

**TITLE TO BE ADDED**

by

**Mahdi Mohammad Al Ajami 202302965**

**Ahmad Ghalayini 202301419**

**Aya Akila 202301091**

**Moustapha Aajineh 202304458**

**Ahmad Khalil 202303762**

**Bilal Shehade 202303538**

**Walid Al Rayis 202201414**

Department of Electrical and Computer Engineering

Faculty of Engineering

Supervised by Dr. Iman Haidar

2024/2025

Table of Contents

[**Project Description** 4](#_Toc196664274)

[**Introduction** 4](#_Toc196664275)

[**Project Statement** 4](#_Toc196664276)

[**Project Overview** 4](#_Toc196664277)

[**Project Objectives** 5](#_Toc196664278)

[**Background** 6](#_Toc196664279)

[**Literature Review** 6](#_Toc196664280)

[**Applications:** 8](#_Toc196664281)

[**Alternative Design:** 8](#_Toc196664282)

[**Project Planning** 9](#_Toc196664283)

[**Project Constraints** 9](#_Toc196664284)

[**Project Issues** 9](#_Toc196664285)

[**Team Members’ Tasks** 9](#_Toc196664286)

[**Ethical Issues** 10](#_Toc196664287)

[**Software Model Process** 10](#_Toc196664288)

[**Feasibility Study** 10](#_Toc196664289)

[**Tools/Technology** 10](#_Toc196664290)

[**Standards** 12](#_Toc196664291)

[**Milestone** 12](#_Toc196664292)

[**Requirements** 13](#_Toc196664293)

[**Functional Requirements** 13](#_Toc196664294)

[**Data Requirements** 13](#_Toc196664295)

[**Non-Functional Requirements** 14](#_Toc196664296)

[**User Cases** 15](#_Toc196664297)

[**System Modeling** 19](#_Toc196664298)

[**Introduction** 19](#_Toc196664299)

[**Context modeling** 20](#_Toc196664300)

[**Interaction Model** 21](#_Toc196664301)

[**Structural Model** 22](#_Toc196664302)

[**Behavioural Model** 23](#_Toc196664303)

[**Model-Driven Engineering** 24](#_Toc196664304)

[**Software Testing** 25](#_Toc196664305)

[**Validation Testing** 25](#_Toc196664306)

[**Defect Testing** 25](#_Toc196664307)

[**Verification Activities** 25](#_Toc196664308)

[**Deployment Testing** 25](#_Toc196664309)

[**Release Testing** 26](#_Toc196664310)

[**User Testing** 26](#_Toc196664311)

[**Code, Frontend & Backend** 26](#_Toc196664312)

[**Conclusion** 41](#_Toc196664313)

[**References:** 42](#_Toc196664314)

**Table of Figures**

[Figure 1: UML Diagram 18](#_Toc196664315)

[Figure 2: Flow Diagram like the UML showing how each button works and its navigations 19](#_Toc196664316)

[Figure 3: Context Model 20](#_Toc196664317)

[Figure 4: Process Model 20](#_Toc196664318)

[Figure 5: Interaction Model that shows the sequence for user booking 21](#_Toc196664319)

[Figure 6: Interaction Model that shows sequence for venue owners 22](#_Toc196664320)

[Figure 7:Structural Model 23](#_Toc196664321)

[Figure 8: Behavioural Model 24](#_Toc196664322)

[Figure 9: Authentication Page Sample Code 27](#_Toc196664323)

[Figure 10: Forget Password Page Sample Code 28](#_Toc196664324)

[Figure 11: Home Page Sample Code 29](#_Toc196664325)

[Figure 12: Booking Page Sample Code 30](#_Toc196664326)

[Figure 13: Category Page Sample Code 31](#_Toc196664327)

[Figure 14: Posting Page Sample Code 32](#_Toc196664328)

[Figure 15: Shows Number of Users 33](#_Toc196664329)

[Figure 16: Shows Tools to be used for authentication 33](#_Toc196664330)

[Figure 17: Shows what devices the authentication works on 33](#_Toc196664331)

[Figure 18: Shows who are logged in 34](#_Toc196664332)

[Figure 19: Shows firebase ID for HJEZLE 34](#_Toc196664333)

[Figure 20: Google Account Permission 34](#_Toc196664334)

[Figure 21: Shows method of logging in 34](#_Toc196664335)

[Figure 22: Communication Page with the user used for confirmation 35](#_Toc196664336)

[Figure 23: Shows whether connection is applied between flutterflow and firebase 35](#_Toc196664337)

[Figure 24: Supabase inside of FlutterFlow where tables were imported 36](#_Toc196664338)

[Figure 25: After uploading an image and add a new venue, it is downloaded on Supabase Storage 36](#_Toc196664339)

[Figure 26: Supabase ID for FlutterFlow Project Integration 36](#_Toc196664340)

[Figure 27:This is were SupaBase storage is taken 37](#_Toc196664341)

[Figure 28: The table given by SupaBase 37](#_Toc196664342)

[Figure 29: Button that saves data inside of SupaBase (FlutterFlow) 38](#_Toc196664343)

[Figure 30: This table is created after uploading a new venue 38](#_Toc196664344)

# **Project Description**

## **Introduction**

HJEZLE is a cutting-edge mobile application developed using flutter that revolutionizes the booking process for sports facilities. The application offers a user-friendly interface where sports enthusiasts can register, explore a wide variety of sports venues, and communicate directly with facility owners to reserve slots and obtain real-time updates. Facility owners, in turn, benefit from a platform that enables them to efficiently list, manage, and promote their venues. By integrating real-time backend services, HJEZLE aims to eliminate inefficiencies in traditional booking systems and foster enhanced communication engagement in the sports and recreational sectors.

## **Project Statement**

In Lebanon, access to sports facilities such as tennis courts, paddle arenas, and football fields remains a significant challenge due to the lack of a centralized and efficient booking system. Currently, individuals seeking to reserve a sports venue must rely on manual, time-consuming processes, often making multiple phone calls to inquire about availability, pricing, and location suitability. This fragmented approach not only results in inconvenience and inefficiency but also discourages people from engaging in sports activities, ultimately affecting community well-being and physical fitness. Moreover, there is a lack of transparency and accessibility regarding sports venues. Many individuals are unaware of the full range of facilities available near them, leading to missed opportunities and underutilized venues. The absence of a unified platform means users struggle to compare options based on location, cost, amenities, and time slots, forcing them to rely on word-of-mouth recommendations or outdated information. This issue is further exacerbated by inconsistent pricing models and limited communication channels between venue owners and potential users. To address these inefficiencies, we propose the development of a comprehensive mobile application that streamlines the sports venue reservation process. This platform will serve as an integrated marketplace, allowing users to seamlessly search for nearby facilities, view real-time availability, compare pricing, and book their preferred slots with minimal effort. By leveraging within Lebanon, digital automation and user-friendly interfaces, our solution aims to eliminate unnecessary hassle, enhance accessibility, and foster a more active sports culture. This application will not only benefit sports enthusiasts but will also provide venue owners with a structured and efficient way to manage their reservations, optimize facility usage, and increase revenue. By bridging the gap between demand and availability, our platform aspires to redefine sports facility booking, making it a streamlined, data-driven, and user-centric experience.

## **Project Overview**

**HJEZLE** is a revolutionary Flutter-built mobile application transforming sports facility booking. It provides a streamlined and intuitive solution for both sports enthusiasts and venue operators. Users can easily browse a wide range of sports venues, check availability, and reserve spaces, all from their mobile devices. Beyond field reservations, the app also allows users to book spots for sports events. **HJEZLE** also provides a user-friendly interface enabling users to explore available sports facilities, check real-time availability, and communicate directly with venue owners. The app aims to replace inefficient manual booking processes with an automated, digital solution that enhances accessibility and convenience.

For venue owners, **HJEZLE** offers an efficient platform to manage facilities, set available timeslots, and streamline the entire booking process. Real-time updates and direct communication features empower owners to effectively promote, manage, and optimize bookings, enhancing operational efficiency.

**HJEZLE** eliminates tedious phone calls and inquiries, providing instant access to information and facilitating a faster, more reliable booking experience. The platform operates on a revenue-sharing model, taking commission from each booking to ensure sustainable growth for both the app and the venues. With **HJEZLE**, finding and booking sports facilities is effortless. Simply open the app, choose your venue, and secure your reservation in seconds. Built using Flutter for cross-platform support and powered by Firebase for real-time backend services, **HJEZLE** bridges the gap between facility providers and users, fostering a more organized and efficient booking experience.

## **Project Objectives**

The objectives of **HJEZLE** revolve around improving the accessibility and efficiency of sports venue reservations. Traditional booking methods often involve phone calls, in-person visits, or social media inquiries, which can be time-consuming and unorganized. By leveraging a digital platform, the project aims to centralize all booking-related tasks, ensuring a seamless experience for both users and venue owners. The platform not only facilitates easy reservations but also provides a marketplace where venue owners can promote their services, set pricing, and communicate with potential customers. Additionally, security and reliability are key aspects of the system, ensuring that users' data is protected and that transactions are carried out smoothly.

The main objectives of the project are the following:

* Developing a mobile application that allows users to browse and book sports venues in a few simple steps.
* Integrating a secure authentication system that enables users to register and log in easily.
* Providing venue owners with a platform to list their sports facilities, including location, pricing, and availability.
* Enabling real-time communication between users and venue owners for better coordination.
* Implementing a secure payment system that supports online transactions for booking confirmations.
* Ensuring scalability and performance by using an efficient backend, such as Firebase.

## **Background**

The sports and recreational industry have seen tremendous growth in recent years, driven by increased health awareness and the popularity of organized sports. In countries like Lebanon, local communities place great value on accessible sports facilities, yet many venues suffer from outdated booking systems and manual processes. Traditional methods often result in double bookings, inefficient resource allocation, and a disconnect between service providers and customers. In this context, digital transformation becomes essential. By adopting modern software solutions such as HJEZLE, stakeholders in the sports industry can enjoy faster, more reliable reservation processes, improved communication channels, and enhanced customer satisfaction. This digital shift not only addresses immediate operational challenges but also supports long-term growth by promoting local sports and recreational activities. Furthermore, integrating real-time communication and data management features fosters a more interactive and responsive environment, which is vital for competitive markets.

In response to the growing need for a streamlined booking system, the HJEZLE mobile application was conceived. Built using Flutter, the app aims to address these pressing challenges by providing an all-encompassing solution for booking sports facilities. HJEZLE allows users to effortlessly browse, and book available sports venues based on location, sport, and availability, eliminating the need for lengthy phone calls and manual coordination. Venue owners, in turn, will be able to manage their facilities, track bookings, and promote their offerings on a centralized platform designed to increase operational efficiency.

The app seeks to provide a seamless and secure booking experience, with features such as real-time availability updates, instant bookings, and a secure payment system, designed to cater to both sports enthusiasts and venue operators. Moreover, HJEZLE’s user feedback system, personalized profiles, and social media sharing will foster greater engagement, enhance user experience, and help users make informed decisions based on reviews and preferences.

By addressing the gaps in the current sports booking system, HJEZLE is poised to revolutionize how sports facilities are booked in Lebanon, making it easier for individuals to enjoy recreational activities and for venue owners to operate their facilities more efficiently. The app stands to contribute to the growth of the sports culture in Lebanon, providing a more connected and active community.

## **Literature Review**

A review of similar products and academic research underscores the need for a sophisticated, user-friendly booking system in the sports industry. In Lebanon, a study by Abdul Rahman et al. (2022) explored the optimization of sports facility booking systems, emphasizing the benefits of localized, cost-effective solutions while also noting challenges related to scalability and user interface design. Commercially, the Sporty app—reviewed by Global Innovations Inc. (2021)—demonstrates strong real-time booking capabilities and an intuitive user interface tailored to its local market; however, despite its usability, it lacks integrated communication tools that can streamline customer support. On an international level, the PlayFinder platform, as analyzed by TechCrunch (2020), offers a comprehensive multi-vendor database and advanced functionalities, yet it suffers from high subscription fees and a complex navigation structure that may deter casual users. These examples illustrate a common trend: while many existing solutions excel in specific areas, they often struggle to provide a holistic experience that combines real-time functionality, ease of use, and integrated communication.

The literature indicates that the ideal solution should merge the benefits of real-time booking and dynamic user interfaces with robust communication features and efficient administrative management. HJEZLE is designed to meet these demands by leveraging modern cross-platform technologies and backend-as-a-service (BaaS) platforms that offer scalable, secure, and flexible support.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ref # | Authors/Sources | Description | Advantages | Problems |
| 1 | Abdul Rahman et al. (2022) | Study on optimizing sports facility booking systems in Lebanon with a focus on localized, cost-effective solutions. | Cost-effective; user-centric; addresses local market needs. | Scalability challenges; occasional interface design issues |
| 2 | Global Innovations Inc. (2021) – Sporty App Review | Review of the Sporty app for sports venue booking, emphasizing real-time booking and an intuitive UI | Real-time booking; simple and intuitive interface; group booking feature. | Lack integrated communication tools; limited administrative functions for venue owner. |
| 3 | TechCrunch (2020) – PlayFinder App Analysis | Analysis of PlayFinder, a global platform offering multi-vendor support and advanced booking functionalities. | Extensive database; robust multi-vendor support; advanced features. | High subscription fees; complex navigation structure. |
| 4 | OpenPlay | OpenPlay is a UK-based app for booking community sports venues like football, tennis, and cricket. | Wide network of sports venues; user-friendly interface; group booking; venue reviews and ratings. | Limited to the UK; lacks comprehensive management tools for venue owners. |
| 5 | BookMySports | BookMySports is an Indian platform for real-time booking of various sports facilities, including cricket, badminton, and football. | Extensive range of facilities; real-time booking; strong customer support; discounts and offers. | Limited international reach; inconsistent availability and pricing; unreliable user reviews. |

## **Applications:**

Real-Time Availability: Without having to phone or come to the facility, users can see the current availability of different pitches in their area, making it easy for them to choose and reserve a time window that works for them.

