JOB RECOMMENDER SYSTEM

KHALID ALI FARAH

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

JOB RECOMMENDER SYSTEM

KHALID ALI FARAH

This report is submitted in partial fulfillment of the requirements for the Bachelor of Computer Science Software Development with Honours.

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2023

DECLARATION

I hereby declare that this project report entitled

JOB RECOMMENDER SYSTEM

is written by me and is my own effort and that no part has been plagiarized without citations.

STUDENT : ________________________________ Date : 19 September 2023
(KHALID ALI FARAH)

I hereby declare that I have read this project report and found this project report is sufficient in term of the scope and quality for the award of Bachelor of [Computer Science (Software Development)] with Honours.

SUPERVISOR : ermahani Date : 20/9/2023
(DR. INTAN ERMAHANI BINTI A. JALIL)

DECLARATION

DEDICATION

I dedicate this final year project to my cherished parents, who have steadfastly accompanied me throughout my academic journey, offering unchanging love and motivation. I extend my dedication to my family members, whose enduring support and belief in me have been a pivotal force propelling me towards knowledge. I would also like to pay tribute to Engineer Najib Ba Alawi, my role model, a consistent pillar of support and a person of remarkable character. Your guidance and unwavering positivity have played a vital role in my pursuit.

To each of you, this project is a token of my appreciation, for it is your unwavering faith in me that has brought me to this stage. Your love and encouragement have ignited my determination, and I aspire to achieve results that will make you all proud."

ACKNOWLEDGEMENTS

I extend my heartfelt gratitude to Dr. Intan Ermahani Binti A. Jalil, my esteemed supervisor, for her invaluable guidance and support that were instrumental in the successful completion of this project. Furthermore, I would like to express my profound appreciation to my beloved parents, whose unwavering support and unwavering motivation served as the driving force behind my project. Their encouragement and belief in my abilities have been a constant source of inspiration. My sincere thanks go out to all those who have been part of this journey, contributing to its success in various ways.

ABSTRACT

The "Job Recommender System" is a web-based platform bridging freelancers, including self-employed professionals and students seeking practical experience, with businesses looking to outsource projects. Freelancers can create profiles showcasing their skills, contact details, work samples, and social media links, which are accessible to registered businesses. These freelancers can explore other profiles and job listings, express their interest in positions, and initiate communication. Businesses, in turn, can register, post job openings, review freelancer profiles, manage interested candidates, and receive curated recommendations for freelancers with the exact skills required for specific projects. Administrators oversee the system, accessing all user information, including personal details, job history, and ratings. Overall, the "Job Recommender System" streamlines talent matching, supports professional growth, and simplifies project outsourcing, benefiting both freelancers and businesses in a dynamic and evolving job market.

TABLE OF CONTENTS

Table of Contents

LIST OF TABLES

PAGE

LIST OF FIGURES

PAGE

LIST OF ABBREVIATIONS

MERN:

MongoDB	**-**	**a NoSQL database management system.**
Express.js	**-**	**Server Framework.**
React	**-**	**Front-End Library.**
Node.js	**-**	**Runtime Environment.**

CHAPTER 1: INTRODUCTION

1.1 Introduction

Businesses often struggle to find the right freelancers for their projects, while freelancers may have difficulty finding suitable job opportunities. This web application aims to bridge that gap by providing a platform that recommends suitable job opportunities to freelancers and allows businesses to find freelancers based on their requirements. Furthermore, the application also aims to provide students with opportunities to work on real-time projects and gain experience. By addressing these needs, the " Job Recommender System “web application can serve as a useful tool for businesses, freelancers, and students alike.

1.2 Problem statement(s)

The lack of a centralized platform for businesses to find suitable freelancers and for freelancers to find job opportunities can be a significant challenge for both parties. This problem can be addressed by developing a web application that matches freelancers with suitable job opportunities based on their skills and experience. Additionally, the lack of opportunities for students to work on real-time projects and gain practical experience can be a major problem that can be addressed by the application. By providing students with access to real-time projects, they can gain valuable experience and enhance their skills, making them more employable in the future.

1.3 Objective

- To provide a platform for freelancers to showcase their skills and connect with businesses that require their services.

- To help businesses find the right freelancers for their projects based on their skill sets.

- To enable freelancers to view and apply for job postings that match their skill sets.

- To facilitate communication between freelancers and businesses.

- To allow freelancers to update their profile information as they acquire new skills and experiences.

1.4 Scope

1.4.1 Homepage

The homepage component is an essential part of an application's user interface, as it is the first page that users interact with when they access the application. The landing page components are the various parts that make up the homepage, including the NavBar, which is responsible for displaying the navigation bar interface. The NavBar typically includes links or buttons that allow users to navigate to different sections of the application.

1.4.2 Signup Page

The signup component is responsible for defining the user interface of an application's sign-up page, permits new users to sign up and access the applications features. Once user has been authenticated and logged in, they will be prompted to create a profile in order to provide more information about themselves or their business. The form of the signup page depending on the type of user accessing the application. For example, if the application caters to both freelancers and businesses, the signup page may appear differently for each type of user. The signup component should be designed in such a way that it accommodates the specific needs of each type of user, with clear and intuitive instructions and feedback messages. The signup component may include various elements, such as text fields for entering personal or business information, dropdown menus for selecting categories or industries, and buttons for submitting the information and completing the sign-up process. The user interface design of the signup component should be clear and easy to use, with intuitive instructions to guide users through the sign-up process.

1.4.3 Login Page

The login component is responsible for defining the user interface of an application's login page, which allows users to authenticate and access to application features. To gain access to the application, users must provide their email address and password through the login page. The appearance of the login page may vary depending on the type of user accessing the application. For example, if the application caters to both freelancers and businesses, the login page may appear differently for each type of user. The login component should be designed in such a way that it accommodates the specific needs of each type of user, with clear and intuitive instructions.

1.4.4 Create Profile Page

Create profile component is responsible for defining the user interface of an application's profile creation page. This page allows users to create and manage their profiles, providing more information about themselves or their business to other users on the application. Create profile component may include various input fields, such as text fields, dropdown menus, checkboxes, and buttons, for entering and submitting profile information. The appearance and content of these input fields may depend on the type of user accessing the application. For example, if the application caters to both freelancers and businesses, create profile component may have different input fields for each type of user. For a freelancer, the input fields may include information about their skills, experience, education, and portfolio, while for a business, the input fields may include information about the company's name, edit profile, view freelancers, and post Jobs .

1.4.5 Freelancer's Dashboard

The freelancer dashboard component is responsible for creating the interface of the freelancer's dashboard. It includes various features and tools that help freelancers. These features may include editing profiles, adding education and experience, viewing available jobs, and browsing other freelancers' profiles.

1.4.6 Edit Profile Page

Profiling editing is the functionality in a freelancer dashboard component that allows freelancers to modify or update their personal and professional information. This may include changing their profile picture, updating their skills or work experience, setting their hourly or project rate, and adding a bio or description of their services. Editing profiles is an essential feature for freelancers to keep their information up-to-date and attract potential clients.

1.4.7 Add Education/Experience

Freelancers have the option to add education or experience details to their profile based on their background and work history. If a freelancer is a student or has recently graduated, adding education details can help demonstrate their academic qualifications and expertise in a particular field. On the other hand, if a freelancer is self-employed or has worked for other companies in the past, adding experience details can help highlight their work history, skills, and accomplishments. The ability to add education and a useful component in a freelancer dashboard is experience details., as it allows freelancers to showcase their strengths and attract potential clients.

1.4.8 View Jobs

A feature in the freelancer dashboard component called "viewing available jobs" enables freelancers to search and apply for job posts or projects that are currently open on the platform. This feature typically displays a list of jobs that are currently open or available for application. Freelancers can filter the job postings based on various criteria such as job type, category, budget, location, or deadline. By viewing available jobs, freelancers can identify opportunities that match their skills and interests. This feature is important for freelancers to find work and build their portfolio, as well as for clients to find qualified professionals for their projects.

1.4.9 View other freelancers

Browsing other freelancers' profiles is another feature of a freelancer dashboard component that allows freelancers to explore the profiles of other professionals on the platform. This feature enables freelancers to view other freelancers' profiles, portfolios, and work history, and learn from their experiences and skills. This feature is valuable for freelancers to build their network, gain insights into their field, and potentially find new opportunities for work.

1.4.10 Business Dashboard

The Business Dashboard component defines the interface of the business dashboard, and typically includes several functionalities to help businesses manage their work and interactions with freelancers. These functionalities may include.

1.4.11 Edit Profile

This feature enables businesses to modify or update their business information, such as the company name, a description, and contact information. By editing their profile, businesses can showcase their brand and attract potential freelancers to their platform.

1.4.12 View Freelancers

This feature allows businesses to browse freelancers' profiles. By viewing freelancers' profiles, businesses can identify potential candidates for their projects and evaluate their qualifications and expertise.

1.4.13 Post Jobs

This feature enables businesses to create and publish job postings on the platform, detailing the requirements, responsibilities. By posting jobs, businesses can attract freelancers and find the best match for their needs.

1.4.14 My Jobs

This feature displays the jobs that the business has posted, as well as the status and progress of each job. By tracking their jobs, businesses can manage their projects more effectively, communicate with freelancers, and ensure timely delivery of the work.

1.4.15 Recommendations

To provide recommendations for suitable freelancers, the platform uses a content-based filtering approach. This means that the recommendations offered by the platform are determined by considering the contents of the user's needs and specifications. Content based filtering involves analyzing the contents of the job requirements and comparing them with the skill sets of available freelancers. The platform uses algorithms to identify relevant skills and experience required for the job and matches them with skills and experience of available freelancers. By using this approach, the platform can recommend freelancers to businesses based on the freelancer's skill set. The platform can ensure that businesses are connected with freelancers who are qualified to do the work, both in terms of experience and skills.

1.4.16 Admin

Administrator, has access to view all the details related to freelancers and businesses. This includes information such as personal and contact details, job history, and ratings and reviews. Additionally, the administrator has the authority to remove jobs, freelancer profiles, and business profiles if necessary. For instance, if a job posting violates the platform's policies, the administrator can remove the job posting. Similarly, if a freelancer or business profile is found to be fraudulent or in violation of the platform's terms and conditions, the administrator can remove the respective profile.

1.5 Project Significance

The "Job Recommender System" project can benefit both freelancers and businesses. Freelancers can benefit from the platform as it provides them with a platform to showcase their skills. They can view and apply for job postings that match their skill sets, update their profile information, and communicate with businesses to discuss project requirements. This can help them to find more job opportunities, grow their professional network, and increase their chances of getting hired for projects that align with their skills and interests. On the other hand, businesses can benefit from the project as it allows them to find the right freelancers for their projects based on their skill sets. This can help them to save time and effort in searching for qualified candidates and increase the efficiency and effectiveness of their outsourcing process.

1.6 Expected Output

The Job Recommender System will have a user-friendly platform for freelancers and businesses to connect and find suitable job opportunities or skilled freelancers. It will have a search and matching algorithm, messaging or communication feature, a way for freelancers to showcase their work, robust security and privacy features. The system will aim to make the process of finding and connecting with suitable freelancers or job opportunities as efficient and effective as possible for both parties involved.

1.7 Conclusion

The project is introduced in chapter one, which also discusses the project's context, problem statement, objective, scope, significance, and expected outcomes. The problem statement describes the issues that directly impact the project's goals, and the objective defines the desired outcomes in a point form format. The scope outlines the project's boundaries, including any specific entities or platforms involved, and the project significance explains who or what stands to benefit from the project and how. Lastly, the expected output summarizes what is expected from the project as a whole.

CHAPTER 2: LITERATURE REVIEW AND PROJECT METHODOLOGY

2.1 Introduction

This chapter of the project report covers the literature review and project methodology. It explores existing systems, research, and case studies in the job recommender system domain. The chapter discusses the selected techniques and justifies the exclusion of other approaches. It also describes the project methodology, activities in each stage, requirements, schedule, and milestones. This chapter establishes a solid foundation based on established practices and research for the subsequent chapters and the overall project.

2.2 Facts and findings

2.2.1 Domain

The domain related to this project is the online job marketplace. It is a platform that connects businesses or organizations seeking specific skills and expertise with freelancers who possess those skills and are looking for work. The platform enables businesses to post job opportunities, and freelancers to create a profile showcasing their skills, experience, and work examples. The system then matches the most suitable freelancers with the job requirements of businesses. The admin has full control over the platform and can view all information related to freelancers and businesses. This project aims to facilitate the process of finding suitable freelancers for businesses and provides a more streamlined and efficient approach to recruitment.

2.2.2 Existing System

2.2.2.1 Research carried out to find the best stack a combination of technologies-for creating applications:

A collection of several technologies used in the creation of an application is referred to as a "tech stack." It includes databases, APIs, back-end and front-end tools, frameworks, programming languages, and more. Making decisions about a tech stack can have a big impact on things like the kinds of integrations that can be developed and the hiring requirements.

Nowadays, a variety of common stacks are used for application development, and the choice is frequently influenced by the particular requirements of the application. Here are a few illustrations. MongoDB, Express, React, and Node.js comprise the MERN stack. For creating full-stack web applications, this stack is common. Linux, Apache, MySQL, and PHP, or LAMP Stack. This stack is popular for creating web apps.particularly those that are built on the LAMP (Linux, Apache, MySQL, and PHP) architecture. MongoDB, Express, Vue, and Node make up the MEVN Stack. Another well-liked stack for creating full-stack online apps, but this time using Vue.js rather than React.js.

Ruby on Rails Stack: Ruby, Rails, and PostgreSQL. This stack is similar to the Django stack, but uses the Ruby programming language and Rails framework. There are many other stacks that can be used for application development, depending on the requirements of the application, including stacks based on Java, Go, and other programming languages. Nikulchev, E., Ilin, D., & Gusev, A. (2021). Technology stack selection model for software design of digital platforms. Mathematics, 9(4), 308.

After evaluating the technical requirements for each stack in this project, I have decided to utilize the MERN stack. MERN stack is a popular JavaScript is a popular choice for web development because it enables the creation of strong and scalable online applications on both the client and server sides. It also includes a NoSQL database (MongoDB), which provides flexibility and scalability for handling large amounts of data. Mehra, M., Kumar, M., Maurya, A., & Sharma, C. (2021). Mern stack web development. Annals of the Romanian Society for Cell Biology, 25(6), 11756-11761.

2.2.2.2 Researching similar web applications:

Researching similar web applications, it's evident that many platforms designed for freelancers offer valuable features but often come with the drawback of requiring freelancers to engage in competitive bidding for projects, leading to potential underpricing. Additionally, there's a prevailing emphasis on team-based registrations, which may limit opportunities for individual freelancers who prefer to work independently and showcase their unique skills. For instance, platforms like Toptal often emphasize team-based collaboration, while websites such as Guru and 99designs also require freelancers to participate in competitive bidding processes. To stand out, a "Job Recommender System" could differentiate itself by prioritizing individual freelancers, offering personalized job recommendations, and fostering transparent pricing negotiations to create a more freelancer-friendly environment, ultimately attracting and retaining a diverse pool of skilled talent while benefiting businesses seeking freelancers."

2.2.2.3 Conducting research on the implementation of a recommendation engine:

The goal of a recommendation system is to present the user with the most relevant item or content. However, the development of a recommendation system poses a challenge in terms of accurately understanding the user's preferences and needs, and providing tailored suggestions based on those requirements. Building a recommendation system that can offer relevant recommendations in real-time is a challenging task. Two common approaches to constructing a recommendation engine are content-based recommendation and collaborative filtering.

Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5

Collaborative filtering:

Generating suggestions for users based on their historical conduct and interests, recommendation systems frequently use a process called collaborative filtering. The primary tenet of collaborative filtering is to identify people who share a user's preferences and then suggest goods that these users have previously appreciated. User-based and item-based collaborative filtering are the two main forms. A user's comparable users are found via user-based collaborative filtering, which also proposes things that these comparable users have previously liked. Conversely, item-based collaborative filtering searches for products that a particular user has previously enjoyed and proposes these relevant products to the user. Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5

Goldberg, K., Roeder, T., Gupta, D. et al. Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval 4, 133–151 (2001). https://doi.org/10.1023/A:1011419012209 Both approaches have their strengths and weaknesses. The implementation of user-based collaborative filtering is straightforward and can offer helpful suggestions to new users who haven't yet given many items a rating.. However, it can be less effective for users with unique tastes or preferences. Item-based collaborative filtering, on the other hand, can be more effective for users with unique preferences, but may require more computational resources to generate recommendations.

Content - Based Filtering:

Content-based filtering is a method used for product or freelancer recommendations. It suggests products or freelancers to customers or businesses by considering their past preferences and the characteristics of the products or skills of the freelancers. In this case, it's employed to match freelancers with jobs that require specific skills, using React Redux for implementation. Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022). <https://doi.org/10.1186/s40537-022-00592-5>

2.2.3 Technique

Another approach to recommendation systems that could be applicable to this project is The knowledge-based recommendation. Users are presented with recommendations by knowledge-based systems based on their expressly stated preferences. or requirements, rather than on their past behavior or attributes. This approach could be useful for businesses looking for freelancers with specific skills and expertise. The system could use a questionnaire or survey to gather information about the business's requirements and preferences and then recommend freelancers who match those criteria. However, I have chosen to implement content-based filtering using React Redux because it is a simpler and more straightforward approach that can still provide accurate recommendations based on the freelancers' skill sets. Additionally, knowledge-based recommendation systems may require more manual input from the user, which could make the system less user-friendly and efficient. In contrast, content-based filtering can automatically generate recommendations based on the data available in the system, making it more efficient and user-friendly.

2.3 Project Methodology

The project methodology selected for this job recommender system is Agile. Agile focuses on iterative development, flexibility in planning, and delivering functional software early in the process. The stages in the Agile methodology for this project include planning, design, development, testing, deployment, and monitoring. Planning involves defining the project scope, setting goals, and creating a roadmap. Design encompasses designing the system's architecture, including the database schema and user interface. Development involves building the system incrementally and continuously testing for bugs. Testing ensures the system performs as planned and complies with standards. Deployment allows for the system to be accessed in a real-world setting, with support for any issues that may arise. Monitoring involves tracking performance, security vulnerabilities, and making improvements based on feedback and observations. The Agile methodology is well-suited for projects with changing needs and emphasizes collaboration throughout the development. The project requirements cover hardware, software, and other requirements for finishing the job recommender system project effectively.

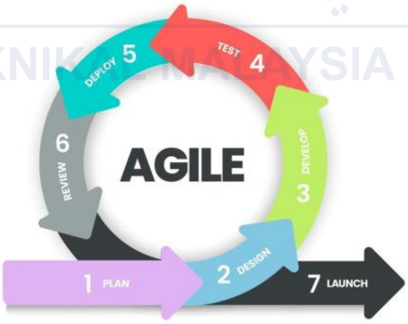


Figure 2.3 1: The Agile Model Process

2.4 Project Requirements

2.4.1 Software Requirement

Table 2.4.1 1: Software Requirement

Category	Tool
Documentation	Microsoft Office Word
	Draw.io
Programming Tool	Visual Studio Code
	MongoDB Compass

2.4.2 Hardware Requirement

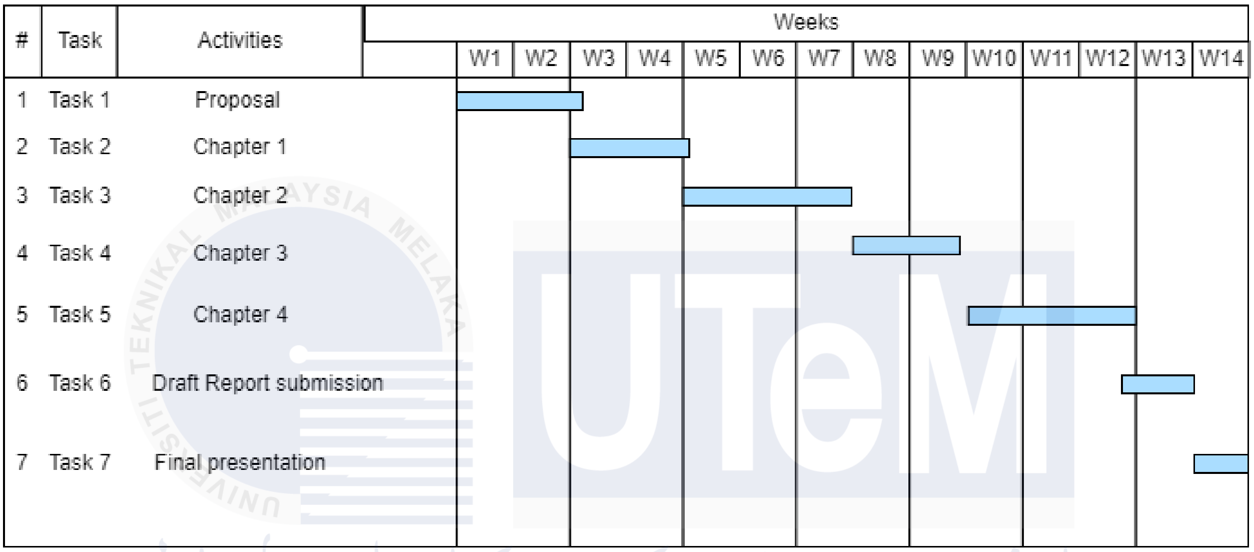
Table 2.4.2 1: Hardware Requirement

Laptop Brand	Type of windows	Type of processor	RAM
MSI	Windows 11	Intel i7	24GB

2.4.3 Other Requirements

2.5 Project Schedule and Milestones

2.5.1 Project Gantt chart:



#	Task	Activities
1	Task 1	Proposal
2	Task 2	Chapter 1
3	Task 3	Chapter 2
4	Task 4	Chapter 3
5	Task 5	Chapter 4
6	Task 6	Draft Report submission
7	Task 7	Final presentation

Figure 2.5.1 1: Gantt diagram for the project schedule

2.6 Conclusion

The study's methodology and literature review are described in depth in this Chapter of the project report.. It identifies the domain related to the project, discusses the existing system and explains the approach taken. It also describes the project methodology, requirements and schedule. The next activities to be developed will be based on the project schedule and milestones outlined in this chapter.

CHAPTER 3: ANALYSIS

3.1 Introduction

This chapter focuses on analyzing the current system, identifying problems, defining data and functional requirements, specifying non-functional requirements, and considering software, hardware, and other system needs.

3.2 Problem Analysis

3.2.1 Activity diagram

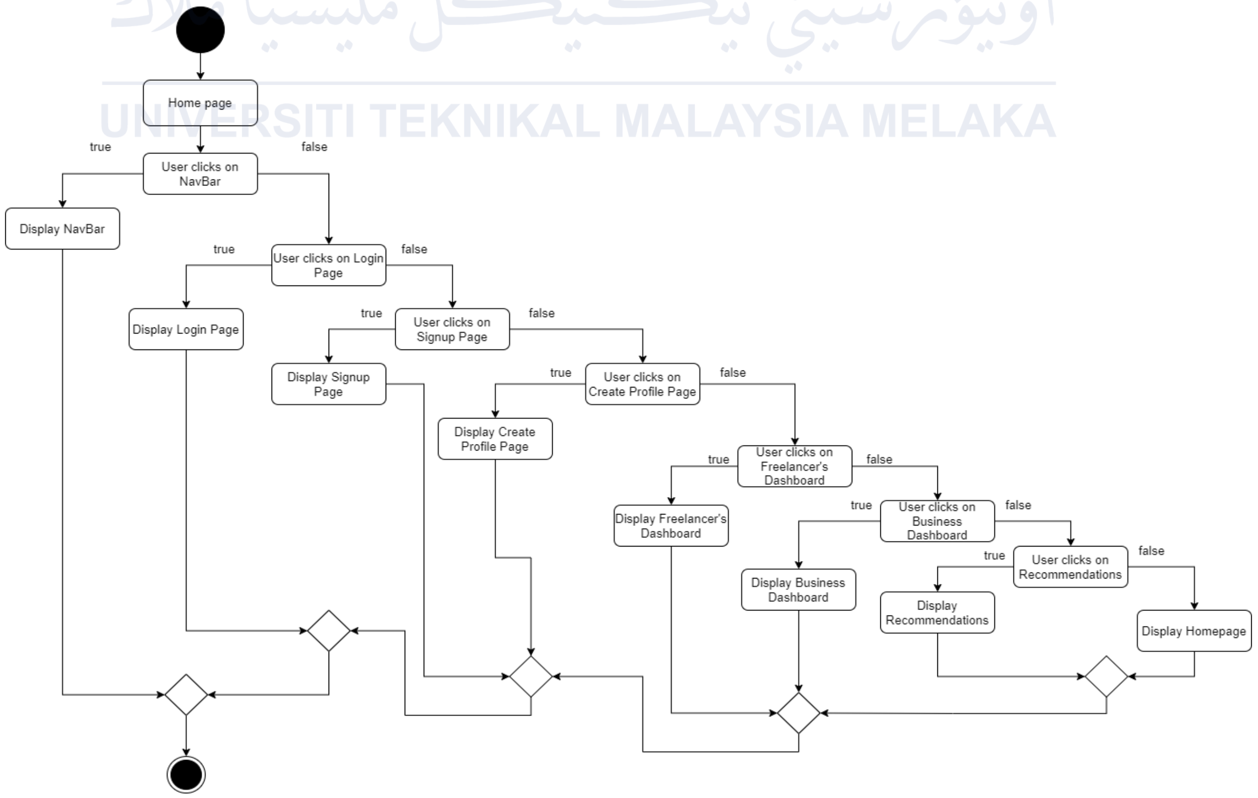

Home page
true
false
User clicks on NavBar
Display NavBar
true
false
User clicks on Login Page
Display Login Page
true
false
User clicks on Signup Page
Display Signup Page
true
false
User clicks on Create Profile Page
Display Create Profile Page
true
false
User clicks on Freelancer's Dashboard
Display Freelancer's Dashboard
true
false
User clicks on Business Dashboard
Display Business Dashboard
true
false
User clicks on Recommendations
Display Recommendations
Display Homepage

