

CHESS APP

A Real-Time Multiplayer Chess Game with Move Evaluation

Ajay Soni Verma (2022btcse006) | Software Engineering Lab| 14-May-2025

Table of Contents

[Project Overview 2](#_Toc198077652)

[Initial Proposal 3](#_Toc198077653)

[**a.** *Purpose* 3](#_Toc198077654)

[**b.** *Users* 3](#_Toc198077655)

[**c.** *Technology to be Used* 3](#_Toc198077656)

[**d.** *Estimated Lines of Code* 3](#_Toc198077657)

[**e.** *Five Functionalities* 3](#_Toc198077658)

[**f.** *Software Delivery Method* 4](#_Toc198077659)

[Software Requirement Specification (SRS) 5](#_Toc198077660)

[**1.** *Introduction* 5](#_Toc198077661)

[**2.** *Functional Requirements* 6](#_Toc198077662)

[**3.** *Non-Functional Requirements* 6](#_Toc198077663)

[**4.** *Constraints and Limitations* 7](#_Toc198077664)

[**5.** *Acceptance Criteria* 8](#_Toc198077665)

[**6.** *Timeline and Roadmap* 8](#_Toc198077666)

[**7.** *References* 8](#_Toc198077667)

[**8.** *Miscellaneous* 9](#_Toc198077668)

[**9.** *Document Information* 9](#_Toc198077669)

[Use Case Diagrams 10](#_Toc198077670)

[User Stories 12](#_Toc198077671)

[Activity and Data Flow Diagrams 14](#_Toc198077672)

[Release 1 Notes 20](#_Toc198077673)

[Test Cases 26](#_Toc198077674)

[Release 2 Notes 32](#_Toc198077675)

[Future Work 33](#_Toc198077676)

[References 34](#_Toc198077677)

# Project Overview

1. The project is a **mobile chess application** built using **Kotlin** for the frontend and the **Ktor framework** for backend services.
2. It offers three modes of play: **Over-the-Board (local two-player)**, **Online Multiplayer**, and **Single-player against a Computer AI**.
3. Users must **log in or register** to access the app, ensuring a personalized and secure experience.
4. The app supports **online multiplayer gameplay with a 10-minute timer** for each player, encouraging competitive real-time matches.
5. A **game analysis feature** allows players to request detailed move evaluations by uploading a PGN file to the backend server.
6. The backend, powered by **Ktor**, handles user authentication, multiplayer game coordination, and PGN-based game analysis.
7. A **community section** is available where users can **post and view content**, fostering interaction among chess enthusiasts.
8. The app **does not store game history** or past match records, keeping the implementation lightweight.
9. The computer opponent provides a basic **AI-based challenge**, allowing users to practice offline.
10. This application is developed **exclusively for Android mobile devices**, offering a responsive and interactive user experience.

# Initial Proposal

## **a.** Purpose

The purpose of this app is to provide an engaging and competitive platform for playing chess. It supports various gameplay modes, including over-the-board, online multiplayer, and against a computer AI. Additionally, it enhances player experience through PGN-based game analysis via a backend server. The app also fosters community interaction through a built-in social feature.

## **b.** Users

* **Primary Users**: Chess players of all levels who want to play casually or competitively, both offline and online.
* **Secondary Users**: Developers and testers exploring multiplayer integration, authentication, and client-server interaction in mobile apps.

## **c.** Technology to be Used

* **Programming Language**: Kotlin
* **Frontend Framework**: Jetpack Compose (Android)
* **Backend Framework**: Ktor (Kotlin server-side)
* **Database**: **PostgreSQL** (used for storing user credentials and authentication data)
* **Development Tools**: Android Studio IDE

## **d.** Estimated Lines of Code

Approximately **3,500–4,500 lines of Kotlin code**, covering UI, game logic, multiplayer integration, and server communication.

## **e.** Five Functionalities

1. **Login and Registration System**
   * *Input*: Email and password
   * *Output*: Authenticated access to app features
2. **Over-the-Board (OTB) Chess Gameplay**
   * *Input*: Two players take turns making moves on the same device
   * *Output*: Validated moves and board updates
3. **Online Multiplayer with Timer**
   * *Input*: Real-time moves over WebSocket connection
   * *Output*: Synchronized gameplay with 10-minute timers per player
4. **Play vs Computer (AI)**
   * *Input*: User plays against computer
   * *Output*: Board updates based on computer responses
5. **PGN-Based Game Analysis**
   * *Input*: Player uploads game PGN
   * *Output*: Server returns move evaluations and feedback

## **f.** Software Delivery Method

* **Mobile Application (Android)**
* Backend hosted using **Ktor** for authentication, multiplayer, and analysis

# Software Requirement Specification (SRS)

**Project Title:** Chess App  
**Written By:** Ajay Soni Verma  
**Version:** 1.0  
**Date:** 5-February-2025  
**Signed By:** *(Instructor/Advisor) Dr. Gourav Somani*

## **1.** Introduction

**1.1 Project Goals**

The goal of this project is to develop a mobile chess application with multiple gameplay modes, including offline (over-the-board and vs computer), online multiplayer, and game analysis using PGN input. The application also offers a simple community feature for public posts and emphasizes real-time interaction and learning through server-based move evaluation.

