
Software Requirements Specification

for

<Plant>

Version 2.0 approved

Prepared by <Wania Naeem, Eman Ali, Nabeeha Shafiq>

<FAST NUCES, Islamabad>

<29-April-2025>

Table of Contents

Table of Contents	ii
Revision History	ii
1. Introduction	1
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Product Scope	1
1.5 References	1
2. Overall Description	2
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment	2
2.5 Design and Implementation Constraints	2
2.6 User Documentation	2
2.7 Assumptions and Dependencies	3
3. External Interface Requirements	3
3.1 User Interfaces	3
3.2 Hardware Interfaces	3
3.3 Software Interfaces	3
3.4 Communications Interfaces	3
4. System Features	4
4.1 System Feature 1	4
4.2 System Feature 2 (and so on)	4
5. Other Nonfunctional Requirements	4
5.1 Performance Requirements	4
5.2 Safety Requirements	5
5.3 Security Requirements	5
5.4 Software Quality Attributes	5
5.5 Business Rules	5
6. Diagrams	5
6.1 Use Case Diagram	5
6.2 Activity Diagram	5
6.3 Sequence Diagram	6
Appendix A: Glossary	6
Appendix B: Analysis Models	6
Appendix C: To Be Determined List	6

Revision History

Project GitHub URL: <https://github.com/Wania-n/Event-Management-System>

Commits		
main	All users	All time
Commits on Mar 23, 2025		
Merge branch 'main' of https://github.com/Wania-n/Event-Management-System	8ceb5	<>
Nabeeha-Shafiq committed 2 minutes ago		
nabs just committed !	9a892a2	<>
Nabeeha-Shafiq committed 4 minutes ago		
Add files via upload	261099d	<>
Emi-Pemi authored 7 minutes ago		
Add files via upload	b28b590	<>
Emi-Pemi authored 31 minutes ago		
Signup form connected to backend now!	c01a0f	<>
Wania-n committed 6 hours ago		
Fixed some bugs	4469486	<>
Wania-n committed 18 hours ago		
Commits on Mar 22, 2025		
Controller classes added!	7b1d968	<>
Wania-n committed yesterday		
Commits on Mar 17, 2025		
Service Layer added!	f2b15f0	<>
Wania-n committed yesterday		
Repository layer added!	2f21230	<>
Wania-n committed yesterday		
SQL file updated	562348d	<>
Wania-n committed yesterday		
Model and factory module created!	8290cce	<>
Wania-n committed yesterday		
Commits on Mar 17, 2025		
Initial commit!	83f17ff	<>
Wania-n committed last week		

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to describe the functional and non-functional requirements for the "PlanIt" event management application. The application will help individuals plan events such as weddings, birthdays, parties, and more by providing one place to integrate all ideas and aspects related to an event.

1.2 Document Conventions

Fonts: All headings are bold with a large font size, and sections are numbered.

Priority: Higher-level requirements will take precedence over detailed requirements.

Terms: "Must" refers to mandatory requirements; "should" refers to tentative features.

1.3 Intended Audience and Reading Suggestions

Developers: To understand the technical specifications and requirements for the system.

Scrum Master/Product Owner: To understand the scope, functionality, and constraints of the application.

End Users: To understand system specifications and benefit from them.

The document is organized as follows:

1. Introduction
2. Overall Description
3. Functional Requirements
4. Non-Functional Requirements
5. Use Case Diagram
6. User Stories
7. Class Diagram
8. Sequence Diagram

1.4 Product Scope

At ENWA, our mission is to revolutionize the event management process with a simple, minimalistic and easy to use application. In today's fast-paced and multitasking world, people need efficient solutions to organize and simplify their complex events easily and that is exactly what our application would do. Our vision is to create a user-friendly event management application that serves as the go-to solution for individuals planning events for example weddings /birthdays/parties etc.

1.5 References

IEEE 830-1998: Software Requirements Specification Standard

2. Overall Description

2.1 Product Perspective

"PlanIt" is an innovative solution that will simplify the event management process. This web application will allow users to manage events efficiently. It will integrate with various services like vendors, venues, caterers etc to bring all ideas/aspects related to planning of any event in one organised, structured app.

2.2 Product Functions

Idea Organization: Provide a space for users to gather and store ideas for easy access and reference.

To-Do Lists: Users can create lists, edit them, and tick off completed tasks.

Vision Boards: Offer tools to create visual mood boards for themes, decor, and more.

Event Components Planning:

- **Decor/Venue Planning:** Tables with attributes like colour theme , list of items ,images, status to decide upon theme .
- **Food Planning:** Attributes include menu , food items , caterers , cutlery , status .
- **Furniture / Layout Planning:** Arrange seating, furniture type , quantity , and plan layout design.
- **Guest Management:** Manage guest lists (names), and seating arrangements (assigned tables).

2.3 User Classes and Characteristics

Primary Stakeholders: Individual users planning their events .

Secondary Stakeholders: Admin , Vendors (caterers, decorators, venues) and service providers (photographers, entertainment).

2.4 Operating Environment

The "PlanIt" event management application will be built upon the following tech stack :

Frontend (UI): React (JavaScript), shall be used for minimal and easy to navigate UI.

Backend: Spring Boot (Java) will be used for building the business layer logic.

Database: MySQL, a relational database, will be used to manage queries.

JPA: JPA (Java Persistence API) will be used for mapping Java objects to database tables, with

Hibernate as the implementation for efficient database interaction.

2.5 Design and Implementation Constraints

Cross-platform Compatibility: The web application should be compatible with major browsers (Chrome, Firefox, Safari, and Edge).

Database Constraints: MySQL schema must be structured to avoid dependencies and redundancy. Must be 3-layer normalised.

2.6 User Documentation

User Manuals: A detailed user manual will be provided to help users understand how to interact with the "PlanIt" system. It will cover functionalities such as event creation, guest management, vendor booking, and task management.

SRS Document: Detailed SRS document as per IEE Standard format for users, developers, project managers, scrum master, and product owners alike to understand the system from scratch.

2.7 Assumptions and Dependencies

Stable Internet Connection: The app assumes users will have secure and stable internet connection for the web application to run seamlessly.

Dependencies/Libraries: Dependencies like React, Spring Boot, Hibernate, and MySQL shall be compatible with each other and up to date. Any version incompatibilities must be resolved prior to implementation.

3. External Interface Requirements

3.1 User Interfaces

The application's UI shall be simple, minimalistic and easy to use and navigate.

Error Messages: User-friendly error messages will help guide the user when incomplete requirements.

Design: A tab-based design, where navigation via related tab /menu is possible.

3.2 Hardware Interfaces

Our web- application shall be compatible to be run on :

Devices: Desktops, laptops.

Operating Systems: Compatible with Windows, macOS, Linux (desktop) and Android, iOS (mobile).

Protocols: HTTP(Hyper Text Transfer Protocol) for communication over the web.

3.3 Software Interfaces

The application shall communicate with necessary software components such as :

Web Browsers: Chrome, Firefox, Safari, Edge

Backend: Built with Spring Boot (Java) and communicates with MySQL via JPA and Hibernate.

3.4 Communications Interfaces

The communication of the application shall be carried out via :

HTTP: For web communication.

Email: Used for notifications, and sending invites to guests.

4. System Features

4.1 User Login

4.1.1 Description and Priority

This feature allows users to log into their existing accounts. It ensures that only authenticated users/admins can access and manage their events.

Priority: High

Benefit: High

Penalty: Medium

Cost: Medium

Risk: Medium

4.1.3 Functional Requirements

REQ-1: The system must provide a login page where users can enter their username/email and password.

REQ-2: The system must authenticate the user credentials by comparing them to stored user data in the database.

REQ-3: The system must provide error messages if the user enters invalid credentials (e.g., incorrect password or username).

4.2 User Account Management

Description:

This feature allows users to manage their accounts by creating new accounts, editing existing user details, and deleting accounts.

Priority: High

Benefit: High

Penalty: High

Cost: Medium

Risk: Medium

4.2.1 Functional Requirements

REQ-1: The system must allow the admin to create a new user account by entering necessary details such as name, email, and role.

REQ-2: The system must allow the admin to edit existing user details (e.g., email, name, role, permissions).

REQ-3: The system must provide confirmation prompts before allowing the admin to delete a user account.

REQ-4: The system must ensure that user data is securely stored and updated in the database when modified.

REQ-5: The system must prevent the deletion of an account if it has dependencies on other schema tables.

4.3 Venue Management

4.3.1 Description and Priority

Description:

This feature allows the admin to manage event venues, including adding, editing, and deleting venue information such as name, location, capacity, and available dates. The admin can ensure that the list of venues is always up-to-date for users to choose from.

Priority: High

Benefit: High

Penalty: High

Cost: Medium

Risk: Medium

4.1.3 Functional Requirements

REQ-1: The system must allow the admin to add new venues by entering venue details (name, location, capacity, etc.).

REQ-2: The system must allow the admin to edit the details of an existing venue, including its availability and capacity.

REQ-3: The system must allow the admin to delete venues that are no longer available for booking.

