A WEB-BASED MANAGEMENT SYSTEM FOR REMOTE WORKERS

FIRORA WORKSPACE

BY

Ariful Islam
ID: 201-15-13843

This Report Presented in Partial Fulfillment of the Requirement of the Degree of Bachelor of Science in Computer Science and Engineering

SUPERVISED BY

Dr. Sheak Rashed Haider Noori
Professor
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

JANUARY 2023

APPROVAL

This Project "**A WEB-BASED MANAGEMENT SYSTEM FOR REMOTE WORKERS"** submitted by Ariful Islam, ID No: 201-15-13843 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on **January 19, 2023**.

DECLARATION

I hereby declare that, this project has done by myself under the supervision of **Dr. Sheak Rashed Haider Noori, Professor,** Department of CSE Daffodil International University. I also declare that neither this project nor any part that solely related to this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:

[illegible] 16.01.2023

Dr. Sheak Rashed Haider Noori
Associate Professor and Associate head
Department of Computer Science and Engineering
Faculty of Science and Information Technology
Daffodil International University

Submitted by:

[illegible]

Ariful Islam
ID: 201-15-13843
Department of Computer Science and Engineering
Faculty of Science and Information Technology
Daffodil International University

ACKNOWLEDGEMENT

First, let me begin by expressing my sincere gratitude to Almighty God for giving us the ability to successfully finish the senior project and internship via His divine grace.

I sincerely thank you and express my sincere gratitude to **Dr. Sheak Rashed Haider Noori, Professor**, Department of CSE, Daffodil International University, Dhaka. Our supervisor has a wealth of knowledge and a genuine interest in the "*Software Engineering*" needed to complete this assignment. His never-ending tolerance, academic leadership, ongoing encouragement, consistent and vigorous supervision, constructive criticism, helpful counsel, reading several subpar drafts and revising them at every level have allowed me to finish this report.

We would like to express our gratitude to the other academics and staff at Daffodil International University's CSE department for their kind support in producing this report. I want to thank everyone of my classmates at Daffodil International University who participated in this discussion while also attending class.

Last but not least, I must respectfully thank my parents for their unwavering love and support.

ABSTRACT

Since 2020 the world is seen a rapid shift to full-time remote workers and hybrid. Due to this shift lots of new and old tools as been reconstructed to meet the new demand. On the other hand, short-time freelancing work and independent freelancers are also increased dramatically. Thousands of individuals start their own freelancing businesses every day. Freelancers use a variety of paid and free products and services every day to advance and outperform their competition. A sizable proportion of independent contractors who operate in groups seek to launch their own businesses. One of the issues they frequently encountered when conducting business outside of the freelance sector. They get disorganized and wind up utilizing an excessive number of services, which causes a stressful working experience.

So far so good I have come across building web-based systems. That follows a client-server architecture with a micro-service in nature. It uses VueJS in the frontend, Express js for REST-API, and Prisma for ORM. Additionally, during the development stage, the system's scalability is given top importance. Both a monolithic and a microservice architecture can be used to run the software. Docker has been used during the development and the package are released as Docker images. So that it may be utilized in the Kubernetes cluster environment. Building the actual system will require many features however, this prototype includes some basic features which will be a great starting point for the further development process.

The purpose of my research and development is to find and build a system that can be used by the vast majority of individuals and businesses to conduct their business online securely and confidently.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Introduction

To build a platform for start-ups and mid-size businesses. The platform will focus user-friendliness while reducing technical overhead. The platform will take advantage of the traditional ERP core system and combine it with modern requirements. This platform will help users to make planning, organizing, implementing, controlling/monitoring, collaborating, marketing, sales, accounting, information gathering, and more. it’s all start with the user, a user could a member or client. All the other services that the system is providing are related to the client. A client is a person or entity who bought any services or products from the provider. However, most of the options can be used solo as well.

The project's development has been split into two parts. I could create the software quickly and simply using full-stack frameworks, but I'm more interested in learning each one separately and being able to run standalone. I frequently considered switching to a simpler framework, like Laravel, during the development. When building the backend with ExpressJS, you will start from scratch, whereas other frameworks will include the majority of the essential components. Even so, I'm glad that I chose the difficult path because everything else will seem simpler once I get acclimated to it.

To work with databases, I utilized Prisma. I prefer to have a single design and the ability to connect to several backend databases, such as MySQL, PostgreSQL, MongoDB, etc. I find Prisma to be a very good ORM, and I really like using and understanding it. It comes with Studio software for visual representation and database manipulation, and it's fairly simple to learn.

Utilizing typescript was a wise choice, but it was also difficult. However, I made an effort to keep my code as secure as I could by using TypeScript and Zod on my project.

1.2 Motivation

During the Covid-19 epidemic, I had the concept while starting my start-up firm. I then gather a small group of close friends and well-known individuals to begin the adventure. As a result of the scenario, every member of our team worked virtually from their homes. For our small team, I was seeking for a solution. There are several options available; nevertheless, I was unable to find a perfect solution at a reasonable cost. Traditional ERP systems are extremely expensive and require additional costly training to use. I've tried a few open-source ERPs, including Odoo, but they are all quite difficult to use and important modules are paid.

We made an effort to employ many open-source applications. They were created to address particular problem categories. Thus, to conduct business online, we end up using 5-7 separate pieces of software. Combining various open-source programs initially looks like a smart idea. However, our data also expands dramatically as team's progress and grow. Individually managing each piece of software becomes time-consuming and difficult. Moreover, since the data are vague, it is harder to draw any conclusive conclusions.

That this isn't simply our issue struck me. There are approximately **137,000** new businesses founded each day, therefore **50 million** new businesses are expected to be founded year. The number of people working as independent contractors has increased dramatically. Freelancers accounted up 36% of the American workforce in 2021, according to study by Upwork® Global Inc., and they helped the economy of the country by **$1.30 trillion** dollars.

As a result, I envision a healthy market demand for software that will enable those companies or individuals to launch and sustain their worldwide businesses. Also, I see great needs for our own team management and prosper.

1.3 Objectives

The development of microservices is my main goal because, in my opinion, doing so will enable me to implement the software they way I have imagined. Since I entered the realm of JavaScript, where there are countless frameworks and packages available, anything can be done in a thousand of different ways. Hold on? I have still had to cope with Typescript. Top priority scalability and sustainability, the software needs to be able easily scale up and down whenever needed. The software needs to meet the demands run on both a monolithic and a microservice architecture. It needs to package as Docker image so that it could run in Kubernetes environment.

Due to the project's complexity and size, which requires to cover most of the major business or individual needs. The first beta version's development may take up to one year, according to my estimation. However, in terms of this project, I am investigating various existing solutions and conducting market research.

I'm attempting to create a prototype for this project to display my idea and a potential means of assisting others.

- Open source
- Learning REST API
- Learning Backend Development
- Working with ORM
- Docker (Orchestration)
- Cross platform
- Cloud and Self-hosting
- Collaborations
- User friendly
- Accessibility for old and color blind

1.4 Expected Outcome

The technology used in this project may be useful to others who are seeking for a way to archive microservices and cross-platform development.

I will try my best to demonstrate system architecture, which will assist others in developing projects of a similar nature.

- REST API
- JWT Authentication
- Microservice architecture
- ORM (Object–relational mapping)
- SPA (Single Page Application)
- SMTP
- Docker image
- Docker-compose

1.5 Project Management and Finance

There are numerous project management services accessible. GitHub, Bitbucket with Jira, and GitLab are potential excellent choices for software development. I have decided to go with GitLab since it is open-source.

- **Project Management**
    - Git is a version controller for distributed system. It has been used on my project to keep track of changes in any set of files.
    - GitLab is DevOps software package suite which has many features that I have utilize on my project development.
    - Kanban board is task scheduling system. Its comes pre-built with GitLab out of the box. I have used keep track different task and create my development plans using it.
    - XP stand for “Extreme Programming”, I like this idea and principle. Since I am the only person working on this project, I am trying to follow as much as I can. However, for the long run I will go with Agile development methodology.

- **Finance**
    - Self-finance, since I am using my own existing equipment, there was no need to buy new equipment for this project.
    - Prototype's assumed cost is $16,500 USD, depending on resources and a $35 hourly rate over the course of three months.

1.6 Report layout

Chapter 1: Introduction

This chapter serves as an introduction to my project and a discussion of its driving forces, goals, and anticipated outcomes.

Chapter 2: Background

I walk you through the business aspect of the project and introduce to competitors. In comparison to many other possible systems, I also discuss the associated work, the project's scope, and the challenges.

Chapter 3: Requirement Specification

Discussion about software requirement as non-functional and deep details about software development dependences.

Chapter 4: Design Specification

I provide screenshots to illustrate the front-end design of our project, along with information on installation, configuration, and other necessary support tools.

Chapter 5: Implementation and Testing

Using GitLab CI/CD tools, I have demonstrated the build and implementation process. conduct some elementary API endpoint tests, too.

Chapter 6: Impact on Society, Environment and Sustainability

I have try described the direct and indirect affect the software might have on the society and environment. Sustainability is one of my prime futures objectives I have also shared my plan on it.

Chapter 7: Conclusion and the future opportunities

I talked about my findings and the potential for further research and development.

CHAPTER 2
BACKGROUND

2.1 Preliminaries/Terminologies

Microservice

Definition: Software is created using an architectural and organizational strategy known as microservices, which consists of small, autonomous services that communicate over defined APIs.

