HALAMAN PENGESAHAN

SIMULASI BLOCKCHAIN UNTUK TRANSPARANSI TRANSAKSI PADA AKUN GAME ONLINE

DARUL KAHARMAN
NIM. 221280080

Telah dipertahankan di depan Komisi Penguji Ujian Skripsi pada tanggal 31 Agustus 2025 dan dinyatakan telah memenuhi syarat

Komisi Penguji

Dr. Eng. Nurdiansyah Sirimorok, S.Kom., MM. (Ketua)(.....)

Wahyu Arthanugraha, S.Kom., M.Kom. (Sekretaris) (.....)

Ade Hastuty, ST., S.Kom., MT. (Anggota) (.....)

Hasnawati, S.Kom., M.Kom. (Anggota) (.....)

Mengetahui:

Plt. Ketua Program Studi
Teknik Informatika

Mughafir Yunus, ST., MT.
NBM. 1252 809

Dekan Fakultas Teknik

Dr. Ir. H. Hakzah, ST., MT.
NBM. 938 317

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertandatangan di bawah ini :

Nama	: **Darul Kaharman**
NIM	: **221280080**
Program Studi	: Teknik Informatika
Fakultas	: Teknik Universitas Muhammadiyah Parepare
Judul Skripsi	: Simulasi Blockchain untuk Transparansi Transaksi pada Akun Game Online

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilalihan tulisan atau pemikiran orang lain. Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini hasil karya orang lain, saya bersedia menerima sanksi sesuai dengan aturan yang berlaku.

Parepare, 31 Agustus 2025

Yang menyatakan



Darul Kaharman
NIM. **221280080**

PRAKATA



Segala puji dan syukur kami panjatkan ke hadirat Allah SWT atas limpahan rahmat, taufik, dan hidayah-Nya, serta nikmat kesehatan dan kesempatan yang telah diberikan, sehingga proposal penelitian ini dapat disusun dan diselesaikan dengan baik. Shalawat dan salam semoga senantiasa tercurah kepada junjungan kita Nabi Besar Muhammad SAW, yang telah membimbing umat manusia dari zaman kegelapan menuju era ilmu pengetahuan yang terang benderang seperti yang kita nikmati saat ini.

Proposal ini disusun sebagai bagian dari upaya pengembangan ilmu pengetahuan, sekaligus merupakan salah satu syarat untuk menyelesaikan studi di Universitas Muhammadiyah Parepare. Penelitian ini berfokus pada topik **"Simulasi Blockchain untuk Transparansi Transaksi pada Akun Game online",** yang diharapkan dapat memberikan kontribusi nyata dalam meningkatkan transparansi transaksi jual beli akun *game online*, serta menciptakan ekosistem perdagangan digital yang lebih terpercaya dan efisien.

Penyusunan proposal ini tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, pada kesempatan ini, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Ayahanda, Kaharman Pammu, dan Ibunda, Rugaiyah, atas segala doa, cinta, kesabaran, serta dukungan yang tiada henti sejak penulis kecil hingga saat ini.

Terima kasih juga kepada kakak dan keluarga besar atas semangat dan dukungan yang selalu diberikan.

2. Bapak Dr. Eng. Nurdiansyah Sirimorok, S.Kom., MM dan Bapak Wahyu Arthanugraha, S.Kom., M.Kom selaku dosen pembimbing, yang dengan sabar memberikan masukan dan arahan selama penulis menyusun dan menyelesaikan skripsi ini.
3. Ibu Ade Hastuty, ST., S.Kom., MT. dan Ibu Hasnawati, S.Kom., M.Kom. selaku dosen penguji atas arahan dan masukan selama penulis menyusun dan menyelesaikan skripsi ini.
4. Bapak Mughafir Yunus, S.T., M.T., selaku Plt. Ketua Prodi Teknik Informatika, atas arahan, dukungan, dan fasilitas selama studi serta penyusunan skripsi, serta kepada seluruh staf Fakultas Teknik atas bantuan administrasi yang ramah dan profesional.
5. Kepada seluruh teman-teman Kelas C, Jurusan Teknik Informatika Angkatan 2021, atas kebersamaan, dukungan, dan bantuan yang begitu berarti selama proses penulisan ini.
6. Kepada rekan-rekan seperjuangan, terima kasih atas kebersamaan dan dukungan. Dari diskusi serius hingga obrolan ringan, kalian selalu ada. Tongkrongan Sweetness menjadi saksi tawa dan keluh-kesah kita. Kehadiran kalian menguatkan dan memaknai perjalanan ini.
7. Last but not least, I wanna thank me, I wanna thank me for believing in me, I wanna thank me for doing all this hard work, I wanna thank me for having no days off, I wanna thank me for never quitting, I wanna thank me for always

being a giver and try give more than I receive, I wanna thank me for try and do more right than wrong, I wanna thank me for just being me at all times.

Penulis menyadari bahwa terdapat banyak kekeliruan dalam penyusunan skripsi ini. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan penelitian ini di masa mendatang. Semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan.

Billahi fii sabililhaq fastabiqul khairat

Wassalamu alaikum warahmatullahi wabarakatuh

Parepare, 31 Agustus 2025

Penulis

Darul Kaharman
NIM. 221280080

ABSTRAK

DARUL KAHARMAN "*Simulasi Blockchain Untuk Transparansi Transaksi Pada Akun Game online"* (dibimbing oleh Nurdiansyah Sirimorok dan Wahyu Arthanugraha).

Penelitian ini bertujuan untuk mengembangkan simulasi penerapan teknologi *blockchain* dalam transaksi jual beli akun *game online* guna meningkatkan transparansi. Permasalahan utama yang dihadapi adalah tingginya risiko penipuan dan ketidakpastian dalam transaksi digital, seperti penjual yang tidak menyerahkan akun setelah menerima pembayaran atau pembeli yang membatalkan transaksi secara sepihak. Solusi yang diusulkan adalah penerapan *smart contract* berbasis *escrow* pada jaringan *ethereum* menggunakan bahasa *solidity*, yang *dideploy* pada *testnet sepolia*, diverifikasi melalui *etherscan*, serta diintegrasikan dengan dompet *web3* dalam aplikasi *web*. Mekanisme *escrow* memastikan dana hanya dilepaskan apabila kedua belah pihak memenuhi kewajiban, sehingga mengurangi potensi penipuan sekaligus meningkatkan kepercayaan. Metode penelitian menggunakan simulasi sistem berbasis blockchain dan pengujian fungsional melalui teknik *black-box* serta *white-box* untuk menilai keandalan alur transaksi. Hasil *usability testing* dengan 20 responden menunjukkan tingkat kepuasan sebesar 85,6% (856 dari 1000 poin), yang termasuk kategori "Sangat Puas". Temuan ini menegaskan bahwa penerapan *blockchain* melalui *smart contract escrow* mampu meningkatkan transparansi, keandalan, dan efisiensi pada ekosistem jual beli akun *game* online, serta berpotensi menjadi solusi praktis yang dapat direplikasi.

Kata Kunci: *Blockchain*, *Smart contract*, *Escrow*, *Ethereum*, *Game online*, Transparansi.

ABSTRACT

DARUL KAHARMAN *"Blockchain Simulation for Transaction Transparency in Online Game Account Sales" (supervised by Nurdiansyah Sirimorok and Wahyu Artanugraha).*

This study aims to develop a simulation of blockchain implementation in online game account trading to enhance transaction transparency. The main problems addressed are the high risk of fraud and uncertainty in digital transactions, such as sellers failing to deliver accounts after receiving payment or buyers unilaterally canceling transactions. The proposed solution applies an escrow-based smart contract on the Ethereum network using Solidity, deployed on the Sepolia testnet, verified through Etherscan, and integrated with a Web3 wallet in a web application. The escrow mechanism ensures that funds are released only when both parties fulfill their obligations, thereby reducing fraud potential and increasing trust. The research method employs a blockchain-based system simulation and functional testing using black-box and white-box techniques to evaluate transaction reliability. Usability testing with 20 respondents recorded a satisfaction level of 85,6% (856 out of 1000 points), categorized as "Very Satisfied." These findings demonstrate that blockchain implementation through escrow smart contracts can improve transparency, reliability, and efficiency in online game account trading, while also offering a practical and replicable solution for digital transactions.

Keywords*: Blockchain, Smart contract, Escrow, Ethereum, Game online, Transparency.*

DAFTAR ISI

PRAKATA	4
ABSTRAK	7
ABSTRACT	8
DAFTAR ISI	9
LAMPIRAN	10

LAMPIRAN

Escrow.sol

```
pragma solidity ^0.8.19;

contract EscrowFixed {
    event EscrowCreated(uint256
indexed escrowId, address indexed
buyer, address indexed seller,
uint256 amount);
    event
PaymentDeposited(uint256 indexed
escrowId, address indexed buyer,
uint256 amount);
    event PaymentReleased(uint256
indexed escrowId, address indexed
seller, uint256 amount);
    event Payment*Refund*ed(uint256
indexed escrowId, address indexed
buyer, uint256 amount);
    event DisputeCreated(uint256
indexed escrowId, address indexed
initiator);
    event DisputeResolved(uint256
indexed escrowId, address indexed
winner, uint256 amount);
    event
AdminPaymentProcessed(uint256
indexed escrowId, address indexed
admin, uint256 amount, string
reason);
    event
Admin*Refund*Processed(uint256
indexed escrowId, address indexed
admin, uint256 amount, string
reason);
    enum EscrowStatus {
        FUNDED,
        COMPLETED,
        DISPUTED,
        *REFUND*ED,
        ADMIN_PROCESSED
    }
    struct EscrowData {
        uint256 escrowId;
        address buyer;
        address seller;
        uint256 amount;
        EscrowStatus status;
        uint256 createdAt;
        bool disputeActive;
        address disputeInitiator;
        string productCode;
        address processedBy;
        uint256 processedAt;
        string adminNote;
    }
    mapping(uint256 =>
EscrowData) public escrows;
    mapping(address => bool)
public admins;
    mapping(address => uint256)
public adminPaymentCount;
    address public owner;
    uint256 public escrowCounter;
    uint256 public
totalAdminPayments;
    modifier onlyOwner() {
        require(msg.sender ==
owner, "Only owner");
    }
    modifier onlyAdmin() {

require(admins[msg.sender] ||
msg.sender == owner, "Only
admin");
    }
    modifier onlyBuyer(uint256
_escrowId) {
        require(msg.sender ==
escrows[_escrowId].buyer, "Only
buyer");
    }
    modifier escrowExists(uint256
_escrowId) {

require(escrows[_escrowId].buyer
!= address(0), "Escrow not
found");
    }
    constructor() {
        owner = msg.sender;
        admins[msg.sender] =
true;
    }
    function createEscrow(address
_seller, string memory
_productCode) external payable
returns (uint256) {
        require(_seller !=
address(0) && _seller !=
msg.sender, "Bad seller");
        require(msg.value > 0,
"Zero amount");
        uint256 newEscrowId =
++escrowCounter;
        escrows[newEscrowId] =
EscrowData({
            escrowId:
newEscrowId,
            buyer: msg.sender,
            seller: _seller,
            amount: msg.value,
            status:
EscrowStatus.FUNDED,
            createdAt:
block.timestamp,
            disputeActive: false,
            disputeInitiator:
address(0),
```

```
            productCode:
_productCode,
            processedBy:
address(0),
            processedAt: 0,
            adminNote: ""});emit
EscrowCreated(newEscrowId,
msg.sender, _seller, msg.value);
        emit
PaymentDeposited(newEscrowId,
msg.sender, msg.value);
        return newEscrowId;
    }
    function
confirmReceived(uint256
_escrowId)
        external
        escrowExists(_escrowId)
        onlyBuyer(_escrowId)
    {
        EscrowData storage e =
escrows[_escrowId];
        require(e.status ==
EscrowStatus.FUNDED, "Not
FUNDED");
        require(!e.disputeActive,
"In dispute");

        e.status =
EscrowStatus.COMPLETED;
        e.processedBy =
msg.sender;
        e.processedAt =
block.timestamp;


payable(e.seller).transfer(e.amou
nt);
        emit
PaymentReleased(_escrowId,
e.seller, e.amount);
    }

    function
adminReleaseFunds(uint256
_escrowId, string memory _reason)
        external
        escrowExists(_escrowId)
        onlyAdmin
    {

require(bytes(_reason).length >=
10, "Reason too short");
        EscrowData storage e =
escrows[_escrowId];
        require(e.status ==
EscrowStatus.FUNDED, "Not
FUNDED");

        e.status =
EscrowStatus.ADMIN_PROCESSED;
        e.processedBy =
msg.sender;
        e.processedAt =
block.timestamp;
        e.adminNote = _reason;
adminPaymentCount[msg.sender]++;
        totalAdminPayments++;
payable(e.seller).transfer(e.amou
nt);
        emit
AdminPaymentProcessed(_escrowId,
msg.sender, e.amount, _reason);
        emit
PaymentReleased(_escrowId,
e.seller, e.amount);
    }
    function adminRefund(uint256
_escrowId, string memory _reason)
        external
        escrowExists(_escrowId)
        onlyAdmin
    {

require(bytes(_reason).length >=
10, "Reason too short");
        EscrowData storage e =
escrows[_escrowId];
        require(e.status ==
EscrowStatus.FUNDED, "Not
FUNDED");

        e.status =
EscrowStatus.REFUNDED;
        e.processedBy =
msg.sender;
        e.processedAt =
block.timestamp;
        e.adminNote = _reason;
adminPaymentCount[msg.sender]++;
payable(e.buyer).transfer(e.amoun
t);
        emit
AdminRefundProcessed(_escrowId,
msg.sender, e.amount, _reason);
        emit
PaymentRefunded(_escrowId,
e.buyer, e.amount);
    }
    function
createDispute(uint256 _escrowId)
        external
        escrowExists(_escrowId)
    {
        EscrowData storage e =
escrows[_escrowId];
        require(msg.sender ==
e.buyer || msg.sender ==
e.seller, "Not party");
        require(e.status ==
EscrowStatus.FUNDED, "Not
FUNDED");
        require(!e.disputeActive,
"Dispute active");
```

```
        e.status =
EscrowStatus.DISPUTED;
        e.disputeActive = true;
        e.disputeInitiator =
msg.sender;

        emit
DisputeCreated(_escrowId,
msg.sender);
    }
    function
adminCreateDispute(uint256
_escrowId, string memory _reason)
        external
        escrowExists(_escrowId)
        onlyAdmin
    {

require(bytes(_reason).length >=
10, "Reason too short");
        EscrowData storage e =
escrows[_escrowId];
        require(e.status ==
EscrowStatus.FUNDED, "Not
FUNDED");
        require(!e.disputeActive,
"Dispute active");

        e.status =
EscrowStatus.DISPUTED;
        e.disputeActive = true;
        e.disputeInitiator =
msg.sender;
        e.adminNote = _reason;

        emit
DisputeCreated(_escrowId,
msg.sender);
    }
    function
resolveDispute(uint256 _escrowId,
address _winner)
        external
        escrowExists(_escrowId)
        onlyAdmin
    {
        EscrowData storage e =
escrows[_escrowId];
        require(e.status ==
EscrowStatus.DISPUTED, "No
dispute");
        require(_winner ==
e.buyer || _winner == e.seller,
"Bad winner");

        e.disputeActive = false;
        e.processedBy =
msg.sender;
        e.processedAt =
block.timestamp;
adminPaymentCount[msg.sender]++;
        if (_winner == e.seller)
{
            e.status =
EscrowStatus.COMPLETED;
payable(e.seller).transfer(e.amou
nt);emit
PaymentReleased(_escrowId,
e.seller, e.amount);
        } else {
            e.status =
EscrowStatus.REFUNDED;

payable(e.buyer).transfer(e.amoun
t);
            emit
PaymentRefunded(_escrowId,
e.buyer, e.amount);
        }
emit DisputeResolved(_escrowId,
_winner, e.amount);
    }
    function addAdmin(address
_admin) external onlyOwner {
        require(_admin !=
address(0), "Zero addr");
        admins[_admin] = true;
    }
    function removeAdmin(address
_admin) external onlyOwner {
        require(_admin != owner,
"Owner protected");
        admins[_admin] = false;
    }
    function isAdmin(address
_addr) external view returns
(bool) {
        return admins[_addr] ||
_addr == owner;
    }
    function
getAdminPaymentCount(address
_admin) external view returns
(uint256) {
        return
adminPaymentCount[_admin];
    }
    function getContractStats()
        external
        view
        returns (uint256
totalEscrows, uint256
totalAdminProcessed, uint256
contractBalance)
    {
        return (escrowCounter,
totalAdminPayments,
address(this).balance);
    }
    function getEscrow(uint256
_escrowId)
        external
        view
        escrowExists(_escrowId)
        returns (EscrowData
memory)
```

```
    {
        return
escrows[_escrowId];
    }
    function getContractBalance()
external view returns (uint256) {
return address(this).balance;
    }
    function emergencyWithdraw()
external onlyOwner {

payable(owner).transfer(address(t
his).balance);
    }
    function
batchProcess(uint256[] memory
_escrowIds, uint8 _action, string
memory _reason) external
onlyAdmin {
        require(_escrowIds.length
> 0 && _escrowIds.length <= 20,
"Bad batch size");
        require(_action <= 1,
"Bad action");

require(bytes(_reason).length >=
10, "Reason too short");

        for (uint256 i = 0; i <
_escrowIds.length; i++) {
            uint256 id =
_escrowIds[i];
            if (id == 0 || id >
escrowCounter) continue;

            EscrowData storage e
= escrows[id];
            if (e.status !=
EscrowStatus.FUNDED ||
e.disputeActive) continue;

            if (_action == 0) {
                e.status =
EscrowStatus.ADMIN_PROCESSED;
                e.processedBy =
msg.sender;
                e.processedAt =
block.timestamp;
                e.adminNote =
_reason;
adminPaymentCount[msg.sender]++;
totalAdminPayments++;payable(e.se
ller).transfer(e.amount);
                emit
AdminPaymentProcessed(id,
msg.sender, e.amount, _reason);
                emit
PaymentReleased(id, e.seller,
e.amount);
            } else {
                e.status =
EscrowStatus.REFUNDED;
                e.processedBy =
msg.sender;
                e.processedAt =
block.timestamp;
                e.adminNote =
_reason;
adminPaymentCount[msg.sender]++;
ayable(e.buyer).transfer(e.amount
);
                emit
AdminRefundProcessed(id,
msg.sender, e.amount, _reason);
                emit
PaymentRefunded(id, e.buyer,
e.amount);
            }
        }
    }
}
```

web3.js

```
import { ethers } from 'ethers';
import EscrowABI from
'../abi/Escrow.json';

const CONTRACT_ADDRESS =
process.env.REACT_APP_CONTRACT_AD
DRESS ||
'0x2100b2dEF2B3d7Dc4B29f8D297C9AA
283b74b1f6';
const NETWORK_NAME =
process.env.REACT_APP_NETWORK ||
'sepolia';
const CHAIN_ID =
parseInt(process.env.REACT_APP_CH
AIN_ID || '11155111');
const RPC_URL =
process.env.REACT_APP_RPC_URL ||
'https://sepolia.infura.io/v3/96f
08bb9b5ba41bda7d5999a33d2d523';
class Web3Service {
  constructor() {
    this.provider = null;
    this.signer = null;
    this.contract = null;
    this.account = null;
  }

  isMetaMaskInstalled() {
    return typeof window !==
'undefined' && typeof
window.ethereum !== 'undefined';
  }

  async connectWallet() {
    try {
      if
(!this.isMetaMaskInstalled())
throw new Error('MetaMask not
installed');
      const accounts = await
window.ethereum.request({ method:
'eth_requestAccounts' });
```

```
      if (accounts.length === 0)
throw new Error('No accounts
found');
      this.provider = new
ethers.BrowserProvider(window.eth
ereum);
      this.signer = await
this.provider.getSigner();
      this.account = accounts[0];
      this.contract = new
ethers.Contract(CONTRACT_ADDRESS,
EscrowABI, this.signer);
      await this.checkNetwork();
      return { success: true,
account: this.account };
    } catch (error) {
      return { success: false,
error: error.message };
    }
  }
  async checkNetwork() {
    try {
      const network = await
this.provider.getNetwork();
      const chainId =
Number(network.chainId);
      if (chainId !== CHAIN_ID) {
        await
window.ethereum.request({
          method:
'wallet_switchEthereumChain',
          params: [{ chainId:
`0x${CHAIN_ID.toString(16)}` }]
        });
      }
      return true;
    } catch (error) {
      if (error.code === 4902)
await this.addSepoliaNetwork();
      else throw error;
    }
  }
  async addSepoliaNetwork() {
    await
window.ethereum.request({
      method:
'wallet_addEthereumChain',
      params: [{
        chainId:
`0x${CHAIN_ID.toString(16)}`,
        chainName: 'Sepolia
Testnet',
        nativeCurrency: { name:
'Ethereum', symbol: 'ETH',
decimals: 18 },
        rpcUrls: [RPC_URL],
        blockExplorerUrls:
['https://sepolia.etherscan.io/']
      }]
    });
  }
  async getCurrentAccount() {
    try {
      if
(!this.isMetaMaskInstalled())
return null;
      const accounts = await
window.ethereum.request({ method:
'eth_accounts' });
      if (accounts &&
accounts.length > 0) {
        if (!this.provider) {
          this.provider = new
ethers.BrowserProvider(window.eth
ereum);
          this.signer = await
this.provider.getSigner();
          this.account =
accounts[0];
          this.contract = new
ethers.Contract(CONTRACT_ADDRESS,
EscrowABI, this.signer);
        }
        return accounts[0];
      }
      return null;
    } catch {
      return null;
    }
  }
  async getBalance(address =
null) {
    if (!this.provider) throw new
Error('Provider not
initialized');
    const targetAddress = address
|| this.account;
    if (!targetAddress) throw new
Error('No address provided');
    const balance = await
this.provider.getBalance(targetAd
dress);
    return
ethers.formatEther(balance);
  }
  async
createEscrow(sellerAddress,
productCode, amountInEth) {
    try {
      if (!this.contract) throw
new Error('Contract not
initialized');
      const amountWei =
ethers.parseEther(amountInEth.toS
tring());
      const tx = await
this.contract.createEscrow(seller
Address, productCode, {
        value: amountWei,
        gasLimit: 300000
      });
      const receipt = await
tx.wait();
      let escrowId = null;
      for (const log of
receipt.logs) {
```

```
        try {
          const parsedLog =
this.contract.interface.parseLog(
log);
          if (parsedLog.name ===
'EscrowCreated') {
            escrowId =
parsedLog.args.escrowId.toString(
);
            break;
          }
        } catch {}
      }
      return { success: true,
escrowId, transactionHash:
tx.hash, blockNumber:
receipt.blockNumber, gasUsed:
receipt.gasUsed.toString() };
    } catch (error) {
      return { success: false,
error: error.message };
    }
  }
  async confirmReceived(escrowId)
{
    try {
      if (!this.contract) throw
new Error('Contract not
initialized');
      const tx = await
this.contract.confirmReceived(esc
rowId, { gasLimit: 200000 });
      const receipt = await
tx.wait();
      return { success: true,
transactionHash: tx.hash,
blockNumber: receipt.blockNumber,
gasUsed:
receipt.gasUsed.toString() };
    } catch (error) {
      return { success: false,
error: error.message };
    }
  }
  async createDispute(escrowId) {
    try {
      if (!this.contract) throw
new Error('Contract not
initialized');
      const tx = await
this.contract.createDispute(escro
wId, { gasLimit: 200000 });
      const receipt = await
tx.wait();
      return { success: true,
transactionHash: tx.hash,
blockNumber: receipt.blockNumber,
gasUsed:
receipt.gasUsed.toString() };
    } catch (error) {
      return { success: false,
error: error.message };
    }
  }
  async getEscrow(escrowId) {
    try {
      if (!this.contract) {
        const readProvider = new
ethers.JsonRpcProvider(RPC_URL);
        const readContract = new
ethers.Contract(CONTRACT_ADDRESS,
EscrowABI, readProvider);
        const e = await
readContract.getEscrow(escrowId);
        return {
          success: true,
          data: {
            escrowId:
e.escrowId.toString(),
            buyer: e.buyer,
            seller: e.seller,
            amount:
ethers.formatEther(e.amount),
            status:
this.getStatusName(e.status),
            statusCode: e.status,
            dibuatPada: new
Date(Number(e.createdAt) * 1000),
            disputeActive:
e.disputeActive,
            disputeInitiator:
e.disputeInitiator,
            productCode:
e.productCode,
            processedBy:
e.processedBy,
            processedAt:
e.processedAt ? new
Date(Number(e.processedAt) *
1000) : null,
            adminNote:
e.adminNote
          }
        };
      }
      const e = await
this.contract.getEscrow(escrowId)
;
      return {
        success: true,
        data: {
          escrowId:
e.escrowId.toString(),
          buyer: e.buyer,
          seller: e.seller,
          amount:
ethers.formatEther(e.amount),
          status:
this.getStatusName(e.status),
          statusCode: e.status,
          dibuatPada: new
Date(Number(e.createdAt) * 1000),
          disputeActive:
e.disputeActive,
          disputeInitiator:
e.disputeInitiator,
```

```
          productCode:
e.productCode,
          processedBy:
e.processedBy,
          processedAt:
e.processedAt ? new
Date(Number(e.processedAt) *
1000) : null,
          adminNote: e.adminNote
        }
      };
    } catch (error) {
      return { success: false,
error: error.message };
    }
  }

  getStatusName(statusCode) {
    const statuses = {
      0: 'CREATED',
      1: 'FUNDED',
      2: 'COMPLETED',
      3: 'DISPUTED',
      4: 'REFUNDED',
      5: 'CANCELLED',
      6: 'ADMIN_PROCESSED'
    };
    return statuses[statusCode]
|| 'UNKNOWN';
  }
  setupEventListeners() {
    if
(!this.isMetaMaskInstalled())
return;
    if
(window.ethereum.removeAllListene
rs) {

window.ethereum.removeAllListener
s('accountsChanged');

window.ethereum.removeAllListener
s('chainChanged');
    }

window.ethereum.on('accountsChang
ed', (accounts) => {
      if (accounts.length === 0)
{
        this.disconnect();
      } else {
        this.account =
accounts[0];
        window.dispatchEvent(new
CustomEvent('walletAccountChanged
', { detail: { account:
accounts[0] } }));
      }
    });

window.ethereum.on('chainChanged'
, (chainId) => {
      window.dispatchEvent(new
CustomEvent('walletChainChanged',
{ detail: { chainId } }));
  disconnect() {
    this.provider = null;
    this.signer = null;
    this.contract = null;
    this.account = null;
  }
}
const web3Service = new
Web3Service();
export default web3Service;
```

