# Software Requirements Specification

**for**

# <HobbyHive>

**Version 1.0 approved**

**Prepared by < Palagiri Afreen Mahtaj, Teo Rong Xuan, Jain Divisha, Chua Jing Yi Jax, Ishita Dhananjaya, Swetha Sudhakar>**

**<SC2006 - SCSC - Team HobbyHive>**

**<02/11/2025>**

# Table of Contents

# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| | | | |
| | | | |

# 1. Introduction

## 1.1 Purpose

This document outlines the Software Requirements Specification (SRS) for “HobbyHive, Version 1.0”. HobbyHive is a digital platform designed to help users discover, organize and participate in hobby-based events and activities. This document specifies the system’s intended functionalities, features and constraints to provide a comprehensive framework for developers, testers and stakeholders.

The SRS serves as a blueprint for the development cycle - from planning and design to implementation, testing and maintenance - ensuring all stakeholders have a unified understanding of the project’s goals and technical requirements.

## 1.2 Document Conventions

This SRS follows standardized documentation conventions to maintain clarity, readability and consistency throughout.

- **Font Styles:** *Arial* for body text, *Times New Roman* for headings.
- **Highlighting: Bold** for key terms, *italics* for emphasis.
- **Requirement Prioritization:** Each requirement is assigned a priority level, **High**, **Medium**, or **Low**, based on its criticality to the system’s success.
- **Requirement Identification:** All requirements are uniquely numbered (e.g., FR-1, NFR-2) for easy cross-referencing.
- **Versioning:** Requirement changes across versions are documented and tracked for traceability and future updates.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for multiple stakeholders involved in the HobbyHive project. Each audience group can focus on sections most relevant to their responsibilities.

- **Project Managers and Stakeholders:** Start with the *Introduction* and *Overall Description* sections for an understanding of the system’s purpose, scope, and objectives.
- **Developers:** Focus on the *System Features* and *External Interface Requirements* to understand functional logic, data flow, and integration details.
- **Testers:** Review *System Features* for test case creation and *Nonfunctional Requirements* for validation of performance, reliability, and security.

- **User Experience Designers:** Refer to *External Interface Requirements*, particularly UI and usability aspects, for designing intuitive user interfaces.
- **Technical Writers:** Consult the *User Documentation* and *Assumptions and Dependencies* sections to ensure accurate guides and help resources.
- **Quality Assurance Teams:** Examine *System Features* and *Other Nonfunctional Requirements* for performance, usability, and compliance testing.

All readers are encouraged to review the document holistically, as sections build upon one another to form a comprehensive view of the HobbyHive system.

## 1.4 Product Scope

HobbyHive is a social event coordination platform that connects individuals with shared hobbies and interests.
It enables users to:

- Create, discover, and join hobby-based events such as sports, music, art, and fitness sessions.
- Manage event logistics including time, location, and capacity.
- Utilize an integrated map picker to locate venues and identify nearby MRT stations for convenience.
- Facilitate community engagement by allowing users to edit or cancel events, ensuring reliability and coordination.

The system's broader objective is to foster community connections, encourage skill - sharing and support active lifestyles through shared passions - enhancing both social engagement and personal growth.

## 1.5 References

**OOP Design Principles:**
https://khalilstemmler.com/articles/object-oriented/programming/4-principles/

**Supabase Documentation:** https://supabase.com/docs

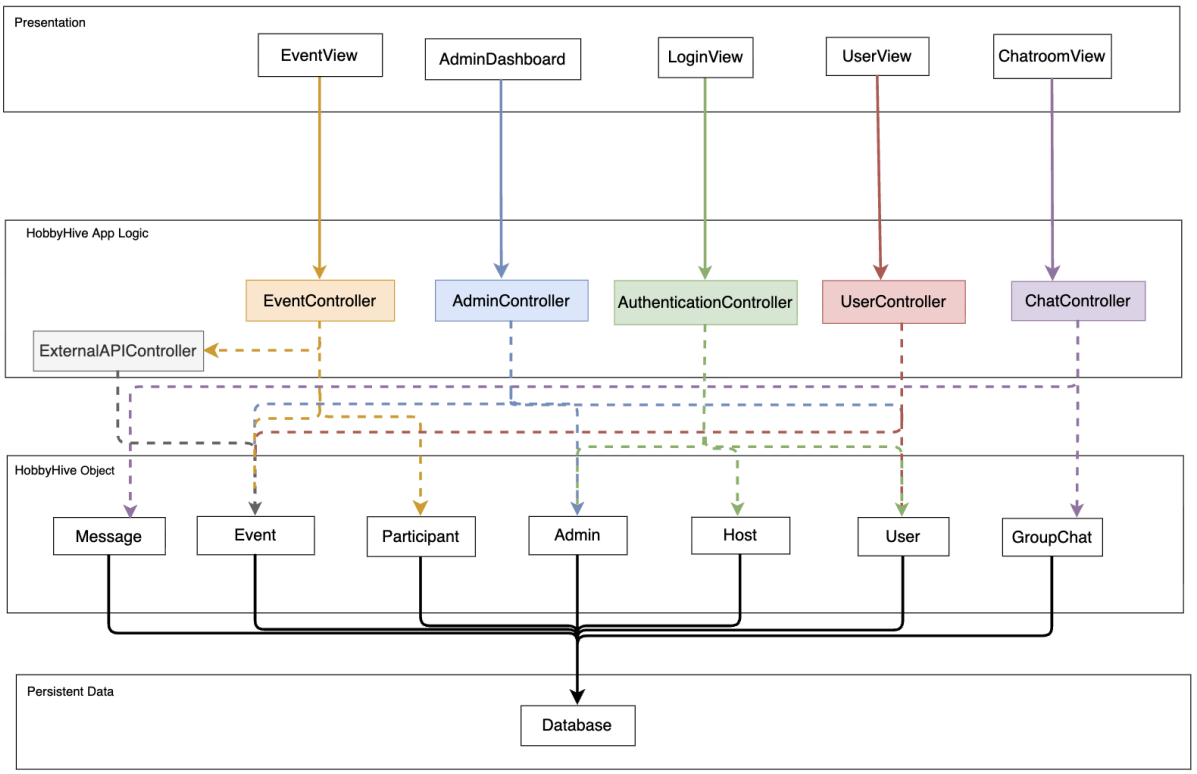
**Next.js Framework Documentation:** https://nextjs.org/docs

# 2. Overall Description

## 2.1 Product Perspective

This SRS specifies the requirements for **HobbyHive**, a web-based event management and community platform designed to connect individuals through shared interests and hobbies. It is a **self-contained product** and not a continuation or replacement of any existing system. HobbyHive aims to create a digital ecosystem that enables users to create, join, and manage hobby-based events, from music and fitness to art and gaming.

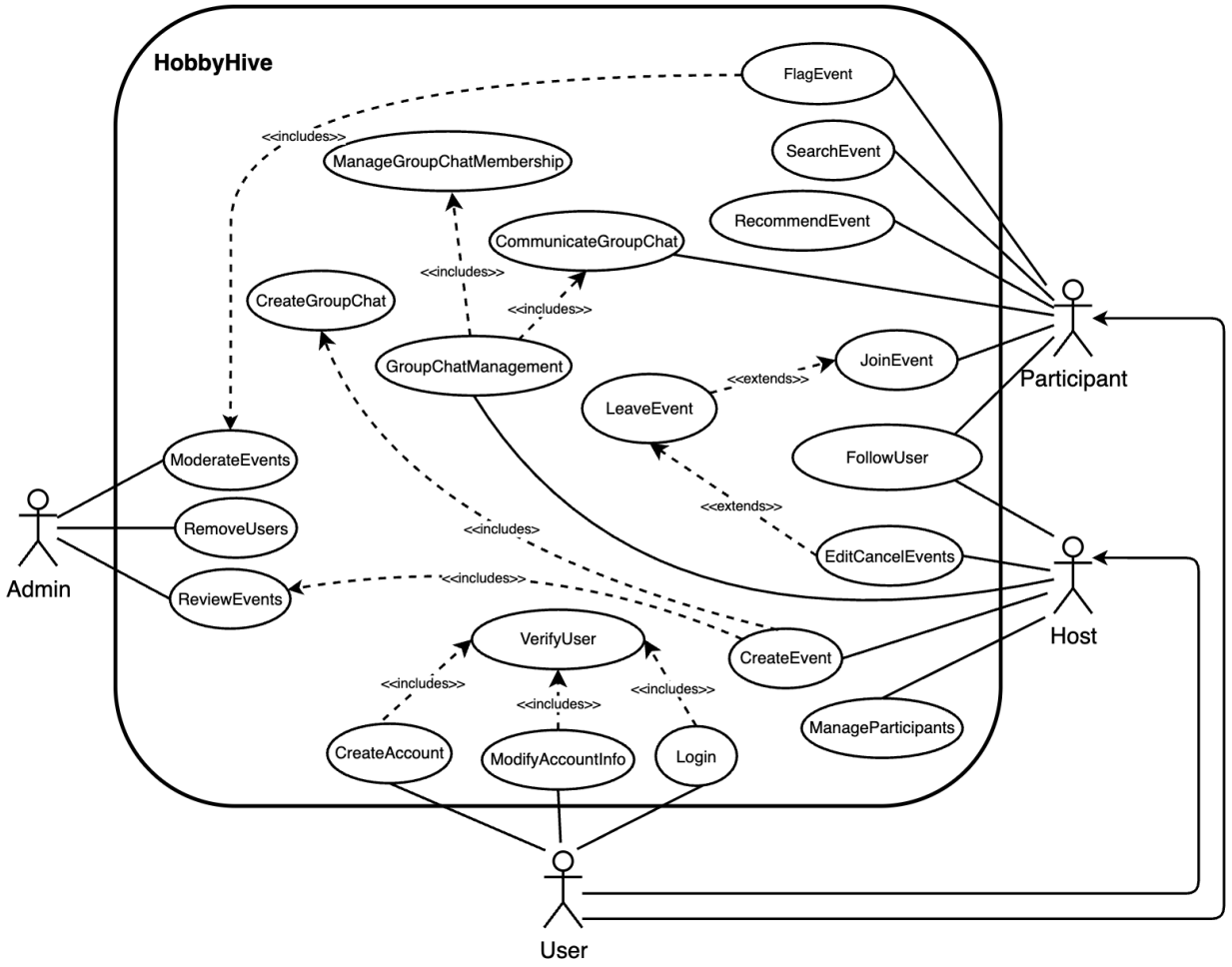
The platform functions independently but can integrate with external services via APIs (e.g., location, authentication). It uses **Supabase** for database and authentication management and **Next.js** for seamless frontend-backend integration.
A system architecture diagram (shown below) illustrates how the frontend, backend, and external APIs interact.




## 2.2 Product Functions

### 2.2.1 Use Case Diagram

The following use case diagram (shown below) represents the main functionalities of **HobbyHive**.
There are **four primary user types**, **Admin, Host, Participant, and User**, each with distinct roles and permissions in the system.




### 2.2.2 Major Product Functions

HobbyHive allows users to perform the following key functions:

1. **Authentication**
   - **Signup** (as Admin, Host, Participant or User)
   - **Login / Logout**

2. **Host Functions**
   - **Create Event**
     - Add title, description, date, time, location, category, and capacity.
     - Choose between manual location input or map picker (integrated with OneMap and data.gov.sg).
   - **Edit Event**
     - Update event details with validation checks.
   - **Cancel Event**
     - Mark an event as cancelled and notify participants.

- **View My Events**
  - Access all hosted events and manage participant lists.

**3. Participant Functions**

- **View and Search Events**
  - Browse events by category, keyword, or nearest MRT station.
- **Join and Unjoin Events**
  - Register or withdraw from events with real-time updates.
- **Flag Events/Users**
  - Flag inappropriate events or users
- **Follow Users**
  - Follow other users to get better event recommendations

**4. Admin Functions**

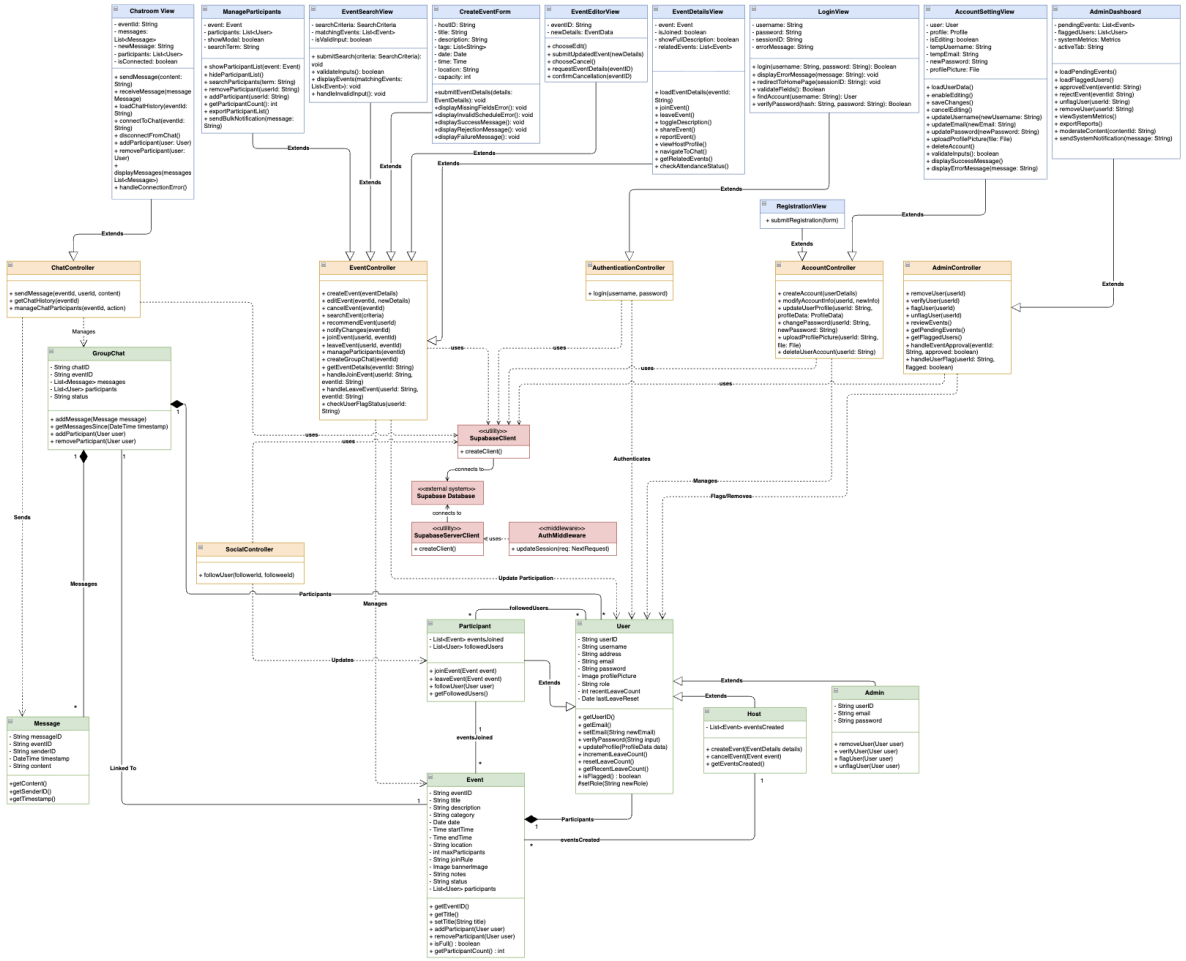
- **Verify and Manage Users**
  - Approve or suspend accounts that violate guidelines.
- **Moderate Events**
  - Remove inappropriate listings or duplicates.
- **System Monitoring**
  - View statistics and manage database integrity.

**5. Other Functions**

- **Profile Management**
  - Edit personal information and preferences.
- **Event Recommendations**
  - Recommendations based on user interests and activity.
- **Map Integration**
  - Suggest places in embedded map and identify nearest MRT stations using data.gov.sg APIs and OneMap API.

