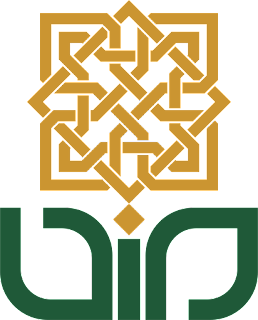
**LAPORAN TUGAS AKHIR**

**MATA KULIAH METODE PENGEMBANGAN PERANGKAT LUNAK SUDOKU4 (PERMAINAN SUDOKU MULTIPLAYER ONLINE BERBASIS**

**WEBSITE)**

****

Disusun Oleh:

|  |  |
| --- | --- |
| Ara Rosalia Safitri | (23106050021) |
| Chairul ‘Azmi Zuhdi Pramono | (23106050024) |
| Aimar Susanto  Fahalliza Nastitie Dewi | (23106050097)  (23106050030) |

**PROGRAM STUDI INFORMATIKA FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM NEGERI SUNAN KALIJAGA YOGYAKARTA**

**2025**

# KATA PENGANTAR

Puji syukur kami panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga kami dapat menyelesaikan laporan tugas akhir mata kuliah Metode Pengembangan Perangkat Lunak dengan judul "SUDOKU4 (PERMAINAN SUDOKU MULTIPLAYER ONLINE BERBASIS WEBSITE)" tepat pada waktunya.

Laporan ini disusun untuk memenuhi tugas akhir mata kuliah Metode Pengembangan Perangkat Lunak di Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Kalijaga Yogyakarta. Melalui pengembangan aplikasi Sudoku4, kami berusaha menerapkan konsep-konsep yang telah dipelajari dalam mata kuliah ini secara praktis, mulai dari tahap perencanaan, analisis, desain, hingga implementasi sebuah perangkat lunak.

Proyek Sudoku4 merupakan sebuah platform permainan Sudoku multiplayer online yang memungkinkan pengguna untuk bermain bersama secara real-time. Pengembangan aplikasi ini diharapkan dapat memberikan kontribusi positif dalam meningkatkan literasi numerasi sekaligus menyediakan platform hiburan yang edukatif dan interaktif.

Kami menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari pembaca untuk perbaikan di masa yang akan datang. Semoga laporan ini dapat bermanfaat bagi pembaca dan dapat menjadi referensi dalam pengembangan perangkat lunak serupa.

Akhir kata, kami mengucapkan terima kasih kepada semua pihak yang telah membantu dalam penyelesaian tugas akhir ini, khususnya kepada Bapak Agus Mulyanto , S.Si., M.Kom. dan Bapak Ir. Daru Prasetyawan, S.T., M.Kom. selaku dosen pengampu mata kuliah Metode Pengembangan Perangkat Lunak yang telah memberikan bimbingan dan arahan selama proses pembelajaran.

Yogyakarta, 12 Juni 2025

Kelompok 7

# DAFTAR ISI

[KATA PENGANTAR ii](#_bookmark0)

[DAFTAR ISI iii](#_bookmark1)

[DAFTAR TABEL v](#_bookmark2)

[DAFTAR GAMBAR vi](#_bookmark3)

[BAB I PENDAHULUAN 2](#_bookmark4)

* 1. [Latar Belakang 2](#_bookmark5)
  2. [Gambaran Tempat Studi Kasus 3](#_bookmark6)
  3. [Rumusan Masalah 3](#_bookmark7)
  4. [Batasan Masalah 3](#_bookmark8)
  5. [Tujuan Penelitian 4](#_bookmark9)
  6. [Manfaat Pengembangan 4](#_bookmark10)

[BAB II LANDASAN TEORI 5](#_bookmark11)

* 1. [Software Development Life Cycle 5](#_bookmark12)
  2. [Metode Pengembangan Sistem 6](#_bookmark13)
     1. [Metode Rapid Application Development 6](#_bookmark14)

[BAB III PERENCANAAN (PLANNING) 7](#_bookmark15)

* 1. [Alat dan Bahan 7](#_bookmark16)
  2. [Indentifikasi Kebutuhan Sistem 7](#_bookmark17)
  3. [Kelayakan Proyek 8](#_bookmark18)
     1. [Kelayakan Teknis 8](#_bookmark19)
     2. [Kelayakan Operasional 8](#_bookmark20)

[BAB IV ANALISIS (ANALYSIS) 9](#_bookmark21)

* 1. [Analisis Kebutuhan Fungsional 9](#_bookmark22)
     1. [Manajemen Pengguna 9](#_bookmark23)
     2. [Manajemen Lobi dan Ruang Permainan (Room): 9](#_bookmark24)
     3. [Fungsionalitas Gameplay 9](#_bookmark25)
     4. [Fitur Sosial dan Kompetitif: 10](#_bookmark26)
  2. [Analisis Kebutuhan Non Fungsional 10](#_bookmark27)
     1. [Kinerja (Performance): 10](#_bookmark28)
     2. [Keamanan (Security): 10](#_bookmark29)
     3. [Kegunaan (Usability): 11](#_bookmark30)
     4. [Ketersediaan (Availability): 11](#_bookmark31)
     5. [Kompatibilitas (Compatibility): 11](#_bookmark32)

[BAB V DESAIN (DESIGN) 12](#_bookmark33)

* 1. [Desain Unified Modelling Language (Diagram UML) 12](#_bookmark34)
  2. [Desain User Interface 12](#_bookmark36)
  3. [Desain Entity Relationship Diagram 15](#_bookmark42)
  4. [Desain Tabel 15](#_bookmark43)

[BAB VI IMPLEMENTASI (IMPLEMENTATION) 16](#_bookmark49)

* 1. [Implementasi Antar Muka 16](#_bookmark50)
  2. [Implementasi Login 18](#_bookmark56)
  3. [Implementasi Register 19](#_bookmark57)
  4. [Implementasi Dashboard 20](#_bookmark58)
  5. [Implementasi Lobby 24](#_bookmark59)
  6. [Implementasi Deathmatch 28](#_bookmark60)
  7. [Hasil pengujian 38](#_bookmark61)

[DAFTAR PUSTAKA 40](#_bookmark63)

[LAMPIRAN 41](#_bookmark64)

# DAFTAR TABEL

[Tabel 5.1 Tabel users 15](#_bookmark45)

[Tabel 5.2 Tabel lobbies 15](#_bookmark46)

[Tabel 5.3 Tabel lobby\_players 15](#_bookmark47)

[Tabel 5.4 Tabel lobby\_puzzles 15](#_bookmark48)

[Tabel 6.5 Hasil pengujian terhadap test case 38](#_bookmark62)

# DAFTAR GAMBAR

## [Gambar 5.1 Use case diagram 12](#_bookmark35)

## [Gambar 5.2 Halaman register 12](#_bookmark37)

## [Gambar 5.3 Halaman login 13](#_bookmark38)

## [Gambar 5.4 Halaman dashboard 13](#_bookmark39)

## [Gambar 5.5 Halaman matchmaking 14](#_bookmark40)

## [Gambar 5.6 Halaman deathmatch/gameplay 14](#_bookmark41)

## [Gambar 5.7 Entity relationship diagram 15](#_bookmark44)

## [Gambar 6.1 Halaman register 16](#_bookmark51)

## [Gambar 6.2 Halaman login 16](#_bookmark52)

## [Gambar 6.3 Halaman dashboard 17](#_bookmark53)

## [Gambar 6.4 Halaman matchmaking 17](#_bookmark54)

## [Gambar 6.5 Halaman deathmatch/gameplay 18](#_bookmark55)

* 1. Latar Belakang

# BAB I PENDAHULUAN

Metode Pengembangan Perangkat Lunak (MPPL) adalah salah satu mata kuliah dalam Program Studi Informatika yang membahas mengenai bagaimana alur suatu perangkat lunak dikembangkan dari tahap perencanaan hingga peluncuran (deploy). Mata kuliah ini berperan penting dalam proses pengembangan perangkat lunak karena pengetahuan mengenai pengelolaan proyek dan tim adalah kunci kesuksesan pengembangan perangkat lunak (Maikel Bolung, 2017). Pengembang juga perlu memahami kelebihan dan kekurangan setiap metode agar dapat menentukan metode yang sesuai karena metode yang digunakan menjadi gambaran perangkat lunak seperti apa yang akan dikembangkan.

Literasi numerasi merupakan kemampuan untuk mengimplementasikan pemahaman konsep bilangan dan operasi aritmatik sekaligus menganalisis informasi kuantitatif yang ada dalam kasus sehari-hari (Lintang Darmastuti, 2024). Selain menjadi tolak ukur implementasi pengetahuan matematika seseorang, literasi numerasi ini juga mengasah ketajaman dalam berpikir kritis dan logis. Sayangnya berdasarkan PISA tahun 2023, Indonesia menduduki peringkat 66 dari total 81 negara dengan skor literasi numerasi sebesar 366 yang mana 106 skor di bawah skor rata-rata internasional. Parahnya literasi numerasi di Indonesia juga terlihat dari pemahaman siswa SMP yang mayoritas hanya bisa memahami satu indikator dari tiga indikator literasi numerasi. Menurut Sirilivia Khunaeni tahun 2024, mayoritas siswa mampu menyelesaikan masalah numerasi dengan memanfaatkan angka atau simbol matematika dasar yang mudah digunakan dalam kehidupan sehari-hari. Sedangkan sebagian besar kesulitan menyelesaikan masalah menggunakan grafik, diagram, atau tabel dan hanya sejumlah siswa yang mampu menafsirkan hasil analisis untuk mengambil sekaligus memprediksi keputusan.