WalletContext.jsx

```
import React, { createContext,
useContext, useState, useEffect }
from 'react';
import web3Service from
'../layanan/web3';
import { useAuth } from
'./AuthContext';
import toast from 'react-hot-
toast';
const WalletContext =
createContext();
export const useWallet = () => {
  const context =
useContext(WalletContext);
  if (!context) {
    throw new Error('useWallet
must be used within a
WalletProvider');
  }
  return context;
};
export const WalletProvider = ({
children }) => {
  const { user, updateProfile } =
useAuth();
  const [isConnected,
setIsConnected] =
useState(false);
  const [account, setAccount] =
useState(null);
  const [balance, setBalance] =
useState('0');
  const [loading, setLoading] =
useState(false);
  const [networkError,
setNetworkError] =
useState(null);
  useEffect(() => {
    checkWalletConnection();
    const cleanup =
setupEventListeners();
    return () => {
      if (cleanup) cleanup();
    };
  }, []);
  useEffect(() => {
    if (user) {
      checkWalletConnection();
```

```
    }
  }, [user]);
  useEffect(() => {
    if (user?.walletAddress &&
account) {
      validateWalletMatch();
    }
  }, [user, account]);
  useEffect(() => {
    if (account) {
      updateBalance();
    }
  }, [account]);
  const checkWalletConnection =
async () => {
    try {
      if
(!web3Service.isMetaMaskInstalled
()) {
        return;
      }
      const currentAccount =
await
web3Service.getCurrentAccount();
      if (currentAccount) {

setAccount(currentAccount);
        setIsConnected(true);
        await updateBalance();
        if (user?.walletAddress
|| user?.alamatWallet) {
          const registeredWallet
= user.walletAddress ||
user.alamatWallet;
          if
(currentAccount.toLowerCase() !==
registeredWallet.toLowerCase()) {
          }
        }
      } else {
        setAccount(null);
        setIsConnected(false);
      }
    } catch (error) {
    }
  };
  const connectWallet = async ()
=> {
    try {
      setLoading(true);
      setNetworkError(null);
      if
(!web3Service.isMetaMaskInstalled
()) {
        const installMetaMask =
window.confirm('MetaMask tidak
terdeteksi. Apakah Anda ingin
menginstall MetaMask sekarang?');
        if (installMetaMask) {

window.open('https://metamask.io/
download/', '_blank');
        }
        toast.error('MetaMask
diperlukan untuk menghubungkan
dompet');
        return { success: false,
error: 'MetaMask not installed'
};
      }
      const loadingToast =
toast.loading('Menghubungkan
dompet... Periksa popup
blockchain');
      const result = await
web3Service.connectWallet();

toast.dismiss(loadingToast);
      if (result.success) {
        if (user?.walletAddress)
{
          if
(result.account.toLowerCase() !==
user.walletAddress.toLowerCase())
{
            toast.error(`AKSES
DITOLAK!\n\nDompet yang
terhubung:
${result.account.slice(0,
6)}...${result.account.slice(-
4)}\nDompet terdaftar:
${user.walletAddress.slice(0,
6)}...${user.walletAddress.slice(
-4)}\n\nSilakan ganti ke dompet
yang benar di blockchain!`, {
duration: 6000 });
            return { success:
false, error: 'Wallet tidak
sesuai dengan yang terdaftar' };
          }

setAccount(result.account);
          setIsConnected(true);
          await updateBalance();
          toast.success(`Dompet
terverifikasi!
${result.account.slice(0,
6)}...${result.account.slice(-
4)}`);
        } else {

setAccount(result.account);
          setIsConnected(true);
          await updateBalance();
          toast.success(`Dompet
terhubung!
${result.account.slice(0,
6)}...${result.account.slice(-
4)}`);
        }
        try {
          await
loadSmartContractData(result.acco
unt);
        } catch (contractError) {
        }
```

```
        return { success: true,
address: result.account };
      } else {
        let errorMessage =
result.error || 'Gagal
menghubungkan dompet';
        if (result.error &&
result.error.includes('User
rejected')) {
          errorMessage = 'Koneksi
dompet dibatalkan oleh pengguna';
        } else if (result.error
&& result.error.includes('Already
processing')) {
          errorMessage =
'MetaMask sedang memproses
permintaan lain';
        } else if (result.error
&& result.error.includes('No
accounts found')) {
          errorMessage = 'Tidak
ada akun ditemukan. Pastikan
MetaMask sudah login';
        }

toast.error(errorMessage);
        return { success: false,
error: errorMessage };
      }
    } catch (error) {
      let errorMessage = 'Gagal
menghubungkan dompet';
      if (error.message &&
error.message.includes('User
rejected')) {
        errorMessage = 'Koneksi
dompet dibatalkan';
      } else if (error.message &&
error.message.includes('MetaMask'
)) {
        errorMessage = 'Error
MetaMask: ' + error.message;
      } else if (error.message) {
        errorMessage =
error.message;
      }
      toast.error(errorMessage);
      return { success: false,
error: errorMessage };
    } finally {
      setLoading(false);
    }
  };
  const disconnectWallet = () =>
{
    web3Service.disconnect();
    setAccount(null);
    setIsConnected(false);
    setBalance('0');
    setNetworkError(null);
    toast.success('Dompet
terputus');
  };

  const updateBalance = async ()
=> {
    try {
      if (account) {
        const balance = await
web3Service.getBalance(account);
        setBalance(balance);
      }
    } catch (error) {
    }
  };
  const linkWalletToProfile =
async (walletAddress) => {
    try {
      const result = await
updateProfile({ alamatWallet:
walletAddress });
      if (result.success) {

setAccount(walletAddress);
        setIsConnected(true);
        toast.success('Dompet
berhasil dikaitkan dengan
akun!');
        return { success: true };
      } else {
        toast.error(result.error
|| 'Gagal mengaitkan dompet
dengan akun');
        return { success: false,
error: result.error };
      }
    } catch (error) {
      toast.error('Gagal
mengaitkan dompet dengan akun');
      return { success: false,
error: error.message };
    }
  };
  const setupEventListeners = ()
=> {

web3Service.setupEventListeners()
;
    const handleAccountChanged =
(event) => {
      const newAccount =
event.detail.account;
      setAccount(newAccount);
      setIsConnected(true);
      updateBalance();
    };
    const handleChainChanged =
(event) => {
      setTimeout(() => {
        checkWalletConnection();
      }, 1000);
    };

window.addEventListener('walletAc
countChanged',
handleAccountChanged);
```

```
window.addEventListener('walletCh
ainChanged', handleChainChanged);
    return () => {

window.removeEventListener('walle
tAccountChanged',
handleAccountChanged);

window.removeEventListener('walle
tChainChanged',
handleChainChanged);
    };
  };
  const validateWalletMatch = ()
=> {
    if (!user?.walletAddress ||
!account) return;
    if (account.toLowerCase() !==
user.walletAddress.toLowerCase())
{
      setIsConnected(false);
      setAccount(null);
      setBalance('0');
      toast.error(`Dompet Tidak
Sesuai!\n\nDompet aktif:
${account.slice(0,
6)}...${account.slice(-
4)}\nDompet terdaftar:
${user.walletAddress.slice(0,
6)}...${user.walletAddress.slice(
-4)}\n\nSilakan ganti ke dompet
yang benar di blockchain!`, {
duration: 8000 });
    }
  };
  const loadSmartContractData =
async (walletAddress) => {
    try {
      const contractBalance =
await web3Service.getBalance();
      toast.success('Data
blockchain berhasil dimuat!');
    } catch (error) {
    }
  };
  const createEscrow = async
(sellerAddress, productCode,
amountInEth) => {
    try {
      if (!isConnected) {
        toast.error('Silakan
hubungkan dompet terlebih
dahulu');
        return { success: false,
error: 'Wallet not connected' };
      }
      setLoading(true);
      const result = await
web3Service.createEscrow(sellerAd
dress, productCode, amountInEth);
      if (result.success) {
        toast.success('Escrow
berhasil dibuat!');
        await updateBalance();
      } else {

toast.error(result.error);
      }
      return result;
    } catch (error) {
      toast.error('Gagal membuat
escrow');
      return { success: false,
error: error.message };
    } finally {
      setLoading(false);
    }
  };
  const confirmReceived = async
(escrowId, buyerAddress) => {
    try {
      if (!isConnected) {
        toast.error('Silakan
hubungkan dompet terlebih
dahulu');
        return { success: false,
error: 'Wallet not connected' };
      }
      if (account.toLowerCase()
!== buyerAddress.toLowerCase()) {
        toast.error('Dompet yang
terhubung bukan dompet pembeli
yang benar');
        return { success: false,
error: 'Wrong wallet connected'
};
      }
      setLoading(true);
      const { default:
escrowService } = await
import('../layanan/escrowService'
);
      const result = await
escrowService.confirmReceived(esc
rowId, buyerAddress);
      if (result.success) {
        await updateBalance();
      }
      return result;
    } catch (error) {
      return { success: false,
error: error.message };
    } finally {
      setLoading(false);
    }
  };
  const createDispute = async
(escrowId, initiatorAddress,
reason) => {
    try {
      if (!isConnected) {
        toast.error('Silakan
hubungkan dompet terlebih
dahulu');
```

```
        return { success: false,
error: 'Wallet not connected' };
      }
      if (account.toLowerCase()
!==
initiatorAddress.toLowerCase()) {
        toast.error('Dompet yang
terhubung bukan dompet yang
berhak membuat sengketa');
        return { success: false,
error: 'Wrong wallet connected'
};
      }
      setLoading(true);
      const { default:
escrowService } = await
import('../layanan/escrowService'
);
      const result = await
escrowService.createDispute(escro
wId, initiatorAddress, reason);
      return result;
    } catch (error) {
      return { success: false,
error: error.message };
    } finally {
      setLoading(false);
    }
  };
  const getEscrow = async
(escrowId) => {
    try {
      const result = await
web3Service.getEscrow(escrowId);
      return result;
    } catch (error) {
      return { success: false,
error: error.message };
    }
  };
  const formatBalance = (balance)
=> {
    return
parseFloat(balance).toFixed(4);
  };
  const formatAddress = (address)
=> {
    if (!address) return '';
    return `${address.slice(0,
6)}...${address.slice(-4)}`;
  };
  const isWalletLinked = () => {
    return user?.profile?.wallet
=== account;
  };
  const value = {
    isConnected,
    account,
    walletAddress: account,
    balance,
    loading,
    networkError,
    isConnecting: loading,
    provider:
web3Service.provider,
    contract:
web3Service.contract,
    signer: web3Service.signer,
    connectWallet,
    disconnectWallet,
    updateBalance,
    linkWalletToProfile,
    createEscrow,
    confirmReceived,
    createDispute,
    getEscrow,
    formatBalance,
    formatAddress,
    isWalletLinked,
    validateWalletMatch,
    web3Service,
  };
  return (
    <WalletContext.Provider
value={value}>
      {children}
    </WalletContext.Provider>
  );
};
```

Dashboard.jsx

```
import React, { useState,
useEffect, useCallback } from
'react'
import { useNavigate } from
'react-router-dom'
import { useAuth } from
'../../konteks/AuthContext'
import { useWallet } from
'../../konteks/WalletContext'
import { transaksiAPI } from
'../../layanan/api'
import { useDashboardUpdates }
from
'../../hooks/useRealTimeUpdates'
import { getProductImageUrl,
createImageErrorHandler } from
'../../utils/imageHelper'
import ModalLihatAkun from
'../../komponen/ModalLihatAkun'
import ModalTerimaAkun from
'../../komponen/ModalTerimaAkun'
import ModalKirimAkun from
'../../komponen/ModalKirimAkun'
import ModalSengketa from
'../../komponen/ModalSengketa'
import ModalSengketaPenjual from
'../../komponen/ModalSengketaPenj
ual'
import ModalDetailSengketa from
'../../komponen/ModalDetailSengke
ta'
import ModalDetailTransaksi from
'../../komponen/ModalDetailTransa
ksi'
```

```
import PembelianKontrakPintar
from
'../../komponen/PembelianKontrakP
intar'
import transactionExpiryManager
from
'../../utils/transactionExpiry'
import toast from 'react-hot-
toast'

const SimpleTimer = ({ transaksi,
onExpired }) => {
  const [timeLeft, setTimeLeft] =
useState(null)
  const [isExpired, setIsExpired]
= useState(false)
  useEffect(() => {
    if (!transaksi ||
transaksi.status !==
'MENUNGGU_PEMBAYARAN') return
    const createdAt = new
Date(transaksi.dibuatPada)
    const expiryTime = new
Date(createdAt.getTime() + 15 *
60 * 1000)
    const updateTimer = () => {
      const now = new Date()
      const difference =
expiryTime.getTime() -
now.getTime()
      if (difference <= 0) {
        setTimeLeft({ minutes: 0,
seconds: 0 })
        setIsExpired(true)
        if (onExpired)
onExpired(transaksi)
        return
      }
      const minutes =
Math.floor(difference / (1000 *
60))
      const seconds =
Math.floor((difference % (1000 *
60)) / 1000)
      setTimeLeft({ minutes,
seconds })
      setIsExpired(false)
    }
    updateTimer()
    const interval =
setInterval(updateTimer, 1000)
    return () =>
clearInterval(interval)
  }, [transaksi, onExpired])
  if (!transaksi ||
transaksi.status !==
'MENUNGGU_PEMBAYARAN') return
null
  if (isExpired) return <div
className="text-xs text-red-600
font-medium">Waktu habis</div>
  if (timeLeft) {
    return (
      <div className={`text-xs
font-medium ${timeLeft.minutes <
5 ? 'text-red-600' :
timeLeft.minutes < 10 ? 'text-
orange-600' : 'text-blue-600'}`}>

{String(timeLeft.minutes).padStar
t(2,
'0')}:{String(timeLeft.seconds).p
adStart(2, '0')}
      </div>
    )
  }
  return <div className="text-xs
text-blue-600 font-
medium">Memuat...</div>
}
const Dashboard = () => {
  const [activeTab, setActiveTab]
= useState('pembelian')
  const [transaksiPembelian,
setTransaksiPembelian] =
useState([])
  const [transaksiPenjualan,
setTransaksiPenjualan] =
useState([])
  const [loading, setLoading] =
useState(true)
  const [error, setError] =
useState(null)
  const [filter, setFilter] =
useState('all')
  const [lastRefresh,
setLastRefresh] = useState(new
Date())
  const [refreshing,
setRefreshing] = useState(false)
  const [hasSellerActivity,
setHasSellerActivity] =
useState(false)
  const [showLihatAkunModal,
setShowLihatAkunModal] =
useState(false)
  const [showTerimaAkunModal,
setShowTerimaAkunModal] =
useState(false)
  const [showKirimAkunModal,
setShowKirimAkunModal] =
useState(false)
  const [showDetailSengketaModal,
setShowDetailSengketaModal] =
useState(false)
  const [showSengketaModal,
setShowSengketaModal] =
useState(false)
  const
[showDetailTransaksiModal,
setShowDetailTransaksiModal] =
useState(false)
  const [showPaymentModal,
setShowPaymentModal] =
useState(false)
```

```
  const [selectedTransaksi,
setSelectedTransaksi] =
useState(null)
  const [actionLoading,
setActionLoading] =
useState(false)
  const { user, isAuthenticated,
loading: authLoading } =
useAuth()
  const { isConnected,
walletAddress } = useWallet()
  const navigate = useNavigate()
  const { triggerUpdate } =
useDashboardUpdates(() => {
    fetchAllTransaksi(false)
    toast.success('Data transaksi
diperbarui', { duration: 2000 })
  })
  useEffect(() => {
    const
handleExpiredTransaction = (_,
expiredTransaksi) =>
fetchAllTransaksi(false)
    const
handleTransactionCreated = () =>
{
      fetchAllTransaksi(false)
      toast.success('Transaksi
berhasil dibuat! Timer 15 menit
dimulai.')
    }
transactionExpiryManager.addListe
ner(handleExpiredTransaction)

window.addEventListener('transact
ion-created',
handleTransactionCreated)
    return () => {

transactionExpiryManager.removeLi
stener(handleExpiredTransaction)

transactionExpiryManager.stopAll(
)

window.removeEventListener('trans
action-created',
handleTransactionCreated)
    }
  }, [])
  useEffect(() => {
    if (user?.role === 'PENJUAL')
setActiveTab('penjualan')
    else if (user?.role ===
'PEMBELI')
setActiveTab('pembelian')
  }, [user?.role])
  useEffect(() => {
    if (authLoading) return
    if (!isAuthenticated ||
!user) {
      navigate('/masuk')
      return
    }
    fetchAllTransaksi()
  }, [isAuthenticated, user,
authLoading, filter])
  const fetchAllTransaksi =
useCallback(async (showLoading =
true) => {
    try {
      if (showLoading)
setLoading(true)
      else setRefreshing(true)
      setError(null)
      if (!user?.id) throw new
Error('User ID tidak ditemukan')
      const responsePembelian =
await
transaksiAPI.getTransaksiSaya({
role: 'pembeli', status: filter
=== 'all' ? undefined : filter,
page: 1, limit: 50, timestamp:
Date.now() })
      const responsePenjualan =
await
transaksiAPI.getTransaksiUser({
role: 'penjual', status: filter
=== 'all' ? undefined : filter,
page: 1, limit: 50, timestamp:
Date.now() })
      let transaksiPembelianData
= []
      if
(responsePembelian.data?.sukses)
transaksiPembelianData =
responsePembelian.data.data?.tran
saksi ||
responsePembelian.data.data || []
      else if
(Array.isArray(responsePembelian.
data)) transaksiPembelianData =
responsePembelian.data
      let transaksiPenjualanData
= []
      if
(responsePenjualan.data?.sukses)
transaksiPenjualanData =
responsePenjualan.data.data?.tran
saksi ||
responsePenjualan.data.data || []
      else if
(Array.isArray(responsePenjualan.
data)) transaksiPenjualanData =
responsePenjualan.data
      const filteredPembelian =
transaksiPembelianData.filter(t
=> t.status !== 'GAGAL')
      const filteredPenjualan =
transaksiPenjualanData.filter(t
=> t.status !== 'GAGAL')

setTransaksiPembelian(filteredPem
belian)
```

```
setTransaksiPenjualan(filteredPen
jualan)
      setLastRefresh(new Date())
      const userHasSellerActivity
= filteredPenjualan.length > 0 ||
(user?.role === 'PENJUAL')

setHasSellerActivity(userHasSelle
rActivity)
      if (!userHasSellerActivity
&& activeTab === 'penjualan')
setActiveTab('pembelian')
      const pendingTransactions =
filteredPembelian.filter(t =>
t.status ===
'MENUNGGU_PEMBAYARAN')
      if
(pendingTransactions.length > 0)
transactionExpiryManager.startMon
itoringMultiple(pendingTransactio
ns)
    } catch (error) {
      let errorMessage = 'Gagal
memuat transaksi'
      if (error.response?.status
=== 401) {

localStorage.removeItem('authToke
n')

localStorage.removeItem('user')
        navigate('/masuk')
        return
      } else if
(error.response?.status === 500)
errorMessage = 'Server
bermasalah. Silakan coba lagi
nanti.'
      else if
(error.response?.status === 404)
errorMessage = 'Endpoint tidak
ditemukan. Hubungi admin.'
      else if (error.code ===
'ERR_NETWORK') errorMessage =
'Koneksi ke server bermasalah.'
      else if
(error.response?.data?.pesan) {
        errorMessage =
error.response.data.pesan
        if
(errorMessage.toLowerCase().inclu
des('token tidak valid')) {

localStorage.removeItem('authToke
n')

localStorage.removeItem('user')
          navigate('/masuk')
          return
        }
      } else if (error.message)
errorMessage = error.message
      setError(errorMessage)
      if
(transaksiPembelian.length === 0
&& transaksiPenjualan.length ===
0) {
        setTransaksiPembelian([])
        setTransaksiPenjualan([])
      }
    } finally {
      if (showLoading)
setLoading(false)
      else setRefreshing(false)
    }
  }, [filter, user,
transaksiPembelian.length,
transaksiPenjualan.length,
navigate, activeTab])
  return <div className="min-h-
screen bg-gray-
50">DASHBOARD</div>
}
export default Dashboard
```

- **ModalDetailTransaksi.jsx**

```
import React, { useState,
useEffect } from 'react';
import { transaksiAPI } from
'../layanan/api';
import { useAuth } from
'../konteks/AuthContext';
import { useEthToIdrRate } from
'../hooks/useEthPrice';
import toast from 'react-hot-
toast';
import PembelianKontrakPintar
from './PembelianKontrakPintar';
const TransactionDetailModal = ({
produk, onConfirm, onCancel,
userWalletAddress, userBalance,
onExpired, onPaymentStart }) => {
  const { user } = useAuth();
  const { rate: ethToIdrRate } =
useEthToIdrRate();
  const [currentStep,
setCurrentStep] = useState(1);
  const [agreed, setAgreed] =
useState(false);
  const [timeLeft, setTimeLeft] =
useState(null);
  const [isExpired, setIsExpired]
= useState(false);
  const [paymentStartTime,
setPaymentStartTime] =
useState(null);
  const [isProcessing,
setIsProcessing] =
useState(false);
  const [hasNotifiedExpired,
setHasNotifiedExpired] =
useState(false);
  useEffect(() => {
    if (currentStep !== 2 ||
!paymentStartTime) {
```

```
      setTimeLeft(null);
      setIsExpired(false);
   setHasNotifiedExpired(false);
      return;
    }
    const expiryTime = new
Date(paymentStartTime.getTime() +
15 * 60 * 1000);
    const updateTimer = () => {
      const now = new Date();
      const difference =
expiryTime.getTime() -
now.getTime();
      if (difference <= 0) {
        setTimeLeft({ minutes: 0,
seconds: 0 });
        setIsExpired(true);
        if (!hasNotifiedExpired)
{

setHasNotifiedExpired(true);
          toast.error('⏰ Waktu
pembayaran habis! Produk
dikembalikan ke market.', {
duration: 6000 });
          if (onExpired) {
            setTimeout(() => {
              onExpired(produk);
            }, 3000);
          }
        }
        return;
      }
      const minutes =
Math.floor(difference / (1000 *
60));
      const seconds =
Math.floor((difference % (1000 *
60)) / 1000);
      setTimeLeft({ minutes,
seconds });
      setIsExpired(false);
    };
    updateTimer();
    const interval =
setInterval(updateTimer, 1000);
    return () =>
clearInterval(interval);
  }, [currentStep,
paymentStartTime,
hasNotifiedExpired, onExpired,
produk]);
  const priceInEth =
produk.hargaEth ?
produk.hargaEth.toString() :
(produk.harga /
1000000).toFixed(4);
  const formatRupiah = (amount)
=> {
    return new
Intl.NumberFormat('id-ID', {
style: 'currency', currency:
'IDR', minimumFractionDigits: 0
}).format(amount);
  };
  const estimatedGasCost = 0.004;
  const totalEstimatedCost =
parseFloat(priceInEth) +
estimatedGasCost;
  const sellerWalletAddress =
produk.penjual?.walletAddress ||
produk.user?.walletAddress ||
produk.walletAddress;
  const handleNextStep = async ()
=> {
    if (currentStep === 1) {
      if (user?.role === 'ADMIN')
{
        toast.error('Admin tidak
dapat membeli produk');
        onCancel();
        return;
      }
      try {
        const response = await
transaksiAPI.buatTransaksi(produk
.id);
        if
(!response.data?.sukses) throw
new Error(response.data?.pesan ||
'Gagal membuat transaksi');
        setCurrentStep(2);
        setPaymentStartTime(new
Date());
        if (onPaymentStart)
onPaymentStart(produk,
response.data.data.transaksi);
        window.dispatchEvent(new
CustomEvent('transaction-
created', { detail: { transaksi:
response.data.data.transaksi,
timestamp: Date.now() } }));
        window.dispatchEvent(new
CustomEvent('product-reserved', {
detail: { produkId: produk.id,
transaksiId:
response.data.data.transaksi.id,
timestamp: Date.now() } }));
      } catch (error) {
        let errorMessage = 'Gagal
membuat transaksi';
        if
(error.response?.status === 404)
errorMessage = 'Endpoint API
tidak ditemukan. Pastikan server
berjalan.';
        else if
(error.response?.status === 401)
errorMessage = 'Anda perlu login
ulang.';
        else if
(error.response?.data?.pesan)
errorMessage =
error.response.data.pesan;
```

```
        else if
(error.message?.includes('Network
Error')) errorMessage = 'Tidak
dapat terhubung ke server.
Pastikan server berjalan di
http://localhost:5000';
        else if (error.message)
errorMessage = error.message;
        toast.error('Gagal
membuat transaksi: ' +
errorMessage);
        return;
      }
    } else if (currentStep === 2)
{
      setCurrentStep(3);
      setIsProcessing(true);
      handlePayment();
    }
  };
  const handlePrevStep = () => {
    if (currentStep === 2) {
      setCurrentStep(1);
      setPaymentStartTime(null);
      setTimeLeft(null);
      setIsExpired(false);

setHasNotifiedExpired(false);
    } else if (currentStep === 3)
{
      setCurrentStep(2);
      setIsProcessing(false);
    }
  };
  const handlePayment = async ()
=> {
    try {
      setIsProcessing(true);
      setCurrentStep(3);
    } catch (error) {
      setIsProcessing(false);
      setCurrentStep(2);
    }
  };
  const handlePaymentSuccess =
async (result) => {
    try {
      if (onConfirm) await
onConfirm(result);
      onCancel();
    } catch (error) {
      setIsProcessing(false);
      setCurrentStep(2);
    }
  };
  const handlePaymentCancel = ()
=> {
    setCurrentStep(2);
    setIsProcessing(false);
  };
  useEffect(() => {
    if (isExpired &&
hasNotifiedExpired) {
      const timer = setTimeout(()
=> {
        onCancel();
      }, 5000);
      return () =>
clearTimeout(timer);
    }
  }, [isExpired,
hasNotifiedExpired, onCancel]);
  const StepIndicator = () => (
    <div className="flex items-
center justify-center mb-6">
      <div className="flex items-
center space-x-4">
        <div className="flex
items-center">
          <div className={`w-8 h-
8 rounded-full flex items-center
justify-center text-sm font-
medium ${currentStep >= 1 ? 'bg-
blue-600 text-white' : 'bg-gray-
300 text-gray-
600'}`}>{currentStep > 1 ? '✓' :
'1'}</div>
          <span className="ml-2
text-sm font-medium text-gray-
700">Detail Transaksi</span>
        </div>
        <span className="text-
gray-400">→</span>
        <div className="flex
items-center">
          <div className={`w-8 h-
8 rounded-full flex items-center
justify-center text-sm font-
medium ${currentStep >= 2 ? 'bg-
blue-600 text-white' : 'bg-gray-
300 text-gray-
600'}`}>{currentStep > 2 ? '✓' :
'2'}</div>
          <span className="ml-2
text-sm font-medium text-gray-
700">Pembayaran</span>
        </div>
        <span className="text-
gray-400">→</span>
        <div className="flex
items-center">
          <div className={`w-8 h-
8 rounded-full flex items-center
justify-center text-sm font-
medium ${currentStep >= 3 ? 'bg-
blue-600 text-white' : 'bg-gray-
300 text-gray-600'}`}>3</div>
          <span className="ml-2
text-sm font-medium text-gray-
700">Proses</span>
        </div>
      </div>
    </div>
  );
  const renderStepContent = () =>
{
```

```
    switch (currentStep) {
      case 1:
        return (
          <div>
            <div className="bg-
white border border-gray-200
rounded-lg p-4 sm:p-6 mb-4 sm:mb-
6">
              <h3
className="text-base sm:text-lg
font-semibold text-gray-900 mb-3
sm:mb-4">Informasi Produk</h3>
              <div
className="grid grid-cols-1
sm:grid-cols-2 gap-3 sm:gap-4">
                <div>
                  <p
className="text-xs sm:text-sm
text-gray-600">Nama Produk</p>
                  <p
className="font-medium text-gray-
900 text-sm sm:text-base break-
words">{produk.judulProduk}</p>
                </div>
                <div>
                  <p
className="text-xs sm:text-sm
text-gray-600">Game</p>
                  <p
className="font-medium text-gray-
900 text-sm sm:text-
base">{produk.namaGame}</p>
                </div>
                <div>
                  <p
className="text-xs sm:text-sm
text-gray-600">Kode Produk</p>
                  <p
className="font-medium text-gray-
900 text-sm sm:text-
base">{produk.kodeProduk}</p>
                </div>
                <div>
                  <p
className="text-xs sm:text-sm
text-gray-600">Status</p>
                  <span
className="bg-green-100 text-
green-800 text-xs sm:text-sm px-2
py-1 rounded">Tersedia</span>
                </div>
              </div>
            </div>
            <div className="bg-
white border border-gray-200
rounded-lg p-4 sm:p-6 mb-4 sm:mb-
6">
              <h3
className="text-base sm:text-lg
font-semibold text-gray-900 mb-3
sm:mb-4">Informasi Penjual</h3>
              <div
className="space-y-3">
                <div>
                  <p
className="text-xs sm:text-sm
text-gray-600">Nama Penjual</p>
                  <p
className="font-medium text-gray-
900 text-sm sm:text-
base">{produk.penjual?.nama ||
produk.user?.nama ||
'Anonim'}</p>
                </div>
                <div>
                  <p
className="text-xs sm:text-sm
text-gray-600">Wallet Address</p>
                  <p
className="font-mono text-xs
sm:text-sm text-gray-900 bg-white
p-2 rounded
border">{sellerWalletAddress ?
`${sellerWalletAddress.substring(
0,
6)}...${sellerWalletAddress.subst
ring(sellerWalletAddress.length -
4)}` : 'Tidak tersedia'}</p>

{sellerWalletAddress && <div
className="text-xs sm:text-sm
text-green-600 mt-2">Wallet
Terhubung</div>}
                </div>
              </div>
            </div>
            <div className="bg-
white border border-gray-200
rounded-lg p-4 sm:p-6 mb-4 sm:mb-
6">
              <h3
className="text-base sm:text-lg
font-semibold text-gray-900 mb-3
sm:mb-4">Rincian Harga</h3>
              <div
className="space-y-3">
                <div
className="flex justify-between
items-center">
                  <span
className="text-sm sm:text-base
text-gray-600">Harga
Produk</span>
                  <div
className="text-right">
                    <p
className="font-bold text-lg
sm:text-xl">{priceInEth} ETH</p>
                    <p
className="text-xs sm:text-sm
text-gray-500">≈
{formatRupiah((produk.hargaEth ||
0) * ethToIdrRate)}</p>
                  </div>
                </div>
```

```
                <div
className="flex justify-between
items-center">
                  <span
className="text-sm sm:text-base
text-gray-600">Estimasi Gas
Fee</span>
                  <span
className="font-medium text-sm
sm:text-base">~{estimatedGasCost}
ETH</span>
                </div>
                <div
className="border-t pt-3">
                  <div
className="flex justify-between
items-center">
                    <span
className="font-semibold text-
gray-900 text-sm sm:text-
base">Total Estimasi</span>
                    <span
className="font-bold text-lg
sm:text-
xl">{totalEstimatedCost.toFixed(4
)} ETH</span>
                  </div>
                </div>
              </div>
            </div>
            <div className="mb-4
sm:mb-6">
              <label
className="flex items-start
space-x-3">
                <input
type="checkbox" checked={agreed}
onChange={(e) =>
setAgreed(e.target.checked)}
className="mt-1 h-4 w-4 text-
primary-600 focus:ring-primary-
500 border-gray-300 rounded flex-
shrink-0" />
                <span
className="text-xs sm:text-sm
text-gray-700">Saya telah membaca
dan menyetujui syarat dan
ketentuan transaksi. Saya
memahami bahwa dana akan disimpan
dalam escrow smart contract dan
akan dilepas setelah saya
mengkonfirmasi penerimaan akun
game. <strong className="text-
orange-600">Saya juga memahami
bahwa saya akan memiliki batas
waktu 15 menit untuk
menyelesaikan pembayaran di tahap
selanjutnya.</strong></span>
              </label>
            </div>
            {!sellerWalletAddress
&& (
              <div className="bg-
red-50 border border-red-200
rounded-lg p-3 sm:p-4 mb-4 sm:mb-
6">
                <div>
                  <h4
className="text-xs sm:text-sm
font-medium text-red-800">Tidak
Dapat Melanjutkan!</h4>
                  <p
className="text-xs sm:text-sm
text-red-700 mt-1">Penjual belum
menghubungkan wallet address.
Transaksi tidak dapat
diproses.</p>
                </div>
              </div>
            )}
          </div>
        );
      case 2:
        return (
          <div>
            {isExpired ? (
              <div className="bg-
red-50 border border-red-200
rounded-lg p-4 mb-6">
                <div>
                  <h4
className="font-semibold text-
red-900"> Waktu Pembayaran Telah
Habis</h4>
                  <p
className="text-sm text-red-700
mt-1">Batas waktu 15 menit untuk
menyelesaikan pembayaran telah
berakhir. Produk akan
dikembalikan ke market dan modal
akan ditutup otomatis.</p>
                  <div
className="mt-3 p-3 bg-red-100
rounded-lg">
                    <p
className="text-xs font-medium
text-red-800">📝 Modal akan
ditutup dalam beberapa detik.
Anda dapat mencoba membeli produk
lagi dari halaman market.</p>
                  </div>
                </div>
              </div>
            ) : (<div
className={`rounded-lg p-4 border
mb-6 ${timeLeft &&
timeLeft.minutes < 5 ? 'bg-red-50
border-red-200' : timeLeft &&
timeLeft.minutes < 10 ? 'bg-
orange-50 border-orange-200' :
'bg-white border-gray-200'}`}>
                <div>
                  <h4
className={`font-semibold
${timeLeft && timeLeft.minutes <
```

```
5 ? 'text-red-900' : timeLeft &&
timeLeft.minutes < 10 ? 'text-
orange-900' : 'text-gray-
900'}`}>{!timeLeft ? '⏳ Memulai
Timer Pembayaran...' :
timeLeft.minutes < 5 ? ' Waktu
Pembayaran Hampir Habis!' :
timeLeft.minutes < 10 ? ' Segera
Lakukan Pembayaran' : 'Siap untuk
Pembayaran'}</h4>{timeLeft &&
<div className="text-3xl font-
bold mt-2 mb-
2">{String(timeLeft.minutes).padS
tart(2,
'0')}:{String(timeLeft.seconds).p
adStart(2, '0')}</div>}
                  <p
className={`text-sm mt-1
${timeLeft && timeLeft.minutes <
5 ? 'text-red-700' : timeLeft &&
timeLeft.minutes < 10 ? 'text-
orange-700' : 'text-gray-
700'}`}>{!timeLeft ? 'Timer akan
dimulai dalam beberapa detik...'
: timeLeft.minutes < 5 ? `Anda
hanya memiliki
${timeLeft.minutes} menit
${timeLeft.seconds} detik lagi!
Jika waktu habis, produk akan
kembali ke market.` : `Anda
memiliki ${timeLeft.minutes}
menit ${timeLeft.seconds} detik
untuk menyelesaikan pembayaran.
Proses akan dilanjutkan di
dashboard pembeli.`}</p>
                </div>
              </div>
            )}
            {!isExpired && (
              <>
                <div
className="bg-white border
border-gray-200 rounded-lg p-4
sm:p-6 mb-4 sm:mb-6">
                  <h3
className="text-base sm:text-lg
font-semibold text-gray-900 mb-3
sm:mb-4">Ringkasan
Pembayaran</h3>
                  <div
className="space-y-3"><div
className="flex justify-between
items-center"><span
className="text-sm sm:text-base
text-gray-600">Produk</span><span
className="font-medium text-sm
sm:text-
base">{produk.judulProduk}</span>
</div>
                    <div
className="flex justify-between
items-center"><span
className="text-sm sm:text-base
text-gray-600">Harga</span><span
className="font-medium text-sm
sm:text-base">{priceInEth}
ETH</span></div>
                    <div
className="flex justify-between
items-center"><span
className="text-sm sm:text-base
text-gray-600">Gas Fee
(estimasi)</span><span
className="font-medium text-sm
sm:text-base">~{estimatedGasCost}
ETH</span></div>
                    <div
className="border-t pt-3"><div
className="flex justify-between
items-center"><span
className="font-semibold text-
gray-900 text-sm sm:text-
base">Total</span><span
className="font-bold text-lg
sm:text-
xl">{totalEstimatedCost.toFixed(4
)} ETH</span></div></div>
                    <div
className="flex justify-between
items-center"><span
className="text-sm sm:text-base
text-gray-600">Saldo
Anda</span><span className="font-
medium text-sm sm:text-
base">{userBalance || '0'}
ETH</span></div>
                    <div
className="flex justify-between
items-center"><span
className="text-sm sm:text-base
text-gray-600">Sisa Setelah
Transaksi</span><span
className={`font-medium text-sm
sm:text-base
${parseFloat(userBalance || 0) -
totalEstimatedCost > 0 ? 'text-
green-600' : 'text-red-
600'}`}>~{(parseFloat(userBalance
|| 0) -
totalEstimatedCost).toFixed(4)}
ETH</span></div>
                  </div>
                </div>
                <div
className="bg-white border
border-gray-200 rounded-lg p-4
sm:p-6 mb-4 sm:mb-6">
                  <h3
className="text-base sm:text-lg
font-semibold text-gray-900 mb-3
sm:mb-4">Instruksi
Pembayaran</h3>
                  <div
className="space-y-2 text-xs
sm:text-sm text-gray-700">
```

```
                    <p>1. Klik
tombol "Bayar Sekarang" di
bawah</p>
                    <p>2. Dompet
Anda akan terbuka untuk
konfirmasi</p>
                    <p>3.
Konfirmasi transaksi di dompet
Anda</p>
                    <p>4. Dana
akan disimpan dalam escrow smart
contract</p>
                    <p>5. Setelah
pembayaran berhasil, produk akan
tersimpan di dashboard
pembeli</p>
                  </div>
                </div>
              </>
            )}
          </div>
        );
      case 3:
        return (
          <div>
            <div className="bg-
white border border-gray-200
rounded-lg p-4 mb-6">
              <h3
className="text-lg font-semibold
text-gray-900 mb-2">Pembayaran
dengan Dompet Digital</h3>
              <p className="text-
sm text-gray-700">Silakan
hubungkan dompet blockchain Anda
dan konfirmasi pembayaran untuk
menyelesaikan transaksi.</p>
            </div>

<PembelianKontrakPintar
produk={produk}
onSuccess={handlePaymentSuccess}
onCancel={handlePaymentCancel} />
          </div>
        );
      default:
        return null;
    }
  };
  return (
    <div className="fixed inset-0
bg-black bg-opacity-50 flex
items-center justify-center z-50
p-2 sm:p-4">
      <div className="bg-white
rounded-lg max-w-2xl w-full max-
h-[95vh] sm:max-h-[90vh]
overflow-y-auto">
        <div className="p-4 sm:p-
6">
          <div className="flex
justify-between items-center mb-4
sm:mb-6">
            <div>
              <h2
className="text-lg sm:text-2xl
font-bold text-gray-
900">{currentStep === 1 ?
'Konfirmasi Pembelian' :
currentStep === 2 ? 'Pembayaran'
: 'Memproses Pembayaran'}</h2>
            </div>
            <button
onClick={onCancel}
className="text-gray-600
hover:text-gray-800 p-1"
disabled={currentStep === 3 &&
isProcessing}>Tutup</button>
          </div>
          <StepIndicator />
          {renderStepContent()}
          {currentStep !== 3 && (
            <div className="flex
flex-col sm:flex-row space-y-3
sm:space-y-0 sm:space-x-4 mt-6">
              {!isExpired && (
                <button
onClick={onCancel} className="w-
full sm:flex-1 bg-gray-300 text-
gray-700 py-3 px-4 sm:px-6
rounded-lg hover:bg-gray-400
transition-colors text-sm
sm:text-base">Batal</button>
              )}
              {currentStep === 1
&& (
                <button
onClick={handleNextStep}
disabled={!agreed ||
!sellerWalletAddress}
className="w-full sm:flex-1 bg-
primary-600 text-white py-3 px-4
sm:px-6 rounded-lg hover:bg-
primary-700 disabled:opacity-50
disabled:cursor-not-allowed
transition-colors flex items-
center justify-center space-x-2
text-sm sm:text-
base"><span>Lanjut ke
Pembayaran</span></button>
              )}
              {currentStep === 2
&& !isExpired && (
                <button
onClick={handleNextStep}
disabled={isProcessing}
className="w-full sm:flex-1 bg-
primary-600 text-white py-3 px-4
sm:px-6 rounded-lg hover:bg-
primary-700 disabled:opacity-50
disabled:cursor-not-allowed
transition-colors flex items-
center justify-center text-sm
sm:text-base"><span>{isProcessing
? 'Memproses...' : 'Bayar
Sekarang'}</span></button>
```

```
              )}
              {isExpired && (
                <button
onClick={onCancel} className="w-
full bg-red-600 text-white py-3
px-4 sm:px-6 rounded-lg hover:bg-
red-700 transition-colors text-sm
sm:text-base">Tutup
Modal</button>
              )}
            </div>
          )}
        </div>
      </div>
    </div>
export default
TransactionDetailModal;
```

