SOFTWARE REQUIREMENTS SPECIFICATION

for

TASK LINK

Deliverable 2

Prepared by : 1. Waseem Zahid 22i-1355)
2. Laiba Raza 22i-2359
2. Shayan Salam 21i-2964

Submitted to : Ma'am Sidra Khalid
Lecturer

April 29, 2025

Contents

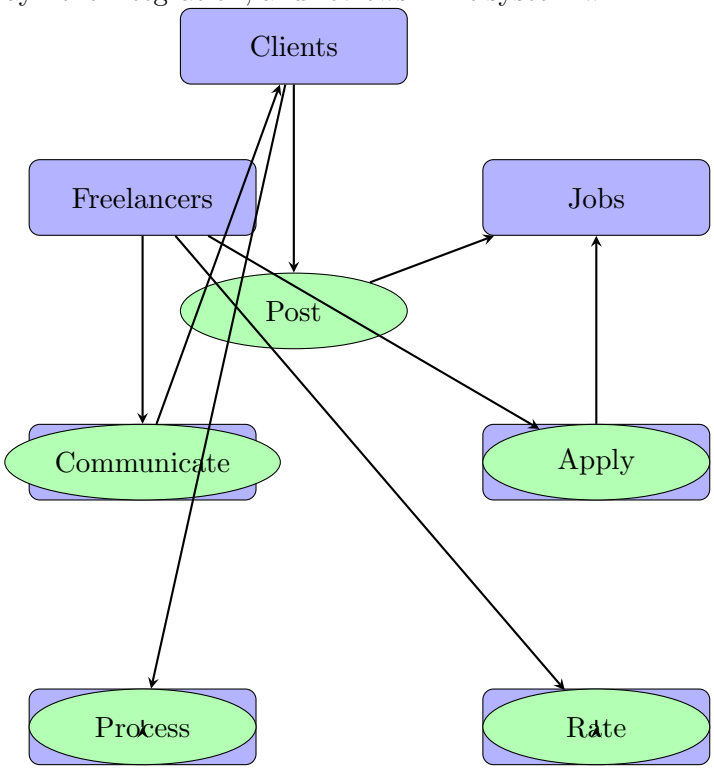
1 Introduction

1.1 Purpose

The purpose of this document is to define the functional and non-functional requirements, one big use case diagram, user stories, sequence diagram and class diagram for the Freelance Job Portal, a platform where clients can post jobs and freelancers can search and apply for jobs. This document will serve as a guide for developers, testers, and stakeholders to ensure the system meets its intended functionality and quality standards.

1.2 Project Scope

The Freelance Job Portal will provide a platform for clients to post jobs and freelancers to find and apply for jobs. The system will include features such as user registration, job posting, job search, application management, messaging, payment integration, and reviews. The system will be web-based and accessible to users globally.



1.3 Definitions, Acronyms, and Abbreviations

- **Client**: A user who posts jobs on the platform.
- **Freelancer**: A user who applies for jobs on the platform.
- **Admin**: A user with administrative privileges to manage the platform.
- **SRS**: Software Requirements Specification.
- **IEEE**: Institute of Electrical and Electronics Engineers.

1.4 References

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.
- Upwork.com as a reference for similar functionality.

1.5 Overview

This document is organized into sections that describe the functional and non-functional requirements,one big use case diagram, user stories, sequence diagram and class diagram of the Freelance Job Portal. Each requirement is uniquely identified and described in detail.

2 Functional Requirements

FR-1 The system shall allow users to register as either a client or a freelancer.

FR-2 The system shall authenticate users using a unique email address and password.

FR-3 The system shall allow clients to post jobs with details such as title, description, budget, category, and deadline.

FR-4 The system shall allow freelancers to search for jobs by keywords, category, budget range, and deadline.

FR-5 The system shall allow freelancers to view detailed information about a job, including the client's profile and job requirements.

FR-6 The system shall allow freelancers to apply to a job by submitting a proposal, including a cover letter and estimated timeline.

FR-7 The system shall allow freelancers to track the status of their applications (e.g., pending, accepted, rejected).

FR-8 The system shall notify freelancers when their application is accepted or rejected.

FR-9 The system shall allow users to create and update their profile, including personal information, skills, portfolio, and work history.

FR-10 The system shall provide a messaging feature for clients and freelancers to communicate about job details.

FR-11 The system shall allow clients to rate and review freelancers after job completion.

FR-12 The system shall allow freelancers to rate and review clients after job completion.

FR-13 The system shall display average ratings and reviews on user profiles.

FR-14 The system shall integrate a payment gateway to facilitate secure transactions between clients and freelancers.

FR-15 The system shall allow clients to release payment to freelancers upon job completion.

FR-16 The system shall provide an admin dashboard to view and manage all users, jobs, and transactions.

FR-17 The system shall allow clients to mark a job as completed once the freelancer delivers the work.

FR-18 The system shall allow freelancers to upload and showcase their previous work samples in their profile.

FR-19 The system shall display the freelancer's portfolio to clients when viewing their profile.

FR-20 The system shall allow clients and freelancers to track the status of a job (e.g., open, in progress, completed).

3 Non-Functional Requirements

3.1 Product Requirements

NFR-1 The system shall support at least 500 concurrent users.

NFR-2 The system shall have a response time of less than 2 seconds for 95% of requests.

NFR-3 The system shall be available 99.9% of the time.

NFR-4 The system shall have a user-friendly interface with a consistent design across all pages.

3.2 Organizational Requirements

NFR-1 The system shall be developed using Agile methodology with bi-weekly sprints.

NFR-2 The system shall be documented thoroughly, including user manuals and technical documentation.

3.3 External Requirements

NFR-1 The system shall integrate with third-party payment gateways (e.g., PayPal, Stripe) for secure transactions.

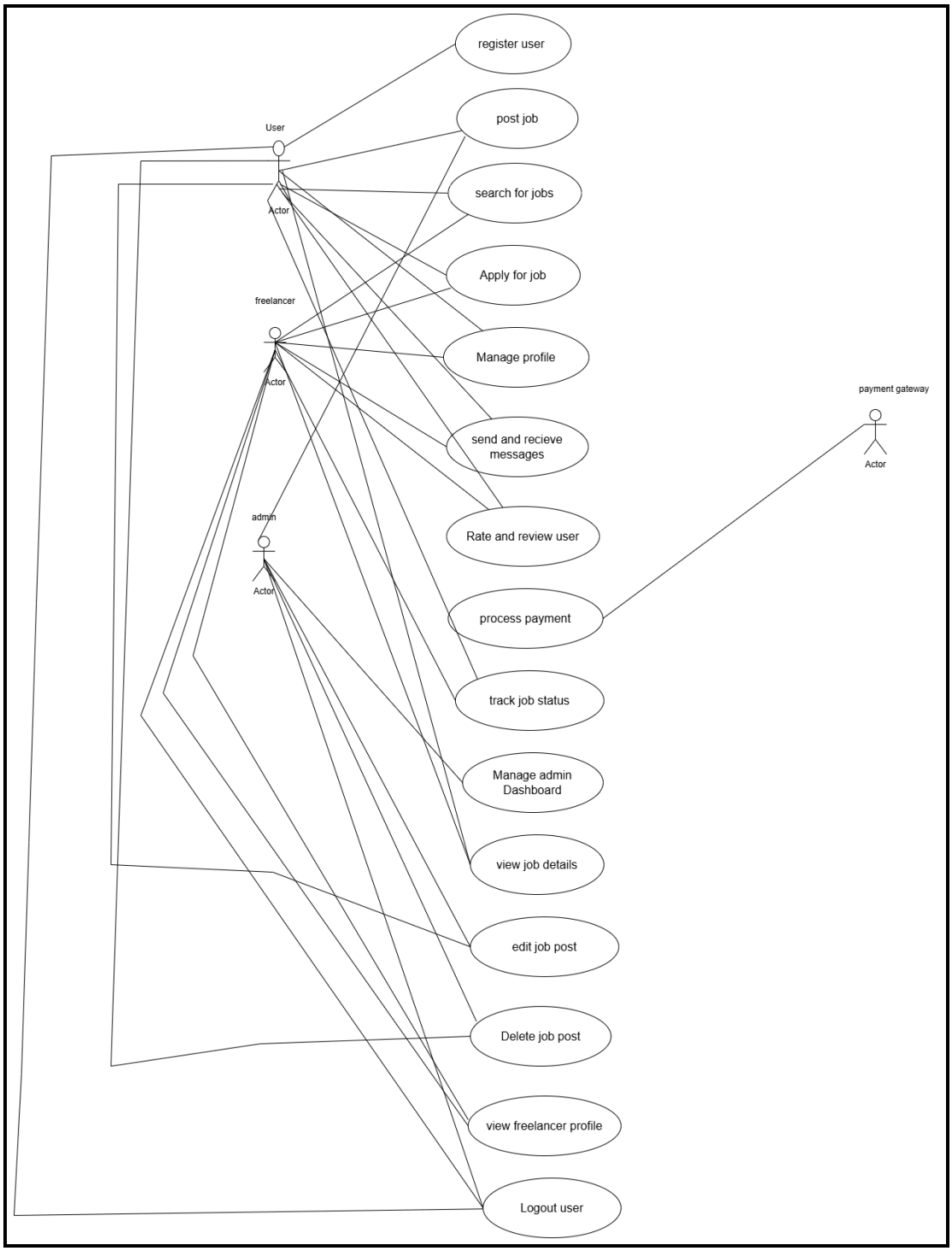
NFR-2 The system shall be compatible with major web browsers (e.g., Chrome).

NFR-3 The system shall use HTTP for secure communication between the client and server.

4 List of All Use Cases

1. Register User
2. Post Job
3. Search for Jobs
4. Apply for Job
5. Manage Profile
6. Send and Receive Messages
7. Rate and Review User
8. Process Payment
9. Track Job Status
10. Manage Admin Dashboard
11. View Job Details
12. Edit Job Post
13. Delete Job Post
14. View Freelancer Profile
15. Logout User

5 Use Case Diagram



6 User Stories

6.1 Register User

- **As a freelancer**, I want to create a profile so that I can showcase my skills.
- **Pre-condition**: User must register.
- **Flow**: User enters details, uploads a profile picture.
- **Post-condition**: Profile saved in the database.

6.2 Post Job

- **As an employer**, I want to post jobs so that freelancers can apply.
- **Pre-condition**: The employer must register.
- **Flow**: The employer fills in the details of the job, sets a budget, and posts the job.
- **Post-condition**: Job is visible to freelancers.

6.3 Search for Jobs

- **As a freelancer**, I want to search for jobs so that I can apply for relevant opportunities.
- **Pre-condition**: Freelancer must be registered and logged in.
- **Flow**: Freelancer searches for jobs using filters and applies to selected jobs.
- **Post-condition**: Application is submitted to the employer.

6.4 Apply for Job

- **As a freelancer**, I want to apply for a job so that I can be considered for the opportunity.
- **Pre-condition**: Freelancer must be registered, logged in, and have a complete profile.
- **Flow**: Freelancer submits a proposal (cover letter, bid amount, timeline) for the selected job.
- **Post-condition**: Application is sent to the employer and saved in the database.

6.5 Manage Profile

- **As a user**, I want to manage my profile so that I can keep my information up-to-date.
- **Pre-condition**: User must be registered and logged in.
- **Flow**: User edits profile details (e.g., skills, portfolio, work history) and saves changes.
- **Post-condition**: Updated profile is saved in the database.

6.6 Send and Receive Messages

- **As a user**, I want to send and receive messages so that I can communicate with employers or freelancers.
- **Pre-condition**: User must be registered and logged in.
- **Flow**: User sends a message to another user and receives replies.
- **Post-condition**: Messages are stored and displayed in the chat history.

6.7 Rate and Review User

- **As an employer**, I want to review freelancer profiles so that I can hire the best candidate.
- **Pre-condition**: Employer must be registered and logged in.
- **Flow**: Employer browses freelancer profiles, views ratings and reviews.
- **Post-condition**: Employers can shortlist or contact freelancers for further discussion.

6.8 Process Payment

- **As an employer**, I want to process payments securely so that freelancers receive their earnings.
- **Pre-condition**: Employer must have a job marked as completed.
- **Flow**: Employer initiates payment through the integrated payment gateway.
- **Post-condition**: Payment is processed, and freelancer receives the amount.

6.9 Track Job Status

- **As a user**, I want to track the status of a job so that I can stay updated on its progress.
- **Pre-condition**: User must be registered, logged in, and have an active job.
- **Flow**: User views the job status (e.g., open, in progress, completed) in their dashboard.
- **Post-condition**: Job status is displayed to the user.

6.10 Manage Admin Dashboard

- **As an admin**, I want to manage users, jobs, and disputes so that I can maintain the platform's integrity.
- **Pre-condition**: Admin must be logged in.
- **Flow**: Admin views and manages user accounts, jobs, and disputes.
- **Post-condition**: Platform data is updated and maintained.

6.11 View Job Details

- **As a freelancer**, I want to view job details so that I can decide whether to apply.
- **Pre-condition**: Freelancer must be registered and logged in.
- **Flow**: Freelancer clicks on a job post to view its full description, requirements, and budget.
- **Post-condition**: Job details are displayed to the freelancer.

6.12 Edit Job Post

- **As an employer**, I want to edit my job post so that I can update or correct its information.
- **Pre-condition**: Employer must be registered, logged in, and own the job post.
- **Flow**: Employer selects the job post, edits the required fields, and saves the changes.
- **Post-condition**: Updated job post is saved in the database.

6.13 Delete Job Post

- **As an employer**, I want to delete a job post so that I can remove outdated or filled job vacancies.
- **Pre-condition**: Employer must be registered, logged in, and own the job post.
- **Flow**: Employer selects the job post and confirms deletion.
- **Post-condition**: Job post is removed from the platform.

