



IT 309 SOFTWARE ENGINEERING

PROJECT DOCUMENTATION

JOB APPLICATION PORTAL

Prepared by:
Eldar Dadić
Arijan Komšić

Proposed to:
Nermina Durmić, Assist. Prof. Dr.
Ajla Korman, Teaching Assistant

3rd April, 2025

Table of Contents

1. Introduction.....	3
1.1. About the Project.....	3
1.2. Product Roadmap	3
1.3. Release Plan	4
1.4. Project Requirements.....	5
1.4.1. Functional Requirements (User Stories).....	5
1.4.2. Non-Functional Requirements	8
1.5. UML diagrams.....	9
1.5.1. Activity Diagrams.....	9
1.5.2. Class Diagram	10
1.5.3. Sequence Diagrams.....	11
2. Project Structure.....	13
2.1. Technologies.....	13
Backend Technologies	13
Frontend Technologies.....	13
Database Technology.....	13
Testing Tools	13
2.2. Architectural Pattern.....	13
Implementation Structure.....	14
Key Components	14
2.3. Database Entities.....	16
2.4. Design Patterns.....	17
Creational Patterns.....	17
Behavioral Patterns.....	17
Structural Patterns	18
2.5. Project Functionalities and Screenshots	18
Authentication Flows.....	18
Employer Features.....	19
Job Seeker Features.....	22
2.6. Tests.....	24
Authentication Tests (auth.spec.js)	24
Profile Management Tests.....	24
Job Management Tests.....	25
Email Communication Tests	25
Screenshot of all passed tests: 3. Conclusion	25
Project Achievements.....	25
Feature Additions.....	26
Learning Outcomes.....	26
4. Individual Contributions	26

1. Introduction

This document provides the complete documentation for the Job Application Portal project. It includes details about the project, a high-level development plan, functional and non-functional requirements, and UML diagrams that represent system design and workflow.

The project follows Agile development principles and ensures the implementation of best software engineering practices.

GitHub Repository: <https://github.com/edadic/JobApplicationPortal.git>

Deployed Application: <https://jobportal101-5b22e9e7c2e5.herokuapp.com/>

1.1. About the Project

The **Job Application Portal** is a web-based platform designed to connect job seekers with employers. The system enables job seekers to browse and apply for job listings, while employers can post jobs and review applicants. The portal ensures an intuitive and efficient job-seeking and hiring process with an easy-to-use interface and seamless functionality.

1.2. Product Roadmap

The project follows an **Agile Development Approach** and will be completed by 15th June. The Job Application Portal will be developed iteratively in multiple sprints, focusing on delivering core functionalities first and refining them in subsequent releases. Below is the product roadmap:

DATE	NAME	GOAL	FEATURES	METRICS
April 2025	Version 1.0	Launching the core functionalities to establish the foundation of the portal, allowing basic interactions between job seekers and employers.	User authentication, job posting, job browsing, application submission. Ensuring a working backend with essential API endpoints.	Measuring initial adoption through user signup/login success rate and the number of job applications submitted.
May 2025	Version 1.1	Enhancing user experience by introducing profile management, advanced job search features, and an improved filtering system.	Users can edit and update their profiles, search for jobs using multiple filters, and job postings automatically expire after a set period.	Tracking profile completion rates, efficiency in job search results, and improvements in application engagement.
June 2025	Final Release 2.0	Deploying a fully optimized and user-friendly system with advanced functionalities, improved design, and administrative controls.	UI/UX improvements for better accessibility, real-time notifications for job updates and application statuses, administrative management features for platform moderation, and system-wide optimizations for performance and security.	Evaluating user engagement duration, system stability through uptime tracking, and overall platform adoption.

1.3. Release Plan

The project will have **three releases** on GitHub:

Version 1.0

- **Features:**
 - Secure user authentication, including account registration and login for both employers and job seekers.
 - Functionality for employers to post job listings with relevant details such as job title, description, and application deadline.
 - Job seekers can browse available job postings and submit applications.
 - A simple yet functional UI layout for core platform interactions.
 - Initial API implementation supporting CRUD operations for users, jobs, and applications.
- **Tasks:**
 - Develop and integrate frontend and backend services.
 - Design and implement the database schema and establish API endpoints.
 - Conduct functional and usability testing to validate core features.
- **Deadline:** April 20th, 2025

Version 1.1

- **Features:**
 - Profile management allowing users to edit their details, upload resumes, and personalize their experience.
 - Advanced job search with filtering capabilities based on categories, location, and salary range.
 - Improved job posting system where expired job listings are automatically removed.
 - Performance optimizations for faster loading times.
- **Tasks:**
 - Implement advanced job filtering and search optimization.
 - Develop profile management features.
 - Conduct testing for usability improvements.
- **Deadline:** May 11, 2025

Version 2.0

- **Features:**
 - Comprehensive UI/UX improvements for better accessibility and a more engaging experience.
 - Real-time notifications for job updates and application statuses.
 - Administrative management features for platform moderation.
 - System-wide optimizations for performance and security.
- **Tasks:**
 - Improve UI design for a modern and intuitive user experience.
 - Implement real-time notification functionality.
 - Conduct extensive testing to ensure functionality and stability.
 - Deploy the final version and establish post-launch support.
- **Deadline:** June 15, 2025

1.4. Project Requirements

1.4.1. Functional Requirements (User Stories)

1. User Registration

As a user, I want to create an account by selecting my role (Employer or Job Seeker) so that I can access relevant features.

Acceptance Criteria:

- User must provide a valid email and password.
- User must select one role: Employer or Job Seeker.
- Registration fails with appropriate error messages if required fields are missing.
- Confirmation email is sent upon successful registration.

2. User Login

As a registered user, I want to log in securely using my email and password so that I can access my account.

Acceptance Criteria:

- Only registered users with correct credentials can log in.
- Users receive error messages for incorrect credentials.
- Passwords must be encrypted in the database.
- Sessions/tokens are created upon successful login.

3. Profile Management

As a user, I want to edit my profile details so that my information stays up to date.

Acceptance Criteria:

- Users can update personal details (e.g., name, bio, contact).
- Changes are saved immediately and reflected in their profile.
- Form validations prevent submission of invalid data.

4. Employer Job Posting

As an employer, I want to post job listings so that I can attract potential candidates.

Acceptance Criteria:

- Employers can input job title, description, requirements, location, and salary.
- Posting appears in the job listings for job seekers.
- Only authenticated employers can post jobs.

5. Edit Job Listings

As an employer, I want to update or delete job postings so that I can manage job openings efficiently.

Acceptance Criteria:

- Employers can edit all fields of an existing job posting.
- Employers can delete job postings.
- Changes reflect in real-time in job listings.

6. View Applicants

As an employer, I want to see a list of applicants for each job posting so that I can evaluate potential hires.

Acceptance Criteria:

- Each job post has an “Applicants” section visible only to the job creator.
- Employers can view applicant names, resumes, and application dates.

7. Job Seeker Job Browsing

As a job seeker, I want to browse available jobs so that I can find suitable opportunities.

Acceptance Criteria:

- All published job postings are visible to job seekers.
- Jobs are listed with key details (title, employer, location, salary).

8. Job Search & Filters

As a job seeker, I want to filter job listings by category, location, and salary range so that I can quickly find relevant jobs.

Acceptance Criteria:

- Job listings can be filtered using dropdowns or sliders.
- Filtered results load within 2 seconds.
- Users can reset filters easily.

9. Job Application

As a job seeker, I want to apply for jobs so that employers can review my application.

Acceptance Criteria:

- Apply button visible on each job post.
- Users can upload resume and write a short message.
- Duplicate applications for the same job are prevented.

10. Application Status Tracking

As a job seeker, I want to see the status of my applications (e.g., pending, reviewed, rejected) so that I can track my progress.

Acceptance Criteria:

- Each application has a status label.
- Employers can update application statuses from their dashboard.
- Status changes are reflected immediately for job seekers.