ProdukAdmin.jsx

```
import React, { useState,
useEffect, useCallback } from
'react';
import { useLocation, useNavigate
} from 'react-router-dom';
import { adminAPI } from
'../../layanan/api';
import AdminNavigation from
'../../komponen/AdminNavigation';
import ConfirmationModal from
'../../komponen/ConfirmationModal
';
import toast from 'react-hot-
toast';
const ProdukAdmin = () => {
  const location = useLocation();
  const navigate = useNavigate();
  const [produk, setProduk] =
useState([]);
  const [loading, setLoading] =
useState(true);
  const [filter, setFilter] =
useState('all');
  const [searchTerm,
setSearchTerm] = useState('');
  const [debouncedSearchTerm,
setDebouncedSearchTerm] =
useState('');
  const [selectedProduk,
setSelectedProduk] =
useState(null);
  const [showDetailModal,
setShowDetailModal] =
useState(false);
  const [currentPage,
setCurrentPage] = useState(1);
  const [totalPages,
setTotalPages] = useState(1);
  const [updateLoading,
setUpdateLoading] =
useState(false);
  const [error, setError] =
useState(null);
  const [showImageViewer,
setShowImageViewer] =
useState(false);
  const [currentImageIndex,
setCurrentImageIndex] =
useState(0);
  const [viewerImages,
setViewerImages] = useState([]);
  const [showDeleteConfirm,
setShowDeleteConfirm] =
useState(false);
  const [productToDelete,
setProductToDelete] =
useState(null);
  const [deleteLoading,
setDeleteLoading] =
useState(false);
  useEffect(() => {
    const searchParams = new
URLSearchParams(location.search);
    const searchParam =
searchParams.get('search');
    const pageParam =
searchParams.get('page');
    if (searchParam) {
      setSearchTerm(searchParam);

setDebouncedSearchTerm(searchPara
m);
    }
    if (pageParam)
setCurrentPage(parseInt(pageParam
) || 1);
  }, [location.search]);
  useEffect(() => {
    const timer = setTimeout(()
=>
setDebouncedSearchTerm(searchTerm
), 500);
    return () =>
clearTimeout(timer);
  }, [searchTerm]);
  useEffect(() => {
    setCurrentPage(1);
  }, [debouncedSearchTerm]);
  useEffect(() => {
    fetchProduk();
  }, [filter, currentPage,
debouncedSearchTerm]);
  const fetchProduk =
useCallback(async () => {
    try {
      setLoading(true);
      setError(null);
      const params = { page:
currentPage, limit: 12,
...(filter !== 'all' && {
statusProduk: filter }),
...(debouncedSearchTerm && {
search: debouncedSearchTerm }),
_t: Date.now() };
```

```
      const response = await
adminAPI.getAllProdukAdmin(params
);
      if (response.data &&
response.data.sukses !== false) {
        const produkData =
response.data.data?.produk ||
response.data.produk || [];
        const paginationData =
response.data.data?.pagination ||
response.data.pagination || {};
        setProduk(produkData);

setTotalPages(paginationData.tota
lPages || 1);
      } else throw new
Error(response.data?.pesan ||
'Gagal memuat data produk');
    } catch (e) {
      setError('Gagal memuat data
produk. Silakan coba lagi.');
      toast.error('Gagal memuat
data produk');
    } finally {
      setLoading(false);
    }
  }, [filter, currentPage,
debouncedSearchTerm]);
  const handleSearchChange =
(value) => setSearchTerm(value);
  useEffect(() => {
    const searchParams = new
URLSearchParams(location.search);
    if (debouncedSearchTerm)
searchParams.set('search',
debouncedSearchTerm); else
searchParams.delete('search');
    searchParams.delete('page');
    const newURL =
searchParams.toString() ?
`${location.pathname}?${searchPar
ams.toString()}` :
location.pathname;
    navigate(newURL, { replace:
true });
  }, [debouncedSearchTerm,
location.pathname, navigate]);
  const handlePaginationChange =
(pageNum) => {
    setCurrentPage(pageNum);
    const searchParams = new
URLSearchParams(location.search);
    if (pageNum > 1)
searchParams.set('page',
pageNum); else
searchParams.delete('page');
    const newURL =
searchParams.toString() ?
`${location.pathname}?${searchPar
ams.toString()}` :
location.pathname;
    navigate(newURL, { replace:
true });
  };
  const handleDeleteProduct =
(produkId) => {
    setProductToDelete(produkId);
    setShowDeleteConfirm(true);
  };
  const confirmDelete = async ()
=> {
    if (!productToDelete) return;
    try {
      setDeleteLoading(true);
      const response = await
adminAPI.deleteProduk(productToDe
lete);
      if (response.data.sukses) {
        await fetchProduk();

setShowDetailModal(false);

setShowDeleteConfirm(false);
        setProductToDelete(null);
        toast.success('Produk
berhasil dihapus!');
      } else throw new
Error(response.data.pesan ||
'Gagal menghapus produk');
    } catch (error) {
      toast.error('Gagal
menghapus produk: ' +
(error.response?.data?.pesan ||
error.message));
    } finally {
      setDeleteLoading(false);
    }
  };
  const cancelDelete = () => {
    setShowDeleteConfirm(false);
    setProductToDelete(null);
    setDeleteLoading(false);
  };
  const getStatusBadge = (status)
=> {
    const statusConfig = { AKTIF:
{ color: 'bg-green-100 text-
green-800', text: 'Aktif' },
PENDING: { color: 'bg-yellow-100
text-yellow-800', text: 'Pending'
}, DITOLAK: { color: 'bg-red-100
text-red-800', text: 'Ditolak' },
TERJUAL: { color: 'bg-blue-100
text-blue-800', text: 'Terjual'
}, DIHAPUS: { color: 'bg-gray-100
text-gray-500', text: 'Dihapus' }
};
    const config =
statusConfig[status] || { color:
'bg-gray-100 text-gray-800',
text: status };
    return <span className={`px-2
py-1 rounded text-xs font-medium
${config.color}`}>{config.text}</
span>;
  };
```

```
  const handleDetailClick =
(produkItem) => {

setSelectedProduk(produkItem);
    setShowDetailModal(true);
  };
  const formatCurrency = (amount)
=> new Intl.NumberFormat('id-ID',
{ style: 'currency', currency:
'IDR', minimumFractionDigits: 0
}).format(amount);
  const openImageViewer =
(images, startIndex = 0) => {
    let processedImages = [];
    if (Array.isArray(images) &&
images.length > 0)
processedImages =
images.filter((img) => img &&
img.trim());
    else if (typeof images ===
'string' && images.trim()) {
      try {
        const parsed =
JSON.parse(images);
        if
(Array.isArray(parsed))
processedImages =
parsed.filter((img) => img &&
img.trim());
        else processedImages =
[images];
      } catch (e) {
        processedImages =
[images];
      }
    }
    if (processedImages.length >
0) {

setViewerImages(processedImages);

setCurrentImageIndex(Math.max(0,
Math.min(startIndex,
processedImages.length - 1)));
      setShowImageViewer(true);
    }
  };
  const nextImage = () =>
setCurrentImageIndex((prev) =>
(prev + 1) %
viewerImages.length);
  const prevImage = () =>
setCurrentImageIndex((prev) =>
(prev - 1 + viewerImages.length)
% viewerImages.length);
  const handleKeyPress = (e) => {
    if (!showImageViewer) return;
    if (e.key === 'ArrowRight')
nextImage(); else if (e.key ===
'ArrowLeft') prevImage(); else if
(e.key === 'Escape')
setShowImageViewer(false);
  };

  React.useEffect(() => {

document.addEventListener('keydow
n', handleKeyPress);
    return () =>
document.removeEventListener('key
down', handleKeyPress);
  }, [showImageViewer,
viewerImages]);
  if (loading) {
    return (
      <div className="min-h-
screen bg-gray-50 p-6">
        <div className="max-w-
[90%] mx-auto">
          <div className="flex
gap-6">
            <div className="flex-
shrink-0"><AdminNavigation
/></div>
            <div className="flex-
1 flex items-center justify-
center h-64"><div
className="animate-spin rounded-
full h-32 w-32 border-b-2 border-
blue-600"></div></div>
          </div>
        </div>
      </div>
    );
  }
  return (
    <div className="min-h-screen
bg-gray-50 p-6">
      <div className="max-w-[90%]
mx-auto">
        <div className="flex gap-
6">
          <div className="flex-
shrink-0"><AdminNavigation
/></div>
          <div className="flex-
1">
            <div className="bg-
white rounded-lg shadow-sm mb-6">
              <div className="px-
6 py-4 border-b border-gray-
200"><h1 className="text-2xl
font-bold text-gray-
900">Manajemen Produk</h1><p
className="text-gray-600 mt-
1">Monitor dan moderasi semua
produk di platform</p></div>
            </div>
            {error && (<div
className="mb-6 bg-red-50 border
border-red-200 rounded-md p-
4"><div className="text-sm text-
red-800">{error}</div><button
onClick={fetchProduk}
className="mt-2 text-sm text-red-
600 hover:text-red-500
underline">Coba
```

```
lagi</button></div>)}<div
className="bg-white rounded-lg
shadow-sm border border-gray-
100"><div className="px-6 py-4
border-b border-gray-200">
<div className="flex flex-col
sm:flex-row sm:items-center
sm:justify-between gap-4"><h2
className="text-lg font-semibold
text-gray-900">Daftar Produk</h2>
<div className="w-full sm:w-
auto"><input type="text"
placeholder="Cari produk..."
value={searchTerm} onChange={(e)
=>
handleSearchChange(e.target.value
)} className="w-full sm:w-64 pl-4
pr-4 py-2 border border-gray-300
rounded-lg focus:ring-2
focus:ring-blue-500 focus:border-
transparent transition" /></div>
</div></div>{produk.length === 0
? (<div className="px-6 py-16
text-center"><h3 className="mt-2
text-lg font-medium text-gray-
900">{searchTerm ? 'Tidak ada
hasil pencarian' : 'Tidak ada
produk'}</h3><p className="mt-1
text-sm text-gray-
500">{searchTerm ? `Tidak
ditemukan produk dengan kata
kunci "${searchTerm}"` : 'Belum
ada produk yang sesuai dengan
filter.'}</p></div>) : (<div
className="overflow-x-auto">
<table className="min-w-full
divide-y divide-gray-200">
<thead className="bg-gray-50">
<tr><th className="px-6 py-3
text-left text-xs font-medium
text-gray-500 uppercase tracking-
wider">No</th>
<th className="px-6 py-3 text-
left text-xs font-medium text-
gray-500 uppercase tracking-
wider">Produk</th>
<th className="px-6 py-3 text-
left text-xs font-medium text-
gray-500 uppercase tracking-
wider">Game</th>
<th className="px-6 py-3 text-
left text-xs font-medium text-
gray-500 uppercase tracking-
wider">Harga</th>
<th className="px-6 py-3 text-
left text-xs font-medium text-
gray-500 uppercase tracking-
wider">Penjual</th>
<th className="px-6 py-3 text-
left text-xs font-medium text-
gray-500 uppercase tracking-
wider">Status</th>
<th className="px-6 py-3 text-
left text-xs font-medium text-
gray-500 uppercase tracking-
wider">Tanggal</th>
<th className="px-6 py-3 text-
left text-xs font-medium text-
gray-500 uppercase tracking-
wider">Aksi</th>
</tr></thead><tbody
className="bg-white divide-y
divide-gray-200">
{produk.map((item, index) => (
<tr key={item.id}
className="hover:bg-gray-50">
<td className="px-6 py-2
whitespace-nowrap"><div
className="text-sm font-medium
text-gray-900">{(currentPage - 1)
* 12 + index + 1}</div></td>
<td className="px-6 py-2
whitespace-nowrap">
<div className="flex items-
center">
<div className="w-10 h-10 bg-
gray-100 rounded-lg flex items-
center justify-center mr-3
overflow-hidden">
{(() => {
let imageUrl = null;
if (Array.isArray(item.gambar) &&
item.gambar.length > 0) imageUrl
= item.gambar[0];
else if (typeof item.gambar ===
'string' && item.gambar.trim()) {
try {
const parsed =
JSON.parse(item.gambar);
if (Array.isArray(parsed) &&
parsed.length > 0) imageUrl =
parsed[0]; else imageUrl =
item.gambar;
} catch (e) { imageUrl =
item.gambar; }
}
return imageUrl ? (
<img src={imageUrl}
alt={item.judulProduk}
className="w-full h-full object-
cover" onError={(e) => {
e.target.style.display = 'none';
e.target.nextElementSibling.style
.display = 'flex'; }} />
) : null;})()}
<div className="w-full h-full
hidden items-center justify-
center text-[10px] text-gray-
400">NO IMAGE</div>
</div><div>
<div className="text-sm font-
medium text-gray-900 max-w-xs
truncate">{item.judulProduk}</div
><div className="text-xs text-
gray-500">Kode:
```

```
{item.kodeProduk}</div></div></di
v></td><td className="px-6 py-2
whitespace-nowrap"><div
className="text-sm text-gray-
900">{item.nama<i>Game</i>}</div></td>
<td className="px-6 py-2
whitespace-nowrap"><div
className="text-sm font-medium
text-blue-600">{item.hargaEth ?
`${item.hargaEth} ETH` :
'N/A'}</div><div className="text-
xs text-gray-500">≈
{formatCurrency(item.harga ||
0)}</div></td>
<td className="px-6 py-2
whitespace-nowrap"><div
className="text-sm text-gray-
900">{item.penjual?.nama ||
'N/A'}</div>{item.penjual?.email
&& <div className="text-xs text-
gray500">{item.penjual.email}</di
v>}</td><td className="px-6 py-2
whitespace-
nowrap">{getStatusBadge(item.stat
usProduk)}</td>
                          <td
className="px-6 py-2 whitespace-
nowrap"><div className="text-sm
text-gray-500">{new
Date(item.dibuatPada).toLocaleDat
eString('id-ID')}</div></td>
<td className="px-6 py-2
whitespace-nowrap">
<button onClick={() =>
handleDetailClick(item)}
className="inline-flex items-
center px-3 py-1 border border-
gray-300 shadow-sm text-xs
leading-4 font-medium rounded-md
text-gray-700 bg-white hover:bg-
gray-50">Detail</button>
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </table>
                </div>
              )}
              {totalPages > 1 &&
(
<div className="px-6 py-4 border-
t border-gray-200"><div
className="flex items-center
justify-between">
<div className="text-sm text-
gray-700">Halaman {currentPage}
dari {totalPages}</div>
<div className="flex items-center
space-x-2">
<button onClick={() =>
handlePaginationChange(Math.max(c
urrentPage - 1, 1))}
disabled={currentPage === 1}
className="px-3 py-2 text-sm
font-medium text-gray-500 bg-
white border border-gray-300
rounded-md hover:bg-gray-50
disabled:opacity-50
disabled:cursor-not-
allowed">Sebelumnya</button>
                    <div
className="flex items-center
space-x-1">
{Array.from({ length: Math.min(5,
totalPages) }, (_, i) => {
const pageNum = i + 1;
const isActive = pageNum ===
currentPage;
                        return
<button key={pageNum} onClick={()
=>
handlePaginationChange(pageNum)}
className={`px-3 py-2 text-sm
font-medium rounded-md ${isActive
? 'bg-primary-600 text-white' :
'text-gray-700 bg-white border
border-gray-300 hover:bg-gray-
50'}`}>{pageNum}</button>;
                      })}
                    </div>
                    <button
onClick={() =>
handlePaginationChange(Math.min(c
urrentPage + 1, totalPages))}
disabled={currentPage ===
totalPages} className="px-3 py-2
text-sm font-medium text-gray-500
bg-white border border-gray-300
rounded-md hover:bg-gray-50
disabled:opacity-50
disabled:cursor-not-
allowed">Selanjutnya</button>
                  </div>
                </div>
              </div>
            )}
          </div>
        </div>
      </div>
      {showDetailModal &&
selectedProduk && (
        <div className="fixed
inset-0 bg-black bg-opacity-50
flex items-center justify-center
z-50 p-4">
          <div className="bg-
white rounded-lg max-w-6xl w-full
max-h-[95vh] overflow-y-auto
shadow-2xl">
            <div className="p-6">
              <div
className="flex justify-between
items-center mb-6">
```

```
                <h3
className="text-xl font-semibold
text-gray-900">Detail Produk</h3>
                <button
onClick={() =>
setShowDetailModal(false)}
className="text-gray-600
hover:text-gray-800 text-sm px-3
py-1 border
rounded">Tutup</button>
              </div>
              <div
className="grid grid-cols-1
lg:grid-cols-2 gap-6">
                <div
className="space-y-4">
                  <div
className="w-full h-64 bg-gray-
100 rounded-lg flex items-center
justify-center overflow-hidden
cursor-pointer hover:bg-gray-200
transition-colors" onClick={() =>
openImageViewer(selectedProduk.ga
mbar, 0)}>
                    {(() => {
                      let
imageUrl = null;
                      if
(Array.isArray(selectedProduk.gam
bar) &&
selectedProduk.gambar.length > 0)
imageUrl =
selectedProduk.gambar[0];
                      else if
(typeof selectedProduk.gambar ===
'string' &&
selectedProduk.gambar.trim()) {
                        try {
                          const
parsed =
JSON.parse(selectedProduk.gambar)
;
                          if
(Array.isArray(parsed) &&
parsed.length > 0) imageUrl =
parsed[0]; else imageUrl =
selectedProduk.gambar;
                        } catch
(e) { imageUrl =
selectedProduk.gambar; }
                      }
                      return
imageUrl ? (
                        <img
src={imageUrl}
alt={selectedProduk.judulProduk}
className="w-full h-full object-
cover" onError={(e) => {
e.target.style.display = 'none';
e.target.nextElementSibling.style
.display = 'flex'; }} />
                      ) : null;
                    })()}
                  <div
className="w-full h-full hidden
items-center justify-center text-
center text-gray-500 text-
sm">Tidak ada gambar</div>
                </div>
                {(() => {
                  let images =
[];
                  if
(Array.isArray(selectedProduk.gam
bar) &&
selectedProduk.gambar.length > 1)
images =
selectedProduk.gambar.slice(1);
                  else if
(typeof selectedProduk.gambar ===
'string' &&
selectedProduk.gambar.trim()) {
                    try {
                      const
parsed =
JSON.parse(selectedProduk.gambar)
;
                      if
(Array.isArray(parsed) &&
parsed.length > 1) images =
parsed.slice(1);
                    } catch (e)
{ images = []; }
                  }
                  if
(images.length === 0) return
null;
                  return (
                    <div
className="grid grid-cols-4 gap-
2">

{images.map((img, index) => (
                        <div
key={index} className="h-20 bg-
gray-200 rounded flex items-
center justify-center overflow-
hidden border border-gray-300">
                          <img
src={img} alt={`Gambar ${index +
2}`} className="w-full h-full
object-cover hover:scale-105
transition-transform duration-200
cursor-pointer" onError={(e) => {
e.target.style.display = 'none';
e.target.nextElementSibling.style
.display = 'flex'; }} onClick={()
=>const allImages = (() => {
if(Array.isArray(selectedProduk.g
ambar)) return
selectedProduk.gambar;
else if (typeof
selectedProduk.gambar ===
'string' &&
selectedProduk.gambar.trim()) {
try { const parsed =
```

```
JSON.parse(selectedProduk.gambar)
; return Array.isArray(parsed) ?
parsed : [selectedProduk.gambar];
} catch (e) { return
[selectedProduk.gambar]; }
return [];
})();
openImageViewer(allImages, index
+ 1);}} />
<div className="w-full h-full
hidden items-center justify-
center text-gray-500 text-xs">NO
IMG</div>
</div>))}</div>);})()}
<div className="bg-white rounded-
lg p-3 shadow-sm border">
                    <div
className="text-xs text-gray-500
mb-2">Informasi Produk</div>
                    <div
className="space-y-1 text-xs">
                      <div
className="flex justify-
between"><span className="text-
gray-600">Status:</span><span
className={`font-medium
${selectedProduk.statusProduk ===
'AKTIF' ? 'text-green-600' :
selectedProduk.statusProduk ===
'TERJUAL' ? 'text-red-600' :
selectedProduk.statusProduk ===
'PENDING' ? 'text-yellow-600' :
selectedProduk.statusProduk ===
'DITOLAK' ? 'text-red-600' :
'text-gray-
600'}`}>{selectedProduk.statusPro
duk === 'TERJUAL' ? 'Terjual' :
selectedProduk.statusProduk ===
'AKTIF' ? 'Aktif' :
selectedProduk.statusProduk ===
'PENDING' ? 'Pending' :
selectedProduk.statusProduk ===
'DITOLAK' ? 'Ditolak' :
'Unknown'}</span></div>
                      <div
className="flex justify-
between"><span className="text-
gray-600">Diposting:</span><span
className="font-medium text-gray-
900">{new
Date(selectedProduk.dibuatPada ||
new
Date().toISOString()).toLocaleDat
eString('id-ID')}</span></div>
                      <div
className="flex justify-
between"><span className="text-
gray-600">Kategori:</span><span
className="font-medium text-gray-
900">{selectedProduk.nama<i>Game</i>}</s
pan></div>
                    </div>
                  </div>
                  </div>
                  <div
className="space-y-4">
                    <div>
                      <h1
className="text-2xl font-bold
text-gray-900 leading-
tight">{selectedProduk.judulProdu
k}</h1>                    <div
className="text-gray-600 text-
base mt-
1">{selectedProduk.nama<i>Game</i>}</div
>
                      <div
className="text-blue-600 text-sm
font-semibold mt-0.5">Kode Akun:
{selectedProduk.kodeProduk}</div>
                    </div>
                    <div
className="border border-gray-400
rounded-lg p-4">
                      <div
className="font-semibold text-
gray-900 mb-3 text-sm uppercase
tracking-wide">DESKRIPSI</div>
                      <div
className="text-sm text-gray-
700"><div className="whitespace-
pre-wrap leading-relaxed min-h-
[100px]">{selectedProduk.deskrips
i ? selectedProduk.deskripsi :
<span className="text-gray-500
italic">Belum ada deskripsi untuk
produk ini.</span>}</div></div>
                    </div>
                    <div
className="py-3">
                      <div
className="flex items-center
justify-between border-b border-
gray-300 pb-2 mb-2"><span
className="text-gray-700 font-
medium">Harga</span><span
className="text-lg font-bold
text-blue-
600">{selectedProduk.hargaEth ?
`${selectedProduk.hargaEth} ETH`
: '0.1 ETH'}</span></div>
<div className="text-right text-
gray-600">≈
{formatCurrency(selectedProduk.ha
rga || 2000000)}</div>
                    </div>
                    <div>
                      <h4
className="font-medium text-gray-
900 mb-1">Penjual</h4>

{selectedProduk.penjual?.namaToko
&& <div className="text-sm text-
blue-700 font-semibold mb-
1">{selectedProduk.penjual.namaTo
ko}</div>}
```

```
                  <div
className="text-sm text-gray-
700">{selectedProduk.penjual?.nam
a || 'Belum
diisi'}{selectedProduk.penjual?.e
mail && <span className="text-
gray-400">
({selectedProduk.penjual.email})<
/span>}</div>
                </div>
                <div
className="bg-gray-50 rounded-lg
p-4 border mt-4">
                  <h4
className="font-medium text-gray-
900 mb-2">Aksi Admin</h4>

{selectedProduk.statusProduk ===
'TERJUAL' ||
selectedProduk.statusProduk ===
'SELESAI' ? (
                    <div
className="bg-blue-50 border
border-blue-200 rounded-lg p-4">
                      <h5
className="font-medium text-blue-
800">Produk Sudah Terjual</h5>
                      <p
className="text-blue-700 text-sm
mt-1">Produk ini sudah terjual
dan transaksi telah selesai.
Tidak dapat dihapus karena sudah
ada riwayat transaksi.</p>
                    </div>
                  ) : (
                    <button
onClick={() =>
handleDeleteProduct(selectedProdu
k.id)} disabled={updateLoading}
className="w-full bg-red-600
text-white py-2 px-4 rounded-lg
hover:bg-red-700
disabled:opacity-
50">{updateLoading ?
'Menghapus...' : 'Hapus
Produk'}</button>
                  )}
                </div>
              </div>
            </div>
          </div>
        </div>
      )}
      {showImageViewer &&
viewerImages.length > 0 && (
        <div className="fixed
inset-0 bg-black bg-opacity-90
flex items-center justify-center
z-[60] p-4">
          <div
className="relative max-w-7xl
max-h-full w-full h-full flex
items-center justify-center">
            <button onClick={()
=> setShowImageViewer(false)}
className="absolute top-4 right-4
z-10 bg-black bg-opacity-50 text-
white px-3 py-1 rounded hover:bg-
opacity-70">Tutup</button>
            {viewerImages.length
> 1 && <button
onClick={prevImage}
className="absolute left-4 top-
1/2 -translate-y-1/2 z-10 bg-
black bg-opacity-50 text-white
px-3 py-2 rounded hover:bg-
opacity-70" aria-label="Gambar
Sebelumnya"><</button>}
            {viewerImages.length
> 1 && <button
onClick={nextImage}
className="absolute right-4 top-
1/2 -translate-y-1/2 z-10 bg-
black bg-opacity-50 text-white
px-3 py-2 rounded hover:bg-
opacity-70" aria-label="Gambar
Selanjutnya"></button>}
            <div
className="relative max-w-full
max-h-full flex items-center
justify-center">
              <img
src={viewerImages[currentImageInd
ex]} alt={`Gambar
${currentImageIndex + 1}`}
className="max-w-full max-h-full
object-contain rounded-lg shadow-
2xl" onError={(e) => {
e.target.style.display = 'none';
e.target.nextElementSibling.style
.display = 'flex'; }} />
              <div className="w-
full h-64 hidden items-center
justify-center text-center bg-
gray-800 rounded-lg text-gray-
300">Gambar tidak dapat
dimuat</div>
            </div>
            <div
className="absolute bottom-4
left-1/2 -translate-x-1/2 z-
10"><div className="bg-black bg-
opacity-50 text-white px-4 py-2
rounded-full text-
sm">{currentImageIndex + 1} dari
{viewerImages.length}</div></div>
            {viewerImages.length
> 1 && (
              <div
className="absolute bottom-16
left-1/2 -translate-x-1/2 z-10">
                <div
className="flex space-x-2 bg-
```

```
black bg-opacity-50 p-2 rounded-
lg max-w-md overflow-x-auto">

{viewerImages.map((img, index) =>
(
                    <button
key={index} onClick={() =>
setCurrentImageIndex(index)}
className={`flex-shrink-0 w-12 h-
12 rounded overflow-hidden
border-2 transition-all ${index
=== currentImageIndex ? 'border-
blue-500 opacity-100' : 'border-
gray-500 opacity-60
hover:opacity-80'}`}>
                      <img
src={img} alt={`Thumbnail ${index
+ 1}`} className="w-full h-full
object-cover" onError={(e) => {
e.target.style.display = 'none';
e.target.nextElementSibling.style
.display = 'flex'; }} />
                      <div
className="w-full h-full hidden
items-center justify-center bg-
gray-700 text-gray-300 text-
[10px]">NO</div>
                    </button>
                  ))}
                </div>
              </div>
            )}
            <div
className="absolute top-4 left-4
z-10 bg-black bg-opacity-50 text-
white p-2 rounded text-xs">
              <div>ESC:
Tutup</div>

{viewerImages.length > 1 &&
<div>← →: Navigasi</div>}
            </div>
          </div>
        </div>
      )}
      <ConfirmationModal
isOpen={showDeleteConfirm}
onClose={cancelDelete}
onConfirm={confirmDelete}
title="Hapus Produk"
message="Yakin ingin menghapus
produk ini? Tindakan ini tidak
dapat dibatalkan."
confirmText="Hapus"
cancelText="Batal" type="danger"
loading={deleteLoading} />
    </div>
  );
export default ProdukAdmin;
```