Figure 2.2.1 1: Activity diagram

3.2.2 Sequence diagram

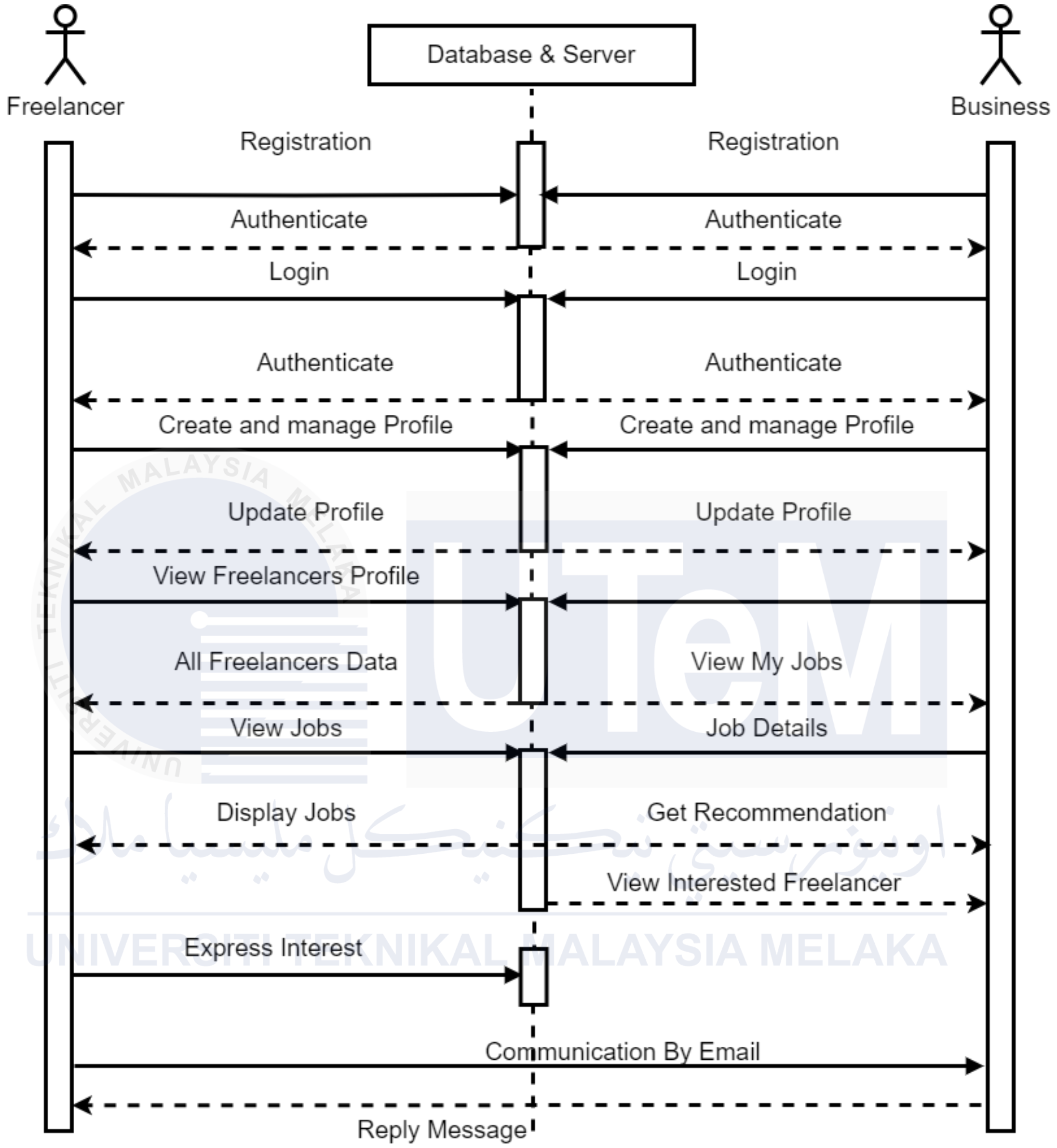


Figure 3.2.2 1: Sequence diagram

3.3 Requirement analysis

3.3.1 Data Requirement

3.3.1.1 Data flow diagram Level "0":

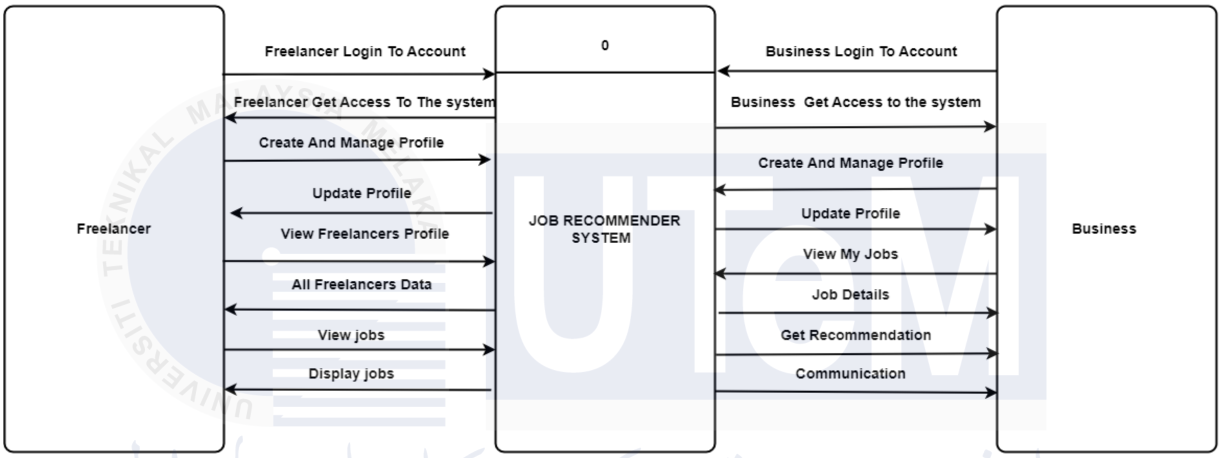


Figure 3.3.1.1 1: Data flow diagram Level "0"

3.3.2 Functional Requirement

Table 3.3.2 1: Functional Requirement

Requirement ID	Requirement Description
FR1	User Registration: Users should be able to register as freelancers or businesses, providing unique login credentials.
FR2	User Profile Creation: Freelancers can create and manage their profiles, including skills, experience, and portfolio.
FR3	Business Profile Creation: Businesses can create and update their company profiles with relevant information.
FR4	Job Posting: Businesses can create and post job opportunities, specifying job details and required skills.
FR5	Job Search: Freelancers can search available job postings based on criteria like skills.
FR6	Skill-Based Job Matching: The system should match freelancers with job requirements, providing suitable recommendations.
FR7	Expressing Interest: Freelancers can express their interest in specific job postings and communicate with businesses.
FR8	Communication: The system should facilitate communication between freelancers and businesses for project discussions.
FR9	Profile Updates: Freelancers can update their profile information, including skills, experience, and portfolio.
FR10	Review and Rating: Businesses can review and rate freelancers' work, providing feedback for future engagements.

3.3.2.1 Use Case Diagram

3.3.2.2 Use-Case Diagram for Freelancers

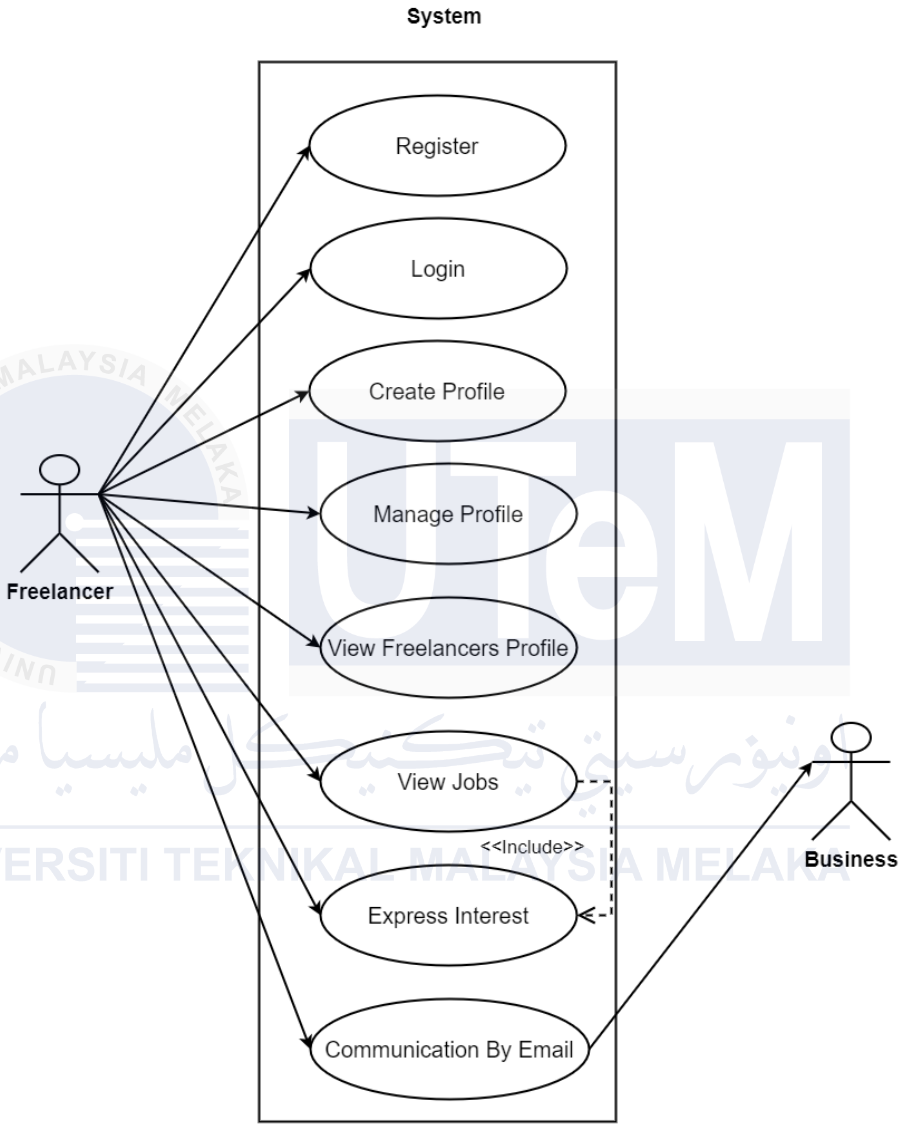


Figure 3.3.2.2 1: Use-Case Diagram for Freelancers

3.3.2.3 Use-Case Diagram for Business

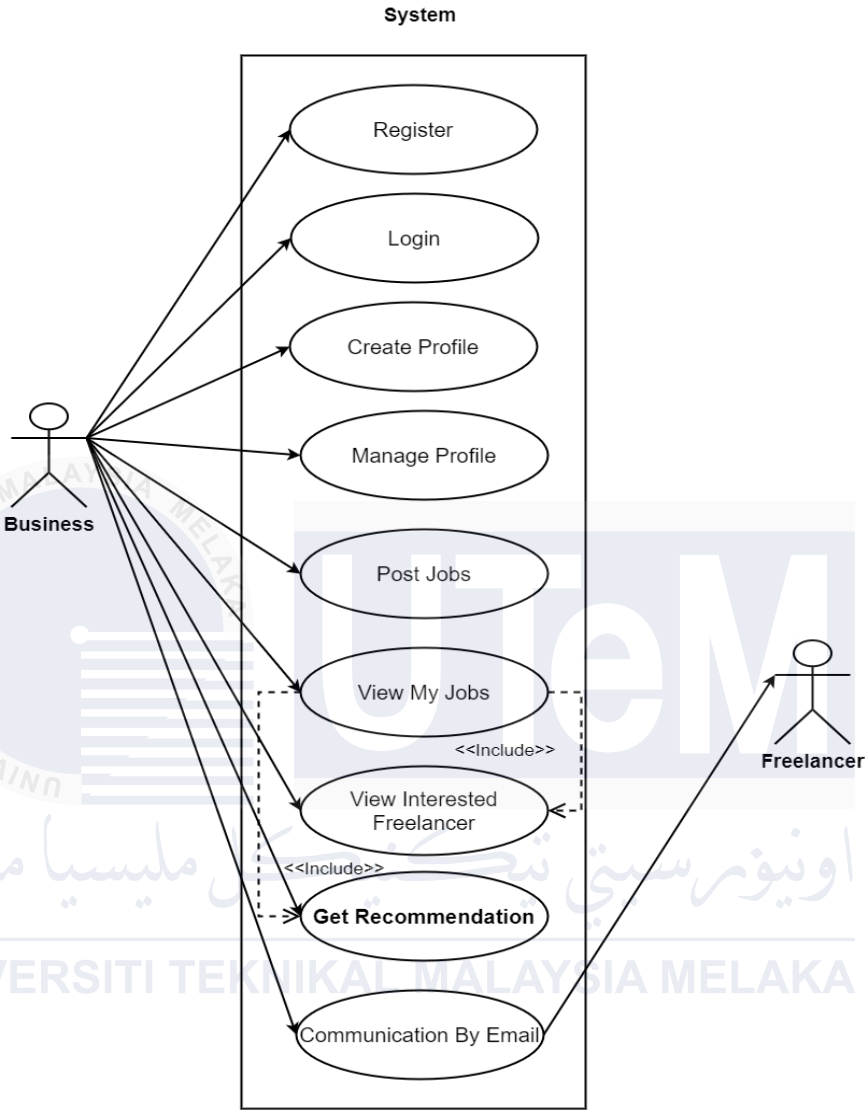


Figure 3.3.2.3 1: Use-Case Diagram for Business

3.3.3 Non-functional Requirement

Table 3.3.3 1: Non-functional Requirement

Requirement ID	Requirement Description
NFR1	Accuracy: The job recommendation algorithm should have a high level of accuracy in matching freelancers with job requirements, ensuring relevant and suitable recommendations.
NFR2	Privacy: The system should respect and protect the privacy of users' personal information, adhering to applicable data protection regulations and providing secure data handling practices.
NFR3	Availability: The system should have high availability, minimizing downtime and ensuring that freelancers and businesses can access the platform and its features reliably and consistently.
NFR4	Scalability: The system must be configured to manage growing user traffic and data volume, enabling seamless performance even as the user base and job postings grow over time.
NFR5	Adaptability: The system should be adaptable to evolving business needs and technological advancements, making it easy to integrate new features, update algorithms, and accommodate future enhancements.

3.3.4 Others Requirement

3.3.4.1 Software Requirement

Table 3.3.4.1 1: Software Requirement

Category	Tools	Description
Documentation	Microsoft Office Word	A comprehensive word processing program used for creating, editing, and printing documents.
	Draw.io	An online diagramming tool that allows for the creation of database schemas, network diagrams, UML diagrams, flowcharts, and more.
Programming Tools	Visual Studio Code	A feature-rich code editor widely used for MERN stack development, offering debugging, IntelliSense, Git integration, and other useful features.
	MongoDB Compass	An application with a graphical user interface (GUI) for controlling, MongoDB databases, providing an intuitive interface for viewing, querying, and analyzing MongoDB data.
	Postman	A popular API development and testing tool that simplifies the testing of MERN application APIs, offering features such as automated testing and collaboration.

3.3.4.2 Hardware Requirement

Table 3.3.4.2: Software Requirement

Laptop Brand	Type of windows	Type of processor	RAM
MSI	Windows 11	Intel i7	24GB

3.4 Conclusion

The chapter focuses on problem analysis and requirement analysis, including examining the current system's scenario, visualizing the system flow, defining non-functional requirements, specifying data and functional requirements, and talking about software, hardware, and other requirements for the system. Next chapter involves design activities such as architecture, interface, database, component. The goal is to create a detailed design that meets the requirements and prepares for implementation.

CHAPTER 4: DESIGN

4.1 Introduction

System architecture is covered in the "System Design" chapter., user interface, database design, software design, and implementation, aiming to bridge the gap between conceptual requirements and actual system development.

4.2 High-Level Design

High-level designs offer a thorough perspective of the entire system. They list the key elements that will be used in the finished project. These designs offer a thorough grasp of the architecture and operation of the system. They act as a model for the user interface and system architecture.

4.2.1 System Architecture

4.2.1.1 Client Server Architecture

The architecture used in this project is client-server. Both the business and the freelancer use the program that is running in the web browser as their client. The MongoDB cluster that displays the database collection serves as the client when it comes to the application's administrator. Express Web Framework is used by the author to create the API end routes, while NodeJS is used as the application server. Data is stored using MongoDB.

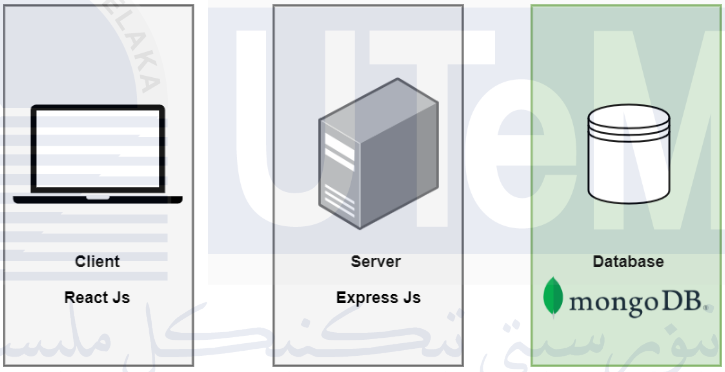


Figure 4.2.1.1 1: Client Server Architecture

4.2.2 User Interface Design

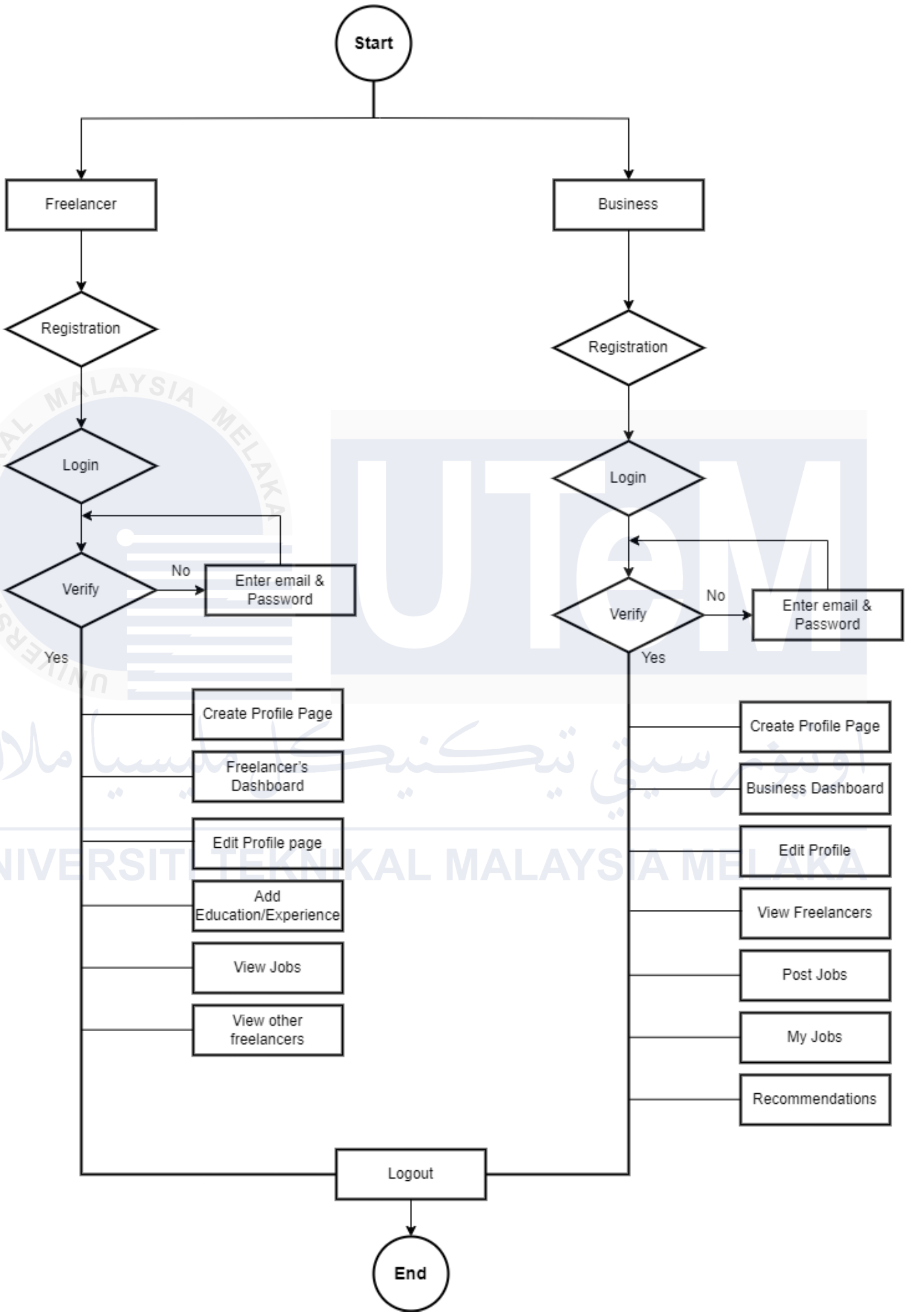


Figure 4.2.2 1: System Flowchart

4.2.3 Database Design

4.2.3.1 Conceptual and Logical Database Design

Table 4.2.3.1 1: Collection: Businesses

Field	Type	Description
_id	Objectid	Unique identifier
companyname	String	Company Name
contactname	Number	Contact Name
contact email	String	Company email address
password	String	Company Password
location	String	Company Location
companydescription	String	Company Description
date	Date	Date

Table 4.2.3.1 2: Collection: Businessprofiles

Field	Type	Description
_id	Objectid	Unique identifier
business	String	Business
website	String	Company Websites
status	String	Company status
employeecount	Number	Number of Employee
established	String	Established data
clients	String	The clients
companycategory	String	Company category

Table 4.2.3.1 3: Collection: jobs

Field	Type	Description
_id	Objectid	Unique identifier
business	String	Business
jobtitle	String	Job title
jobdescription	String	Job description
skillsetreq	Object	Skills required
jobbudget	Number	Job budget
jobduration	Number	Job duration
interested	Array	Interested freelancers
date	Date	date

Table 4.2.3.1 3: Collection: user

Field	Type	Description
_id	Objectid	Unique identifier
firstname	String	Freelancer First Name
lastname	String	Freelancer Last Name
username	String	User name
email	String	Freelancer email
password	String	Password
linkdeln	String	FreelancerLinkdin account
location	String	Freelancers Location
age	String	Freelancer Age
description	String	Freelancer Description
date	Date	Date

Table 4.2.3.1 4: Collection: freelancerprofiles

Field	Type	Description
_id	Objectid	Unique identifier
user	String	User
company	String	Company name
website	String	User website
status	String	Status
githubusername	String	Github username
skills	Array	User skills
student	True	Student or freelancer
social	Object	User social media account
experience	String	User experience
education	Object	User education
date	Date	Date

4.3 Detailed Design

4.3.1 Software Design

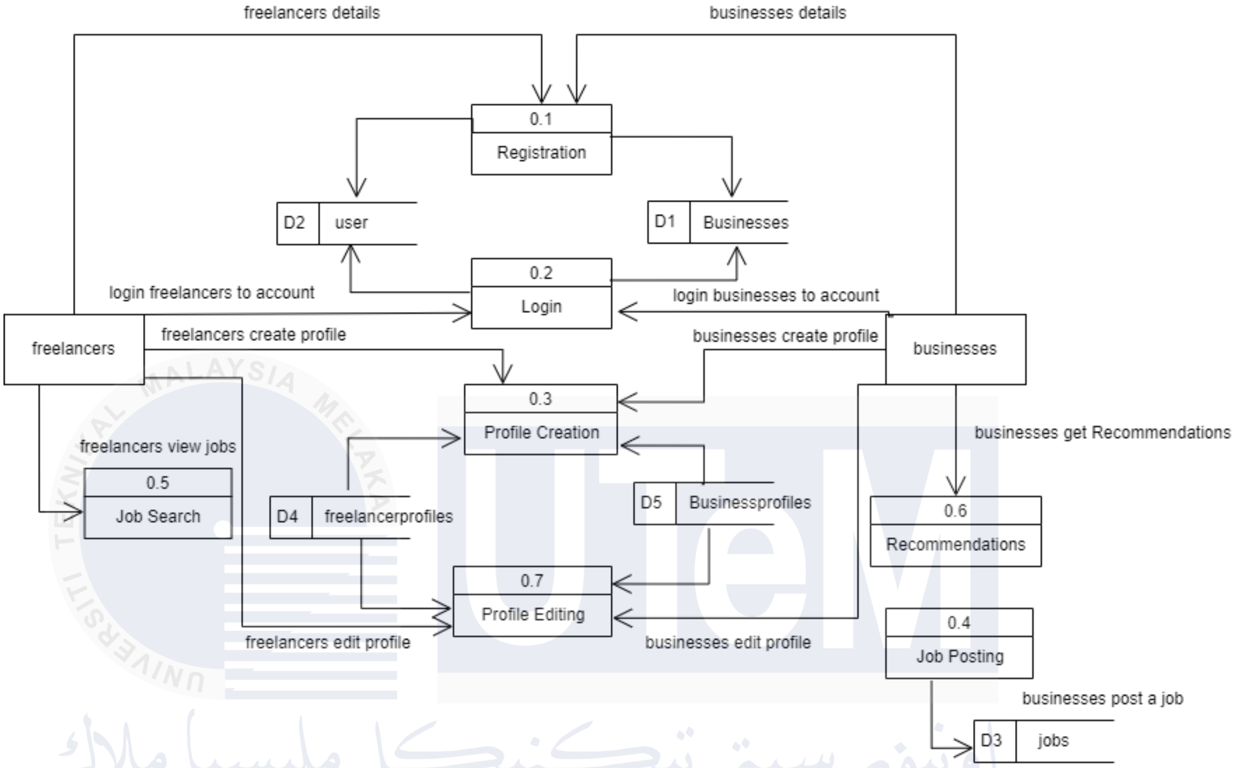


Figure 4.3.1 1: Data Flow Diagram Level 1

4.3.2 Physical Database Design

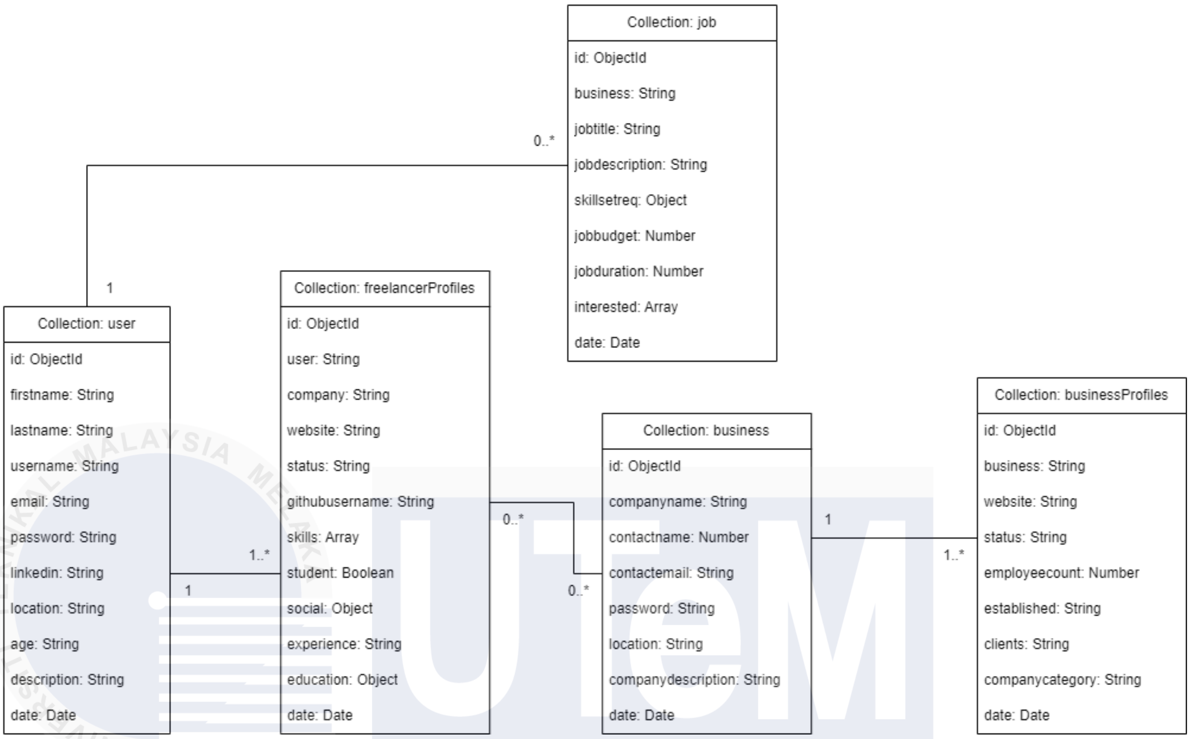


Figure 4.3.2 1: Physical Data Modeling

4.4 Conclusion

This chapter covers system design, database design, and software design, all of which encompass user interface and system architecture. The work to be developed after the design phase is the implementation phase, which comprises turning the design into usable software components.