REST API

Definition: Representational State Transfer (REST also knows as RESTful) an application programming interface (API) that complies with the restrictions of the REST architectural style and enables communication with RESTful web services.

ORM

Definition: Object-Relational Mapping (ORM) is a method that enables addressing, accessing, and manipulating objects without taking into account how those objects relate to their data sources.

SPA

Definition: Single-Page Application (SPA) is a web app implementation technique that only load a single web document and update the DOM for body content. Invention of this technique played a major role in cross-platform development with web technologies.

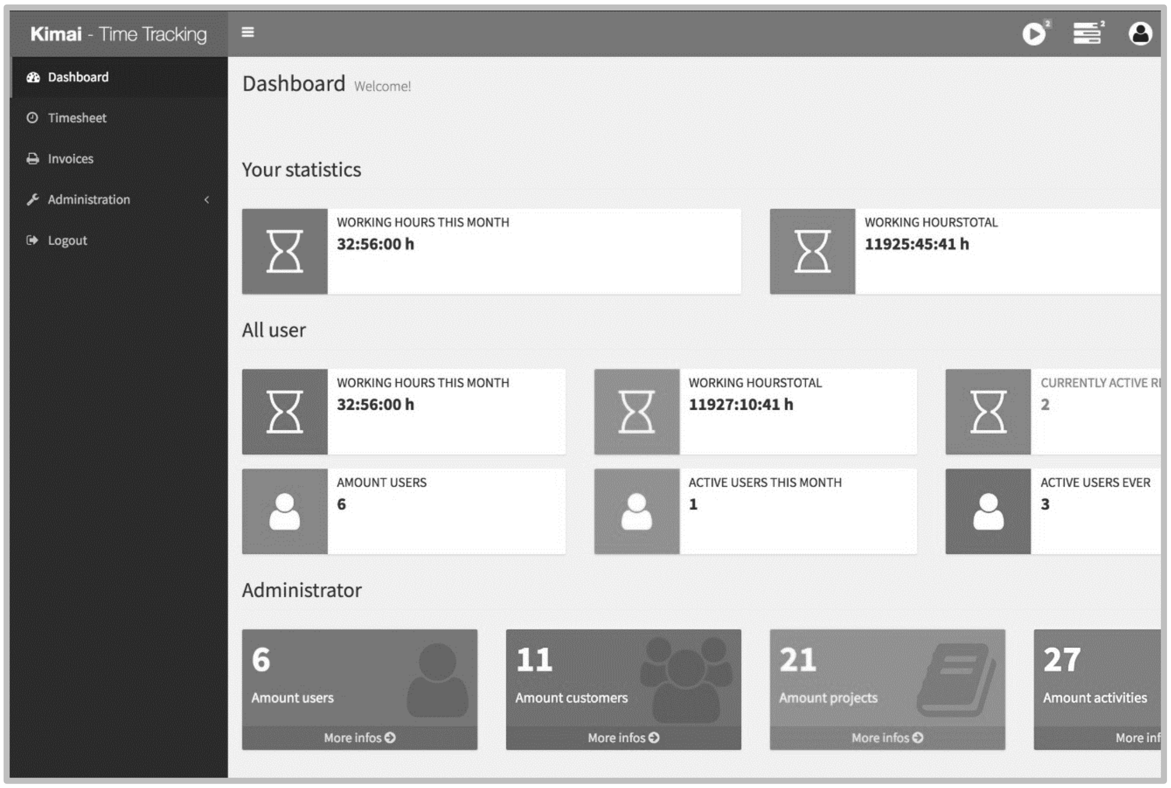
CI/CD

Definition: Continuous Integration/Continuous Deployment (CI/CD) is a set of technique that make possible the frequently deliveries through automation to the various stages of software development process.

2.2 Related Works

There are only a few businesses out there that are offering solutions specific to my project. I am conducting extensive study on them, which is greatly assisting me in obtaining the user needs.

Kimai: Is an open source time tracking software that has the concept of client, service and charge client based hourly or monthly.



WORKING HOURS THIS MONTH 32:56:00 h

WORKING HOURSTOTAL 11925:45:41 h

WORKING HOURS THIS MONTH 32:56:00 h

WORKING HOURSTOTAL 11927:10:41 h

CURRENTLY ACTIVE R 2

AMOUNT USERS 6

ACTIVE USERS THIS MONTH 1

ACTIVE USERS EVER 3

6 Amount users

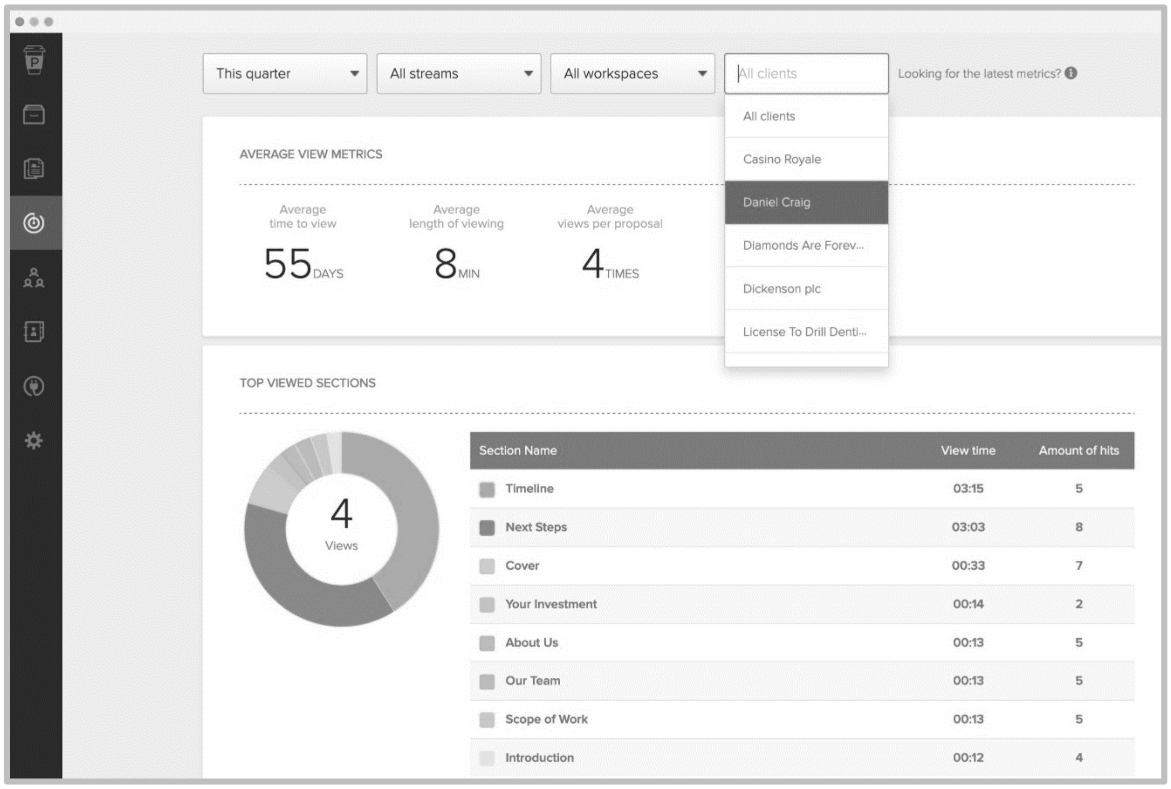
11 Amount customers

21 Amount projects

27 Amount activities

Figure 2.1: Kimai – Time Tracking software

Profosify: Is a proposal software for sales team or contractor. Growing teams can eliminate document bottlenecks and gain visibility into the close, which is the most crucial stage of sales cycle, with the aid of Proposify proposal software.

