SOFTWARE REQUIREMENTS SPECIFICATION FOR EVENTHUB

Software process and tools

RMIT University

Yusuf Kaan Yigiter, Christian Nabati, Anthony Vo, Thomas Thai Ta, Pee jay Cabale, Jensen Ly

Table of Contents

1. Introduction 2
1.1 Purpose 2
1.2 Document Conventions 2
1.3 Intended Audience 2
1.4 Product Scope 2
1.5 References 3
2. Overall Description 3
2.1 Product Perspective 3
2.2 Product functions 3
2.3 User Classes and Characteristics 4
2.4 Operating Environment 4
2.5 Design and Implementation Constraints 4
2.6 Assumptions and Dependencies 4
3. External Interface Requirements 4
3.1 User Interfaces 5
3.2 Hardware interfaces 5
3.3 Software Interfaces 5
3.4 Communications Interfaces 6
4. Nonfunctional Requirements 6
4.1 Performance Requirements 6
4.2 Safety Requirements 6
4.3 Security Requirements 7
4.4 Software Quality Attributes 7
4.5 Business Rules 7
5. System Architecture 7
5.1 Architecture Overview 9
5.2 Architectural Decisions 9
6. User Interface Design 11
6.1 Navigation flow of UI wireframe: 18
Appendix A: Glossary 18
Appendix B: Analysis Models 19
APPENDIX C: Definition of done 19
Appendix Milestone 2: 20

1. Introduction

1.1 Purpose

The Software Requirements Specification (SRS) aims to define foundational requirements for EventHub, a web-based event management that is to be developed as part of the course's Major Project. The product objective is to support university clubs and organisers in the event management process while enabling students to browse, RSVP and participate in activities across different categories. This SRS covers the principal system as underlined in the project brief, including event management, user participation, administrative functions and other optional extension features. By having this document, it serves as a reference for all relevant stakeholders (development team, product owner, clients), suggesting a thorough, clear and shared understanding regarding EventHub's objectives.

1.2 Document Conventions

The requirements mentioned in this SRS are expressed adopting the standard IEEE SRS conventions. All functional requirements are stated with a "shall" format whereas optional or non-functional requirements follow "should" or "may" where applicable. Major features are defined and elaborated through the provided user stories and their corresponding acceptance criteria. Priorities are deprived from the higher-level requirements, consulted and approved by Product Owner.

1.3 Intended Audience

This document is intended for a range of audiences:

- **Developers:** This document serves as a guideline for features/requirements implementation and verification of Event Hub.
- **Scrum Master:** This document is intended for the purpose of progress tracking, tasks allocations to ensure that the end-product closely aligns with the agreed scope.
- **Product Owner:** This document acts as a mutual agreement on the features, ensuring deliverables meet stakeholders' vision and expectations.
- **Users:** This document provides a detailed explanation on how the users' requirements are addressed in the system design.

1.4 Product Scope

EventHub is an all-in-one, state-of-the-art event management system that promotes student engagement by consolidating all event information into one single destination. The system's capabilities allow event organisers to create and manage events with detailed information while students can browse, filter, RSVP and generate a personalised dashboard for event recommendations based on interests. Administrators can manage and moderate both the users and events to ensure the integrity of the system. The system aims to greatly improved student engagement and participation as well as streamlining the process of coordinating and registering across a wide range of activities. Furthermore, extension features may be implemented including

but not limited to calendar/maps integration, QR code generation for check-ins, badge systems, to which line up with the project's vision of deploying an intuitive, innovative, scalable and user-centric solution.

1.5 References

- COSC2299 Major Project Assignment Specification, RMIT University, 2025.
- IEEE Std 830-1998, *IEEE Recommended Practice for Software Requirements Specifications*.

2. Overall Description

2.1 Product Perspective

EventHub is a new, self-contained web platform designed to provide opportunities for university students to be able to enjoy their university life. The platform will serve as an intermediate between club organisers and university students. The event organisers will be able to create, manage and publish their events, while students will have the option to browse events, RSVP to said events, and see a list of their upcoming RSVP'd events.

2.2 Product functions

FUNCTIONAL REQIUIREMENTS:
1/ The system shall allow the event organiser to create, edit, and delete any of their own events.

2/The system shall allow the students to browse a list of upcoming events on the homepage.

3/The user shall search and filter the event by date, and categories to find the event that satisfies them.

4/The user should RSVP to the event and receive email confirmation with all the necessary information.

5/The event organiser shall see the list of RSVP'd students in the specific event, so that they can mark who attended the event.

6/The club organiser shall have the dashboard to show the list of their events.

7/The system shall allow the student user to submit event feedback (rating and comments) after the event.

8/The system shall allow users to share event details (via link or social media pop up).

9/The system shall recommend the event based on the user's interests and categories.

10/The system shall allow club organisers to upload event photos to a gallery visible on the event page

11/The system shall provide a unique QR code as a check-in to an RSVP event.

12/The system shall export the CSV file of attendee list of specific events.

13/The admin shall manage all the events and users (deactivate / banning account).

14/The system shall have an interactive map where it will display the location of the event.

2.3 User Classes and Characteristics

The three main user Classes that will use our product (in different ways) will be University students, club organisers, and admins. The most important user class for EventHub would be the University students, since they provide a profit for the events.

Students: Highly frequent use of EventHub, to browse or RSVP to events.

Club organisers: Moderately frequent use of EventHub, to create, manage, and publish events periodically.

System Administrators: Least frequent use of EventHub, only when managing and/or deleting inappropriate events or banning users.

2.4 Operating Environment

The software must be able to operate under windows, mac, or Linux in non-obsolete versions. The system should be able to scale appropriately for mobile devices like android and apple. The software must be able to operate under internet browsers, such as google chrome or edge.

2.5 Design and Implementation Constraints

The development team are restricted to only using Springboot and java for the backend technology and databases must be a RDBMS.