Sejak akhir abad ke-20, sebuah permainan papan untuk mengombinasikan bilangan 1 hingga 9 telah dikembangkan di New York dan kemudian dipopulerkan di Jepang dengan nama Sudoku. Permainan Sudoku ini mengharuskan pemainnya untuk menganalisis pola bilangan dalam 9 buah tabel berukuran 3x3 agar bisa menempatkan bilangan 1 hingga 9 dalam semua kolom dan baris tanpa adanya duplikat bilangan serupa. Sehingga pemain perlu menganalisis informasi bilangan

yang tersedia di papan Sudoku untuk menemukan pola yang tepat dari 6,670,903,752,021,072,936,960 kemungkinan yang dapat terbentuk. Oleh karena itu, perlunya pemahaman mengenai pola dari setiap bilangan membuat permainan Sudoku ini cocok untuk melatih kemampuan literasi numerasi.

Pengembangan Sudoku4 berbasis website yang memanfaatkan sistem PvP ini diharapkan mampu memberikan pemahaman literasi numerasi secara tidak sadar kepada siswa dan mahasiswa di Indonesia sembari bersenang-senang. Sudoku4 ini juga diharapkan mampu meningkatkan rasa kompetitif yang sehat antar pemain dengan menghadirkan leaderboard. Penambahan batas waktu dalam bermain Sudoku4 ini juga diharapkan dapat meningkatkan kemampuan berpikir kritis.

## Gambaran Tempat Studi Kasus

Studi kasus pengembangan website “Sudoku4" ini dilakukan pada Program Studi Informatika di Universitas Islam Negeri Sunan Kalijaga. Pengembangan website ini memanfaatkan bahasa pemrograman native dan database online dalam sistem matchmaking untuk mendukung permainan real-time. Pengujian aplikasi dilakukan di Ruang 102 Fakultas Sains dan Teknologi dengan melibatkan mahasiswa sebagai responden dalam uji coba aplikasi.

## Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah pengembangan ini adalah:

* + 1. Bagaimana cara merancang dan mengembangkan website Sudoku4 yang bisa dimainkan secara real-time oleh empat pemain?
    2. Sistem apa saja yang dibutuhkan agar permainan menjadi optimal?
    3. Apakah pemain dapat memahami dan menguasai cara bermain Sudoku4 secara mudah?

## Batasan Masalah

Batasan masalah yang digunakan pengembangan ini agar hasilnya terfokus adalah:

* + 1. Pengembangan website ini lebih berfokus pada platform desktop daripada mobile.
    2. Sudoku4 ini hanya dikembangkan untuk permainan mode PvP secara real- time.
    3. Fitur yang dimiliki Sudoku4 hanya mencakup edit username, papan permainan, matchmaking, cek bilangan, perhitungan skor, dan leaderboard.
    4. Enkripsi dan autentikasi data yang dikembangkan hanya di tingkat sederhana saja.
  1. Tujuan Penelitian Pengembangan ini bertujuan:
     1. Untuk mengembangkan Sudoku yang dapat dimainkan dalam mode PvP secara real-time dengan metode pengembangan yang sesuai.
     2. Untuk menerapkan sistem yang tepat dalam pengembangan fitur-fitur utama permainan yang mencakup edit username, papan permainan, matchmaking, cek bilangan, perhitungan skor, dan leaderboard.
     3. Untuk menghasilkan sistem Sudoku yang mudah dipahami dan dikuasai oleh pemain awam serta meningkatkan ketertarikannya dalam bermain Sudoku.

## Manfaat Pengembangan

Manfaat dari pengembangan ini antara lain:

* + 1. Meningkatkan kemampuan literasi numerasi yang menyenangkan dengan cara bermain game.
    2. Memberikan pengalaman bermain yang kompetitif antar pemain.
    3. Mengalihkan rasa kecanduan bermain game online ke arah yang lebih bermanfaat.

# BAB II LANDASAN TEORI

## Software Development Life Cycle

Software Development Life Cycle (SDLC) adalah kerangka kerja yang menjadi acuan pengembang mengenai tahapan yang harus dilalui untuk mengembangkan perangkat lunak yang efisien. Kehadiran SDLC ini membantu pengembang untuk menata alur kerja yang terstruktur dan sistematis sehingga hasil pengembangan sesuai dengan kebutuhan pengguna dengan tetap mempertahankan kualitas yang tinggi. Selain itu, SDLC juga mempermudah pengembang dalam mengelola dan memelihara perangkat lunak yang dikembangkan. Tahapan yang perlu dilalui dalam SDLC secara umum terdiri dari:

1. Perencanaan: Menentukan arah dari pengembangan perangkat lunak serta sumber daya yang diperlukan.
2. Requirement Analysis: Mengumpulkan dan menentukan kebutuhan fungsional dan non-fungsional yang dibutuhkan pengguna.
3. Design: Merancang struktur sistem dan antar muka yang memenuhi kebutuhan pengguna.
4. Implementasi: Proses pengkodean dan pengembangan perangkat lunak yang sesuai dengan hasil design.
5. Testing: Menguji perangkat lunak agar optimal dan sesuai dengan kebutuhan pengguna sekaligus mencari bug atau error yang mungkin muncul agar tidak mengganggu performa perangkat lunak ketika digunakan pengguna.
6. Deploy dan Maintenance: Merilis perangkat lunak agar bisa digunakan di lingkungan pengguna serta mengawasi kemungkinan munculnya bug selama penggunaan dan penambahan fitur di masa depan.

Pada pengembangan ini, SDLC digunakan sebagai acuan dalam proses pengembangan Sudoku4 agar hasilnya sesuai dengan kebutuhan awal yang direncanakan yaitu permainan Sudoku yang dapat dimainkan dalam mode PvP secara real-time.

## Metode Pengembangan Sistem

Secara umum metode pengembangan yang menerapkan SDLC terbagi menjadi tiga jenis, yaitu metode Waterfall, Agile, dan Iteratif. Dalam pengembangan ini, metode yang digunakan adalah metode Rapid Application Development (RAD) yang merupakan salah satu turunan dari metode iteratif. Metode ini dipilih karena pengembangan ini berfokus pada hasil jadi yang dapat segera diuji untuk melakukan evaluasi di iterasi berikutnya.

## Metode Rapid Application Development

Rapid Application Development (RAD) adalah metode pengembangan perangkat lunak yang menekankan pada kecepatan dalam pembuatan sistem melalui penggunaan prototipe, keterlibatan aktif pengguna, dan pengulangan proses pengembangan (iteratif). Tahapan yang perlu dilalui dalam metode ini antara lain:

* + - 1. Requirement Analysis: Mengumpulkan dan menganalisa kebutuhan pengguna secara cepat dan general.
      2. Design: Merancang database, desain sistem, dan desain antar muka secara cepat.
      3. Prototipe: Mengembangkan sebagian perangkat lunak yang dapat diuji langsung oleh pengguna.
      4. Testing: Prototipe akan diuji untuk mencari bug atau bagian yang belum sesuai kebutuhan awal kemudian proses akan diulang kembali dari design.
      5. Implementasi: Setelah prototipe dirasa memenuhi kebutuhan pengguna dan tidak ada bug yang memengaruhi performa perangkat lunak, maka dilakukan konfigurasi keseluruhan prototipe menjadi perangkat lunak final yang dapat dirilis.

Berdasarkan tahapan di atas, metode RAD ini sangat cocok untuk pengembangan Sudoku4 yang perlu banyak feedback dari pengguna. Metode ini juga mendukung perubahan sistem yang diperlukan secepatnya berdasarkan feedback pengguna.

* 1. Alat dan Bahan

# BAB III PERENCANAAN (PLANNING)

Adapun alat dan bahan yang dibutuhkan untuk membuat program Sudoku4 ini adalah:

1. Figma

Figma digunakan untuk merancang tampilan antarmuka pengguna (UI) secara visual. Alat ini memungkinkan perancangan prototipe halaman seperti login, dashboard, lobby, dan gameplay secara kolaboratif dan interaktif sebelum diimplementasikan ke dalam kode program.

1. Visual Studio Editor Code

Visual Studio Code (VS Code) merupakan editor teks utama yang digunakan untuk menulis dan mengelola kode sumber dari proyek ini. Editor ini mendukung berbagai bahasa pemrograman dan dilengkapi dengan ekstensi seperti Live Server, Firebase Tools, dan Git Integration yang sangat membantu dalam proses pengembangan web.

1. Git & Github

Git digunakan sebagai sistem kontrol versi untuk mencatat dan mengelola setiap perubahan kode selama proses pengembangan. GitHub digunakan sebagai repositori daring untuk menyimpan, mengamankan, dan memfasilitasi kolaborasi kode antar pengembang secara real-time.

1. Firebase

Firebase digunakan sebagai layanan backend untuk menyimpan data pengguna dan permainan secara real-time. Layanan ini juga digunakan untuk autentikasi pengguna serta hosting aplikasi web. Penggunaan Firebase sangat membantu mengurangi kompleksitas dan biaya dalam pengelolaan server.

