NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF SOFTWARE ENGINEERING

STUDENT'S JOB PORTAL WEB APPLICATION

GRADUATION PROJECT – II

SWE-492

STUDENT: ADAM NG'ANZI KIBOKO

SUPERVISOR: PROF. DR. FADI AL-TURJMAN

Nicosia - 2025

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

There are a lot of students that are currently pursuing their education here in Turkish Republic of North Cyprus who come from other countries abroad such as Nigeria, Pakistan, Sudan, UAE and

even Russia with Poland. One of the key points of students picking TRNC for their higher education is the ability to pursue part-time jobs and being able to earn some money to help accommodate their living expenses on the island.

But it has not been easy, especially for the afore mentioned international students as jobs are not easily secured due to lack of proper channels to advertise such positions, hence leaving it to word of mouth and having acquaintances with connections to such jobs.

Now my project aims to overcome that hurdle and ensure that students can be able to apply to jobs straight to the firm instead of relying on agents or connections to secure such jobs, hence level the field where all get equal opportunity to apply for such jobs.

It will help the students to secure jobs and not only that but also diversify the available positions. This is because not only farm jobs will be available but will open opportunities for offices to also recruit interns, assistants etc. to work in their firms. The impact of the project will help improve quality of life for some international students and prevent involvement in illegal methods of acquiring income.

CHAPTER 1

INTRODUCTION

The introduction part of the report will include an overview of the Job portal meaning that it will talk about its purpose, design and implementation. The portal is still under development so it will be using web technologies such as React and Node.js, with Express.js as backend. MongoDB handles the database component of the project. This may be subject to change upon suggestions by the faculty.
Problem statement is eliminating the lack of part-time jobs for individuals, especially international students in TRNC, which also tackles the problem of low quality of life for some of the students residing on the island. Also diversify job opportunities not only having students do hard farm work which might prevent women from also participating.
Objective here is to help more international students to get equal part-time job opportunities and not only that but also have them get good jobs such as cashiers, waiters and assistants which are often only given to natives or Turkish speaking students.
The project is important because it helps students to have more equal opportunities to part-time jobs and earn income that can help them maintain their quality of life on the island. Bridge the gap between natives and international students to job opportunities. Cause literally huge percentage of students are drawn to study here on promise of part-time jobs. Also, it helps keep them from activities such as gambling, prostitution and scams to earn money to cater to their needs here. As costs of living here are rising, making it difficult to survive only on allowance money from back home Here's a concise summary of each chapter:

Chapter 2: Project Description

Details the project's functionalities, goals, and significance. Discusses objectives, target audience, development methodology, and expected outcomes.

Chapter 3: Development

Outlines the tools and technologies used, describes the system architecture and data storage, and covers testing methods and results.

Chapter 4: Implementation

Provides a guide for setting up the development environment, including hardware and software requirements, installation steps, and component integration.

Chapter 5: Conclusion

Recaps the project's objectives and achievements, discusses challenges and solutions, suggests future improvements, and offers final reflections.

CHAPTER 2

PROJECT DESCRIPTION

2.1: Project Overview

The server-side application project is a job portal for students in TRNC to help them secure jobs on the island smoothly. Built on a technology stack comprising Express.js, React, Node.js and MongoDB, the application provides a comprehensive solution for managing users, recruiters, applications and job posts.

Main Functionalities and Components:

- **User Management**: The application handles user sessions and authentication, enabling users to securely log in and access their own accounts. To help keep everything simple the usernames and passwords can link with Uzebim and Genius portals since it will be held under the university
- **Applications and Job Handling**: It offers functionalities to create and manage job posts and applications. Admins can add and remove certain roles for users such as recruiters or just recruits.
- **Profile management**: This means that all user roles are given the ability to edit their profiles when in the system.
- **Allow file attachments**: The app will allow its users to share files within the application which means that users can upload resumes or links to their resume profiles.

Unique Features:

- **Integration**: The messenger app can be integrated with existing university platforms to share login credentials as seen with Uzebim and Genius portal IDs.
- **Automated Sorting**: Though a work in progress feature but several scripts are going to be implemented to help sorting the jobs in various criteria such as part-time, full-time or even internships.

- **Real time updates**: This means that the application will be able to give out real time updates once an application has been done by a recruit and he/she can see the status as pending, rejected or accepted

2.2 Objectives

The primary objectives of the project are outlined below:

• Verifying the Listed Employers to ensure that they follow the work/hour requirements, especially for students. And put a filter to help separate part-time and full-time jobs or even internship.
• English support this is to help students to access the sites with ease for now we can push English as another language instead of only Turkish as seen on most of these sites.
• Partnership with university to ensure that the students work in proper fields and trusted employers as it can help solve problems for students if anything happens.
• Diversify available opportunities as not every student would work on the campus some wish to work outside due to their own reasons.

2.3 Significance

The significance of the project lies in its ability to ensure that people on the island especially international students can get a fair chance to apply to jobs on the island almost at the same chance as the locals.

Impact on Interaction:

- **Easily accessible**: This means that if the application can be accepted by the university it can included in student email. So new students can have easy access to such job opportunities.

- **Real time Updates**: The job portal app will be able to send out notification in real time helping students to stay up to date with job opportunities. This helps quicken application processes.
- **Proper language support:** This means that the job portal application will feature proper language support compared to other alternatives here on the island

2.4 Application Features.

The application encompasses a range of features organized into specific categories, each designed to enhance functionality, security, and user experience. This detailed examination covers how these features contribute to the overall effectiveness of the application.

2.4.1 User Management

- **User Login and Functionalities**: The application supports registered user logins access. Registered users can log in using their credentials and access the application's features
- **User Management**: Sessions are managed securely, with mechanisms in place for automatic cleanup to prevent data leakage and unauthorized access. Also in terms of the groups they can added or removed accordingly.
- **Logout Processes and Data Security Measures**: The logout process ensures that user sessions are properly terminated, and any associated data is securely cleared from the session. This helps maintain user privacy and data integrity.