2.6 Assumptions and Dependencies

The following are a list of assumptions:

- Users will have access to the internet in order to use EventHub.
- Users will have access to internet browsers.

The following are a list of dependencies:

- The system depends on the reliability of the database management system for event and user data storage.
- The system relies on properly rendering across web browsers to ensure consistent functionality.

3. External Interface Requirements

3.1 User Interfaces

The user interface is designed at the wireframe stage and will act as support during the development phase. The system will mainly support two primary users:

1. Normal users: browsing, searching, and registering for events.
2. Event organisers: create and manage events

Logical Characteristics:

Consistent layout: the navigation bar and footer and standardized across all pages.

Planned features:

- Inline validation messages for all forms (Login, Sign-up, Contact, Create event).
- Search bar and filters on normal user's homepage.

UI components included

- Normal User Homepage
- Event Organiser Homepage
- Contact Page
- Login Page
- Sign-Up Page
- Create Event Page
- About Us Page
- Event organisers list of events page

Admin interface decision:

A separate admin dashboard page will not be developed for the initial version, all admin tasks such as data moderation, event approval, and user management will be handled manually via the backend or by editing the database directly. This decision will simplify development, and a dashboard interface may be added in future versions.

3.2 Hardware interfaces

Logical characteristics: runs on modern devices and browsers with internet access.
Support devices: desktop and laptops and mobile devices. (Windows, macOS, linux , ios and android)
Input methods support keyboard, mouse, and touchscreen inputs.
Communication protocols: Secure HTTPS protocol for all client server

3.3 Software Interfaces

- The application is built using Springboot for the backend and Thymeleaf for server-side rendering in the frontend.

Frontend:

- Thymeleaf for server-side rendering for HTML elements.
- CSS and JavaScript

Backend:

- Springboot: will handle most business logic

Database:

Our database would be using MySQL

Build tool: Maven, CI/CD tool (github action), Testing framework (JUnit5)

3.4 Communications Interfaces

The system will support secure and efficient communication between spring boot server and client browser.

Protocols & formats:

HTTPS with TLS encryption for all traffic.

HTTP/1.1 or HTTP/2 for browser compatibility

HTML for server-side rendering of pages

Core features:

User login, signup, session management, create event, update events, browsing, and contact form submission.

4. Nonfunctional Requirements

4.1 Performance Requirements

1) The system shall maintain 99.5% uptime for the year:
The reason

2) Average page load time shall be 3 seconds under normal usage and ≤ 5 seconds during peak traffic (e.g., university club week).

3) For searching for the event in the search bar, the event should appear in less than 3 seconds.

4) The event should update the RSVP in real time (less than 2 seconds)

4.2 Safety Requirements

1) Sensitive actions (e.g., event deletion, user banning) shall require confirmation dialogs to prevent accidental loss or harm

2) Event and RSVP data shall be retained for at least 2 years for reporting purposes

4.3 Security Requirements

1) All user passwords shall be stored using salted hashing (e.g., bcrypt)

2) Organizer and admin shall have role-based access control (RBAC) to restrict permission according to their role

4.4 Software Quality Attributes

1) Portability: The web platform must be compatible with modern browsers (Chrome, Firefox, Edge, Safari) and responsive on mobile devices.

2) *Maintainability:* The system shall use monolithic architecture

3) *reliability: The system shall use MYSQL as the primary database*

4) *reliability: The system should use springboot as the main framework*

5) *Testability*: The system should support automated deployment pipeline (CI/CD) for efficient updates

4.5 Business Rules

1) Only registered university students may RSVP to events (email end with .edu.au)

2) The system shall prevent duplicate RSVPs for the same event by the same user.

3) Admins have authority to delete inappropriate events, deactivate users, and override bookings.
4) User may only provide 1 feedback entry per event attended

5) Suspended or banned accounts after login will be redirected to a warning page where they can't see any event and they only see the message that they are being banned

5. System Architecture

Presentation Layer (UI)

This is what the end-user interacts with.

- RSVP/Feedback UI: RSVP to events, check RSVP list, give ratings/comments.
- Authentication UI: Login, signup, and account/profile management.
- Event UI: Browse upcoming events, filter/search, and view details.

- Gallery UI: Displays uploaded event photos.
- Admin UI: Lets admins manage events and users.

Interaction:

- These UIs send HTTP requests to the Business Layer services.
- They receive structured responses (JSON) and render the data visually for the user.

Business Layer (Logic/Services)

Contains the core logic of the system.

- RSVP Service: Manages RSVP actions, attendee lists, and user-event relationships.
- Feedback Service: Saves/retrieves ratings and comments for events.
- Authentication Service: Validates credentials, manages sessions, and controls roles.
- Event Service: Handles CRUD (create, update, delete, view) operations for events.
- Gallery Service: Processes photo uploads, links them to events, and serves them.
- Search Engine: Generates event suggestions using RSVP history, tags, and interests and data from User preference repository).
- Admin Service: Lets admins manage users and events (ban, delete).

Interaction:

- Each service talks to its matching repository in the Persistence Layer.
- Services also interact with each other. For example:
    - RSVP Service: checks with Auth Service to confirm user identity.
    - Search Engine: pulls data from RSVP, Event repositories and User Preference Repo.

Persistence Layer (Repositories)

Acts as the data access layer (ports). It translates service calls into SQL queries.

- RSVP Repository: Reads/writes RSVP records (user ↔ event mapping).
- Feedback Repository: Saves/retrieves event ratings and comments.
- User Repository: Manages user accounts, credentials, and roles.
- Event Repository: Handles event table queries.
- Gallery Repository: Stores metadata of event photos.
- User preference repo: Keeps interest/tag data for suggestions.
- Audit Logs: Tracks admin actions for security/compliance.

Interaction:

- Repositories are only called by services (not directly by UI).
- They issue SQL queries against MySQL and return results.

Database

Have 2 separate server, but both use MySQL:

- One for most of the information like event information (excluding images), user information, user's preference, audit logs, etc....
- Another one only for saving images which feeds into Gallery Repository.