**1.2 Stakeholders and Users**

**Stakeholders:**

* Developers
* Testers
* End-users (Chess players and learners)
* Instructor/Advisor

**Users:**

* Casual Chess Players
* Competitive Chess Players
* Chess Enthusiasts and Learners

**1.3 Definitions and Acronyms**

* **PGN** – Portable Game Notation (used for sending game data to the server for analysis)
* **GUI** – Graphical User Interface
* **Ktor** – Kotlin server-side framework
* **Jetpack Compose** – Modern UI toolkit for Android
* **OTB** – Over-the-Board chess (local two-player mode)

## **2.** Functional Requirements

**2.1 Features and Expected Behavior**

* User login and registration using email and password
* Over-the-board chess gameplay on a single device
* Play against a computer
* Online multiplayer chess with 10-minute timers
* PGN-based game analysis via backend server
* Access to online features (analysis, multiplayer) requires login

**2.2 Inputs and Outputs**

**Inputs:**

* User credentials (email, password)
* User moves (via touch-based UI)
* PGN text for analysis

**Outputs:**

* Validated chess moves and updated board states
* Move feedback from server (in analysis mode)

**2.3 User Interaction**

* Touch-based piece movement on a responsive chessboard
* Login screen for user authentication
* Analysis screen showing annotated move feedback

## **3.** Non-Functional Requirements

**3.1 Performance**

**3.1.1 Response Time**

* Offline gameplay response: <100ms
* Computer move response: <1 second
* Server analysis: <2 minutes on average

**3.1.2 Throughput**

* Support up to 100 concurrent online multiplayer games in testing phase

**3.2 Security Considerations**

* Passwords securely stored in PostgreSQL using hashing
* Authentication required for all online features
* Community feature available only to logged-in users

**3.3 Usability and Accessibility**

* Simple and mobile-friendly UI using Jetpack Compose
* Easy for users of all ages and skill levels
* Basic accessibility supported (high contrast UI)

**3.4 Reliability and Availability**

* Offline features fully functional without internet
* Multiplayer backend designed to recover gracefully from brief disconnects
* Target uptime: 99% during usage

**3.5 Format**

* Game data uses PGN format
* Application deployed as Android APK
* Backend server runs independently using Ktor framework

## **4.** Constraints and Limitations

**4.1 Time**

* Project to be completed in approximately 6 weeks

**4.2 Budget**

* No external funding or paid services used
* Fully developed using open-source tools (Android Studio, Kotlin, Ktor, PostgreSQL)

## **5.** Acceptance Criteria

**5.1 Verification and Validation**

* User login/registration functional and secure
* Local and computer game modes work with correct rules and move validation
* Multiplayer mode supports real-time play with timer
* PGN analysis returns valid server responses

## **6.** Timeline and Roadmap

* **Week 1:** Project setup, over-the-board gameplay
* **Week 2:** AI opponent integration
* **Week 3:** User authentication and login system
* **Week 4:** Online multiplayer implementation with timer
* **Week 5:** Game analysis (PGN-based) and community features
* **Week 6:** final bug fixing and testing

## **7.** References

* [https://lichess.org](https://lichess.org/)
* [https://ktor.io](https://ktor.io/)
* <https://developer.android.com/jetpack/compose>
* FIDE official chess rules
* PGN standard documentation