6.14 View Freelancer Profile

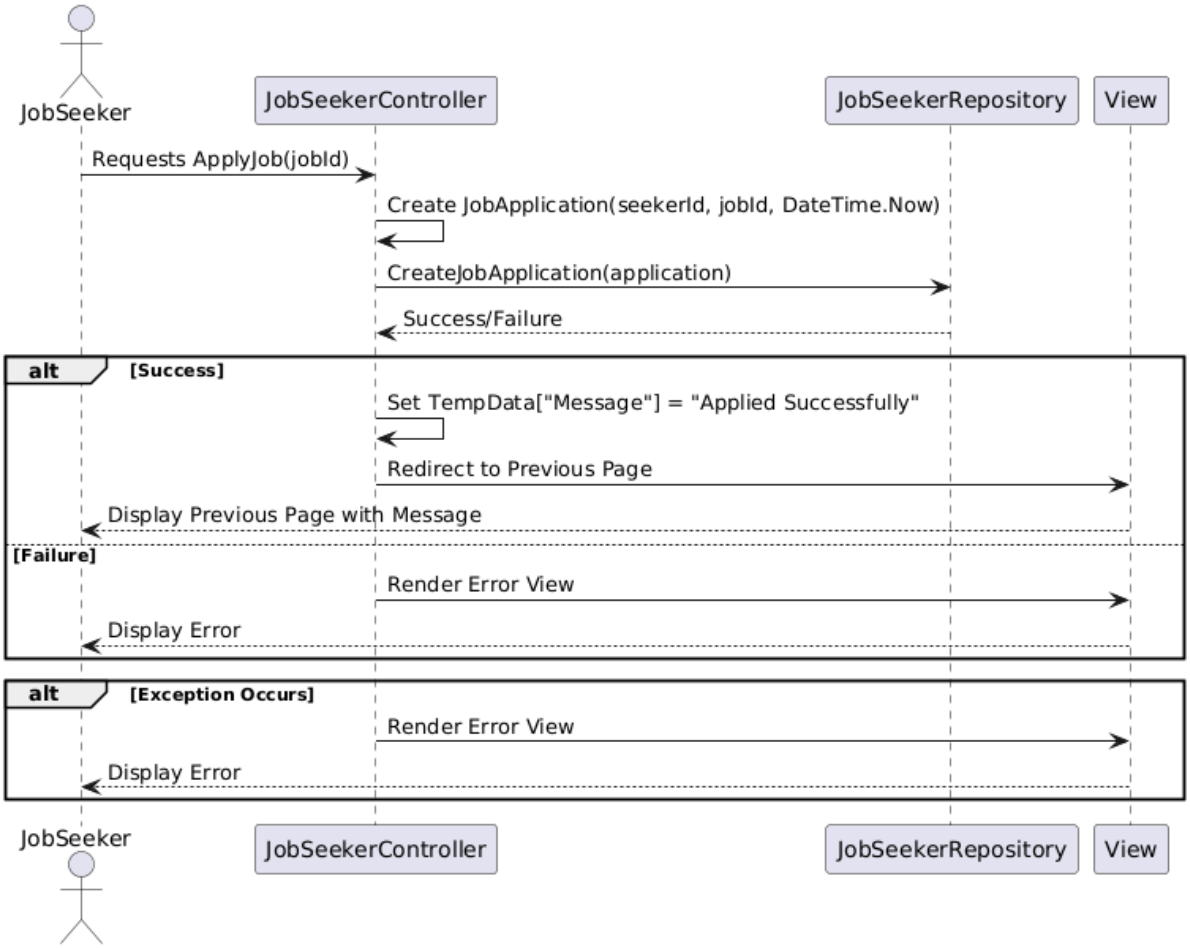
- **As an employer**, I want to view freelancer profiles so that I can evaluate suitable candidates.
- **Pre-condition**: Employer must be registered and logged in.
- **Flow**: Employer searches or selects a freelancer and views their profile information.
- **Post-condition**: Freelancer profile details are displayed to the employer.

6.15 Logout User

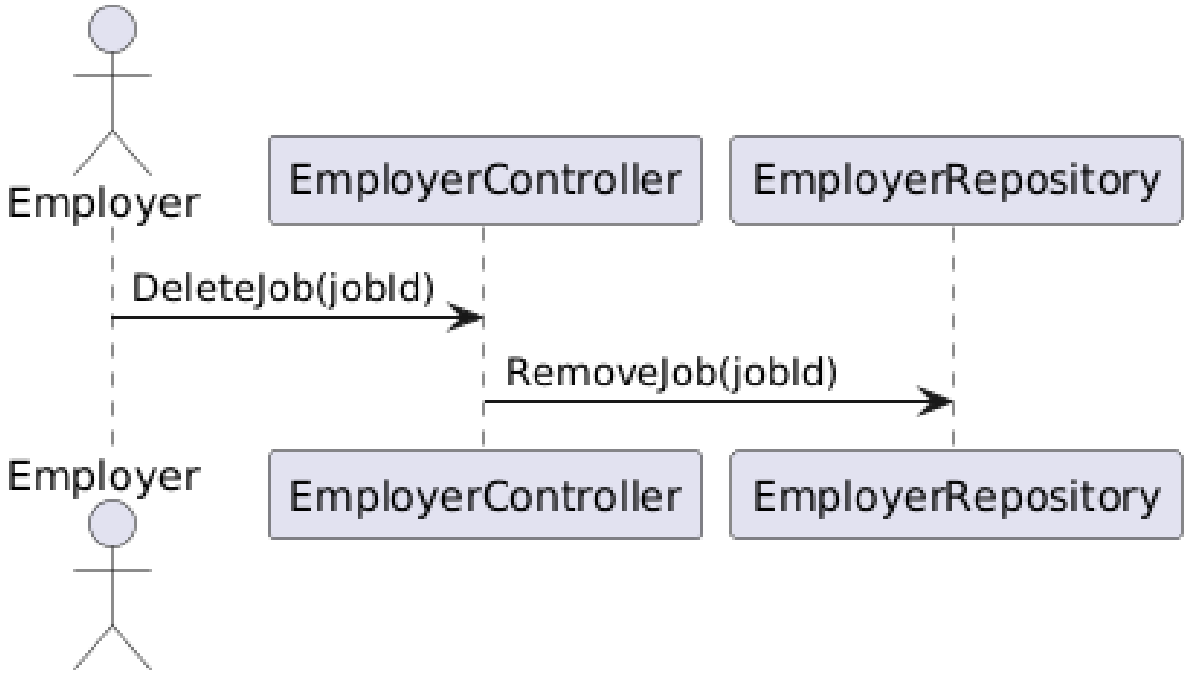
- **As a user**, I want to logout from the platform so that I can end my session securely.
- **Pre-condition**: User must be logged in.
- **Flow**: User clicks on the logout option to end the current session.
- **Post-condition**: User is logged out and redirected to the homepage or login page.

