

BAB I

PENDAHULUAN

A. Latar Belakang

Seiring berjalannya perkembangan teknologi komputer, teknologi perangkat lunak juga berkembang. Salah satu teknologi perangkat lunak yang berkembang adalah aplikasi game. Game adalah sebuah aktivitas menarik yang dibutuhkan manusia untuk bersantai atau mengisi waktu luang. Aplikasi game sudah bermunculan berbagai macam jenis. Mulai dari game kartu, papan game, sampai game-game lainnya yang lebih banyak. Aplikasi game ini dapat dimainkan, baik itu dari kalangan muda hingga orang-orang dewasa (Ilham, A., 2008).

Semua perkembangan game itu diperlukan sebuah kecerdasan yang memungkinkan komputer untuk melawan pemain. Cabang ilmu teknik komputer yang menciptakan untuk menirukan kecerdasan manusia untuk yang bukan pemain adalah kecerdasan buatan atau *Artificial Intelligence (A.I.)*. Salah satu kelebihanannya adalah kita tidak harus mencari orang untuk menjadi lawan tanding karena ini sudah mendukung *single-player mode* dimana kita dapat bermain sendiri melawan komputer atau vs AI (Salmon, R., & Nugraha, I., 2017).

Game *Checkers* merupakan salah satu permainan papan game yang menggunakan strategi, taktik, dan bidak sebagai pemainnya. Game *Checkers* yang dapat dimainkan oleh dua orang, yang dilakukan dipapan game 8x8 kotak dengan ukuran yang berbeda, yang disesuaikan dengan ukuran bidak. Salah satu pemain menggunakan bidak warna hitam dan yang satunya menggunakan bidak warna

terang. Game dimulai dengan bidak warna hitam yang jalan terlebih dahulu. Bidak hanya dapat bergerak secara diagonal dan hanya dapat menangkap musuh dengan meloncatinya secara diagonal dan musuh yang tertangkap harus dihilangkan dari papan game. Dalam game ini setiap pemain dapat mengetahui semua kondisi permainan, seperti jumlah bidak yang dimiliki oleh pemain dan lawan, posisi bidak, dan langkah-langkah apa saja yang digunakan oleh pemain maupun lawan (Effendi, A. K., Delima, R., & Chrismanto, A. R., 2012).

Berdasarkan uraian latar belakang diatas, maka dalam penelitian ini penulis mengusulkan dengan judul “**Aplikasi Game *Checkers* Dengan Menggunakan Metode *Alpha Beta* Dan *Minimax***”.

B. Rumusan Masalah

Berdasarkan uraian latar belakang yang telah diuraikan, maka akan dibuat suatu penelitian pada masalah yang sedang dihadapi. Adapun latar belakang masalah yang dapat diidentifikasi sebagai berikut :

1. Bagaimana menganalisis, dan membangun Aplikasi Game *Checkers* Dengan Menggunakan Metode *Alpha Beta* Dan *Minimax*?
2. Bagaimana menerapkan suatu aplikasi game menggunakan algoritma *Alpha Beta* dan *Minimax*?

C. Batasan Masalah

Adapun agar pembahasan menjadi lebih terarah dan tidak menyimpang dan juga sesuai dengan latar belakang yang sudah di uraikan sebelumnya, maka penulis membatasi permasalahan penelitian pada suatu Aplikasi Game *Checkers* Dengan Menggunakan Metode *Alpha Beta* dan *Minimax*, antara lain :

1. Aplikasi ini digunakan hanya untuk permainan game *Checkers*.
2. Aplikasi *game checkers* ini memuat tiga menu utama yaitu bermain, petunjuk dan keluar.
3. Tidak dilakukan pembahasan terhadap permainan *multiplayer* secara *online*.
4. Aplikasi ini dibangun menggunakan bahasa pemrograman PHP, HTML dan Javascript.
5. Aplikasi ini dirancang dan dibangun menggunakan algoritma *Alpha Beta* dan *Minimax*.
6. Perancangan aplikasi game *Checkers* ini hanya menggunakan berbasis Web Server.

D. Tujuan Penelitian

Berdasarkan rumusan masalah yang telah didefinisikan sebelumnya tujuan dari penelitian dan perancangan sistem dalam penelitian ini adalah :

1. Merancang dan membangun Aplikasi Game *Checkers* Dengan Menggunakan Algoritma *Alpha Beta* Dan *Minimax*.
2. Menerapkan suatu aplikasi game menggunakan algoritma *Alpha Beta* dan *Minimax*.

E. Manfaat Penelitian

Salah satu fungsi dari penelitian adalah bisa memberikan manfaat dan kegunaan yang secara nyata akan diterima oleh berbagai pihak yang terkait dengan obyek penelitian seperti aspek akademis dan praktis. Dalam penelitian kali ini peneliti berharap bisa memberikan kegunaan, antara lain :

1. Aspek Akademis

- a. Untuk mengaplikasikan teori yang di dapat di bangku perkuliahan dengan kondisi *real* di lapangan, khususnya terkait dengan permasalahan yang dibahas terkait penelitian dan *tools* atau metode penyelesaian masalah.
- b. Diharapkan Sebagai bahan masukan bagi para pengkaji ilmu komunikasi di bidang *Public Relations* yang berminat untuk meneliti masalah yang sama.

2. Aspek Praktis

- a. Dapat sebagai penerapan ilmu yang telah didapat selama kuliah terutama dalam bidang Informatika.
- b. Memberi pengetahuan tentang cara bermain *checkers* dengan melawan *vs AI* pada *single-player mode*.
- c. Bagi peneliti diharapkan dapat memberikan pengalaman dan menerapkan ilmu pengetahuan yang diterima selama mengikuti perkuliahan.
- d. Sebagai tugas akhir untuk mendapatkan gelar sarjana strata 1 (S.1) pada Fakultas Teknik, Program Studi Informatika, Universitas Muhammadiyah Parepare.

F. Sistematika Penulisan

Sistematika penulisan skripsi ini dibagi ke dalam lima bab yang masing-masing menjelaskan bagian-bagian yang berbeda namun tetap memiliki korelasi satu dengan yang lain. Setiap bab-nya terdiri dari beberapa sub tersendiri. Diawali dengan bab pendahuluan dan di akhiri dengan bab penutup yang berisi kesimpulan dan saran-saran. Secara singkat akan diuraikan sebagai berikut :

BAB I : PENDAHULUAN

Pada bab ini akan dijelaskan tentang latar belakang penulisan, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Bab ini berisi uraian tentang landasan teori-teori dari perancangan “**Aplikasi Game Checkers Dengan Menggunakan Metode Alpha Beta Dan Minimax**”. Bab ini juga membahas mengenai berbagai teori yang mendasari dari analisis permasalahan yang berhubungan dengan topik yang dibahas.

BAB III : METODOLOGI PENELITIAN

Dalam bab ini membahas mengenai metode penelitian yang digunakan dalam merancang dan membangun sistem. Pada bab ini juga membahas tentang identifikasi masalah, analisis kebutuhan aplikasi yang meliputi kebutuhan masukan, proses dan keluaran aplikasi yang akan dibangun. Penjelasan mengenai proses perancangan dengan menggunakan rancangan dan *flowchart*.

BAB IV : HASIL DAN PEMBAHASAN

Pada bab ini membahas mengenai hasil dari tahap analisis dan perancangan sesuai dengan metode yang digunakan pada alat dan sistem yang dibuat serta hasil dari pengujian. Pada bab ini berisi tentang implementasi dan pengujian dua unit testing.

BAB V : PENUTUP

Pada bab akhir ini berisi tentang kesimpulan dari penelitian yang telah dilakukan dan saran yang diberikan untuk pengembangan lebih lanjut agar tercapai hasil yang lebih maksimal.

DAFTAR PUSTAKA

Pada bagian ini akan dipaparkan tentang sumber-sumber literatur, tutorial, buku maupun situs-situs yang digunakan dalam pembuatan laporan tugas akhir ini.

LAMPIRAN

BAB II

TINJAUAN PUSTAKA

A. Penelitian Terdahulu

Berbagai penelitian sebelumnya merupakan hal yang sangat perlu dan dapat dijadikan sebagai data pendukung. Salah satu data pendukung yang menurut peneliti perlu dijadikan bagian tersendiri adalah penelitian terdahulu yang relevan dengan permasalahan yang sedang dibahas dalam penelitian ini. Oleh karena itu, peneliti melakukan kajian terhadap beberapa hasil penelitian berupa tugas akhir/skripsi atau jurnal melalui *internet*.

Tabel 2.1 Penelitian Terdahulu

NO	JUDUL PENELITIAN, TAHUN	METODE	HASIL
1	Pembuatan Aplikasi Permainan <i>Othello</i> 16x16 Berbasis Desktop Dengan Algoritma <i>Alpha Beta Pruning</i> (Nugroho Agus, 2020)	<i>Alpha Beta Pruning</i>	Dalam penelitian ini, peneliti akan mengimplementasikan algoritma <i>Alpha Beta Pruning</i> . Algoritma <i>Alpha Beta Pruning</i> dapat memangkas pohon permainan secara efisien, dan hasil yang dilakukan oleh algoritma ini merupakan langkah terbaik.
2	Penerapan Algoritma <i>Minimax</i> Dan <i>Alpha Beta Pruning</i> Pada Game <i>Connect Four</i> Berbasis Android (Gunawan, L. 2019)	<i>Minimax Dan Alpha Beta Pruning</i>	Dalam penelitian ini, peneliti menggunakan algoritma <i>Minimax</i> Dan <i>Alpha Beta Pruning</i> diterapkan untuk mengkalkulasikan Langkah terbaik bagi AI dengan memangkas kemungkinan node yang tidak mengandung

Tabel 2.1 Penelitian Terdahulu

NO	JUDUL PENELITIAN, TAHUN	METODE	HASIL
			langkah terbaik pada pohon permainan. Berdasarkan hasil pengujian fungsionalitas aplikasi yang dilakukan 5 orang responden, Tingkat keberhasilan aplikasi sebesar 100% dan berjalan dengan baik pada perangkat pemilik.
3	Algoritma Minimax Dalam Permainan Checkers (Nadhira Ayuningtyas, 2008)	Algoritma Minimax	Dalam Penelitian ini, peneliti menerapkan algortima <i>Minimax</i> . Algoritma <i>Minimax</i> memiliki dasar berupa <i>zero – sum game</i> , dimana jika pemain mendapatkan nilai tertentu maka pemain lain akan kehilangan nilai yang sama dengan pemain tersebut. Dengan menambahkan nilai <i>Alpha – beta</i> , algoritma <i>Minimax</i> akan memiliki pohon pencarian yang lebih singkat sehingga akan membutuhkan waktu singkat untuk melakukan aksinya. Berdasarkan hasil dari penelitian ini disimpulkan bahwa dengan menambahkan fungsi heuristik yang tepat pengambilan langkah yang dilakukan oleh <i>A.I.</i> akan memberikan hasil pendapatan yang lebih baik. Sehingga penerapan algoritma akan mempersulit pemain untuk melawan <i>A.I.</i> , karena <i>A.I.</i> akan mengevaluasi setiap langkah agar pendapatan yang diambil olehnya maksimum dengan memperhitungkan seluruh kemungkinan langkah pemain sehingga langkah pemain akan menghasilkan nilai minimum.

Penelitian Terdahulu dengan penulis yaitu :

1. Dalam penelitian ini terdapat perbedaan dimana peneliti membuat aplikasi permainan *Othello*, sedangkan penulis membuat aplikasi game *Checkers* serta dari sisi metode yang digunakan peneliti terdahulu ialah *Alpha – Beta Prunning* sedangkan penulis menggunakan *Alpha – Beta* dan *Minimax*. Adapun sisi perbedaan yang digunakan dalam merancang aplikasi dari penelitian terdahulu ialah berbasis desktop sedangkan penulis merancang berbasis web.
2. Adapun terdapat perbedaan penelitian ini dimana peneliti menggunakan Game *Connect Four*, sedangkan penulis menggunakan game *Checkers* serta dari sisi teknologi menggunakan aplikasi android sedangkan penulis menggunakan web. Adapun persamaan metode yang digunakan ialah *Alpha – Beta* dan *Minimax*.
3. Dari penelitian ini terdapat perbedaan yaitu peneliti menggunakan metode *Minimax*, sedangkan penulis menggunakan *Alpha – Beta* dan *Minimax* serta adapun persamaan penelitian terdahulu dengan penulis saat ini ialah sama – sama membuat aplikasi game *Checkers*.

B. Tinjauan Teori

1. Aplikasi

Aplikasi merupakan program siap pakai yang dapat membantu penggunanya dan melaksanakan suatu tugas khusus fungsi bagi penggunanya serta dapat digunakan sesuai dengan kebutuhan.

Menurut (Kadir, 2008), program aplikasi adalah program siap pakai atau program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi

yang lain. Aplikasi juga diartikan sebagai penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan atau sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu.

2. Game

Game berasal dari bahasa Inggris yang berarti permainan. Game secara umum adalah sebuah aktivitas rekreasi dengan tujuan mengisi waktu luang, bersenang-senang, atau berolahraga ringan. Bermain game biasa dilakukan sendiri atau Bersama-sama.

Game memiliki beberapa peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Game menurut beberapa para ahli sebagai berikut :

a. Samuel Hendry

Game merupakan bagian tak terpisahkan dari keseharian anak, sedangkan sebagian orang tua menuding game sebagai penyebab nilai anak turun, anak tak mampu bersosialisasi, dan tindakan kekerasan yang dilakukan anak.

b. Ivan C. Sibero

Game merupakan aplikasi yang paling banyak digunakan dan dinikmati para pengguna media elektronik saat ini.

c. Albert Einstein

Game adalah bentuk investigasi paling tinggi.

d. Mitchell Wade

Game adalah lingkungan pelatihan yang baik bagi dunia nyata dalam organisasi yang menuntut pemecahan masalah secara kolaborasi.

e. John Naisbitt

Game merupakan sistem partisipatoris dinamis karena game memiliki tingkat penceritaan yang tidak dimiliki film.

f. Fauzi A.

Game merupakan suatu bentuk hiburan yang sering kali dijadikan sebagai penyegar pikiran dari rasa lelah yang disebabkan oleh aktivitas dan rutinitas kita.

3. *Checkers*

Menurut (Aditya Kurniawan Efendi, 2012) *Checkers* atau dam adalah sebuah papan menyerupai papan catur yang pertama kali dimainkan sekitar 3000 tahun sebelum masehi. Game *checkers* dimainkan dipapan yang digunakan berukuran 8x8 (dengan jumlah total 64 kotak) dengan 12 bidak pada setiap sisi (total bidak 24).



Gambar 2.1 Papan *Checkers*

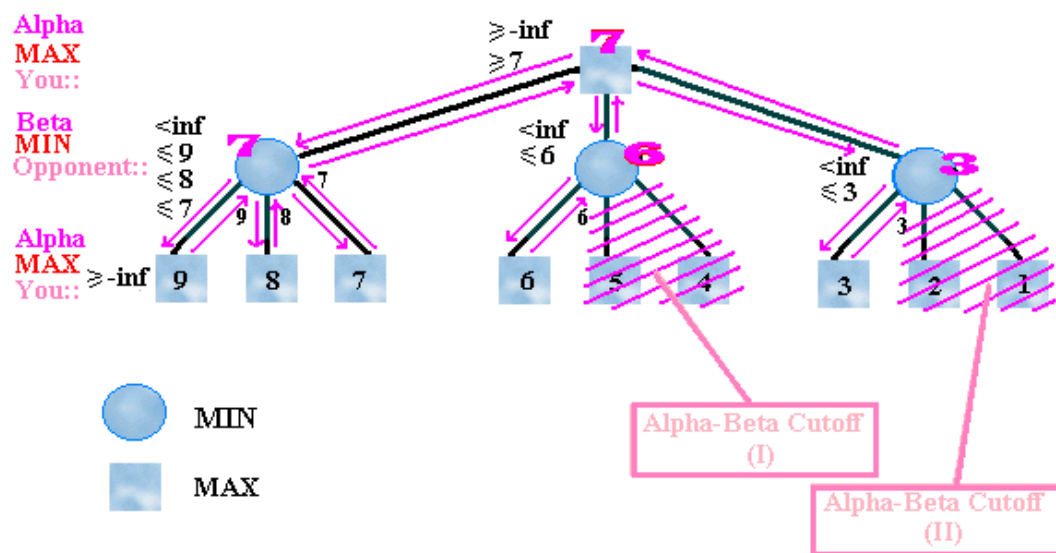
Berdasarkan buku *How to Win at Checkers*, setiap pemain Menyusun 12 bidak pada papan permainan pada kotak hitam di 3 baris paling dekat dengan pemain. Baris paling dekat dengan pemain disebut dengan *crownhead* atau *king's row*. Pemain dengan bidak warna hitam jalan terlebih dahulu. Ada dua jenis gerakan dalam game *checkers* ini yaitu :

- a. *Simple move*, yaitu gerakan diagonal satu langkah ke kotak kosong di depan.
- b. *Jump move*, yaitu gerakan diagonal melompati satu bidak lawan ke kotak, kosong di depan. Bidak yang akan dilompati akan dihilangkan dari papan, dan apabila terdapat *jump move* setelah gerakan *jump move*, pemain harus melakukan gerakan tersebut sampai tidak terdapat *jump move* lagi.

Ketika pemain memiliki kesempatan untuk melakukan *jump move*, pemain harus melakukan gerakan itu. Apabila pemain memiliki beberapa Gerakan *jump move*, pemain bebas memilih *jump move* mana yang akan pemain ambil. (Fred Reinfeld, 2020).