REQ-5: The system must store all venue data securely and ensure it is updated across all schema.

System Feature: Manage Vendors

4.1.1 Description and Priority

Description:

This feature allows the admin to manage vendors, including adding, updating, and removing vendors from the system. The admin can add vendor details such as name, services offered, contact information, and availability.

Priority: High

Benefit: High

Penalty: High

Cost: Medium

Risk: Medium

4.1.3 Functional Requirements

REQ-1: The system must allow the admin to add new vendors by entering vendor details.

REQ-2: The system must allow the admin to edit existing vendor details.

REQ-3: The system must allow the admin to delete vendors who are no longer available for events.

REQ-5: The system must store all vendor data securely and ensure it is updated across all schema.

System Feature: Manage Guests

4.1.1 Description and Priority

Description:

This feature allows users to manage their guest lists by adding, editing, and removing guests. Users can track guest details, including names list with contact details.

Priority: High

Benefit: High

Penalty: Medium

Cost: Medium

Risk: Low

4.1.3 Functional Requirements

REQ-1: The system must allow users to add guests with details.

REQ-2: The system must allow users to edit existing guest details.

REQ-3: The system must allow users to delete guests from the list.

The system must store all guest data securely and ensure it is updated across all schema.

Send Invites

4.1.1 Description and Priority

Description:

This feature allows users to send invitations to their guests via email. The system automatically generates the invitation content, which includes event details and RSVP

options, and sends it to the guests' registered email addresses.

Priority: High

Benefit: High

Penalty: Medium

Cost: Medium

Risk: Low

4.1.3 Functional Requirements

REQ-1: The system must generate an email template for invitations with event details (date, time, location, etc.).

REQ-2: The system must allow users to select guests from the guest list to send invitations.**REQ-3:** The system must send the email invitation to the selected guests' email addresses.

REQ-4: The system must allow users to customize the invitation content before sending.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

Response Time: The application should load any user interaction within 2 seconds.

Throughput: The application should support up to 10,000 concurrent users without performance degradation.

5.2 Safety Requirements

Data Integrity: The application must ensure the safety and accuracy of event and user data.

Safety Protocols: No unsafe operations should be allowed that could cause loss of data or system crashes.

5.3 Security Requirements

Authentication: Users must authenticate via username and password before accessing the app.

Data Privacy: All sensitive user data must be encrypted and comply with data protection regulations.

Authorization: Different levels of access must be granted to users and admins for data security.

5.4 Software Quality Attributes

Reliability: The application must be available 99.9% of the time.

Usability: The system should be easy to use for non-technical users with minimal training.

Maintainability: Code should follow Object-Oriented Programming (OOP) standards and be easy to refactor. It should follow rule "open for extension , closed for modification"

5.5 Business Rules

Admin Privileges: Only admins can add, edit, or delete vendors, venues, and event details.

User Management: Users can only manage their own events, guest lists, and invitations.

5.6 Organizational Requirements

Project Management: The project will follow an Agile development methodology with **Scrum**. Daily Scrum meetings, sprint backlogs, and sprint planning will be used to manage the development process.

Version Control: The project will use GitHub for version control to manage code changes and collaboration.

Collaboration Tools: The team will use **Trello** for task management and tracking progress, Google Meet for remote meetings, and WhatsApp for quick communication.

Documentation Standards: All documentation will be created in alignment with the IEEE 830 standards, ensuring clarity, consistency, and thoroughness.

5.7 External Requirements

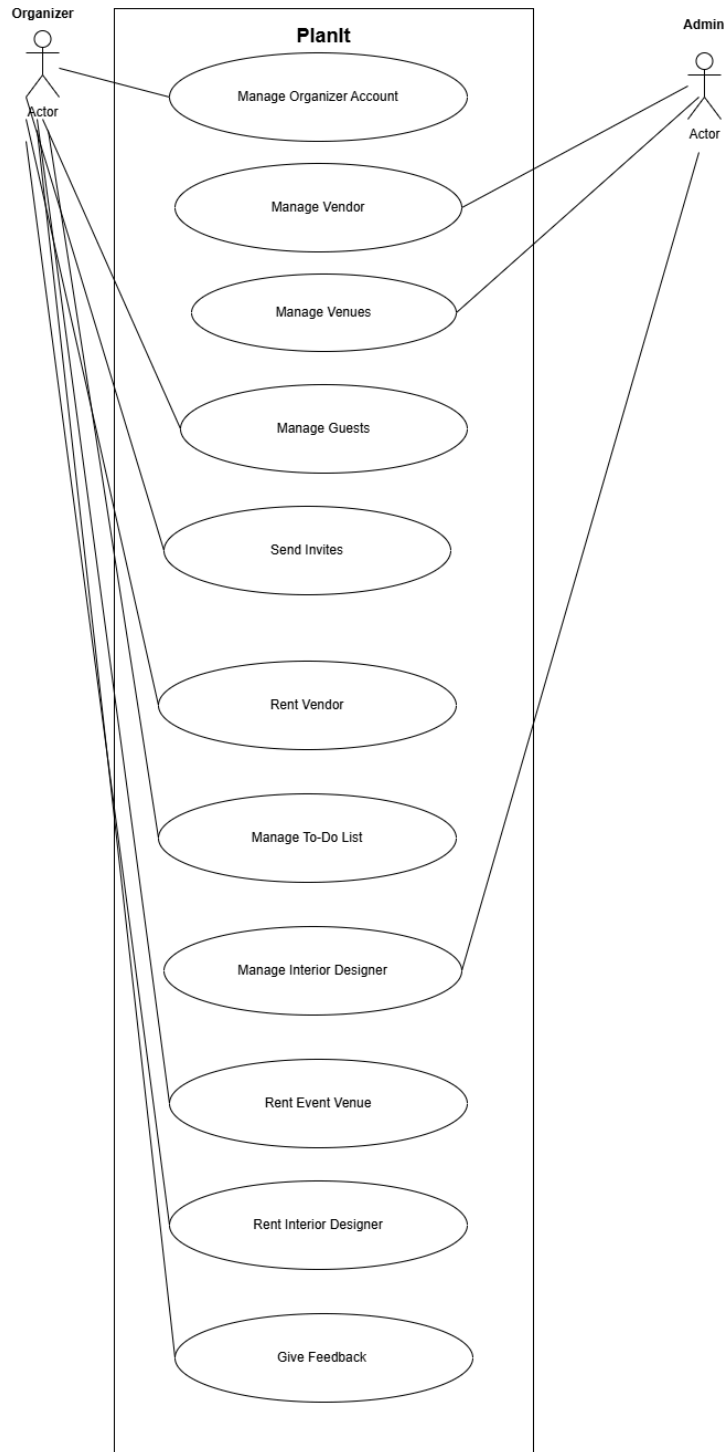
Integration with Third-Party Services: The application should integrate with external services for sending email invitations (e.g. SMTP).

Compliance: The system should comply with data privacy regulations such as GDPR (General Data Protection Regulation) and local data protection laws for user information.

Browser Compatibility: The application must be compatible with major web browsers (Chrome, Firefox, Safari, Edge).

5. Diagrams

5.1 Use Case Diagram



5.2 User Stories

1. Use Case: User Login

Description:

Users (both regular and admin) can log into the system to access it and perform their functionalities.

User Story 1: User Login

- *As a user/admin*
- *I want to log in to the system*
- *So that I can access my account and use the app's features.*

Pre-conditions:

User has an existing account with valid credentials.

Post-conditions:

User is authenticated and granted access to the system.

2. Use Case: User Account Management

Description:

Users can create (sign up), manage, and delete their accounts.

User Story 2: User Account Creation (Sign Up)

- *As a user*
- *I want to sign up for a new account*
- *So that I can access the app and manage my events.*

Pre-conditions:

The user is not registered in the system.

Post-conditions:

A new user account is created and the user can log in.

User Story 3: User Account Management (Edit/Delete)

- *As a user*

- **I want to** edit or delete my account
- **So that** I can manage my personal details or remove my account from the system.

Pre-conditions:

User is logged into their account.

Post-conditions:

The user's details are updated or the account is deleted from the system.

3. Use Case: Event Venue Management

Description:

Admins can manage event venues, including adding, editing, and deleting venue information such as name, location, capacity, and available dates.

User Story 4: Add Event Venues (Admin)

- **As an admin**
- **I want to** add venues to the system
- **So that** users can rent venues for their events.

Pre-conditions:

Admin is logged into the system.

Post-conditions:

New venue details are added and available for user selection.

User Story 5: Edit/Delete Venue Info (Admin)

- **As an admin**
- **I want to** edit or delete venue details
- **So that** the venue information stays up-to-date.

Pre-conditions:

Admin is logged into the system.

Post-conditions:

Vendor details are updated or the vendor is removed from the system.

4. Use Case: Manage Vendors

Description:

Admins can manage vendors by adding, editing, or removing vendors from the system. Admins can add vendor details such as name, services offered, and contact info.

User Story 6: Add Vendors (Admin)

- **As an admin**
- **I want to** add vendors to the system
- **So that** users can rent services for their events.

