

# “CAREER QUEST”

*A Job Board Platform*

*Done By Team – 06,*

Pranav Reddy Gudipati, 1002162501

Pakshal Mahavir Ranawat, 1002152810

Rohan Keshav Harish, 1002170085

Harshavardhan Narayanaswamy, 1002206475

## **TABLE OF CONTENTS**

<b><u>Index</u></b>	<b><u>Description</u></b>	<b><u>Page No.</u></b>
1.	Project Initiation	1
	1.1 Abstract	
	1.2 Project Description	
	1.3 Project Scope	
	1.4 Project Resources	
	1.5 Software Development Lifecycle	
	1.6 Cost and Effort Estimation	
2.	Project Planning	5
	2.1 Work Breakdown Structure	
	2.2 Project Timeline	
	2.3 Product Backlog	
	2.4 Comparison of Ideal and Actual Progress	
3.	Risk Management	12
	3.1 Risk Identification	
	3.2 Risk Analysis	
	3.3 Risk Mitigation Plan	
4.	Project Quality Assurance	15
	4.1 Quality Factors	
	4.2 Quality Planning	
	4.3 Quality Control	
5.	Design	17
	5.1 E.R Diagram	
	5.2 U.M.L Diagram	
6.	Implementation	19
	6.1 Tools, Frameworks & Libraries	
	6.2 Supporting Tools	
	6.3 Application Implementation	
7.	Testing	23
	7.1 Manual Testing	
	7.2 Automated Testing	
	7.3 Conducting Different Types of Testing	
	7.4 Test Cases Screenshots	
8.	Maintenance	30
	8.1 Corrective Maintenance	
	8.2 Adaptive Maintenance	
	8.3 Perfective Maintenance	
	8.4 Preventative Maintenance	
9.	Roles and Responsibilities	31
10.	Conclusion	33
11.	Future Scope	34
12.	References	35

# **1. PROJECT INITIATION**

## **1.1 Abstract**

Career Quest is a job board platform developed to simplify the employment process for both job seekers and employers by providing a seamless, interactive environment. The primary objectives are to allow job seekers to discover and apply for relevant job opportunities and enable employers to attract, manage, and evaluate candidates efficiently. The platform supports end-to-end recruitment by allowing job seekers to create detailed profiles, submit applications, and receive real-time notifications on application progress. Employers can post job openings, review applicants, and manage hiring decisions, while admins monitor user activity to ensure a secure and reliable platform experience. With these features, Career Quest aims to enhance the job search and recruitment journey, providing a connected space for users on both sides of the hiring process.

## **1.2 Project Description**

1. Career Quest is a job board platform aimed at making the job search and hiring process easier for both job seekers and employers. It provides a streamlined experience, connecting job seekers with the right opportunities.
2. The platform comes with key functionalities like secure login, job search options with filters, and dedicated employer dashboards. Job seekers can apply for jobs, while employers can create and manage job postings.
3. Additional features include tracking job applications, sending real-time updates, and allowing users to create personalized profiles. This helps both the job seekers and employers manage the recruitment process more effectively.

## **1.3 Project Scope**

1. *User Authentication:* Implement login functionality for job seekers, employers, and admins.
2. *Job Search:* Enable job seekers to search for jobs using filters.
3. *Employer Dashboard:* Allow employers to create and manage job postings.
4. *Application Submission:* Allow job seekers to apply for jobs with resumes
5. *Job Posting Management:* Allow employers to update or delete job postings as needed.
6. *Admin Dashboard:* Provide admins with tools to manage users and monitor platform activity.
7. *Accept or Reject Applicants:* Employers can accept or reject job applications based on their suitability for the position.
8. *Notification System:* Notify users about application status, new job postings, and platform updates.
9. *Job Seeker's Profile:* Employees can create detailed profiles, including skills and education.
10. *Employer Reviews:* Job Seekers can leave reviews and ratings for employers.

## 1.4 Project Resources:

### 1.4.1 *Technological Requirements:*

- Frontend: HTML, CSS, Bootstrap, JavaScript.
- Backend: PHP, MariaDB.
- Version Control: Git, GitHub.
- Development Environment: Visual Studio Code, XAMPP (Apache, MariaDB, PHP).

### 1.4.2 *System Requirements:*

- Operating System: Cross-platform (Windows).
- RAM: Minimum 4GB (8GB recommended).
- Processor: Minimum 2 GHz dual-core processor.
- Storage: At least 10GB of available space for project files and databases.

### 1.4.3 *Software Requirements:*

- Web Server: Apache (included in XAMPP).
- Database: MariaDB.
- IDE: Visual Studio Code (or any preferred text editor).
- Version Control Software: Git.

### 1.4.4 *Libraries & Tools:*

- Bootstrap: For responsive design and pre-built UI components.
- phpMyAdmin: For database management and query execution.

## 1.5 Software Development Life Cycle

We will implement an Agile methodology using Scrum for the software development lifecycle of our project.

1. This method enables us to adapt quickly to changes as they arise.
2. Our team will work in short sprints, consistently delivering working parts of the system.
3. By recognizing risks early on, we can address them proactively.
4. Scrum facilitates strong collaboration among team members.
5. This approach will help us achieve a high-quality product in the end

## 1.6 Cost and Time Estimation

### COCOMO II - Constructive Cost Model

**Software Size**      Sizing Method

[SLOC](#)

	% Design Modified	% Code Modified	% Integration Required	Assessment and Assimilation (0% - 8%)	Software Understanding (0% - 50%)	Unfamiliarity (0-1)
New	<input type="text" value="2000"/>					
Reused	<input type="text" value="500"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value=""/>	<input type="text" value=""/>	
Modified	<input type="text" value="500"/>	<input type="text" value="10"/>	<input type="text" value="13"/>	<input type="text" value="15"/>	<input type="text" value="2"/>	<input type="text" value="20"/>

**Software Scale Drivers**

Precedentedness	<input type="text" value="Nominal"/>	Architecture / Risk Resolution	<input type="text" value="Nominal"/>	Process Maturity	<input type="text" value="Nominal"/>
Development Flexibility	<input type="text" value="Nominal"/>	Team Cohesion	<input type="text" value="Nominal"/>		

**Software Cost Drivers**

Product	Personnel	Platform			
Required Software Reliability	<input type="text" value="Nominal"/>	Analyst Capability	<input type="text" value="Nominal"/>	Time Constraint	<input type="text" value="Nominal"/>
Data Base Size	<input type="text" value="Nominal"/>	Programmer Capability	<input type="text" value="Nominal"/>	Storage Constraint	<input type="text" value="Nominal"/>
Product Complexity	<input type="text" value="Nominal"/>	Personnel Continuity	<input type="text" value="Nominal"/>	Platform Volatility	<input type="text" value="Nominal"/>
Developed for Reusability	<input type="text" value="Nominal"/>	Application Experience	<input type="text" value="Nominal"/>		
Documentation Match to Lifecycle Needs	<input type="text" value="Nominal"/>	Platform Experience	<input type="text" value="High"/>	<b>Project</b>	
		Language and Toolset Experience	<input type="text" value="Nominal"/>	Use of Software Tools	<input type="text" value="Nominal"/>
				Multisite Development	<input type="text" value="Nominal"/>
				Required Development Schedule	<input type="text" value="Nominal"/>

**Maintenance**

**Software Labor Rates**

Cost per Person-Month (Dollars)

### Results

#### Software Development (Elaboration and Construction)

Effort = 6.0 Person-months  
Schedule = 6.5 Months  
Cost = \$17883

Total Equivalent Size = 2072 SLOC  
Effort Adjustment Factor (EAF) = 0.91

#### Acquisition Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.4	0.8	0.4	\$1073
Elaboration	1.4	2.4	0.6	\$4292
Construction	4.5	4.0	1.1	\$13591
Transition	0.7	0.8	0.9	\$2146

#### Software Effort Distribution for RUP/MBASE (Person-Months)

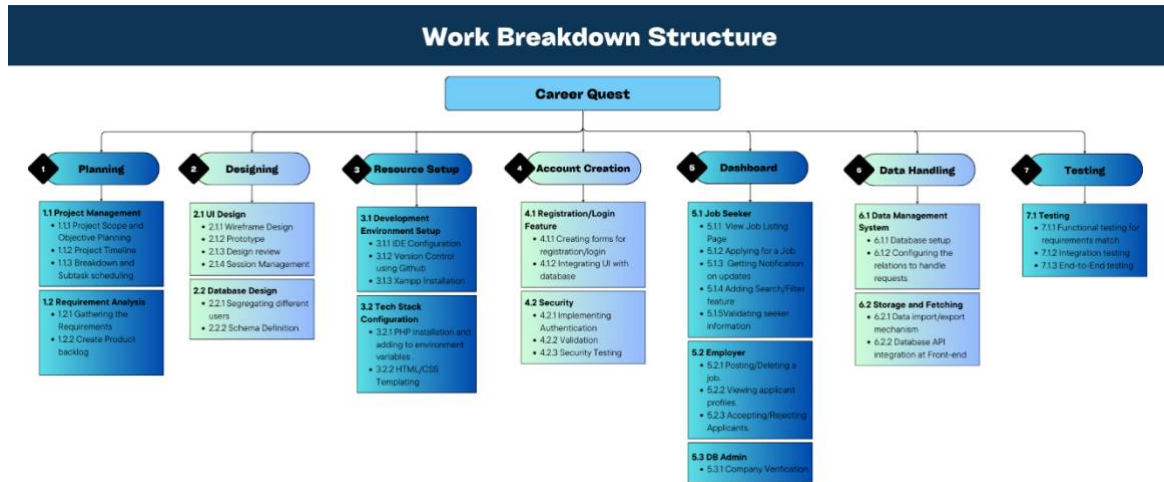
Phase/Activity	Inception	Elaboration	Construction	Transition
Management	0.1	0.2	0.5	0.1
Environment/CM	0.0	0.1	0.2	0.0
Requirements	0.1	0.3	0.4	0.0
Design	0.1	0.5	0.7	0.0
Implementation	0.0	0.2	1.5	0.1
Assessment	0.0	0.1	1.1	0.2
Deployment	0.0	0.0	0.1	0.2

The reference for this estimate comes from a similar model but with functionality to view new job postings.