4. Alpha – Beta

Dalam algoritma *Minimax* pencarian dilakukan pada seluruh bagian pohon, sementara Sebagian pohon tidak seharusnya diperiksa. *Alpha-beta pruning* merupakan modifikasi dari algoritma *Minimax*, yang akan mengurangi jumlah node yang dievaluasi oleh pohon pencarian. Pencarian untuk node berikutnya akan dipikirkan terlebih dahulu. Algoritma ini akan berhenti mengevaluasi langkah ketika terdapat paling tidak satu kemungkinan yang ditemukan dan membuktikan bahwa langkah tersebut lebih buruk jika dibandingkan dengan langkah yang diperiksa sebelumnya. Sehingga, langkah berikutnya tidak perlu dievaluasi lebih jauh. Dengan algoritma ini hasil optimasi tidak akan berubah. Berikut merupakan pohon dengan algoritma *alpha-beta pruning*. (Ayuningtyas, N., 2008)



Gambar 2.2 *Alpha-beta Pruning*

Diperlihatkan, pada pohon tersebut, bahwa terdapat pemotongan pencarian dengan menggunakan algoritma ini.

Pada algoritma ini, terdapat dua nilai yang diatur, yaitu alpha dan beta, yang merepresentasikan nilai minum dari max yang diyakini dan nilai maksimum dari min yang diyakini. Nilai awal alpha adalah tak hingga negatif dan nilai awal beta adalah tak hingga positif. Sebagai hasil dari proses rekursif, area pencarian akan semakin kecil. Ketika beta menjadi lebih kecil dari alpha, akan berarti posisi saat itu tidak dapat menjadi hasil terbaik permainan untuk kedua pemain dan pencarian tidak perlu dilakukan lebih jauh.

```

function ALPHA-BETA-SEARCH(state) returns an action
   $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$ 
  return the action in ACTIONS(state) with value v



---


function MAX-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow -\infty$ 
  for each a in ACTIONS(state) do
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$ 
    if  $v \geq \beta$  then return v
     $\alpha \leftarrow \text{MAX}(\alpha, v)$ 
  return v



---


function MIN-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow +\infty$ 
  for each a in ACTIONS(state) do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$ 
    if  $v \leq \alpha$  then return v
     $\beta \leftarrow \text{MIN}(\beta, v)$ 
  return v

```

Gambar 2.3 Pseudocode Algoritma Alpha Beta

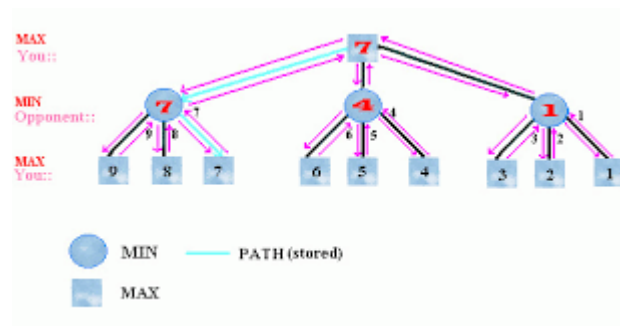
5. Algoritma *Minimax*

Algoritma *Minimax* merupakan algoritma yang digunakan untuk menentukan pilihan agar memperkecil kemungkinan kehilangan nilai maksimal. Algoritma ini diterapkan dengan baik dalam permainan yang melibatkan dua pemain yang saling

bergantian seperti *tic-tac-toe*, *othello*, *checkers*, *chess*, *go*, dan permainan yang menggunakan strategi atau logika lainnya (Ayuningtyas, N., 2008).

Algoritma *Minimax* merupakan algoritma dasar pencarian *DFS* (*Depth-First Search*) untuk melakukan travelsar dalam pohon. *DFS* akan mengekspanasi simpul paling dalam terlebih dahulu. Setelah simpul akar dibangkitkan, algoritma ini akan membangkitkan pada tingkat kedua, yang akan dilanjutkan pada tingkat ketiga, dan seterusnya.

Dalam representasi pohon dalam algoritma *Minimax*, terdapat dua jenis node, yaitu node *min* dan node *max*. *Max* node akan memilih langkah dengan nilai tertinggi dan *min* node akan memilih langkah dengan nilai terendah. Berikut merupakan gambar pohon untuk algoritma *Minimax*. (Ayuningtyas, N., 2008)



Gambar 2.4 Pohon Algoritma *Minimax*

C. Tinjauan Perangkat Lunak

1. PHP (*Hypertext Preprocessor*)

Menurut (Diar Puji Octavian, 2010) “*PHP* (*PHP Hypertext Prosesor*) adalah akronim dari *Hypertext Preprocessor*, yaitu suatu bahasa pemrograman berbasis kode-kode (*script*) yang di gunakan untuk mengolah suatu data dan

mengirimkannya kembali ke *web browser* menjadi kode HTML. Kode PHP mempunyai ciri-ciri khusus, yaitu :

- a. Hanya dapat dijalankan menggunakan *web server* misalnya : *Apache*.
- b. Kode PHP dapat diletakan dan dijalankan di *web server*.
- c. Kode PHP dapat digunakan untuk mengakses database, seperti: *MYSQL*, *PostgreSQL*, *Oracle*, dan lain-lain.
- d. Merupakan *software* yang bersifat *open source*.
- e. Gratis untuk di *download* dan digunakan.
- f. Memiliki sistem *multiplatform*, artinya dapat dijalankan menggunakan sistem operasi apapun, seperti *Linux*, *Unix*, *Windows*, dan lain-lain.

PHP merupakan *software Open-Source* yang disebarakan dan dilisensikan secara gratis serta dapat di-*download* secara bebas dari situs resminya <http://www.php.net>. PHP ditulis menggunakan bahasa C. PHP memiliki banyak kelebihan yang tidak dimiliki oleh bahasa *script* sejenis. PHP difokuskan pada pembuatan *script server-side*, yang bisa melakukan apa saja yang dapat dilakukan oleh CGI, seperti mengumpulkan data *form*, menghasilkan isi halaman *web* dinamis, dan kemampuan mengirim serta menerima *cookies*, bahkan lebih daripada kemampuan CGI.

PHP dapat digunakan pada semua *system operasi*, antara lain *Linux*, *Unix* (termasuk variannya *HP-UX*, *Solaris*, dan *OpenBSD*), *Microsoft Windows*, *Mac OS X*, *RISC OS*. PHP juga mendukung banyak *Web Server*, seperti *Apache*, *Microsoft Internet Information Server (MIIS)*, *Personal Web Server (PWS)*, *Netscape and iPanet servers*, *Oreilly Website Pro Server*, *audium*, *Xitami*, *OmniHTTPd*, dan

masih banyak lagi lainnya, bahkan PHP dapat bekerja sebagai suatu *CGI processor*. PHP tidak terbatas pada hasil keluaran HTML (*HyperText Markup Languages*). (Hasan S., Muhammad N., 2020)

2. JavaScript

JavaScript adalah bahasa pemrograman tingkat tinggi dan dinamis. *JavaScript* populer di internet dan dapat bekerja di sebagian besar penjelajah web populer seperti *Google Chrome*, *Internet Explorer (IE)*, *Mozilla Firefox*, *Netscape* dan *Opera*. Kode *JavaScript* dapat disisipkan dalam halaman web menggunakan tag *SCRIPT*. *JavaScript* merupakan salah satu teknologi inti *World Wide Web* selain *HTML* dan *CSS*. *JavaScript* membantu membuat halaman web interaktif dan merupakan bagian aplikasi web yang esensial.

Awalnya hanya diimplementasi sebagai *client-side* dalam penjelajah web, kini *engine JavaScript* disisipkan ke dalam perangkat lunak lain seperti dalam *server-side* dalam server web dan basis data, dalam program *non web* seperti perangkat lunak pengolah kata dan pembaca *PDF*, dan sebagai *runtime environment* yang memungkinkan penggunaan *JavaScript* untuk membuat aplikasi *desktop* maupun *mobile*. (Pratama A. P., Marpaung K. A., 2021).

3. HTTP (*Hypertext Transfer Protocol*)

HTTP (Hypertext Transfer Protocol) adalah *protocol* yang digunakan untuk mentransfer dokumen dalam *World Wide Web (WWW)*. *Protocol* ini adalah *protocol* ringan, tidak berstatus dan generik yang dapat dipergunakan berbagai

macam *tipe* dokumen. Pengembangan HTTP dikoordinasikan oleh *Konsorsium World Wide Web (W3C)* dan grup bekerja *Internet Engineering Task Force (IETF)*, bekerja dalam publikasi satu seri RFC, yang paling terkenal RFC yang menjelaskan HTTP/1,1, versi HTTP yang digunakan umum sekarang ini. HTTP adalah sebuah *protocol* meminta / menjawab antara *client* dan *server*.

Sebuah *client* HTTP seperti web *browser*, biasanya melalui permintaan dengan membuat hubungan *TPC/IP* ke port tertentu. Sebuah server HTTP yang mendengarkan di *port* tersebut menunggu *client* mengirim kode permintaan di ikuti dengan pesan *MIME* yang memiliki beberapa informasi data tertentu. (Setiawardhana, S., 2010).

4. Adobe Dreamweaver CS6

Adobe Dreamweaver adalah sebuah HTML (*Hyper Text Markup Language*) *editor* profesional untuk mendesain web secara visual dan mengelola situs atau halaman web. Saat ini terdapat *software* dari kelompok *adobe* yang belakangan banyak digunakan untuk mendesain suatu situs web. Versi terbaru dari *adobe dreamweaver* saat ini adalah *Dreamweaver CS6*. Pada *Dreamweaver CS6*, terdapat beberapa kemampuan bukan hanya sebagai *software* untuk mendesain web saja. (Madcoms Madiun, 2012)

D. Tahap-tahap Perancangan Sistem

1. UML (*Unified Modelling Language*)

UML (Unified Modelling Language) menurut (Munawar, 2005), dalam buku “Pemodelan Visual dengan UML” menyatakan UML adalah salah satu alat bantu yang sangat handal didunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

Sedangkan menurut menurut (Adi Nugroho, 2010), “*Unified Modelling Language* adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berpradigma berorientasi objek”. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Jurnal CCIT (2008:70) *Unified Modelling Language* (UML) didefinisikan oleh *Good Corporate* sebagai berikut :

Unified Modelling Language (UML) adalah “suatu alat bantu yang sangat handal dalam bidang pengembangan yang berorientasi objek. Hal ini disebabkan karena uml menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem untuk membuat cetak cetak biru atas visi mereka dalam bentuk baku”. *Metode Unified Modelling Language* (UML) menggunakan tiga bangunan dasar untuk mendeskripsikan sistem atau perangkat lunak yang akan dikembangkan, yaitu :

a. Sesuatu (*things*)

Ada empat *things* dalam *Unified Modelling Language* (UML) :

- 1) *Structural things*, bagian yang relative statis dapat berupa elemen- elemen yang bersifat fisik maupun konseptual.
- 2) *Behaviorial things*, bagian dinamis biasanya merupakan kata kerja dari model UML yang mencerminkan perilaku sepanjang ruang dan waktu.
- 3) *Grouping things*, bagian pengorganisasian dalam UML. Dalam penggambaran model UML yang rumit diperlukan penggambaran paket yang menyederhanakan model. Paket-paket ini kemudian dapat didekomposisi lebih lanjut. Paket berguna bagi pengelompokan sesuatu, misalnya model-model serta *subsitem-subsistem*.
- 4) *An notational things*, merupakan bagian yang meperjelas model UML. Dapat berisi komentar yang menjelaskan fungsi serta ciri-ciri tiap element dalam model UML.

b. Relasi (*relationship*)

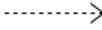
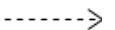
Untuk upaya tersebut UML menyediakan 9 jenis diagram yang dapat dikelompokkan berdasarkan sifatnya *statis* atau *dinamis*. Ke 9 jenis diagram dalam UML itu adalah :

- 1) *Diagram kelas* Bersifat statis Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula *diagram kelas* memuat kelas-kelas aktif.

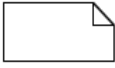
- 2) *Diagram Objek* Bersifat statis. *Diagram* ini memperlihatkan objek-objek serta relasi-relasi antar objek. Diagram objek memperlihatkan instalasi statis dari segala sesuatu yang dijumpai pada diagram kelas.
- 3) *Use-Case Diagram* Bersifat statis. *Diagram* ini memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku dari suatu sistem yang dibutuhkan serta diharapkan pengguna.
- 4) *Sequence Diagram* Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
- 5) *Collaboration Diagram* Bersifat dinamis. Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.
- 6) *Statechart Diagram* Bersifat dinamis. Diagram state ini memperlihatkan *state-state* pada sistem memuat *state*, *transisi*, *event*, serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka, kelas, kolaborasi dan terutama penting pada pemodelan sistem-sistem yang reaktif.
- 7) *Activity Diagram* Bersifat dinamis. *Diagram* aktivitas ini adalah tipe khusus dari *diagram state* yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi dalam suatu sistem dan memberi tekanan pada aliran kendali antar objek.

- 8) *Component Diagram* Bersifat statis. Dengan komponen ini memperlihatkan organisasi serta kebergantungan sistem atau perangkat lunak pada komponen-komponen yang telah ada sebelumnya. *Diagram* ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan kedalam satu atau lebih kelas-kelas, antarmuka-antarmuka serta kolaborasi-kolaborasi.
- 9) *Deployment Diagram* Bersifat statis. *Diagram* ini memperlihatkan konfigurasi saat aplikasi dijalankan. *Diagram* ini membuat simpul-simpul beserta komponen-komponen yang ada di dalamnya. *Diagram* ini sangat berguna pada banyak mesin. (Sholiq, 2006)

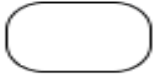
Tabel 2.2 Table Use Case Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (<i>independent</i>).
3		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek.
4		<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
5		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.

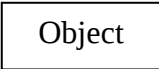
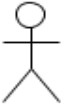
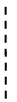
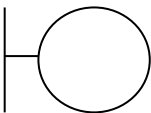
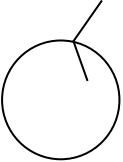
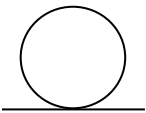
Tabel 2.2 Table Use Case Diagram

NO	GAMBAR	NAMA	KETERANGAN
7		<i>System</i>	Menspesifikasikan suatu paket yang menampilkan sistem secara terbatas.
8		<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan dalam sistem yang menghasilkan suatu hasil yang terukur bagi suatu <i>actor</i> .
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 2.3 Symbol Activity Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Activity</i>	Memperlihatkan cara bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2		<i>Action</i>	State dari suatu sistem yang mencerminkan eksekusi dari suatu aksi
3		<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali.
4		<i>Activity Final Node</i>	Bagaimana objek dibentuk dan dihancurkan
5		<i>Fork Node</i>	Satu aliran yang pada tahap- tahap tertentu dan berubah menjadi beberapa aliran

Tabel 2.4 *Symbol Sequence Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Objekct</i> (Partisipan)	<i>Object</i> atau biasa juga disebut partisipan merupakan <i>instance</i> dari sebuah <i>class</i> dan dituliskan tersusun secara <i>horizontal</i> .
2		<i>Actor</i>	<i>Actor</i> juga dapat berkomunikasi dengan <i>object</i> , maka <i>actor</i> juga dapat diurutkan sebagai kolom.
3		<i>Life line</i>	<i>Life line</i> mengidentifikasi keberadaan sebuah <i>object</i> dalam basis waktu. Notasi untuk <i>lifeline</i> adalah garis putus-putus <i>vertical</i> yang di tarik oleh sebuah <i>object</i> .
4		<i>Collaborations</i>	<i>Activication</i> dinotasikan sebagai kotak segi empat yang digambar pada sebuah <i>lifeline</i> <i>action</i> mengidentifikasi sebuah <i>object</i> yang akan melakukan sebuah aksi.
5		<i>Boundary</i>	<i>Boundary</i> terletak diantara <i>system</i> dengan dunia di sekelilingnya. Semua form, laporan-laporan, antara muka ke perangkat keras.
6		<i>Control</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		<i>Entity</i>	<i>Entity</i> digunakan menengani informasi yang mungkin akan disimpan secara permanen. <i>Entity</i> biasa juga merupakan sebuah <i>table</i> .

2. Use Case

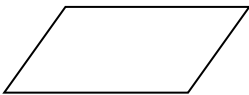
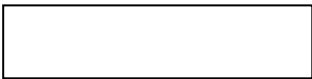
Use case diagram adalah abstraksi dari interaksi antara sistem dan *actor*. *Use case* bekerja mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.

Use Case Diagram merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata user. Sedangkan *Use Case* memfasilitasi komunikasi diantara analisis dan pengguna serta antara analis dan *client*. *Use Case* dapat sangat membantu bila kita sedang Menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan *client*, dan merancang *Test Case* untuk semua *feature* yang ada pada sistem. (Afrianto, I., 2015).

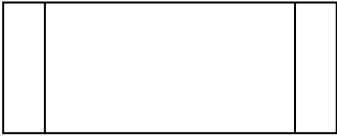
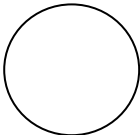
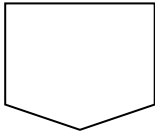
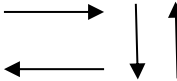
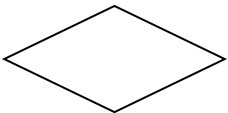
3. Bagan Alir Program

Bagan alir program (*program flowchart*) merupakan bagan yang menjelaskan rinci langkah-langkah dari proses program. Bagan alir program dibuat dari derifikasi bagan alir system. Bagan alir program dibuat dengan menggunakan simbol-simbol sebagai berikut :

Tabel 2.5 Simbol *Flowchart*

SIMBOL	KETERANGAN
	Simbol <i>input/output</i> digunakan untuk mewakili data <i>input/output</i> .
	Simbol proses, digunakan untuk mewakili suatu proses.

Tabel 2.5 Simbol *Flowchart*

SIMBOL	KETERANGAN
	Simbol proses terdefinisi, digunakan untuk menunjukkan suatu operasi yang rinciannya ditunjukkan ditempat lain.
	Simbol titik terminal, digunakan untuk awal dan akhir suatu proses.
	Simbol penghubung, digunakan untuk menunjukkan sambungan dari alir yang terputus dihalaman yang masih sama.
	Simbol penghubung, digunakan untuk menunjukkan sambungan dari alir yang terputus dihalaman yang berbeda.
	Simbol garis alir, menunjukkan arus dari proses.
	Simbol keputusan, digunakan untuk suatu penyelesaian kondisi didalam program.