CHAPTER 5: IMPLEMENTATION

5.1 5.1 Introduction

Using the MERN stack to develop application features is explained in this chapter. It covers both client and server components. Routes, Middleware, Configuration, and server.js comprise the server. Additionally covered are front-end technologies and design.

5.2 5.2 Software Development Environment setup

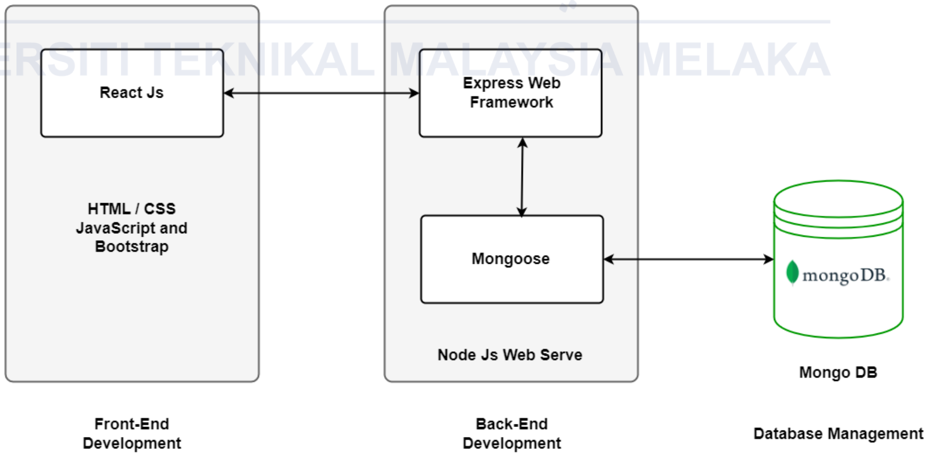


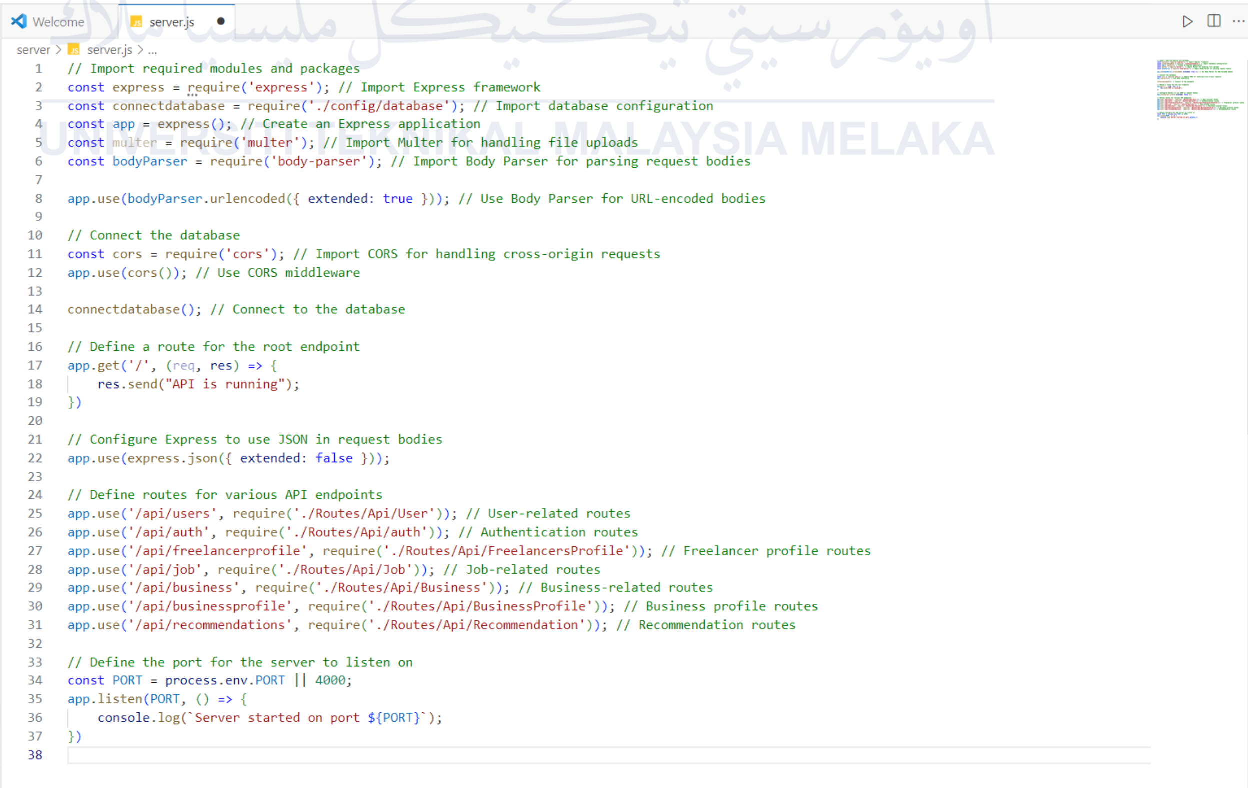
Figure 5.2 1:diagram

5.2.1 MongoDB setup

The first step in setting up MongoDB involves creating a cluster using MongoDB Atlas, a managed database service that automates deployment. You choose settings like cloud provider, memory, and region. The cluster acts as an online database accessible through a MONGO_URI. This link goes into the app's config for connecting. A code snippet is used to establish connections, showing "Database is Connected" on success.

5.2.2 Setting up the server

The next step is to set up the server using the Express framework. The entry point to the application is represented by the code fragment labelled "Server.js" that is provided below. API routes are configured using the app.use() function, necessitating both the location to the file containing the route specifications as well as the actual route. The connectdatabase() function is used, and the server is configured to run on port 4000. is executed to establish a linkage between the server and the database.



```
// Import required modules and packages
const express = require('express'); // Import Express framework
const connectdatabase = require('./config/database'); // Import database configuration
const app = express(); // Create an Express application
const multer = require('multer'); // Import Multer for handling file uploads
const bodyParser = require('body-parser'); // Import Body Parser for parsing request bodies

app.use(bodyParser.urlencoded({ extended: true })); // Use Body Parser for URL-encoded bodies

// Connect the database
const cors = require('cors'); // Import CORS for handling cross-origin requests
app.use(cors()); // Use CORS middleware

connectdatabase(); // Connect to the database

// Define a route for the root endpoint
app.get('/', (req, res) => {
    res.send("API is running");
})

// Configure Express to use JSON in request bodies
app.use(express.json({ extended: false }));

// Define routes for various API endpoints
app.use('/api/users', require('./Routes/Api/User')); // User-related routes
app.use('/api/auth', require('./Routes/Api/auth')); // Authentication routes
app.use('/api/freelancerprofile', require('./Routes/Api/FreelancersProfile')); // Freelancer profile routes
app.use('/api/job', require('./Routes/Api/Job')); // Job-related routes
app.use('/api/business', require('./Routes/Api/Business')); // Business-related routes
app.use('/api/businessprofile', require('./Routes/Api/BusinessProfile')); // Business profile routes
app.use('/api/recommendations', require('./Routes/Api/Recommendation')); // Recommendation routes

// Define the port for the server to listen on
const PORT = process.env.PORT || 4000;
app.listen(PORT, () => {
    console.log(`Server started on port ${PORT}`);
})

```

Figure 5.2.2 1: Server.js File

5.3 Software Configuration Management

5.3.1 Configuration environment setup

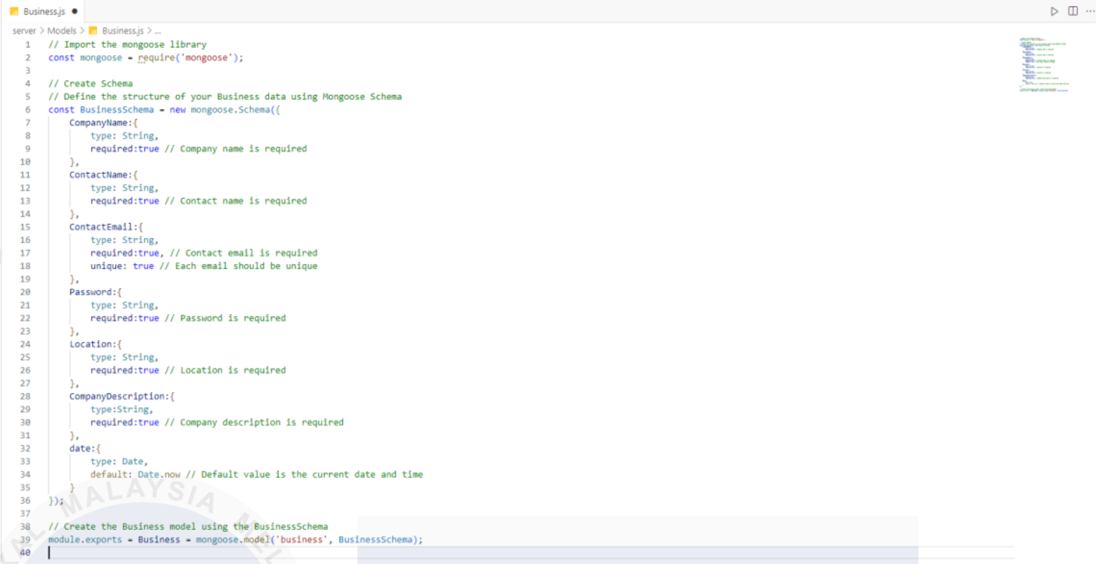
5.3.1.1 Mongoose Schemas

The next step was to create mongoose Schema when the server was configured and the database was connected.

Table 5.3.1.1 1: Five schemas are used by the program

Schema	Description
Business	Only the properties listed in the schema are accepted and permitted for database storage during registration for business. Before being saved, passwords are encrypted.
Business Profile	Building a profile inside the application requires the properties in the company profile. This model is connected to the user model in question through a business schema. types of monkey.Schema.The ID of actual user models can be matched by a property type called ObjectID.
Job	The property type mongoose connects the company and the job schema, as was already mentioned.Schema.Types. ObjectID. The list of potential freelancers interested in the position that is tied to the user Freelancer Schema model is also included in the Job Schema.
Freelancer	When a freelancer registers, the freelancers schema only allows the database to store properties that are defined by the schema. Passwords are encrypted before being stored.
Freelancer Profile	The characteristics of the profile for a freelancer are those that must be present in an application profile. The mongoose is what we employ.Schema.Types.To connect this model to the Freelancer Schema's actual user model, use the ObjectID property type, which connects to the id of actual user models.

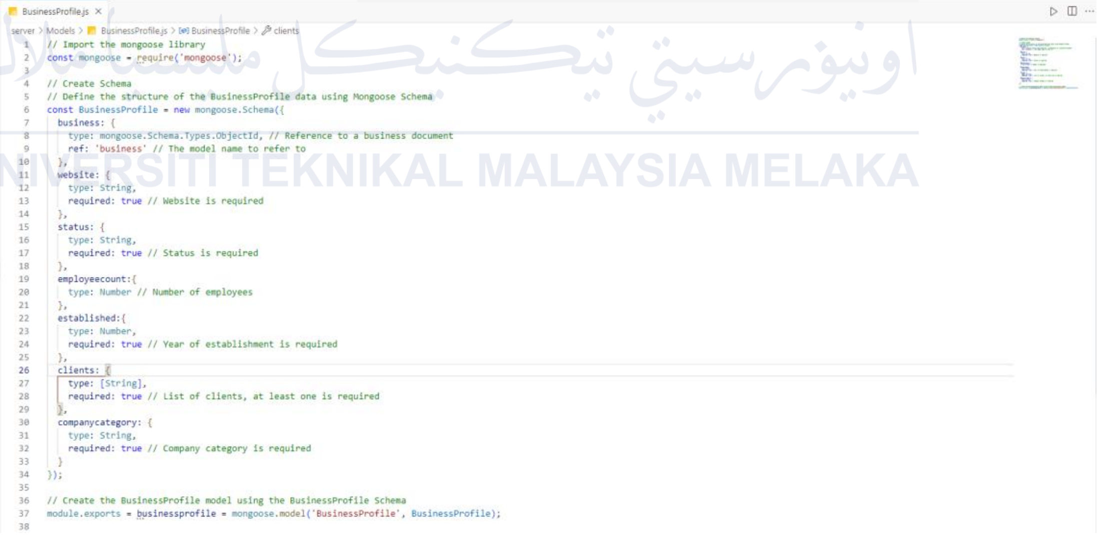
Business Schema



```javascript
// Import the mongoose library
const mongoose = require('mongoose');

// Create Schema
// Define the structure of your Business data using Mongoose Schema
const BusinessSchema = new mongoose.Schema({
    CompanyName:{
        type: String,
        required:true // Company name is required
    },
    ContactName:{
        type: String,
        required:true // Contact name is required
    },
    ContactEmail:{
        type: String,
        required:true, // Contact email is required
        unique: true // Each email should be unique
    },
    Password:{
        type: String,
        required:true // Password is required
    },
    Location:{
        type: String,
        required:true // Location is required
    },
    CompanyDescription:{
        type:String,
        required:true // Company description is required
    },
    date:{
        type: Date,
        default: Date.now // Default value is the current date and time
    }
});

// Create the Business model using the BusinessSchema
module.exports = Business = mongoose.model('business', BusinessSchema);

```

Figure 5.3.1.1 1: Business.js File

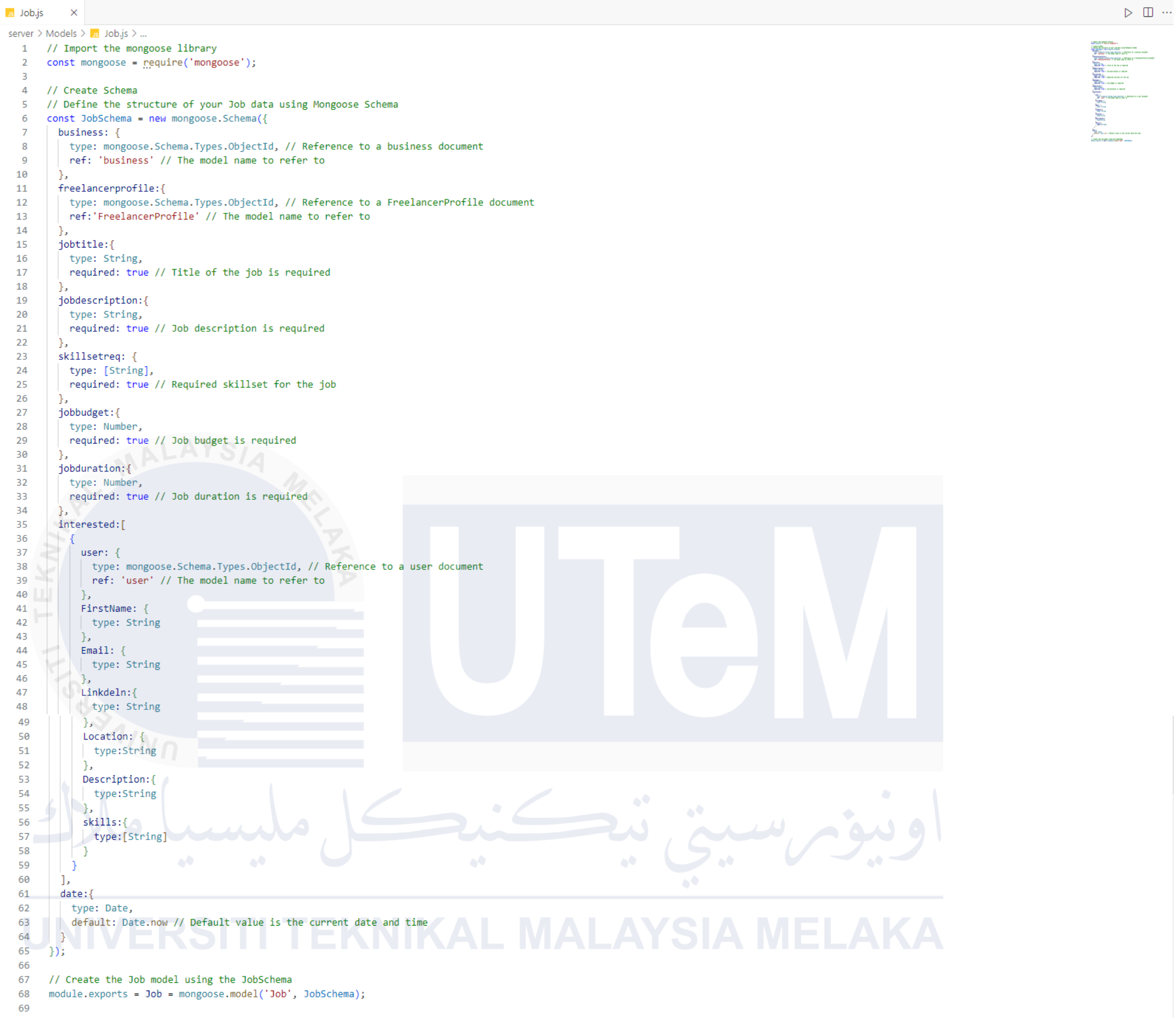
Business Profile Schema:


BusinessProfile.js
server > Models > BusinessProfile.js > BusinessProfile > clients

```javascript
// Import the mongoose library
const mongoose = require('mongoose');

// Create Schema
// Define the structure of the BusinessProfile data using Mongoose Schema
const BusinessProfile = new mongoose.Schema({
  business: {
    type: mongoose.Schema.Types.ObjectId, // Reference to a business document
    ref: 'business' // The model name to refer to
  },
  website: {
    type: String,
    required: true // Website is required
  },
  status: {
    type: String,
    required: true // Status is required
  },
  employeecount:{
    type: Number // Number of employees
  },
  established:{
    type: Number,
    required: true // Year of establishment is required
  },
  clients: {
    type: [String],
    required: true // List of clients, at least one is required
  },
  companycategory: {
    type: String,
    required: true // Company category is required
  }
});

// Create the BusinessProfile model using the BusinessProfile Schema
module.exports = businessprofile = mongoose.model('BusinessProfile', BusinessProfile);

```

Figure 5.3.1.1 2: BusinessProfile.js File

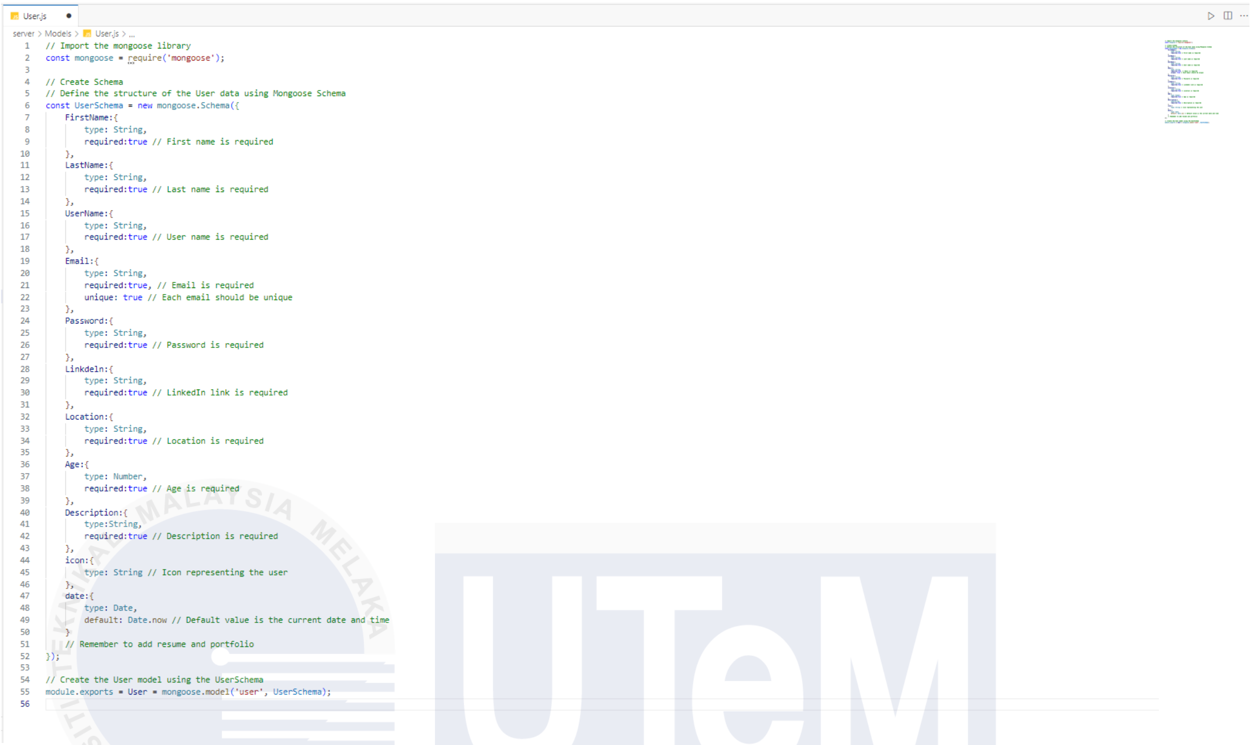
Job Schema



```javascript
// Import the mongoose library
const mongoose = require('mongoose');

// Create Schema
// Define the structure of your Job data using Mongoose Schema
const JobSchema = new mongoose.Schema({
  business: {
    type: mongoose.Schema.Types.ObjectId, // Reference to a business document
    ref: 'business' // The model name to refer to
  },
  freelancerprofile:{
    type: mongoose.Schema.Types.ObjectId, // Reference to a FreelancerProfile document
    ref:'FreelancerProfile' // The model name to refer to
  },
  jobtitle:{
    type: String,
    required: true // Title of the job is required
  },
  jobdescription:{
    type: String,
    required: true // Job description is required
  },
  skillsetreq: {
    type: [String],
    required: true // Required skillset for the job
  },
  jobbudget:{
    type: Number,
    required: true // Job budget is required
  },
  jobduration:{
    type: Number,
    required: true // Job duration is required
  },
  interested:[
    {
      user: {
        type: mongoose.Schema.Types.ObjectId, // Reference to a user document
        ref: 'user' // The model name to refer to
      },
      FirstName: {
        type: String
      },
      Email: {
        type: String
      },
      Linkdeln:{
        type: String
      },
      Location: {
        type:String
      },
      Description:{
        type:String
      },
      skills:{
        type:[String]
      }
    }
  ],
  date:{
    type: Date,
    default: Date.now // Default value is the current date and time
  }
});

// Create the Job model using the JobSchema
module.exports = Job = mongoose.model('Job', JobSchema);
```