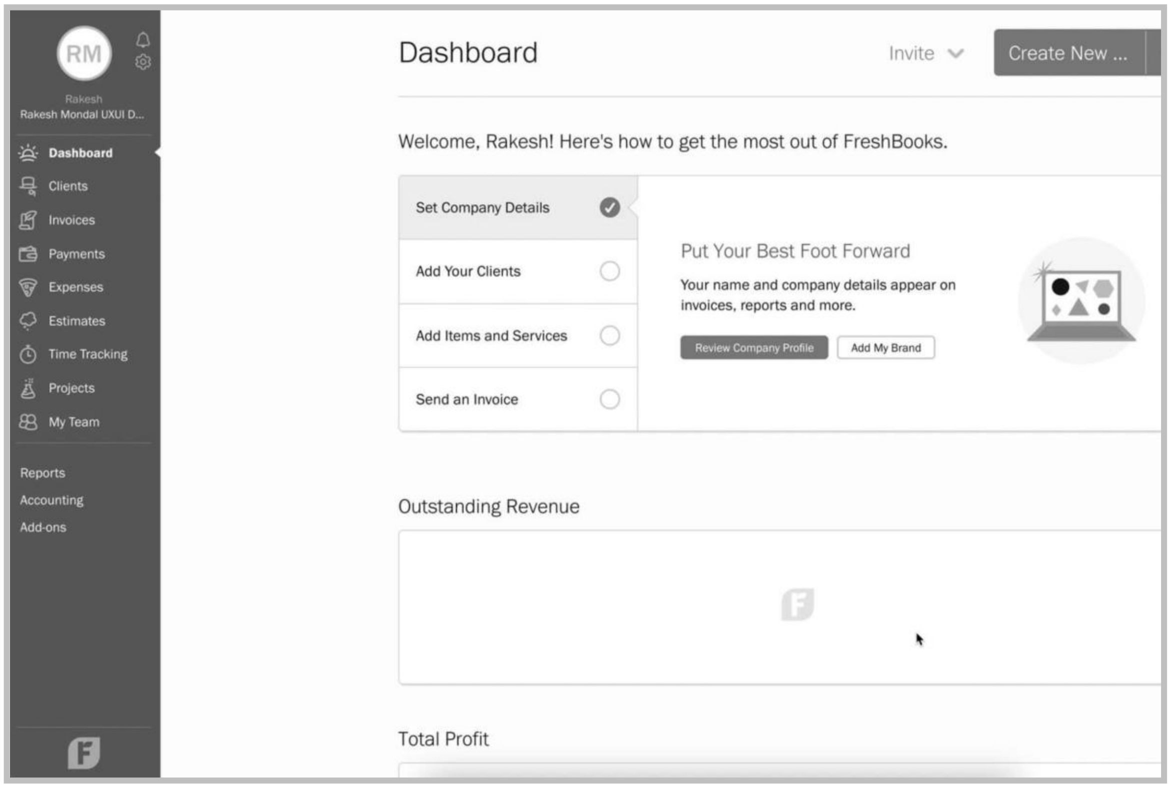


Section Name	View time	Amount of hits
Timeline	03:15	5
Next Steps	03:03	8
Cover	00:33	7
Your Investment	00:14	2
About Us	00:13	5
Our Team	00:13	5
Scope of Work	00:13	5
Introduction	00:12	4

Figure 2.2: Profosify Software

Freshbooks: The accounting program FreshBooks is run by 2ndSite Inc. and is largely used by small and medium-sized enterprises. It is accessible from a desktop or mobile device and is a web-based software as a service model.

Freshbooks stands out from the prior solution as being more beneficial and packed with features, both of which I was considering including in my project.



Welcome, Rakesh! Here's how to get the most out of FreshBooks.

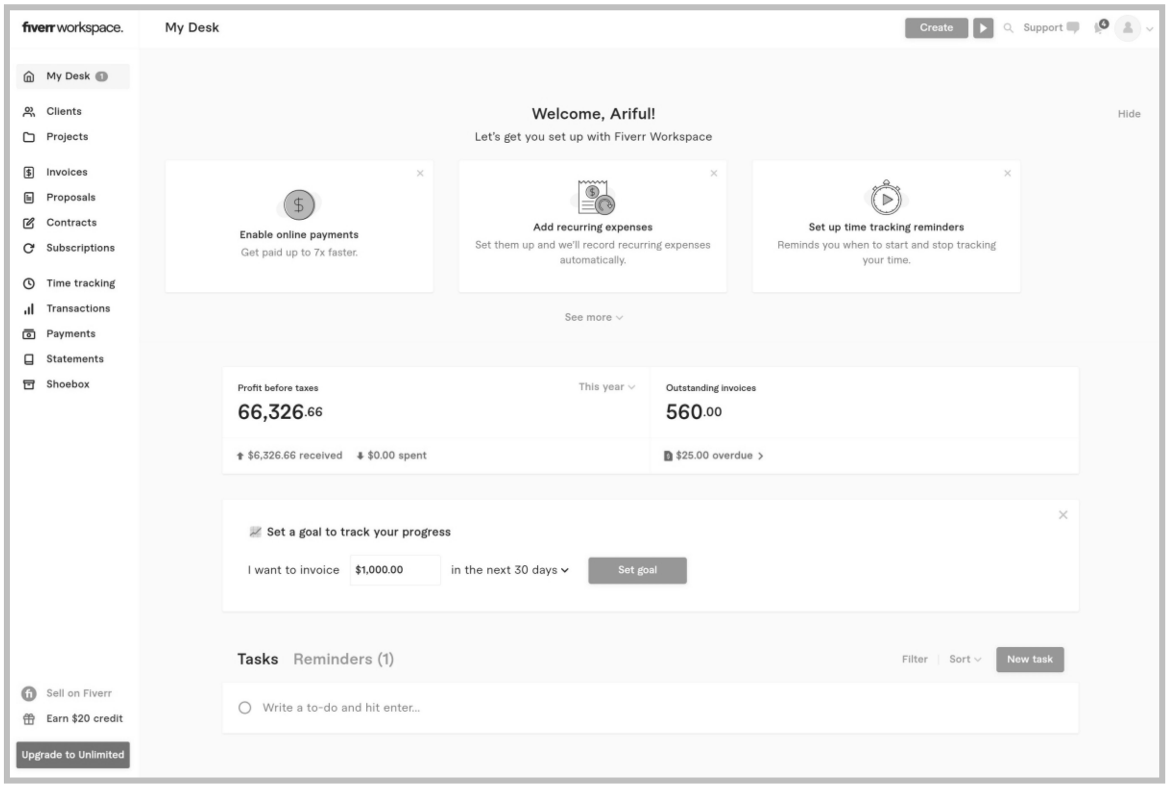
Put Your Best Foot Forward

Your name and company details appear on invoices, reports and more.

Figure 2.3: Freshbook Software

AND.CO: The freelancing program AND CO, designed to assist you in managing your offline business from proposal to payment, was acquired by Fiverr in 2018. Now called AND CO from Fiverr, by reducing friction, this SaaS tool enables freelancers to earn more money.

This service is a perfect example of what I am trying to build from ground up.



Welcome, Ariful!

Let's get you set up with Fiverr Workspace

Get paid up to 7x faster.

Set them up and we'll record recurring expenses automatically.

Reminds you when to start and stop tracking your time.

66,326.66

$6,326.66 received $0.00 spent

560.00

$25.00 overdue

Set a goal to track your progress

I want to invoice $1,000.00 in the next 30 days

Write a to-do and hit enter...

Figure 2.4: AND.CO software

2.3 Comparative Studies

There are other businesses offering features, and I'll be included them in my project as well. However, there are no businesses operating in Bangladesh or in our region of Asia.

A program that will assist this new firm in becoming legally established is in high demand due to the daily growth of commerce-based businesses in our nation. Our government also loses VAT and TAX because there isn't any specific software for them. Note don't compare extensive ERP system with this lightweight software for small and mid-size business. If all businesses operated responsibly, it would be simpler for them to analyze data and make wiser business decisions. They will then automatically have reports and records that can be given to authorities.

2.4 Scope of the Problem

To avoid the usual monolithic development approach, I have divided the project into server side (backend) and client side (frontend) components. Start creating API, so that they can talk to each other over API and that will allow me later develop the desktop and mobile client for the system. There for it took me longer time to even started. Additionally, I avoided selecting an all-encompassing backend framework so that I could study each one separately and consider other solutions as opposed to simply seeing through a predefined view. According to what I've learned, this path is harder, but if you go through it, everything else will seem simpler.

Some key technical problem of the project as follow:

- Protecting the API endpoint from unauthorized access
- Connecting the server API with the client
- Choosing correct NPM packages
- Creating authentication mechanism
- Validating user input
- Development pattern and architecture
- Setting up development environment

2.5 Challenges

I encountered difficulties at every stage of the process, and I frequently became irritated. Collecting and analyzing user requirement then creating different features that are interconnect and keeping relation all in your mind is quite good challenging.

Also need to collect and analysis similar software, then come to conclusion which one is the best approach that will solve people’s problem.

Some key challenges that I tried to overcome:

- Finding the best “tech stack” for the client side
- Finding the best “tech stack” for server side
- Creating a pattern or workflow, that will be followed
  by the system to do any actions.
- Creating a Single-Page Application for web app
- Securing REST-API end point
- Validating user input and protecting from SQL injection

CHAPTER 3
REQUIREMENT SPECIFICATION

3.1 Business Process Modeling

By providing data-driven visual representations of the most important business processes, business process modeling (BPM) provides organizations with an easy approach to comprehend and optimize workflows.

I have created some basic BPM; those are given below.

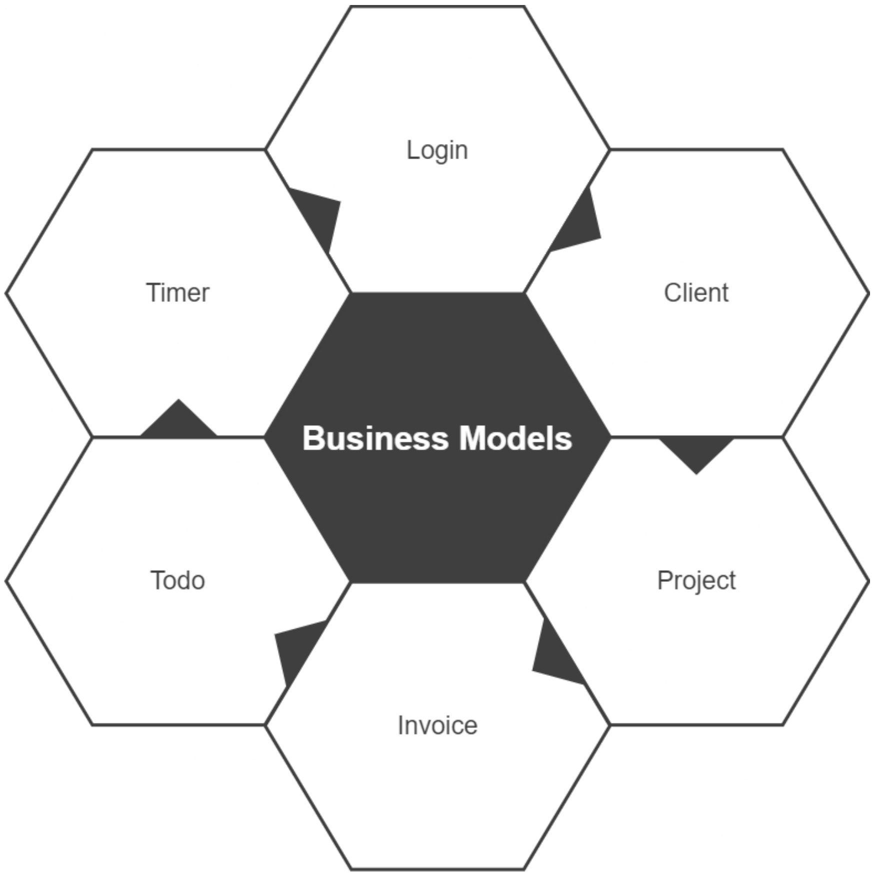


Figure 3.1: Business Models

The illustration below demonstrates the secure authentication process. If the user credentials are accurate and saved in the database, the end user attempted to log in. If the user information is missing from the database, they will receive the wrong login information and, if they haven't already signed up, be presented with a registration screen.