Group Booking and Management: By allowing users to invite friends or teammates to join a game, the app can help with group bookings. In addition, it can send out notifications for next games and handle payments.

Rating and Reviews: Based on their personal experiences, users are able to evaluate and review pitches. This feature encourages facility owners to uphold high standards and assists others in making well-informed decisions.

Discounts and Promotions: To entice regular users to make reservations through the app, it may provide exclusive discounts, loyalty plans, or promotions.

Event Planning: Using the app, users can plan tournaments or informal get-togethers, setting up games, inviting players, and even keeping score or statistics for competitive play.

## **Alternative Design:**

The "Community Sports Hub" is an innovative design concept for a pitch reservation app that emphasizes user engagement and social interaction. It features a personalized dashboard that displays upcoming bookings, favorite pitches, and recent activities, along with a map integration that highlights nearby pitches filtered by sport type, availability, and user ratings. Users can click on each pitch for detailed information, including photos and amenities. The app also incorporates social features, allowing users to connect, form teams, and challenge friends to matches, while enabling the creation and sharing of community events. Additionally, it offers skill level filters to help users find suitable matches and groups, and includes push notifications for booking confirmations, reminders for upcoming matches, and alerts for available pitches nearby. This design aims to foster a vibrant sports community while streamlining the pitch reservation process.

# **Project Planning**

Project planning is the foundation upon which successful initiatives are built. It involves developing a comprehensive roadmap that outlines objectives, deliverables, timelines, resource allocation, risk management, and communication strategies. Effective planning considers both the big picture and granular details, integrating elements such as a Work Breakdown Structure (WBS), Gantt charts, and stakeholder analysis. By breaking the project into manageable tasks and milestones, the planning phase not only forecasts potential challenges but also sets a clear path to achieving strategic objectives. In today’s competitive environment, project planning also emphasizes adaptability, integrating iterative review points and contingency measures to respond to unforeseen issues while staying aligned with organizational goals.

## **Project Constraints**

Hjezle’s development faces several key constraints that must be managed carefully throughout the project lifecycle. On the technical side, the app is required to run smoothly on both iOS and Android platforms, using Flutter and FlutterFlow as the designated front-end frameworks. The backend infrastructure must support robust user authentication, real-time reservation capabilities, and secure payment processing through reliable payment gateways. Additionally, a scalable cloud-based hosting solution is necessary to manage user data and reservations, while strict adherence to standard security protocols is essential for protecting sensitive information. Time constraints dictate that the entire development cycle, including comprehensive testing and debugging, must be completed within a fixed period to ensure a timely launch. Budget limitations are also in place, with recurring infrastructure costs related to cloud services, API integrations, and payment gateways; expenses may further increase due to paid plugins and services required by FlutterFlow, as well as funds allocated for post-launch marketing and customer support. Finally, user constraints mandate that the application remain simple and accessible, ensuring an intuitive UI/UX design that caters effectively to both venue owners and players.

## **Project Issues**

Every project encounters a range of issues throughout its lifecycle. Common challenges include integration difficulties between various systems, misinterpretation or incomplete documentation of requirements, and scope creep driven by uncontrolled changes. Communication breakdowns and stakeholder misalignments further exacerbate these issues. To mitigate risks, effective projects implement regular status reviews, risk assessment sessions, and change control procedures. Some specific issues often observed are:  
• Inadequate user involvement  
• Unrealistic deadline and budget estimations  
• Poor interdepartmental coordination  
• Resistance to change from stakeholders  
Addressing these issues early through comprehensive monitoring and adaptive management techniques is essential for steering the project back on course.

## **Team Members’ Tasks**

A project’s success relies on a well-coordinated team where each member has clearly defined roles and responsibilities. The project manager oversees the overall process, ensuring that objectives are met and that communication flows smoothly between stakeholders. Designers create intuitive and visually appealing interfaces, while developers implement the core functionalities, integrating backend systems and ensuring technical robustness. Quality assurance teams conduct rigorous testing, and support staff handle deployment and post-launch maintenance. Employing tools like the RACI matrix (Responsible, Accountable, Consulted, Informed) ensures that each task is assigned appropriately and that accountability is maintained throughout the project lifecycle. Clear task delegation fosters collaboration and minimizes redundancies.

## **Ethical Issues**

Ethical considerations are paramount in project management, as decisions made at every stage can affect the welfare of both the organization and its stakeholders. Project managers are expected to adhere to professional codes of conduct, ensuring honesty, fairness, and transparency in all dealings. Ethical issues might include conflicts of interest, data privacy concerns, and the potential for bias in decision-making. Establishing clear ethical guidelines and having an independent review process can help resolve dilemmas when they arise. For instance, when budget pressures tempt managers to compromise quality, a robust ethical framework ensures that the interests of end users and the broader community are not sacrificed. This commitment to ethical behavior builds trust and fosters long-term sustainability.

## **Software Model Process**

For Hjezle, we adopted a hybrid software development model that combines the strengths of agile methodologies and iterative development. This approach enabled us to benefit from agile’s rapid iterations, continuous user feedback, and flexibility in handling evolving requirements, while also incorporating the structured review and refinement cycles typical of iterative models. By merging these frameworks, we could implement robust quality assurance, effective risk management, and proactive process improvements throughout the project lifecycle. This model was particularly well-suited to managing Hjezle’s complex requirements, technical constraints, and dynamic stakeholder needs, ensuring a balanced and responsive development process that met both short-term deadlines and long-term strategic objectives.

## **Feasibility Study**

Before significant resources are committed, a feasibility study assesses the viability of the project from multiple perspectives. Commonly evaluated factors include:  
• **Technical Feasibility:** Can the organization’s current technology and expertise support the project requirements?  
• **Economic Feasibility:** Does the cost-benefit analysis justify the investment?  
• **Legal Feasibility:** Are there regulatory or legal constraints that could impact the project?  
• **Operational Feasibility:** How well will the project integrate with existing operations?  
• **Scheduling Feasibility:** Can the project be completed within the required timeframe?  
Often summarized by the TELOS framework (Technical, Economic, Legal, Operational, Scheduling), the study provides a clear decision-making basis for proceeding, revising, or abandoning a project.

## **Tools/Technology**

To develop **Hjezle** efficiently, a diverse and carefully selected array of software tools, frameworks, and cloud services is employed to ensure a robust, scalable, and user-friendly mobile application for both iOS and Android platforms. The development process is structured into two key components: **front-end** and **back-end**, each utilizing industry-leading technologies to maximize performance, security, and efficiency.

**Front-End Development**

The front-end of **Hjezle** focuses on delivering a seamless and intuitive user experience. **Flutter**, a powerful cross-platform framework, was chosen for its ability to compile to native code, offering **high performance and smooth animations** across both iOS and Android devices. Complementing this, **FlutterFlow**, a no-code/low-code development platform, accelerates UI design and front-end implementation, enabling rapid iterations without extensive manual coding. This approach significantly reduces development time while maintaining **pixel-perfect design consistency**.

For location-based functionalities, the **Google Maps API** is integrated to allow users to **search for sports venues, access navigation features, and retrieve real-time location data**. Authentication is handled by **Firebase Authentication**, ensuring secure and streamlined login options via **email, phone number, or third-party providers (Google, Facebook, Apple, etc.)**. To enhance the development workflow, version control is managed through **GitHub or GitLab**, facilitating efficient code collaboration. Internal team communication is streamlined via **Discord and WhatsApp**, ensuring effective coordination throughout the development cycle.

**Back-End Development**

The back-end architecture of **Hjezle** is designed to be **scalable, secure, and optimized for real-time interactions**. At its core, **Firebase** is utilized to handle database management, authentication, and cloud storage. Specifically, **Firestore (Firebase’s NoSQL database)** is implemented to **provide real-time syncing of reservations, user data, and venue availability**, making it ideal for a sports booking platform. An alternative **Supabase (Postgres-based open-source database)** is considered for cases where relational database functionalities are needed.

To ensure **secure payments and transactions**, the platform integrates trusted payment gateways such as **Stripe, PayPal, and local providers**, offering multi-currency support and fraud protection. File storage is managed through **Firebase Storage and AWS S3**, enabling users to **upload and retrieve images, documents, and other media securely**. Additionally, **Node.js** is available as a back-end processing option for scenarios where additional server-side logic is required beyond Firebase’s serverless capabilities.

**Testing, Security, and Deployment**

Comprehensive testing is a critical aspect of **Hjezle’s** development, ensuring reliability and performance across all user interactions. The **Flutter Testing Framework** is employed for **unit and integration testing**, verifying the stability of core functionalities. Security testing and API validation are conducted using **OWASP ZAP and Postman**, while **Firebase Performance Monitoring** continuously tracks the app’s responsiveness and load times.

Finally, **deployment and hosting** are managed through the **Google Play Store and Apple App Store**, ensuring that **Hjezle** reaches its intended audience efficiently. With **Firebase’s auto-scaling capabilities**, the platform can handle increasing user demand without performance degradation, making it a highly **resilient and scalable** solution for sports facility reservations.

By leveraging these cutting-edge tools and technologies, **Hjezle** is designed to be a **high-performance, secure, and scalable** platform that offers a seamless experience for users while optimizing operational efficiency for venue owners.

## **Standards**

Adherence to industry standards ensures consistency, reliability, and quality across all project phases. Standards such as the Project Management Body of Knowledge (PMBOK® Guide), ISO 21500:2012 for project management, and specific technical standards (e.g., IEEE, ISO/IEC standards) provide a structured framework for project execution. These guidelines define best practices for risk management, quality assurance, and stakeholder communication, among other areas. Compliance with such standards not only streamlines internal processes but also enhances credibility with clients and stakeholders. The integration of standard operating procedures and regular audits ensures that projects remain aligned with both regulatory requirements and organizational objectives.

## **Milestone**

Below is a table summarizing the key project milestones for Hjezle. These milestones serve as critical checkpoints throughout the project lifecycle, enabling the team to monitor progress, ensure accountability, and make timely adjustments to meet strategic objectives.

|  |  |
| --- | --- |
| **Milestone** | **Description** |
| Completion of Initial Planning Phase | Finalize project scope, objectives, and a detailed project plan. |
| Finalization of Design Phase | Complete UI/UX design, system architecture, and technical specifications. |
| Development of Working Prototype | |  | | --- | |  |  |  | | --- | | Build a functional prototype to validate core features and design assumptions. | |
| Completion of Integration and Testing | |  | | --- | |  |  |  | | --- | | Integrate all components and conduct thorough testing to ensure quality, performance, and reliability. | |
| Formal Deployment and Launch | |  | | --- | |  |  |  | | --- | | Publish the app on the Google Play Store and Apple App Store after final validations. | |
| Post-Implementation Review | Evaluate project outcomes, capture lessons learned, and identify opportunities for future improvements. |

This table clearly outlines each milestone along with its description, providing a concise view of the project's major achievements and their significance. Each milestone acts as a target that supports effective monitoring and timely adjustments throughout the Hjezle development process.

# **Requirements**

## **Functional Requirements**

The **User Management System** in Hjezle will allow users to register using their email addresses with built-in verification mechanisms to ensure account validity. It will implement a role-based authentication system that distinguishes between general users and venue owners. Users will be able to manage their profiles, including updating personal information, preferences, and viewing their booking history. The system will include secure procedures for password resetting, supported by verification protocols, and will also enable third-party authentication using services such as Google, Facebook, and Apple for increased accessibility and convenience.

For **Venue Discovery and Booking**, the system will offer geolocation-based search functionality to help users find nearby sports venues quickly. Users will be able to filter venues by sport type, pricing range, and available amenities. The system will display real-time venue availability integrated with an interactive calendar interface and allow instant bookings, complete with confirmation mechanisms. Group bookings will also be supported, with features that facilitate participant management. Additionally, users will have the option to save their favorite venues for quicker access during future bookings.

The **Venue Management Portal** will enable venue owners to register their venues through a formal verification process. Once onboarded, owners can define and update available time slots, pricing structures, and upload multimedia content—such as photos or virtual tours—to better showcase their facilities. The system will provide comprehensive booking management tools that include approval workflows and allow venue owners to review and respond to bookings. Furthermore, operational analytics and utilization reports will be automatically generated to aid owners in optimizing their offerings.

The system will also include a robust **Communication Module**. It will allow direct messaging between users and venue owners, streamlining inquiries and clarifications. Automated notifications and reminders will inform users of booking confirmations and upcoming reservations. A ratings and review feature will enable community feedback, while moderation mechanisms will maintain quality and trust. Additionally, venue owners will be able to broadcast announcements for events, promotions, or schedule changes.

Regarding **Payment Processing**, the system will integrate with secure payment gateways, enabling users to complete transactions safely and efficiently. It will support a variety of payment methods, including credit cards and mobile wallets. Digital receipts and booking confirmations will be generated for every transaction. A transparent revenue-sharing model will be in place to calculate and display financial distributions to venue owners. Lastly, refund processing capabilities will be incorporated, governed by appropriate authorization workflows to ensure fairness and security.

## **Data Requirements**

In terms of **User Data**, the system will store authentication credentials in encrypted form to safeguard user privacy. Profiles will be maintained with relevant contact information and preference settings, while booking history and transaction details will be archived to allow users to review past activity. Engagement metrics such as frequently visited venues or search behavior will be collected to enable personalized recommendations and experiences.