2.4.2 Jobs Management

- **Job Creation and Deletion**: Users with admin or recruiter access are able to create new jobs on the platform or delete the already existing ones

- **Handling of Applications for the Job posts**: The users who registered as recruiters are able to sift through the applications to get the candidates that they need for that specific job post by accepting to rejecting them.

2.4.3 Roles Management

- **This means that the admin can give or take away role privileges.** .

2.4.4 Applications History

- **Storing Application metrics:** This means that the application will be able to collect metrics about the statistics of the job applications and applicants too and together with their status.

2.4.5 Session and Data Management

- **Session Expiration and Data Cleanup**: The application implements session expiration policies and data cleanup processes to ensure that inactive sessions do not pose a security risk and that data is managed efficiently.
- **Cache Control and Static File Serving**: The application uses cache control mechanisms and serves static files efficiently, optimizing performance and reducing load times for users.

2.4.6 Frontend Integration

- **React**: The application employs this framework to hold different components of the front end all the UI elements too.

2.4.7 Security

- **Secure Session Handling and Cookie Management**: The application ensures that sessions are handled securely and cookies are managed properly to prevent unauthorized access and protect user data.
- **Management of Environment Variables**: Environment variables are managed securely to protect sensitive information such as API keys and database credentials.

2.4.8 Error Handling

- **Comprehensive Error Handling and Logging**: The application includes mechanisms for comprehensive error handling and logging. This ensures that errors are detected, logged, and addressed efficiently, enhancing the stability and reliability of the application.

2.4.9 Helper Functions

- **Unique ID Generation**: Unique IDs are generated for users to facilitate tracking and interaction requiring full user registration. This feature supports access while maintaining some level of user differentiation.

2.5 Target Audience

University Staff :
The application supports university staff to acts as administrative figure which can oversee overall running of the system and even assign roles as recruits and recruiters

Students:
Students benefit from easy access to job opportunities on the island as full-time, part-time or internships and help the gain income to sustain their lives.

Developers:
Developers can utilize the structured codebase, secure session management, and modular design for further development and customization, ensuring adaptability and maintainability.

2.6 Development Methodology

The project was developed using an **agile methodology**, which emphasizes iterative progress, flexibility, and collaboration. The development process involved breaking down the project into manageable tasks and features, allowing for continuous improvement and adaptation based on feedback.

Iterative Development: Each iteration involved planning, developing, testing, and reviewing components such as user management, job handling, and etc. This approach ensured that each feature was refined and improved incrementally.

Frameworks and Tools:

- **Node.js**: Used for building the server-side application, providing a robust and scalable foundation for handling requests and server logic.
- **React:** This component was used to build the frontend framework
- **MongoDB**: Implemented as lightweight, file-based database for managing user data, courses, and session information.
- **Express.js**: It was used to help build up the backend framework in general

2.7 Challenges and Solutions

2. **Session and Data Management:**
    - **Challenge:** Ensuring secure and efficient session management with automatic cleanup to prevent unauthorized access and data bloat.
    - **Solution:** Implemented secure session handling sessions with automatic expiration. Set up data cleanup routines to remove stale sessions and optimize database performance.

4. **Security Concerns:**
    - **Challenge:** Protecting sensitive user data and securing session management to prevent vulnerabilities.
    - **Solution:** Employed environment variables for secure configuration, managed cookies with secure flags, and used encryption for data transmission. Conducted regular security audits and implemented best practices for data protection.

5. **User Interface and Experience:**
    - **Challenge:** Designing a responsive and intuitive frontend that seamlessly integrates with the backend features.
    - **Solution:** Developed dynamic content rendering using EJS templates, ensuring compatibility across different devices and providing localized translations to enhance user experience.

6. **Error Handling and Debugging:**
    - **Challenge:** Managing comprehensive error handling and logging for effective troubleshooting.
    - **Solution:** Implemented a robust error handling mechanism with centralized logging and monitoring tools. Developed detailed error messages and logging practices to facilitate debugging and maintain application stability.

2.8 Expected Outcomes.

- **Efficient Management:** Streamlined tools for creating, retrieving, and managing job applications and applicants, including resume uploads.

- **Enhanced User Experience:** Improved interface for a more engaging user experience.
- **Secure Sessions:** Robust session management ensuring data security.
- **Optimized Data Management:** Effective data and cache management, reducing bloat and improving performance.
- **Future Enhancements:** Foundation for future improvements, including Excel parsing to allow adding bulk users at a time to help save time and effort. Also, further improvement of UI/UX for the application.

CHAPTER 3

DEVELOPMENT

This chapter outlines the development of the project, including the tools and technologies used, system architecture, data storage methods, and testing procedures. It details how the application was built and refined to ensure functionality and performance.

3.1 Tools and Technologies

- **Node.js:** Node.js is utilized for server-side scripting, allowing JavaScript to be run outside the browser. Its non-blocking, event-driven architecture ensures efficient handling of multiple simultaneous connections and real-time data processing. This is crucial for managing user sessions and interactions efficiently.
- **Frontend:** React.js was used to build up the frontend with other components to help like Vite which was a build tool and Tailwind CSS for appearance too.
- **Docker:** is a self-contained, server database engine used for storing and managing data in a lightweight, file-based format. Its simplicity and zero-configuration nature make it ideal for applications requiring a small-to-medium scale database solution, such as managing user data, course groups, and session information.
- **Tools:** VS Code
- **Backend** – MongoDB was used as the database component for the application specifically Atlas which runs remotely. APIs were used too and Authentications and Authorizations too.