E. Metode Pengujian Perangkat Lunak

Beberapa *test-case* harus dilaksanakan dengan beberapa perbedaan strategi, *query*, atau jalur navigasi yang mewakili penggunaan sistem yang *tipical*, kritis atau abnormal. Isu kunci pada pengembangan sistem adalah pemilihan sekelompok *test-case* yang cocok, sekecil dan secepat mungkin, untuk meyakinkan perilaku sistem secara detail. Pengujian harus mencakup *unit testing*, yang mengecek validasi dari prosedur dan fungsi-fungsi secara independen dari komponen sistem yang lain. Kemudian modul testing harus menyusul dilakukan untuk mengetahui apakah

Ciri – ciri pengujian *white box testing* :

- a. Pengujian berbasis pengetahuan. Penguji mengetahui cara kerja internal sistem dan menggunakan informasi ini untuk membuat tes yang tepat.
- b. Fokus pada struktur. Metode ini menguji arsitektur, aliran data, jalur eksekusi, dan logika kode yang digunakan dalam program.
- c. Deteksi bug mendalam. Dapat menemukan bug yang tersembunyi dalam kode, seperti kesalahan logika atau cacat dalam aliran kontrol program.
- d. Alat dan teknik. Teknik yang umum digunakan dalam *whitebox testing* termasuk *path testing*, *control flow testing*, *data flow testing*, dan *branch testing*.

Berikut adalah langkah – langkah umum dalam pengujian *white box testing* :

- a. Analisis kode sumber. Pelajari kode sumber aplikasi untuk memahami struktur internal dan alur logika. Identifikasi fungsi, alur kontrol, dan alur data.
- b. Identifikasi kasus uji. Buat kasus uji yang mencakup semua jalur dan kondisi yang diidentifikasi. Pastikan kasus uji dirancang untuk menguji semua cabang dan jalur eksekusi yang mungkin terjadi.
- c. Buat skripsi uji. Kembangkan skrip uji atau prosedur uji yang diperlukan untuk menjalankan kasus uji. Skrip ini akan menjalankan pengujian sesuai dengan desain yang telah dibuat.
- d. Jalankan kasus uji. Eksekusi kasus uji dengan menggunakan skrip uji yang telah dibuat. Pastikan untuk memeriksa hasil dan bandingkan dengan hasil yang diharapkan.

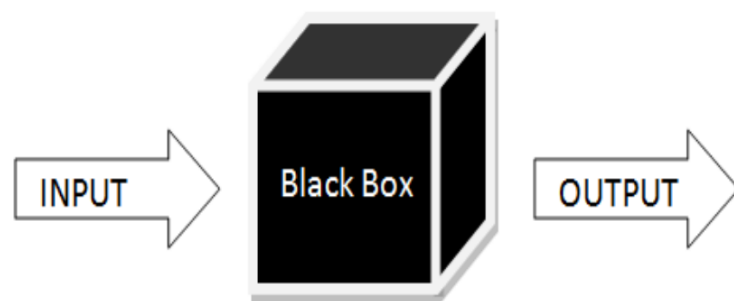
- e. Tulis laporan hasil uji. Dokumentasikan hasil uji, termasuk hasil yang berhasil dan masalah yang ditemukan. Laporkan masalah yang memerlukan perhatian lebih lanjut.
- f. Perbaiki dan uji ulang. Jika ditemukan masalah, koordinasikan dengan pengembang untuk memperbaikinya. Setelah perbaikan dilakukan, jalankan uji ulang untuk memastikan bahwa masalah telah teratasi dan tidak ada masalah baru yang muncul.
- g. Lakukan pengujian tambahan. Jika perlu, lakukan pengujian tambahan untuk memastikan bahwa perubahan atau perbaikan tidak memengaruhi bagian lain dari aplikasi.
- h. Evaluasi dan penutupan. Setelah semua pengujian selesai, evaluasi keseluruhan hasil pengujian dan pastikan semua kasus uji telah berhasil. Dokumentasikan proses pengujian secara menyeluruh dan tutup fase pengujian.

Contoh penggunaan *white box testing* sebagai berikut :

- a. Menguji bagaimana variabel diproses dalam suatu program
- b. Memeriksa jalur logika dalam algoritma, seperti memastikan semua cabang *if – else* dieksekusi.
- c. Digunakan dalam pengujian unit (*unit testing*), dimana pengembang memeriksa bagian kecil dari kode, seperti fungsi atau modul.

2. *Black Box Testing*

Black box testing adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak. Jadi dianalogikan seperti kita melihat suatu kotak hitam, kita hanya bisa melihat penampilan luarnya saja, tanpa tau ada apa dibalik bungkus hitam nya. Sama seperti pengujian *black box*, mengevaluasi hanya dari tampilan luarnya (*interfacenya*), fungsionalitas tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detilnya (hanya mengetahui *input* dan *output*).



Gambar 2.6 Pengujian *Black Box*

Ciri – ciri pengujian *blackbox testing* :

- a. Pendekatan berbasis fungsional. Pengujian dilakukan berdasarkan fungsionalitas eksternal sistem tanpa memperhatikan bagaimana sistem tersebut diprogram.
- b. Fokus pada hasil. Penguji memvalidasi bahwa sistem bekerja sesuai dengan yang diharapkan berdasarkan input yang diberikan dan keluaran yang dihasilkan.
- c. Pengujian pengguna akhir. *Blackbox testing* sering kali mirip dengan cara pengguna akhir akan menggunakan perangkat lunak.

- d. Alat dan teknik. Teknik umum termasuk *equivalence partitioning*, *boundary value analysis* dan *decision table testing*.

Berikut adalah langkah – langkah umum dalam pengujian *blackbox testing* :

- a. Memahami spesifikasi dan persyaratan
 - 1) Pelajari dokumentasi spesifikasi fungsional, persyaratan sistem, dan dokumen desain untuk memahami apa yang diharapkan dari sistem.
 - 2) Identifikasi fungsionalitas utama, alur kerja, dan kebutuhan pengguna.
- b. Perencanaan pengujian
 - 1) Buat rencana pengujian yang mencakup tujuan pengujian, ruang lingkup, sumber daya yang diperlukan, dan jadwal.
 - 2) Tentukan jenis pengujian *blackbox* yang akan dilakukan, seperti pengujian fungsional, pengujian batas, pengujian komponen, dll.
- c. Desain kasus uji
 - 1) Buat kasus uji berdasarkan spesifikasi fungsional dan persyaratan sistem.
Pastikan untuk mencakup berbagai skenario pengguna, input yang valid, dan input yang tidak valid.
 - 2) Rancang kasus uji untuk menguji fungsi-fungsi sistem, alur kerja, dan interaksi antar komponen.
- d. Persiapkan lingkungan pengujian
 - 1) Siapkan lingkungan pengujian yang mencakup perangkat keras, perangkat lunak, dan konfigurasi yang diperlukan untuk menjalankan pengujian.
 - 2) Pastikan semua data yang diperlukan dan konfigurasi sistem sesuai dengan spesifikasi pengujian.

e. Jalankan kasus uji

- 1) Eksekusi kasus uji yang telah dirancang. Masukkan data uji dan amati bagaimana sistem merespons.
- 2) Bandingkan hasil yang diperoleh dengan hasil yang diharapkan untuk menentukan apakah fungsionalitas sistem berfungsi dengan baik.

f. Dokumentasikan hasil uji

- 1) Catat hasil pengujian, termasuk kasus uji yang berhasil dan yang gagal.
- 2) Dokumentasi setiap masalah yang ditemukan, lengkap dengan deskripsi masalah, langkah-langkah untuk mereproduksi, dan hasil yang diharapkan versus yang diperoleh.

g. Tindak lanjut dan perbaikan

- 1) Laporkan masalah kepada tim pengembangan untuk diperbaiki.
- 2) Setelah perbaikan dilakukan, jalankan uji ulang untuk memastikan masalah telah teratasi dan tidak ada masalah baru.

h. Pengujian regresi

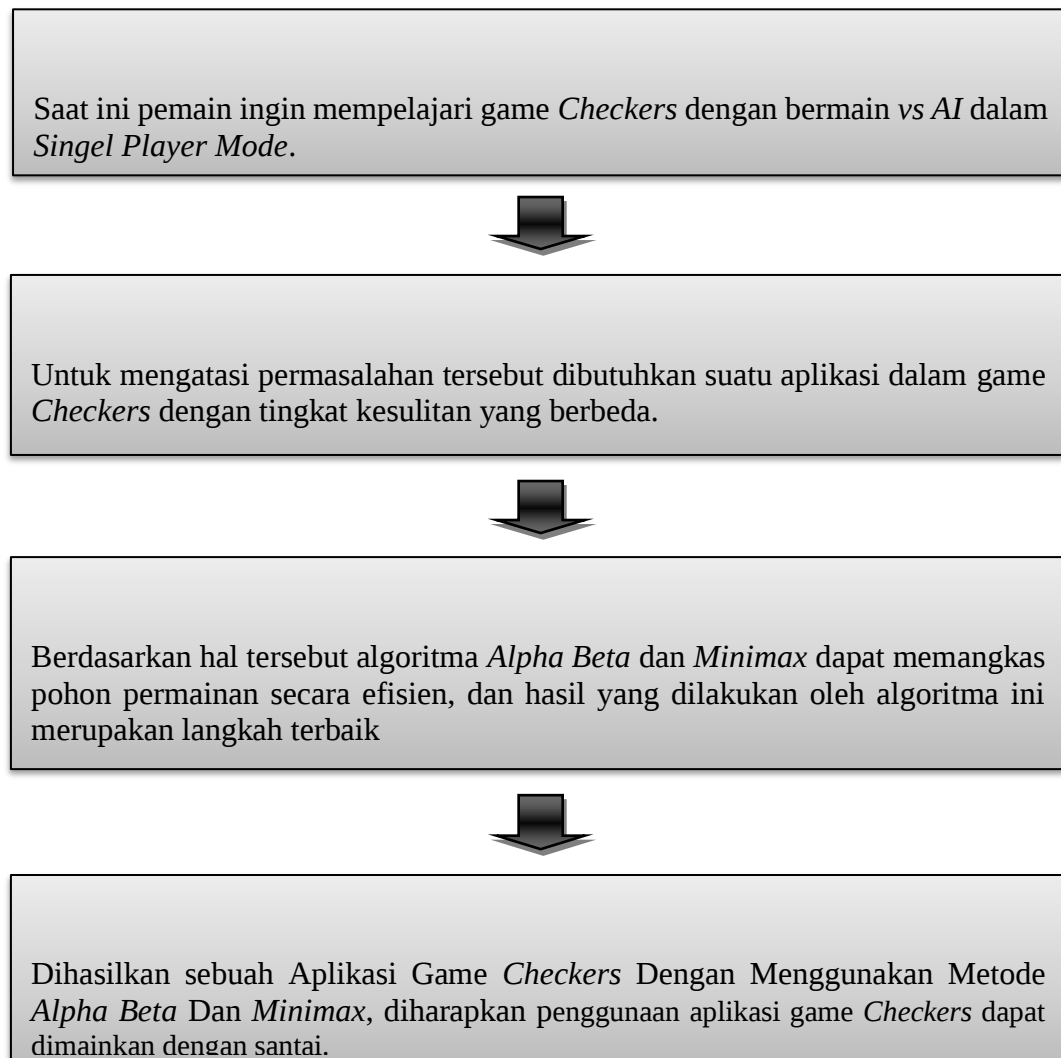
- 1) Lakukan pengujian regresi untuk memastikan bahwa perbaikan atau perubahan tidak memengaruhi bagian lain dari sistem yang sudah ada.

i. Evaluasi dan penutupan

- 1) Tinjau hasil pengujian secara keseluruhan dan pastikan semua kasus uji telah dieksekusi.
- 2) Dokumentasikan proses pengujian dan hasilnya secara menyeluruh. Persiapkan laporan akhir yang mencakup ringkasan temuan dan rekomendasi.

F. Kerangka Fikir

Untuk lebih memperjelas kerangka berpikir yang akan ditulis dalam penelitian akhir ini. Maka berikut ini digambarkan kerangka pikir tersebut pada gambar 2.7.



Gambar 2.7 Kerangka Fikir

BAB III

METODOLOGI PENELITIAN

A. Tempat dan Waktu Penelitian

Rencana waktu yang digunakan untuk penelitian ini berlangsung selama $\pm$ 2 bulan.

B. Jenis Penelitian

Jenis penelitian ini merupakan penelitian eksperimental yang bersifat aplikatif sehingga dari ruang lingkup masalah dapat dilakukan dengan metode studi Pustaka (*library research*), metode pengumpulan data lapangan (*field research*).

C. Metode Pengumpulan Data

Pada tahap ini peneliti melakukan wawancara dengan pihak yang terkait dengan penelitian, serta melakukan pencatatan dan pengamatan (*observasion*) langsung di tempat penelitian. Adapun uraian dari tahapan tersebut adalah sebagai berikut :

1. Persiapan Penelitian

Tahap persiapan adalah tahap yang dilakukan sebelum melakukan penelitian. Pada tahapan ini dimulai dengan mengkaji permasalahan yang telah ada kemudian melakukan studi literatur mengenai permasalahan yang sedang diteliti.

2. Studi *Literature*

Pada tahapan ini ada empat langkah yang harus dipenuhi guna untuk mencapai hasil maksimal dalam penelitian yaitu tahapan pengumpulan data dengan menggunakan beberapa metode pengumpulan data yang telah dijelaskan pada poin di atas, pengolahan data, analisis data dan selanjutnya penafsiran hasil analisis. Setelah kegiatan tersebut, proses selanjutnya yaitu dengan melakukan tugas lapangan dalam rangka mengumpulkan data, untuk kemudian akan diproses. Proses yang dimaksud meliputi penyuntingan, penerapan masalah dalam aplikasi program, serta analisis sebagai penarikan kesimpulan hasil akhir.

3. Pengumpulan Data

Pada tahap ini peneliti melakukan pencarian data-data di berbagai sumber untuk dikumpulkan lalu dikaji lebih lanjut.

4. Analisis

Pada tahap analisis, peneliti melakukan analisa terhadap sistem yang diterapkan sekarang berdasarkan kemudian merumuskan masalah yang menjadi pokok penelitian sehingga dapat dibuat alternatif pemecahan masalah.

5. Perancangan

Peneliti kemudian merancang aplikasi yang ingin dibuat berdasarkan alternatif pemecahan masalah.

6. Pengujian

Setelah melakukan perancangan, peneliti kemudian menguji hasil perancangan yang telah dibuat. Jika hasil perancangan terdapat kekurangan atau kelemahan maka kembali ke tahap analisis.

D. Sumber Data Penelitian

Adapun jenis dan sumber data yang dibutuhkan dalam penelitian ini adalah sebagai berikut :

1. Data Primer yaitu mengumpulkan data-data atau informasi yang terkait dengan game *Checkers* dan melakukan perancangan program aplikasi.
2. Data sekunder yaitu merupakan data yang diperoleh melalui studi kepustakaan, sosial media atau berbagai literatur dengan menelaah buku-buku dan tulisan-tulisan atau internet, jurnal hukum, serta peraturan perundang-undangan yang relevan dengan permasalahan yang diteliti.

E. Alat dan Bahan

Dalam pembuatan program aplikasi tersebut penulis membutuhkan perangkat keras (*Hardware*) dan perangkat lunak (*Software*). Adapun *Hardware* dan *Software* yang digunakan adalah :

1. Perangkat keras (*hardware*)

Perangkat keras yang digunakan untuk membuat aplikasi menggunakan PC (*Personal Computer*) dengan spesifikasi yang dapat dilihat pada tabel 3.1 berikut :

Tabel 3.1 Spesifikasi Perangkat Keras

Jenis	Spesifikasi
Laptop	ACER ASPIRE 5 A514-52K
Processor	Processor Intel Core i3-8130U Processor (up to 2.20GHz)
Ram	12 GB DDR 4 2400 MHZ
Hardisk	1 TB

2. Perangkat Lunak (*software*)

Perangkat lunak yang digunakan untuk membangun aplikasi ini dapat dilihat pada tabel 3.2 berikut :

Tabel 3.2 Spesifikasi Perangkat Lunak

Jenis	Spesifikasi
Sistem Operasi	<i>Windows 10 ultimate 64 bit</i>
Bahasa Pemograman	<i>PHP, Javascript</i>
<i>Tools</i>	<i>XAMPP, Visual Studio Code</i>

3. Bahan Penelitian

Adapun data penelitian berupa data-data penelitian yang didapat langsung melalui pedoman buku petunjuk dan mekanisme permainan checkers serta data-data gambar papan permainan checkers seperti pada gambar 3.1.