kontrolerProduk.js

```
const { PrismaClient } =
require('@prisma/client');
const { cleanImageData,
parseImageData,
formatProductWithImages,
validateImageRequest } =
require('../utils/imageHelper');
const prisma = new
PrismaClient();
const simplePriceService = {
  validateEthAmount: (ethAmount)
=> {
    const amount =
parseFloat(ethAmount);
    if (isNaN(amount) || amount
<= 0) return { valid: false,
message: 'Jumlah ETH harus berupa
angka positif' };
    if (amount < 0.001) return {
valid: false, message: 'Jumlah
minimal 0.001 ETH' };
    if (amount > 10) return {
valid: false, message: 'Jumlah
maksimal 10 ETH' };
    return { valid: true };
  },
  ethToIdr: async (ethAmount) =>
parseFloat(ethAmount) * 65000000,
  formatEth: (amount) =>
parseFloat(amount).toFixed(6) + '
ETH',
  formatRupiah: (amount) => new
Intl.NumberFormat('id-
ID',{style:'currency',currency:'I
DR',minimumFractionDigits:0,maxim
umFractionDigits:0}).format(amoun
t)
};
let priceConversionService;
try { priceConversionService =
require('../services/priceConvers
ionService'); } catch {
priceConversionService =
simplePriceService; }
const generateKodeProduk = async
(nama_Game_) => {
  const _game_CodeMap = {'Mobile
Legends':'ML','Free
Fire':'FF','PUBG
Mobile':'PM','Genshin
Impact':'GI'};
  let _game_Code =
_game_CodeMap[nama_Game_];
  if (!_game_Code) _game_Code =
nama_Game_.replace(/[^a-zA-
Z]/g,'').substring(0,3).toUpperCa
se();
  const count = await
prisma.produk.count({ where: {
nama_Game_ } });
  let kode; let counter = count +
1;
  do {
    kode = counter <= 999 ?
`${_game_Code}-
```

```
${String(counter).padStart(3,'0')
}` : `${gameCode}-${counter}`;
    const existing = await
prisma.produk.findUnique({ where:
{ kodeProduk: kode } });
    if (!existing) break;
    counter++;
  } while (true);
  return kode;
};
const tambahProduk = async
(req,res) => {
  try {
    const userId =
req.user.userId;
    const { judulProduk,
namaGame, deskripsi, hargaEth,
gambar } = req.body;
    const user = await
prisma.user.findUnique({ where: {
id: userId } });
    if (!user) return
res.status(404).json({
sukses:false, pesan:'User tidak
ditemukan' });
    if (user.role !== 'PENJUAL')
return res.status(403).json({
sukses:false, pesan:'Hanya
penjual yang dapat menambahkan
produk.' });
    if (!judulProduk || !namaGame
|| !hargaEth) return
res.status(400).json({
sukses:false, pesan:'Judul
produk, nama game, dan harga ETH
wajib diisi' });
    const ethValidation =
priceConversionService.validateEt
hAmount(hargaEth);
    if (!ethValidation.valid)
return res.status(400).json({
sukses:false, pesan:
ethValidation.message });
    let hargaIdr;
    try { hargaIdr = await
priceConversionService.ethToIdr(h
argaEth); } catch { hargaIdr =
parseFloat(hargaEth) * 65000000;
}
    let kodeProduk;
    try { kodeProduk = await
generateKodeProduk(namaGame); }
catch { kodeProduk = `PROD-
${Date.now()}`; }
    const produkId =
`prod_${Date.now()}_${Math.random
().toString(36).substr(2,9)}`;
    const imageValidation =
validateImageRequest(gambar);
    if (!imageValidation.valid)
return res.status(400).json({
sukses:false,
pesan:imageValidation.message });
    const gambarData =
imageValidation.cleanData;
    const produkData = {
id:produkId, kodeProduk,
judulProduk:judulProduk.trim(),
namaGame:namaGame.trim(),
deskripsi:deskripsi ?
deskripsi.trim() : null,
harga:Math.round(hargaIdr),
hargaEth:parseFloat(hargaEth),
gambar:gambarData,
penjualId:userId,
statusJual:true, dibuatPada:new
Date() };
    const produk = await
prisma.produk.create({ data:
produkData, include:{ user:{
include:{ profile:true } } } });
    res.status(201).json({
sukses:true, pesan:'Produk
berhasil ditambahkan!', data:{
produk:{ id:produk.id,
kodeProduk:produk.kodeProduk,
judulProduk:produk.judulProduk,
namaGame:produk.namaGame,
deskripsi:produk.deskripsi,
harga:produk.harga,
hargaEth:produk.hargaEth,
hargaFormatted:{
eth:priceConversionService.format
Eth(produk.hargaEth),
idr:priceConversionService.format
Rupiah(produk.harga) },
gambar:produk.gambar,
statusJual:produk.statusJual,
dibuatPada:produk.dibuatPada,
penjual:{
nama:produk.user.profile?.nama ||
'Anonim', email:produk.user.email
} } } });
  } catch (error) {
    if (error.code === 'P2002')
return res.status(400).json({
sukses:false, pesan:'Kode produk
sudah ada, silakan coba lagi' });
    if (error.code === 'P2003')
return res.status(400).json({
sukses:false, pesan:'User tidak
ditemukan' });
    res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server: ' +
error.message });
  }
};
const ambilSemuaProduk = async
(req,res) => {
  try {
    const { page=1, limit=12,
search, namaGame, hargaMin,
hargaMax, hargaEthMin,
hargaEthMax, sortBy='dibuatPada',
sortOrder='desc' } = req.query;
```

```
    const where = {
      statusJual:true,
      statusProduk:'AKTIF',
      ...(search && { OR:[{
judulProduk:{ contains:search }
},{ deskripsi:{ contains:search }
},{ nama_Game:{ contains:search }
}] }),
      ...(nama_Game && {
nama_Game:{ contains:nama_Game }
}),
      ...(hargaEthMin && {
hargaEth:{
gte:parseFloat(hargaEthMin) } }),
      ...(hargaEthMax && {
hargaEth:{
lte:parseFloat(hargaEthMax) } }),
      ...(!hargaEthMin &&
!hargaEthMax && hargaMin && {
harga:{ gte:parseInt(hargaMin) }
}),
      ...(!hargaEthMin &&
!hargaEthMax && hargaMax && {
harga:{ lte:parseInt(hargaMax) }
})
    };
    if (hargaEthMin &&
(isNaN(parseFloat(hargaEthMin))
|| parseFloat(hargaEthMin) < 0))
return res.status(400).json({
sukses:false, pesan:'Harga ETH
minimum tidak valid' });
    if (hargaEthMax &&
(isNaN(parseFloat(hargaEthMax))
|| parseFloat(hargaEthMax) < 0))
return res.status(400).json({
sukses:false, pesan:'Harga ETH
maksimum tidak valid' });
    if (hargaEthMin &&
hargaEthMax &&
parseFloat(hargaEthMin) >
parseFloat(hargaEthMax)) return
res.status(400).json({
sukses:false, pesan:'Harga ETH
minimum tidak boleh lebih besar
dari maksimum' });
    const validSortFields =
['dibuatPada','harga','hargaEth',
'judulProduk','nama_Game'];
    const validSortOrders =
['asc','desc'];
    if
(!validSortFields.includes(sortBy
)) return res.status(400).json({
sukses:false, pesan:'Field
sorting tidak valid' });
    if
(!validSortOrders.includes(sortOr
der)) return
res.status(400).json({
sukses:false, pesan:'Order
sorting tidak valid' });
    const orderBy = {};
orderBy[sortBy] = sortOrder;
    const pageNum = Math.max(1,
parseInt(page) || 1);
    const limitNum = Math.min(50,
Math.max(1, parseInt(limit) ||
12));
    const skip = (pageNum - 1) *
limitNum;
    const [produk,total] = await
Promise.all([
      prisma.produk.findMany({
where, orderBy, skip,
take:limitNum, include:{ user:{
include:{ profile:true } } } }),
      prisma.produk.count({ where
})
    ]);
    const produkFormatted =
produk.map(p => ({
      id:p.id,
      kodeProduk:p.kodeProduk,
      judulProduk:p.judulProduk,
      nama_Game:p.nama_Game,
      deskripsi:p.deskripsi,
      harga:p.harga,
      hargaEth:p.hargaEth,
      gambar:p.gambar,
      statusJual:p.statusJual,
      dibuatPada:p.dibuatPada,
      penjual:{ id:p.user.id,
nama:p.user.profile?.nama ||
'Anonim', email:p.user.email,
walletAddress:p.user.walletAddres
s }
    }));
    res.json({ sukses:true,
data:{ produk:produkFormatted,
pagination:{ currentPage:pageNum,
totalPages:Math.ceil(total/limitN
um), totalItems:total,
itemsPerPage:limitNum } } });
  } catch (error) {
    if (error.code === 'P2025')
return res.status(404).json({
sukses:false, pesan:'Data tidak
ditemukan' });
    if (error.code === 'P2002')
return res.status(400).json({
sukses:false, pesan:'Data
duplikat ditemukan' });
    if (error.name ===
'ValidationError') return
res.status(400).json({
sukses:false, pesan:'Data tidak
valid: ' + error.message });
    const page = Math.max(1,
parseInt(req.query.page) || 1);
    const limit = Math.min(50,
Math.max(1,
parseInt(req.query.limit) ||
12));
```

```
    res.status(200).json({
sukses:true, data:{ produk:[],
pagination:{ currentPage:page,
totalPages:0, totalItems:0,
itemsPerPage:limit } },
pesan:'Tidak ada produk
ditemukan' });
  }
};
const ambilProdukById = async
(req,res) => {
  try {
    const { id } = req.params;
    const produk = await
prisma.produk.findUnique({
where:{ id }, include:{ user:{
include:{ profile:true } } } });
    if (!produk) return
res.status(404).json({
sukses:false, pesan:'Produk tidak
ditemukan' });
    res.json({ sukses:true,
data:{ produk:{ id:produk.id,
kodeProduk:produk.kodeProduk,
judulProduk:produk.judulProduk,
nama_Game_:produk.nama_Game_,
deskripsi:produk.deskripsi,
harga:produk.harga,
hargaEth:produk.hargaEth,
gambar:produk.gambar,
statusJual:produk.statusJual,
statusProduk:produk.statusProduk,
dibuatPada:produk.dibuatPada,
diperbaruiPada:produk.diperbaruiP
ada, penjual:{ id:produk.user.id,
nama:produk.user.profile?.nama ||
'Anonim',
email:produk.user.email,
walletAddress:produk.user.walletA
ddress } } } });
  } catch {
res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server' }); }
};
const updateProduk = async
(req,res) => {
  try {const userId =
req.user.userId;const { id } =
req.params;
    const { judulProduk,
nama_Game_, deskripsi, hargaEth,
gambar, statusJual } = req.body;
    const produkExisting = await
prisma.produk.findUnique({
where:{ id }, include:{
transaksi:{ where:{ status:{
in:['DIBAYAR_SMARTCONTRACT','MENU
NGGU_KIRIM_AKUN','DIKIRIM','DIKON
FIRMASI_PEMBELI','SELESAI'] } } }
} });
    if (!produkExisting) return
res.status(404).json({
sukses:false, pesan:'Produk tidak
ditemukan' });
    if (produkExisting.penjualId
!== userId) return
res.status(403).json({
sukses:false, pesan:'Anda tidak
memiliki akses untuk mengubah
produk ini' });
    if
(produkExisting.statusProduk ===
'TERJUAL') return
res.status(400).json({
sukses:false, pesan:'Produk yang
sudah terjual tidak dapat diedit'
});
    const transaksiDibayar =
produkExisting.transaksi.find(t
=>
['DIBAYAR_SMARTCONTRACT','MENUNGG
U_KIRIM_AKUN','DIKIRIM','DIKONFIR
MASI_PEMBELI','SELESAI'].includes
(t.status));
    if (transaksiDibayar) return
res.status(400).json({
sukses:false, pesan:'Produk yang
sudah dibayar tidak dapat diedit.
Transaksi sedang berlangsung.'
});
    let updateData = {};
    if (judulProduk !==
undefined) updateData.judulProduk
= judulProduk.trim();
    if (nama_Game_ !== undefined)
updateData.nama_Game_ =
nama_Game_.trim();
    if (deskripsi !== undefined)
updateData.deskripsi = deskripsi
? deskripsi.trim() : null;
    if (statusJual !== undefined)
updateData.statusJual =
statusJual;
    if (hargaEth !== undefined) {
      const ethValidation =
priceConversionService.validateEt
hAmount(hargaEth);
      if (!ethValidation.valid)
return res.status(400).json({
sukses:false,
pesan:ethValidation.message });
      let hargaIdr;
      try { hargaIdr = await
priceConversionService.ethToIdr(h
argaEth); } catch { hargaIdr =
parseFloat(hargaEth) * 65000000;
}
      updateData.hargaEth =
parseFloat(hargaEth);
      updateData.harga =
Math.round(hargaIdr);
    }
    if (gambar !== undefined) {
```

```
      const imageValidation =
validateImageRequest(gambar,false
);
      if (!imageValidation.valid)
return res.status(400).json({
sukses:false,
pesan:imageValidation.message });
      if
(imageValidation.cleanData !==
null) updateData.gambar =
imageValidation.cleanData;
    }
    const produkUpdate = await
prisma.produk.update({ where:{ id
}, data:updateData, include:{
user:{ include:{ profile:true } }
} });
    res.json({ sukses:true,
pesan:'Produk berhasil
diperbarui!', data:{ produk:{
id:produkUpdate.id,
kodeProduk:produkUpdate.kodeProdu
k,
judulProduk:produkUpdate.judulPro
duk,
namaGame:produkUpdate.namaGame,
deskripsi:produkUpdate.deskripsi,
harga:produkUpdate.harga,
hargaEth:produkUpdate.hargaEth,
hargaFormatted:{
eth:priceConversionService.format
Eth(produkUpdate.hargaEth),
idr:priceConversionService.format
Rupiah(produkUpdate.harga) },
gambar:produkUpdate.gambar,
statusJual:produkUpdate.statusJua
l,
statusProduk:produkUpdate.statusP
roduk,
diperbaruiPada:produkUpdate.diper
baruiPada, penjual:{
nama:produkUpdate.user.profile?.n
ama || 'Anonim' } } } });
  } catch {
res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server' }); }
};
const hapusProduk = async
(req,res) => {
  try {
    const userId =
req.user.userId;
    const { id } = req.params;
    const produk = await
prisma.produk.findUnique({
where:{ id }, include:{
transaksi:{ select:{ id:true,
status:true, kodeTransaksi:true }
} } });
    if (!produk) return
res.status(404).json({
sukses:false, pesan:'Produk tidak
ditemukan' });
    if (produk.penjualId !==
userId) return
res.status(403).json({
sukses:false, pesan:'Anda tidak
memiliki akses untuk menghapus
produk ini' });
    if (produk.statusProduk ===
'TERJUAL') return
res.status(400).json({
sukses:false, pesan:'Produk yang
sudah terjual tidak dapat
dihapus' });
    const transaksiAktif =
produk.transaksi.find(t =>
['MENUNGGU_PEMBAYARAN','DIBAYAR_S
MARTCONTRACT','MENUNGGU_KIRIM_AKU
N','DIKIRIM','DIKONFIRMASI_PEMBEL
I'].includes(t.status));
    if (transaksiAktif) {
      const statusMessage =
transaksiAktif.status ===
'MENUNGGU_PEMBAYARAN' ? 'menunggu
pembayaran' : 'sudah dibayar dan
sedang diproses';
      return
res.status(400).json({
sukses:false, pesan:`Tidak dapat
menghapus produk yang memiliki
transaksi ${statusMessage}. Kode
transaksi:
${transaksiAktif.kodeTransaksi}`
});
    }
    await prisma.produk.delete({
where:{ id } });
    res.json({ sukses:true,
pesan:'Produk berhasil dihapus!'
});
  } catch {
res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server' }); }
};
const ambilProdukSaya = async
(req,res) => {
  try {
    const userId =
req.user?.userId;
    if (!userId) return
res.status(401).json({
sukses:false, pesan:'User ID
tidak ditemukan. Silakan login
ulang.' });
    const { page=1, limit=10 } =
req.query;
    const skip = (parseInt(page)
- 1) * parseInt(limit);
    const [produk,total] = await
Promise.all([
      prisma.produk.findMany({
where:{ penjualId:userId },
```

```
orderBy:{ dibuatPada:'desc' },
skip, take:parseInt(limit),
include:{ transaksi:{ select:{
id:true, status:true,
kodeTransaksi:true,
dibuatPada:true,
user_transaksi_pembeliIdTouser:{
select:{ profile:{ select:{
nama:true } } } } } } }),
      prisma.produk.count({
where:{ penjualId:userId } })
    ]);
    const produkFormatted =
produk.map(p => {
      try {
        const transaksiDibayar =
p.transaksi?.find(t =>
['DIBAYAR_SMARTCONTRACT','MENUNGG
U_KIRIM_AKUN','DIKIRIM','DIKONFIR
MASI_PEMBELI','SELESAI'].includes
(t.status));
        const transaksiAktif =
p.transaksi?.filter(t =>
['MENUNGGU_PEMBAYARAN','DIBAYAR_S
MARTCONTRACT','DIKIRIM'].includes
(t.status)) || [];
        return {
          id:p.id,

kodeProduk:p.kodeProduk,
judulProduk:p.judulProduk,
          nama_Game_:p.nama_Game_,
          deskripsi:p.deskripsi,
          harga:p.harga,
          hargaEth:p.hargaEth,
          gambar:p.gambar,

statusJual:p.statusJual,

statusProduk:p.statusProduk,

dibuatPada:p.dibuatPada,

diperbaruiPada:p.diperbaruiPada,

totalTransaksi:p.transaksi?.lengt
h || 0,
transaksiAktif:transaksiAktif.len
gth,canEdit:!transaksiDibayar &&
p.statusProduk !== 'TERJUAL',
canDelete:!transaksiAktif.length
&& p.statusProduk !== 'TERJUAL',
canToggleStatus:!transaksiDibayar
&& p.statusProduk !== 'TERJUAL',

isPaid:!!transaksiDibayar,
          isSold:p.statusProduk
=== 'TERJUAL',
          statusInfo:{ message:
transaksiDibayar ? `Produk sudah
dibayar
(${transaksiDibayar.kodeTransaksi
})` : p.statusProduk ===
'TERJUAL' ? 'Produk sudah
terjual' : transaksiAktif.length
> 0 ? `Ada
${transaksiAktif.length}
transaksi aktif` : 'Produk
tersedia untuk dijual',
canModify: !transaksiDibayar &&
p.statusProduk !== 'TERJUAL' },
          latestTransaction:
p.transaksi?.length > 0 ? {
kodeTransaksi:p.transaksi[0].kode
Transaksi,
status:p.transaksi[0].status,
pembeli:p.transaksi[0].user_trans
aksi_pembeliIdTouser?.profile?.na
ma || 'Anonim',
dibuatPada:p.transaksi[0].dibuatP
ada } : null
        };
      } catch {
        return { id:p.id,
kodeProduk:p.kodeProduk,
judulProduk:p.judulProduk,
nama_Game_:p.nama_Game_,
deskripsi:p.deskripsi,
harga:p.harga,
hargaEth:p.hargaEth,
gambar:p.gambar,
statusJual:p.statusJual,
statusProduk:p.statusProduk,
dibuatPada:p.dibuatPada,
diperbaruiPada:p.diperbaruiPada,
totalTransaksi:0,
transaksiAktif:0, canEdit:false,
canDelete:false,
canToggleStatus:false,
isPaid:false, isSold:false,
statusInfo:{ message:'Error
memuat status', canModify:false
}, latestTransaction:null };
      }
    });
    res.json({ sukses:true,
data:{ produk:produkFormatted,
pagination:{
currentPage:parseInt(page),
totalPages:Math.ceil(total/parseI
nt(limit)), totalItems:total,
itemsPerPage:parseInt(limit) } }
});
  } catch (error)
{res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server: ' +
error.message, error:
process.env.NODE_ENV ===
'development' ? error.stack :
undefined });
  }
};
const ambil_Game_Populer = async
(req,res) => {
  try {
```

```
    const gameStats = await
prisma.produk.groupBy({
by:['namaGame'], where:{
statusJual:true, statusProduk:{
not:'TERJUAL' } }, _count:{
namaGame:true }, orderBy:{
_count:{ namaGame:'desc' } } });
    const defaultGames = ['Mobile
Legends','Free Fire','PUBG
Mobile','Genshin Impact'];
    const gameStatsMap = new
Map(); gameStats.forEach(g => {
gameStatsMap.set(g.namaGame,
g._count.namaGame); });
    const gamePopuler =
defaultGames.map(n => ({
namaGame:n,
jumlahProduk:gameStatsMap.get(n)
|| 0 }));
    gameStats.forEach(g => { if
(!defaultGames.includes(g.namaGam
e)) gamePopuler.push({
namaGame:g.namaGame,
jumlahProduk:g._count.namaGame
}); });
    res.json({ sukses:true,
data:{ gamePopuler } });
  } catch {
res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server' }); }
};
const ambilSuggestedPrices =
async (req,res) => {
  try {
    const suggestions = await
priceConversionService.getSuggest
edPrices();
    const rates = await
priceConversionService.getCurrent
Rates();
    res.json({ sukses:true,
data:{ suggestions, rates, info:{
minEth:'0.001', maxEth:'10.000',
lastUpdate:rates.lastUpdate } }
});
  } catch {
res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server' }); }
};const konversiEthKeIdr = async
(req,res) => {
  try {
    const { ethAmount } =
req.query;
    if (!ethAmount) return
res.status(400).json({
sukses:false, pesan:'Parameter
ethAmount diperlukan' });
    const validation =
priceConversionService.validateEt
hAmount(ethAmount);
    if (!validation.valid) return
res.status(400).json({
sukses:false,
pesan:validation.message });
    const idrAmount = await
priceConversionService.ethToIdr(e
thAmount);
    const rates = await
priceConversionService.getCurrent
Rates();
    res.json({ sukses:true,
data:{
ethAmount:parseFloat(ethAmount),
idrAmount, formatted:{
eth:priceConversionService.format
Eth(ethAmount),
idr:priceConversionService.format
Rupiah(idrAmount) }, rates } });
  } catch {
res.status(500).json({
sukses:false, pesan:'Terjadi
kesalahan server' }); }
};
module.exports = { tambahProduk,
ambilSemuaProduk,
ambilProdukById, updateProduk,
hapusProduk, ambilProdukSaya,
ambilGamePopuler,
ambilSuggestedPrices,
konversiEthKeIdr };
```

- **Daftar.jsx**

```
import React, { useState,
useEffect } from 'react';
import { Link, useNavigate,
useLocation } from 'react-router-
dom';
import { useAuth } from
'../../konteks/AuthContext';
const Daftar = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const { register, loading,
isAuthenticated } = useAuth();
  const [formData, setFormData] =
useState({
    nama: '',
    email: '',
    password: '',
    confirmPassword: '',
    nomor_telepon: '',
    role: 'PEMBELI'
  });
  const [errors, setErrors] =
useState({});
  const [showPassword,
setShowPassword] =
useState(false);
  const [showConfirmPassword,
setShowConfirmPassword] =
useState(false);

  useEffect(() => {
    if (isAuthenticated) {
```

```
      navigate('/beranda');
    }
  }, [isAuthenticated,
navigate]);

  const handleChange = (e) => {
    const { name, value } =
e.target;
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));
    if (errors[name]) {
      setErrors(prev => ({
        ...prev,
        [name]: ''
      }));
    }
  };

  const validateForm = () => {
    const newErrors = {};
    if (!formData.nama.trim()) {
      newErrors.nama = 'Nama
wajib diisi';
    }
    if (!formData.email.trim()) {
      newErrors.email = 'Email
wajib diisi';
    } else if
(!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.te
st(formData.email)) {
      newErrors.email = 'Format
email tidak valid';
    }
    if (!formData.password) {
      newErrors.password =
'Password wajib diisi';
    } else if
(formData.password.length < 8) {
      newErrors.password =
'Password minimal 8 karakter';
    } else if (!/(?=.*[a-
z])(?=.*[A-
Z])(?=.*\d)/.test(formData.passwo
rd)) {
      newErrors.password =
'Password harus mengandung huruf
besar, huruf kecil, dan angka';
    }
    if
(!formData.confirmPassword)newErr
ors.confirmPassword = 'Konfirmasi
password wajib diisi';
    } else if (formData.password
!== formData.confirmPassword) {
      newErrors.confirmPassword =
'Password tidak cocok';
    }
    if (formData.role ===
'PENJUAL') {
      if (!formData.nomor_telepon
||
!formData.nomor_telepon.trim()) {
        newErrors.nomor_telepon =
'Nomor telepon wajib diisi untuk
penjual';
      } else {
        const phoneRegex =
/^(\+62|62|0)[0-9]{9,13}$/;
        if
(!phoneRegex.test(formData.nomor_
telepon.replace(/\s/g, ''))) {
          newErrors.nomor_telepon
= 'Format nomor telepon tidak
valid (contoh: 08123456789)';
        }
      }
    } else if
(formData.nomor_telepon &&
formData.nomor_telepon.trim()) {
      const phoneRegex =
/^(\+62|62|0)[0-9]{9,13}$/;
      if
(!phoneRegex.test(formData.nomor_
telepon.replace(/\s/g, ''))) {
        newErrors.nomor_telepon =
'Format nomor telepon tidak valid
(contoh: 08123456789)';
      }
    }
    setErrors(newErrors);
    return
Object.keys(newErrors).length ===
0;
  };

  const handleSubmit = async (e)
=> {
    e.preventDefault();
    if (!validateForm()) {
      return;
    }
    try {
      const { confirmPassword,
...dataToSend } = formData;
      if
(!dataToSend.nomor_telepon ||
!dataToSend.nomor_telepon.trim())
{
        delete
dataToSend.nomor_telepon;
      }
      const result = await
register(dataToSend);
      if (result.success) {
        navigate('/masuk');
      } else {
        setErrors({ general:
result.message || 'Registrasi
gagal' });
      }
    } catch (error) {
```

```
      setErrors({ general:
'Terjadi kesalahan. Silakan coba
lagi.' });
    }
  };

  return (
    <div className="min-h-screen
bg-gray-50 flex flex-col">
      <main className="flex-grow
flex items-center justify-center
py-12 px-4 sm:px-6 lg:px-8">
        <div className="max-w-md
w-full space-y-8 bg-white p-8
rounded-lg shadow-md">
          <div className="flex
justify-start">
            <button
              onClick={() => {
                const from =
location.state?.from;
                if (from ===
'/masuk') {

navigate('/masuk');
                } else {

navigate('/beranda');
                }
              }}
              className="inline-
flex items-center text-sm text-
gray-600 hover:text-gray-900
transition-colors"
            >
              Kembali
            </button>
          </div>
          <h2 className="text-
center text-3xl font-bold text-
gray-900 mb-4">Daftar Akun</h2>
          <form className="mt-8
space-y-6"
onSubmit={handleSubmit}>
            {errors.general && (
              <div className="bg-
red-50 border border-red-200
text-red-600 px-4 py-3 rounded-md
text-sm">
                {errors.general}
              </div>
            )}
            <div
className="space-y-4">
              <div>
                <label
htmlFor="nama" className="block
text-sm font-medium text-gray-
700">
                  Nama <span
className="text-red-500">*</span>
                </label>
                <input
                  id="nama"
                  name="nama"
                  type="text"

value={formData.nama}

onChange={handleChange}
                  className={`mt-
1 appearance-none relative block
w-full px-3 py-2 border ${
                    errors.nama ?
'border-red-300' : 'border-gray-
300'
                  } placeholder-
gray-500 text-gray-900 rounded-lg
focus:outline-none focus:ring-
blue-500 focus:border-blue-500
focus:z-10 sm:text-sm`}

placeholder="Masukkan nama"
                />
                {errors.nama && (
                  <p
className="mt-1 text-xs text-red-
600">{errors.nama}</p>
                )}
              </div>
              <div>
                <label
htmlFor="email" className="block
text-sm font-medium text-gray-
700">
                  Email <span
className="text-red-500">*</span>
                </label>
                <input
                  id="email"
                  name="email"
                  type="email"

autoComplete="email"

value={formData.email}

onChange={handleChange}
                  className={`mt-
1 appearance-none relative block
w-full px-3 py-2 border ${
                    errors.email
? 'border-red-300' : 'border-
gray-300'
                  } placeholder-
gray-500 text-gray-900 rounded-lg
focus:outline-none focus:ring-
blue-500 focus:border-blue-500
focus:z-10 sm:text-sm`}

placeholder="nama@email.com"
                />
                {errors.email &&
(
```

```
                  <p
className="mt-1 text-xs text-red-
600">{errors.email}</p>
                )}
              </div>
              <div>
                <label
htmlFor="nomor_telepon"
className="block text-sm font-
medium text-gray-700">
                  Nomor Telepon
{formData.role === 'PENJUAL' ? (
                    <span
className="text-red-500">*</span>
                  ) : (
                    <span
className="text-gray-400 text-
xs">(opsional)</span>
                  )}
                </label>
                <input

id="nomor_telepon"

name="nomor_telepon"
                  type="tel"

autoComplete="tel"

value={formData.nomor_telepon}

onChange={handleChange}
                  className={`mt-
1 appearance-none relative block
w-full px-3 py-2 border ${

errors.nomor_telepon ? 'border-
red-300' : 'border-gray-300'
                  } placeholder-
gray-500 text-gray-900 rounded-lg
focus:outline-none focus:ring-
blue-500 focus:border-blue-500
focus:z-10 sm:text-sm`}

placeholder={formData.role ===
'PENJUAL' ? "08xxxxxxxxxx (wajib
untuk penjual)" : "08xxxxxxxxxx
(bisa diisi nanti)"}
                />

{errors.nomor_telepon && (
                  <p
className="mt-1 text-xs text-red-
600">{errors.nomor_telepon}</p>
                )}
                <p className="mt-
1 text-xs text-gray-500">
                  {formData.role
=== 'PENJUAL'
                    ? 'Nomor
telepon diperlukan untuk
verifikasi penjual'
                    : 'Nomor
telepon dapat diisi sekarang atau
nanti di halaman profil'
                  }
                </p>
              </div>
              <div>
                <label
htmlFor="role" className="block
text-sm font-medium text-gray-
700">
                  Role <span
className="text-red-500">*</span>
                </label>
                <select
                  id="role"
                  name="role"

value={formData.role}

onChange={handleChange}
                  className={`mt-
1 appearance-none relative block
w-full px-3 py-2 border ${
                    errors.role ?
'border-red-300' : 'border-gray-
300'
                  } placeholder-
gray-500 text-gray-900 rounded-lg
focus:outline-none focus:ring-
blue-500 focus:border-blue-500
focus:z-10 sm:text-sm`}
                >
                  <option
value="PEMBELI">Pembeli - Membeli
akun *game*</option>
                  <option
value="PENJUAL">Penjual - Menjual
akun *game*</option>
                </select>
                {errors.role && (
                  <p
className="mt-1 text-xs text-red-
600">{errors.role}</p>
                )}
                <p className="mt-
1 text-xs text-gray-500">
                  Role dapat
diubah nanti di pengaturan profil
                </p>
              </div>
              <div>
                <label
htmlFor="password"
className="block text-sm font-
medium text-gray-700">
                  Password <span
className="text-red-500">*</span>
                </label>
                <div
className="relative">
                  <input
                    id="password"
```

```
name="password"

type={showPassword ? 'text' :
'password'}

autoComplete="new-password"

value={formData.password}

onChange={handleChange}

className={`mt-1 appearance-none
relative block w-full px-3 py-2
border ${

errors.password ? 'border-red-
300' : 'border-gray-300'
                    }
placeholder-gray-500 text-gray-
900 rounded-lg focus:outline-none
focus:ring-blue-500 focus:border-
blue-500 focus:z-10 sm:text-sm
pr-10`}

placeholder="Minimal 8 karakter"
                  />
                  <button
                    type="button"
                    onClick={()
=>
setShowPassword(!showPassword)}

className="absolute inset-y-0
right-0 pr-3 flex items-center"
                  >
                    {showPassword
? 'Hide' : 'Show'}
                  </button>
                </div>
                {errors.password
&& (
                  <p
className="mt-1 text-xs text-red-
600">{errors.password}</p>
                )}
              </div>
              <div>
                <label
htmlFor="confirmPassword"
className="block text-sm font-
medium text-gray-700">
                  Konfirmasi
Password <span className="text-
red-500">*</span>
                </label>
                <div
className="relative">
                  <input

id="confirmPassword"

name="confirmPassword"

type={showConfirmPassword ?
'text' : 'password'}

autoComplete="new-password"

value={formData.confirmPassword}

onChange={handleChange}

className={`mt-1 appearance-none
relative block w-full px-3 py-2
border ${

errors.confirmPassword ? 'border-
red-300' : 'border-gray-300'
                    }
placeholder-gray-500 text-gray-
900 rounded-lg focus:outline-none
focus:ring-blue-500 focus:border-
blue-500 focus:z-10 sm:text-sm
pr-10`}

placeholder="Masukkan ulang
password"
                  />
                  <button
                    type="button"
                    onClick={()
=>
setShowConfirmPassword(!showConfi
rmPassword)}

className="absolute inset-y-0
right-0 pr-3 flex items-center"
                  >

{showConfirmPassword ? 'Hide' :
'Show'}
                  </button>
                </div>

{errors.confirmPassword && (
                  <p
className="mt-1 text-xs text-red-
600">{errors.confirmPassword}</p>
                )}
              </div>
              <div>
                <button
                  type="submit"

disabled={loading}
                  className="w-
full py-3 px-4 bg-blue-600 text-
white rounded-lg hover:bg-blue-
700 disabled:bg-gray-300
transition duration-300"
                >
                  {loading ?
'Memproses...' : 'Daftar'}
                </button>
              </div>
```

```
            <div
className="text-center text-sm
text-gray-600 mt-4">
                Sudah punya
akun?{' '}
                <Link
                  to="/masuk"
                  state={{ from:
'/daftar' }}

className="text-primary-600
hover:text-primary-700
transition-colors font-medium"
                >
                  Masuk
                </Link>
              </div>
            </div>
          </form>
        </div>
      </main>
    </div>
  );
};
export default Daftar;
```