Dependencies

- SpringBoot (Framework)
- Thymeleaf (template engine)
- JAR
- Maven
- Argon (for encryption)

5.1 Architecture Overview

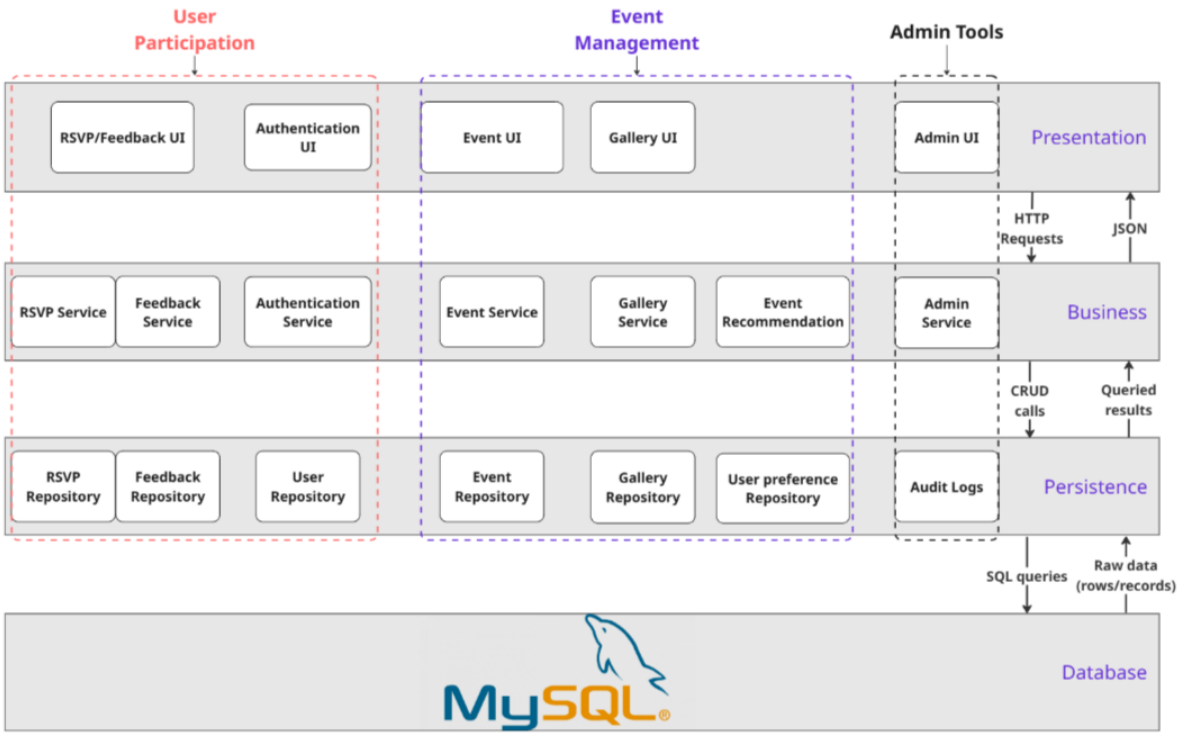

Figure 1: Software Architecture

5.2 Architectural Decisions

Decision 1: Using Layered Architecture

- **Rationale:** Using layered architecture ensures clean separation of concerns between the presentation, business, persistence and database layers. This drastically improves team collaboration, maintainability and testability. For EventHub, this allows the front-end developers to focus on the user interface without interfering with the backend developers managing attendance and event logic.

- **Risk Consideration:** The primary risk of using architecture is the abundance of potential overhead, which will decrease the application's performance since layered architecture will have multiple calls between layers. However, this is acceptable for EventHub's scale and is outtraded by the maintainability benefits.

Decision 2: Single/Centralized Database for Events and Users

- **Rationale:** Having a single database provides strong consistency (e.g., a student cannot attend a non-existent event) and simplifies query management.
- **Risk Consideration:** A single database can become a bottleneck as usage increases. If EventHub grows exponentially, we may consider using sharding or replication, but since this is only for a school project, we do not think that this will be a problem at all. For now, this is the most cost effective and reliable solution.

Decision 3: Having 2 separate servers for images and raw data.

- **Rationale:** Separating the database into 2 separate servers, one for almost all the data that is used for making the website functional, the other one solely for images. This separation reduces the downtime of the application overall. For EventHub, this will prevent the application from shutting down completely when the "Image server" is under maintenance, since the application can default to a placeholder image when the needed image is not accessible.
- **Risk Consideration:** Having 2 separate servers can increase the complexity of the application since "Business layer" now has to pull data from 2 different servers.

Decision 4: Web Based Frontend with Responsive Design

- **Rationale:** Our application is web based to ensure accessibility across multiple platforms (e.g., desktop, mobile) without requiring a separate native app. This aligns with student use case, where most users will access EventHub on laptops and phones;
- **Risk Consideration:** Solely relying on web app may limit offline use cases. Future iterations may add a mobile app, but this is not critical in the current scope.

Decision 5: Role-based access control

- **Rationale:** Using a role-based access control system makes it simpler to manage user's access, reducing probability of having an error. For EventHub, this means having a hierarchy (admin -> event organizer -> user) where administrators have access to almost, if not all features of the application. Event organizer will have less permissions than an administrator, but they will still be able to create or edit events and users will have the least permissions.
- **Risk Assessment:** A huge security risk will happen if an admin's account is compromised since they have access to most of the features and have the permission to do almost anything.