## 2.2.3 Class Diagram

The following Class Diagram (as shown below) illustrates the relationship between the key classes in **HobbyHive**, including **User**, **Event**, **Host** and **Participant**. The **Event Controller** manages creation, updates, and deletion, while the **User Controller** handles authentication and authorization processes.



## 2.3 User Classes and Characteristics

Users of **HobbyHive** fall into four categories:

- **Hosts**: Users who create and manage events. They need an intuitive interface for event creation, editing, and participant management.
- **Participants**: Users who discover, join, and attend events. They require smooth navigation and filters.
- **Admins**: Platform administrators who moderate users and events. They need a dashboard for oversight and analytics.
- **Visitors**: Unregistered users who can browse public events but must register to join or host.

Each user type contributes to the HobbyHive ecosystem, organizers create activities, participants engage, admins ensure platform integrity, and visitors sustain user growth.

## 2.4 Operating Environment

HobbyHive is a **web-based platform**, optimized for both **desktop and mobile browsers**.
It operates across all major browsers, Chrome, Firefox, Edge, and Safari, and is hosted on **Vercel** or other cloud services for high scalability and availability.
The system integrates with external APIs such as:

- **Supabase** (authentication, database, and storage)
- **OneMap and data.gov.sg API** (for map and MRT station data)

The application supports **cross-device responsiveness** using modern UI frameworks.

## 2.5 Design and Implementation Constraints

### 2.5.1 Frontend Constraints

#### 2.5.1.1 Framework (Frontend)

The user interface will be developed using **Next.js (React)**, **TypeScript**, and **TailwindCSS** for fast and modular development.
*TypeScript* ensures strong typing for maintainability and error reduction.

## 2.5.2 Backend Constraints

### 2.5.2.1 Framework (Backend)

HobbyHive uses **Supabase** (built on **PostgreSQL**) as its backend for user authentication, data storage, and API integration.
Supabase's RESTful API simplifies communication between the frontend and backend.

### 2.5.2.2 Database

Data is stored in a **PostgreSQL** database, which is relational and well-suited for structured event and user data.
Supabase provides built-in data access control and real-time synchronization.

### 2.5.2.3 Code Formatting

Backend logic within Supabase edge functions or any server scripts will follow standard linting and formatting rules to ensure maintainability.

## 2.5.3 Language Constraints

- All system documentation, user interfaces, and support materials will be provided in English.

- English was chosen due to its accessibility among Singapore's and global users, ensuring consistency and clarity across all communications.

## 2.6 User Documentation

HobbyHive will include detailed user documentation to ensure ease of use for both end-users and developers.

### 2.6.1 README Files

#### 2.6.1.1 Main Repository

Provides an overview of system architecture, setup instructions, environment configuration, deployment steps, and repository structure.

#### 2.6.1.2 Lab Deliverables Repository

Explain files in the folder for each lab's deliverables.

#### 2.6.1.3 API.md and DATABASE.md

Contain API documentation and database documentation.

### 2.6.2 Code Comments

- Code will include clear inline comments for complex logic, validation, and event handling functions.

### 2.6.3 Backend API Documentation

- The backend of HobbyHive is implemented using custom Next.js API routes alongside a managed Supabase (PostgreSQL) database. There are two main backend layers:
- The API layer, which exposes RESTful endpoints documented in Markdown (see docs/API.md)
- The database layer, which models data in Supabase and is described in docs/DATABASE.md

**Next.js Custom API Routes**

- All business logic (e.g., events, community clubs, parks, libraries, geocoding, group chat) is handled through custom endpoints defined in app/api/.
- These API routes act as the interface for the frontend to interact with event data and third-party sources.
- The API supports common RESTful operations, authenticates users via Supabase Auth, and returns data in JSON format.

**Supabase Database Integration**

- Data is persisted in Supabase Postgres using a normalized relational schema (see DATABASE.md for structure).
- Supabase is used primarily as a database and authentication provider.
- Direct use of Supabase's auto-generated REST API ("/rest/v1/tables") is not exposed to the frontend, instead, all access goes through the custom API routes. (refer utils/supabase)

**Third-Party Integrations**

- Some API routes (e.g., geocoding, MRT station lookup) serve as proxies to external APIs like OneMap and data.gov.sg.
- These external integrations are transparently incorporated into API, so frontend developers only deal with unified endpoints.

**Documentation Format**

- API.md contains a complete list of available endpoints, request/response formats, query/body parameter definitions, and example payloads.
- DATABASE.md describes the data model, table relationships, field constraints, and indexes.
- All documentation is maintained in Markdown (docs/ folder), enabling easy updates and codebase versioning.

## 2.7 Assumptions and Dependencies

### 2.7.1 Assumptions

- Users will have stable internet connectivity to access cloud-hosted services.
- Browser support will remain consistent with current standards for ES6+ and modern CSS.

### 2.7.2 Dependencies

#### 2.7.2.1 Frontend Libraries

The user interface of HobbyHive depends on numerous front-end libraries:

| **Library** | **Purpose** |
|---|---|
| React / Next.js | Framework for building the SPA with server-side rendering |
| TailwindCSS | For rapid and responsive UI design |
| TypeScript | Ensures strong typing and fewer runtime errors |
| react-icons | Provides icons for better UI aesthetics |
| react-hook-form | Manages form validation and input handling |
| next/navigation | For client-side routing and navigation |
| dynamic (Next.js) | Lazy-loads map picker and components for performance optimization |
| shadcn/ui | Provides styled UI components |
| lucide-react | Supplies modern and lightweight icons |

#### 2.7.2.2 Backend Libraries

The backend of HobbyHive depends on numerous back-end libraries:

| **Library** | **Purpose** |
|---|---|
| Supabase | Provides authentication, database, and storage |
| PostgreSQL | Stores relational data (events, users, categories) |
| bcrypt | Secures passwords via hashing (used internally by Supabase) |
| jsonwebtoken | Manages secure authentication tokens |
| node-fetch | Handles API requests (if custom API calls are needed) |

#### 2.7.2.3 External Services

HobbyHive depends on several external services to ensure smooth functionality:

- **Third-party APIs:**
    - Singapore's OneMap API
    - Singapore's OneMap Nearest MRT API
    - Community Clubs GEOJSON dataset by PA (People's Association)
    - Libraries GEOJSON dataset by NLB (National Library Board)
    - Parks GEOJSON data set by NPARKS (National Parks Board)

- **Cloud Hosting Services:**
    - Vercel (for frontend deployment)
    - Supabase (for backend database and storage hosting)

Hence, HobbyHive's operations rely on the continuous availability and reliability of these external services.

# 3. External Interface Requirements

## 3.1 User Interfaces

### 3.1.1 Style Guides

#### 3.1.1.1 Theme Colours

| Primary Accent | Secondary Accent | Background Base 1 | Neutral 1 | Status Indicator Background | Warning Background | Clear Actions Button |
|---|---|---|---|---|---|---|
| #1DDACA | #CCF3F6 | #FFFFFF | #363636 | #DBFCE7 | #FEFCE8 | #FB2C36 |

| Primary Accent (Hover) | Secondary Accent (Hover) | Background Base 2 | Neutral 2 | Status Indicator Text | Warning Border |
|---|---|---|---|---|---|
| #30DDCE | #E0F9F7 | #F3F4F6 | #E5E7EB | #008236 | #FFF085 |

| Background Gradient 2 | Background Gradient 1 | Background Base 3 | Warning Text |
|---|---|---|---|
| #00B8DA | #00BBA7 | #A8F0EB | #894B00 |

### 3.1.1.2 Typography

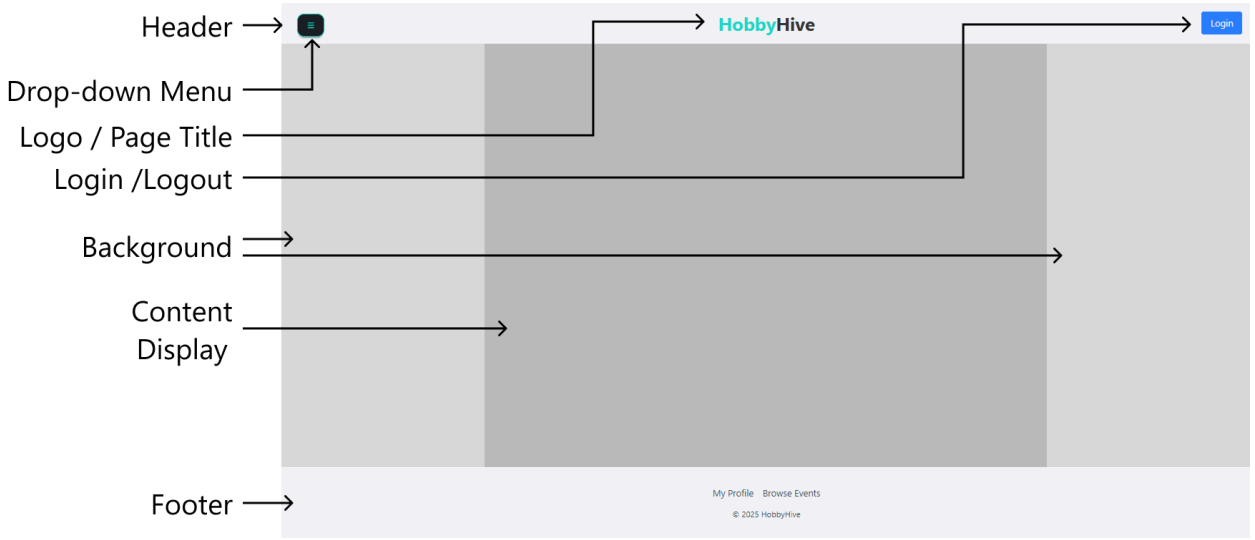
Typography differs across devices based on their operating system, using their native system font for best readability and performance.