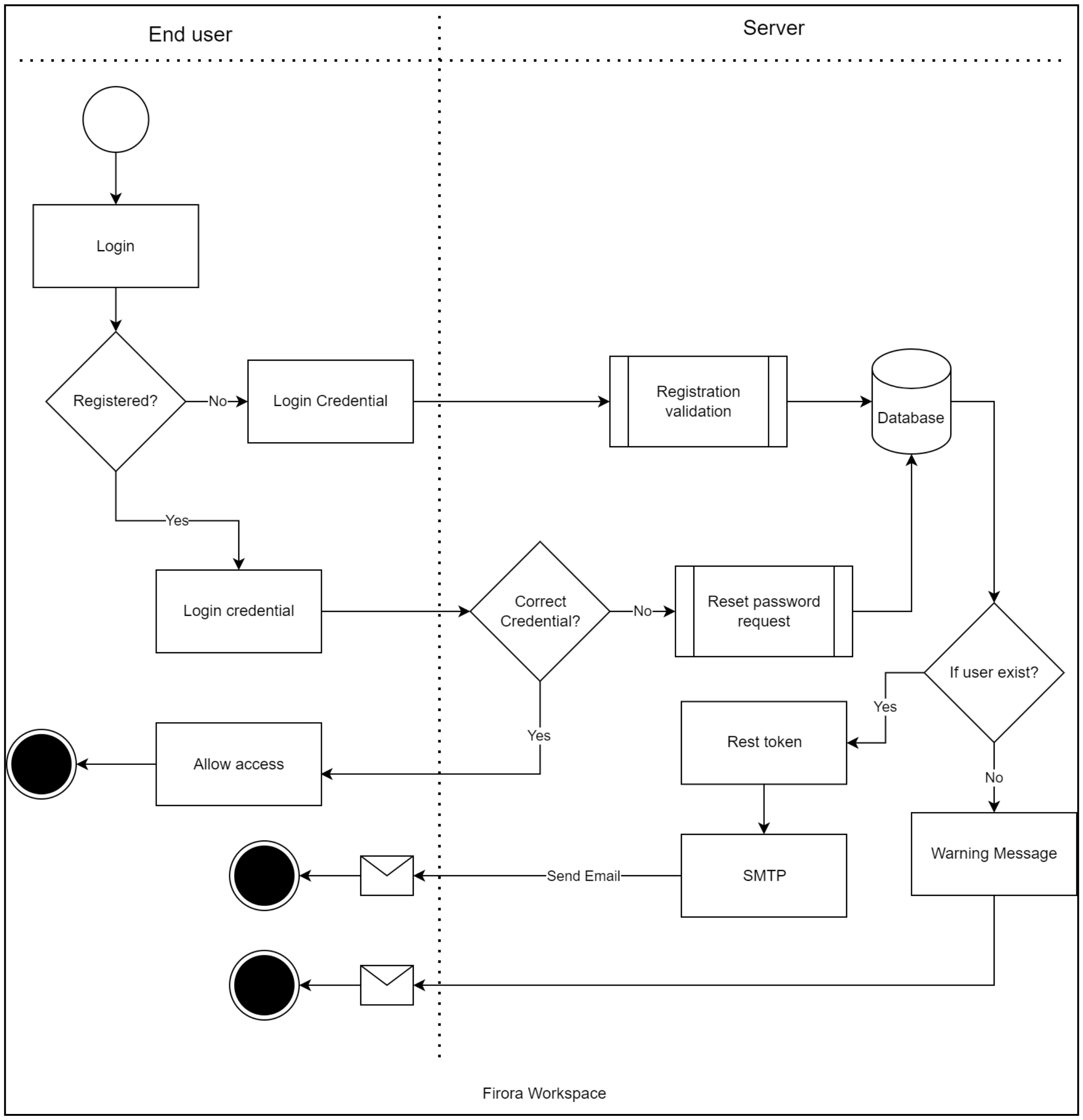


Figure 3.2: Login activity Diagram

The BPM diagram was pretty challenging to create. Please accept my sincere apologies for making the component appear so tiny and difficult to read because there were so many components to depict.

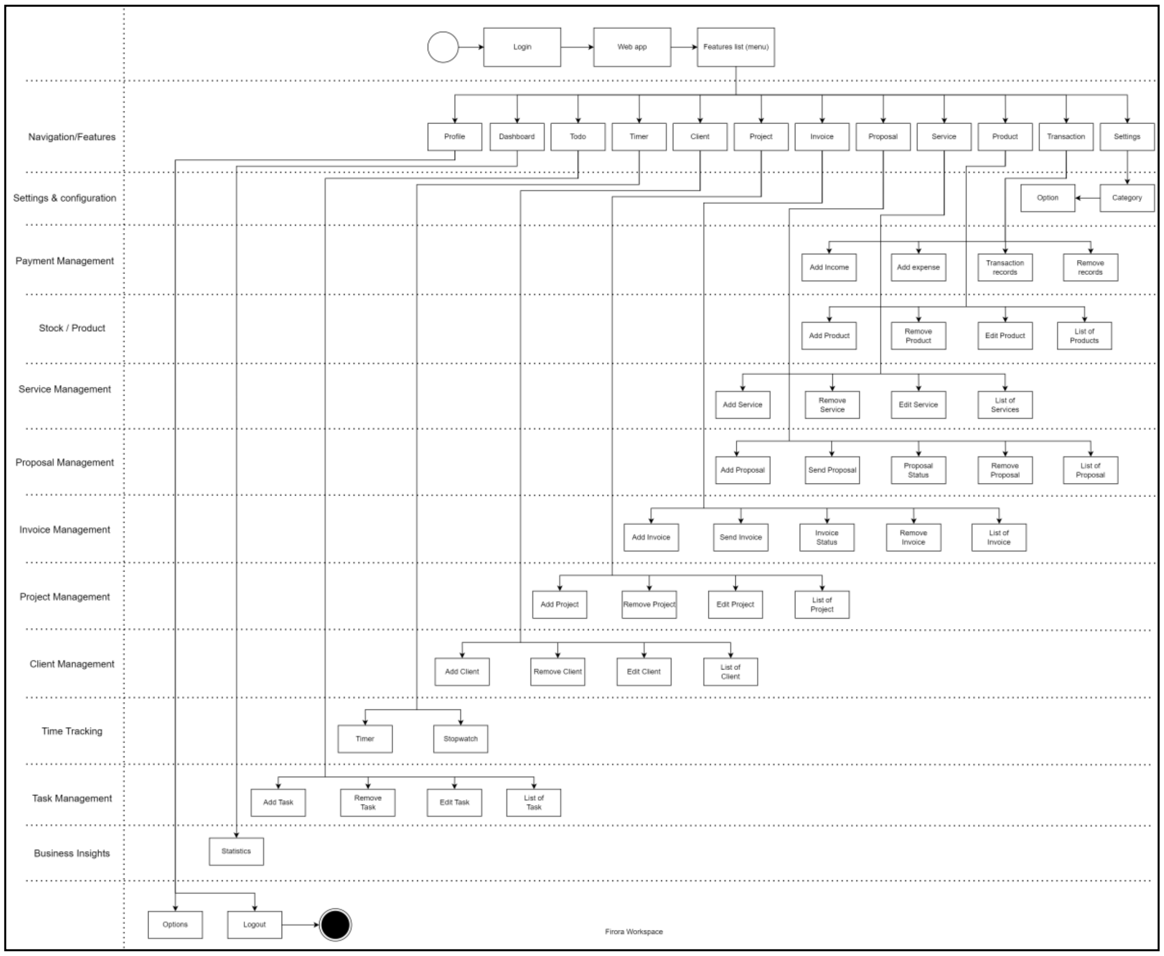


Figure 3.3: Business Processing model

3.2 Requirement Collection and Analysis

The first step in creating any system is gathering user requirements. There are various methods for gathering requirements. I decided to conduct online research, analyze current projects, and my personal working experience (since I am freelancer myself, therefor I am a perfect candidate).

Non-technical requirement:

- Easy access to the system (Suggested by an old man)
- Visible color for UX/UX (Suggested by an old man)
- Searching option
- Client/Customer management
- Invoice generator
- Product/Service or Stock management
- Easy proposal creation and tracking
- Easy payment option
- Multiple organization management
- Good performance and great user experience
- Being able to access from different platform (Desktop, Phone)
- Require simple overview or business insight

Based on the user requirement I have find some technical requirement to develop such system.