Pre-conditions:

Admin is logged into the system.

Post-conditions:

New vendor details are added and available for user selection.

User Story 7: Edit/Delete Vendor Info (Admin)

- **As an admin**
- **I want to** edit or delete vendor details
- **So that** the vendor information stays up-to-date.

Pre-conditions:

Admin is logged into the system.

Post-conditions:

Vendor details are updated or the vendor is removed from the system.

5. Use Case: Manage Guests

Description:

Users can manage their guest lists by adding, editing, and removing guests.

User Story 8: Add Guest to List (User)

- **As a user**
- **I want to** add guests to my event's guest list
- **So that** I can keep track of who is invited.

Pre-conditions:

User is logged into their account.

Post-conditions:

New guest is added to the guest list.

User Story 9: Edit/Delete Guest from List (User)

- **As a user**
- **I want to** edit or delete guests from my list
- **So that** I can update the guest list if needed.

Pre-conditions:

User is logged into their account.

Post-conditions:

Guest details are updated or the guest is removed from the list.

6. Use Case: Send Invites

Description:

Users can send email invitations to their guests with event details .

User Story 10: Send Invitations (User)

- **As a user**
- **I want to** send invites to all my guests via email
- **So that** they can receive event details .

Pre-conditions:

User has a guest list created.

Post-conditions:

Guests receive the email with event details .

7. Use Case: Rent Vendors

Description:

Users can rent vendors for their event via the app, selecting from the available vendor list.

User Story 11: Rent Vendor (User)

- **As a user**
- **I want to** rent vendors (such as caterers, and decorators) for my event
- **So that** I can ensure all services for my event are booked.

Pre-conditions:

The user is logged in and has selected the desired vendor.

Post-conditions:

Vendor is successfully booked for the event.

8. Use Case: Rent Event Venue

Description:

Users can rent an event venue through the app.

User Story 11: Rent Event Venue (User)

- **As a user**
- **I want to** rent a venue for my event
- **So that** I can secure a location for my event.

Pre-conditions:

User is logged in and has selected a venue.

Post-conditions:

Venue is successfully booked for the event.

User Story 12: Manage Event Venue Booking (User)

- **As a user**
- **I want to edit/delete** my booking for an event

- So that I can update the booking as per my preference.

Pre-conditions:

The user is logged in and has rented a venue.

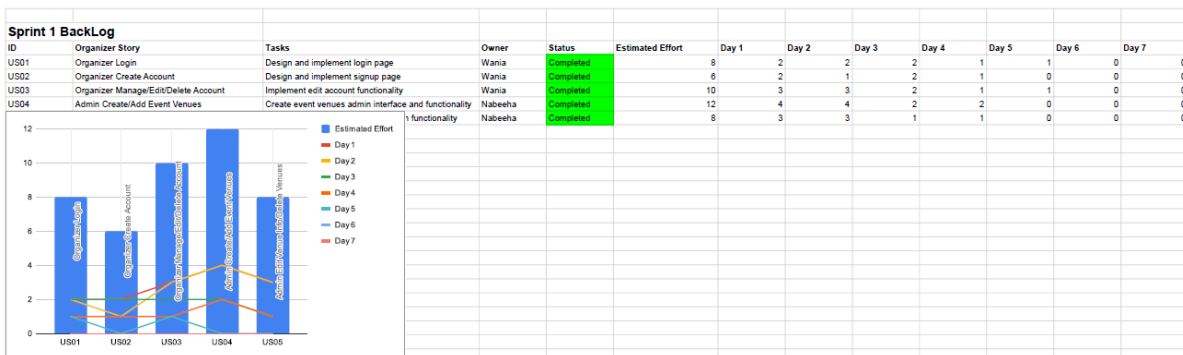
Post-conditions:

The venue is successfully edited/deleted for the event.

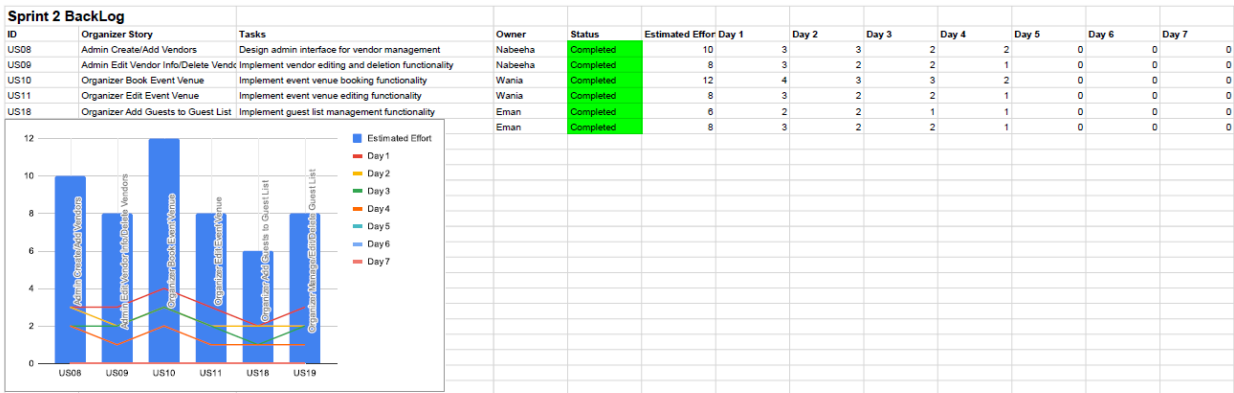
6.0 Product Backlog

"PlanIt"						
Product BackLog						
ID	As a...	I want to be able to...	So that...	Priority	Sprint	Status
1	Organizer	Login to the system	I can access and manage my events	High	Sprint 1	Completed
2	Organizer	Create an account/sign up	I can use the app and manage my events	High	Sprint 1	Completed
3	Organizer	Manage/edit/delete my account	I can keep my account up-to-date	High	Sprint 1	Completed
4	Admin	Create/add event venues to the system	organizers can book available venues	High	Sprint 1	Completed
5	Admin	Edit venue info/delete venues	Venue details are up-to-date and accurate	Medium	Sprint 1	Completed
6	Admin	Create/add interior designers to the system	Organizers can rent them for events	High	Sprint 4	Completed
7	Admin	Edit interior designers' info/delete designers	Designer details are updated and accurate	Medium	Sprint 4	Completed
8	Admin	Create/add vendors to the system	Organizers can rent them for events	High	Sprint 2	Completed
9	Admin	Edit vendor info/delete vendors	Vendors' details are up-to-date and accurate	Medium	Sprint 2	Completed
10	Organizer	Book an event venue available on the app	I can host my event at the chosen venue	High	Sprint 2	Completed
11	Organizer	Edit the event venue available on the app	The venue details are correct for my event	Medium	Sprint 2	Completed
12	Organizer	Rent a vendor	My event can have the necessary vendors	High	Sprint 3	Completed
13	Organizer	Edit/delete my rented vendor	I can make changes to the vendor selection	Medium	Sprint 3	Completed
14	Organizer	Rent an interior designer	My event can have the desired theme and decor	High	Sprint 3	Completed
15	Organizer	Edit/delete my rented interior designer	I can make changes to my designer selection	Medium	Sprint 3	Completed
16	Organizer	Make a to-do list	I can organize and keep track of tasks for my event	Medium	Sprint 4	Completed
17	Organizer	Edit/delete my to-do list	I can update or remove tasks as needed	Medium	Sprint 4	Completed