For **Venue Data**, the platform will maintain detailed venue profiles that include location coordinates, photos, and virtual tours. Availability calendars will reflect booking status in real time, helping both users and owners manage schedules effectively. Historical booking data will be archived to enable trend analysis and forecasting. Additionally, the system will support dynamic pricing, accounting for seasonal variations and promotional periods.

Regarding **Transaction Data**, each booking transaction will be logged with an audit trail for transparency. Payment records will be securely stored using encryption to ensure compliance with financial data protection standards. Commission structures and payouts to venue owners will be calculated and recorded systematically. The system will also preserve logs of cancellations, modifications, and refund requests for accountability and analysis.

In terms of **Analytics Data**, the system will continuously collect and aggregate metrics related to venue utilization, user engagement, and behavioral patterns such as search trends and booking conversion rates. These analytics will feed into performance monitoring systems that help optimize the platform’s responsiveness and service quality over time.

## **Non-Functional Requirements**

**Performance Requirements**

The application will be optimized to respond to user interactions within two seconds under normal network conditions, ensuring a smooth user experience. It will support concurrent operations by multiple users without notable performance degradation. Image loading will be enhanced using progressive rendering techniques, and data-heavy screens will utilize efficient pagination. The app will be lightweight enough to function smoothly on mid-range and budget mobile devices.

**Reliability Requirements**

Hjezle will maintain an uptime of 99.5%, excluding scheduled maintenance windows. Offline operation will be partially supported through local caching and synchronization mechanisms. Automated backups will be scheduled regularly to prevent data loss, and the application will degrade gracefully during internet outages. Comprehensive error handling and recovery protocols will ensure resilience during unexpected failures.

**Security Requirements**

User passwords will be encrypted using industry-standard hashing algorithms, and all communication between the client and server will be secured using HTTPS protocols. Only authenticated users will have access to booking and venue management functionalities. Session management will include timeouts to prevent unauthorized access. All payment processes will comply with PCI DSS standards, and the system will implement rate limiting to guard against brute-force attacks. Sensitive user data will be secured both during transmission and while at rest.

**Usability and Humanity Requirements**

Hjezle will feature an intuitive interface that caters to users of all technical skill levels. It will provide clear feedback for all actions and present complex functionalities using progressive disclosure to reduce cognitive overload. The user interface will be designed for minimal workflow friction, ensuring that common tasks require the fewest possible steps. Contextual help will be available throughout the app, and the design will conform to accessibility guidelines to support users with disabilities. Multiple input methods, such as touch and voice where applicable, will be supported.

**Look and Feel Requirements**

The user interface will adhere to a clean and consistent visual design, applying standardized typography, color palettes, and layout schemes. Responsive design principles will ensure that the application adapts to different screen sizes and resolutions. The visual identity will align with Hjezle’s branding guidelines, and navigation will remain consistent across different screens. The layout will make effective use of white space to enhance readability, and the app will offer both light and dark modes for user preference.

**Operational and Environmental Requirements**

The application will be fully compatible with both Android and iOS operating systems. It will be optimized for devices with limited processing capabilities, and its data usage will be efficient to accommodate low-bandwidth environments. The application will maintain essential functionality even during intermittent connectivity. Furthermore, battery consumption will be minimized through optimized resource usage, and the app will adapt intelligently to various device and environmental conditions.

**Cultural and Political Requirements**

Hjezle will support interfaces in Arabic, English, and French to reflect the linguistic diversity of its user base. All interface elements—including icons and terminology—will be culturally appropriate. Regional formatting standards for dates, times, and currencies will be applied. The application will avoid the inclusion of politically sensitive material and will respect local norms in imagery and content presentation. Support for right-to-left text orientation will be available for Arabic language users.

**Legal Requirements**

The platform will implement data protection measures that comply with Lebanese privacy laws. Users will be presented with transparent terms of service and privacy policies that are accessible throughout the application. Explicit user consent will be obtained for all data collection and processing activities. Age verification will be implemented where legally necessary. The application will adhere to international standards for digital accessibility and will respect intellectual property rights related to user-generated content. The system will also maintain documentation demonstrating compliance with relevant regulations for auditing purposes.

## **User Cases**

Use Case 1: User Registration and Authentication

**Actors:** General User, Venue Owner  
**Description:** A user visits the application and chooses to register either via email or through a third-party service (Google, Facebook, Apple). Upon registration, the system sends a verification link. Once verified, the user can log in and access the dashboard according to their assigned role.  
**Functional Requirements Addressed:**

* Email verification
* Role-based authentication
* Secure password handling
* Third-party authentication integration

Use Case 2: Profile Management

**Actors:** Registered User  
**Description:** After logging in, a user accesses the profile section to update personal details such as name, contact info, sport preferences, and view past bookings. The system saves these changes and applies them to future interactions and personalized suggestions.  
**Functional Requirements Addressed:**

* Profile updates
* Booking history display
* Preference customization

Use Case 3: Venue Discovery and Search

**Actors:** General User  
**Description:** A user opens the app and uses the venue search feature to discover available sports venues near their location. The system uses geolocation and advanced filters (e.g., price range, sport type, amenities) to present a curated list.  
**Functional Requirements Addressed:**

* Geolocation-based discovery
* Advanced search filtering
* Real-time venue availability

Use Case 4: Booking a Venue

**Actors:** General User  
**Description:** A user selects a venue, views available time slots via the integrated calendar, and proceeds to book a slot. The system processes the payment securely and sends a booking confirmation.  
**Functional Requirements Addressed:**

* Instant booking
* Real-time calendar integration
* Payment processing
* Booking confirmation

Use Case 5: Group Booking and Participant Management

**Actors:** General User  
**Description:** While booking a venue, the user opts for a group reservation. The system allows the addition of participants, sends notifications to them, and confirms the group reservation upon successful payment.  
**Functional Requirements Addressed:**

* Group booking
* Participant management
* Notification system

Use Case 6: Venue Registration and Management

**Actors:** Venue Owner  
**Description:** A venue owner registers their venue, adds multimedia content, defines available time slots, and sets pricing. The system verifies the venue before making it visible to users.  
**Functional Requirements Addressed:**

* Venue registration
* Availability setup
* Media upload
* Admin approval workflow

Use Case 7: Booking Management (Owner Side)

**Actors:** Venue Owner  
**Description:** A venue owner receives new booking requests and either accepts or declines them based on their schedule. The system updates availability and notifies the user accordingly.  
**Functional Requirements Addressed:**

* Booking approval workflows
* Calendar updates
* User-owner communication

Use Case 8: Messaging and Communication

**Actors:** General User, Venue Owner  
**Description:** Users and venue owners exchange messages regarding booking details or venue-related inquiries. The system maintains message history and sends notifications for unread messages.  
**Functional Requirements Addressed:**

* Direct messaging
* Notification system
* Review moderation

Use Case 9: Payment and Refund

**Actors:** General User, System, Venue Owner  
**Description:** Users pay using their preferred method (credit card, mobile wallet). The system handles transactions securely and calculates revenue shares. In case of cancellations, a refund process is triggered based on authorization rules.  
**Functional Requirements Addressed:**

* Multi-method payments
* Revenue sharing
* Refund and audit trail

Use Case 10: Rating and Reviews

**Actors:** General User  
**Description:** After a booking, the user rates the venue and leaves a review. The system publishes the review after moderation and adds it to the venue’s public profile.  
**Functional Requirements Addressed:**

* Rating system
* Review moderation
* Feedback visibility

Here are the following Figures and Diagrams Concerning the Use Cases:

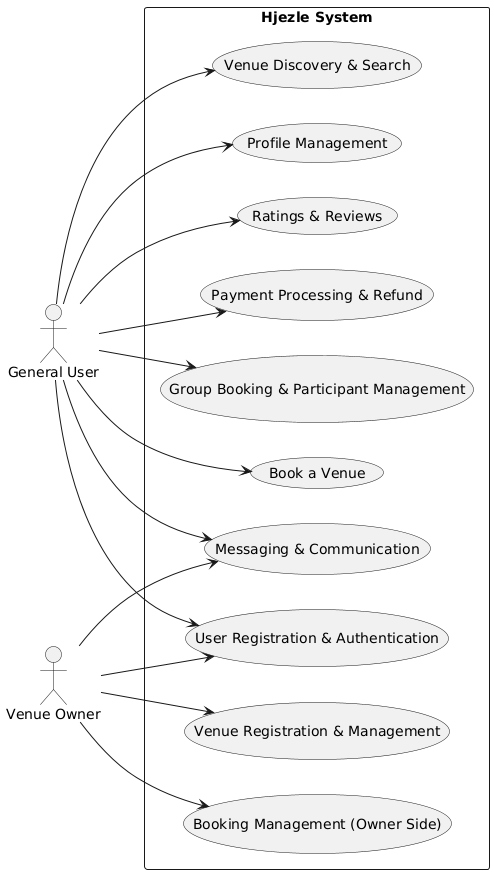


Figure : UML Diagram

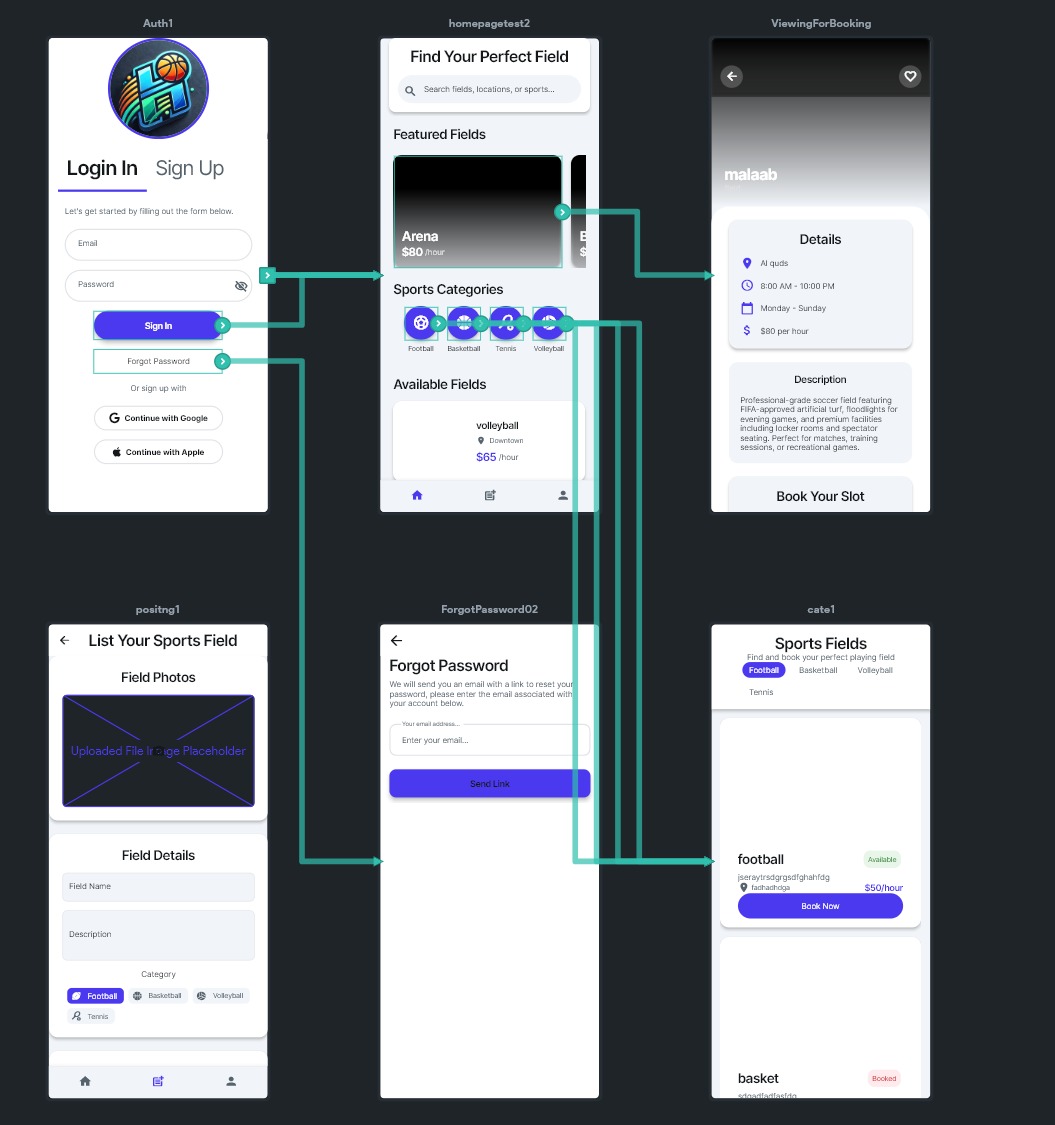


Figure : Flow Diagram like the UML showing how each button works and its navigations

# **System Modeling**

## **Introduction**

System modeling is a fundamental practice in software engineering that provides abstract representations of a system’s structure, behavior, and interactions. It helps stakeholders visualize and understand how the system operates, what components are involved, and how they communicate. By using modeling techniques, developers can validate requirements, identify potential issues early, and ensure alignment between user needs and system functionality. For the **Hjezle** application, system modeling played a crucial role in bridging the gap between conceptual design and implementation, especially given its real-time nature and multi-role architecture.

## **Context modeling**

Context models define the **scope and external boundaries** of a system. They illustrate the interaction between the system and external entities such as users, third-party services, and external databases. In the case of **Hjezle**, the context model highlights the relationships between the system and actors including **general users**, **venue owners**, **payment gateways**, **Google Maps API**, and **Firebase services**.

This model helps identify the **external data flow** and clarifies what lies inside the system's control and what exists externally. It supports better understanding of integration points and ensures that all stakeholder interactions are accurately represented and planned for.