7 Sequence Diagrams

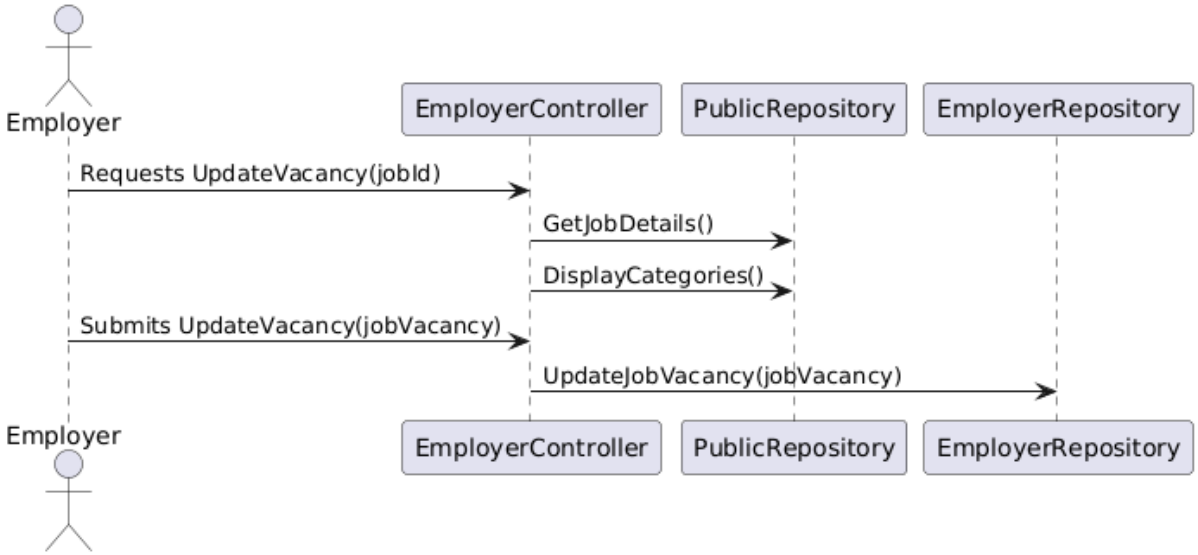
7.1 Apply Job



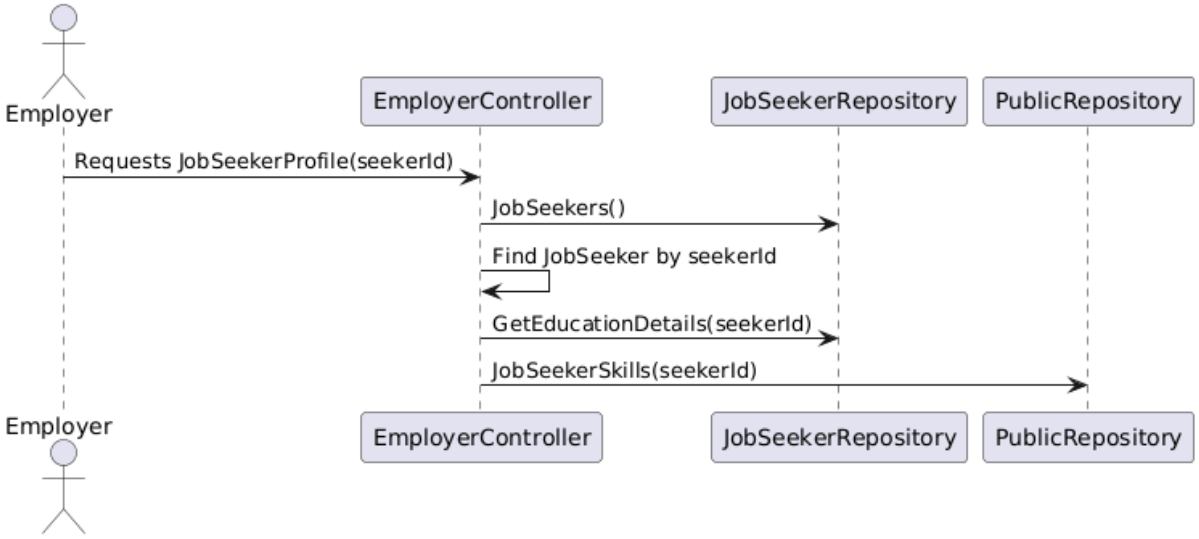
7.2 Delete Job Post



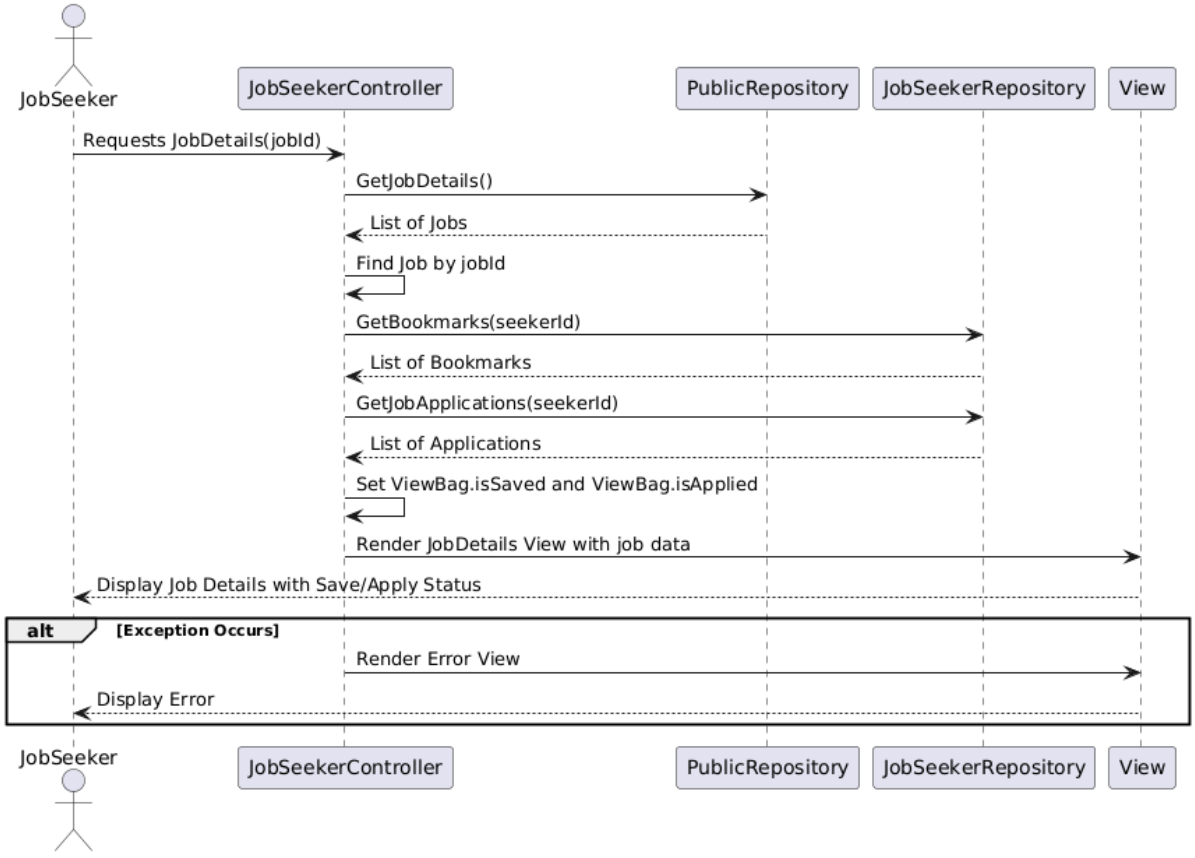
7.3 Edit Job Post



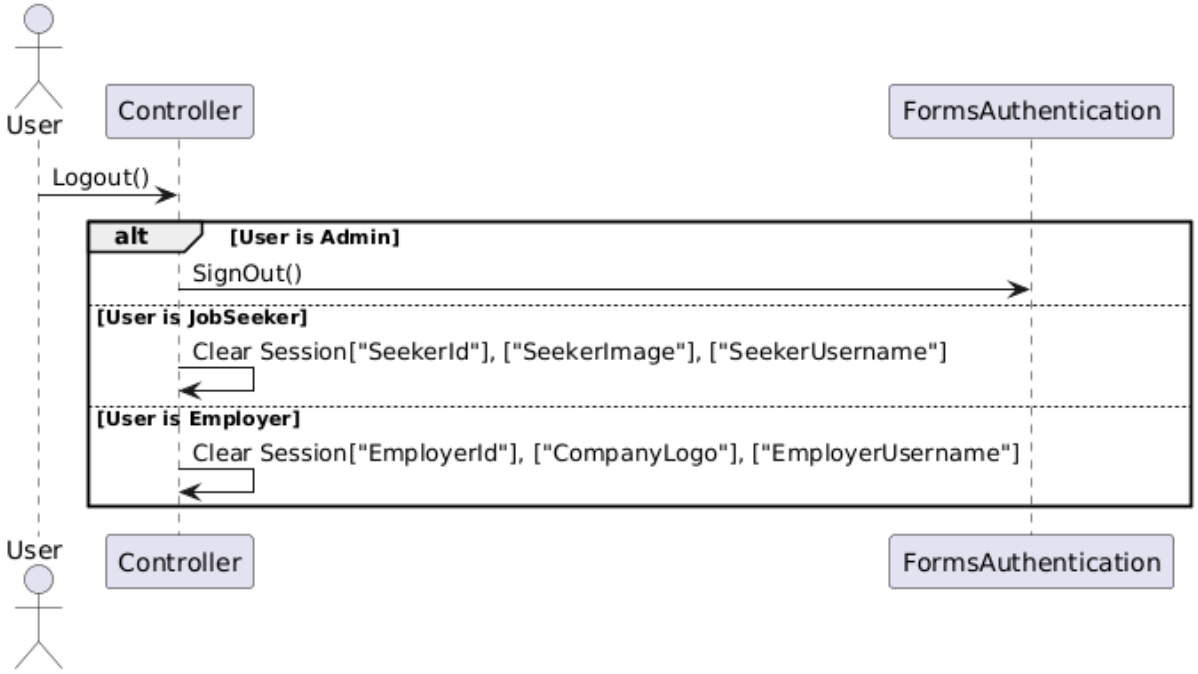
7.4 Freelancer Profile



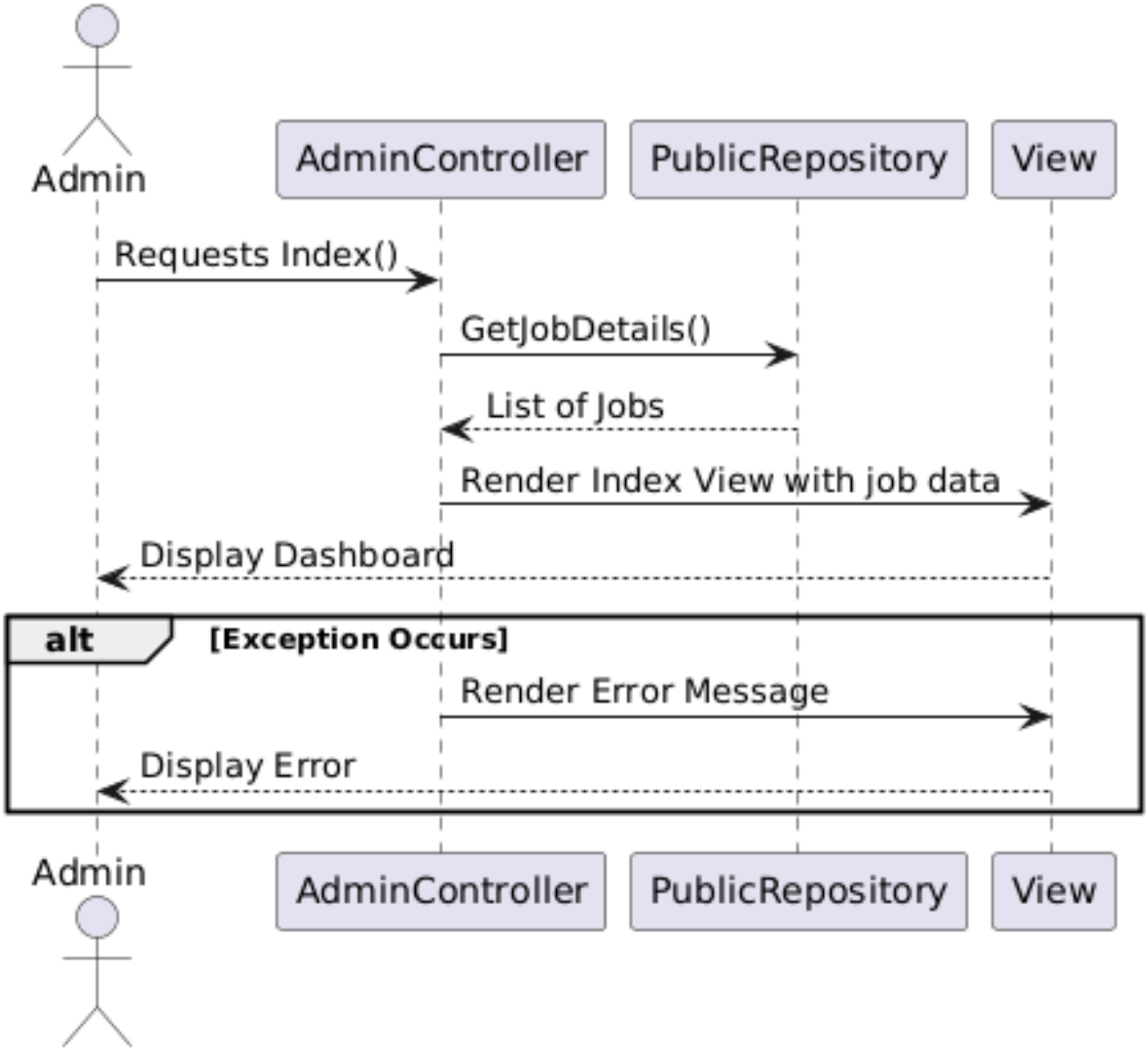
7.5 Job Details



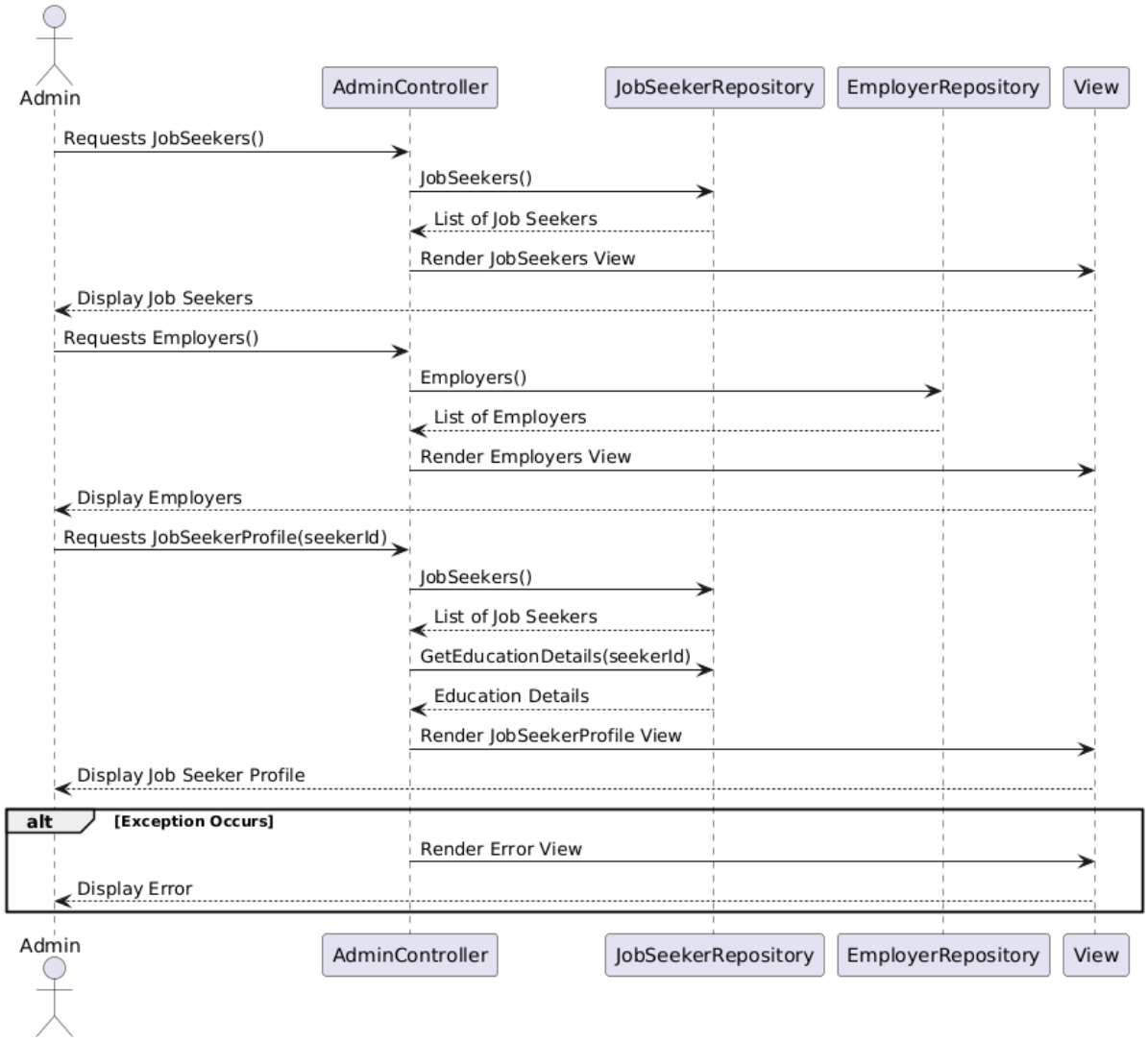
7.6 Logout User



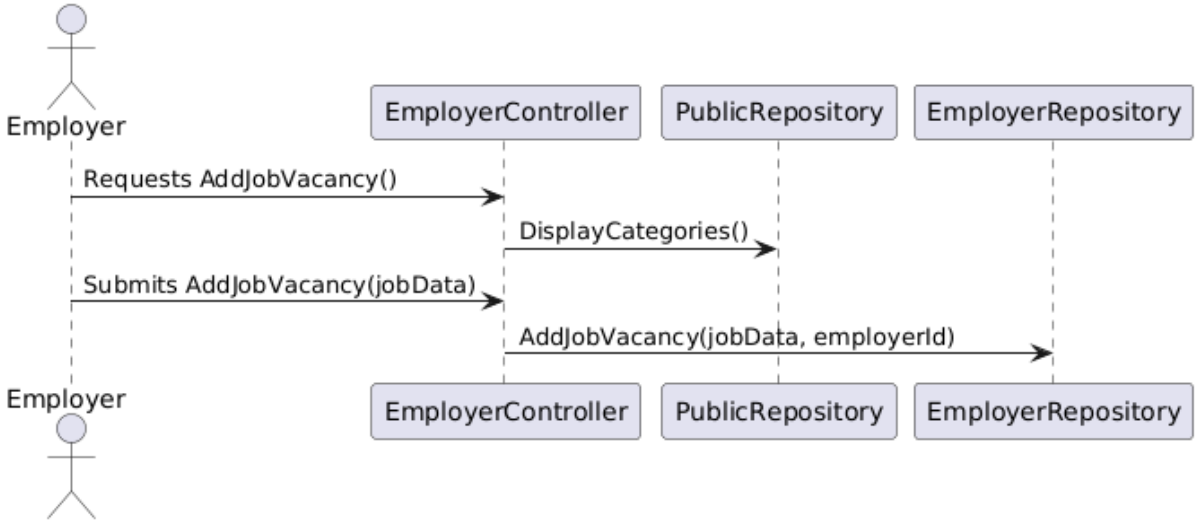
7.7 Manage Admin Dashboard



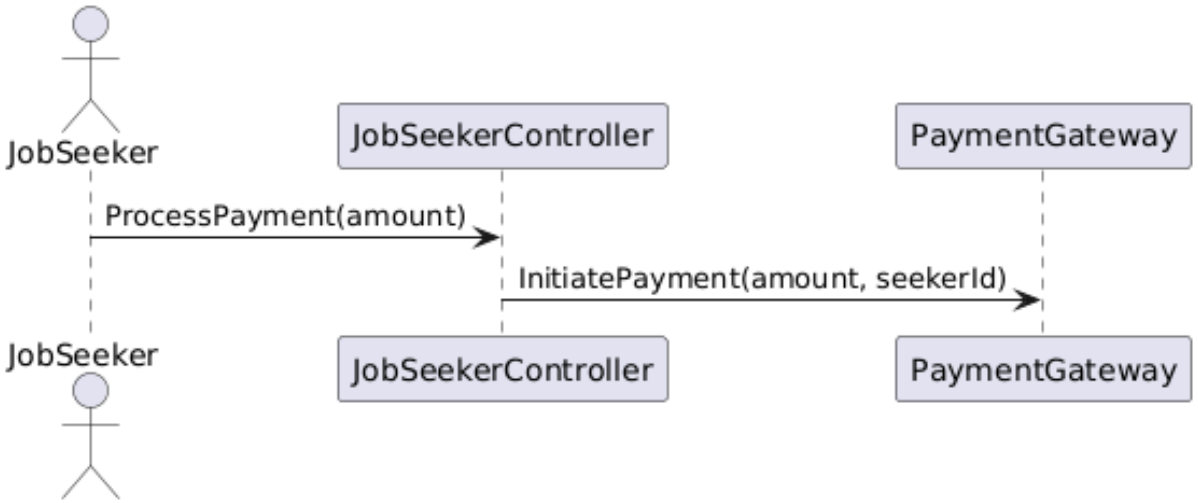
7.8 Manage Profiles



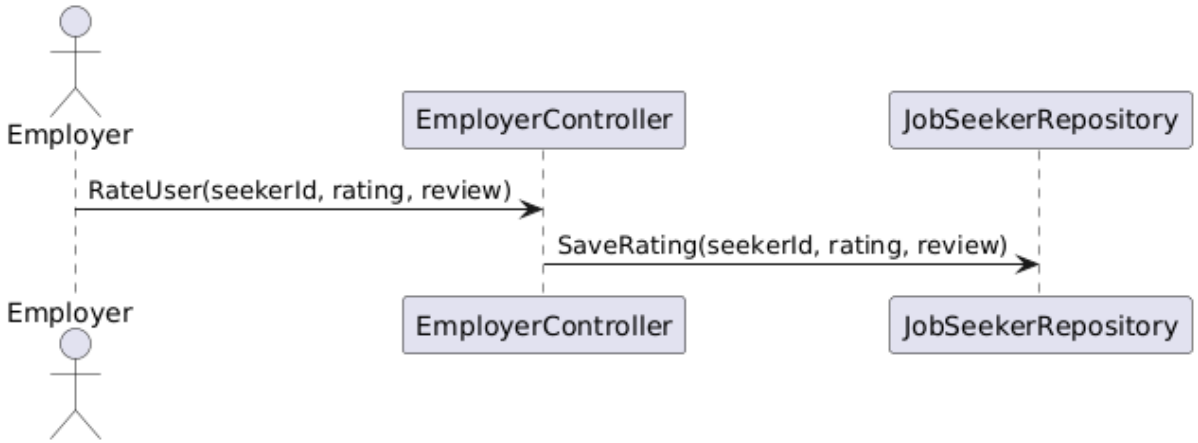
7.9 Post Job



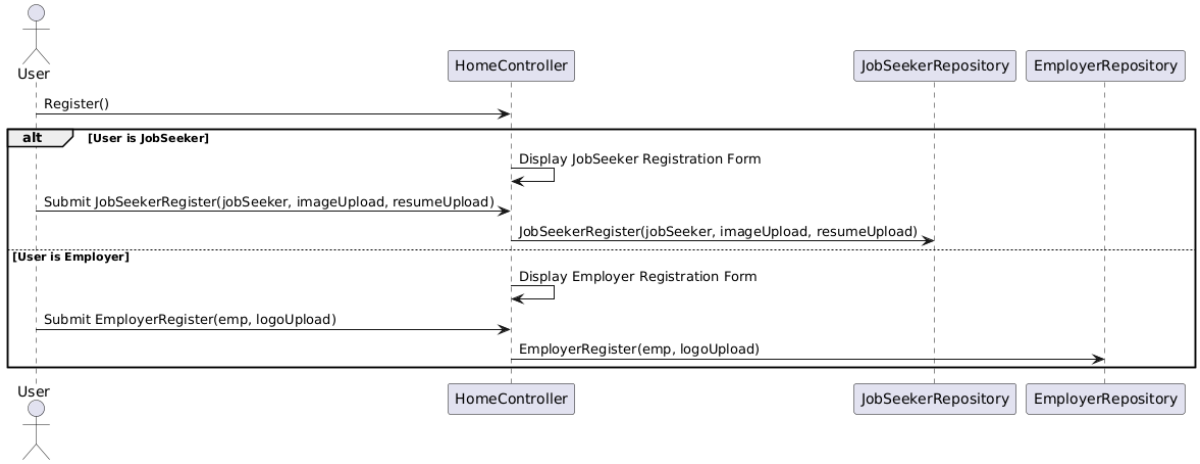
7.10 Process Payment



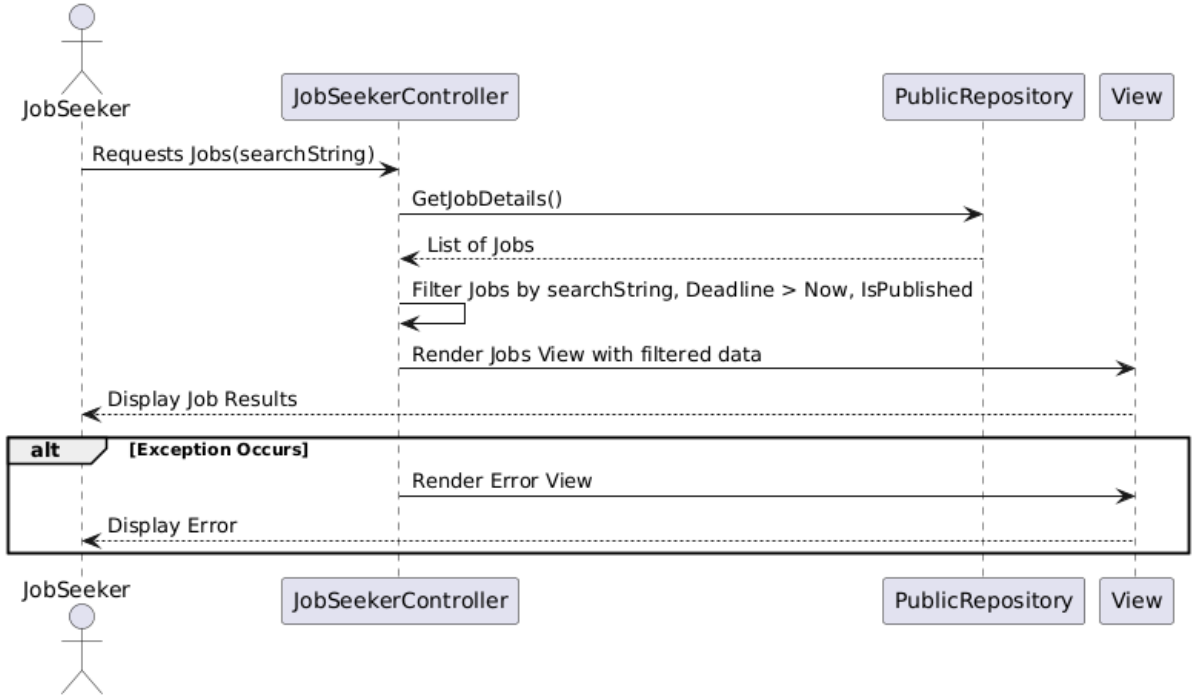
7.11 Rate and Review User



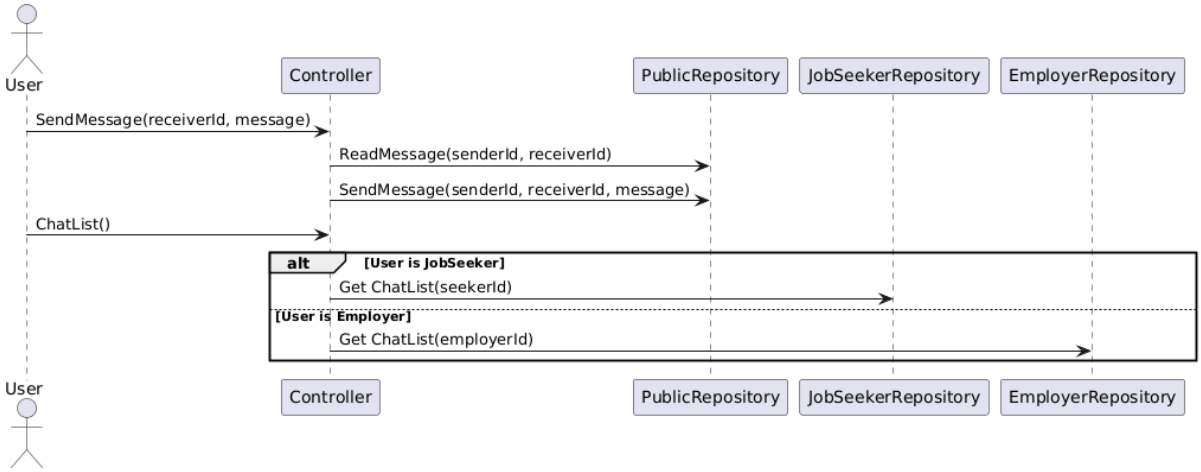
7.12 Register User



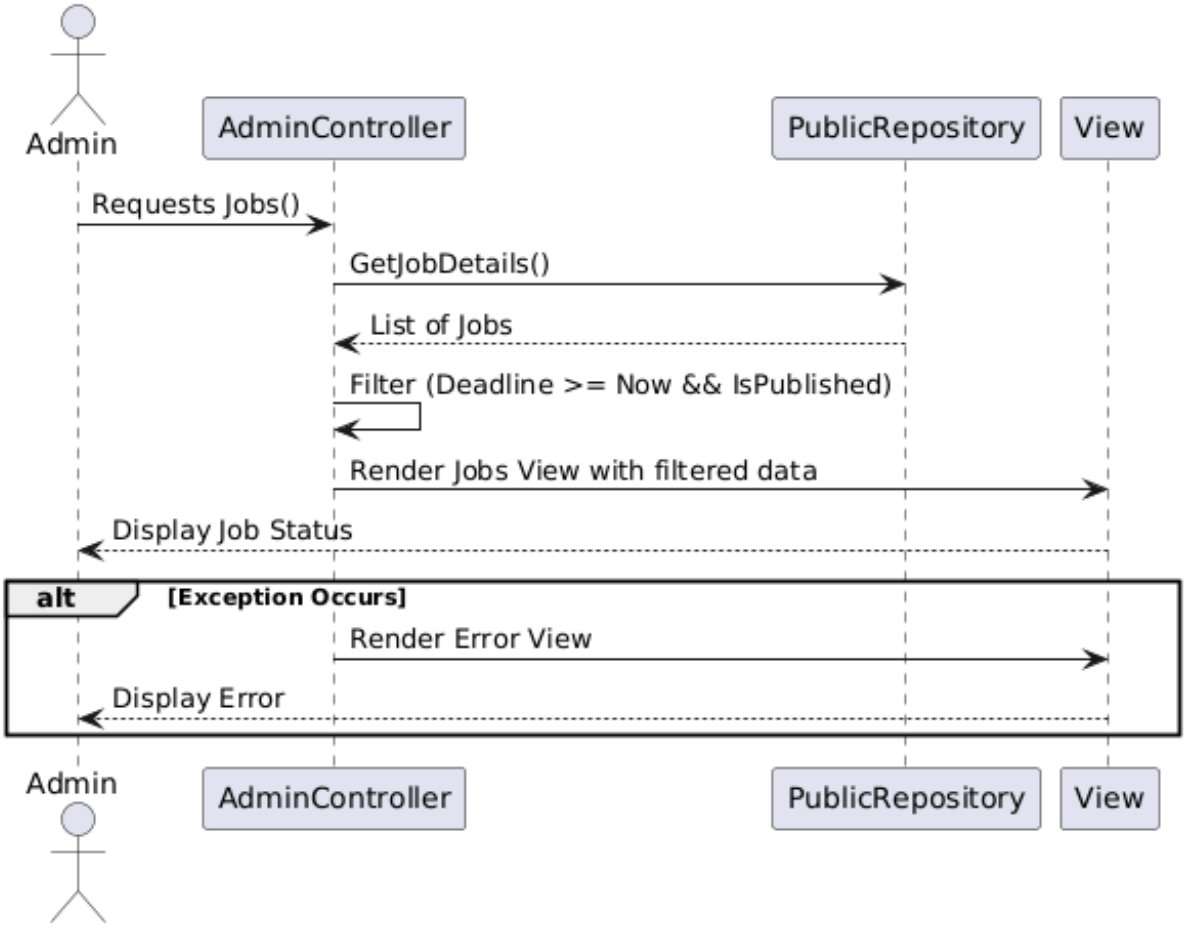
7.13 Search for Job



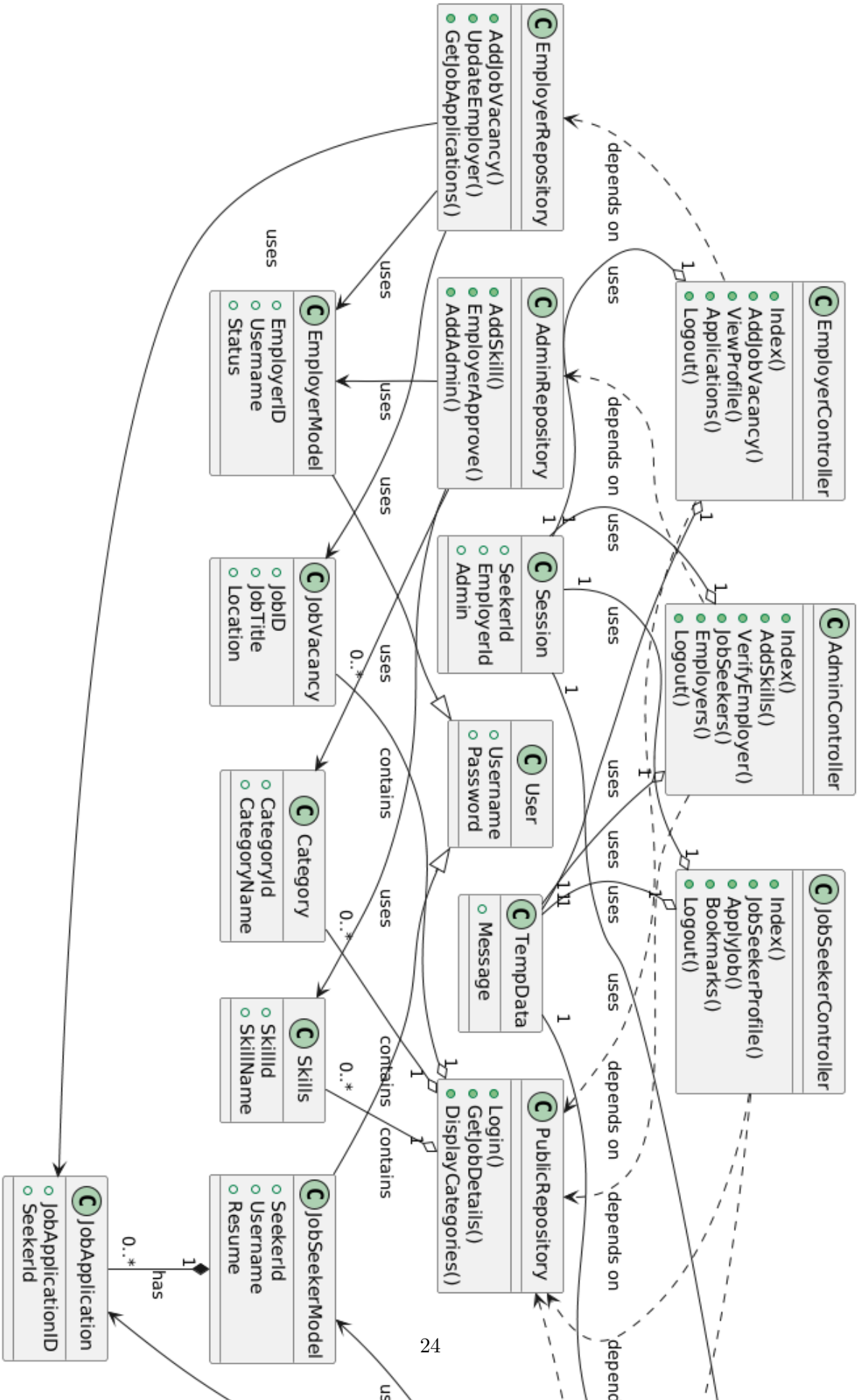
7.14 Send and Receive Messages



7.15 Track Job Status



8 Class Diagrams



9 Sprint Backlog

9.1 Sprint 01

ID	User Story	Tasks	Assigned to	Status	Estimated Effort (Hours)	Day 1	Day 2	Day 3
1.1	As a User, I want to register as a freelancer so that I can apply for jobs.	Create a Freelancer Table in the database	Shayan	Completed	1	1	0	0
1.2		Develop Freelancer Registration UI	Waseem	Completed	3	1	2	0
1.3		Implement backend logic for freelancer registration	Waseem	Completed	1	0.5	0	0.5
2.1	As a User, I want to register as a client so that I can post jobs.	Create a Client Table in the database	Laiba	Completed	2	1	1	0
2.2		Develop Client Registration UI	Laiba	Completed	2	0	1	1
3.1	As a User, I want to log in so that I can access my account.	Develop Login UI	Shayan	Completed	7	4	3	0