6. User Interface Design

The following wireframe diagrams are shown below:

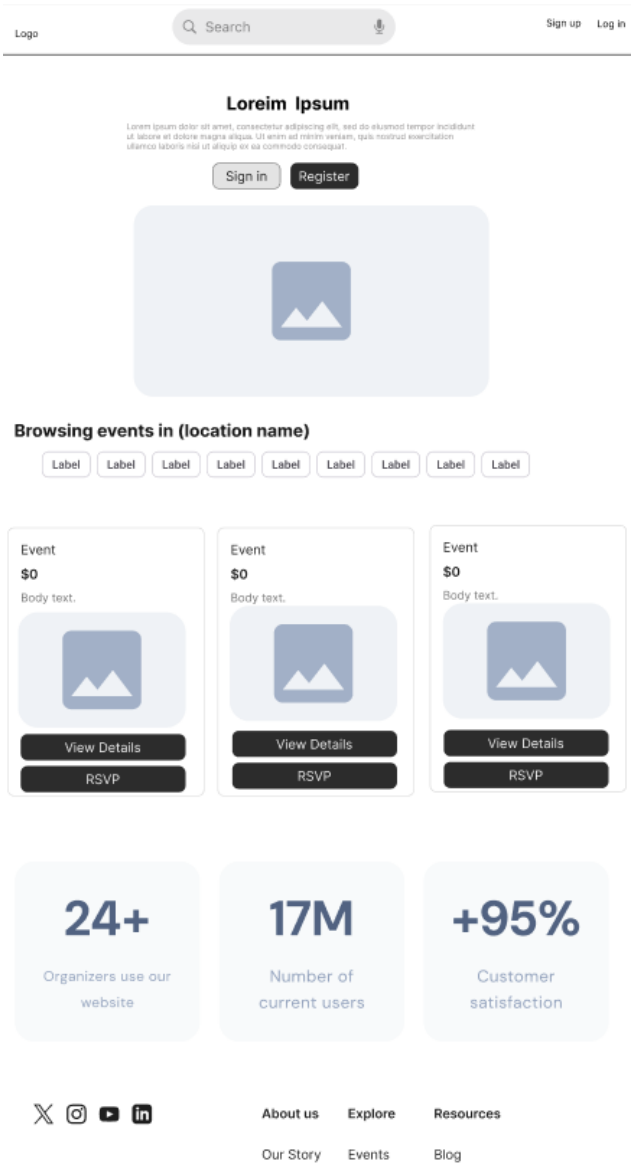

Figure 2: Browse Event Page updated to have RSVP button and Event details button

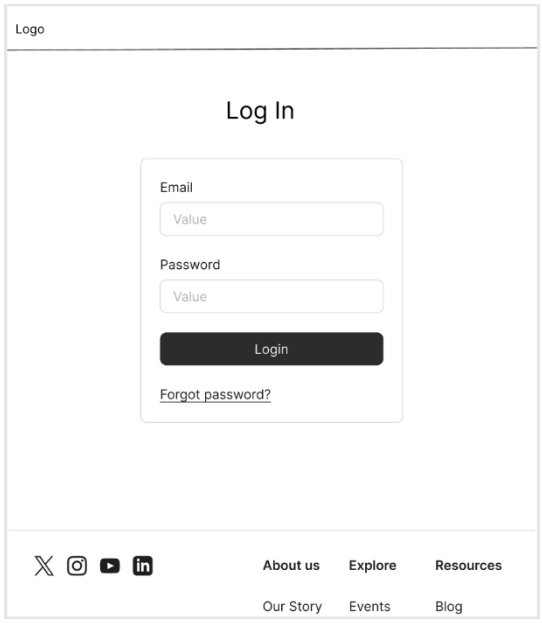

Figure 3: login page

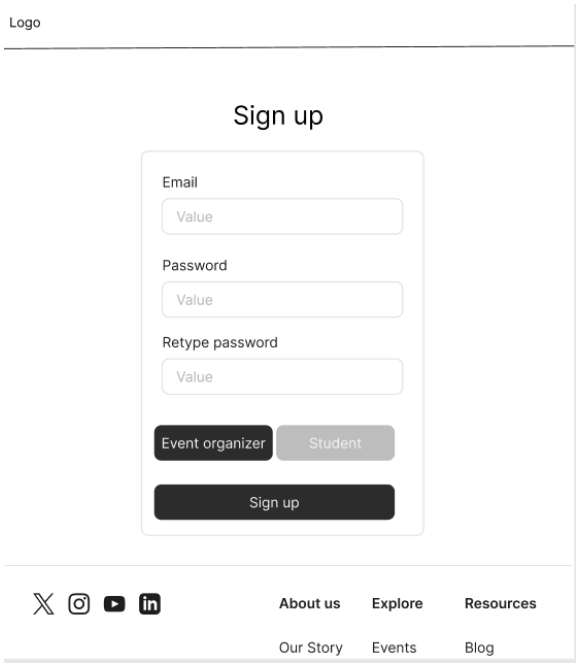

Figure 4: Sign up page

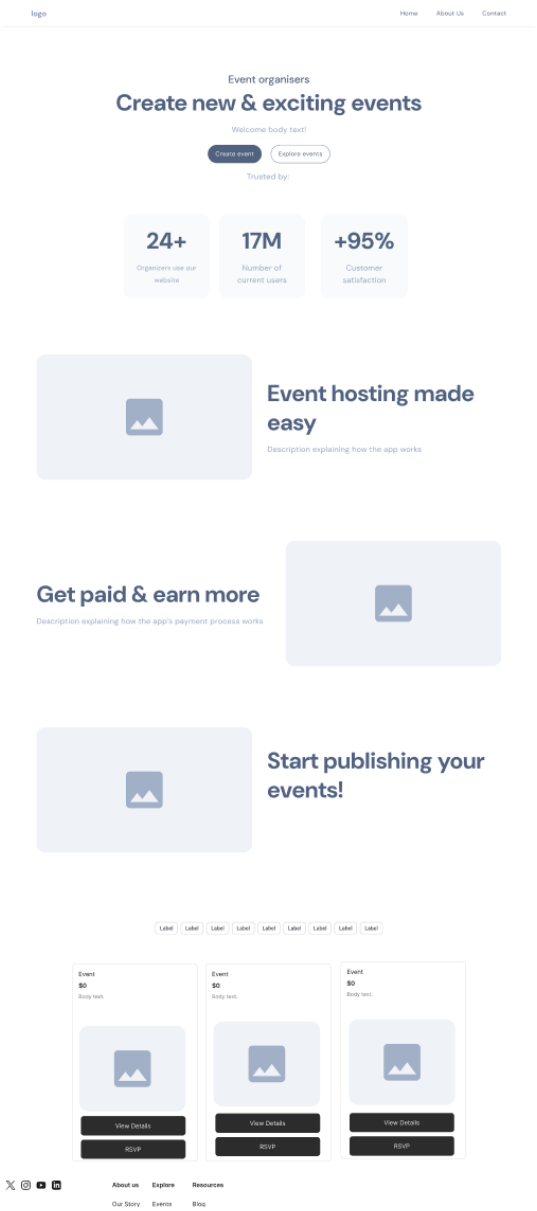