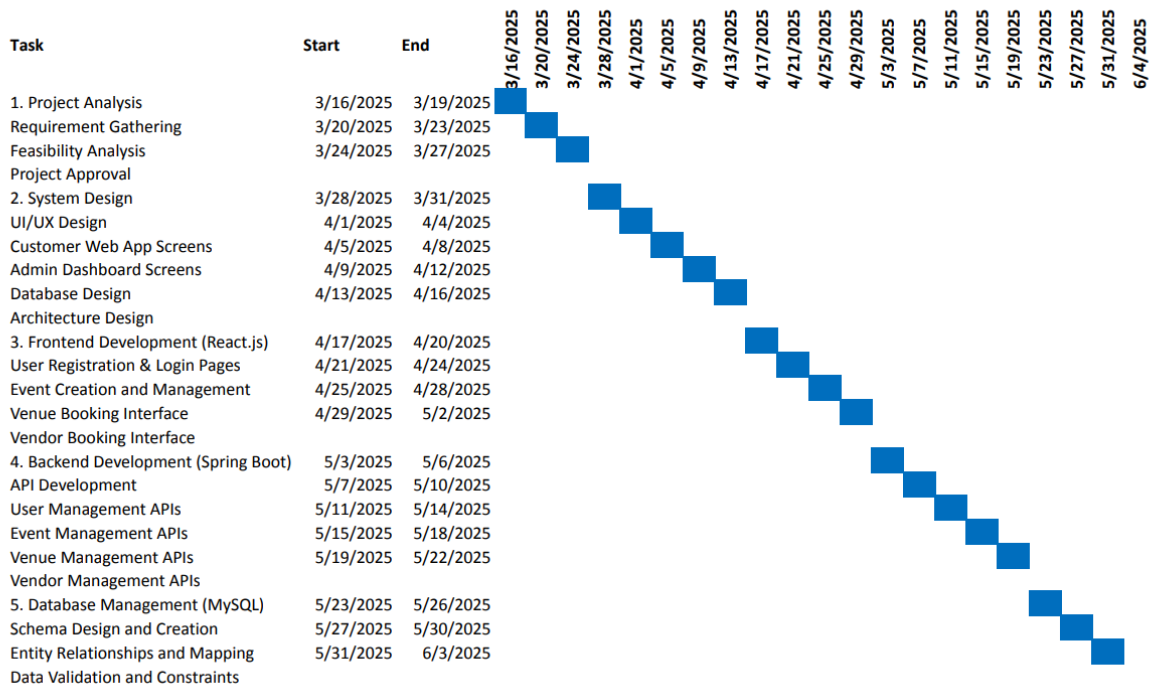
6.1 Sprint 1 Backlog



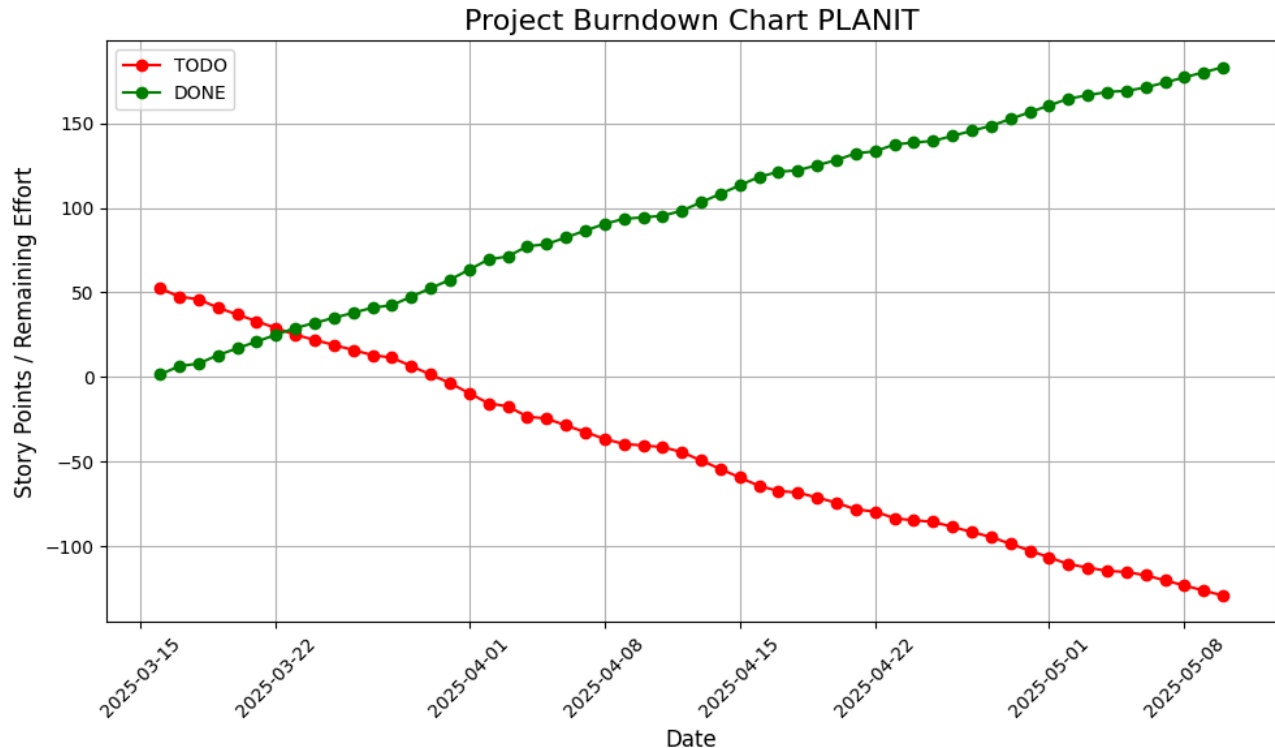
6.2 Sprint 2 BackLog



Gantt Chart



Project Burndown Chart



7.0 Project Plan (Work Breakdown Structure)

1. Project Analysis

1.1 Requirement Gathering

We conducted detailed requirement gathering by analysing **user needs** and requirements and understanding problems faced during event management. We gathered information by brainstorming ideas, competitor analysis, and user expectations.

1.2 Feasibility Analysis

A feasibility study was carried out to assess the project's technical, operational, and financial viability. We made sure our technologies (React, Spring Boot, MySQL) were capable of building a scalable system within our timeline and resources and were free too.

1.3 Project Approval

After analysing feasibility and finalising project requirements, the proposal was approved by our instructors to move into the design and development phase.

2. System Design

2.1 User Interface (UI/UX) Design

We designed simple, clean, and user-friendly interfaces using CSS and JavaScript. Special focus was given to making the application minimalistic and easy to navigate for all user types.

2.1.1 Customer Web App Screens

We created user-facing screens for event creation, guest management, vendor selection, and venue booking. Each screen was designed with intuitive layouts to provide a seamless user experience.

2.1.2 Admin Dashboard Screens

Admin screens were designed to allow administrators to manage venues, vendors, user accounts, and system notifications. Admin dashboard follows a clean and organized design.

2.2 Database Design

Database tables were carefully planned to manage different entities like Users, Events, Guests, Vendors, Venues, and Invitations, ensuring normalisation up to the 3rd normal form (3nf).

2.3 Architecture Design

We adopted a three-tier architecture where the frontend is built using React.js, backend services are developed in Spring Boot (Java), and data is managed using MySQL with JPA/Hibernate ORM.

3. Frontend Development (React.js)

3.1 User Registration & Login Pages

React pages were developed for user signup and login using secure authentication mechanisms.

3.2 Event Creation and Management

Users can create new events, set event dates, manage guest lists, and assign vendors through interactive forms and tables.

3.3 Venue Booking Interface

Users are provided with a venue listing interface where they can view available venues, check capacities, and book accordingly.

3.4 Vendor Booking Interface

Vendor services such as catering, photography, and decor are listed for users to book easily through dedicated pages.

3.7 Responsive Design Testing

We tested and optimised our web pages to ensure proper responsiveness on various devices like laptops, tablets, and smartphones.

4. Backend Development (Spring Boot)

4.1 API Development

We developed RESTful APIs to handle all application communication between frontend and backend.

4.1.1 User Management APIs (api/organizer)

These APIs allow users to register, login, edit profiles, and manage their event-related data.

4.1.2 Event Management APIs (catered in organizer API)

Integrated within the organizer module, these APIs allow users to create events, manage event details, and perform updates.

4.1.3 Venue Management APIs (catered in organizer API)

Users can view available venues, book venues, and manage their booked venues via APIs.

4.1.4 Vendor Management APIs (api/admin)

Admin APIs allow adding, updating, and removing vendors. Admin manages the entire vendor list.

5. Database Management (MySQL)

5.1 Schema Design and Creation

Tables were designed with primary and foreign keys to maintain relational integrity between users, events, venues, vendors, and guests.

5.2 Entity Relationships and Mapping (Hibernate, JPA)

Hibernate ORM was used for entity mapping, ensuring smooth interaction between Java objects and database tables.

5.3 Data Validation and Constraints

Constraints like NOT NULL, UNIQUE, and FOREIGN KEY were enforced to maintain data correctness and avoid redundancy.

6. Testing

6.1 Unit Testing (Frontend and Backend)

Each component and API endpoint was unit tested to ensure they perform their individual functionalities correctly.

6.2 Integration Testing

We performed integration testing to check whether frontend and backend work together seamlessly.

6.3 User Acceptance Testing (UAT)

Sample users as well as developers tested the application to confirm that it meets all user stories and functional expectations.

7. Deployment

7.1 Hosting Web App on Server

The frontend build and backend APIs will be deployed on a cloud server for live usage.

7.2 Database Deployment

MySQL database will be deployed and connected securely with the backend APIs.

7.3 Production Environment Setup

We will configure server security, database backups, and API monitoring to ensure a stable production environment.

8. Documentation

8.1 SRS Document

The Software Requirements Specification (SRS) document was prepared to align the whole team and future developers.

8.3 Deployment and Maintenance Guides

Developers will maintain deployment documentation and keep the app updated by fixing bugs and adding improvements as needed.

9. Project Management

9.1 Agile Scrum Management (Daily Scrum, Sprint Planning)

We followed Agile Scrum methodology, conducting daily stand-ups, weekly sprint reviews, and sprint planning sessions.

9.2 Version Control (GitHub Repository Management)

Our GitHub repository stores the entire source code, managing commits, pull requests, and branching for collaborative development.

9.3 Progress Tracking (Trello Board)

We used Trello to manage tasks, user stories, and deadlines effectively, ensuring clear visibility and smooth tracking of project progress.