11. Job Expiration

As an employer, I want job postings to automatically expire after a specified deadline so that outdated listings do not appear.

Acceptance Criteria:

- Employers set a deadline when posting a job.
- Jobs automatically move to an “Expired” state after the deadline.
- Expired jobs are not visible to job seekers.

12. Job Bookmarking

As a job seeker, I want to save jobs to my favorites so that I can apply later.

Acceptance Criteria:

- Job seekers can bookmark/unbookmark jobs.
- Bookmarked jobs appear in the user dashboard.
- Bookmark status persists across sessions.

13. Email Notifications

As a user, I want to receive email notifications for account updates, job applications, and employer responses so that I stay informed.

Acceptance Criteria:

- Users receive emails for registration, password reset, and application activity.
- Email preferences can be managed from settings.
- Emails are delivered within 1 minute of the event.

14. Resume Upload

As a job seeker, I want to upload my resume when applying for jobs so that employers can review my qualifications.

Acceptance Criteria:

- Acceptable file types: PDF, DOCX (limit 5MB).
- Uploaded resume is stored securely.
- Employers can download/view uploaded resumes.

15. Profile Visibility Control

As a job seeker, I want to set my profile visibility (public or private) so that I can control who sees my information.

Acceptance Criteria:

- Toggle switch available for visibility settings.
- Private profiles are hidden from employer search.
- Changes to visibility settings apply immediately.

16. Employer Dashboard

As an employer, I want a dashboard to manage job postings and applications efficiently.

Acceptance Criteria:

- Dashboard shows list of job postings with status.
- Employers can click into each job to see applicants.
- Employers can edit/delete postings from dashboard.

17. Job Seeker Dashboard

As a job seeker, I want a dashboard to track my applications and saved jobs.

Acceptance Criteria:

- Dashboard shows list of applications with status.
- Saved jobs are accessible with quick apply option.
- Dashboard shows number of active applications.

18. Admin Management

As an admin, I want to manage users and moderate job listings to ensure compliance with policies.

Acceptance Criteria:

- Admin panel lists all users and jobs.
- Admins can block users and remove job posts.
- Audit logs record all admin actions.

19. User Logout

As a user, I want to log out securely so that my account remains protected.

Acceptance Criteria:

- Session/token is invalidated after logout.
- Users are redirected to login page after logout.
- No access to protected routes after logout.

20. Responsive Design

As a user, I want the application to be mobile-friendly so that I can access it from any device.

Acceptance Criteria:

- Site works on screens from 320px to 1920px wide.
- All buttons and forms are accessible via touch.
- No horizontal scrolling on mobile devices.

1.4.2. Non-Functional Requirements

1. Security

The system must encrypt sensitive user data and follow authentication best practices to prevent unauthorized access.

Acceptance Criteria:

- Passwords are hashed using bcrypt or similar algorithm.
- JWT/session tokens expire after inactivity.

2. Performance

The platform should load job listings and profiles within 2 seconds for a seamless user experience.

Acceptance Criteria:

- Job listings and profile pages load within 2 seconds under normal conditions.

3. Availability

The application should have 99.9% uptime to ensure users can access it anytime.

Acceptance Criteria:

- Application monitored with uptime monitoring service.
- SLA is established with hosting provider.

4. Data Backup

The system should have automated daily backups to prevent data loss.

Acceptance Criteria:

- Daily backups scheduled via automated process.
- Backups are stored securely and tested monthly for recovery.

5. Compliance

The system must comply with GDPR regulations to ensure user data privacy.

Acceptance Criteria:

- Users can download or delete their personal data.
- Privacy policy and terms of use are visible to all users.