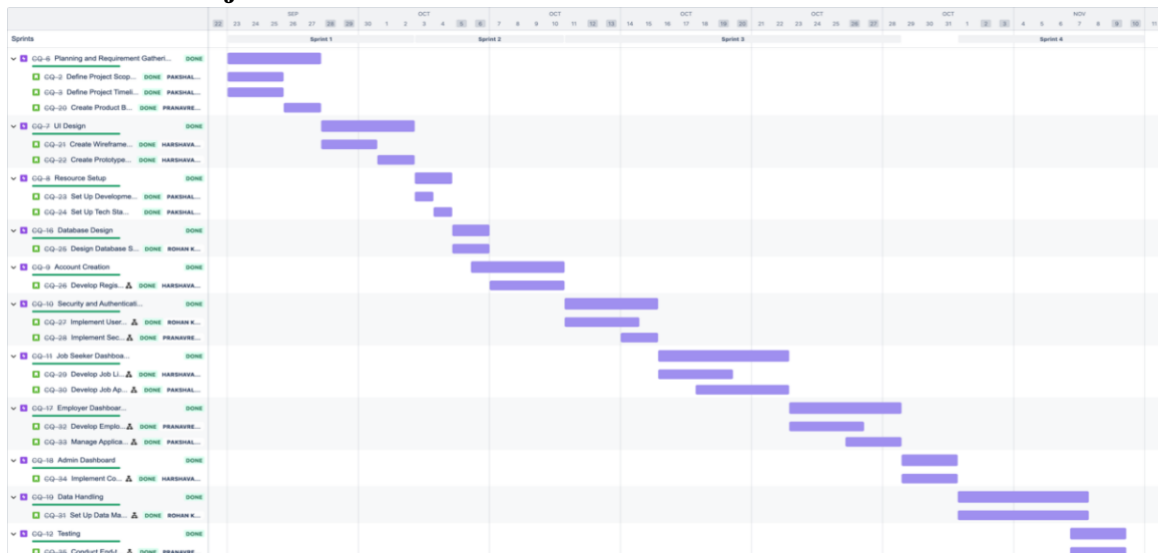
1. *New Lines of Code (LOC)*: Estimated at 2,000 new lines of code, this is based on the previous projects with certain similar functionality.
2. *Reused Lines of Code*: 500 lines of code will be reused from previous projects for components like the registration, login and other generic functionality.
3. *Modified Lines of Code*: 500 lines of code will require modification as these are the components from previous projects and will need to be tweaked to serve this project.
4. *Precedentedness*: Our team has a high familiarity with the technology stack, indicating a strong understanding of the tools, languages, and frameworks being used.
5. *Assessment and Assimilation Familiarity*: Our team has carefully assessed its familiarity with the project's software components, ensuring a smooth assimilation process throughout development.
6. *Estimated Total Project Cost*: The total estimated cost of the project is \$17,883. This estimate is based on a salary of \$3,000 per employee involved in the project.

## 2. PROJECT PLANNING

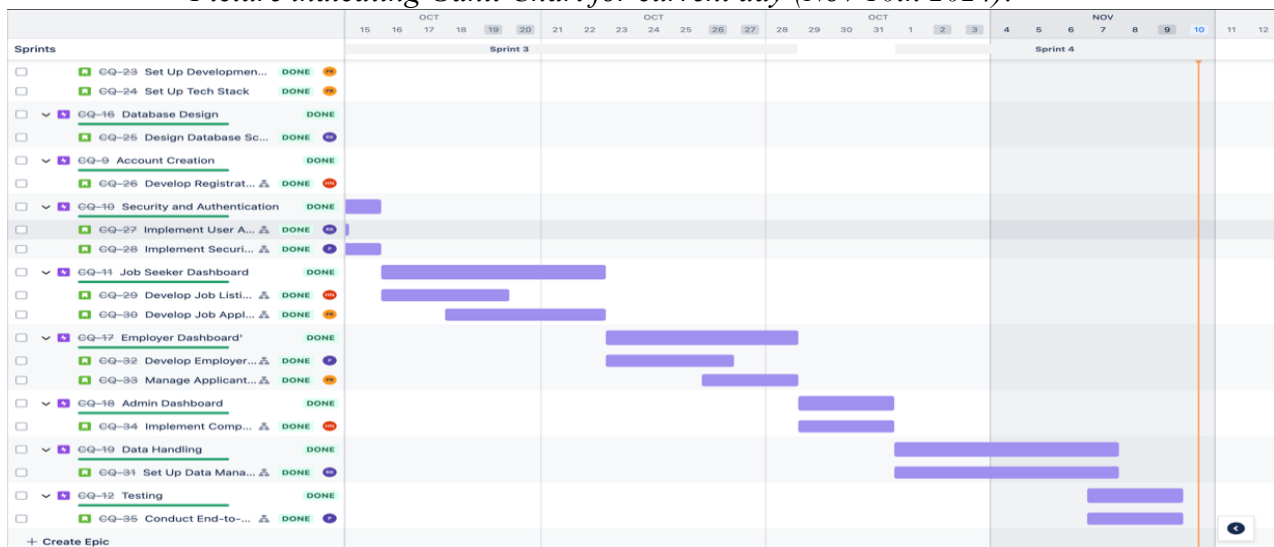
### 2.1 Work Breakdown Structure



### 2.2 Project Timeline



Picture indicating Gantt Chart for current day (Nov 10th 2024):



## 2.3 Product Backlog

The Planning/Scheduling tool used is Jira.

### ***Sprint 1: 23rd Sept – 2nd Oct***

1. Epic: Planning and Requirement Gathering (23rd Sept – 27th Sept)
  - Define Project Scope and Objectives: Identify and document the goals, objectives, and scope of the project.
  - Define Project Timeline: Establishing key milestones and delivery dates for the overall project.
  - Create Product Backlog: Organizing all the necessary features, tasks, and issues into a comprehensive backlog to be used for future sprints.
2. Epic: UI Design (28th Sept – 2nd Oct)
  - Create basic Design for Core Pages: Creating ER Diagram for knowing the basic design goals and what are the functionalities of the system and how each module is connected to each other.
  - Create Prototypes for Core Pages: Develop prototypes of the design

### ***Sprint 2: 3rd Oct – 10th Oct***

1. Epic: Resource Setup (3rd Oct – 4th Oct)
  - Set Up Development Environment: Install and configure tools like Xampp Server on the machine and login to phpMyAdmin.
  - Set Up Tech Stack: Ensure that php is installed on the machine and configure in the environment variables.
2. Epic: Database Design (5th Oct – 6th Oct)
  - Design Database Schema: Designing the MySQL schema.
3. Epic: Account Creation (5th Oct – 10th Oct)
  - Develop Registration/Login Feature: Implementing user authentication login and register feature.

### ***Sprint 3: 11th Oct – 28th Oct***

1. Epic : Security and Authentication (11th Oct – 15th Oct)
  - Implement User Authentication: Securing user login/registration function with hashing of the password and using local session storage to keep the user login even if he closes the tab.
  - Implement Security Testing: Conduct penetration testing and vulnerability checks on user login.
2. Epic: Job Seeker Dashboard (16th Oct – 22nd Oct)
  - Develop Job Listing Page: Creating the job listing page where users can view available events/jobs.
  - Develop Job Application Feature: Enable job seekers to apply for events/jobs and submit applications.



3. Epic: Employer Dashboard (23rd Oct – 28th Oct)
  - Develop Employer Job Posting: Allowing employers to post new job listings/events.
  - Manage Applicants for Job Posting: Provide functionality to manage applicants.

***Sprint 4: 29th Oct – 9th Nov***

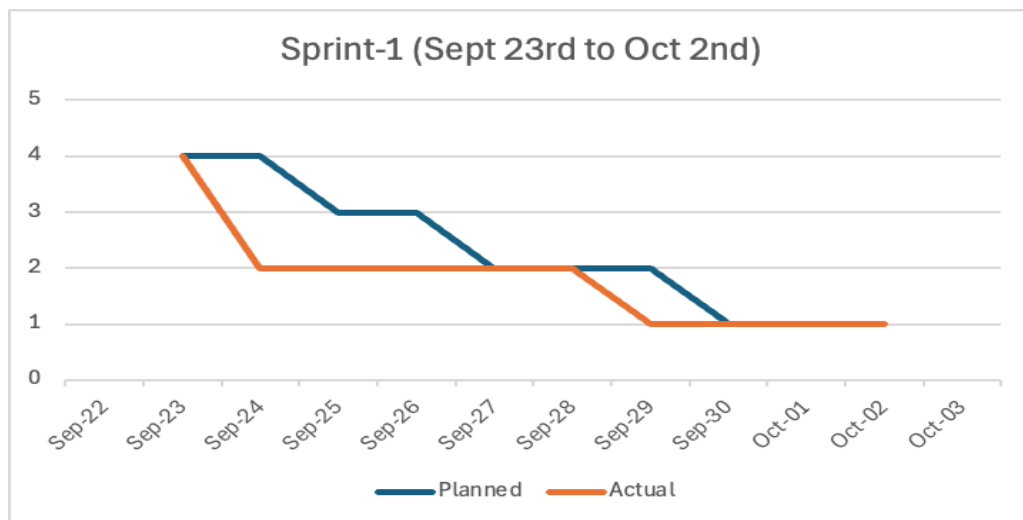
1. Epic: Admin Dashboard (29th Oct – 31st Oct)
  - Implement Company Verification Process: Creating a process where admins can verify the authenticity of companies posting jobs.
2. Epic: Data Handling (1st Nov – 8th Nov)
  - Set Up Data Management System: Ensuring a scalable and secure data storage with MariaDB for managing job data.
3. Epic: Testing (7th Nov – 9th Nov)
  - Conduct End-to-End Testing: Perform tests on the entire application, covering user registration, login, job posting, job applications, and admin functionalities.