3.2		Implement authentication backend	Waseem	Completed	7	3	3	1
3.3		Design Job Posting UI	Shayan	Completed	7	2	3	2

9.2 Sprint 02

Sprint 2 Backlog

ID	User Story	Tasks	Assigned to	Status	Hours	Day 1	Day 2	Day 3
4.1	Apply for Job	Implement Proposal UI	Waseem	Completed	3	2	1	0
4.2		Develop Backend for Proposal	Laiba	Completed	4	0	2	2
5.1	Manage Profile	Create Profile Edit Page	Shayan	Completed	3	1	2	0
5.2		Backend for Profile Updates	Waseem	Completed	4	2	2	0
6.1	Messaging Feature	Design Messaging UI	Laiba	Completed	3	1	1	1
6.2		Backend for Messaging	Shayan	Completed	5	1	2	2
7.1	Review Freelancer	Develop Rating UI	Waseem	Completed	3	2	1	0
7.2		Backend for Ratings	Laiba	Completed	4	0	2	2
8.1	Process Payment	Integrate Payment Gateway	Shayan	Completed	6	2	2	2
8.2		Backend for Payment Processing	Waseem	Completed	5	1	2	2

Table 1: Sprint 2 Backlog

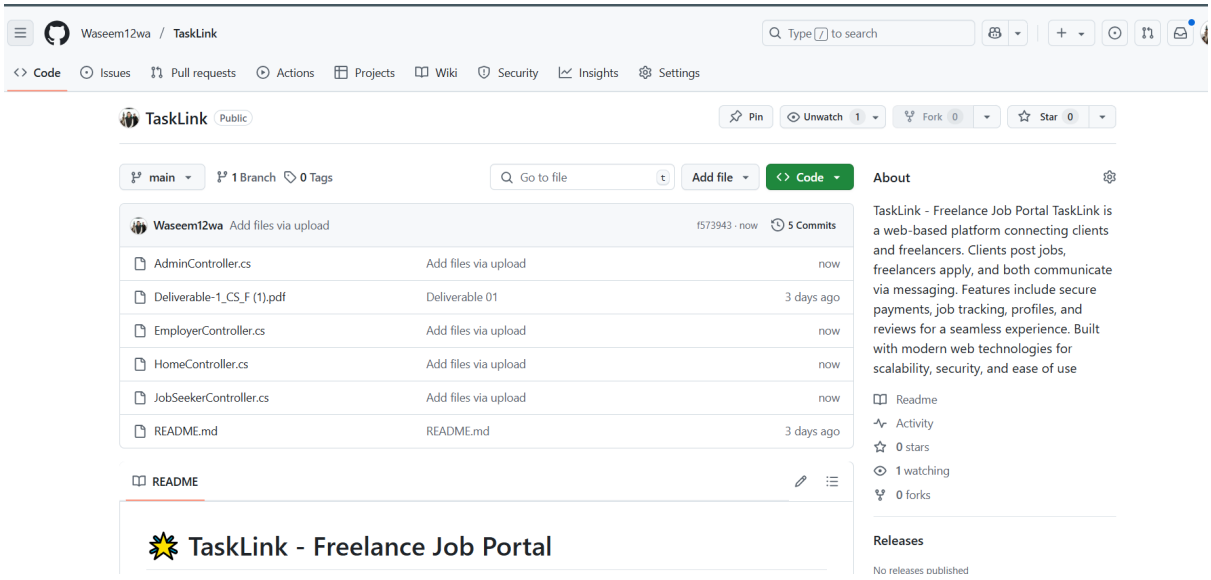
10 Product Backlog

Product Backlog

ID	User Story	Priority	Estimated Hours	Status
6.1	Register User	High	4	Completed
6.2	Post Job	High	5	Completed
6.3	Search for Jobs	Medium	4	Completed
6.4	Apply for Job	High	6	Completed
6.5	Manage Profile	Medium	4	Completed
6.6	Send and Receive Messages	Medium	4	Completed
6.7	Rate and Review User	Low	3	Completed
6.8	Process Payment	High	5	Completed
6.9	Track Job Status	Medium	4	Completed
6.10	Manage Admin Dashboard	High	6	Completed

Table 1: Product Backlog

11 Version Control



You can access the GitHub repository here: GitHub Repository: TaskLink

12 Project Plan

12.1 Work Breakdown Structure (WBS)

The WBS is hierarchical, covering all tasks for D1, D2, and D3, with deliverables, responsibilities, and durations.