3.2 System Architecture

The system architecture for the server-side application is designed to efficiently manage user applications, applicants, resumes, and metrics. The architecture follows a modular and scalable approach, leveraging a combination of Node.js, React, MongoDB, and other technologies to ensure seamless integration and performance.

Components and Interactions

1. **Client-Side Interface**

   - **Frontend:** Users interact with the system through a interface that provides functionalities for user management, job handling, resume uploads, and user interactions due to backend integration.
   - **Communication:** The frontend communicates with the backend server via to perform operations such as logging in, uploading files, and retrieving applications content.

2. **Backend Server**
   - **Node.js & Express.js:** The backend is built on Node.js and Express,js to help execute backend operations like middleware and routes with APIs too
3. **Database**
   - **MongoDB:** serves as the relational database management system, storing user data, applications and jobs. It operates in a server mode, maintaining data persistence through a file-based system.
4. **Session Management**
   - **User Sessions:** The server manages user sessions using secure session handling mechanisms. Sessions are created upon user login and are used to track user activities and preferences. Automatic cleanup is performed to ensure session security and prevent data leaks.
5. **Error Handling and Logging**
   - **Error Handling:** Comprehensive error handling is implemented throughout the backend to manage exceptions and provide meaningful error messages to users. This includes handling invalid requests, file processing errors, and database issues.
   - **Logging:** Server-side logging captures operational details and errors, aiding in debugging and monitoring system performance.

System Flow

1. **User Interaction:**
   - Users access the interface through a browser/virtual device, interacting with various features such as logging in, uploading files, and engaging by chatting in the app.
2. **Request Handling:**

- Requests from the front end are routed to the appropriate Nodejs endpoints. The server processes these requests, invoking corresponding controllers to perform the required operations.

3. **Data Processing:**
    - For this means that data is sent back and forth from database to the app and vice versa. The extracted data is then stored in the MongoDB database.
4. **Response Delivery:**
    - After processing, the server sends responses back to the frontend, updating the user interface with relevant information, confirmation messages, or error details.
5. **Session Management:**
    - Sessions are managed throughout user interactions, ensuring that user data and activities are securely tracked and maintained.

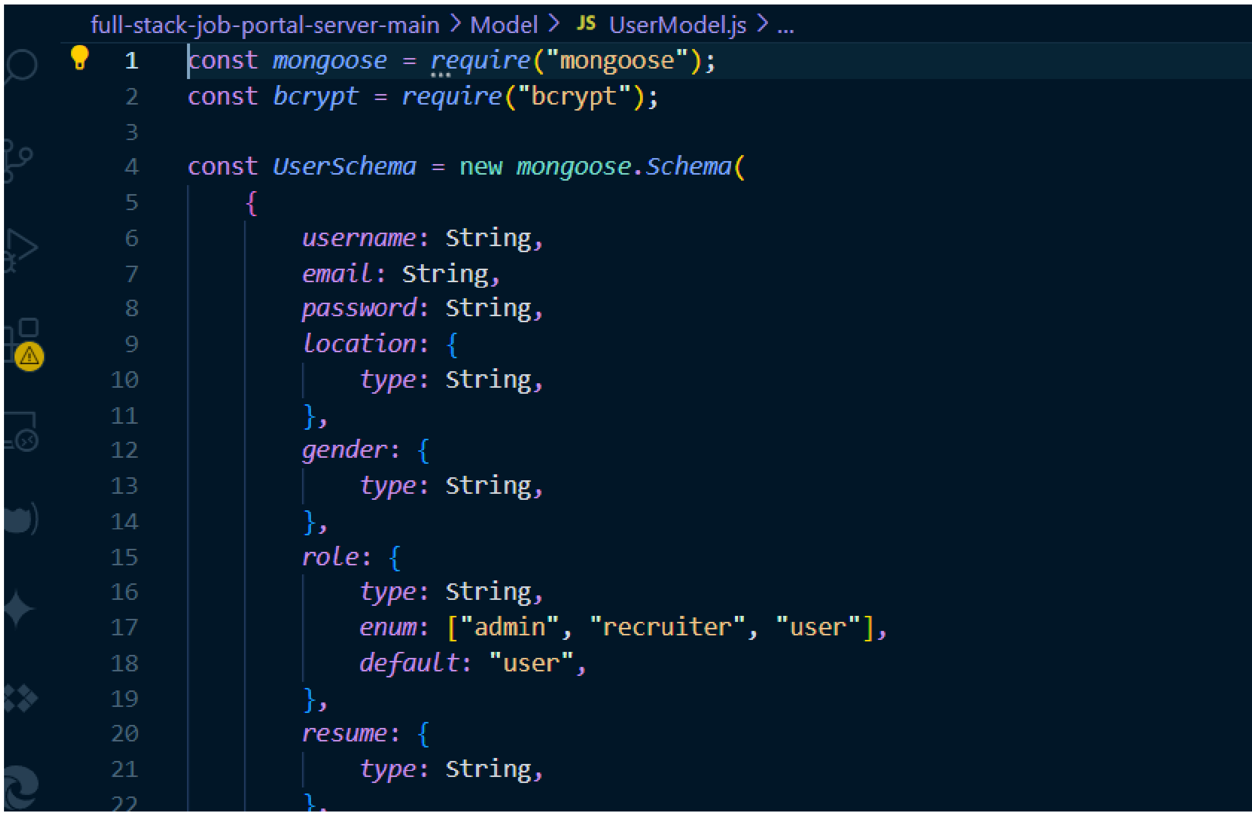
3.3 Data Storage

The data storage solution for the project relies on MongoDB (Atlas), a lightweight and serverless relational database management system. MongoDB is chosen for its simplicity, ease of integration, and sufficient capability for handling the data requirements of this application. This section outlines the database schema, data models, and the management and retrieval of data within the application.

Database Schema

The database schema is designed to manage various entities such as users, courses, chapters, and chat history. Below is a description of the key tables in the MongoDB database:

1. **Users Table**
    - **Schema:**


full-stack-job-portal-server-main > Model > JS UserModel.js > ...