Masuk.jsx

```
import React, { useState,
useEffect } from 'react';
import { Link, useNavigate,
useLocation } from 'react-router-
dom';
import { useAuth } from
'../../konteks/AuthContext';
const Masuk = () => {
  const navigate = useNavigate();
  const location = useLocation();
  const { login, loading,
isAuthenticated, user } =
useAuth();
  const [formData, setFormData] =
useState({
    email: '',
    password: ''
  });
  const [showPassword,
setShowPassword] =
useState(false);
  const [errors, setErrors] =
useState({});

  useEffect(() => {
    if (isAuthenticated && user)
{
      if (user.role === 'ADMIN')
{navigate('/admin', { replace:
true });
      } else if (user.role ===
'PENJUAL') {
        navigate('/produk-saya',
{ replace: true });
      } else {
        navigate('/beranda', {
replace: true });
      }
    }
  }, [isAuthenticated, user,
navigate]);

  const handleChange = (e) => {
    const { name, value } =
e.target;
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));
    if (errors[name]) {
      setErrors(prev => ({
        ...prev,
        [name]: ''
      }));
    }
  };

  const validateForm = () => {
    const newErrors = {};
    if (!formData.email) {
      newErrors.email = 'Email
harus diisi';
    } else if
(!/\S+@\S+\.\S+/.test(formData.em
ail)) {
      newErrors.email = 'Format
email tidak valid';
    }
    if (!formData.password) {
      newErrors.password =
'Password harus diisi';
    } else if
(formData.password.length < 6) {
      newErrors.password =
'Password minimal 6 karakter';
    }
    setErrors(newErrors);
    return
Object.keys(newErrors).length ===
0;
  };

  const handleSubmit = async (e)
=> {
    e.preventDefault();
    if (!validateForm()) {
      return;
    }
    const result = await
login(formData);
    if (result.success) {
      if (result.user?.role ===
'ADMIN') {
        navigate('/admin', {
replace: true });
      } else if
(result.user?.role === 'PENJUAL')
{
```

```
        navigate('/produk-saya',
{ replace: true });
      } else {
        navigate('/beranda', {
replace: true });
      }
    }
  };

  return (
    <div className="min-h-screen
bg-gray-50 flex items-center
justify-center py-12 px-4 sm:px-6
lg:px-8">
      <div className="max-w-md w-
full space-y-8 bg-white p-8
rounded-lg shadow-md">
        <div className="flex
justify-start">
          <button
            onClick={() => {
              const from =
location.state?.from;
              if (from ===
'/daftar') {

navigate('/daftar');
              } else {

navigate('/beranda');
              }
            }}
            className="inline-
flex items-center text-sm text-
gray-600 hover:text-gray-900
transition-colors"
          >
            Kembali
          </button>
        </div>
        <div className="text-
center">
          <h2 className="mt-2
text-center text-3xl font-
extrabold text-gray-900">
            Masuk ke akun Anda
          </h2>
        </div>
        <form className="mt-8
space-y-6"
onSubmit={handleSubmit}>
          <div className="space-
y-4">
            <div>
              <label
htmlFor="email" className="block
text-sm font-medium text-gray-
700">
                Email
              </label>
              <input
                id="email"
                name="email"
                type="email"

autoComplete="email"

value={formData.email}

onChange={handleChange}
                className={`mt-1
appearance-none relative block w-
full px-3 py-2 border ${
                  errors.email ?
'border-red-300' : 'border-gray-
300'
                } placeholder-
gray-500 text-gray-900 rounded-md
focus:outline-none focus:ring-
primary-500 focus:border-primary-
500 focus:z-10 sm:text-sm`}

placeholder="Masukkan email Anda"
              />
              {errors.email && (
                <p className="mt-
1 text-sm text-red-
600">{errors.email}</p>
              )}
            </div>
            <div>
              <label
htmlFor="password"
className="block text-sm font-
medium text-gray-700">
                Password
              </label>
              <div className="mt-
1 relative">
                <input
                  id="password"
                  name="password"

type={showPassword ? 'text' :
'password'}

autoComplete="current-password"

value={formData.password}

onChange={handleChange}

className={`appearance-none
relative block w-full px-3 py-2
pr-10 border ${

errors.password ? 'border-red-
300' : 'border-gray-300'
                } placeholder-
gray-500 text-gray-900 rounded-md
focus:outline-none focus:ring-
primary-500 focus:border-primary-
500 focus:z-10 sm:text-sm`}

placeholder="Masukkan password
Anda"
```

```
                />
                <button
                  type="button"
                  
className="absolute inset-y-0
right-0 pr-3 flex items-center"
                  onClick={() =>
setShowPassword(!showPassword)}
                >
                  {showPassword ?
'Hide' : 'Show'}
                </button>
              </div>
              {errors.password &&
(
                <p className="mt-
1 text-sm text-red-
600">{errors.password}</p>
              )}
            </div>
          </div>
          <div className="flex
items-center justify-between">
            <div className="text-
sm">
              <a href="#"
className="font-medium text-
primary-600 hover:text-primary-
500">
                Lupa password?
              </a>
            </div>
          </div>
          <div>
            <button
              type="submit"
              disabled={loading}
              className={`group
relative w-full flex justify-
center py-2 px-4 border border-
transparent text-sm font-medium
rounded-md text-white ${
                loading
                  ? 'bg-gray-400
cursor-not-allowed'
                  : 'bg-primary-
600 hover:bg-primary-700
focus:outline-none focus:ring-2
focus:ring-offset-2 focus:ring-
primary-500'
              } transition-
colors`}
            >
              {loading ? (
                <div
className="flex items-center">
                  <div
className="animate-spin rounded-
full h-4 w-4 border-b-2 border-
white mr-2"></div>
                  Memproses...
                </div>
              ) : (
                'Masuk'
              )}
            </button>
          </div>
          <div className="text-
center mt-4">
            <p className="text-sm
text-gray-600">
              Belum punya akun?{'
'}
              <Link
                to="/daftar"
                state={{ from:
'/masuk' }}
                className="font-
medium text-primary-600
hover:text-primary-500
transition-colors"
              >
                Daftar sekarang
              </Link>
            </p>
          </div>
        </form>
      </div>
    </div>
  );
};
export default Masuk;
```

Daftar.jsx

```
import React, { useState,
useEffect, useCallback } from
'react';
import { useNavigate } from
'react-router-dom';
import { useAuth } from
'../../konteks/AuthContext';
import { useWallet } from
'../../konteks/WalletContext';
import { transaksiAPI, apiService
} from '../../layanan/api';
import { useDashboardUpdates }
from
'../../hooks/useRealTimeUpdates';
import { accountDataHelper } from
'../../utils/accountDataHelper';
import ModalLihatAkun from
'../../komponen/ModalLihatAkun';
import ModalTerimaAkun from
'../../komponen/ModalTerimaAkun';
import ModalSengketa from
'../../komponen/ModalSengketa';
import ModalDetailTransaksi from
'../../komponen/ModalDetailTransa
ksi';
import PembelianKontrakPintar
from
'../../komponen/PembelianKontrakP
intar';
import transactionExpiryManager
from
'../../utils/transactionExpiry';
```

```
import toast from 'react-hot-
toast';

const SimpleTimer = ({ transaksi,
onExpired }) => {
  const [timeLeft, setTimeLeft] =
useState(null);
  const [isExpired, setIsExpired]
= useState(false);

  useEffect(() => {
    if (!transaksi ||
transaksi.status !==
'MENUNGGU_PEMBAYARAN') return;

    const createdAt = new
Date(transaksi.dibuatPada);
    const expiryTime = new
Date(createdAt.getTime() + 15 *
60 * 1000);

    const updateTimer = () => {
      const now = new Date();
      const difference =
expiryTime.getTime() -
now.getTime();

      if (difference <= 0) {
        setTimeLeft({ minutes: 0,
seconds: 0 });
        setIsExpired(true);
        if (onExpired)
onExpired(transaksi);
        return;
      }
      const minutes =
Math.floor(difference / (1000 *
60));
      const seconds =
Math.floor((difference % (1000 *
60)) / 1000);
      setTimeLeft({ minutes,
seconds });
      setIsExpired(false);
    };

    updateTimer();
    const interval =
setInterval(updateTimer, 1000);
    return () =>
clearInterval(interval);
  }, [transaksi, onExpired]);

  if (!transaksi ||
transaksi.status !==
'MENUNGGU_PEMBAYARAN') return
null;

  if (isExpired) return <p
className="text-xs text-red-600
font-medium">Waktu habis</p>;

  if (timeLeft) {
    return (
      <p className={`text-xs
font-medium ${
        timeLeft.minutes < 5 ?
'text-red-600' :
        timeLeft.minutes < 10 ?
'text-orange-600' : 'text-blue-
600'
      }`}>

{String(timeLeft.minutes).padStar
t(2,
'0')}:{String(timeLeft.seconds).p
adStart(2, '0')}
      </p>
    );
  }
  return <p className="text-xs
text-blue-600 font-
medium">Memuat...</p>;
};

const DashboardPembeli = () => {
  const [transaksi, setTransaksi]
= useState([]);
  const [loading, setLoading] =
useState(true);
  const [error, setError] =
useState(null);
  const [filter, setFilter] =
useState('all');
  const [lastRefresh,
setLastRefresh] = useState(new
Date());
  const [refreshing,
setRefreshing] = useState(false);
  const [showLihatAkunModal,
setShowLihatAkunModal] =
useState(false);
  const [showTerimaAkunModal,
setShowTerimaAkunModal] =
useState(false);
  const [showSengketaModal,
setShowSengketaModal] =
useState(false);
  const
[showDetailTransaksiModal,
setShowDetailTransaksiModal] =
useState(false);
  const [showPaymentModal,
setShowPaymentModal] =
useState(false);
  const [selectedTransaksi,
setSelectedTransaksi] =
useState(null);
  const [actionLoading,
setActionLoading] =
useState(false);

  const { user, isAuthenticated,
loading: authLoading } =
useAuth();
```

```
  const { isConnected,
walletAddress } = useWallet();
  const navigate = useNavigate();

  const { triggerUpdate } =
useDashboardUpdates(() => {
    fetchTransaksi(false);
    toast.success('Data transaksi
diperbarui', { duration: 2000 });
  });
  useEffect(() => {
    const
handleExpiredTransaction =
(event, expiredTransaksi) => {
      fetchTransaksi(false);
    };

    const
handleTransactionCreated = () =>
{
      fetchTransaksi(false);
      toast.success('Transaksi
berhasil dibuat! Timer 15 menit
dimulai.');
    };
transactionExpiryManager.addListe
ner(handleExpiredTransaction);

window.addEventListener('transact
ion-created',
handleTransactionCreated);
    return () => {

transactionExpiryManager.removeLi
stener(handleExpiredTransaction);

transactionExpiryManager.stopAll(
);

window.removeEventListener('trans
action-created',
handleTransactionCreated);
    };
  }, []);
  useEffect(() => {
    if (authLoading) return;
    if (!isAuthenticated ||
!user) {
      navigate('/masuk');
      return;
    }
    fetchTransaksi();
  }, [isAuthenticated, user,
authLoading, filter, navigate]);

  const fetchTransaksi =
useCallback(async (showLoading =
true) => {
    try {
      showLoading ?
setLoading(true) :
setRefreshing(true);
      setError(null);
      if (!user?.id) throw new
Error('User ID tidak ditemukan');

      let response;
      try {
        response = await
transaksiAPI.getTransaksiSaya({
          role: 'pembeli',
          status: filter ===
'all' ? undefined : filter,
          page: 1,
          limit: 50,
          timestamp: Date.now()
        });
      } catch (primaryError) {
        response = await
transaksiAPI.getTransaksiUser({
          role: 'pembeli',
          status: filter ===
'all' ? undefined : filter,
          page: 1,
          limit: 50,
          timestamp: Date.now()
        });
      }
      let transaksiData = [];
      if (response.data?.sukses)
{
        transaksiData =
response.data.data?.transaksi ||

(Array.isArray(response.data.data
) ? response.data.data : []);
      } else if
(Array.isArray(response.data)) {
        transaksiData =
response.data;
      }
      let filteredTransaksi =
transaksiData;
      if (filter === 'all') {
        filteredTransaksi =
transaksiData.filter(t =>
t.status !== 'GAGAL');
      } else {
        filteredTransaksi =
transaksiData.filter(t =>
t.status !== 'GAGAL');
      }

setTransaksi(filteredTransaksi);
      setLastRefresh(new Date());

      const pendingTransactions =
transaksiData.filter(t =>
t.status ===
'MENUNGGU_PEMBAYARAN');
if (pendingTransactions.length >
0) {
transactionExpiryManager.startMon
itoringMultiple(pendingTransactio
ns); }
    } catch (error) {
```

```
      let errorMessage = 'Gagal
memuat transaksi';
      if (error.response?.status
=== 401) {

localStorage.removeItem('authToke
n');

localStorage.removeItem('user');
        navigate('/masuk');
        return;
      } else if
(error.response?.status === 500)
{
        errorMessage = 'Server
bermasalah. Silakan coba lagi
nanti.';
      } else if
(error.response?.status === 404)
{
        errorMessage = 'Endpoint
tidak ditemukan. Hubungi admin.';
      } else if (error.code ===
'ERR_NETWORK') {
        errorMessage = 'Koneksi
ke server bermasalah.';
      } else if
(error.response?.data?.pesan) {
        errorMessage =
error.response.data.pesan;
        if
(errorMessage.toLowerCase().inclu
des('token tidak valid')) {

localStorage.removeItem('authToke
n');

localStorage.removeItem('user');
          navigate('/masuk');
          return;
        }
      } else if (error.message) {
        errorMessage =
error.message;
      }

      setError(errorMessage);
      if (transaksi.length === 0)
setTransaksi([]);
    } finally {
      showLoading ?
setLoading(false) :
setRefreshing(false);
    }
  }, [filter, user,
transaksi.length, navigate]);

const handleRefresh = async () =>
{
    await fetchTransaksi(false);
    toast.success('Data transaksi
diperbarui');
  };

  const handlePaymentExpired =
async (expiredTransaksi) => {
    try {

transactionExpiryManager.stopMoni
toring(expiredTransaksi.id);
      await
fetchTransaksi(false);
      toast.error(`Waktu
pembayaran untuk
${expiredTransaksi.produk?.judulP
roduk} telah habis. Produk
dikembalikan ke market.`);
      window.dispatchEvent(new
CustomEvent('transaction-
expired', {
        detail: {
          transaksiId:
expiredTransaksi.id,
          produkId:
expiredTransaksi.produkId,
          timestamp: Date.now()
        }
      }));
    } catch (error) {
      console.error('Error
handling expired payment:',
error);
    }
  };

  const handlePayment =
(transaksi) => {
    if (!isConnected) {
      toast.error('Silakan
connect wallet di navbar terlebih
dahulu');
      return;
    }

    if
(walletAddress?.toLowerCase() !==
user.walletAddress?.toLowerCase()
) {
      toast.error('Wallet yang
terhubung tidak sesuai dengan
akun Anda');
      return;
    }

setSelectedTransaksi(transaksi);
    setShowPaymentModal(true);
  };

  const handlePaymentSuccess =
(result) => {
    toast.success('Pembayaran
berhasil! Transaksi telah dibuat
di blockchain.');
    setShowPaymentModal(false);
    setSelectedTransaksi(null);
```

```
    if (selectedTransaksi)
transactionExpiryManager.stopMoni
toring(selectedTransaksi.id);
    fetchTransaksi(false);
  };
  const handlePaymentCancel = ()
=> {
    setShowPaymentModal(false);
    setSelectedTransaksi(null);
  };
  const formatDate = (dateString)
=> {
    return new
Date(dateString).toLocaleDateStri
ng('id-ID', {
      year: 'numeric',
      month: 'short',
      day: 'numeric',
      hour: '2-digit',
      minute: '2-digit'
    });
  };
  const getStatusBadge = (status)
=> {
    const statusConfig = {
      'MENUNGGU_PEMBAYARAN': {
        color: 'bg-yellow-100
text-yellow-800',
        text: 'Menunggu
Pembayaran'
    const config =
statusConfig[status] || {
      color: 'bg-gray-100 text-
gray-800',
      text: status
    };
    return (
      <span className={`inline-
flex items-center px-2.5 py-0.5
rounded-full text-xs font-medium
${config.color}`}>
        {config.text}
      </span>
    );
  };
  const handleDetailClick =
(transaksi) => {

setSelectedTransaksi(transaksi);

setShowDetailTransaksiModal(true)
;
  };
  const handleLihatAkun =
(transaksi) => {
setSelectedTransaksi(transaksi);
    setShowLihatAkunModal(true);
  };
  const handleTerimaAkun =
(transaksi) => {

setSelectedTransaksi(transaksi);
    setShowTerimaAkunModal(true);
  };
  const
handleKonfirmasiPenerimaan =
async () => {
    if (!selectedTransaksi)
return;
    try {
      setActionLoading(true);
      await
transaksiAPI.konfirmasiPenerimaan
(selectedTransaksi.id);
      await
fetchTransaksi(false);
setShowTerimaAkunModal(false);
setSelectedTransaksi(null);toast.
success('Penerimaan akun berhasil
dikonfirmasi!');
    } catch (error) {
      toast.error('Gagal
konfirmasi penerimaan: ' +
error.message);
    } finally {
      setActionLoading(false);
    }
  };
  const handleSengketa = async
(sengketaData) => {
    if (!selectedTransaksi)
return;
    try {
      setActionLoading(true);
      const formattedData = {
        alasan:
sengketaData.deskripsi ||
sengketaData.alasan,
        bukti: sengketaData.bukti
|| null
      };

      if (!formattedData.alasan
|| formattedData.alasan.length <
20) {
        toast.error('Alasan
sengketa minimal 20 karakter');
        return;
      }
      await
transaksiAPI.buatSengketa(selecte
dTransaksi.id, formattedData);
      await
fetchTransaksi(false);

setShowSengketaModal(false);
      setSelectedTransaksi(null);
      toast.success('Sengketa
berhasil dilaporkan!');
    } catch (error) {const
errorMessage =
error.response?.data?.pesan ||
error.message;
      toast.error('Gagal membuat
sengketa: ' + errorMessage);
    } finally {
```

```
      setActionLoading(false);
    }
  };
  const handleBuatSengketa =
(transaksi) => {

setSelectedTransaksi(transaksi);
    setShowSengketaModal(true);
  };
  const canCreateDispute =
(transaksi) => {
    return transaksi.status ===
'DIKIRIM' &&
           transaksi.status !==
'SENGKETA' &&
           !transaksi.sengketa;
  };
  const getDisputeStatusInfo =
(transaksi) => {
    if (transaksi.status ===
'SENGKETA') {
      return { message: 'Sengketa
sedang diproses admin' };
    }
    if (transaksi.status ===
'DIMENANGKAN_PEMBELI') {
      return { message: 'Sengketa
dimenangkan - Dana dikembalikan'
};
    }
    if (transaksi.status ===
'DIMENANGKAN_PENJUAL') {
      return { message: 'Sengketa
dimenangkan penjual' };
    }
    return { message: null };
  };

  if (authLoading) {
    return (
      <div className="min-h-
screen bg-gray-50 flex items-
center justify-center">
        <div className="text-
center">
          <div
className="animate-spin rounded-
full h-32 w-32 border-b-2 border-
blue-600 mx-auto"></div>
          <p className="mt-4
text-gray-600">Memuat
dashboard...</p>
        </div>
      </div>
    );
  }
  if (!isAuthenticated || !user)
{
    navigate('/masuk');
    return null;}
  if (loading) {
    return (
      <div className="min-h-
screen bg-gray-50 flex items-
center justify-center">
        <div className="text-
center">
          <div
className="animate-spin rounded-
full h-32 w-32 border-b-2 border-
blue-600 mx-auto"></div>
          <p className="mt-4
text-gray-600">Memuat
transaksi...</p>
        </div>
      </div>
    );
  }
  return (
    <div className="min-h-screen
bg-gray-50 py-4 sm:py-8">
      <div className="max-w-[90%]
mx-auto px-2 sm:px-4 lg:px-8">
        <div className="bg-white
rounded-lg shadow-sm mb-4 sm:mb-
6">
          <div className="px-4
sm:px-6 py-4 border-b border-
gray-200">
            <div className="flex
flex-col sm:flex-row sm:justify-
between sm:items-center space-y-3
sm:space-y-0">
              <div>
                <h1
className="text-xl sm:text-2xl
font-bold text-gray-
900">Pesanan</h1>
                <p
className="text-sm text-gray-600
mt-1">
                  Kelola pesanan
pembelian Anda dengan timer
pembayaran 15 menit
                </p>
              </div>
              <div
className="flex flex-col sm:text-
right">
                <button

onClick={handleRefresh}

disabled={loading || refreshing}

className="inline-flex items-
center justify-center px-3 py-2
border border-gray-300 shadow-sm
text-sm leading-4 font-medium
rounded-md text-gray-700 bg-white
hover:bg-gray-50
disabled:opacity-50">{refreshing
? 'Memperbarui...' : 'Perbarui'}
</button><p className="text-xs
text-gray-500 mt-1 text-center
```

```
sm:text-right">Terakhir
diperbarui:
{lastRefresh.toLocaleTimeString('
id-ID')}
                </p>
              </div>
            </div>
          </div>
        </div>
<div className="ml-3"><h3
className="text-sm font-medium
text-red-800">Gagal memuat
transaksi</h3>
<div className="mt-2 text-sm
text-red-
700"><p>{error}</p></div>
<div className="mt-4 space-x-2">
<button
onClick={() =>
fetchTransaksi(true)}
className="bg-red-100 px-3 py-2
rounded-md text-sm font-medium
text-red-800 hover:bg-red-200"
>Coba Lagi </button>
<buttononClick={() => {
setError(null);
setTransaksi([]);
fetchTransaksi(true);}}
className="bg-blue-100 px-3 py-2
rounded-md text-sm font-medium
text-blue-800 hover:bg-blue-200"
>Reset & Muat
Ulang</button>{error &&
error.toLowerCase().includes('tok
en') && (<buttononClick={() =>
navigate('/masuk')}
className="bg-green-100 px-3 py-2
rounded-md text-sm font-medium
text-green-800 hover:bg-green-
200">
Login Ulang</button>)}
                    </div>
                  </div>
                </div>
              </div>
            )}
            <div className="mb-
4"><h2 className="text-lg font-
semibold text-gray-900">Riwayat
Pesanan</h2></div>transaksi.lengt
h === 0 ? (<div className="px-6
py-12 text-center"><path
strokeLinecap="round"
strokeLinejoin="round"
strokeWidth="2" d="M16 11V7a4 4 0
00-8 0v4M5 9h14l1 12H4L5 9z"
/></svg><h3 className="mt-2 text-
sm font-medium text-gray-900">
                  {error ? 'Tidak
dapat memuat transaksi' : 'Belum
ada transaksi'}</h3><p
className="mt-1 text-sm text-
gray-500">{error? 'Terjadi
masalah saat memuat data. Silakan
coba lagi.': 'Mulai berbelanja
untuk melihat transaksi di sini.'
}</p><div className="mt-6">{error
? (<div className="space-x-2">
<buttononClick={() =>
fetchTransaksi(true)}
className="inline-flex items-
center px-4 py-2 border border-
transparent shadow-sm text-sm
font-medium rounded-md text-white
bg-red-600 hover:bg-red-700">Coba
Lagi</button><buttononClick={()
=> navigate('/masuk')}
className="inline-flex items-
center px-4 py-2 border border-
gray-300 shadow-sm text-sm font-
medium rounded-md text-gray-700
bg-white hover:bg-gray-50" >Login
Ulang</button></div>) : (<button
onClick={() =>
navigate('/produk')}
className="inline-flex items-
center px-4 py-2 border border-
transparent shadow-sm text-sm
font-medium rounded-md text-white
bg-blue-600 hover:bg-blue-
700">Mulai
Belanja</button>)}</div></div>) :
(<div className="space-y-
3">{transaksi.map((item) => (<div
key={item.id} className="bg-gray-
50 rounded-lg p-3 hover:bg-gray-
100 transition-colors"><div
className="flex flex-col space-y-
2"><div className="flex items-
start space-x-3"><img
src={item.produk?.gambar ||
'/placeholder-game.jpg'}
alt={item.produk?.judulProduk ||
'Produk'}className="w-12 h-12
sm:w-16 sm:h-16 rounded-lg
object-cover flex-shrink-0"/>
<div className="flex-1 min-w-0">
<h3 className="text-sm font-
medium text-gray-900 line-clamp-
2">{item.produk?.judulProduk ||
'Produk Tidak Ditemukan'}</h3>
<p className="text-xs text-gray-
500">{item.produk?.namaGame ||
'Game'}</p><p className="text-xs
text-gray-500">
                          Kode:
{item.kodeTransaksi || item.id}
                        </p>
                        <div
className="flex items-center
space-x-2 mt-1">
<p className="text-sm font-
semibold text-blue-600">
{item.escrowAmount ?
`${parseFloat(item.escrowAmount).
toFixed(4)} ETH` :
```

```
item.produk?.hargaEth ?
`${item.produk.hargaEth} ETH` :
'N/A'}</p></div></div></div>
<div className="flex flex-col
sm:flex-row sm:items-center
sm:justify-between space-y-1
sm:space-y-0"><div
className="flex items-center
space-x-2">
   {getStatusBadge(item.status)}
                        </div>
                        <div
className="text-left sm:text-
right"><p className="text-xs
text-gray-
500">formatDate(item.dibuatPada)}
</p>{item.escrowId && (<p
className="text-xs text-blue-
600">Escrow ID: {item.escrowId}
</p>)}{item.status ===
'MENUNGGU_PEMBAYARAN' &&
(<SimpleTimertransaksi={item}onEx
pired={handlePaymentExpired}/>)}<
/div></div>
<div className="flex flex-wrap
gap-2"><buttononClick={() =>
handleDetailClick(item)}
className="flex-1 sm:flex-none
inline-flex items-center justify-
center px-3 py-2 border border-
gray-300 text-xs font-medium
rounded-md text-gray-700 bg-white
hover:bg-gray-50">Detail</button>
{(() => {const hasData =
accountDataHelper.hasAccountData(
item); const validStatus =
['DIKIRIM',
'DIKONFIRMASI_PEMBELI',
'SELESAI'].includes(item.status);
return hasData && validStatus &&
(<buttononClick={() =>
handleLihatAkun(item)}
className="flex-1 sm:flex-none
inline-flex items-center justify-
center px-3 py-2 border border-
green-300 text-xs font-medium
rounded-md text-green-700 bg-
green-50 hover:bg-green-
100">Lihat Akun </button>);
  })()}
  {item.status === 'DIKIRIM' && (
<button
               onClick={() =>
handleTerimaAkun(item)}
className="flex-1 sm:flex-none
inline-flex items-center justify-
center px-3 py-2 border border-
transparent text-xs font-medium
rounded-md text-white bg-green-
600 hover:bg-green-
700"onClick={() =>
handleBuatSengketa(item)}
className="flex-1 sm:flex-none
inline-flex items-center justify-
center px-3 py-2 border border-
red-300 text-xs font-medium
rounded-md text-red-700 bg-red-50
hover:bg-red-100"Sengketa
</button>){item.status ===
'DIKONFIRMASI_PEMBELI' && (<div
className="flex-1 sm:flex-none
inline-flex items-center justify-
center px-3 py-2 border border-
blue-300 text-xs font-medium
rounded-md text-blue-700 bg-blue-
50">Dana Ditahan</div>
{getDisputeStatusInfo(item).messa
ge && (<div className="flex-1
sm:flex-none inline-flex items-
center justify-center px-3 py-2
border border-orange-300 text-xs
font-medium rounded-md text-
orange-700 bg-orange-50"
title={getDisputeStatusInfo(item)
.message}>
Sengketa
Aktif</div>)}</div></div></div>
))} </div>)}
          </div>
        </div>
        <ModalLihatAkun
isOpen={showLihatAkunModal}
          onClose={() => {
setShowLihatAkunModal(false);
setSelectedTransaksi(null);
          }}
transaksi={selectedTransaksi}
        />
        <ModalTerimaAkun
isOpen={showTerimaAkunModal}
          onClose={() => {

setShowTerimaAkunModal(false);

setSelectedTransaksi(null);
          }}
onConfirm={handleKonfirmasiPeneri
maan}
onDispute={handleSengketa}
          loading={actionLoading}
transaksi={selectedTransaksi}
        />
        <ModalSengketa
isOpen={showSengketaModal}
          onClose={() => {
setShowSengketaModal(false);
setSelectedTransaksi(null);
          }}
onSubmit={handleSengketa}
          loading={actionLoading}

transaksi={selectedTransaksi}/>
        <ModalDetailTransaksi
isOpen={showDetailTransaksiModal}
          onClose={() => {
```

```
setShowDetailTransaksiModal(false
);
setSelectedTransaksi(null);
          }}
transaksi={selectedTransaksi}/>
        {showPaymentModal &&
selectedTransaksi && (
          <div className="fixed
inset-0 bg-black bg-opacity-50
flex items-center justify-center
z-50 p-4">
            <div className="bg-
white rounded-lg max-w-md w-full
max-h-[90vh] overflow-y-auto">
              <div className="p-
6"><div className="flex justify-
between items-center mb-6"><h3
className="text-xl font-semibold
text-gray-900">Pembayaran
Transaksi</h3><buttononClick={han
dlePaymentCancel}className="text-
gray-400 hover:text-gray-600"
><svg className="w-6 h-6"
fill="none" stroke="currentColor"
viewBox="0 0 24 24"><path
strokeLinecap="round"
strokeLinejoin="round"
strokeWidth="2" d="M6 18L18 6M6
6l12 12"
/></svg></button></div><Pembelian
KontrakPintar
produk={selectedTransaksi.produk}
oSuccess={handlePaymentSuccess}
onCancel={handlePaymentCancel}
                />
              </div>
            </div>
          </div>
        )}
      </div>
    </div>
  );
};
export default DashboardPembeli;
```