1.5. UML diagrams

1.5.1. Activity Diagrams

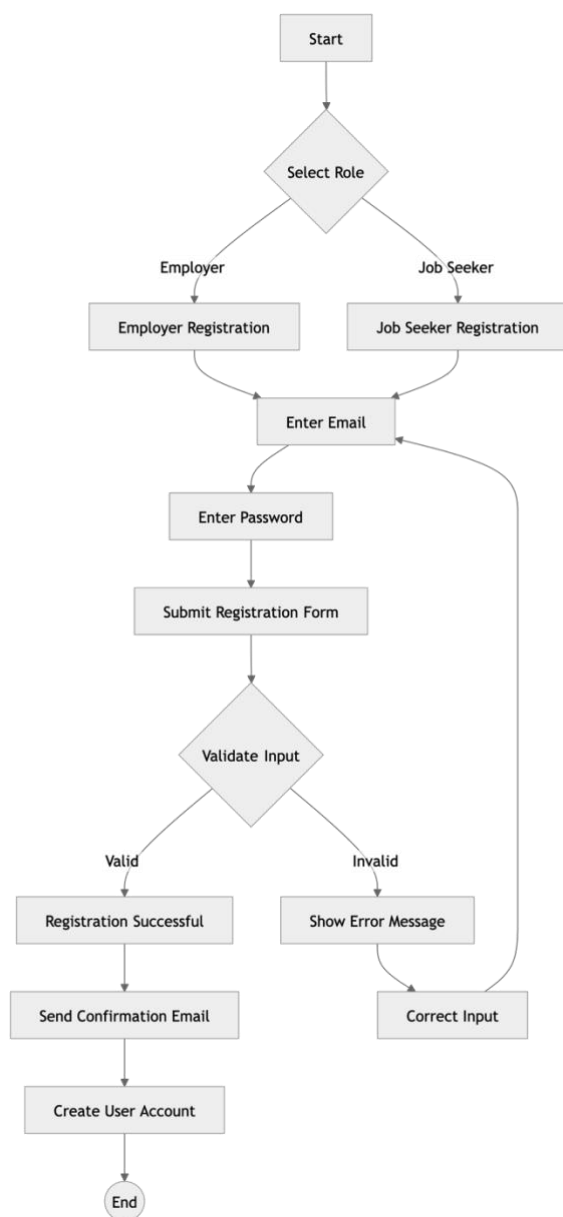


Diagram 2 User Registration Activity Diagram

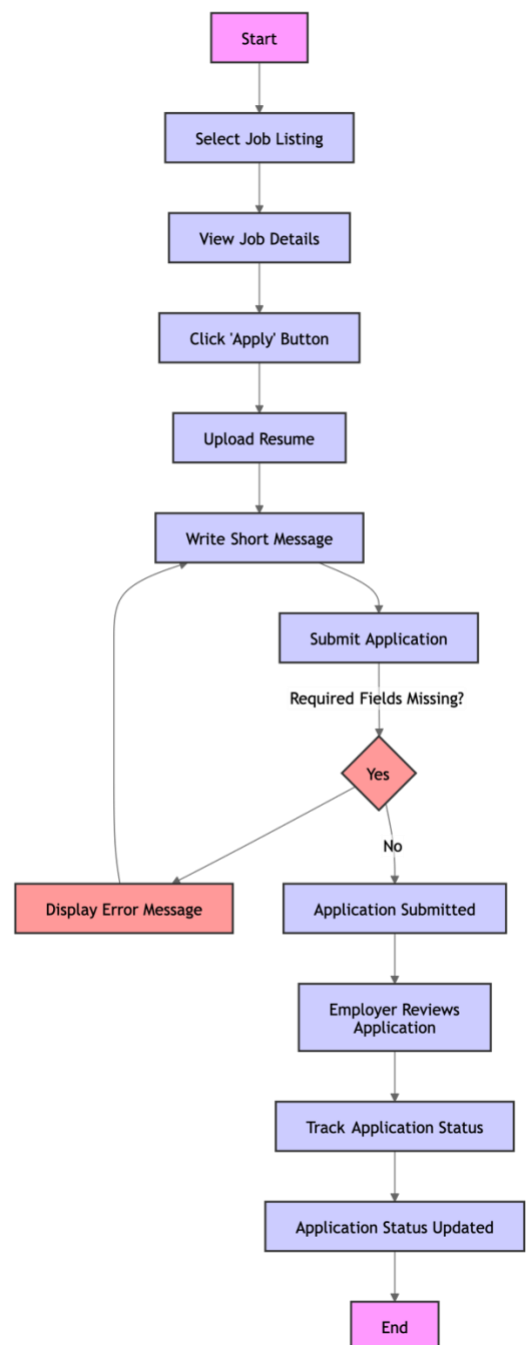


Diagram 1 Job Application Activity Diagram

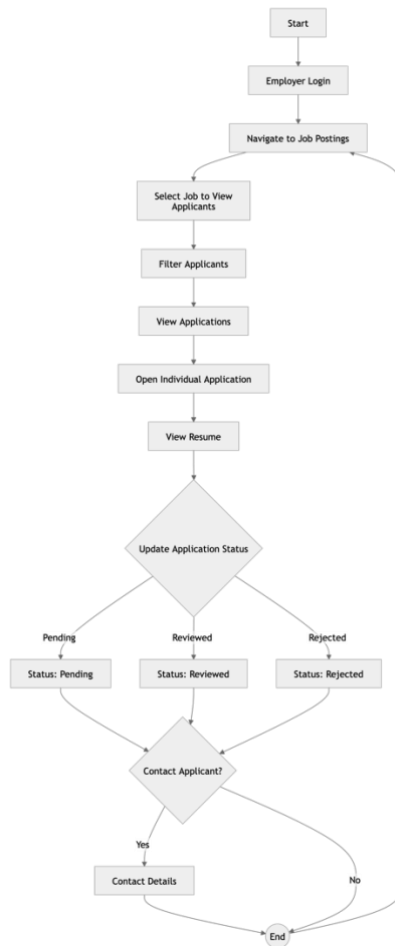


Diagram 3 Employer Reviewing Applicants Activity Diagram

1.5.2. Class Diagram

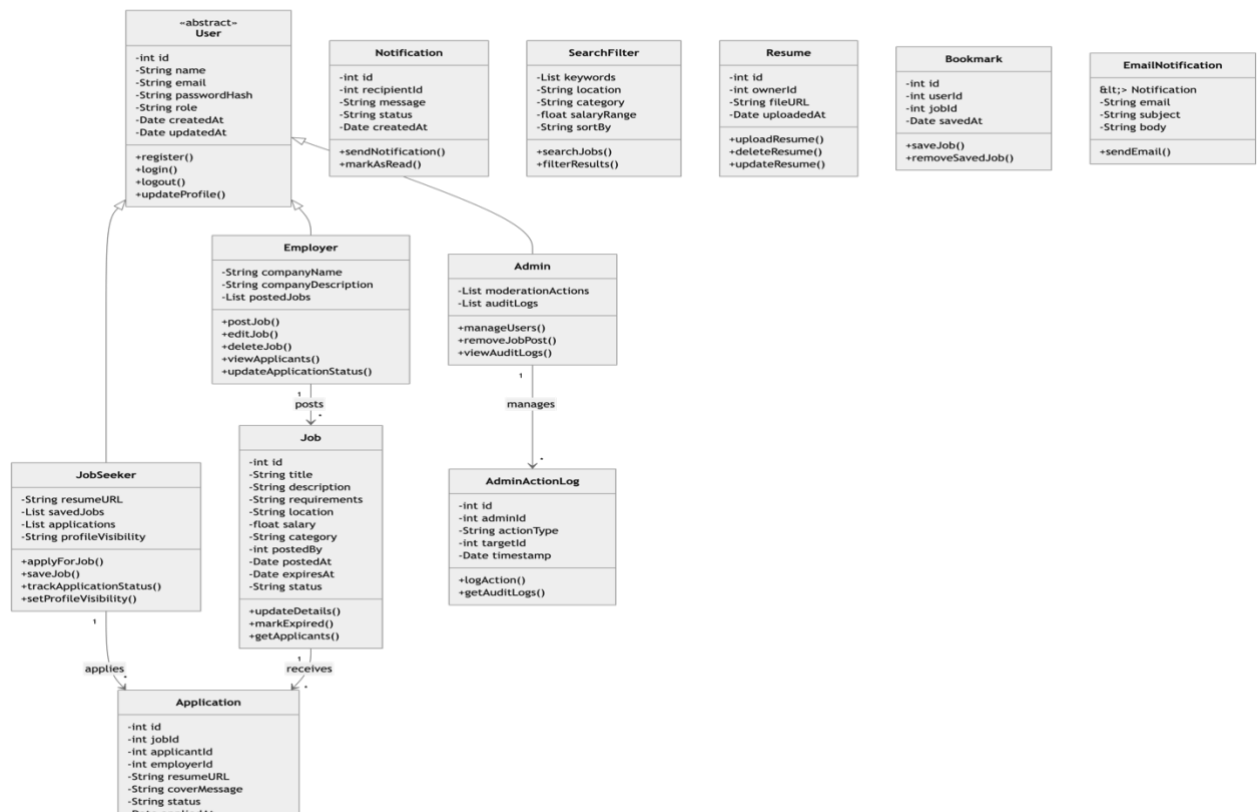


Diagram 4 System Class Diagram

1.5.3. Sequence Diagrams

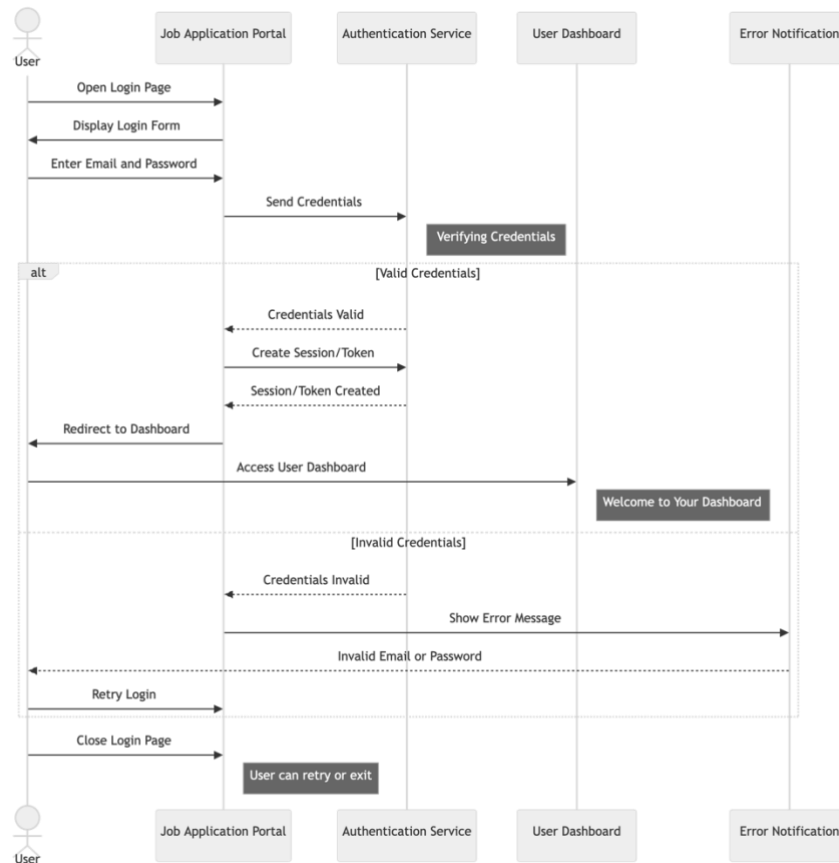


Diagram 5 User Login Sequence Diagram

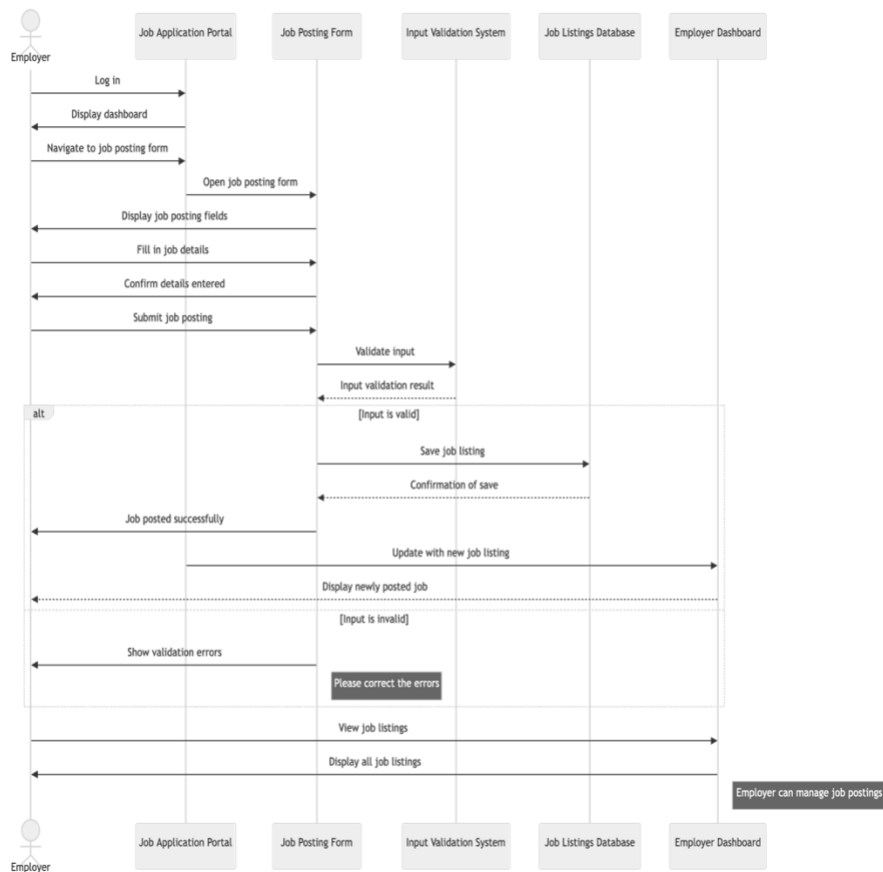


Diagram 6 Posting a Job Sequence Diagram

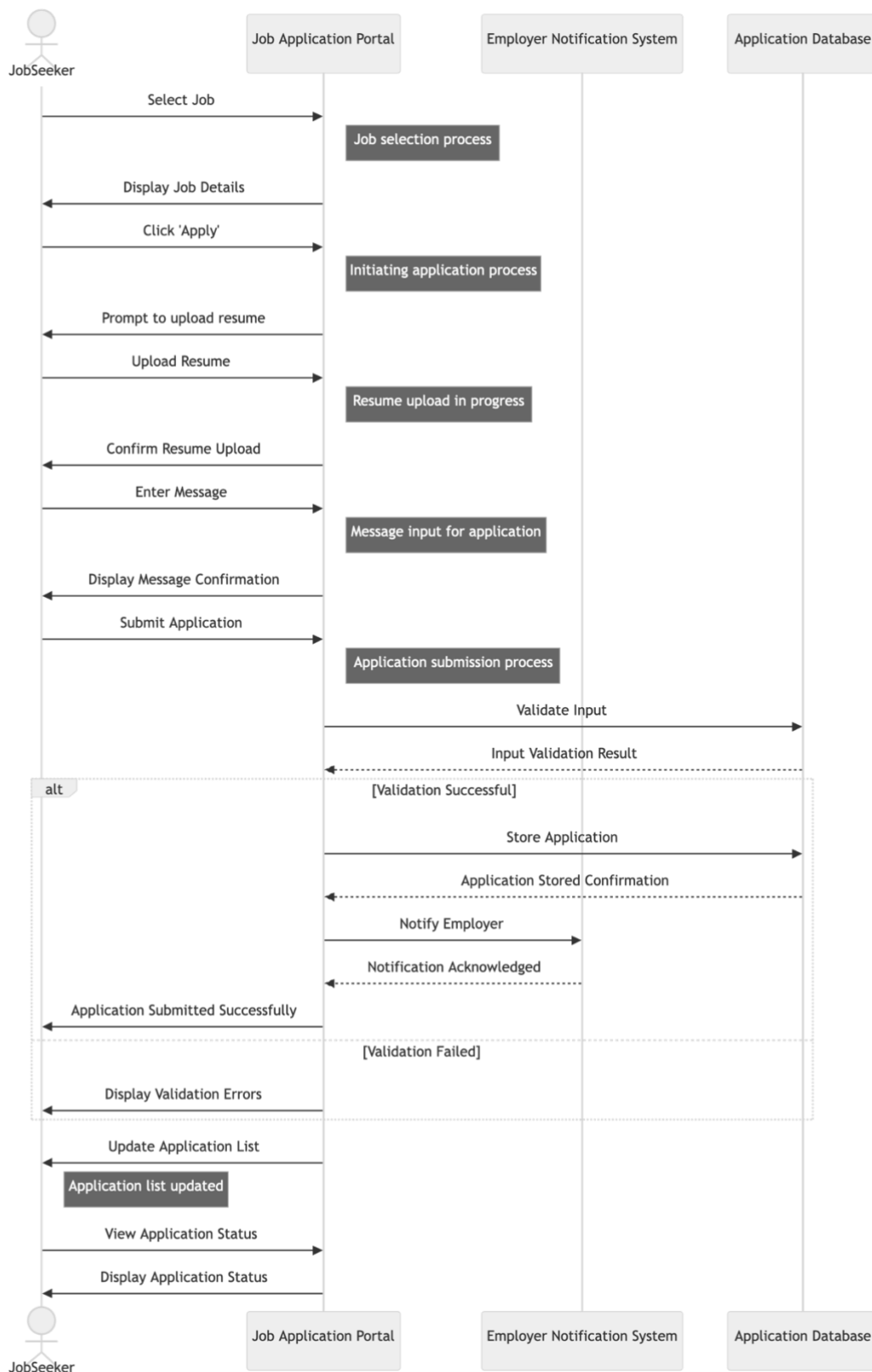


Diagram 7 Applying for a Job Sequence Diagram

2. Project Structure

2.1. Technologies

Backend Technologies

The backend of our application is built using Node.js as the runtime environment, paired with Express.js as the web application framework. This combination provides a robust and scalable server-side architecture. We implemented JWT (JSON Web Tokens) for secure authentication and authorization. For file handling and uploads (particularly resumes), we integrated Multer middleware. The backend also utilizes Nodemailer for handling email communications between employers and job seekers.

Frontend Technologies

Our frontend is developed using React.js, specifically utilizing Vite as the build tool for enhanced development experience and faster build times. For styling, we employed Tailwind CSS, which provided a utility-first approach to building our user interface. React Router handles client-side routing, enabling smooth navigation between different components. We used Axios for making HTTP requests to our backend API, chosen for its promise-based architecture and wide browser support.

Database Technology

The application's data is managed through a MySQL database, selected for its reliability and robust relationship management capabilities. This relational database efficiently handles complex relationships between users (job seekers and employers), job listings, and applications. The structured nature of our data made MySQL an ideal choice over NoSQL alternatives.

Testing Tools

For testing, we implemented Playwright for end-to-end testing, allowing us to automate browser testing across multiple browser engines. Jest was used for unit testing backend components, providing comprehensive test coverage for our API endpoints and business logic.

