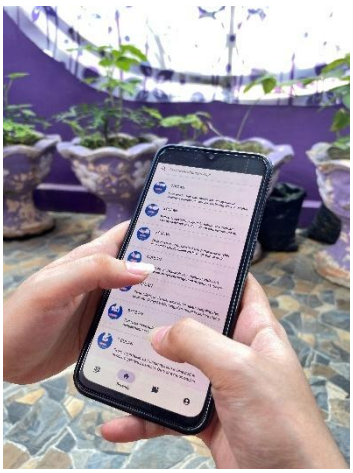
LAMPIRAN

A. Penggunaan Aplikasi











B. Script Program

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('users', function (Blueprint $table)
{
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')-
>nullable();
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('password_reset_tokens', function
(Blueprint $table) {
            $table->string('email')->primary();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });

        Schema::create('sessions', function (Blueprint
$table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()-
>index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('users');
        Schema::dropIfExists('password_reset_tokens');
        Schema::dropIfExists('sessions');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
```

```
        Schema::create('cache', function (Blueprint $table)
{
            $table->string('key')->primary();
            $table->mediumText('value');
            $table->integer('expiration');
        });

        Schema::create('cache_locks', function (Blueprint
$table) {
            $table->string('key')->primary();
            $table->string('owner');
            $table->integer('expiration');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('cache');
        Schema::dropIfExists('cache_locks');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('artikels', function (Blueprint
$table) {
            $table->id();
            $table->string('judul');
            $table->text('isi');
            $table->string('foto')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('artikels');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
```

```
    {
        Schema::create('bayis', function (Blueprint $table)
{
            $table->id();
            $table->string('judul');
            $table->text('isi');
            $table->string('foto')->nullable();
            $table->string('video')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('bayis');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('chats', function (Blueprint $table)
{
            $table->id();
            $table->foreignId('user_one_id')-
>constrained('users')->onDelete('cascade');
            $table->foreignId('user_two_id')-
>constrained('users')->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('chats');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('messages', function (Blueprint
$table) {
            $table->id();
```

```
            $table->foreignId('chat_id')->constrained('chats')->onDelete('cascade');
            $table->foreignId('sender_id')->constrained('users')->onDelete('cascade');
            $table->text('message')->nullable();
            $table->string('attachment')->nullable(); // Path ke file gambar/video
            $table->enum('type', ['text', 'image', 'video'])->default('text');
            $table->boolean('is_read')->default(false);
            $table->string('auto_reply_type')->nullable();

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('messages');
    }
};
```

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        $teams = config('permission.teams');
        $tableNames = config('permission.table_names');
        $columnNames = config('permission.column_names');
        $pivotRole = $columnNames['role_pivot_key'] ?? 'role_id';
        $pivotPermission = $columnNames['permission_pivot_key'] ?? 'permission_id';

        if (empty($tableNames)) {
            throw new \Exception('Error: config/permission.php not loaded. Run [php artisan config:clear] and try again.');
        }
        if ($teams && empty($columnNames['team_foreign_key'] ?? null)) {
            throw new \Exception('Error: team_foreign_key on config/permission.php not loaded. Run [php artisan config:clear] and try again.');
        }

        Schema::create($tableNames['permissions'], static function (Blueprint $table) {
            // $table->engine('InnoDB');
            $table->bigIncrements('id'); // permission id
            $table->string('name');       // For MyISAM use string('name', 225); // (or 166 for InnoDB with Redundant/Compact row format)
            $table->string('guard_name'); // For MyISAM use string('guard_name', 25);
            $table->timestamps();

            $table->unique(['name', 'guard_name']);
        });
```

```
        Schema::create($tableNames['roles'], static function (Blueprint $table) use ($teams, $columnNames) {
            // $table->engine('InnoDB');
            $table->bigIncrements('id'); // role id
            if ($teams || config('permission.testing')) { // permission.testing is a fix for sqlite testing
                $table->unsignedBigInteger($columnNames['team_foreign_key'])->nullable();
                $table->index($columnNames['team_foreign_key'], 'roles_team_foreign_key_index');
            }
            $table->string('name');       // For MyISAM use string('name', 225); // (or 166 for InnoDB with Redundant/Compact row format)
            $table->string('guard_name'); // For MyISAM use string('guard_name', 25);
            $table->timestamps();
            if ($teams || config('permission.testing')) {
                $table->unique([$columnNames['team_foreign_key'], 'name', 'guard_name']);
            } else {
                $table->unique(['name', 'guard_name']);
            }
        });

Schema::create($tableNames['model_has_permissions'], static function (Blueprint $table) use ($tableNames, $columnNames, $pivotPermission, $teams) {
            $table->unsignedBigInteger($pivotPermission);

            $table->string('model_type');
            $table->unsignedBigInteger($columnNames['model_morph_key']);
            $table->index([$columnNames['model_morph_key'], 'model_type'],
'model_has_permissions_model_id_model_type_index');

            $table->foreign($pivotPermission)
                ->references('id') // permission id
                ->on($tableNames['permissions'])
                ->onDelete('cascade');
            if ($teams) {
                $table->unsignedBigInteger($columnNames['team_foreign_key']);
                $table->index($columnNames['team_foreign_key'], 'model_has_permissions_team_foreign_key_index');

                $table->primary([$columnNames['team_foreign_key'], $pivotPermission, $columnNames['model_morph_key'], 'model_type'],
'model_has_permissions_permission_model_type_primary');
            } else {
                $table->primary([$pivotPermission, $columnNames['model_morph_key'], 'model_type'],
'model_has_permissions_permission_model_type_primary');
            }

        });

        Schema::create($tableNames['model_has_roles'], static function (Blueprint $table) use ($tableNames, $columnNames, $pivotRole, $teams) {
            $table->unsignedBigInteger($pivotRole);

            $table->string('model_type');
            $table->unsignedBigInteger($columnNames['model_morph_key']);
```

```
            $table->index([$columnNames['model_morph_key'], 'model_type'], 'model_has_roles_model_id_model_type_index');

            $table->foreign($pivotRole)
                ->references('id') // role id
                ->on($tableNames['roles'])
                ->onDelete('cascade');
            if ($teams) {
                $table->unsignedBigInteger($columnNames['team_foreign_key']);
                $table->index($columnNames['team_foreign_key'], 'model_has_roles_team_foreign_key_index');

                $table->primary([$columnNames['team_foreign_key'], $pivotRole, $columnNames['model_morph_key'], 'model_type'],
                    'model_has_roles_role_model_type_primary');
            } else {
                $table->primary([$pivotRole, $columnNames['model_morph_key'], 'model_type'],
                    'model_has_roles_role_model_type_primary');
            }
        });

        Schema::create($tableNames['role_has_permissions'], static function (Blueprint $table) use ($tableNames, $pivotRole, $pivotPermission) {
            $table->unsignedBigInteger($pivotPermission);
            $table->unsignedBigInteger($pivotRole);

            $table->foreign($pivotPermission)
                ->references('id') // permission id
                ->on($tableNames['permissions'])
                ->onDelete('cascade');

            $table->foreign($pivotRole)
                ->references('id') // role id
                ->on($tableNames['roles'])
                ->onDelete('cascade');

            $table->primary([$pivotPermission, $pivotRole], 'role_has_permissions_permission_id_role_id_primary');
        });

        app('cache')
            ->store(config('permission.cache.store') != 'default' ? config('permission.cache.store') : null)
            ->forget(config('permission.cache.key'));
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        $tableNames = config('permission.table_names');

        if (empty($tableNames)) {
            throw new \Exception('Error: config/permission.php not found and defaults could not be merged. Please publish the package configuration before proceeding, or drop the tables manually.');
        }

        Schema::drop($tableNames['role_has_permissions']);
        Schema::drop($tableNames['model_has_roles']);
        Schema::drop($tableNames['model_has_permissions']);
```

```
        Schema::drop($tableNames['roles']);
        Schema::drop($tableNames['permissions']);
    }
};
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Artikel extends Model
{
    use HasFactory;

    protected $fillable = [
        'judul',
        'isi',
        'foto',
    ];
}
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Factories\HasFactory;

class Bayi extends Model
{
    use HasFactory;

    protected $fillable = [
        'judul',
        'isi',
        'foto',
        'video',
    ];
}
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Chat extends Model
{
    protected $fillable = [
        'user_one_id',
        'user_two_id',
    ];

    public function userOne(): BelongsTo
    {
        return $this->belongsTo(User::class, 'user_one_id');
    }

    public function userTwo(): BelongsTo
    {
        return $this->belongsTo(User::class, 'user_two_id');
```

```
    }

    public function messages(): HasMany
    {
        return $this->hasMany(Messages::class);
    }
}
```

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Messages extends Model
{
    protected $fillable = [
        'chat_id',
        'sender_id',
        'message',
        'attachment',
        'type',
        'is_read',
        'auto_reply_type',
    ];

    public function chat(): BelongsTo
    {
        return $this->belongsTo(Chat::class);
    }

    public function sender(): BelongsTo
    {
        return $this->belongsTo(User::class, 'sender_id');
    }

}
```

```
<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use
Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as
Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    /** @use
HasFactory<\Database\Factories\UserFactory> */
    use HasFactory, Notifiable,HasApiTokens,HasRoles;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
```

```
    * The attributes that should be hidden for
serialization.
    *
    * @var list<string>
    */
   protected $hidden = [
      'password',
      'remember_token',
   ];

   /**
    * Get the attributes that should be cast.
    *
    * @return array<string, string>
    */
   protected function casts(): array
   {
      return [
         'email_verified_at' => 'datetime',
         'password' => 'hashed',
      ];
   }
   public function chatsAsUserOne()
{
   return $this->hasMany(Chat::class, 'user_one_id');
}

public function chatsAsUserTwo()
{
   return $this->hasMany(Chat::class, 'user_two_id');
}

public function messages()
{
   return $this->hasMany(Messages::class, 'sender_id');
}

}

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\Request;
use Illuminate\Http\Response;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;
use Spatie\Permission\Models\Role;

class RegisteredUserController extends Controller
{
   /**
    * Handle an incoming registration request.
    *
    * @throws
\Illuminate\Validation\ValidationException
    */
   public function store(Request $request): Response
   {
      $request->validate([
         'name' => ['required', 'string', 'max:255'],
         'email' => ['required', 'string', 'lowercase', 'email',
'max:255', 'unique:'.User::class],
         'password' => ['required', 'confirmed',
Rules\Password::defaults()],
      ]);
```

```
        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->string('password')),
        ]);
        $Role = Role::findByName('user');
        $user->assignRole($Role);
        event(new Registered($user));

        Auth::login($user);

        return response()->noContent();
    }
}
```

```
<?php

namespace App\Http\Controllers;

use App\Models\Artikel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Storage;

class ArtikelController extends Controller
{
    // Menampilkan semua artikel
    public function index()
    {
        $artikels = Artikel::all()->map(function ($artikel) {
            $artikel->foto_url = $artikel->foto ? asset('storage/' . $artikel->foto) : null;
            return $artikel;
        });
        return response()->json($artikels);
    }

    // Menyimpan artikel baru
    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'judul' => 'required|string|max:255',
            'isi' => 'required|string',
            'foto' => 'nullable|image|mimes:jpg,png,jpeg,gif|max:10240',
        ]);

        if ($validator->fails()) {
            return response()->json(['errors' => $validator->errors()], 422);
        }

        $data = $validator->validated();

        if ($request->hasFile('foto')) {
            $data['foto'] = $request->file('foto')->store('artikels', 'public');
        }

        $artikel = Artikel::create($data);
        $artikel->foto_url = $artikel->foto ? asset('storage/' . $artikel->foto) : null;

        return response()->json(['message' => 'Artikel berhasil dibuat', 'data' => $artikel], 201);
    }

    // Menampilkan detail artikel
    public function show(Artikel $artikel)
    {
```

```
        $artikel->foto_url = $artikel->foto ? asset('storage/' . $artikel->foto) : null;

        return response()->json($artikel);
    }

    // Mengupdate artikel
    public function update(Request $request, Artikel $artikel)
    {
        $validator = Validator::make($request->all(), [
            'judul' => 'sometimes|required|string|max:255',
            'isi' => 'sometimes|required|string',
            'foto' => 'nullable|image|mimes:jpeg,png,jpg,gif|max:10048',
        ]);

        if ($validator->fails()) {
            return response()->json(['errors' => $validator->errors()], 422);
        }

        $data = $validator->validated();

        if ($request->hasFile('foto')) {
            // Hapus foto lama jika ada
            if ($artikel->foto && Storage::disk('public')->exists($artikel->foto)) {
                Storage::disk('public')->delete($artikel->foto);
            }

            $data['foto'] = $request->file('foto')->store('artikels', 'public');
        }

        $artikel->update($data);
        $artikel->refresh();

        $artikel->foto_url = $artikel->foto ? asset('storage/' . $artikel->foto) : null;

        return response()->json(['message' => 'Artikel berhasil diupdate', 'data' => $artikel]);
    }

    // Menghapus artikel
    public function destroy(Artikel $artikel)
    {
        // Hapus foto dari storage
        if ($artikel->foto && Storage::disk('public')->exists($artikel->foto)) {
            Storage::disk('public')->delete($artikel->foto);
        }

        $artikel->delete();

        return response()->json(['message' => 'Artikel berhasil dihapus']);
    }
}

<?php

namespace App\Http\Controllers;

use App\Models\Bayi;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Storage;

class BayiController extends Controller
{
    // Menampilkan semua Bayi
```

```
    public function index()
    {
        return response()->json(Bayi::all());
    }