Windows


macOS

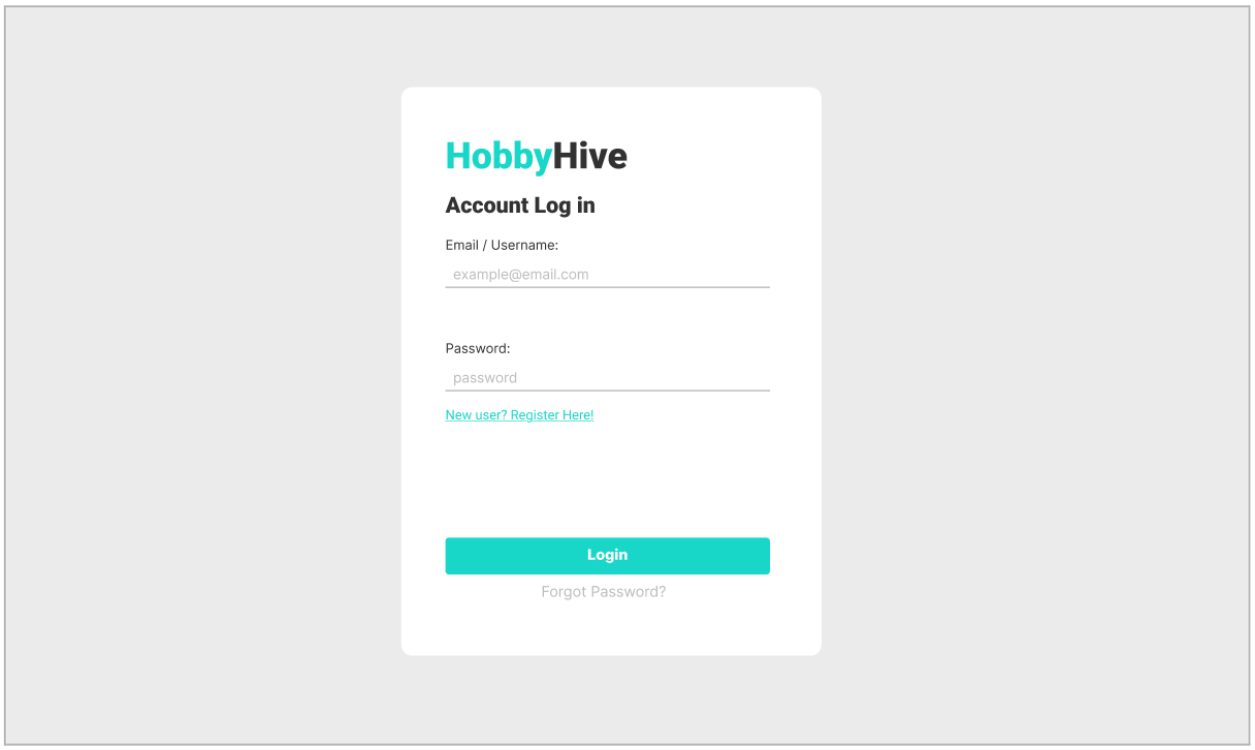

Ubuntu

### 3.1.1.3 Website Template

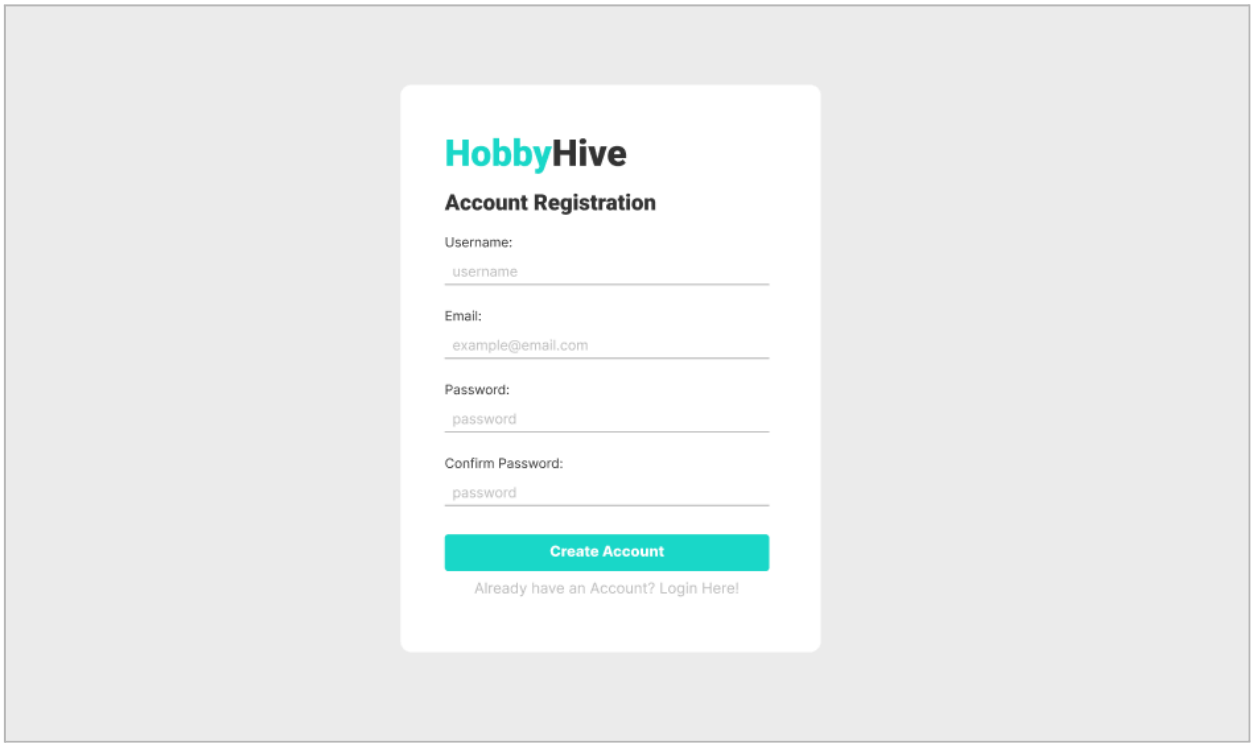



## 3.1.2 UI Mockups (Figma)

### 3.1.2.1 Login Screen

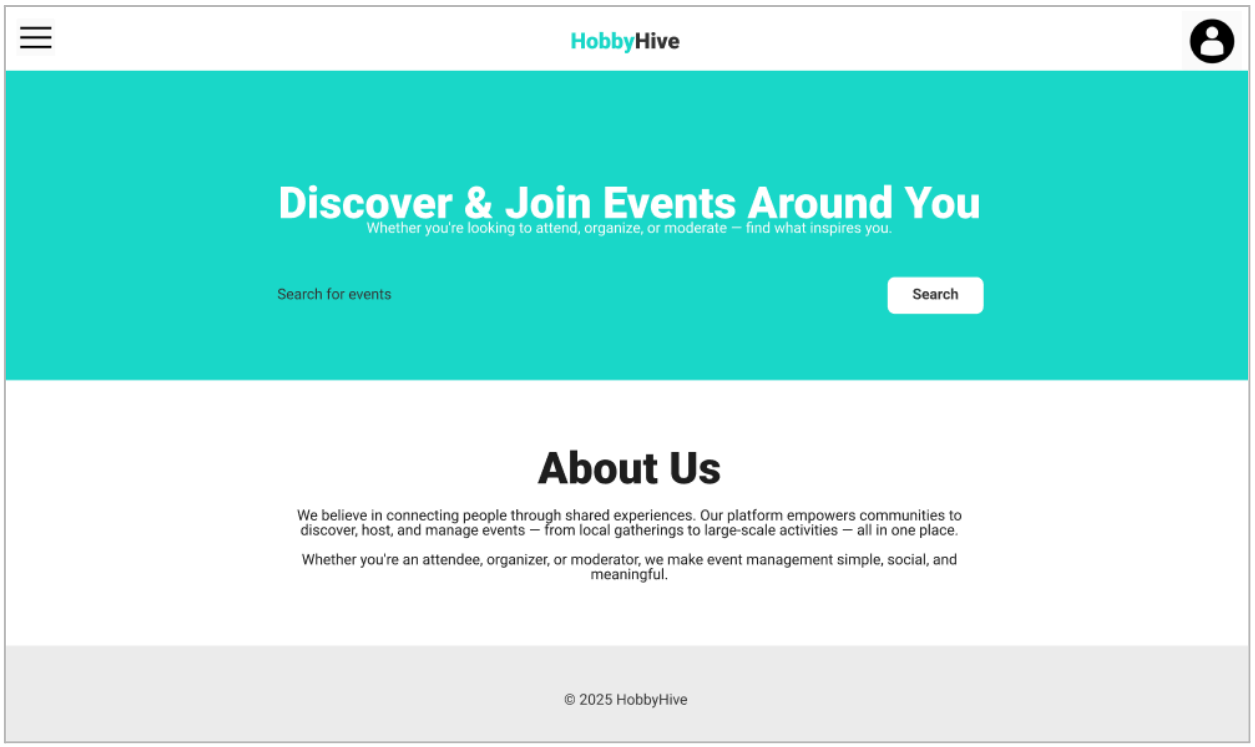


#### 3.1.2.2 Registration Screen



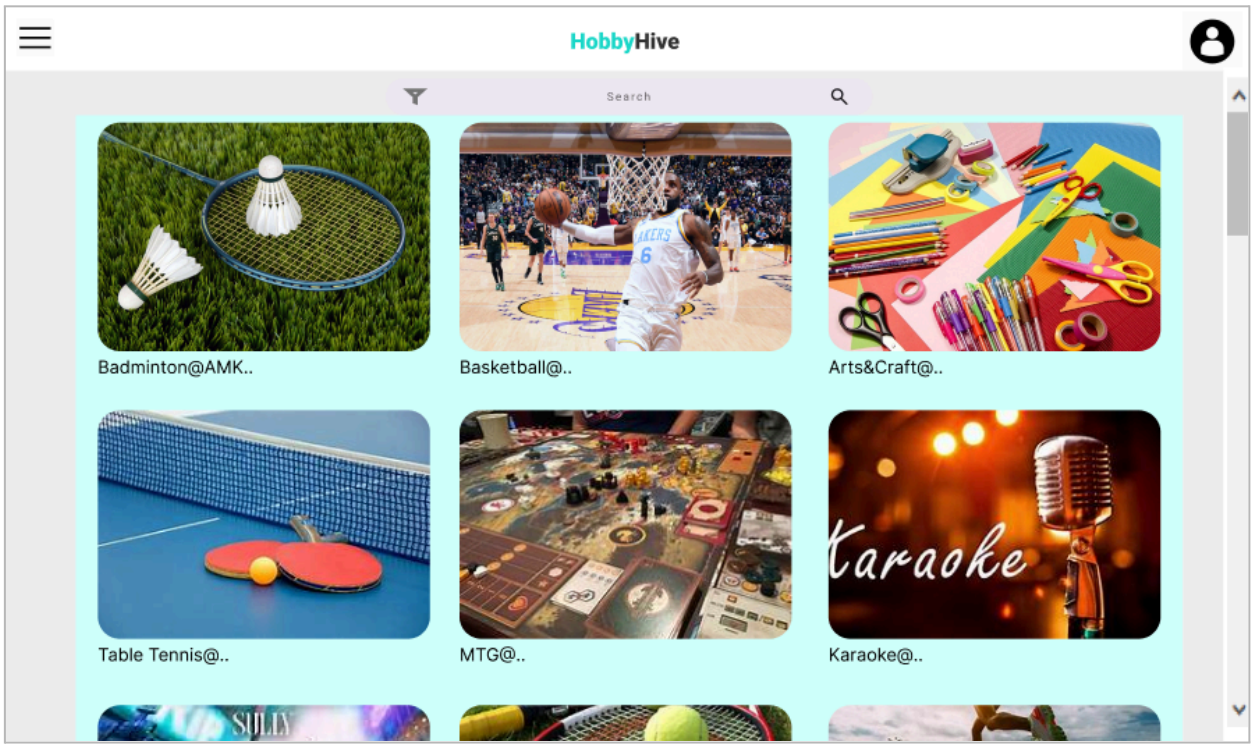

### 3.1.3 User Screens

#### 3.1.3.1 Landing Screen

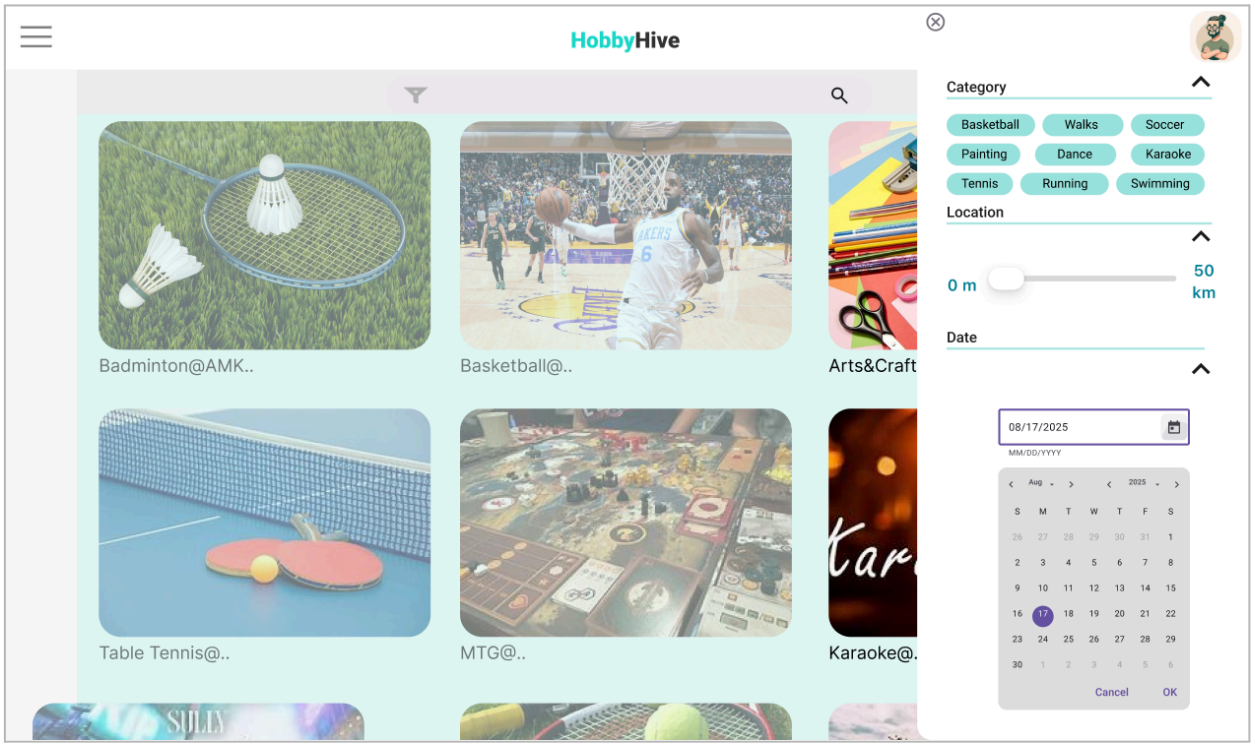


### 3.1.3.2 Search Events

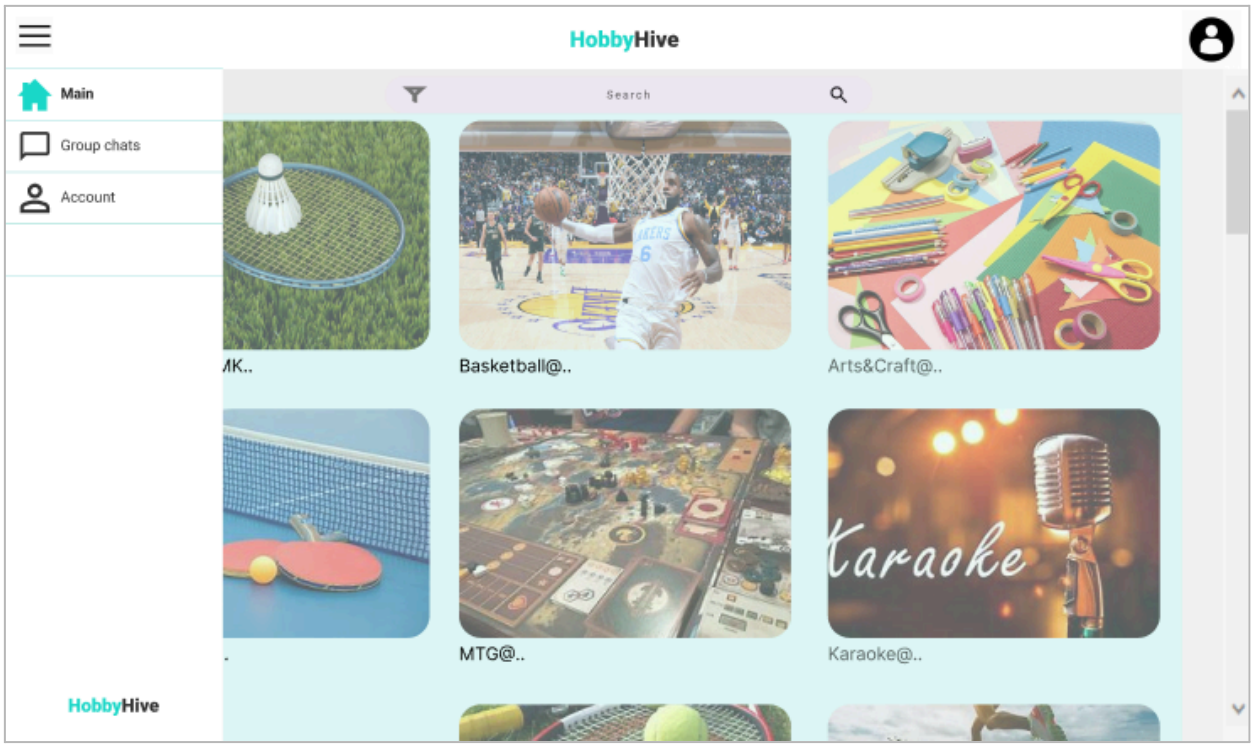
#### 3.1.3.2.1 Default Event Screen



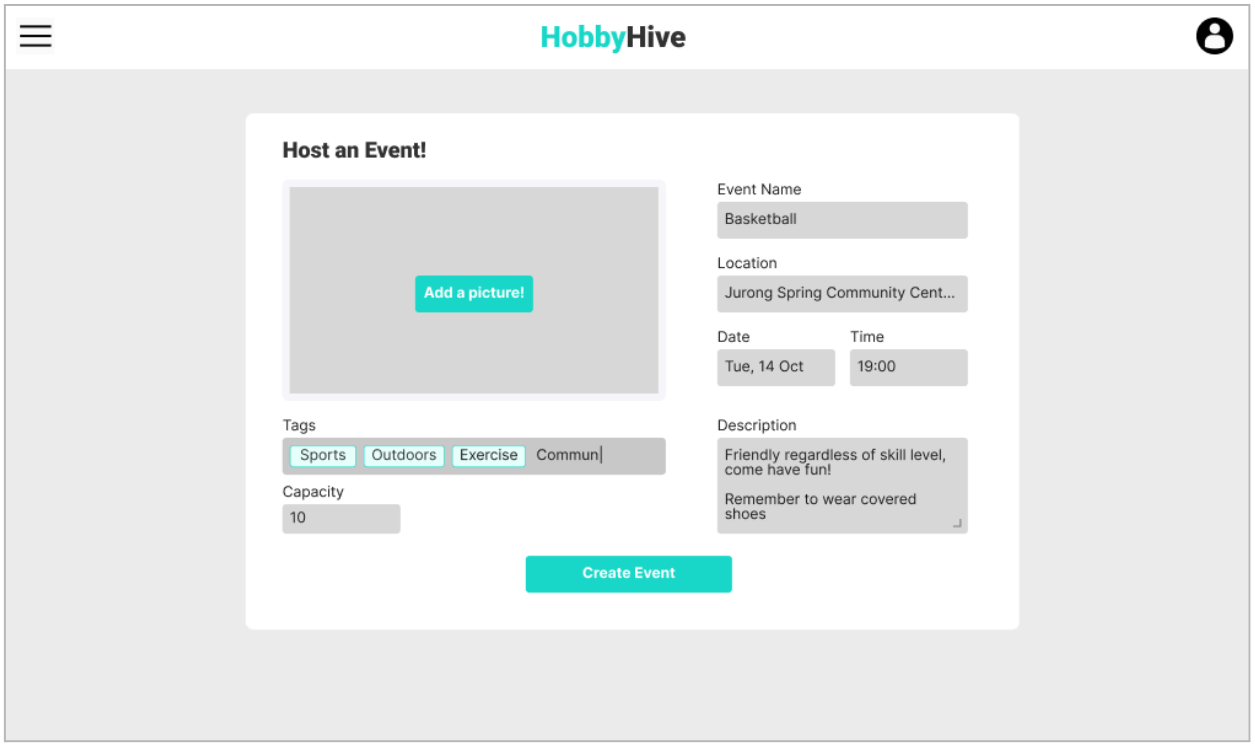

#### 3.1.3.2.2 Event Screen with Search Filter



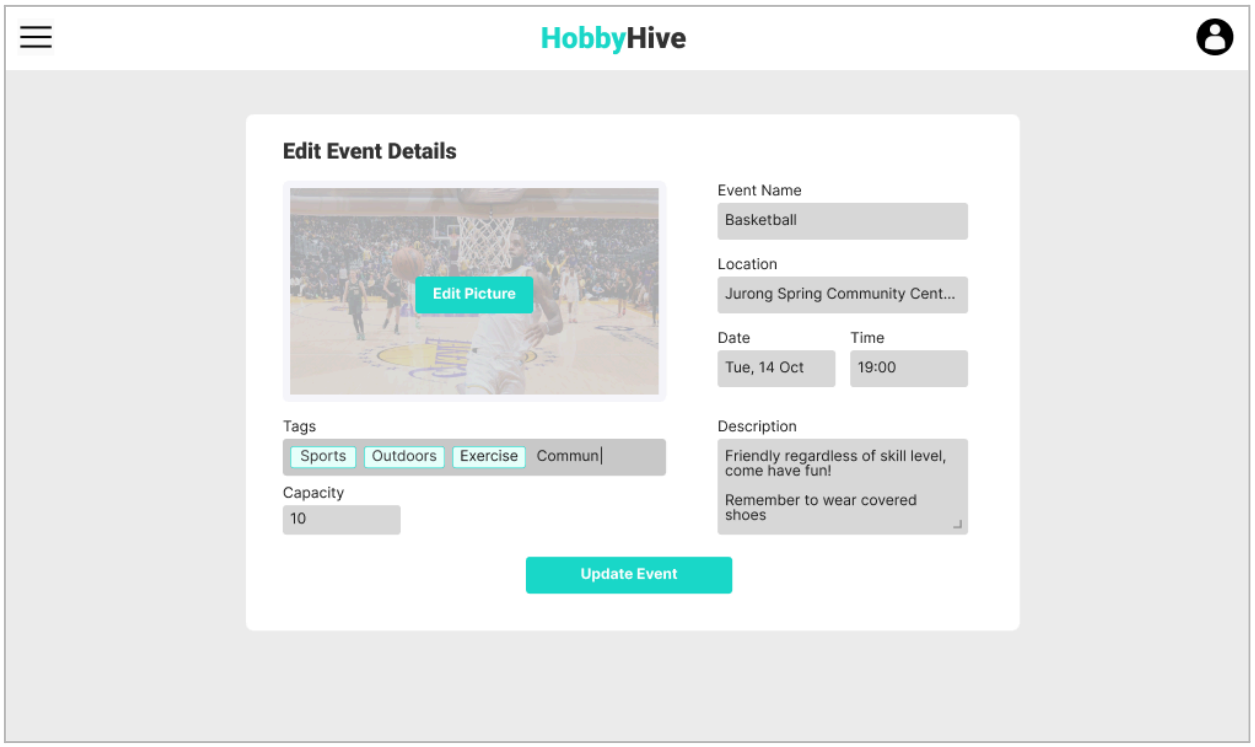
#### 3.1.3.2.3 Event Screen with Dropdown Menu