12.1.1 1.0 Project Initiation (D1)

- **1.1 Team Setup**
  - **Description**: Create team info (name, logo, bios, GitHub accounts, roles).
  - **Responsible**: Laiba, Waseem Zahid, Shayan Salam.
  - **Duration**: 2 days (Feb 5–6, 2025).
  - **Deliverable**: Team info PDF (D1).
- **1.2 Define Scope & Vision**
  - **Description**: Outline project goals, stakeholders (job seekers, employers, admins), and vision.
  - **Responsible**: Waseem Zahid (Scrum Master).
  - **Duration**: 1 day (Feb 7, 2025).
  - **Deliverable**: Vision document (D1).
- **1.3 Gather Requirements**
  - **Description**: Collect functional (registration, listings) and non-functional (responsive UI) requirements, user stories.
  - **Responsible**: Laiba (Requirement Analyst).
  - **Duration**: 2 days (Feb 8–9, 2025).
  - **Deliverable**: Requirements, user stories (D1).

12.1.2 2.0 Sprint 1 Setup (D1)

- **2.1 Sprint 1 Backlog**
  - **Description**: Define Sprint 1 tasks (user registration, job listing) and user stories.
  - **Responsible**: Waseem Zahid, Laiba.
  - **Duration**: 1 day (Feb 10, 2025).
  - **Deliverable**: Sprint 1 backlog (D1).
- **2.2 Trello Board Setup**
  - **Description**: Create Trello board, upload Sprint 1 backlog, take initial snapshot.
  - **Responsible**: Waseem Zahid (Scrum Master).
  - **Duration**: 1 day (Feb 10, 2025).
  - **Deliverable**: Trello snapshot 1 (D1).

12.1.3 3.0 Sprint 1 Development (D1, D2)

- **3.1 UI Design (Sprint 1)**
  - **Description**: Design wireframes for registration and listing pages (HTML, CSS, Bootstrap).
  - **Responsible**: Laiba (UI Designer).
  - **Duration**: 2 days (Feb 11–12, 2025).
  - **Deliverable**: Wireframes (D1, D2).
- **3.2 Database Design (Sprint 1)**
  - **Description**: Design SQL Server schema for Users, Jobs tables.
  - **Responsible**: Shayan Salam (Database Administrator).
  - **Duration**: 2 days (Feb 11–12, 2025).
  - **Deliverable**: ERD (D1, D2).
- **3.3 Frontend Development (Sprint 1)**
  - **Description**: Implement registration, listing UI (HTML, CSS, Bootstrap, JavaScript).
  - **Responsible**: Laiba, Waseem Zahid (Developers).
  - **Duration**: 3 days (Feb 14–16, 2025).
  - **Deliverable**: Frontend code (D2).
- **3.4 Backend Development (Sprint 1)**
  - **Description**: Develop registration, listing APIs (C#, Node.js).
  - **Responsible**: Waseem Zahid, Shayan Salam (Developers).
  - **Duration**: 3 days (Feb 14–16, 2025).
  - **Deliverable**: Backend code (D2).
- **3.5 Database Implementation (Sprint 1)**
  - **Description**: Set up SQL Server tables for Users, Jobs.
  - **Responsible**: Shayan Salam.
  - **Duration**: 2 days (Feb 15–16, 2025).
  - **Deliverable**: Database (D2).

12.1.4 4.0 Sprint 2 Development (D2)

- **4.1 Sprint 2 Backlog**
  - **Description**: Define Sprint 2 tasks (job applications, search) and user stories.
  - **Responsible**: Waseem Zahid, Laiba.
  - **Duration**: 1 day (Feb 17, 2025).
  - **Deliverable**: Sprint 2 backlog (D2).
- **4.2 UI Design (Sprint 2)**
  - **Description**: Design wireframes for application, search pages (Bootstrap).
  - **Responsible**: Laiba (UI Designer).
  - **Duration**: 2 days (Feb 18–19, 2025).

- **Deliverable**: Wireframes (D2).

- **4.3 Database Design (Sprint 2)**
  - **Description**: Add Applications table to SQL Server schema.
  - **Responsible**: Shayan Salam.
  - **Duration**: 2 days (Feb 18–19, 2025).
  - **Deliverable**: Updated ERD (D2).
- **4.4 Frontend Development (Sprint 2)**
  - **Description**: Implement application, search UI (JavaScript, Bootstrap).
  - **Responsible**: Laiba, Waseem Zahid.
  - **Duration**: 3 days (Feb 20–22, 2025).
  - **Deliverable**: Frontend code (D2).
- **4.5 Backend Development (Sprint 2)**
  - **Description**: Develop application, search APIs (C#, Node.js).
  - **Responsible**: Waseem Zahid, Shayan Salam.
  - **Duration**: 3 days (Feb 20–22, 2025).
  - **Deliverable**: Backend code (D2).
- **4.6 Database Implementation (Sprint 2)**
  - **Description**: Set up Applications table in SQL Server.
  - **Responsible**: Shayan Salam.
  - **Duration**: 2 days (Feb 23–24, 2025).
  - **Deliverable**: Database (D2).

12.1.5 5.0 Deliverable 2 Compilation (D2)

- **5.1 Software Requirements Specification**
  - **Description**: Create SRS (IEEE 830) for functional/non-functional requirements.
  - **Responsible**: Laiba (Requirement Analyst).
  - **Duration**: 2 days (Feb 25–26, 2025).
  - **Deliverable**: SRS (D2).
- **5.2 Product Backlog**
  - **Description**: Update product backlog with all features.
  - **Responsible**: Waseem Zahid.
  - **Duration**: 1 day (Feb 25, 2025).
  - **Deliverable**: Product backlog (D2).
- **5.3 Trello Board Updates**
  - **Description**: Update Trello with Sprint 1/2 tasks, take 3 snapshots.
  - **Responsible**: Waseem Zahid.
  - **Duration**: 2 days (Feb 17–26, 2025).
  - **Deliverable**: Trello snapshots (D2).

- **5.4 GitHub Repository**
  - **Description**: Commit Sprint 1/2 code, provide revision history.
  - **Responsible**: Waseem Zahid, Shayan Salam.
  - **Duration**: 2 days (Feb 20–26, 2025).
  - **Deliverable**: GitHub link (D2).
- **5.5 Compile Deliverable 2**
  - **Description**: Assemble D2 (SRS, backlogs, Trello, GitHub, code).
  - **Responsible**: Laiba, Waseem Zahid, Shayan Salam.
  - **Duration**: 2 days (Feb 27–28, 2025).
  - **Deliverable**: D2 PDF.

12.1.6 6.0 Deliverable 3 (D3)

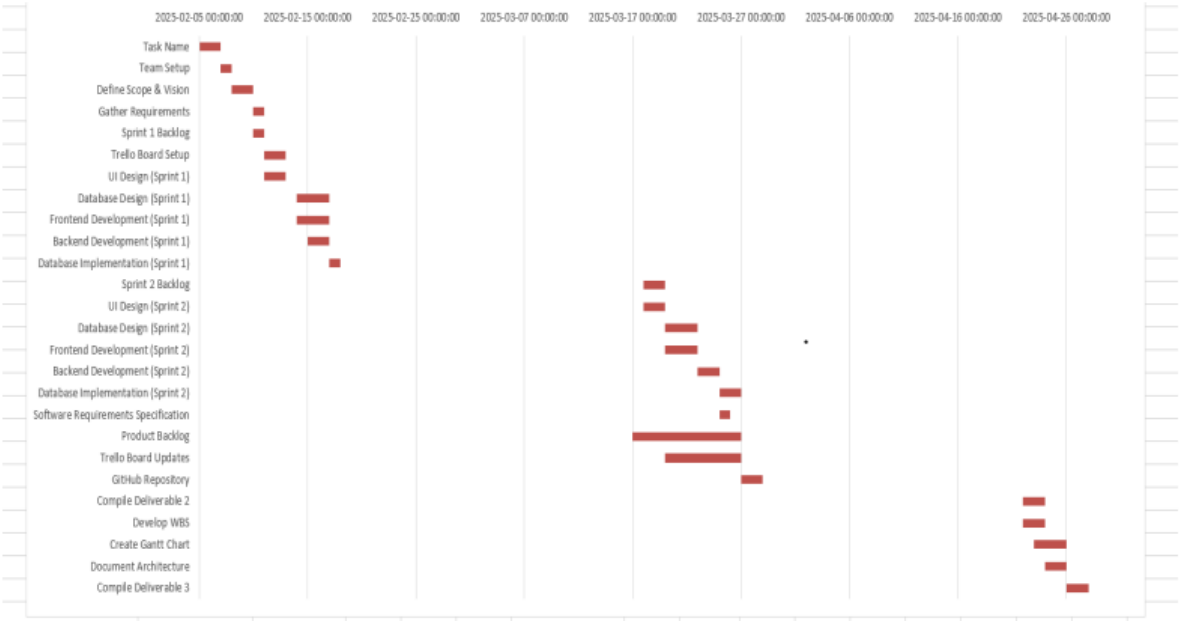
- **6.1 Develop WBS**
  - **Description**: Create WBS for entire project (Feb 5–Apr 28, 2025).
  - **Responsible**: Waseem Zahid (Scrum Master).
  - **Duration**: 2 days (Mar 2–3, 2025).
  - **Deliverable**: WBS (D3: Software Project Plan).
- **6.2 Create Gantt Chart**
  - **Description**: Build Gantt chart in Excel for project timeline.
  - **Responsible**: Laiba (UI Designer).
  - **Duration**: 2 days (Mar 4–5, 2025).
  - **Deliverable**: Gantt chart (D3: Software Project Plan).
- **6.3 Document Architecture**
  - **Description**: Define architecture (C#, Node.js, SQL Server, Bootstrap).
  - **Responsible**: Shayan Salam, Waseem Zahid.
  - **Duration**: 3 days (Mar 6–8, 2025).
  - **Deliverable**: Architecture diagram (D3: System Architecture).
- **6.4 Compile Deliverable 3**
  - **Description**: Assemble D3 (project plan, architecture) into PDF.
  - **Responsible**: Laiba, Waseem Zahid, Shayan Salam.
  - **Duration**: 2 days (Mar 9–10, 2025).
  - **Deliverable**: D3 PDF.

12.1.7 7.0 Final Report Preparation

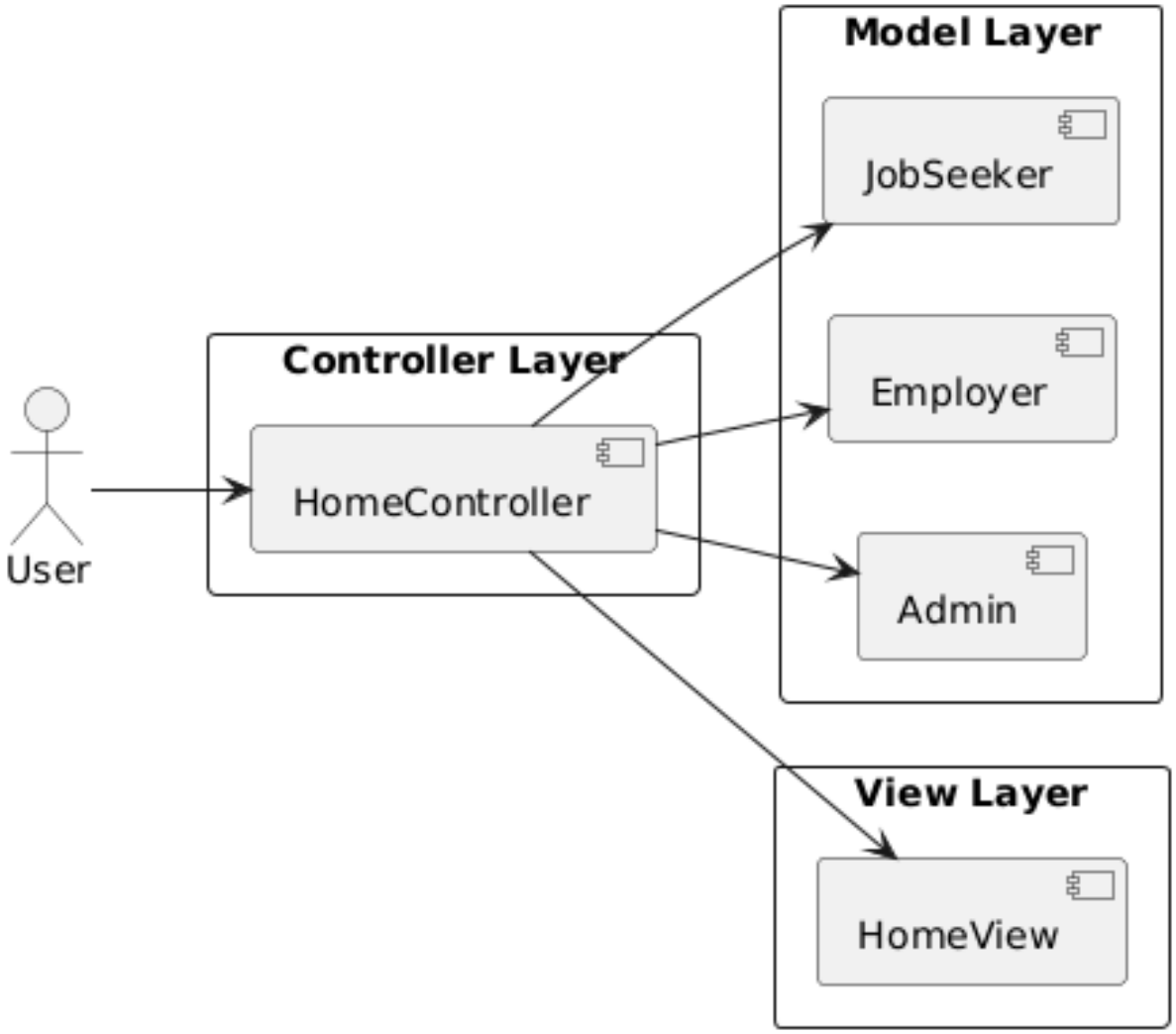
- **7.1 Compile Final Report**
  - **Description**: Combine D1–D3, add user stories, backlogs, Trello, screenshots, work division for Apr 28 submission.
  - **Responsible**: Laiba, Waseem Zahid, Shayan Salam.
  - **Duration**: 3 days (Apr 26–28, 2025).