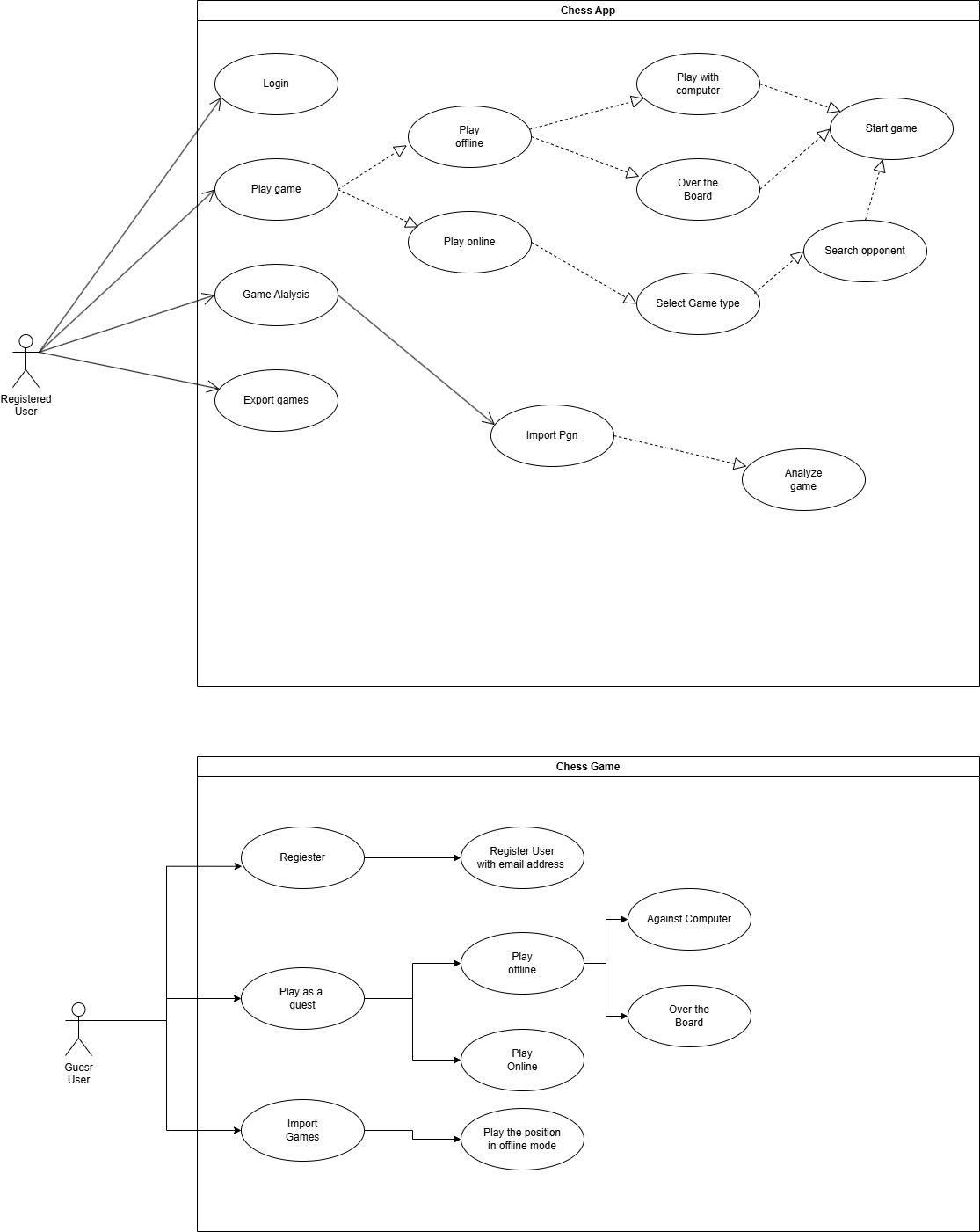
## **8.** Miscellaneous

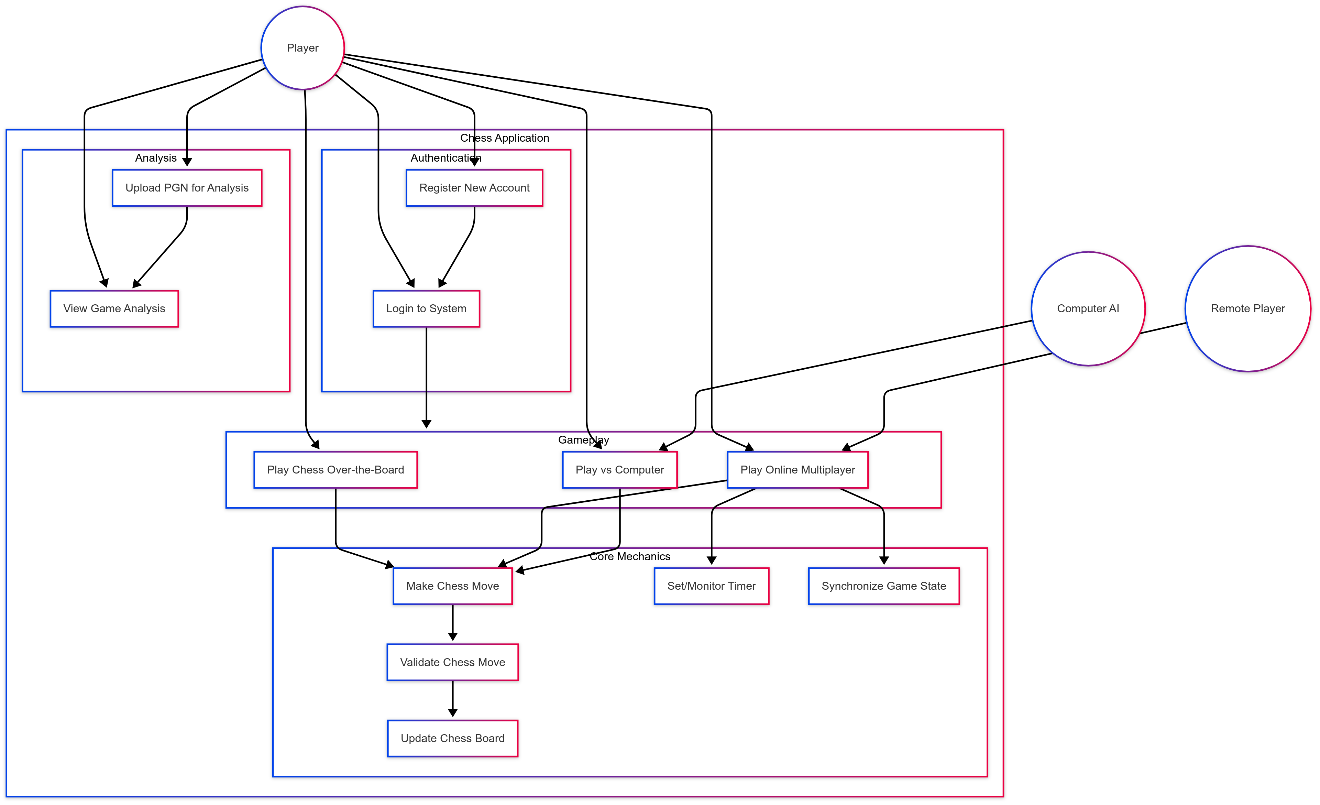
* No notifications or background services implemented
* Additional features may be considered based on user feedback and feasibility.

## **9.** Document Information

| **Written By** | **Version** | **Date** | **Signed By** |
| --- | --- | --- | --- |
| Ajay Soni Verma | 1.0 | 5-February-2025 | *(Instructor/Advisor)* |

# Use Case Diagrams





# User Stories

**🔐 Login and Registration System**

**User Registration**  
A new user opens the app and is shown the registration form. The form asks for an email and a password. If the email format is invalid or the password is too weak, an appropriate error message is shown. If the email is already registered, the user is prompted to log in instead. Once the credentials are valid and unique, a verification otp is sent via email. The user is informed that they must verify their otp before accessing app features.

**User Login**  
An existing user can log in by entering their email and password. If the credentials match the database, the user is granted access to all available features. If login fails, the user is shown an error and allowed to retry.

**♟️ Over-the-Board (OTB) Chess Gameplay**

Two players on the same device can take turns playing chess. The app alternates the board view depending on the player turn, flipping the orientation accordingly. The system validates each move according to chess rules and updates the board visually. Illegal moves are blocked, and players are notified when their move is invalid. The game continues until a checkmate, draw, or resignation occurs.

**🌐 Online Multiplayer with Timer**

A user selects the online multiplayer mode. They are matched with another online player through a WebSocket connection. Once matched, a 10-minute timer starts for each player. Players take turns making moves in real time, and each move resets the opponent’s timer. The board is updated simultaneously on both players' devices to ensure synchronization. If a player disconnects or runs out of time, the system ends the game and notifies both players.