## 2.4 Comparison of Ideal and Actual Progress

*Sprint 1:*

Epic	Ideal Timeline	Actual Timeline	On Track?	Analysis and Justification
Planning and Requirement Gathering	23rd Sept – 27th Sept	23rd Sept – 24th Sept	Yes	All tasks were completed within the planned timeline without any delays.
UI Design	28th Sept – 2nd Oct	24th Sept – 2nd Oct	Yes	The design and prototypes were created as per schedule.

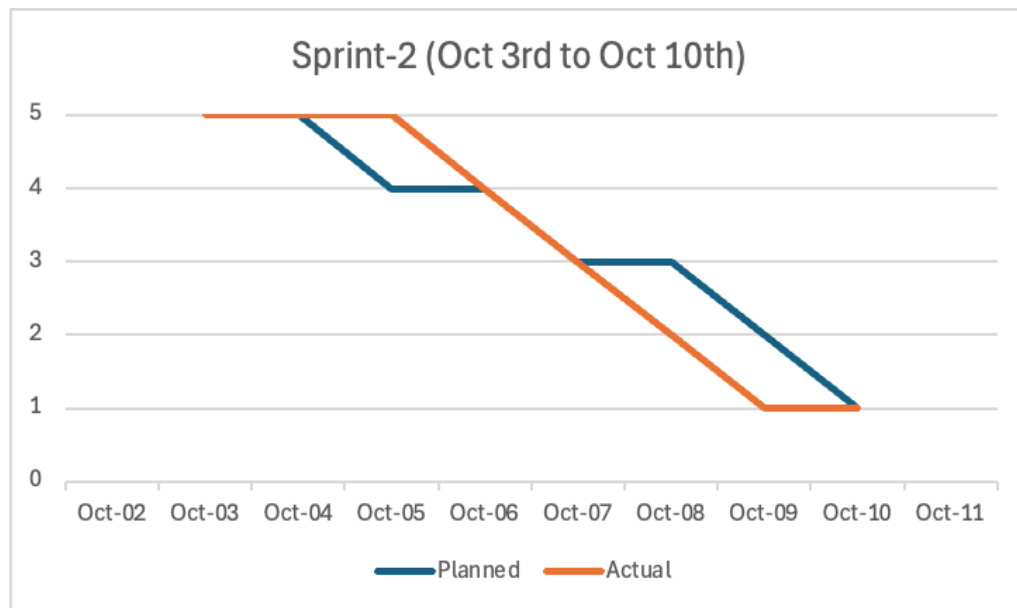
*Burndown chart of Sprint-1*



*Sprint 2:*

<b>Epic</b>	<b>Ideal Timeline</b>	<b>Actual Timeline</b>	<b>On Track?</b>	<b>Analysis and Justification</b>
Resource Setup	3rd Oct – 4th Oct	3rd Oct – 6th Oct	No	Environment and tech stack setup was delayed due to mid-term exams but caught up shortly after.
Database Design	5th Oct – 7th Oct	5th Oct – 7th Oct	Yes	Database schema design was completed as planned.
Account Creation	5th Oct – 10th Oct	5th Oct – 10th Oct	Yes	User login/registration was completed on time

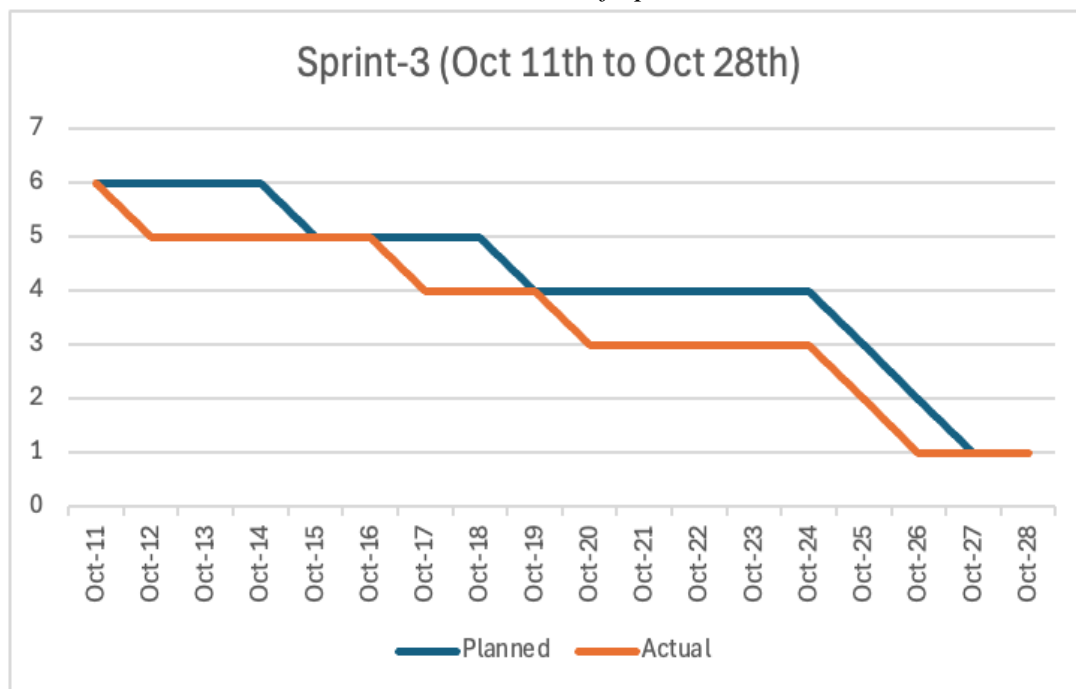
*Burndown chart of Sprint-2*



*Sprint 3:*

<b>Epic</b>	<b>Ideal Timeline</b>	<b>Actual Timeline</b>	<b>On Track?</b>	<b>Analysis and Justification</b>
Security and Authentication	11th Oct – 15th Oct	11th Oct – 12th Oct	Yes	Security and Authentication has been done within the time planned.
Job Seeker Dashboard	16th Oct – 22nd Oct	12th Oct - 17th Oct	Yes	Job Seeker .Dashboard was completed as planned, despite minor delays from previous task
Employer Dashboard	23rd Oct – 28th Oct	17th Oct - 28th Oct	Yes	The employer dashboard was implemented on schedule and the functionality met the requirements.

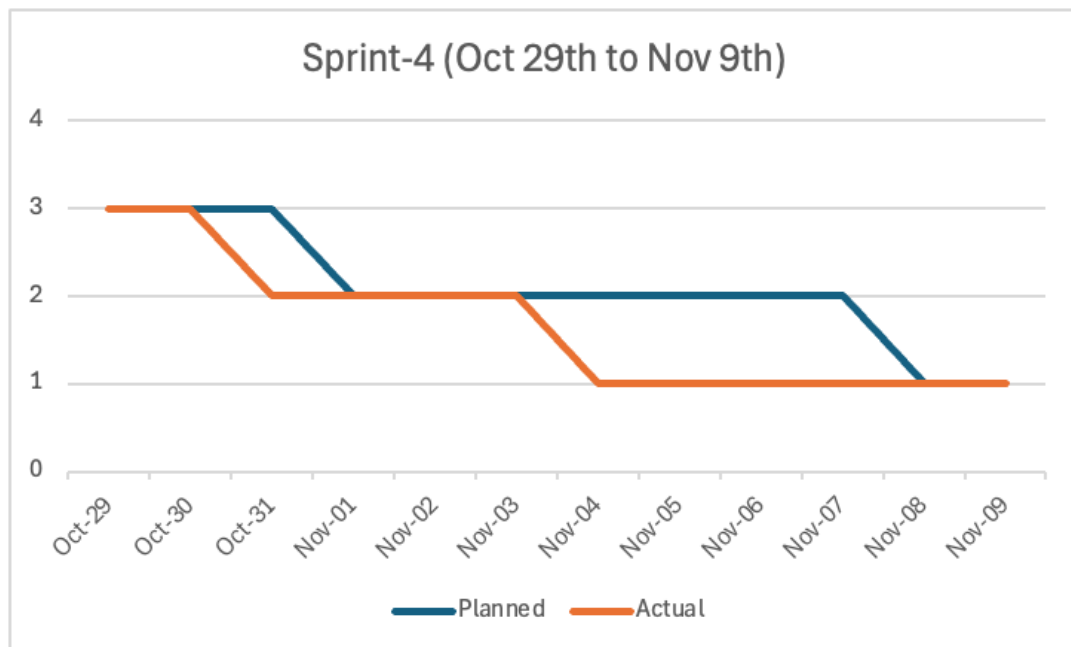
*Burndown chart of Sprint-3*



#### *Sprint 4:*

<b>Epic</b>	<b>Ideal Timeline</b>	<b>Actual Timeline</b>	<b>On Track?</b>	<b>Analysis and Justification</b>
Admin Dashboard	29th Oct – 31st Oct	29th Oct - 31st Oct	Yes	The Admin Dashboard was developed and tested within the scheduled time, meeting all requirements.
Data Handling	1st Nov - 7th Nov	31st Oct - 4th Nov	Yes	Data processing and handling features were implemented as planned, ensuring smooth data flow.
Testing	7th Nov - 9th Nov	4th Nov- 9th Nov	Yes	Comprehensive testing was conducted, and all major bugs were resolved on time, preparing the project for demo.