    // Menyimpan Bayi baru
    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'judul' => 'required|string|max:255',
            'isi' => 'required|string',
            'foto' =>
'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
            'video' =>
'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        if ($validator->fails()) {
            return response()->json(['errors' => $validator-
>errors()], 422);
        }

        $data = $validator->validated();

        if ($request->hasFile('foto')) {
            $path = $request->file('foto')->store('bayis/foto',
'public');
            $data['foto'] = $path;
        }

        if ($request->hasFile('video')) {
            $path = $request->file('video')-
>store('bayis/video', 'public');
            $data['video'] = $path;
        }

        $bayi = Bayi::create($data);
        return response()->json(['message' => 'Bayi
berhasil dibuat', 'data' => $bayi], 201);
    }

    // Menampilkan detail satu Bayi
    public function show(Bayi $bayi)
    {
        return response()->json($bayi);
    }

    // Mengupdate Bayi
    public function update(Request $request, Bayi $bayi)
    {
        $validator = Validator::make($request->all(), [
            'judul' => 'sometimes|required|string|max:255',
            'isi' => 'sometimes|required|string',
            'foto' =>
'nullable|image|mimes:jpeg,png,jpg,gif|max:10048',
            'video' =>
'nullable|mimes:mp4,mov,avi|max:200240',
        ]);

        if ($validator->fails()) {
            return response()->json(['errors' => $validator-
>errors()], 422);
        }

        $data = $validator->validated();

        if ($request->hasFile('foto')) {
            if ($bayi->foto && Storage::disk('public')-
>exists($bayi->foto)) {
                Storage::disk('public')->delete($bayi->foto);
            }
```

```
            $path = $request->file('foto')->store('bayis/foto',
'public');
            $data['foto'] = $path;
        }

        if ($request->hasFile('video')) {
            if ($bayi->video && Storage::disk('public')-
>exists($bayi->video)) {
                Storage::disk('public')->delete($bayi->video);
            }

            $path = $request->file('video')-
>store('bayis/video', 'public');
            $data['video'] = $path;
        }

        $bayi->update($data);
        $bayi->refresh();

        return response()->json(['message' => 'Bayi
berhasil diupdate', 'data' => $bayi]);
    }

    // Menghapus Bayi
    public function destroy(Bayi $bayi)
    {
        if ($bayi->foto && Storage::disk('public')-
>exists($bayi->foto)) {
            Storage::disk('public')->delete($bayi->foto);
        }

        if ($bayi->video && Storage::disk('public')-
>exists($bayi->video)) {
            Storage::disk('public')->delete($bayi->video);
        }

        $bayi->delete();
        return response()->json(['message' => 'Bayi
berhasil dihapus']);
    }
}
```

```
<?php

// app/Http/Controllers/ChatController.php

namespace App\Http\Controllers;

use App\Models\Chat;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class ChatController extends Controller
{
    // Get all chats for logged-in user
    public function index()
    {
        $userId = Auth::id();

        $chats = Chat::where('user_one_id', $userId)
            ->orWhere('user_two_id', $userId)
            ->with(['userOne', 'userTwo', 'messages' =>
function ($query) {
                $query->latest()->first();
            }])
            ->get();

        return response()->json($chats);
    }
```

```
    // Start or get existing chat between two users
    public function store(Request $request)
    {
        $request->validate([
            'receiver_id' => 'required|exists:users,id',
        ]);

        $senderId = Auth::id();
        $receiverId = $request->receiver_id;

        // Cek apakah chat sudah ada
        $chat = Chat::where(function ($q) use ($senderId, $receiverId) {
            $q->where('user_one_id', $senderId)
              ->where('user_two_id', $receiverId);
        })->orWhere(function ($q) use ($senderId, $receiverId) {
            $q->where('user_one_id', $receiverId)
              ->where('user_two_id', $senderId);
        })->first();

        if (!$chat) {
            $chat = Chat::create([
                'user_one_id' => $senderId,
                'user_two_id' => $receiverId,
            ]);
        }

        return response()->json($chat->load(['userOne', 'userTwo']));
    }
}
```

```
<?php

// app/Http/Controllers/MessageController.php

namespace App\Http\Controllers;

use App\Models\Chat;
use App\Models\Messages;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
use Carbon\Carbon;
use Illuminate\Support\Facades\DB;

class MessageController extends Controller
{
    // Get messages for a chat
    public function index($chatId)
    {
        $messages = Messages::where('chat_id', $chatId)
            ->with('sender')
            ->orderBy('created_at')
            ->get();

        return response()->json($messages);
    }

    // Send a message (text, image, or video)
    public function store(Request $request)
    {
        $request->validate([
            'chat_id' => 'required|exists:chats,id',
            'type' => 'required|in:text,image,video',
            'message' => 'nullable|string',
            'attachment' =>
'nullable|file|mimes:jpg,jpeg,png,mp4,mov|max:90240',
// max 10MB
        ]);
```

```
        $data = $request->only(['chat_id', 'message', 'type']);
        $data['sender_id'] = Auth::id();

        if ($request->hasFile('attachment')) {
            $path = $request->file('attachment')->store('chat_attachments', 'public');
            $data['attachment'] = $path;
        }

        $message = Messages::create($data);

        return response()->json($message->load('sender'));
    }

    public function clear($chatId)
    {
        $chat = Chat::findOrFail($chatId);

        // Hapus file media dari storage
        $mediaMessages = Messages::where('chat_id', $chatId)
            ->whereIn('type', ['image', 'video'])
            ->get();

        foreach ($mediaMessages as $msg) {
            if ($msg->attachment && Storage::disk('public')->exists($msg->attachment)) {
                Storage::disk('public')->delete($msg->attachment);
            }
        }

        // Hapus semua pesan dalam chat
        Messages::where('chat_id', $chatId)->delete();

        return response()->json(['message' => 'Chat berhasil dibersihkan']);
    }

    public function sendAutoReply()
    {
        $thresholdTime = Carbon::now()->subMinutes(1);

        $userMessages = Messages::where('is_read', false)
            ->where('created_at', '<=', $thresholdTime)
            ->whereHas('sender', function ($query) {
                $query->role('user');
            })
            ->get();

        $repliedCount = 0;

        foreach ($userMessages as $userMessage) {
            // Cek apakah sudah dibalas admin
            $adminReplyExists = Messages::where('chat_id', $userMessage->chat_id)
                ->where('created_at', '>', $userMessage->created_at)
                ->whereHas('sender', function ($query) {
                    $query->role('admin');
                })
                ->exists();

            // Cek apakah auto-reply pertama sudah pernah dikirim
            $firstReplyExists = Messages::where('chat_id', $userMessage->chat_id)
                ->where('auto_reply_type', 'first')
                ->exists();

            // Cek apakah auto-reply kedua sudah pernah dikirim
```

```
        $secondReplyExists = Messages::where('chat_id', $userMessage->chat_id)
            ->where('auto_reply_type', 'second')
            ->exists();

        if (!$adminReplyExists && !$firstReplyExists) {
            // Kirim auto-reply pertama
            Messages::create([
                'chat_id' => $userMessage->chat_id,
                'sender_id' => 1,
                'type' => 'text',
                'message' => "Silakan pilih salah satu pertanyaan berikut:\n
1. Apakah manfaat ASI bagi bayi lahir dini ?\n
2. Normalkah jika ASI merembes waktu mendengar tangisan bayi ?\n
3. Bayi baru sering menangis dan terlihat tidak bahagia, apakah ini pertanda dia bayi sulit ?\n
4. Mengapa bayi sering menggeliat ?\n
5. Dapatkah bayi baru lahir mengenali suara ibunya ?",
                'auto_reply_type' => 'first',
            ]);

            $userMessage->is_read = true;
            $userMessage->save();

            $repliedCount++;
        }

        // Kirim auto-reply kedua hanya jika:
        // - Sudah ada auto-reply pertama
        // - User mengirim balasan SETELAH auto-reply pertama
        if ($firstReplyExists && !$secondReplyExists) {
            $afterFirstReply = Messages::where('chat_id', $userMessage->chat_id)
                ->whereHas('sender', function ($query) {
                    $query->role('user');
                })
                ->where('created_at', '>', function ($query) use ($userMessage) {
                    $query->select('created_at')
                        ->from('messages')
                        ->where('chat_id', $userMessage->chat_id)
                        ->where('auto_reply_type', 'first')
                        ->limit(1);
                })
                ->exists();

            if ($afterFirstReply) {
                Messages::create([
                    'chat_id' => $userMessage->chat_id,
                    'sender_id' => 1,
                    'type' => 'text',
                    'message' => 'Terima kasih sudah menanggapi, kami akan segera menindaklanjuti pesan Anda.',
                    'auto_reply_type' => 'second',
                ]);

                $repliedCount++;
            }
        }
    }

    return response()->json([
        'message' => "Auto-reply dikirim ke {$repliedCount} pesan."
    ]);
}
```

```
    public function handleAutoQuestion(Request $request)
    {
        $request->validate([
            'chat_id' => 'required|exists:chats,id',
            'message' => 'required|string',
        ]);

        $message = strtolower($request->message);
        $chatId = $request->chat_id;

        $response = null;

        if (str_contains($message, '1') || str_contains($message, 'Apakah manfaat ASI bagi bayi lahir dini ?')) {
            $response = "ASI sangat dibutuhkan oleh semua bayi, terlebih lagi bayi lahir dini (prematur) . Ketika melewati masa kritis, yaitu minggu pertama kelahirannya , bayi lahir dini sedang berjuang untuk tumbuh. Tak ada yang dapat membantu perjuangannya selain ASI. Sebaiknya, bayi tetap disusui, sekalipun bayi lahir dini umumnya tidak banyak minum. Si kecil sangat membutuhkan ASI, karena ASI menyediakan antibody, kalori, vitamin dan protein sesuai kebutuhan si kecil di masa kritis itu.";
        } elseif (str_contains($message, '2') || str_contains($message, 'Normalkah jika ASI merembes waktu mendengar tangisan bayi ?')) {
            $response = " Normal, Bahkan jika yang menangis itu bukan bayi Anda. Hal ini terjadi karena reaksi oxytocin, hormon yang bertugas mengatur produksi ASI, akan segera diproduksi sesaat Anda teringat pada si kecil. atau mengalami sesuatu yang mengingatkan Anda pada si kecil.";
        } elseif (
            str_contains($message, '3') || str_contains($message, 'Bayi baru sering menangis dan terlihat tidak bahagia, apakah ini pertanda dia bayi sulit ?')
        ) {
            $response = "Hal ini normal terjadi, dan belum dapat digolongkan sebagai bayi sulit. Penelitian menunjukkan bahwa tangis si kecil membentuk pola tertentu. Si kecil akan sering menangis setelah memasuki usia 2 - 3 minggu kelahirannya, dan akan mencapai puncaknya pada usia 6 - 8 minggu, untuk kemudian mereda kembali, hingga mencapai titik terendah pada bulan keempat.Selain itu, bayi biasanya menangis untuk sekedar melepas ketegangan.";
        } elseif (str_contains($message, '4') || str_contains($message, 'Mengapa bayi sering menggeliat ?')) {
            $response = "Bayi, terutama yang masih sangat kecil, banyak bergerak. Umumnya gerakan mereka terjadi diluar kendali, dengan hentakan-hentakan yang disebabkan oleh pesatnya perkembangan neurologis di bulan-bulan pertama. Namun, jika menurut Anda gerakannya lebih daripada gerakan biasa - seperti kejang-kejang segera konsultasikan ke dokter. Pada beberapa kasus, gerakan spontan mengindikasikan adanya serangan atau gangguan neurologis.";
        } elseif (str_contains($message, '5') || str_contains($message, 'Dapatkah bayi baru lahir mengenali suara ibunya ?')) {
            $response = "Penelitian menunjukkan bahwa bayi dapat mengenali suara (suara orang tua, misalnya) dan bereaksi pada bulan pertama kelahirannya, namun tak dapat dipastikan dapat memahami emosi dan makna dari intonasi yang menyertai suara tersebut.";
        }

        if ($response) {
            Messages::create([
                'chat_id' => $chatId,
                'sender_id' => 1,
                'type' => 'text',
                'message' => $response,
                'auto_reply_type' => 'answer',
            ]);

            return response()->json(['message' => 'Jawaban otomatis dikirim.']);
        }

        return response()->json(['message' => 'Tidak ada pertanyaan yang cocok untuk dijawab.'], 400);
    }
```