Gambar 3.1 Papan *Checkers*

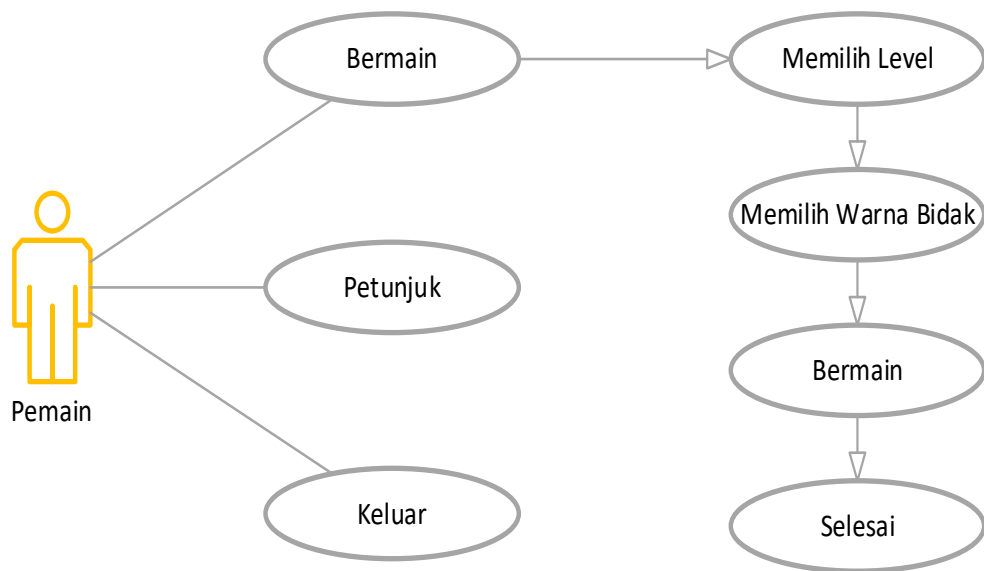
BAB IV

HASIL DAN PEMBAHASAN

A. Rancangan Sistem

1. Use Case

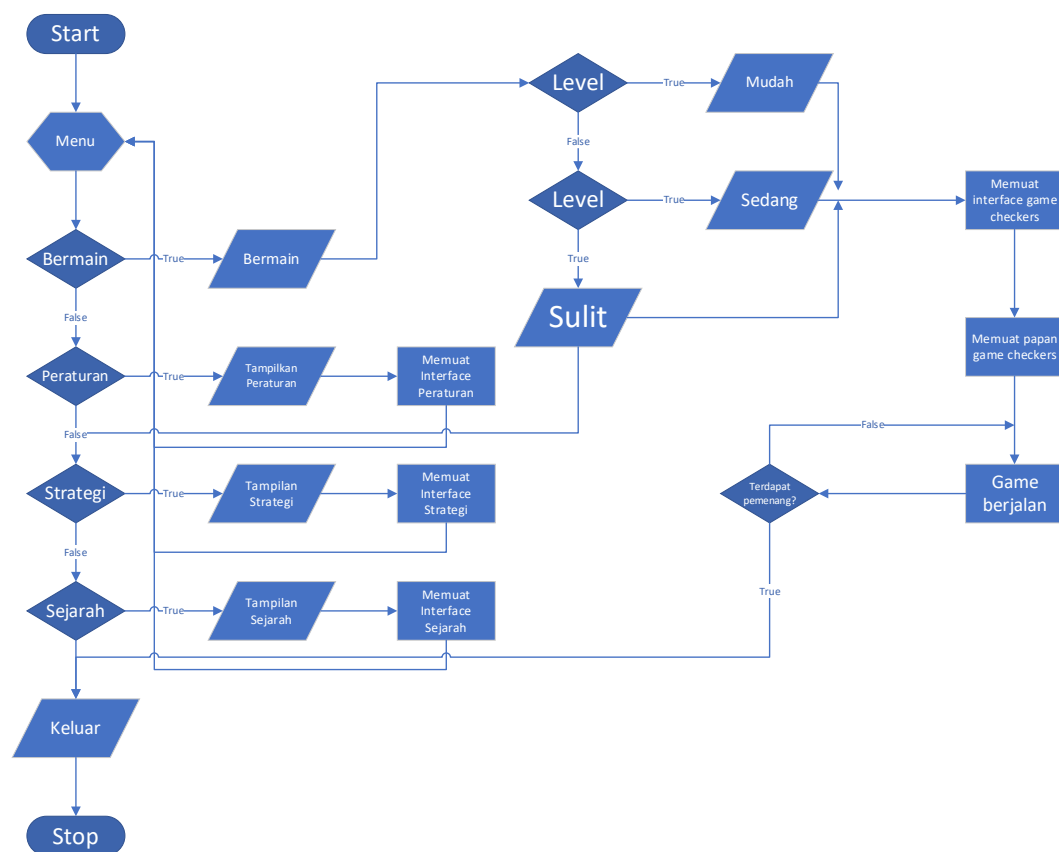
Use Case merupakan bagan yang menunjukkan alur kerja sistem jika dilihat menurut pandangan orang yang berada diluar sistem.



Gambar 4.1 *Use Case* Alur Penelitian

2. Analisis Aplikasi Diusulkan

Analisis aplikasi merupakan penguraian dari suatu sistem yang utuh kedalam bagian komponennya untuk mengidentifikasi dan mengevaluasi permasalahan. *Flowchart* aplikasi merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam aplikasi secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam aplikasi.



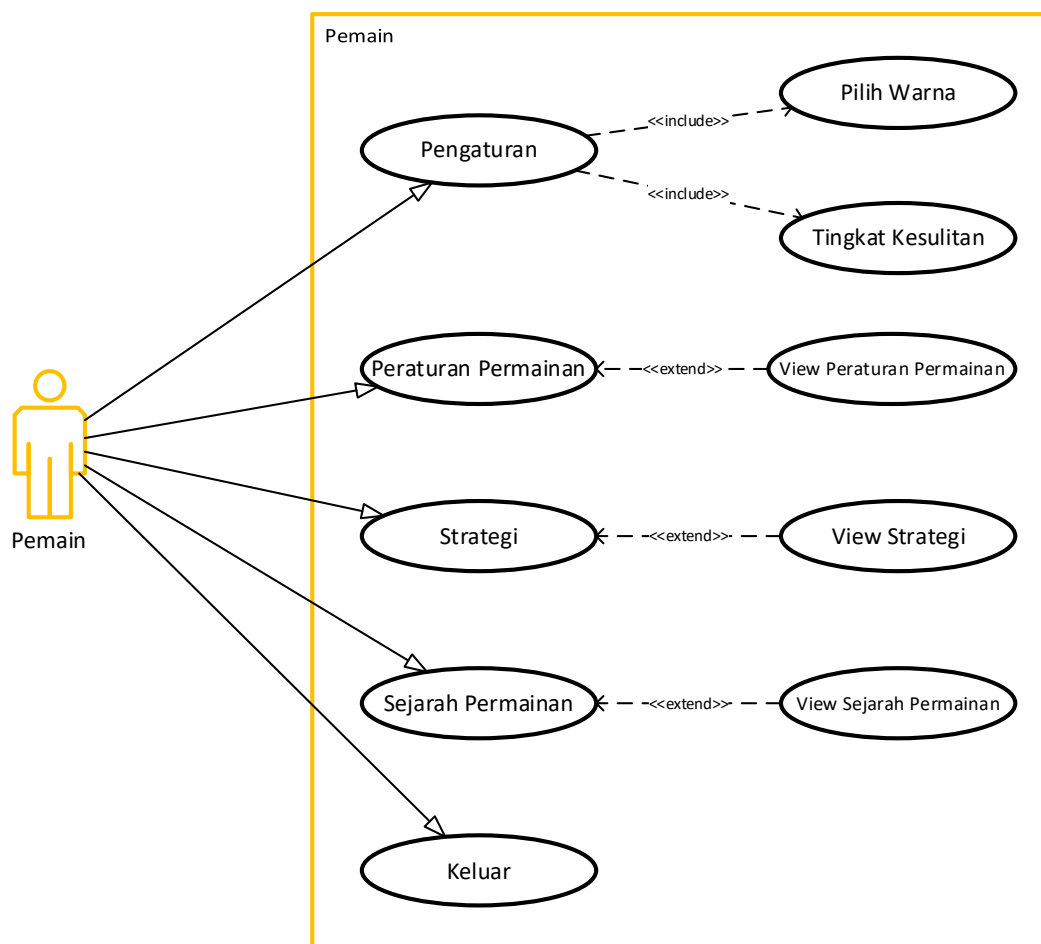
Gambar 4.2 *Flowchart* Aplikasi Diusulkan

B. Analisis Aliran Data UML

Analisis aliran data ini bertujuan mengetahui alur proses sistem. Dalam analisis sistem ini, penulis menggunakan pengembangan orientasi objek sehingga menggunakan *Use Case Diagram*, *Activity Diagram*.

1. *Use Case Diagram*

Use Case Diagram berfungsi untuk menjelaskan alur sistem jika dilihat menurut pandangan orang yang berada diluar sistem (actor).



Gambar 4.3 *Use Case Diagram* Pemain

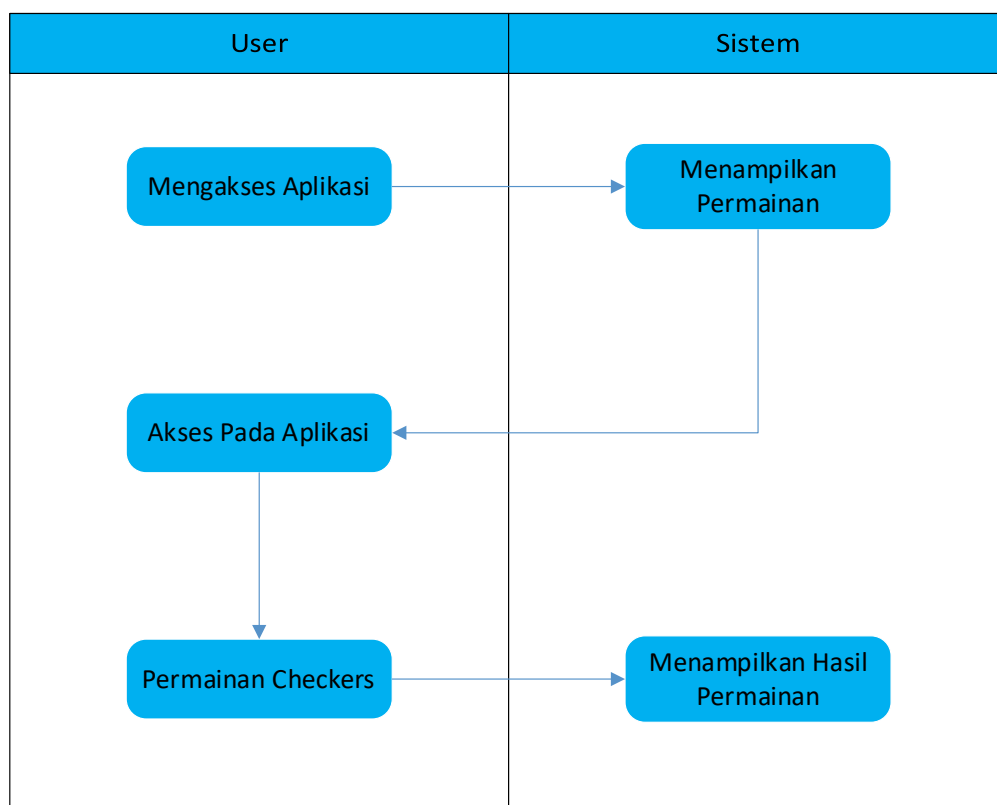
Tabel 4.1 Penjelasan *Use Case Diagram*

Nama <i>Use Case</i>	Deskripsi <i>Use Case</i>
Pengaturan	Pemain dapat mengakses opsi pengaturan untuk menyesuaikan preferensi permainan, termasuk memilih warna bidak dan tingkat kesulitannya.
Pilih Warna	Pemain memilih warna bidak yang akan digunakan dalam permainan. <i>Use Case</i> ini merupakan bagian dari pengaturan permainan.
Tingkat Kesulitan	Pemain memilih tingkat kesulitan permainan (misalnya, mudah, sedang, sulit) sebelum memulai. Ini juga termasuk dalam pengaturan permainan.
Peraturan Permainan	Pemain dapat mengakses dan melihat peraturan permainan. Ini dapat mencakup aturan dasar, tata cara bermain, dan informasi lain yang relevan.
View Peraturan Permainan	Pemain melihat detail peraturan permainan setelah memilih opsi peraturan permainan. <i>Use Case</i> ini memperluas fungsi peraturan permainan.
Strategi	Pemain dapat mengakses informasi atau panduan mengenai strategi yang dapat digunakan dalam permainan.
View Strategi	Pemain melihat detail dari strategi yang tersedia setelah memilih opsi Strategi. <i>Use case</i> ini memperluas fungsi Strategi.
Sejarah Permainan	Pemain dapat melihat catatan atau sejarah dari permainan sebelumnya.
View Sejarah Permainan	Pemain melihat detail sejarah permainan setelah memilih opsi Sejarah Permainan. <i>Use case</i> ini memperluas fungsi Sejarah Permainan.
Keluar	Pemain memilih untuk keluar dari permainan atau aplikasi. Ini menghentikan sesi permainan saat ini.

2. Activity Diagram

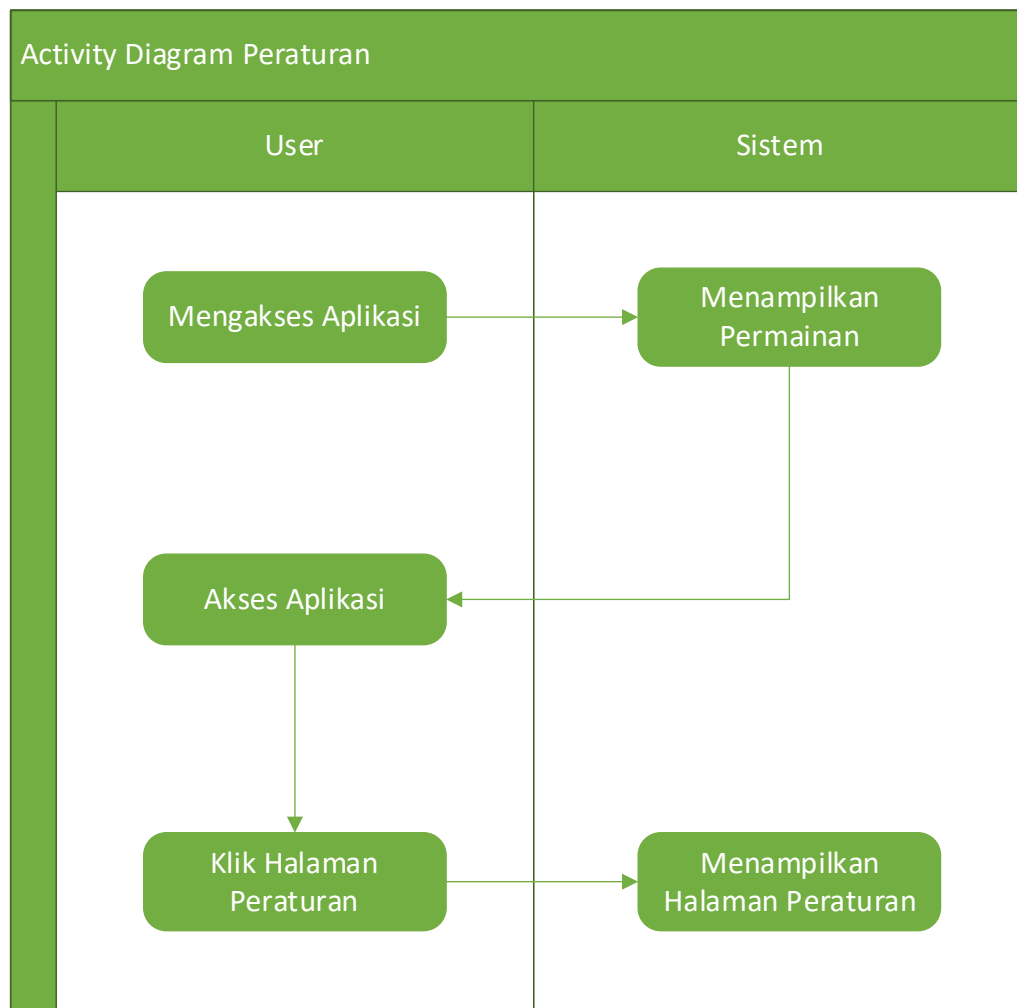
Activity Diagram adalah suatu bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan, dan membangun perangkat lunak. *Activity Diagram* ini menjelaskan tentang aktifitas – aktifitas terjadi dalam sebuah aliran proses pada sebuah sistem.