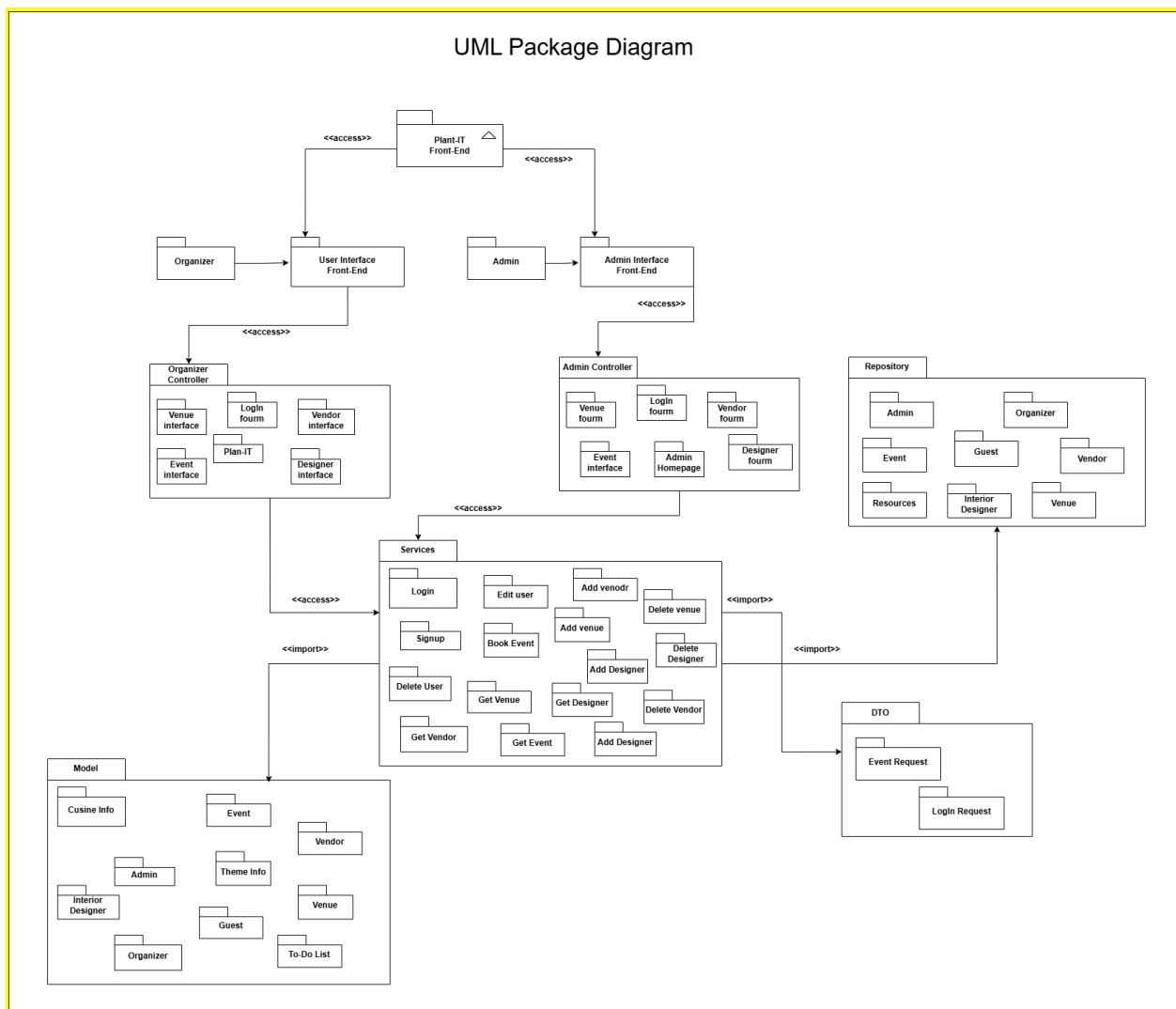
8.0 System Architecture

1. Identifying Subsystems

1. **User Management Subsystem:** Handles registration, login, profile management of organizer.
2. **Admin Management Subsystem** :Handles registration, login, profile management of the admin.
3. **Event Management Subsystem:** Manages event creation, decor, food, furniture, and layout planning.
4. **Venue Management Subsystem:** Manages venue listings, bookings, and availability.

5. **Vendor Management Subsystem:** Manages vendors like caterers, decorators, photographers.
6. **Interior Designer Management Subsystem :** Manages designers with their specific theme and styles.
7. **Admin Subsystem:** Allows administrators to manage interior designers, venues, and vendors.

UML Package Diagram for Subsystems:



Each of these is a logical package/module in your system.

2. Architecture Styles Used

1- Client-Server Architecture:

Client	Server
Frontend (React.js) acts as a client.	Backend (Spring Boot REST APIs) acts as a server.

2- Layered Architecture:

Presentation Layer: React frontend for users.

Business Logic Layer: Spring Boot services handling business operations.

Data Access Layer: JPA/Hibernate for database interactions.

Database Layer: MySQL database for storing event, user, vendor, and guest data.

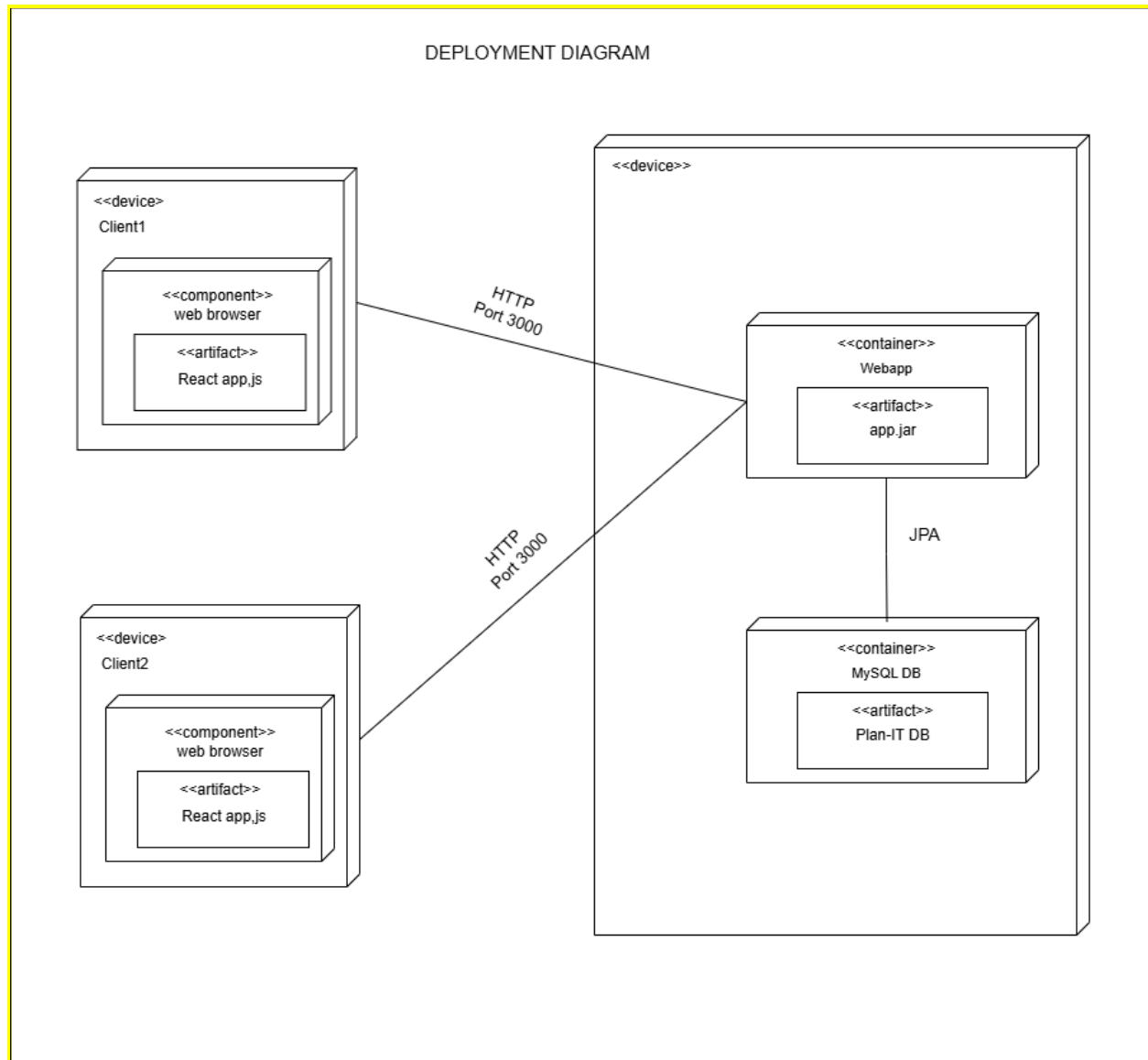
3- Model-View-Controller (MVC)

Model: Java entities such as **Event**, **Vendor**, **Guest**, **Task**, and **Venue** that represent the core data structures of the event management system.

View: The user interfaces are built with **React**, offering dynamic and responsive web pages that consume data via **REST APIs**. These views include event schedules, guest lists, venue details, and vendor information.