```
}
```

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rule;

class ProfileController extends Controller
{
    /**
     * Update the authenticated user's profile.
     */
    public function update(Request $request)
    {
        $user = $request->user();

        $validated = $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'email', 'max:255',
Rule::unique('users')->ignore($user->id)],
            'password' => ['nullable', 'string', 'min:8',
'confirmed'], // use password_confirmation
        ]);

        $user->name = $validated['name'];
        $user->email = $validated['email'];

        if (!empty($validated['password'])) {
            $user->password =
Hash::make($validated['password']);
        }

        $user->save();

        return response()->json([
            'message' => 'Profile updated successfully.',
            'user' => $user,
        ]);
    }
}
```

```
<?php

use Illuminate\Http\Request;
use
App\Http\Controllers\Auth\AuthenticatedSessionContro
ller;
use
App\Http\Controllers\Auth\EmailVerificationNotificatio
nController;
use
App\Http\Controllers\Auth\NewPasswordController;
use
App\Http\Controllers\Auth\PasswordResetLinkControll
er;
use
App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use App\Http\Controllers\ArtikelController;
use App\Http\Controllers\BayiController;
use App\Http\Controllers\ChatController;
use App\Http\Controllers\MessageController;
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

Route::middleware(['auth:sanctum'])->get('/user',
function (Request $request) {
    return response()->json([
```

```
        'user' => [
            'id' => $request->user()->id,
            'name' => $request->user()->name,
            'email' => $request->user()->email,
            'roles' => $request->user()->getRoleNames()->toArray() // Ubah ke array
        ]
    ]);
});

Route::middleware('auth:sanctum')->get('/users', function () {
    return \App\Models\User::where('id', '!=', auth()->id())->get(); // Kecuali dirinya sendiri
});

Route::middleware('auth:sanctum')->put('/profile', [ProfileController::class, 'update']);


Route::post('/register', [RegisteredUserController::class, 'store'])
    ->middleware('guest')
    ->name('register');


Route::post('/login', [AuthenticatedSessionController::class, 'store'])
    ->middleware('guest')
    ->name('login');


Route::post('/forgot-password', [PasswordResetLinkController::class, 'store'])
    ->middleware('guest')
    ->name('password.email');


Route::post('/reset-password', [NewPasswordController::class, 'store'])
    ->middleware('guest')
    ->name('password.store');


Route::get('/verify-email/{id}/{hash}', VerifyEmailController::class)
    ->middleware(['auth', 'signed', 'throttle:6,1'])
    ->name('verification.verify');


Route::post('/email/verification-notification', [EmailVerificationNotificationController::class, 'store'])
    ->middleware(['auth', 'throttle:6,1'])
    ->name('verification.send');


Route::post('/logout', [AuthenticatedSessionController::class, 'destroy'])
    ->middleware('auth')
    ->name('logout');


Route::get('artikels/', [ArtikelController::class, 'index']); // Menampilkan semua artikel
Route::post('artikels/', [ArtikelController::class, 'store']); // Menyimpan artikel baru
Route::get('artikels/{artikel}', [ArtikelController::class, 'show']);      // Menampilkan satu artikel
Route::put('artikels/{artikel}', [ArtikelController::class, 'update']);    // Mengupdate artikel
Route::delete('artikels/{artikel}', [ArtikelController::class, 'destroy']); // Menghapus artikel


Route::get('/bayis', [BayiController::class, 'index']); // Menampilkan semua Bayi
Route::post('/bayis', [BayiController::class, 'store']); // Menyimpan Bayi baru
Route::get('/bayis/{bayi}', [BayiController::class, 'show']);     // Menampilkan satu Bayi
Route::put('/bayis/{bayi}', [BayiController::class, 'update']);   // Mengupdate Bayi
Route::delete('/bayis/{bayi}', [BayiController::class, 'destroy']);// Menghapus Bayi


// Chat
Route::get('/chats', [ChatController::class, 'index']); // List semua chat user
```

```
Route::post('/chats', [ChatController::class, 'store']); // Buat atau ambil chat dengan user lain

// Messages
Route::get('/chats/{chat}/messages', [MessageController::class, 'index']); // Ambil semua pesan dalam chat
Route::post('/messages', [MessageController::class, 'store']); // Kirim pesan

Route::post('/auto-reply', [MessageController::class, 'sendAutoReply']); //pesan otomatis
Route::post('/messages/auto-answer', [MessageController::class, 'handleAutoQuestion']); //jawaban otomatis

Route::delete('/chats/{chat}/clear', [MessageController::class, 'clear']);
```

```
import axios from 'axios';
import AsyncStorage from '@react-native-async-storage/async-storage';

const API_URL = 'https://balita.proyekk.com/public/api';
// const API_URL = 'http://192.168.1.6/api'; // Jika ingin tes di lokal

// Konfigurasi instance axios
const api = axios.create({
  baseURL: API_URL,
  headers: {
    'Content-Type': 'application/json',
    'Accept': 'application/json'
  }
});

// Interceptor untuk menambahkan token otomatis di setiap request
api.interceptors.response.use(response => {
  if (response.data.token) {
    AsyncStorage.setItem('userToken', response.data.token);
    api.defaults.headers.common['Authorization'] = `Bearer ${response.data.token}`;
  }
  return response;
}, error => Promise.reject(error));

// 🔹 Registrasi
export const register = async (data) => {
  try {
    const response = await api.post('/register', data);
    return response.data;
  } catch (error) {
    throw error.response?.data || { message: 'Registrasi gagal!' };
  }
};

// 🔹 Login
export const login = async (data) => {
  try {
    const response = await api.post('/login', data);
    const token = response.data.token;
    if (!token) throw { message: 'Token tidak diterima dari server' };
    await AsyncStorage.setItem('userToken', token);
    return response.data;
  } catch (error) {
    throw error.response?.data || { message: 'Login gagal!' };
  }
};

// 🔹 Logout
```

```
export const logout = async () => {
  try {
    const token = await AsyncStorage.getItem('userToken');
    if (!token) throw { message: 'Tidak ada token untuk logout' };
    await api.post('/logout', {}, { headers: { Authorization: `Bearer ${token}` } });
    await AsyncStorage.removeItem('userToken');
    return { message: 'Logout berhasil' };
  } catch (error) {
    throw error.response?.data || { message: 'Logout gagal!' };
  }
};

// 🔹 Mendapatkan data user
export const getUser = async () => {
  try {
    const token = await AsyncStorage.getItem('userToken');
    if (!token) throw new Error('Token tidak ditemukan');

    console.log("Token ditemukan:", token);

    const response = await api.get('/user', {
      headers: { Authorization: `Bearer ${token}` }
    });

    console.log("Response dari API:", response.data);

    const { user } = response.data;
    const roles = user.roles && user.roles.length ? user.roles : ["user"]; // Default role jika kosong

    console.log("User:", user);
    console.log("Roles:", roles);

    await AsyncStorage.setItem('userRole', JSON.stringify(roles));

    return { user, roles };
  } catch (error) {
    console.error("Error saat mendapatkan user:", error);
    return { user: null, roles: ["user"] }; // Jangan throw, tapi kembalikan default
  }
};

// 🔹 Update profil user (name, email, password)
export const updateProfile = async (data) => {
  try {
    const token = await AsyncStorage.getItem('userToken');
    if (!token) throw { message: 'Token tidak tersedia' };

    const response = await api.put('/profile', data, {
      headers: {
        Authorization: `Bearer ${token}`
      }
    });

    return response.data;
  } catch (error) {
    console.error('❌ Error saat update profile:', error.response?.data || error);
    throw error.response?.data || { message: 'Gagal memperbarui profil!' };
  }
};
```

```
// 🔹 Ambil semua chat milik user
export const getChats = async () => {
  const token = await AsyncStorage.getItem('userToken');
  const response = await api.get('/chats', {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
};

// 🔹 Buat atau dapatkan chat antara 2 user
export const createOrGetChat = async (receiver_id) => {
  try {
    const token = await AsyncStorage.getItem('userToken');
    const response = await api.post(
      '/chats', // ← ini disesuaikan dengan route kamu
      { receiver_id },
      { headers: { Authorization: `Bearer ${token}` } }
    );
    return response.data;
  } catch (error) {
    throw error.response?.data || { message: 'Gagal membuat chat!' };
  }
};

// 🔹 Ambil semua pesan dari chat tertentu
export const getMessages = async (chatId) => {
  const token = await AsyncStorage.getItem('userToken');
  const response = await api.get(`/chats/${chatId}/messages`, {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });

  return response.data;
};

export const sendAutoAnswer = async (chatId, message) => {
  try {
    const response = await api.post('/messages/auto-answer', {
      chat_id: chatId,
      message: message,
    });
    return response.data;
  } catch (error) {
    console.error('Error sending auto answer:', error.response?.data || error.message);
    throw error;
  }
};

// 🔹 Kirim pesan (text/gambar/video)
export const sendMessage = async (data) => {
  const token = await AsyncStorage.getItem('userToken');
  const formData = new FormData();

  formData.append('chat_id', data.chat_id);
  formData.append('type', data.type);
  if (data.message) formData.append('message', data.message);
```

```
  if (data.attachment) {
    formData.append('attachment', {
      uri: data.attachment.uri,
      name: data.attachment.name,
      type: data.attachment.type,
    });
  }

  const response = await api.post('/messages', formData,
{
    headers: {
      Authorization: `Bearer ${token}`,
      'Content-Type': 'multipart/form-data'
    }
  });

  return response.data;
};
export const clearChat = async (chatId) => {
  const token = await
AsyncStorage.getItem('userToken');

  const response = await
axios.delete(`${API_URL}/chats/${chatId}/clear`, {
    headers: {
      Authorization: `Bearer ${token}`,
    },
  });

  return response.data;
};

export const getAllUsers = async () => {
  try {
    const token = await
AsyncStorage.getItem('userToken');
    const response = await api.get('/users', {
      headers: { Authorization: `Bearer ${token}` },
    });
    return response.data;
  } catch (error) {
    throw error.response?.data || { message: 'Gagal
mengambil daftar user!' };
  }
};

// ==========================
// 🔹 API ARTIKELS
// ==========================

// 🔹 Get semua artikel
export const getArtikels = async () => {
  const token = await
AsyncStorage.getItem('userToken');
  const response = await api.get('/artikels', {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
};

// 🔹 Get satu artikel berdasarkan ID
export const getArtikel = async (id) => {
  const token = await
AsyncStorage.getItem('userToken');
  const response = await api.get(`/artikels/${id}`, {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
```

```
};

// 🔹 Create artikel baru
export const createArtikel = async (data) => {
  const token = await AsyncStorage.getItem('userToken');

  const formData = new FormData();
  formData.append('judul', data.judul);
  formData.append('isi', data.isi);

  if (data.foto) {
    let localUri = data.foto.uri;
    let filename = localUri.split('/').pop();
    let match = /\.(\w+)$/.exec(filename ?? '');
    let type = match ? `image/${match[1]}` : `image/jpeg`;

    console.log('Foto Upload Info:', { localUri, filename, type });

    formData.append('foto', {
      uri: localUri,
      name: filename ?? `photo-${Date.now()}.jpg`,
      type: type,
    });
  }

  const response = await api.post('/artikels', formData, {
    headers: {
      Authorization: `Bearer ${token}`,
      'Content-Type': 'multipart/form-data', // <<< WAJIB ADA DI SINI
      Accept: 'application/json', // optional
    },
  });
  return response.data;
};

export const updateArtikel = async (id, judul, isi, foto, oldFoto) => {
  const token = await AsyncStorage.getItem('userToken');

  const formData = new FormData();
  formData.append('_method', 'PUT');
  formData.append('judul', judul);
  formData.append('isi', isi);

  // Cek jika ada foto baru
  if (foto) {
    let localUri = foto.uri;
    let filename = localUri.split('/').pop();
    let match = /\.(\w+)$/.exec(filename ?? '');
    let type = match ? `image/${match[1]}` : `image/jpeg`;

    formData.append('foto', {
      uri: localUri,
      name: filename ?? `photo-${Date.now()}.jpg`,
      type: type,
    });
  }
  // ❌ Tidak perlu else oldFoto lagi
  // Biarkan server pertahankan foto lama kalau tidak ada input foto baru

  try {
```

```
    const response = await api.post(`/artikels/${id}`, formData, {
      headers: {
        Authorization: `Bearer ${token}`,
        'Content-Type': 'multipart/form-data',
        Accept: 'application/json',
      },
    });
    console.log('Response:', response.data);
    return response.data;
  } catch (error) {
    console.log('Error Response:', error.response?.data || error.message);
    throw new Error(error.response?.data?.message || 'Gagal mengupdate artikel!');
  }
};




// 🔹 Delete artikel berdasarkan ID
export const deleteArtikel = async (id) => {
  const token = await AsyncStorage.getItem('userToken');
