

Sports Court Reservation App Documentation

A user-friendly platform for reserving sport courts



December 2024

Contents

[**Section 1: System Overview:** 4](#_Toc185708917)

[**Section 2: Functional Requirements:** 4](#_Toc185708918)

[**2.1** **Login (username):** 4](#_Toc185708919)

[**2.2** **Sign up / Create Account:** 4](#_Toc185708921)

[**2.3** **Check Courts:** 5](#_Toc185708923)

[**2.4** **Select Court:** 5](#_Toc185708925)

[**2.5** **Reserve Courts:** 5](#_Toc185708927)

[**2.6** **Cancel Reservation:** 5](#_Toc185708929)

[**Section 3: Non-Functional Requirements** 5](#_Toc185708931)

[**3.1 Performance:** 5](#_Toc185708932)

[**3.2 Scalability:** 5](#_Toc185708934)

[**3.3 Simplicity:** 6](#_Toc185708936)

[**3.4 Security:** 6](#_Toc185708938)

[**3.5 Reliability:** 6](#_Toc185708940)

[**Section 4: Tools and Technology** 6](#_Toc185708942)

[**4.1 Frontend:** 6](#_Toc185708943)

[**4.2 Backend:** 6](#_Toc185708948)

[**4.3 Version Control and Collaboration:** 7](#_Toc185708957)

[**4.4 Development Tools:** 7](#_Toc185708960)

[**Section 5: Risk Analysis** 8](#_Toc185708963)

[**5.1 Risk Analysis Table:** 9](#_Toc185708964)

[**Section 6: Test Cases** 9](#_Toc185708965)

[**6.1 Test Case Table:** 9](#_Toc185708966)

[**Section 7: Screenshots of Frontend** 10](#_Toc185708997)

[**Section 8: Diagrams** 11](#_Toc185708998)

[**8.1 Class Diagram:** 11](#_Toc185708999)

[**8.2 Architecture Diagram:** 12](#_Toc185709000)

[**8.3 Sequence Diagram:** 12](#_Toc185709001)

[**8.4 Use Case Diagram:** 13](#_Toc185709002)

[**Section 9: Design Pattern** 14](#_Toc185709003)

[**9.1 Model** 15](#_Toc185709004)

[**9.2 View** 15](#_Toc185709011)

[**9.3 Controller** 15](#_Toc185709017)

[**9.4 Benefits of MVC** 15](#_Toc185709024)

Sports Court Reservation App Documentation

# **Section 1: System Overview:**

The **Sports Court Reservation App** is a user-friendly platform designed to assemble the process of reserving sports courts. The app supports users in browsing real-time availability, selecting their preferred court and time slot, and completing the booking process.

# **Section 2: Functional Requirements:**

## **Login (username):**

### **Users should be able to log in either via their Google account or their phone number. The login process should be quick, and users should have the option to reset their password if they forget it.**

## **Sign up / Create Account:**

### **New users should be able to create an account by providing their basic information (e.g., username, email). The sign-up process should be straightforward.**

## **Check Courts:**

### **Users should be able to view available courts for reservation, with real-time updates showing the availability for different times.**

## **Select Court:**

### **Users can choose a specific court from available options, and the interface should show key details (e.g., location, type of court, etc.).**

## **Reserve Courts:**

### **After selecting a court, users should be able to reserve it by selecting a time slot and confirming the reservation.**

## **Cancel Reservation:**

### **Users should be able to cancel or modify their reservation if needed.**

# **Section 3: Non-Functional Requirements**

## **3.1 Performance:**

### **The application should be designed for optimal performance, ensuring that users have a smooth experience with minimal delays.**

## **3.2 Scalability:**

**The system should be scalable, meaning it can handle an increasing number of users without performance degradation**.

## **3.3 Simplicity:**

### **The user interface should be simple and intuitive, requiring minimal effort from users to navigate.**

## **3.4 Security:**

### **Security is critical to protect user data, especially when dealing with sensitive information such as personal details and payment data.**

## **3.5 Reliability:**

### **The system should be reliable and always available for users to book their courts.**

# **Section 4: Tools and Technology**

## **4.1 Frontend:**

### **- Framework: Flutter**

### **Flutter is used for building the frontend of the application. It is a powerful open-source UI toolkit from Google for building natively compiled applications for mobile, web, and desktop from a single codebase.**