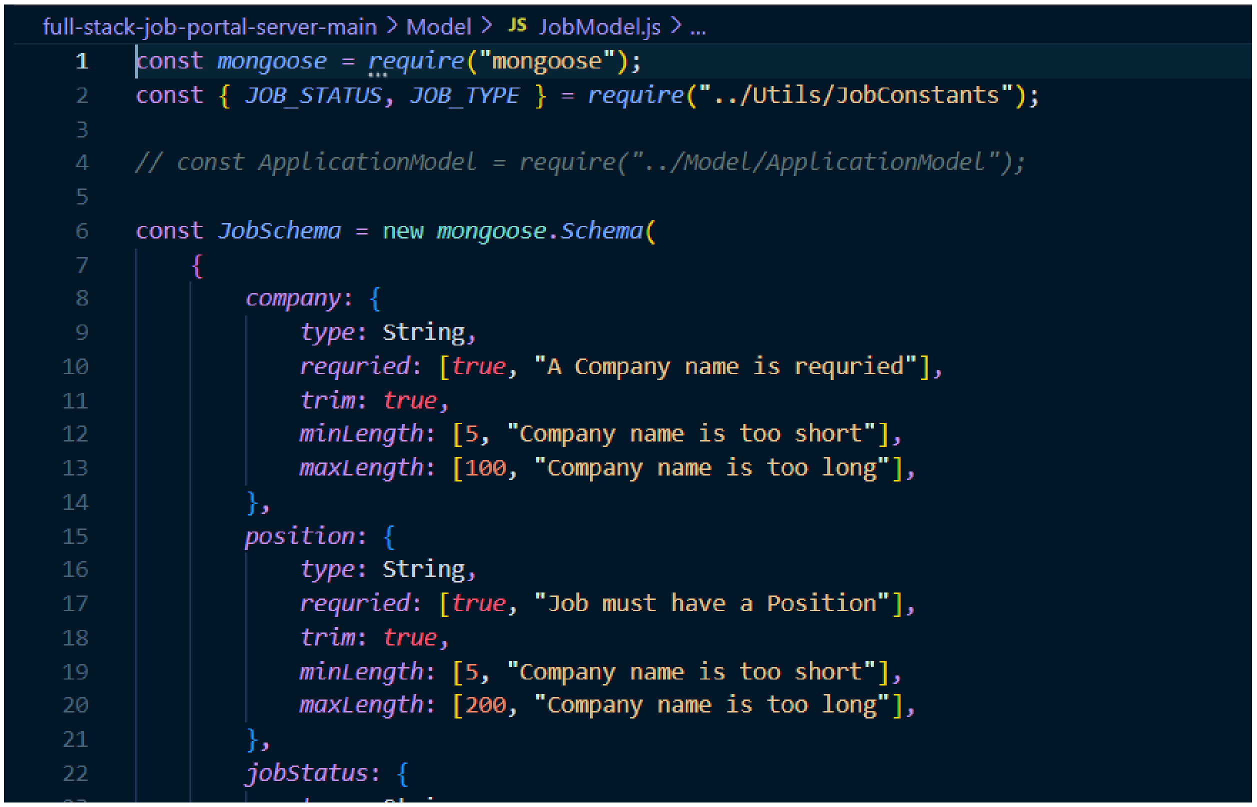
```
const mongoose = require("mongoose");
const bcrypt = require("bcrypt");

const UserSchema = new mongoose.Schema(
    {
        username: String,
        email: String,
        password: String,
        location: {
            type: String,
        },
        gender: {
            type: String,
        },
        role: {
            type: String,
            enum: ["admin", "recruiter", "user"],
            default: "user",
        },
        resume: {
            type: String,
```

Figure 1 – Creating Users Table which Stores user information including a unique identifier.

Jobs Table

- **Schema:**



full-stack-job-portal-server-main > Model > JS JobModel.js > ...

```
const mongoose = require("mongoose");
const { JOB_STATUS, JOB_TYPE } = require("../Utils/JobConstants");

// const ApplicationModel = require("../Model/ApplicationModel");

const JobSchema = new mongoose.Schema(
    {
        company: {
            type: String,
            requried: [true, "A Company name is requried"],
            trim: true,
            minLength: [5, "Company name is too short"],
            maxLength: [100, "Company name is too long"],
        },
        position: {
            type: String,
            requried: [true, "Job must have a Position"],
            trim: true,
            minLength: [5, "Company name is too short"],
            maxLength: [200, "Company name is too long"],
        },
        jobStatus: {
```

Figure 2 –This is responsible for making new jobs.

Data Management and Retrieval

1. **Data Insertion:**
   - **Users:** When a new user registers or logs in, their information is inserted or updated in the users table. Passwords are hashed before storage.
   - **Applications:** New applications are added through administrative interfaces, inserting records into the table.
   - **Jobs:** Jobs that are added by recruiters
   - **Resumes:** User can add their resumes as file or links

2. **Data Retrieval:**
   - **User Data:** Retrieved during login and session validation to verify credentials and manage sessions.
   - **Applications:** Retrieved for displaying application and applicants details to users.
   - **Metrics:** Retrieved for displaying metrics such as applications and their statuses as applications and their statuses

3. **Data Updates:**
    - **User Sessions:** Session tokens are updated during user login and logout processes.

4. **Data Deletion:**
    - **Cascade Deletion:** When users or applications are deleted, all associated applications are automatically removed due to the cascading foreign key constraint. So data can be pruned based on user actions or administrative policies.

Integration with Application

The application uses MongoDB for seamless data management and retrieval. Node.js with MongoDB integration libraries is employed to execute Database queries, manage connections, and perform CRUD (Create, Read, Update, and Delete) operations. The use of database queries ensures efficient data handling and retrieval, supporting the application's functionality.