*Burndown chart of Sprint-4*



### **3. RISK MANAGEMENT**

#### **3.1 Risk Identification**

<b>No.</b>	<b>Risk Description</b>	<b>Category</b>
1.	Basic Styling/Design Issues : The website might have inconsistent or unappealing design elements, affecting the user experience.	Technical
2.	Lack of Data Validation : Users might be able to submit incorrect or incomplete information (e.g., missing resume uploads, empty job postings).	Technical
3.	Mismanagement of User Roles and Permissions : Users may gain unauthorized access to features meant for different roles, leading to misuse or data leaks.	Organizational
4.	Limited Error Handling : Users may encounter bugs or errors, such as submitting a form that doesn't go through, without any clear feedback or guidance on what went wrong.	Technical
5.	Data Breach or Security Vulnerabilities : User data, including personal details and resumes, could be exposed to unauthorized access.	Technical

### 3.2 Risk Analysis

No.	Risk	Likelihood (1-5)	Impact (1-5)	Risk Level (L) x (I)
1.	Basic Styling/Design Issues	2	2	Low (4)
2.	Lack of Data Validation	3	4	High (12)
3.	Mismanagement of User Roles and Permissions	2	5	Medium (10)
4.	Limited Error Handling	3	3	Medium (9)
5.	Data Breach or Security Vulnerabilities	1	5	Low (5)

### 3.3 Risk Mitigation Plan

No.	Risk	Priority	Mitigation
1.	Lack of Data Validation	High	Implement client-side and server-side validation to ensure all required fields are filled in and in the correct format.
2.	Mismanagement of User Roles and Permissions	Medium	Clearly define user roles and permissions within the system. Implement role-based access controls and regularly review access logs for discrepancies.
3.	Limited Error Handling	Medium	Implement basic error handling and feedback messages (e.g., "Form submission failed. Please try again.") for key user actions.
4.	Data Breach or Security Vulnerabilities	Low	Implement robust security measures such as hashing/encryption for sensitive data, SSL for secure communication, strong password policies, and regular security audits
5.	Basic Styling/Design Issues	Low	Use simple, responsive CSS frameworks (like Bootstrap) to ensure a consistent and user-friendly layout. Perform basic testing on different screen sizes.



## **4. PROJECT QUALITY ASSURANCE**

### **4.1 Quality Factors**

1. *Functionality*: We will ensure that our platform meets all user and system requirements, with a focus on delivering all promised features.
2. *Usability*: Providing a user-friendly, intuitive, and responsive design to make navigation easy and enhance user experience.
3. *Reliability*: Maintain high system uptime, especially during peak usage times, and minimize occurrences of crashes.
4. *Security*: Implementing secure login methods using hashing and conduct regular security audits to prevent data breaches.
5. *Maintainability*: We will use modular, well-structured code, making future updates and feature expansions easier to implement.
6. *Compatibility*: Ensuring cross-platform compatibility (including Windows) and consistent functionality across various devices for a seamless experience.

### **4.2 Quality Planning**

1. *Safety*: As the reviews posted about different companies might be both positive and negative it might be impactful on their role if they are a current employee. So, we have implemented anonymous reviews.
2. *Adaptability*: Users might not always be carrying laptops, so we have implemented a dynamic application which scales based on the screen they use to access the portal and still provide the same experience.
3. *Modularity*: The application follows a modular design making it easy to track the source for any bugs, maintenance and also further enhancements.
4. *Learnability*: Simple interactive and informative interface makes it a smooth experience for new users not bombarding them with too much information.

### **4.3 Quality Control**

1. *Testing*:
  - Conduct automation and functional testing to validate that each feature performs as expected.
  - User acceptance testing (UAT) will be carried out to verify that the platform is user-friendly and intuitive.
  - Security testing will be performed to identify and resolve potential vulnerabilities before release, protecting user data and system integrity.
2. *Code Reviews*:
  - We will have regular peer reviews to ensure code quality, consistency, and adherence to best practices.

- Code reviews will be scheduled frequently, allowing early identification and correction of potential issues.

3. *User Feedback:*

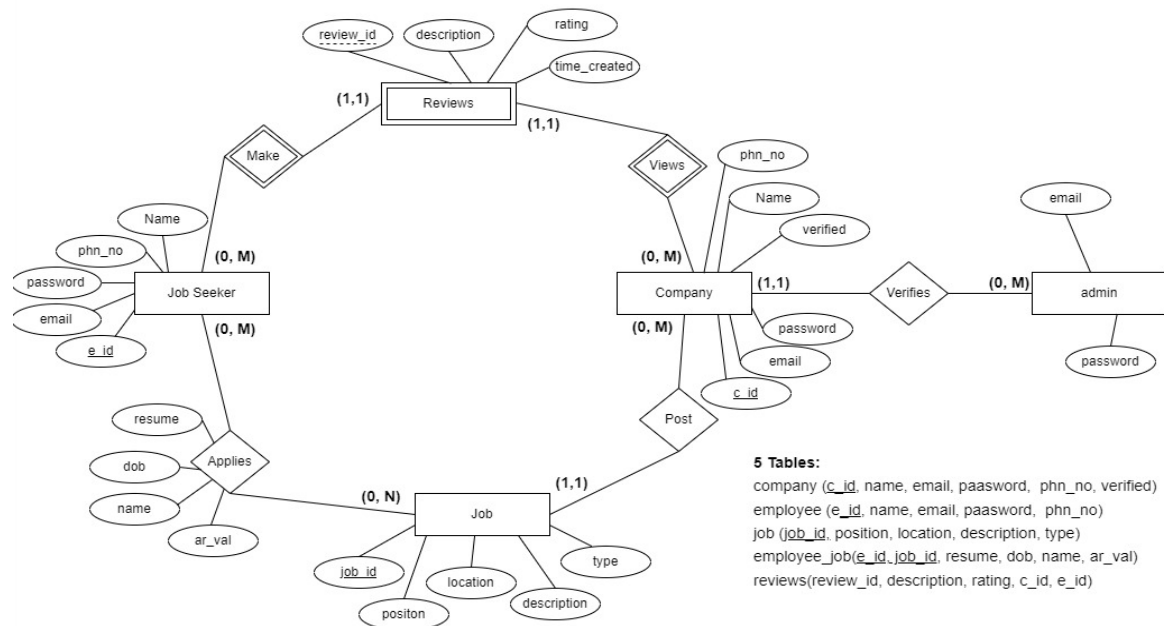
- Collecting user feedback after each sprint to ensure that the application is functioning as expected and that user needs are being met.
- This iterative feedback loop helps the team make adjustments promptly and improve the user experience over time.

4. *Version Control:*

- Use Git for version control to manage code changes efficiently and facilitate collaboration among team members.
- Adopting a structured branching strategy and pull request process ensures code quality and maintains a clear, organized history of project developments.

## 5. DESIGN

### 5.1 ER Diagram

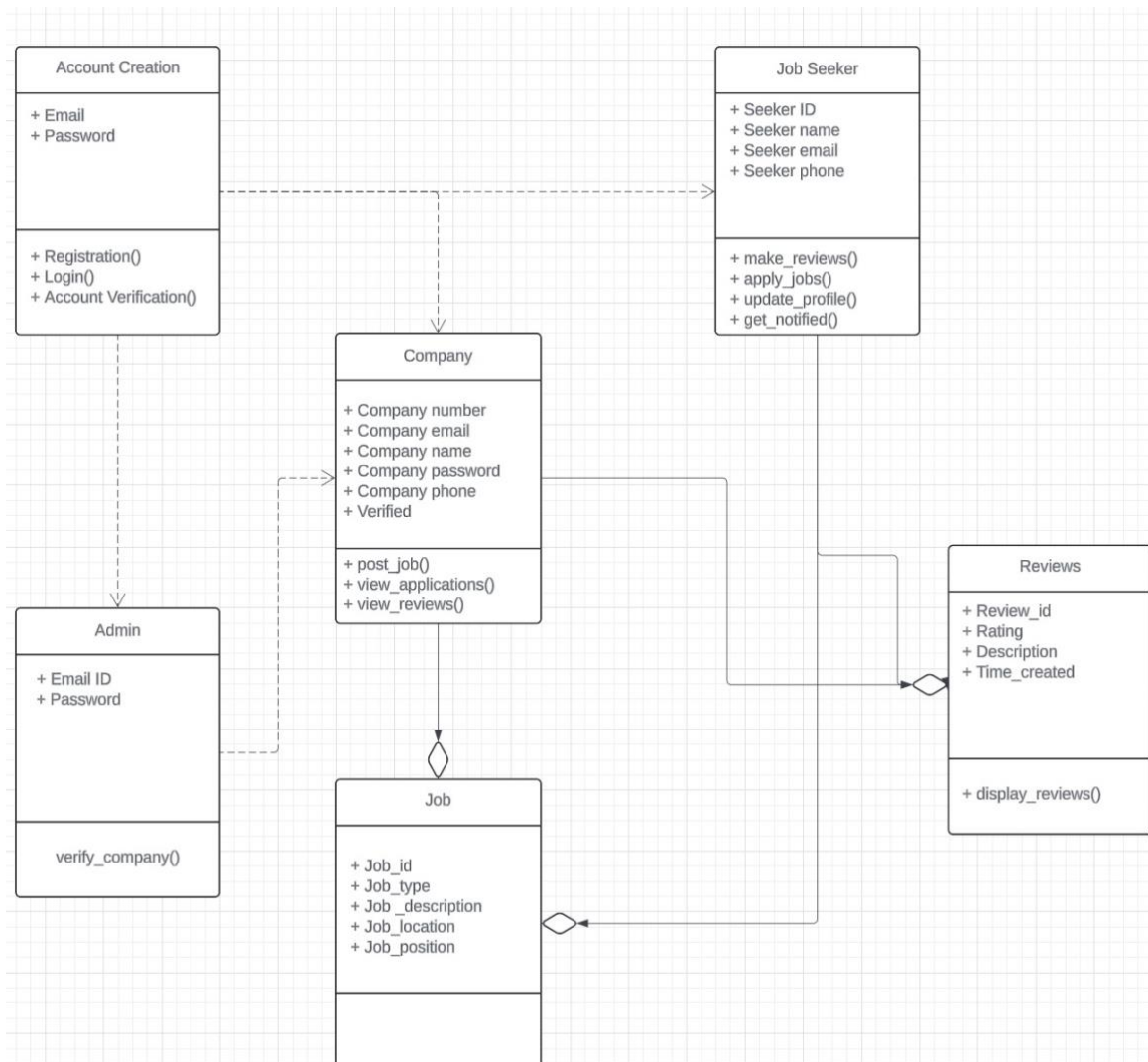


The Entity Relation Diagram represents the five entities of the application and their relationships.