### **Language: Dart**

### **Dart is the programming language used in conjunction with Flutter for frontend development. It is optimized for building fast, expressive UIs.**

## **4.2 Backend:**

### **- Framework: Django**

### **Django is a high-level Python web framework that promotes rapid development and clean, pragmatic design. It is used for building the backend of the application.**

### **- Language: Python**

### **Python is the primary programming language used for backend development in conjunction with Django. It provides simplicity and ease of use for creating and maintaining the server-side logic.**

### **- Database: PostgreSQL**

### **PostgreSQL is a powerful, open-source relational database management system used for handling data storage, queries, and database management for applications.**

### **- Database Hosting: Neon Serverless PostgreSQL**

### **Neon is a serverless PostgreSQL platform that automatically scales and provides high availability for the database, making it suitable for handling varying loads efficiently.**

## **4.3 Version Control and Collaboration:**

### **Tool: Git**

### **Git is used as the version control system to manage changes to the project codebase. It allows collaboration between team members and tracks revisions, ensuring seamless development.**

## **4.4 Development Tools:**

### **Tool: Visual Studio Code (VS Code)**

### **VS Code is the primary code editor used for the project. It provides a robust environment with extensions for Python, Dart, and Git integration, ensuring efficient workflow during development.**

# **Section 5: Risk Analysis**

## **5.1 Risk Analysis Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Description | Likelihood | Impact | Mitigation Strategy |
| Data Security Breaches | Unauthorized access to sensitive user data during authentication or payments. | High | Critical | Implement secure authentication methods |
| System Downtime | Server or database outages disrupting reservations or payments. | Medium | High | Use Neon Serverless PostgreSQL and implement failover mechanisms. |
| Scalability Issues | Inefficiency in handling a large number of users. | Medium | High | Optimize queries, utilize Django’s scalability features, plan scaling. |
| Integration Challenges | Issues with integrating third-party APIs like Google login or payment gateways. | Medium | Moderate | Test APIs thoroughly |
| User Experience Problems | Complex or unintuitive UI reducing app usability. | High | Moderate | Gather user feedback. |
| Version Control Conflicts | Merge conflicts due to poor Git branch management. | Medium | Low | Enforce branch conventions and conduct timely code reviews. |