Figure 53: Organiser Dashboard Homepage (Merged the organiser dashboard(figure 10) and organiser home page into a single page)

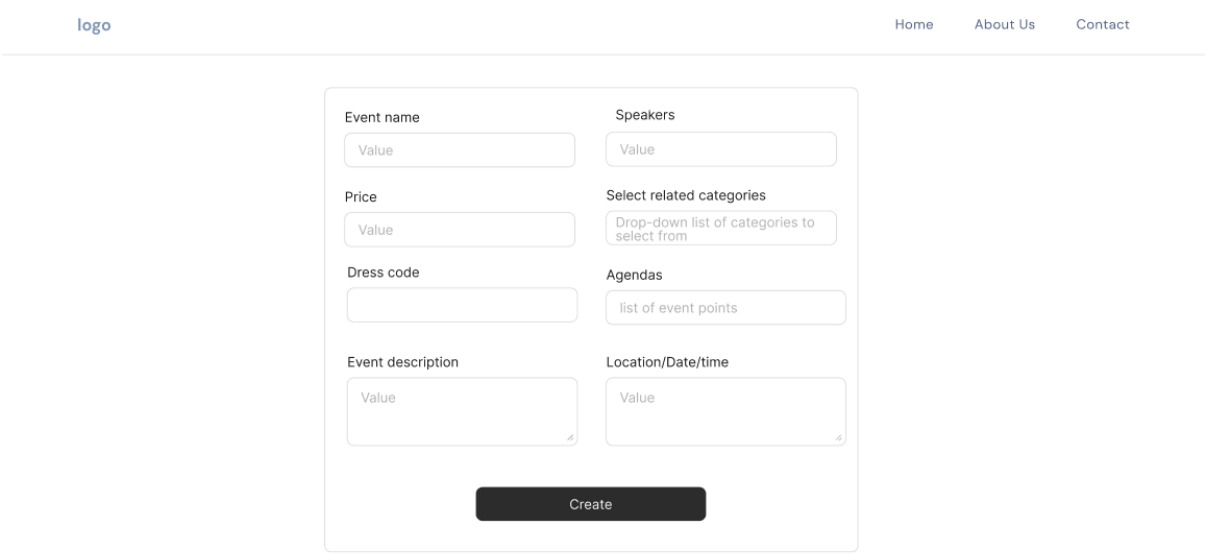

Figure 64: Create event page

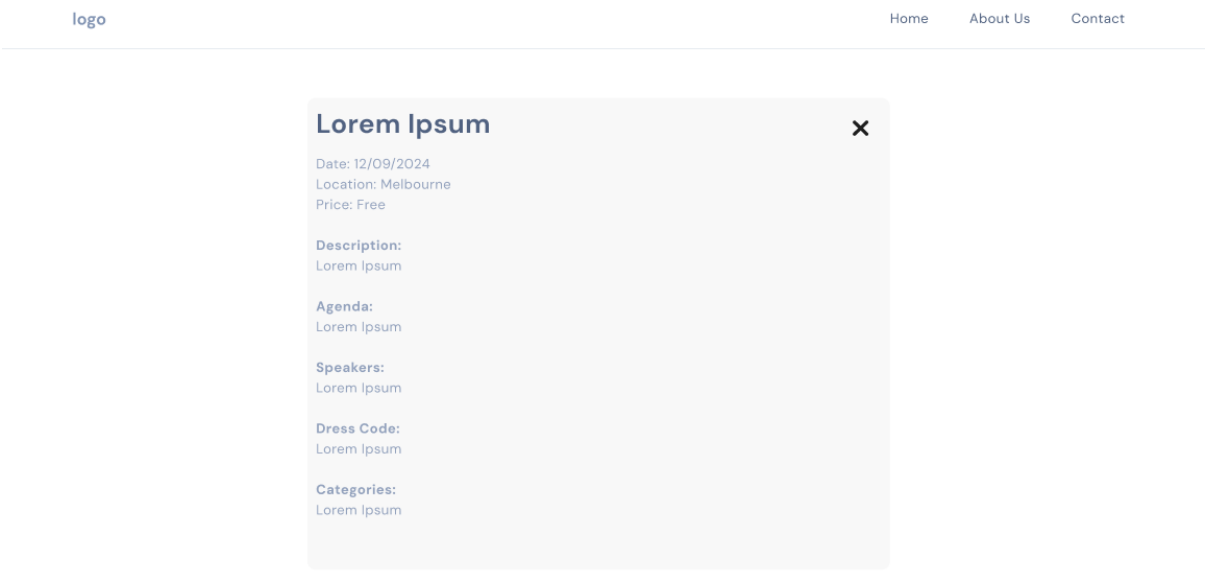

Figure 75: Specific event page (Changed into a pop-up window when user clicks on event details button)