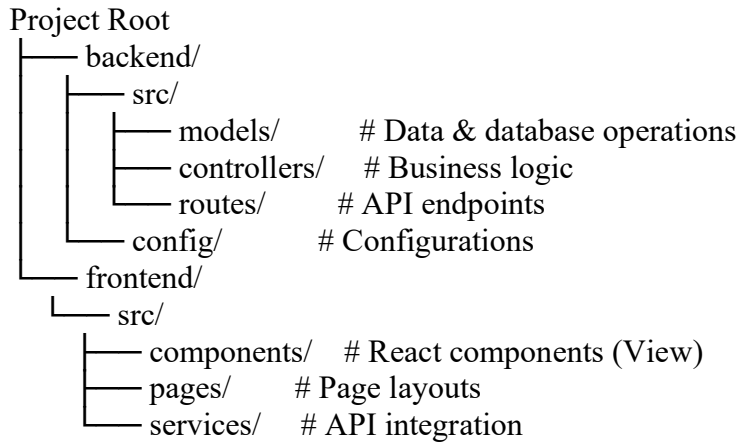
2.2. Architectural Pattern

The Job Application Portal implements the Model-View-Controller (MVC) pattern within a client-server architecture. This separation allows for better code organization, maintainability, and scalability.

Why MVC?

- Clear separation of data, business logic, and presentation layers
- Independent component testing
- Parallel development capability
- Enhanced maintainability

Implementation Structure



Key Components

Model Layer (Database Interactions)

```
class User {
  async findByEmail(email) {
    return await db.query('SELECT * FROM users WHERE email = ?', [email]);
  }
}
```

View Layer (React Components)

```
const JobList = () => {
  return (
    <div className="job-listings">
      {jobs.map(job => (
        <JobCard key={job.id} job={job} />
      ))}
    </div>
  );
};
```

Controller Layer (Business Logic)

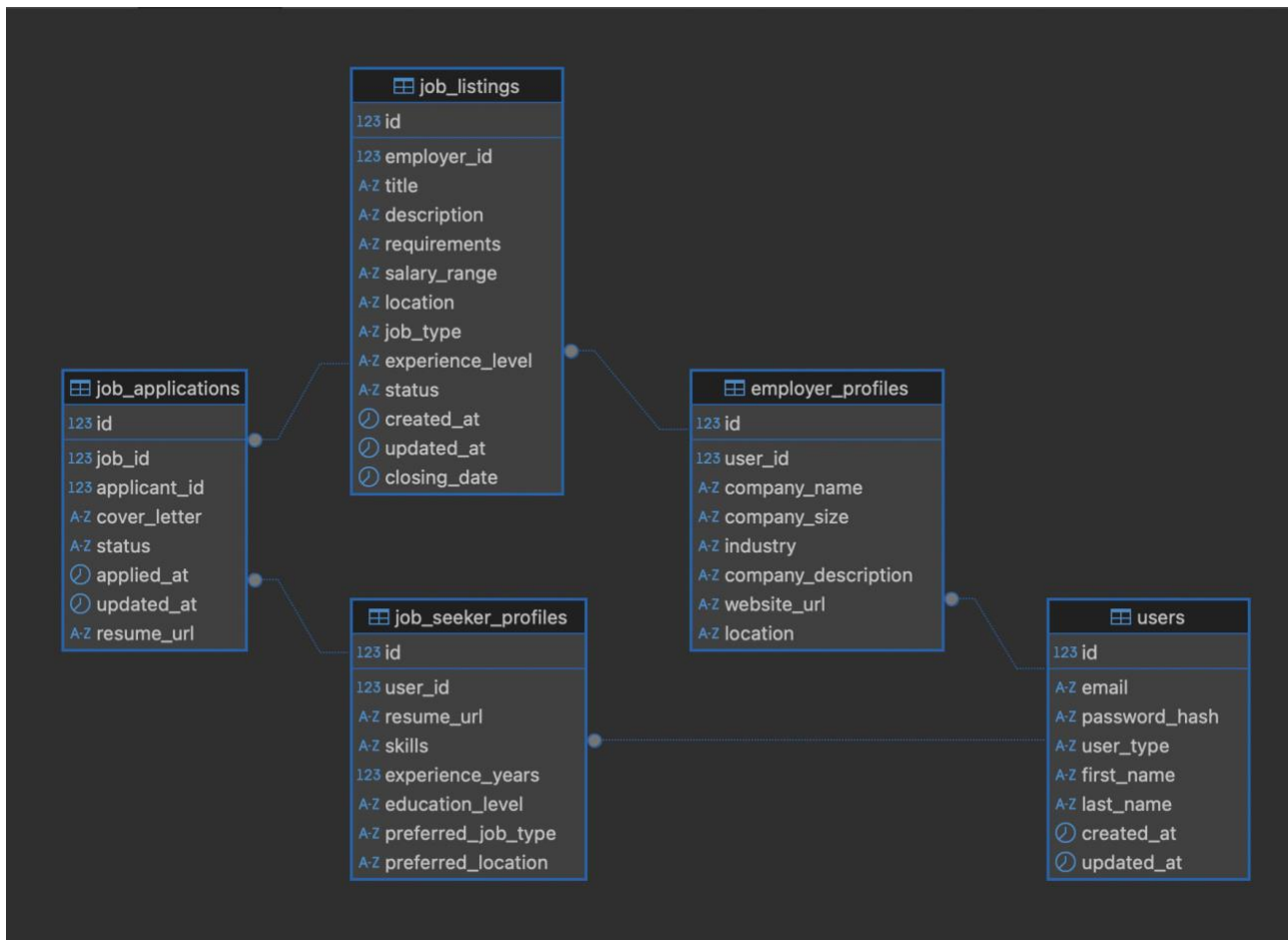
```
class JobController {
  async createJob(req, res) {
    const job = await jobModel.create(req.body);
    res.status(201).json(job);
  }
}
```

This architecture enables:

- Independent scaling of frontend and backend
- Clear separation of responsibilities
- Easier maintenance and testing

- Efficient team collaboration

2.3. Database Entities



Based on the database schema shown in the image, here are the tables and their purposes:

1. users
 - Stores basic user information (email, password, name, type)
 - Common table for both employers and job seekers
2. job_listings
 - Contains all job postings
 - Includes details like title, description, requirements, salary
 - Tracks job status and application deadlines
3. job_applications
 - Records all job applications submitted by job seekers
 - Tracks application status, cover letters, and timestamps
 - Links job seekers to specific job listings
4. employer_profiles
 - Stores detailed information about employer companies
 - Contains company descriptions, size, industry details
 - Includes company contact and location information
5. job_seeker_profiles
 - Maintains professional information for job seekers
 - Stores resumes, skills, education, and experience
 - Tracks job preferences and desired locations

Each table uses relationships through foreign keys:

- job_listings links to employer_profiles through employer_id
- job_applications connects to both job_listings (job_id) and users (applicant_id)

- Both profile tables link to users through user_id

2.4. Design Patterns

Creational Patterns

Factory Pattern

Located in User.js, we implemented the Factory pattern for user creation. This pattern was chosen to handle the creation of different types of users (Job Seekers and Employers) while encapsulating the creation logic. The factory determines the appropriate user type and initializes the corresponding profile, making our user creation process more flexible and maintainable.

```
static async createProfile(userId, profileData, userType) {  
  const table = userType === 'job_seeker' ? 'job_seeker_profiles' : 'employer_profiles';  
  const fields = userType === 'job_seeker'  
    ? '(user_id, resume_url, skills, experience_years, education_level, preferred_job_type, preferred_location)'  
    : '(user_id, company_name, company_size, industry, company_description, website_url, location)';  
  // ...factory implementation for creating different types of user profiles  
}
```

Behavioral Patterns

Observer Pattern

Implemented in the frontend components for real-time application status updates. We chose this pattern to maintain loose coupling between the job application status and the UI components that need to react to these changes. This allows for real-time updates without tightly coupling the status management to the display logic.

```
// Observer pattern implementation for real-time status updates  
const [applications, setApplications] = useState([]);  
  
// Status change notification  
setApplications((prev) =>  
  prev.map((a, i) =>  
    i === idx ? { ...a, application_status: newStatus } : a  
  )  
);  
  
// Observers subscribing to changes  
useEffect(() => {  
  fetchMyJobs()  
}, [])
```