a. Activity Diagram Pemain



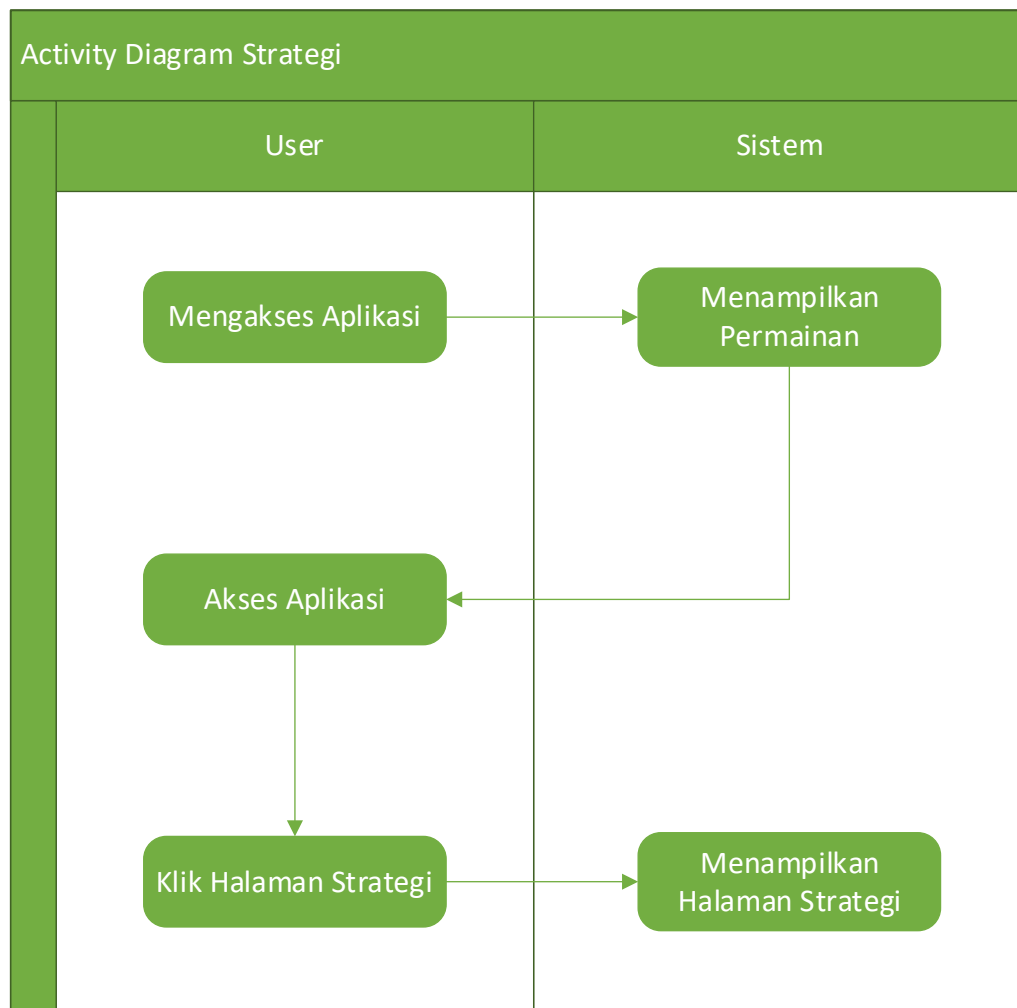
Gambar 4.4 Activity Diagram Pemain

b. *Activity Diagram* Peraturan

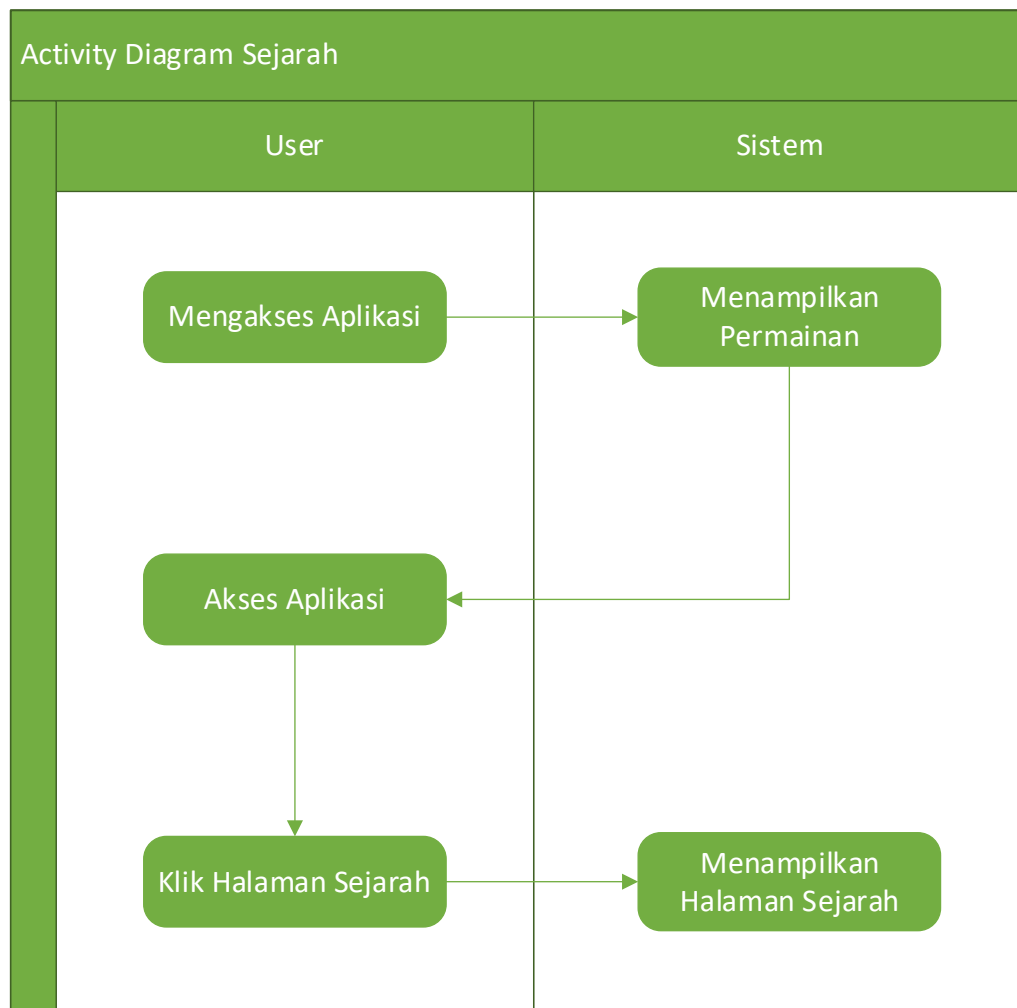


Gambar 4.5 *Activity Diagram* Peraturan

c. *Activity Diagram Strategi*

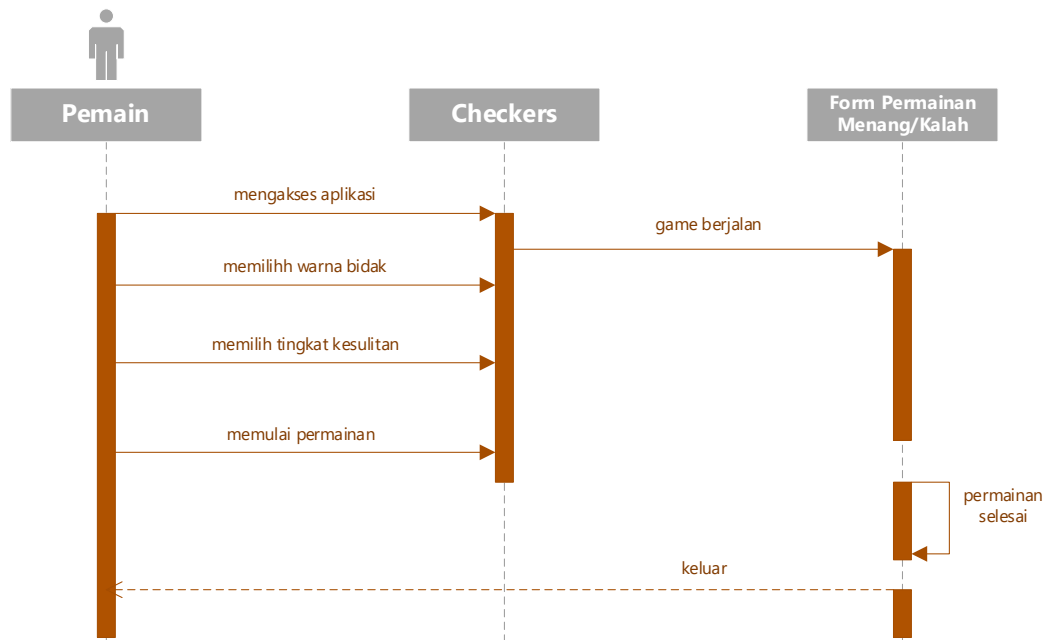


Gambar 4.6 *Activity Diagram Strategi*

d. *Activity Diagram Sejarah*Gambar 4.7 *Activity Diagram Sejarah*

3. Sequence Diagram

Sequence Diagram merupakan salah satu *diagram Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan *message* (pesan) apa yang dikirim dan kapan pelaksanaannya.

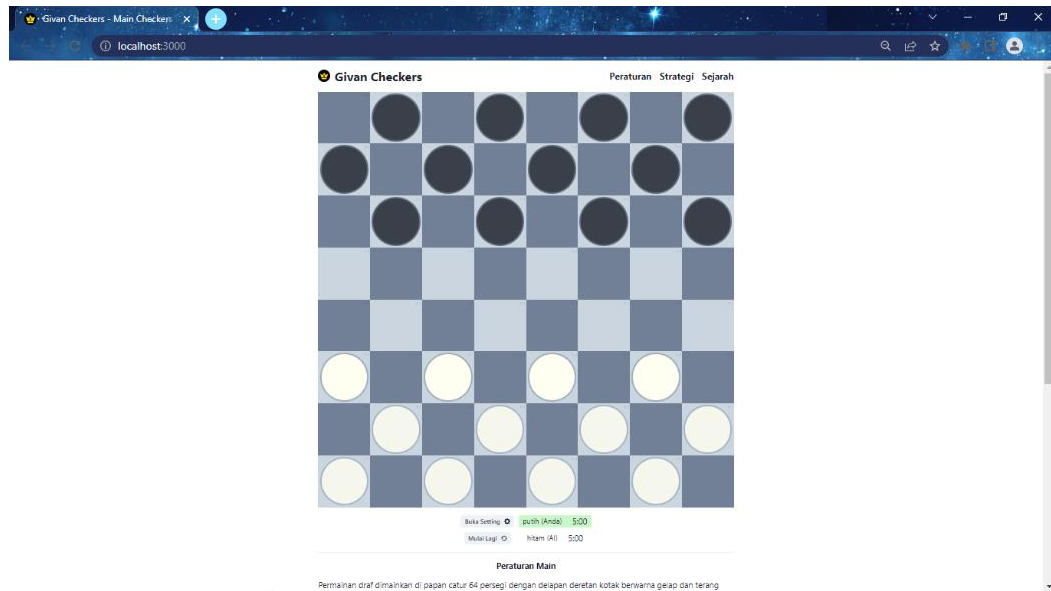


Gambar 4.8 *Sequence Diagram*

C. Rancangan *Output* Sistem

1. Halaman *Home*

Halaman *Home* merupakan tampilan awal yang ada pada sistem, untuk bermain sekaligus menampilkan tombol yang terdapat di halaman *Home*.

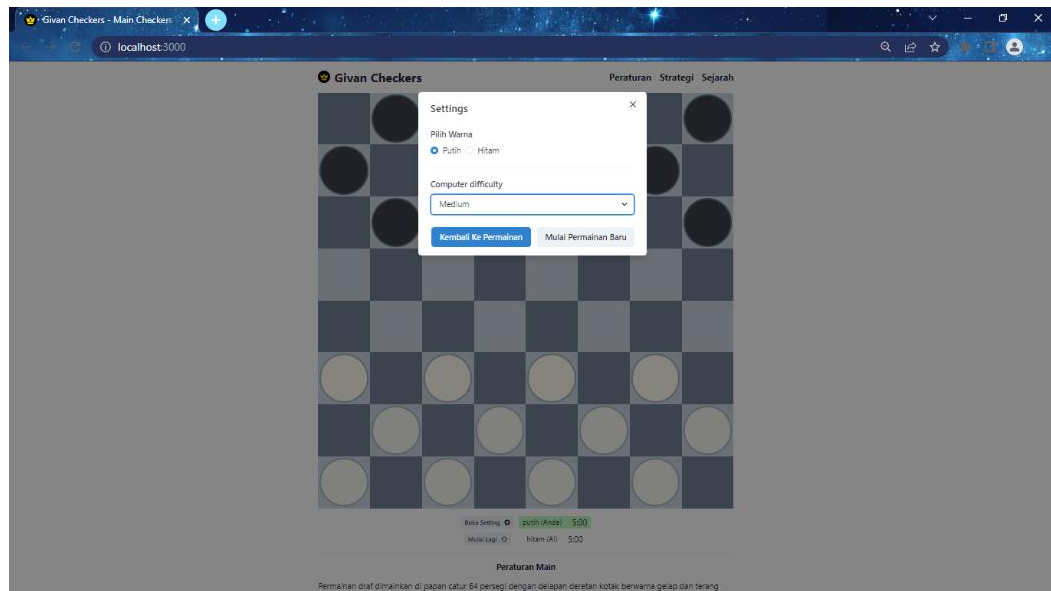


Gambar 4.9 Halaman *Home*

Didalam Halaman *Home* ini akan tampil ketika pemain mengakses/membuka aplikasi. Pada halaman ini terdapat beberapa *button* seperti, buka pengaturan, mulai lagi, peraturan, strategi, sejarah.

2. *Form* Buka Pengaturan

Form buka setting merupakan *form* yang berada di halaman *home*, dimana *form* ini berisikan inputan pilih warna dan pilih tingkat kesulitan.

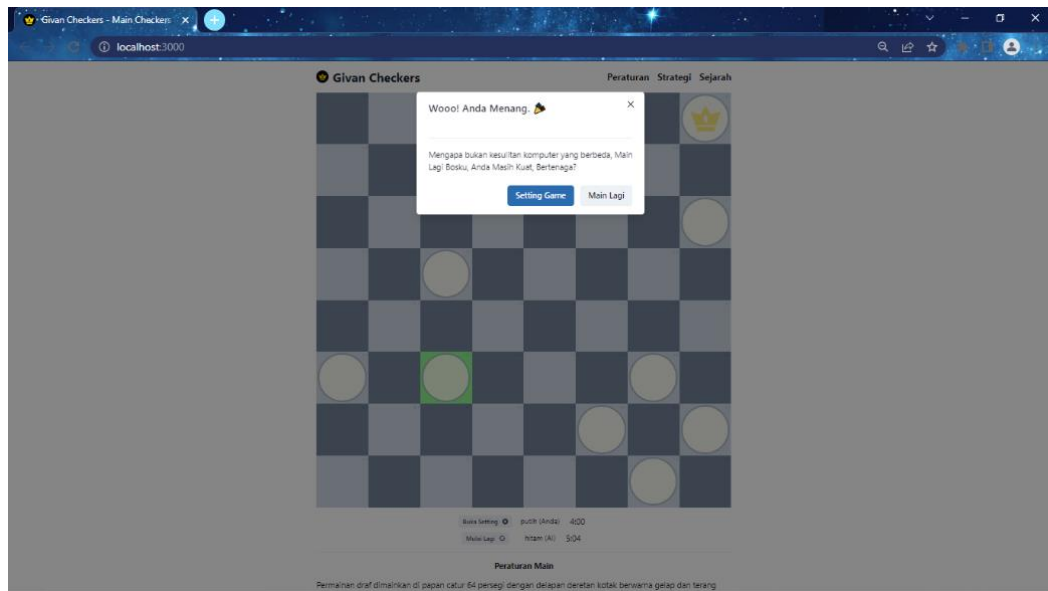


Gambar 4.10 Tampilan *Form* Buka Pengaturan

Terdapat beberapa pilihan didalam *form* buka pengaturan seperti, dibagian pilih warna ada dua jenis pilihan antara putih atau hitam, dan dibagian tingkat kesulitan ada tiga jenis pilihan antara mudah, sedang, sulit.

3. *Form* Permainan Menang

Form permainan menang merupakan tampilan *form* ketika pemain berhasil memenangkan dalam salah satu permainan.

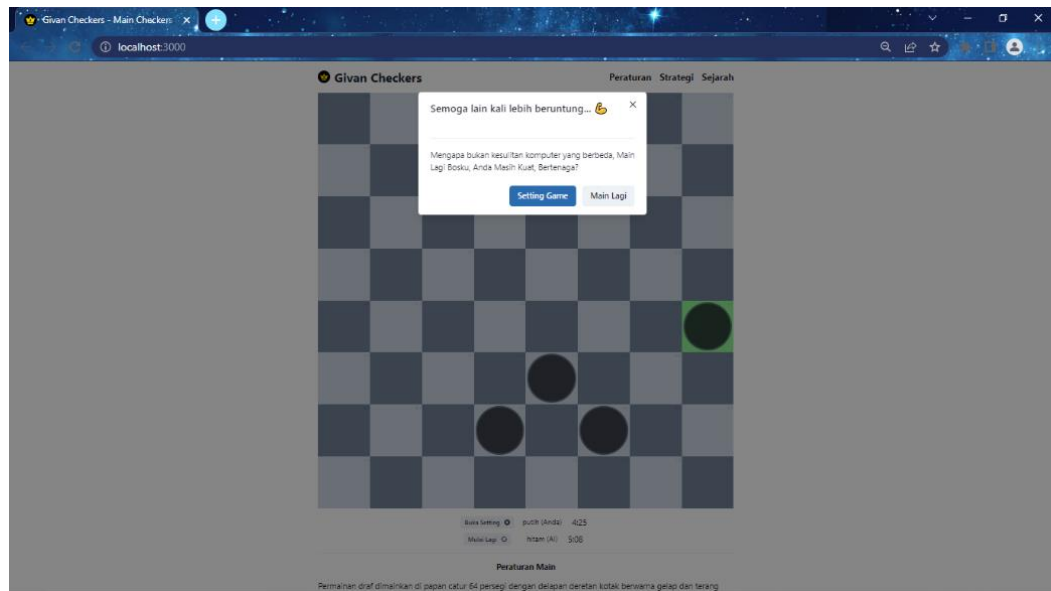


Gambar 4.11 Tampilan *Form* Permainan Menang

Didalam *form* permainan menang terdapat dua *button* pilihan yaitu antara buka pengaturan dan mulai lagi dengan kembali ke awal permainan.

4. *Form* Permainan Kalah

Form permainan kalah merupakan tampilan *form* ketika pemain kalah dalam salah satu permainan.

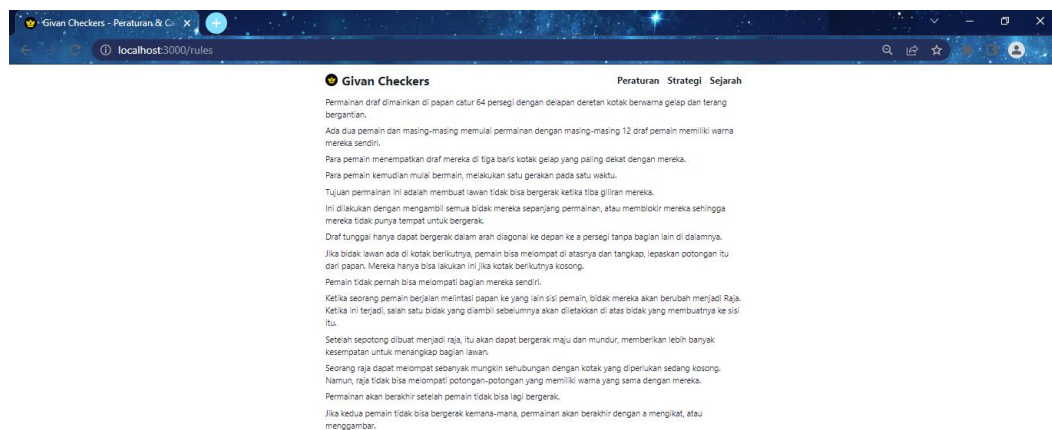


Gambar 4.12 Tampilan *Form* Permainan Kalah

Sama seperti dengan *form* permainan menang, didalam *form* permainan kalah juga terdapat dua *button* pilihan yaitu antara buka pengaturan dan mulai lagi dengan Kembali ke awal permainan.