1. Laptop

Laptop berfungsi sebagai perangkat utama untuk seluruh aktivitas pengembangan, mulai dari perancangan UI, penulisan kode, integrasi database, hingga pengujian aplikasi.

## Indentifikasi Kebutuhan Sistem

Saat ini, permainan Sudoku banyak tersedia dalam format *single-player*. Keterbatasan ini mengurangi aspek sosial dan kompetitif dari permainan. Penggemar

Sudoku tidak memiliki platform yang terdedikasi untuk dapat bermain bersama teman secara *real-time*, membandingkan kecepatan dan ketepatan dalam menyelesaikan teka-teki, atau berkompetisi dalam sebuah papan peringkat. Oleh karena itu, diperlukan sebuah platform berbasis web yang menyediakan pengalaman bermain Sudoku secara multiplayer, interaktif, dan kompetitif.

## Kelayakan Proyek

## Kelayakan Teknis

Dari sisi teknis, proyek ini dinilai cukup layak untuk dikembangkan. Secara umum, seluruh komponen yang dibutuhkan telah tersedia dan mudah diimplementasikan dengan teknologi yang umum digunakan. Pada sisi frontend, aplikasi ini dapat dibangun menggunakan teknologi HTML, CSS, dan JavaScript. Untuk bagian backend, dapat menggunakan JavaScript yang terintegrasi dengan layanan Firebase. Pemanfaatan Firebase sebagai layanan basis data dan autentikasi memberikan keuntungan dalam hal kemudahan penggunaan serta mengurangi biaya pengelolaan infrastruktur secara keseluruhan.

## Kelayakan Operasional

Secara operasional, sistem ini juga dinilai layak untuk diterapkan. Dari sisi penerimaan pengguna, permainan Sudoku sudah memiliki basis pengguna yang cukup luas. Penambahan fitur multiplayer akan menjadi nilai tambah yang signifikan, karena memungkinkan pengalaman bermain yang lebih seru dan kompetitif. Antarmuka yang dirancang secara intuitif akan memudahkan pengguna baru dalam mengakses dan memainkan permainan. Sementara itu, dari sisi pemeliharaan, sistem ini tidak memerlukan prosedur operasional harian yang kompleks. Pemeliharaan lebih difokuskan pada pemantauan server, pembaruan keamanan, serta penanganan bug yang dilaporkan pengguna. Seluruh operasional tersebut dapat dikelola oleh satu orang atau tim pengembang dalam skala kecil.

# BAB IV ANALISIS (ANALYSIS)

## Analisis Kebutuhan Fungsional

Kebutuhan fungsional adalah fungsi utama yang harus dimiliki oleh sistem agar dapat digunakan sesuai tujuan. Pada sistem Website Game Online Multiplayer Sudoku4, kebutuhan fungsional dikategorikan ke dalam beberapa bagian berikut.

## Manajemen Pengguna

Sistem harus menyediakan halaman registrasi yang memungkinkan pengguna baru untuk membuat akun dengan cara mengisi alamat email dan password. Proses registrasi harus disertai dengan validasi agar sistem dapat mendeteksi apabila email atau username yang dimasukkan sudah pernah digunakan sebelumnya. Setelah registrasi berhasil, pengguna dapat melakukan login melalui halaman login yang disediakan, dengan menggunakan kombinasi email dan password yang valid. Selain itu, sistem harus menyediakan halaman profil bagi pengguna yang telah login, di mana pengguna dapat melihat informasi pribadi dan statistik permainan, seperti total skor atau jumlah kemenangan yang telah diperoleh selama bermain.

## Manajemen Lobi dan Ruang Permainan (Room):

Dalam sistem ini, pengguna tidak memilih ruang permainan secara manual, melainkan ditempatkan secara otomatis ke dalam ruang permainan publik secara acak oleh sistem. Pengguna yang masuk ke dalam lobby akan terdaftar sebagai bagian dari sebuah ruang permainan yang dikelola oleh sistem. Di dalam lobby tersebut, sistem akan mengatur daftar pemain, status masing-masing pemain, dan menetapkan salah satu pengguna sebagai host permainan.

## Fungsionalitas Gameplay:

Sistem harus menampilkan papan permainan Sudoku yang identik untuk semua pemain dalam satu ruang permainan. Pemain dapat mengisi angka dari satu hingga sembilan ke dalam sel kosong pada papan Sudoku. Ketika pemain memasukkan angka yang tidak valid atau bertentangan

dengan aturan Sudoku, sistem harus memberikan umpan balik visual secara real-time agar pemain mengetahui kesalahan yang dilakukan. Sistem juga harus mampu mendeteksi ketika seorang pemain berhasil menyelesaikan seluruh papan Sudoku dengan benar dan memberikan penilaian berdasarkan waktu atau ketepatan.

## Fitur Sosial dan Kompetitif:

Untuk mendukung interaksi antar pemain, sistem perlu menyediakan fitur chat sederhana di dalam ruang permainan, sehingga pemain dapat saling berkomunikasi selama permainan berlangsung. Setelah permainan berakhir, sistem harus menghitung skor dari setiap pemain berdasarkan performa mereka dan menampilkan hasil tersebut di layar. Sistem juga harus menyediakan leaderboard atau papan peringkat global yang menampilkan posisi pemain berdasarkan skor tertinggi atau jumlah kemenangan yang telah dicapai selama bermain.

## Analisis Kebutuhan Non Fungsional

Kebutuhan non-fungsional mencakup aspek-aspek kualitas sistem yang tidak berhubungan langsung dengan fitur, namun berpengaruh terhadap pengalaman pengguna, keandalan, dan efektivitas sistem secara keseluruhan. Berikut adalah kebutuhan non-fungsional dari sistem ini:

## Kinerja (Performance):

Sistem harus mampu memberikan respon cepat dalam setiap interaksi pengguna, terutama saat pemain memasukkan angka ke dalam sel Sudoku dan saat sistem melakukan validasi. Respon waktu yang ideal adalah kurang dari 500 milidetik agar permainan tetap terasa lancar dan real-time. Selain itu, sistem harus dirancang untuk dapat menangani minimal empat pemain yang bermain secara bersamaan dalam satu ruang permainan.

## Keamanan (Security):

Demi menjaga keamanan data pengguna, sistem harus menyimpan password dalam bentuk terenkripsi menggunakan algoritma hashing yang kuat seperti bcrypt. Setiap proses otentikasi dan penyimpanan data harus

dilindungi agar tidak mudah diretas atau disalahgunakan oleh pihak yang tidak berwenang.

## Kegunaan (Usability):

Antarmuka pengguna harus dirancang agar intuitif dan mudah digunakan, bahkan oleh pengguna yang baru pertama kali menggunakan aplikasi. Sistem perlu menyediakan panduan mengenai aturan permainan dan cara menggunakan fitur-fitur utama. Desain antarmuka juga harus responsif sehingga tetap dapat digunakan dengan nyaman di berbagai ukuran layar, baik desktop maupun tablet.

## Ketersediaan (Availability):

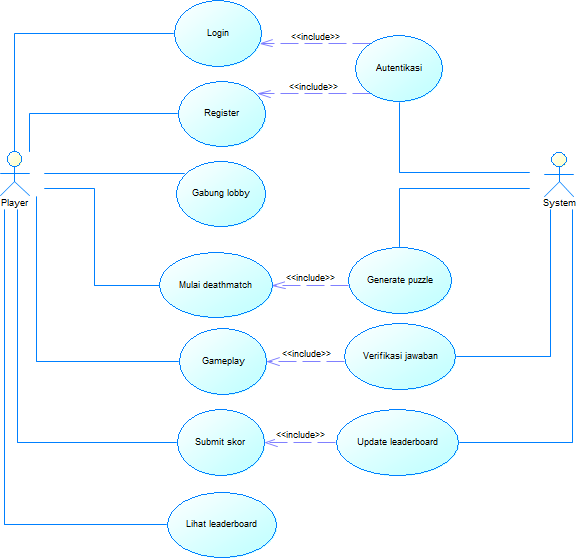
Website Sudoku4 ditargetkan memiliki tingkat ketersediaan sebesar 99% selama waktu operasional. Jika dibutuhkan pemeliharaan sistem, maka downtime harus dilakukan di luar jam sibuk untuk meminimalisir gangguan terhadap pengguna.

## Kompatibilitas (Compatibility):

Sistem harus kompatibel dan dapat dijalankan dengan lancar pada berbagai browser web populer seperti Google Chrome, Mozilla Firefox, Microsoft Edge, dan Safari versi terbaru. Hal ini memastikan bahwa pengguna tidak mengalami kendala hanya karena perbedaan platform atau perangkat yang digunakan.