Figure 5.3.1.1 3: Job.js File

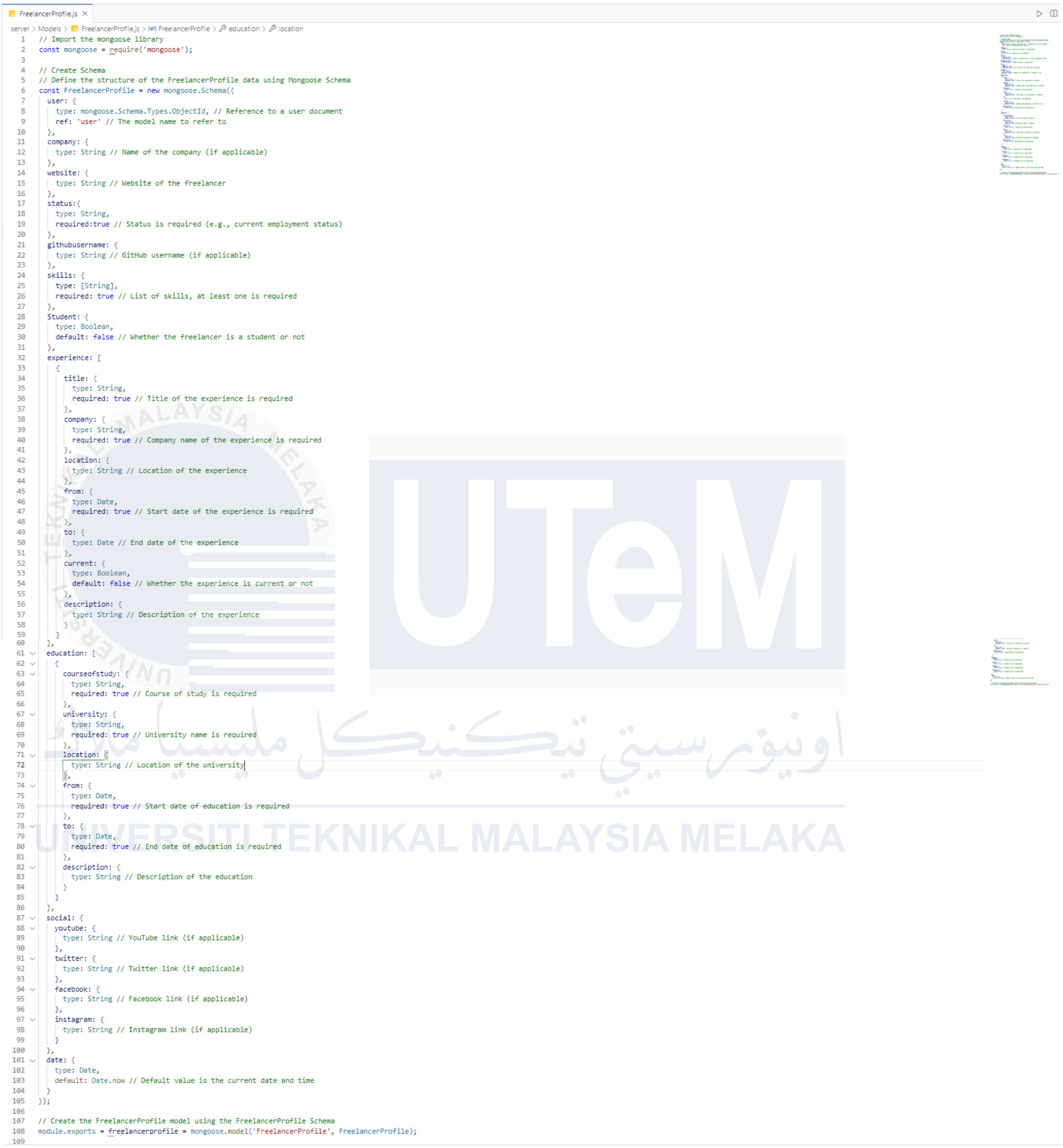
Freelancer Schema



```javascript
// Import the mongoose library
const mongoose = require('mongoose');

// Create Schema
// Define the structure of the User data using Mongoose Schema
const UserSchema = new mongoose.Schema({
    FirstName:{
        type: String,
        required:true // First name is required
    },
    LastName:{
        type: String,
        required:true // Last name is required
    },
    UserName:{
        type: String,
        required:true // User name is required
    },
    Email:{
        type: String,
        required:true, // Email is required
        unique: true // Each email should be unique
    },
    Password:{
        type: String,
        required:true // Password is required
    },
    Linkdeln:{
        type: String,
        required:true // LinkedIn link is required
    },
    Location:{
        type: String,
        required:true // Location is required
    },
    Age:{
        type: Number,
        required:true // Age is required
    },
    Description:{
        type:String,
        required:true // Description is required
    },
    icon:{
        type: String // Icon representing the user
    },
    date:{
        type: Date,
        default: Date.now // Default value is the current date and time
    }
    // Remember to add resume and portfolio
});

// Create the User model using the UserSchema
module.exports = User = mongoose.model('user', UserSchema);

```

Figure 5.3.1.1 4: User.js File

Freelancer Profile Schema



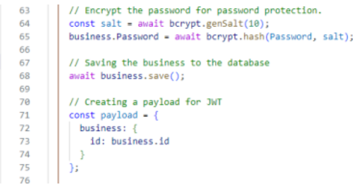
FreelancerProfile.js ×

server > Models > FreelancerProfile.js > FreelancerProfile > education > location

```javascript
// Import the mongoose library
const mongoose = require('mongoose');

// Create Schema
// Define the structure of the FreelancerProfile data using Mongoose Schema
const FreelancerProfile = new mongoose.Schema({
  user: {
    type: mongoose.Schema.Types.ObjectId, // Reference to a user document
    ref: 'user' // The model name to refer to
  },
  company: {
    type: String // Name of the company (if applicable)
  },
  website: {
    type: String // Website of the freelancer
  },
  status:{
    type: String,
    required:true // Status is required (e.g., current employment status)
  },
  githubusername: {
    type: String // GitHub username (if applicable)
  },
  skills: {
    type: [String],
    required: true // List of skills, at least one is required
  },
  Student: {
    type: Boolean,
    default: false // Whether the freelancer is a student or not
  },
  experience: [
    {
      title: {
        type: String,
        required: true // Title of the experience is required
      },
      company: {
        type: String,
        required: true // Company name of the experience is required
      },
      location: {
        type: String // Location of the experience
      },
      from: {
        type: Date,
        required: true // Start date of the experience is required
      },
      to: {
        type: Date // End date of the experience
      },
      current: {
        type: Boolean,
        default: false // Whether the experience is current or not
      },
      description: {
        type: String // Description of the experience
      }
    }
  ],
  education: [
    {
      courseofstudy: {
        type: String,
        required: true // Course of study is required
      },
      university: {
        type: String,
        required: true // University name is required
      },
      location: {
        type: String // Location of the university
      },
      from: {
        type: Date,
        required: true // Start date of education is required
      },
      to: {
        type: Date,
        required: true // End date of education is required
      },
      description: {
        type: String // Description of the education
      }
    }
  ],
  social: {
    youtube: {
      type: String // YouTube link (if applicable)
    },
    twitter: {
      type: String // Twitter link (if applicable)
    },
    facebook: {
      type: String // Facebook link (if applicable)
    },
    instagram: {
      type: String // Instagram link (if applicable)
    }
  },
  date: {
    type: Date,
    default: Date.now // Default value is the current date and time
  }
});

// Create the FreelancerProfile model using the FreelancerProfile Schema
module.exports = freelancerprofile = mongoose.model('FreelancerProfile', FreelancerProfile);
```

Figure 5.3.1.1 5: FreelancerProfile.js File

Password Encryption

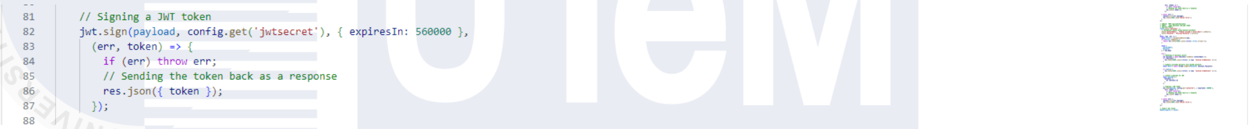


```
// Encrypt the password for password protection.
const salt = await bcrypt.genSalt(10);
business.Password = await bcrypt.hash(Password, salt);

// Saving the business to the database
await business.save();

// Creating a payload for JWT
const payload = {
  business: {
    id: business.id
  }
};
```

Figure 5.3.1.1 6: Using bcrypt to encrypt passwords

User passwords are secured using bcryptjs for hashing. Using bcrypt.genSalt() and bcrypt.hash(), passwords are encrypted and stored. This adds a layer of security by avoiding plain text storage. Encrypted passwords are saved in the database.

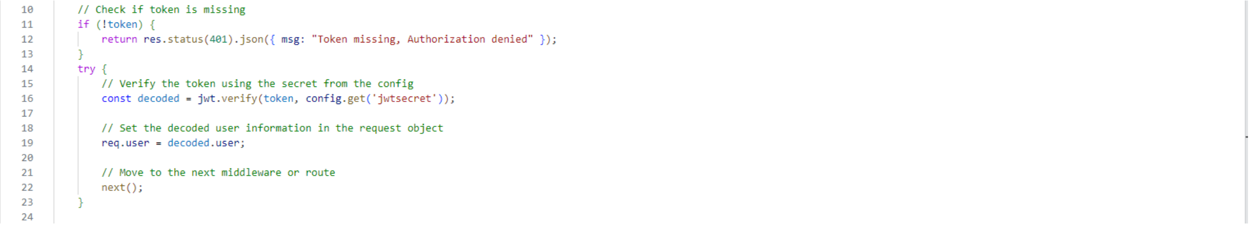
Authentication with JSON web token



```
    // Signing a JWT token
    jwt.sign(payload, config.get('jwtsecret'), { expiresIn: 560000 },
      (err, token) => {
        if (err) throw err;
        // Sending the token back as a response
        res.json({ token });
      });
```

Figure 5.3.1.1 7: Authentication with JWT sign ()

We must provide a JSON web token after the user registers with the application. These tokens can be used to access system-protected routes and to verify the user's identity. Because generating the JWT token requires the user's id, we are utilising it as the payload in our application. Whatever is related to the user can be the payload.

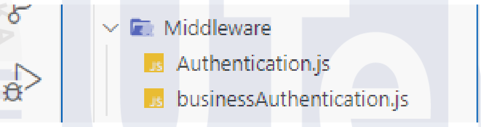


```
    // Check if token is missing
    if (!token) {
        return res.status(401).json({ msg: "Token missing, Authorization denied" });
    }
    try {
        // Verify the token using the secret from the config
        const decoded = jwt.verify(token, config.get('jwtsecret'));

        // Set the decoded user information in the request object
        req.user = decoded.user;

        // Move to the next middleware or route
        next();
    }
```

Figure 5.3.1.1 8: Authentication with Verify ()

JWT was installed with the command "npm install jsonwebtoken." After installation, we can construct the web token using jwt.sign() and check it using jwt.verify() when a user tries to access protected routes. JWT is integrated into the application in a manner akin to how node packages are added. Through a route in an API, user authentication takes place. The payload includes the user ID, JWT secret, and expiration. Reauthentication is necessary if the token expires. Upon creation, the token is sent in the response. When used in API queries, this token enables the server to verify the user's identity for protected routes.

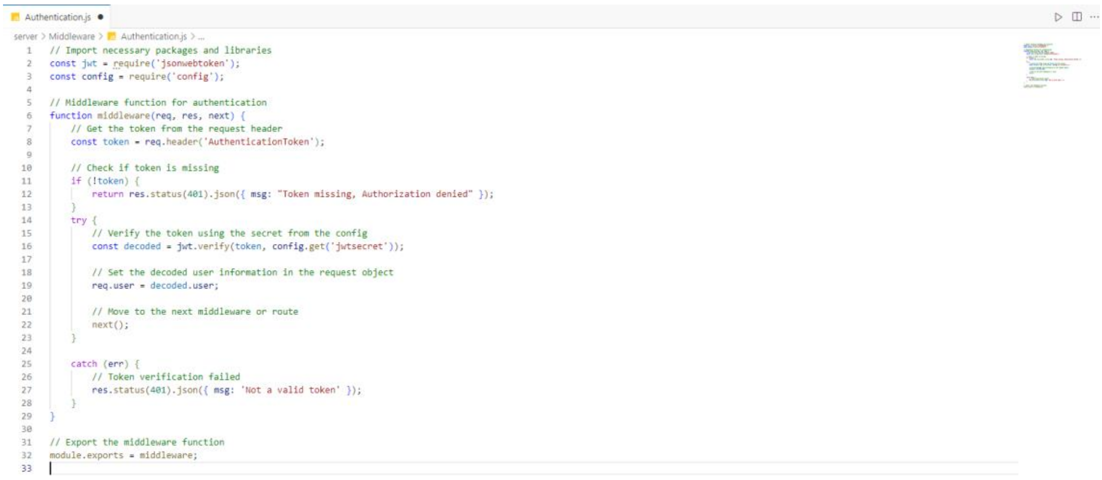
Middleware



Middleware
Authentication.js
businessAuthentication.js

Figure 5.3.1.1 9: Middleware

As previously stated, jwt.sign() provides a token; however, We must now return the token in order to authenticate the user and provide access to secured routes. We have developed bespoke middleware to accomplish this. Due to the middleware we are utilising, it requires three arguments (req, res, next). We can utilise the callback function next (), which allows us move on to the following piece of middleware, once the request has been fulfilled.



Authentication.js ●

server > Middleware > Authentication.js > ...

```javascript
// Import necessary packages and libraries
const jwt = require('jsonwebtoken');
const config = require('config');

// Middleware function for authentication
function middleware(req, res, next) {
    // Get the token from the request header
    const token = req.header('AuthenticationToken');

    // Check if token is missing
    if (!token) {
        return res.status(401).json({ msg: "Token missing, Authorization denied" });
    }
    try {
        // Verify the token using the secret from the config
        const decoded = jwt.verify(token, config.get('jwtsecret'));

        // Set the decoded user information in the request object
        req.user = decoded.user;

        // Move to the next middleware or route
        next();
    }

    catch (err) {
        // Token verification failed
        res.status(401).json({ msg: 'Not a valid token' });
    }
}

// Export the middleware function
module.exports = middleware;

```

Figure 5.3.1.1 10: Code for middleware used in user authentication

We have a request object with a header property that must contain the token when accessing protected routes. The header key for sending the token is "AuthenticationTokenjwt.verify(), which accepts the token from the header and the JWT secret that we have decided to use, may be used to decode the JWT token. By including the middleware as a second argument when establishing routes, we can secure the routes. When using secured routes, this aids in user authentication.

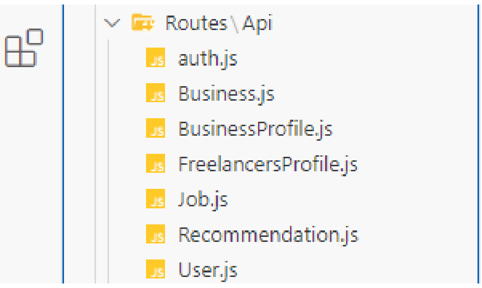
How JWT and Middleware Operate

- *Posting user login details*
- *Make a JSON web token that contains a secret.*
- *Send the JSON web token back to the browser.*
- *In order to confirm the token, create middleware.*
- *Middleware is used to send JWT to the authentication header.*
- *Sending user information and JWT token verification using middleware*
- *Sending a response back to the client.*

Routes:

In our application, a variety of route files are present, encompassing both private and public routes in their composition.

Files:



Routes \ Api
auth.js
Business.js
BusinessProfile.js
FreelancersProfile.js
Job.js
Recommendation.js
User.js

Figure 5.3.1.1 11: Route files

APIs:

Application Programming Interface (API) facilitates the communication between two programmes. Data sharing between two systems is a critical component of software development. Any time a user interacts with the programme in any way, the application's server receives an API request. The server examines the request after receiving it and

Each route's HTTP method, route path, and middleware used to verify the user before accessing that route are all included in the tables below. Input fields are defined for this route along with a description and, if the HTTP method is POST, the possibility that the user may need to supply details.

Table 5.3 1: Business.js

HTTP Method	Route Path	Middleware	Description	Possible input fields
POST	api/business	None	The business is registered with the application using this POST route. Prior to the password is encrypted before it is kept in the database. When a user accesses a protected route, a JWT token is issued to verify their identity.	CompanyDes cription CompanyNa me, ContactName , ContactEmail ,Password, Location,

Table 5.3 2: BusinessProfile.js

HTTP Method	Route Path	Middleware	Description	Possible input fields
GET	api/businessprofile/ me	Business Authentication	This method retrieves all business-related profile information.	None
POST	api/businessprofile	Business Authentication	After the company registers, a profile-creation prompt will appear. By gathering the relevant information, this POST route assists in building profiles for the company. By taking this approach, a company profile will be created and added to the database.	Website, Status, Establishe d, clients, company category and employee count

GET	api/businessprofile	None	This method retrieves the database with all the companies and their profiles.	None
GET	api/businessprofile/:business_id	None	Using the business id, this method retrieves all of the information for each particular firm. As a parameter, the business id is received.	None
DELETE	api/businessprofile	Admin Authentication	This method removes the database's business profile and company information.	None

Table 5.3 3: Freelancer.js

HTTP Method	Route Path	Middleware	Description	Possible input fields
POST	api/freelancers	None	The freelancers are registered with the application via this POST channel. The database stores the password after it has been encrypted. When using protected routes, a JWT token is issued to verify the user's identity.	UserName, FirstName, LastName, Email, Password, Linkedln, Location, Age, About

Table 5.3 4: FreelancerProfile.js

HTTP Method	Route Path	Middleware	Description	Possible input fields
GET	api/freelancerprofile/me	Freelancer Authentication	This route retrieves the freelancer's entire profile information.	None
POST	api/freelancerprofile	Freelancer Authentication	After registering, the user will be asked to build their profile. By gathering the relevant information, this POST route assists in helping freelancers create profiles. By taking this route, a freelancer's profile will be created and added to the database.	Qualification, website, status, github username, social media links
GET	api/freelancerprofile	None	This method populates the database with all of the freelancers' profiles. it can be used to show businesses freelancers	None

GET	api/freelancerprofile/:user_id	None	Using the freelancer's ID, this method retrieves all of their information. A parameter called the freelancer id is obtained.	None
DELETE	api/freelancerprofile	Admin Authentication	By using this method, the database's freelancer profile and freelancer information are removed.	None
PUT	api/freelancerprofile/experience	Freelancer Authentication	The freelancer can add experience to their profile after it has been built. By including experience, this method aids in updating the freelancer profile.	Title, company, location, from, to, and description
DELETE	api/freelancerprofile/experience/:exp_id	Freelancer Authentication	After adding experience, the freelancer will have the option to remove it if necessary. This route extracts the freelancer experience's id from the database and deletes it.	None
PUT	api/freelancerprofile/education	Freelancer Authentication	If the freelancer is still in school after creating their profile, they can substitute education for experience. By adding schooling information, this method assists with improving the freelancer profile.	university , location, Course of study, from, to, and description
DELETE	api/freelancerprofile/education/:edu_id	Freelancer Authentication	After the student or freelancer adds their education, they will be able to remove it if they like.	None

Table 5.3 4: Job.js

HTTP Method	Route Path	Middleware	Description	Possible input fields
POST	api/job	Business Authenticatio n	Middleware for business authentication is used since this route is protected. This route obtains the necessary information and adds a job to the database. after the job	job description, Job title, skill sets required,
			When posted, it becomes linked to the company profile.	job budget and job duration
GET	api/job	Business and freelancer authentication	Due to the security of this route, identity verification is required for the middleware of the system as well as the freelancer and the business.. This route collects all the job postings made by companies.	None
GET	api/job/:id	Authentication of businesses	Through this route, data pertaining to a single job is retrieved. The task id is supplied as an input parameter. This strategy aims to provide all job-related information on a single page.	None

DELETE	api/job/:id	Business Authenticatio n	This method aids the company in removing the jobs they have placed. Job ID is supplied as a parameter.	None
PUT	Api/job/interested /:id	Freelancer Authentication	If a freelancer prefers to work on-site, they might use this method to show interest in a position. This increases the number of freelancers that are interested in the task. The parameter "freelancer id" is received.	None
PUT	Api/job/uninterest ed/:id	Freelancer Authenticatio n	By using this method, independent contractors can withdraw their indicated interest in a job if they'd rather not accept it.	None

Table 5.3 5: Auth.js

HTTP Method	Route Path	Middleware	Description	Possible input fields
GET	api/auth/freelancer	Authentication of Freelancers	The information that freelancers provided upon registration is returned when their identities have been verified. Given its	None

			confidentiality, the password is not sent alone.	
GET	api/auth/business	Authentication of businesses	This route verifies the legitimacy of the business and returns the registration information for the business. Because it is confidential, the password is not sent alone.	None
POST	api/auth/freelancer	None	By comparing the password and email, this method authorises freelancers to access the application. It creates the JWT token needed to verify the freelancer's identity.	Email and Password
POST	api/auth/business	None	Through a comparison of the password and email, this route logs users into the application. It creates the JWT token needed to verify the legitimacy of the company.	Email and Password

5.3.1.2 Front-end Implementation:

The back-end of the project made use of several technologies. React.js is used to build the front end. Given the dissertation's focus, front-end details won't be explored in great length. Important elements will be described. We used "npm create-react-app devhub" to start a React app. "npm install packagename" is used to add dependencies to the package.json file.

Table 5.3.1.2 1: Front-end Implementation

Topic	Description
React Router	"react-router-dom" package was installed from the npm registry. used on the client side in the "app.js" file. 'Router' and 'Routes' were used to define the routes.
Private Route	made a component to verify the progress of user authentication. React Redux was used to pass the authentication state. only authenticated users have access to protected routes.
React Redux	used to transfer and control application-wide state. It was chosen because it works well with different reducers. regarded as the best option due to the utilisation of many reducers. Context API is an alternative state management choice.
Reducer	a procedure for managing state updates based on actions sent forth. identifies state changes and sends information to UI components. additional components that depend on the same state are updated. by calling "actions" to begin.
Actions	functions are charge of supplying data from Axios requests to the reducer. inherited as prop types by components that need them. The reducer's state is updated with response information after use in components.
Redux Dev Tools	Extension for the browser made specifically for monitoring and troubleshooting state changes. provides visibility into actions that have been taken and accessible data. Easy to add as a browser extension, open-source tool.
Components	components built as JavaScript functions that produce UI elements for the React application. organised into many files, each of which corresponds to different functionalities.
Styling	decided against using separate CSS files for styling. used components from the react-bootstrap framework to construct the user interface. All bootstrap component style requirements are specified inline within component tags.
	Employed React Icons library to incorporate icons as components within the UI elements.

5.3.2 Version Control Procedure

5.4 Implementation Status

Table 5.4 1: Implementation Status

Component/Module	Description	Duration to Complete	Date Completed
Homepage	The homepage is the first page users interact with. It includes the NavBar for navigation to different sections.	6 Days	April 5, 2023
Login Page	User authentication through email and password. Different appearances for various user types.	5 Days	April 13, 2023

Signup Page	New user registration with customizable forms based on user type.	7 Days	May 5, 2023
Create Profile Page	Users create and manage profiles with various input fields. Different fields for different user types.	8 Days	May 28, 2023
Freelancer's Dashboard	Interface for freelancers with features like profile editing, adding education/experience, viewing jobs and profiles.	6 Days	June 7, 2023
Business Dashboard	Interface for businesses to manage work and interactions with freelancers. Features include editing profiles, viewing freelancers, posting jobs, and tracking job statuses.	13 Days	June 15, 2023
Recommendations	Content-based filtering to recommend freelancers to businesses based on skills and job requirements.	19 Days	June 26 2023

5.5 Conclusion

The chapter covered setting up a development environment, establishing Software Configuration Management (SCM) processes, designing a configuration management system, and monitoring development progress. It emphasized creating a suitable environment, managing software changes effectively, organizing components, and tracking project advancement. The next step is testing. This ensures the software is error-free and meets requirements.

CHAPTER 6: TESTING

6.1 Introduction

In the forthcoming chapter, We'll cover the testing plan, team organization, environment setup, schedule, rationale, test design, and data details. we'll delve into our black-box testing strategy, evaluating the project's functionality, security, and user experience across diverse scenarios. This approach focuses on uncovering hidden issues, enhancing performance, and aligning with user expectations.

6.2 Test Plan

6.2.1 Test Organization

6.2.1.1 Project information

Table 6.2.1.1 1: Project information

Project Name	Job Recommender System
Project Supervisor	Dr. Intan Ermahani Binti A. Jalil
Project Manager	Khalid Ali
Test Manager	Khalid Ali
Start Date	August 15, 2023
End Date	September 20, 2023

Tester InformationTable 6.2.2.1 1

Table 6.2..1.2 1: Tester Information

Tester ID	First Name	Last Name	Status	Role
TST-001	Khalid	Ali	Student	Author

6.2.2 Test Environment

6.2.2.1 Hardware:

Table 6.2.2.1 2:Testing Machines

Component	Specifications
Processor:	Up to 8th Gen. Intel® Core™ i7 Processor
RAM:	24GB
Storage:	1TB NVMe SSD
Graphics Card:	NVIDIA GeForce RTX 3080

6.2.2.2 Software:

Table 6.2.2.2 1: Software

Component	Version
Operating System	Windows 11 Pro (Version 21H1)
Browser	Chrome 116.0.5845.97 (Official Build) (64-bit)

6.2.3 Test Schedule

Table 6.2.3 1: Test Schedule

Cycle	Focus	Duration	Activities	Outcome
1	Login Page	3 days	Execute login-related test cases.	Validate login functionality. Verify error handling for various scenarios.
2	Signup Page, Create Profile Page	4 days	Execute signup-related test cases.	Ensure successful registration and profile creation.
3	Create Profile Page, Freelancer's Dashboard	3 days	Test profile creation functionality. Test profile data in Freelancer's Dashboard.	Validate accurate profile creation and display in dashboard.
4	Freelancer's Dashboard, Business Dashboard	4 days	Test freelancer dashboard functionality. Test navigation in Business Dashboard.	Ensure smooth functionality of both dashboards. Verify navigation and interactions.
5	Business Dashboard, Recommendations	4 days	Test business dashboard functionality. Test recommendations feature.	Validate proper functioning of the dashboard. Verify accuracy of recommendations.
6	Recommendations	2 days	Test recommendations feature.	Verify relevance and accuracy of recommendations.

6.3 Test Strategy

I intend to use a black-box testing technique to thoroughly evaluate the functionality and reliability of this project. This approach involves evaluating the system's behavior through various scenarios and user interactions, without delving into its internal code. By generating diverse test cases, examining functionality alignment, probing for edge cases, verifying security measures, testing user experience, identifying performance bottlenecks, and documenting defects, I aim to uncover hidden issues, security vulnerabilities, and usability flaws that might not be apparent from code inspection alone. Through this method, I will ensure the project's quality, align it with user expectations, and enhance its overall performance.