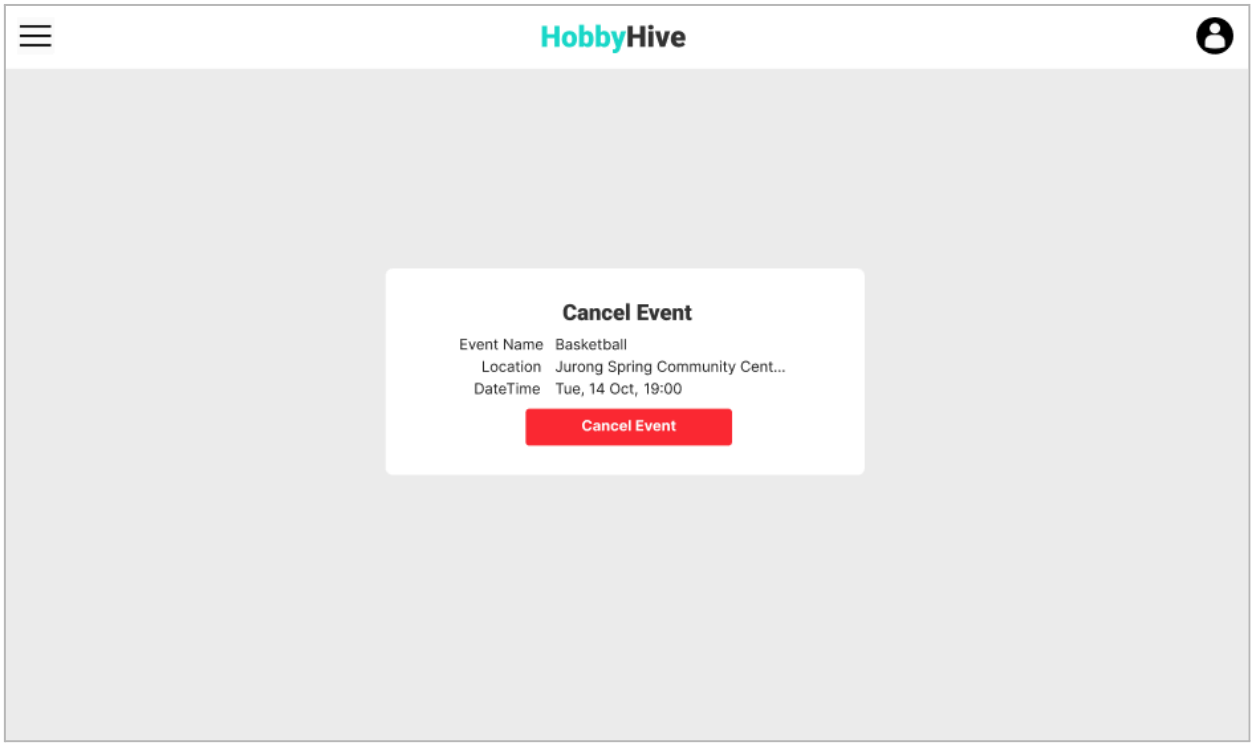

### 3.1.3.3 Create Event



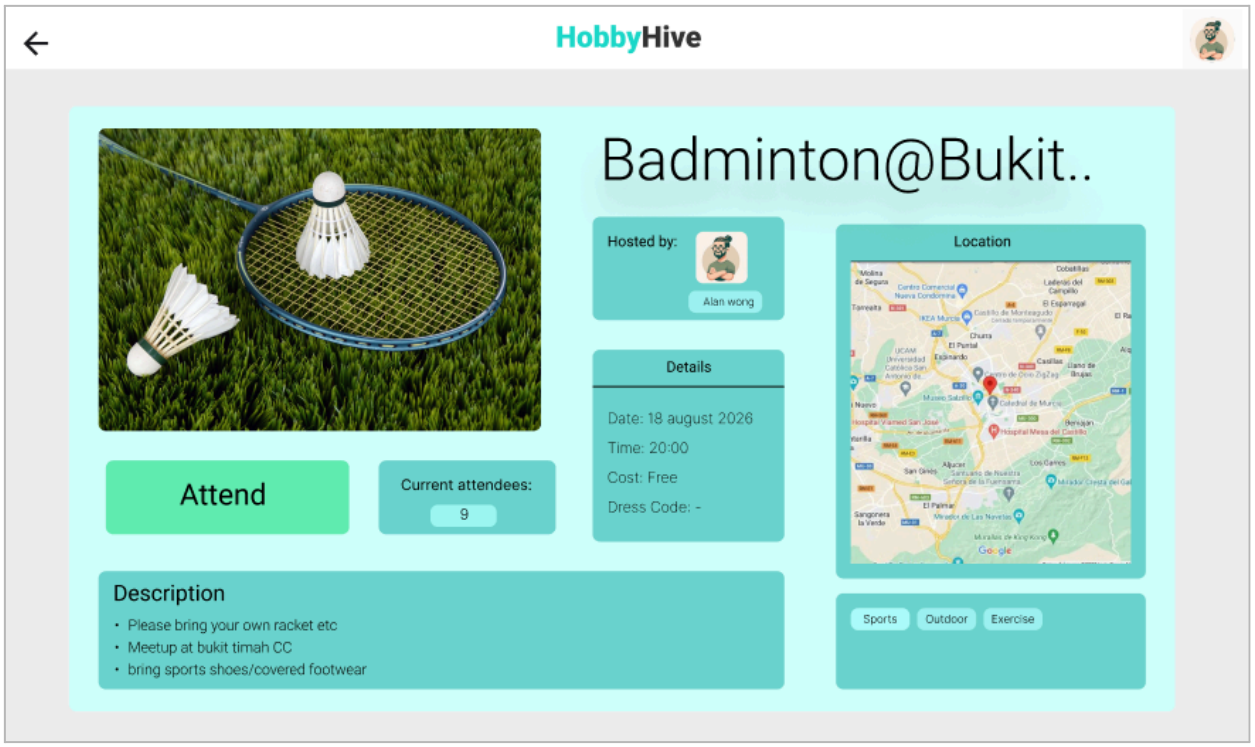
### 3.1.3.4 Edit Event



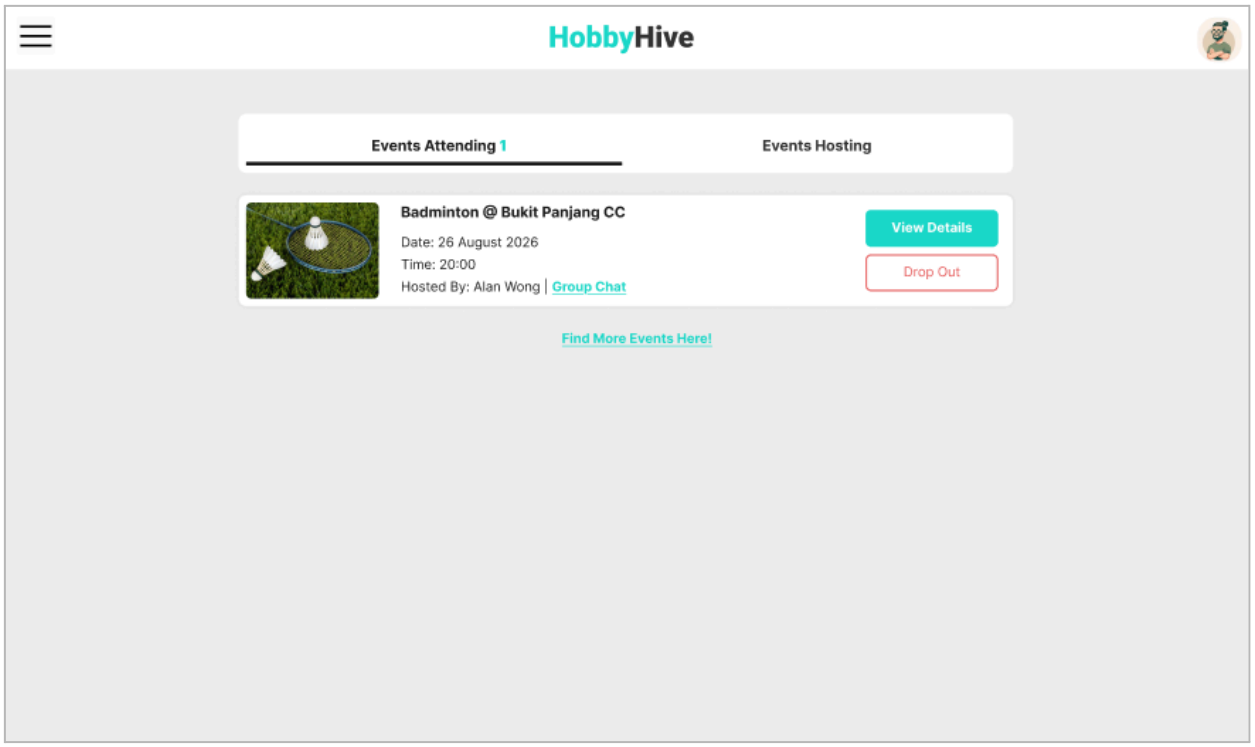

### 3.1.3.5 Cancel Event

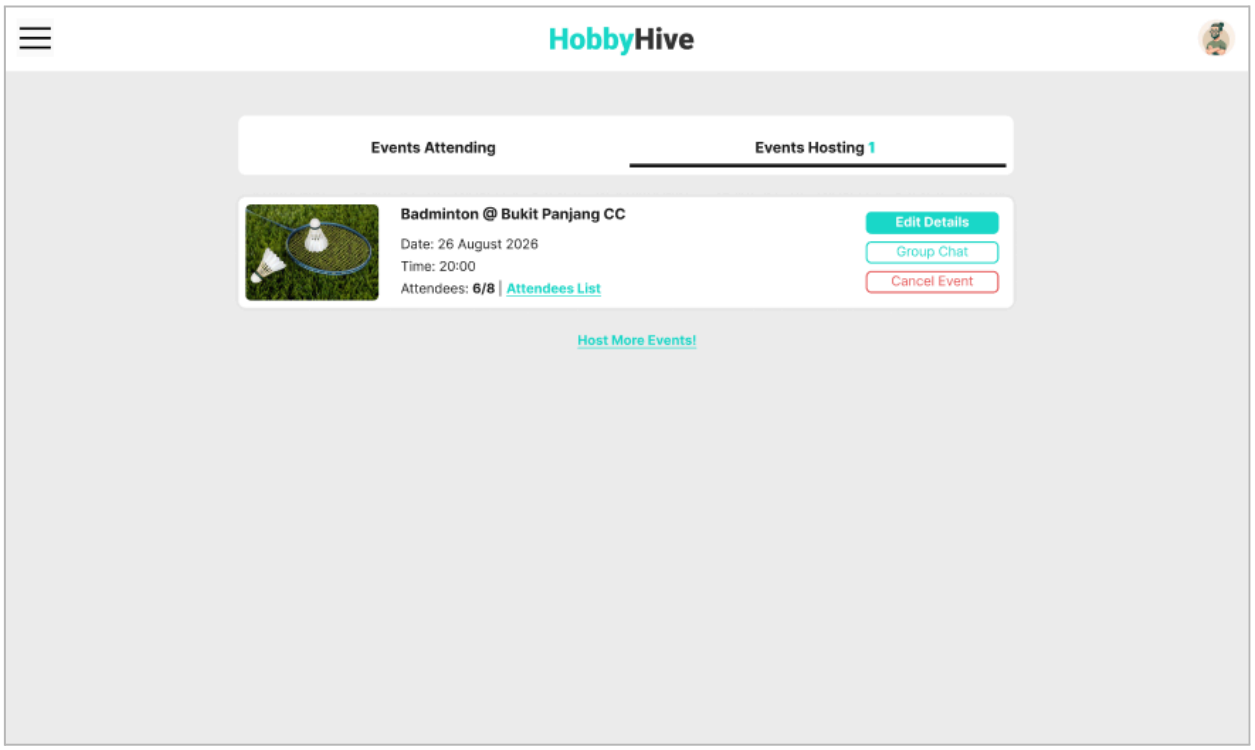


### 3.1.3.6 View Event Page

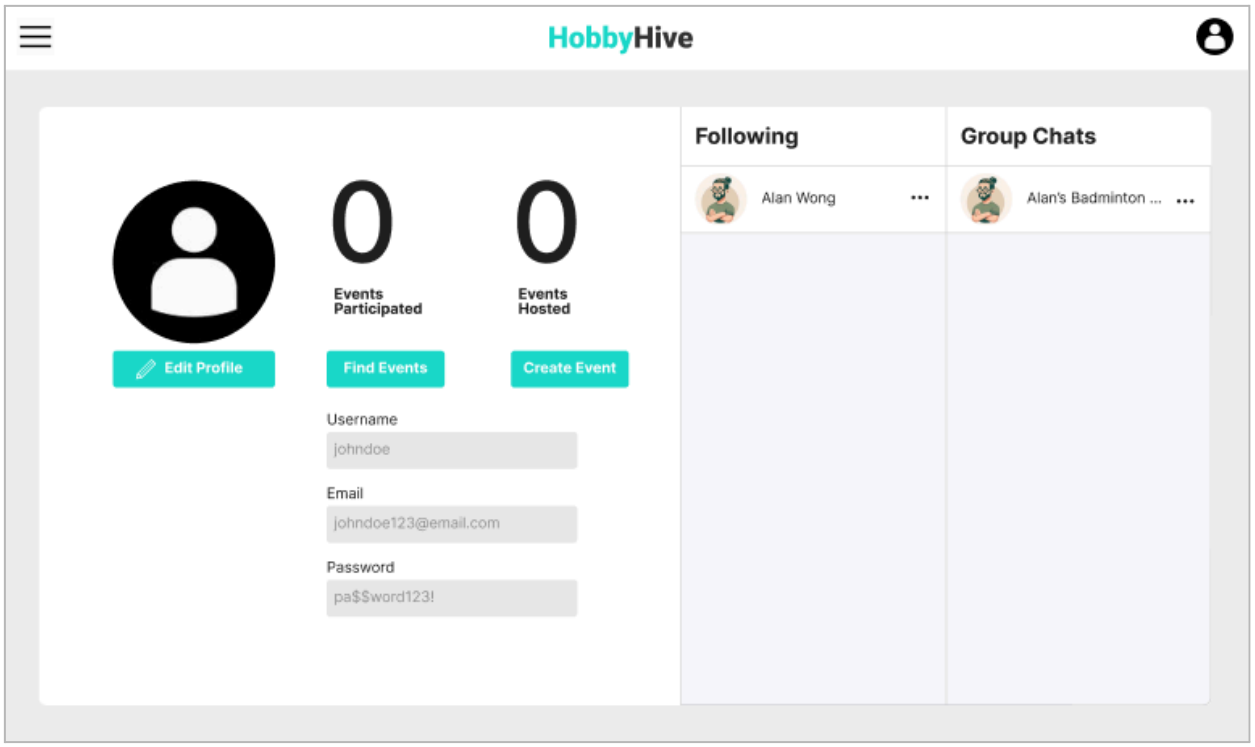


### 3.1.3.7 My Events Page

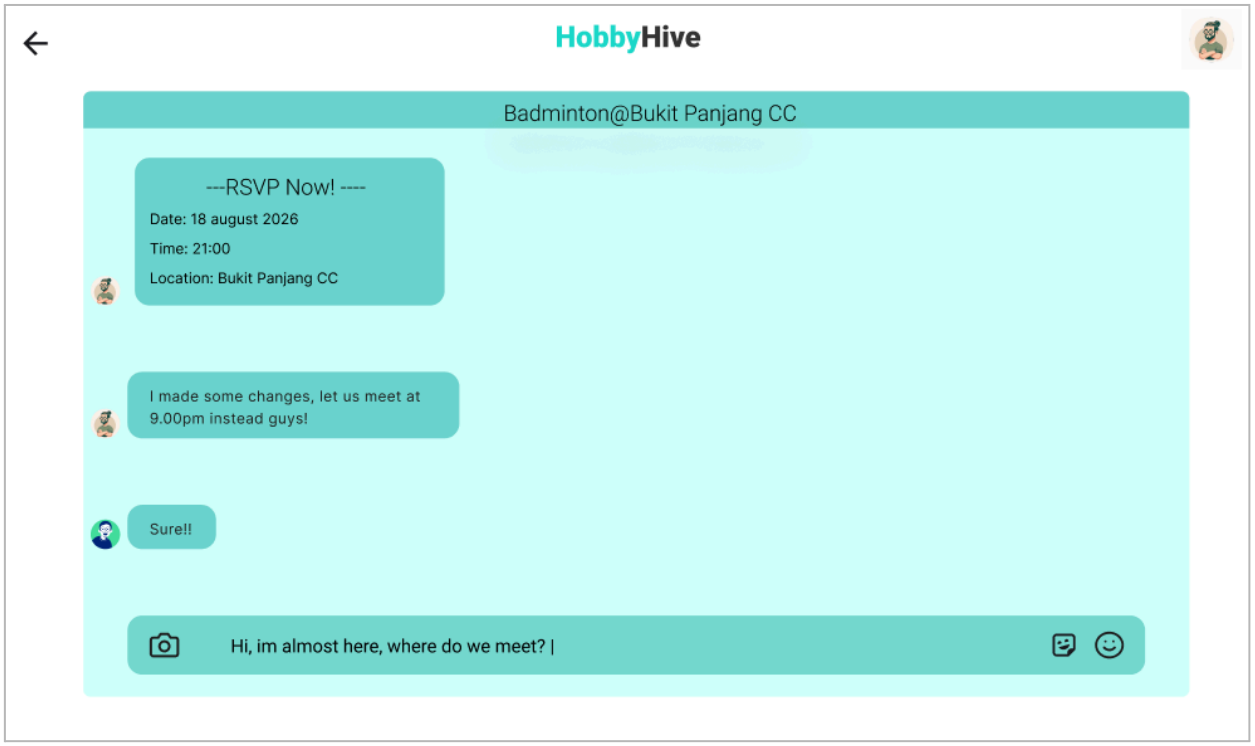


### 3.1.3.8 Profile Page

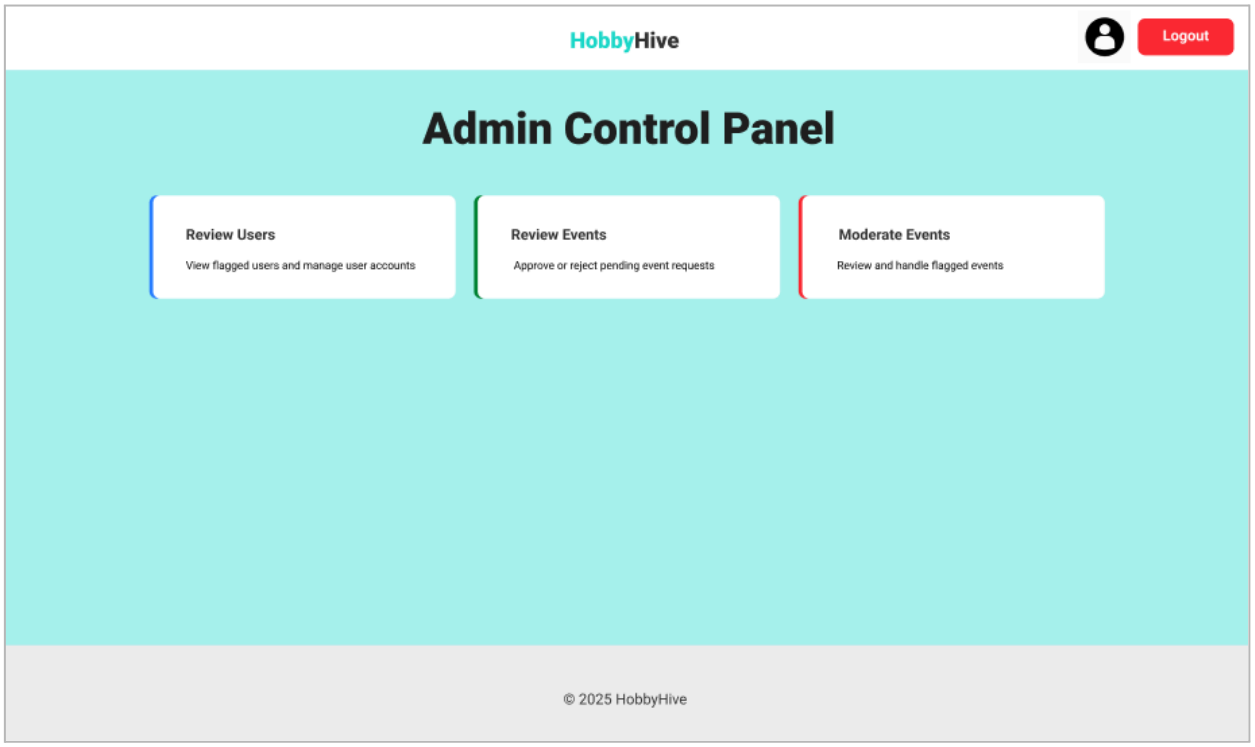



### 3.1.3.9 Group Chat Page

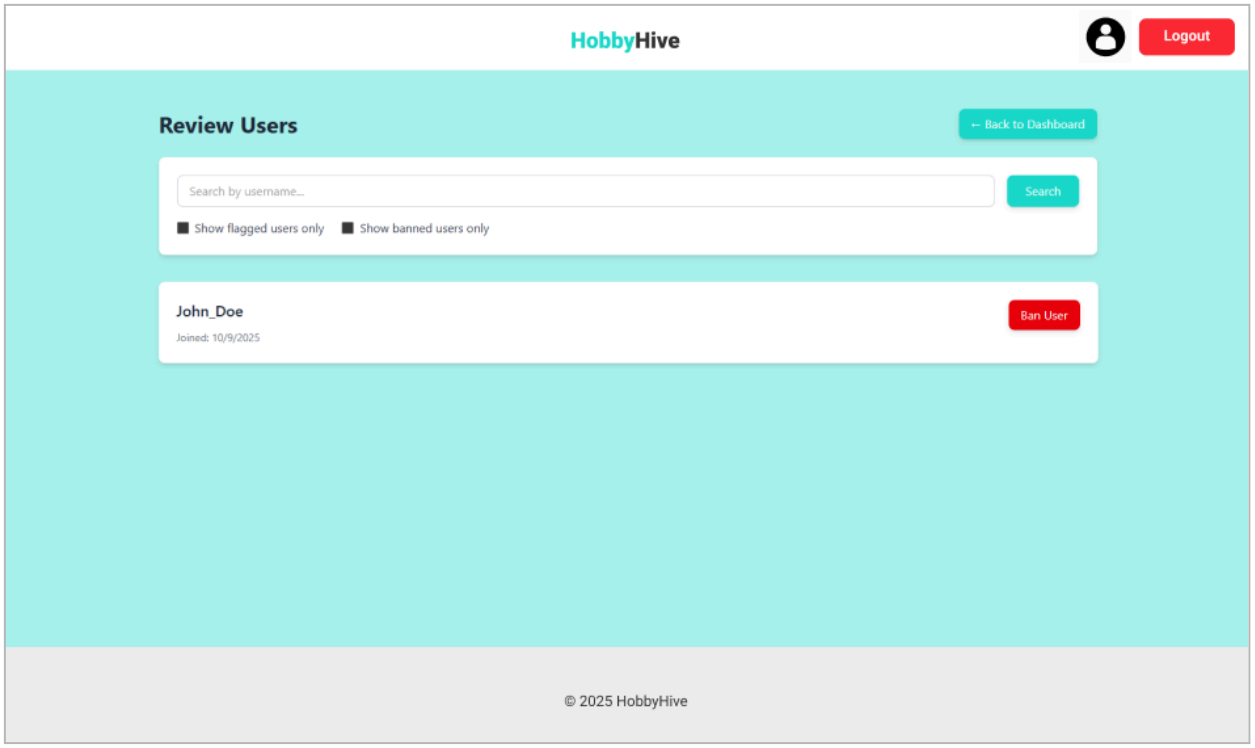


### 3.1.4 Admin Screen

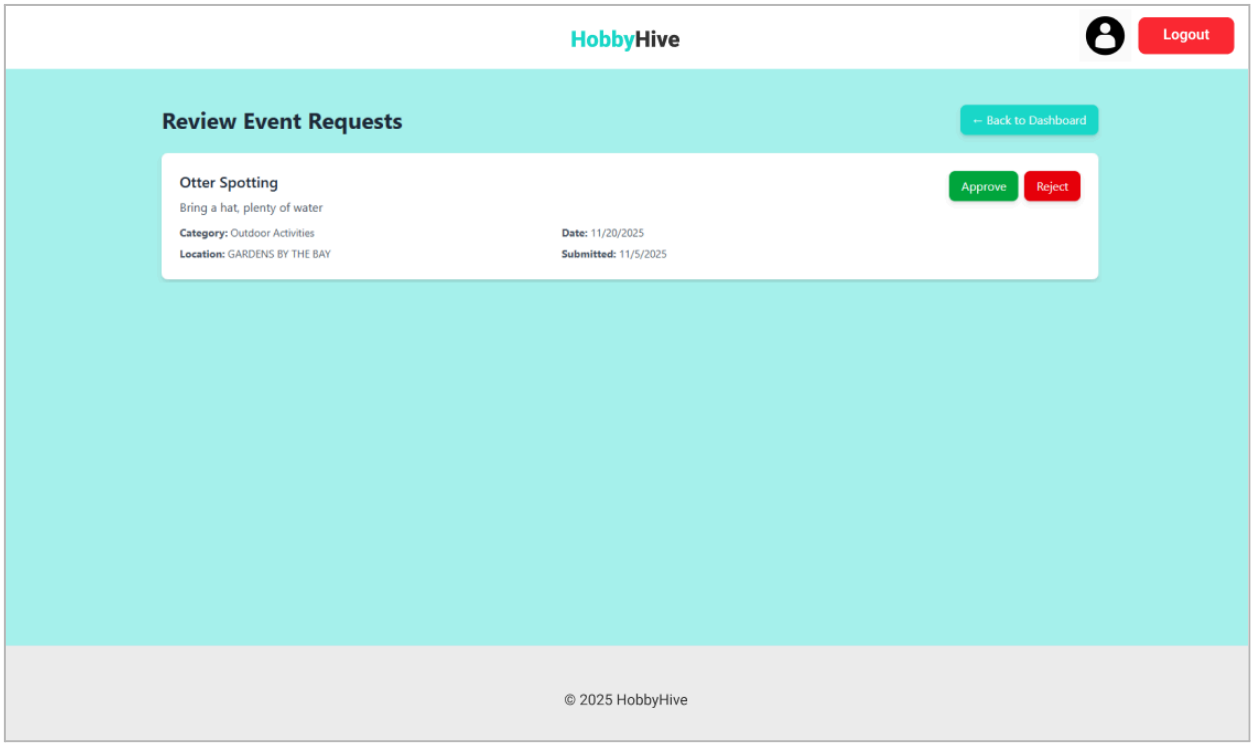
#### 3.1.4.1 Admin Dashboard



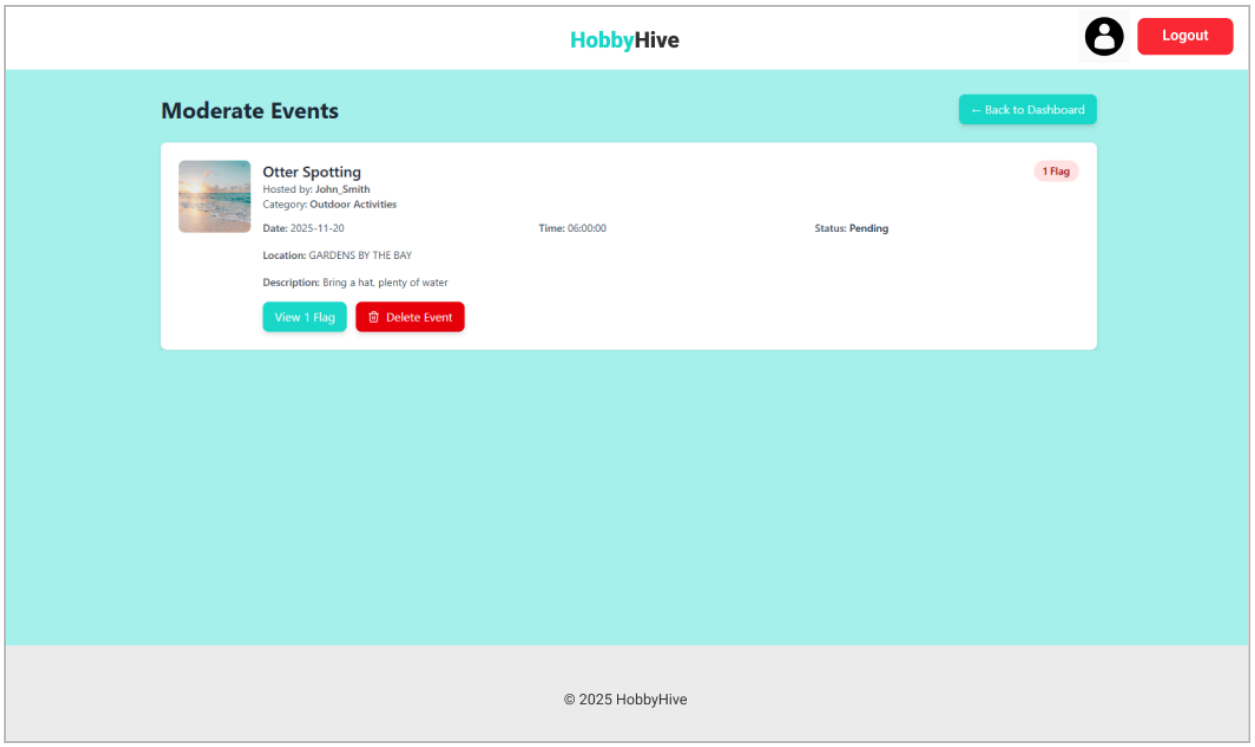
### 3.1.4.2 Review users

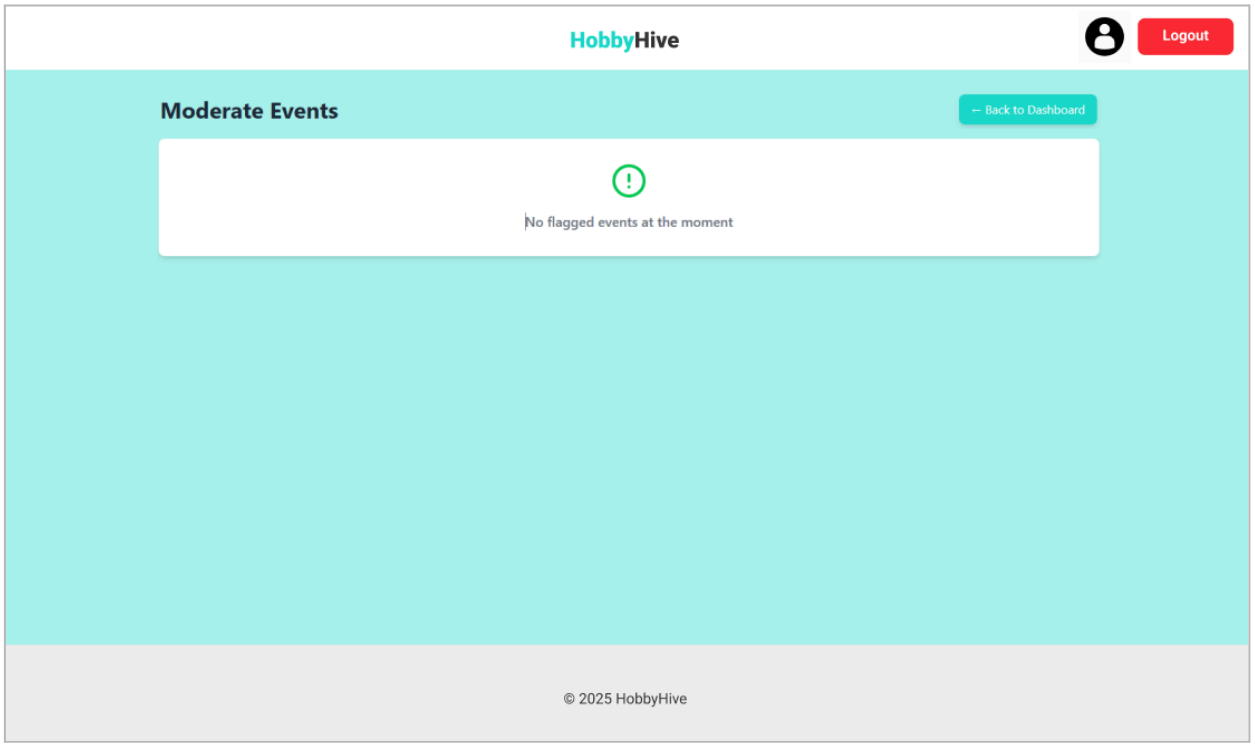



### 3.1.4.3 Review events



### 3.1.4.4 Moderate Events



### 3.1.5 Success/Error Message Display (Toast Notification)

Toast Notifications are used for bringing attention to important information regarding status updates. Examples include invalid login credentials (Fig. 3.1.5.a), successfully joined event (Fig. 3.1.5.b), and disclaimers (Fig. 3.1.5.c).

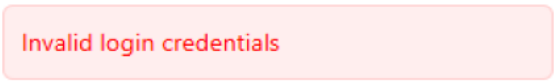



Figure 3.1.5.a – Invalid Login

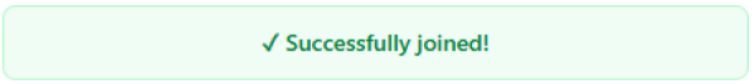
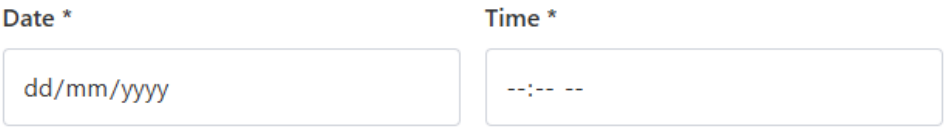



Figure 3.1.5.b – Successfully joined event

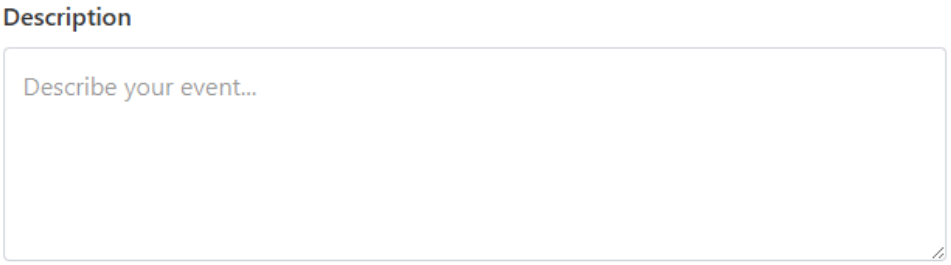

Figure 3.1.5.c – Disclaimer

### 3.1.6 Common Components

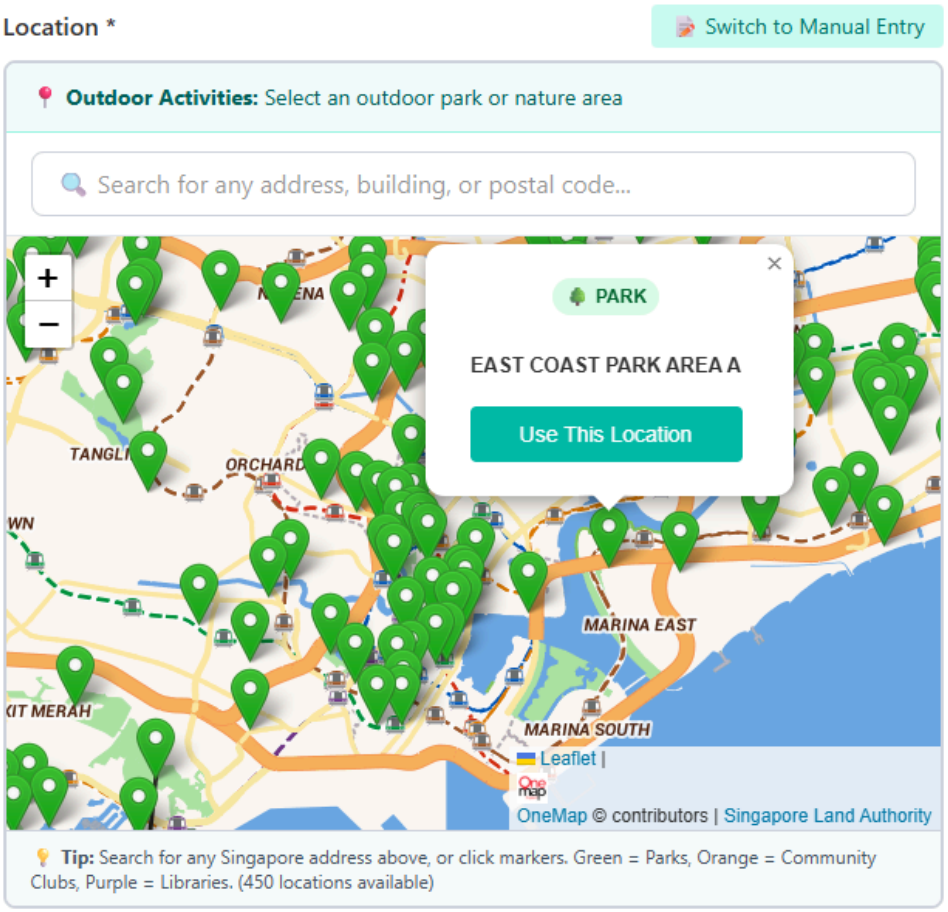

Form Text Input




Form Date Time Input




Form Textarea Input




Form Location input


Form Proximity Slider Input


Form Buttons

## 3.2 Hardware Interfaces

HobbyHive is designed as a fully web-based application, with no direct interaction with hardware interfaces. The following describes the physical and logical characteristics of the interfaces between the software and client devices.

### 3.2.1 Device Types

The application is designed to be compatible with readily available devices capable of running modern web browsers, such as desktops, laptops, tablets, and smartphones.

### 3.2.2 Data and Control Interactions

Users can interact with HobbyHive through standard I/O peripherals such as:

- Input: Keyboard, mouse, touchscreen
- Output: Display screens
- Network Access: Wi-Fi or mobile data for communication with backend services

No computation is handled directly by hardware beyond the browser.

### 3.2.3 Communication Protocols

The system uses HTTPS over TCP/IP for secure communication between the client and the server and external APIs. All data exchanges follow REST conventions using JSON formats, including authentication requests, which use JWT tokens.

## 3.3 Software Interfaces

This section illustrates the system interaction with other software components and external services.

### 3.3.1 Operating System

The application will be deployed on Vercel Cloud Platform, running on a Linux-based environment. Client interacts through a web browser supported on operating systems, including:

- Windows 10 and above
- macOS 12 and above
- Linux distributions (e.g., Ubuntu, Debian)

No OS-specific installation is required for the end-user.

### 3.3.2 Web Browsers

The system supports modern browsers, including the latest versions of Google Chrome, Firefox, and Safari. Older browsers, such as Internet Explorer, are not supported due to incompatibility issues with Next.js outputs.

### 3.3.3 Databases

Supabase is used as the database and authentication service.

- PostgreSQL database for structured data.
- Authentication using JWT tokens
- File storage for media uploads

### 3.3.4 Backend Tools

Backend logic is implemented using Next.js API Routes, which handle business logic processing, incoming data validation, and middleware for route protection and authentication.