# **Section 6: Test Cases**

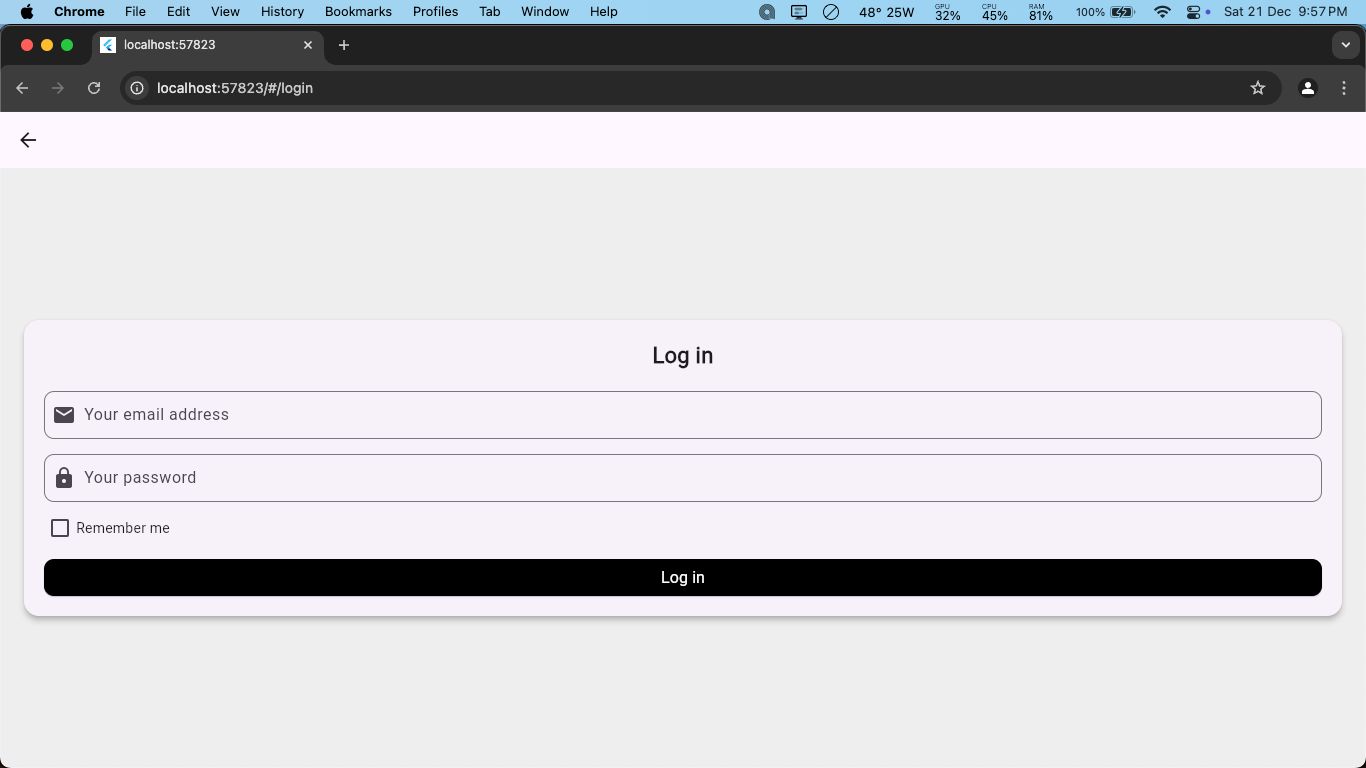
## **6.1 Test Case Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Description** | **Input Data** | **Expected Result** | **Result** | | **Status** |
| **T-001** | **Login with valid credentials** | **Valid email and password** | **User successfully logs in** | **Successfully logged in** | **Pass** | |
| **T-002** | **Login with invalid credentials** | **Invalid email and password** | **Display “Invalid credentials”** | **Displayed “Invalid Credentials”** | **Pass** | |
| **T-003** | **Reserve court with valid inputs** | **Select date and time** | **Court is successfully reserved** | **Court was successfully reserved** | **Pass** | |
| **T-004** | **Reserve court with invalid inputs** | **Past date** | **Display "Invalid date selection"** | **Displayed “Invalid date selection”** | **Pass** | |