6.3.1 Classes of tests

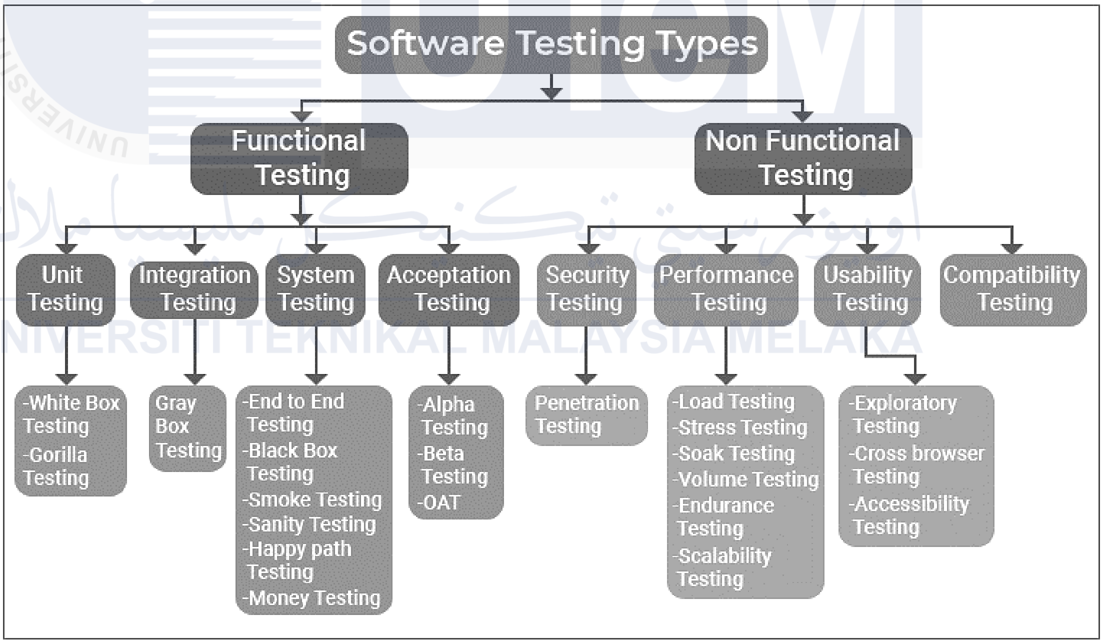


Figure 6.3.1 1: Classes of tests

6.4 Test Design

6.4.1 Test Description

Table 6.4.1 1: Page Name: Signup Page For Business Test Cases

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-SPB-001	Valid Company Name	Company Name: UTeM Holdings.	Company Name is accepted without errors.	No errors observed.
TC-SPB-002	Empty Company Name	Company Name: empty	User receives an error message indicating the Company Name is required.	Error message displayed.
TC-SPB-003	Invalid Company Name	Company Name: *@#/Company Name	User receives an error message indicating an invalid Company Name.	Error message displayed.
TC-SPB-004	Valid Contact Name	Contact Name: Khalid Ali	Contact Name is accepted without errors.	No errors observed.
TC-SPB-005	Empty Contact Name	Contact Name: empty	User receives an error message indicating the Contact Name is required.	Error message displayed.
TC-SPB-006	Invalid Contact Name	Contact Name: 156456	The user receives an error message stating that the Contact Name is necessary.	Error message displayed.
TC-SPB-007	Valid Contact's Email	Contact's Email: khalid@gemail.com	Contact's Email is accepted without errors.	No errors observed.
TC-SPB-008	Invalid Contact's Email	Contact's Email: Khalid-email	The user receives an error notice indicating that the email format is incorrect.	Error message displayed.
TC-SPB-009	Empty Contact's Email	Contact's Email: empty	User receives an error message indicating the Contact's Email is required.	Error message displayed.
TC-SPB-010	Valid Password	Password: Kk123456	Password is accepted without errors.	No errors observed.
TC-SPB-011	Empty Password	Password: empty	The user is informed with an error notice that the password is necessary.	Error message displayed.
TC-SPB-012	Invalid Password	Password: short	User receives an error message indicating an invalid Password.	Error message displayed.

TC-SPB-013	Valid Confirm Password	Password: Kk123456 Confirm Password: Kk123456	Confirm Password is accepted without errors.	No errors observed.
TC-SPB-014	Mismatched Confirm Password	Password: Kk123456 Confirm Password: Cc789012	User receives an error message indicating passwords do not match.	Error message displayed.
TC-SPB-015	Empty Confirm Password	Password: Kk123456 Confirm Password: empty	User receives an error message indicating Confirm Password is required.	Error message displayed.
TC-SPB-016	Valid Location	Location: Melaka	Location is accepted without errors.	No errors observed.
TC-SPB-017	Empty Location	Location: empty	User receives an error message indicating the Location is required.	Error message displayed.
TC-SPB-018	Invalid Location	Location: !@#IMelaka	User receives an error message indicating an invalid Location.	Error message displayed.
TC-SPB-019	Valid Company's Description	Company's Description: We provide innovative e-commerce solutions.	Company's Description is accepted without errors.	No errors observed.
TC-SPB-020	Empty Company's Description	Company's Description: empty	Company's Description is accepted without errors (assuming it's not a required field).	No errors observed.
TC-SPB-021	Invalid Agreement	This test case requires no input.	Registration process detects invalid agreement state and prevents registration.	Invalid agreement detected.

Table 6.4.1 2: Page Name: Business Login Page Test Cases

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-BLP-001	Valid Login	Email: khalid@gmail.com Password: Kk15k0kk	A successful login has occurred, and the user is then taken to the appropriate dashboard.	User logged in successfully.
TC-BLP-002	Empty Email and Password	Email: empty Password: empty	User receives error messages indicating that both fields are required.	Error messages displayed.
TC-BLP-003	Empty Email	Email: empty Password: Pa55w0rd	An error notice notifies the user that an email address is necessary.	Error message displayed.
TC-BLP-004	Empty Password	Email: khalid@gmail.com Password: empty	User receives an error message indicating that the Password is required.	Error message displayed.
TC-BLP-005	Invalid Email Format	Email: invalid-email Password: Pa55w0rd	A notice alerting the user to an invalid email format is received.	Error message displayed.
TC-BLP-006	Invalid Email and Password	Email: invalid-email Password: incorrect	An error notice informing the user that the login credentials are wrong is displayed.	Error message displayed.
TC-BLP-007	Valid Email, Invalid Password	Email: khalid@gmail.com Password: incorrect	The wrong login credentials are disclosed to the user via an error message.	Error message displayed.
TC-BLP-008	"Check me out" Checkbox	Email: khalid@gmail.com Password: Pa55w0rd Checkbox: Checked	The checkbox status is remembered and the user is logged in.	Checkbox status remembered.
TC-BLP-009	Unchecked "Check me out" Checkbox	Email: khalid@gmail.com Password: Pa55w0rd Checkbox: Unchecked	The checkbox status is not remembered after logging in.	Checkbox status not remembered.
TC-BLP-010	Submit Button Clicked	Email: empty Password: empty	User receives error messages indicating that both Email and Password are required.	Error messages displayed.

Table 6.4.1 3: Page Name: Signup Page For Freelance Test Cases

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-SFF-001	Valid User Name	User Name: freelancer123	User Name is accepted without errors.	No errors observed.
TC-SFF-002	Empty User Name	User Name: empty	User receives an error message indicating the User Name is required.	Error message displayed.
TC-SFF-003	Invalid User Name	User Name: 12345	User receives an error message indicating an invalid User Name.	Error message displayed.
TC-SFF-004	Valid First Name	First Name: Khalid	First Name is accepted without errors.	No errors observed.
TC-SFF-005	Empty First Name	First Name: empty	User receives an error message indicating the First Name is required.	Error message displayed.
TC-SFF-006	Invalid First Name	First Name: 12345	User receives an error message indicating an invalid First Name.	Error message displayed.
TC-SFF-007	Valid Last Name	Last Name: Smith	Last Name is accepted without errors.	No errors observed.
TC-SFF-008	Empty Last Name	Last Name: empty	User receives an error message indicating the Last Name is required.	Error message displayed.
TC-SFF-009	Invalid Last Name	Last Name: 12345	User receives an error message indicating an invalid Last Name.	Error message displayed.
TC-SFF-010	Valid Email	Email: khalid@gmail.com	Email is accepted without errors.	No errors observed.
TC-SFF-011	Invalid Email	Email: invalid-email	A notice alerting the user to an invalid email format is received.	Error message displayed.
TC-SFF-012	Empty Email	Email: empty	An error notice informs the user that the Email is necessary.	Error message displayed.
TC-SFF-013	Valid Personal Email	Personal Email: khalid.personal@gmail.com	Personal Email is accepted without errors.	No errors observed.
TC-SFF-014	Invalid Personal Email	invalid-email	An error notice indicating an improper Personal Email format is delivered to the user.	Error message displayed.
TC-SFF-015	Empty Personal Email	Personal Email: empty	Personal Email is accepted as optional.	No errors observed.
TC-SFF-016	Valid Password	Password: Kk125556	Password is accepted without errors.	No errors observed.
TC-SFF-017	Empty Password	Password: empty	An error notice informs the user that a password is necessary..	Error message displayed.
TC-SFF-018	Invalid Password	Password: short	User receives an error message indicating an invalid Password.	Error message displayed.
TC-SFF-019	Valid Confirm Password	Password: Kk125556 Confirm Password: Kk125556	Confirm Password is accepted without errors.	No errors observed.
TC-SFF-020	Mismatched Confirm Password	Password: Kk125556 Confirm Password: Bb789012	User receives an error message indicating passwords do not match.	Error message displayed.

TC-SFF-021	Empty Confirm Password	Password: Kk125556 Confirm Password: empty	Confirm Password is necessary, according to the error message the user receives.	Error message displayed.
TC-SFF-022	Valid LinkedIn Profile	LinkedIn: linkedin.com/in/khalidsmith	LinkedIn link is accepted without errors.	No errors observed.
TC-SFF-023	Invalid LinkedIn Profile	LinkedIn: notalinkedinprofile	User receives an error message indicating an invalid LinkedIn profile link.	Error message displayed.
TC-SFF-024	Empty LinkedIn Profile	LinkedIn: empty	LinkedIn link is accepted as optional.	No errors observed.
TC-SFF-025	Valid Location	Location: Melaka	Location is accepted without errors.	No errors observed.
TC-SFF-026	Empty Location	Location: empty	User receives an error message indicating the Location is required.	Error message displayed.
TC-SFF-027	Invalid Location	Location: !@#Incorrect	User receives an error message indicating an invalid Location.	Error message displayed.
TC-SFF-028	Valid Age	Age: 23	Age is accepted without errors.	No errors observed.
TC-SFF-029	Empty Age	Age: empty	Age is accepted as optional.	No errors observed.
TC-SFF-030	Invalid Age	Age: -4	User receives an error message indicating an invalid Age.	Error message displayed.
TC-SFF-031	Valid Description	Describe yourself: I'm a skilled designer...	Description is accepted without errors.	No errors observed.
TC-SFF-032	Empty Description	Describe yourself: empty	User receives an error message indicating the description is required.	Error message displayed.
TC-SFF-033	Valid Agreement	Agree to terms and conditions: Checked	Agreement is checked without errors.	No errors observed.
TC-SFF-034	Invalid Agreement	Agree to terms and conditions: Unchecked	User receives an error message indicating agreement is required.	Error message displayed.
TC-SFF-035	Submit Form	form fields filled correctly	Form is successfully submitted.	Form submitted successfully.

Table 6.4.1 4: Page Name: Freelancer Login Page Test Cases

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-FLP-001	Valid Email and Password	Email: khalid@gmail.com Password: Kk125556	User is successfully logged in.	User logged in successfully.
TC-FLP-002	Empty Email and Password	Email: empty Password: empty	User receives an error message indicating fields are required.	Error message displayed.
TC-FLP-003	Valid Email, Empty Password	Email: khalid@gmail.com Password: empty	User receives an error message indicating password is required.	Error message displayed.
TC-FLP-004	Empty Email, Valid Password	Email: empty Password: Kk125556	User receives an error message indicating email is required.	Error message displayed.
TC-FLP-005	Invalid Email Format	Email: khalid-email Password: Kk125556	An error message indicating an improper email format is delivered to the user.	Error message displayed.
TC-FLP-006	Incorrect Password	Email: khalid@gmail.com Password: wrong pass	User receives an error message indicating incorrect password.	Error message displayed.
TC-FLP-007	Valid Email, Check Me Out	Email: khalid@gmail.com Password: Kk125556 Check Me Out: Checked	User is successfully logged in and the "Check Me Out" preference is recorded.	User logged in successfully.
TC-FLP-008	Remember Me Unchecked	Email: khalid@gmail.com Password: Kk125556 Check Me Out: Unchecked	User is successfully logged in without remembering the session.	User logged in successfully.
TC-FLP-009	Invalid Email and Password	Email: invalid-email Password: wrongpass	User receives an error message indicating invalid email and password.	Error message displayed.
TC-FLP-010	Submit Form	form fields filled correctly	Form is successfully submitted for login.	Form submitted successfully.

Table 6.4.1 5: Page Name: Business Create Profile Test Cases:

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-BCP-001	Valid Profile Creation	valid data for all fields	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-002	Empty Fields	empty fields	User receives error messages for required fields.	Error messages displayed.
TC-BCP-003	Valid Company Website	Company Website: www.utem.com	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-004	Invalid Company Website Format	Company Website: invalid-website-format	User receives an error message indicating invalid website format.	Error message displayed.
TC-BCP-005	Valid No of Employees	No of Employees: 100	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-006	Invalid No of Employees Format	No of Employees: wrong-format	User receives an error message indicating invalid number format.	Error message displayed.
TC-BCP-007	Valid Established Year	Established Year: 2023	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-008	Invalid Established Year Format	Established Year: invalid-format	User receives an error message indicating invalid year format.	Error message displayed.
TC-BCP-009	Valid Company Status	Company Status: Choose...	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-010	Valid Company's Category	Company's Category: Choose...	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-011	Valid Major Clients	Major Clients: ABC Corp, XYZ Inc.	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-012	Empty Major Clients	Major Clients: empty	Profile is created successfully without errors (assuming it's optional).	Profile created successfully.
TC-BCP-013	Checkbox Agreement Checked	Acceptance of the conditions: Checked	Profile is created successfully without errors.	Profile created successfully.
TC-BCP-014	Checkbox Agreement Unchecked	Acceptance of the conditions: Unchecked	User receives an error message indicating agreement is required.	Error message displayed.

Table 6.4.1 6: Page Name: Post Jobs Page Test Cases

Case ID	Description	Inputs	Expected Output	Actual Output
TC-PJP-001	Valid Job Title	Job Title: Software Developer	Job is posted successfully without errors.	Job posted successfully.
TC-PJP-002	Empty Job Title	Job Title: empty	User receives an error message indicating the Job Title is required.	Error message displayed.
TC-PJP-003	Invalid Job Title Format	Job Title: $%&*@#	An error notice indicating an improper Job Title format is presented to the user.	Error message displayed.
TC-PJP-004	Valid Salary	Salary: RM70000	Salary is accepted without errors.	No errors observed.
TC-PJP-005	Empty Salary	Salary: empty	User receives an error message indicating the Salary is required.	Error message displayed.
TC-PJP-006	Invalid Salary Format	Salary: invalid	A notification alerting the user to an invalid salary format is received.	Error message displayed.
TC-PJP-007	Valid Duration in Months	6	Duration in Months is accepted without errors.	No errors observed.
TC-PJP-008	Empty Duration in Months	Duration in Months: empty	User receives an error message indicating Duration in Months is required.	Error message displayed.
TC-PJP-009	Invalid Duration in Months	-3	User receives an error message indicating an invalid Duration in Months.	Error message displayed.
TC-PJP-010	Valid Job Description	Job Description: Seeking a skilled...	Job Description is accepted without errors.	No errors observed.
TC-PJP-011	Empty Job Description	Job Description: empty	User receives an error message indicating the Job Description is required.	Error message displayed.

Table 6.4.1 7: Page Name: View My Jobs Test Cases:

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-VMJ-001	View Job List	existing job list	List of jobs is displayed successfully.	List of jobs displayed.
TC-VMJ-002	View Job Details	clickable job details	Job details are displayed successfully.	Job details displayed.
TC-VMJ-003	Interested Freelancers	clickable interested freelancers	List of interested freelancers is displayed.	List of freelancers displayed.
TC-VMJ-004	View Recommendations	recommendation list	List of recommended freelancers is displayed.	List of freelancers displayed.
TC-VMJ-005	Send Job Details to Freelancer	clickable "Send Job Details to Freelancer" button	Job details are sent to the freelancer successfully.	Job details sent successfully.

Table 6.4.1 8: Page Name: View All Jobs Test Cases

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-VAJ-001	View Job List	existing job list	List of jobs is displayed successfully.	List of jobs displayed.
TC-VAJ-002	View Business Details	clickable business details	Business details are displayed successfully.	Business details displayed.
TC-VAJ-003	Interested Button	clickable "Interested" button	User expresses interest in the job successfully.	Interest expressed.
TC-VAJ-004	View Freelancers Button	clickable "View Freelancers" button	List of freelancers is displayed.	List of freelancers displayed.
TC-VAJ-005	Edit Profile Button	clickable "Edit Profile" button	User is directed to the profile editing page.	Profile editing page displayed.

Table 6.4.1 9: Page Name: Create Profile Test Cases

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-CP-001	Create Profile	valid profile data	Profile is created successfully.	Profile created successfully.
TC-CP-002	Add Website/Portfolio	valid URL	Website/portfolio is added successfully.	URL added successfully.
TC-CP-003	Add Qualification	valid qualification	Qualification is added successfully.	Qualification added.
TC-CP-004	Add GitHub UserID	valid GitHub UserID	GitHub UserID is added successfully.	GitHub UserID added.
TC-CP-005	Choose Status	select valid status	Status is selected and saved successfully.	Status selected and saved.
TC-CP-006	Check Student Checkbox	click on "Student" checkbox	Student checkbox is checked successfully.	Checkbox checked.
TC-CP-007	Add Social Media Links	add valid social media links	Social media links are added successfully.	Links added successfully.
TC-CP-008	Agree to Terms and Conditions	click "Agree" checkbox	Terms and conditions are agreed to.	Checkbox checked.
TC-CP-009	Submit Profile	complete profile data	Profile is submitted successfully.	Profile submitted.

Table 6.4.1 10: Page Name: Add Education Test Cases

Test Case ID	Description	Inputs	Expected Output	Actual Output
TC-AE-001	Valid Course of Study	Computer Science	Course of Study is accepted without errors.	No errors observed.
TC-AE-002	Empty Course of Study	empty	User receives an error message indicating Course of Study is required.	Error message displayed.
TC-AE-003	Invalid Course of Study	Course of Study: 12345	User receives an error message indicating invalid Course of Study.	Error message displayed.
TC-AE-004	Valid University	University: Example University	University is accepted without errors.	No errors observed.
TC-AE-005	Empty University	University: empty	User receives an error message indicating University is required.	Error message displayed.
TC-AE-006	Invalid University	University: 12345	User receives an error message indicating invalid University.	Error message displayed.
TC-AE-007	Valid Location	Location: Melaka	Location is accepted without errors.	No errors observed.
TC-AE-008	Empty Location	Location: empty	User receives an error message indicating Location is required.	Error message displayed.
TC-AE-009	Invalid From Date Format	From Date: wrong-format	User receives an error message indicating invalid From Date format.	Error message displayed.
TC-AE-010	Valid From Date	From Date: 2020-09-01	From Date is accepted without errors.	No errors observed.
TC-AE-011	Empty From Date	From Date: empty	User receives an error message indicating From Date is required.	Error message displayed.
TC-AE-012	Invalid To Date Format	To Date: invalid-format	User receives an error message indicating invalid To Date format.	Error message displayed.
TC-AE-013	To Date Before From Date	From Date: 2022-06-30, To Date: 2020-03-15	User receives an error message indicating To Date must be after From Date.	Error message displayed.
TC-AE-014	Valid To Date	From Date: 2020-09-01, To Date: 2021-03-15	To Date is accepted without errors.	No errors observed.
TC-AE-015	Valid Course Description	Course Description: Studied various...	Course Description is accepted without errors.	No errors observed.
TC-AE-016	Empty Course Description	Course Description: empty	User receives an error message indicating Course Description is required.	Error message displayed.

6.4.2 Test Data

Table 6.4.2 1: Test Data

Data	Explanation
Test Data	Specific values and inputs are used to test an application's functionality.
Test Case ID	A unique identifier, such as TC-SPB-001, for tracking and reference.
Description	Explains briefly the goal of the test case and the functionality being tested.
Inputs	Actual test data for the test case, including example values.
Action	Sequence of steps to imitate a scenario during test case execution.
Expected Output	The expected outcome of the application using the available data and specified procedures.
Actual Output	Actual results are observed when the test case is run with the provided data and steps.
Status	The test case's current status, such as Pass, Fail, and so on..

6.5 Test Results and Analysis

Table 6.5 1: Page Name: Signup Page For Business

Test Case ID	Tester ID	Test Result	Detailed Documentation (if applicable)
TC-SPB-001	TST-001	Passed	No issues encountered. System correctly accepts valid Company Name.
TC-SPB-002	TST-001	Passed	No issues encountered. System correctly shows error message for missing Company Name.
TC-SPB-003	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Company Name.
TC-SPB-004	TST-001	Passed	No issues encountered. System correctly accepts valid Contact Name.
TC-SPB-005	TST-001	Passed	No issues encountered. System correctly shows error message for missing Contact Name.
TC-SPB-006	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Contact Name.
TC-SPB-007	TST-001	Passed	No issues encountered. System correctly accepts valid Contact's Email.
TC-SPB-008	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Contact's Email format.
TC-SPB-009	TST-001	Passed	No issues encountered. System correctly shows error message for missing Contact's Email.
TC-SPB-010	TST-001	Passed	No issues encountered. System correctly accepts valid Password.
TC-SPB-011	TST-001	Passed	No issues encountered. System correctly shows error message for missing Password.
TC-SPB-012	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Password.
TC-SPB-013	TST-001	Passed	No issues encountered. System correctly accepts valid Confirm Password.
TC-SPB-014	TST-001	Passed	No issues encountered. System correctly shows error message for mismatched Confirm Passwords.

TC-SPB-015	TST-001	Passed	No issues encountered. System correctly shows error message for missing Confirm Password.
TC-SPB-016	TST-001	Passed	No issues encountered. System correctly accepts valid Location.
TC-SPB-017	TST-001	Passed	No issues encountered. System correctly shows error message for missing Location.
TC-SPB-018	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Location.
TC-SPB-019	TST-001	Passed	No issues encountered. System correctly accepts valid Company's Description.
TC-SPB-020	TST-001	Passed	No issues encountered. System correctly accepts empty Company's Description.
TC-SPB-021	TST-001	Passed	No issues encountered. System detects and prevents invalid manipulation attempts of agreement checkbox.

Table 6.5 2: Page Name: Business Login Page

Test Case ID	Tester ID	Test Result	Detailed Documentation (if applicable)
TC-BLP-001	TST-001	Passed	No issues encountered. User successfully logged in and redirected to the relevant dashboard.
TC-BLP-002	TST-001	Passed	No issues encountered. User receives error messages indicating that both Email and Password fields are required.
TC-BLP-003	TST-001	Passed	There were no problems. The email address is necessary, according to the error message the user receives.
TC-BLP-004	TST-001	Passed	No problems were found. The user is informed via error message that the password is necessary.
TC-BLP-005	TST-001	Passed	There were no problems. An error message indicating an improper email format is delivered to the user.
TC-BLP-006	TST-001	Passed	No problems were found. An error notice informing the user that the login credentials are wrong is displayed.
TC-BLP-007	TST-001	Passed	There were no problems. The wrong login credentials are disclosed to the user via an error message.
TC-BLP-008	TST-001	Passed	No issues encountered. The checkbox status is remembered, and the user is logged in.
TC-BLP-009	TST-001	Passed	No issues encountered. The checkbox status is not remembered after logging in.
TC-BLP-010	TST-001	Passed	No issues encountered. User receives error messages indicating that both Email and Password fields are required.

Table 6.5 3: Page Name: Signup Page For Freelancer

Test Case ID	Tester ID	Test Result	Detailed Documentation
TC-SFF-001	TST-001	Passed	No issues encountered. System correctly accepts valid User Name.
TC-SFF-002	TST-001	Passed	No issues encountered. System correctly shows error message for missing User Name.
TC-SFF-003	TST-001	Passed	No issues encountered. System correctly shows error message for invalid User Name.
TC-SFF-004	TST-001	Passed	No issues encountered. System correctly accepts valid First Name.
TC-SFF-005	TST-001	Passed	No issues encountered. System correctly shows error message for missing First Name.
TC-SFF-006	TST-001	Passed	No issues encountered. System correctly shows error message for invalid First Name.
TC-SFF-007	TST-001	Passed	No issues encountered. System correctly accepts valid Last Name.
TC-SFF-008	TST-001	Passed	No issues encountered. System correctly shows error message for missing Last Name.
TC-SFF-009	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Last Name.
TC-SFF-010	TST-001	Passed	No issues encountered. System correctly accepts valid Email.
TC-SFF-011	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Email format.
TC-SFF-012	TST-001	Passed	No issues encountered. System correctly shows error message for missing Email.
TC-SFF-013	TST-001	Passed	No issues encountered. System correctly accepts valid Personal Email.
TC-SFF-014	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Personal Email format.
TC-SFF-015	TST-001	Passed	No issues encountered. System correctly accepts empty Personal Email.
TC-SFF-016	TST-001	Passed	No issues encountered. System correctly accepts valid Password.
TC-SFF-017	TST-001	Passed	No issues encountered. System correctly shows error message for missing Password.
TC-SFF-018	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Password.
TC-SFF-019	TST-001	Passed	No issues encountered. System correctly accepts valid Confirm Password.
TC-SFF-020	TST-001	Passed	No issues encountered. System correctly shows error message for mismatched Confirm Password.
TC-SFF-021	TST-001	Passed	No issues encountered. System correctly shows error message for missing Confirm Password.
TC-SFF-022	TST-001	Passed	No issues encountered. System correctly accepts valid LinkedIn Profile.
TC-SFF-023	TST-001	Passed	No issues encountered. System correctly shows error message for invalid LinkedIn Profile.
TC-SFF-024	TST-001	Passed	No issues encountered. System correctly accepts empty LinkedIn Profile.
TC-SFF-025	TST-001	Passed	No issues encountered. System correctly accepts valid Location.
TC-SFF-026	TST-001	Passed	No issues encountered. System correctly shows error message for missing Location.
TC-SFF-027	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Location.