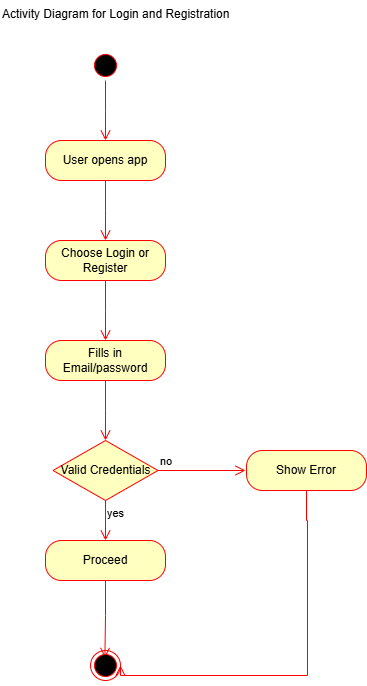
**🤖 Play vs Computer (AI)**

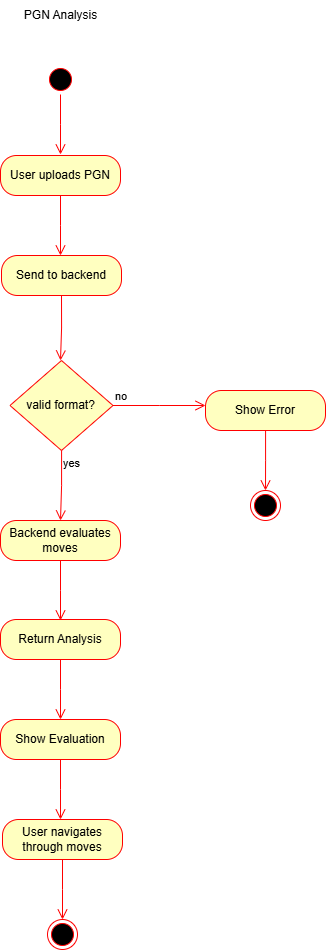
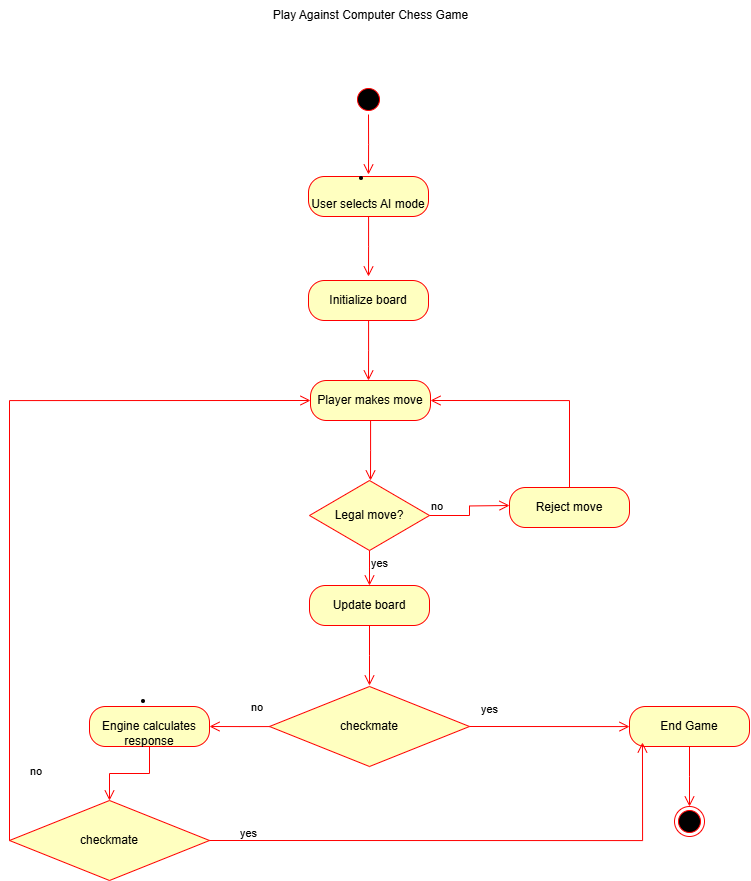
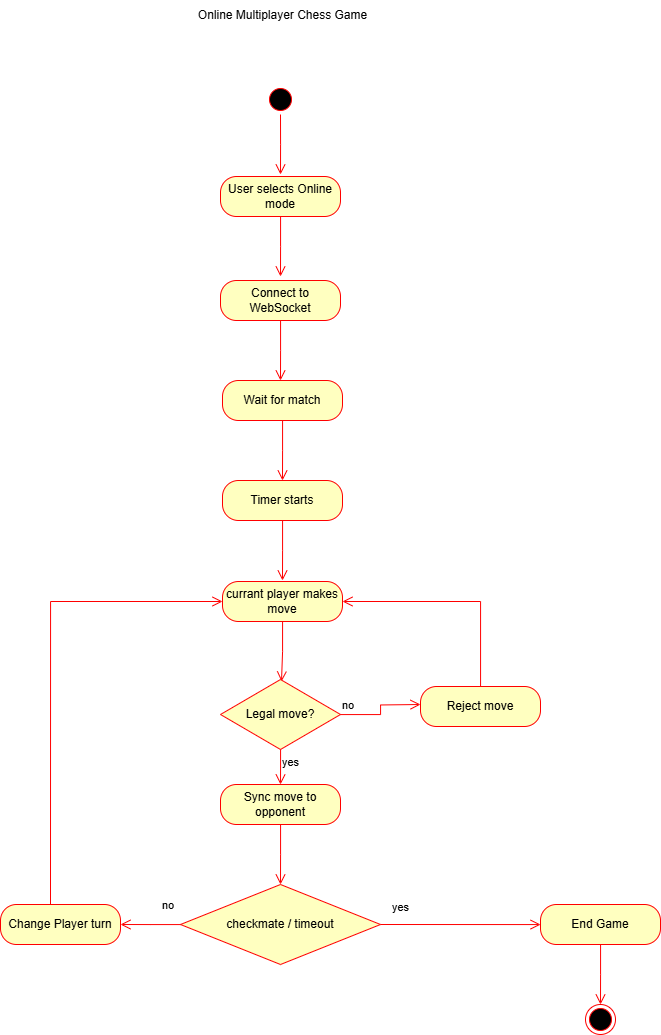
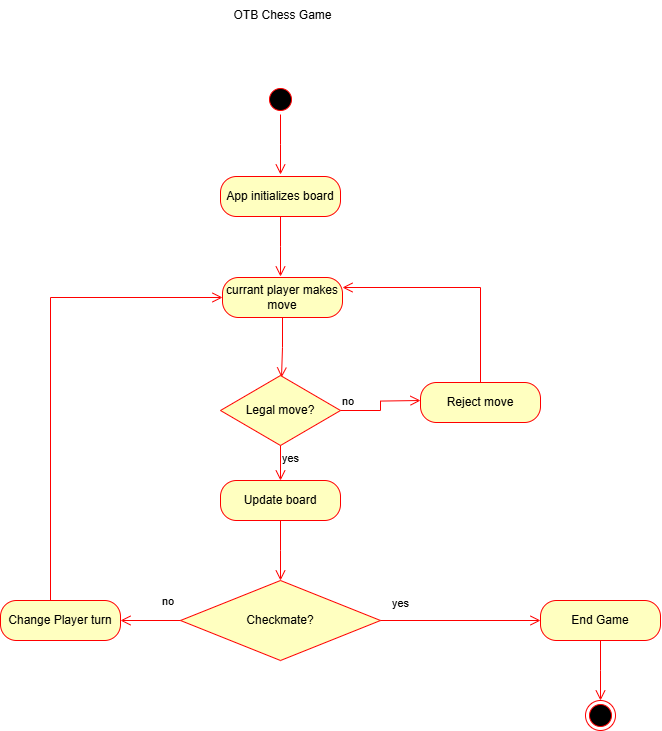
A user selects the option to play against the computer. The app starts a new game with the computer as the opponent. The user plays first (default white), and after each move, the system calculates and displays the AI’s move based on predefined difficulty settings. The user can undo a move or resign at any time. The game continues until a win, draw, or resignation is registered.