Technical requirement analysis report:

Table 3.1: Backend node packages

SN	Name	Minify size	Minify + GZIP	Load Time 4G
1	@prisma/client	233^{B}	171^{B}	$191^{\mu\text{s}}$
2	bcryptjs	21.2^{KB}	9.6^{KB}	11^{ms}
3	config	14.6^{KB}	5.3^{KB}	6^{ms}
4	cookie-parser	3.4^{KB}	1.3^{KB}	2^{ms}
5	cors	4.3^{KB}	1.8^{KB}	2^{ms}
6	dotenv	2.7^{KB}	1.3^{KB}	2^{ms}
7	envalid	14.8^{KB}	4.9^{KB}	6^{ms}
8	express	572.8^{KB}	229.4^{KB}	262^{ms}
9	html-to-text	229.7^{KB}	74.4^{KB}	85^{ms}
10	jsonwebtoken	39.9^{KB}	11.6^{KB}	13^{ms}
11	lodash	69.9^{KB}	24.5^{KB}	28^{ms}
12	nodemailer	197.6^{KB}	52.4^{KB}	60^{ms}
13	pug	731.9^{KB}	170.1^{KB}	194^{ms}
14	redis	249.8^{KB}	45.7^{KB}	52^{ms}
15	ts-node-dev	416.5^{KB}	133.1^{KB}	152^{ms}
16	zod	45.3^{KB}	11^{KB}	13^{ms}
Total		2.6^{MB}	776.5^{KB}	888.19^{ms}

Note: This are backend packages, so will not reflect the frontend load time on client side.

Table 3.2: Frontend node packages

SN	Name	Minify size	Minify + GZIP	Load Time 4G
1	quasar	505^{KB}	143.9^{KB}	164^{ms}
2	chart.js	194.1^{KB}	65^{KB}	74^{ms}
3	vue	78.5^{KB}	28.7^{KB}	33^{ms}
4	vue-i18n	48.6^{KB}	14.6^{KB}	17^{ms}
5	vue-router	30.6^{KB}	11.2^{KB}	13^{ms}
6	axios	13.5^{KB}	4.6^{KB}	5^{ms}
7	pinia	11.4^{KB}	4.5^{KB}	5^{ms}
Total		881.6^{KB}	272.3^{KB}	314^{ms}

3.3 Use Case Modeling and Description

Use Case Modeling: This model diagram shows an overview of who have access to which services of the system.

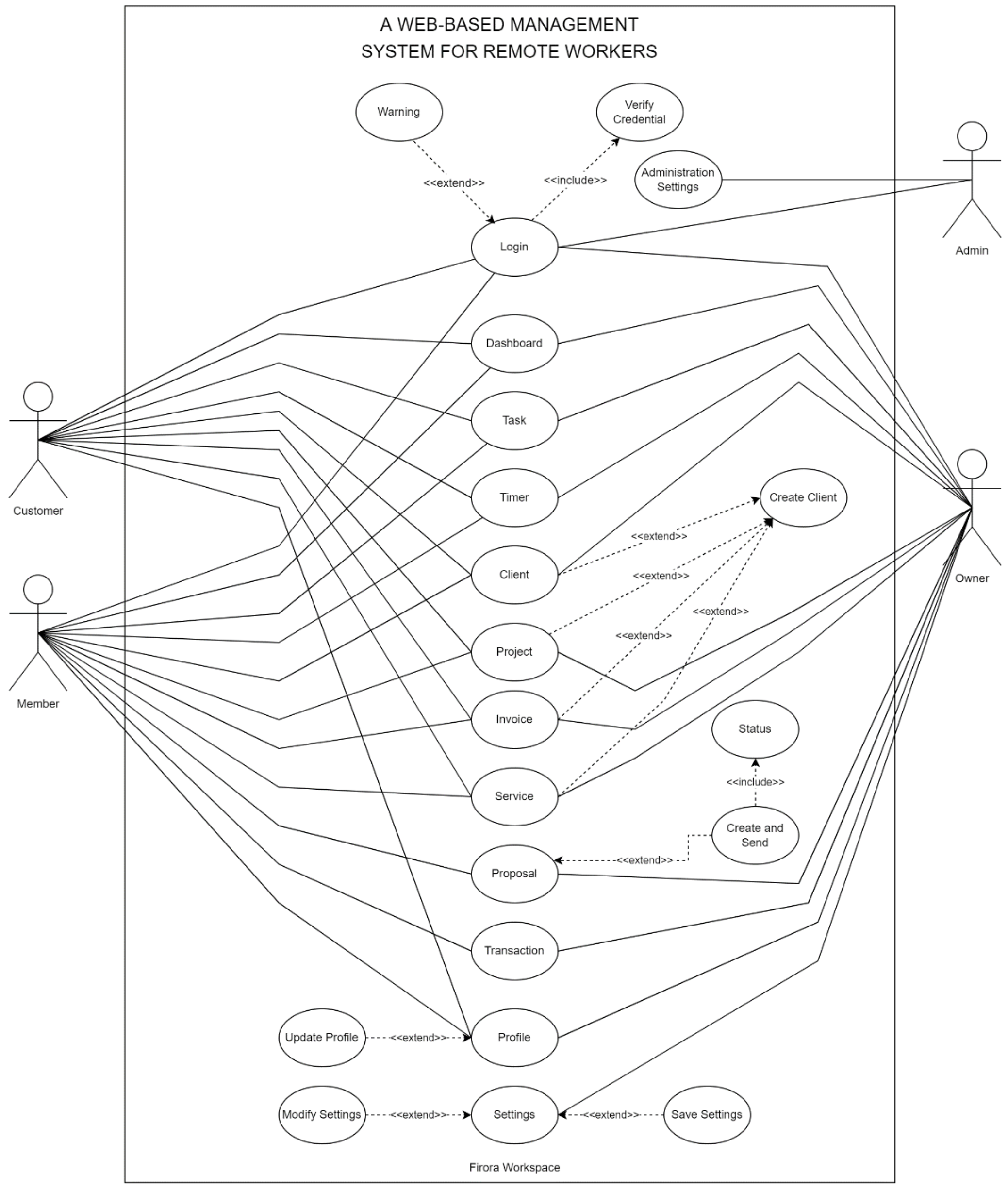


Figure 3.4: Use case modeling

Use Case Description:

Table 3.3: Registration

Use-Case ID	UC-01
Use-Case Title	Registration
Precondition	▪ Need a valid email address ▪ Strong password
Actors	User
Success End State	Show successful registration message
Failure End State	Show error message
Trigger	Redirect to login page
Description	To access the system, users must first register.

Table 3.4: Login

Use-Case ID	UC-02
Use-Case Title	Login
Precondition	▪ User was pre-registered
Actors	User, Owner, Administrator
Success End State	Allow user to access portal
Failure End State	Show error message
Trigger	Redirect to portal
Description	Only those who have been granted access may use this private system.

Use Case Description (Continued):

Table 3.5: Dashboard

Use-Case ID	UC-03
Use-Case Title	Dashboard statistics
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to access dashboard statistics
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	Dashboard is a collection of different statistics about user business. That give a quick insight for business decision.

Table 3.6: Task

Use-Case ID	UC-04
Use-Case Title	Task
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to create and see task list
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	Task are small activity or work related to user projects. User can create task and assign them to any project.

Use Case Description (Continued):

Table 3.7: Timer

Use-Case ID	UC-05
Use-Case Title	Timer
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to track time and set stopwatch
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	A timer is a tool that lets users keep track of the time they spend on various tasks. Set a stopwatch as well to aid in maintaining focus and completing the most crucial tasks.

Table 3.9: Client

Use-Case ID	UC-06
Use-Case Title	Client
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to create and see client list
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	Client are the primary consumer of the business services. User will be able to create client records to track and grow their business.

Use Case Description (Continued):

Table 3.8: Project

Use-Case ID	UC-07
Use-Case Title	Project
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to create and see project list
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	Projects are short- or long-term business agreements based on specific terms and conditions between the seller and the buyer. In my system user can track each project and assign them to any client.

Table 3.10: Client

Use-Case ID	UC-08
Use-Case Title	Invoice
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to create and see client list
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	a list of the products or services sent, along with a summary of the payment owed; a bill.

Use Case Description (Continued):

Table 3.11: Proposal

Use-Case ID	UC-09
Use-Case Title	Proposal
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to create and see proposal list
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	A sales proposal is a written or electronic document that is used to promote goods and services to potential customers. User will be able to create, send and track the status of the proposal.

Table 3.12: Service

Use-Case ID	UC-10
Use-Case Title	Service
Precondition	▪ The end users is currently logged into the system.
Actors	User, Owner
Success End State	Allow user to create and see service list
Failure End State	Show unauthorized warning and redirect
Trigger	Redirect to login page
Description	In my system service are treated as business service, that user provide to their client.

3.4 Logical Data Model

The relationship between each entity and its main and foreign keys is depicted in this entity diagram. It's a fantastic method to examine the intricate database and comprehend it quickly.