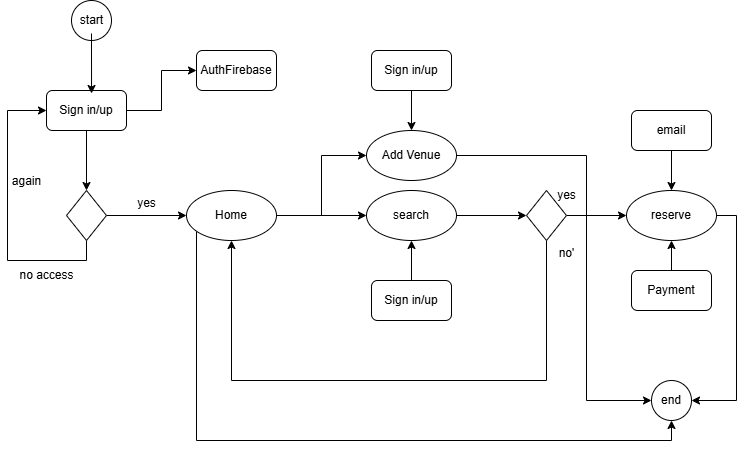


Figure : Context Model

A black background with white rectangles

AI-generated content may be incorrect.

Figure : Process Model

## **Interaction Model**

Interaction modeling is essential for understanding how users and system components communicate, and it plays a crucial role in identifying user requirements, validating system behavior, and resolving communication-related issues. It focuses on three main types of interactions: **user-to-system**, **system-to-system**, and **component-to-component**. Two key UML diagram types are commonly used for this purpose: **use case diagrams** and **sequence diagrams**.  
Use case diagrams help define the functional scope of the system by illustrating discrete tasks performed by actors—whether human users or external systems—that interact with the system. Sequence diagrams complement use cases by detailing the step-by-step flow of interactions between actors and system objects during specific use case scenarios. These diagrams are read top-to-bottom and depict the chronological sequence of messages exchanged, helping stakeholders understand the dynamic behavior of the system. The following diagrams illustrate these interaction models in the context of the **Hjezle** application.

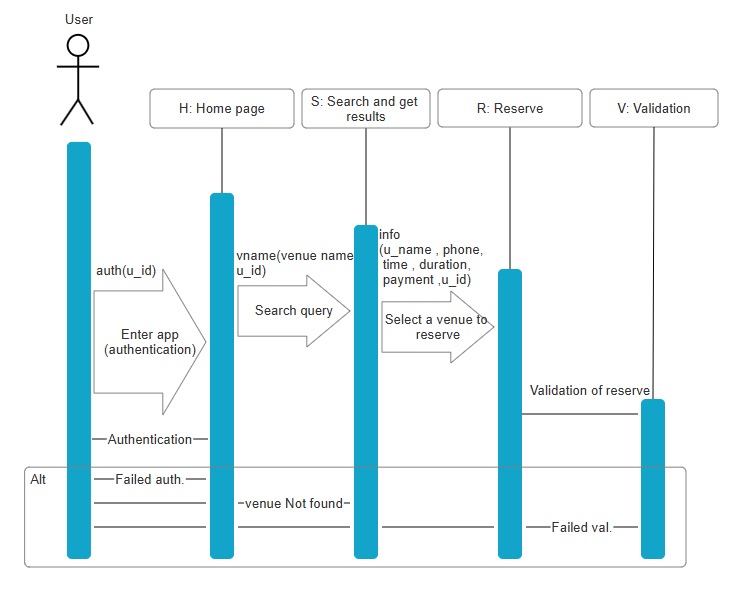


Figure : Interaction Model that shows the sequence for user booking

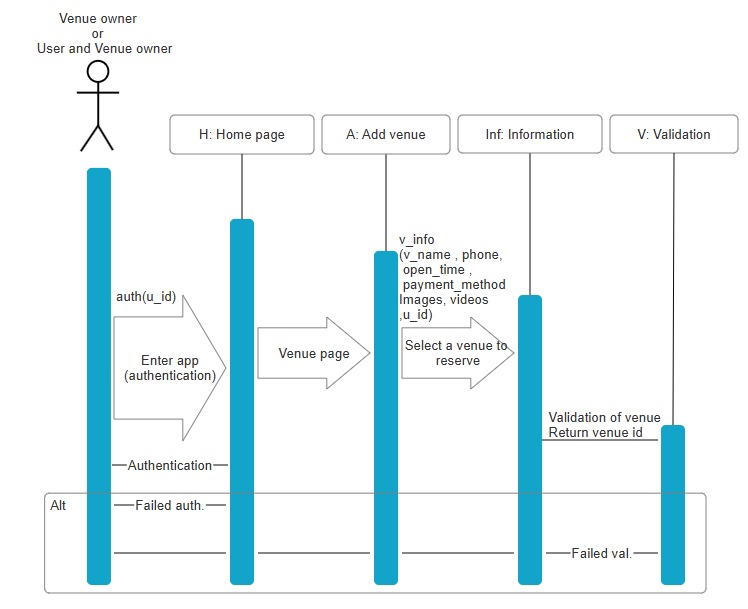


Figure : Interaction Model that shows sequence for venue owners

## **Structural Model**

Structural modeling focuses on the static organization of a system, illustrating the components that make up the system and the relationships among them. These models are typically created during the design phase to define and document the system architecture. In object-oriented systems, **UML class diagrams** are the primary tool used for structural modeling. They represent the system's classes, their attributes and operations, and the associations or links between them, which reflect real-world relationships.

Each class in the diagram corresponds to a real-world concept—such as a user, venue, or booking—and associations represent the interactions or dependencies between them. **Generalization** is used to manage complexity by grouping classes into hierarchical structures, allowing lower-level subclasses to inherit attributes and behaviors from higher-level super classes. This enables code reuse and simplifies maintenance by centralizing shared logic.

Additionally, **aggregation models** are used to express "whole part" relationships, where a class (e.g., a venue) is composed of other classes (e.g., schedules, amenities). Aggregation provides clarity on how different components are structured and how they contribute to larger entities. By modeling these structural elements early in the design process, the development team can ensure that the system’s architecture is scalable, maintainable, and aligned with the domain it represents.

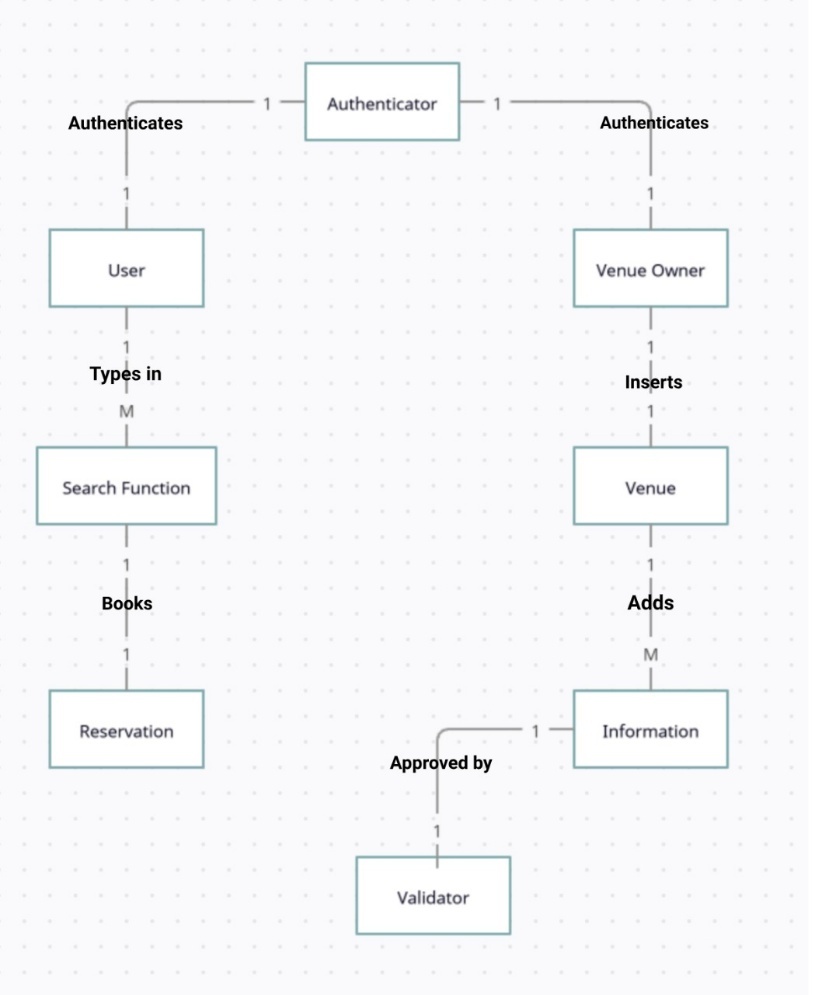


Figure :Structural Model

## **Behavioural Model**

The following state transition diagram professionally illustrates the behavioral model of a booking and venue management system. It clearly defines the discrete states within the system lifecycle and the explicit transitions triggered by specific events or user interactions. The model encompasses key functionalities such as user authentication, venue discovery and registration, reservation specification, confirmation processes, and exception handling mechanisms denoted by warning states. This visual representation provides a structured understanding of the system's dynamic behavior and its responses to various operational scenarios.

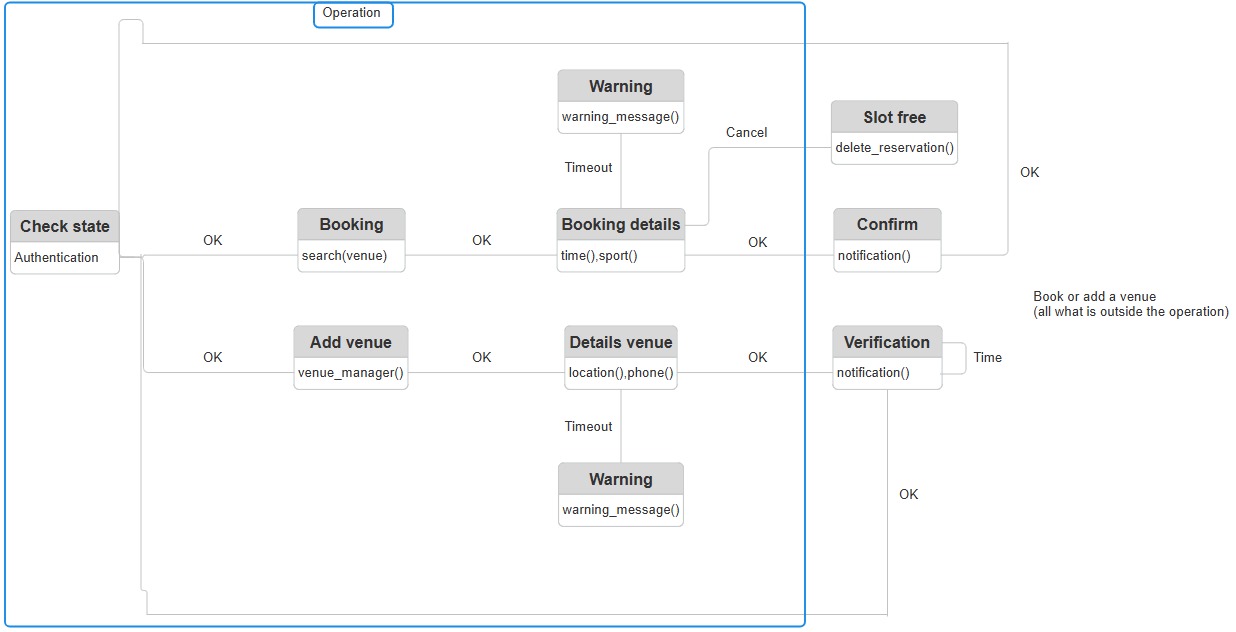


Figure : Behavioural Model

## **Model-Driven Engineering**

Model-Driven Engineering (MDE) is a modern approach to software development where models, rather than traditional programs, are the main outputs of the development process. In this methodology, the programs that execute on a hardware or software platform are automatically generated from the models created during the development stages. MDE aims to raise the level of abstraction in software engineering, allowing developers to focus on system design and functionality instead of being burdened by the specifics of programming languages or execution platforms.

Although MDE is a promising and innovative concept, it remains at an early stage of adoption, and it is still unclear whether it will significantly impact mainstream software engineering practices in the long term. Proponents argue that MDE enables engineers to work at higher levels of abstraction, making it easier to visualize, design, and modify complex systems. Additionally, by generating code automatically from models, it becomes cheaper and faster to adapt software systems to new hardware and software platforms.

However, MDE also presents certain challenges. The models created for abstraction purposes are not always perfectly suited for direct implementation, which may introduce inefficiencies or design issues. Furthermore, while automatic code generation reduces manual development costs, these savings may be outweighed by the high costs involved in building and maintaining translators that can generate code for multiple and evolving platforms.

In the development of Hjezle, we embraced the principles of Model-Driven Engineering by using FlutterFlow to design our app visually and generate high-quality code. This approach allowed us to accelerate development, reduce errors, and focus more on enhancing the user experience. By applying MDE concepts, we were able to build a flexible, reliable, and scalable architecture for Hjezle, setting a strong foundation for future growth and feature expansion.

# **Software Testing**

Throughout the development of **Hjezle**, rigorous testing and verification processes were applied to ensure that the application not only met its functional requirements but also delivered reliable and satisfying user experience. The theoretical principles of validation, defect testing, verification, and various testing levels were customized to fit the unique needs of our project.