5. Halaman Peraturan

Halaman peraturan merupakan tampilan yang menampilkan aturan-aturan dalam permainan *checkers*.

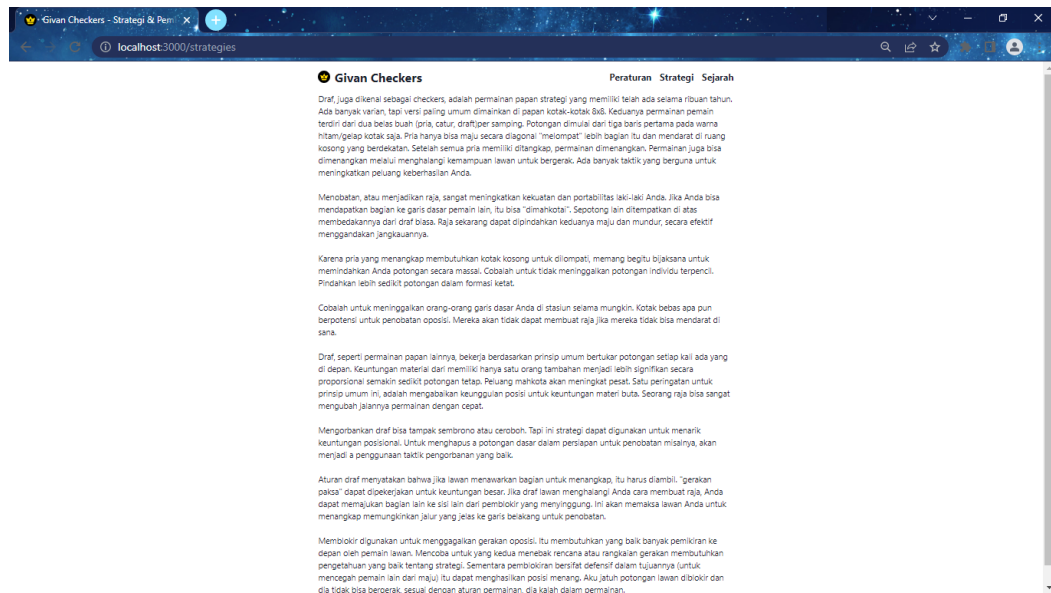


Gambar 4.13 Halaman Peraturan

Dalam halaman peraturan, pemain dapat mengetahui apa saja aturan dalam permainan *checkers* dan terdapat beberapa *button* seperti, *checkers*, peraturan, strategi dan sejarah.

6. Halaman Strategi

Halaman strategi merupakan tampilan jenis-jenis strategi dalam permainan *checkers*.

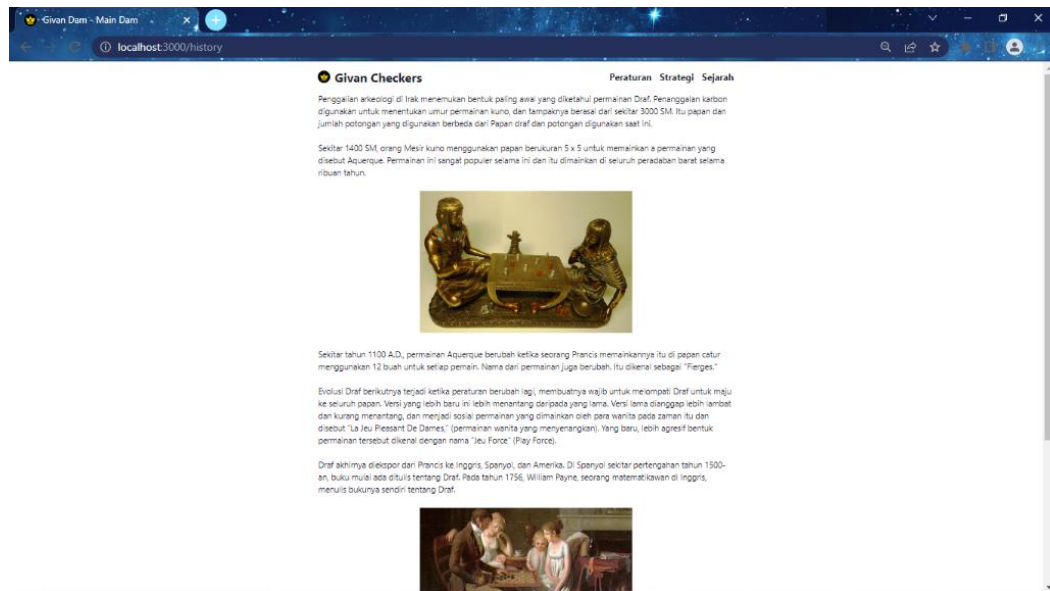


Gambar 4.14 Halaman Strategi

Didalam halaman strategi pemain dapat mengetahui taktik apa yang ingin digunakan dalam permainan *checkers* dan terdapat beberapa button seperti, *checkers*, peraturan, strategi, dan sejarah.

7. Halaman Sejarah

Halaman sejarah merupakan tampilan halaman yang menjelaskan dari awal sejarah dalam permainan *checkers*.

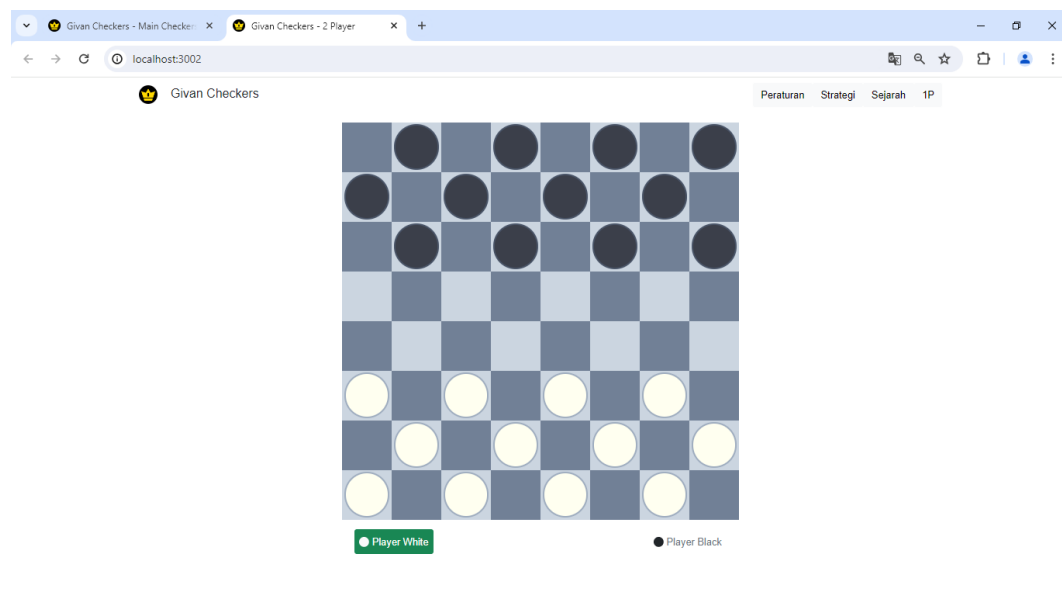


Gambar 4.15 Halaman Sejarah

Pemain dapat mengetahui bagaimana awal mulai *game checkers* dari awal hingga bisa sampai dikembangkan dengan melawan komputer atau *vs A.I. (Artificial Intelligence)* Terdapat beberapa *button* seperti, *checkers*, peraturan, strategi, dan sejarah.

8. Dua Pemain

Halaman Dua Pemain atau Two Player merupakan dua mode pemain melawan pemain.



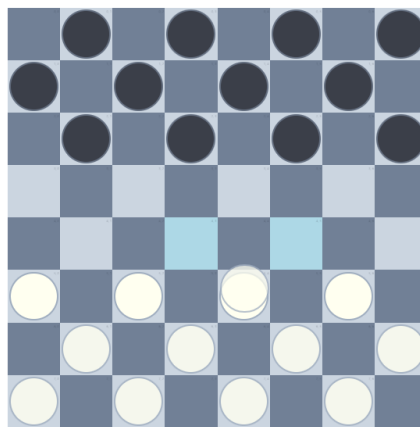
Gambar 4.16 Dua Pemain

Di mode ini pemain dapat bermain dua orang dengan pemain lainnya untuk dalam satu perangkat.

D. Detail Aplikasi

Tampilan utama aplikasi merupakan antarmuka interaktif yang dirancang untuk memberikan pengalaman bermain yang intuitif dan menyenangkan bagi pengguna. Berikut adalah deskripsi detail dari elemen – elemen yang terdapat pada tampilan halaman utama :

1. Papan Permainan

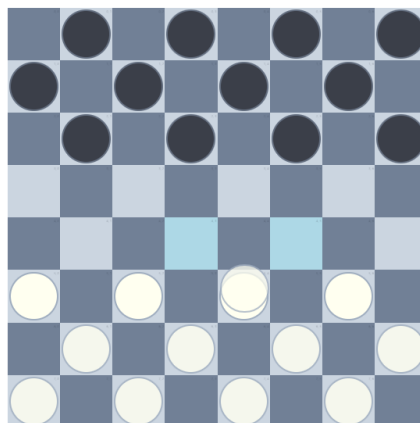


Gambar 4.17 Papan Permainan

```
<DndProvider backend={backend}>
  <DraughtsGameOverModal />
  <Grid
    templateRows="repeat(8, 1fr)"
    templateColumns="repeat(8, 1fr)"
    w="100%"
    h="100%"
    style={{ aspectRatio: 1 }}
  >
    {rows.map(({ row, rowIndex }) =>
      row.map((piece, colIndex) => (
        // eslint-disable-next-line react/no-array-index-key
        <GridItem key={` ${rowIndex}:${colIndex}:${piece} `}>
          <DraughtsCell
            piece={piece}
            rowIndex={rowIndex}
            colIndex={colIndex}
          />
        </GridItem>
      ))
    )}
  </Grid>
</DndProvider>
```

Fungsi ini adalah bagian dari antarmuka pengguna untuk permainan *checkers* yang menggunakan *React* dan *Chakra UI*. Komponen **‘DndProvider’** menyediakan konteks untuk *drag – and – drop*, sementara **‘DraughtsGameOverModal’** menangani tampilan akhir permainan. Grid 8x8 dibuat menggunakan komponen **‘Grid’** dari *Chakra UI*, dengan setiap sel diwakili oleh komponen **‘GridItem’** yang berisi komponen **‘DraughtsCell’** untuk menampilkan bidak. Setiap sel diidentifikasi oleh kombinasi indeks baris dan kolom serta tipe bidaknya untuk memastikan kunci unik.

2. Bidak

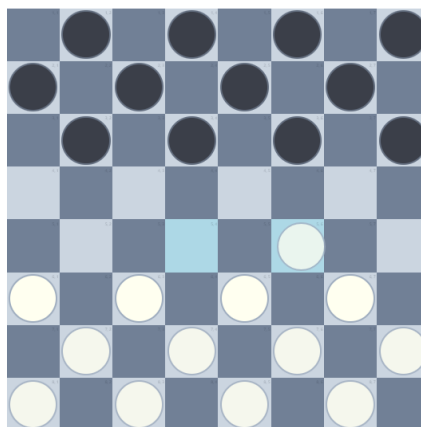


Gambar 4.18 Bidak

```
<Center
  ref={dragRef}
  zIndex="1"
  w="100%"
  h="100%"
  opacity={canDrag ? 1 : 0.8}
  borderWidth="0.2em"
  borderStyle="solid"
  borderColor={isWhite ? 'gray.400' : 'gray.600'}
  borderRadius="50%"
  bgColor={isWhite ? 'yellow.50' : 'gray.900'}
>
  {pieceIsQueen(props.piece) && (
    <DraughtsCrown w="70%" h="70%" opacity="0.6" userSelect="none" />
  )}
</Center>
```

Fungsi ini adalah bagian dari antarmuka pengguna untuk permainan *checkers* yang menggunakan komponen **‘Center’** dari *Chakra UI* untuk menampilkan bidak. Komponen **‘Center’** menggunakan **‘dragRef’** untuk mendukung *drag – and – drop* dan mengatur berbagai gaya berdasarkan apakah bidak bisa digerakkan (**‘canDrag’**) dan apakah bidak berwarna putih (**‘isWhite’**). Jika bidak adalah ratu, ditentukan oleh fungsi **‘pieceIsQueen’**, komponen **‘DraughtsCrown’** ditampilkan di dalamnya dengan ukuran dan gaya yang sesuai.

3. Langkah Bidak



Gambar 4.19 Langkah Bidak

```
_getMoves(starts) {
  const moves = [];
  // const captures = [];

  for (const start of starts) {
    const piece = this.position[start.row][start.col];
    const possibleDeltaRow = pieceIsQueen(piece) ? [-1, 1] : [1];
    for (const deltaRow of possibleDeltaRow) {
      for (const deltaCol of [-1, 1]) {
        const end = {
          col: start.col + deltaCol * this.direction,
          row: start.row + deltaRow * this.direction,
        };
        if (cellWithinBounds(end) && cellIsEmpty(this.position, end)) {
          moves.push({ captures: [], path: [start, end] });
        }
      }
    }
  }
}
```

```

    // captures = [
    //   ...captures,
    //   ...this._getNextCaptures(this.position, {
    //     captures: [],
    //     path: [start],
    //   }),
    // ];
  }
  // const cpt = filterToLongestCaptures(captures)

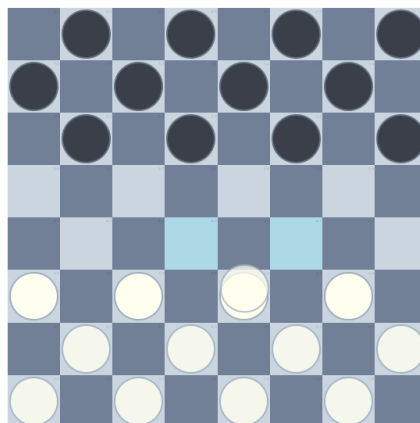
  // if (captures > 0) moves.push(cpt);

  return moves;
}

```

Fungsi ini menentukan langkah – langkah valid untuk bidak dalam permainan *checkers*. Fungsi iterasi melalui setiap posisi awal bidak, menentukan apakah bidak tersebut ratu atau bukan, dan menghitung posisi tujuan yang mungkin dengan bergerak diagonal ke kiri atau ke kanan. Fungsi memvalidasi setiap gerakan dengan memastikan posisi tujuan berada dalam batas papan permainan dan kosong. Jika valid, Gerakan tersebut ditambahkan ke dalam array ‘*moves*’, yang kemudian dikembalikan sebagai hasil.

4. *First Move* Bidak



Gambar 4.20 *First Move* Bidak

```

export class Board {
  constructor(position, playerToMove, firstMove = true) {
    this.position = position;
    this.playerToMove = playerToMove;
    this.firstMove = firstMove;
  }
}

```

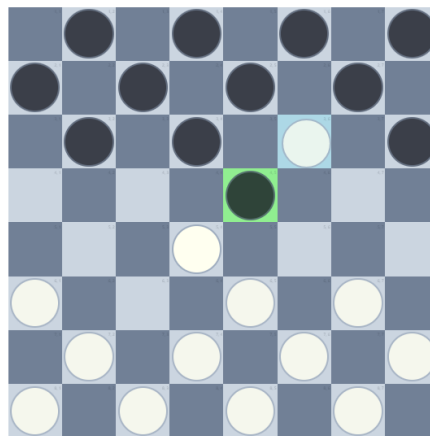
```

    this.direction = this.playerToMove === Players.WHITE ? -1 : 1;
    this.moves = this._computeMoves();
    this.state = this._computeState();
  }

```

Fungsi dari kelas '**Board**' menginisialisasi papan permainan *checkers* dengan posisi bidak, pemain yang akan bergerak, dan status apakah ini adalah Langkah pertama. Konstruktor menetapkan arah gerakan berdasarkan pemain yang akan bergerak (putih atau hitam), menghitung langkah yang mungkin dengan memanggil metode '**_computeMoves()**', dan menentukan status permainan dengan memanggil metode '**_computeState()**'.

5. Memakan Bidak



Gambar 4.21 Memakan Bidak

```

_getCaptures(starts) {
  let captures = [];
  // const getmv = this._getMoves(starts);
  for (const start of starts) {
    captures = [
      ...captures,
      ...this._getNextCaptures(this.position, {
        captures: [],
        path: [start],
      }),
    ];
  }
  // console.log('ftlc executed');
  var ftlc = filterToLongestCaptures(captures);
  // ftlc.moves.push(getmv);
}

```

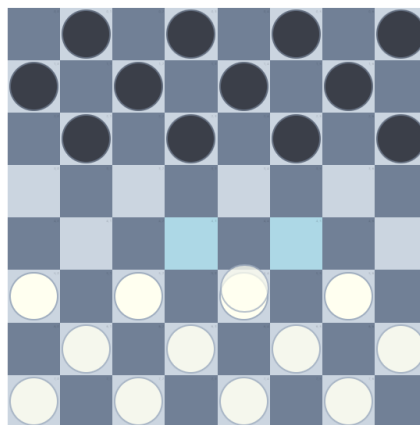
```

// return filterToLongestCaptures(captures);
return ftlc;
}

```

Fungsi dari ‘*_getCaptures*’ menentukan Langkah penangkapan bidak dalam permainan *checkers*. Untuk setiap posisi awal dalam ‘*starts*’, fungsi ini memperluas daftar ‘*captures*’ dengan hasil dari ‘*_getNextCaptures*’, yang menghitung langkah penangkapan lanjutan. Setelah semua penangkapan dihitung, fungsi menyaring penangkapan terpanjang menggunakan ‘*filterToLongestCaptures*’ dan mengembalikannya.