3.4 System Testing and Evaluation

System testing and evaluation are crucial for verifying that the application functions correctly, meets user requirements, and is free from critical issues. This section outlines the testing methodologies used, including unit testing, integration testing, and user acceptance testing, along with the specific tools and frameworks employed in the project.

Testing Methodologies

1. **Unit Testing**
    - **Description:** Unit testing involves testing individual functions and modules in isolation to ensure they work as intended. This helps in early detection of bugs and verification of component behavior.
2. **Integration Testing**
    - **Description:** Integration testing assesses the interaction between different components of the application. This ensures that combined parts work together seamlessly and that data flows correctly across the system.
3. **User Acceptance Testing (UAT)**

- **Description:** UAT involves real users interacting with the application to ensure it meets their needs and is user-friendly. This phase verifies that the application aligns with user expectations and requirements.

Testing Process

1. **Test Planning**
   - **Description:** Define the testing scope, including features to be tested, test cases, and success criteria. Develop a comprehensive test plan outlining objectives, resources, and timelines.
2. **Test Case Development**
   - **Description:** Create detailed test cases covering various scenarios, including positive and negative tests. Ensure that test cases thoroughly check all functionalities of the application.
3. **Test Execution**
   - **Description:** Run the test cases as per the test plan, recording results and identifying any issues or discrepancies. Automated tests help in frequent checks, while manual tests are performed during development.
4. **Defect Reporting and Fixing**
   - **Description:** Report any defects or issues discovered during testing. The development team investigates, addresses, and re-tests the issues to ensure resolution.
5. **Test Evaluation**
   - **Description:** Evaluate testing results to determine if the application is ready for release. Assess whether critical issues are resolved, performance standards are met, and user feedback is integrated.

Evaluation Metrics

- **Test Coverage:** Measures how much of the application code is covered by tests, ensuring thorough testing.
- **Bug Rate:** Tracks the frequency and severity of bugs identified during testing.

- **User Feedback:** Collects input from users during UAT to evaluate usability and functionality.

3.5 Results and Analysis.

The results and analysis section provides a detailed examination of the outcomes from the testing phase. It includes a summary of issues discovered, their resolutions, and an assessment of the application's overall performance based on the metrics collected during testing.

Testing Outcomes

1. **Unit Testing Results**
    - **Issues Found:** Initial unit testing revealed several issues related to function logic, such as incorrect handling of edge cases and validation errors in API endpoints.
    - **Resolutions:** The identified issues were addressed by refining function implementations and correcting validation logic. Additional test cases were created to cover previously untested edge cases, ensuring comprehensive coverage.
2. **Integration Testing Results**
    - **Issues Found:** Integration testing highlighted problems with data flow between components, particularly with session management and processing. Some issues also surfaced in the interaction between the application storage.
    - **Resolutions:** The issues were resolved by adjusting the data handling mechanisms and improving error handling in the integration points. Mocking and stubbing were used to isolate problematic interactions during testing.
3. **User Acceptance Testing (UAT) Results**
    - **Issues Found:** UAT feedback indicated usability issues, including difficulties in navigating the course management interface and inconsistencies in multi-language support.

- **Resolutions:** The user interface was refined based on feedback, improving navigation and usability. Localization was enhanced to ensure consistent and accurate language support across different regions.

Performance Analysis

1. **Application Performance**
   - **Metrics Collected:** Metrics such as response times for application load times, and memory usage were monitored during testing.

   - **Results:** Performance testing showed that the application meets performance benchmarks with average response and acceptable load times. Memory usage was optimized to handle high loads effectively.

2. **Error Rates and Stability**
   - **Metrics Collected:** Error rates, including failed requests and system crashes, were tracked.

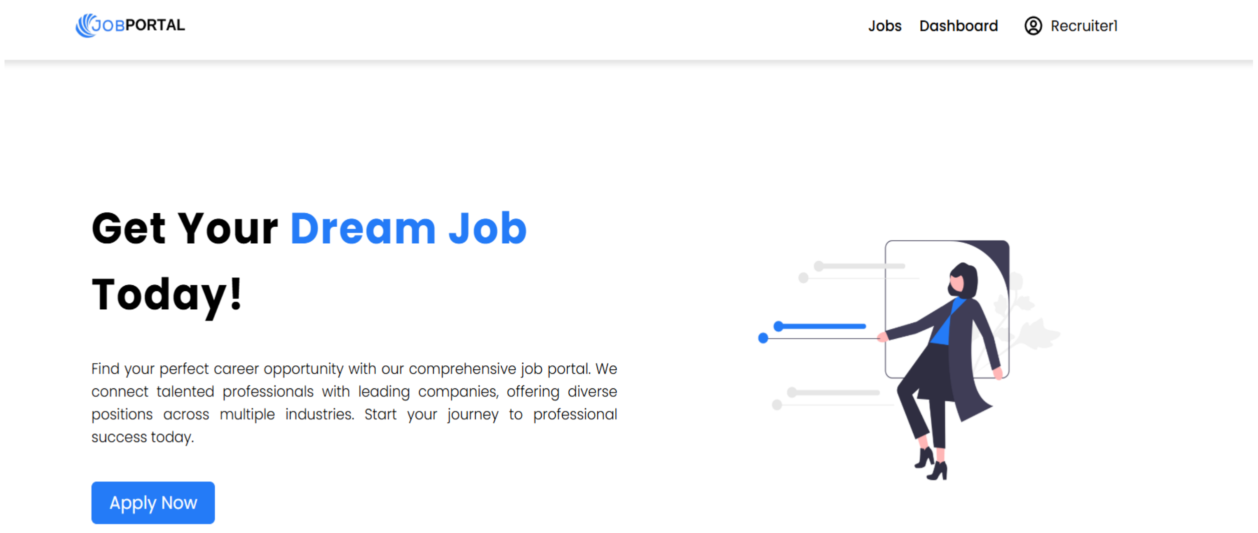
   - **Results:** Error rates were low, with less than 1% of requests resulting in errors. Stability tests confirmed that the application remained operational under stress conditions, with no significant crashes or failures.