- Admin: Admins acts like the Database Administrator and verifies the company profiles based on the background check as we do not want any sort of mis happenings with the candidates who apply for those companies.
- Company: This is the organization that posts the job on the portal based on their requirements. They can have multiple postings for different roles which can attract multiple applications.
- Job Seeker: These are individuals who are trying to acquire the opportunity posted by companies based on their qualification and interest. A single job seeker can apply to multiple job roles and companies at the same time.
- Job: This is the listing itself and gives insight on what the candidate applying for it can expect from that position and the skills they might need to get into that position with other relevant information like the application deadline, salary and more.
- Reviews: These are made by the current or past employees of a company, based on their experience in terms of work culture and balance.

## 5.2 UML Diagram

The UML Diagram represents the different attributes of the classes and the methods that they provide as the functionality to different users of the application. The different types of connection represent the dependency between the different classes.



## 6. IMPLEMENTATION

### 6.1 Tools, Frameworks, and Libraries

- *PHP*: Core back-end language for building server-side logic, handling form data, session management, and database interactions.
- *HTML*: Structures webpage content, including form fields, buttons, tables, and page elements.
- *CSS*: Styles and organizes page layout for consistent appearance across browsers and screen sizes.
- *Bootstrap*: CSS framework providing responsive, pre-styled components (e.g., navigation, forms, buttons) for quick, mobile-friendly designs.
- *JavaScript*: Enhances client-side interactivity with dynamic content, form validation, and interactive elements.
- *MariaDB*: Database management system to store essential project data such as user profiles, job postings, reviews, and applications.

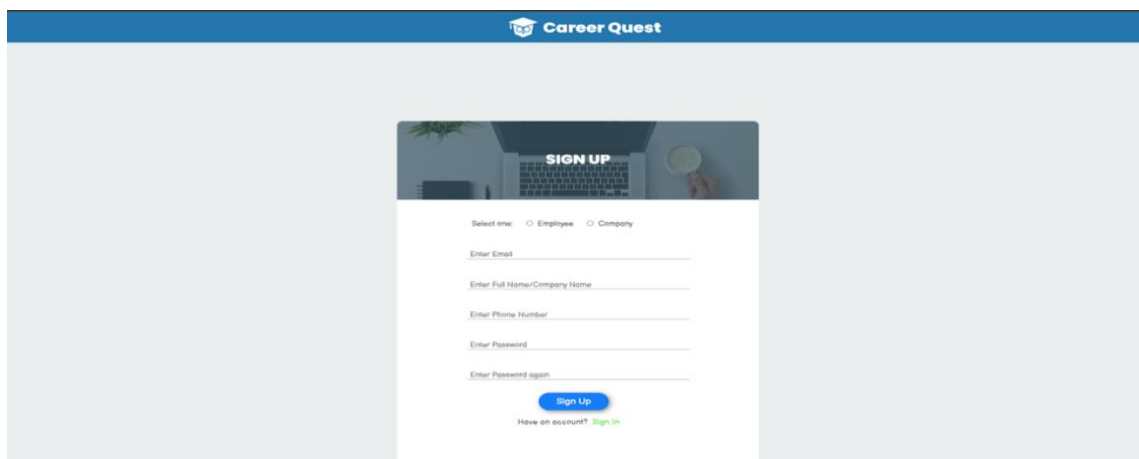
### 6.2 Supporting Tools

- *Visual Studio Code*: Main code editor for organizing and editing project files.
- *XAMPP*: Local server environment for testing PHP scripts and connecting with MariaDB.
- *phpMyAdmin*: Web interface to manage the MariaDB database, enabling easy setup, table management, and data inspection.

### 6.3 Application Implementation

#### 6.3.1 Register:

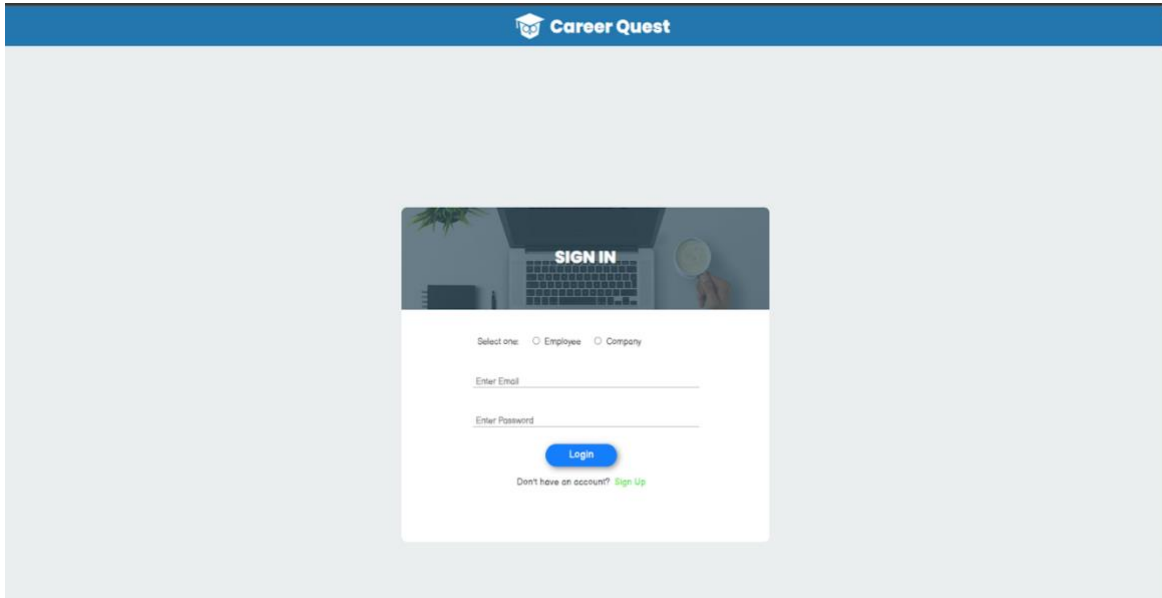
- We implemented validation for required fields like email, password, and user details during registration.
- Users are notified with messages like "Please fill all required fields" or "Email already exists" for guidance.
- This ensures accurate data entry and smooth account creation, enhancing security.



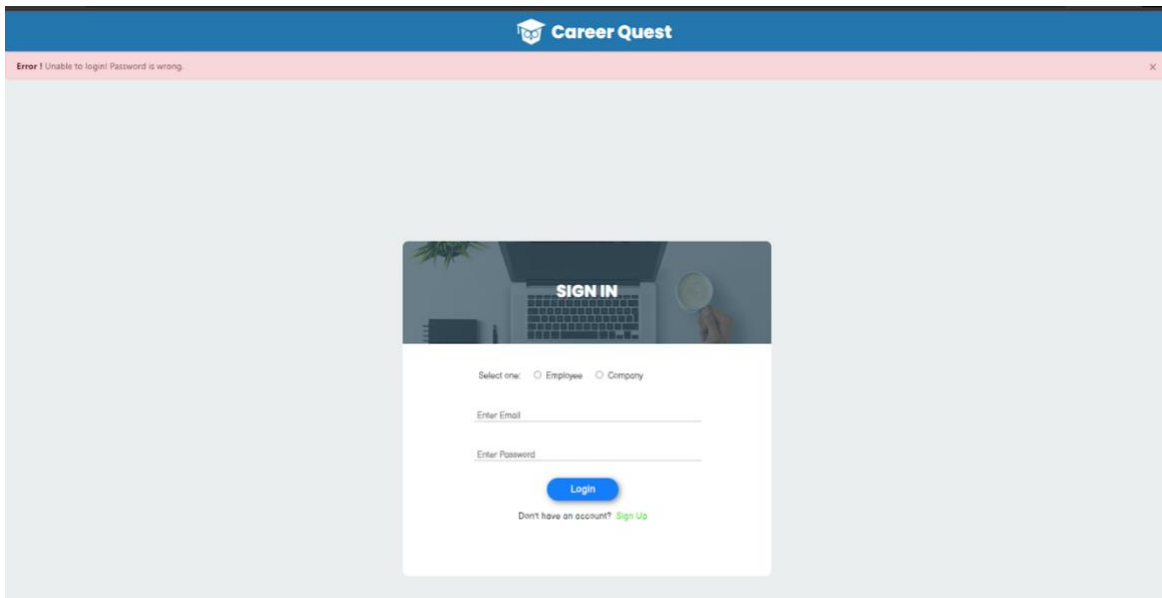
The screenshot displays the 'Sign Up' form for 'Career Quest'. The form is centered on a light gray background. At the top, there's a blue header with the 'Career Quest' logo. The form itself has a white background and a blue border. It starts with a 'SIGN UP' title in bold. Below it, there's a 'Select one:' label with two radio buttons: 'Employee' and 'Company'. The form then has five input fields: 'Enter Email', 'Enter Full Name/Company Name', 'Enter Phone Number', 'Enter Password', and 'Enter Password again'. Each field has a placeholder text. At the bottom, there's a blue 'Sign Up' button and a link that says 'Have an account? Sign In'.

### 6.3.2 Login:

- Added validation to ensure correct email format and password strength during login.
- Users receive messages such as "Invalid username or password" when credentials are incorrect.
- This helps prevent unauthorized access and improves the overall user experience.