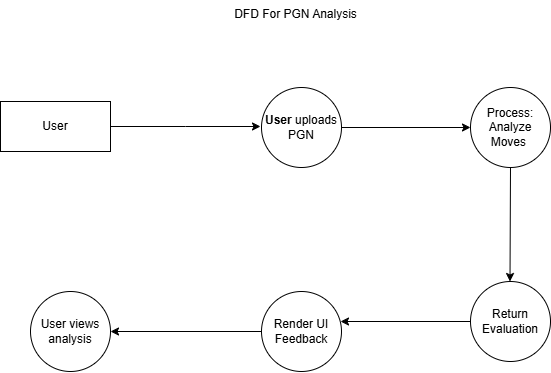
**📊 PGN-Based Game Analysis**

A user uploads a game in PGN (Portable Game Notation) format through a file selector. Once uploaded, the system sends the file to a backend analysis engine. The server processes the PGN and returns detailed evaluation data for each move, including best alternatives, score evaluations, and categorized feedback (e.g., blunder, inaccuracy). The user can scroll through the moves and view suggestions and insights to understand their strengths and mistakes.

# Activity and Data Flow Diagrams



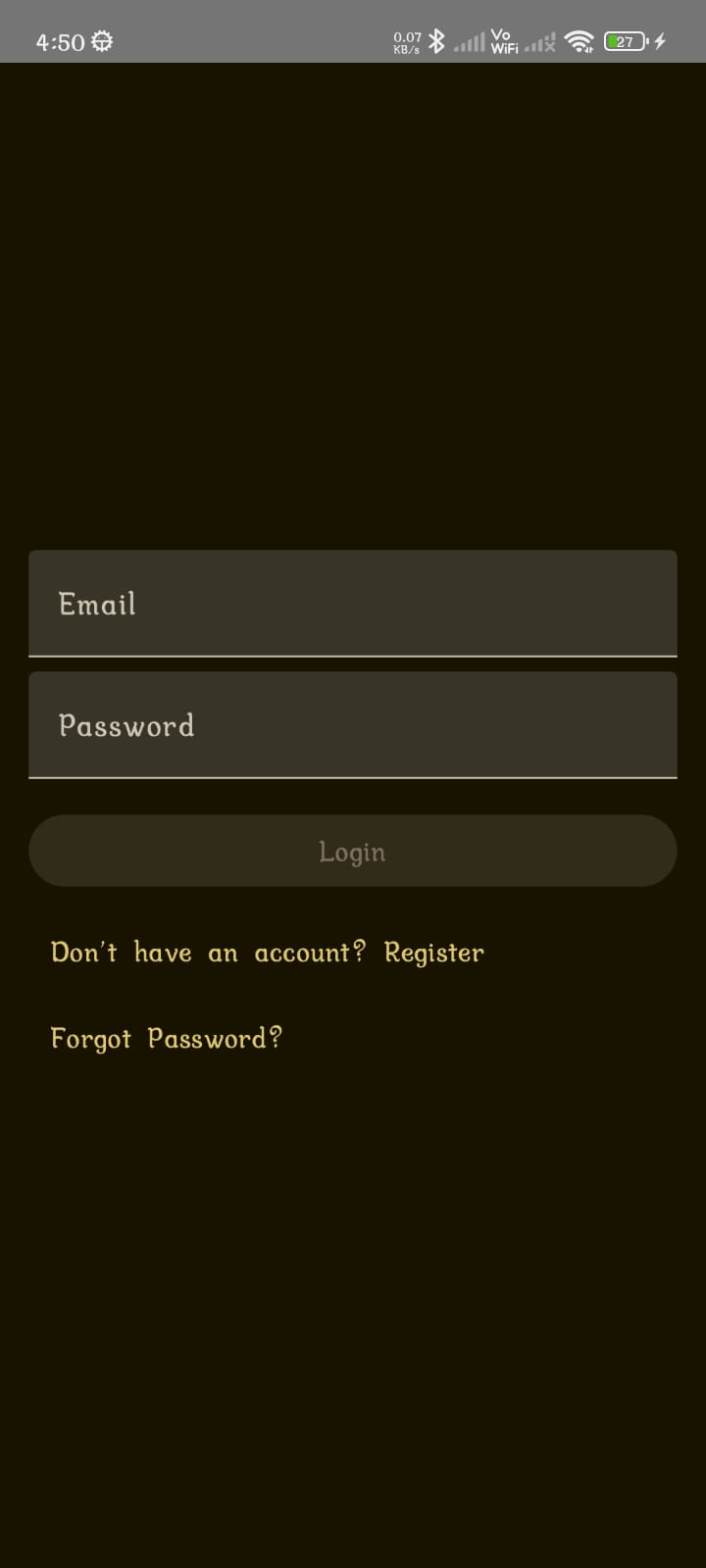
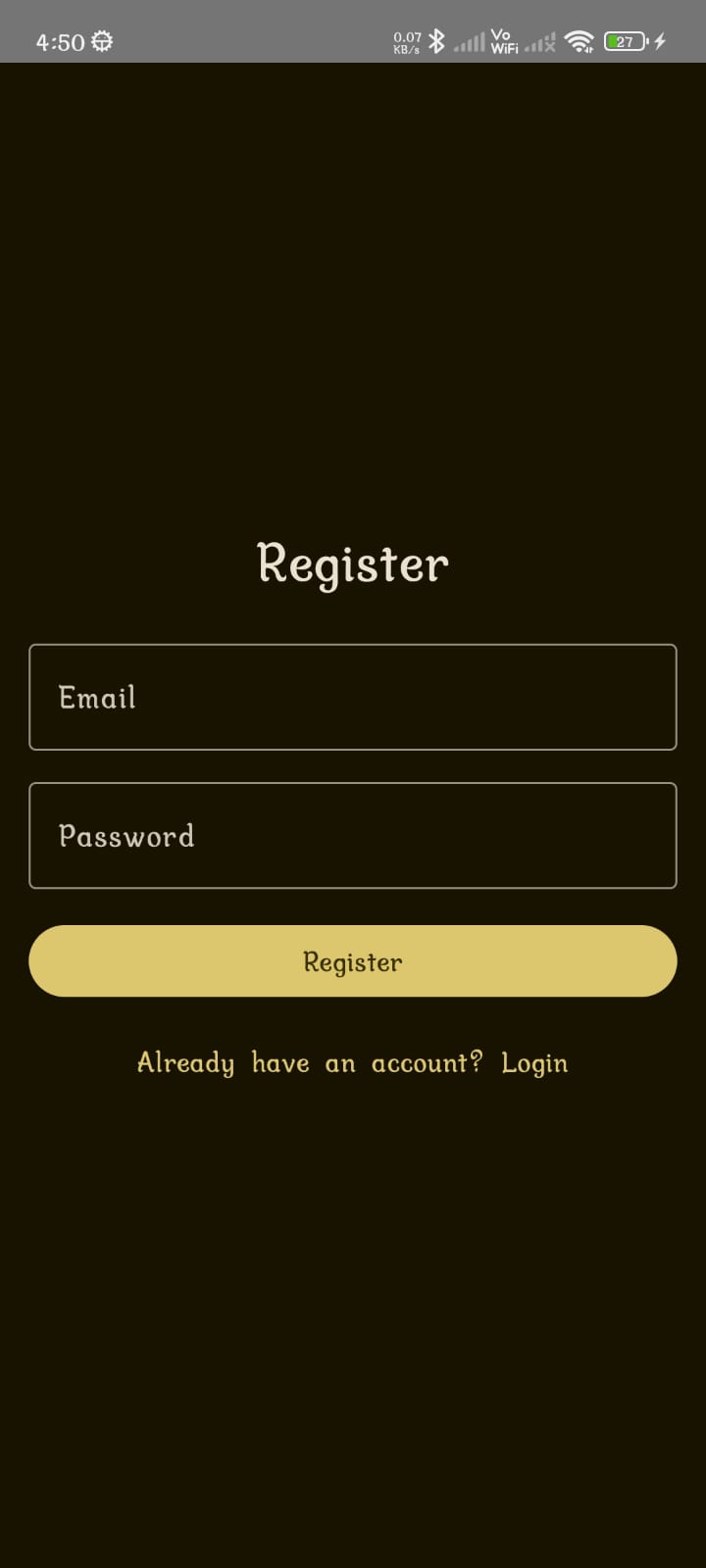