### 3.3.5 Frontend Tools

- Next.js (v15.5.3) for routing and API integration.
- React (v19.1.0) is the component library for building UI components.
- Tailwind CSS (v4) for styling.
- DaisyUI (v5.1.29) is integrated as a Tailwind plugin for prebuilt UI components.

### 3.3.6 Data Sharing

- User data is shared only through secure Supabase queries
- Protected by authentication and RLS policies
- Media files are publicly accessible only when accessed via signed URLs or public buckets

### 3.3.7 External Data Integration

The system retrieves real-world data provided from Data.gov.sg and publicly available APIs, including:

- Singapore's OneMap API
- Singapore's OneMap Nearest MRT API
- Community Clubs GEOJSON dataset by PA (People's Association)
- Libraries GEOJSON dataset by NLB (National Library Board)
- Parks GEOJSON data set by NPARKS (National Parks Board)

## 3.4 Communications Interfaces

HobbyHive relies on widely adopted web communication standards to ensure reliable connectivity, secure data transmission, and smooth integration with external services.

1) HTTP/HTTPS enables client-server communication, Next.js API routes, the Supabase backend, and publicly available APIs. HTTPS (TLS encryption enforced) is used for secure data transmission over port 443, and automatic redirection from HTTP (port 80) to HTTPS when needed. This ensures the confidentiality and integrity of user data.

2) RESTful APIs are used for all backend communication. Supabase APIs are used to validate authentication, query, and update data. External APIs such as Data.gov.sg and OneMap are used for real-world, real-time geolocation data. JSON is used as a message format for all requests and responses

3) Electronic form submissions (e.g., event creation, profile updates) are transmitted through HTTPS and validated before being stored in the database.

4) TCP/IP is used for all communication under the web protocols for reliable, standards-compliant communication.

5) TLS encryption is used for all transmitted data, Supabase manages JWT-based authentication, and Row Level Security Policies are enforced on Supabase database operations. This ensures data integrity and client privacy.

These communication interfaces provide robust, seamless, and secure interaction with the platform.

# 4. System Features

## 4.1 Login Features

### 4.1.1 Login

#### 4.1.1.1 Description and Priority

**Description:**

- Users can securely log in to the system using their registered email and password.
- The system authenticates credentials through Supabase Authentication before granting access.
- The system verifies if the user is temporarily or permanently banned before login.

**Priority:**

- High

#### 4.1.1.2 Stimulus/Response Sequences

1. The user enters a registered email and password.
2. System verifies the provided credentials with Supabase Authentication.
3. If credentials are valid, the system checks for active bans in the profiles table.
4. If no ban is active, the user is redirected to the main dashboard.
5. If credentials are invalid or ban is active, the system displays appropriate error or ban message.

#### 4.1.1.3 Functional Requirements

REQ-1: System must allow users to enter login credentials (email and password).
REQ-2: System must verify login credentials via Supabase Authentication.
REQ-3: System must check for active bans (is_banned, banned_until) before granting access.
REQ-4: System must redirect authenticated users to the dashboard upon successful login.
REQ-5: System must display appropriate error messages for incorrect credentials or active bans.

### 4.1.2 Register

#### 4.1.2.1 Description and Priority

**Description:**

- Users can create a new account by entering an email, password, and username.