## **Validation Testing**

Validation testing was carried out to demonstrate that Hjezle’s core functionalities operated according to the defined requirements. Specific test cases reflected real-world user journeys, such as logging into the application, searching for nearby venues, booking a time slot, making a secure online payment, and receiving a booking confirmation. Additional validation focused on ensuring that group bookings were processed correctly, allowing users to invite multiple participants and manage team reservations effortlessly. The successful completion of these tests provided evidence to the development team and project stakeholders that the application met its intended functional specifications and delivered a seamless user experience.

## **Defect Testing**

Defect testing played a critical role in uncovering faults that might arise under abnormal or less predictable circumstances. We deliberately designed test cases to simulate edge conditions, such as two users attempting to book the same venue simultaneously. In these scenarios, the system correctly handled the conflict by accepting only the first transaction and notifying the second user appropriately. Additional defect tests included simulating invalid payment processes to verify that the application could gracefully handle failed transactions without crashing, and testing under very slow network conditions to assess the application's stability during delays. This approach allowed us to expose weaknesses early and strengthen Hjezle’s resilience.

## **Verification Activities**

Verification activities were systematically conducted to ensure that the implementation remained consistent with our architectural and functional design specifications. Static verification was performed through regular software inspections, where FlutterFlow page structures, backend Firestore database rules, and security constraints were reviewed. Moreover, during the structural modeling phase, the relationships between Users, Venues, and Bookings were thoroughly checked against UML class diagrams to validate the system's logical design. These verification efforts were essential in maintaining traceability between design artifacts and the developed system, ensuring that "we were building the product right."

## **Deployment Testing**

Development testing encompassed multiple levels of testing performed during system construction. Unit testing focused on verifying the correctness of individual Flutter classes, such as screens for login, venue listing, and payment integration. Each function and method was independently tested to ensure expected behavior. Component testing then followed, where integrated modules such as the booking flow (venue selection, time slot management, and payment gateway interaction) were tested collectively to validate their interoperability. System testing concluded the internal testing phase by executing complete end-to-end workflows, covering login, browsing, booking, payment, notification, and cancellation scenarios, ensuring that all system components functioned correctly when combined.

## **Release Testing**

Prior to public release, a dedicated round of release testing was conducted. This phase involved simulating realistic operational environments to validate that the near-final application could handle live usage scenarios. Special focus was placed on performance testing, where system response times were measured under increasing loads. Tests were conducted by simulating hundreds of users searching for venues and making concurrent bookings, confirming that the application maintained acceptable responsiveness under heavy load. Additionally, stress testing was performed by artificially overloading the system with excessive bookings and network activity, allowing the team to observe failure behavior and verify that the system degraded gracefully without data loss or corruption.

## **User Testing**

In the final phase, user testing was organized to obtain practical feedback from individuals representing the app’s target audience. Users were tasked with booking venues, managing team bookings, and performing cancellations or modifications within their personal environments. Their feedback highlighted areas for minor improvements, particularly in notification clarity and error handling in payment flows. By incorporating their input, the system was further refined to ensure intuitive usability and satisfaction for real-world customers.

# **Code, Frontend & Backend**

After we finished modeling, designing, decision making and testing, we can finally say that our codes were finished. Here are some samples of our codes for our pages.



Figure 9: Authentication Page Sample Code

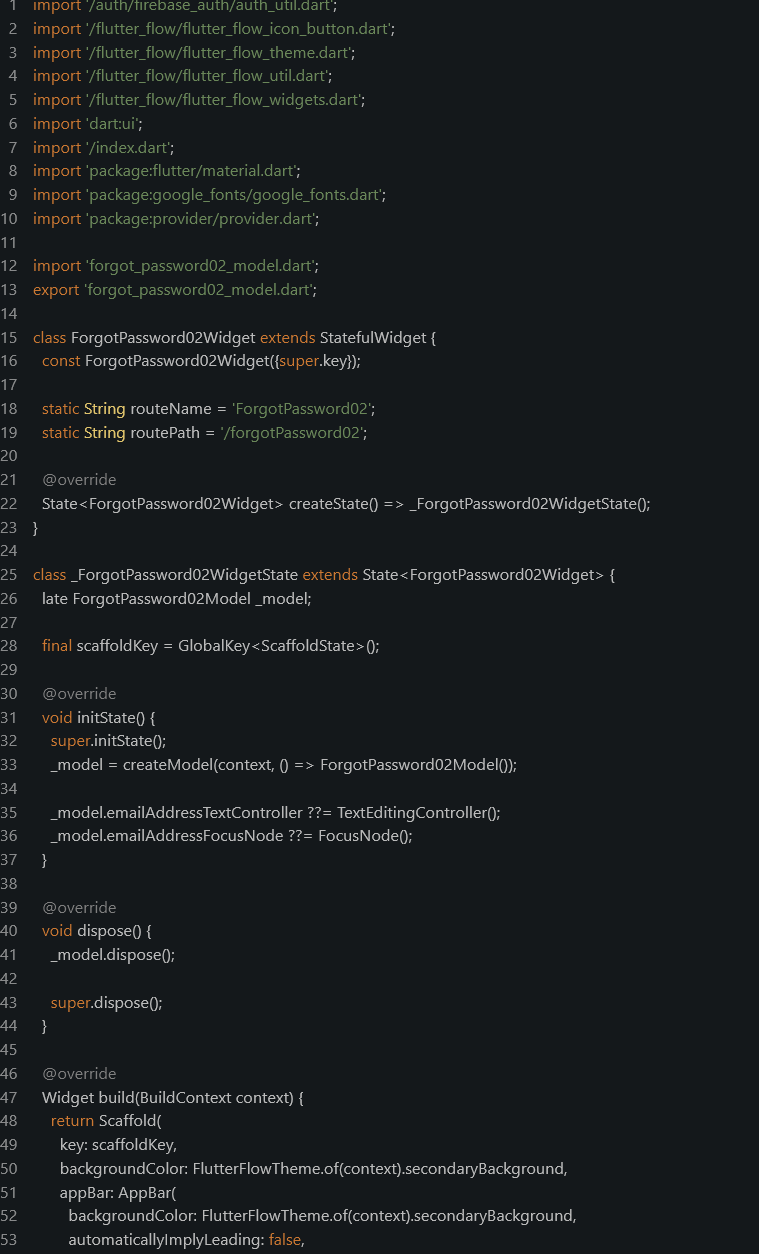


Figure 10: Forget Password Page Sample Code



Figure 11: Home Page Sample Code

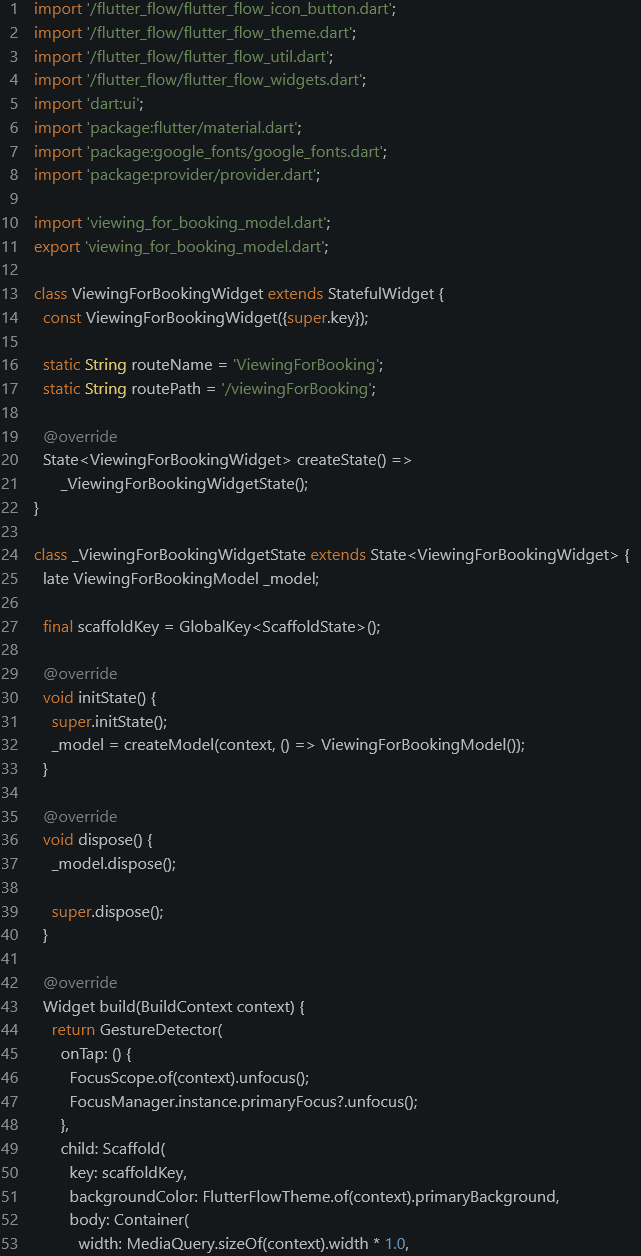


Figure : Booking Page Sample Code

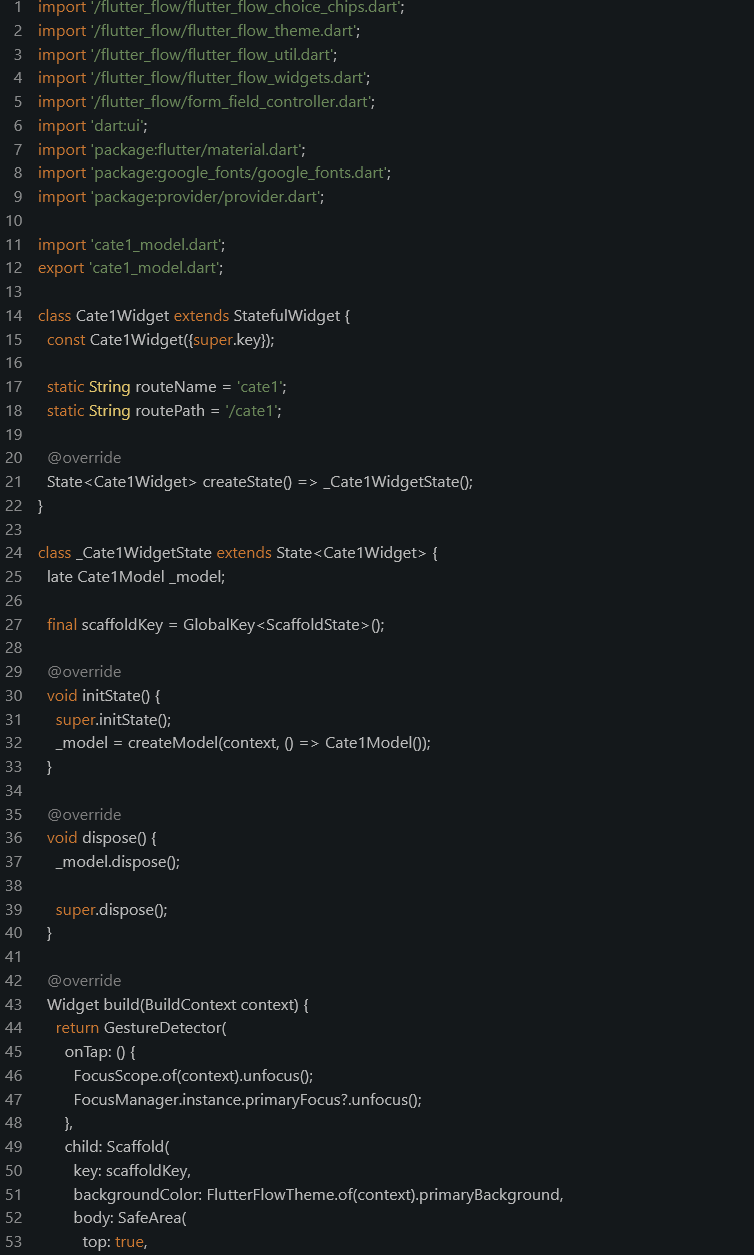


Figure : Category Page Sample Code

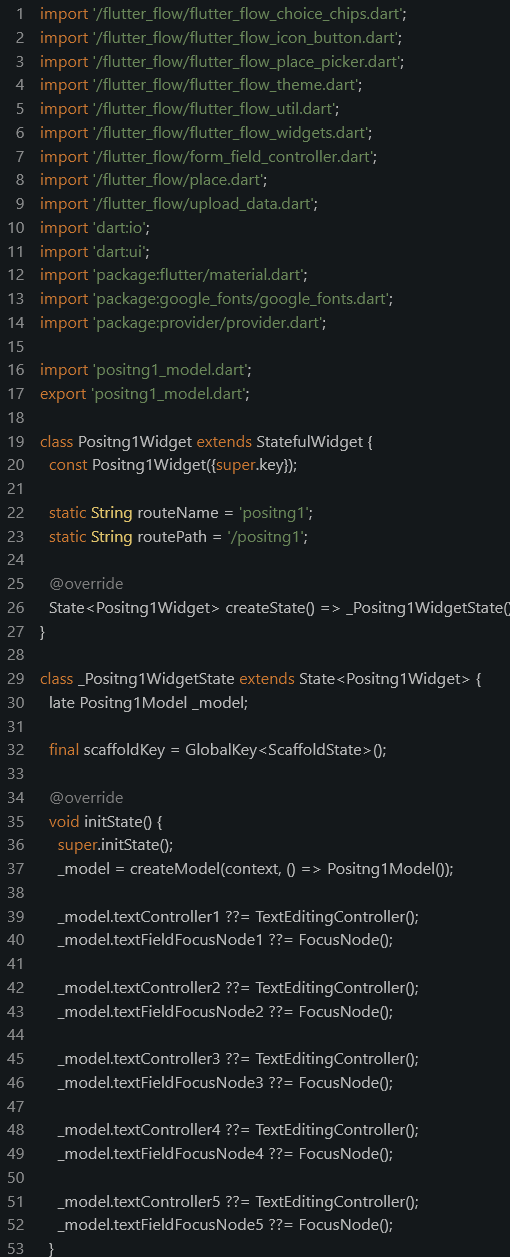


Figure 14: Posting Page Sample Code

Below is a comprehensive overview of the process used to implement the integration between our backend database and frontend application. The figures provided visually represent the steps involved in the integration process and are designed to be self-explanatory.