DashboardPenjual.jsx

```
import React, { useState,
useEffect, useCallback } from
'react';
import { useNavigate } from
'react-router-dom';
import { useAuth } from
'../../konteks/AuthContext';
import { transaksiAPI, produkAPI
} from '../../layanan/api';
import { useDashboardUpdates }
from
const DashboardPenjual = () => {
  const [transaksi, setTransaksi]
= useState([]);
  const [produk, setProduk] =
useState([]);
  const [loading, setLoading] =
useState(true);
  const [error, setError] =
useState(null);
  const [filter, setFilter] =
useState('all');
  const [lastRefresh,
setLastRefresh] = useState(new
Date());
  const [refreshing,
setRefreshing] = useState(false);
  const [showLihatAkunModal,
setShowLihatAkunModal] =
useState(false);
  const [showTerimaAkunModal,
setShowTerimaAkunModal] =
useState(false);
  const [showKirimAkunModal,
setShowKirimAkunModal] =
useState(false);
  const [showDetailSengketaModal,
setShowDetailSengketaModal] =
useState(false);
  const [showSengketaModal,
setShowSengketaModal] =
useState(false);
  const
[showDetailTransaksiModal,
setShowDetailTransaksiModal] =
useState(false);
  const [selectedTransaksi,
setSelectedTransaksi] =
useState(null);
  const [actionLoading,
setActionLoading] =
useState(false);
  const { user, isAuthenticated,
loading: authLoading } =
useAuth();
  const navigate = useNavigate();

  const { triggerUpdate } =
useDashboardUpdates(() => {
    fetchTransaksi(false);
    fetchProduk(false);
    toast.success('Data transaksi
dan produk diperbarui', {
duration: 2000 });
  });
  useEffect(() => {
    if (authLoading) return;
    if (!isAuthenticated ||
!user) {
      navigate('/masuk');
      return;
    }
    fetchTransaksi();
    fetchProduk();
  }, [isAuthenticated, user,
authLoading, filter]);
```

```
  const fetchTransaksi =
useCallback(async (showLoading =
true) => {
    try {
      if (showLoading) {
        setLoading(true);
      } else {
        setRefreshing(true);
      }
      setError(null);
      if (!user?.id) {
        throw new Error('User ID
tidak ditemukan');
      }
      const response = await
transaksiAPI.getTransaksiUser({
        role: 'penjual',
        status: filter === 'all'
? undefined : filter,
        page: 1,
        limit: 50,
        timestamp: Date.now()
      });
      let transaksiData = [];
      if (response.data?.sukses)
{
        transaksiData =
response.data.data?.transaksi ||
response.data.data || [];
      } else if
(Array.isArray(response.data)) {
        transaksiData =
response.data;
      }
      let filteredTransaksi =
transaksiData;
      if (filter === 'all') {
        filteredTransaksi =
transaksiData.filter(t =>
t.status !== 'GAGAL');
      } else {
        filteredTransaksi =
transaksiData.filter(t =>
t.status !== 'GAGAL');
      }

setTransaksi(filteredTransaksi);
      setLastRefresh(new Date());
    } catch (error) {
      const errorMessage =
error.response?.data?.pesan ||
error.message || 'Gagal memuat
transaksi';
      setError(errorMessage);
      if (transaksi.length === 0)
{
        setTransaksi([]);
      }
    } finally {
      if (showLoading) {
        setLoading(false);
      } else {
        setRefreshing(false);
      }
    }
  }, [filter, user,
transaksi.length]);

  const fetchProduk =
useCallback(async (showLoading =
true) => {
    try {
      if (showLoading) {
        setLoading(true);
      } else {
        setRefreshing(true);
      }
      setError(null);
      if (!user?.id) {
        throw new Error('User ID
tidak ditemukan');
      }
      const response = await
produkAPI.getProdukPenjual({
        userId: user.id
      });
      let produkData = [];
      if (response.data?.sukses)
{
        produkData =
response.data.data?.produk ||
response.data.data || [];
      } else if
(Array.isArray(response.data)) {
        produkData =
response.data;
      }
      setProduk(produkData);
    } catch (error) {
      const errorMessage =
error.response?.data?.pesan ||
error.message || 'Gagal memuat
produk';
      setError(errorMessage);
      if (produk.length === 0) {
        setProduk([]);
      }
    } finally {
      if (showLoading) {
        setLoading(false);
      } else {
        setRefreshing(false);
      }
    }
  }, [user, produk.length]);
  const handleRefresh = async ()
=> {
    await fetchTransaksi(false);
    await fetchProduk(false);
    toast.success('Data transaksi
dan produk diperbarui');
  };

  const formatCurrency = (amount)
=> {
    if (!amount) return 'Rp 0';
```

```
    return new
Intl.NumberFormat('id-ID', {
      style: 'currency',
      currency: 'IDR',
      minimumFractionDigits: 0,
      maximumFractionDigits: 0
    }).format(amount);
  const formatDate = (dateString)
=> {
    return new
Date(dateString).toLocaleDateStri
ng('id-ID', {
      year: 'numeric',
      month: 'short',
      day: 'numeric',
      hour: '2-digit',
      minute: '2-digit'
    });
  };const getStatusBadge =
(status) => {const statusConfig =
{MENUNGGU_PEMBAYARAN: {color:
'bg-yellow-100 text-yellow-800',
text: 'Menunggu Pembayaran',
description: 'Transaksi dibuat,
menunggu pembayaran'
      },DIBAYAR_SMARTCONTRACT:
{color: 'bg-blue-100 text-blue-
800', text: 'Pembayaran
Berhasil', description:
'Pembayaran berhasil, menunggu
pengiriman akun'},DIKIRIM:
{color: 'bg-purple-100 text-
purple-800',
        text: 'Akun Dikirim',
        description: 'Akun sudah
dikirim, perlu konfirmasi
penerimaan'
      },
      DIKONFIRMASI_PEMBELI: {
        color: 'bg-green-100
text-green-800', text:
'Dikonfirmasi', description:
'Pembeli sudah konfirmasi
penerimaan'
      },SELESAI: {color: 'bg-
green-100 text-green-800', text:
'Selesai',
description: 'Transaksi selesai'
      },
      SENGKETA: {
        color: 'bg-red-100 text-
red-800',
        text: 'Sengketa',
        description: 'Transaksi
dalam sengketa'
      },
      GAGAL: {
        color: 'bg-gray-100 text-
gray-800',
        text: 'Gagal',
        description: 'Transaksi
gagal'},REFUNDED: {color: 'bg-
blue-100 text-blue-800', text:
'Dana Dikembalikan ke Pembeli',
description: 'Dana telah
dikembalikan ke pembeli via smart
contract', isBlockchainStatus:
true},COMPLETED_DISPUTE: {
        color: 'bg-green-100
text-green-800',
        text: 'Selesai (Resolusi
Sengketa)',
        description: 'Dana telah
dikirim ke penjual via resolusi
sengketa',
        isBlockchainStatus: true
      }
    };
    const config =
statusConfig[status] || {
      color: 'bg-gray-100 text-
gray-800',
      text: status,
      description: 'Status tidak
diketahui'
    };
    const badgeClass =
config.isDisputeOutcome ||
config.isBlockchainStatus
      ? `inline-flex items-center
px-3 py-1 rounded-full text-sm
font-semibold ${config.color}
border-2
${config.color.includes('blue') ?
'border-blue-300' : 'border-
green-300'}`
      : `inline-flex items-center
px-2.5 py-0.5 rounded-full text-
xs font-medium ${config.color}`;
    return (<span
className={badgeClass}
title={config.description}>
{config.text}</span>);
  };
  const handleDetailClick =
(transaksi) => {
setSelectedTransaksi(transaksi);
setShowDetailTransaksiModal(true)
;  };const handleLihatAkun =
(transaksi) => {
 setSelectedTransaksi(transaksi);
    setShowLihatAkunModal(true);
  };const handleTerimaAkun =
(transaksi) => {
setSelectedTransaksi(transaksi);
    setShowTerimaAkunModal(true);
  };const handleKirimAkun =
(transaksi) => {
setSelectedTransaksi(transaksi);
    setShowKirimAkunModal(true);
  };
const handleKirimAkunSubmit =
async (kirimData) => {
    if (!selectedTransaksi)
return;
    try {setActionLoading(true);
```

```
      await
transaksiAPI.kirimAkun(selectedTr
ansaksi.id, kirimData); await
fetchTransaksi(false);
setShowKirimAkunModal(false);
      setSelectedTransaksi(null);
      toast.success('Data akun
berhasil dikirim ke pembeli');
    } catch (error) {
      toast.error('Gagal mengirim
akun: ' + (error.message ||
'Terjadi kesalahan'));
    } finally {
      setActionLoading(false);
    }};
  const
handleKonfirmasiPenerimaan =
async () => {
    if (!selectedTransaksi)
return;
    try {
      setActionLoading(true);
      await
transaksiAPI.konfirmasiPenerimaan
(selectedTransaksi.id); await
fetchTransaksi(false);
setShowTerimaAkunModal(false);
      setSelectedTransaksi(null);
      toast.success('Penerimaan
akun berhasil dikonfirmasi');
    } catch (error) {
      toast.error('Gagal
konfirmasi penerimaan: ' +
(error.message || 'Terjadi
kesalahan'));
    } finally {
      setActionLoading(false);
    }};
  const handleSengketa = async
(sengketaData) => {
    if (!selectedTransaksi)
return;
    try {
      setActionLoading(true);
      if
(selectedTransaksi.status ===
'SENGKETA') {
        let sengketaId =
selectedTransaksi.sengketa?.id;
        if (!sengketaId) {
          try {
            const
sengketaResponse = await
transaksiAPI.getDetailSengketa(se
lectedTransaksi.id);
            if
(sengketaResponse.data?.sukses &&
sengketaResponse.data.data?.id) {
              sengketaId =
sengketaResponse.data.data.id;
            } else {
              toast.error('Data
sengketa tidak ditemukan. Silakan
refresh halaman dan coba lagi.');
 return; }} catch {
            toast.error('Gagal
mengambil data sengketa. Silakan
refresh halaman dan coba lagi.');
            return; }}const
        await
transaksiAPI.buatPembelaan(sengke
taId, pembelaanData);
        toast.success('Pembelaan
berhasil dikirim');
        await new Promise(resolve
=> setTimeout(resolve, 1000));
      } else {
        toast.error('Penjual
tidak bisa membuat sengketa.
Hanya pembeli yang dapat membuat
sengketa. Penjual hanya bisa
memberikan pembelaan jika ada
sengketa dari pembeli.');
        return; }
      await
fetchTransaksi(false);
setShowTerimaAkunModal(false);
setShowSengketaModal(false);
      setSelectedTransaksi(null);
    } catch (error) {
      const errorMessage =
error.response?.data?.pesan ||
error.message || 'Terjadi
kesalahan';
      if
(errorMessage.includes('Hanya
pembeli yang dapat membuat
sengketa')) {
        toast.error('Penjual
tidak bisa membuat sengketa.
Hanya pembeli yang dapat membuat
sengketa. Penjual hanya bisa
memberikan pembelaan jika ada
sengketa dari pembeli.');
      } else if
(errorMessage.includes('sudah
memberikan pembelaan')) {
        toast.error('Anda sudah
memberikan pembelaan untuk
sengketa ini.');
      } else {
        toast.error('Gagal: ' +
errorMessage);
      }
    } finally {
      setActionLoading(false);
    }
  };
  const handleBuatSengketa =
(transaksi) =>
{setSelectedTransaksi(transaksi);
    setShowSengketaModal(true);
  };
```

```
  const handleBuatPembelaan =
(transaksi) => {
setSelectedTransaksi(transaksi);
    setShowSengketaModal(true);
  };
  const handleLihatDetailSengketa
= (transaksi) =>
{setSelectedTransaksi(transaksi);
setShowDetailSengketaModal(true);
  };
  if (authLoading) {
    return (
      <div className="min-h-
screen bg-gray-50 flex items-
center justify-center">
        <div className="text-
center">
          <div
className="animate-spin rounded-
full h-32 w-32 border-b-2 border-
blue-600 mx-auto"></div>
          <p className="mt-4
text-gray-600">Memuat
dashboard...</p></div></div>);}
  if (!isAuthenticated || !user)
{navigate('/masuk');
    return null;
  if (loading) {
    return (
      <div className="min-h-
screen bg-gray-50 flex items-
center justify-center">
        <div className="text-
center">
          <div
className="animate-spin rounded-
full h-32 w-32 border-b-2 border-
blue-600 mx-auto"></div>
          <p className="mt-4
text-gray-600">Memuat
transaksi...</p></div></div>);}
  return (
    <div className="min-h-screen
bg-gray-50 py-4 sm:py-8">
      <div className="max-w-[90%]
mx-auto px-2 sm:px-4 lg:px-8">
        <div className="bg-white
rounded-lg shadow-sm mb-4 sm:mb-
6"><div className="px-4 sm:px-6
py-4 border-b border-gray-200">
           <div className="flex
flex-col sm:flex-row sm:justify-
between sm:items-center space-y-3
sm:space-y-0"><div><h1
className="text-xl sm:text-2xl
font-bold text-gray-
900">Dashboard Penjual</h1>
              <p
className="text-sm text-gray-600
mt-1">Kelola transaksi dan
penjualan Anda</p>
            </div>
            <div
className="flex flex-col sm:flex-
row gap-2 sm:text-right">
              <button
                onClick={() =>
className="inline-flex items-
center justify-center px-3 py-2
border border-gray-300 shadow-sm
text-sm leading-4 font-medium
rounded-md text-gray-700 bg-white
hover:bg-gray-50
disabled:opacity-50">{refreshing
? 'Memperbarui...' :
'Perbarui'}</button><p
className="text-xs text-gray-500
mt-1 text-center sm:text-right">
Terakhir diperbarui:
{lastRefresh.toLocaleTimeString('
id-ID')}
              </p>
            </div>
          </div>
        </div>
<div className="bg-white rounded-
lg shadow-sm">
<div className="p-3 sm:p-<div
className="flex"><div
className="ml-0"><h3
className="text-sm font-medium
text-red-800">Gagal memuat
transaksi</h3><div className="mt-
2 text-sm text-red-
700"><p>{error}</p></div><div
className="mt-4 flex flex-col
sm:flex-row gap-
2"><buttononClick={() =>
fetchTransaksi(true)}
className="bg-red-100 px-3 py-2
rounded-md text-sm font-medium
text-red-800 hover:bg-red-200">
Coba Lagi</button><button
           onClick={() => {
              setError(null)
setTransaksi([]);
fetchTransaksi(true);}}
className="bg-blue-100 px-3 py-2
rounded-md text-sm font-medium
text-blue-800 hover:bg-blue-200"
>Reset & Muat
Ulang</button>{error &&
error.toLowerCase().includes('tok
en') && (<buttononClick={() =>
navigate('/masuk')}
className="bg-green-100 px-3 py-2
rounded-md text-sm font-medium
text-green-800 hover:bg-green-
200">Login Ulang</button>)}
                  </div>
                </div>
              </div>
            </div>
)}<div className="mb-4">
```

```
<h2 className="text-base sm:text-lg font-semibold text-gray-900">Riwayat Transaksi</h2></div>
{transaksi.length === 0 ? (
              <div className="px-3 sm:px-6 py-12 text-center">
                <h3 className="mt-2 text-sm font-medium text-gray-900">{error ? 'Tidak dapat memuat transaksi' : 'Belum ada transaksi'}</h3><p className="mt-1 text-sm text-gray-500">{error'Terjadi masalah saat memuat data.
: 'Transaksi penjualan akan muncul di sini setelah ada pembeli.'}</p><div className="mt-6">error ? (<div className="flex flex-col sm:flex-row gap-2 justify-center"><buttononClick={() => fetchTransaksi(true)} className="inline-flex items-center justify-center px-4 py-2 border border-transparent shadow-sm text-sm font-medium rounded-md text-white bg-red-600 hover:bg-red-700">Coba Lagi</button><buttononClick={() => navigate('/masuk')} className="inline-flex items-center justify-center px-4 py-2 border border-gray-300 shadow-sm text-sm font-medium rounded-md text-gray-700 bg-white hover:bg-gray-50">Login Ulang</button></div>) : (<buttononClick={() => navigate('/produk-saya')}
     className="inline-flex items-center justify-center px-4 py-2 border border-transparent shadow-sm text-sm font-medium rounded-md text-white bg-blue-600 hover:bg-blue-700">Jual Akun Pertama</button>)}
                </div>
              </div>
            ) : (
              <div
```

```
<table className="min-w-full divide-y divide-gray-200"><thead className="bg-gray-50"><tr><th className="px-4 py-2 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">Produk</th><th
```

```
<th
```

```
<th
```

```
<
```

```
{item.pembeli?.walletAddress && (
<div className="text-xs text-
gray-500 font-
mono">{item.pembeli.walletAddress
.substring(0,
6)}...{item.pembeli.walletAddress
.substring(item.pembeli.walletAdd
ress.length - 4)}
</div>)}</td><td className="px-4
py-2 whitespace-
nowrap">{getStatusBadge(item.stat
us)} {item.status === 'SENGKETA'
&& (<div className="text-xs text-
red-600">Sengketa
Aktif</div>)}</td><td
className="px-4 py-2 whitespace-
nowrap"><div className="text-sm
text-gray-
500">{formatDate(item.dibuatPada)
} </div></td><td className="px-4
py-2 whitespace-nowrap"><div
className="flex flex-wrap gap-1">
     <button onClick={() =>
handleDetailClick(item)}
className="inline-flex items-
center px-2 py-1 border border-
gray-300 text-xs font-medium
rounded text-gray-700 bg-white
hover:bg-gray-50"title="Detail
Transaksi">Detail</button>{(() =>
{const hasData =
accountDataHelper.hasAccountData(
item); const validStatus =
['DIKIRIM',
'DIKONFIRMASI_PEMBELI',
'SELESAI'].includes(item.status);
return hasData && validStatus &&
(<buttononClick={() =>
handleLihatAkun(item)}
className="inline-flex items-
center px-2 py-1 border border-
green-300 text-xs font-medium
rounded text-green-700 bg-green-
50 hover:bg-green-
100"{item.status ===
'DIBAYAR_SMARTCONTRACT' &&
(<buttononClick={() =>
handleKirimAkun(item)}
className="inline-flex items-
center px-2 py-1 border border-
transparent text-xs font-medium
rounded text-white bg-blue-600
hover:bg-blue-700"
          title="Kirim Akun"
>Kirim</button>
hasPembelaan =
item.sengketa?.penjualBukti
||item.sengketa?.resolution?.incl
udes('[PEMBELAAN PENJUAL]') ||
(item.sengketa?.status ===
'DIPROSES' &&
item.sengketa?.penjualAlasan);
return item.status === 'SENGKETA'
&& !hasPembelaan &&
(<buttononClick={() =>
handleBuatPembelaan(item)}
className="inline-flex items-
center px-2 py-1 border border-
red-300 text-xs font-medium
rounded text-red-700 bg-red-50
hover:bg-red-100"title="Beri
Pembelaan"{item.status ===
'SENGKETA' &&
(<buttononClick={()handleLihatDet
ailSengketa(item)}
className="inline-flex items-
center px-2 py-1 border border-
orange-300 text-xs font-medium
rounded text-orange-700 bg-
orange-50 hover:bg-orange-
100"title="Detail
Sengketa"<ModalLihatAkunisOpen={s
howLihatAkunModal}
          onClose={() =>
{setShowLihatAkunModal(false);
setSelectedTransaksi(null);
}}transaksi={selectedTransaksi}/>
<ModalTerimaAkunisOpen={showTerim
aAkunModal}onClose={() =>
{setShowTerimaAkunModal(false);
setSelectedTransaksi(null);
}}onConfirm={handleKonfirmasiPene
rimaan}onDispute={handleSengketa}
loading={actionLoading}transaksi=
{selectedTransaksi}/><ModalKirimA
kunisOpen={showKirimAkunModal}
          onClose={() =>
{setShowKirimAkunModal(false);
setSelectedTransaksi(null);
}}onSubmit={handleKirimAkunSubmit
}loading={actionLoading}transaksi
={selectedTransaksi}/><ModalDetai
lSengketaisOpen={showDetailSengke
taModal}onClose={() =>
{setShowDetailSengketaModal(false
); setSelectedTransaksi(null);
}}transaksi={selectedTransaksi}/>
<ModalSengketaPenjualisOpen={show
SengketaModal}onClose={() =>
{setShowSengketaModal(false);
setSelectedTransaksi(null);
}}onSubmit={handleSengketa}loadin
g={actionLoading}transaksi={selec
tedTransaksi}/><ModalDetailTransa
ksiisOpen={showDetailTransaksiMod
al}onClose={setSelectedTransaksi(
null);
}}transaksi={selectedTransaksi}on
Refresh={fetchTransaksi}
        />
      </div>
    </div>
};
export default DashboardPenjual;
```

Produk.jsx

```
import React, { useState,
useEffect, import React, {
useState, useEffect } from
const Beranda = () => {
  const { isAuthenticated, user }
= useAuth()
  const { isConnected } =
useWallet()
  const [gameStats, setGameStats]
= useState([])
  const [loading, setLoading] =
useState(true)
  useEffect(() => {
    const fetchGameStats = async
() => {try {
        const API_BASE_URL =
process.env.REACT_APP_API_URL ||
'http://localhost:5000/api'
        const response = await
fetch(`${API_BASE_URL}/produk/game-populer`)
        if (!response.ok) {
          throw new Error(`HTTP
error! status:
${response.status}`)
        }const data = await
response.json()
        if (data.sukses)
{setGameStats(data.data.gamePopuler) }} catch (error) {
        setGameStats([])
      } finally {
        setLoading(false)
      }}
    fetchGameStats()
  }, [])
  const gameImages = {
    'Mobile Legends': MLImage,
    'Free Fire': FFImage,
    'PUBG Mobile': PUBGImage,
    'Genshin Impact': GIImage}
  const games = gameStats.length
> 0
    ? gameStats.map(game => ({
        name: game.namaGame,
        image:
gameImages[game.namaGame] ||
MLImage,
        count:
`${game.jumlahProduk} akun`,
        description:
gameDescriptions[game.namaGame]
|| 'Akun game berkualitas tinggi'
      }))
    : defaultGames.map(game => {
        const apiGame =
gameStats.find(g => g.namaGame
=== game.name)
        return {game,
count: apiGame ?
`${apiGame.jumlahProduk} akun` :
game.count}})
  const loadingGames =
defaultGames.map(game => ({
    ...game,
    count: 'Loading...'
  }))
  const displayGames = loading ?
loadingGames : games
  if (isAuthenticated &&
user?.role === 'ADMIN') {
    return <Navigate to="/admin"
replace />}
  return (
    <div className="bg-white">
      <div className="relative
bg-gradient-to-r from-primary-600
to-primary-800"><div
className="absolute inset-0 bg-
black opacity-20"></div><div
className="relative max-w-[90%]
mx-auto py-8 px-4 sm:py-12 md:py-
16 sm:px-6 lg:px-8">
        <div className="mt-6
sm:mt-8 md:mt-10 flex flex-col
sm:flex-row gap-3 sm:gap-4">
{!isAuthenticated ?
(<><Linkto="/masuk"className="inl
ine-flex items-center justify-
center px-4 sm:px-6 py-2.5 sm:py-
3 border border-transparent text-
sm sm:text-base font-medium
rounded-md text-primary-700 bg-
white hover:bg-gray-50
transition-colors w-full sm:w-
auto">Mulai
Sekarang</Link><Linkto="/produk"
className="inline-flex items-
center justify-center px-4 sm:px-
6 py-2.5 sm:py-3 border border-
white text-sm sm:text-base font-
medium rounded-md text-white
hover:bg-white hover:text-
primary-700 transition-colors w-
full sm:w-auto">Lihat
Produk</Link></>) :
(<Linkto="/produk"className="inli
ne-flex items-center justify-
center px-4 sm:px-6 py-2.5 sm:py-
3 border border-transparent text-
sm sm:text-base font-medium
rounded-md text-primary-700 bg-
white hover:bg-gray-50
transition-colors w-full sm:w-
auto">Jelajahi Produk
              </Link>
            )}
          </div>
        </div>
      </div>
      <div className="py-12
sm:py-16 bg-white">
        <div className="max-w-
[90%] mx-auto px-4 sm:px-6 lg:px-
8"><div className="text-center">
```

```
<h2 className="text-xs sm:text-sm
md:text-base text-primary-600
font-semibold tracking-wide
uppercase">Game Populer</h2><p
className="mt-2 text-2xl sm:text-
3xl leading-7 sm:leading-8 font-
extrabold tracking-tight text-
gray-900 md:text-4xl">Akun Game
Terlengkap</p><p className="mt-3
sm:mt-4 max-w-2xl text-base
sm:text-lg md:text-xl text-gray-
500 mx-auto leading-
relaxed">Temukan akun game impian
Anda dari berbagai game
populer</p></div><div
className="mt-10 sm:mt-12 md:mt-
16 grid grid-cols-1 gap-6
sm:grid-cols-2 lg:grid-cols-4
sm:gap-
8">{displayGames.map((game,
index) => (<Link key={`game-
${index}-${game.name}`}
to={`/produk?game=${encodeURIComp
onent(game.name)}`}
className="relative group cursor-
pointer"><div className="relative
w-full h-56 bg-gray-200 rounded-
xl overflow-hidden group-
hover:opacity-75 transition-all
duration-300 shadow-lg group-
hover:shadowxl"><imgsrc={game.ima
ge}alt={game.name}className="w-
full h-full object-cover object-
center group-hover:scale-105
transition-transform duration-
300"onError={(e) =>
{e.target.style.display = 'none'
e.target.nextSibling.style.displa
y = 'flex'}}/><div className="w-
full h-full bg-gradient-to-br
from-primary-400 to-primary-600
flex items-center justify-center"
style={{display: 'none'}}><span
className="text-white font-bold
text-xl">{game.name}</span></div>
<div className="absolute inset-0
bg-gradient-to-t from-black/20
to-transparent opacity-0 group-
hover:opacity-100 transition-
opacity duration-300"></div><div
className="absolute inset-0 bg-
black/40 opacity-0 group-
hover:opacity-100 transition-
opacity duration-300 flex items-
center justify-center"><div
className="text-white text-
center"><div className="text-lg
font-semibold mb-1">Lihat
Produk</div><div className="text-
sm">{game.name}</div></div>
                  </div>
                </div><div
className="mt-4 text-center">
<h3 className="text-lg font-
semibold text-gray-900 mb-1
group-hover:text-primary-600
transition-
colors">{game.name}</h3><p
className="text-sm font-medium
text-primary-
600">{game.count}</p></div>
              </Link>
            ))}
          </div>
          <div className="mt-12
text-center"><Linkto="/produk"
              className="inline-
flex items-center px-6 py-3
border border-transparent text-
base font-medium rounded-md text-
white bg-primary-600 hover:bg-
primary-700 transition-colors"
>Lihat Semua Produk</Link>
          </div>
        </div>
      </div><div className="py-12
sm:py-16 bg-gray-50">
        <div className="max-w-
[90%] mx-auto px-4 sm:px-6 lg:px-
8"><div className="text-center
lg:text-center"><h2
className="text-xs sm:text-sm
md:text-base text-primary-600
font-semibold tracking-wide
uppercase">Keunggulan</h2><p
className="mt-2 text-2xl sm:text-
3xl leading-7 sm:leading-8 font-
extrabold tracking-tight text-
gray-900 md:text-4xl">Mengapa
Memilih Jubel? </p><
className="space-y-8 sm:space-y-
10 md:space-y-0 md:grid md:grid-
cols-2 md:gap-x-6 lg:gap-x-8
md:gap-y-8 lg:gap-y-
10">features.map((feature, index)
=> (<div key={`feature-${index}-
${feature.name}`}
className="relative"><dt><p
className="ml-12 sm:ml-16 text-
base sm:text-lg leading-5
sm:leading-6 font-medium text-
gray-900">{feature.name}</p></dt>
<dd className="mt-2 ml-12 sm:ml-
16 text-sm sm:text-base text-
gray-500 leading-
relaxed">{feature.description}
</dd>
              </div>
            ))}
          </dl>
        </div>
      </div>
    </div>
  )export default Beranda export
default Produk;
```