- Upon registration, the system automatically creates a corresponding record in the profiles table.

**Priority:**

- High

#### 4.1.2.2 Stimulus/Response Sequences

1. The user selects “Register” on the login screen.
2. The system displays the registration form with required fields.
3. The user enters their email, password, and desired username.
4. The system validates all inputs and checks for duplicate accounts.

5. Upon successful validation, an account is created in Supabase Authentication and an entry is added to the profiles table.
6. System redirects the new user to the main dashboard or login page.

#### 4.1.2.3 Functional Requirements

REQ-1: System must allow users to input email, password, and username for registration.
REQ-2: System must validate that the email is unique and the password meets security requirements.
REQ-3: System must create a new authentication record in Supabase.
REQ-4: System must create a corresponding user profile in the profiles table.
REQ-5: System must redirect users to the dashboard or login screen after successful registration.

### 4.1.3 Logout

#### 4.1.3.1 Description and Priority

**Description:**

- Users can log out of the system securely at any time.
- Logging out terminates the current session and clears any cached authentication tokens.

**Priority:**

- Medium

#### 4.1.3.2 Stimulus/Response Sequences

1. User selects the “Logout” button.
2. System invalidates the user’s active session and clears local authentication tokens.
3. System redirects the user to the homepage or login page.

#### 4.1.3.3 Functional Requirements

REQ-1: System must allow users to log out from any screen.
REQ-2: System must clear session data and invalidate tokens upon logout.
REQ-3: System must redirect the user to the login or home page after logout.

## 4.2 Participant Features

### 4.2.1 Join Event

#### 4.2.1.1 Description and Priority

**Description:**

- Participants can join any available event listed on the platform.
- The system records the participant's ID and event ID into the event_participants table.
- Duplicate entries for the same event are prevented.

**Priority:**

- High

#### 4.2.1.2 Stimulus/Response Sequences

1. After successful login, participant selects an event and clicks "Join Event."
2. System verifies that the participant has not already joined the event.
3. System inserts a new record in the event_participants table linking participant and event.
4. Participant receives confirmation that they have successfully joined the event.

#### 4.2.1.3 Functional Requirements

REQ-1: System must allow participants to join events.
REQ-2: System must verify that a participant has not already joined the event.
REQ-3: System must insert participant and event details into event_participants.
REQ-4: System must display a success message after joining.

### 4.2.2 Leave Event

#### 4.2.2.1 Description and Priority

**Description:**

- Participants can leave an event they have previously joined.
- The system removes the participant's record from the event_participants table.

**Priority:**

- Medium

#### 4.2.2.2 Stimulus/Response Sequences

1. Participant selects an event from the "My Events" list and clicks "Leave Event."
2. System verifies that the participant is currently enrolled in the event.
3. System deletes the record linking the participant and event from the event_participants table.
4. Participant receives confirmation that they have left the event.

#### 4.2.2.3 Functional Requirements

REQ-1: System must allow participants to leave any event they have joined.
REQ-2: System must verify that the participant is enrolled before deletion.
REQ-3: System must remove the participant-event record from event_participants.
REQ-4: System must display confirmation after successful removal.