Structural Patterns

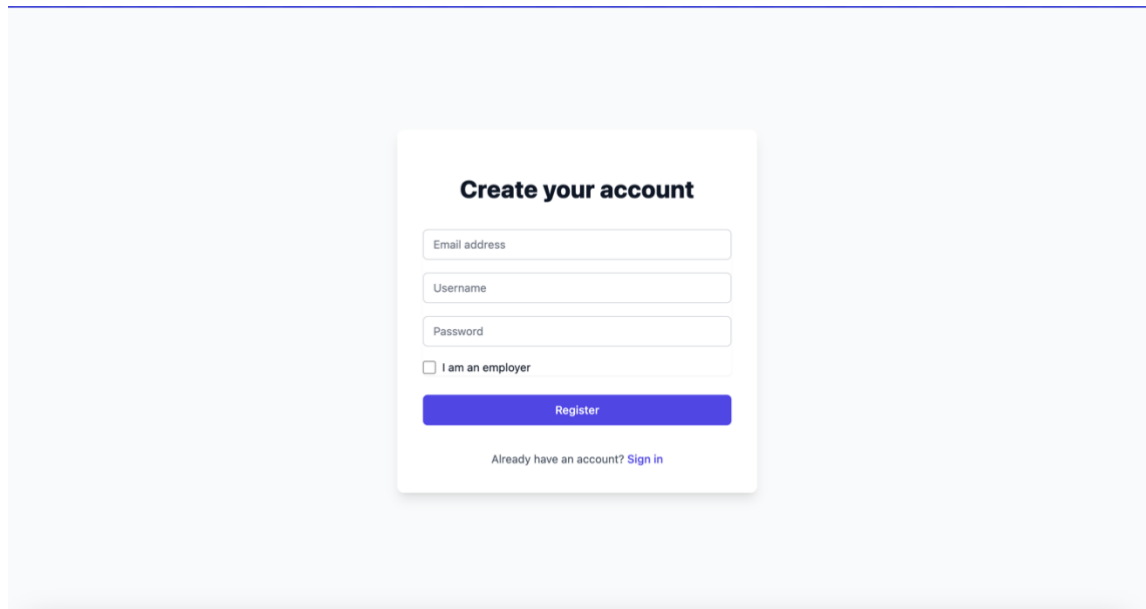
Singleton Pattern

Located in app.js:

```
const app = express();  
  
// Single instance of express application  
  
// Single database connection instance  
const db = require('../config/database');
```

The database connection in database.js maintains a single connection throughout the application:

```
const mysql = require('mysql2/promise');  
const pool = mysql.createPool({  
  // Single database connection pool configuration  
});
```

A screenshot of a web form titled "Create your account". The form is centered on a light gray background. It contains four input fields: "Email address", "Username", "Password", and a checkbox labeled "I am an employer". Below the checkbox is a blue "Register" button. At the bottom of the form, there is a link that says "Already have an account? Sign in".

2.5.

Project Functionalities and Screenshots

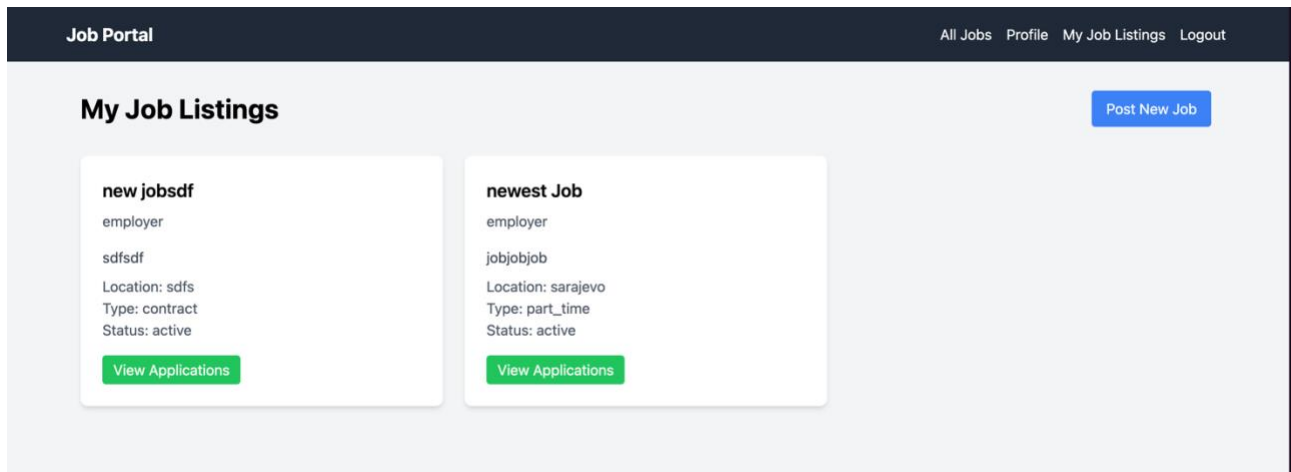
Authentication Flows

Registration Page

- User type selection (Job Seeker/Employer)
- Form validation
- Success/error messages

Login Page

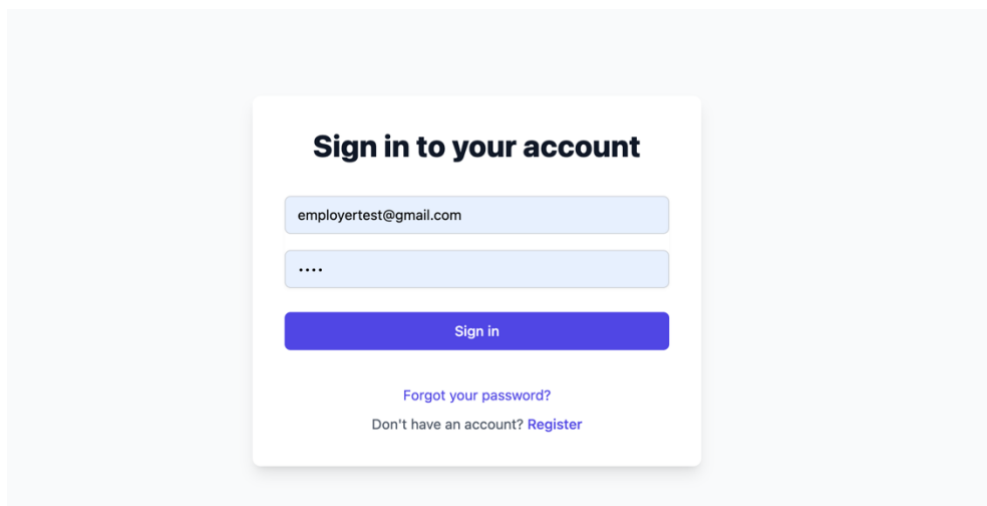
- Email/password input
- Authentication feedback



Employer Features

Job Listing Management

- Create new job posting form
- List of posted jobs
- Edit/delete job options



My Job Listings

new jobsdf

employer

sdfsdf

Location: sdfs

Type: contract

Status: active

[View Applications](#)

[Post New Job](#)