6. Posisi Bidak



Gambar 4.22 Posisi Bidak

```

export const INITIAL_POSITION = [
  [ // row 1
    Pieces.NONE, // col 1 => INITIAL_POSITION = 1,1
    Pieces.BLACK, // col 2
    Pieces.NONE,
    Pieces.BLACK,
    Pieces.NONE,
    Pieces.BLACK,
    Pieces.NONE,
    Pieces.BLACK,
  ],
  [ // row 2
    Pieces.BLACK, // col 1
    Pieces.NONE, // col 2
    Pieces.BLACK,
    Pieces.NONE,
    Pieces.BLACK,
  ],

```

```

    Pieces.NONE,
    Pieces.BLACK,
    Pieces.NONE,
  ],
  [
    Pieces.NONE,
    Pieces.BLACK,
    Pieces.NONE,
    Pieces.BLACK,
    Pieces.NONE,
    Pieces.BLACK,
    Pieces.NONE,
    Pieces.BLACK,
  ],
  [
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
  ],
  [
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
    Pieces.NONE,
  ],
  [
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
  ],
  [
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
  ],
  [

```

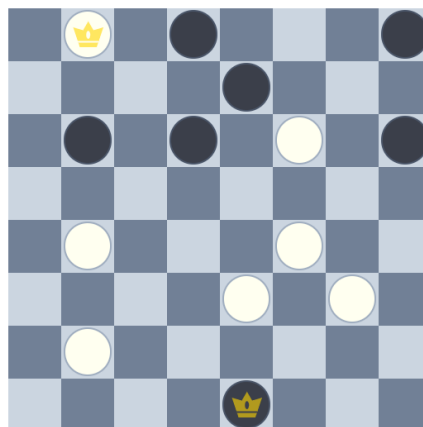
```

    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
    Pieces.WHITE,
    Pieces.NONE,
  ],
];

```

‘**INITIAL_POSITION**’ adalah konstanta yang mendefinisikan posisi awal bidak dalam permainan *checkers*. Ini adalah array dua dimensi yang merepresentasikan papan permainan 8x8, di mana setiap elemen menunjukkan posisi bidak hitam (**Pieces.BLACK**), bidak putih (**Pieces.WHITE**), atau posisi kosong (**Pieces.NONE**). Bidak hitam ditempatkan pada tiga baris teratas dengan pola selang – seling, sementara bidak putih ditempatkan pada tiga baris terbawah dengan pola yang sama. Batas Tengah papan dibiarkan kosong.

7. Menjadikan Raja



Gambar 4.23 Menjadikan Raja

```

doMove({ path, captures }) {
  const start = path.at(0);
  const startPiece = this.position[start.row][start.col];
  const end = path.at(-1);

  const endPiece = shouldQueen(end, startPiece)

```

```

// const endPiece = shouldQueen(end, this.position[6][2])
? queenPiece(startPiece)
: startPiece;

const newPosition = clonePosition(this.position);
newPosition[path.at(0).row][path.at(0).col] = Pieces.NONE;
newPosition[end.row][end.col] = endPiece;

for (const capture of captures) {
  newPosition[capture.row][capture.col] = Pieces.NONE;
}

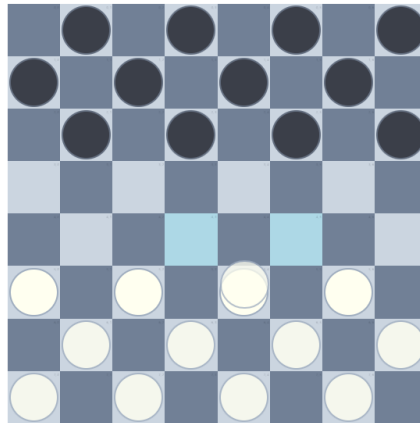
const newPlayerToMove =
  this.playerToMove === Players.WHITE ? Players.BLACK :
  Players.WHITE;

return new Board(newPosition, newPlayerToMove, false);
}
}

```

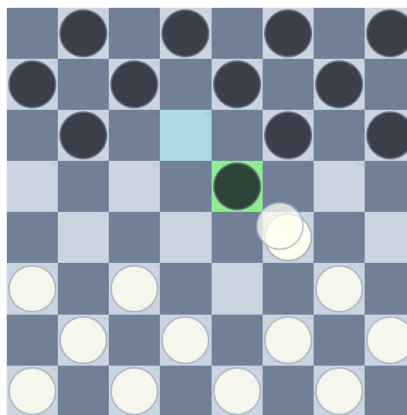
Fungsi **‘doMove’** menjalankan langkah dalam permainan *checkers* dengan memindahkan bidak dari posisi awal ke posisi akhir yang ditentukan oleh **‘path’**. Fungsi ini memeriksa apakah bidak harus diubah menjadi raja di posisi akhir dan memperbarui papan permainan dengan menghapus bidak dari posisi awal, menempatkannya di posisi akhir, serta menghapus semua bidak yang ditangkap. Akhirnya, giliran pemain diperbarui dan fungsi mengembalikan objek papan baru (**‘Board’**) dengan posisi dan giliran pemain yang diperbarui.

E. Detail Permainan



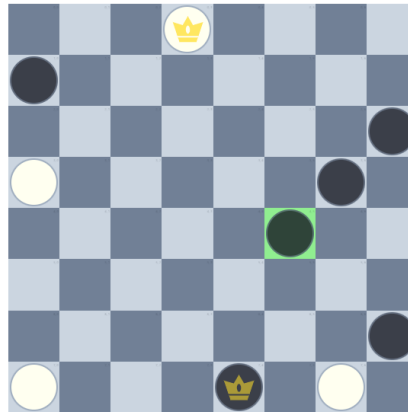
Gambar 4.24 Memulai Permainan

Pemain yang menggunakan bidak putih mendapatkan giliran pertama. Pada awal permainan, bidak hanya boleh digerakkan satu petak secara diagonal ke depan (dengan array yang telah dibuat).



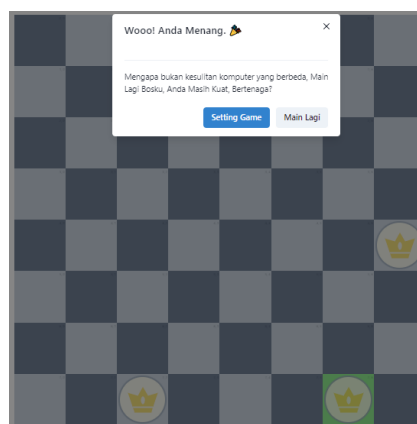
Gambar 4.25 Melompat dan Mengambil Bidak

Jika bidak pemain putih berhenti di petak diagonal yang dekat dengan bidak lawan, pemain dapat melompati dan mengambil bidak lawan tersebut. Untuk mengambil bidak lawan, pemain hanya perlu melewatinya dengan memindahkan bidak secara diagonal sejauh dua petak (searah dengan letak bidak lawan dengan array yang dibuat).



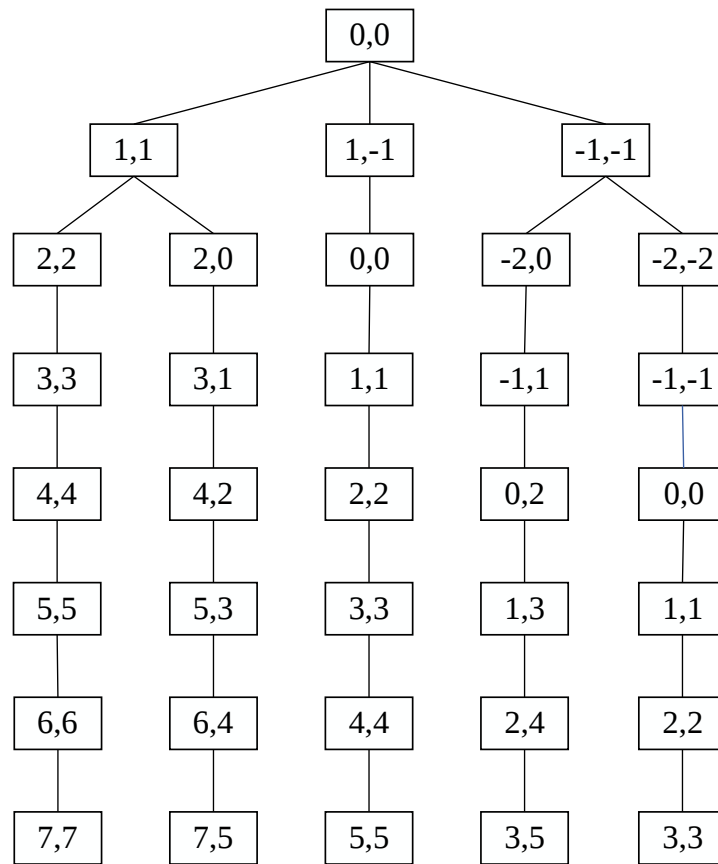
Gambar 4.26 Menjadikan Bidak Sebagai Raja

Untuk menjadikan bidak raja dan memahkotainya, letakkan salah satu bidak mencapai ujung papan (sisi lawan). Status bidak tersebut menjadi lebih tinggi, sehingga pemain dapat membedakannya dengan mudah dari bidak – bidak yang lain. Bidak raja dapat bergerak maju dan mundur, sehingga memudahkan pemain untuk mengambil bidak – bidak lawan.



Gambar 4.27 Menyelesaikan Permainan

Terus maju dan mengambil bidak – bidak lawan hingga tidak ada yang tersisa. Jika semua bidak lawan berhasil diambil, pemain berhasil memenangkan permainan.



Gambar 4.28 Langkah – langkah *Alpha - Beta*

Berikut ini adalah langkah-langkah untuk menghitung menggunakan *Alpha – Beta* pada aplikasi game *Checkers* :

1. Mulai dari simpul awal (0,0) dan lakukan pencarian ke bawah pohon permainan dengan mempertimbangkan setiap kemungkinan langkah yang dapat dilakukan oleh pemain putih.
2. Setiap langkah yang mungkin dilakukan oleh pemain putih dicoba dan pencarian dilanjutkan ke bawah pohon permainan dengan mempertimbangkan langkah – langkah yang mungkin dilakukan oleh pemain hitam.
3. Pada setiap simpul pada kedalaman ganjil, nilai *alpha* dan *beta* dihitung. *Alpha* adalah nilai terbaik.

F. Implementasi

Implementasi sistem merupakan tahap penerapan dari suatu teknologi yang didesain untuk siap dioperasikan. Tahap ini merupakan terjemahan perancangan dari bab hasil analisis sebelumnya dalam suatu bahasa pemrograman.

Bahasa pemrograman yang digunakan untuk membuat “Implementasi Algoritma *Negascout* dan *Minimax* Dalam Aplikasi Game *Checkers*” adalah bahasa pemrograman *PHP*.

1. Kebutuhan Perangkat Keras

Spesifikasi perangkat keras sebagai berikut :

Tabel 4.2 Spesifikasi Perangkat Keras

Jenis	Spesifikasi
Laptop	<i>ACER ASPIRE 5 A514-52K</i>
<i>Processor</i>	<i>Processore Intel Core i3-8130U Processor (up to 2.20GHZ)</i>
<i>Ram</i>	12 GB DDR 4 2400 MHZ
<i>Hardisk</i>	1 TB

2. Kebutuhan Perangkat Lunak

Spesifikasi perangkat lunak sebagai berikut :

Tabel 4.3 Spesifikasi Perangkat Lunak

Jenis	Spesifikasi
Sistem Operasi	<i>Windows 10 ultimate 64 bit</i>
Bahasa Pemograman	<i>PHP, Javascript</i>
<i>Tools</i>	<i>XAMPP, Visual Studio Code</i>

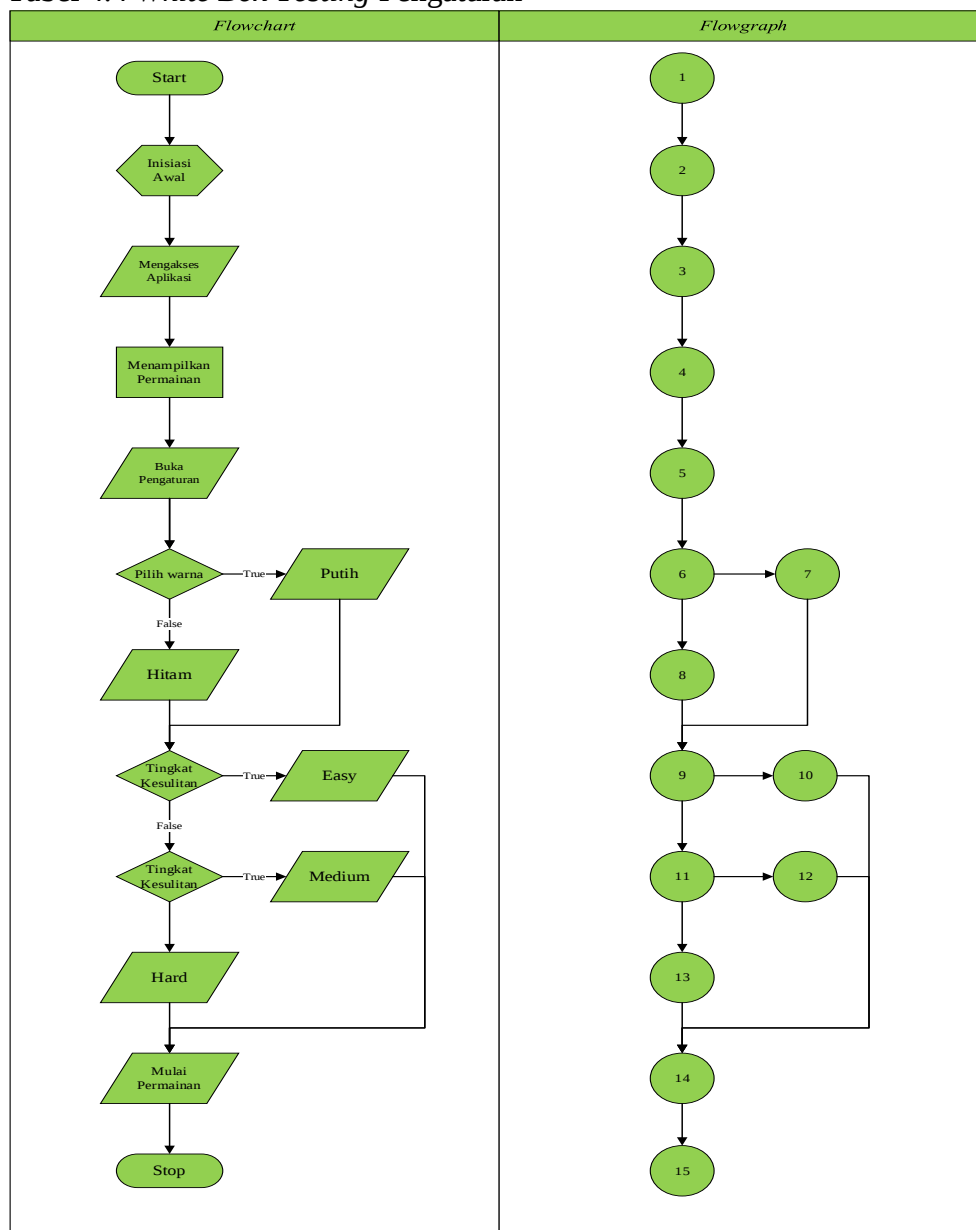
G. Pengujian

1. White Box

Pada metode pengujian ini akan ditampilkan *flowchart* dan *flowgraph* pada aplikasi yang telah dibuat. Berikut dibawah ini merupakan hasil dari pengujian yang menggunakan metode pengujian *White Box*.

a. White Box Testing Pengaturan

Tabel 4.4 White Box Testing Pengaturan



Dari *flowgraph* Buka Pengaturan di atas dapat dilakukan proses perhitungan sebagai berikut :

1) Menghitung *Cyclomatic Complexity* $V(G)$ dari *Edge* dan *Node* :

Dengan Rumus : $V(G) = E - N + 2$

Dengan E (*edge*) = 19

N (*node*) = 16

P (*Predikat Node*) = 3

Penyelesaian : $V(G) = E - N + 2$

= $19 - 16 + 2$

= 5

Predikat Node (P) = $P + 1$

= $3 + 1$

= 4

2) Berdasarkan perhitungan *Cyclomatic Complexity* dari *Flowgraph* di atas memiliki *Region* = 4

3) *Independent Path* pada *Flowgraph* di atas adalah :

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 14 – 15

Path 2 = 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 10 – 14 – 15

Path 3 = 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 11 – 12 – 14 – 15

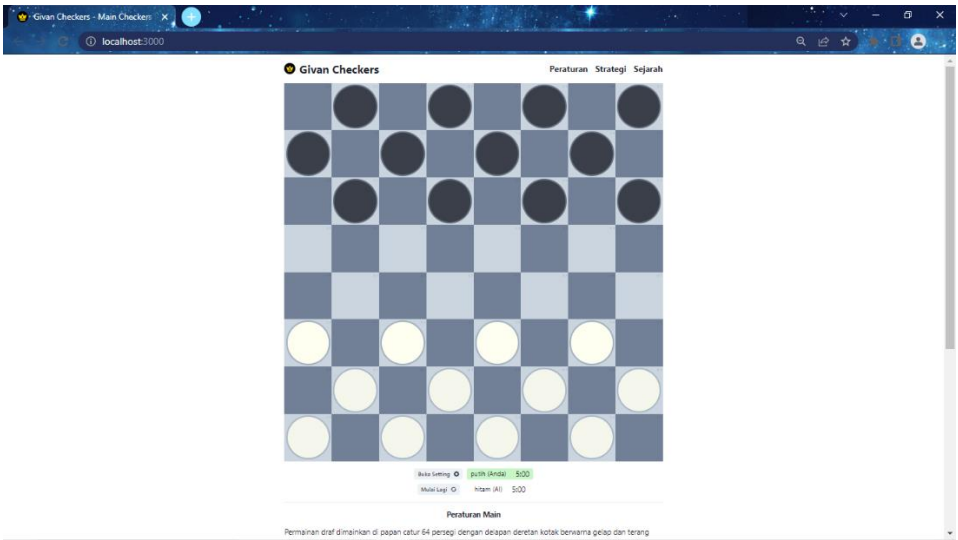
Path 4 = 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 11 – 13 – 14 – 15