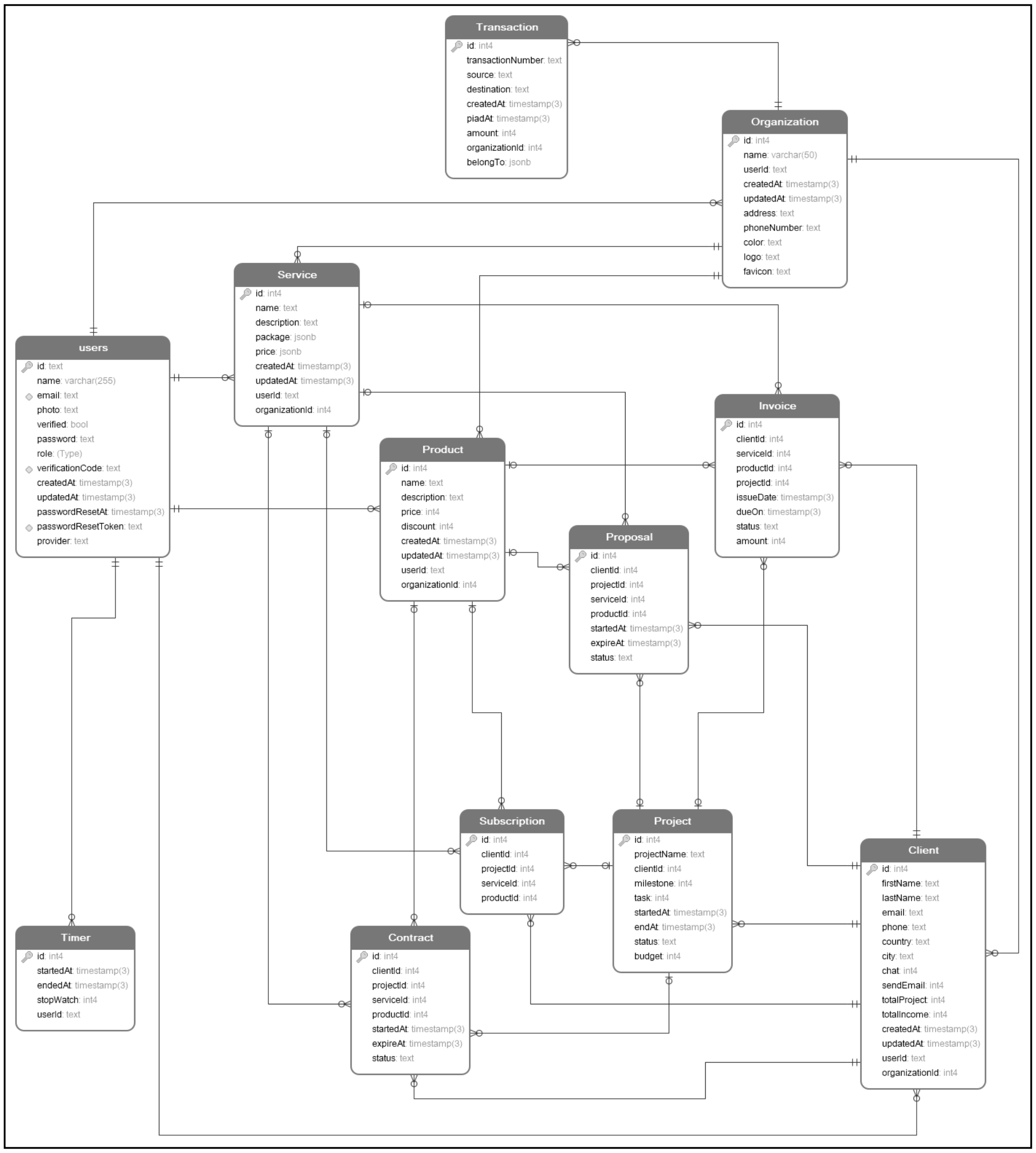


Figure 3.5: ERD Diagram

3.5 Design Requirement

As far as my study goes, I've discovered several system design requirements. I also research their function and relationship. I'll attempt to describe them below.

- **Role based access:** This system has concept of roles per user. Users can have a role of owner, member, customer or admin. Base on the user role the system will provide different user interface and information the user.
- **Service and Product:** Owners and members can create, modify and share service, product to the customer. Customer will be able to see it and place order.
- **Proposal and Invoice:** Proposal are a convenient way to get business leads from customer. And invoice will help them to get paid for the services or product they are selling to their respective customers.
- **Subscription:** To have repeated business user can sell subscription to their customer. They subscription option in the system will automatically charge the customer based on what they subscribe to.
- **Timer and Todo:** Task list manager or To-do is a great way to get list of work need to done per project and by whom. Owner and customer both will be able to create task and assign them to projects, users. Timer is user depended, user can turn on and off. It will allow them to get paid hourly basis and customer will have clear idea about how much hours took per tasks or project.
- **Transaction:** Its give a history of all financial transaction is occurred respective to per user. Owner will get much brought view from the Transaction function.
- **Client:** Customer or client are the same in this system. Only business owner should see the client list and how many and information related to them.
- **Organization:** The concept behind having organization function is that the system can be used by an entity who might own multiple business and want them to manage from a single system. So that make organization a top node of the system tree, everything else connected to the organization.

CHAPTER 4
DESIGN SPECIFICATION

4.1 Front-end Design

I developed the frontend using VueJS and followed SPA pattern to give a seamless user experience. SPA provides end users with the same experience as using a mobile application.

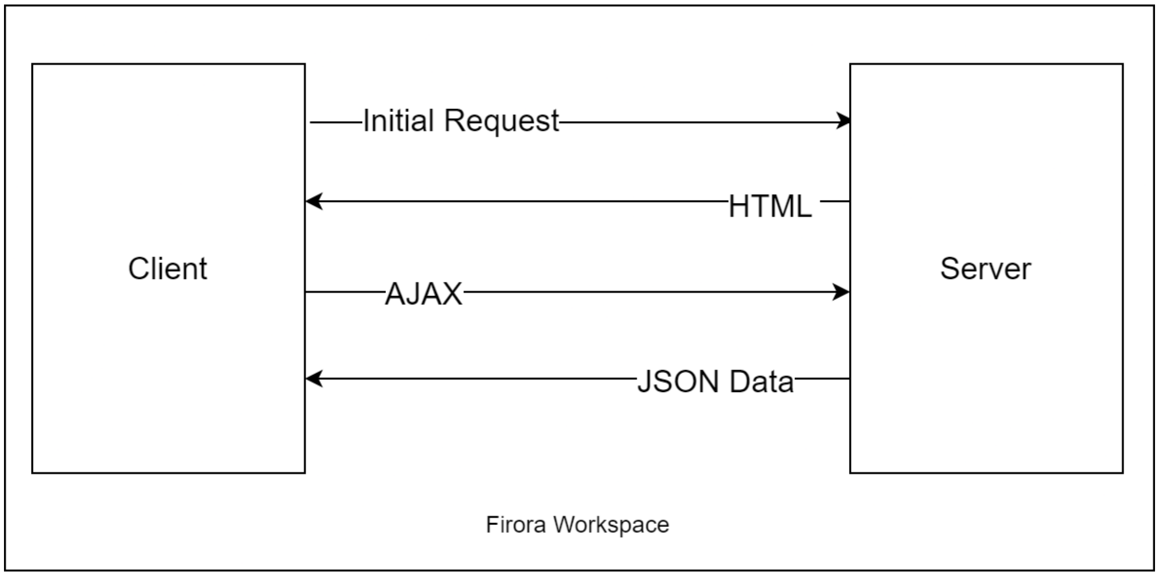


Figure 4.1: SPA Architecture

The frontend initaliy loads a empty htlm page and manuplate the browser DOM to renender different view or component on the display. This way user nerver see blank screen when navigating from one page to another. Since in SPA there is only one single page.

There will be other SPA for Admin panel, will develop it later on. The resion I want to have separate code base for user and admin is reduce the load time. So I don't load any unnecassary code that wasn't required for that specific user.

Project structure: There are more files in the project than those mentioned here. It would take up a lot of room to display every file and folder, which is unnecessary.

```
➜ frontend git:(master)
├── index.html
├── jsconfig.json
├── package.json
├── public
│   ├── favicon.ico
│   └── icons
└── src
    ├── App.vue
    ├── assets
    ├── boot
    │   ├── axios.js
    │   └── i18n.js
    ├── components
    │   ├── charts
    │   ├── settings
    │   └── WYSIWYG.vue
    ├── css
    │   ├── app.sass
    ├── i18n
    │   ├── en-US
    │   └── index.js
    ├── layouts
    │   ├── LoginLayout.vue
    │   └── MainLayout.vue
    ├── pages
    │   ├── ClientPage.vue
    │   ├── DashboardPage.vue
    │   ├── ErrorNotFound.vue
    │   ├── InvoicePage.vue
    │   ├── ProjectPage.vue
    │   ├── ProposalPage.vue
    │   ├── ServicesPage.vue
    │   ├── _SettingsPage.vue
    │   ├── SigninPage.vue
    │   ├── SignupPage.vue
    │   ├── SubscriptionPage.vue
    │   ├── TimerPage.vue
    │   ├── TodoPage.vue
    │   └── TransactionPage.vue
    ├── router
    │   ├── index.js
    │   └── routes.js
    └── stores
```

Lifecycle Diagram of Frontend: Since the frontend build with VueJS, basically a JavaScript framework, that allow me manuplate the DOM at different stage. Each component I have build have this stages and this is how the component is renderd and update data on the DOM.