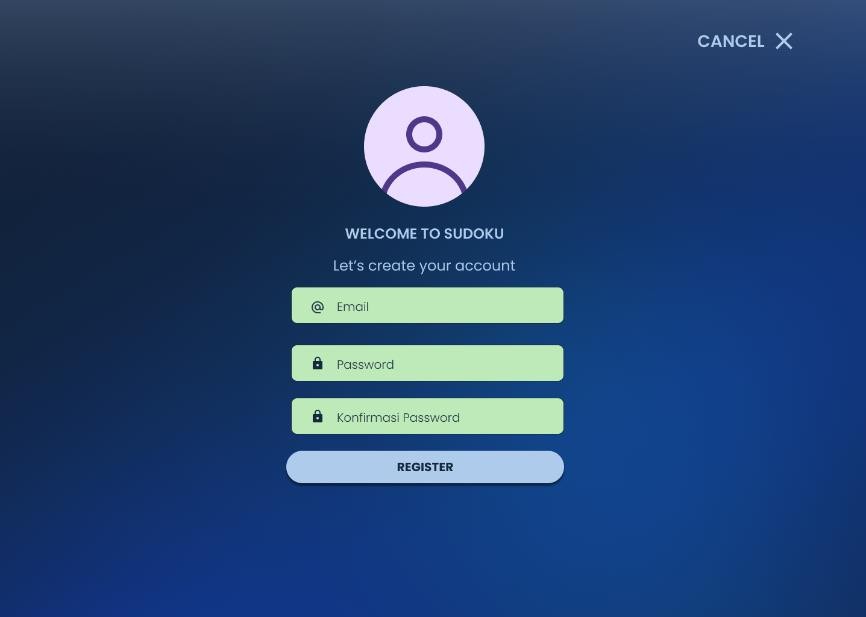
# BAB V DESAIN (DESIGN)

## Desain Unified Modelling Language (Diagram UML)

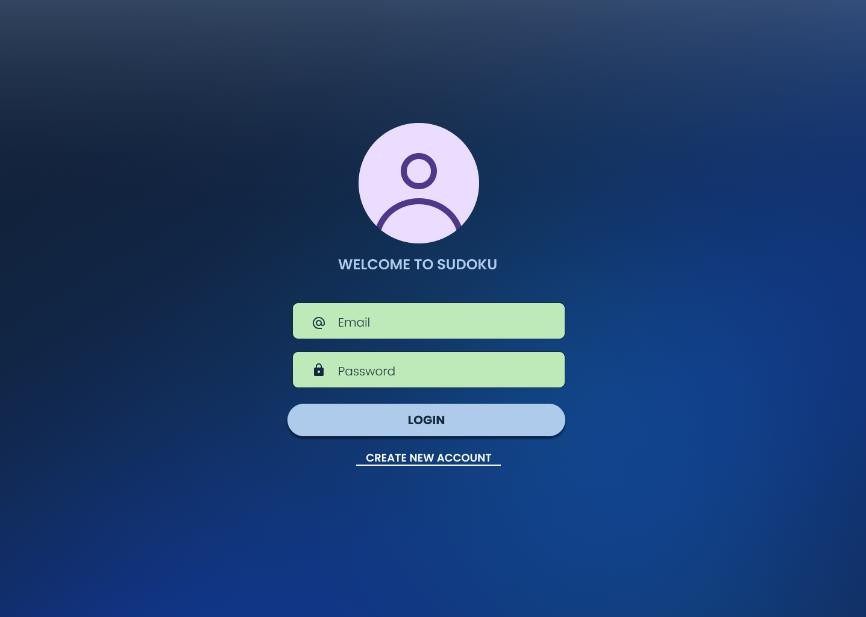


Gambar 5.1 Use case diagram

## Desain User Interface



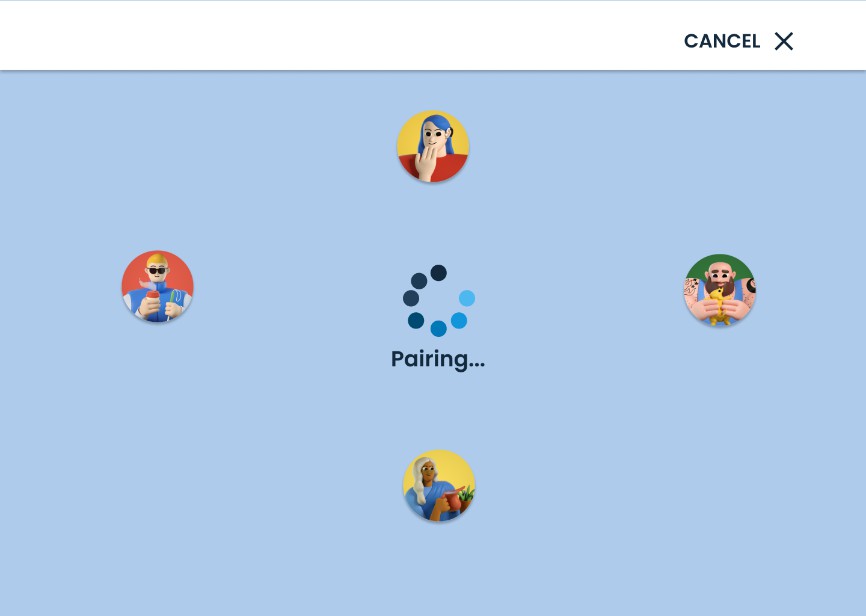
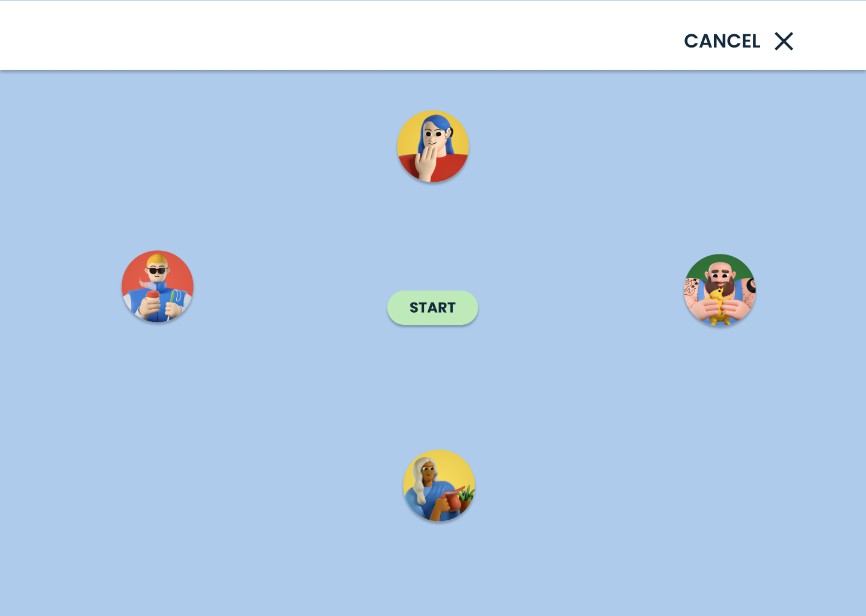
Gambar 5.*2* Halaman register



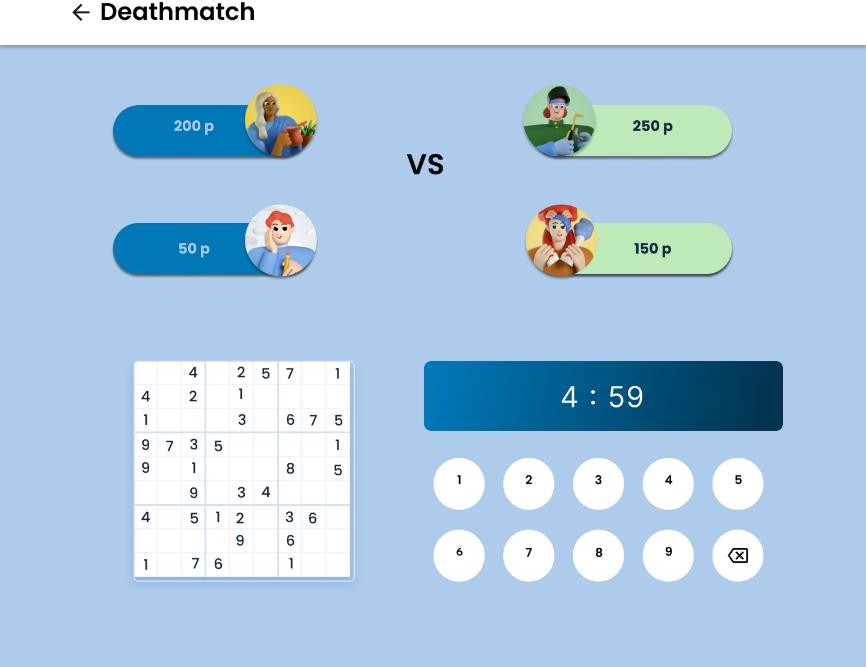
Gambar 5.3 Halaman login



Gambar 5.4 Halaman dashboard

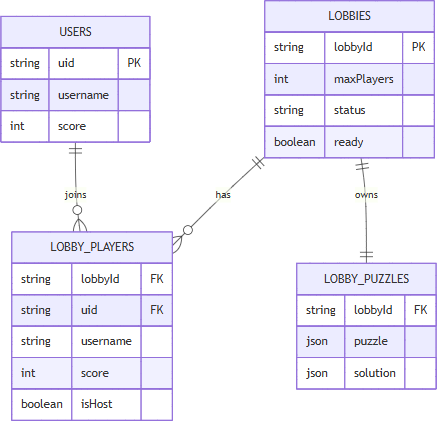


Gambar 5.5 Halaman matchmaking



Gambar 5.6 Halaman deathmatch/gameplay

## Desain Entity Relationship Diagram



## Desain Tabel

Gambar 5.7 Entity relationship diagram

Tabel 5.1 Tabel users

|  |  |  |
| --- | --- | --- |
| USERS | | |
| string | uid | PK |
| string | username |  |