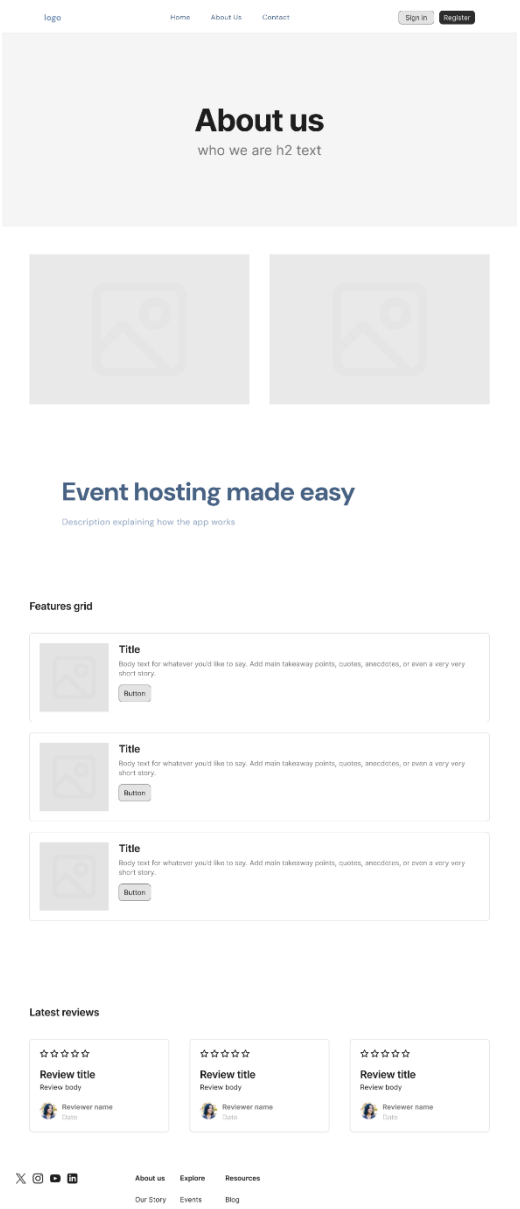

Figure 86: About us page

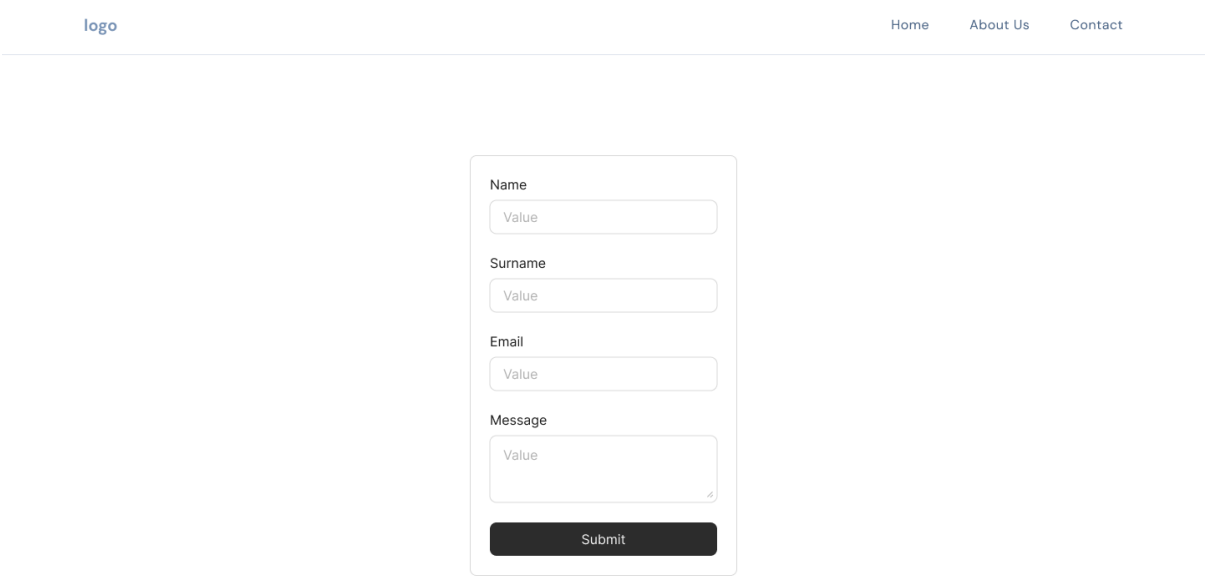

Figure 97: Contact Form

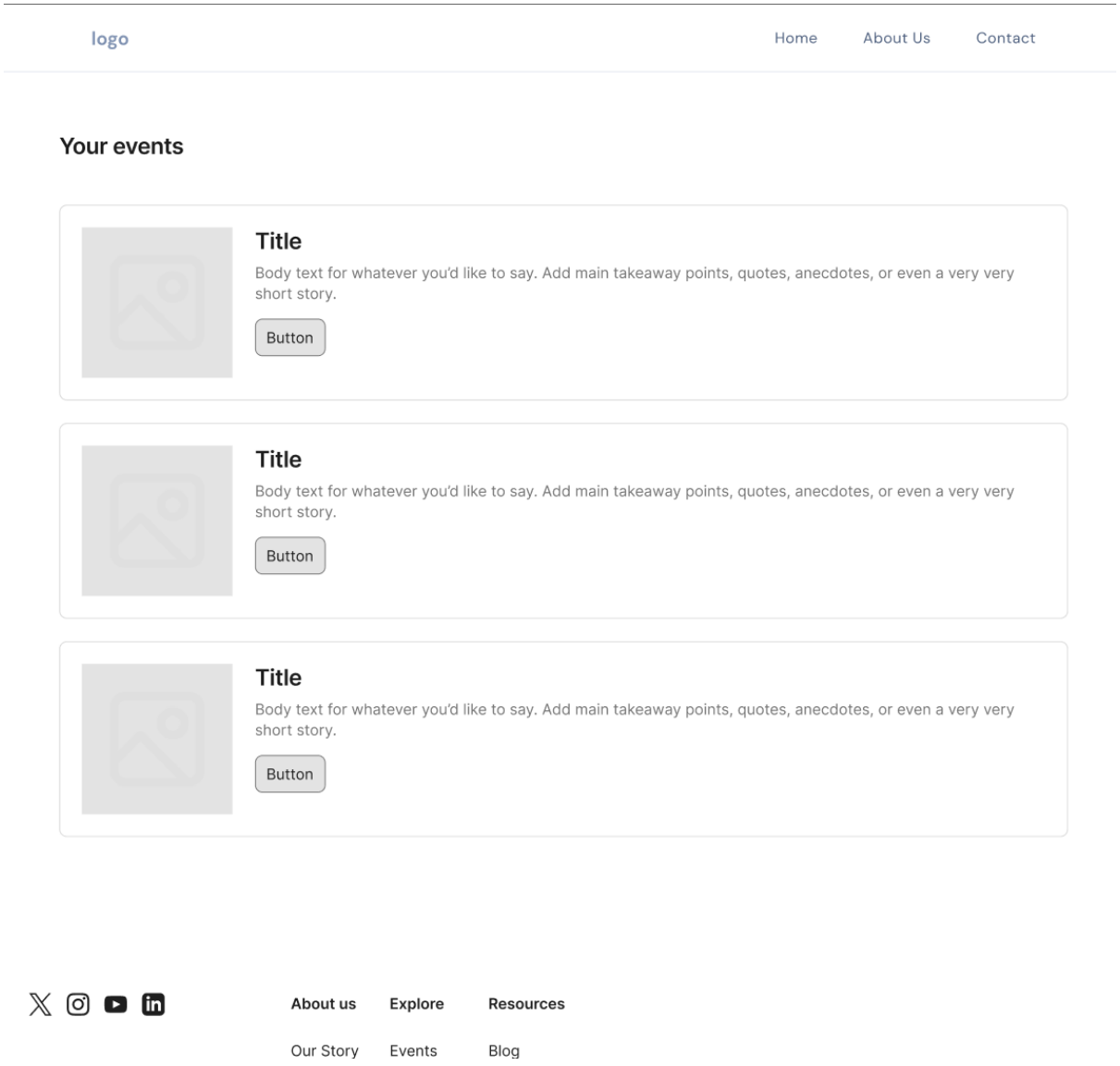

Figure 108: Event manager dashboard of list

Figure 911: The header when user logs in

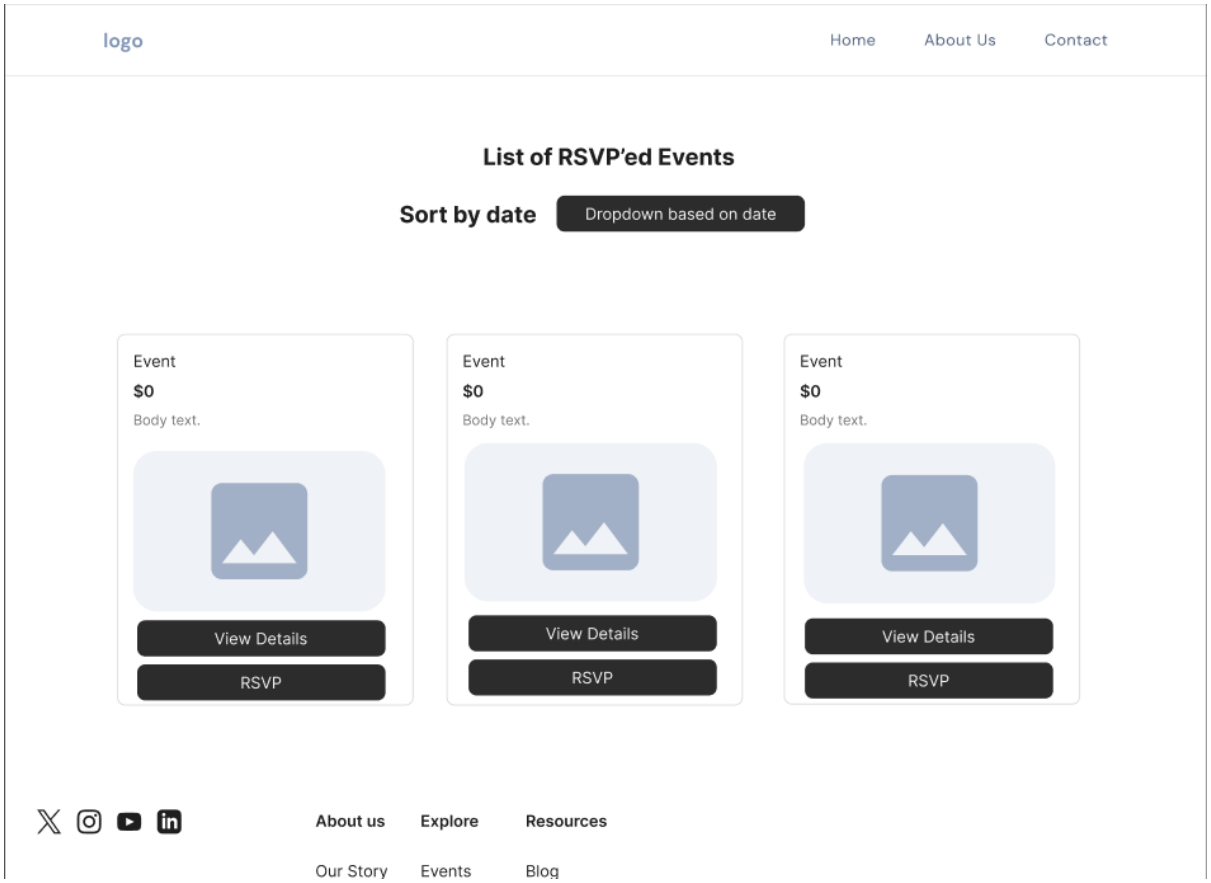

Figure 1102: My rsvp list page(showcases all RSVPed event of a user)

6.1 Navigation flow of UI wireframe:

The user is initially on the homepage of EventHub as shown on figure 2. The user clicks on the log-in button which redirects the user to the log-in page (figure 3) and enters their credentials. The user is then redirected to a specific page depending on whether the user is a student or a club orgainser. If the user is a student, they will be redirected to essentially the homepage but will have the ability to RSVP to events (figure 12). However, if the user is a club organiser, then they will be redirected to the club organiser dashboard page (figure 5). If the user is a student, then the user will click on RSVP button which will take them to the RSVP form. If the user is an organiser, then the user can click on the create event button and will redirect them to the create event page (figure 6).