### 4.2.3 View Joined Events

#### 4.2.3.1 Description and Priority

**Description:**

- Participants can view a list of all events they have joined.
- The system retrieves relevant event information such as title, date, time, and host.

**Priority:**

- High

#### 4.2.3.2 Stimulus/Response Sequences

1. Participant navigates to the “My Events” page.
2. System queries the event_participants and events tables for events associated with the participant.
3. System displays all joined events with relevant event details.

#### 4.2.3.3 Functional Requirements

REQ-1: System must retrieve all events linked to the participant’s ID.
REQ-2: System must display each event’s title, date, time, location, and host.
REQ-3: System must update the displayed list when participants join or leave events.

### 4.2.4 View Available Events

#### 4.2.4.1 Description and Priority

**Description:**

- Participants can view all events that they have not yet joined.
- The system filters out events that already exist in the participant’s joined list.

**Priority:**

- Medium

#### 4.2.4.2 Stimulus/Response Sequences

1. Participant accesses the “Events” page.
2. System retrieves all events from the events table.
3. System filters out any events already joined by the user.
4. System displays the remaining available events for participation.

#### 4.2.4.3 Functional Requirements

REQ-1: System must retrieve all events from the events table.
REQ-2: System must filter out events the participant has already joined.
REQ-3: System must display all remaining available events with essential details.

### 4.2.5 View Group Chats

#### 4.2.5.1 Description and Priority

**Description:**

- Participants can access group chats associated with events they have joined.
- Group chats allow members of the same event to communicate in real time.

**Priority:**

- High

#### 4.2.5.2 Stimulus/Response Sequences

1. Participant navigates to “My Events” and selects “Group Chat.”
2. System verifies that the participant is a member of the event.
3. System retrieves messages from the messages table linked to the event’s chat_id.
4. Participant can view and send messages within the chatroom.

#### 4.2.5.3 Functional Requirements

REQ-1: System must retrieve group chats linked to joined events.
REQ-2: System must allow participants to send and receive messages in real time.
REQ-3: System must restrict chat access to event participants only.
REQ-4: System must update chatrooms with new messages dynamically.

### 4.2.6 View Recommended Events

#### 4.2.6.1 Description and Priority

**Description:**

- The system displays a list of recommended events based on category.

**Priority:**

- Medium.

#### 4.2.6.2 Stimulus/Response Sequences

1. Participant visits the “Events” page.
2. System queries the events table, sorting by category.
3. System displays the recommended events list.

#### 4.2.6.3 Functional Requirements

REQ-1: System must retrieve event data from the events table.
REQ-2: System must sort events by popularity or category before displaying.
REQ-3: System must display recommended events on the main events page.

### 4.2.7 Follow Users

#### 4.2.7.1 Description and Priority

**Description:**

- Participants can follow or unfollow other users on the platform.
- The system records relationships between users in the follows table.

**Priority:**

- Medium

#### 4.2.7.2 Stimulus/Response Sequences

1. Participant views another user’s profile and selects “Follow.”
2. System records the follow relationship in the follows table.
3. Participant can later select “Unfollow” to remove the relationship.
4. System updates both follower and following lists accordingly.

#### 4.2.7.3 Functional Requirements

REQ-1: System must allow participants to follow and unfollow other users.
REQ-2: System must record relationships in the follows table.
REQ-3: System must prevent duplicate follow relationships.
REQ-4: System must update and display follower/following counts dynamically.

## 4.3 Host Features

### 4.3.1 Create Event

#### 4.3.1.1 Description and Priority

**Description:**

- Hosts can create events by providing details such as title, description, category, date, time, location, and capacity.
- Created events are submitted in a Pending state and require Admin approval before becoming visible to participants.

**Priority**

- High

#### 4.3.1.2 Stimulus/Response Sequences

1. Host clicks "Create Event" and is prompted to enter event details (title, description, category, date, time, location, capacity).
2. Host optionally uploads an event image/banner.
3. Host submits the event for creation.
4. System validates all required fields and date/time formatting.
5. System creates the event in Pending state and stores it in the database.
6. Admin reviews the event via ReviewEvent.
7. If approved, system updates event status to Active and makes it visible to participants.
8. If rejected, event is not visible to participants and the Host can see the rejection reason in the MyEvents page.
9. Host receives confirmation of event creation and current status.

#### 4.3.1.3 Functional Requirements

REQ-1: System must allow Hosts to input all required event details (title, description, category, date, time, location, capacity).
REQ-2: System must validate that all mandatory fields are filled before submission.
REQ-3: System must validate date/time to ensure they are not in the past and are properly formatted.
REQ-4: System must create events in Pending state awaiting Admin approval.
REQ-5: System must notify Host of approval or rejection status.
REQ-6: System must display appropriate error messages for missing fields or invalid data.

### 4.3.2 Manage Participants

#### 4.3.2.1 Description and Priority

**Description:**

- Hosts can view and manage participants enrolled in their events.
- This includes viewing participant lists and removing participants when necessary.

**Priority**

- High

#### 4.3.2.2 Stimulus/Response Sequences

1. Host selects their event's groupchat from the GroupChats Page.
2. Host opens the View Members tab to view enrolled participants.
3. System displays list of current participants with their details.
4. Host reviews participant information.
5. If needed, Host selects a participant and clicks "Remove Participant."
6. System removes the participant record from event_participants table.
7. System updates the participant list display.

#### 4.3.2.3 Functional Requirements

REQ-1: System must display complete list of participants for each Host's event.
REQ-2: System must allow Hosts to remove participants from their events.
REQ-3: System must verify Host owns the event before allowing participant management.
REQ-4: System must update event_participants table when participants are removed.
REQ-5: System must enforce participant capacity limits and display warning when capacity is reached.
REQ-6: System must update participant count dynamically after additions or removals.

### 4.3.3 Edit/Cancel Events

#### 4.3.3.1 Description and Priority

**Description:**

- Hosts can edit event details or cancel events they have created.

**Priority**

- High

#### 4.3.3.2 Stimulus/Response Sequences

1. Host selects an event from their event list.
2. Host clicks either "Edit Event" or "Cancel Event.
3. If Edit:
   - 3.1. System displays editable fields with current event information.
   - 3.2. Host updates desired fields (title, description, date, time, location, capacity).
   - 3.3. Host confirms changes.
   - 3.4. System validates updated information.
   - 3.5. System updates event record in database.
4. If Cancel:
   - 4.1. System prompts Host to confirm cancellation.
   - 4.2. Host confirms cancellation decision.
   - 4.3. System updates event status to Cancelled.
   - 4.4. System closes associated group chat.
5. Host receives confirmation of successful update or cancellation.

#### 4.3.3.3 Functional Requirements

REQ-1: System must allow Hosts to edit all event details except past event dates.
REQ-2: System must validate edited information to ensure data integrity.
REQ-3: System must prevent setting event dates in the past.
REQ-4: System must allow Hosts to cancel their events with confirmation.

REQ-5: System must update event status to Cancelled upon cancellation.
REQ-6: System must automatically close group chats when events are cancelled.
REQ-7: System must maintain data consistency during updates and cancellations.
REQ-8: System must display appropriate error messages for invalid edits or database errors.

## 4.4 Admin Features

### 4.4.1 Remove Users

#### 4.4.1.1 Description and Priority

**Description:**

- Admins can remove users from the platform by banning their accounts for a specified duration.

**Priority:**

- High

#### 4.4.1.2 Stimulus/Response Sequences

1. Admin navigates to Review Users section.
2. Admin can
   2.1. see flagged users and choose the Ban User option.
   2.2. input the username of the user to be banned.
      2.2.1. System verifies username exists in the database.
      2.2.2. Admin clicks Ban User.
3. Admin inputs the duration of the ban (e.g., days, weeks, permanent).
4. Admin clicks "Confirm" to proceed with the ban.
5. System updates user account status to Banned in the database.
6. System records ban duration and banned email address.
7. Admin receives confirmation that user has been successfully banned.

#### 4.4.1.3 Functional Requirements

REQ-1: System must display all flagged users at the top of the list of users
REQ-2: System must allow Admins to search and select users by username.
REQ-3: System must verify that the username exists before proceeding with ban.
REQ-4: System must allow Admins to specify ban duration (temporary or permanent).
REQ-5: System must update user account status to Banned upon confirmation.
REQ-6: System must allow Admins to cancel the ban action before confirmation.
REQ-7: System must display confirmation message after successful ban.
REQ-8: System must log all ban actions with timestamps and Admin username for audit purposes.

### 4.4.2 Review Events

#### 4.4.2.1 Description and Priority

**Description:**

- Admins review event creation requests submitted by Hosts and decide whether to approve or reject.

**Priority:**

- High

#### 4.4.2.2 Stimulus/Response Sequences

1. System displays list of pending event creation requests to Admin.
2. Admin selects an event request to review.

3. System displays complete event details including title, description, category, date, time, location, and capacity.
4. Admin reviews the event information for appropriateness and completeness.
5. Admin selects one of two actions:
    5.1. Approve: System updates event status to Active, makes it visible to participants, and notifies the Host of approval.
    5.2. Reject: System updates event status to Rejected, notifies Host with rejection reason.
6. Admin receives confirmation that the action has been processed.

#### 4.4.2.3 Functional Requirements

REQ-1: System must display all pending event creation requests to Admins.
REQ-2: System must show complete event details for Admin review.
REQ-3: System must allow Admins to approve event requests.
REQ-4: System must allow Admins to reject event requests with a reason.
REQ-5: System must update event status based on Admin's decision (Approved or Rejected).
REQ-6: System must make approved events visible to participants.

### 4.4.3 Moderate Events

#### 4.4.3.1 Description and Priority

**Description:**

- Admins review and moderate flagged events that may violate platform guidelines, contain inappropriate content, promote illegal activities, or have misleading information. Admins can remove problematic events or mark them as reviewed.

**Priority:**

- High

#### 4.4.3.2 Stimulus/Response Sequences

1. System displays list of flagged events to Admin.
2. Admin selects an event for moderation review.
3. System displays event details including description, schedule, user reports, and flagged content indicators.
4. Admin reviews the event content against platform guidelines.
5. Admin selects one of two actions:
    5.1. Remove Event: System deletes event from platform, closes associated group chat.
    5.2. Keep Event: System updates event status to "Reviewed," clears flags.
6. System logs moderation action with timestamp and Admin username.
7. Admin receives confirmation that moderation action has been completed.

#### 4.4.3.3 Functional Requirements

REQ-1: System must display all flagged events to Admins.
REQ-2: System must show complete event details, user reports, and flagged content for review.
REQ-3: System must allow Admins to remove events that violate guidelines.
REQ-4: System must allow Admins to mark events as "Reviewed" and keep them active.

REQ-5: System must delete event listings from the platform when removed.
REQ-6: System must close associated group chats when events are removed.
REQ-7: System must log all moderation actions with timestamps and Admin username for audit trail.
REQ-8: System must clear flags from events marked as "Reviewed."

## 4.5 Group Chat Features

### 4.5.1 Group Chat Management

#### 4.5.1.1 Description and Priority

**Description:**

- Manages the complete lifecycle of event-based group chats, including automatic creation, membership management, and closure. Group chats are created when events are approved, updated as participants join or leave, and closed after event cancellation.

**Priority:**

- High

#### 4.5.1.2 Stimulus/Response Sequences

1. Host creates an event and Admin approves it via ReviewEvents.
2. System automatically generates a group chat linked to the event.
3. System assigns unique group chat ID and associates it with the event.
4. When a participant joins the event, system adds them to the group chat.
5. When a participant leaves the event, system removes them from the group chat.
6. When Host removes a participant via ManageParticipants, system removes them from group chat.
7. System monitors event status for closure conditions.
8. When event is cancelled, system closes the group chat.

#### 4.5.1.3 Functional Requirements

REQ-1: System must automatically create a group chat when an event is approved.
REQ-2: System must link each group chat to its corresponding event with a unique identifier.
REQ-3: System must add participants to group chat when they join an event.
REQ-4: System must remove participants from group chat when they leave an event.
REQ-5: System must remove participants from group chat when Host removes them from event.
REQ-6: System must support real-time messaging between all chat members.
REQ-7: System must automatically close group chats after event cancellation.

### 4.5.2 Create Group Chat

#### 4.5.2.1 Description and Priority

**Description:**

- Automatically creates a unique group chat for each newly approved event, establishing the communication channel for the Host and future participants.

**Priority:**

- High

#### 4.5.2.2 Stimulus/Response Sequences

1. Host creates an event through CreateEvent use case.
2. Admin approves the event via ReviewEvents.
3. System generates a unique group chat identifier.
4. System creates a new group chat instance in the chat service.

5. System links the group chat to the event in the database.
6. System adds the Host as the first member of the group chat.
7. Group chat becomes available for participant messaging.

#### 4.5.2.3 Functional Requirements

REQ-1: System must automatically create a group chat upon event approval.
REQ-2: System must generate a unique group chat ID for each event.
REQ-3: System must link the group chat to its corresponding event in the database.
REQ-4: System must ensure each event has exactly one unique group chat.
REQ-5: System must add the Host as the initial group chat member.
REQ-6: System must integrate group chat creation with the event service.
REQ-7: System must retry group chat creation automatically if initial attempt fails.

### 4.5.3 Manage Group Chat Membership

#### 4.5.3.1 Description and Priority

**Description:**

- Controls the addition and removal of users from event group chats based on participant join/leave actions and Host management decisions, ensuring chat membership accurately reflects event enrollment.

**Priority:**

- High

#### 4.5.3.2 Stimulus/Response Sequences

1. Participant joins an event through JoinEvent use case.
2. System retrieves the associated group chat ID for the event.
3. System adds the participant to the group chat membership list.
4. Participant gains access to view and send messages in the group chat.
5. System updates chat member count in real time.
6. When participant leaves:
   - 6.1. Participant uses LeaveEvent functionality.
   - 6.2. System removes participant from group chat membership.
   - 6.3. Participant loses access to the group chat.
7. When Host removes participant:
   - 7.1. Host removes participant via ManageParticipants.
   - 7.2. System removes participant from group chat.
8. System logs all membership changes for auditing purposes.

#### 4.5.3.3 Functional Requirements

REQ-1: System must automatically add participants to group chat when they join an event.
REQ-2: System must automatically remove participants from group chat when they leave an event.
REQ-3: System must remove participants from group chat when Host removes them from event.
REQ-4: System must verify that event and group chat exist before updating membership.
REQ-5: System must update group chat membership in real time.

REQ-6: System must prevent duplicate membership entries for the same user.
REQ-7: System must log all membership changes with timestamps for audit trail.

### 4.5.4 Communicate in Group Chat

#### 4.5.4.1 Description and Priority

**Description:**

- Enables Hosts and Participants to exchange real-time text messages within event group chats, facilitating coordination and communication among event members.

**Priority:**

- High

#### 4.5.4.2 Stimulus/Response Sequences

1. User (Host or Participant) opens the group chat for their event.
2. System verifies user is a current member of the group chat.
3. System displays chat history and active members.
4. User types a text message.
5. User sends the message.
6. System transmits message to chat service.
7. System delivers message to all group chat members in real time.
8. All members see the message appear in their chat interface.
9. If message fails to send, system automatically retries transmission.

#### 4.5.4.3 Functional Requirements

REQ-1: System must allow Hosts and Participants to send messages in group chats.
REQ-2: System must verify user is a member of the group chat before allowing messaging.
REQ-3: System must deliver messages to all chat members in real time.
REQ-4: System must display messages chronologically in the chat interface.
REQ-5: System must show chat history to members when they open the chat.
REQ-6: System must retry message transmission automatically if initial send fails.
REQ-7: System must prevent non-members from accessing or sending messages to the group chat.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

1. Each page, including event discovery, user authentication, and profile loading, should load in no more than 3 seconds under normal network conditions.

2. Synchronization conflicts such as two or more users attempting to join the same event simultaneously shall be limited to fewer than 100 occurrences per month.

3. The HobbyHive web application should be scalable to support at least 300,000 concurrent users while maintaining optimal performance and responsiveness.