Tabel 5.2 Tabel lobbies

|  |  |  |
| --- | --- | --- |
| LOBBIES | | |
| string | lobbyId | PK |
| int | maxPlayers |  |
| string | status |  |
| boolean | ready |  |

Tabel 5.3 Tabel lobby\_players

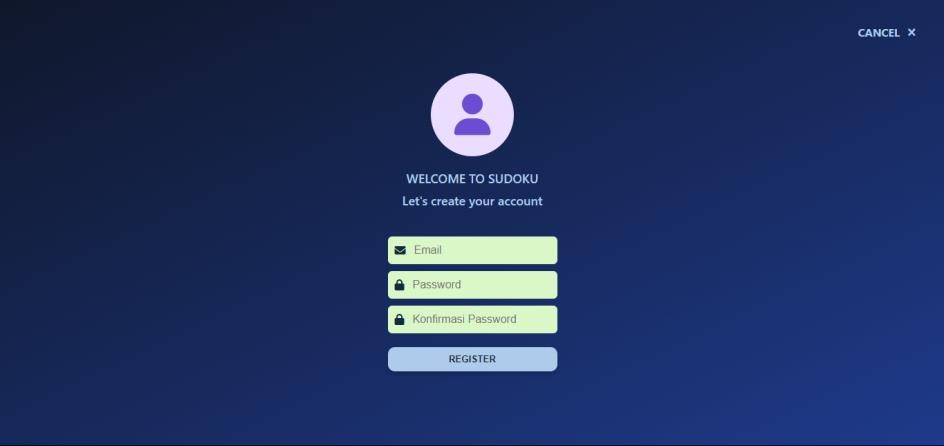
|  |  |  |
| --- | --- | --- |
| LOBBY\_PLAYERS | | |
| string | lobbyId | FK |
| string | uid | FK |
| string | username | FK |
| int | score |  |
| boolean | isHost |  |

Tabel 5.4 Tabel lobby\_puzzles

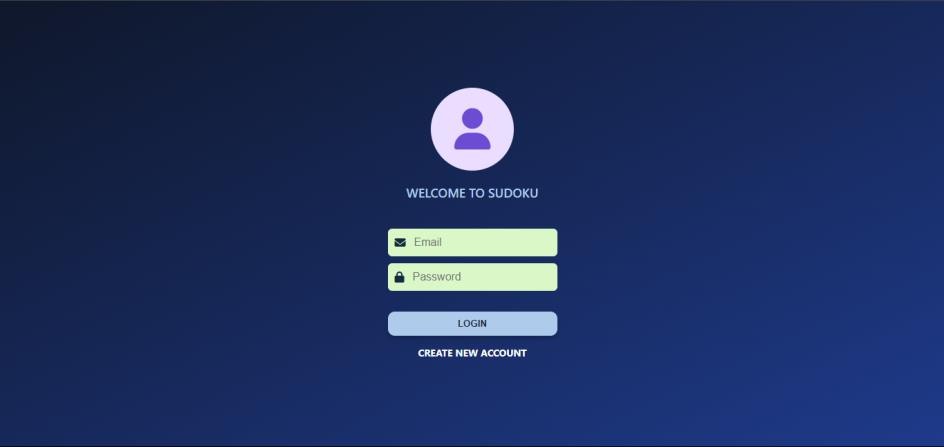
|  |  |  |
| --- | --- | --- |
| LOBBY\_PUZZLES | | |
| string | lobbyId | FK |
| json | puzzle |  |
| json | solution |  |

# BAB VI IMPLEMENTASI (IMPLEMENTATION)

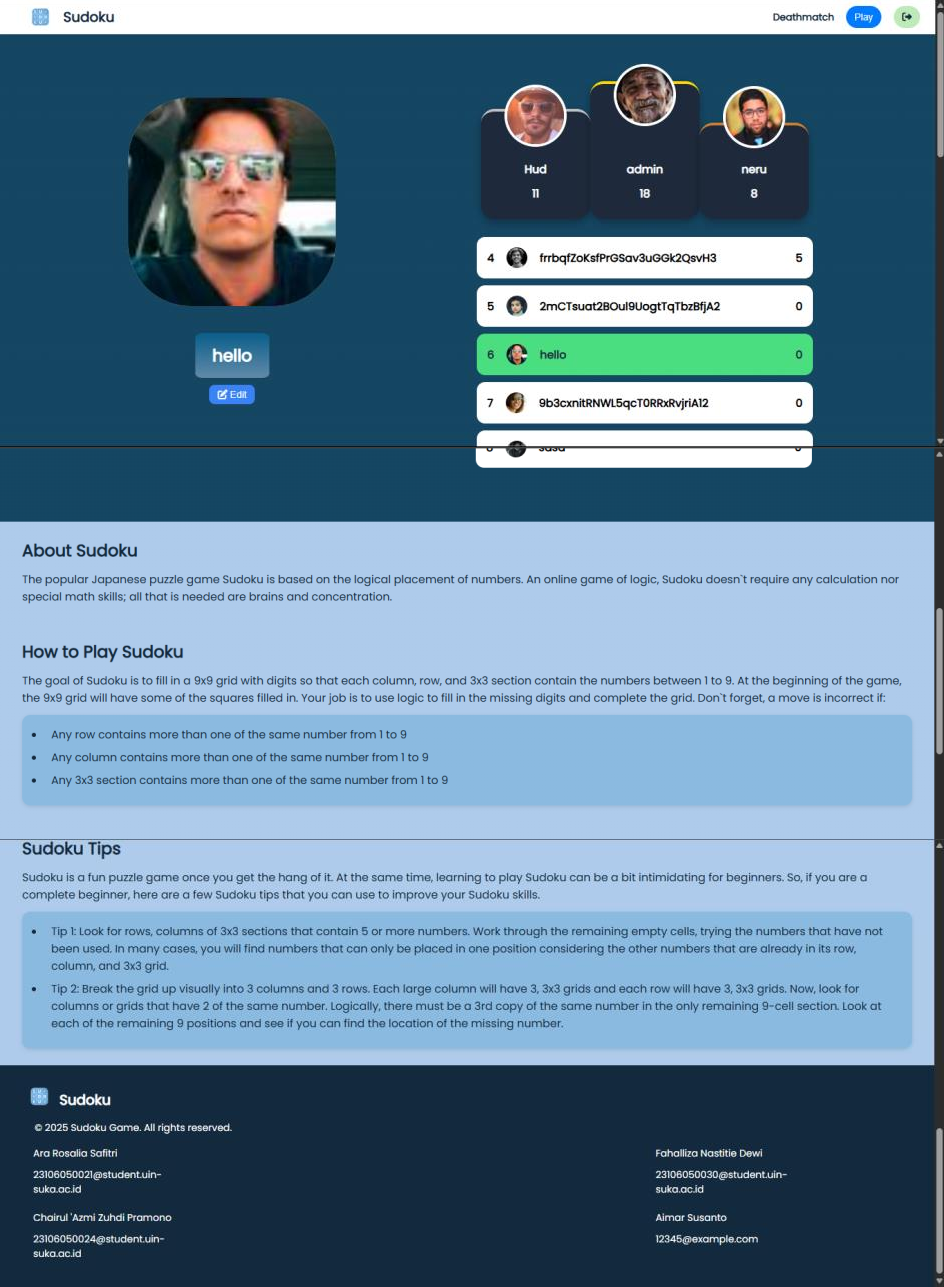
## Implementasi Antar Muka



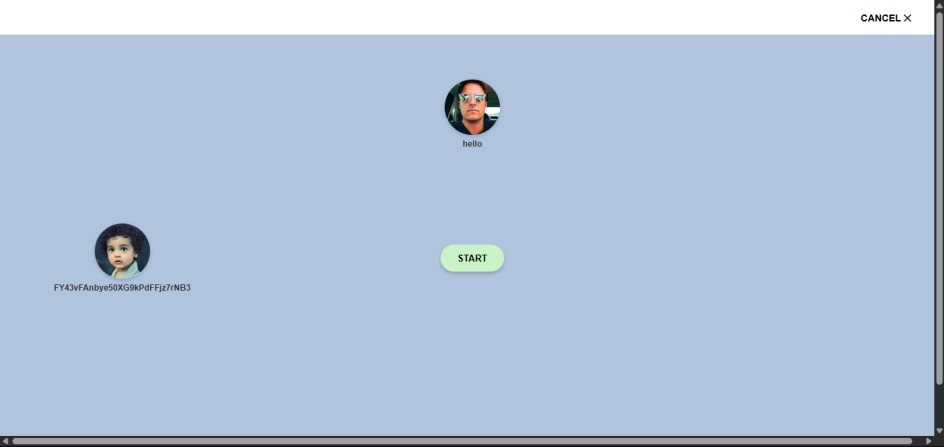
Gambar 6.1 Halaman register



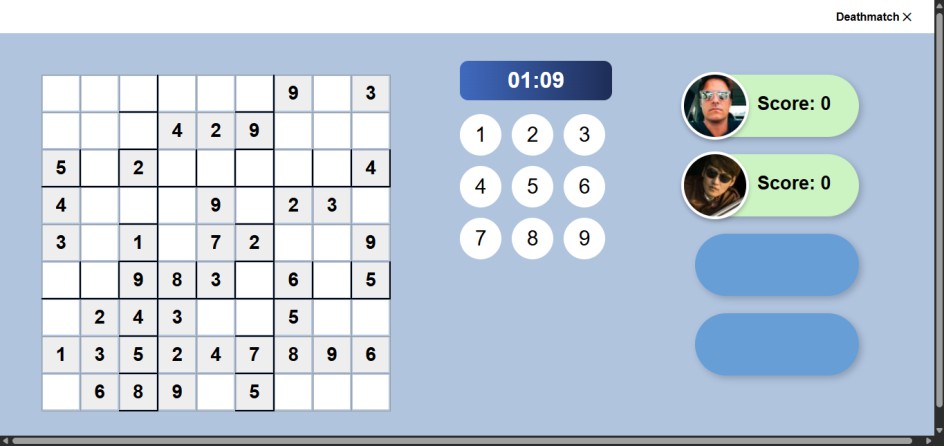
Gambar 6.2 Halaman login



Gambar 6.3 Halaman dashboard



Gambar 6.4 Halaman matchmaking



Gambar 6.5 Halaman deathmatch/gameplay

## Implementasi Login

<!-- Firebase SDK -->

<script type="module">

import { initializeApp } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-app.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-app.js)