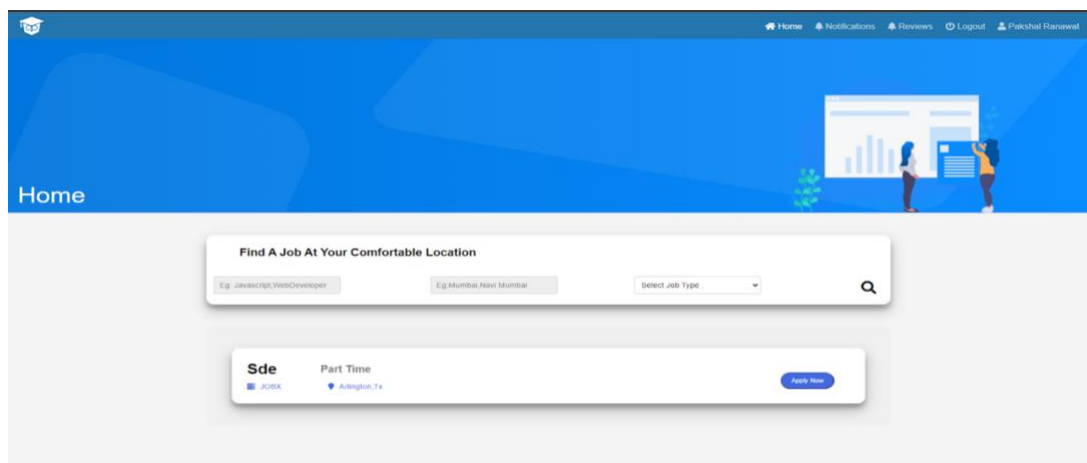
The image shows the 'Sign In' form on the Career Quest website. The form is centered on a light gray background. At the top, there is a blue header with the 'Career Quest' logo. The form itself has a white background and a blue border. It features a 'SIGN IN' title, a 'Select one:' label with radio buttons for 'Employee' and 'Company', and two input fields for 'Enter Email' and 'Enter Password'. A blue 'Login' button is positioned below the password field. At the bottom of the form, there is a link that says 'Don't have an account? Sign Up'.



The image shows the 'Sign In' form on the Career Quest website, identical to the one above, but with an error message displayed at the top. The error message is in a red box and reads: 'Error! Unable to login! Password is wrong.' The form itself is the same as in the previous image, with the 'SIGN IN' title, 'Select one:' label, radio buttons, input fields, and 'Login' button.

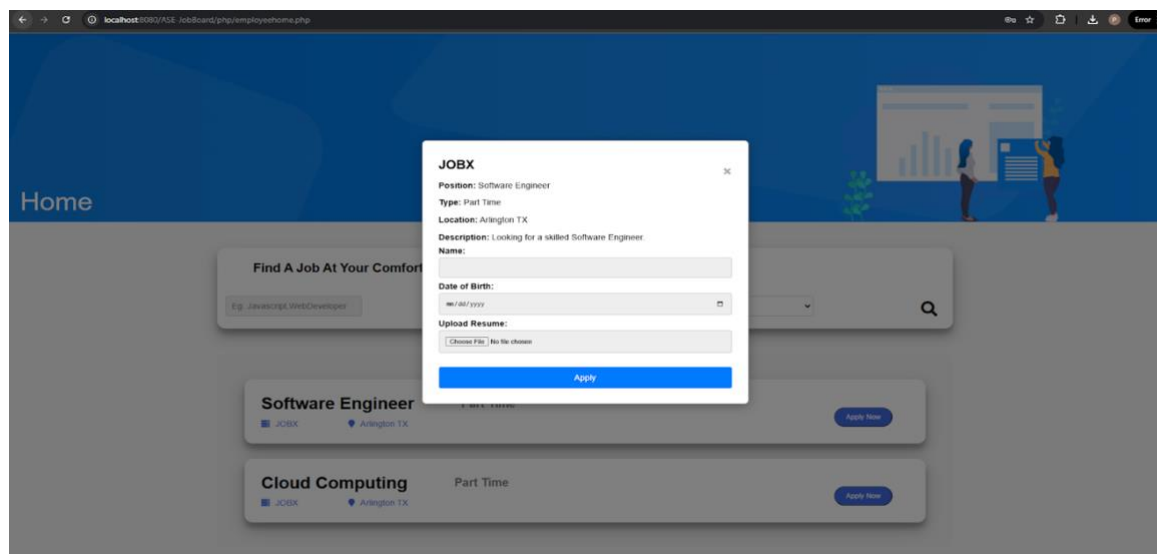
### 6.3.3 Employee Home Page:

- This is the landing page for the user Employee after they login into their account. It has a search bar which lets the user look for their desired job roles. It has advanced filtering by city and also job type which makes it ideal for all sorts of users.
- Notifications: Gives real time updates over the applicants status for a particular role. The reviews takes them to a different page where they can share their experience and give feedback to different organizations.
- Reviews: A way for the former or current employees to provide feedback to different companies and to read about the same.



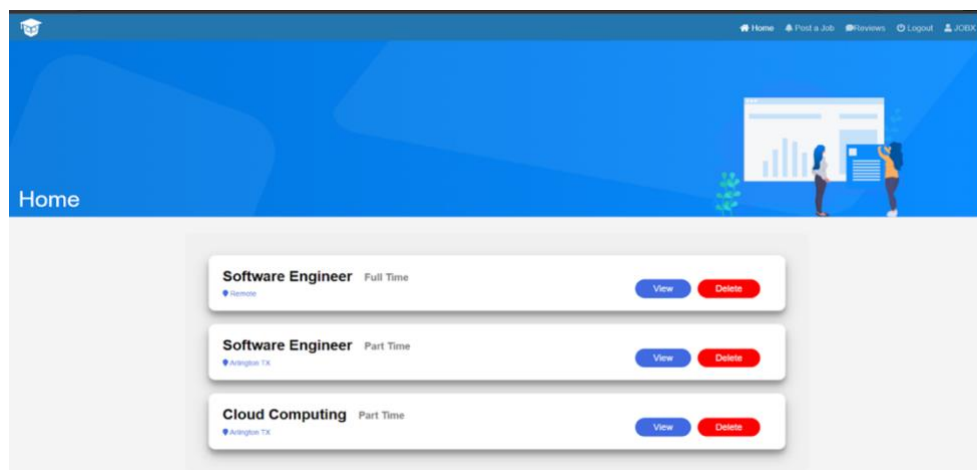
### 6.3.4 Employee Apply Job:

- This is the follow up page in the employee home, after finding their desired job role, company and city.
- Once they click on the tile of the job posting that they wish for, it gives a pop up window for the applicant to enter their personal details and upload their resume.
- Validation for the document format submission is ensured, so that they do not submit wrong formats or files.



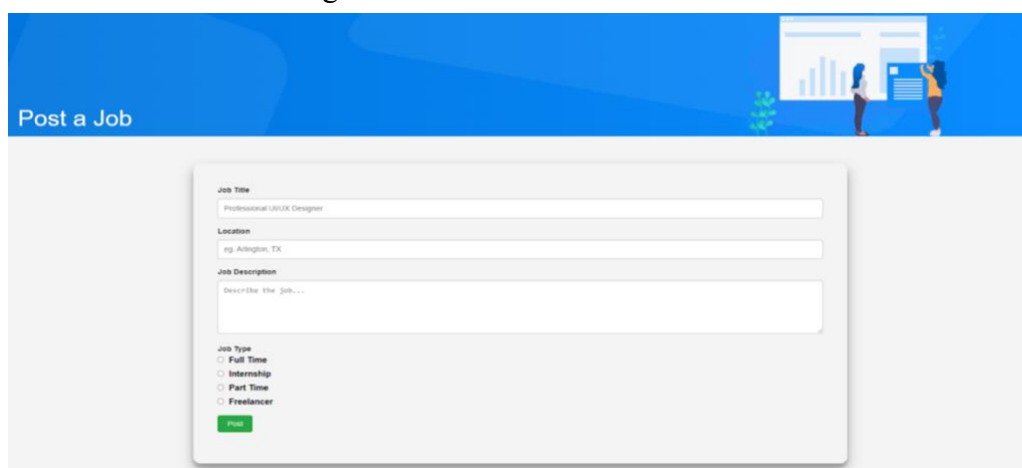
### 6.3.5 *Company Home Page*

- This is the landing page of our other user, company. It has a similar interface to that of the job seeker to ensure consistency but has few changes according to the requirement.
- It has a home page, where all their listings are present and they can expand it further to view the applications that they have received for a particular job role and proceed with the shortlisting. They can delete the job posting once they have found the right candidate.
- Reviews: They can view the reviews made by employees about them to understand how well the company is being perceived in their perspective.
- Post a Job: This lets the companies create new job postings, which will be visible to the candidates trying to secure a position.



### 6.3.6 *Company Post a Job*

- Post a Job mainly deals with the specifics of the role and position that the company is hiring for.
- It has the job title, location, job type and description which gives the complete information over what the candidate will be doing on a daily basis and what would make them a great fit for this role.





## 7. TESTING

### 7.1 Manual Testing

- Performed by QA personnel or users to validate functions.
- Covers real-world usability checks, like signing up and logging in.
- Ensures the application behaves as expected in hands-on scenarios.

### 7.2 Automated Testing

- Uses tools (e.g., PHPUnit for PHP) to check functionality with code scripts.
- Faster, repeatable, and ideal for testing updates without human intervention.
- Validates specific parts (unit tests) and user flows (integration/system tests).

### 7.3 Conducting Different Types of Testing

#### 7.3.1 Unit Testing

- Focuses on testing individual functionalities such as the signup forms and validations for both company and employee registrations.
- Each function or component is tested in isolation to ensure proper behaviour of input fields, validation rules, and submission buttons.
- PHP unit tests are used to validate each field in the forms, verifying that the data is correctly handled and that error messages appear as expected for invalid inputs.
- Ensures accuracy in data handling, particularly for key fields like company name, email, and password, in both company and employee signup forms.

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post Condition	Actual Result	Status
TC001	Company Signup	User is on the signup page	1. Open signup page. 2. Fill in company details. 3. Submit form.	Company Name, Email, Password	Company should be signed up successfully.	Company account is created, user is logged in	Passed: Company details were successfully submitted, account created.	Pass
TC002	Employee Signup	User is on the signup page	1. Open signup page. 2. Fill in employee details. 3. Submit form.	Name, Email, Password	Employee should be signed up successfully.	Employee account is created, user is logged in	Passed: Employee details were successfully submitted, account created.	Pass

### 7.3.2 Integration Testing

- Integration testing focuses on verifying that interconnected modules work well together for smooth functionality.
- Key scenarios tested include a seamless flow from signup to login, posting jobs, and viewing applicants to confirm end-to-end user workflows.
- This includes validating data consistency, session handling, and database interactions across login, dashboard, job postings, and notification views.
- Tests are designed to ensure users can move through multiple steps and pages without errors, reinforcing functionality in combined modules like job applications, reviews, and notifications.