# Release 1 Notes

Working Functionalities -

Login - Sign Up -

Main Menu –

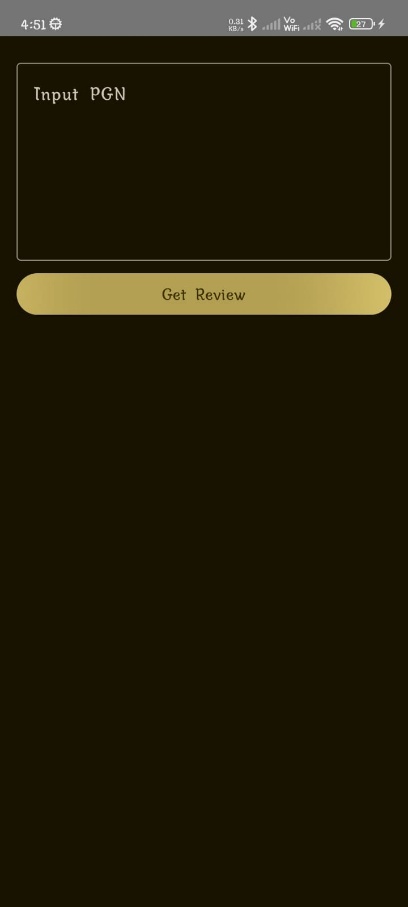


Gameplay:

Over the Board / Against Computer - Online Multiplayer -

Game Review –

**Q1. What has been done?**

The current release of the chess app includes the implementation of the core and advanced features essential for a complete chess experience:

* **User Authentication:** Login and registration functionality is fully integrated, allowing users to securely access their accounts.
* **Over-the-Board Play:** Players can play chess locally on the same device.
* **Play Against Computer:** Users can challenge an AI opponent with various difficulty levels.
* **Online Multiplayer:** Real-time online matches are supported via WebSocket-based communication.
* **Game Review and Analysis:** Users can review their completed games with move-by-move analysis using Stockfish.
* **Export Game:** Players can export their games in a standard format for external use or sharing.

**Q2. Target for the Next Incremental Delivery**

The focus for the upcoming release is to enhance usability, increase community engagement, and improve performance. Key targets include:

* **Feature Optimization:** Streamline and polish existing features based on user feedback and internal testing.
* **Bug Fixes:** Address known issues and edge cases affecting gameplay and connectivity.
* **Community Feature:** Introduce a community component to foster interaction among users.
* **Performance Improvements:** Optimize app speed, reduce latency in multiplayer, and ensure efficient background processing.