import { getAuth, signInWithEmailAndflassword, signOut, onAuthStateChanged

} from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js)

const firebaseConfig = { apiKey: "", authDomain: "", projectId: "", storageBucket: "", messagingSenderId: "", appId: ""

};

const app = initializeApp(firebaseConfig); const auth = getAuth(app);

window.login = () => {

const email = document.getElementById("email").value;

const password = document.getElementById("password").value;

signInWithEmailAndflassword(auth, email, password)

.then(() => {

document.getElementById("status").innerText = "Login berhasil!";

})

.catch((error) => {

document.getElementById("status").innerText = "Email/flassword

Invalid";

});

};

onAuthStateChanged(auth, (user) => { if (user) {

window.location.href = "dashboard.html";

}

});

</script>

## Implementasi Register

<script type="module">

import { initializeApp } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-app.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-app.js)

import { getAuth, createUserWithEmailAndflassword, signInWithEmailAndflassword, signOut, onAuthStateChanged } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js)

import { getDatabase, ref, set, get } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-database.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-database.js)

const firebaseConfig = { apiKey: "", authDomain: "", projectId: "", storageBucket: "", messagingSenderId: "", appId: "", databaseURL:""

};

const app = initializeApp(firebaseConfig); const auth = getAuth(app);

const db = getDatabase(app);

window.register = () => {

const email = document.getElementById("email").value;

const password = document.getElementById("password").value; const passwordtry = document.getElementById("passwordtry").value;

if (password === passwordtry) { createUserWithEmailAndflassword(auth, email, password)

.then((userCredential) => {

const user = userCredential.user; const uid = user.uid;

const userRef = ref(db, 'users/' + uid);

return set(userRef, { email: user.email, username: uid, score: 0

});

}).then(() => {

document.getElementById("status").innerText = "Akun berhasil dibuat!

Silakan login.";

window.location.href = "login.html";

}).catch((error) => { document.getElementById("status").innerText = `Gagal daftar:

${error.message}`;

});

} else {

document.getElementById("status").innerText = "flassword tidak sesuai.";

}

};

</script>

## Implementasi Dashboard

<script type="module">

import { initializeApp } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-app.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-app.js)

import { getAuth, onAuthStateChanged, signOut } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js)

import { getDatabase, ref, set, get, child } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-database.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-database.js)

const firebaseConfig = { apiKey: "", authDomain: "", databaseURL: "", projectId: "", storageBucket: "", messagingSenderId: "", appId: ""

};

const app = initializeApp(firebaseConfig); const auth = getAuth(app);

const db = getDatabase(app); const dbRef = ref(db);

let currentUID = null;

window.toggleUserEdit = function () { document.getElementById("Userfllace").classList.toggle("hidden"); document.getElementById("UserEdit").classList.toggle("hidden");

};

document.addEventListener("DOMContentLoaded", () => { onAuthStateChanged(auth, (user) => {

if (user) {

const email = user.email; const uid = user.uid; currentUID = uid;

const avatarImg = document.querySelector(".user-avatar"); if (avatarImg) {

avatarImg.src = `https://i.pravatar.cc/80ou=${uid}`;

}

const userRef = ref(db, 'users/' + uid); get(userRef).then((snapshot) => {

if (snapshot.exists()) {

const data = snapshot.val(); document.getElementById("displayUsername").innerText = data.username

|| "Blanc";

document.getElementById("username").value = data.username || "";

} else {

window.location.href = "login.html";

}

});

document.getElementById("change").addEventListener("click", () => { const newUsername = document.getElementById("username").value; const updatedData = {

email: email, username: newUsername, score: 0

};

set(userRef, updatedData).then(() => { alert("Username diperbarui!");

document.getElementById("displayUsername").innerText = newUsername; toggleUserEdit();

});

});

} else {

window.location.href = "login.html";

}

});

get(child(dbRef, 'users')).then((snapshot) => { if (snapshot.exists()) {

const users = snapshot.val(); const leaderboardArray = [];

for (const uid in users) { const user = users[uid]; leaderboardArray.push({

uid: uid,

username: user.username || "Blanc", score: user.score || 0

});

}

leaderboardArray.sort((a, b) => b.score - a.score);

const top3 = leaderboardArray.slice(0, 3); const rest = leaderboardArray.slice(3);

const top3Container = document.getElementById("top3"); top3Container.innerHTML = "";

const positions = ["second", "first", "third"]; const rankumbers = [2, 1, 3];

[1, 0, 2].forEach((index, pos) => { const user = top3[index];

const div = document.createElement("div");

const avatarUrl = `https://i.pravatar.cc/80ou=${user.uid}`; div.classame = `top-card ${positions[pos]}`; div.innerHTML = `

<img src="${avatarUrl}" alt="Avatar"/>

<div class="username">${user.username}</div>

<div class="score">${user.score}</div>

`; top3Container.appendChild(div);

});

const leaderboardList = document.getElementById("leaderboard"); leaderboardList.innerHTML = "";

const middleRanks = rest.slice(0, 5); let currentUserRendered = false;

middleRanks.forEach((user, index) => { const li = document.createElement("li");

const isCurrentUser = user.uid === currentUID;

const avatarUrl = `https://i.pravatar.cc/80ou=${user.uid}`; li.classame = isCurrentUser o "you" : "";

li.innerHTML = `

<span class="rank">${index + 4}</span>

<img src="${avatarUrl}" alt="Avatar" class="avatar"/>

<span class="uname">${user.username}</span>

<span class="scores">${user.score}</span>

`; leaderboardList.appendChild(li);

if (isCurrentUser) { currentUserRendered = true;

}

});

// Cek apakah user ada di top3

const inTop3 = top3.some(user => user.uid === currentUID);

currentUID);

if (!inTop3 fifi !currentUserRendered) { const allUsers = [...top3, ...rest];

const currentUserIndex = allUsers.findIndex(user => user.uid ===

if (currentUserIndex !== -1) {

const currentUser = allUsers[currentUserIndex]; const avatarUrl =

`https://i.pravatar.cc/80ou=${currentUser.uid}`;

const li = document.createElement("li"); li.classame = "you";

li.innerHTML = `

<span class="rank">${currentUserIndex + 1}</span>

<img src="${avatarUrl}" alt="Avatar" class="avatar"/>

<span class="uname">${currentUser.username}</span>

<span class="scores">${currentUser.score}</span>

`; leaderboardList.appendChild(li);

}

}

}

});

});

window.logout = () => { signOut(auth).then(() => {

window.location.href = "login.html";

}).catch((error) => { console.error("Logout gagal:", error);

});

};

</script>

## Implementasi Lobby

<script type="module">

import { initializeApp } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-app.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-app.js)

import { getAuth, onAuthStateChanged } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js)

import { getDatabase, ref, get, push, set, update, onValue } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-database.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-database.js)

const firebaseConfig = { apiKey: "", authDomain: "", databaseURL: "", projectId: "", storageBucket: "", messagingSenderId: "", appId: ""

};

const app = initializeApp(firebaseConfig); const auth = getAuth(app);

const db = getDatabase(app);

let currentRoomId = "";

onAuthStateChanged(auth, async (user) => { if (user) {

const uid = user.uid;

// Cari room yang belum penuh const lobbyRef = ref(db, 'lobby');

const snapshot = await get(lobbyRef); let found = false;

if (snapshot.exists()) {

const lobbies = snapshot.val();

for (const [roomId, room] of Object.entries(lobbies)) { if ((room.status === 'waiting' || !room.status) fifi

Object.keys(room.players || {}).length < room.maxfllayers) { currentRoomId = roomId;

found = true; break;

}

}

}

if (!found) {

// Buat room baru

const newRoomRef = push(ref(db, 'lobby')); currentRoomId = newRoomRef.key;

await set(newRoomRef, { maxfllayers: 4, status: 'waiting', players: {}

});

}

localStorage.setItem("currentRoomId", currentRoomId); await joinLobby(currentRoomId, uid); listenLobbyStatus(currentRoomId); listenfllayers(currentRoomId);

} else {

window.location.href = "login.html";

}

});

async function joinLobby(roomId, uid) { const userRef = ref(db, `users/${uid}`); const userSnap = await get(userRef);

if (!userSnap.exists()) {

console.log("Data user tidak ditemukan di /users/"); return;