  const response = await api.delete(`/artikels/${id}`, {
    headers: {
      Authorization: `Bearer ${token}`,
      'Content-Type': 'multipart/form-data', // <<< WAJIB ADA DI SINI
      Accept: 'application/json', // optional
    },
  });
  return response.data;
};

// ============================
// 🔹 API BAYIS
// ============================

// 🔹 Get semua artikel
export const getBayi = async () => {
  const token = await AsyncStorage.getItem('userToken');
  const response = await api.get('/bayis', {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
};

// 🔹 Get satu artikel berdasarkan ID
export const getBayis = async (id) => {
  const token = await AsyncStorage.getItem('userToken');
  const response = await api.get(`/bayis/${id}`, {
    headers: {
      Authorization: `Bearer ${token}`
    }
  });
  return response.data;
};

// 🔹 Create artikel baru
export const createBayis = async (data) => {
  const token = await AsyncStorage.getItem('userToken');

  const formData = new FormData();
  formData.append('judul', data.judul);
```

```
  formData.append('isi', data.isi);

  if (data.foto) {
    let localUri = data.foto.uri;
    let filename = localUri.split('/').pop();
    let match = /\.(\w+)$/.exec(filename ?? '');
    let type = match ? `image/${match[1]}` : `image/jpeg`;

    formData.append('foto', {
      uri: localUri,
      name: filename ?? `photo-${Date.now()}.jpg`,
      type: type,
    });
  }

  if (data.video) {
    let localUri = data.video.uri;
    let filename = localUri.split('/').pop();
    let match = /\.(\w+)$/.exec(filename ?? '');
    let type = match ? `video/${match[1]}` : `video/mp4`;

    console.log('Video Upload Info:', { localUri, filename, type });

    formData.append('video', {
      uri: localUri,
      name: filename ?? `video-${Date.now()}.mp4`,
      type: type,
    });
  }

  const response = await api.post('/bayis', formData, {
    headers: {
      Authorization: `Bearer ${token}`,
      'Content-Type': 'multipart/form-data',
      Accept: 'application/json',
    },
  });

  return response.data;
};

export const updateBayis = async (id, judul, isi, foto, oldFoto, video, oldVideo) => {
  const token = await AsyncStorage.getItem('userToken');

  const formData = new FormData();
  formData.append('_method', 'PUT');
  formData.append('judul', judul);
  formData.append('isi', isi);

  // Validasi foto
  if (foto) {
    let localUri = foto.uri;
    let filename = localUri.split('/').pop();
    let match = /\.(\w+)$/.exec(filename ?? '');
    let ext = match ? match[1].toLowerCase() : '';

    // Validasi format gambar
    const allowedImageTypes = ['jpg', 'jpeg', 'png', 'webp'];
    if (!allowedImageTypes.includes(ext)) {
      throw new Error('Format foto tidak valid. Hanya JPG, JPEG, PNG, atau WEBP yang diizinkan.');
    }

    // Validasi ukuran gambar
    if (foto.fileSize && foto.fileSize > 5 * 1024 * 1024) { // 5MB max
```

```
      throw new Error('Ukuran foto terlalu besar. Maksimal 5MB.');
    }

    let type = `image/${ext || 'jpeg'}`;

    formData.append('foto', {
      uri: localUri,
      name: filename ?? `photo-${Date.now()}.jpg`,
      type: type,
    });
  }

  // Validasi video
  if (video) {
    let localUri = video.uri;
    let filename = localUri.split('/').pop();
    let match = /\.(\w+)$/.exec(filename ?? '');
    let ext = match ? match[1].toLowerCase() : '';

    // Validasi format video
    const allowedVideoTypes = ['mp4', 'mov', 'avi', 'mkv'];
    if (!allowedVideoTypes.includes(ext)) {
      throw new Error('Format video tidak valid. Hanya MP4, MOV, AVI, atau MKV yang diizinkan.');
    }

    // Validasi ukuran video
    if (video.fileSize && video.fileSize > 20 * 1024 * 1024) { // 20MB max
      throw new Error('Ukuran video terlalu besar. Maksimal 20MB.');
    }

    let type = `video/${ext || 'mp4'}`;
    formData.append('video', {
      uri: localUri,
      name: filename ?? `video-${Date.now()}.mp4`,
      type: type,
    });
  }

  try {
    const response = await api.post(`/bayis/${id}`, formData, {
      headers: {
        Authorization: `Bearer ${token}`,
        'Content-Type': 'multipart/form-data',
        Accept: 'application/json',
      },
    });
    console.log('Response:', response.data);
    return response.data;
  } catch (error) {
    console.log('Error Response:', error.response?.data || error.message);

    let errorMessage = 'Gagal mengupdate artikel!';

    if (error.response?.data?.errors) {
      // Ambil error pertama yang ada di errors
      const firstKey = Object.keys(error.response.data.errors)[0];
      if (firstKey) {
        errorMessage = error.response.data.errors[firstKey][0];
      }
    } else if (error.response?.data?.message) {
      errorMessage = error.response.data.message;
    }
```

```
    throw new Error(errorMessage);
  }


};




// 🔹 Delete artikel berdasarkan ID
export const deleteBayis = async (id) => {
  const token = await
AsyncStorage.getItem('userToken');
  const response = await api.delete(`/bayis/${id}`, {
    headers: {
      Authorization: `Bearer ${token}`,
     'Content-Type': 'multipart/form-data', // <<< WAJIB
ADA DI SINI
      Accept: 'application/json', // optional
    },
  });
  return response.data;
};


export default api;


import { enableScreens } from 'react-native-screens';
enableScreens();
import { StatusBar } from 'expo-status-bar';
import { NavigationContainer } from '@react-
navigation/native';
import { createNativeStackNavigator } from '@react-
navigation/native-stack';
import { useEffect, useState } from 'react';
import { getUser } from './api';
import { View, ActivityIndicator, Text } from 'react-
native';
import tw from 'twrnc';
import AsyncStorage from '@react-native-async-
storage/async-storage';
import { AuthContext } from
'./Component/AuthContext';
import LoginScreen from './Page/LoginScreen';
import RegisterScreen from './Page/RegisterScreen';
import BottomNavbar from
'./Component/BottomNavbar';
import Header from './Component/Header';
import WelcomeScreen from './Page/WelcomeScreen';
import GlobalErrorBoundary from
'./Component/GlobalErrorBoundary';
import ChatListScreen from './Page/ChatlistScreen';
import ChatDetailScreen from
'./Page/ChatDetailScreen';
import NewChatScreen from './Page/NewChatScreen';
import IndexAr from './Page/IndexAr';
import ArtikelTambah from './Page/ArtikelTambah';
import ArtikelEdit from './Page/ArtikelEdit';
import ArtikelDetail from './Page/ArtikelDetail';
import IndexB from './Page/IndexB';
import BayiTambah from './Page/BayiTambah';
import BayiDetail from './Page/BayiDetail';
import BayiEdit from './Page/BayiEdit';


const Stack = createNativeStackNavigator();


export default function App() {
  const [user, setUser] = useState(null);
  const [role, setRole] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [checkingToken, setCheckingToken] =
useState(true);


  const checkAuth = async () => {
    setLoading(true);
```

```
      try {
        const token = await AsyncStorage.getItem('userToken');
        if (!token) {
          setUser(null);
          setRole([]);
          console.log("Token tidak ditemukan, set role ke []");
        } else {
          const { user, roles } = await getUser();
          console.log("User yang login:", user);
          console.log("Role sebelum diset:", roles);

          setUser(user);
          setRole(roles && roles.length ? roles : ["user"]);
          await AsyncStorage.setItem('userRole', JSON.stringify(roles));

          console.log("Role setelah diset di AsyncStorage:", roles);
        }
      } catch (err) {
        setUser(null);
        setRole(["user"]);
        console.log("Gagal mendapatkan user, set role ke ['user']");
      } finally {
        setLoading(false);
      }
    };

    // **Pastikan `role` selalu diperbarui saat `user` berubah**
    useEffect(() => {
      if (user) {
        setRole(user.roles || ["user"]);
        console.log("Role di AuthContext diperbarui:", user.roles || ["user"]);
      }
    }, [user]);

    useEffect(() => {
      checkAuth();
    }, []);
    if (loading) {
      return (
        <View style={tw`flex-1 justify-center items-center bg-gray-100`}>
          <ActivityIndicator size="large" color="#0000ff" />
          <Text style={tw`mt-4 text-lg text-gray-700`}>Memuat aplikasi...</Text>
        </View>
      );
    }

    return (
      <GlobalErrorBoundary>
        <AuthContext.Provider value={{ user, role, setUser }}>
          <NavigationContainer>
            <Header />
            <Stack.Navigator screenOptions={{ headerShown: false }}>
              {!user ? (
                <>
                  <Stack.Screen name="Welcome" component={WelcomeScreen} />
                  <Stack.Screen name="Login" component={LoginScreen} />
                  <Stack.Screen name="Register" component={RegisterScreen} />
                </>
```

```
                ) : (
                  <>
                    <Stack.Screen name="Main" component={BottomNavbar} />
                    <Stack.Screen name="Inbox" component={ChatListScreen} />
                    <Stack.Screen name="ChatDetail" component={ChatDetailScreen} options={({ route }) => ({
                      title: route.params?.user?.name || 'Chat'
                    })} />
                    <Stack.Screen name="Artikel" component={IndexAr} />
                    <Stack.Screen name="ArtikelDetail" component={ArtikelDetail} />
                    <Stack.Screen name="Bayi" component={IndexB} />
                    <Stack.Screen name="BayiDetail" component={BayiDetail} />
                    <Stack.Screen name="NewChat" component={NewChatScreen} options={{ title: 'Buat Chat Baru' }} />

                    {/* Menampilkan halaman admin hanya jika role termasuk 'admin' */}
                    {role.includes('admin') && (
                      <>
                        <Stack.Screen name="ArtikelTambah" component={ArtikelTambah} />
                        <Stack.Screen name="ArtikelEdit" component={ArtikelEdit} />
                        <Stack.Screen name="BayiTambah" component={BayiTambah} />
                        <Stack.Screen name="BayiEdit" component={BayiEdit} />
                      </>
                    )}
                  </>
                )}
              </Stack.Navigator>
              <StatusBar style="auto" />
            </NavigationContainer>
          </AuthContext.Provider>
        </GlobalErrorBoundary>
    );
}
```

```
import React, { useContext } from 'react';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';
import { FontAwesome } from '@expo/vector-icons';
import { AuthContext } from './AuthContext';
import DashboardScreen from '../Page/DashboardScreen';
import ChatListScreen from '../Page/ChatlistScreen';
import IndexAr from '../Page/IndexAr';
import IndexB from '../Page/IndexB';
import ProfilScreen from '../Page/ProfilScreen';

const Tab = createBottomTabNavigator();

export default function BottomNavbar() {
  const { role } = useContext(AuthContext);  // Ambil role dari context

  return (
<Tab.Navigator
 screenOptions={({ route }) => ({
  tabBarIcon: ({ color, size }) => {
    let iconName = route.name === 'Home' ? 'home' : 'user';
    return <FontAwesome name={iconName} size={size} color={color} />;
  },
  tabBarActiveTintColor: '#2196F3',
  tabBarInactiveTintColor: 'gray',
  tabBarLabelStyle: {
```

```
    fontSize: 14, // Ganti ukuran sesuai keinginan, misalnya 16 untuk lebih besar
    fontWeight: 'bold', // Opsional
  },
 })}
>

        <Tab.Screen name="Home" component={DashboardScreen} />
        <Tab.Screen name="Chat" component={ChatListScreen} />
        <Tab.Screen name="Artikel" component={IndexAr} />
        <Tab.Screen name="Bayi" component={IndexB} />
        <Tab.Screen name="Profile" component={ProfilScreen} />


      </Tab.Navigator>
  );
}

import { View, Text, TouchableOpacity, Image } from 'react-native';
import tw from 'twrnc';

export default function WelcomeScreen({ navigation }) {
  return (
    <View style={tw`flex-1 justify-center items-center bg-blue-50 p-4`}>
        <Image
          source={{ uri: 'https://i.pinimg.com/474x/3e/ed/38/3eed380389c458c1a3b7865796a8d105.jpg' }} // Ganti dengan URL gambar yang sesuai
          style={tw`w-32 h-32 mb-6 rounded-full`}
        />
        <Text style={tw`text-3xl font-extrabold text-blue-600 mb-6`}>
          Selamat Datang di teman Balita
        </Text>
        <Text style={tw`text-lg text-gray-700 text-center mb-8 px-6`}>
          Semua yang perlu Anda ketahui tentang merawat bayi dan balita dalam satu aplikasi.
        </Text>

        <TouchableOpacity
          style={tw`bg-green-500 px-8 py-3 rounded-lg mb-4`}
          onPress={() => navigation.navigate('Login')}
        >
          <Text style={tw`text-white text-xl font-bold`}>Masuk</Text>
        </TouchableOpacity>

        <TouchableOpacity
          style={tw`border-2 border-green-500 px-8 py-3 rounded-lg`}
          onPress={() => navigation.navigate('Register')}
        >
          <Text style={tw`text-green-500 text-xl font-bold`}>Daftar</Text>
        </TouchableOpacity>
      </View>
  );
}

import { useState } from 'react';
import { View, Text, TextInput, Button, Alert, TouchableOpacity } from 'react-native';
import { register } from '../api';
import tw from 'twrnc';
import { FontAwesome } from '@expo/vector-icons';

export default function RegisterScreen({ navigation }) {
```

```