**Q3. Difficulties Faced**

During development, several technical challenges were encountered:

* **WebSocket Integration in Multiplayer:** Ensuring reliable and synchronized real-time communication between players proved complex due to connection handling, latency issues, and state consistency.
* **Stockfish Integration for Game Review:** Embedding and interacting with the Stockfish engine required significant effort, especially around parsing and evaluating moves efficiently.
* **Conversion Between PGN, UCI, and FEN:** Translating move data between these notation formats involved dealing with nuanced rules, inconsistencies, and lossless data conversion to maintain accuracy during analysis.

GitHub Repo Link – https://github.com/AjayGovindSoni/ChessApp.git

# Test Cases

**Test Case Id:**

Test\_Signup\_1

**Test Scenario:**

Verify signing up

**Test Case:**

Enter invalid email address

**Pre-Condition:**

Nan

**Test Steps:**

1.Enter a string that does not resemble an email address

2. Click "Register"

**Test Data:**

A string that does not represent an email address

**Expected Result:**

A message that "given email address is not an email" and prevent the user from signing up

**Post Condition:**

Nan

**Actual Result:**

An alert comes up with the message "Registration failed" and user is not able to sign up

**Status:**

Pass

**Test Case Id:**

Test\_Signup\_2

**Test Scenario:**

Verify signing up

**Test Case:**

Enter an email that already has an account

**Pre-Condition:**

An email address that already exists

**Test Steps:**

1. Enter an email address already registered

**Test Data:**

An email address which is already registered

**Expected Result:**

Prevent the user from signing up and alert that email already in use

**Post Condition:**

Nan

**Actual Result:**

User is not allowed to sign up and is informed that "Email already in use"

**Status:**

Pass

**Test Case Id:**

Test\_Signup\_3

**Test Scenario:**

Verify signing up

**Test Case:**

Enter a valid email that is not in use but for otp verification enter incorrect otp

**Pre-Condition:**

A valid email address

**Test Steps:**

1. Enter an email address which is not registered.

2. After reading the OTP sent on mail, type in an incorrect otp in the next page

**Test Data:**

An email address which is not registered

**Expected Result:**

Alert the user that OTP is incorrect and prevent sign up

**Post Condition:**

Nan

**Actual Result:**

Form submission prevented and the user was informed that "OTP entered was not correct"

**Status:**

Pass

**Test Case Id:**

Test\_Signup\_4

**Test Scenario:**

Verify signing up

**Test Case:**

Enter a valid email that is not in use and for otp verification enter correct otp

**Pre-Condition:**

A valid email address

**Test Steps:**

1. Enter an email address which is not registered.

2. After reading the OTP sent on mail, type in correct otp in the next page

**Test Data:**

An email address which is not registered

**Expected Result:**

Allow the user to sign up

**Post Condition:**

Check password policy

**Actual Result:**

User allowed

**Status:**

Pass

**Test Case Id:**

Test\_Signup\_5

**Test Scenario:**

Verify signing up

**Test Case:**

Enter a valid email that is not in use, correct OTP and choose a password that fits the password policy (6 chars, 1 special, 1 Uppercase, 1 lowercase, 1 digit)

**Pre-Condition:**

A valid email address, Correct OTP, Suitable password

**Test Steps:**

1. Enter an email address which is not registered.

2. After reading the OTP sent on mail, type in correct otp in the next page.

3. Enter an appropriate password

**Test Data:**

An email address which is not registered

**Expected Result:**

Complete the process of signing up

**Post Condition:**

Redirect user to login page

**Actual Result:**

Sign up successful, user redirected to login page

**Status:**

Pass

**Test Case Id:**

Test\_Signup\_6

**Test Scenario:**

Verify signing up

**Test Case:**

Enter a valid email that is not in use, correct OTP and choose a password that does not fit the password policy

**Pre-Condition:**

A valid email address, Correct OTP

**Test Steps:**

1. Enter an email address which is not registered.

2. After reading the OTP sent on mail, type in correct otp in the next page.

3. Enter a password that does not correspond to password policy

**Test Data:**

An email address which is not registered

**Expected Result:**

User is not able to sign up until an appropriate password is chosen

**Post Condition:**

Nan

**Actual Result:**

Sign up prevented

**Status:**

Pass

**Test Case Id:**

Test\_Login\_1

**Test Scenario:**

Verify Login

**Test Case:**

Enter an email which is not registered

**Pre-Condition:**

Nan

**Test Steps:**

1. Enter a random email address or a mail which is not registered

**Test Data:**

An email address which is not registered

**Expected Result:**

Prevent login, and inform the user that the mail is not in use

**Post Condition:**

Nan

**Actual Result:**

Login prevented, user informed

**Status:**

Pass

**Test Case Id:**

Test\_Login\_2

**Test Scenario:**

Verify Login

**Test Case:**

Enter a registered email and incorrect password

**Pre-Condition:**

Registered email

**Test Steps:**

1. Enter registered email

2. Enter incorrect password

**Test Data:**

Registered email

**Expected Result:**

Prevent login and inform user that the password is incorrect

**Post Condition:**

Nan

**Actual Result:**

Login prevented, user informed

**Status:**

Pass

**Test Case Id:**

Test\_Login\_3

**Test Scenario:**

Verify Login

**Test Case:**

Enter registered email and correct password

**Pre-Condition:**

Registered email

**Test Steps:**

1. Enter registered email

2. Enter correct password

**Test Data:**

Registered email

**Expected Result:**

Allow user to login

**Post Condition:**

User is redirected to home page

**Actual Result:**

User allowed to login and redirected to home page

**Status:**

Pass

**Test Case Id:**

Test\_GameAnalysis\_1

**Test Scenario:**

Verify game evaluation of chess game

**Test Case:**

Enters invalid moves in the pgn

**Pre-Condition:**

Nan

**Test Steps:**

1. Navigate to input pgn screen and fill pgn.

2.Enter “get review” button.

**Test Data:**

Pgn with invalid moves.