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post Condition	Actual Result	Status
TC003	Company Login	Company has an existing account	1. Open login page. 2. Enter valid company credentials. 3. Submit.	Company Email, Password	Company should be logged in successfully.	User is logged in and redirected to dashboard	Passed: Valid credentials entered, login successful, redirected to company dashboard.	Pass
TC004	Employee Login	Employee has an existing account	1. Open login page. 2. Enter valid employee credentials. 3. Submit.	Employee Email, Password	Employee should be logged in successfully.	User is logged in and redirected to dashboard	Passed: Valid credentials entered, login successful, redirected to employee dashboard	Pass
TC006	Company Post Job	Company is logged in	1. Navigate to job posting page. 2. Fill in job details. 3. Submit.	Job Title, Job Position,  Job Location,  Job type, Description,	Job post should be created successfully.	Job post is visible in the job listing	Passed: Job post was created successfully, visible on the company home page.	Pass
TC007	Company Delete Job	Company has posted a job	1. Navigate to company home page. 2. Select a job. 3. Click "Delete".	Job ID	Job should be deleted successfully.	Job is no longer visible in the job listing	Passed: Job was successfully deleted and removed.	Pass

TC008	Employee Apply Job	Employee is logged in	1. Navigate to employee home page. 2. Select a job. 3. Click "Apply".	Job ID, Employee ID	Employee should be able to apply for the job.	Application is submitted for review	Passed: Employee successfully applied for the job. Application submitted.	Pass
TC010	Employee Add Reviews	Employee is logged in	1. Navigate to review page. 2. Select a company. 3. Submit review.	Company Name, Rating, Comment	Review should be successfully submitted.	Review is visible in the company reviews	Passed: Employee able to add reviews.	Pass
TC012	Employee View Notification	Employee is logged in	1. Navigate to notifications page. 2. View new notifications.	Notification content	Employee should see all the notifications.	Notifications should be visible.	Passed: Able to see the notification	Pass

### 7.3.3 System Testing

- System testing validates the complete application in real-world scenarios to ensure it meets all functional and non-functional requirements.
- Tests simulate end-to-end workflows, such as admin company verification, job posting management, and applicant viewing, covering full platform functionality.
- This phase checks for data consistency, accurate navigation across pages, and UI consistency across modules, confirming smooth operation for both company and employee accounts.
- Key test areas include company and employee module interactions, ensuring each user action (like posting jobs, viewing applicants, or leaving reviews) functions as expected in the live environment.

Test Case ID	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post Condition	Actual Result	Status
TC005	Company Verification (by Admin)	Company has signed up	1. Admin Logs in. 2. Admin reviews company details. 3. Admin clicks "Verify".	Company ID	Admin should successfully verify the company account.	Company account status is updated to "Verified"	Passed: Admin verified the company successfully, status updated to "Verified".	Pass
TC009	Company View Applicants	Company is logged in	1. Navigate to company home page.	Job ID	Company should see the list of	Applicants are listed for	Passed: Applicants are	Pass

		and job is posted	2. Select the job and click view.		applicants for the job.	the particular job role.	listed in the table.	
TC011	Company View Reviews	Company is logged in	1. Navigate to "Reviews" page.	Company ID	Company should see all reviews submitted by employees.	Reviews are displayed for the company	Passed: Company can view the reviews.	Pass
TC013	Logout	User is logged in	1. Click "Logout" button.	-	User should be logged out successfully.	User is redirected to the login page	Passed: User is logged out successfully	Pass

### 7.3.4 User Acceptance Testing

- UAT confirms the project meets business requirements and user expectations before deployment.
- End-users test real-world functionality, including signup, login, job applications, and notifications, to ensure ease of use.
- Testing focuses on validating usability, workflows, and feature accessibility from a user perspective.
- Feedback gathered here determines if the project is ready for launch or needs further adjustments, ensuring a smooth, user-friendly experience.

## 7.4 Test Cases Screenshots

### 7.4.1 main.py

- main.py serves as the entry point for running all test cases. It contains the necessary logic to invoke the testing framework (like pytest) and is responsible for executing the test scripts in the correct order.
- This file can be run directly to execute all the tests associated with the application.

```

main.py > ...
1 import pytest
2 from selenium import webdriver
3 from test_login import login_test
4 from test_signup import signup_test
5 from test_post_a_job import post_job_test
6 from test_apply_a_job import apply_job_test
7 from test_delete_job import delete_job_test
8 from test_company_verification import company_verification_test
9 import random
10
11 # URL for the application
12 url = "http://localhost:8080/ASE-JobBoard/index.php"
13
14
15 def test_main_signup():
16     driver = webdriver.Chrome() # Start a new browser session
17     try:
18         randomNum = str(random.randint(1000, 9999))
19         signup_test(
20             driver,
21             url,
22             user_type="Company",
23             email=f"xyz{randomNum}@example.com",
24             full_name=f"XYZ {randomNum}",
25             phone="1234567890",
26             password=f"xyz@{randomNum}"
27         )
28     finally:
29         driver.quit() # Close the browser after signup test is done
30

```

### 7.4.2 *test\_register*

- This Test file includes test cases for validating the user registration functionality. These tests ensure that the registration process works as expected for various user roles (such as employers and job seekers).
- It verifies that the user data is correctly processed and that any required fields, such as email, password, and name, are validated properly.

```
def signup_test(driver, url, user_type, email, full_name, phone, password):
    driver.get(url)

    signup_btn = driver.find_element(By.XPATH, "//a[contains(text(), 'Sign Up')]")
    signup_btn.click()

    # Select radio button dynamically based on user_type
    radio_button = driver.find_element(By.XPATH, f"//input[@type='radio' and @value='{user_type}']")
    radio_button.click()

    # Fill in the signup form
    email_field = driver.find_element(By.NAME, "email")
    full_name_field = driver.find_element(By.NAME, "full_name")
    phone_field = driver.find_element(By.NAME, "phone")
    password_field = driver.find_element(By.NAME, "password")
    confirm_password_field = driver.find_element(By.NAME, "cpassword")

    email_field.send_keys(email)
    full_name_field.send_keys(full_name)
    phone_field.send_keys(phone)
    password_field.send_keys(password)
    confirm_password_field.send_keys(password)

    # Click register button
    register_button = driver.find_element(By.NAME, "submit")
    register_button.click()
```

### 7.4.3 *test\_apply\_job*

- This test contains test cases that validate the job application process for job seekers. It ensures that the user can apply for a job listing through the platform.
- The tests verify that the job application form is correctly submitted, the job application is saved in the database, and notifications or feedback are provided after submission.

```
def apply_job_test(driver, url, user_type, email, password, job_position, job_type, name, dob, resume_path):
    # Login as an employee
    login_test(driver, url, user_type, email, password)

    # Wait for the home page to load
    WebDriverWait(driver, 10).until(
        EC.visibility_of_element_located((By.XPATH, "//div[text()='Home']"))
    )

    # Locate the job based on position and location, and click apply
    apply_button_xpath = f"//div[h3[contains(text(), '{job_position}')]//following-sibling::h4[contains(text(), '{job_type}')]//ancestor::div[@class='job']//following-sibling::div[@class='application']"

    try:
        apply_button = WebDriverWait(driver, 10).until(
            EC.element_to_be_clickable((By.XPATH, apply_button_xpath))
        )
        apply_button.click() # Click the apply button
        print("Apply button clicked.")
    except Exception as e:
        print("Apply button not found or not clickable.")
        raise e

    # Wait for the modal to appear
    try:
        modal = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.ID, "jobDetailsModal")) # Adjust modal ID as needed
        )
        print("Application modal is displayed.")
    except Exception as e:
        print("Application modal did not appear.")
        raise e

    # Fill out the application form
    name_field = driver.find_element(By.ID, "name")
    dob_field = driver.find_element(By.ID, "dob")
    name_field.send_keys(name)
    dob_field.send_keys(dob) # Enter date in mm/dd/yyyy format

    # Upload resume
    resume_upload_field = driver.find_element(By.ID, "resume") # Adjust ID if necessary
    resume_upload_field.send_keys(os.path.abspath(resume_path)) # Use absolute path for the resume file

    # Click submit button in the modal
    submit_button = driver.find_element(By.XPATH, "//button[contains(text(), 'Apply') and @type='submit']") # Adjust if needed
    submit_button.click()
```

#### 7.4.4 *test\_post\_job*

- This testcase contains tests for the job posting functionality available to employers.
- This ensures that companies can create new job listings by providing the required information such as job title, description, and application deadline.

```
def post_job_test(driver, url, user_type, email, password, job_position, job_location, job_description, job_type):
    # log in as a company
    login_test(driver, url, user_type, email, password)

    # Wait for the page to load and ensure login was successful
    WebDriverWait(driver, 10).until(
        EC.visibility_of_element_located((By.XPATH, "//div[text()='Home']"))
    )

    # Click on "Post a Job"
    post_job_button = driver.find_element(By.XPATH, "//a[contains(text(),'Post a Job')]")
    post_job_button.click()

    # Fill out the job posting form
    job_position_field = driver.find_element(By.NAME, "jobposition")
    job_location_field = driver.find_element(By.NAME, "joblocation")
    job_description_field = driver.find_element(By.NAME, "jobdescription")

    job_position_field.send_keys(job_position)
    job_location_field.send_keys(job_location)
    job_description_field.send_keys(job_description)

    # Select "Full Time" radio button
    full_time_radio = driver.find_element(By.XPATH, f"//input[@type='radio' and @value='{job_type}']")
    full_time_radio.click()

    # Click on the "Post Job" button
    post_job_submit_button = driver.find_element(By.NAME, "postjob")
    post_job_submit_button.click()
```

#### 7.4.5 *Results*

```
===== 1 passed in 10.86s =====
PS F:\files\Languages\Python\ase-testing> python main.py
===== test session starts =====
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0 -- C:\Python312\python.exe
cachedir: .pytest_cache
metadata: {'Python': '3.12.6', 'Platform': 'Windows-11-10.0.22631-SP0', 'Packages': {'pytest': '8.3.3', 'pluggy': '1.5.0', '3.1.1'}, 'JAVA_HOME': 'C:\\Program Files\\Java\\jdk-17'}
rootdir: F:\files\Languages\Python\ase-testing
plugins: html-4.1.1, metadata-3.1.1
collected 4 items

main.py::test_main_signup
DevTools listening on ws://127.0.0.1:61887/devtools/browser/4642e3cf-76bb-4dce-a35c-b53dfc2c102f
PASSED
main.py::test_main_login
DevTools listening on ws://127.0.0.1:61919/devtools/browser/70ac969e-88ea-4cea-9415-8fcd958aadbe
PASSED
main.py::test_post_job
DevTools listening on ws://127.0.0.1:61951/devtools/browser/77b4bcd5-4fbb-4bd5-b2af-1eea5b670136
PASSED
main.py::test_apply_job
DevTools listening on ws://127.0.0.1:61984/devtools/browser/fbd7f7da-1c37-4aed-8494-59cb7d1626c4
PASSED
```

### 7.4.6 Automation Testing Report

- The report compiles all results and provides a structured summary of test passes, failures, and overall application performance. With all tests passing, this indicates a stable release.