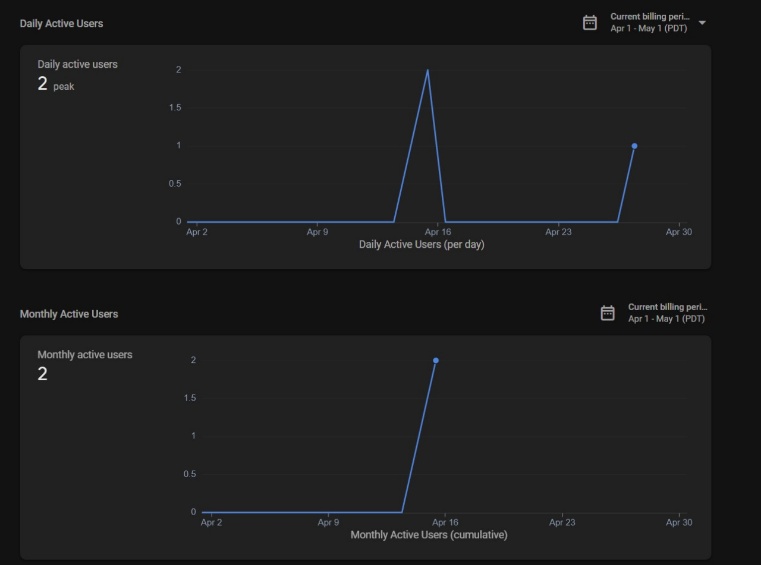


Figure : Shows Number of Users

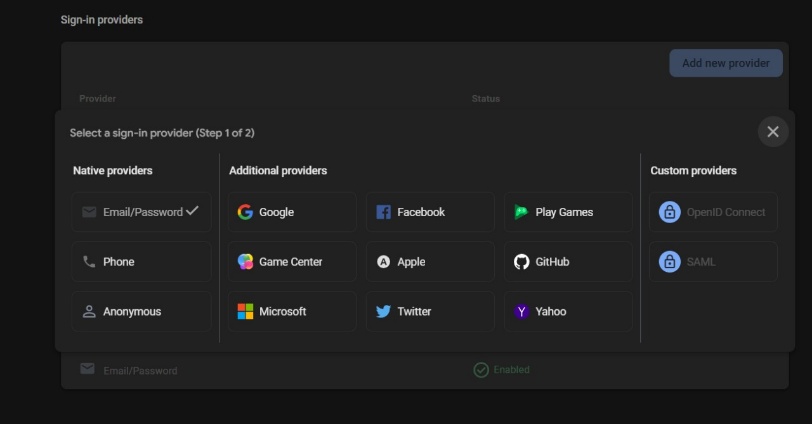


Figure : Shows Tools to be used for authentication

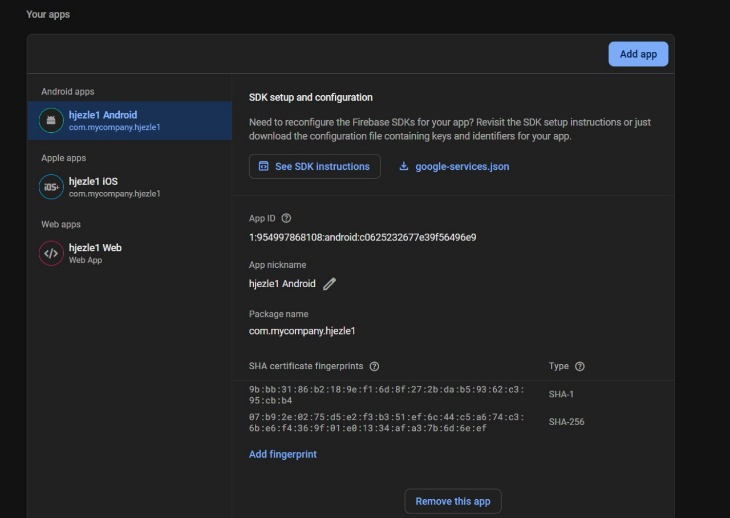


Figure : Shows what devices the authentication works on

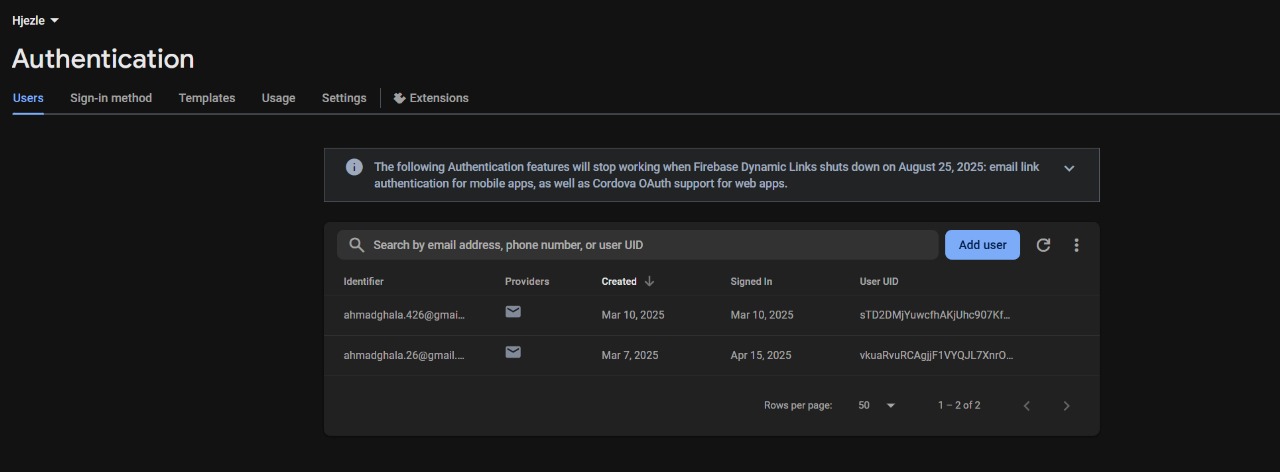


Figure : Shows who are logged in

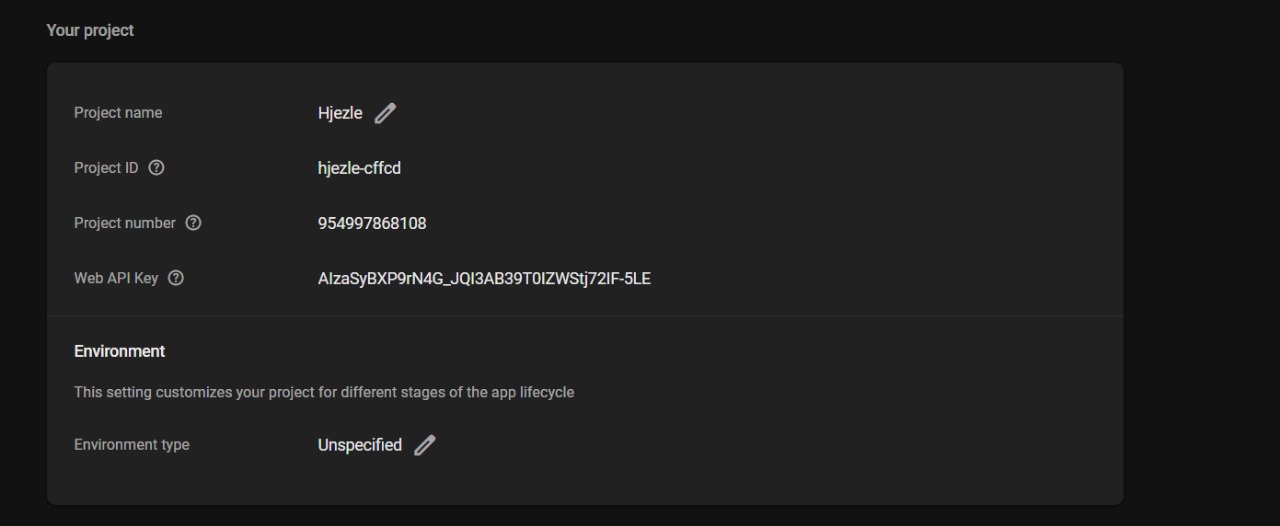


Figure : Shows firebase ID for HJEZLE

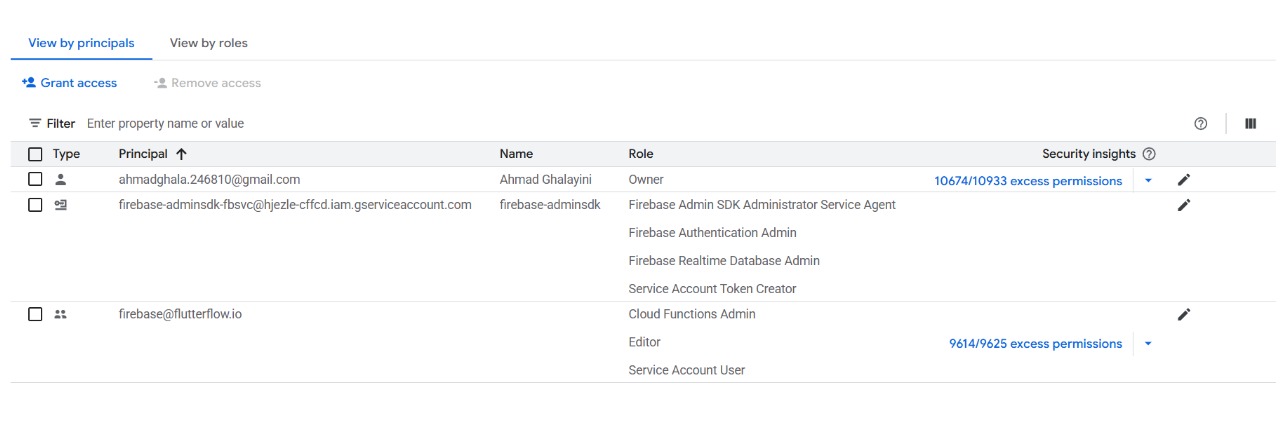


Figure : Google Account Permission

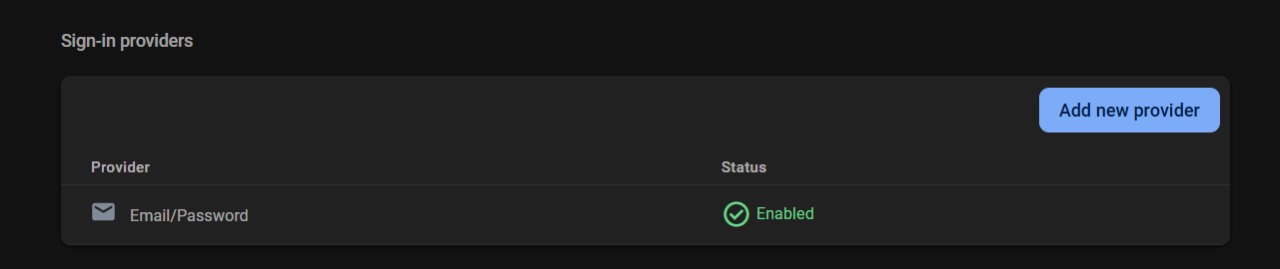


Figure : Shows method of logging in

A screenshot of a computer

AI-generated content may be incorrect.

Figure : Communication Page with the user used for confirmation

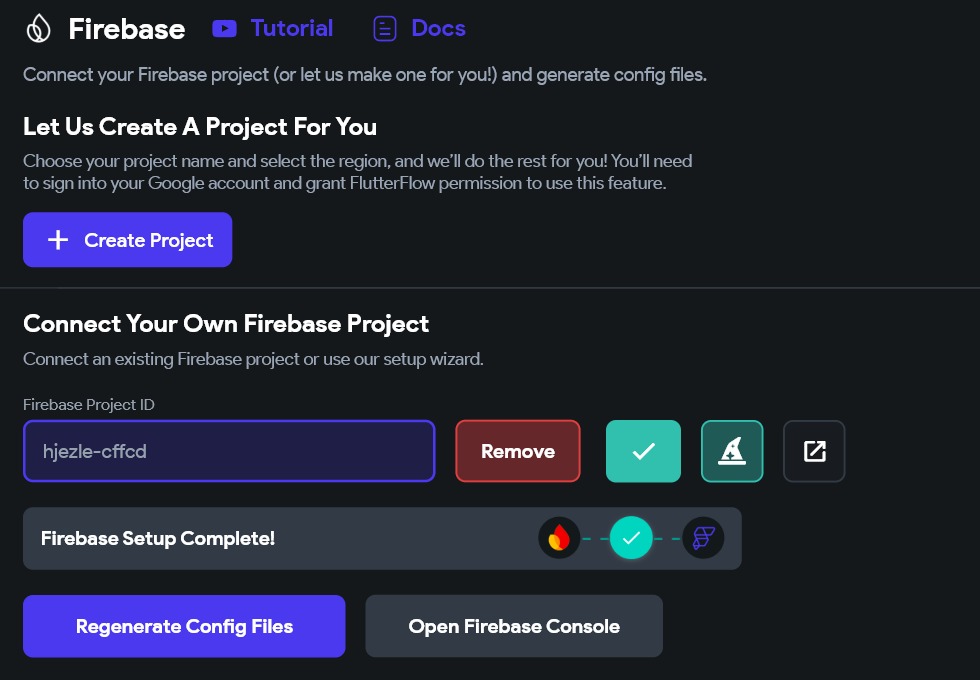


Figure : Shows whether connection is applied between flutterflow and firebase

Having established the Firebase integration for authentication and real-time data synchronization, we now turn our attention to the storage solution implemented using Supabase. This complementary backend service handles media storage requirements, particularly for venue images and user profile pictures that require secure and efficient hosting.