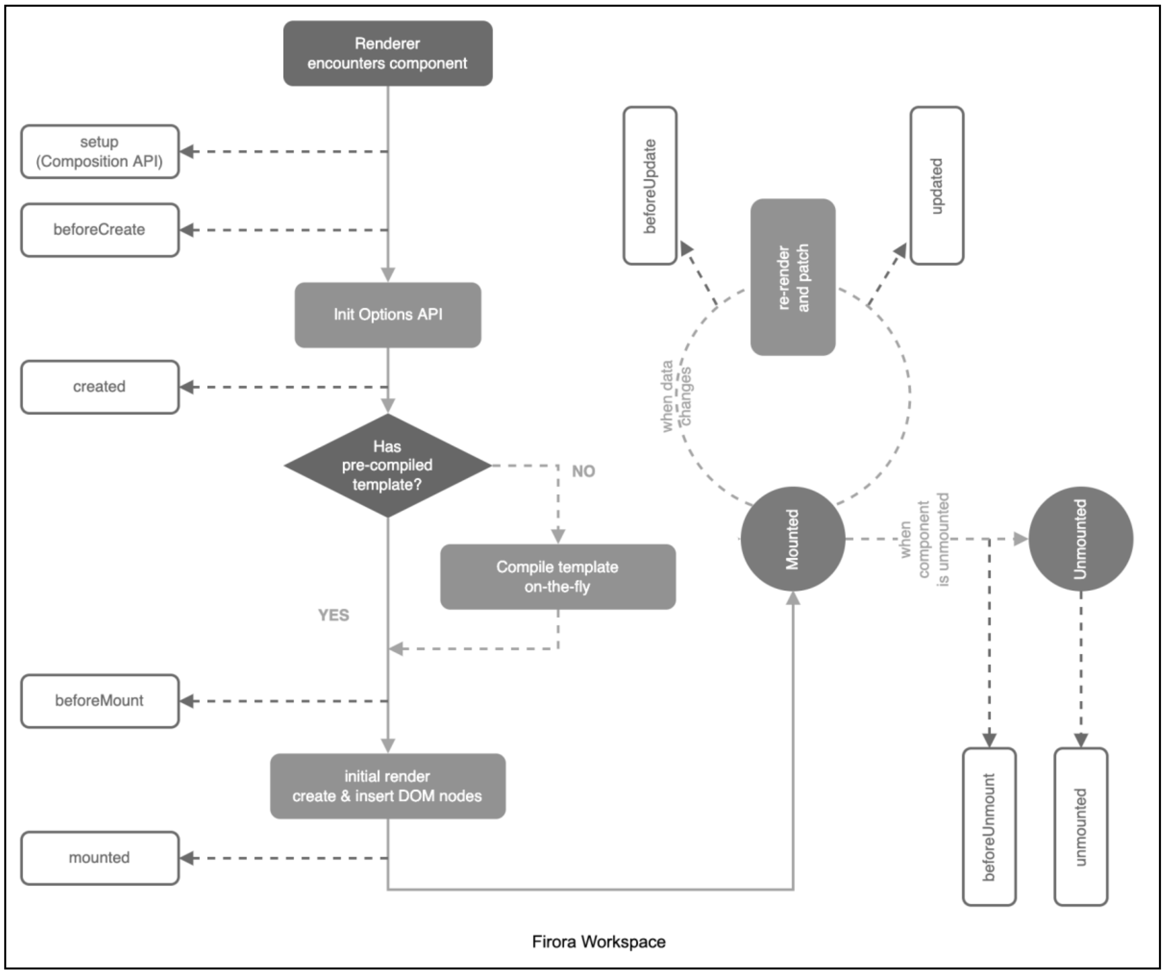


Figure 4.2: Component Lifecycle

4.2 Back-end Design

ExpressJS doesn’t comes with pre-defined development pattern like Larval with MVC (Model, View, Controller). I have developed the backend following pattern mention below in the diagram.

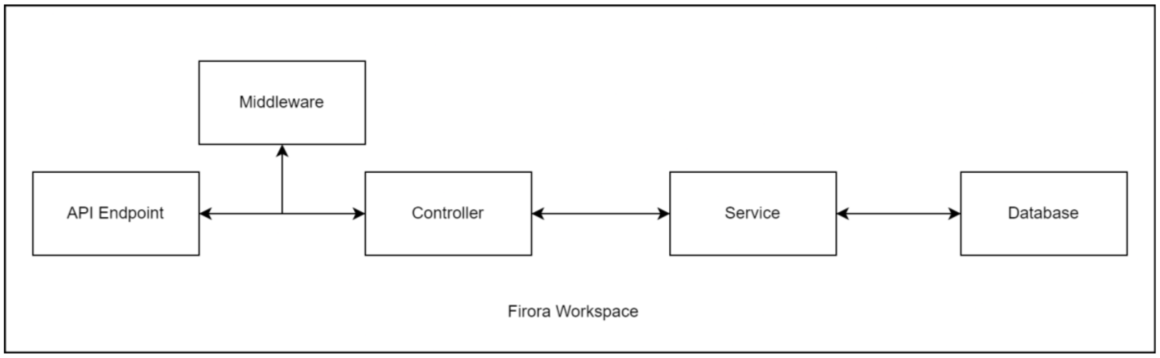


Figure 4.3: Backend development pattern

In this pattern each API endpoint is first primarily manage by a controller and Middleware can be called if it was required base the API route. Controller then forward the request to Service and Service has connection to database. If service need to make any changes and make it persistent then Service store that information into database. And its follow the same path to return respond back to the API client.

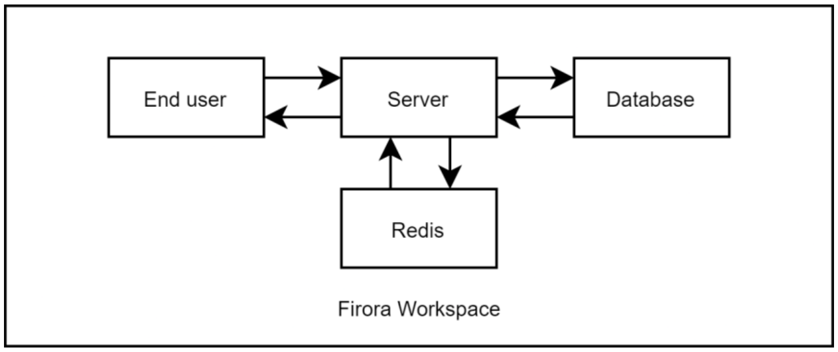


Figure: 4.4: Caching example

Optionally I have also used redis to cache necessary quires. Redis is a great solution cache data from database and serve the cache to end user instead making requires to database all the time for same data.

Project structure: Backend file and directory structure is cleaner and simpler than the frontend. Most of them are displayed here as a level three tree list.

```
➜ server git:(master)
├── 📂 config
│   ├── 📜 custom-environment-variables.ts
│   └── 📜 default.ts
├── 📜 dev-db.yml
├── 📜 package.json
├── 📂 prisma
│   └── 📜 schema.prisma
├── 📂 source
│   ├── controller
│   │   ├── 📜 auth.ctrl.ts
│   │   └── 📜 user.ctrl.ts
│   ├── 📂 middle-ware
│   │   ├── 📜 userDeserializer.ts
│   │   ├── 📜 user.ts
│   │   └── 📜 validate.ts
│   ├── 📂 routes
│   │   ├── 📜 auth.routes.ts
│   │   └── 📜 users.routes.ts
│   ├── 📂 schemas
│   │   └── 📜 users.schema.ts
│   ├── 📜 server.ts
│   ├── 📂 services
│   │   └── 📜 users.services.ts
│   ├── 📂 utils
│   │   ├── 📜 appError.ts
│   │   ├── 📜 connectRedis.ts
│   │   ├── 📜 jwt.ts
│   │   ├── 📜 smtp.ts
│   │   └── 📜 validateEnv.ts
│   └── 📂 views
│       ├── 📜 base.pug
│       ├── 📜 resetPassword.pug
│       ├── 📜 _styles.pug
│       └── 📜 verificationCode.pug
├── 📜 tsconfig.json
└── 📜 yarn.lock
```

I have used JWT (JSON Web Token) for this project. This below flowchart show how the authorization process is completed and renewing process of access token.