#### report.html

Report generated on 06-Nov-2024 at 19:41:20 by [pytest-html](#) v4.1.1

##### Environment

Python	3.12.6
Platform	Windows-11-10.0.22631-SP0
Packages	<ul style="list-style-type: none"><li>• pytest: 8.3.3</li><li>• pluggy: 1.5.0</li></ul>
Plugins	<ul style="list-style-type: none"><li>• html: 4.1.1</li><li>• metadata: 3.1.1</li></ul>
JAVA_HOME	C:\Program Files\Java\jdk-17


##### Summary

4 tests took 00:00:39.

(Un)check the boxes to filter the results.

☐ 0 Failed, ☒ 4 Passed, ☐ 0 Skipped, ☐ 0 Expected failures, ☐ 0 Unexpected passes, ☐ 0 Errors, ☐ 0 Reruns

[Show all details](#) / [Hide all details](#)

Result 	Test	Duration	Links
Passed	main.py::test_main_signup	00:00:10	
Passed	main.py::test_main_login	00:00:09	
Passed	main.py::test_post_job	00:00:10	
Passed	main.py::test_apply_job	00:00:10	

## **8. MAINTENANCE**

### **8.1 Corrective Maintenance**

- Resolving issues in the job application process, such as form submission errors.
- Fixing login issues or bugs in user role permissions.
- Addressing any bugs in the notification or tracking systems for job applications.

### **8.2 Adaptive Maintenance**

- Updating the system to work with new versions of MariaDB, PHP, or Apache if they release updates that impact functionality.
- Modifying features to comply with new business rules or hiring practices.
- Making adjustments to ensure compatibility with new security standards, browsers, or mobile devices.

### **8.3 Perfective Maintenance**

- Adding new features, like advanced job matching algorithms or personalized job recommendations.
- Improving the employer dashboard with analytics to track hiring trends.
- Expanding profile options for job seekers, like adding video introductions or portfolio uploads.

### **8.4 Preventative Maintenance**

- Regularly refactoring code to improve performance and readability, reducing technical debt.
- Conducting periodic security audits to prevent potential vulnerabilities.
- Setting up automated backups and redundancies to avoid data loss.



## **9. ROLES AND RESPONSIBILITIES**

### **Pranav Reddy Gudipati**

#### *Sprint 1(Completed):*

- *Create Product Backlog* (23rd Sept – 27th Sept): Organized necessary features, tasks, and issues into a comprehensive backlog for future sprints.

#### *Sprint 3 (Completed):*

- *Implement Security Testing* (11th Oct – 15th Oct): Conducted penetration testing and vulnerability checks on the user login feature.

#### *Sprint 4 (Completed):*

- *Conduct End-to-End Testing* (7th Nov – 9th Nov): Performed complete tests covering user registration, login, job posting, applications, and admin functionalities

### **Pakshal Ranawat**

#### *Sprint 1 (Completed):*

- *Define Project Scope and Objectives* (23rd Sept – 27th Sept) : Identified and documented the goals, objectives, and scope of the project.
- *Define Project Timeline* (23rd Sept – 27th Sept) : Established key milestones and delivery dates for the overall project.

#### *Sprint 2 (Completed):*

- *Set Up Development Environment* (3rd Oct – 4th Oct) : Installed and configured tools like Xampp Server and logged into phpMyAdmin.
- *Set Up Tech Stack* (3rd Oct – 4th Oct) : Ensured PHP was installed and configured in environment variables.

#### *Sprint 3 (Completed):*

- *Develop Job Application Feature* (16th Oct – 22nd Oct) : Enabled job seekers to apply for jobs/events and submit applications.
- *Manage Applicants for Job Posting* (23rd Oct – 28th Oct) : Developed the functionality to manage job applicants within the employer dashboard.

### Harshavardhan Narayanaswamy

#### *Sprint 1 (Completed):*

- *Create Basic Design for Core Pages* (28th Sept – 2nd Oct) : Created the ER diagram to understand system functionalities and module interactions.
- *Create Prototypes for Core Pages* (28th Sept – 2nd Oct) : Developed design prototypes for core pages of the application.

#### *Sprint 2 (Completed):*

- *Develop Registration/Login Feature* (5th Oct – 10th Oct) : Implemented user authentication for login and registration.

#### *Sprint 3 (Completed):*

- *Develop Job Listing Page* (16th Oct – 22nd Oct) : Created the job listing page for job seekers to view available listings.

#### *Sprint 4 (Completed):*

- *Implement Company Verification Process* (29th Oct – 31st Oct) : Created a process to verify company authenticity before they post jobs.

### Rohan Keshav Harish

#### *Sprint 2 (Completed):*

- *Design Database Schema* (5th Oct – 6th Oct) : Created the MySQL schema for the application database.

#### *Sprint 3 (Completed):*

- *Implement User Authentication* (11th Oct – 15th Oct) : Secured user authentication by hashing passwords and using session storage.

#### *Sprint 4 (Completed):*

- *Set Up Data Management System* (1st Nov – 8th Nov) : Ensured scalable and secure data storage with MariaDB for managing job data.

## **10. CONCLUSION**

Career Quest was developed to create a more straightforward and efficient experience for both job seekers and employers, focusing on making the hiring process smooth and accessible. With features like secure login, personalized profiles, and easy-to-use job search tools, the platform brings everything into one place for a user-friendly experience. Job seekers can quickly find opportunities that match their skills, while employers have a reliable way to post job openings, manage applications, and connect with potential candidates.

During development, we put a lot of emphasis on building core features such as the employer dashboard, job posting management, and notifications to keep users informed. This setup allows for clear communication between employers and job seekers, ensuring everyone stays updated on the progress of applications. Through consistent testing, we made sure that each part of the platform works together effectively, creating a smooth journey for users from login to job application and beyond.

Looking ahead, Career Quest has a maintenance plan in place to address any technical issues, adapt to software updates, and continuously improve based on user feedback. These plans aim to keep the platform stable and relevant for the long term, making it a helpful tool that grows with its users' needs.

## **11. FUTURE SCOPE**

### **1. AI-Powered Job Recommendations**

- Use machine learning to match job seekers with jobs based on their profiles, previous applications, skills, and experience. A recommendation engine could consider user behaviour to suggest the most relevant roles.
- Implement a “Best Fit” score on job listings, showing how well a job aligns with a user’s profile.

### **2. Enhanced Employer Analytics and Candidate Insights**

- Provide advanced analytics for employers, such as applicant conversion rates, engagement metrics on job listings, and the ability to filter applicants based on activity or qualification.
- Allow employers to view candidates’ engagement levels, like how active they are on the platform or how frequently they update their profiles.

### **3. Skill Gap Analysis for Job Seekers**

- Offer a tool that compares a job seeker's profile with target job requirements, identifying skill gaps and suggesting resources to bridge them.
- Partner with online learning platforms to recommend courses based on the skill gap analysis, encouraging career growth and readiness for target roles.

### **4. Company Culture Matchmaking**

- Use AI to analyze user preferences for work culture, values, and mission alignment, then match job seekers to companies based on cultural fit in addition to role requirements.

### **5. Skill Exchange and Peer Learning Programs**

- Implement a peer learning or skill exchange feature where users can offer and receive mentoring in specific skills, facilitating growth and community engagement.

### **6. Employee Referral System with Referral Tracking**

- Offer a referral program where current employees can refer potential candidates. If the referred candidate is hired, the referrer could receive rewards, which could be a gamified point system or incentives.

## 12. REFERENCES

1. Previous Submissions – Used content from the project proposal and project progress reports.
2. *Bootstrap Documentation* – Official documentation for Bootstrap, covering grid systems, components, and utilities used in responsive design. Available at: <https://getbootstrap.com/docs/5.3/getting-started/introduction/>
3. *PHP Official Documentation* – A complete guide to PHP functions, libraries, and code examples for server-side scripting. Available at: <https://www.php.net/manual/en/>
4. *Upload Files* – Tutorial on how to upload files using PHP and MariaDB, providing step-by-step guidance. Available at: [https://www.youtube.com/watch?v=JaRq73y5MJk&ab\\_channel=DaniKrossing](https://www.youtube.com/watch?v=JaRq73y5MJk&ab_channel=DaniKrossing)
5. *XAMPP Documentation* – Information on setting up the local server environment using XAMPP, including PHP and MySQL. Available at: <https://www.apachefriends.org/index.html>
6. *MariaDB Documentation* – Guide for working with MariaDB, used in managing the database for the project. Available at: <https://mariadb.com/kb/en/documentation/>
7. *Previous Project Reference* – GitHub repository for a similar project used for cost estimation. Available at: <https://github.com/Md-SabbirHosen/Job-Portal-Project-PHP-MySql-CSS>
8. *Pytest Documentation* – Official documentation for pytest, providing a comprehensive guide on installation, usage, and advanced configurations. Available at: <https://docs.pytest.org>
9. *Selenium WebDriver Documentation* – Selenium’s official WebDriver documentation offering extensive information on setting up automated browser interactions. Available at: <https://www.selenium.dev/documentation/en/>