3. **User Feedback**
   - **Metrics Collected:** User satisfaction scores and feedback from UAT were analyzed.

   - **Results:** Users reported a high level of satisfaction with the application's functionality and ease of use. Feedback indicated positive responses to the integration and overall system performance, though minor improvements in UI were suggested.

3.6 Visual Representations

- **Home Page:** User home page.

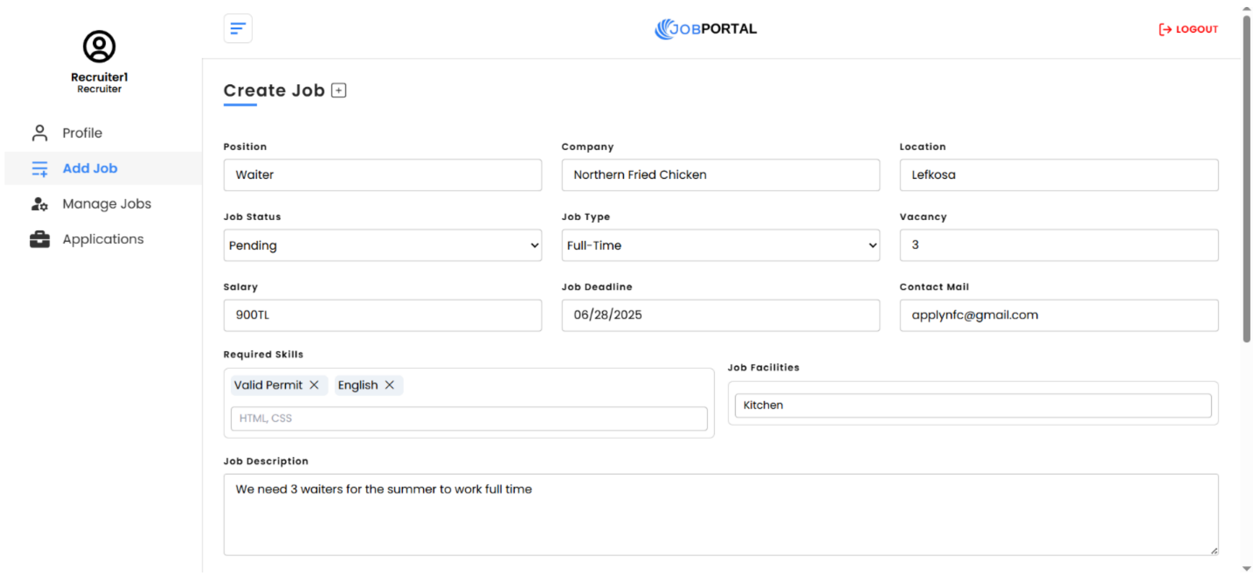


Get Your Dream Job Today!

Find your perfect career opportunity with our comprehensive job portal. We connect talented professionals with leading companies, offering diverse positions across multiple industries. Start your journey to professional success today.

Figure 3 – Showing the user home page

- **Add job page:** Where recruits can add new jobs themselves.



Create Job

Position	Company	Location
Waiter	Northern Fried Chicken	Lefkosa
Job Status	Job Type	Vacancy
Pending	Full-Time	3
Salary	Job Deadline	Contact Mail
900TL	06/28/2025	applynfc@gmail.com

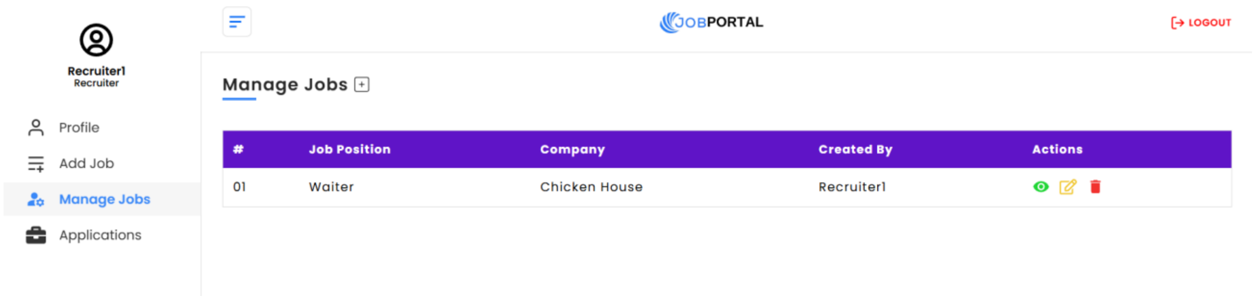
Required Skills: Valid Permit, English (HTML, CSS)