TC-SFF-028	TST-001	Passed	No issues encountered. System correctly accepts valid Age.
TC-SFF-029	TST-001	Passed	No issues encountered. System correctly accepts empty Age.
TC-SFF-030	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Age.
TC-SFF-031	TST-001	Passed	No issues encountered. System correctly accepts valid self-description.
TC-SFF-032	TST-001	Passed	No issues encountered. System correctly shows error message for missing self-description.
TC-SFF-033	TST-001	Passed	No issues encountered. System correctly accepts Agreement.
TC-SFF-034	TST-001	Passed	No issues encountered. System correctly shows error message for missing Agreement.
TC-SFF-035	TST-001	Passed	No issues encountered. Form is successfully submitted.

Table 6.5 4: Page Name: Freelancer Login Page

Test Case ID	Tester ID	Test Result	Detailed Documentation
TC-FLP-001	TST-001	Passed	No issues encountered. System successfully logged in with valid email and password.
TC-FLP-002	TST-001	Passed	No issues encountered. System correctly shows error message for missing email and password.
TC-FLP-003	TST-001	Passed	No issues encountered. System correctly shows error message for missing password with valid email.
TC-FLP-004	TST-001	Passed	No issues encountered. System correctly shows error message for missing email with valid password.
TC-FLP-005	TST-001	Passed	No issues encountered. System correctly shows error message for invalid email format.
TC-FLP-006	TST-001	Passed	No issues encountered. System correctly shows error message for incorrect password.
TC-FLP-007	TST-001	Passed	No issues encountered. System successfully logged in and remembered session.
TC-FLP-008	TST-001	Passed	No issues encountered. System successfully logged in without remembering session.
TC-FLP-009	TST-001	Passed	No issues encountered. System correctly shows error message for invalid email and password.
TC-FLP-010	TST-001	Passed	No issues encountered. Form is successfully submitted.

Table 6.5 5: Module: Business Create Profile

Test Case ID	Tester ID	Test Result	Detailed Documentation (if applicable)
TC-BCP-001	TST-001	Passed	No issues encountered. Profile created successfully.
TC-BCP-002	TST-001	Passed	No issues encountered. User receives error messages for required fields.
TC-BCP-003	TST-002	Passed	No issues encountered. Profile created successfully.
TC-BCP-004	TST-002	Passed	No issues encountered. User receives an error message for invalid format.
TC-BCP-005	TST-003	Passed	No issues encountered. Profile created successfully.
TC-BCP-006	TST-003	Passed	No issues encountered. User receives an error message for invalid format.
TC-BCP-007	TST-004	Passed	No issues encountered. Profile created successfully.
TC-BCP-008	TST-004	Passed	No issues encountered. User receives an error message for invalid format.
TC-BCP-009	TST-005	Passed	No issues encountered. Profile created successfully.
TC-BCP-010	TST-005	Passed	No issues encountered. Profile created successfully.
TC-BCP-011	TST-006	Passed	No issues encountered. Profile created successfully.
TC-BCP-012	TST-006	Passed	No issues encountered. Profile created successfully.
TC-BCP-013	TST-007	Passed	No issues encountered. Profile created successfully.
TC-BCP-014	TST-007	Passed	No issues encountered. User receives an error message for agreement.

Table 6.5 6: Page Name: Post Jobs Page

Test Case ID	Tester ID	Test Result	Detailed Documentation
TC-PJP-001	TST-001	Passed	No issues encountered. System successfully posts job with valid Job Title.
TC-PJP-002	TST-001	Passed	No issues encountered. System correctly shows error message for missing Job Title.
TC-PJP-003	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Job Title format.
TC-PJP-004	TST-001	Passed	No issues encountered. System successfully posts job with valid Salary.
TC-PJP-005	TST-001	Passed	No issues encountered. System correctly shows error message for missing Salary.
TC-PJP-006	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Salary format.
TC-PJP-007	TST-001	Passed	No issues encountered. System successfully posts job with valid Duration in Months.
TC-PJP-008	TST-001	Passed	No issues encountered. System correctly shows error message for missing Duration in Months.
TC-PJP-009	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Duration in Months.
TC-PJP-010	TST-001	Passed	No issues encountered. System successfully posts job with valid Job Description.
TC-PJP-011	TST-001	Passed	No issues encountered. System correctly shows error message for missing Job Description.

Table 6.5 7: Module: View My Jobs

Test Case ID	Tester ID	Test Result	Detailed Documentation (if applicable)
TC-VMJ-001	TST-001	Passed	No issues encountered. List of jobs displayed successfully.
TC-VMJ-002	TST-001	Passed	No issues encountered. Job details displayed successfully.
TC-VMJ-003	TST-002	Passed	No issues encountered. List of freelancers displayed successfully.
TC-VMJ-004	TST-002	Passed	No issues encountered. List of recommended freelancers displayed.
TC-VMJ-005	TST-003	Passed	No issues encountered. Job details sent to freelancer successfully.

Table 6.5 8: Module: Create Profile

Test Case ID	Tester ID	Test Result	Detailed Documentation (if applicable)
TC-CP-001	TST-001	Passed	No issues encountered. Profile created successfully.
TC-CP-002	TST-001	Passed	No issues encountered. Website/portfolio URL added successfully.
TC-CP-003	TST-002	Passed	No issues encountered. Qualification added successfully.
TC-CP-004	TST-002	Passed	No issues encountered. GitHub UserID added successfully.
TC-CP-005	TST-003	Passed	No issues encountered. Status selected and saved successfully.
TC-CP-006	TST-003	Passed	No issues encountered. Student checkbox checked successfully.
TC-CP-007	TST-004	Passed	No issues encountered. Social media links added successfully.
TC-CP-008	TST-004	Passed	No issues encountered. Terms and conditions agreed to successfully.
TC-CP-009	TST-001	Passed	No issues encountered. Profile submitted successfully.

Table 6.5 9: Module: View All Jobs

Test Case ID	Tester ID	Test Result	Detailed Documentation (if applicable)
TC-VAJ-001	TST-001	Passed	No issues encountered. List of jobs displayed successfully.
TC-VAJ-002	TST-001	Passed	No issues encountered. Business details displayed successfully.
TC-VAJ-003	TST-002	Passed	No issues encountered. Interest expressed successfully.
TC-VAJ-004	TST-002	Passed	No issues encountered. List of freelancers displayed successfully.
TC-VAJ-005	TST-003	Passed	No issues encountered. Profile editing page displayed successfully.

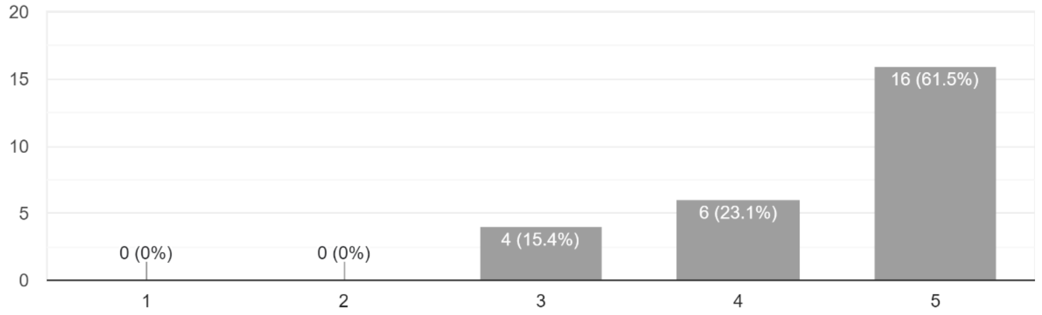
Table 6.5 10: Page Name: Add Education

Test Case ID	Tester ID	Test Result	Detailed Documentation
TC-AE-001	TST-001	Passed	No issues encountered. System successfully adds education with valid Course of Study.
TC-AE-002	TST-001	Passed	No issues encountered. System correctly shows error message for missing Course of Study.
TC-AE-003	TST-001	Passed	No issues encountered. System correctly shows error message for invalid Course of Study.
TC-AE-004	TST-001	Passed	No issues encountered. System successfully adds education with valid University.
TC-AE-005	TST-001	Passed	No issues encountered. System correctly shows error message for missing University.
TC-AE-006	TST-001	Passed	No issues encountered. System correctly shows error message for invalid University.
TC-AE-007	TST-001	Passed	No issues encountered. System successfully adds education with valid Location.
TC-AE-008	TST-001	Passed	No issues encountered. System correctly shows error message for missing Location.
TC-AE-009	TST-001	Passed	No issues encountered. System correctly shows error message for invalid From Date format.
TC-AE-010	TST-001	Passed	No issues encountered. System successfully adds education with valid From Date.
TC-AE-011	TST-001	Passed	No issues encountered. System correctly shows error message for missing From Date.
TC-AE-012	TST-001	Passed	No issues encountered. System correctly shows error message for invalid To Date format.
TC-AE-013	TST-001	Passed	No issues encountered. System correctly shows error message for To Date before From Date.
TC-AE-014	TST-001	Passed	No issues encountered. System successfully adds education with valid To Date greater than From Date.
TC-AE-015	TST-001	Passed	No issues encountered. System successfully adds education with valid Course Description.
TC-AE-016	TST-001	Passed	No issues encountered. System correctly shows error message for missing Course Description.

6.6 Usability Test

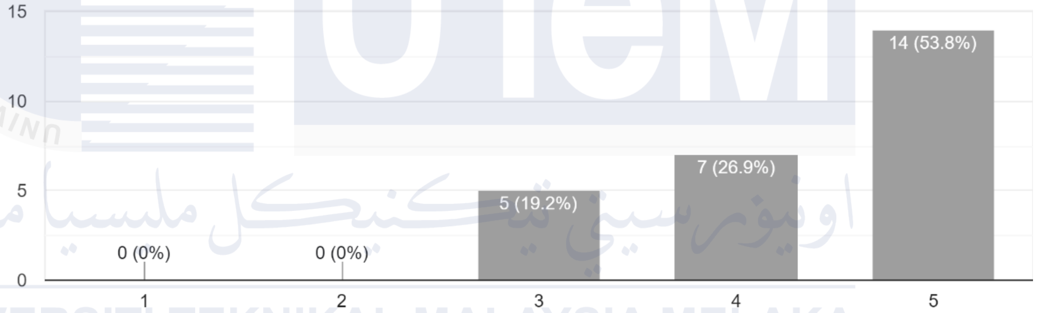
Overall, how would you rate your experience using Job Recommender System ?

26 responses



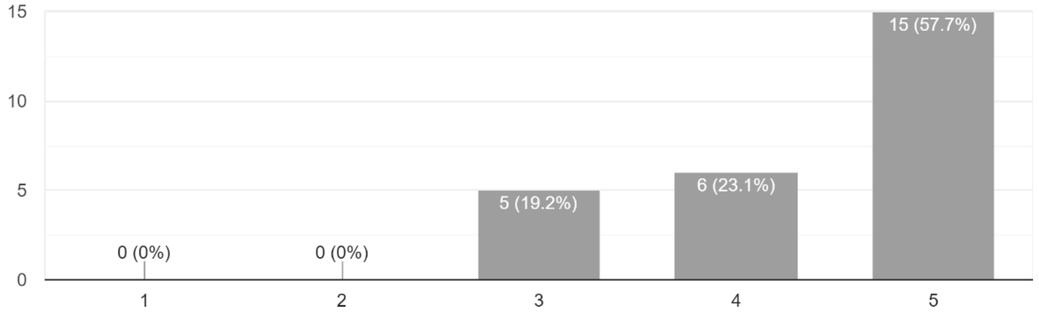
How challenging or confusing were the specific tasks or features of the "Job Recommender System"?

26 responses

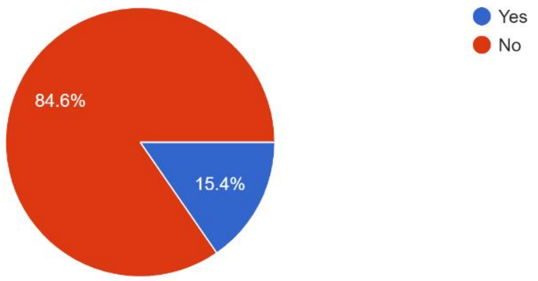


How much did you like the aspects of the "Job Recommender System"?

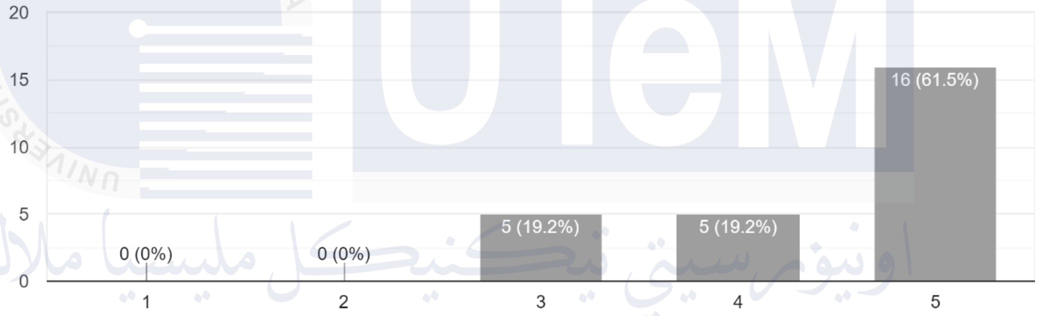
26 responses



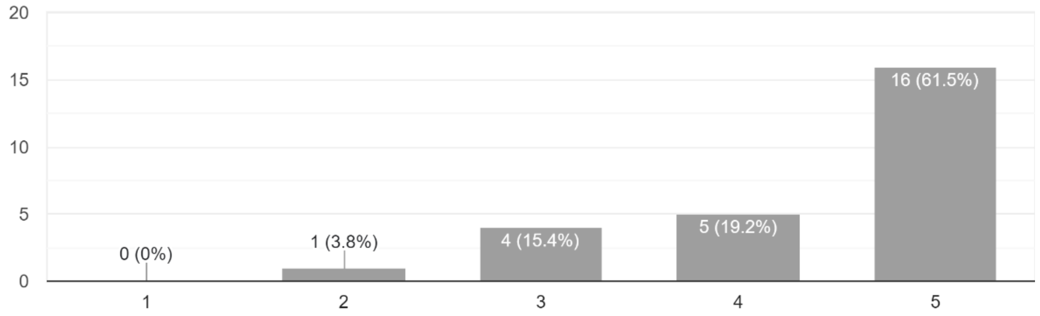
Did you encounter any technical issues or glitches while using the platform?
26 responses

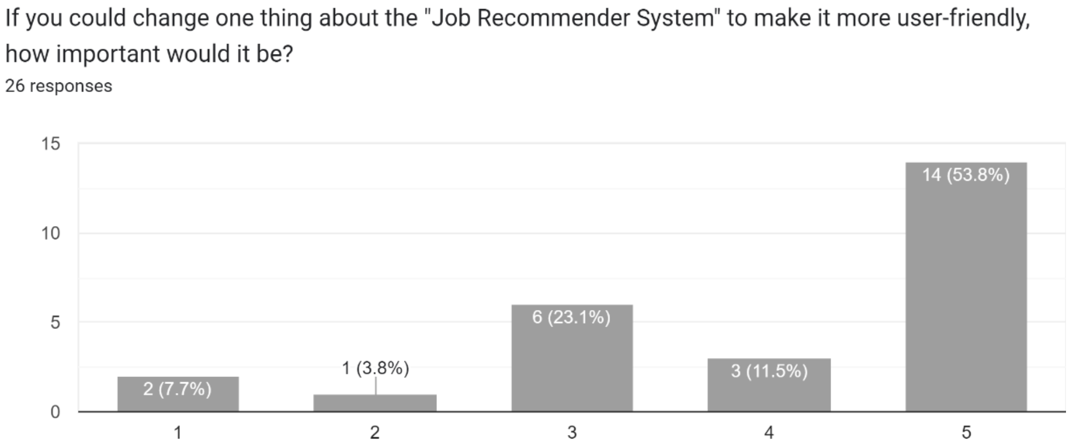


How well did the instructions provided help you complete the tasks on the "Job Recommender System"?
26 responses



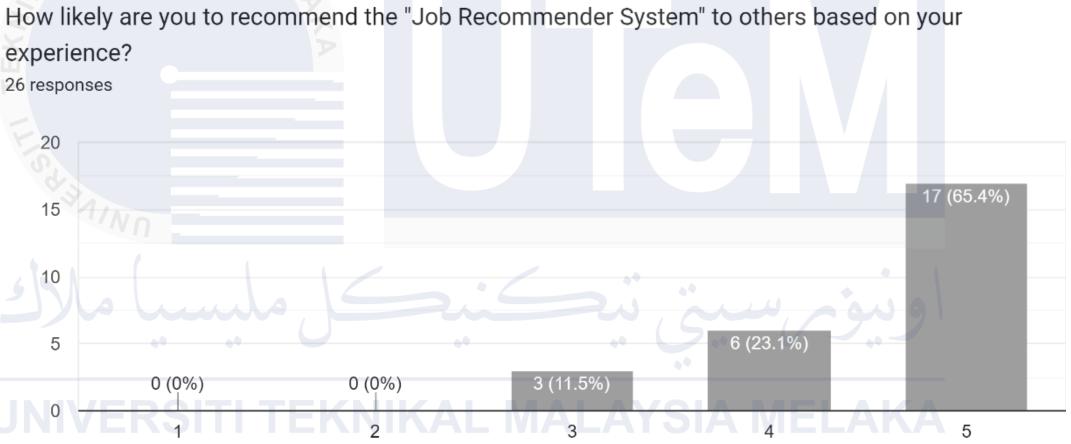
How clear and intuitive was the user interface of the "Job Recommender System"?
26 responses





If you could change one thing about the "Job Recommender System" to make it more user-friendly, how important would it be?

26 responses



How likely are you to recommend the "Job Recommender System" to others based on your experience?

26 responses

6.7 Conclusion

This chapter covered our testing plan, organization, test environment, and schedule. We emphasized black-box testing to assess functionality, security, and user experience across scenarios, aiming to uncover issues, enhance performance, and align with user expectations, ultimately delivering a robust end product. Next, we'll write the conclusion of this project.

CHAPTER 7: PROJECT CONCLUSION

7.1 Observation on Weaknesses and Strengths

7.1.1 Strengths:

The "Job Recommender System" fills a critical gap by linking businesses with qualified freelancers and providing students with hands-on experience. This platform's strengths lie in its ability to create profiles, showcase skills, and facilitate communication between freelancers and businesses. The system's suggestion component offers value by recommending freelancers who have the exact skill sets needed for specific jobs.

7.1.2 Weaknesses:

Possible challenges in policing the caliber of freelancers' work and verifying their claims of ability are among the system's shortcomings. It is crucial to guarantee precise talent matching and avoid misrepresentation. Furthermore, communication could be restricted to email and lack real-time engagement capabilities. It's critical to strike a balance between user friendliness and strong security measures. Data security and privacy concerns can arise, especially in light of the sensitive information involved. To safeguard user data, certain security precautions must be taken.

7.2 Propositions for Improvement

1. Streamlined Communication Tools: To speed up conversations and improve cooperation, integrate real-time communication tools like chat, audio calls, and video calls into the platform. Discussion boards created specifically for a project help centralize discussions and maintain interest among all parties.

2. Resources for Continuous Learning: Include a learning hub that integrates industry webinars, courses, and skill-building activities into the platform to support freelancer growth and provide firms with knowledgeable personnel.

3. Personalized Job Alerts: Use a system that notifies freelancers of pertinent possibilities based on their profiles and interests to maximize engagement and efficiency in locating acceptable tasks.

7.3 Project Contribution

7.3.1 Project Contribution:

My involvement in the project is limited to the creation and implementation of the "Job Recommender System." I worked on the system architecture design, user profile development, authentication mechanisms, communication feature integration, and talent matching algorithm optimization. I also played a significant part in developing data privacy policies and putting the rating and review system in place. My work has helped close the gap between corporations and freelancers by offering a useful platform for effective teamwork and fruitful project outputs.

7.3.2 User Manual:

The user manual for the "Job Recommender System" can be found in Appendix of the project documentation and provides comprehensive instructions on how to use it. This manual offers detailed instructions on how to use the system's essential features, including registration, profile creation, job posting, skill verification, communication tools, and others. It aims to assist companies and freelancers contractors in maximizing the platform's advantages and functionalities.

7.4 Conclusion

In conclusion, this project successfully achieves its objective of creating a web platform that connects businesses with freelancers. Thorough exploration of cutting-edge web technologies and a focus on the MERN stack led to a robust application. The documentation provides a comprehensive guide to MERN stack development, ensuring that anyone can build similar platforms. This project's success lies in its ability to bridge the gap between businesses and freelancers, offering an intuitive and impactful collaboration platform.

REFERENCES

Alam, S. M., Hasan, A. R., & Borman, T. (2021). IT Freelancing in Bangladesh: Assessment of Present Status and Future Needs. Journal of Economics and Business, 4(1).

Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. J Big Data 9, 59 (2022). <https://doi.org/10.1186/s40537-022-00592-5>

Goldberg, K., Roeder, T., Gupta, D. et al. Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval 4, 133–151 (2001). <https://doi.org/10.1023/A:1011419012209>

Z. I. Paramarini Hardianto and Karmilasari, "Analysis and Design of User Interface and User Experience (UI / UX) E-Commerce Website PT Pentasada Andalan Kelola Using Task System Centered Design (TCSD) Method," 2019 Fourth International Conference on Informatics and Computing (ICIC), Semarang, Indonesia, 2019, pp. 1-8, doi: 10.1109/ICIC47613.2019.8985854.

Nikulchev, E., Ilin, D., & Gusev, A. (2021). Technology stack selection model for software design of digital platforms. Mathematics, 9(4), 308.

Mehra, M., Kumar, M., Maurya, A., & Sharma, C. (2021). Mern stack web development. Annals of the Romanian Society for Cell Biology, 25(6), 11756-11761.

BIBLIOGRAPHY

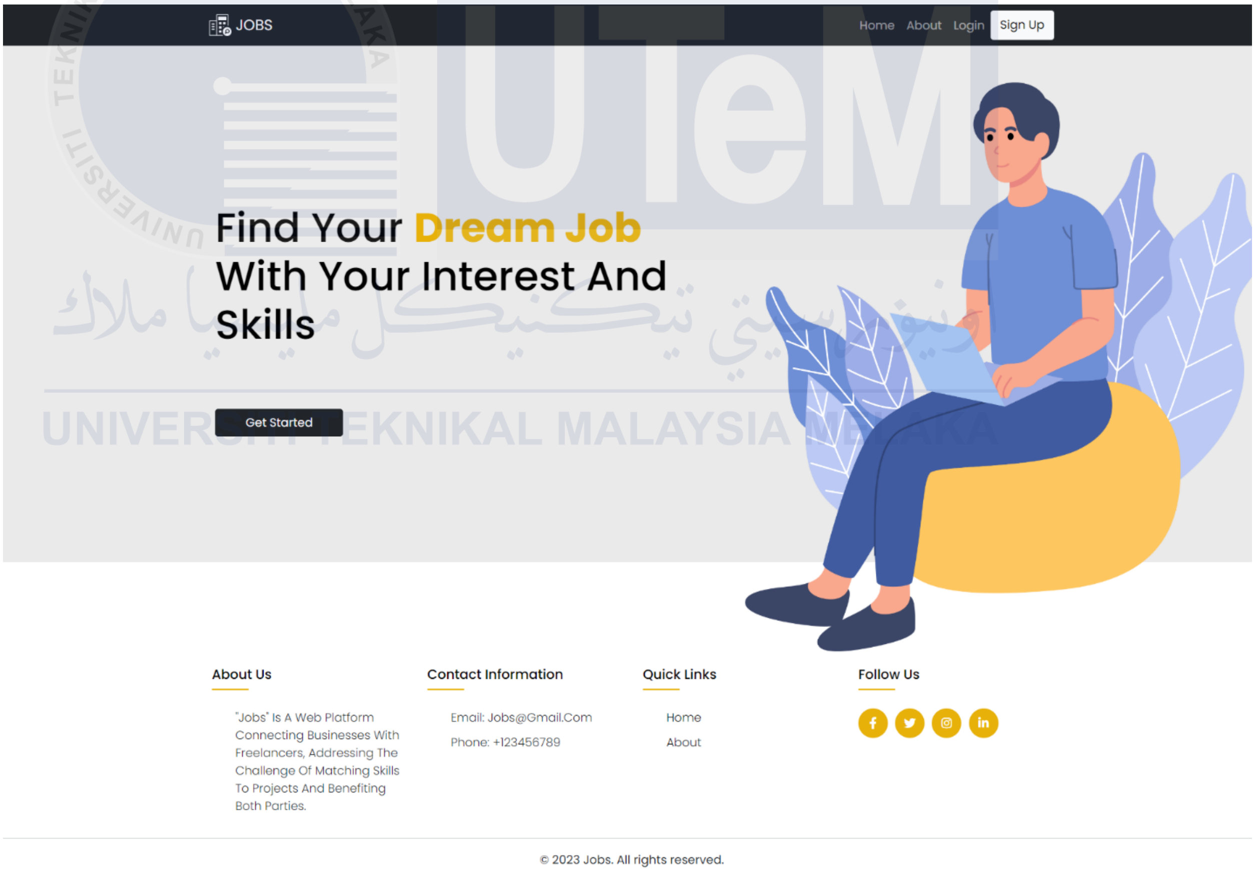
Bloesch, A. (2019). Overview of Verification and Validation for Embedded Software in Medical Systems. In Handbook of Medical Device Design (pp. 579-600). CRC Press.

Ismail, S. S., Mansour, R. F., El-Aziz, A., Rasha, M., & Taloba, A. I. (2022). Efficient E-mail spam detection strategy using genetic decision tree processing with NLP features. Computational Intelligence and Neuroscience, 2022.

Wu, Z., Li, C., Cao, J., & Ge, Y. (2020). On scalability of association-rule-based recommendation: A unified distributed-computing framework. ACM Transactions on the Web (TWEB), 14(3), 1-21.

APPENDICES

THE USER MANUAL FOR THE "JOB RECOMMENDER SYSTEM"



Find Your Dream Job With Your Interest And Skills

About Us

"Jobs" Is A Web Platform Connecting Businesses With Freelancers, Addressing The Challenge Of Matching Skills To Projects And Benefiting Both Parties.