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [passwordConfirmation, setPasswordConfirmation] = useState('');
  const [showPassword, setShowPassword] = useState(false);
  const [showPasswordConfirmation, setShowPasswordConfirmation] = useState(false);

  const handleRegister = async () => {
    if (password !== passwordConfirmation) {
      Alert.alert('Gagal', 'Password dan Konfirmasi Password tidak sama!');
      return;
    }
    try {
      await register({
        name,
        email,
        password,
        password_confirmation: passwordConfirmation
      });
      Alert.alert('Berhasil', 'Registrasi sukses!');
      navigation.navigate('Login');
    } catch (error) {
      Alert.alert('Gagal', error.message || 'Registrasi gagal!');
    }
  };

  return (
    <View style={tw`flex-1 items-center justify-center bg-gray-100 p-6`}>
      <Text style={tw`text-2xl font-bold mb-6`}>Register</Text>
      <TextInput
        placeholder="Nama"
        value={name}
        onChangeText={setName}
        style={tw`w-full bg-white p-4 mb-4 rounded-xl shadow-sm`}
      />
      <TextInput
        placeholder="Email"
        value={email}
        onChangeText={setEmail}
        style={tw`w-full bg-white p-4 mb-4 rounded-xl shadow-sm`}
        keyboardType="email-address"
        autoCapitalize="none"
      />

      <View style={tw`w-full bg-white flex-row items-center p-4 mb-4 rounded-xl shadow-sm`}>
        <TextInput
          placeholder="Password"
          value={password}
          onChangeText={setPassword}
          style={tw`flex-1`}
          secureTextEntry={!showPassword}
        />
        <TouchableOpacity onPress={() => setShowPassword(!showPassword)}>
          <FontAwesome name={showPassword ? "eye-slash" : "eye"} size={20} />
        </TouchableOpacity>
      </View>

      <View style={tw`w-full bg-white flex-row items-center p-4 mb-6 rounded-xl shadow-sm`}>
        <TextInput
          placeholder="Konfirmasi Password"
          value={passwordConfirmation}
```

```
onChangeText={setPasswordConfirmation}
                style={tw`flex-1`}
secureTextEntry={!showPasswordConfirmation}
              />
              <TouchableOpacity onPress={() =>
setShowPasswordConfirmation(!showPasswordConfir
mation)}>
                <FontAwesome
name={showPasswordConfirmation ? "eye-slash" :
"eye"} size={20} />
              </TouchableOpacity>
            </View>

            <View style={tw`w-full mt-2`}>
              <Button title="Register"
onPress={handleRegister} color="#4CAF50" />
            </View>
            <TouchableOpacity onPress={() =>
navigation.navigate('Login')}>
              <Text style={tw`text-sm mt-4 text-blue-
600`}>Sudah punya akun? Login di sini</Text>
            </TouchableOpacity>
        </View>
    );
}

import { useState, useEffect } from 'react';
import { View, Text, TextInput, TouchableOpacity,
Alert, ScrollView } from 'react-native';
import tw from 'twrnc';
import { getUser, updateProfile } from '../api';

export default function ProfilScreen({ navigation }) {
    const [name, setName] = useState('');
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [passwordConfirmation,
setPasswordConfirmation] = useState('');

    // Ambil data user saat pertama kali dibuka
    useEffect(() => {
      (async () => {
        try {
          const user = await getUser();
      const userData = user.user;
      setName(userData.name);
      setEmail(userData.email);
        } catch (err) {
          Alert.alert('Gagal mengambil data user',
err.message || 'Terjadi kesalahan');
        }
      })();
    }, []);

    const handleUpdate = async () => {
      if (password && password !==
passwordConfirmation) {
        return Alert.alert('Error', 'Konfirmasi password
tidak cocok.');
      }

      try {
        const data = {
          name,
          email,
          password: password || undefined,
          password_confirmation:
passwordConfirmation || undefined,
        };

        const response = await updateProfile(data);
        Alert.alert('Sukses', response.message);
      } catch (error) {
```

```
      Alert.alert('Gagal update profil', error.message || 'Terjadi kesalahan');
    }
  };

  return (
    <ScrollView contentContainerStyle={tw`flex-grow justify-center bg-gray-100 p-6`}>
      <View style={tw`bg-white rounded-2xl p-6 shadow`}>
        <Text style={tw`text-2xl font-bold mb-4 text-center`}>Edit Profil</Text>

        <Text style={tw`mb-1 text-gray-700`}>Nama</Text>
        <TextInput
          style={tw`border border-gray-300 rounded-xl px-4 py-2 mb-4`}
          value={name}
          onChangeText={setName}
        />

        <Text style={tw`mb-1 text-gray-700`}>Email</Text>
        <TextInput
          style={tw`border border-gray-300 rounded-xl px-4 py-2 mb-4`}
          value={email}
          onChangeText={setEmail}
          keyboardType="email-address"
          autoCapitalize="none"
        />

        <Text style={tw`mb-1 text-gray-700`}>Password Baru</Text>
        <TextInput
          style={tw`border border-gray-300 rounded-xl px-4 py-2 mb-4`}
          value={password}
          onChangeText={setPassword}
          secureTextEntry
        />

        <Text style={tw`mb-1 text-gray-700`}>Konfirmasi Password</Text>
        <TextInput
          style={tw`border border-gray-300 rounded-xl px-4 py-2 mb-6`}
          value={passwordConfirmation}
          onChangeText={setPasswordConfirmation}
          secureTextEntry
        />

        <TouchableOpacity
          onPress={handleUpdate}
          style={tw`bg-blue-500 rounded-xl py-3`}
        >
          <Text style={tw`text-white text-center text-lg font-semibold`}>Simpan Perubahan</Text>
        </TouchableOpacity>
      </View>
    </ScrollView>
  );
}

import React, { useEffect, useState } from 'react';
import { View, Text, FlatList, TouchableOpacity, ActivityIndicator, Alert } from 'react-native';
import { getAllUsers, createOrGetChat, getChats } from '../api';
import { useNavigation } from '@react-navigation/native';
import AsyncStorage from '@react-native-async-storage/async-storage';

const NewChatScreen = () => {
  const [users, setUsers] = useState([]);
```

```
  const [chats, setChats] = useState([]);
  const [loading, setLoading] = useState(true);
  const [currentUserId, setCurrentUserId] = useState(null);
  const [userRole, setUserRole] = useState(null);  // Menyimpan role pengguna
  const navigation = useNavigation();

  useEffect(() => {
    const fetchData = async () => {
      try {
        // Ambil data pengguna dari AsyncStorage
        const userData = await AsyncStorage.getItem('user');
        const parsedUser = JSON.parse(userData);
        setCurrentUserId(parsedUser?.id);

        // Ambil data role pengguna
        const roleData = await AsyncStorage.getItem('userRole');
        const parsedRole = JSON.parse(roleData);
        setUserRole(parsedRole);

        const [allUsers, existingChats] = await Promise.all([
          getAllUsers(),
          getChats()
        ]);

        setChats(existingChats);

        // Filter user agar hanya menampilkan ID 1 jika bukan admin
        const filteredUsers = parsedRole.includes('admin')
          ? allUsers.filter((user) => user.id !== parsedUser?.id)  // Admin bisa chat dengan siapa saja
          : allUsers.filter((user) => user.id === 1);  // Non-admin hanya bisa chat dengan ID 1
        setUsers(filteredUsers);
      } catch (err) {
        console.error('❌ Error saat mengambil data:', err);
        Alert.alert('Gagal', 'Tidak dapat mengambil data');
      } finally {
        setLoading(false);
      }
    };

    fetchData();
  }, []);

  const handleSelectUser = async (userId, userData) => {
    try {
      const chat = await createOrGetChat(userId);
      navigation.navigate('ChatDetail', { chatId: chat.id, user: userData });
    } catch (err) {
      console.error('❌ Gagal membuat chat:', err);
      Alert.alert('Gagal', err.message || 'Terjadi kesalahan');
    }
  };

  if (loading) {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <ActivityIndicator size="large" color="#000" />
      </View>
    );
  }
```

```
 return (
   <FlatList
     data={users}
     keyExtractor={(item) => item.id.toString()}
     renderItem={({ item }) => {
       const hasExistingChat = chats.some(
         chat =>
           (chat.user_one_id === currentUserId &&
chat.user_two?.id === item.id) ||
           (chat.user_two_id === currentUserId &&
chat.user_one?.id === item.id)
       );

       return (
         <TouchableOpacity
           onPress={() => handleSelectUser(item.id,
item)}
           style={{
             padding: 16,
             borderBottomWidth: 1,
             borderColor: '#ddd',
           }}
         >
           <Text style={{ fontWeight: 'bold'
}}>{item.name}</Text>
           <Text style={{ color: '#555'
}}>{item.email}</Text>
           {hasExistingChat && <Text style={{ color:
'green' }}>✔ Sudah ada chat</Text>}
         </TouchableOpacity>
       );
     }}
     contentContainerStyle={{ paddingBottom: 40 }}
   />
 );
};
```

```
export default NewChatScreen;


import { useState, useContext, useEffect } from 'react';
import { View, Text, TextInput, Button, Alert,
TouchableOpacity } from 'react-native';
import { login } from '../api';
import { AuthContext } from
'../Component/AuthContext';
import tw from 'twrnc';
import { FontAwesome } from '@expo/vector-icons';
import AsyncStorage from '@react-native-async-
storage/async-storage';
import { useNavigation } from '@react-
navigation/native'; // ⬅️ Tambahkan ini


export default function LoginScreen() {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const { setUser } = useContext(AuthContext);
  const [showPassword, setShowPassword] =
useState(false);
  const navigation = useNavigation(); // ⬅️ Gunakan
ini untuk navigasi


  const handleLogin = async () => {
    try {
      if (!email.trim() || !password.trim()) {
        Alert.alert('Error', 'Email dan password tidak
boleh kosong!');
        return;
      }


      const response = await login({ email, password
});


      await AsyncStorage.setItem('userToken',
response.token);
      setUser(response.user);
```

```
      Alert.alert('Berhasil', 'Login sukses!');


    } catch (error) {

      console.error('❌ Login gagal:', error);

      Alert.alert('Gagal', error.message || 'Login gagal!');

    }

  };


  return (

    <View style={tw`flex-1 items-center justify-center bg-gray-100 p-6`}>

      <Text style={tw`text-2xl font-bold mb-6 text-red-500`}>Login</Text>


      <TextInput

        placeholder="Email"

        value={email}

        onChangeText={setEmail}

        style={tw`w-full bg-white p-4 mb-4 rounded-xl shadow-sm`}

        keyboardType="email-address"

        autoCapitalize="none"

      />

      <View style={tw`w-full bg-white flex-row items-center p-4 mb-4 rounded-xl shadow-sm`}>

        <TextInput

          placeholder="Password"

          value={password}

          onChangeText={setPassword}

          style={tw`flex-1`}

          secureTextEntry={!showPassword}

        />

        <TouchableOpacity onPress={() => setShowPassword(!showPassword)}>

          <FontAwesome name={showPassword ? "eye-slash" : "eye"} size={20} />

        </TouchableOpacity>

      </View>


      <View style={tw`w-full mt-2`}>

        <Button title="Login" onPress={handleLogin} color="#4CAF50" />

      </View>


      <TouchableOpacity onPress={() => navigation.navigate('Register')}>

        <Text style={tw`text-sm mt-4 text-blue-600`}>Belum punya akun? Daftar di sini</Text>

      </TouchableOpacity>

    </View>

  );

}


import React, { useEffect, useState, useContext } from 'react';

import { View, Text, TouchableOpacity, FlatList, Alert } from 'react-native';

import tw from 'twrnc';

import { getBayi, deleteBayis } from '../api'; // pastikan path ke api.js benar

import { useNavigation } from '@react-navigation/native';

import { AuthContext } from '../Component/AuthContext';


const IndexB = () => {

  const [Bayi, setBayi] = useState([]);

  const navigation = useNavigation();

  const { user, role } = useContext(AuthContext);


  useEffect(() => {

    console.log("User di Navbar:", user);
```

```
  console.log("Role di Navbar:", role);
 }, [user, role]); // Debugging setiap `user` atau `role` berubah

 if (!user || !role) {
  return null; // Hindari error saat role masih undefined
 }

 useEffect(() => {
  fetchBayi();
 }, []);

 const fetchBayi = async () => {
  try {
   const data = await getBayi();
   setBayi(data);
  } catch (error) {
   console.error('Gagal mengambil Bayi:', error);
  }
 };

 const handleDelete = (id) => {
  Alert.alert(
   'Konfirmasi',
   'Apakah kamu yakin ingin menghapus Bayi ini?',
   [
    { text: 'Batal', style: 'cancel' },
    {
     text: 'Hapus',
     style: 'destructive',
     onPress: async () => {
      try {
       await deleteBayis(id);
       fetchBayi(); // refresh list
      } catch (error) {
       console.error('Gagal menghapus Bayi:', error);
      }
     }
    }
   ]
  );
 };

 const truncateIsi = (isi) => {
  // Membatasi isi bayi hanya sampai 30 karakter
  return isi.length > 30 ? isi.substring(0, 30) + '...' : isi;
 };

 const renderItem = ({ item }) => (
  <View style={tw`bg-white p-4 mb-4 rounded-lg shadow`}>
   <Text style={tw`text-2xl font-bold mb-2`}>{item.judul}</Text>
   <Text style={tw`text-lg mb-2`}>{truncateIsi(item.isi)}</Text>