Job Facilities: Kitchen

Job Description: We need 3 waiters for the summer to work full time

Figure 4 – Showing the Add job page

- **Manage jobs page:** Showing how jobs are managed by recruiter

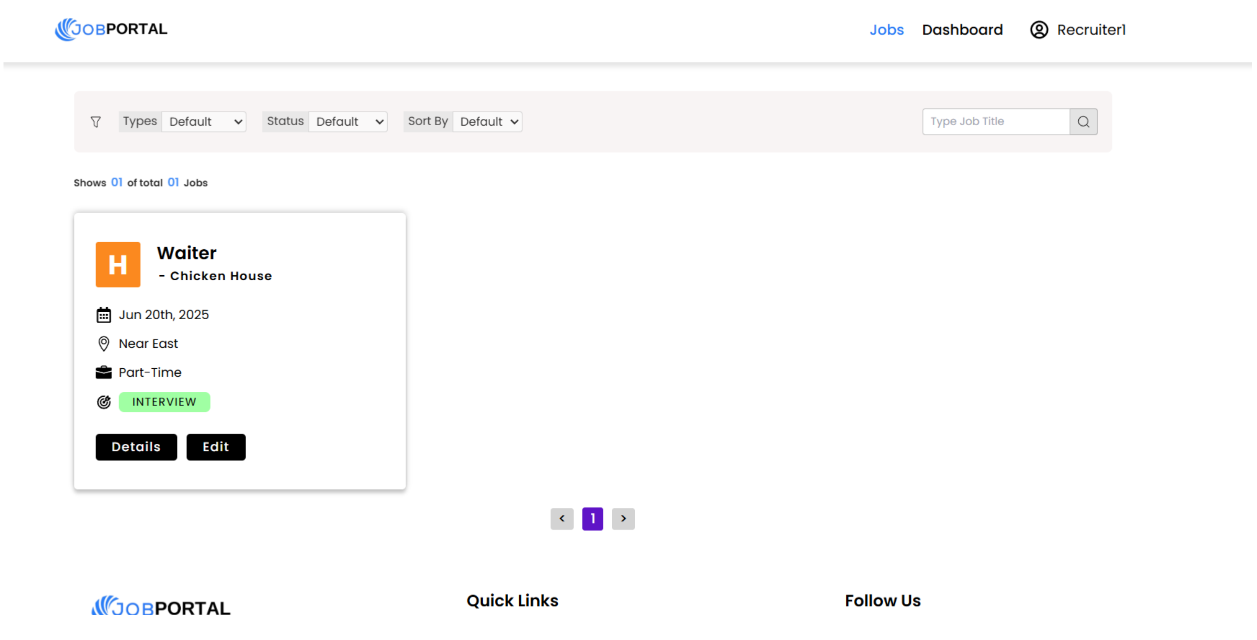


Manage Jobs

#	Job Position	Company	Created By	Actions
01	Waiter	Chicken House	Recruiter1	

Figure 5 – A page for managing jobs

- **Jobs page:** Where jobs can be seen by users



Shows 01 of total 01 Jobs

Waiter
- Chicken House
Jun 20th, 2025
Near East
Part-Time
INTERVIEW

Figure 6 – Showing an available job in the system.

CHAPTER 4

IMPLEMENTATION

4.1 Setting Up the Development Environment

This section categorizes the setup process into hardware and software requirements, libraries and dependencies, and installation procedures to streamline the development environment configuration.

Hardware and Software Requirements

1. **Node.js and npm**
   - **Description:** Node.js is required to run the server-side application, and npm (Node Package Manager) is included for managing dependencies.
   - **Installation:** Download and install the latest LTS version of Node.js from the official Node.js website, which includes npm.

2. **MongoDB**
   - **Description:** MongoDB is database technology used for data storage. Ensure that it's installed for managing the database.
   - **Installation:** Download and install MongoDB from the official MongoDB website.

3. **Git (Optional)**
   - **Description:** Git is used for version control and collaboration. It is optional but recommended for managing the project codebase.
   - **Installation:** Download and install Git from the official Git website.

4. **Text Editor or IDE**
   - **Description:** A text editor or Integrated Development Environment (IDE) is needed for coding and debugging.
   - **Recommendations:** VSCode, Sublime Text, Atom, or any preferred IDE.

5. **Web Browser/Virtual Device**
   - **Description:** A web browser is required to access and interact with the application during development or even a virtual device to emulate a mobile device.
   - **Recommendations:** Chrome, Firefox, or Edge.

Libraries and Dependencies

1. **Project Dependencies**
    a. **Description:** The project relies on various Node.js libraries and packages to function correctly.
    b. **Installation:** Use npm to install all required packages specified in *package.json*.
        i. `"npm install"`

2. **Key Libraries**
    a. **MongoDB:** Mongoose Library for interfacing with MongoDB databases.

Installation

1. **Install Node.js and npm**

    a. **Instructions:**
        i. Download and run the installer from the official Node.js website.
        ii. Verify installation:
            a. `"node -v"`
            b. `"npm -v"`

2. **Clone the Project Repository**

- **Instructions:**
    - Clone the repository from version control:

      ```
      "Git clone https://github.com/yourusername/yourproject.git"
      ```

    - Navigate to the project directory:

      ```
      "Cd yourproject"
      ```

3. **Install Project Dependencies**

- **Instructions:**
    - Run npm to install all dependencies listed in package.json:

      ```
      "Npm install"
      ```

4. **Set up MongoDB Database**

- **Instructions:**
    - Ensure MongoDB is installed and its drivers are installed.

6. **Start the Development Server**

- **Instructions:**
    - Launch the server using npm:

      ```
      "Npm start"
      ```

    - Open a web browser and navigate to *http://localhost:5432* (or the port specified in your configuration).

7. **Access the Application**

- **Instructions:**
    - Proceed to run the application through a web browser.

4.2 Integrating System Components.

This section outlines the integration of various system components, including backend, frontend, database, and to ensure a unified and functional application.