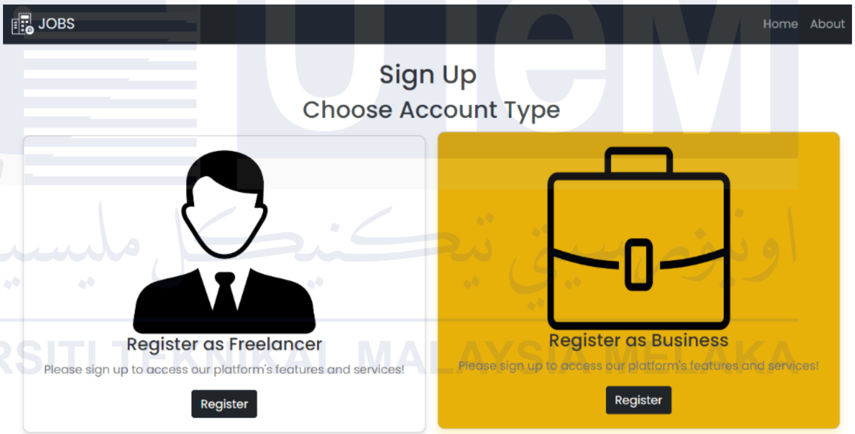
Contact Information

Email: Jobs@Gmail.Com

Phone: +123456789

© 2023 Jobs. All rights reserved.

Figure 1: Homepage



Sign Up

Choose Account Type

Register as Freelancer

Please sign up to access our platform's features and services!

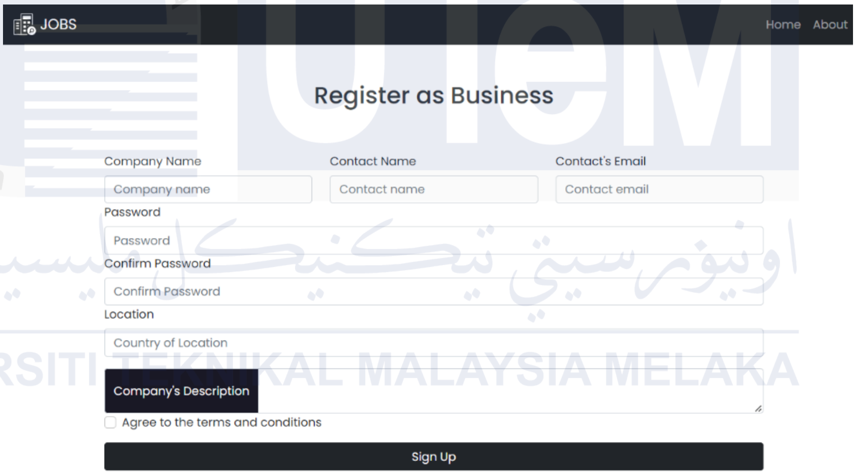
Register

Register as Business

Please sign up to access our platform's features and services!

Register

Figure 2: Signup page



JOBS

Home About

Register as Business

Company Name: Company name

Contact Name: Contact name

Contact's Email: Contact email

Password: Password

Confirm Password: Confirm Password

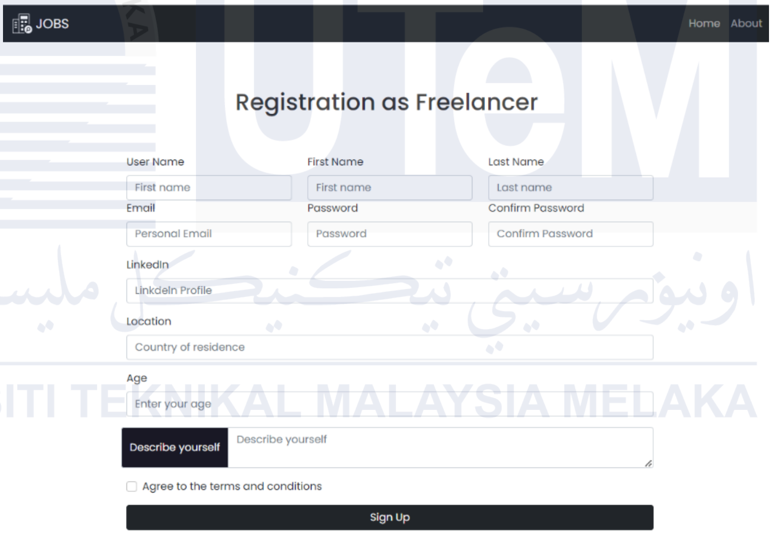
Location: Country of Location

Company's Description

□ Agree to the terms and conditions

Sign Up

Figure 3: Signup page as Business



Registration as Freelancer

User Name	First Name	Last Name
First name	First name	Last name
Email	Password	Confirm Password
Personal Email	Password	Confirm Password

LinkedIn

LinkedIn Profile

Location

Country of residence

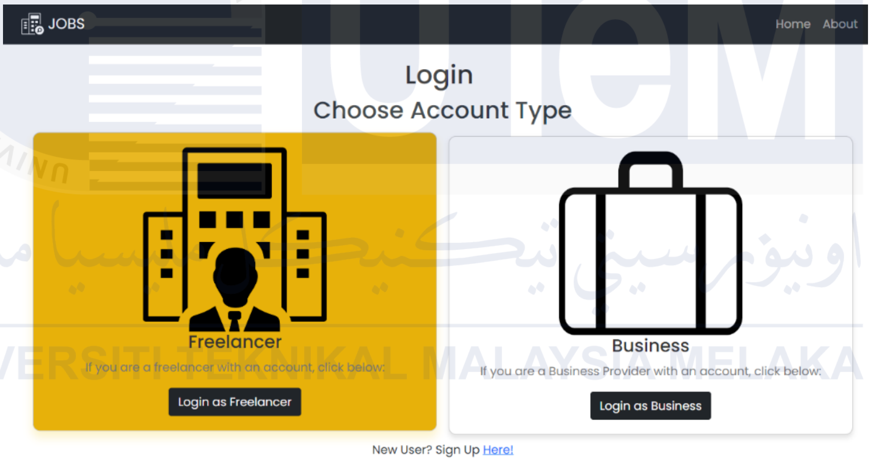
Age

Enter your age

Describe yourself: Describe yourself

□ Agree to the terms and conditions

Figure 4: Signup page as Freelancer



JOBS

Home About

Login

Choose Account Type

Freelancer

If you are a freelancer with an account, click below:

Login as Freelancer

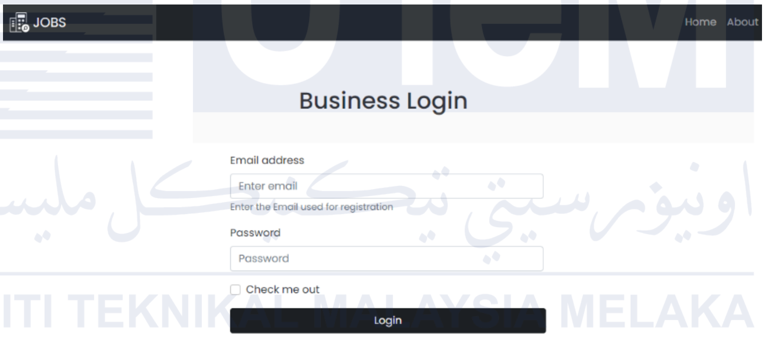
Business

If you are a Business Provider with an account, click below:

Login as Business

New User? Sign Up Here!

Figure 5: Login page



Business Login

Email address

Enter email

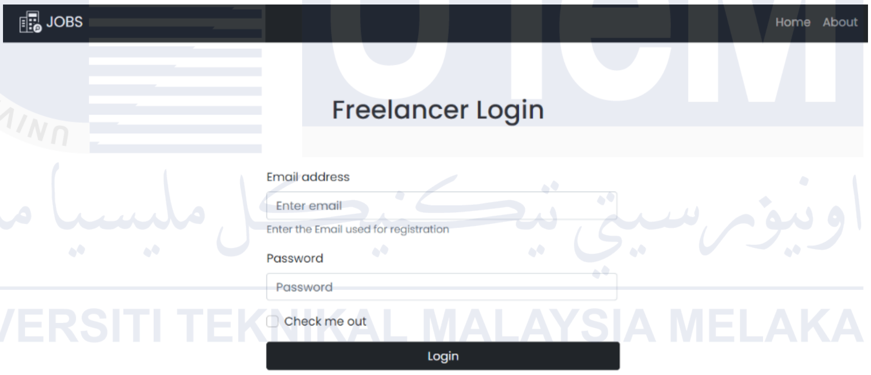
Enter the Email used for registration

Password

Password

□ Check me out

Figure 6: Login page us Business



Freelancer Login

Email address

Enter email

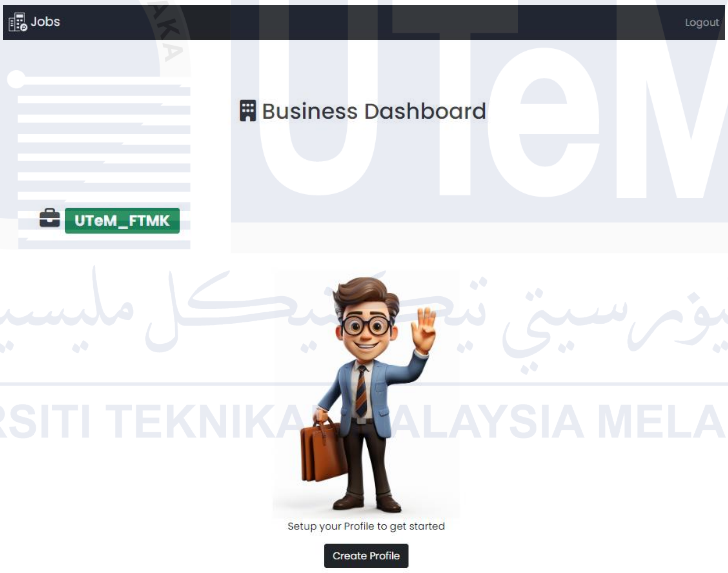
Enter the Email used for registration

Password

Password

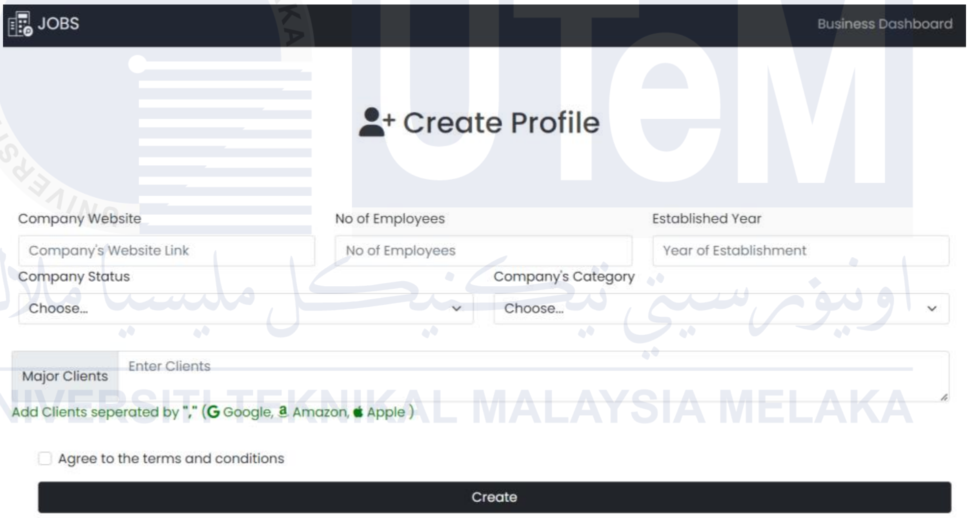
□ Check me out

Figure 7: Login page us Freelancer



Business Dashboard

Setup your Profile to get started



JOBS

Business Dashboard

Create Profile

Company Website

Company's Website Link

No of Employees

No of Employees

Established Year

Year of Establishment

Company Status

Choose...

Company's Category

Choose...

Major Clients

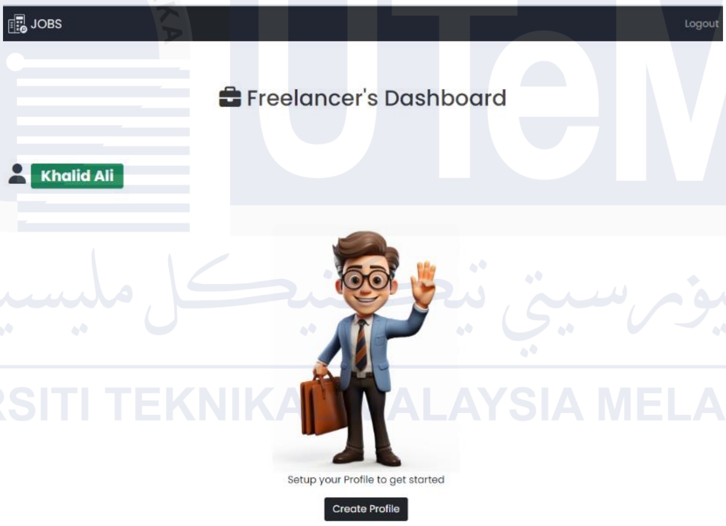
Enter Clients

Add Clients seperated by "," (Google, Amazon, Apple)

□ Agree to the terms and conditions

Create

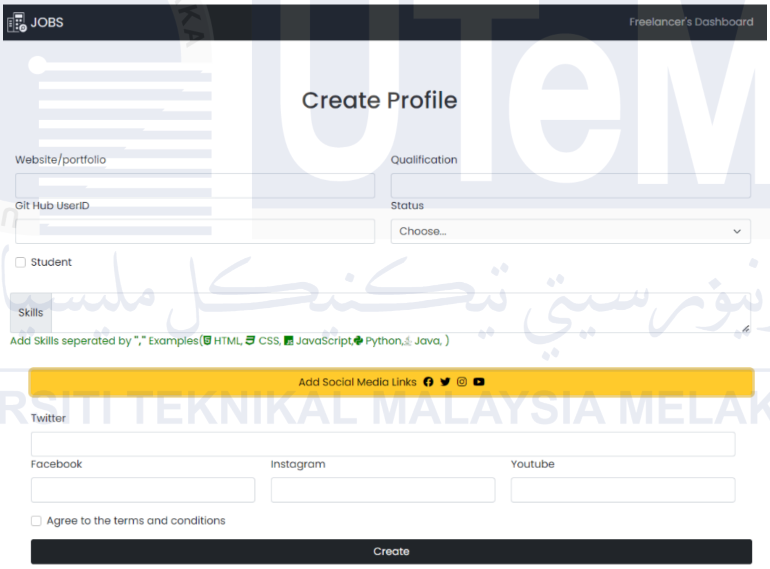
Figure 8: Create Profile Page For Business



Freelancer's Dashboard

Khalid Ali

Setup your Profile to get started



JOBS

Freelancer's Dashboard

Create Profile

Website/portfolio

Qualification

Git Hub UserID

Status

Choose...

□ Student

Skills

Add Skills seperated by "," Examples(HTML, CSS, JavaScript, Python, Java,)

Add Social Media Links

Twitter

Facebook

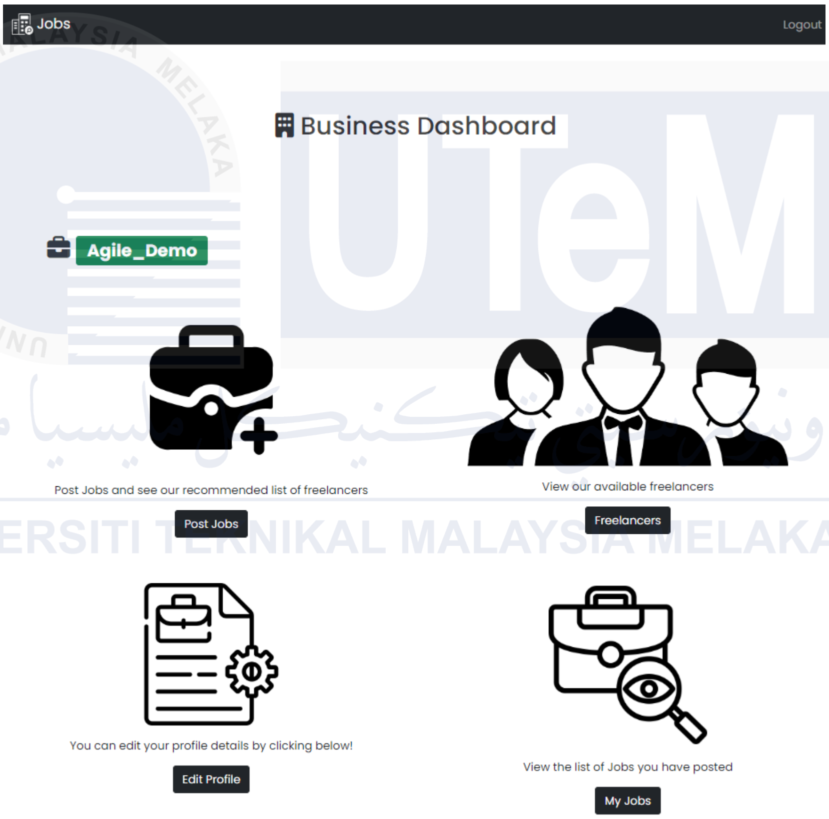
Instagram

Youtube

□ Agree to the terms and conditions

Create

Figure 9: Create Profile Page For Freelancers



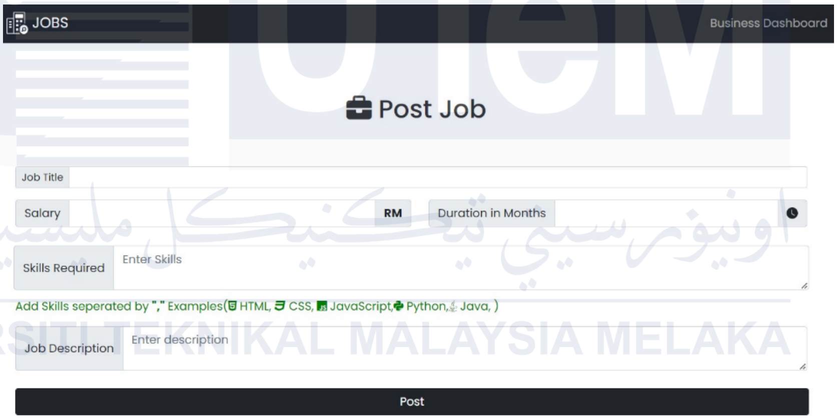
Post Jobs and see our recommended list of freelancers

View our available freelancers

You can edit your profile details by clicking below!

View the list of Jobs you have posted

Figure 10: Business Dashboard



JOBS

Business Dashboard

Post Job

Job Title

Salary RM

Duration in Months

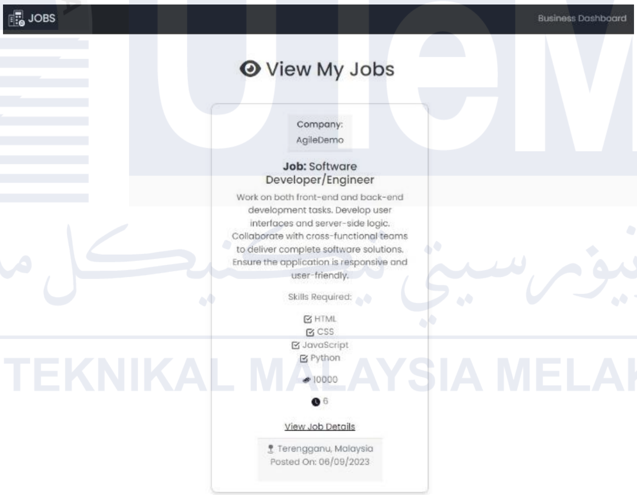
Skills Required: Enter Skills

Add Skills seperated by "," Examples(HTML, CSS, JavaScript, Python, Java,)

Job Description: Enter description

Post

Figure 11 : Post Job



View My Jobs

Company:
AgileDemo

Job: Software Developer/Engineer

Work on both front-end and back-end development tasks. Develop user interfaces and server-side logic. Collaborate with cross-functional teams to deliver complete software solutions. Ensure the application is responsive and user-friendly.

Skills Required:

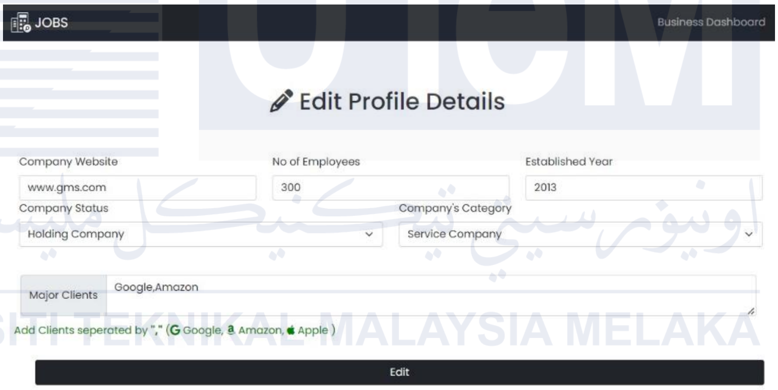
☑ HTML
☑ CSS
☑ JavaScript
☑ Python

10000

6

Terengganu, Malaysia
Posted On: 06/09/2023

Figure 12 : View My jobs



Edit Profile Details

Company Website: www.gms.com

No of Employees: 300

Established Year: 2013

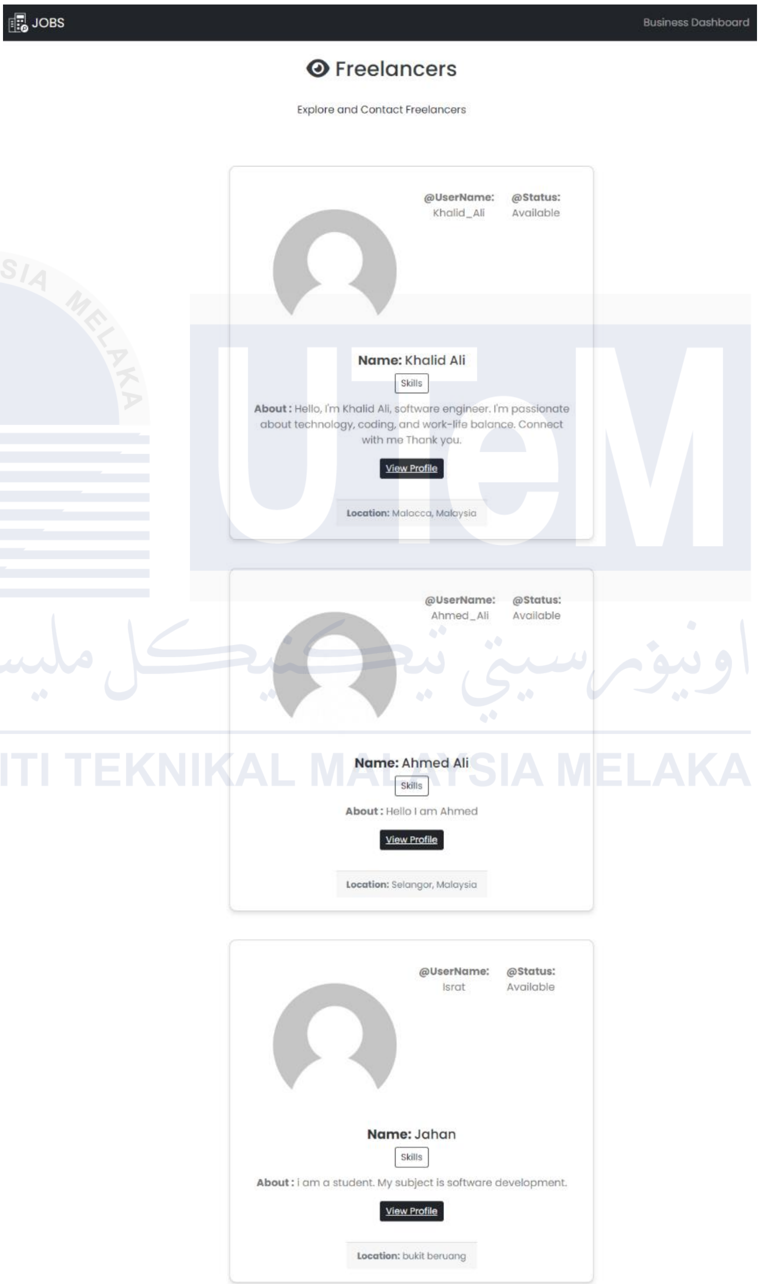
Company Status: Holding Company

Company's Category: Service Company

Major Clients: Google,Amazon

Add Clients seperated by "," (Google, Amazon, Apple)

Figure 13: Edit Job Profile Details



Freelancers

Explore and Contact Freelancers

@UserName: Khalid_Ali **@Status:** Available

Name: Khalid Ali

About : Hello, I'm Khalid Ali, software engineer. I'm passionate about technology, coding, and work-life balance. Connect with me Thank you.