The Supabase implementation provides HJEZLE with robust object storage capabilities, allowing venue owners to upload high-quality images of their facilities that can be quickly retrieved by users browsing the application. The integration process involved establishing secure API connections between our Flutter frontend and the Supabase backend, implementing proper access controls, and optimizing image processing for mobile delivery.

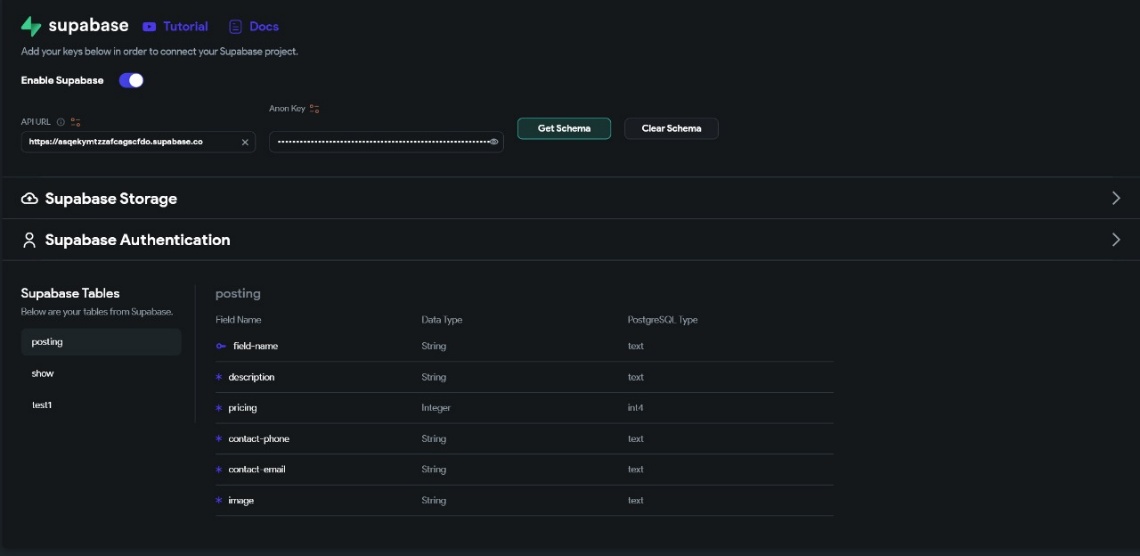


Figure : Supabase inside of FlutterFlow where tables were imported

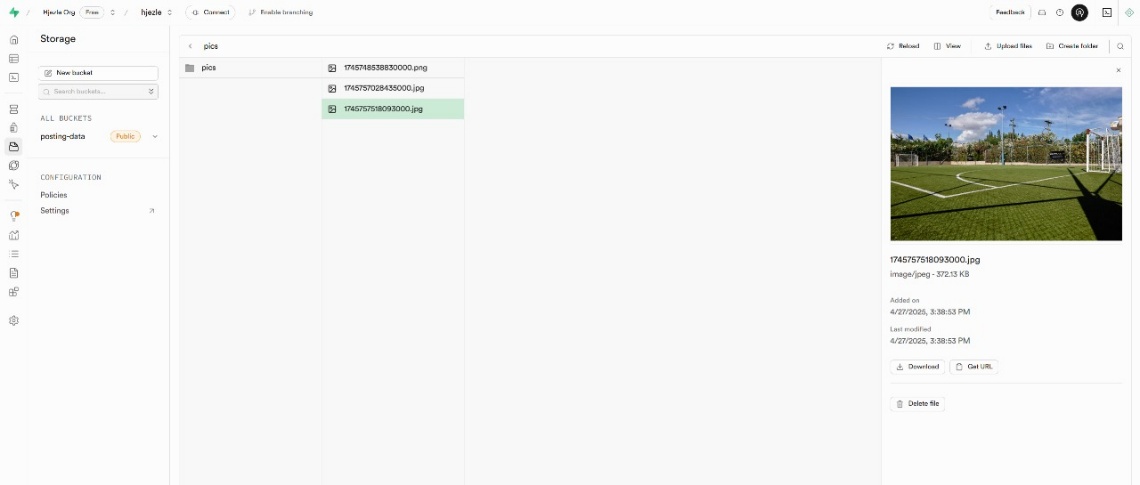


Figure : After uploading an image and add a new venue, it is downloaded on Supabase Storage

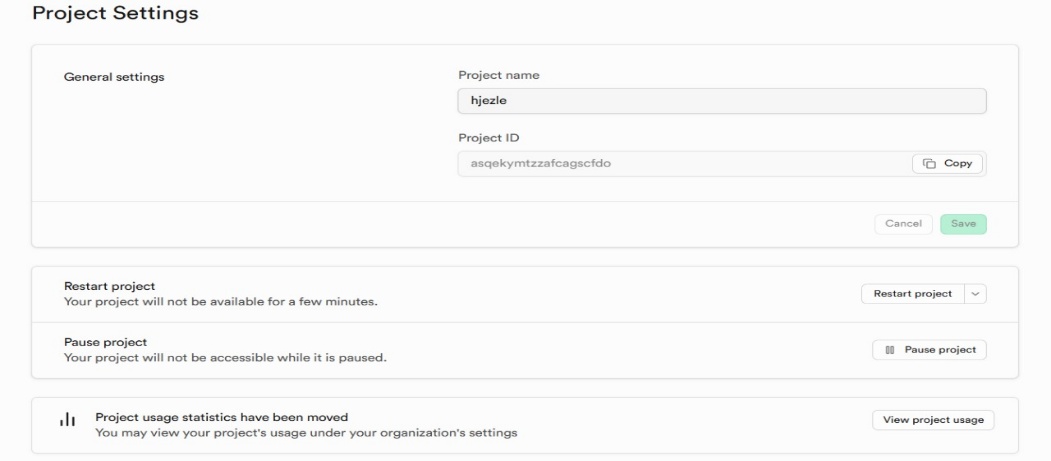


Figure : Supabase ID for FlutterFlow Project Integration

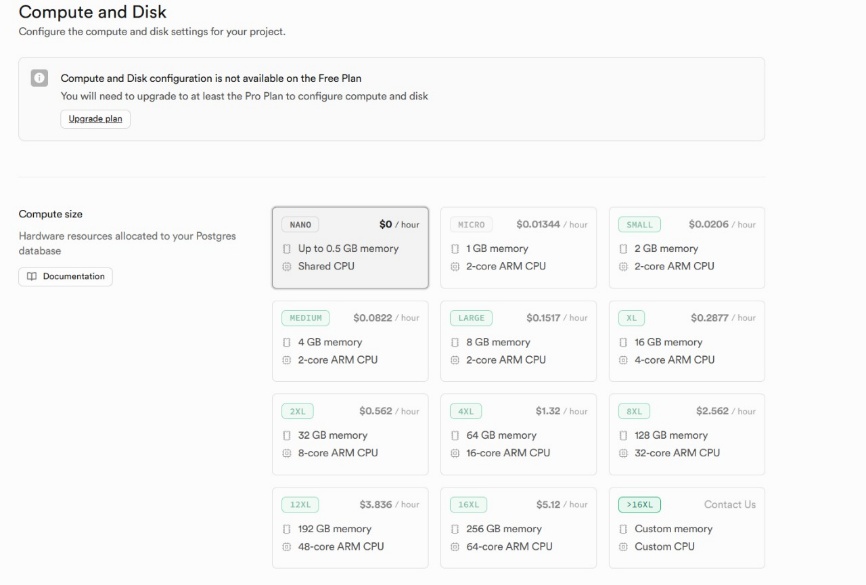


Figure :This is were SupaBase storage is taken

A screenshot of a computer

AI-generated content may be incorrect.

Figure : The table given by SupaBase

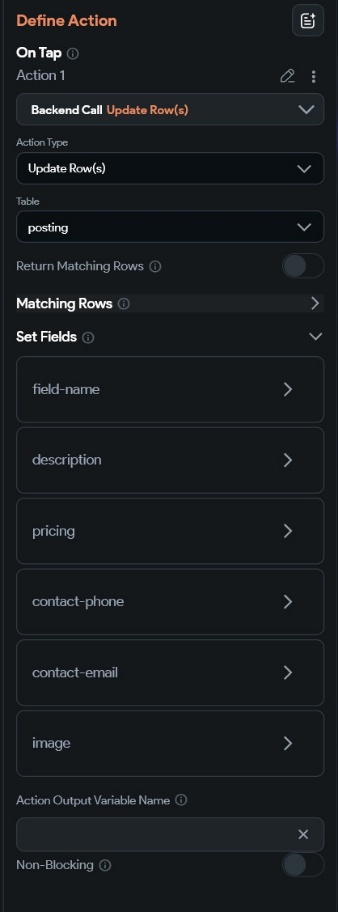


Figure : Button that saves data inside of SupaBase (FlutterFlow)