   <View style={tw`flex-row justify-between`}>
    <TouchableOpacity
     style={tw`bg-blue-500 px-4 py-2 rounded`}
     onPress={() => navigation.navigate('BayiDetail', { id: item.id })}
    >
     <Text style={tw`text-white text-center`}>Lihat Selengkapnya</Text>
    </TouchableOpacity>
    {role.includes('admin') && (  // Periksa role sebagai array
     <>
      <TouchableOpacity
       style={tw`bg-yellow-500 px-4 py-2 rounded`}
       onPress={() => navigation.navigate('BayiEdit', { id: item.id })}
      >
```

```
            <Text style={tw`text-white text-
center`}>Edit</Text>
          </TouchableOpacity>

          <TouchableOpacity
            style={tw`bg-red-500 px-4 py-2 rounded`}
            onPress={() => handleDelete(item.id)}
          >
            <Text style={tw`text-white text-
center`}>Hapus</Text>
          </TouchableOpacity>
        </>
      )}
    </View>
  </View>
 );

 return (
  <View style={tw`flex-1 p-4 bg-gray-100`}>
    {role.includes('admin') && ( // Periksa role sebagai
array
      <TouchableOpacity
        style={tw`bg-green-500 p-3 rounded-lg mb-4`}
        onPress={() =>
navigation.navigate('BayiTambah')}
      >
        <Text style={tw`text-white text-center font-bold
text-lg`}>+ Tambah Bayi</Text>
      </TouchableOpacity>
    )}

    <FlatList
      data={Bayi}
      keyExtractor={(item) => item.id.toString()}
      renderItem={renderItem}
      refreshing={false}
      onRefresh={fetchBayi}
      ListEmptyComponent={
        <Text style={tw`text-center text-gray-500 mt-
10`}>Belum ada Bayi</Text>
      }
    />
  </View>
 );
};

export default IndexB;

import React, { useEffect, useState, useContext } from
'react';

import { View, Text, TouchableOpacity, FlatList, Alert,
TextInput } from 'react-native';

import tw from 'twrnc';

import { getArtikels, deleteArtikel } from '../api'; //
pastikan path ke api.js benar

import { useNavigation } from '@react-
navigation/native';

import { AuthContext } from
'../Component/AuthContext';

const IndexAr = () => {
 const [artikels, setArtikels] = useState([]);
 const [filteredArtikels, setFilteredArtikels] =
useState([]); // Menyimpan artikel yang sudah difilter
 const [searchQuery, setSearchQuery] = useState(''); //
Menyimpan query pencarian
 const navigation = useNavigation();
 const { user, role } = useContext(AuthContext);

 useEffect(() => {
  console.log("User di Navbar:", user);
  console.log("Role di Navbar:", role);
 }, [user, role]); // Debugging setiap `user` atau `role`
berubah

 if (!user || !role) {
```

```
  return null; // Hindari error saat role masih undefined
 }

 useEffect(() => {
  fetchArtikels();
 }, []);

 useEffect(() => {
  if (searchQuery === '') {
   setFilteredArtikels(artikels); // Menampilkan semua artikel jika tidak ada pencarian
  } else {
   const results = artikels.filter((artikel) =>
artikel.judul.toLowerCase().includes(searchQuery.toLowerCase()) ||
artikel.isi.toLowerCase().includes(searchQuery.toLowerCase())
   );
   setFilteredArtikels(results);
  }
 }, [searchQuery, artikels]);

 const fetchArtikels = async () => {
  try {
   const data = await getArtikels();
   setArtikels(data);
   setFilteredArtikels(data); // Menampilkan semua artikel setelah data diambil
  } catch (error) {
   console.error('Gagal mengambil artikel:', error);
  }
 };

 const handleDelete = (id) => {
  Alert.alert(
   'Konfirmasi',
   'Apakah kamu yakin ingin menghapus artikel ini?',
   [
    { text: 'Batal', style: 'cancel' },
    {
     text: 'Hapus',
     style: 'destructive',
     onPress: async () => {
      try {
       await deleteArtikel(id);
       fetchArtikels(); // refresh list
      } catch (error) {
       console.error('Gagal menghapus artikel:', error);
      }
     }
    }
   ]
  );
 };

 const truncateIsi = (isi) => {
  // Membatasi isi artikel hanya sampai 30 karakter
  return isi.length > 30 ? isi.substring(0, 30) + '...' : isi;
 };

 const renderItem = ({ item }) => (
  <View style={tw`bg-white p-4 mb-4 rounded-lg shadow`}>
   <Text style={tw`text-2xl font-bold mb-2`}>{item.judul}</Text>
   <Text style={tw`text-lg mb-2`}>{truncateIsi(item.isi)}</Text>
   <View style={tw`flex-row justify-between`}>
    <TouchableOpacity
     style={tw`bg-blue-500 px-4 py-2 rounded`}
```

```
        onPress={() =>
navigation.navigate('ArtikelDetail', { id: item.id })}
      >
        <Text style={tw`text-white text-center`}>Lihat
Selengkapnya</Text>
      </TouchableOpacity>
      {role.includes('admin') && (  // Menyembunyikan
tombol hanya untuk admin
        <>
          <TouchableOpacity
            style={tw`bg-yellow-500 px-4 py-2 rounded`}
            onPress={() =>
navigation.navigate('ArtikelEdit', { id: item.id })}
          >
            <Text style={tw`text-white text-
center`}>Edit</Text>
          </TouchableOpacity>

          <TouchableOpacity
            style={tw`bg-red-500 px-4 py-2 rounded`}
            onPress={() => handleDelete(item.id)}
          >
            <Text style={tw`text-white text-
center`}>Hapus</Text>
          </TouchableOpacity>
        </>
      )}
    </View>
  </View>
 );

 return (
  <View style={tw`flex-1 p-4 bg-gray-100`}>
    {role.includes('admin') && (  // Menyembunyikan
tombol hanya untuk admin
      <TouchableOpacity
        style={tw`bg-green-500 p-3 rounded-lg mb-4`}
        onPress={() =>
navigation.navigate('ArtikelTambah')}
      >
        <Text style={tw`text-white text-center font-bold
text-lg`}>+ Tambah Artikel</Text>
      </TouchableOpacity>
    )}

    {/* Filter pencarian */}
    <TextInput
      style={tw`bg-white p-3 rounded-lg mb-4`}
      placeholder="Cari artikel..."
      value={searchQuery}
      onChangeText={setSearchQuery}
    />

    <FlatList
      data={filteredArtikels}
      keyExtractor={(item) => item.id.toString()}
      renderItem={renderItem}
      refreshing={false}
      onRefresh={fetchArtikels}
      ListEmptyComponent={
        <Text style={tw`text-center text-gray-500 mt-
10`}>Belum ada artikel</Text>
      }
    />
  </View>
 );
};

export default IndexAr;

import { View, Text, Button, Alert, TouchableOpacity }
from 'react-native';
import { useEffect, useState, useContext } from 'react';
```

```
import { getUser, logout } from '../api';

import tw from 'twrnc';

import { AuthContext } from
'../Component/AuthContext';


export default function DashboardScreen({ navigation
}) {

  const { user, setUser } = useContext(AuthContext);


  useEffect(() => {

    const fetchUser  = async () => {

      try {

        const user = await getUser();

        const userDatas = user.user;

        setUser (userDatas);

      } catch (error) {

        setUser (null);

      }

    };

    fetchUser ();

  }, []);


  const handleLogout = () => {

    logout();

    setUser(null);

    navigation.navigate('Welcome');

  };


  return (

    <View style={tw`flex-1 bg-blue-50 p-6 justify-
center items-center`}>

      <Text style={tw`text-3xl font-extrabold text-
blue-600 mb-6`}>

        Dashboard

      </Text>

      {user ? (

        <View style={tw`bg-white p-6 rounded-xl
shadow-lg mb-6`}>

          <Text style={tw`text-lg font-semibold text-
blue-700`}>

            Selamat datang, Bunda!

          </Text>

          <Text style={tw`text-sm text-gray-500 mb-
4`}>Email: {user.email}</Text>



        </View>

      ) : (

        <View style={tw`bg-white p-6 rounded-xl
shadow-lg mb-6`}>

          <Text style={tw`text-lg text-center text-
gray-700 mb-4`}>

            Anda belum login

          </Text>

          <TouchableOpacity

            style={tw`bg-green-500 px-8 py-3
rounded-lg`}

            onPress={() =>
navigation.navigate('Login')}

          >

            <Text style={tw`text-white text-xl font-
bold`}>Masuk</Text>

          </TouchableOpacity>

        </View>

      )}

    </View>

  );

}


import React, { useEffect, useState } from 'react';

import { View, Text, FlatList, TouchableOpacity,
ActivityIndicator } from 'react-native';

import { getChats } from '../api';
```

```
import { useNavigation } from '@react-navigation/native';


const ChatListScreen = () => {
  const [chats, setChats] = useState([]);
  const [loading, setLoading] = useState(true);
  const navigation = useNavigation();


  useEffect(() => {
    const fetchChats = async () => {
      try {
        const data = await getChats();
        setChats(data);
      } catch (err) {
        console.error('❌ Gagal mengambil chat:', err);
      } finally {
        setLoading(false);
      }
    };


    fetchChats();
  }, []);


  const renderItem = ({ item }) => {
    const otherUser = item.user_one_id === item.user_two?.id ? item.user_one : item.user_two;


    return (
      <TouchableOpacity
        onPress={() => navigation.navigate('ChatDetail', { chatId: item.id, user: otherUser })}
        style={{
          padding: 16,
          borderBottomWidth: 1,
          borderColor: '#ddd',
        }}
      >
        <Text style={{ fontWeight: 'bold' }}>{otherUser?.name || 'Pengguna'}</Text>
        {item.messages && item.messages.length > 0 && (
          <Text numberOfLines={1} style={{ color: '#666' }}>
            {item.messages[0]?.message || '[Attachment]'}
          </Text>
        )}
      </TouchableOpacity>
    );
  };


  if (loading) {
    return (
      <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
        <ActivityIndicator size="large" color="#000" />
      </View>
    );
  }


  return (
    <View style={{ flex: 1 }}>
      {chats.length === 0 ? (
        <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
          <Text style={{ marginBottom: 12 }}>Belum ada chat</Text>
          <TouchableOpacity
            onPress={() => navigation.navigate('NewChat')}
            style={{
              backgroundColor: '#007AFF',
              paddingHorizontal: 20,
              paddingVertical: 12,
```

```
            borderRadius: 8,
          }}
        >
          <Text style={{ color: '#fff' }}>+ Buat Chat</Text>
        </TouchableOpacity>
      </View>
    ) : (
      <FlatList
        data={chats}
        keyExtractor={(item) => item.id.toString()}
        renderItem={renderItem}
        contentContainerStyle={{ paddingBottom: 40 }}
      />
    )}
  </View>
 );
};

export default ChatListScreen;
import React, { useEffect, useState } from 'react';
import {
 View,
 Text,
 TextInput,
 TouchableOpacity,
 ActivityIndicator,
 Alert,
 FlatList,
 Image,
} from 'react-native';
import * as ImagePicker from 'expo-image-picker';
import { sendMessage, getMessages, clearChat, sendAutoAnswer } from '../api';
import { Video } from 'expo-av';
import { useNavigation } from '@react-navigation/native';

const ChatDetailScreen = ({ route }) => {
 const { chatId, user } = route.params;
 const navigation = useNavigation(); // Untuk navigasi kembali
 const [message, setMessage] = useState('');
 const [uploading, setUploading] = useState(false);
 const [messages, setMessages] = useState([]);
 const [loadingMessages, setLoadingMessages] = useState(true);

 useEffect(() => {
  fetchMessages();
 }, []);

 const fetchMessages = async () => {
  try {
   const data = await getMessages(chatId);
   setMessages(data);
  } catch (err) {
   console.error('❌ Gagal mengambil pesan:', err);
  } finally {
   setLoadingMessages(false);
  }
 };

 const handleSendText = async () => {
  if (!message.trim()) return;

  try {
   setUploading(true);
   await sendMessage({
    chat_id: chatId,
    type: 'text',
```

```
      message,
    });
    setMessage('');
    fetchMessages();
   } catch (error) {
    Alert.alert('Gagal', 'Tidak dapat mengirim pesan');
   } finally {
    setUploading(false);
   }
  };
  const handleAutoReplyResponse = async (questionText) => {
   try {
    await sendAutoAnswer(chatId, questionText);
    fetchMessages(); // ambil ulang pesan setelah kirim jawaban
   } catch (error) {
    Alert.alert('Gagal', 'Tidak bisa mengirim jawaban otomatis');
   }
  };

  const pickMedia = async () => {
   const result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.All,
    allowsEditing: false,
    quality: 1,
   });

   if (!result.canceled) {
    const asset = result.assets[0];
    const type = asset.type === 'video' ? 'video' : 'image';
    handleSendMedia(asset, type);
   }
  };

const handleSendMedia = async (asset, type) => {
 try {
  setUploading(true);

  const uriParts = asset.uri.split('.');
  const fileType = uriParts[uriParts.length - 1];

  await sendMessage({
   chat_id: chatId,
   type,
   attachment: {
    uri: asset.uri,
    name: `upload.${fileType}`,
    type: `${type}/${fileType}`,
   },
  });

  fetchMessages();
 } catch (error) {
  Alert.alert('Gagal', 'Tidak dapat mengirim media');
 } finally {
  setUploading(false);
 }
};

const styles = {
 autoBtn: {
  backgroundColor: '#007bff',
  padding: 10,
  borderRadius: 8,
  marginTop: 6,
 },
 autoBtnText: {
```

```
    color: '#fff',
    textAlign: 'center',
  },
});

const renderItem = ({ item }) => {
  const isQuestionPrompt = item.message?.includes('Silakan pilih salah satu pertanyaan');

  return (
    <View style={{ marginVertical: 8 }}>
      <Text style={{ fontWeight: 'bold' }}>{item.sender?.name || 'User'}:</Text>

      {item.type === 'text' && (
        <Text style={{ fontSize: 16 }}>{item.message}</Text>
      )}

      {isQuestionPrompt && (
        <View style={{ marginTop: 10 }}>
          <TouchableOpacity
            onPress={() => handleAutoReplyResponse('1. Apakah manfaat ASI bagi bayi lahir dini ?')}