1. Backend and Frontend Integration

I. **Nodejs Server Setup**
  - Configure to serve static files and EJS templates by updating `server.js` or `app.`

3. **Connecting the Database**

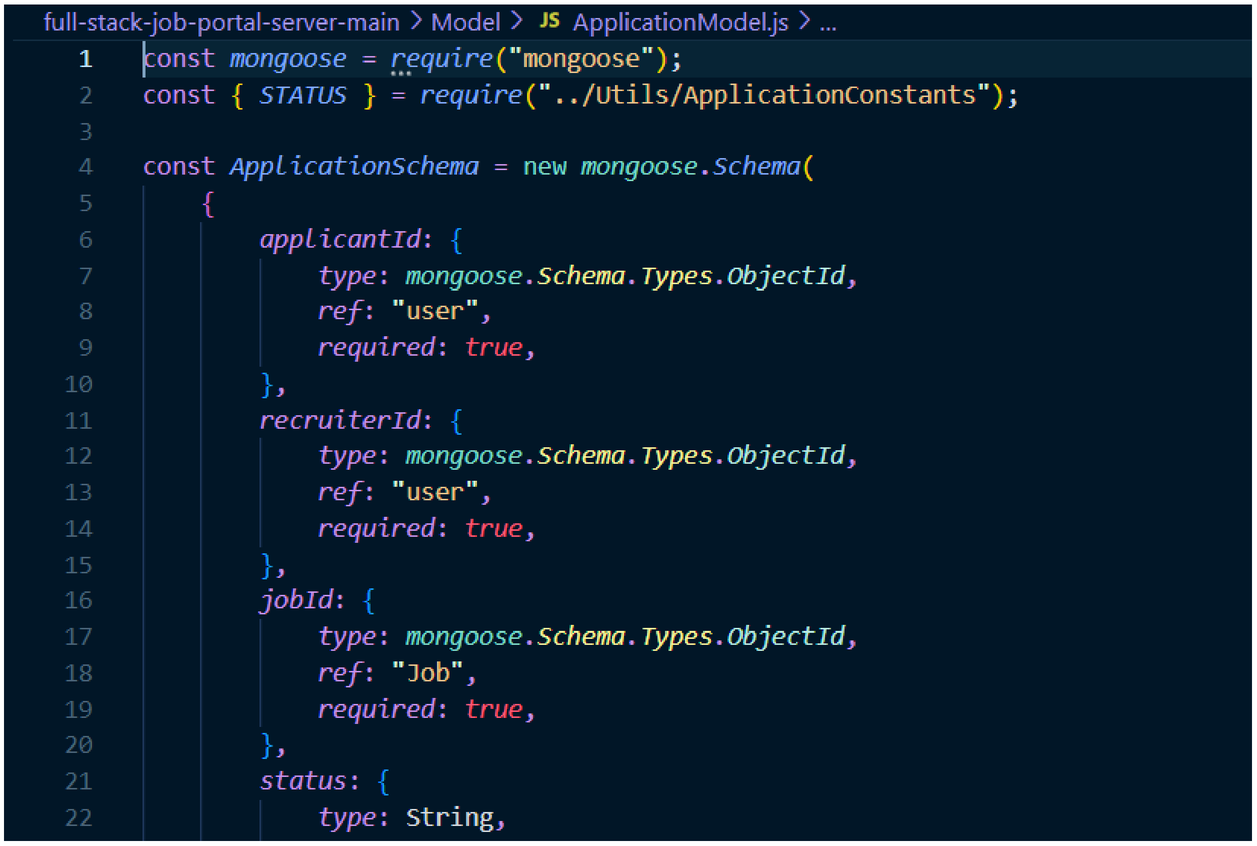
I. **Database Configuration**
   - Reference the MongoDb database in `config.js` or. env:

II. **Database Initialization**
   - Execute migration scripts to set up the schema and tables:

```
npm run migrate
```

III. **Database Interaction**
   - Perform queries using the `Mongoose` library:

IV. **Backend-Database Integration**
   - Link database queries to backend routes


full-stack-job-portal-server-main > Model > JS ApplicationModel.js > ...

```javascript
const mongoose = require("mongoose");
const { STATUS } = require("../Utils/ApplicationConstants");

const ApplicationSchema = new mongoose.Schema(
    {
        applicantId: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "user",
            required: true,
        },
        recruiterId: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "user",
            required: true,
        },
        jobId: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "Job",
            required: true,
        },
        status: {
            type: String,
```

Figure 7 – Showing Applications on Database

CHAPTER 5
CONCLUSION

5.1 Summary of the Project

This project developed a server-side application using Node.js, React, and MongoDB to handle user sessions, applications, job posts, and metrics. Key features include secure user management, session handling with unique users, jobs and applications with real time updates thanks to database helping pushing real-time updates.

5.2 Key Findings and Achievements

Major achievements include:

- **Secure User Management**: Effective session handling and login.
- **Functionality**: Successful user and applications creation for use.
- **Responsive Frontend**: Efficient use database for real-time updates.

5.3 Challenges and Solutions

Challenges and their solutions include:

- **Database Reliability:** MongoDB(Atlas) has to be run with internet to ensure it runs smoothly.
- **Session Security**: Applied best practices for secure session and management.

5.4 Future Work and Enhancements

Future enhancements may involve:

- **Performance Optimization**: Caching and query optimization.
- **Feature Expansion**: advanced search and video/audio calls.
- **UI/UX Improvements**: Modern design and interactive elements.
- **Scalability**: Exploring database optimization and scaling solutions.

5.5 Final Thoughts

The project highlighted effective application development practices and technology integration. Key lessons include the value of planning, testing, and iterative development. The application's success in managing user applications and user information underscores its potential impact and effectiveness in modern university job environment.

REFERENCES

Node.js Documentation
Node.js Foundation. (n.d.). *Node.js Documentation*. Retrieved from *https://nodejs.org/api/*

React.js Documentation

Meta. (n.d.) *React.js Documentation*. Retrieved from *https://reactjs.org/docs/getting-started.html*

Vite Documentation

Vite Documentation Retrieved from *https://www.vitejs.dev/guide/*

Tailwind CSS Documentation

Tailwind Labs (n.d.). *Tailwind CSS Documentation*. Retrieved from *https://www.tailwindcss.com/docs*

Express.js Documentation

OpenJS Foundation (n.d.). *Express.js Documentation*. Retrieved from *https://www.expressjs.com/en/starter/installing.html*

MongoDB Documentation

MongoDB, Inc (n.d.). *MongoDB Documentation*. Retrieved from *https://www.mongodb.com/docs/*