4. The error rate for actions such as event creation, joining/leaving a group, or updating profiles should remain below 1 percent.

5. The system must support real-time communication (chats) with minimal latency of under 2 seconds per message transmission.

## 5.2 Safety Requirements

1. The system shall not enforce strict time limits for event participation or group communication to ensure a safe and non-stressful user experience.

2. The platform shall allow hosts to clearly state event requirements, age suitability, and safety disclaimers, enabling users to make informed participation decisions.

3. The platform shall not disclose users' personal contact information (phone numbers, emails, or addresses) in group chats or event listings, ensuring participant safety.

4. HobbyHive shall include report and block functions to allow users to respond to harassment, inappropriate behavior, or unsafe content.

## 5.3 Security Requirements

1. All personal data (such as email addresses, passwords, and location data) shall be stored securely in compliance with Singapore's PDPA (Personal Data Protection Act).

2. User authentication shall be required for all interactions involving profile access, event participation, messaging, or creation of new listings.

3. Passwords shall be hashed and salted before being stored in the database to protect against unauthorized access.

4. Access control mechanisms will restrict data visibility to only authorized users

5. The database will be secured with role-based access control (RBAC) and limited to approved developers and system administrators.

6. The system must ensure zero tolerance for data leaks or unauthorized disclosure of user-generated content.

## 5.4 Software Quality Attributes

1. Source code shall be well-commented and follow clean coding practices to ensure readability, reusability, and maintainability.

2. All API endpoints implemented by the backend server must include documentation specifying the endpoint URL, required request body, and expected response structure.

3. The system shall display clear success and error messages for user actions (e.g., event creation, or chat message failure) to enhance usability and testability.

4. The web application shall maintain cross-platform compatibility across desktop browsers (Chrome, Edge, Safari) and mobile devices, ensuring consistent UI/UX.

5. To ensure reliability, the application shall fail to start or crash less than 1 percent of the time.

6. The system shall support continuous integration and deployment (CI/CD) for easy updates and maintenance without disrupting user activity.

## 5.5 Business Rules

The following business rules define the operational boundaries and functional permissions of the HobbyHive platform, ensuring that the system operates ethically and transparently.

### 5.5.1 Review and Rating System

1. Users may leave reviews and ratings for events or hosts only after attending the event.

2. All reviews must comply with community guidelines and will be subject to moderation; inappropriate or offensive content may be removed.

3. Hosts may flag reviews they deem inappropriate, which will be reviewed by an administrator.

### 5.5.2 Transactional Integrity

1. All system transactions, including event creation, membership requests, and chat messages, shall be logged and time-stamped for traceability.

2. Changes to events, groups, or profiles should be version-controlled, allowing administrators to audit and restore historical records if necessary.

### 5.5.3 Operational Hours

1. HobbyHive will operate 24/7 for user access.

2. Scheduled maintenance periods will be announced in advance and should not exceed 2 hours per occurrence.

# 6. Other Requirements

## 6.1 Internationalisation Requirements

1) Hobbyhive will initially support only the English language at launch. Future releases will explore multi-language support to cater to Singapore's diversity.

2) All data and time formats will follow Singapore Standard Time (GMT+8).

## 6.2 Legal Requirements

1) Hobbyhive shall comply with Singapore's Personal Data Protection Act (PDPA) when handling user-sensitive data.

2) Users must provide consent before any personal data is collected or stored.

## 6.3 Accessibility Requirements

1) The user interface shall maintain a readable contrast, clear and dynamic font sizes, and scalable elements to support different display sizes.

## 6.4 Reuse Objectives

1) UI components and modules should be modular and reusable across the platform to support future enhancements.

2) APIs shall follow REST API standards for maintainability to facilitate potential integration opportunities.

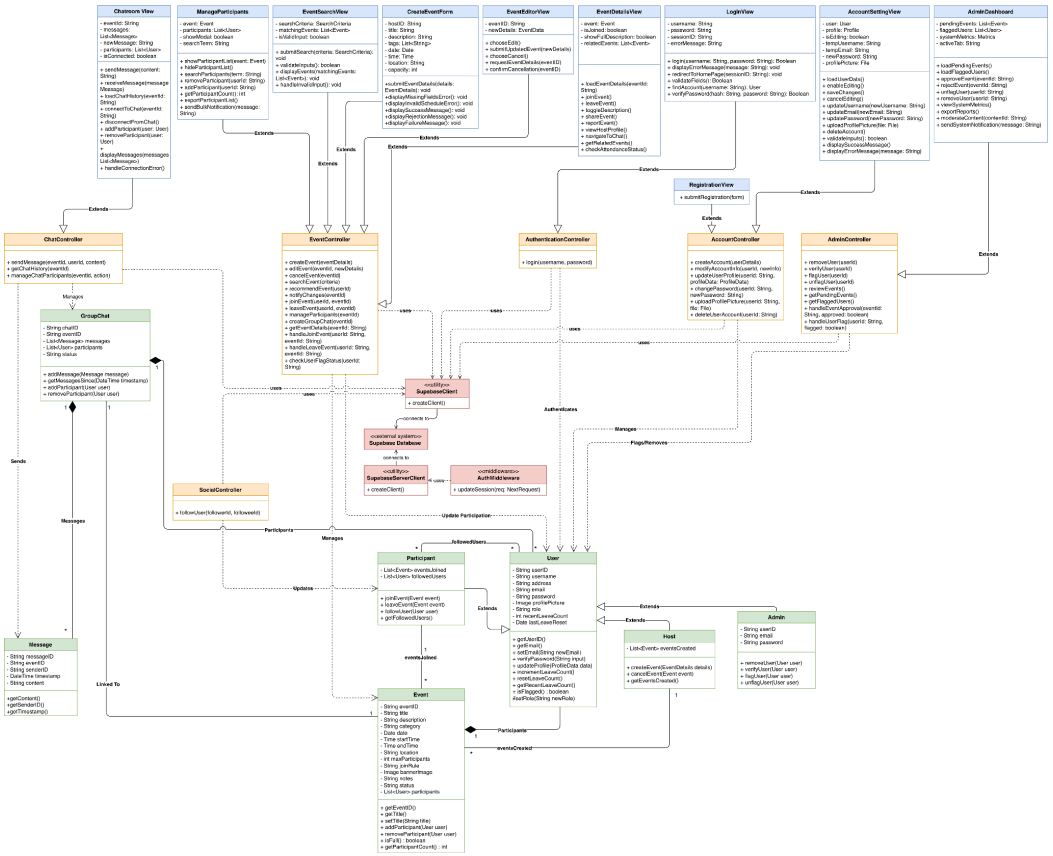
## 6.5 Security & Data Requirements

1) All authentication and data storage shall be securely managed by Supabase's services

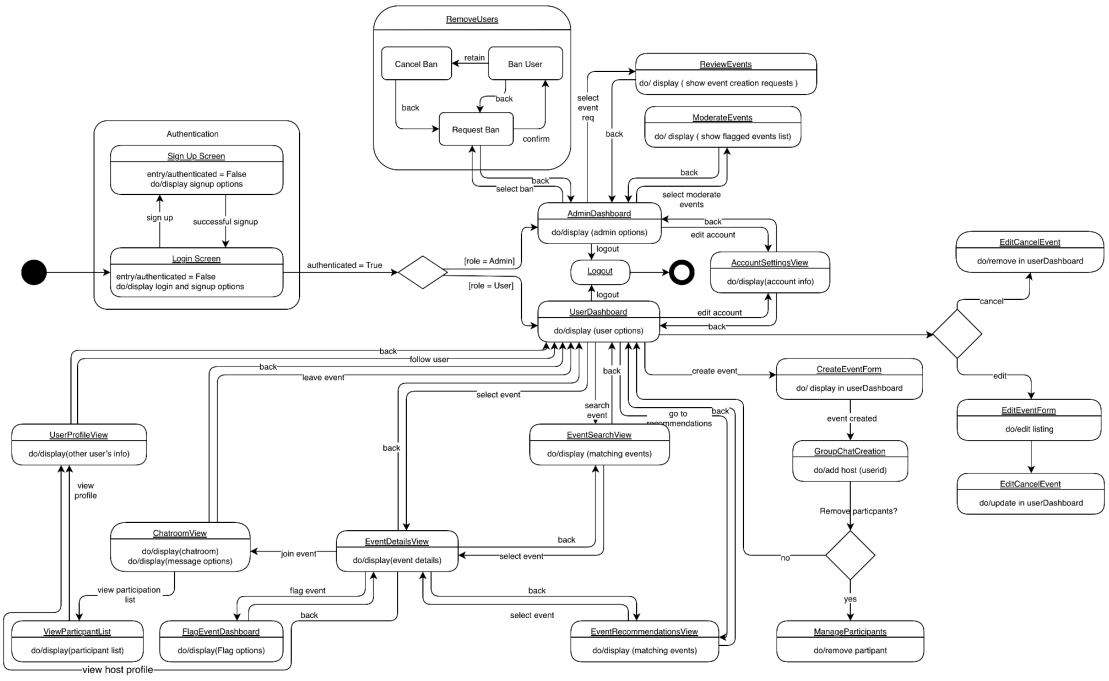
# Appendix A: Glossary

| **Acronym / Abbreviation** | **Meaning** |
|---|---|
| API | Application Programming Interface<br>- A set of protocols for building and interacting with software applications. |
| PDPA | Personal Data Protection Act<br>- Provides a baseline standard of protection for personal data in Singapore. |
| PostgreSQL | Postgre Structured Query Language<br>- It is a powerful open-source database system that stores, manages, and retrieves data using SQL.<br>- It supports both relational (tables, rows, columns) and non-relational (JSON) data for modern applications. |
| Front-end | The client-side part of a web application that users interact with directly<br>- It consists of everything the user experiences directly: from text and images to sliders and buttons. |
| Back-end | The server-side of a web application that handles database interactions, server logic, and application integration. |
| SRS | Software Requirements Specification<br>- A document that describes what the software will do and how it will be expected to perform.<br>- It includes a set of use cases that describe all the interactions the users will have with the software. |
| RBAC | Role Based Access Control<br>- It manages user permissions by assigning specific roles with defined privileges to users or groups.<br>- This allows for scalable, secure access management by granting users only the permissions they need to perform their tasks. |

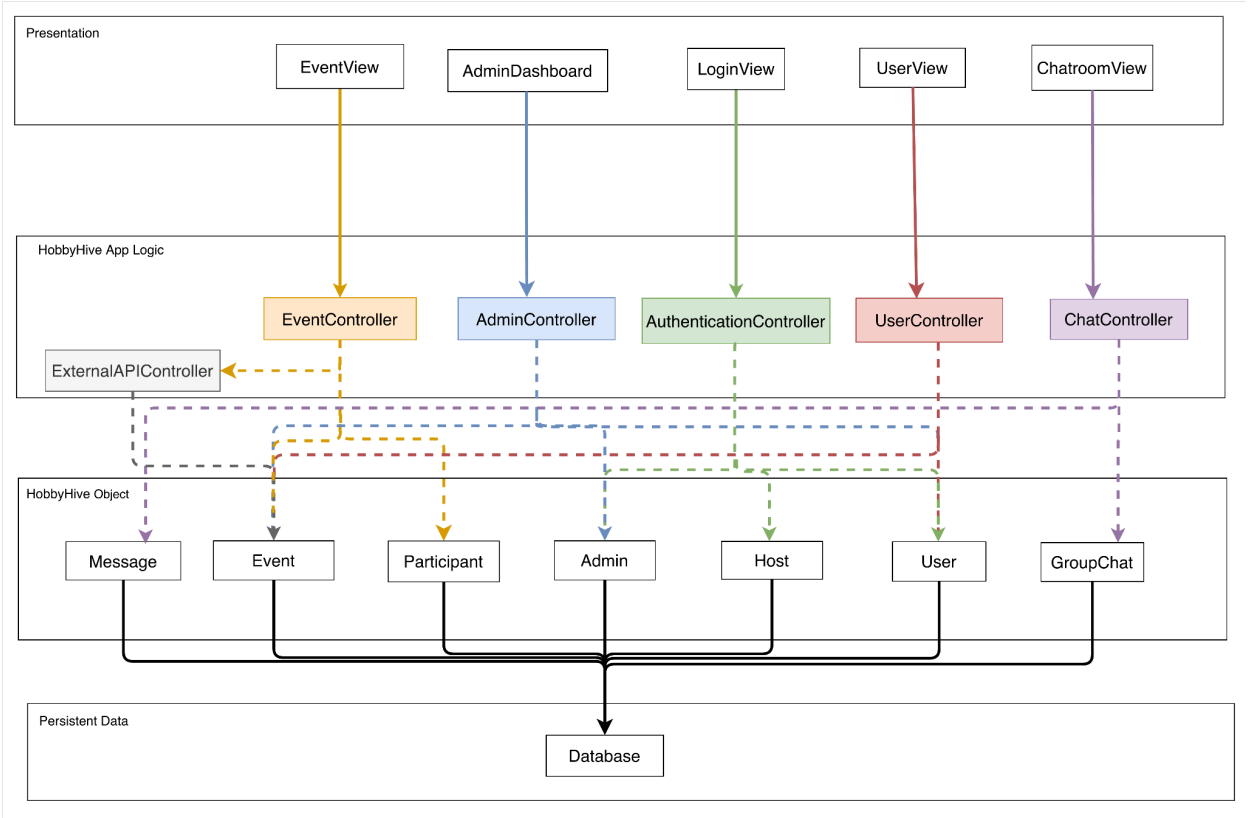
# Appendix B: Analysis Models



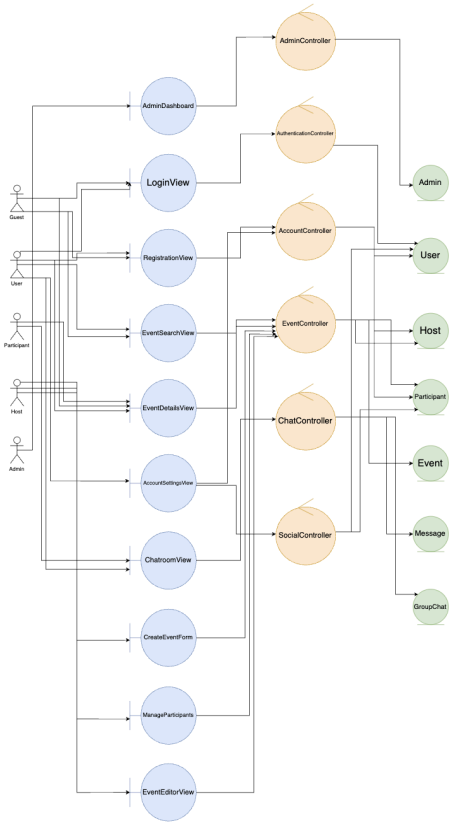
*Class Diagram*




*Dialog Map*




*System Architecture*




*Stereotype Class Diagram*

# Appendix C: To Be Determined List

There are no TBD items for this version of Software Requirement Specification.

# References

People's Association. (2023). Community Clubs (2024) [Dataset]. data.gov.sg. Retrieved November 4, 2025 from https://data.gov.sg/datasets/d_f706de1427279e61fe41e89e24d440fa/view

National Library Board. (2023). Libraries (2024) [Dataset]. data.gov.sg. Retrieved November 4, 2025 from https://data.gov.sg/datasets/d_27b8dae65d9ca1539e14d09578b17cbf/view

National Parks Board. (2023). Parks (2024) [Dataset]. data.gov.sg. Retrieved November 4, 2025 from https://data.gov.sg/datasets/d_0542d48f0991541706b58059381a6eca/view

Singapore Land Authority. (2023). OneMap Search API [Web API]. OneMap. Retrieved November 4, 2025 from https://www.onemap.gov.sg/docs

Singapore Land Authority. (2023). OneMap Nearest MRT API [Web API]. OneMap. Retrieved November 4, 2025 from https://www.onemap.gov.sg/docs

Source: http://www.frontiernet.net/~kwiegers/process_assets/srs_template.doc