Post New Job

Job Title

Job Title

Description

Description

Location

Location

Job Type

Full Time

Salary Range

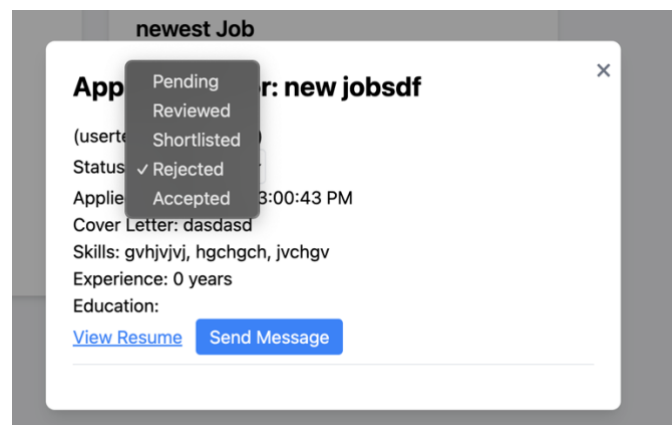
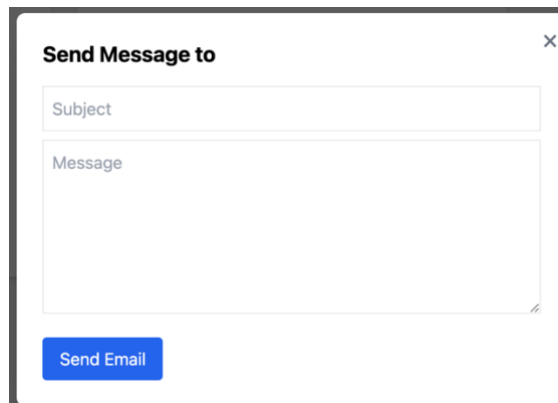
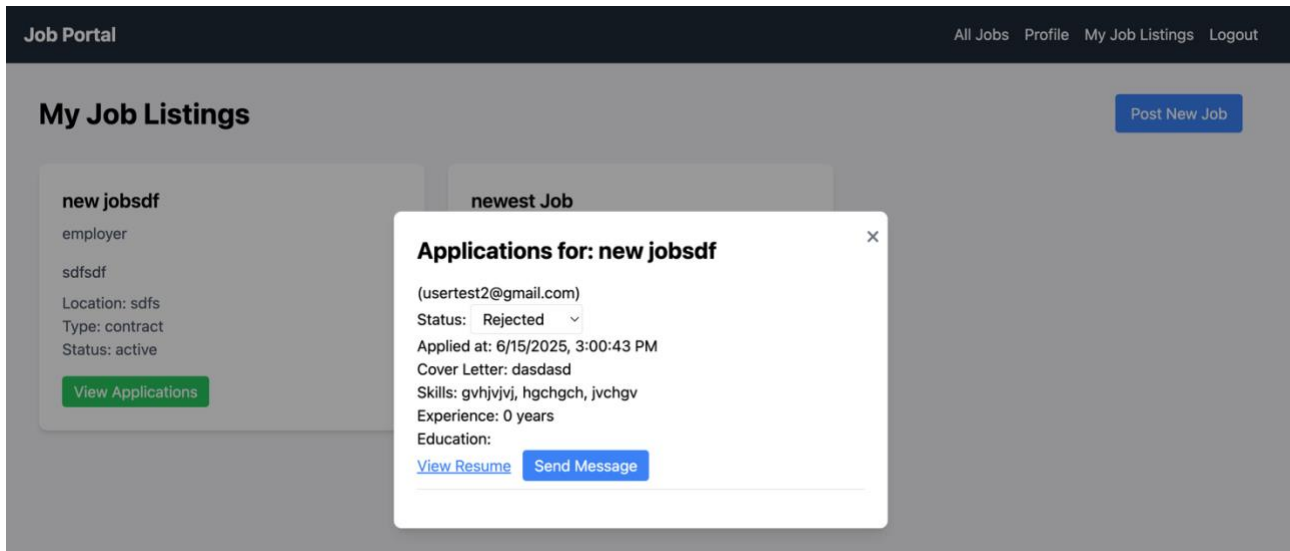
Salary Range

[Cancel](#)

[Post Job](#)

Application Review

- Applications list view
- Status update interface
- Candidate details view
- Email sending modal



Job Seeker Features

Profile Management

- Profile completion form
- Resume upload interface
- Skills and experience input

Job Portal

All JobsProfileMy ApplicationsLogout

My Profile

Name:
Email: usertest2@gmail.com
Account Type: Job Seeker
Skills: gvhjvjvj, hgchgch, jvchg
Experience: 0 years
Education:
Preferred Job Type:
Preferred Location:
Resume: [View Resume](#)

Choose File

No file chosen

Upload Resume

See My Applications

Edit Profile

Job Portal

All JobsProfileMy ApplicationsLogout

My Profile

Name:
Email: usertest2@gmail.com
Account Type: Job Seeker

gvhjvjvj, hgchgch, jvchg

Experience Years

Education Level

Preferred Job Type

Preferred Location

Save

Cancel

Job Search & Application

- Job listings view
- Application submission form
- Cover letter input

Apply Now

Senior Software Engineer

Tech Corp

full_time

San Francisco, CA

We are looking for an experienced software engineer...

Salary Range: \$100,000 - \$150,000

Requirements:

5+ years experience in web development...

Apply Now

Medior Software Engineer

Tech Corp

full_time

San Francisco, CA

We are looking for an experienced software engineer...

Salary Range: \$100,000 - \$150,000

Requirements:

5+ years experience in web development...

Apply Now

Senior Software Engineer

Tech Corp

full_time

San Francisco, CA

We are looking for an experienced software engineer...

Salary Range: \$100,000 - \$150,000

Requirements:

5+ years experience in web development...

Apply Now

Medior Software Engineer

Tech Corp

full_time

San Francisco, CA

Apply for Position

Cover Letter

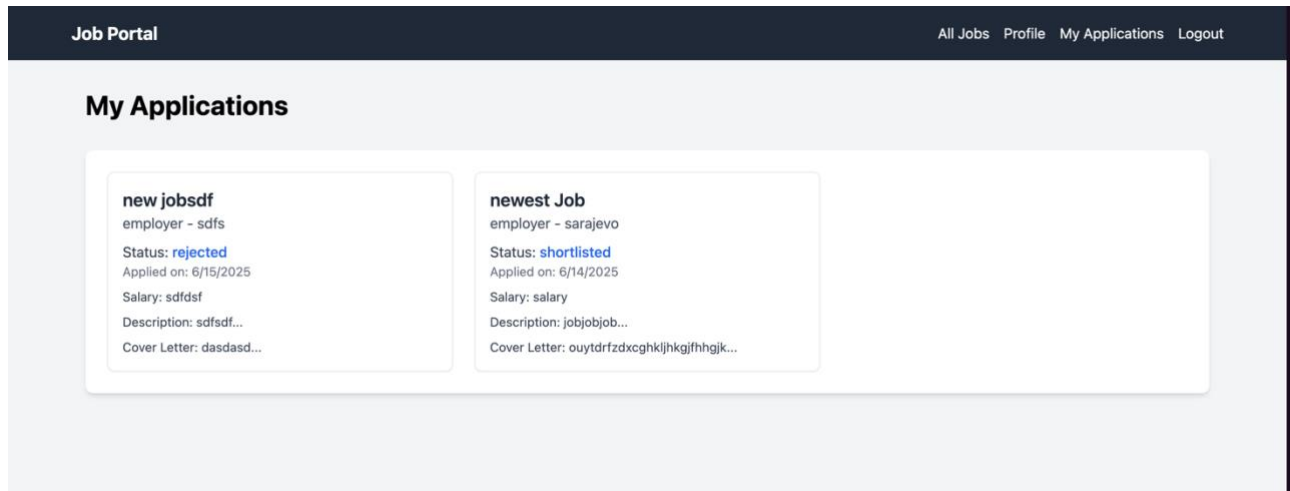
i am very interested

Cancel

Submit Application

Application Tracking

- My Applications dashboard
- Status indicators
- Application history



2.6. Tests

Our Job Application Portal implements end-to-end (E2E) testing using Playwright, located in the e2e directory. The tests cover critical user flows and functionality:

Authentication Tests (auth.spec.js)

```
test.describe('Authentication flows', () => {
  test('User can login as employer', async ({ page }) => {
    // Tests login functionality
  });
  test('Employer can logout', async ({ page }) => {
    // Tests logout functionality
  });
  test('User cannot login with invalid credentials', async ({ page }) => {
    // Tests authentication validation
  });
});
```