            style={styles.autoBtn}
          >
            <Text style={styles.autoBtnText}>1. Apakah manfaat ASI bagi bayi lahir dini ?</Text>
          </TouchableOpacity>

          <TouchableOpacity
            onPress={() => handleAutoReplyResponse('2. Normalkah jika ASI merembes waktu mendengar tangisan bayi ?')}
            style={styles.autoBtn}
          >
            <Text style={styles.autoBtnText}>2. Normalkah jika ASI merembes waktu mendengar tangisan bayi ?</Text>
          </TouchableOpacity>

          <TouchableOpacity
            onPress={() => handleAutoReplyResponse('3. Bayi baru sering menangis dan terlihat tidak bahagia, apakah ini pertanda dia bayi sulit ?')}
            style={styles.autoBtn}
          >
            <Text style={styles.autoBtnText}>3. Bayi baru sering menangis dan terlihat tidak bahagia, apakah ini pertanda dia bayi sulit ?</Text>
          </TouchableOpacity>

          <TouchableOpacity
            onPress={() => handleAutoReplyResponse('4. Mengapa bayi sering menggeliat ?')}
            style={styles.autoBtn}
          >
            <Text style={styles.autoBtnText}>4. Mengapa bayi sering menggeliat ?</Text>
          </TouchableOpacity>

          <TouchableOpacity
            onPress={() => handleAutoReplyResponse('5. Dapatkah bayi baru lahir mengenali suara ibunya ?')}
            style={styles.autoBtn}
          >
            <Text style={styles.autoBtnText}>5. Dapatkah bayi baru lahir mengenali suara ibunya ?</Text>
          </TouchableOpacity>
        </View>
      )}

      {item.type === 'image' && (
        <Image
```

```
          // source={{ uri:
`http://192.168.1.2/public/storage/${item.attachment}`
}}

          source={{ uri:
`https://balita.proyekk.com/public/storage/${item.attach
ment}` }}

          style={{ width: 200, height: 200, borderRadius:
8 }}

          resizeMode="cover"

         />

        )}



        {item.type === 'video' && (

         <Video

          // source={{ uri:
`http://192.168.1.2/public/storage/${item.attachment}`
}}

          source={{ uri:
`hhttps://balita.proyekk.com/public/storage/${item.attac
hment}` }}

          style={{ width: 300, height: 200 }}

          useNativeControls

          resizeMode="contain"

          shouldPlay={false}

         />

        )}

    </View>

  );

 };


 const handleGoBack = () => {

   navigation.goBack(); // Kembali ke halaman
sebelumnya

 };


 const handleRefresh = () => {

  setLoadingMessages(true);

  fetchMessages(); // Menyegarkan daftar pesan

 };


 return (

  <View style={{ flex: 1, padding: 16 }}>

   {/* Tombol Kembali */}

   <TouchableOpacity

    onPress={() => navigation.navigate('Main')}

    style={{

     marginBottom: 10,

     backgroundColor: '#007AFF',

     padding: 12,

     borderRadius: 8,

     alignItems: 'center',

    }}

   >

    <Text style={{ color: '#fff' }}>{"< "} Kembali ke
Dashboard</Text>

   </TouchableOpacity>



   {/* Tombol Refresh */}

   <TouchableOpacity

    onPress={handleRefresh}

    style={{

     marginBottom: 10,

     backgroundColor: '#28a745',

     padding: 12,

     borderRadius: 8,

     alignItems: 'center',

    }}

   >

    <Text style={{ color: '#fff' }}>🔄 Segarkan
Pesan</Text>

   </TouchableOpacity>


   <Text style={{ fontWeight: 'bold', marginBottom:
10, fontSize: 20, }}>
```

```
      Chat dengan {user.name}
    </Text>

    {/* Tombol Bersihkan Chat */}
    <TouchableOpacity
      onPress={() => {
        Alert.alert(
          'Konfirmasi',
          'Apakah kamu yakin ingin menghapus semua pesan?',
          [
            { text: 'Batal', style: 'cancel' },
            {
              text: 'Ya',
              onPress: async () => {
                try {
                  await clearChat(chatId);
                  setMessages([]);
                  Alert.alert('Berhasil', 'Chat berhasil dibersihkan');
                } catch (error) {
                  console.error('❌ Gagal bersihkan chat:', error);
                  Alert.alert('Gagal', 'Tidak dapat membersihkan chat');
                }
              },
            },
          ]
        );
      }}
      style={{
        marginBottom: 10,
        backgroundColor: '#dc3545',
        padding: 12,
        borderRadius: 8,
        alignItems: 'center',
      }}
    >
      <Text style={{ color: '#fff' }}>🗑️ Bersihkan Chat</Text>
    </TouchableOpacity>

    {/* Menampilkan pesan */}
    {loadingMessages ? (
      <ActivityIndicator />
    ) : (
      <FlatList
        data={messages}
        keyExtractor={(item) => item.id.toString()}
        renderItem={renderItem}
        contentContainerStyle={{ paddingBottom: 20 }}
      />
    )}

    <View style={{ flexDirection: 'row', marginTop: 12, alignItems: 'center' }}>
      <TextInput
        value={message}
        onChangeText={setMessage}
        placeholder="Tulis pesan..."
        style={{
          flex: 1,
          borderWidth: 1,
          borderColor: '#ccc',
          padding: 10,
          borderRadius: 8,
          marginRight: 8,
        }}
      />
      <TouchableOpacity
```

```
        onPress={handleSendText}
        style={{ backgroundColor: '#007AFF', padding: 12, borderRadius: 8 }}
      >
        <Text style={{ color: 'white' }}>Kirim</Text>
      </TouchableOpacity>
    </View>

    <TouchableOpacity
      onPress={pickMedia}
      style={{
        marginTop: 10,
        backgroundColor: '#28a745',
        padding: 12,
        borderRadius: 8,
        alignItems: 'center',
      }}
    >
      <Text style={{ color: '#fff' }}>📎 Kirim Gambar/Video</Text>
    </TouchableOpacity>
  </View>
 );
};

export default ChatDetailScreen;

import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, Image, ScrollView, Alert, Video } from 'react-native';
import tw from 'twrnc';
import * as ImagePicker from 'expo-image-picker';
import { createBayis } from '../api';
import { useNavigation } from '@react-navigation/native';

const BayiTambah = () => {
 const [judul, setJudul] = useState('');
 const [isi, setIsi] = useState('');
 const [foto, setFoto] = useState(null);
 const [video, setVideo] = useState(null);
 const navigation = useNavigation();

 const pickImage = async () => {
  const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
  if (status !== 'granted') {
   alert('Izin akses galeri diperlukan!');
   return;
  }

  let result = await ImagePicker.launchImageLibraryAsync({
   mediaTypes: ImagePicker.MediaTypeOptions.Images,
   allowsEditing: true,
   aspect: [4, 3],
   quality: 1,
  });

  if (!result.canceled) {
   setFoto(result.assets[0]);
  }
 };

 const pickVideo = async () => {
  const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
  if (status !== 'granted') {
   alert('Izin akses galeri diperlukan!');
   return;
  }
```

```
    let result = await
ImagePicker.launchImageLibraryAsync({
      mediaTypes:
ImagePicker.MediaTypeOptions.Videos,
      allowsEditing: false,
      quality: 1,
    });

    if (!result.canceled) {
      setVideo(result.assets[0]);
    }
  };

  const handleSubmit = async () => {
    if (!judul || !isi) {
      Alert.alert('Validasi', 'Judul dan Isi wajib diisi!');
      return;
    }

    const data = {
      judul: judul,
      isi: isi,
      foto: foto,   // foto boleh null
      video: video, // video juga boleh null
    };

    try {
      await createBayis(data);
      Alert.alert('Sukses', 'Artikel berhasil dibuat');
      navigation.goBack();
    } catch (error) {
      console.error('Gagal menambahkan artikel:',
error.response?.data || error.message);
      Alert.alert('Error', error.message || 'Gagal membuat
artikel');
    }
  };

  return (
    <ScrollView contentContainerStyle={tw`p-4 bg-
gray-100`}>
      <View style={tw`mb-4`}>
        <Text style={tw`text-lg font-bold mb-2`}>Judul
Artikel</Text>
        <TextInput
          value={judul}
          onChangeText={setJudul}
          placeholder="Masukkan judul"
          style={tw`bg-white p-3 rounded-lg shadow`}
        />
      </View>

      <View style={tw`mb-4`}>
        <Text style={tw`text-lg font-bold mb-2`}>Isi
Artikel</Text>
        <TextInput
          value={isi}
          onChangeText={setIsi}
          placeholder="Masukkan isi artikel"
          style={tw`bg-white p-3 rounded-lg shadow h-
32`}
          multiline
        />
      </View>

      {/* FOTO */}
      <View style={tw`mb-4`}>
        <Text style={tw`text-lg font-bold mb-2`}>Foto
(opsional)</Text>
        {foto && (
          <Image
            source={{ uri: foto.uri }}
```

```
          style={tw`w-full h-48 mb-2 rounded-lg`}
          resizeMode="cover"
        />
      )}
      <TouchableOpacity
        style={tw`bg-blue-500 p-3 rounded-lg`}
        onPress={pickImage}
      >
        <Text style={tw`text-white text-center`}>
          {foto ? 'Ganti Foto' : 'Pilih Foto'}
        </Text>
      </TouchableOpacity>
    </View>

    {/* VIDEO */}
    <View style={tw`mb-4`}>
      <Text style={tw`text-lg font-bold mb-2`}>Video (opsional)</Text>
      {video && (
        <Text style={tw`text-green-600 mb-2`}>
          Video dipilih: {video.fileName ?? 'Video'}
        </Text>
      )}
      <TouchableOpacity
        style={tw`bg-purple-500 p-3 rounded-lg`}
        onPress={pickVideo}
      >
        <Text style={tw`text-white text-center`}>
          {video ? 'Ganti Video' : 'Pilih Video'}
        </Text>
      </TouchableOpacity>
    </View>

    <TouchableOpacity
      style={tw`bg-green-500 p-4 rounded-lg`}
      onPress={handleSubmit}
    >
      <Text style={tw`text-white text-center text-lg font-bold`}>Simpan Artikel</Text>
    </TouchableOpacity>
  </ScrollView>
 );
};

export default BayiTambah;

import React, { useState, useEffect } from 'react';
import { View, Text, TextInput, TouchableOpacity, Image, ScrollView, Alert } from 'react-native';
import tw from 'twrnc';
import * as ImagePicker from 'expo-image-picker';
import { updateBayis, getBayis } from '../api';
import { useNavigation, useRoute } from '@react-navigation/native';
import { Video } from 'expo-av';

const BayiEdit = () => {
 const [judul, setJudul] = useState('');
 const [isi, setIsi] = useState('');
 const [foto, setFoto] = useState(null);
 const [oldFoto, setOldFoto] = useState(null);
 const [video, setVideo] = useState(null);
const [oldVideo, setOldVideo] = useState(null);
 const route = useRoute();
 const navigation = useNavigation();

 useEffect(() => {
  const fetchBayi = async () => {
   try {
    const bayi = await getBayis(route.params.id);
    setJudul(bayi.judul);
    setIsi(bayi.isi);
```

```
      setOldFoto(bayi.foto);
      setOldVideo(bayi.video); // ambil video lama juga
    } catch (error) {
      console.error('Gagal mengambil artikel:', error);
      Alert.alert('Error', 'Gagal mengambil artikel');
    }
  };

  fetchBayi();
}, [route.params.id]);

// Fungsi untuk memilih gambar
const pickImage = async () => {
  const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
  if (status !== 'granted') {
    alert('Izin akses galeri diperlukan!');
    return;
  }

  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.Images,
    allowsEditing: true,
    aspect: [4, 3],
    quality: 1,
  });

  if (!result.canceled) {
    setFoto(result.assets[0]);  // Menyimpan file gambar yang dipilih
  }
};

const pickVideo = async () => {
  const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
  if (status !== 'granted') {
    alert('Izin akses galeri diperlukan!');
    return;
  }

  let result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.Videos,
    allowsEditing: true,
    quality: 1,
  });

  if (!result.canceled) {
    setVideo(result.assets[0]);
  }
};

// Fungsi untuk mengirim data Bayi
const handleSubmit = async () => {
  if (!judul || !isi) {
    Alert.alert('Validasi', 'Judul dan Isi wajib diisi!');
    return;
  }

  try {
    await updateBayis(route.params.id, judul, isi, foto, oldFoto, video, oldVideo);
    Alert.alert('Sukses', 'Artikel berhasil diupdate');
    navigation.goBack();
  } catch (error) {
    console.error('Gagal mengupdate artikel:', error.response || error.message);
    Alert.alert('Error', error.message || 'Gagal mengupdate artikel');
```

```
  }
 };

 return (
  <ScrollView contentContainerStyle={tw`p-4 bg-gray-100`}>
   <View style={tw`mb-4`}>
    <Text style={tw`text-lg font-bold mb-2`}>Judul Bayi</Text>
    <TextInput
     value={judul}
     onChangeText={setJudul}
     placeholder="Masukkan judul"
     style={tw`bg-white p-3 rounded-lg shadow`}
    />
   </View>

   <View style={tw`mb-4`}>
    <Text style={tw`text-lg font-bold mb-2`}>Isi Bayi</Text>
    <TextInput
     value={isi}
     onChangeText={setIsi}
     placeholder="Masukkan isi Bayi"
     style={tw`bg-white p-3 rounded-lg shadow h-32`}
     multiline
    />
   </View>

   <View style={tw`mb-4`}>
    <Text style={tw`text-lg font-bold mb-2`}>Foto (opsional)</Text>
    {foto ? (
     <Image
      source={{ uri: foto.uri }}
      style={tw`w-full h-48 mb-2 rounded-lg`}
      resizeMode="cover"
     />
    ) : oldFoto ? (
     <Image
      source={{ uri: oldFoto }}
      style={tw`w-full h-48 mb-2 rounded-lg`}
      resizeMode="cover"
     />
    ) : null}
    <TouchableOpacity
     style={tw`bg-blue-500 p-3 rounded-lg`}
     onPress={pickImage}
    >
     <Text style={tw`text-white text-center`}>{foto ? 'Ganti Foto' : 'Pilih Foto'}</Text>
    </TouchableOpacity>
   </View>
   <View style={tw`mb-4`}>