Appendix A: Glossary

- Monolithic architecture: a single unified software application structure that is self-contained.
- Wireframe: A visual representation of the user interface.
- Frontend: everything the user either sees or interacts with in a website.
- Backend: data management and processing logic.
- RDBMS: a software system that stores and organizes data in a relational database format.

Appendix B: Analysis Models

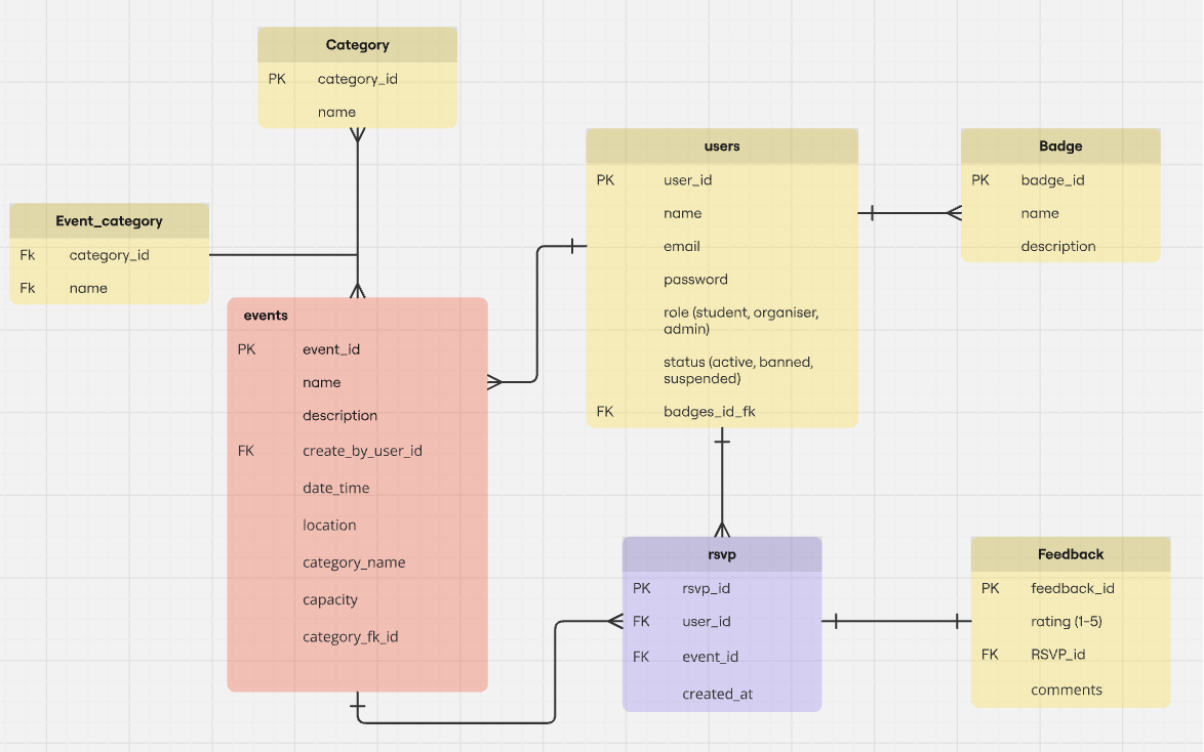

Figure 11: Entity-relationship diagram

APPENDIX C: Definition of done

- All unit tests derived from the acceptance criteria have been successfully executed and passed
- Integration tests verify end-to-end correctness of critical workflows.
- Code compiles and builds with no errors/warnings.
- API documentation updated with endpoint path and method, request params and query options, examples. Fields definition and data types are clarified.
- Approved by Product Owner.
- Necessary documentation (user guide, technical notes, feature updates) is updated.
- No high/critical severity bugs remain unresolved
- Test cases expected outcome is documented in detail

Appendix Milestone 2:

Updated DOD:

“Unit tests that are based on acceptance criteria are passed” - has been rephrased to – “All unit tests derived from the acceptance criteria have been successfully executed and passed”

“Integration tests verify correct data from storage layer.” - has been updated to – “Integration tests verify end-to-end correctness of critical workflows.”

Estimation for user stories:

The user stories were estimated in time (hours) instead of story points, in that case the development team have decided to convert time to story points in a Fibonacci scale; the conversion rate is shown below:

Estimated in hours	Estimated in story points
1-4	1
5-8	2
9-16	3
17-20	5
21-28	8

Note: this has been updated on the github board

Updated tasks and user stories:

The following tasks and user stories has been updated, as well as additional user stories which have been requested by the product owner.

Task 88: has been altered when cancelling rsvp, the entries will be removed from the rsvp table, instead of changing the status to cancelled.

Task 146: has been changed from allowing users to select multiple categories to only select one category and will dynamically display all events selected under that specified category.

Task 90: removed the status row from the RSVP event.

Task 92: removed the dropdown of the category filter feature as it was deemed unnecessary, since the development decided to implement a category select feature instead.

Task 102: “Filter date” has been added to the task to avoid confusion.

User story 49: has been removed entirely from the board, as it was considered as a redundant feature from the development team.

Task 169: this task has been added to the board in order to avoid ghost work.

The following user stories have been added to the board per the product owners’ request:

User story: 160, 161, 162, 163, 164, 165, 166, 167, 168