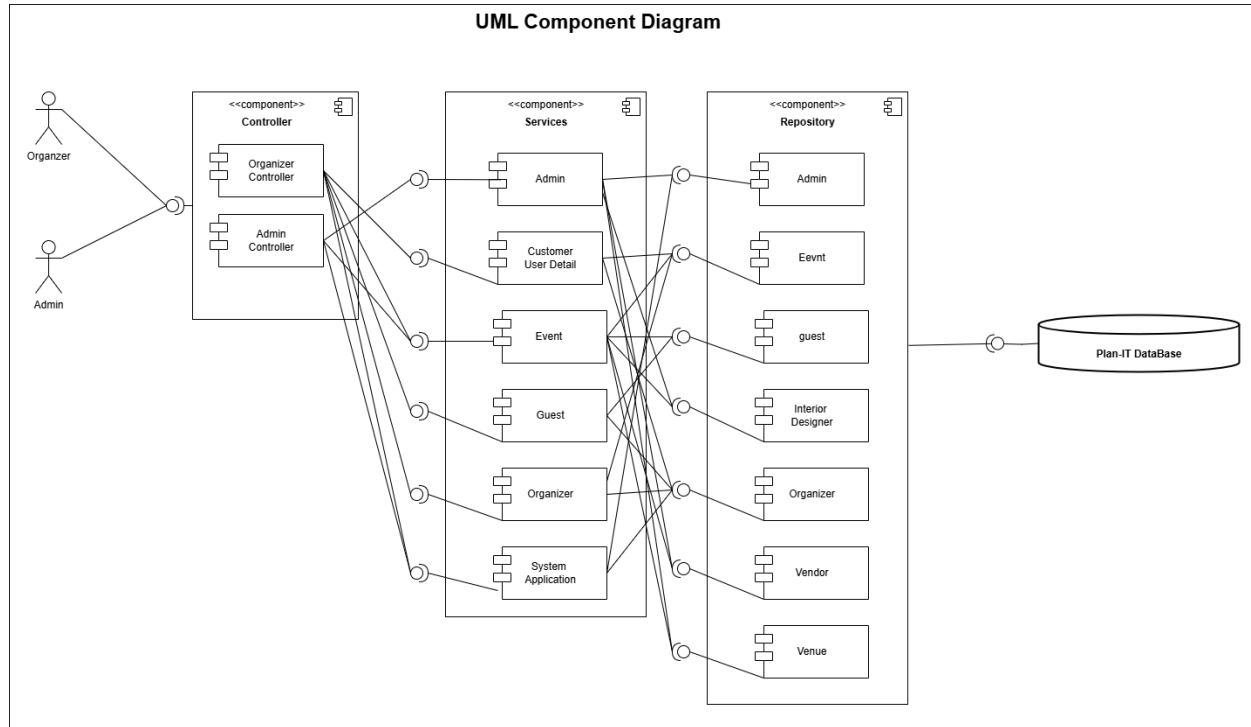
Controller: **Spring REST Controllers** manage incoming HTTP requests, interact with service layers, and return the appropriate responses. These controllers handle actions like event creation, vendor management, and task tracking.

3. Deployment Diagram

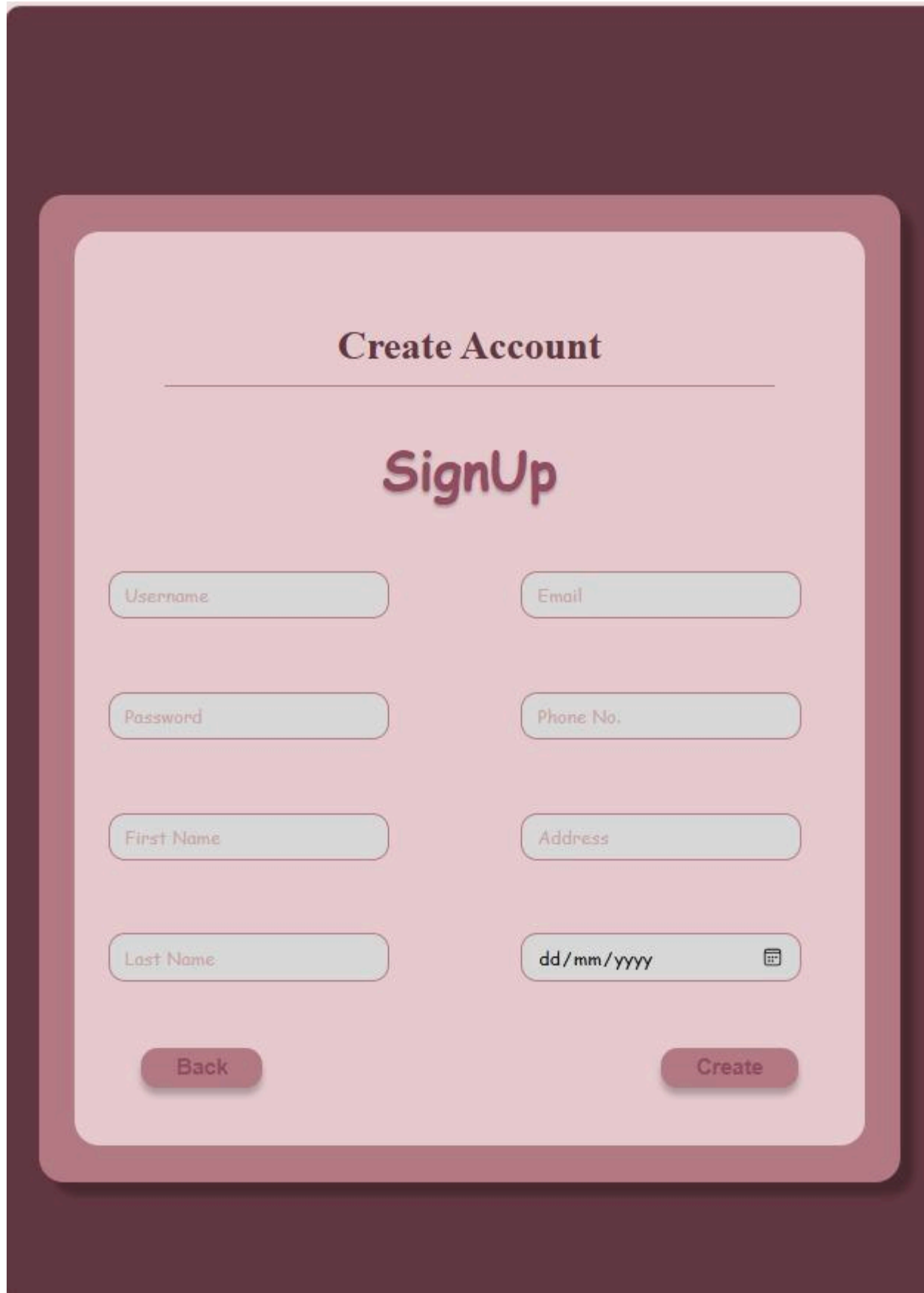
**Flow Of Deployment Diagram :**

- User PC runs a Web Browser (artifact: PlanIt Web App UI) and accesses the system via HTTPS.
- Web Server hosts React Build Files.
- Application Server runs Spring Boot REST APIs.
- Database Server hosts the PlanIt Database (MySQL).
- Communication happens over HTTPS, REST API Calls, and JPA/Hibernate internally.

4. Component Diagram



9.0 Actual Implementation ScreenShots



The image shows a 'Create Account' sign-up form. It has a dark maroon background with a lighter pink rounded rectangle in the center. The title 'Create Account' is at the top, followed by a horizontal line and the word 'SignUp' in a large, bold, maroon font. Below this are eight input fields arranged in two columns. The left column contains 'Username', 'Password', 'First Name', and 'Last Name'. The right column contains 'Email', 'Phone No.', 'Address', and a date field with the placeholder 'dd/mm/yyyy' and a calendar icon. At the bottom are two buttons: 'Back' on the left and 'Create' on the right.

Create Account

SignUp

Username

Email

Password

Phone No.