– **Deliverable**: Final Report PDF.

GANT_CHART:



13 Architecture Design



13.1 MVC (Model-View-Controller)

The architecture style used for the JB Portal System is the **Model-View-Controller (MVC)** Architecture.

13.1.1 Model

The **Model** represents the core data and business logic of the system.
Examples in the JB Portal include classes such as:

- Admin.cs
- Employer.cs
- JobSeeker.cs

13.1.2 View

The **View** handles the presentation layer (what users see).
In C# console or web projects, this could include:

- Razor pages
- HTML files
- Front-end templates

You can mention that there are views such as:

- Home page
- Views for Admin, Employer, and Job Seeker interactions

13.1.3 Controller

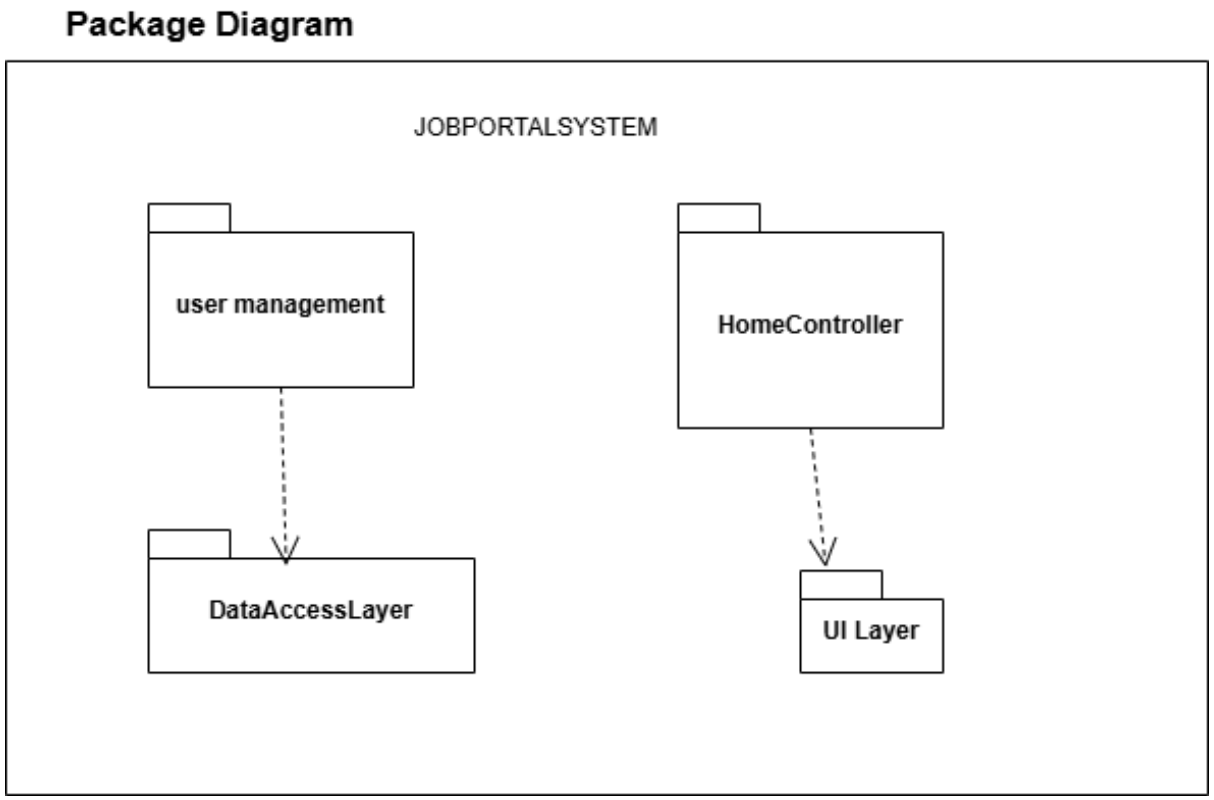
The **Controller** acts as the bridge between the Model and the View.
An example in the JB Portal is:

- HomeController.cs

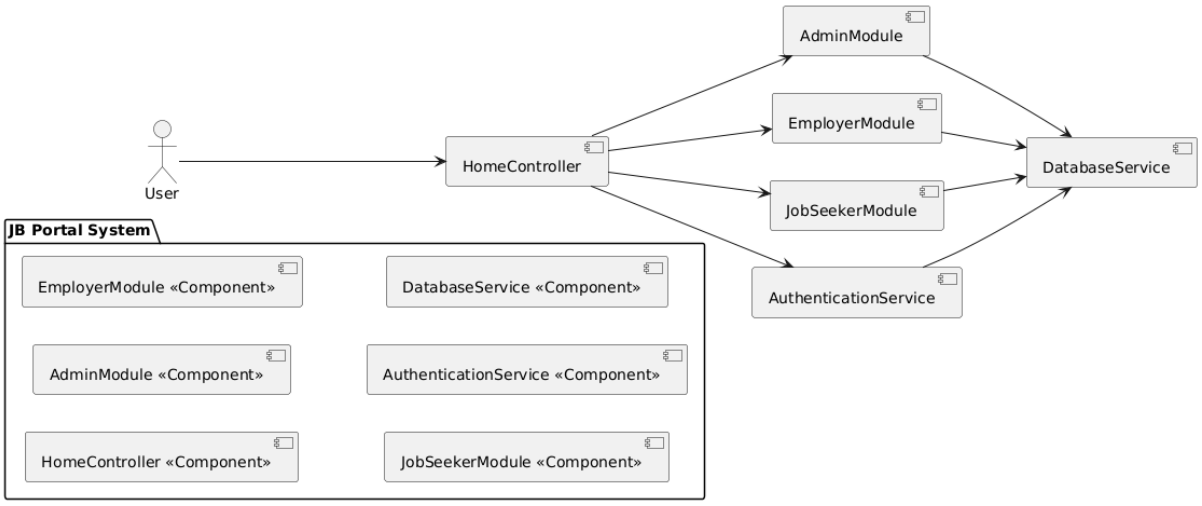
The Controller processes input from the user, updates the Model, and refreshes the View.

14 Design (all sprint 3 items

14.1 Package Diagram



14.2 Component Diagram



14.2.1 Description

A **component diagram** in UML represents the structure of a system by showing its components and their relationships.

Components

Components represent modular parts of the system, such as:

- Frontend
- Backend
- Database

They are depicted as rectangles.

Interfaces

Interfaces define the services provided or required by components:

- Provided interfaces are shown as circles.
- Required interfaces are shown as half-circles.

Dependencies

Dependencies indicate how components rely on each other, depicted using dashed arrows.

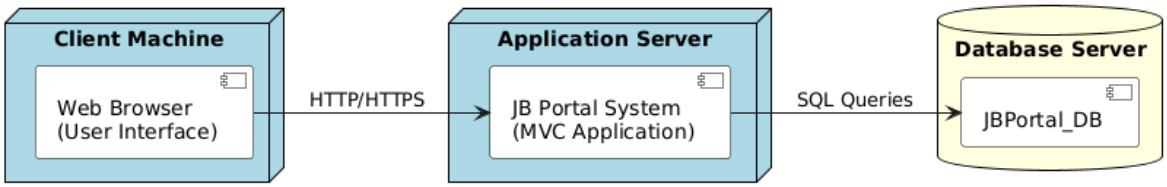
Ports

Ports are interaction points on a component for communication and are represented as small squares.

Connections

Connections are solid lines between components, showing communication paths.

14.3 Deployment Diagram



14.3.1 Description

Client Machine

The **Client Machine** runs a web browser where the user interacts with the system.

Application Server

The **Application Server** runs the JB Portal System, which is developed using the **Model-View-Controller (MVC)** architecture.

Database Server

The **Database Server** is responsible for storing all the data, including:

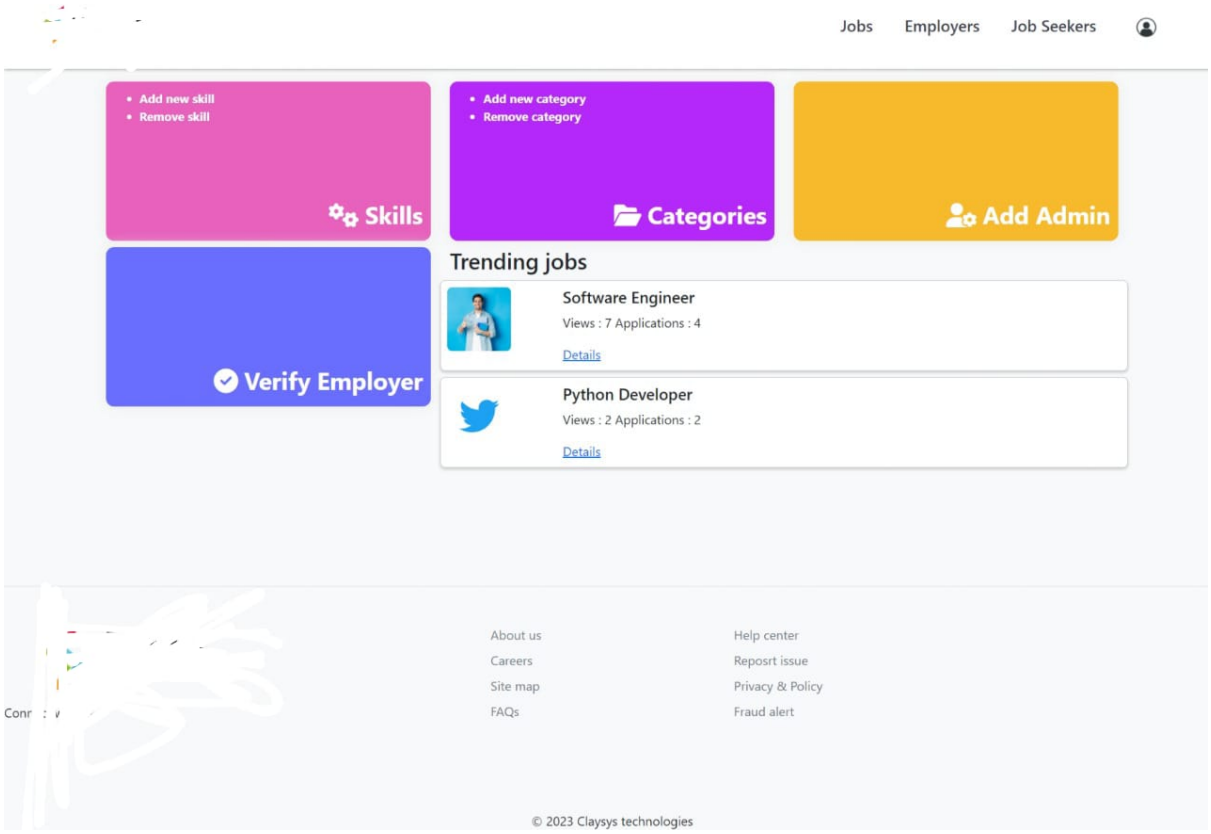
- User accounts
- Job listings
- Applications

Communication

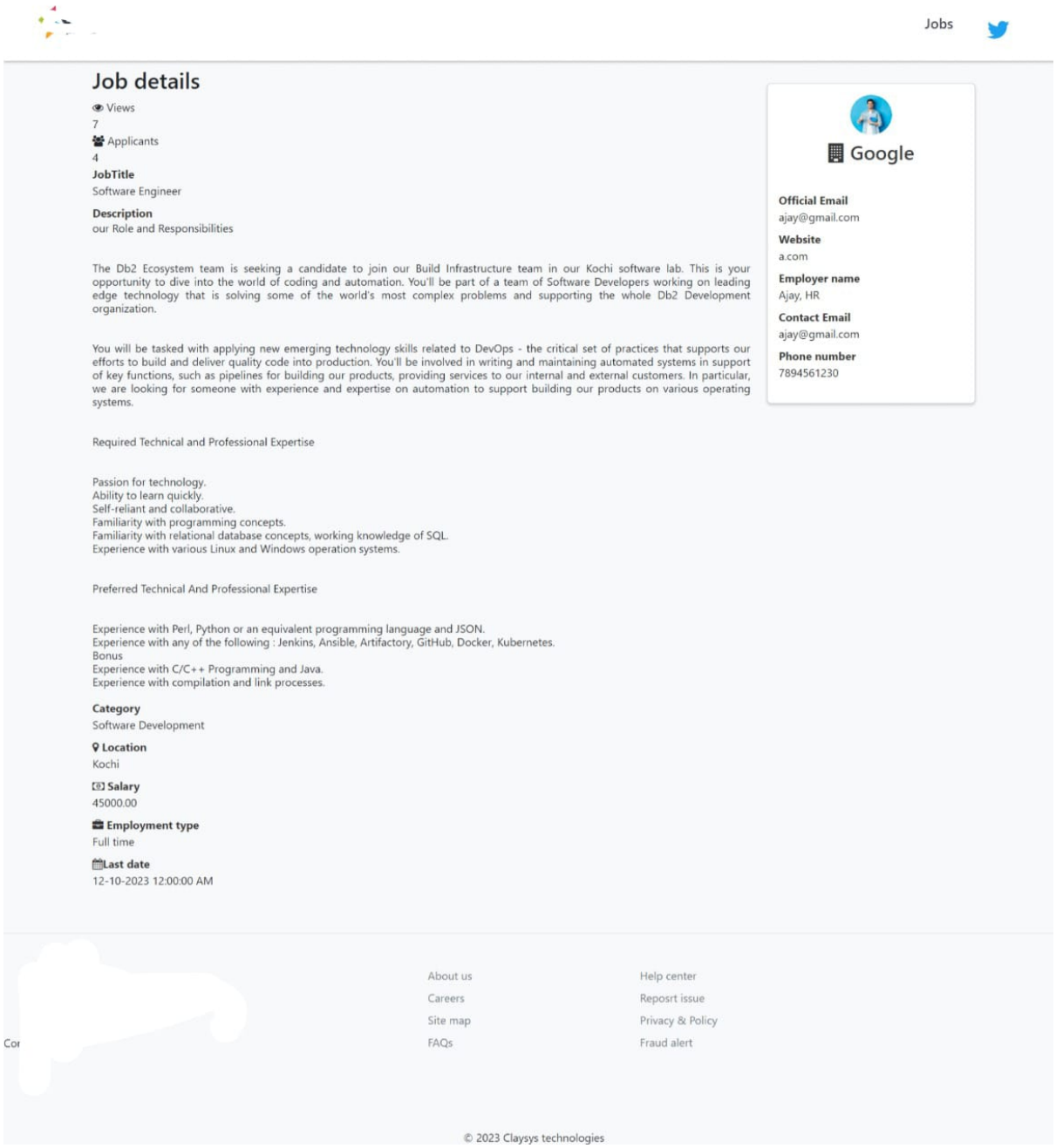
- The client communicates with the application server over **HTTP/HTTPS**.
- The application server interacts with the database server using **SQL queries**.