  <Text style={tw`text-lg font-bold mb-2`}>Video (opsional)</Text>

  {video ? (
   <Video
    source={{ uri: video.uri }}
    style={tw`w-full h-48 mb-2 rounded-lg`}
    useNativeControls
    resizeMode="contain"
   />
  ) : oldVideo ? (
   <Video
    source={{ uri: oldVideo }}
    style={tw`w-full h-48 mb-2 rounded-lg`}
    useNativeControls
    resizeMode="contain"
   />
```

```
 ) : null}


 <TouchableOpacity

  style={tw`bg-purple-500 p-3 rounded-lg`}

  onPress={pickVideo}

 >

  <Text style={tw`text-white text-center`}>{video ?
'Ganti Video' : 'Pilih Video'}</Text>

 </TouchableOpacity>

</View>



    <TouchableOpacity

     style={tw`bg-green-500 p-4 rounded-lg`}

     onPress={handleSubmit}

    >

     <Text style={tw`text-white text-center text-lg font-
bold`}>Simpan Bayi</Text>

    </TouchableOpacity>

  </ScrollView>

 );

};


export default BayiEdit;


import React, { useState, useEffect } from 'react';

import { View, Text, Image, ScrollView,
ActivityIndicator, Alert, TouchableOpacity } from
'react-native';

import tw from 'twrnc';

import { useRoute, useNavigation } from '@react-
navigation/native';

import { getBayis } from '../api';

import { Video } from 'expo-av';


const BayiDetail = () => {

  const [bayi, setBayi] = useState(null);

  const [loading, setLoading] = useState(true);

  const route = useRoute();

  const navigation = useNavigation(); // Untuk navigasi
kembali

  const { id } = route.params;


  useEffect(() => {

    const fetchBayi = async () => {

      try {

        const data = await getBayis(id); // Mengambil
Bayi berdasarkan ID

        setBayi(data);

      } catch (error) {

        console.error('Gagal memuat Bayi:', error); //
Menampilkan log error

        Alert.alert('Error', 'Gagal memuat Bayi!'); //
Menampilkan alert error

      } finally {

        setLoading(false); // Menghentikan loading
setelah request selesai

      }

    };


    fetchBayi();

  }, [id]);


  const handleGoBack = () => {

    navigation.goBack(); // Kembali ke halaman
sebelumnya (dashboard)

  };


  if (loading) {

    return (

      <View style={tw`flex-1 justify-center items-
center`}>

        <ActivityIndicator size="large"
color="#3B82F6" />

      </View>
```

```
    );
  }

  if (!bayi) {
    return (
      <View style={tw`flex-1 justify-center items-center`}>
        <Text style={tw`text-lg text-red-500`}>Bayi tidak ditemukan</Text>
      </View>
    );
  }

  return (
    <ScrollView style={tw`bg-gray-100`} contentContainerStyle={tw`p-4`}>
      {/* Tombol Kembali */}
      <TouchableOpacity onPress={handleGoBack} style={tw`mb-4 flex-row items-center`}>
        <Text style={tw`text-blue-500 text-lg`}>{"< "} Kembali ke Dashboard</Text>
      </TouchableOpacity>

      {/* Render Foto Bayi */}
      {bayi.foto && (
        <Image
          source={{ uri: `https://balita.proyekk.com/public/storage/${bayi.foto}` }} // Pastikan URL ini valid

          // source={{ uri: `http://192.168.1.2/public/storage/${bayi.foto}` }}
          style={tw`w-full h-60 rounded-lg mb-4`}
          resizeMode="cover"
        />
      )}

      {/* Render Video Bayi */}
      {bayi.video && (
        <Video
          source={{ uri: `https://balita.proyekk.com/public/storage/${bayi.video}` }}
          // source={{ uri: `http://192.168.1.2/public/storage/${bayi.video}` }}
          style={{ width: '100%', height: 200 }}
          useNativeControls
          resizeMode="contain"
          shouldPlay={false}
        />
      )}

      {/* Judul Bayi */}
      <Text style={tw`text-3xl font-bold mb-2`}>{bayi.judul}</Text>

      {/* Isi Bayi */}
      <Text style={tw`text-base text-xl text-gray-700 leading-relaxed`}>{bayi.isi}</Text>
    </ScrollView>
  );
};

export default BayiDetail;

import React, { useState } from 'react';
import { View, Text, TextInput, TouchableOpacity, Image, ScrollView, Alert } from 'react-native';
import tw from 'twrnc';
import * as ImagePicker from 'expo-image-picker';
import { createArtikel } from '../api'; // Pastikan ada function createArtikel di api.js
import { useNavigation } from '@react-navigation/native';

const ArtikelTambah = () => {
```

```
  const [judul, setJudul] = useState('');
  const [isi, setIsi] = useState('');
  const [foto, setFoto] = useState(null);
  const navigation = useNavigation();

  const pickImage = async () => {
    const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
    if (status !== 'granted') {
      alert('Izin akses galeri diperlukan!');
      return;
    }

    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.Images,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });

    if (!result.canceled) {
      setFoto(result.assets[0]);
    }
  };

  const handleSubmit = async () => {
    if (!judul || !isi) {
      Alert.alert('Validasi', 'Judul dan Isi wajib diisi!');
      return;
    }

    const data = {
      judul: judul,
      isi: isi,
      foto: foto, // foto bisa null, tidak masalah
    };

    try {
      await createArtikel(data);
      Alert.alert('Sukses', 'Artikel berhasil dibuat');
      navigation.goBack();
    } catch (error) {
      console.error('Gagal menambahkan artikel:', error.response?.data || error.message);
      Alert.alert('Error', error.message || 'Gagal membuat artikel');
    }
  };

  return (
    <ScrollView contentContainerStyle={tw`p-4 bg-gray-100`}>
      <View style={tw`mb-4`}>
        <Text style={tw`text-lg font-bold mb-2`}>Judul Artikel</Text>
        <TextInput
          value={judul}
          onChangeText={setJudul}
          placeholder="Masukkan judul"
          style={tw`bg-white p-3 rounded-lg shadow`}
        />
      </View>

      <View style={tw`mb-4`}>
        <Text style={tw`text-lg font-bold mb-2`}>Isi Artikel</Text>
        <TextInput
          value={isi}
          onChangeText={setIsi}
```

```
        placeholder="Masukkan isi artikel"
        style={tw`bg-white p-3 rounded-lg shadow h-32`}
        multiline
      />
    </View>

    <View style={tw`mb-4`}>
      <Text style={tw`text-lg font-bold mb-2`}>Foto (opsional)</Text>
      {foto && (
        <Image
          source={{ uri: foto.uri }}
          style={tw`w-full h-48 mb-2 rounded-lg`}
          resizeMode="cover"
        />
      )}
      <TouchableOpacity
        style={tw`bg-blue-500 p-3 rounded-lg`}
        onPress={pickImage}
      >
        <Text style={tw`text-white text-center`}>{foto ? 'Ganti Foto' : 'Pilih Foto'}</Text>
      </TouchableOpacity>
    </View>

    <TouchableOpacity
      style={tw`bg-green-500 p-4 rounded-lg`}
      onPress={handleSubmit}
    >
      <Text style={tw`text-white text-center text-lg font-bold`}>Simpan Artikel</Text>
    </TouchableOpacity>
  </ScrollView>
 );
};

export default ArtikelTambah;


import React, { useState, useEffect } from 'react';
import { View, Text, TextInput, TouchableOpacity, Image, ScrollView, Alert } from 'react-native';
import tw from 'twrnc';
import * as ImagePicker from 'expo-image-picker';
import { updateArtikel, getArtikel } from '../api';
import { useNavigation, useRoute } from '@react-navigation/native';

const ArtikelEdit = () => {
 const [judul, setJudul] = useState('');
 const [isi, setIsi] = useState('');
 const [foto, setFoto] = useState(null);
 const [oldFoto, setOldFoto] = useState(null);
 const route = useRoute();
 const navigation = useNavigation();

 useEffect(() => {
  const fetchArtikel = async () => {
   try {
    const artikel = await getArtikel(route.params.id);
    setJudul(artikel.judul);
    setIsi(artikel.isi);
    setOldFoto(artikel.foto); // Set foto lama jika ada
   } catch (error) {
    console.error('Gagal mengambil artikel:', error);
    Alert.alert('Error', 'Gagal mengambil artikel');
   }
  };

  fetchArtikel();
 }, [route.params.id]);
```

```
  // Fungsi untuk memilih gambar
  const pickImage = async () => {
    const { status } = await ImagePicker.requestMediaLibraryPermissionsAsync();
    if (status !== 'granted') {
      alert('Izin akses galeri diperlukan!');
      return;
    }

    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.Images,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });

    if (!result.canceled) {
      setFoto(result.assets[0]);  // Menyimpan file gambar yang dipilih
    }
  };

  // Fungsi untuk mengirim data artikel
  const handleSubmit = async () => {
    if (!judul || !isi) {
      Alert.alert('Validasi', 'Judul dan Isi wajib diisi!');
      return;
    }

    try {
      // Mengirim data artikel ke API
      await updateArtikel(route.params.id, judul, isi, foto, oldFoto);
      Alert.alert('Sukses', 'Artikel berhasil diupdate');
      navigation.goBack();
    } catch (error) {
      console.error('Gagal mengupdate artikel:', error.response || error.message);
      Alert.alert('Error', error.message || 'Gagal mengupdate artikel');
    }
  };

  return (
    <ScrollView contentContainerStyle={tw`p-4 bg-gray-100`}>
      <View style={tw`mb-4`}>
        <Text style={tw`text-lg font-bold mb-2`}>Judul Artikel</Text>
        <TextInput
          value={judul}
          onChangeText={setJudul}
          placeholder="Masukkan judul"
          style={tw`bg-white p-3 rounded-lg shadow`}
        />
      </View>

      <View style={tw`mb-4`}>
        <Text style={tw`text-lg font-bold mb-2`}>Isi Artikel</Text>
        <TextInput
          value={isi}
          onChangeText={setIsi}
          placeholder="Masukkan isi artikel"
          style={tw`bg-white p-3 rounded-lg shadow h-32`}
          multiline
        />
      </View>

      <View style={tw`mb-4`}>
```

```
      <Text style={tw`text-lg font-bold mb-2`}>Foto (opsional)</Text>
      {foto ? (
        <Image
          source={{ uri: foto.uri }}
          style={tw`w-full h-48 mb-2 rounded-lg`}
          resizeMode="cover"
        />
      ) : oldFoto ? (
        <Image
          source={{ uri: oldFoto }}
          style={tw`w-full h-48 mb-2 rounded-lg`}
          resizeMode="cover"
        />
      ) : null}
      <TouchableOpacity
        style={tw`bg-blue-500 p-3 rounded-lg`}
        onPress={pickImage}
      >
        <Text style={tw`text-white text-center`}>{foto ? 'Ganti Foto' : 'Pilih Foto'}</Text>
      </TouchableOpacity>
    </View>

    <TouchableOpacity
      style={tw`bg-green-500 p-4 rounded-lg`}
      onPress={handleSubmit}
    >
      <Text style={tw`text-white text-center text-lg font-bold`}>Simpan Artikel</Text>
    </TouchableOpacity>
  </ScrollView>
 );
};

export default ArtikelEdit;
```

```
import React, { useState, useEffect } from 'react';
import { View, Text, Image, ScrollView, ActivityIndicator, Alert, TouchableOpacity } from 'react-native';
import tw from 'twrnc';
import { useRoute, useNavigation } from '@react-navigation/native';
import { getArtikel } from '../api';
import * as Speech from 'expo-speech'; // Import expo-speech

const ArtikelDetail = () => {
 const [artikel, setArtikel] = useState(null);
 const [loading, setLoading] = useState(true);
 const [isSpeaking, setIsSpeaking] = useState(false); // Untuk melacak status bicara
 const route = useRoute();
 const navigation = useNavigation();
 const { id } = route.params;

 useEffect(() => {
  const fetchArtikel = async () => {
   try {
    const data = await getArtikel(id);
    setArtikel(data);
   } catch (error) {
    console.error('Gagal memuat artikel:', error);
    Alert.alert('Error', 'Gagal memuat artikel!');
   } finally {
    setLoading(false);
   }
  };

  fetchArtikel();
 }, [id]);
```

```
  const handleGoBack = () => {
    navigation.goBack();
  };

  const handlePlay = () => {
    if (artikel?.isi) {
      Speech.speak(artikel.isi); // Mulai membaca isi artikel
      setIsSpeaking(true); // Tandai sedang berbicara
    }
  };

  const handlePause = () => {
    Speech.stop(); // Berhenti berbicara
    setIsSpeaking(false); // Tandai berhenti berbicara
  };

  if (loading) {
    return (
      <View style={tw`flex-1 justify-center items-center`}>
        <ActivityIndicator size="large" color="#3B82F6" />
      </View>
    );
  }

  if (!artikel) {
    return (
      <View style={tw`flex-1 justify-center items-center`}>
        <Text style={tw`text-lg text-red-500`}>Artikel tidak ditemukan</Text>
      </View>
    );
  }

  return (
    <ScrollView style={tw`bg-gray-100`} contentContainerStyle={tw`p-4`}>
      <TouchableOpacity onPress={handleGoBack} style={tw`mb-4 flex-row items-center`}>
        <Text style={tw`text-blue-500 text-lg`}>{"< "} Kembali ke Dashboard</Text>
      </TouchableOpacity>

      {artikel.foto && (
        <Image
          source={{ uri: `https://balita.proyekk.com/public/storage/${artikel.foto}` }}

          // source={{ uri: `http://192.168.1.2/public/storage/${artikel.foto}` }}
          style={tw`w-full h-60 rounded-lg mb-4`}
          resizeMode="cover"
        />
      )}

      <Text style={tw`text-3xl font-bold mb-2`}>{artikel.judul}</Text>
      <Text style={tw`text-base text-xl text-gray-700 leading-relaxed`}>{artikel.isi}</Text>

      {/* Tombol Play dan Pause */}
      <View style={tw`mt-4 flex-row justify-between`}>
        <TouchableOpacity onPress={handlePlay} style={tw`bg-blue-500 p-3 rounded`}>
          <Text style={tw`text-white`}>Play</Text>
        </TouchableOpacity>
        <TouchableOpacity onPress={handlePause} style={tw`bg-red-500 p-3 rounded`}>
          <Text style={tw`text-white`}>Pause</Text>
        </TouchableOpacity>
      </View>
    </ScrollView>
```

```
  );
};

export default ArtikelDetail;
```