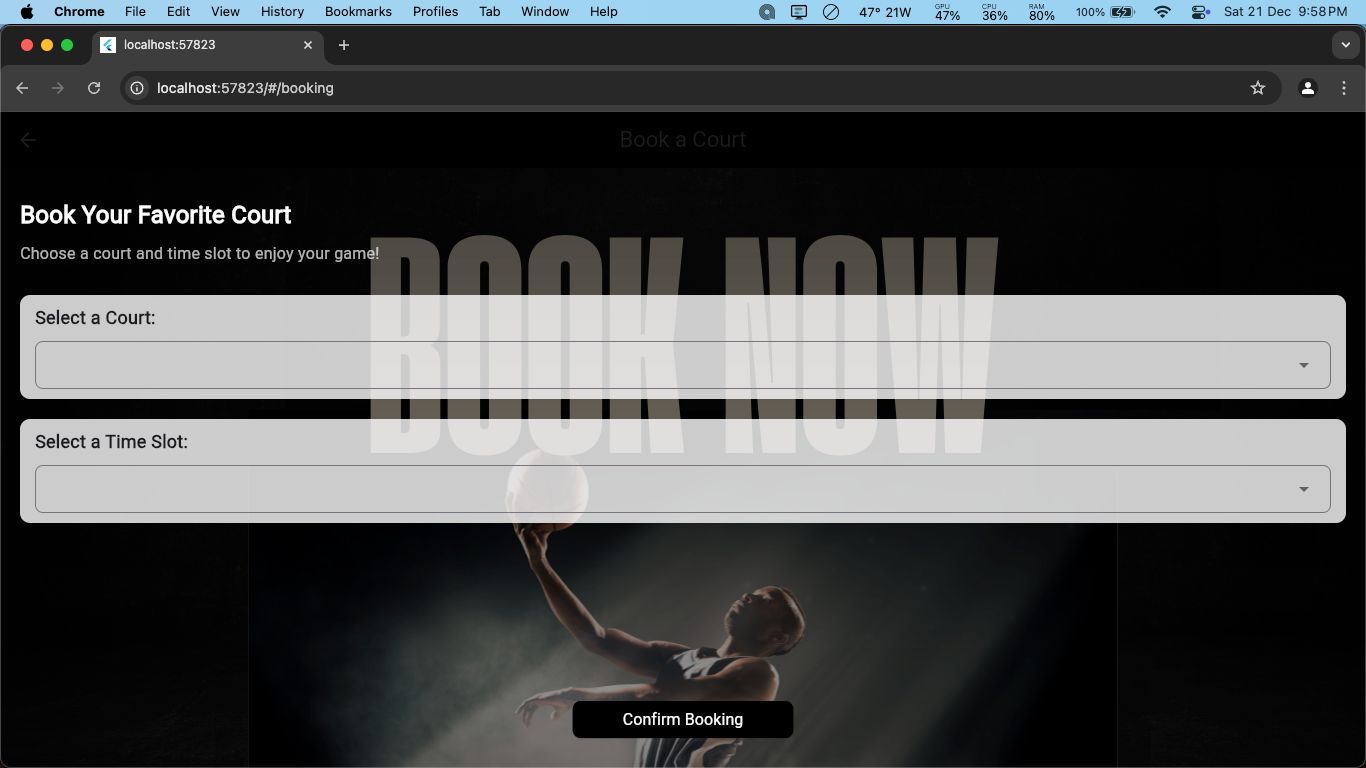
# 

# **Section 7: Screenshots of Frontend**

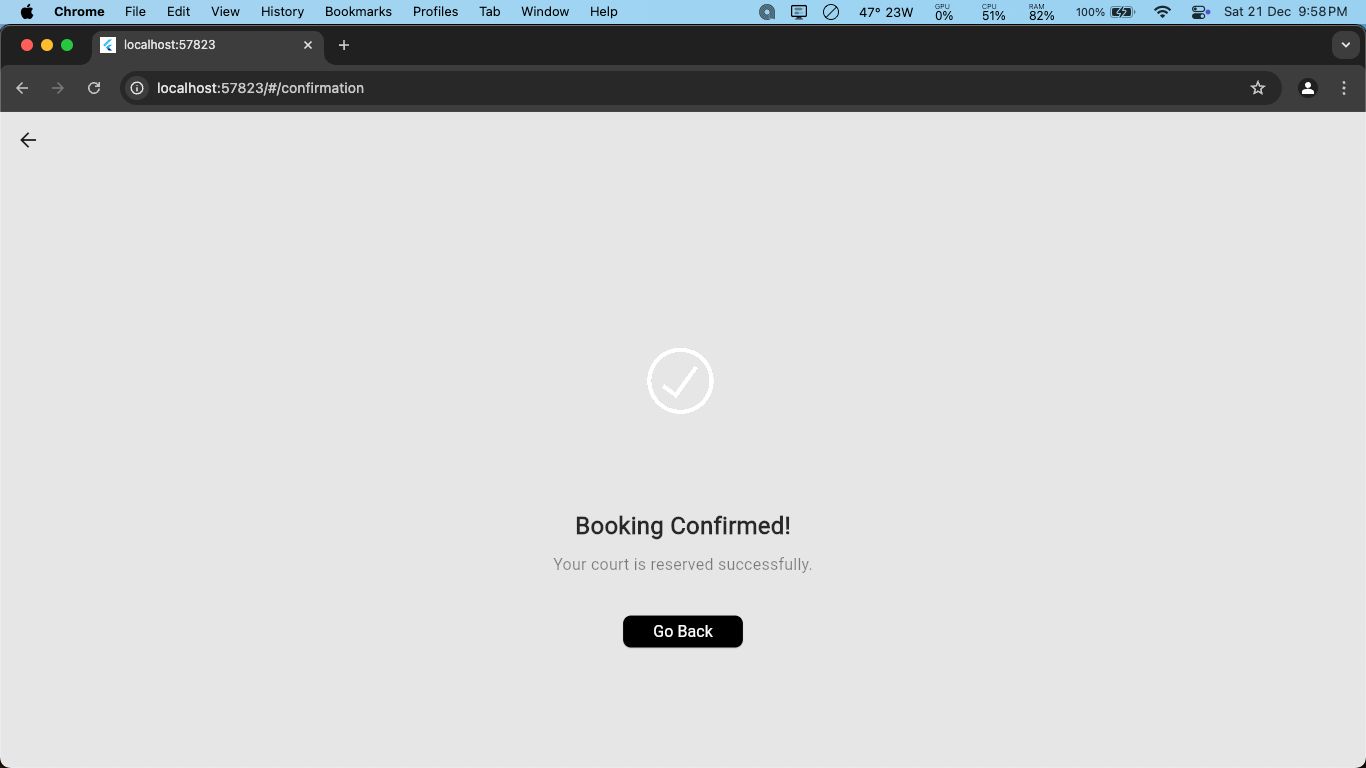
**7.1 Login/Sign-up Screen:**

****

**7.2 Court Selection Page:**

****

**7.3 Reservation Confirmation Page:**

****

# **Section 8: Diagrams**

## **8.1 Class Diagram:**

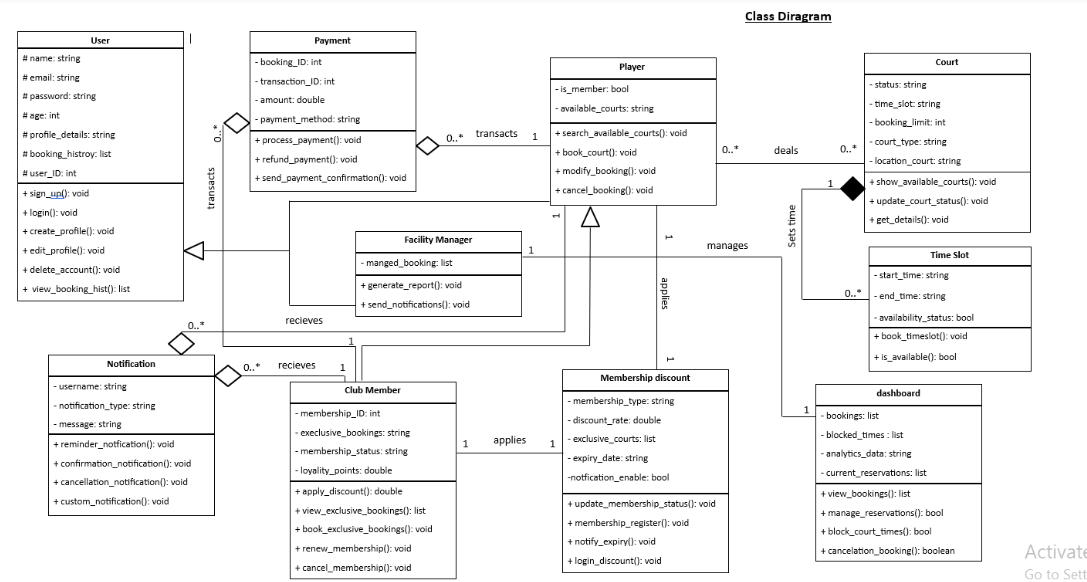


Figure Class Diagram

## **8.2 Architecture Diagram:**

A diagram of a service

Description automatically generated

Figure Architecture Diagram

## **8.3 Sequence Diagram:**

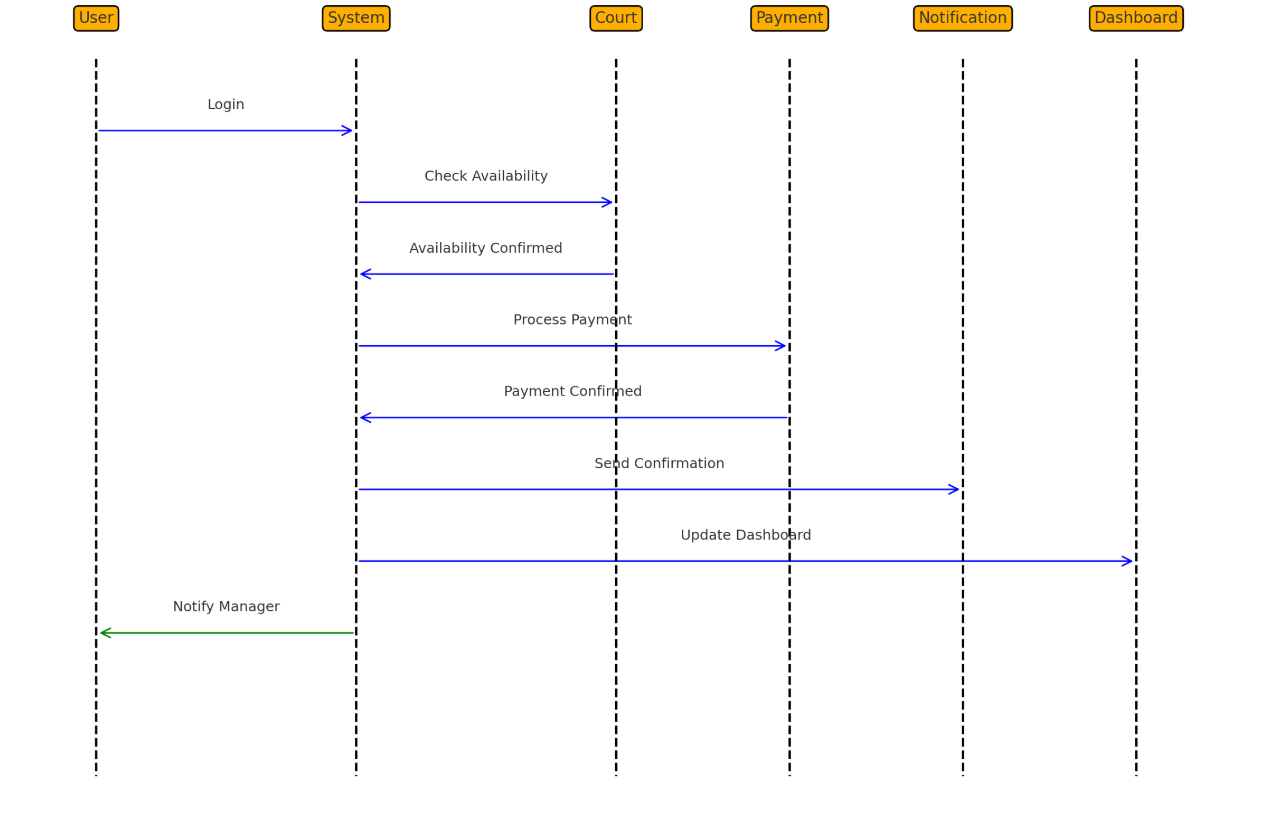


Figure Sequence Diagram

## **8.4 Use Case Diagram:**

A diagram of a sports court reservation

Description automatically generated

Figure Use Case Diagram

# **Section 9: Design Pattern**

## **9.1 Model**

### **Definition: The Model represents the data and business logic of the application.**

### **Responsibilities:**

### **Manages data and logic.**

### **Retrieves data from the database and sends it to the View for display.**

### **Updates data when needed.**

### **Implementation: In our app, Django’s ORM (Object-Relational Mapping) is used to define the model and interact with the database.**

## **9.2 View**

### **Definition: The View is responsible for displaying the user interface.**

### **Responsibilities:**

### **Displays the data to the user.**

### **Receives user input and sends it to the Controller.**

### **Implementation: The frontend is built with Flutter and Dart, which communicates with the backend through API calls.**

## **9.3 Controller**

### **Definition: The Controller manages the interaction between the Model and View.**

### **Responsibilities:**

### **Processes user input from the View.**

### **Updates the Model as necessary.**

### **Sends updated data to the View.**

### **Implementation: In Django, the controllers (views) handle the API requests from the frontend, process them, and return the data to the View.**

## **9.4 Benefits of MVC**

### **Separation of Concerns: Each component is independent, making the code easier to maintain and scale.**

### **Reusability: Logic can be reused across multiple components.**

### **Scalability: New features can be added with minimal changes to existing components.**