First Name

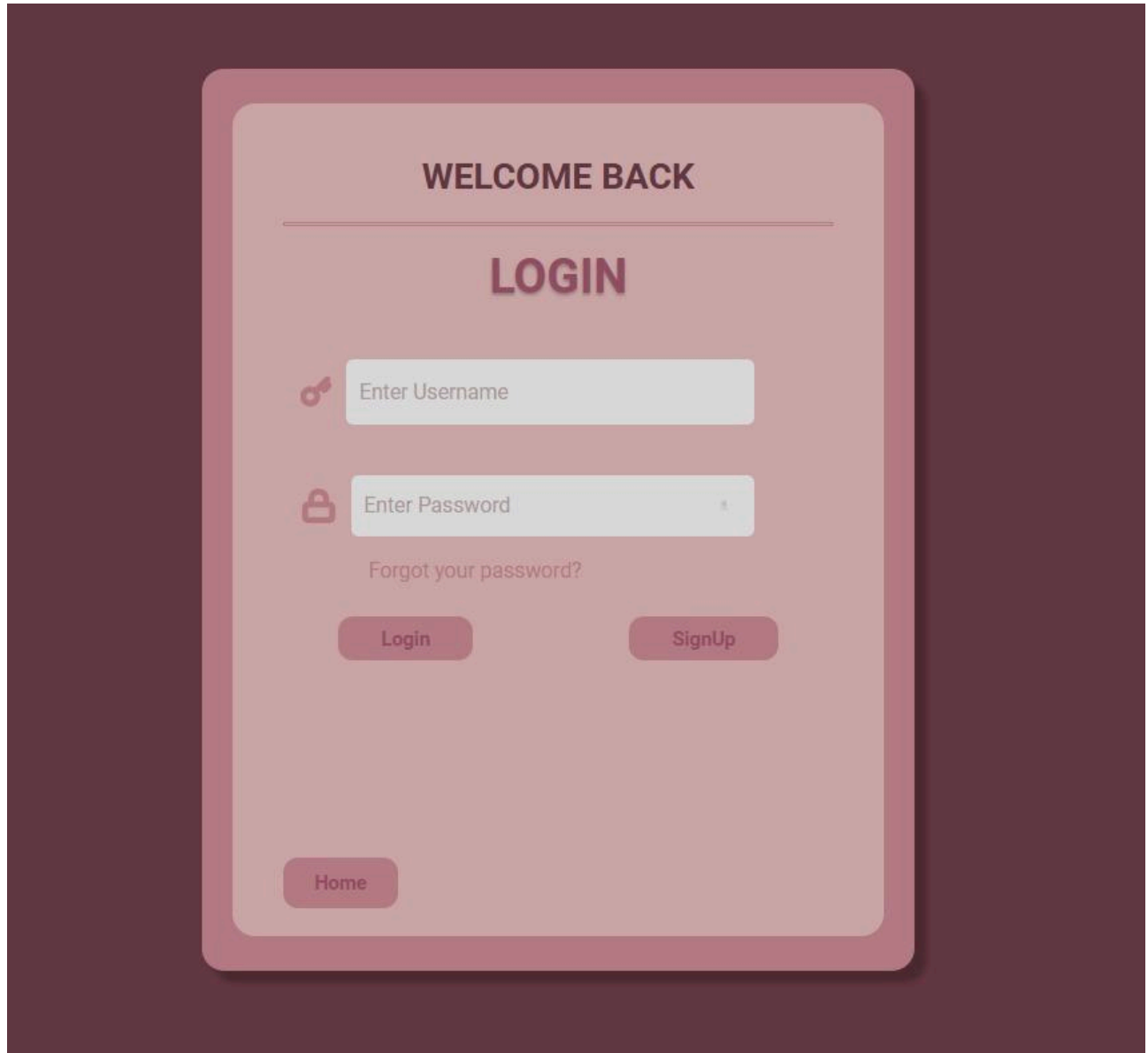
Address


Last Name

dd/mm/yyyy

Back

Create





User

Book Event

Book Vendor

Book Designer

Welcome to Organizer HomePage

Book Your Event

Date and Start Time:

dd/mm/yyyy --:-- --

End Time:

--:-- --

Username:

Event Name:

Theme:

Select a theme

Venue:


Select a venue

Back

Book Vendor

Book Designer

Submit Booking



Vendor

Venue

Designer

Welcome to Admin Homepage

Added Vendors

Floral Dreams - Decor

Catering Delights - Food

Added Venues

Grand Ballroom - Capacity: 500

Garden Terrace - Capacity: 200

Added Designers

Elegant Designs - Bridal Wear

Creative Styles - Party Outfits

Logout

Add/Delete Vendor

Name:

Phone Number:

Email:

Address:

Price:

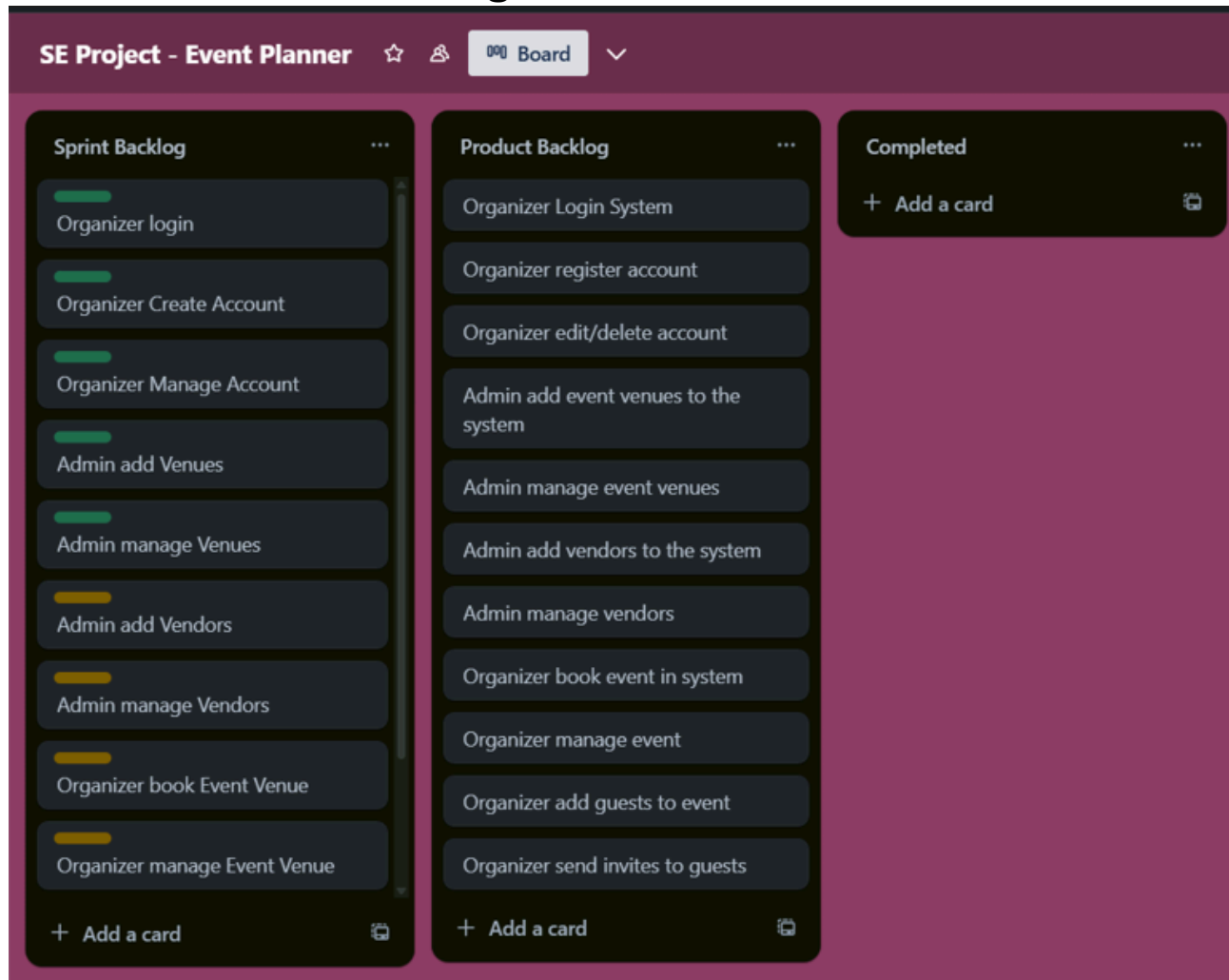
Service Type:

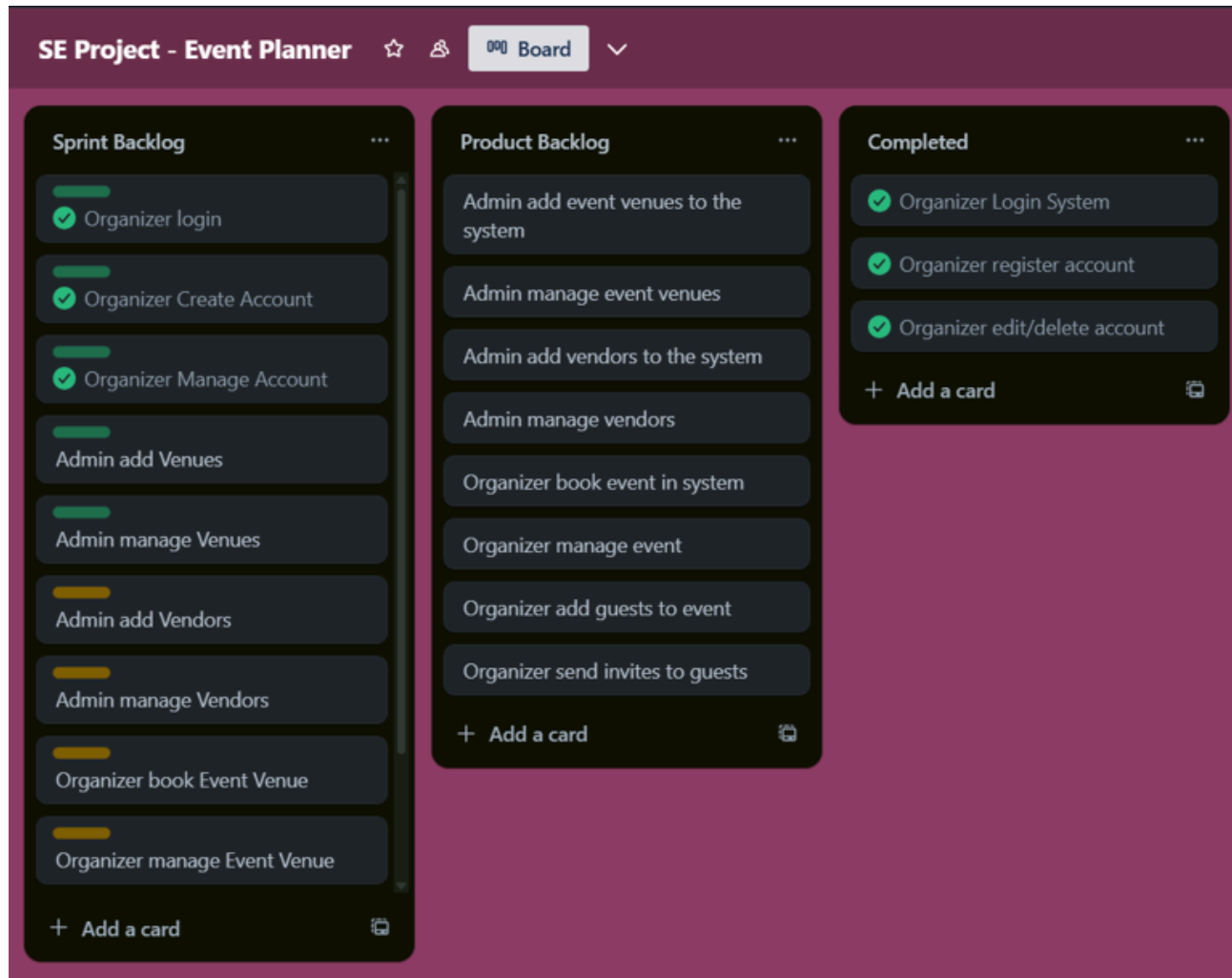
Add Vendor

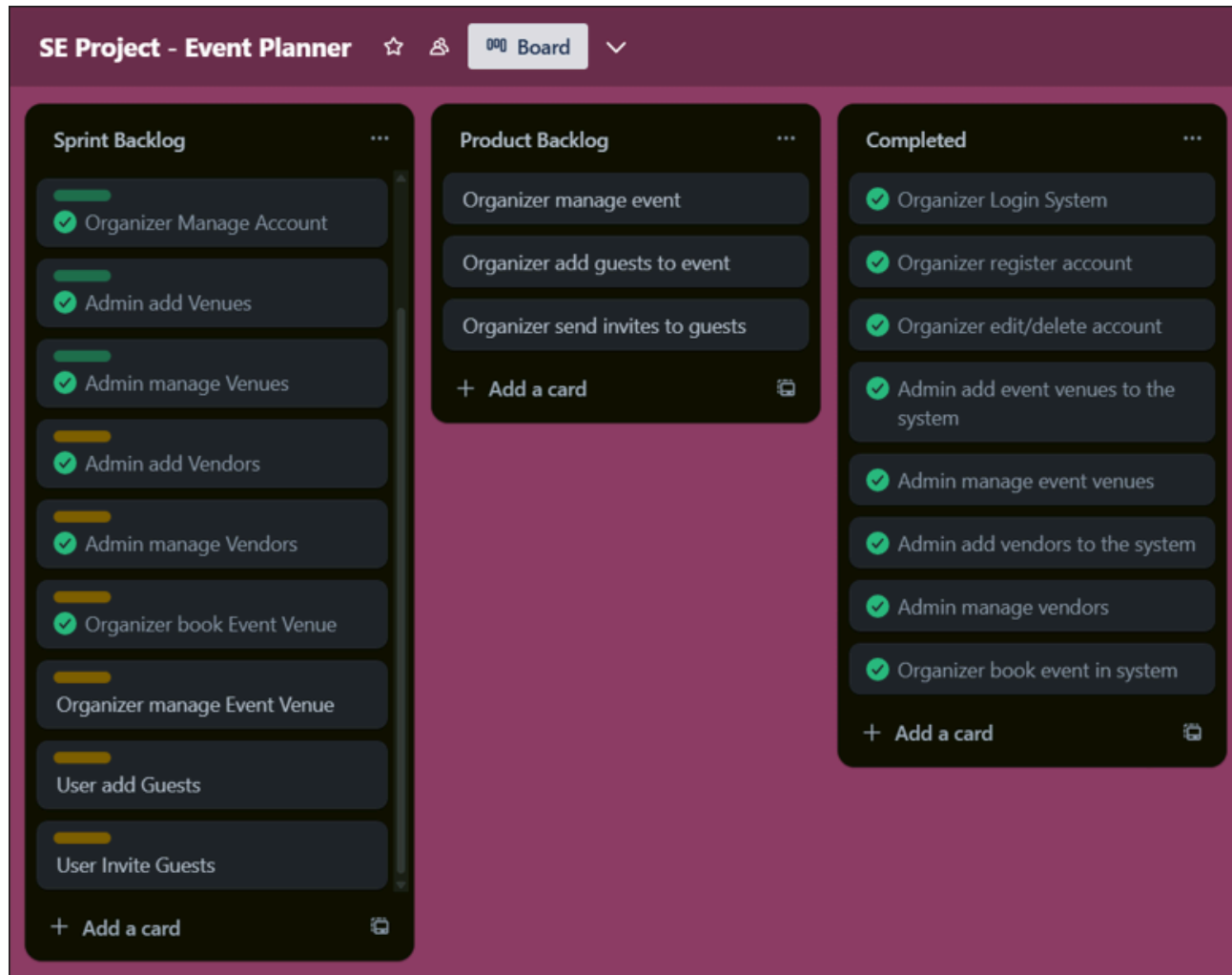
Delete Vendor

Close

10.0 Trello Board Progress







11.0 BlackBox Testing

Equivalence Class Partitioning

Class	Valid Range	Invalid Range
Name	Length $2 \leq x \leq 50$, alphabets and spaces only	$x < 2$, $x > 50$, non-alphabetic characters
Username	Length $3 \leq x \leq 20$, starts with letter, contains letters/numbers/._ only	$x < 3$, $x > 20$, starts with non-letter, invalid characters
Password	Length $6 \leq x \leq 20$, at least one letter and one number, allowed symbols @#\$%^&+=!	$x < 6$, $x > 20$, only letters or only numbers

Email	Proper email format: abc@domain.com	missing '@', missing domain, invalid special characters
Address	Length $5 \leq x \leq 100$, letters/numbers/ ,.- allowed	$x < 5$, $x > 100$, invalid characters
Date of Birth (DOB)	Age ≥ 18 years	Age < 18 years

Boundary Value Analysis

Class	Boundary Value Analysis	Expected Result
First Name Length = 2	Valid	Pass
First Name Length = 1	Invalid	Fail
Username Length = 3	Valid	Pass
Username Length = 2	Invalid	Fail
Password Length = 6	Valid	Pass
Password Length = 5	Invalid	Fail
Address Length = 5	Valid	Pass
Address Length = 4	Invalid	Fail
DOB exactly 18 years ago	Valid	Pass
DOB 17 years 11 months ago	Invalid	Fail

Test Cases (Validating System Functionality)

Test Case ID	Input	Expected Output
TC01	All fields valid	Organizer registered successfully
TC02	First Name = "A" (1 character)	Validation failed (First Name)
TC03	Last Name = "1234" (numbers only)	Validation failed (Last Name)
TC04	Username = "12user" (starts with number)	Validation failed (Username)
TC05	Password = "abcdef" (no number)	Validation failed (Password)
TC06	Email = "user@domain" (missing .com)	Validation failed (Email)
TC07	Address = "abc" (only 3 characters)	Validation failed (Address)
TC08	DOB = 17 years old	Validation failed (DOB)
TC09	Correct username and password for login	Login successful
TC10	Wrong username and password for login	Login failed

12.0 WhiteBox Testing

Implemented in Code, please.

13.0 Work Division

Wania:

Product Manager / Scrum Master / Developer

Emaan:

UI/UX / Developer / Tester

Nabeeha:

Analyst / Developer / Tester

14.0 Lessons Learnt By Group

This Web development project was actually a good experience in terms of learning a new tech stack. So far, we had been developing desktop applications, but this time we tried out a web application; hence, learning wise it was a valuable learning experience. Although the UI for the web application was very different on React but our skilled member Eman did it anyway. Understanding the layers and integrating all layers to communicate with one another was a new thing to implement for us, but our developer and product scrum master, Wania, did it all. For the first time in CS degree, we finally understand what an API is, how GitHub is supposed to be used actually (learnt the hard way by abandoning one repo due to git clashes), how important it is to start working on a project early on, and how good project management leads to in time submissions.