15 Actual Implementation screenshots

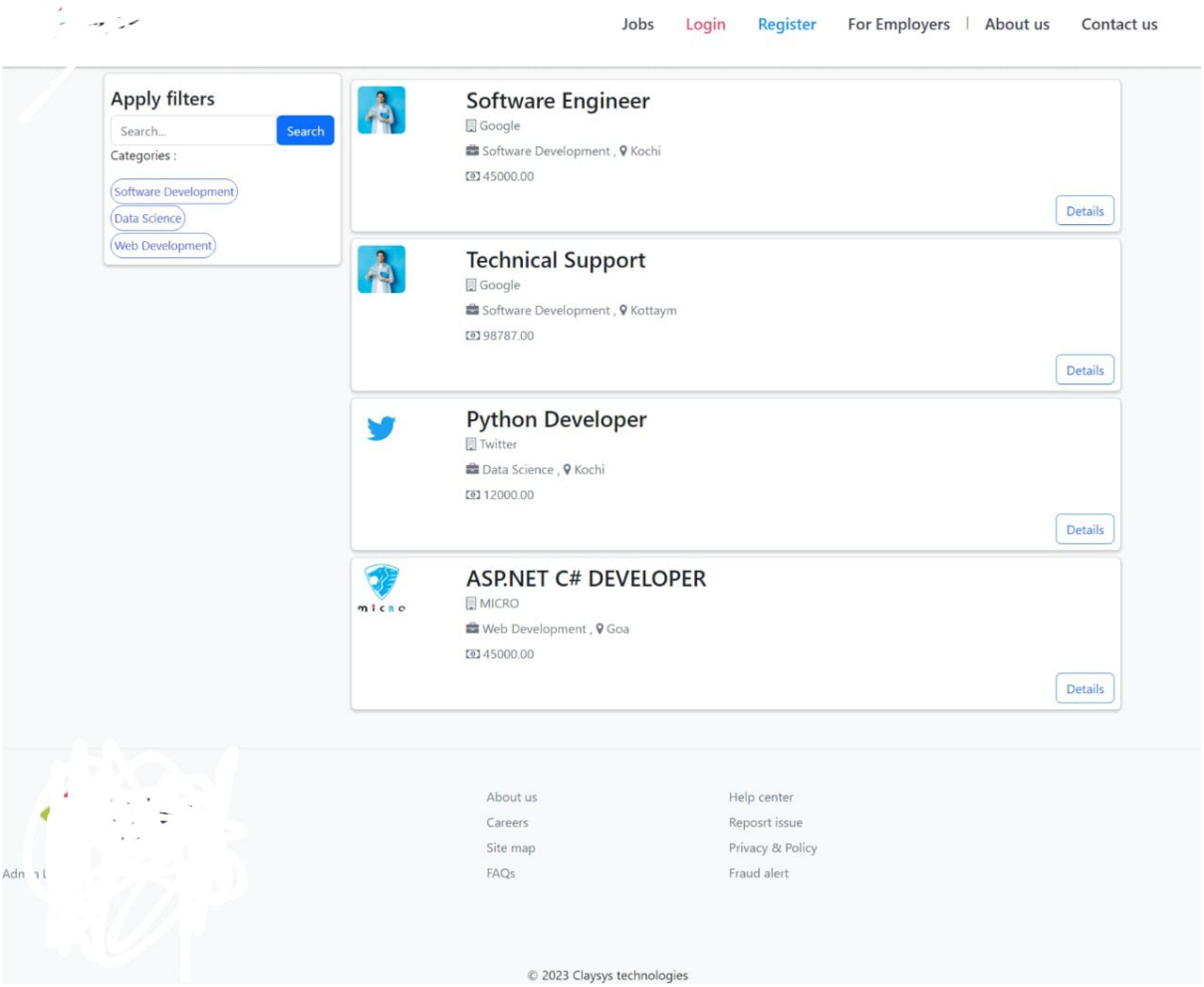
15.1 Admin



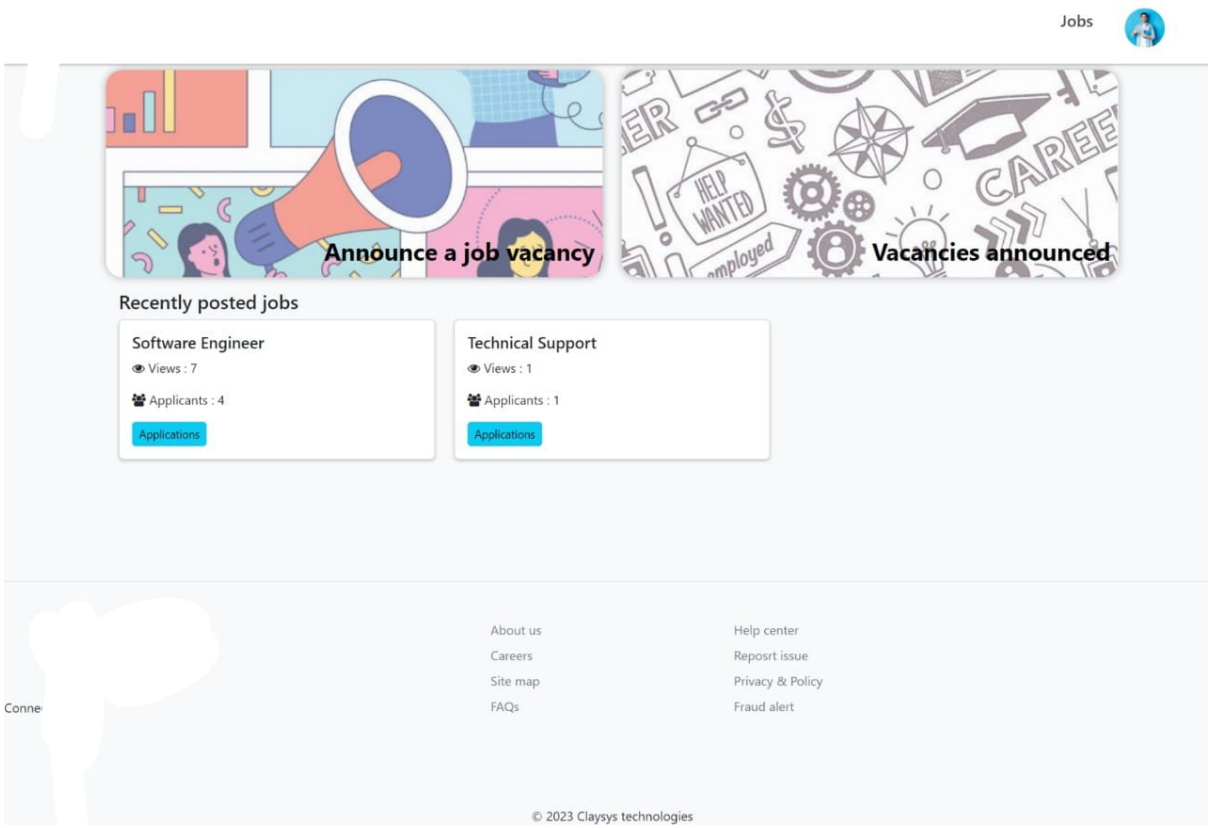
15.2 Job Details



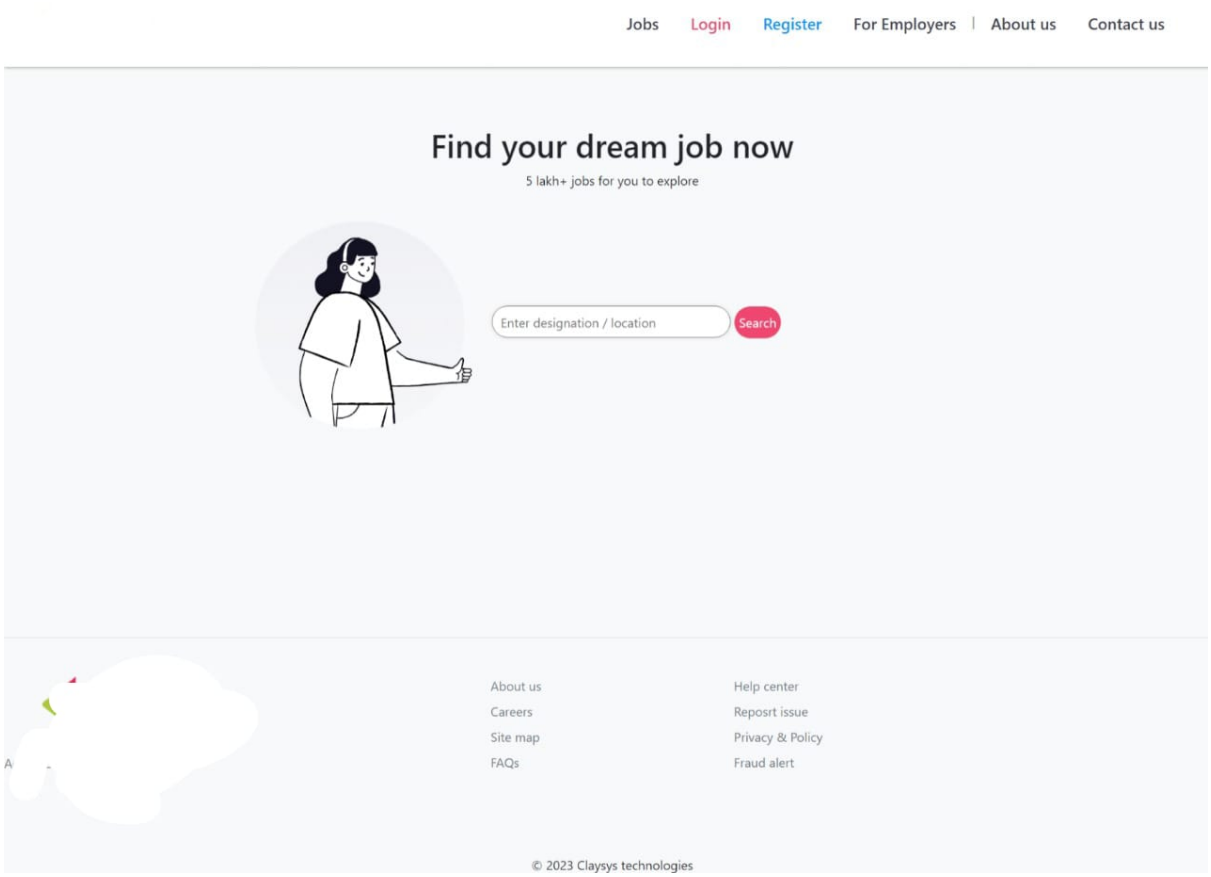
15.3 Jobs


Jobs
Login
Register
For Employers
About us
Contact us
Apply filters
Search...
Search
Categories :
Software Development
Data Science
Web Development
Software Engineer
Google
Software Development , Kochi
45000.00
Details
Technical Support
Google
Software Development , Kottaym
98787.00
Details
Python Developer
Twitter
Data Science , Kochi
12000.00
Details
ASP.NET C# DEVELOPER
MICRO
Web Development , Goa
45000.00
Details
About us
Careers
Site map
FAQs
Help center
Reposrt issue
Privacy & Policy
Fraud alert
© 2023 Claysys technologies

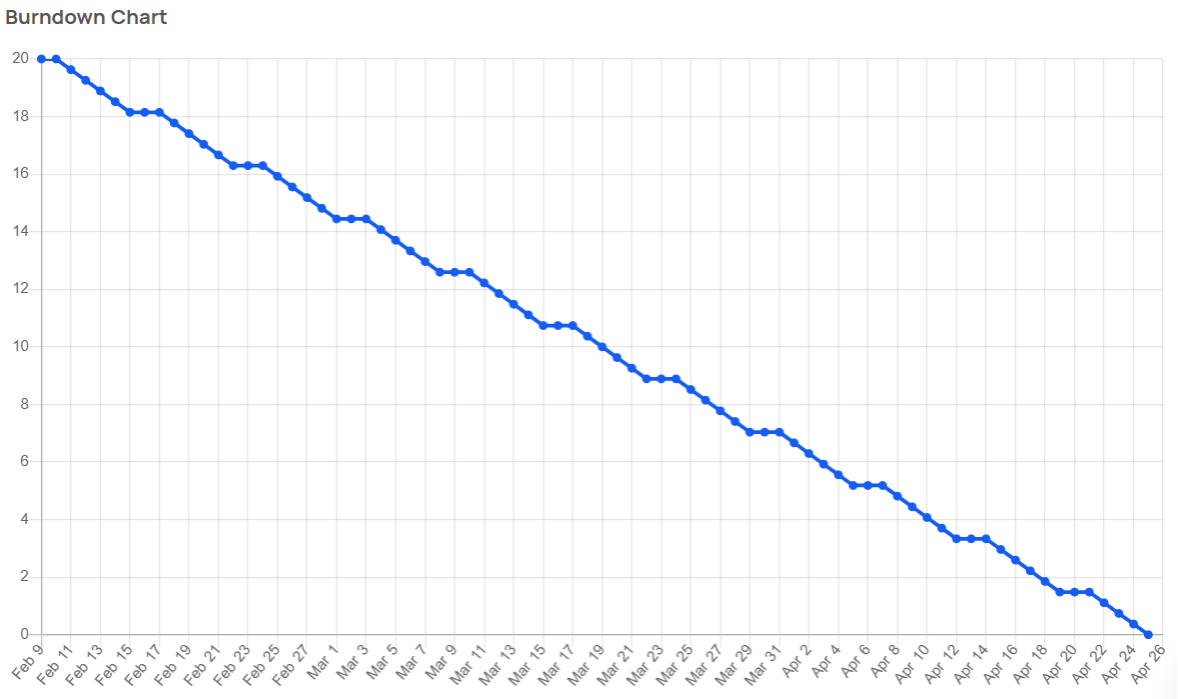
15.4 Employer



15.5 Home

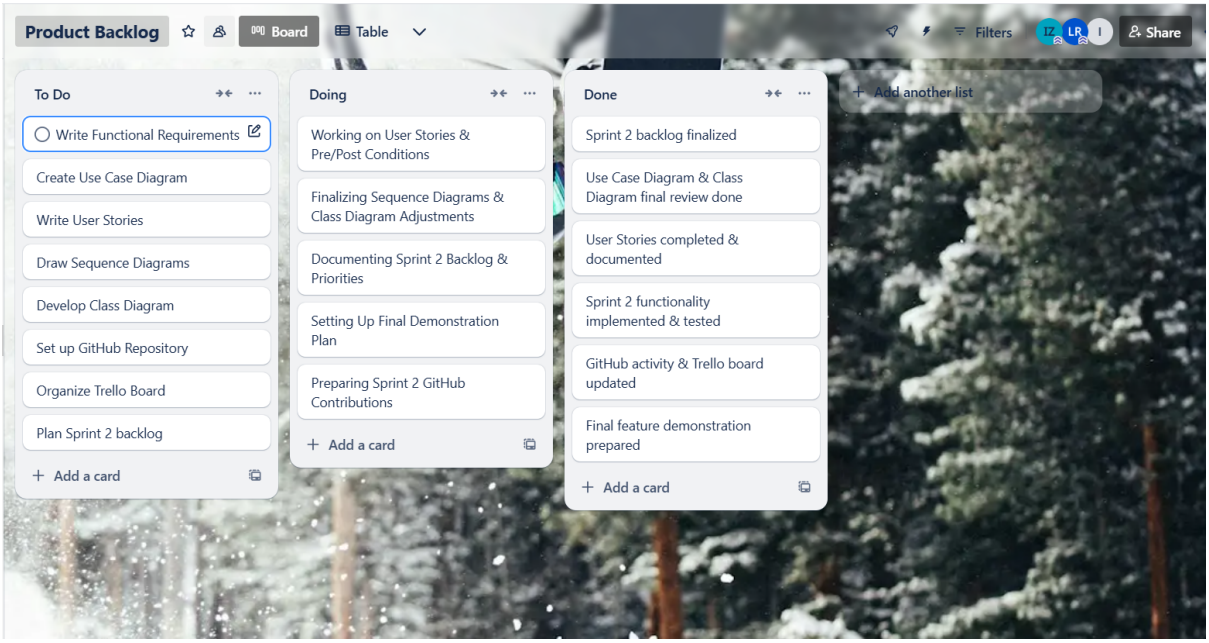

Jobs
Login
Register
For Employers
About us
Contact us
Find your dream job now
5 lakh+ jobs for you to explore
Enter designation / location
Search
About us
Careers
Site map
FAQs
Help center
Reposrt issue
Privacy & Policy
Fraud alert
© 2023 Claysys technologies