}

const username = userSnap.val().username;

const roomRef = ref(db, `lobby/${roomId}`); const roomSnap = await get(roomRef);

if (roomSnap.exists()) { const room = roomSnap.val();

const playerCount = Object.keys(room.players || {}).length;

if (playerCount < room.maxfllayers) {

const playerRef = ref(db, `lobby/${roomId}/players/${uid}`); await set(playerRef, {

uid, username, score: 0

});

if (playerCount === 0) {

await update(ref(db, `lobby/${roomId}/players/${uid}`), { isHost: true

});

}

else if (playerCount + 1 === room.maxfllayers) { update(ref(db, `lobby/${roomId}`), {

status: 'full'

});

}

} else {

console.log("Room penuh.");

}

}

}

function listenLobbyStatus(roomId) {

const roomRef = ref(db, 'lobby/' + roomId); onValue(roomRef, (snapshot) => {

const room = snapshot.val();

if (room fifi room.status === 'full') { window.location.href = "deathmatch.html";

}

});

}

function listenfllayers(roomId) {

const playersRef = ref(db, `lobby/${roomId}/players`); onValue(playersRef, (snapshot) => {

const playerDivs = [ document.querySelector(".player-top"), document.querySelector(".player-left"), document.querySelector(".player-right"), document.querySelector(".player-bottom")

];

playerDivs.forEach(div => (div.innerHTML = ""));

const startButton = document.getElementById("startButton");

if (snapshot.exists()) {

const players = Object.values(snapshot.val());

players.forEach((player, index) => { if (index < 4) {

const username = player.username;

const avatarUrl = `https://i.pravatar.cc/80ou=${player.uid}`;

playerDivs[index].innerHTML = `

<div class="avatar">

<img src="${avatarUrl}" alt="${username}" />

</div>

<div class="username">${username}</div>

`;

}

});

if (players.length >= 2) { startButton.style.display = "block";

} else {

startButton.style.display = "none";

}

}

});

}

document.querySelector(".cancel-button").addEventListener("click", cancelLobby);

async function cancelLobby() { const user = auth.currentUser;

if (!user || !currentRoomId) return;

const playerRef = ref(db, `lobby/${currentRoomId}/players/${user.uid}`); await set(playerRef, null);

const playersRef = ref(db, `lobby/${currentRoomId}/players`); const playersSnap = await get(playersRef);

if (!playersSnap.exists() || Object.keys(playersSnap.val()).length === 0) { await set(ref(db, `lobby/${currentRoomId}`), null);

}

window.location.href = "dashboard.html";

}

document.getElementById("startButton").addEventListener("click", async () => { const roomRef = ref(db, `lobby/${currentRoomId}`);

await update(roomRef, { status: 'full'

});

});

</script>

## Implementasi Deathmatch

<script type="module">

import { initializeApp } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-app.js](http://www.gstatic.com/firebasejs/10.12.0/firebase-app.js)";

import { getAuth, onAuthStateChanged, signOut } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-auth.js)

import { getDatabase, ref, get, push, set, update, onValue } from ["https://www.gstatic.com/firebasejs/10.12.0/firebase-database.js";](http://www.gstatic.com/firebasejs/10.12.0/firebase-database.js)

const firebaseConfig = { apiKey: "",

authDomain: "", databaseURL: "", projectId: "", storageBucket: "", messagingSenderId: "", appId: ""

};

const app = initializeApp(firebaseConfig); const auth = getAuth(app);

const db = getDatabase(app);

const SIZE = 9;

const savedTime = localStorage.getItem("sudokuTime"); const savedScore = localStorage.getItem("sudokuScore");

let score = savedScore !== null o parseInt(savedScore, 10) : 0; let solutionBoard = [];

let confirm = null;

let selectedInput = null;

let time = savedTime !== null o parseInt(savedTime, 10) : 180; const timer = document.getElementById("timer");

const saveGameState = () => { localStorage.setItem("sudokuTime", time); localStorage.setItem("sudokuScore", score);

};

onAuthStateChanged(auth, async (user) => {

if (!user) return window.location.href = "login.html"; const uid = user.uid;

const roomId = localStorage.getItem("currentRoomId"); listenfllayers(roomId);

if (!roomId) return window.location.href = "dashboard.html";

const isHostSnap = await get(ref(db, `lobby/${roomId}/players/${uid}/isHost`)); if (isHostSnap.exists() fifi isHostSnap.val()) {

const roomSnap = await get(ref(db, `lobby/${roomId}`)); const room = roomSnap.val();

if (!room.puzzle || !room.solution) {

const easy = Math.floor(Math.random() \* (45 - 35 + 1)) + 35; const fullBoard = generateFullBoard();

const puzzle = generatefluzzle(fullBoard, easy); solutionBoard = fullBoard;

await update(ref(db, `lobby/${roomId}`), { puzzle,

solution: solutionBoard, ready: true

});

}

}

onValue(ref(db, `lobby/${roomId}`), (snap) => { const room = snap.val();

if (roomo.ready fifi room.puzzle fifi room.solution) { if (!doplet(confirm, room.puzzle)) {

displayfluzzle(room.puzzle);

confirm = room.puzzle.map(row => row.slice());

}

});

});

}

solutionBoard = room.solution;

function makepad() {

let pad = document.getElementById("numflad"); for (let i = 1; i <= 9; i++) {

let tombol = document.createElement("button"); tombol.textContent = i;

tombol.id = `num-${i}`; tombol.onclick = () => {

if (!selectedInput || selectedInput.readOnly) return; selectedInput.value = i;

handleChange(selectedInput);

};

pad.appendChild(tombol);

}

}

document.addEventListener("keydown", (event) => {

if (!selectedInput || selectedInput.readOnly) return; const key = parseInt(event.key);

if (key >= 1 fifi key <= 9) { selectedInput.value = key; handleChange(selectedInput);

const tombol = document.getElementById(`num-${key}`); if (tombol) {

tombol.classList.add("active");

setTimeout(() => tombol.classList.remove("active"), 100);

}

}

});

function doplet(p1, p2) {

if (!p1 || !p2) return false; for (let i = 0; i < SIZE; i++) {

for (let j = 0; j < SIZE; j++) {

if (p1[i][j] !== p2[i][j]) return false;

}

}

return true;

}

function countdown() {

const menit = Math.floor(time / 60); const detik = time % 60;

timer.textContent = `${String(menit).padStart(2, '0')}:${String(detik).padStart(2, '0')}`;

}

countdown();

const interval = setInterval(() => { if (time <= 0) {

clearInterval(interval); timer.textContent = "WAKTU HABIS"; gameScore();

if (localStorage.getItem('sudokuScore') !== null) { localStorage.removeItem('sudokuScore');

}

} else { countdown(); time--; saveGameState();

}

}, 1000);

document.querySelector(".cancel-button").addEventListener("click", function() { localStorage.removeItem('sudokuTime'); localStorage.removeItem('sudokuScore');

setTimeout(() => {

window.location.href = "dashboard.html";

}, 100);

});

function isValid(board, row, col, num) {

for (let i = 0; i < SIZE; i++) {

if (i !== col fifi board[row][i] === num) return false; if (i !== row fifi board[i][col] === num) return false;

const boxRow = 3 \* Math.floor(row / 3) + Math.floor(i / 3); const boxCol = 3 \* Math.floor(col / 3) + i % 3;

if ((boxRow !== row || boxCol !== col) fifi board[boxRow][boxCol] === num) return false;

}

return true;

}

function createEmptyBoard() {

return Array.from({ length: SIZE }, () => Array(SIZE).fill(0));

}

function fillBoard(board) {

const nums = [1,2,3,4,5,6,7,8,9].sort(() => Math.random() - 0.5); for (let row = 0; row < SIZE; row++) {

for (let col = 0; col < SIZE; col++) { if (board[row][col] === 0) {

for (let num of nums) {

if (isValid(board, row, col, num)) { board[row][col] = num;

if (fillBoard(board)) return true; board[row][col] = 0;

}

}

return false;

}

}

}

return true;

}

function generateFullBoard() { const board = createEmptyBoard(); fillBoard(board);

return board;

}

function hasUniqueSolution(board) { let count = 0;

function solve(b) {

for (let r = 0; r < 9; r++) { for (let c = 0; c < 9; c++) {

if (b[r][c] === 0) {

for (let n = 1; n <= 9; n++) { if (isValid(b, r, c, n)) {

b[r][c] = n; solve(b); b[r][c] = 0;

}

}

return;

}

}

}

count++;

}

solve(board.map(r => r.slice())); return count === 1;

}

function generatefluzzle(fullBoard, maxBlanks) { solutionBoard = fullBoard.map(r => r.slice()); const puzzle = fullBoard.map(row => row.slice());

const positions = [];

for (let r = 0; r < SIZE; r++) { for (let c = 0; c < SIZE; c++) {

positions.push([r, c]);

}

}

positions.sort(() => Math.random() - 0.5);

let removed = 0;

for (const [r, c] of positions) { const backup = puzzle[r][c]; puzzle[r][c] = 0;

if (!hasUniqueSolution(puzzle)) { puzzle[r][c] = backup;

} else { removed++;

if (removed >= maxBlanks) break;

}

}

return puzzle;