2. Black Box

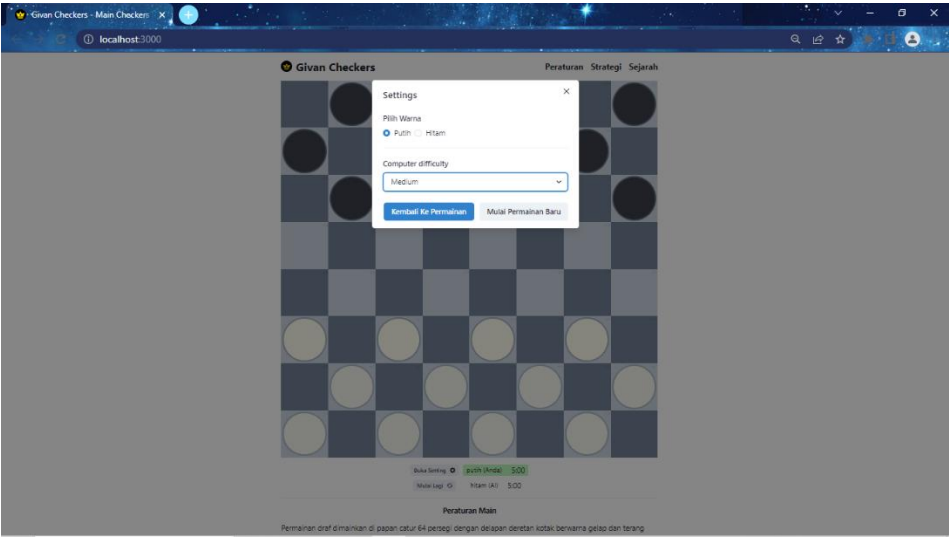
Pengujian *Black Box* adalah metode pengujian perangkat lunak yang menguji fungsionalitas aplikasi yang bertentangan dengan struktur *internal* atau kerja (lihat pengujian *black box*). Pengetahuan khusus dari kode aplikasi struktur *internal* dan pengetahuan pemrograman pada umumnya tidak diperlukan.

a. Black Box Testing Halaman Utama

Table 4.6 *Black Box Testing* Halaman Utama

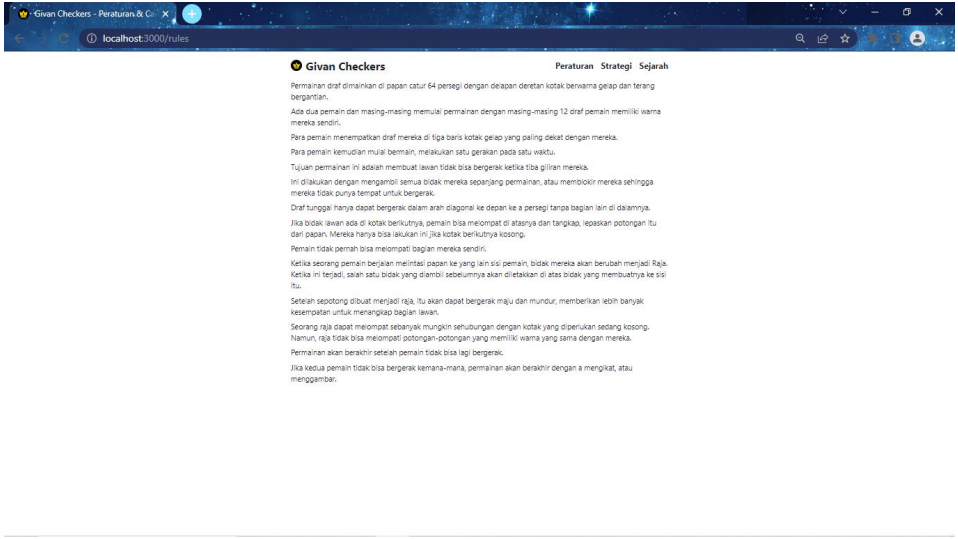
Teks Faktor	Hasil	Keterangan
User mulai bermain	✓	Berhasil, menampilkan halaman utama bermain
Screenshot		
		

b. *Black Box Testing Form Buka Pengaturan*Table 4.7 *Black Box Testing Buka Pengaturan*

Teks Faktor	Hasil	Keterangan
User membuka <i>form</i> pengaturan	✓	Berhasil, menampilkan <i>form</i> pengaturan
Screenshot		
		

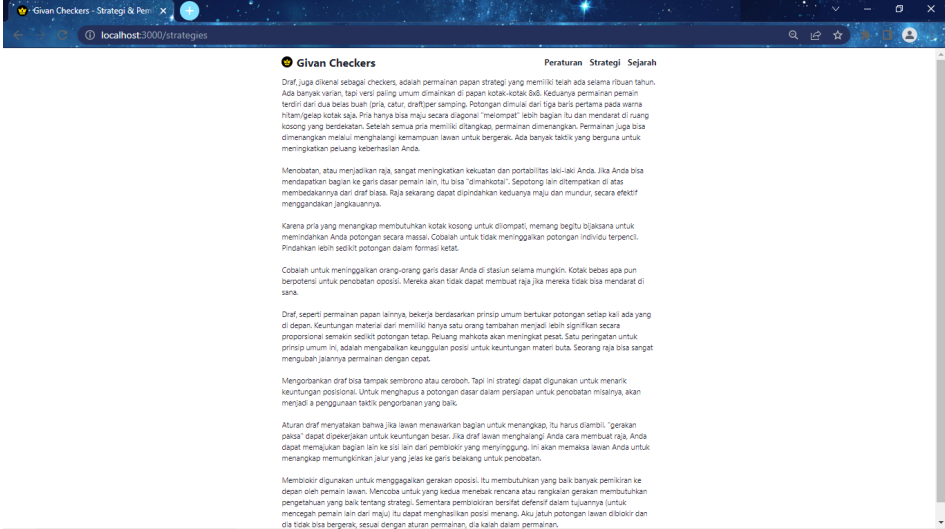
c. *Black Box Testing* Halaman Peraturan

Table 4.8 *Black Box Testing* Halaman Peraturan

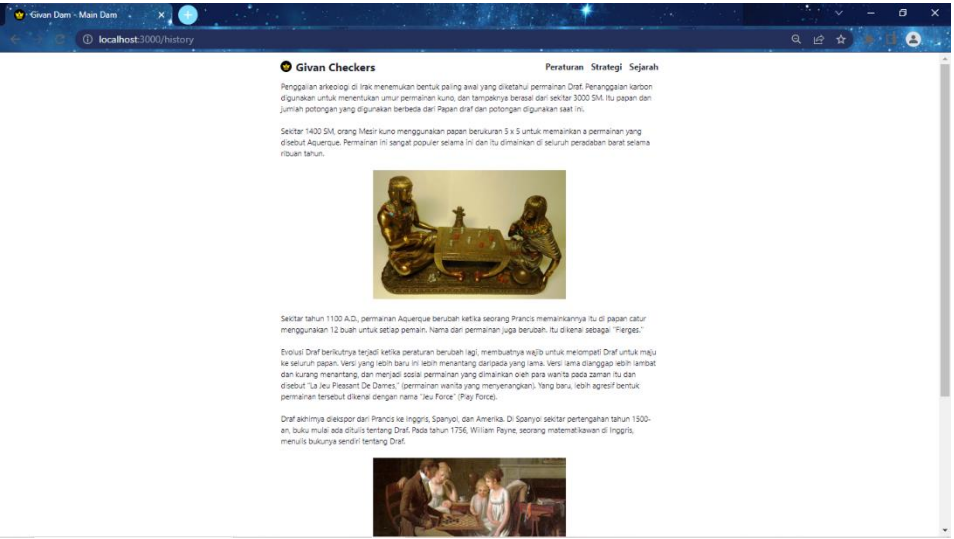
Teks Faktor	Hasil	Keterangan
User membuka halaman peraturan	✓	Berhasil, menampilkan halaman peraturan
Screenshot		
		

d. *Black Box Testing* Halaman Strategi

Table 4.9 *Black Box Testing* Halaman Strategi

Teks Faktor	Hasil	Keterangan
User membuka halaman strategi	✓	Berhasil, menampilkan menampilkan halaman strategi
Screenshot		
		

e. *Black Box Testing* Halaman SejarahTable 4.10 *Black Box Testing* Halaman Sejarah

Teks Faktor	Hasil	Keterangan
User membuka halaman sejarah	✓	Berhasil, menampilkan halaman sejarah
Screenshot		
		

H. Uji Aplikasi

Tabel 4.11 Pemain dan Komputer

No.	Tingkat Kesulitan	Menang – Seri – Kalah	Keterangan
1	Mudah	7 – 0 – 3 (70%)	10 permainan dari 10 orang
2	Sedang	4 – 0 – 6 (40%)	10 permainan dari 10 orang
3	Sulit	2 – 0 – 8 (20%)	10 permainan dari 10 orang

Dari tabel 4.11 dapat disimpulkan bahwa tingkat kemenangan pemain pada setiap tingkat kesulitan semakin menurun karena kedalaman yang digunakan semakin tinggi. Hasilnya adalah komputer menjadi lebih pintar karena dapat melakukan lebih banyak langkah terbaik.

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan penelitian yang telah dilakukan maka dapat diberikan kesimpulan sebagai berikut :

1. Peneliti berhasil merancang Aplikasi Game *Checkers* Dengan Menggunakan Metode *Alpha Beta*. Implementasi algoritma ini memungkinkan pengambilan keputusan yang lebih cerdas dan efisien, sehingga menghasilkan permainan yang lebih menantang bagi pengguna.
2. Dengan menggunakan *Alpha Beta*, aplikasi game *Checkers* akan dapat memberikan pengalaman bermain yang menantang dan menyenangkan bagi pemain. Algoritma ini tidak hanya meningkatkan kualitas permainan dari segi tantangan, tetapi juga membantu memperkuat keterampilan logika dan pemecahan masalah pemain. *Pruning* (pemangkasan) yang diterapkan dalam algoritma *Alpha – Beta* mempercepat proses pencarian langkah yang paling optimal, sehingga pemain dapat merasakan dinamika permainan yang lebih responsif dan menarik.

B. Saran

Pada penelitian ini penulis menyadari bahwa masih ada beberapa kekurangan yang sangat di perlu perbaikan dan pengembangan di penelitian selanjutnya. Oleh

karena itu, penulis memiliki beberapa saran untuk pengembangan selanjutnya sebagai berikut :

Untuk pengembangan lebih lanjut dari aplikasi game checkers ini, disarankan agar peningkatan antarmuka pengguna (*UI*) dan pengalaman pengguna (*UX*) sangat diperlukan agar aplikasi lebih menarik dan nyaman digunakan, misalnya melalui tutorial interaktif dan grafik yang lebih baik. Penambahan fitur analisis permainan juga penting untuk membantu pemain belajar dari setiap langkah yang diambil. Disarankan pula untuk melakukan penelitian pengguna secara berkala guna mengidentifikasi kebutuhan dan preferensi pemain yang dapat digunakan untuk pengembangan lebih lanjut. Terakhir, mengembangkan versi edukasi dari aplikasi ini dapat memberikan nilai tambah, terutama dalam melatih keterampilan logika dan pemecahan masalah bagi anak-anak atau pelajar, dengan fitur-fitur seperti level kesulitan yang dapat disesuaikan dan mode pelatihan khusus pendidikan.

DAFTAR PUSTAKA

- Adi Nugroho, (2005). *Konsep Pengembangan Sistem Basis Data Informatika*, Bandung.
- Adi Nugroho. (2010). *Rekayasa Perangkat Lunak Berbasis Objek dengan Metode USDP*. Andi. Yogyakarta.
- Aditya, Alan Nur. (2011). *Jago PHP & MySQL Dalam Hitungan Menit*. Jakarta: Dunia Komputer.
- Afrianto, I. (2015). *Rancang Bangun Aplikasi ChatBot Informasi Objek Wisata Bandung Dengan Pendekatan Natural Language Processing*. Komputa : Jurnal Ilmiah Komputer dan Informatika, 4(1), 49 – 54.
- Ardiansa, Susanto, Rahman, A., Yohannes, (2011). *Perbandingan Peforma Algoritma Minimax dan Negascout Pada Permainan Checkers Berbasis Android*.
- Arifianto, Teguh. (2011). *Membuat Interface Aplikasi Android Lebih Keren dengan LWUIT*. Yogyakarta: Andi Publisher.
- Ayuningtyas, N. (2008). *Algortima Minimax Dalam Permainan Checkers*. Dalam Strategi Algoritmik.
- Darmawan, R., & Hermawan, G. (2016). *Solusi Langkah pada Game Congklak Menggunakan Metode Negascout*. Komputa: Jurnal Ilmiah Komputer dan Informatika, 5(2), 99-105.
- David, (2013). *Perbandingan Algoritma Minimax dan Negascout Pada Permainan Catur Sederhana*. Jurnal Informatika Mulawarman.
- Diar Puji, Octavian. (2010). *Menjadi Programmer Jempolan Menggunakan PHP*. Penerbit MediaKom. Yogyakarta.
- Effendi, A. K., Delima, R., & Chrismanto, A. R. (2012). Implementasi Algoritma Negascout Untuk Permainan Checkers. *Jurnal Informatika*, 8(1).
- Gunawan, L. (2019). *Penerapan Algoritma Minimax Dan Alpha-Beta Pruning Pada Game Connect Four Berbasis Android* (Disertasi Doktor, STMIK ATMA LUHUR).
- Hartono, Jogiyanto. (1999). *Pengenalan Komputer*. Andi:Yogyakarta.

- Hasan, S., & Muhammad, N. (2020). *Sistem Informasi Pembayaran Biaya Studi Berbasis Web Pada Politeknik Sains Dan Teknologi Wiratama Maluku Utara. IJIS – Indonesian Jurnal On Information System*, 5(1), 44 – 55.
- Ilham, A. (2008). *Penerapan Algoritma Minimax dengan Optimasi MTD (f) pada Permainan Catur*. Jurnal, ITB, Bandung.
- Jogiyanto Hartono, (2005). *Analisis dan Desain Sistem Informasi. Pendekatan terstruktur teori dan praktis aplikasi bisnis*. Andi. Yogyakarta.
- Kadir, A. (2008). *Dasar Pemograman dan Implementasi Database Relational*. Yogyakarta: ANDI.
- KBBI, (2015). *Kamus Besar Bahasa Indonesia (KBBI)*. [Online] Available at: <http://kbbi.web.id/di> [Diakses 21 Juni 2020].
- Madcoms. (2012). *Manajemen Sistem Jaringan Komputer. Ed I*. Yogyakarta: Andi, Madiun: Madcoms.
- Munawar. (2005), *Pemodelan Visual dengan UML*, Graha Ilmu, Yogyakarta, 17-100.
- Nurdiansyah, A., Pratama, B. T., & Farhan, L. M. A. (2016). *Pembuatan Aplikasi Permainan Othello 16x16 Berbasis Desktop dengan Algoritma Alpha Beta Prunning*. SEMNASTEKNOMEDIA ONLINE, 4(1), 3-5.
- Pratama, A., P., & Marpaung, K. A. (2021). *Sistem Informasi Pendataan Warga Desa Berbasis Web Perum Surya Residence Blok 4E RT. 06/RW. 31* (Doctoral dissertation, UPN “VETERAN” Jatim).
- Putra, A. P. (2019). *Culture Shock dalam Komunikasi Antar Budaya* (Studi pada Penerima Beasiswa Erasmus + Periode 2018 Universitas Muhammadiyah Malang di Eropa) (Doctoral dissertation, University of Muhammadiyah Malang).
- Putra, G. F. A. (2016). *TA: Pembuatan Game Tentang Permainan Tradisional" Patil Lele" Bergenre Sports Game Sebagai Media Untuk Mengenalkan Kembali Bagi Anak-Anak* (Doctoral dissertation, Institut Bisnis dan Informatika Stikom Surabaya).
- Reinfeld, F. (2011). *How To win at Checkers*. Diakses 13 Juni 2022, dari <http://www.bobnewel.net/checkers/howtowin.pdf>
- Rianingtias, O. (2019). *Pengembangan Game Edukasi Berbasis Android Sebagai Media Pembelajaran Biologi Bernuansa Motivasi Siswa Kelas XI Di SMA/MA*. Universitas Islam Negeri Raden Intan Lampung.

- Salmon, R., & Nugraha, I. (2017). Penerapan Intelegensi Buatan Dengan Algoritma Negamax pada Aplikasi Permainan Deret Lima. *Jurnal Bangkit Indonesia*, 6(2).
- Sanjaya, E., Wonggo, T., & Angreni, R. (2015). *Penerapan Algoritma Minimax dan Memory-Enhanced Test Driver With Value pada Permainan Checkers*.
- Sebastian, V., S., Joko, P., M.Kom., & Nugroho, A., H., M.Si. (2020). *Implementasi Algoritma Negascout Pada Permainan Animal Chess*. *Jurnal Informatika* Vol. 9, No. 2.
- Septa, & Saifudin, A. (2019). *Penerapan Algoritma Finite State Machine Pada Game Horro 3D Untuk Melestarikan Budaya Tradisional Bangsa Berbasis Android*. *Jurnal Informatika Universitas Pamulang*.
- Setiawardhana, S. (2010). *Aplikasi Biro Jodoh Berbasis Web dan J2ME. Eepis final project*.
- Sholih. (2006). *Pemodelan Sistem Informasi Berorientasi Objek dengan UML*. Graha Ilmu. Yogyakarta.
- Sugiyono. (2001). *Metode Penelitian kuantitatif kualitatif dan R&D*. Bandung: Alfabeta.
- Syapnika, D., & Siagian, E. R. (2015). *Penerapan Algoritma Minimax Pada Permainan Checkers*. *JURIKOM (Jurnal Riset Komputer)*, 2(6).
- Triono, H., & Minarso, B. (2018). *Analisis Faktor-faktor yang Berpengaruh pada Minat Pemilihan Karir Akuntan Publik dan Akuntan Non Publik Mahasiswa Akuntansi*. *Jurnal Ilmiah Aset*, 20(2), 103-107.
- Universitas Muhammadiyah Parepare. (2021). *Panduan Akademik 2020-2021*, Parepare : UMPAR Pr