Beranda.jsx

```
import React, { useState,
useEffect } from 'react'
import { Link, Navigate } from
'react-router-dom'
import { useAuth } from
'../../konteks/AuthContext'
import { useWallet } from
'../../konteks/WalletContext'
import MLImage from
'../../gambar/ML.jpg'
import FFImage from
'../../gambar/FF.jpg'
import PUBGImage from
'../../gambar/PUBGM.jpg'
import GIImage from
'../../gambar/GI.jpg'
const Beranda = () => {
  const { isAuthenticated, user }
= useAuth()
  const { isConnected } =
useWallet()
  const [gameStats, setGameStats]
= useState([])
  const [loading, setLoading] =
useState(true)

  useEffect(() => {
    const fetchGameStats = async
() => {
      try {
        const API_BASE_URL =
process.env.REACT_APP_API_URL ||
'http://localhost:5000/api'
        const response = await
fetch(`${API_BASE_URL}/produk/game-populer`)
        if (!response.ok) throw
new Error(`HTTP error! status:
${response.status}`)
        const data = await
response.json()
        if (data.sukses)
setGameStats(data.data.gamePopuler)
      } catch (error) {
        setGameStats([])
      } finally {
        setLoading(false)
      }
    }
    fetchGameStats()
  }, [])
  const gameImages = {
    'Mobile Legends': MLImage,
    'Free Fire': FFImage,
    'PUBG Mobile': PUBGImage,
    'Genshin Impact': GIImage
  }
  const gameDescriptions = {
    'Mobile Legends': 'Akun ML
dengan rank tinggi dan skin
rare',
    'Free Fire': 'Akun FF dengan
diamond dan karakter premium',
    'PUBG Mobile': 'Akun PUBG
dengan tier tinggi dan outfit
eksklusif',
    'Genshin Impact': 'Akun
Genshin dengan karakter 5-star
dan weapon rare'
  }

  const features = [
    { name: 'Transaksi Aman',
description: 'Menggunakan smart
contract escrow untuk
transparansi transaksi maksimal'
},
    { name: 'Pembayaran Crypto',
description: 'Pembayaran
menggunakan Ethereum di Sepolia
testnet' },
    { name: 'Marketplace
Terpercaya', description:
'Platform jual beli akun game
yang terpercaya dan transparan'
},
    { name: 'Komunitas Aktif',
description: 'Bergabung dengan
ribuan gamer yang sudah
mempercayai kami' }
  ]
  const defaultGames = [
    { name: 'Mobile Legends',
image: MLImage, count: '0 akun',
description: 'Akun ML dengan rank
tinggi dan skin rare' },
    { name: 'Free Fire', image:
FFImage, count: '0 akun',
description: 'Akun FF dengan
diamond dan karakter premium' },
    { name: 'PUBG Mobile', image:
PUBGImage, count: '0 akun',
description: 'Akun PUBG dengan
tier tinggi dan outfit eksklusif'
},
    { name: 'Genshin Impact',
image: GIImage, count: '0 akun',
description: 'Akun Genshin dengan
karakter 5-star dan weapon rare'
}
  ]
  const games = gameStats.length
> 0
    ? gameStats.map(game => ({
      name: game.namaGame,
      image:
gameImages[game.namaGame] ||
MLImage,
      count:
`${game.jumlahProduk} akun`,
      description:
gameDescriptions[game.namaGame]
|| 'Akun game berkualitas tinggi'
    }))
```

```
    : default{Games}.map({game} => {
      const api{Game} =
{game}Stats.find(g => g.nama{Game}
=== {game}.name)
      return { ...{game}, count:
api{Game} ?
`${api{Game}.jumlahProduk} akun` :
{game}.count }})
  const loading{Games} =
default{Games}.map({game} => ({
...{game}, count: 'Loading...' }))
  const display{Games} = loading ?
loading{Games} : {games}
  if (isAuthenticated &&
user?.role === 'ADMIN') {
    return <Navigate to="/admin"
replace />
  }return (<div className="bg-
white"><div className="relative
bg-gradient-to-r from-primary-600
to-primary-800"><div
className="absolute inset-0 bg-
black opacity-20"></div><div
className="relative max-w-[90%]
mx-auto py-8 px-4 sm:py-12 md:py-
16 sm:px-6 lg:px-8">
          <h1 className="text-xl
font-extrabold tracking-tight
text-white sm:text-2xl md:text-
3xl lg:text-4xl xl:text-5xl">Jual
Beli Akun {Game}<span
className="block text-primary-200
mt-1">Dengan Transparansi
Transaksi
Blockchain</span></h1><p
className="mt-4 sm:mt-6 max-w-3xl
text-base sm:text-lg md:text-xl
text-primary-100 leading-
relaxed">Platform marketplace
pertama di Indonesia yang
menggunakan teknologi smart
contract untuk menjamin
transparansi transaksi jual beli
akun {game}.
          </p>
          <div className="mt-6
sm:mt-8 md:mt-10 flex flex-col
sm:flex-row gap-3 sm:gap-
4">{!isAuthenticated ? (<><Link
to="/masuk" className="inline-
flex items-center justify-center
px-4 sm:px-6 py-2.5 sm:py-3
border border-transparent text-sm
sm:text-base font-medium rounded-
md text-primary-700 bg-white
hover:bg-gray-50 transition-
colors w-full sm:w-auto">Mulai
Sekarang</Link><Link to="/produk"
className="inline-flex items-
center justify-center px-4 sm:px-
6 py-2.5 sm:py-3 border border-
white text-sm sm:text-base font-
medium rounded-md text-white
hover:bg-white hover:text-
primary-700 transition-colors w-
full sm:w-auto">Lihat
Produk</Link></>) : (<Link
to="/produk" className="inline-
flex items-center justify-center
px-4 sm:px-6 py-2.5 sm:py-3
border border-transparent text-sm
sm:text-base font-medium rounded-
md text-primary-700 bg-white
hover:bg-gray-50 transition-
colors w-full sm:w-auto">
                Jelajahi Produk
              </Link>)}
          </div>
        </div>
      </div>
      <div className="py-12
sm:py-16 bg-white">
        <div className="max-w-
[90%] mx-auto px-4 sm:px-6 lg:px-
8"><div className="text-
center"><h2 className="text-xs
sm:text-sm md:text-base text-
primary-600 font-semibold
tracking-wide uppercase">{Game}
Populer</h2><p className="mt-2
text-2xl sm:text-3xl leading-7
sm:leading-8 font-extrabold
tracking-tight text-gray-900
md:text-4xl">Akun {Game}
Terlengkap</p><p className="mt-3
sm:mt-4 max-w-2xl text-base
sm:text-lg md:text-xl text-gray-
500 mx-auto leading-
relaxed">Temukan akun {game} impian
Anda dari berbagai {game}
populer</p></div>
          <div className="mt-10
sm:mt-12 md:mt-16 grid grid-cols-
1 gap-6 sm:grid-cols-2 lg:grid-
cols-4 sm:gap-
8">{display{Games}.map(({game},
index) => (<Link key={`{game}-
${index}-${{game}.name}`}
to={`/produk?{game}=${encodeURIComp
onent({game}.name)}`}
className="relative group cursor-
pointer"><div className="relative
w-full h-56 bg-gray-200 rounded-
xl overflow-hidden group-
hover:opacity-75 transition-all
duration-300 shadow-lg group-
hover:shadow-xl"><img
src={{game}.image} alt={{game}.name}
className="w-full h-full object-
cover object-center group-
hover:scale-105 transition-
transform duration-300"
onError={e => {
e.target.style.display = 'none';
e.target.nextSibling.style.displa
y = 'flex' }} />
```

```
<div className="w-full h-full bg-gradient-to-br from-primary-400 to-primary-600 flex items-center justify-center" style={{ display: 'none' }}><span className="text-white font-bold text-xl">{game.name}</span></div><div className="absolute inset-0 bg-gradient-to-t from-black/20 to-transparent opacity-0 group-hover:opacity-100 transition-opacity duration-300"></div><div className="absolute inset-0 bg-black/40 opacity-0 group-hover:opacity-100 transition-opacity duration-300 flex items-center justify-center"><div className="text-white text-center"><div className="text-lg font-semibold mb-1">Lihat Produk</div><div className="text-sm">{game.name}</div>
                    </div>
                  </div>
                </div><div
className="mt-4 text-center"><h3
className="text-lg font-semibold
text-gray-900 mb-1 group-
hover:text-primary-600
transitioncolors">{game.name}</h3
><p className="text-sm font-
medium text-primary-
600">{game.count}</p>
                </div>
              </Link>
            ))}
          </div>
          <div className="mt-12
text-center">
            <Link to="/produk"
className="inline-flex items-
center px-6 py-3 border border-
transparent text-base font-medium
rounded-md text-white bg-primary-
600 hover:bg-primary-700
transition-colors">
              Lihat Semua Produk
            </Link>
          </div>
        </div>
      </div>
      <div className="py-12
sm:py-16 bg-gray-50"><div
className="max-w-[90%] mx-auto
px-4 sm:px-6 lg:px-8">div
className="text-center lg:text-
center"><h2 className="text-xs
sm:text-sm md:text-base text-
primary-600 font-semibold
tracking-wide
uppercase">Keunggulan</h2><p
className="mt-2 text-2xl sm:text-
3xl leading-7 sm:leading-8 font-
extrabold tracking-tight text-
gray-900 md:text-4xl">Mengapa
Memilih Jubel?</p>
            <p className="mt-3
sm:mt-4 max-w-2xl text-base
sm:text-lg md:text-xl text-gray-
500 mx-auto leading-relaxed">Kami
menggunakan teknologi blockchain
terdepan untuk memberikan
pengalaman jual beli yang aman
dan transparan.</p>
          </div>
          <div className="mt-10
sm:mt-12 md:mt-16">
            <dl className="space-
y-8 sm:space-y-10 md:space-y-0
md:grid md:grid-cols-2 md:gap-x-6
lg:gap-x-8 md:gap-y-8 lg:gap-y-
10">{features.map((feature,
index) => (<div key={`feature-
${index}-${feature.name}`}
className="relative"> <dt><p
className="ml-12 sm:ml-16 text-
base sm:text-lg leading-5
sm:leading-6 font-medium text-
gray-900">{feature.name}</p>
                  </dt>
                  <dd
className="mt-2 ml-12 sm:ml-16
text-sm sm:text-base text-gray-
500 leading-
relaxed">{feature.description}</d
d></div>))}
            </dl>
          </div>
        </div>
      </div>
    </div>
  )
}
export default Beranda
```