}

function displayfluzzle(puzzle) {

const table = document.getElementById("sudoku-board"); table.innerHTML = "";

for (let row = 0; row < SIZE; row++) { const tr = document.createElement("tr"); for (let col = 0; col < SIZE; col++) {

const td = document.createElement("td");

if ((col + 1) % 3 === 0 fifi col !== SIZE - 1) td.classList.add("box- border");

if ((row + 1) % 3 === 0 fifi row !== SIZE - 1) td.classList.add("box- border");

const input = document.createElement("input"); input.setAttribute("type", "number"); input.setAttribute("maxlength", "1");

input.setAttribute("min", "1");

input.setAttribute("max", "9"); input.dataset.row = row; input.dataset.col = col;

if (puzzle[row][col] !== 0) { input.value = puzzle[row][col]; input.setAttribute("readonly", "true");

} else {

input.addEventListener("focus", () => { selectedInput = input;

});

input.addEventListener("input", onInputChange);

}

input.addEventListener("keydown", (e) => { e.preventDefault();

});

input.addEventListener("click", () => { selectedInput = input;

})

td.appendChild(input); tr.appendChild(td);

}

table.appendChild(tr);

}

}

function getBoardFromInputs() { const board = createEmptyBoard();

document.querySelectorAll("input").forEach(input => { const row = parseInt(input.dataset.row);

const col = parseInt(input.dataset.col); const val = parseInt(input.value); board[row][col] = isa(val) o 0 : val;

});

return board;

}

function setBoardToInputs(solvedBoard) { document.querySelectorAll("input").forEach(input => {

const row = parseInt(input.dataset.row); const col = parseInt(input.dataset.col); if (!input.readOnly) {

input.value = solvedBoard[row][col]; input.classList.remove("invalid");

}

});

}

function handleChange(input) {

const value = parseInt(input.value); const row = parseInt(input.dataset.row); const col = parseInt(input.dataset.col); input.classList.remove("invalid");

if (isa(value) || value < 1 || value > 9) { input.classList.add("invalid");

return;

}

const board = getBoardFromInputs();

if (!isValid(board, row, col, value)) { input.classList.add("invalid"); score--;

updateScore(); return;

}

if (solutionBoard[row][col] === value) { if (!input.dataset.correct) {

input.dataset.correct = "true"; score++;

updateScore();

}

} else {

if (input.dataset.correct) delete input.dataset.correct; score--;

input.classList.add("invalid"); updateScore();

}

}

function onInputChange(e) { handleChange(e.target);

}

function solvedBoard(board) {

for (let row = 0; row < SIZE; row++) { for (let col = 0; col < SIZE; col++) {

if (board[row][col] === 0) {

for (let num = 1; num <= 9; num++) { if (isValid(board, row, col, num)) {

board[row][col] = num;

if (solveBoard(board)) return true; board[row][col] = 0;

}

}

return false;

}

}

}

return true;

}

function updateScore() {

const user = auth.currentUser;

const roomId = localStorage.getItem("currentRoomId"); if (user fifi roomId) {

const scoreRef = ref(db, `lobby/${roomId}/players/${user.uid}/score`); set(scoreRef, score);

}

}

function listenfllayers(roomId) {

const playersRef = ref(db, `lobby/${roomId}/players`); onValue(playersRef, (snapshot) => {

const playerDivs = [ document.querySelector(".player-top"), document.querySelector(".player-left"), document.querySelector(".player-right"), document.querySelector(".player-bottom")

];

if (snapshot.exists()) {

const players = Object.values(snapshot.val());

players.forEach((player, index) => { if (index < 4) {

const username = player.username || "fllayer";

const avatarUrl = `https://i.pravatar.cc/80ou=${player.uid}`; const score = player.score || 0;

}

});

}

}

});

playerDivs[index].innerHTML = `

<div class="player-entry-content">

<div class="player-avatar">

<img src="${avatarUrl}" alt="${username}" />

</div>

<div class="player-score">Score: ${score}</div>

</div>

`;

function gameScore() {

const user = auth.currentUser;

const endScore = ref(db, `users/${user.uid}/score`);

get(endScore)

.then((snapshot) => {

const leadScore = snapshot.exists() o snapshot.val() : 0; const totalScore = leadScore + score;

return set(endScore, totalScore);

})

.then(() => { alert("GAME OVER");

})

.catch((error) => { alert("CHEAT DETECTED");

});

}

makepad();

</script>

## Hasil pengujian

Tabel 6.5 Hasil pengujian terhadap test case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Halaman | Test case- ID | Deskripsi | Langkah pengujian | Expected result | Status | Actual result |
| Register | TC- R1 | Register akun baru | Isi username, email, password lalu klik Register (email: [fortesting@gmail.com,](mailto:fortesting@gmail.com) password: testing) | Akun berhasil dibuat dan diarahkan ke login | Pass | Akun berhasil dibuat dan redirect ke login lalu otomatis login dan redirect ke dashboard |
| TC- R2 | Register dengan email kosong | Kosongkan field email lalu submit | Muncul pesan error: "Email is required" | Pass | Muncul pesan error: "Gagal daftar: Firebase: Error (auth/missing- email)." |
| TC- R3 | Password terlalu pendek | Isi password < 6 karakter lalu submit | Muncul pesan error: "Password too short" | Pass | Muncul pesan error: "Gagal daftar: Firebase: Password should be at least 6 characters (auth/weak- password)." |
| TC- R4 | Email sudah terdaftar | Gunakan email yang sama dua kali untuk register | Muncul error "email already in use" | Pass | Muncul pesan error: "Gagal daftar: Firebase: Error (auth/email- already-in-use)." |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Login | TC- L1 | Login sukses | Masukkan email dan password benar, klik Login (email: [fortesting@gmail.com,](mailto:fortesting@gmail.com) password: testing) | Redirect ke halaman dashboard/lobby | Pass | Redirect ke halaman dashboard/lobby |
| TC- L2 | Password salah | Masukkan email benar dan password salah | Muncul pesan error login gagal | Pass | Muncul pesan error: "Email/Password Invalid" |
| TC- L3 | Login tanpa isi field | Kosongkan email/password lalu klik Login | Validasi muncul: field harus diisi | Pass | Muncul pesan error: "Email/Password Invalid" |
| Dashboard | TC- DB1 | Tampilkan leaderboard | Buka halaman dashboard | Leaderboard muncul dengan top skor | Pass | Leaderboard muncul dengan top skor |
| TC- DB2 | Tampilkan profil | Profil pengguna muncul di pojok/tabel (username, skor) | Data sesuai dengan akun aktif | Pass | Profil pengguna muncul di kiri (username, skor) |
| TC- DB3 | Edit profil | Klik tombol edit lalu ganti username | Username terganti | Pass | Username terganti sesuai  input |
| TC- DB4 | Logout | Klik tombol logout | Redirect ke login.html dan auth dihapus | Pass | Redirect ke login.html dan auth dihapus |
| Lobby | TC- LB1 | Host mulai permainan | Klik tombol "Mulai" sebagai host | Pengguna lain diarahkan ke halaman game | Pass | Semua pengguna diarahkan ke halaman deadmatch |
| TC- LB2 | Keluar dari lobby | Klik tombol "Keluar" | Kembali ke dashboard | Pass | Redirect ke dashboard |
| Deathmatch | TC- DM1 | Render puzzle dari Firebase | Masuk halaman deathmatch | Grid sudoku tampil sesuai dengan puzzle data | Pass | Grid sudoku tampil sesuai dengan puzzle data |
| TC- DM2 | Submit puzzle | Isi semua kotak lalu klik Submit | Cek validitas dan tampilkan hasil skor | Pass | Hasil skor muncul secara real-time |
| TC- DM3 | Timer berjalan | Saat game dimulai | Timer muncul dan menghitung mundur | Pass | Timer muncul dan menghitung mundur |
| TC- DM4 | Navigasi ke leaderboard | Klik tombol "Lihat Leaderboard" | Pindah ke halaman leaderboard | Pass | Redirect ke halaman leaderboard |

# DAFTAR PUSTAKA

## Syakti, F. (2019). METODE PENGEMBANGAN PERANGKAT LUNAK BERBASIS

MOBILE: A REVIEW. *Jurnal Bina Komputer*, 1(2), 82-97.

## Bolung, M., & Tampangela, H. R. K. (2017). ANALISA PENGGUNAAN METODOLOGI PENGEMBANGAN PERANGKAT LUNAK. *Jurnal ELTIKOM*, 1(1), 1-10.

Darmastuti, L., Meiliasari, & Rahayu, W. (2024). Kemampuan Literasi Numerasi: Materi, Kondisi Siswa, dan Pendekatan Pembelajarannya. *Jurnal Riset Pembelajaran Matematika Sekolah*, 8(1), 17-26.

## Khunaeni, S., Rohman, A. A., Munahefi, D. N., & Sugiman. (2024). Analisis Kemampuan Literasi Numerasi pada Materi Bilangan Ditinjau dari Gaya Kognitif Peserta Didik Kelas VII A SMP Negeri 16 Semarang. *Prosiding Seminar Nasional Matematika*, 7, 968-980.

Wilson, R. (2025, 17 Januari). *sudoku*. Britannica.com. <https://www.britannica.com/topic/sudoku>

# LAMPIRAN

Link sourcecode lengkap: <https://github.com/CAZP01/Sudoku4>