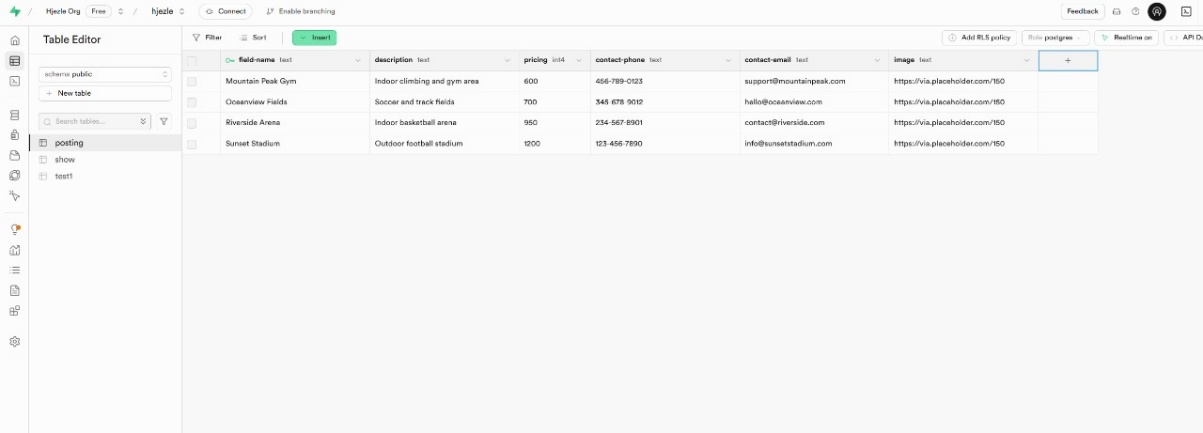
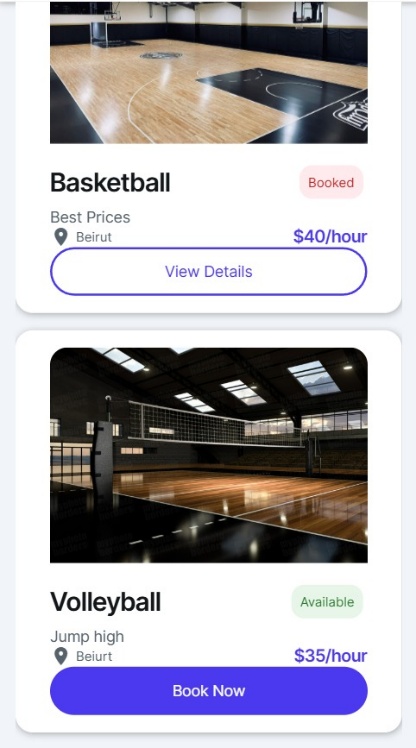
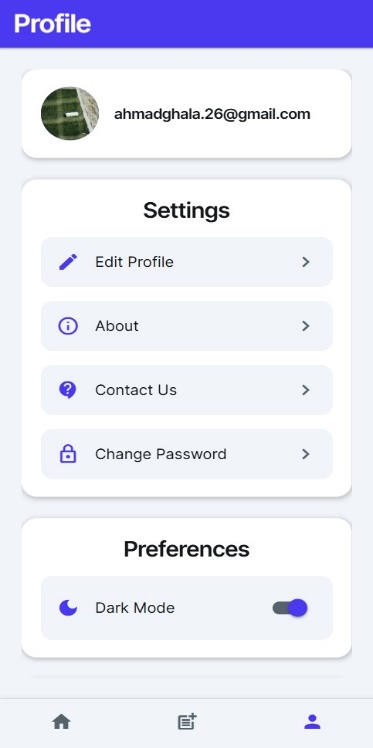
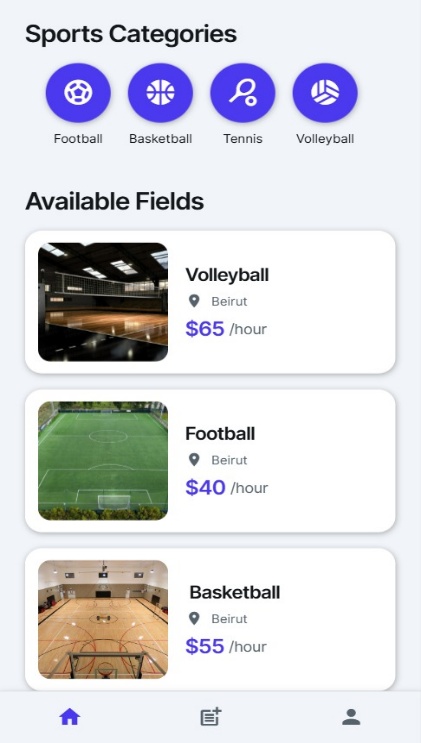
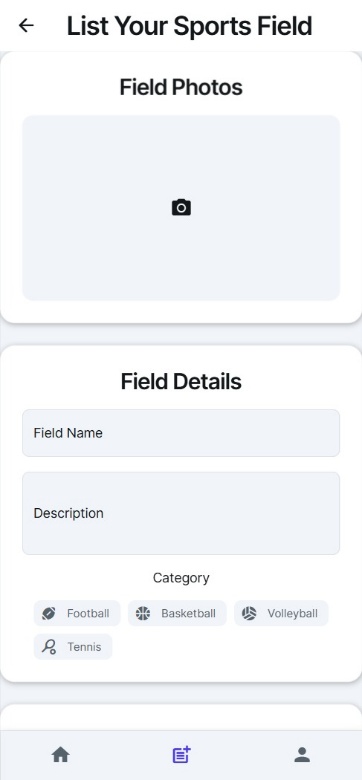
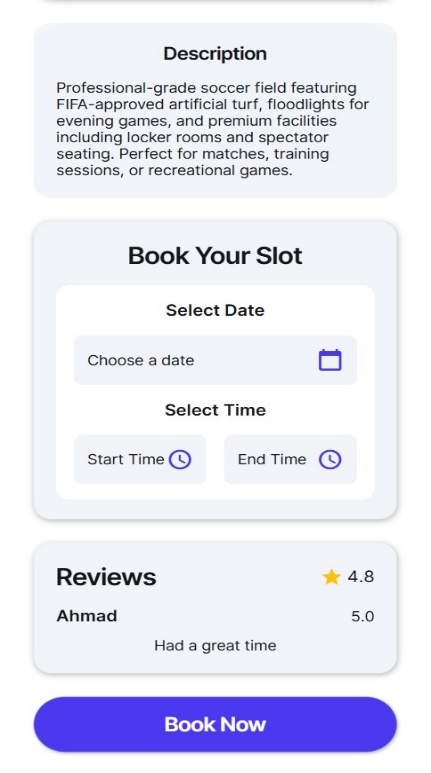
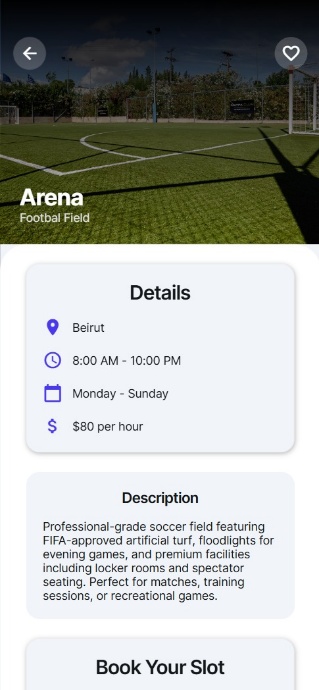
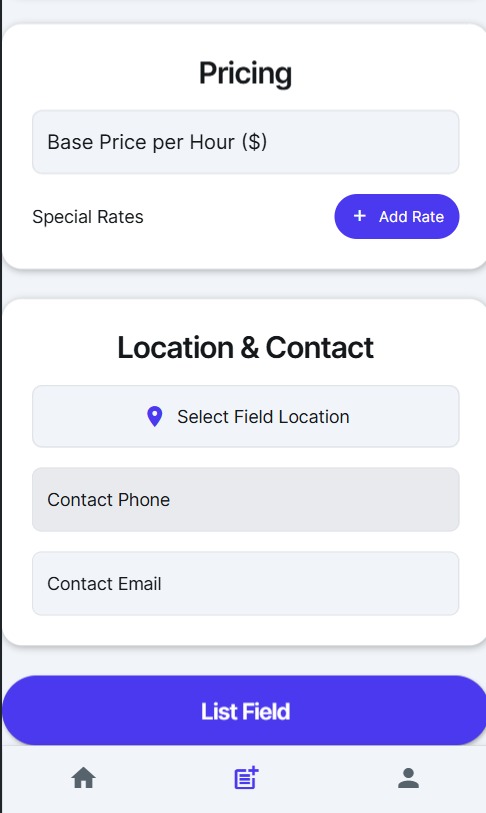
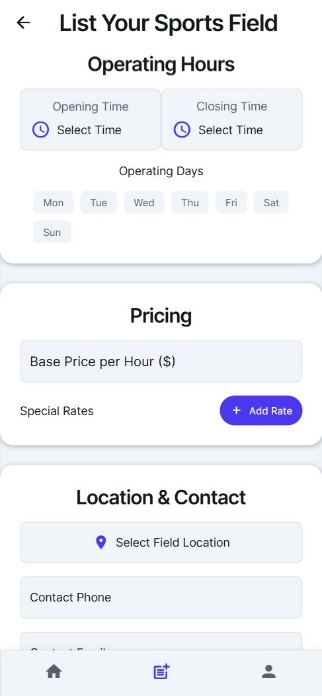
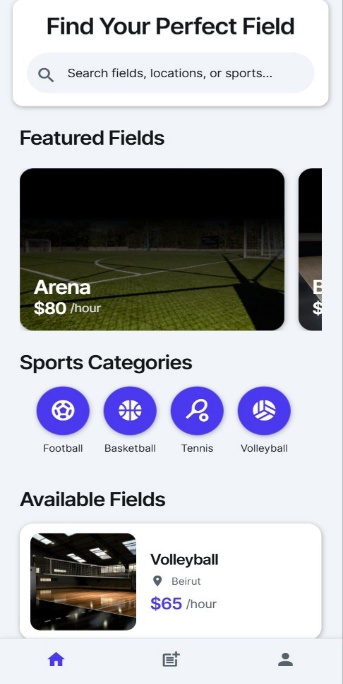
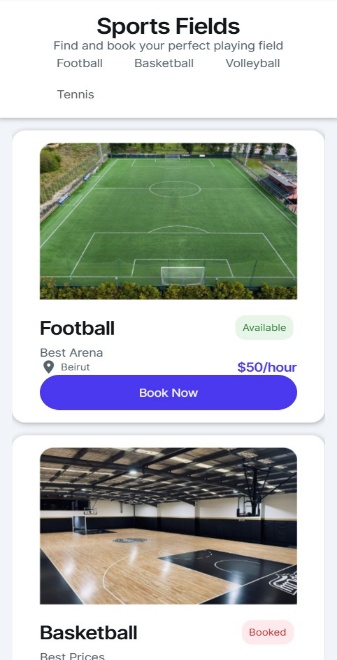
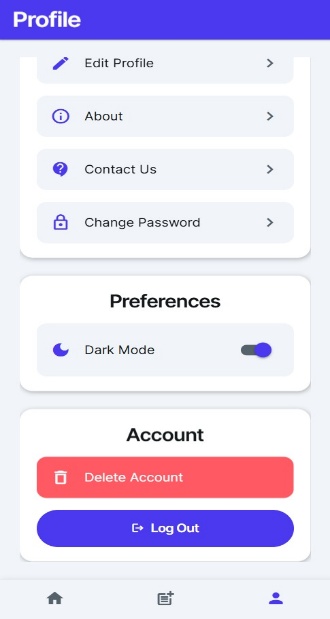
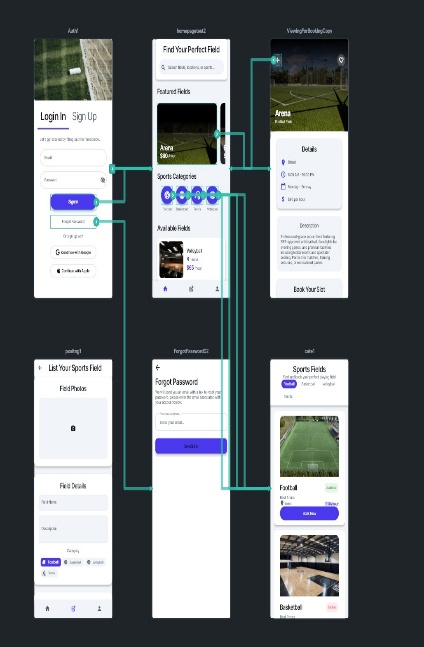


Figure : This table is created after uploading a new venue

Having outlined the backend integration, we now shift focus to the frontend aspect of the application. The frontend serves as the interface through which users interact with the system, and it has been designed to ensure a seamless and intuitive experience. Below, we will walk through the key components of the frontend, showcasing the user interface and how it communicates with the backend to deliver a cohesive user experience.

 A screenshot of a phone

AI-generated content may be incorrect. 

# **Conclusion**

The development of HJEZLE represents a significant advancement in sports facility booking technology within Lebanon's recreational landscape. By addressing the critical gaps in existing booking processes, this application effectively bridges the disconnect between venue owners and sports enthusiasts through a comprehensive, user-friendly platform. The project successfully integrates sophisticated technical capabilities with intuitive user experience design to create a seamless booking ecosystem.

Throughout the development lifecycle, our team maintained rigorous adherence to software engineering principles, from initial requirements gathering through to final testing and deployment. The modular architecture of HJEZLE ensures scalability to accommodate future growth in user base and feature expansion. The implementation of Firebase as the backend infrastructure provides real-time synchronization capabilities essential for a reservation system, while Flutter delivers a consistent cross-platform experience for end users.

The extensive testing regimen applied to HJEZLE—including validation, defect, verification, deployment, release, and user testing—has confirmed the application's reliability and performance under various operational conditions. This thorough approach has yielded a robust platform capable of handling concurrent bookings, secure payments, and efficient communication between stakeholders.

As HJEZLE moves from development into the marketplace, it stands poised to transform the sports venue reservation experience in Lebanon. By eliminating inefficiencies in traditional booking methods and introducing transparency in facility availability and pricing, the application not only streamlines operations for venue owners but also encourages greater community participation in sports activities. The social and economic impact of this digital solution extends beyond convenience, potentially contributing to increased utilization of sports facilities and promotion of healthier lifestyles.

In conclusion, HJEZLE demonstrates how thoughtful software engineering practices, when applied to real-world challenges, can produce technological solutions with meaningful societal benefits. The project's success serves as a testament to the power of innovative digital platforms in addressing longstanding inefficiencies in traditional service sectors.

# **References:**

1. Abdul Rahman, et al. "Optimizing Sports Facility Booking Systems in the Middle East." 2022.
2. Global Innovations Inc. "Sporty App Review." 2021.
3. TechCrunch. "PlayFinder App Analysis." 2020.
4. Firebase. "Firebase Documentation." Google, <https://firebase.google.com/docs>.
5. OpenPlay. "OpenPlay: Book Your Sports Venue." OpenPlay, n.d. <https://www.openplay.co.uk>.
6. BookMySports. "BookMySports – Real-Time Booking for Sports Venues." BookMySports, n.d. <https://www.bookmysports.com>.
7. Project Management Institute. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide).* 6th ed., Project Management Institute, 2017.
8. Sommerville, Ian. *Software Engineering.* 10th ed., Pearson, 20
9. Atkinson, Roger. "Project Management: Cost, Time, and Quality, Two Best Guesses and a Phenomenon." *International Journal of Project Management*, vol. 17, no. 6, 1999, pp. 337-342.
10. ISO. *ISO 21500:2012 – Guidance on Project Management.* International Organization for Standardization, 2012.
11. McLeod, Sam. "Interrelated Attributes of Project Feasibility: Visualizing the TELOS Framework." *ScienceOpen Posters*, 29 June 2021.
12. Bentley, Lonnie, and Jeffrey Whitten. *System Analysis & Design for the Global Enterprise.* 7th ed., McGraw-Hill, 2007.
13. IEEE Computer Society. *Software Engineering Code of Ethics and Professional Practice.* IEEE, 1999, <https://ethics.acm.org/code-of-ethics/software-engineering-code/>.
14. OWASP Foundation. *Application Security Verification Standard (ASVS).* v4.0.3, 2021, <https://owasp.org/www-project-application-security-verification-standard/>.
15. Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed., Addison-Wesley, 2003.
16. Larman, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. 3rd ed., Prentice Hall, 2004.
17. Booch, Grady, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. 2nd ed., Addison-Wesley, 2005.
18. **PlantUML.** "Hjezle Use Case Diagram." *PlantUML*, n.d. Web. Retrieved, from <https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa700001>
19. Beck, Kent. *Extreme Programming Explained: Embrace Change*. 2nd ed., Addison-Wesley Professional, 2004.
20. Cockburn, Alistair. *Agile Software Development: The Cooperative Game*. 2nd ed., Addison-Wesley Professional, 2006.
21. Nielsen, Jakob. "10 Usability Heuristics for User Interface Design." *Nielsen Norman Group*, 24 Apr. 1994, <https://www.nngroup.com/articles/ten-usability-heuristics/>.
22. Schwaber, Ken, and Jeff Sutherland. *The Scrum Guide*. Scrum.org, 2020, <https://scrumguides.org/scrum-guide.html>.
23. Wiegers, Karl E., and Joy Beatty. *Software Requirements*. 3rd ed., Microsoft Press, 2013.
24. Robertson, Suzanne, and James Robertson. *Mastering the Requirements Process: Getting Requirements Right*. 3rd ed., Addison-Wesley Professional, 2012.
25. Bass, Len, et al. *Software Architecture in Practice*. 3rd ed., Addison-Wesley Professional, 2012.