PembayaranAdmin.jsx

```
import React, { useState,
useEffect } from 'react'
import { useNavigate } from
'react-router-dom'
import { adminAPI as oldAdminAPI
} from '../../layanan/api'
import AdminNavigation from
'../../komponen/AdminNavigation'
import ModalDetailSengketaAdmin
from
'../../komponen/ModalDetailSengke
taAdmin'
const API_BASE_URL =
process.env.REACT_APP_API_URL ||
'http://localhost:5000/api'
const adminAPI = {
  getAllTransaksi: async () => {
    try {const response = await
fetch(`${API_BASE_URL}/admin-
new/transaksi`, {
```

```
        method: 'GET', headers:
{Authorization: `Bearer
${localStorage.getItem('authToken
')}`, 'Content-Type':
'application/json'}
      })
      const data = await
response.json()
      if (!response.ok) throw
datareturn data
    } catch (error) {
      return await
oldAdminAPI.getAllTransaksi()
    }
  },
  getAllSengketa: async (params =
{}) => {try {
      const queryString = new
URLSearchParams(params).toString(
)const response = await
fetch(`${API_BASE_URL}/admin/seng
keta${queryString ?
`?${queryString}` : ''}`, {
        method: 'GET',
        headers: {
          Authorization: `Bearer
${localStorage.getItem('authToken
')}`, 'Content-Type':
'application/json'}
      })
      const data = await
response.json()
      if (!response.ok) throw
data return data
    } catch (error) {
      return await
oldAdminAPI.getAllSengketa(params
)}},
  releaseFunds: async
(transaksiId, reason) => {
    const response = await
fetch(`${API_BASE_URL}/admin-
new/transaksi/${transaksiId}/rele
ase-funds`, {
      method: 'POST',
      headers: {
        Authorization: `Bearer
${localStorage.getItem('authToken
')}`, 'Content-Type':
'application/json'},
      body: JSON.stringify({
reason })
    })const data = await
response.json()
    if (!response.ok) throw data
    return data
  },resolveDisputeToSeller: async
(transaksiId, reason) => {
    const response = await
fetch(`${API_BASE_URL}/admin-
new/transaksi/${transaksiId}/reso
lve-dispute-seller`, {
      method: 'POST',headers: {
Authorization: `Bearer
${localStorage.getItem('authToken
')}`, 'Content-Type':
'application/json'},
      body: JSON.stringify({
reason })
    })
    const data = await
response.json()
    if (!response.ok) throw data
    return data},
  resolveDisputeToBuyer: async
(transaksiId, reason) => {
    const response = await
fetch(`${API_BASE_URL}/admin-
new/transaksi/${transaksiId}/reso
lve-dispute-buyer`, {method:
'POST' headers: {Authorization:
`Bearer
${localStorage.getItem('authToken
')}`, 'Content-Type':
'application/json'
      },body: JSON.stringify({
reason })
    })
    const data = await
response.json()
    if (!response.ok) throw data
    return data
  }
}
const PembayaranAdmin = () => {
  const [transaksi, setTransaksi]
= useState([])
  const [displayData,
setDisplayData] = useState([])
  const [loading, setLoading] =
useState(true)
  const [paymentLoading,
setPaymentLoading] =
useState(false)
  const [filter, setFilter] =
useState('all')
  const [selectedTransaksi,
setSelectedTransaksi] =
useState(null)
  const [showPaymentModal,
setShowPaymentModal] =
useState(false)
  const [showDisputeModal,
setShowDisputeModal] =
useState(false)
  const [showReasonModal,
setShowReasonModal] =
useState(false)
  const [showConfirmModal,
setShowConfirmModal] =
useState(false)
  const [paymentReason,
setPaymentReason] = useState('')
  const [notification,
setNotification] = useState(null)
  const navigate = useNavigate()
```

```
  useEffect(() => {
    if (notification) {
      const timer = setTimeout(()
=> setNotification(null), 5000)
      return () =>
clearTimeout(timer) }},
[notification])
  useEffect(() => {
    fetchTransaksi()
  }, [])
  useEffect(() => {
    fetchTransaksi()
  }, [filter])
  const fetchTransaksi = async ()
=> {try {setLoading(true)
      if (filter === 'SENGKETA')
{const [transaksiResponse,
sengketaResponse] = await
Promise.all([adminAPI.getAllTrans
aksi(),
adminAPI.getAllSengketa()
        ]) let combinedData = []
if (transaksiResponse.data &&
transaksiResponse.data.sukses !==
false) {const transaksiData =
transaksiResponse.data.data?.tran
saksi ||
transaksiResponse.data.transaksi
|| []const sengketaTransaksi =
transaksiData.filter(t => {
const hasValidEscrowId =
t.escrowId && t.escrowId !== null
&& t.escrowId !== ''const
isSengketaStatus = t.status ===
'SENGKETA'const isNotFailedStatus
= t.status !== 'GAGAL' &&
t.status !== 'DIBATALKAN'return
hasValidEscrowId &&
isSengketaStatus &&
isNotFailedStatus})combinedData =
[...combinedData,
...sengketaTransaksi] }if
(sengketaResponse.data &&
sengketaResponse.data.sukses !==
false) {const sengketaData =
sengketaResponse.data.data?.sengk
eta ||
sengketaResponse.data.sengketa ||
[]const formattedSengketa =
sengketaData.map(s =>
({s.transaksi, sengketaData: s,
isSengketaItem:
true}))combinedData =
[...combinedData,
...formattedSengketa]
}setDisplayData(combinedData)
      } else {
        const response = await
adminAPI.getAllTransaksi()
        if (response.data &&
response.data.sukses !== false)
{const transaksiData =
response.data.data?.transaksi ||
response.data.transaksi || []
          const validTransaksi =
transaksiData.filter(t => {const
hasValidEscrowId = t.escrowId &&
t.escrowId !== null && t.escrowId
!== ''const isNotFailedStatus =
t.status !== 'GAGAL' && t.status
!== 'DIBATALKAN'const
isValidStatus = t.status ===
'SELESAI' || t.status ===
'DIKONFIRMASI_PEMBELI' ||
t.status === 'SENGKETA'return
hasValidEscrowId &&
isNotFailedStatus &&
isValidStatus})
          let filteredTransaksi =
validTransaksiif (filter ===
'DIKONFIRMASI_PEMBELI')
{filteredTransaksi =
validTransaksi.filter(t =>
t.status ===
'DIKONFIRMASI_PEMBELI') } else if
(filter === 'SELESAI')
{filteredTransaksi =
validTransaksi.filter(t =>
t.status === 'SELESAI')
}setTransaksi(Array.isArray(filte
redTransaksi) ? filteredTransaksi
:[])setDisplayData(Array.isArray(
filteredTransaksi) ?
filteredTransaksi : [])
        } else {
          throw new
Error(response.data?.pesan ||
'Failed to load transaksi')
        }
      }
      setTransaksi([])
      setDisplayData([])
    } finally {
      setLoading(false)
    }
  const handlePayment = async ()
=> {if (!selectedTransaksi)
returnif(!selectedTransaksi.escro
wId) {setNotification({ type:
'error', message: 'Escrow ID
tidak ditemukan' })return}
    const sellerWallet =
selectedTransaksi.penjual?.wallet
Addressif (!sellerWallet) {
      setNotification({ type:
'error', message: 'Alamat wallet
penjual tidak ditemukan'
})return}if (!/^0x[a-fA-F0-
9]{40}$/.test(sellerWallet)) {
      setNotification({ type:
'error', message: 'Format alamat
wallet penjual tidak valid' })
return}setShowReasonModal(true)
  }const processPayment = async
reason => {
```

```
    if (!reason || reason.length
< 10) { setNotification({ type:
'error', message: 'Alasan
pembayaran harus diisi minimal 10
karakter' })return}
    setShowReasonModal(false)
    setShowConfirmModal(true)
    setPaymentReason(reason)
  }
  const confirmPayment = async ()
=> {try {
      setPaymentLoading(true)
      setShowConfirmModal(false)
      const response = await
adminAPI.releaseFunds(selectedTra
nsaksi.id, paymentReason)
      if (response.sukses)
{setShowPaymentModal(false)
setSelectedTransaksi(null)
        await fetchTransaksi()
        setNotification({
          type: 'success',
          message: `Pembayaran
berhasil! Dana
${selectedTransaksi.escrowAmount}
ETH telah dikirim ke
${selectedTransaksi.penjual.nama}
        })} else {
        throw new
Error(response.pesan || 'Failed
to release funds')
      }} catch (error) {
      let errorMessage =
'Pembayaran gagal. '
      if (error.name ===
'TypeError' &&
error.message?.includes('fetch'))
{errorMessage += 'Periksa koneksi
internet Anda.'
      } else if (typeof error ===
'object' && error.pesan) {
        errorMessage +=
error.pesan
      } else if (error.message) {
        errorMessage +=
error.message
      } else {
        errorMessage += 'Terjadi
kesalahan yang tidak diketahui'
      }
      setNotification({ type:
'error', message: errorMessage })
    } finally {
      setPaymentLoading(false)
    }
  }
  const handleDisputePayment =
async payTo => {
    if (!selectedTransaksi)
return
    if
(!selectedTransaksi.escrowId)
{setNotification({ type: 'error',
message: 'Escrow ID tidak
ditemukan' })return}
    const recipient = payTo ===
'seller' ?
selectedTransaksi.penjual :
selectedTransaksi.pembeli
    const recipientWallet = payTo
=== 'seller' ?
selectedTransaksi.penjual?.wallet
Address :
selectedTransaksi.pembeli?.wallet
Address
    if (!recipientWallet) {
      setNotification({ type:
'error', message: `Alamat wallet
${payTo === 'seller' ? 'penjual'
: 'pembeli'} tidak ditemukan` })
      return}
    if (!/^0x[a-fA-F0-
9]{40}$/.test(recipientWallet)) {
      setNotification({ type:
'error', message: `Format alamat
wallet ${payTo === 'seller' ?
'penjual' : 'pembeli'} tidak
valid` })return}try {
      setPaymentLoading(true)
      const reason = `Admin
menyelesaikan sengketa untuk
transaksi
${selectedTransaksi.kodeTransaksi
}`const response = payTo ===
'seller'? await
adminAPI.resolveDisputeToSeller(s
electedTransaksi.id, reason):
awaitadminAPI.resolveDisputeToBuy
er(selectedTransaksi.id, reason)
if (response.sukses)
{setShowPaymentModal(false)
setSelectedTransaksi(null)
        await fetchTransaksi()
        setNotification({
          type: 'success',
          message: `Sengketa
berhasil diselesaikan! Dana
${selectedTransaksi.escrowAmount}
ETH telah dikirim ke
${recipient.nama}`
        })} else {throw new
Error(response.pesan || 'Failed
to resolve dispute') }} catch
(error) {let errorMessage =
'Penyelesaian sengketa gagal. '
      if (error.name ===
'TypeError' &&
error.message?.includes('fetch'))
{errorMessage += 'Periksa koneksi
internet Anda.'} else if (typeof
error === 'object' &&
error.pesan) {errorMessage +=
error.pesan} else if
(error.message) {errorMessage +=
error.message} else {errorMessage
```

```
+= 'Terjadi kesalahan yang tidak
diketahui'}
      setNotification({ type:
'error', message: errorMessage })
    } finally {
      setPaymentLoading(false)
    }
  }
  const getStatusBadge = status
=> {const statusConfig = {
DIKONFIRMASI_PEMBELI: { color:
'bg-blue-100 text-blue-800',
text: 'Perlu Bayar Penjual' },
      SELESAI: { color: 'bg-
green-100 text-green-800', text:
'Selesai' },
      SENGKETA: { color: 'bg-red-
100 text-red-800', text:
'Sengketa Aktif' },
      DIPROSES: { color: 'bg-
yellow-100 text-yellow-800',
text: 'Perlu Review' },
      DIMENANGKAN_PEMBELI: {
color: 'bg-blue-100 text-blue-
800', text: 'Pembeli Menang' },
      DIMENANGKAN_PENJUAL: {
color: 'bg-green-100 text-green-
800', text: 'Penjual Menang' }}
    const config =
statusConfig[status] || { color:
'bg-gray-100 text-gray-800',
text: status }return <span
className={`px-2 py-1 rounded
text-xs font-medium
${config.color}`}>{config.text}</
span>}
  const handleDetailClick =
transaksiItem =>
{setSelectedTransaksi(transaksiIt
em)
    if
(transaksiItem.sengketaData ||
transaksiItem.status ===
'SENGKETA') {
      setShowDisputeModal(true)
    } else {
      setShowPaymentModal(true)
    }
  }
  const handleDisputeResolve =
async winner => {
    if (!selectedTransaksi)
return
    try {
      const result = await
handleDisputePaymentWithReturn(wi
nner)
      return result
    } catch (error) {
      throw error
    }}const
handleDisputePaymentWithReturn =
async payTo => {
    if (!selectedTransaksi) throw
new Error('Transaksi tidak
ditemukan') if
(!selectedTransaksi.escrowId)
throw new Error('Escrow ID tidak
ditemukan')
    const recipient = payTo ===
'seller' ?
selectedTransaksi.penjual :
selectedTransaksi.pembeli
    const recipientWallet = payTo
=== 'seller' ?
selectedTransaksi.penjual?.wallet
Address :
selectedTransaksi.pembeli?.wallet
Address
    if (!recipientWallet) throw
new Error(`Alamat wallet ${payTo
=== 'seller' ? 'penjual' :
'pembeli'} tidak ditemukan`)
    if (!/^0x[a-fA-F0-
9]{40}$/.test(recipientWallet))
throw new Error(`Format alamat
wallet ${payTo === 'seller' ?
'penjual' : 'pembeli'} tidak
valid`)
    try {
      setPaymentLoading(true)
      const reason = `Admin
menyelesaikan sengketa untuk
transaksi
${selectedTransaksi.kodeTransaksi
}`const response = payTo ===
'seller'? await
adminAPI.resolveDisputeToSeller(s
electedTransaksi.id, reason):
awaitadminAPI.resolveDisputeToBuy
er(selectedTransaksi.id, reason)
if (response.sukses) {
setShowDisputeModal(false)
setSelectedTransaksi(null)
        await fetchTransaksi()
        setNotification({
          type: 'success',
          message: `Sengketa
berhasil diselesaikan! Dana
${selectedTransaksi.escrowAmount}
ETH telah dikirim ke
${recipient.nama}`
        })return {
          success: true,
          transactionHash:
response.transactionHash ||
response.txHash || null,
          recipient:
recipient.nama, amount:
selectedTransaksi.escrowAmount
        }} else {
        throw new
Error(response.pesan || 'Failed
to resolve dispute') }
} catch (error) {let errorMessage
= 'Penyelesaian sengketa gagal. '
```

```
if (error.name === 'TypeError' &&
error.message?.includes('fetch'))
{
        errorMessage += 'Periksa
koneksi internet Anda.'
      } else if (typeof error ===
'object' && error.pesan) {
        errorMessage +=
error.pesan
      } else if (error.message) {
        errorMessage +=
error.message
      } else {errorMessage +=
'Terjadi kesalahan yang tidak
diketahui'}throw new
Error(errorMessage) } finally {
      setPaymentLoading(false)
    }
  }  if (loading) {return (
      <div className="min-h-
screen bg-gray-50 flex items-
center justify-center">
        <div className="animate-
spin rounded-full h-32 w-32
border-b-2 border-blue-
600"></div></div>)}
  return (
    <div className="min-h-screen
bg-gray-50 p-6">
      <div className="max-w-[90%]
mx-auto"><div className="flex
gap-6"><div className="flex-
shrink-0"><AdminNavigation />
</div><div className="flex-
1"><div className="bg-white
rounded-lg shadow-sm mb-6"><div
className="px-6 py-4 border-b
border-gray-200"><h1
className="text-2xl font-bold
text-gray-900">Manajemen
Pembayaran & Sengketa</h1><p
className="text-gray-600 mt-
1">Kelola pembayaran escrow dan
sengketa untuk transaksi yang
sudah dibayar pembeli</p></div>
</div><div className="bg-white
rounded-lg shadow-sm border
border-gray-100"><div
className="px-6 py-4 border-b
border-gray-200"><h2
className="text-lg font-semibold
text-gray-900">{filter ===
'SENGKETA' ? 'Daftar Sengketa &
Transaksi Sengketa' : 'Daftar
Transaksi
Pembayaran'}</h2>div><div
className="overflow-x-
auto"><table className="min-w-
full divide-y divide-gray-
200"><thead className="bg-gray-
50"><tr><th className="px-6 py-3
text-left text-xs font-medium
text-gray-500 uppercase tracking-
wider">Produk</th><th
className="px-6 py-3 text-left
text-xs font-medium text-gray-500
uppercase tracking-
wider">Transaksi</th><th
className="px-6 py-3 text-left
text-xs font-medium text-gray-500
uppercase tracking-
wider">Harga</th><th
className="px-6 py-3 text-left
text-xs font-medium text-gray-500
uppercase tracking-
wider">Penjual</th><th
className="px-6 py-3 text-left
text-xs font-medium text-gray-500
uppercase tracking-
wider">Pembeli</th><th
className="px-6 py-3 text-left
text-xs font-medium text-gray-500
uppercase tracking-
wider">Status</th><th
className="px-6 py-3 text-left
text-xs font-medium text-gray-500
uppercase tracking-
wider">Tanggal</th><th
className="px-6 py-3 text-left
text-xs font-medium text-gray-500
uppercase tracking-
wider">Aksi</th></tr></thead><tbo
dy className="bg-white divide-y
divide-gray-
200">{!Array.isArray(displayData)
|| displayData.length === 0 ?
(<tr><td colSpan="8"
className="px-6 py-12 text-
center"><div className="text-
gray-500"><h3 className="mt-2
text-sm font-medium text-gray-
900">{filter === 'SENGKETA' ?
'Tidak ada sengketa' : 'Tidak ada
transaksi'}</h3><p className="mt-
1 text-sm text-gray-500">{filter
=== 'SENGKETA' ? 'Belum ada
sengketa yang perlu ditinjau.' :
'Belum ada transaksi dengan
escrow yang perlu dikelola
admin.'}</p></div></td></tr>
                    ) :
(Array.isArray(displayData)
&&displayData.map((item, index)
=> {const isSengketaItem =
item.isSengketaItem || falseconst
sengketaData =
item.sengketaDataconst uniqueKey
= isSengketaItem && sengketaData?
`sengketa-${sengketaData.id ||
sengketaData.transaksiId ||
item.id}-${index}`: `transaksi-
${item.id || item.kodeTransaksi}-
${index}`return (<tr
key={uniqueKey}
className="hover:bg-
gray50"><tdclassName="px-6 py-2
```

```
whitespace-nowrap"><div
className="flex items-
center"><imgsrc={item.produk?.gam
bar || '/placeholder-
game.jpg'}alt={item.produk?.judul
Produk || 'Produk'}className="w-
10 h-10 rounded-lg object-cover
mr-3"/><div><div className="text-
sm font-medium text-gray-
900">{item.produk?.judulProduk ||
'-'}</div><div className="text-xs
text-gray-
500">{item.produk?.namaGame || '-
'}</div></div></div></td><td
className="px-6 py-2 whitespace-
nowrap"><div className="text-sm
text-gray-
900">{item.kodeTransaksi || '-
'}</div>{item.escrowId && <div
className="text-xs text-gray-
500">Escrow:
{item.escrowId}</div>}</td><td
className="px-6 py-2 whitespace-
nowrap"><div className="text-sm
font-medium text-gray-
900">{item.escrowAmount ?
`${parseFloat(item.escrowAmount).
toFixed(4)} ETH` : '-
'}</div>{item.produk?.harga &&
<div className="text-xs text-
gray-500">≈ Rp
{item.produk.harga.toLocaleString
()}</div>}</td><td className="px-
6 py-2 whitespace-nowrap"><div
className="text-sm text-gray-
900">{item.penjual?.nama || '-
'}</div>{item.penjual?.walletAddr
ess ? (<div className="text-xs
text-gray-500 font-
mono">{item.penjual.walletAddress
.substring(0,
6)}...{item.penjual.walletAddress
.substring(item.penjual.walletAdd
ress.length - 4)} </div>) : (<div
className="text-xs text-gray-
400">-</div>
</td> <td className="px-6 py-2
whitespace-nowrap"><div
className="text-sm text-gray-
900">{item.pembeli?.nama || '-
'}</div>{item.pembeli?.walletAddr
ess ? (<div className="text-xs
text-gray-500 font-
mono">{item.pembeli.walletAddress
.substring(0,
6)}...{item.pembeli.walletAddress
.substring(item.pembeli.walletAdd
ress.length - 4)} </div>) : (<div
className="text-xs text-gray-
400">-</div>)}</td><td
className="px-6 py-2 whitespace-
nowrap">{isSengketaItem &&
sengketaData ?
getStatusBadge(sengketaData.statu
s) : getStatusBadge(item.status)}
</td><td className="px-6 py-2
whitespace-nowrap"><div
className="text-sm text-gray-
500">{isSengketaItem &&
sengketaData? new
Date(sengketaData.dibuatPada).toL
ocaleDateString('id-ID'): new
Date(item.dibuatPada).toLocaleDat
eString('id-ID')} </div></td><td
className="px-6 py-2 whitespace-
nowrap"><buttononClick={() =>
handleDetailClick(item)}
className={`inline-flex items-
center px-3 py-1 border shadow-sm
text-xs leading-4 font-medium
rounded-md text-white
${isSengketaItem && sengketaData?
sengketaData.status ===
'DIPROSES'? 'bg-red-600 hover:bg-
red-700 border-red-600': 'bg-
gray-600 hover:bg-gray-700
border-gray-600': item.status ===
'DIKONFIRMASI_PEMBELI'? 'bg-blue-
600 hover:bg-blue-700 border-
blue-600': item.status ===
'SENGKETA'? 'bg-red-600 hover:bg-
red-700 border-red-600': 'bg-
gray-600 hover:bg-gray-700
border-gray-600'}`}
{isSengketaItem && sengketaData?
sengketaData.status ===
'DIPROSES' ? 'Review' : 'Detail':
item.status ===
'DIKONFIRMASI_PEMBELI'? 'Bayar':
item.status === 'SENGKETA'?
'Sengketa':
'Detail'}</button></td></tr>
                        })
                      )}
                    </tbody>
                  </table>
                </div>
              </div>
            </div>
          </div>
        </div>
{showPaymentModal &&
selectedTransaksi && (<div
className="fixed inset-0 bg-black
bg-opacity-50 flex items-center
justify-center z-50 p-4"><div
className="bg-white rounded-lg
max-w-2xl w-full max-h-[90vh]
overflow-y-auto"><div
className="p-6"><div
className="flex justify-between
items-center mb-6"><h3
className="text-xl font-semibold
text-gray-900">
{selectedTransaksi.status ===
'SENGKETA'Penyelesaian Sengketa -
```

```
${selectedTransaksi.kodeTransaksi
}`: `Pembayaran ke Penjual -
${selectedTransaksi.kodeTransaksi
}`}</h3><button onClick={() =>
setShowPaymentModal(false)}
className="text-gray-600 px-2 py-
1">
</button></div><div
className="space-y-4"><div
className="bg-gray-50 border
border-gray-200 rounded-lg p-4">
<h4 className="font-medium text-
gray-900 mb-2">Informasi
Transaksi</h4><div
className="space-y-2 text-
sm"><p><span className="font-
medium">Produk:</span>
{selectedTransaksi.produk?.judulP
roduk}</p><p><span
className="font-
medium">Game:</span>
{selectedTransaksi.produk?.namaGa
me}</p><p><span className="font-
medium">Harga:</span> Rp
{selectedTransaksi.produk?.harga?
.toLocaleString()}</p><p><span
className="font-
medium">Status:</span>
{selectedTransaksi.status}</p>
                  </div>
                </div>
                <div
className="bg-gray-50 border
border-gray-200 rounded-lg p-4">
<h4 className="font-medium text-
gray-900 mb-2">Penerima
Pembayaran (Penjual)</h4><div
className="space-y-2 text-sm">
<p><span className="font-
medium">Nama:</span>
{selectedTransaksi.penjual?.nama}
</p><p><span className="font-
medium">Email:</span>
{selectedTransaksi.penjual?.email
}</p><p><span className="font-
medium">Wallet:</span>{selectedTr
ansaksi.penjual?.walletAddress ?
(<span className="text-xs ml-1
font-mono bg-white px-2 py-1
rounded
border">{selectedTransaksi.penjua
l.walletAddress}</span>) : (<span
className="text-red-600 ml-
1">Tidak ada wallet
address</span>)}</p>{!selectedTra
nsaksi.penjual?.walletAddress &&
(<div className="bg-red-100 p-2
rounded mt-2">p className="text-
red-800 text-xs"><span
className="font-
medium">MASALAH:</span> Penjual
belum menghubungkan wallet
address. Pembayaran tidak dapat
diproses tanpa wallet address
penjual. </p></div>)}</div>
                </div>
                <div
className="bg-gray-50 border
border-gray-200 rounded-lg p-
4"><h4 className="font-medium
text-gray-900 mb-2">Informasi
Escrow & Pembayaran</h4><div
className="space-y-2 text-
sm">{selectedTransaksi.escrowId ?
(<><p><span className="font-
medium">Escrow ID:</span>
{selectedTransaksi.escrowId}</p>{
selectedTransaksi.escrowAmount &&
(<p><span className="font-
medium">Jumlah Dana:</span>
{parseFloat(selectedTransaksi.esc
rowAmount).toFixed(4)}
ETH</p>)}{selectedTransaksi.smart
ContractTxHash && (<p><span
className="font-medium">TX Hash
Deposit:</span><ahref={`https://s
epolia.etherscan.io/tx/${selected
Transaksi.smartContractTxHash}`}t
arget="_blank"rel="noopener
noreferrer"className="text-gray-
600 hover:text-gray-800 ml-1
break-all text-
xs">{selectedTransaksi.smartContr
actTxHash.substring(0, 20)}...
 </p>)}<div className="bg-white
border border-gray-300 p-2
rounded mt-2"><p className="text-
gray-800 text-xs"><strong>Dana
Siap Dibayar:</strong> Dana
{parseFloat(selectedTransaksi.esc
rowAmount || 0).toFixed(4)} ETH
sudah terkunci di smart contract
dan siap untuk dibayarkan ke
penjual. </p></div>
) : (<div className="bg-red-100
p-3 rounded"><p className="text-
red-800 text-sm"><span
className="font-
medium">PERINGATAN:</span></p>
                        <p
className="text-red-700 text-xs
mt-1">Transaksi ini tidak
memiliki Escrow ID. Dana tidak
dikelola oleh smart contract dan
tidak dapat dibayar secara
otomatis.</p></div>)}</div></div>
{selectedTransaksi.status ===
'DIKONFIRMASI_PEMBELI' ? (<div
className="bg-gray-50 border
border-gray-200 rounded-lg p-
4"><h4 className="font-medium
text-gray-900 mb-2">Status
Konfirmasi</h4><div
className="space-y-2 text-
sm"><p><span className="font-
medium">Pembeli:</span>
```

```
{selectedTransaksi.pembeli?.nama}
</p><p className="text-gray-
700"><span className="font-
medium">Konfirmasi:</span>
Pembeli telah mengkonfirmasi
penerimaan akun</p><p
className="text-gray-700"><span
className="font-
medium">Menunggu:</span> Admin
melepas dana ke penjual</p>
                    </div>
                  </div>
) : selectedTransaksi.status ===
'SENGKETA' ? (<div className="bg-
gray-50 border border-gray-200
rounded-lg p-4"><h4
className="font-medium text-gray-
900 mb-2">Status Sengketa</h4>
<div className="space-y-2 text-
sm"><p><span className="font-
medium">Pembeli:</span>
{selectedTransaksi.pembeli?.nama}
</p><p><span className="font-
medium">Penjual:</span>
{selectedTransaksi.penjual?.nama}
</p><p className="text-gray-
700"><span className="font-
medium">Status:</span> Transaksi
dalam sengketa</p><p
className="text-gray-700"><span
className="font-
medium">Menunggu:</span> Admin
menyelesaikan sengketa</p><div
className="bg-white border
border-gray-300 p-2 rounded mt-
2"><p className="text-gray-800
text-xs"><span className="font-
medium">Tindakan:</span> Pilih
pihak yang menang untuk
menyelesaikan sengketa dan
melepas dana.</p></div>
                    </div>
                  </div>
                ) : (
                <div
className="bg-gray-50 border
border-gray-200 rounded-lg p-4">
<h4 className="font-medium text-
gray-900 mb-2">Status
Transaksi</h4><div
className="space-y-2 text-
sm"><p><span className="font-
medium">Pembeli:</span>
{selectedTransaksi.pembeli?.nama}
</p><p><span className="font-
medium">Penjual:</span>
{selectedTransaksi.penjual?.nama}
</p><p><span className="font-
medium">Status:</span>
{selectedTransaksi.status}</p>
                    </div>
                  </div>
                )}</div>

{selectedTransaksi.status ===
'DIKONFIRMASI_PEMBELI' &&
selectedTransaksi.escrowId &&
(<div className="mt-6 pt-6
border-t border-gray-200"><h4
className="font-medium text-gray-
900 mb-4">Konfirmasi
Pembayaran</h4>{selectedTransaksi
.penjual?.walletAddress ? (<>
<buttononClick={handlePayment}
disabled={paymentLoading}className
e="w-full bg-green-600 text-white
py-3 px-4 rounded-lg hover:bg-
green-700 disabled:opacity-50
font-medium">{paymentLoading ?
'Memproses Pembayaran...' :
`Bayar
${parseFloat(selectedTransaksi.es
crowAmount || 0).toFixed(4)} ETH
ke Penjual`}
                  </button>
                <p
className="text-sm text-gray-500
mt-2 text-center">Dana akan
dikirim dari smart contract ke
wallet penjual menggunakan sistem
baru</p>
            </>
          ) : (
         <div className="bg-red-
50 border border-red-200 rounded-
lg p-4"><h5 className="font-
medium text-red-800 mb-
2">Pembayaran Tidak Dapat
Diproses</h5><p className="text-
red-700 text-sm mb-3">Penjual
belum menghubungkan wallet
address ke akun mereka.
Pembayaran tidak dapat dilakukan
tanpa alamat wallet
penjual.</p><p className="text-
red-600 text-xs"><span
className="font-
medium">Solusi:</span> Minta
penjual untuk <i>login</i> dan
menghubungkan wallet address di
halaman profil mereka.</p>
                    </div>
                  )}
                </div>
              )}
{selectedTransaksi.status ===
'SENGKETA' &&
selectedTransaksi.escrowId &&
(<div className="mt-6 pt-6
border-t border-gray-200"> <h4
className="font-medium text-gray-
900 mb-4">Penyelesaian
Sengketa</h4>div className="bg-
yellow-50 border border-yellow-
200 rounded-lg p-4 mb-4">
<h5 className="font-medium text-
yellow-800 mb-2">Perhatian</h5>
```

```
<p className="text-yellow-700 text-sm">Anda akan menyelesaikan sengketa ini dengan memilih pihak yang menang. Dana akan dikirim ke pihak yang dipilih dan sengketa akan ditutup secara permanen.</p></div>{!selectedTransaksi.penjual?.walletAddress || !selectedTransaksi.pembeli?.walletAddress ? (<div className="bg-red-50 border border-red-200 rounded-lg p-4"><h5 className="font-medium text-red-800 mb-2">Tidak Dapat Menyelesaikan Sengketa</h5><div className="text-red-700 text-sm space-y-1">{!selectedTransaksi.penjual?.walletAddress && <p>- Penjual belum menghubungkan wallet address</p>}{!selectedTransaksi.pembeli?.walletAddress && <p>- Pembeli belum menghubungkan wallet address</p>}</div><p className="text-red-600 text-xs mt-2"><span className="font-medium">Solusi:</span> Minta kedua pihak untuk menghubungkan wallet address di profil mereka.</p>
                  </div>
                ) : (
                  <div
className="grid grid-cols-1 md:grid-cols-2 gap-4"><div
className="bg-gray-50 border border-gray-200 rounded-lg p-4">
<h5 className="font-medium text-gray-800 mb-2">Menangkan Penjual</h5><p className="text-gray-700 text-sm mb-3">Dana {parseFloat(selectedTransaksi.escrowAmount || 0).toFixed(4)} ETH akan dikirim ke penjual: {selectedTransaksi.penjual?.nama}</p><buttononClick={() => handleDisputePayment('seller')} disabled={paymentLoading}className="w-full bg-green-600 text-white py-2 px-4 rounded-lg hover:bg-green-700 disabled:opacity-50 font-medium text-sm">{paymentLoading ? 'Memproses...' : 'Bayar ke Penjual'}</button></div><div
className="bg-gray-50 border border-gray-200 rounded-lg p-4"><h5 className="font-medium text-gray-800 mb-2">Menangkan Pembeli</h5><p className="text-gray-700 text-sm mb-3">Dana {parseFloat(selectedTransaksi.escrowAmount || 0).toFixed(4)} ETH akan dikembalikan ke pembeli: {selectedTransaksi.pembeli?.nama}</p><buttononClick={() => handleDisputePayment('buyer')} disabled={paymentLoading}className="w-full bg-gray-600 text-white py-2 px-4 rounded-lg hover:bg-gray-700 disabled:opacity-50 font-medium text-sm">{paymentLoading ? 'Memproses...' : 'Refund ke Pembeli'}</button></div>
                  </div>
                )} <p
className="text-sm text-gray-500 mt-4 text-center">Penyelesaian sengketa menggunakan smart contract dan tidak dapat dibatalkan</p>
                </div>
              )}
            </div>
          </div>
        </div>
      )}
      {notification && (<div
className="fixed top-4 right-4 z-50"><div className={`max-w-sm w-full shadow-lg rounded-lg pointer-events-auto ${notification.type === 'success' ? 'bg-green-50 border border-green-200' : 'bg-red-50 border border-red-200'}`}><div
className="p-4"><div
className="flex items-start"><div
className="ml-1 w-0 flex-1"><p
className={`text-sm font-medium ${notification.type === 'success' ? 'text-green-800' : 'text-red-800'}`}>{notification.message}</p></div><div className="ml-4 flex-shrink-0 flex"><button
onClick={() => setNotification(null)}
className={`${notification.type === 'success' ? 'text-green-600 hover:text-green-700' : 'text-red-600 hover:text-red-700'} px-2`}>
                    ×
                  </button>
                </div>
              </div>
            </div>
          </div>
        </div>
      )}
      {showReasonModal && (
        <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50 p-4">
```

```
<div className="bg-white rounded-
lg max-w-md w-full"><div
className="p-6"><h3
className="text-lg font-semibold
text-gray-900 mb-4">Alasan
Pembayaran</h3><p
className="text-sm text-gray-600
mb-4">Anda akan mengirim dana
{selectedTransaksi?.escrowAmount}
ETH ke penjual:
{selectedTransaksi?.penjual?.nama
}</p><textareaplaceholder="Masukk
an alasan pembayaran (minimal 10
karakter)" className="w-full p-3
border border-gray-300 rounded-lg
resize-none h-24 focus:ring-2
focus:ring-blue-500 focus:border-
transparent"value={paymentReason}
                onChange={e =>
setPaymentReason(e.target.value)}
              />
              <div
className="flex gap-3 mt-4">
                <button
                  onClick={() =>
{setShowReasonModal(false)
setPaymentReason('')
}}className="flex-1 px-4 py-2
text-gray-700 bg-gray-100
rounded-lg hover:bg-gray-
200">Batal</button>
                <button
                  onClick={() =>
processPayment(paymentReason)}
className="flex-1 px-4 py-2 text-
white bg-blue-600 rounded-lg
hover:bg-blue-700"
                >
                  Lanjutkan
                </button>
              </div>
            </div>
          </div>
        </div>
      )}
      {showConfirmModal &&
selectedTransaksi && (
        <div className="fixed
inset-0 bg-black bg-opacity-50
flex items-center justify-center
z-50 p-4"><div className="bg-
white rounded-lg max-w-md w-
full"><div className="p-6"><h3
className="text-lg font-semibold
text-gray-900 mb-4">Konfirmasi
Pembayaran</h3><div
className="space-y-3 text-sm">
                <div><span
className="font-medium">Escrow
ID:</span>
{selectedTransaksi.escrowId}</div
><div><span className="font-
medium">Jumlah:</span>
{selectedTransaksi.escrowAmount}
ETH</div><div><span
className="font-
medium">Penjual:</span>
{selectedTransaksi.penjual?.nama}
</div><div><span className="font-
medium">Wallet:</span> <span
className="font-mono text-
xs">{selectedTransaksi.penjual?.w
alletAddress}</span></div><div><s
pan className="font-
medium">Alasan:</span>
{paymentReason}</div>
              </div>
              <div
className="flex gap-3 mt-
6"><button onClick={() =>
setShowConfirmModal(false)}
className="flex-1 px-4 py-2 text-
gray-700 bg-gray-100 rounded-lg
hover:bg-gray-200">Batal
</button><button
onClick={confirmPayment}
disabled={paymentLoading}
className="flex-1 px-4 py-2 text-
white bg-green-600 rounded-lg
hover:bg-green-700
disabled:opacity-50">
                  {paymentLoading
? 'Memproses...' : 'Bayar
Sekarang'}</button>
              </div>
            </div>
          </div>
        </div>
      )}
      <ModalDetailSengketaAdmin
        isOpen={showDisputeModal}
        onClose={() => {
setShowDisputeModal(false)
setSelectedTransaksi(null)
}}transaksi={selectedTransaksi}on
Resolve={handleDisputeResolve}/>
    </div>
  )
}
export default PembayaranAdmin
```

Api.js

```
import axios from 'axios';
const API_BASE_URL =
process.env.REACT_APP_API_URL ||
'http://localhost:5000/api';
const api = axios.create({
  baseURL: API_BASE_URL,
  headers: { 'Content-Type':
'application/json' },
  timeout: 30000
});
api.interceptors.request.use(
  (config) => {const token =
localStorage.getItem('authToken')
;
```

```
    if (token)
config.headers.Authorization =
`Bearer ${token}`;
    config.metadata = {
startTime: Date.now() };
    return config;
  },
  (error) =>
Promise.reject(error)
);
api.interceptors.response.use(
  (response) => {
    const start =
response?.config?.metadata?.start
Time;
    if (start) void (Date.now() -
start); // keep timing guard, no
logging
    return response;
  },
  async (error) => {
    const originalRequest =
error?.config || {};

    // Timeout with retry
    if (error?.code ===
'ECONNABORTED' &&
!originalRequest._retry) {
      originalRequest._retry =
true;
      originalRequest.timeout =
45000;
      try {
        const retryResponse =
await api(originalRequest);
        return retryResponse;
      } catch (retryError) {
        return Promise.reject({
          ...retryError,
          userMessage: 'Koneksi
timeout. Server mungkin sedang
sibuk, silakan coba lagi.'
        });
      }
    }
    if (error?.message ===
'Network Error' || error?.code
=== 'ERR_NETWORK') {
      return Promise.reject({
        ...error,
        userMessage: 'Koneksi ke
server bermasalah. Pastikan
server berjalan dan coba lagi.'
      });
    }
    if (error?.response?.status
=== 401)
{localStorage.removeItem('authTok
en');
localStorage.removeItem('user');
      if (typeof window !==
'undefined') window.location.href
= '/masuk';
      return Promise.reject({
        ...error,
        userMessage: 'Sesi Anda
telah berakhir. Silakan login
kembali.'
      });
    }
    if (error?.response?.status
>= 500) {
      return Promise.reject({
        ...error,
        userMessage: 'Server
sedang bermasalah. Silakan coba
lagi dalam beberapa saat.'
      });
    }message
    if (error?.response?.status
=== 408 || (typeof error?.message
=== 'string' &&
error.message.toLowerCase().inclu
des('timeout'))) {
      return Promise.reject({
        ...error,
        userMessage: 'Operasi
memakan waktu lebih lama dari
biasanya. Silakan periksa status
transaksi di Etherscan.'
      });
    }if
(error?.response?.data?.pesan ||
error?.response?.data?.message) {
      return Promise.reject({
        ...error,
        userMessage:
error.response.data.pesan ||
error.response.data.message
      });
    }return
Promise.reject(error); }
);
const apiWithRetry = async
(apiCall, maxRetries = 2) => {
  let lastError;
  for (let attempt = 1; attempt
<= maxRetries; attempt++) {
    try {const result = await
apiCall();
      return result;
    } catch (error) {
      lastError = error;
      if (error?.response?.status
>= 400 && error?.response?.status
< 500 && error?.response?.status
!== 408) {throw error;
      }if (attempt ===
maxRetries) throw error;
      const delay = Math.pow(2,
attempt) * 1000;
      await new Promise((r) =>
setTimeout(r, delay)); }  }
  throw lastError;};
export const authAPI = {login:
(credentials) =>
```

```
api.post('/auth/masuk',
credentials),
  register: (userData) =>
api.post('/auth/daftar',
userData),
  logout: () =>
api.post('/auth/keluar'),
  getProfile: () =>
api.get('/auth/profil'),
  updateProfile: (profileData) =>
api.put('/auth/profil',
profileData),
  validateToken: () =>
api.get('/auth/validasi'),
  checkWalletAvailability:
(alamatWallet) =>
    api.get('/auth/cek-wallet', {
params: { alamatWallet },
timeout: 10000 })
};
export const produkAPI = {
  getAllProduk: (params = {}) =>
api.get('/produk', { params }),
  getProdukById: (id) =>
api.get(`/produk/${id}`),
  getProdukPenjual: () =>
api.get('/produk/saya/daftar'),
  createProduk: (produkData) =>
api.post('/produk', produkData),
  updateProduk: (id, produkData)
=> api.put(`/produk/${id}`,
produkData),
  deleteProduk: (id) =>
api.delete(`/produk/${id}`),
  searchProduk: (query) =>
api.get(`/produk/search?q=${encod
eURIComponent(query)}`)
};

// Transaksi API
export const transaksiAPI = {
  buatTransaksi: (produkId) =>
apiWithRetry(() =>
api.post('/transaksi', { produkId
})),
  getTransaksiUser: (params = {})
=> api.get('/transaksi', { params
}),
  getTransaksiSaya: (params = {})
=> api.get('/transaksi/saya', {
params }),
  getTransaksiPembeli: (params =
{}) => api.get('/transaksi', {
params: { ...params, role:
'pembeli' } }),
  getTransaksiPenjual: (params =
{}) => api.get('/transaksi', {
params: { ...params, role:
'penjual' } }),
  getDetailTransaksi: (id) =>
api.get(`/transaksi/${id}`),
  konfirmasiPembayaran: (id,
data) => apiWithRetry(() =>
api.put(`/transaksi/${id}/konfirm
asi-pembayaran`, data), 3),
  kirimAkun: (id, data) =>
api.put(`/transaksi/${id}/kirim-
akun`, data),
  konfirmasiPenerimaan: (id) =>
api.put(`/transaksi/${id}/konfirm
asi-penerimaan`),
  buatSengketa: (id, data) =>
api.post(`/transaksi/${id}/sengke
ta`, data),
  getDetailSengketa:
(transaksiId) =>
api.get(`/transaksi/${transaksiId
}/sengketa`),
  checkSengketa: (transaksiId) =>
api.get(`/transaksi/${transaksiId
}/sengketa/check`),
  buatPembelaan: (sengketaId,
data) =>
api.post(`/transaksi/sengketa/${s
engketaId}/pembelaan`, data),
  getStatistikTransaksi: () =>
api.get('/transaksi/statistik'),
  getRiwayatTransaksi: (params =
{}) =>
api.get('/transaksi/riwayat', {
params }),
  updateStatus: (id, data) =>
api.put(`/transaksi/${id}/status`
, data)
};

// Admin API
export const adminAPI = {
  getDashboardStats: () =>
api.get('/admin/dashboard'),
  getAllUsers: (params = {}) =>
api.get('/admin/users', { params
}),
  getUserStats: (userId) =>
api.get(`/admin/users/${userId}/s
tats`),
  updateUserStatus: (userId,
status) =>
api.put(`/admin/users/${userId}/s
tatus`, { status }),
  updateUserRole: (userId, role)
=>
api.put(`/admin/users/${userId}/r
ole`, { role }),
  deleteUser: (userId) =>
api.delete(`/admin/users/${userId
}`),
  getAllTransaksi: (params = {})
=> api.get('/admin/transaksi', {
params }),
  getAllSengketa: (params = {})
=> api.get('/admin/sengketa', {
params }),
  resolveSengketa: (id, data) =>
api.put(`/admin/sengketa/${id}/re
solve`, data),
```

```
  getSengketaDetail: (id) =>
api.get(`/admin/sengketa/${id}`),
  getAllProdukAdmin: (params =
{}) => api.get('/admin/produk', {
params }),
  deleteProduk: (id) =>
api.delete(`/admin/produk/${id}`)
,
  getKonfigurasi: (kunci) =>
api.get(`/admin/konfigurasi/${kun
ci}`),
  payToSeller: (id, data) =>
    apiWithRetry(() =>
api.post(`/admin-
payment/payment/${id}`, { reason:
data.note || 'Admin processing
payment' }, { timeout: 90000 }),
2)
};

// Upload API
export const uploadAPI = {
  uploadImage: (file) => {
    const formData = new
FormData();
    formData.append('gambar',
file);
    return
api.post('/upload/single',
formData, {
      headers: { 'Content-Type':
'multipart/form-data' },
      timeout: 60000
    });
  },
  uploadMultipleImages: (files)
=> {
    const formData = new
FormData();
    files.forEach((file) =>
formData.append('gambar', file));
    return
api.post('/upload/multiple',
formData, {
      headers: { 'Content-Type':
'multipart/form-data' },
      timeout: 120000
    });
  },
  uploadBukti: (file) => {
    const formData = new
FormData();
    formData.append('gambar',
file);
    return
api.post('/upload/single',
formData, {
      headers: { 'Content-Type':
'multipart/form-data' },
      timeout: 60000
    });},deleteImage: (url) =>
api.delete('/upload/cloudinary',
{ data: { url } })
};
export const notificationAPI = {
  getNotifications: () =>
api.get('/notifications'),
  markAsRead: (id) =>
api.put(`/notifications/${id}/rea
d`),
  markAllAsRead: () =>
api.put('/notifications/read-
all'),
  deleteNotification: (id) =>
api.delete(`/notifications/${id}`
)
};
export const smartContractAPI = {
  prepareEscrow: (data) =>
apiWithRetry(() =>
api.post('/escrow/prepare', data,
{ timeout: 60000 }), 2),
  createEscrow: (data) =>
apiWithRetry(() =>
api.post('/escrow/create', data,
{ timeout: 120000 }), 3),
  verifyEscrow: (data) =>
apiWithRetry(() =>
api.post('/escrow/verify', data,
{ timeout: 60000 }), 2),
  confirmReceived: (escrowId,
buyerAddress) =>
api.post('/escrow/confirm', {
escrowId, buyerAddress }),
  createDispute: (escrowId, data)
=> api.post('/escrow/dispute', {
escrowId, ...data }),
  confirmReceivedCallback: (data)
=> apiWithRetry(() =>
api.post('/escrow/confirm-
callback', data), 2),
  createDisputeCallback: (data)
=> apiWithRetry(() =>
api.post('/escrow/dispute-
callback', data), 2),
  resolveDispute: (escrowId,
data) => apiWithRetry(() =>
api.post('/escrow/resolve-
dispute', { escrowId, ...data },
{ timeout: 90000 }), 2),
  getEscrowStatus: (escrowId) =>
api.get(`/escrow/${escrowId}`),
  getContractStats: () =>
api.get('/escrow/stats'),
  healthCheck: () =>
api.get('/escrow/health')
};
export const apiUtils = {
  getErrorMessage: (error) => {
    if (error?.userMessage)
return error.userMessage;
    if
(error?.response?.data?.pesan)
return error.response.data.pesan;
    if
(error?.response?.data?.message)
```

```
return
error.response.data.message;
    if (error?.code ===
'ECONNABORTED') return 'Koneksi
timeout. Silakan coba lagi.';
    if (error?.code ===
'ERR_NETWORK') return 'Koneksi ke
server bermasalah. Pastikan
server berjalan.';
    if (error?.response?.status
=== 500) return 'Server sedang
bermasalah. Silakan coba lagi
dalam beberapa saat.';
    if (error?.response?.status
=== 404) return 'Data yang
diminta tidak ditemukan.';
    if (error?.response?.status
=== 403) return 'Anda tidak
memiliki akses untuk melakukan
operasi ini.';
    if (error?.message) return
error.message;
    return 'Terjadi kesalahan
yang tidak diketahui';
  },
  isAuthenticated: () =>
!!localStorage.getItem('authToken
'),
  getCurrentUser: () => {
    try {
      const user =
localStorage.getItem('user');
      if (!user) return null;
      return JSON.parse(user);
    } catch {

localStorage.removeItem('user');
      return null;
    }
  },
  setAuthData: (token, user) => {

localStorage.setItem('authToken',
token);
    localStorage.setItem('user',
JSON.stringify(user));
  },
  clearAuthData: () => {

localStorage.removeItem('authToke
n');

localStorage.removeItem('user');
  },
  isRetryableError: (error) =>
    error?.code ===
'ECONNABORTED' ||
    error?.code === 'ERR_NETWORK'
||
    error?.response?.status >=
500 ||
    error?.response?.status ===
408,
  formatResponseTime: (ms) => (ms
< 1000 ? `${ms}ms` : `${(ms /
1000).toFixed(1)}s`)
};
export const apiService = {
  auth: authAPI,
  produk: produkAPI,
  transaksi: transaksiAPI,
  admin: adminAPI,
  upload: uploadAPI,

  notification: notificationAPI,

  smartContract:
smartContractAPI,

  utils: apiUtils

};

export default api;
```