Location: Malocca, Malaysia

@UserName: Ahmed_Ali **@Status:** Available

Name: Ahmed Ali

About : Hello I am Ahmed

Location: Selangor, Malaysia

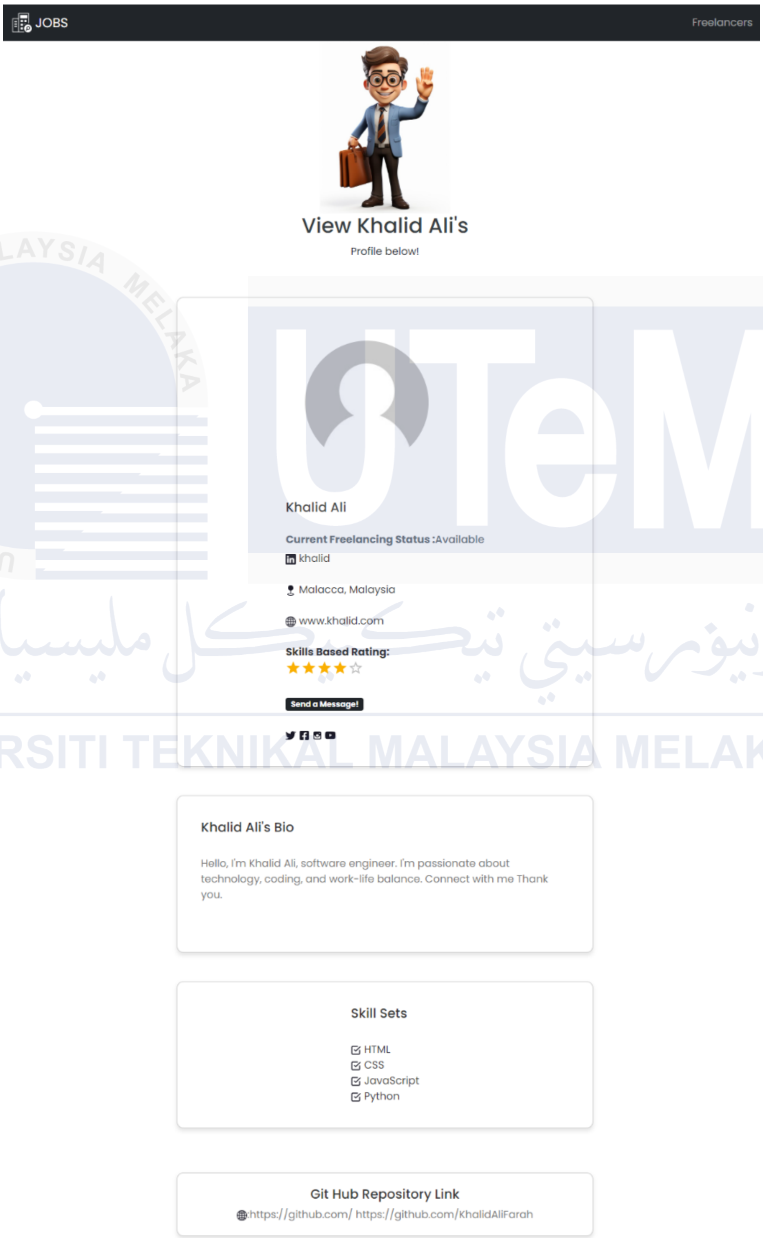
@UserName: Israt **@Status:** Available

Name: Jahan

About : i am a student. My subject is software development.

Location: bukit beruang

Figure 14: View Freelancers in business side



View Khalid Ali's

Profile below!

Khalid Ali

Current Freelancing Status : Available

khalid

Malacca, Malaysia

www.khalid.com

Skills Based Rating:

★★★★☆

Khalid Ali's Bio

Hello, I'm Khalid Ali, software engineer. I'm passionate about technology, coding, and work-life balance. Connect with me Thank you.

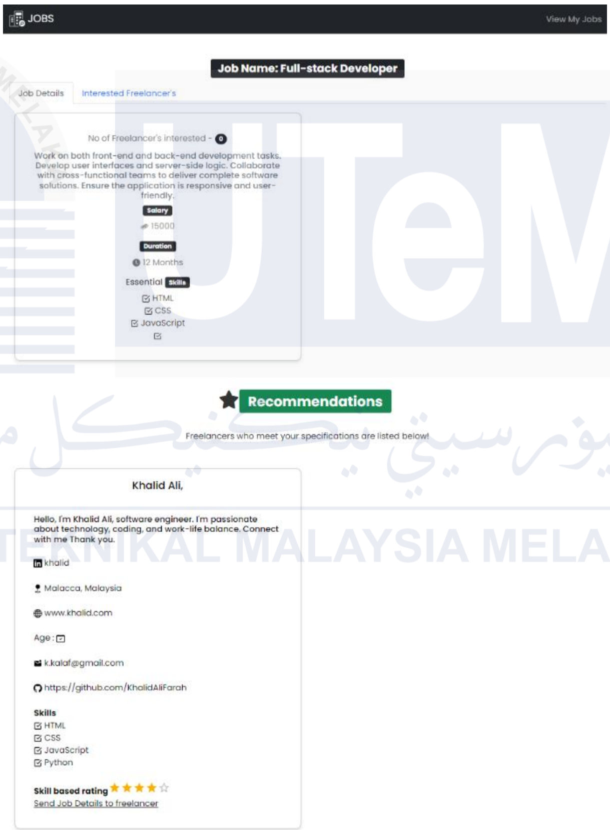
Skill Sets

☑ HTML
☑ CSS
☑ JavaScript
☑ Python

Git Hub Repository Link

:https://github.com/ https://github.com/KhalidAliFarah

Figure 15: Freelancers View Profile



Job Name: Full-stack Developer

No of Freelancer's interested - 0

Work on both front-end and back-end development tasks. Develop user interfaces and server-side logic. Collaborate with cross-functional teams to deliver complete software solutions. Ensure the application is responsive and user-friendly.

Salary

15000

Duration

12 Months

Essential Skills

☑ HTML
☑ CSS
☑ JavaScript
☑

★ Recommendations

Freelancers who meet your specifications are listed below!

Khalid Ali,

Hello, I'm Khalid Ali, software engineer. I'm passionate about technology, coding, and work-life balance. Connect with me Thank you.

khalid

Malacca, Malaysia

www.khalid.com

Age :

k.kalaf@gmail.com

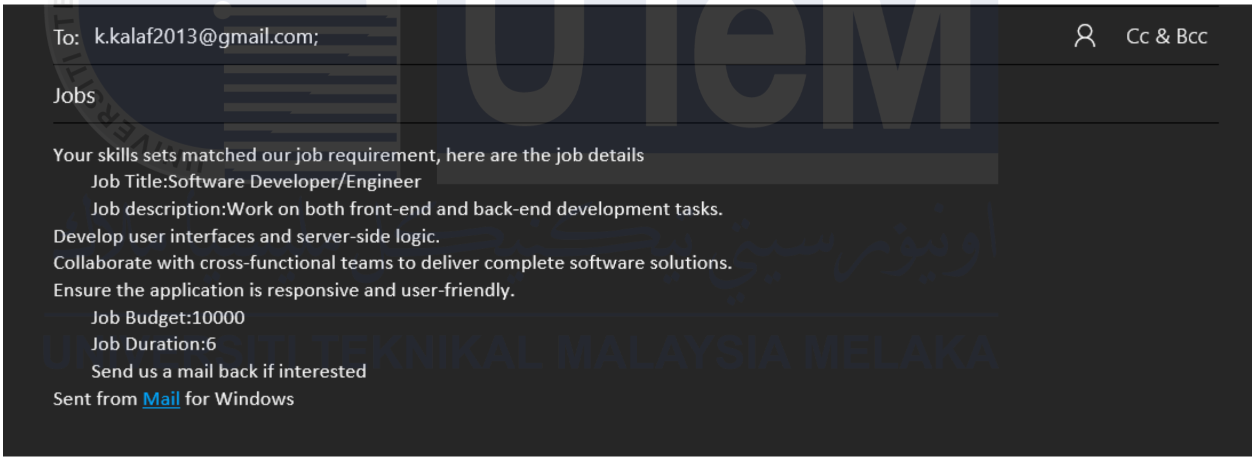
https://github.com/KhalidAliFarah

Skills
☑ HTML
☑ CSS
☑ JavaScript
☑ Python

Skill based rating ★★★★☆

Send Job Details to freelancer

Figure 16: Recommendations



To: k.kalaf2013@gmail.com;

Jobs

Your skills sets matched our job requirement, here are the job details

Job Title:Software Developer/Engineer

Job description:Work on both front-end and back-end development tasks.

Develop user interfaces and server-side logic.

Collaborate with cross-functional teams to deliver complete software solutions.

Ensure the application is responsive and user-friendly.

Job Budget:10000

Job Duration:6

Send us a mail back if interested

Sent from Mail for Windows

Figure 17: Send job Details To Freelancers by Email

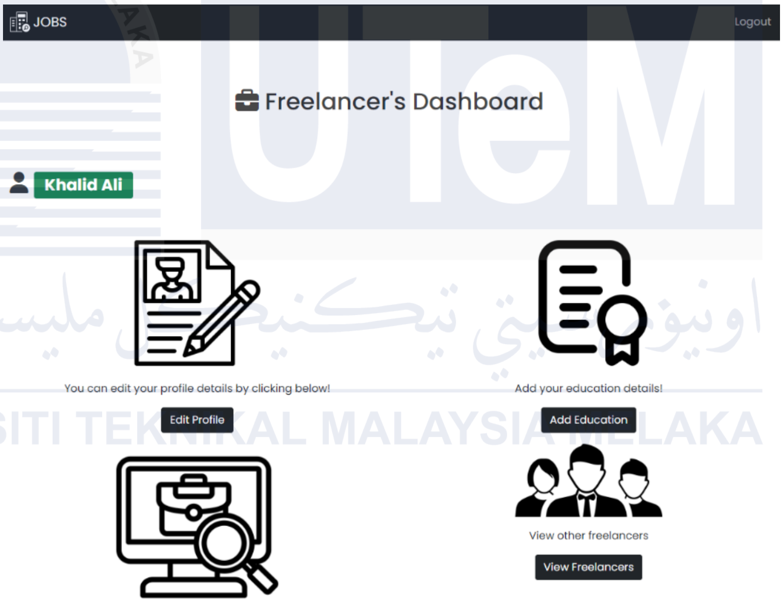
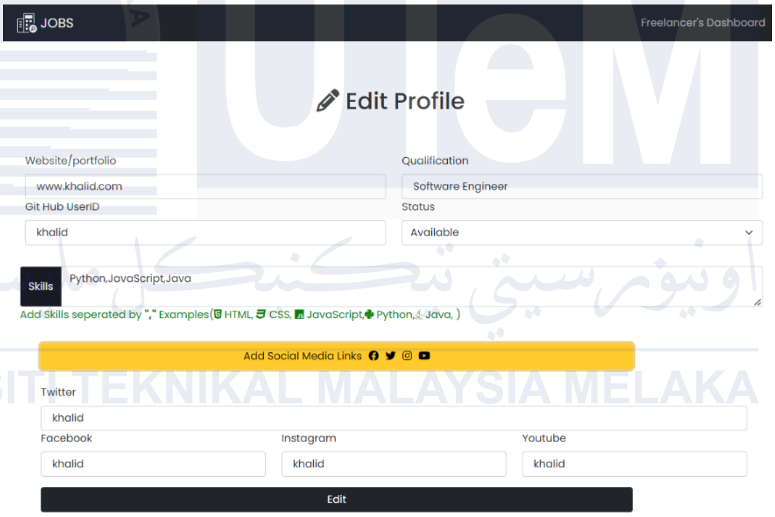


Figure 18: Freelancer's Dashboard:



Edit Profile

Website/portfolio

www.khalid.com

Qualification

Software Engineer

Git Hub UserID

khalid

Status

Available

Skills

Python,JavaScript,Java

Add Skills seperated by "," Examples(HTML, CSS, JavaScript, Python, Java,)

Twitter

khalid

Facebook

khalid

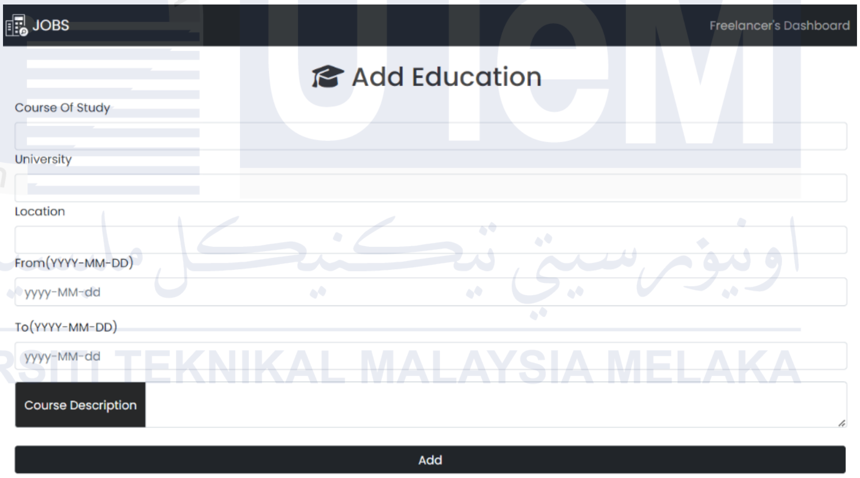
Instagram

khalid

Youtube

khalid

Figure 19: Freelancer Edit Profile Page



JOBS

Freelancer's Dashboard

Add Education

Course Of Study

University

Location

From(YYYY-MM-DD)

yyyy-MM-dd

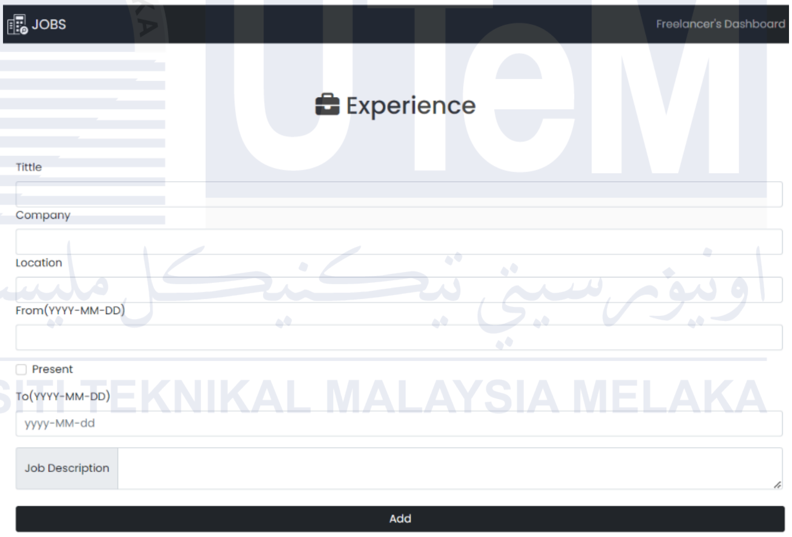
To(YYYY-MM-DD)

yyyy-MM-dd

Course Description

Add

Figure 20: Add Education If The User Is Student Form



Tittle

Company

Location

From(YYYY-MM-DD)

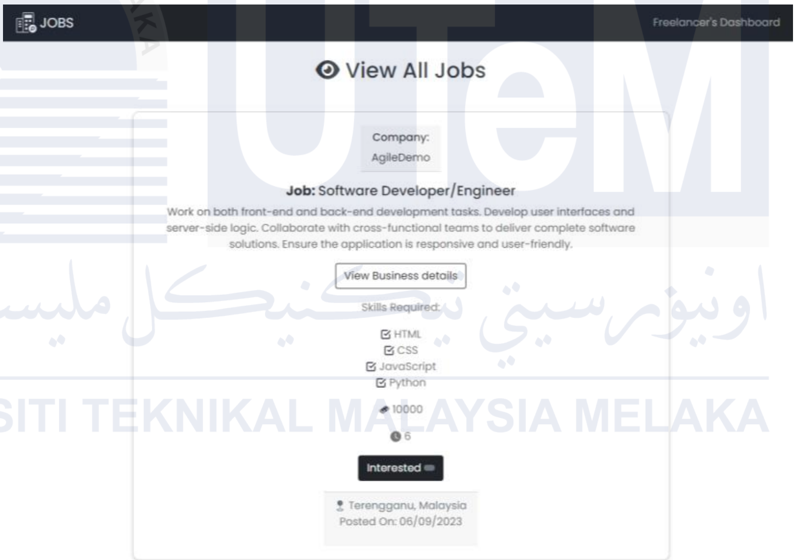
□ Present

To(YYYY-MM-DD)

yyyy-MM-dd

Job Description

Figure 21: Add Experience If The User Is self-employed Form



JOBS

Freelancer's Dashboard

View All Jobs

Company:
AgileDemo

Job: Software Developer/Engineer

Work on both front-end and back-end development tasks. Develop user interfaces and server-side logic. Collaborate with cross-functional teams to deliver complete software solutions. Ensure the application is responsive and user-friendly.

View Business details

Skills Required:

☑ HTML
☑ CSS
☑ JavaScript
☑ Python

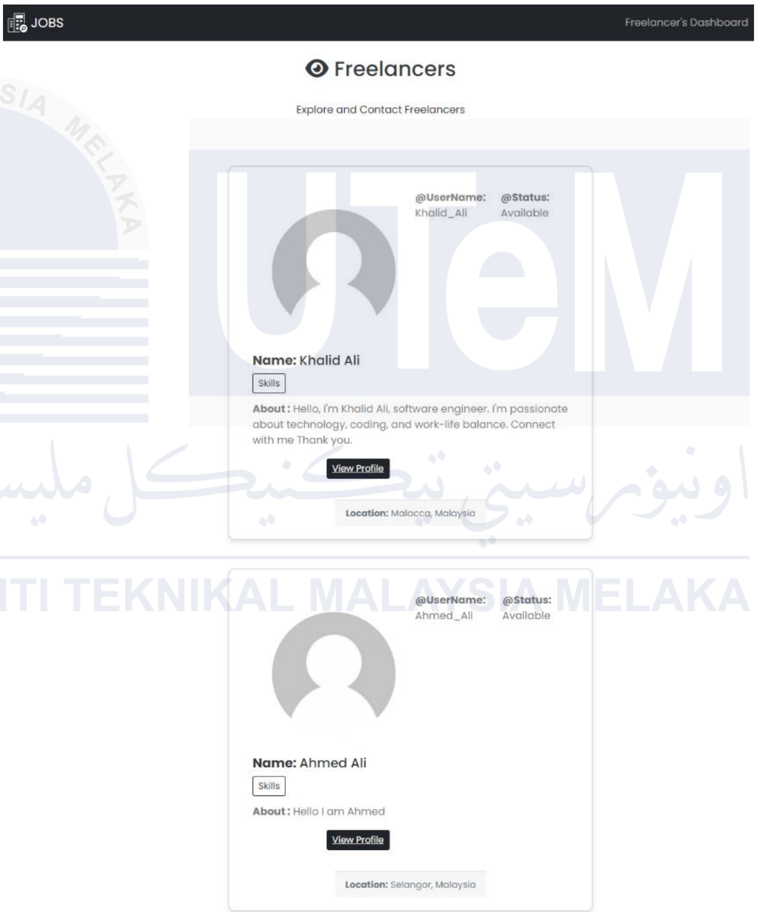
10000

6

Interested

Terengganu, Malaysia
Posted On: 06/09/2023

Figure 22: View Jobs Page



Freelancers

Explore and Contact Freelancers

@UserName: Khalid_Ali **@Status:** Available

Name: Khalid Ali

About : Hello, I'm Khalid Ali, software engineer. I'm passionate about technology, coding, and work-life balance. Connect with me Thank you.

Location: Malacca, Malaysia

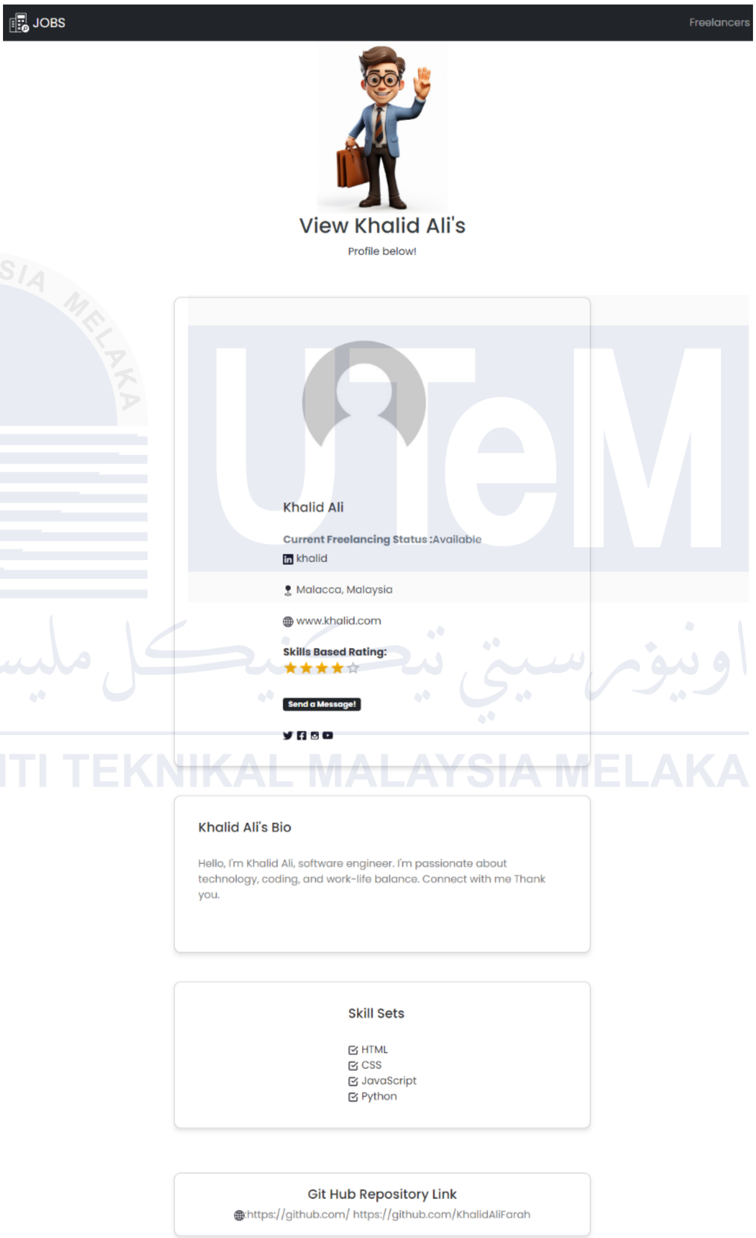
@UserName: Ahmed_Ali **@Status:** Available

Name: Ahmed Ali

About : Hello I am Ahmed

Location: Selangor, Malaysia

Figure 23: View other freelancers



View Khalid Ali's

Profile below!

Khalid Ali

Current Freelancing Status :Available

khalid

Malacca, Malaysia

www.khalid.com

Skills Based Rating:

★★★★☆

Khalid Ali's Bio

Hello, I'm Khalid Ali, software engineer. I'm passionate about technology, coding, and work-life balance. Connect with me Thank you.

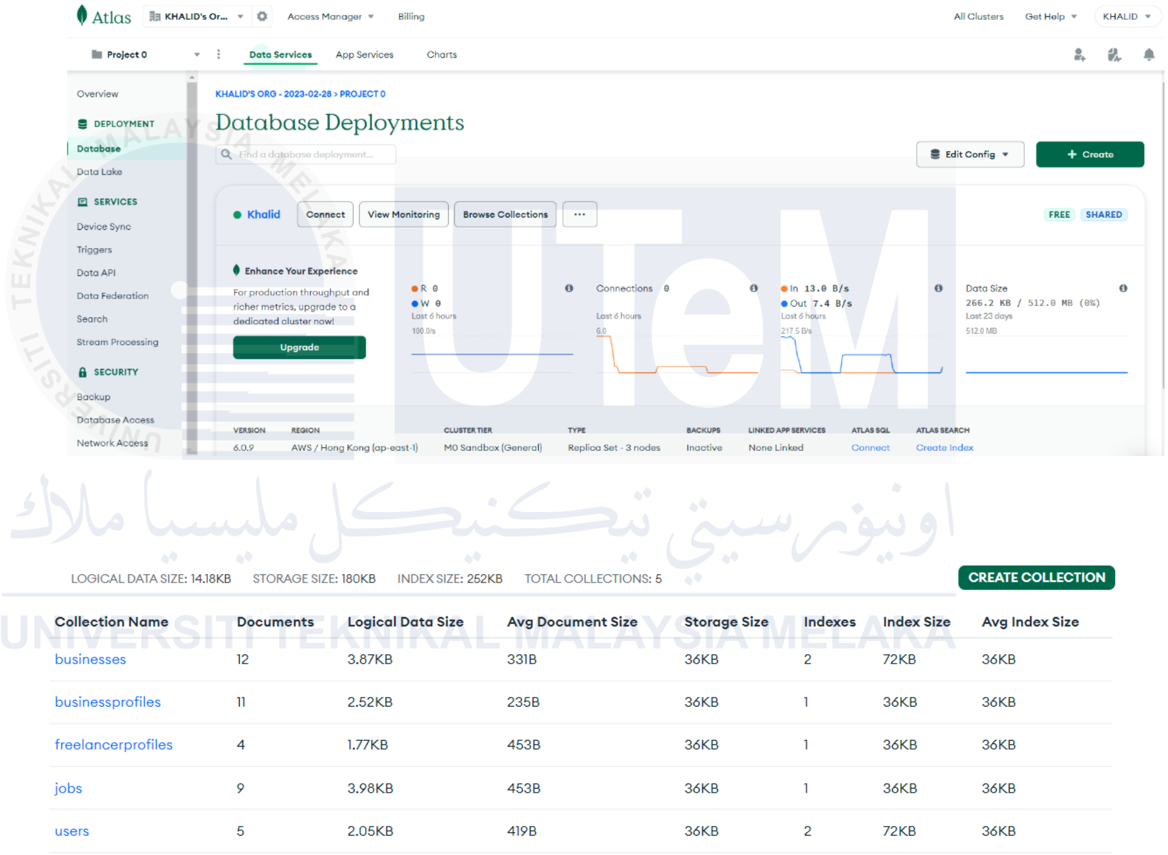
Skill Sets

☑ HTML
☑ CSS
☑ JavaScript
☑ Python

Git Hub Repository Link

https://github.com/ https://github.com/KhalidAliFarah

Figure 24: Freelancers View Profile



KHALID'S ORG - 2023-02-28 > PROJECT 0

Database Deployments

● Khalid

Enhance Your Experience

For production throughput and richer metrics, upgrade to a dedicated cluster now!

R 0
W 0

Connections 0

In 13.0 B/s
Out 7.4 B/s

Data Size
266.2 KB / 512.0 MB (0%)

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SQL	ATLAS SEARCH
6.0.9	AWS / Hong Kong (ap-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Connect	Create Index

LOGICAL DATA SIZE: 14.18KB STORAGE SIZE: 180KB INDEX SIZE: 252KB TOTAL COLLECTIONS: 5

CREATE COLLECTION

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
businesses	12	3.87KB	331B	36KB	2	72KB	36KB
businessprofiles	11	2.52KB	235B	36KB	1	36KB	36KB
freelancerprofiles	4	1.77KB	453B	36KB	1	36KB	36KB
jobs	9	3.98KB	453B	36KB	1	36KB	36KB
users	5	2.05KB	419B	36KB	2	72KB	36KB

Figure 25: Admin