**Expected Result:**

Game review with no moves

**Post Condition:**

Nan

**Actual Result:**

Game review with no moves

**Status:**

Pass

**Test Case Id:**

Test\_GameAnalysis\_2

**Test Scenario:**

Verify game evaluation of chess game

**Test Case:**

Enters valid moves in the pgn

**Pre-Condition:**

Nan

**Test Steps:**

1. Navigate to input pgn screen and fill pgn.

2.Enter “get review” button.

**Test Data:**

Pgn with valid moves.

**Expected Result:**

analysis screen should display each move with:

* Evaluation score (e.g., +1.2, -0.5)
* Best alternative move (if applicable)
* A comment or suggestion if the move was suboptimal

**Post Condition:**

Nan

**Actual Result:**

The game was successfully analyzed. Evaluation and annotations matched backend response.

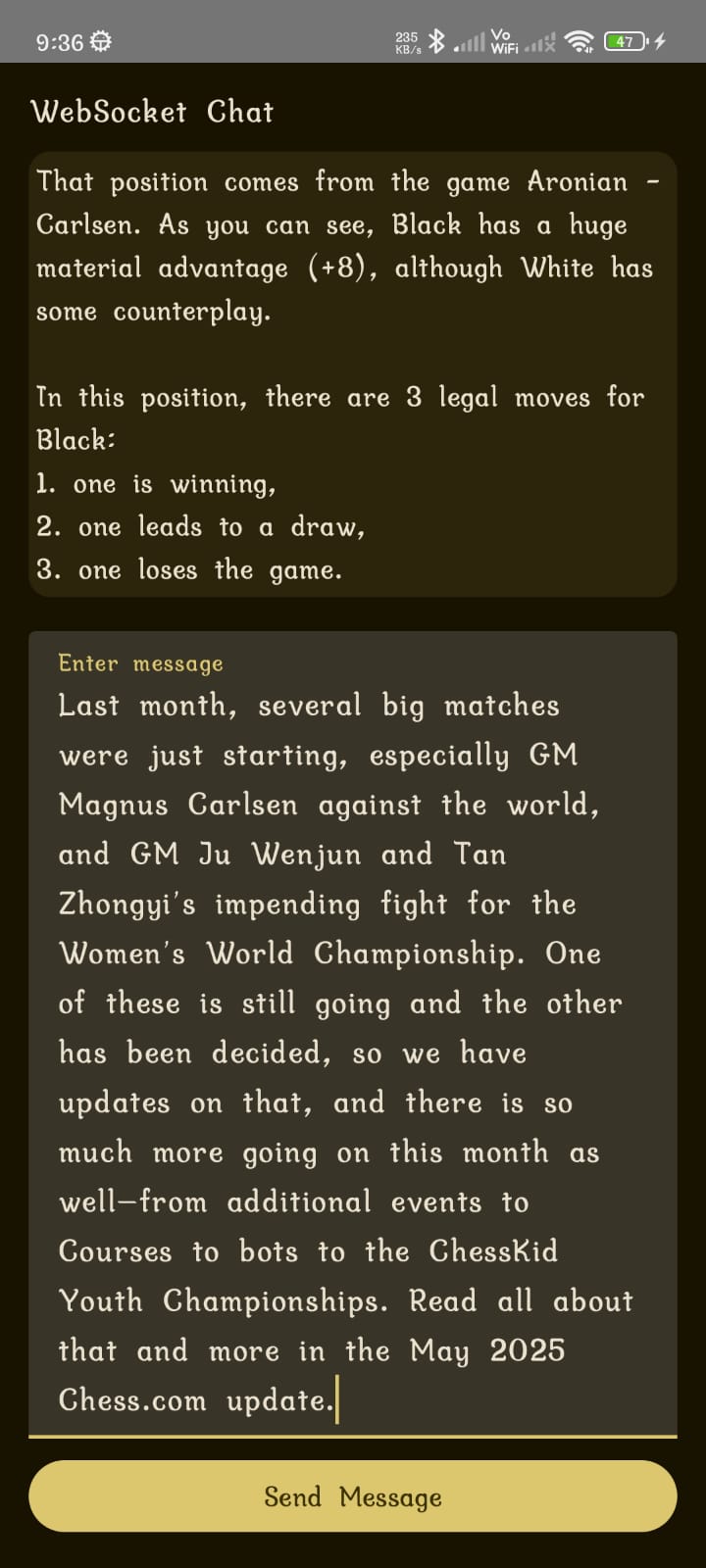
**Status:**

Pass

# Release 2 Notes

As per Release 1 notes, the targets that were to be implemented for the next delivery have been implemented.

Community Feature:



This release focuses on enhancing overall usability, addressing bugs identified through feedback and testing, and introducing new features to foster a stronger community presence within the app. Performance enhancements have also been made to improve responsiveness and reduce multiplayer latency.

# Future Work

All functionalities outlined in the SRS and user stories have been implemented as planned. In terms of future scope, several feature ideas are under consideration to further enhance the platform. These include a **Player Rating System** to reflect performance, a **Learning Section** with puzzles and tutorials, and an **Opening Explorer** to help users study common chess openings. While these features are still in the ideation or early planning phase, they offer potential for improving user engagement and the overall gameplay experience in upcoming releases.

# References

1. **Ktor Documentation (Official)**  
   https://ktor.io/docs
2. **Jetpack Compose – Android Developers**  
   <https://developer.android.com/jetpack/compose>
3. **Chess.com – Chess Rules, Puzzles, and Community**  
   <https://www.chess.com>
4. **Lichess.org – Free Online Chess Server and Open Source Tools**  
   <https://lichess.org>
5. **FIDE – International Chess Federation Rules and Regulations**  
   https://handbook.fide.com
6. **Stockfish Chess Engine (Open Source)**  
   <https://stockfishchess.org>
7. **WebSocket Basics – MDN Web Docs**  
   <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket>
8. **Kotlin Language Reference – KotlinLang.org**  
   https://kotlinlang.org/docs/home.html
9. **JetBrains Exposed – Kotlin SQL Library**  
   <https://github.com/JetBrains/Exposed>
10. **PostgreSQL Official Documentation**  
    <https://www.postgresql.org/docs>