Profile Management Tests

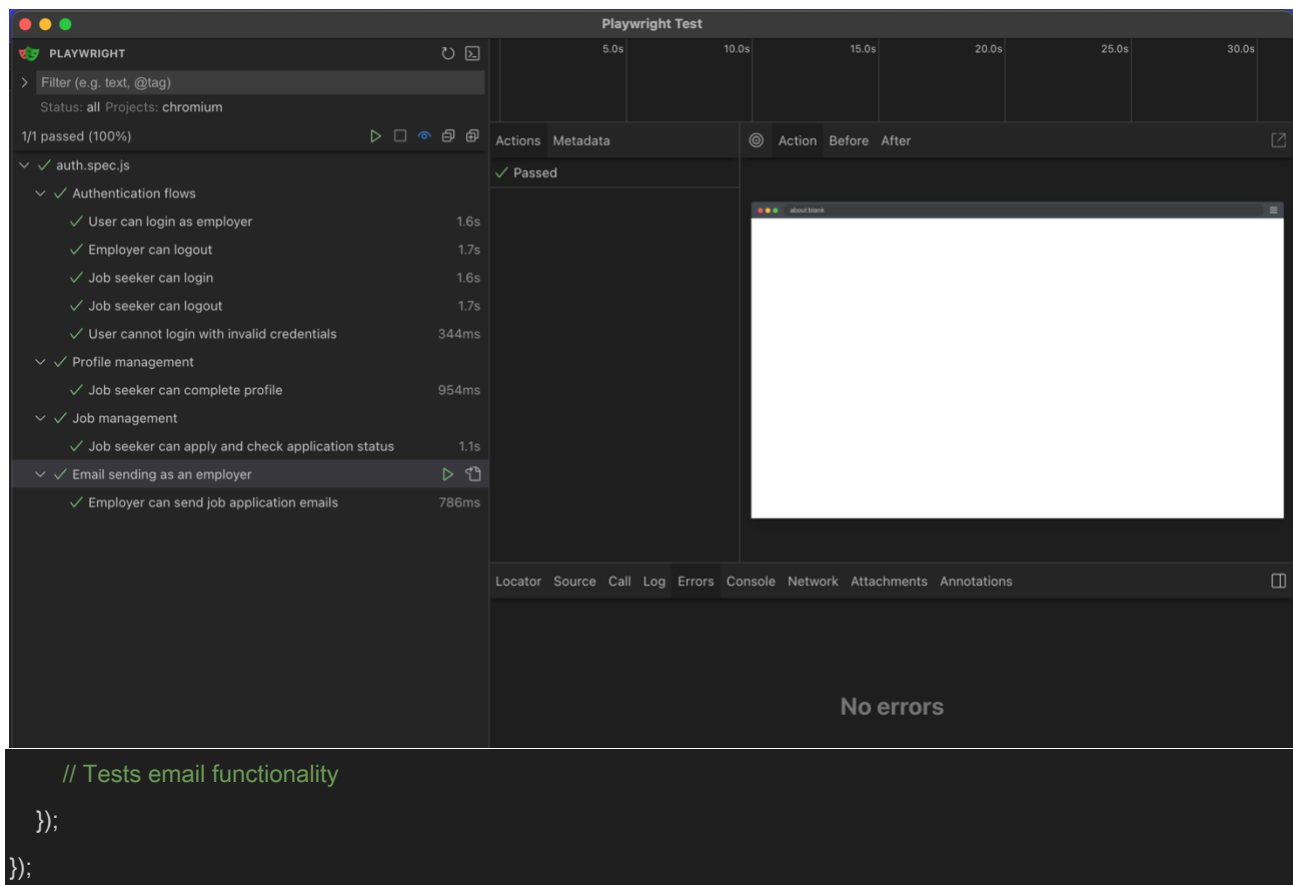
```
test.describe('Profile management', () => {
  test('Job seeker can complete profile', async ({ page }) => {
    // Tests profile creation and updates
  });
});
```


Job Management Tests

```
test.describe('Job management', () => {  
  test('Job seeker can apply and check application status', async ({ page }) => {  
    // Tests job application flow  
  });  
});
```

Email Communication Tests

```
test.describe('Email sending as an employer', () => {  
  test('Employer can send job application emails', async ({ page }) => {
```



The screenshot displays the Playwright Test interface. On the left, a list of test suites and individual tests is shown, all marked with green checkmarks to indicate they passed. The tests include authentication flows, profile management, job management, and email sending. The 'Email sending as an employer' suite is currently selected. The main panel on the right shows a 'Passed' status with a large empty white box representing the browser window. At the bottom, a 'No errors' message is displayed. The interface includes a filter bar, a progress indicator (1/1 passed), and various tabs for Actions, Metadata, and a detailed view of the selected test.

```
    // Tests email functionality  
  });  
});
```

Each test ensures proper functionality across different user roles (Job Seekers and Employers) and validates both successful and error scenarios.

Screenshot of all passed tests:

3. Conclusion

Project Achievements

The implementation of the Job Application Portal successfully achieved its core objectives of connecting job seekers with employers through a modern web application. The use of React with Vite for the frontend and Node.js with Express for the backend proved to be an efficient technology stack, enabling rapid development and good performance.

Key Successes

- Successfully implemented role-based authentication
- Created a responsive and intuitive user interface
- Established real-time application status updates
- Integrated file upload functionality for resumes
- Implemented email notification system

Feature Additions

Advanced Search

- Add filters for job type, location, salary range
- Implement keyword-based search
- Add company ratings and reviews

Analytics Dashboard

- Track application success rates
- Monitor user engagement
- Generate hiring trend reports
-

While the core functionality meets the requirements, there's always room for improvement. The project successfully demonstrates the use of modern web technologies and design patterns, but could benefit from:

- More comprehensive testing coverage
- Enhanced error handling
- Better mobile responsiveness
- Advanced security features

Learning Outcomes

This project provided valuable experience in:

- Implementing design patterns in a real-world application
- Managing full-stack development workflows
- Handling file uploads and storage
- Creating secure authentication systems
- Working with real-time updates

The foundation is solid, and the architecture allows for easy scaling and addition of new features in the future.

4. Individual Contributions

Eldar Dadić (60%)

Eldar was primarily responsible for the backend development, overall system architecture, and core functionalities of the Job Application Portal. His contributions included:

- **System Analysis and Design (SAD) Diagrams** : Created detailed SAD diagrams, including Use Case Diagrams, Sequence Diagrams, and Class Diagrams, to ensure a clear understanding of system requirements and lay the groundwork for development.
- **Backend Development** : Designed and implemented the database schema and built the server-side logic using Node.js and Express. This involved setting up the application's core structure and handling data persistence.
- **API Development** : Developed robust RESTful APIs for various operations such as user authentication, job listing management, and application submissions, ensuring smooth and secure communication between the frontend and backend.

- **Deployment and Infrastructure** : Managed the deployment process and ensured the application was properly hosted and accessible, including setting up necessary server configurations.
- **Collaboration** : Coordinated extensively with Arijan to ensure the frontend design and functionality aligned perfectly with the backend APIs, regularly testing and debugging integrations to maintain system integrity.

Arijan Komšić (40%)

Arijan focused on the frontend development, user experience, and integration aspects of the project. Her contributions included:

- **Frontend Development** : Designed and developed the user interface using React.js, focusing on creating a responsive, intuitive, and visually appealing experience for users. This included implementing components like the JobFormModal as seen in `MyListings.jsx` .
- **Integration with Backend** : Connected frontend components with the API endpoints developed by Eldar, ensuring seamless data flow and dynamic content display between the client and server.
- **User Experience (UX) Enhancements** : Implemented various UX improvements, including form validations, interactive elements, and responsive design, to enhance the overall usability of the portal.
- **Testing and Debugging** : Assisted in comprehensive testing of the frontend application, identifying and resolving bugs to ensure a smooth and error-free user experience.
- **Collaboration** : Worked closely with Eldar to ensure UI elements correctly displayed the data retrieved from the backend, conducting joint debugging sessions and providing feedback to refine API interactions.