16 Product Burn down chart for the project

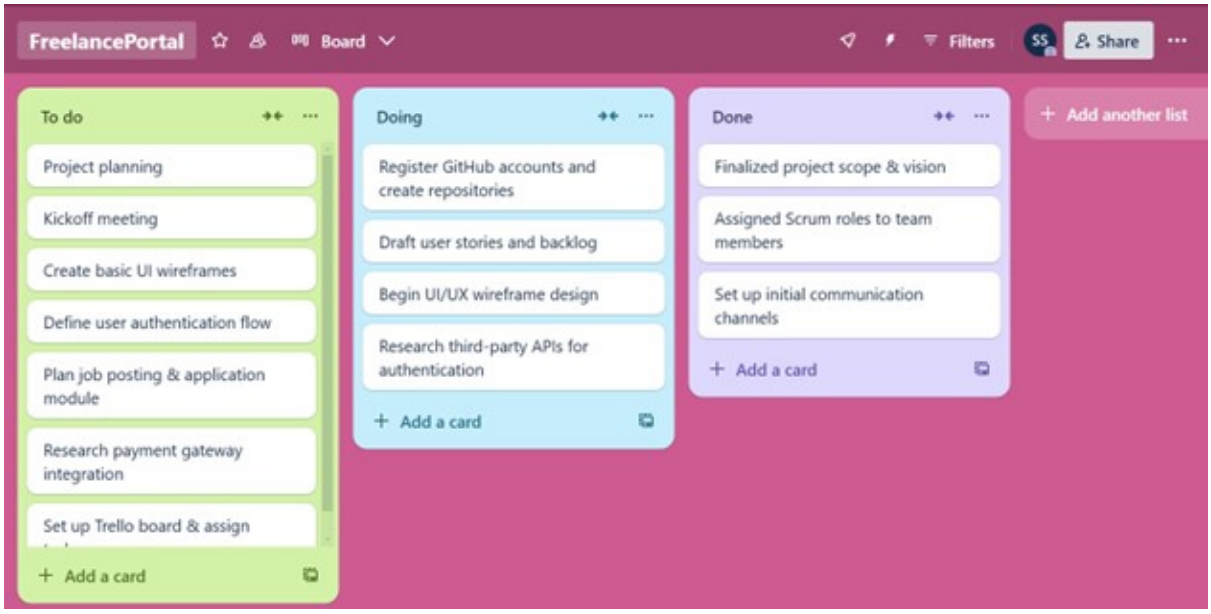


17 Trello board screen shots

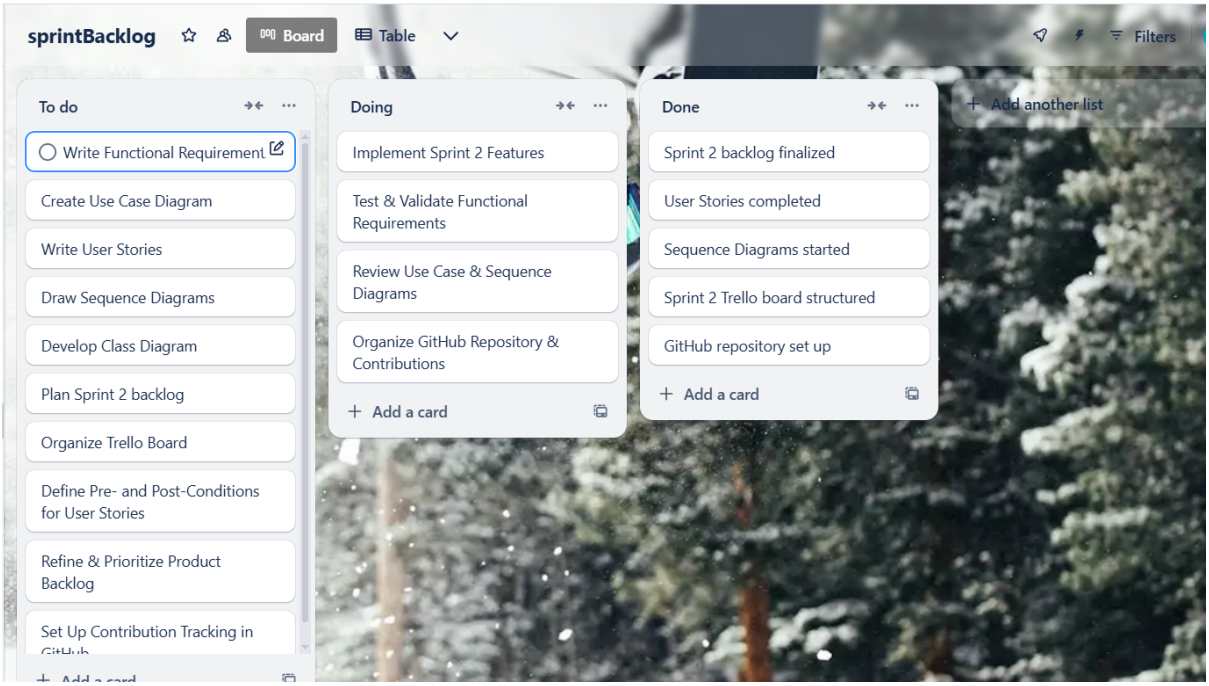
17.1 Product



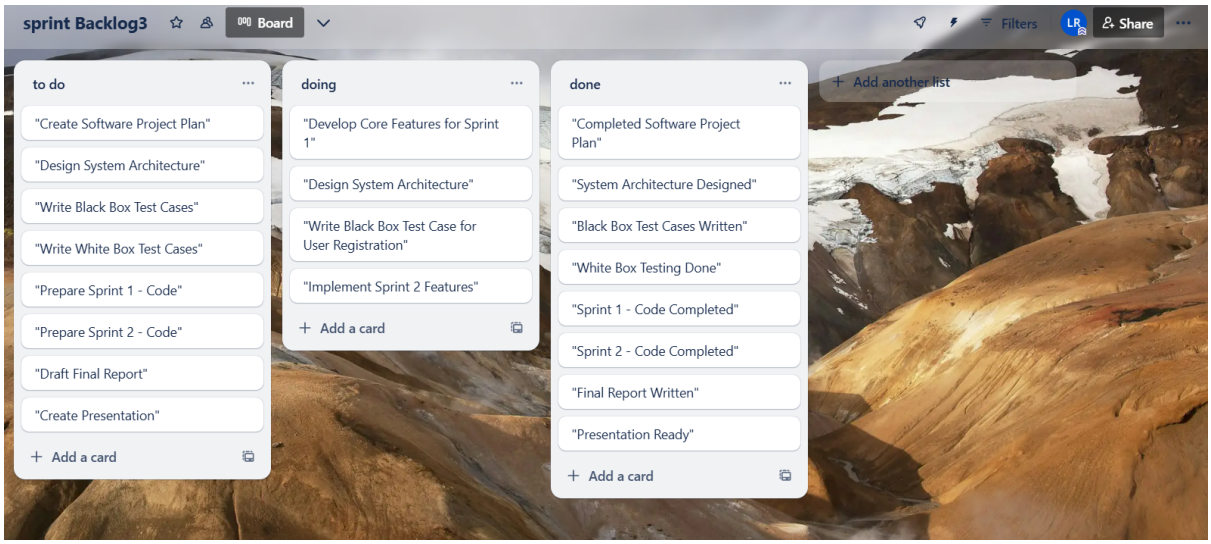
17.2 Sprint 01



17.3 Sprint 02



17.4 Sprint 03



18 Testcases -Black box

This chapter includes black box test cases for the Job Portal application. These test cases cover a range of functional scenarios focusing on input/output behavior without internal code knowledge.

Test Case ID	Test Scenario	Input	Expected Output	Status	Type of Test
B1	Job seeker register with invalid email	Email: "invalid-email", Username: "newuser", Password: "pass123"	Error message: 'Please enter a valid email address.'	Pass	ECP
B2	Job seeker register with valid email	Email: "new@email.com", Username: "newuser", Password: "pass123"	Success message: 'Registration successful.' Redirected to login page.	Pass	ECP
B3	Job seeker register with short password	Email: "new2@email.com", Username: "newuser2", Password: "pass"	Error message: 'Password must be at least 6 characters long.'	Pass	BVA
B4	Job seeker register with existing email	Email: "existing@email.com", Username: "newuser3", Password: "pass123"	Error message: 'Email already registered.'	Pass	ECP
B5	Employer post job with past deadline	Job Title: "Developer", Application Deadline: "2024-04-27" (past date)	Error message: 'Application deadline must be a future date.'	Pass	BVA

Test Case ID	Test Scenario	Input	Expected Output	Status	Type of Test
B6	Employer post job with future deadline	Job Title: "Developer", Application Deadline: "2025-05-01"	Success message: 'Job vacancy published.' Job appears in the job listings.	Pass	ECP
B7	Employer post job with empty title	Job Title: "", Application Deadline: "2025-05-01"	Error message: 'Job title is required.'	Pass	ECP
B8	Job seeker apply for job (not logged in)	Job ID: 1 (exists), User: Not logged in	Redirected to login page.	Pass	ECP
B9	Job seeker apply for job (logged in)	Job ID: 1 (exists), User: Job Seeker (logged in)	Success message: 'Applied Successfully.' Application submitted.	Pass	ECP
B10	Admin add skill with empty name	Skill Name: ""	Error message: 'Skill name is required.'	Pass	ECP
B11	Admin add skill with valid name	Skill Name: "Java"	Success message: 'Skill added.' Skill appears in the skills list.	Pass	ECP
B12	Admin delete skill	Skill ID: 1 (exists)	Success message: 'Skill deleted.' Skill removed from the list.	Pass	ECP
B13	Guest user submit contact form with long message	Name: "John", Email: "john@email.com", Message: (201 characters)	Error message: 'Message must be 200 characters or less.'	Pass	BVA
B14	Guest user submit contact form with valid message	Name: "John", Email: "john@email.com", Message: "Help needed"	Success message: 'Message sent.' Redirected to home page.	Pass	ECP

Test Case ID	Test Scenario	Input	Expected Output	Status	Type of Test
B15	Job seeker search with empty search term	Search Term: ""	All available jobs displayed.	Pass	ECP
B16	Job seeker search with specific term	Search Term: "Developer"	List of jobs with 'Developer' in the title displayed.	Pass	ECP
B17	Admin login with invalid credentials	Username: "admin", Password: "wrongpass"	Error message: 'Invalid username or password.'	Pass	ECP
B18	Admin login with valid credentials	Username: "admin", Password: "admin123"	Admin dashboard page displayed.	Pass	ECP
B19	Job seeker login with short username	Username: "ab", Password: "pass123"	Error message: 'Username must be at least 3 characters long.'	Pass	BVA
B20	Employer approve application	Application ID: 1 (exists), Job ID: 10 (exists)	Success message: 'Application Approved.' Application status updated to 'Approved.'	Pass	ECP

19 Testcases - White box

This document contains white box test cases for the Job Portal application, covering the **AdminController**, **EmployerController**, **HomeController**, and **JobSeekerController** classes. Each test case targets specific code paths, conditions, and branches to ensure comprehensive testing.

TC #	Test Scenario	Input	Expected Output	Status	Type of TC
T1	Admin Home	No input (GET request to /Admin/Index)	Code path for successful job retrieval executed. View with job details rendered.	Pass	ECP
T2	Admin Job Listing Failure	Simulate exception in AdminJobRepository.GetJobListings()	Code path for exception handling executed with error message returned.	Pass	ECP
T3	Add Skills with invalid model	Submit only ID with invalid data (null, SkillName)	Code path for `ModelState.IsValid` failure. View returned (AddSkill).	Pass	ECP
T4	Add Skills success path	Valid Skills info (e.g., SkillName="Java")	Code path for successful skill addition executed. Redirect to AddSkill view.	Pass	ECP
T5	Edit Skill Failure	Simulate exception in AdminRepository.EditSkill()	Code path for exception handling executed. View (Error) returned.	Pass	ECP
T6	Edit Skill Success	Skill ID: 1 (exists)	Code path for editing skill details executed. Redirect to Skills, TempData["Message"].	Pass	ECP
T7	View Employer	No input (GET request to /ViewEmployer)	Code path for retrieving employer listing executed. View rendered.	Pass	ECP
T8	Employer Approve	Employer ID: 1 (exists, pending)	Code path for successful approval executed. Redirect to previous page. TempData["Message"]: "Employer approved."	Pass	ECP
T9	Admin Change Password (wrong password)	Old Password: "wrong", New Password: "NewPassword", Session["AdminId"]="admin"	Code path for incorrect password executed. View returned, TempData["Message"]: "Wrong password."	Pass	ECP
T10	Admin Check Username (invalid)	Username: "nonuser" (does not exist)	Code path for `JobSeekerRepository.CheckUserName()` executed. HTTP 200 returned.	Pass	ECP

Notes

- Status: Marked as “Pass” assuming the code behaves as expected for these inputs.
- Type of Test: Primarily used ECP for test valid/invalid partitions (e.g., model state, session values) and some BVA for boundary conditions (e.g., dates, lists).
- Assumptions: Assumed repository methods return ’true’ for successful operations and throw exceptions for failures, as per the controller logic.

20 Work Division between group members

- **Laiba**
  - Gather and document project requirements.
  - Design UI/UX for the portal (wireframes, layouts).
  - Develop frontend components (HTML, CSS, Bootstrap, JavaScript).
- **Waseem Zahid**
  - Act as team lead and Scrum Master (manage tasks, sprint planning).
  - Develop backend APIs (C#, Node.js) and integrate frontend-backend.
  - Perform software testing and ensure quality assurance.
- **Shayan Salam**
  - Manage database design and implementation (SQL Server).
  - Develop backend functionalities alongside Waseem.
  - Handle documentation and testing of system components.

21 Lesson Learnt by Group

- Importance of clear communication and collaboration among team members.
- Managing time effectively is critical for meeting project deadlines.
- Proper planning and task division help in reducing workload and confusion.
- Hands-on practice with tools like Trello and GitHub improves project organization.
- Understanding both frontend and backend development deepens technical skills.
- Regular sprint meetings help in tracking progress and resolving issues early.
- Preparing complete documentation is important for clarity and future reference.