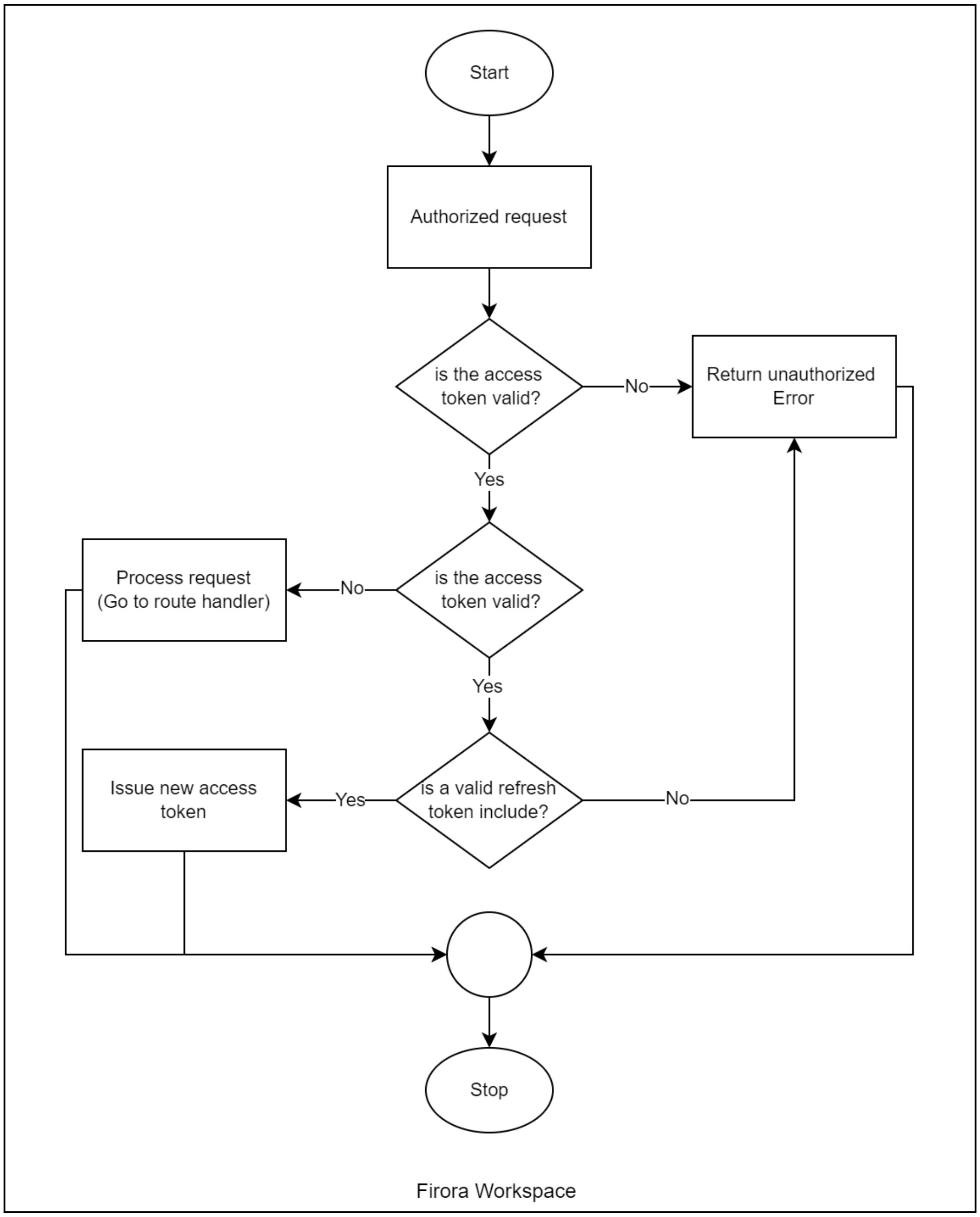

Start
Authorized request
is the access token valid?
No
Return unauthorized Error
Yes
Process request (Go to route handler)
No
is the access token valid?
Yes
Issue new access token
Yes
is a valid refresh token include?
No
Stop
Firora Workspace

Figure 4.5: Authorization Flowchart

This below diagram show the lifecycle of API end point with HTTP request method for user authentication and getting access to the system.

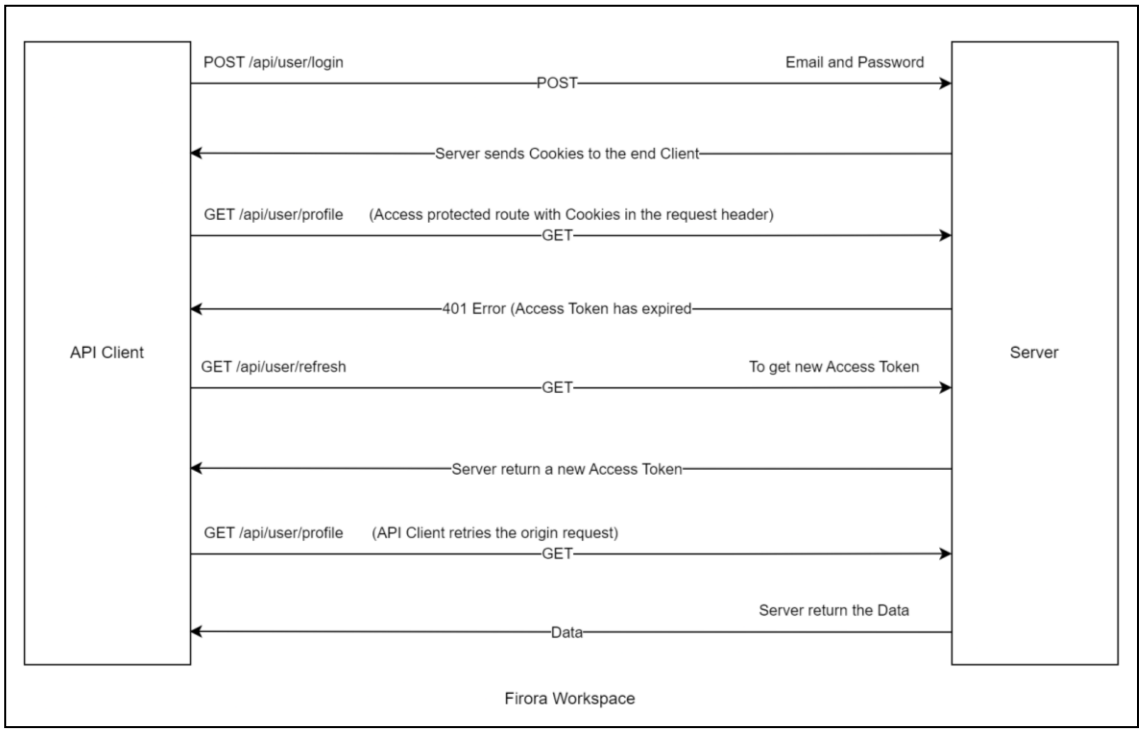

API Client
Server
POST /api/user/login
Email and Password
POST
Server sends Cookies to the end Client
GET /api/user/profile
(Access protected route with Cookies in the request header)
GET
401 Error (Access Token has expired
GET /api/user/refresh
To get new Access Token
GET
Server return a new Access Token
GET /api/user/profile
(API Client retries the origin request)
GET
Server return the Data
Data
Firora Workspace

Figure 4.6: Authorization lifecycle with API endpoint

4.3 Interaction Design and user Experience (UX)

This is a registration page where users can register themselves. the user will need to have a valid email address and a strong password to register himself.

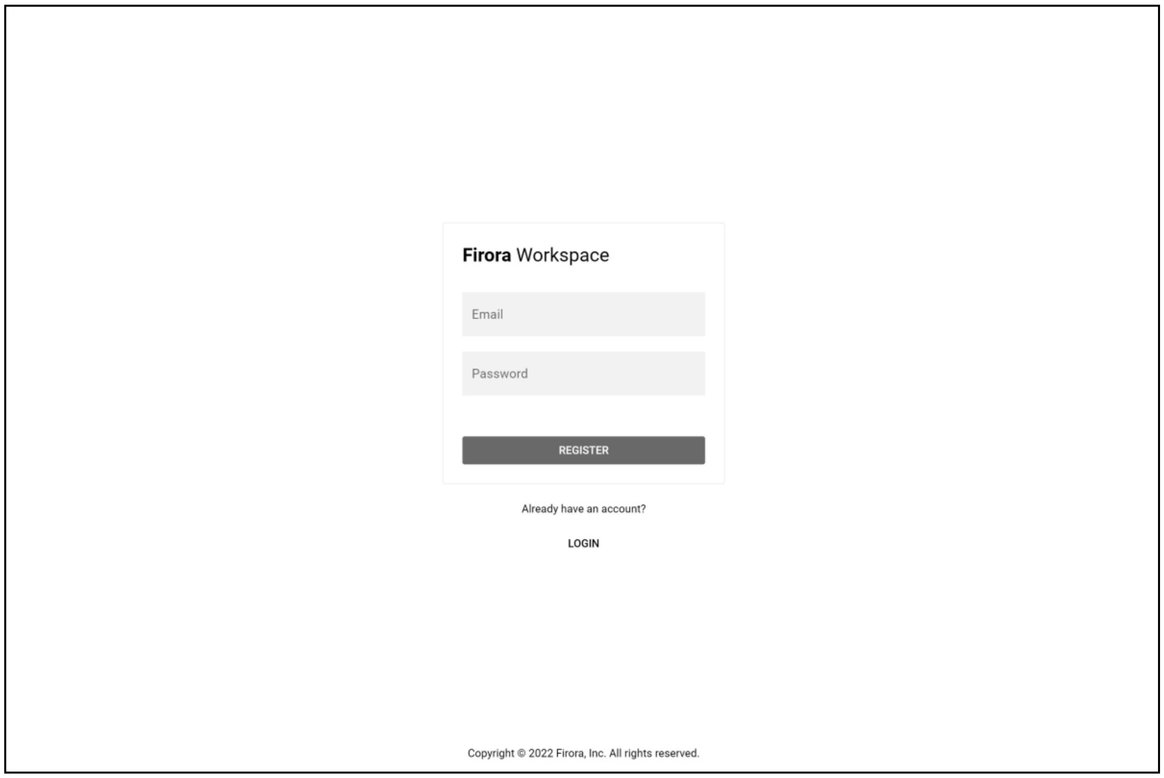


Figure 4.7: Registration page

4.3 Interaction Design and user Experience (Continued)

The validation process involves two phases. One occurred prior to the data being transmitted to the server, and the other when the data had already arrived at the server and was being reviewed before being stored in the database.

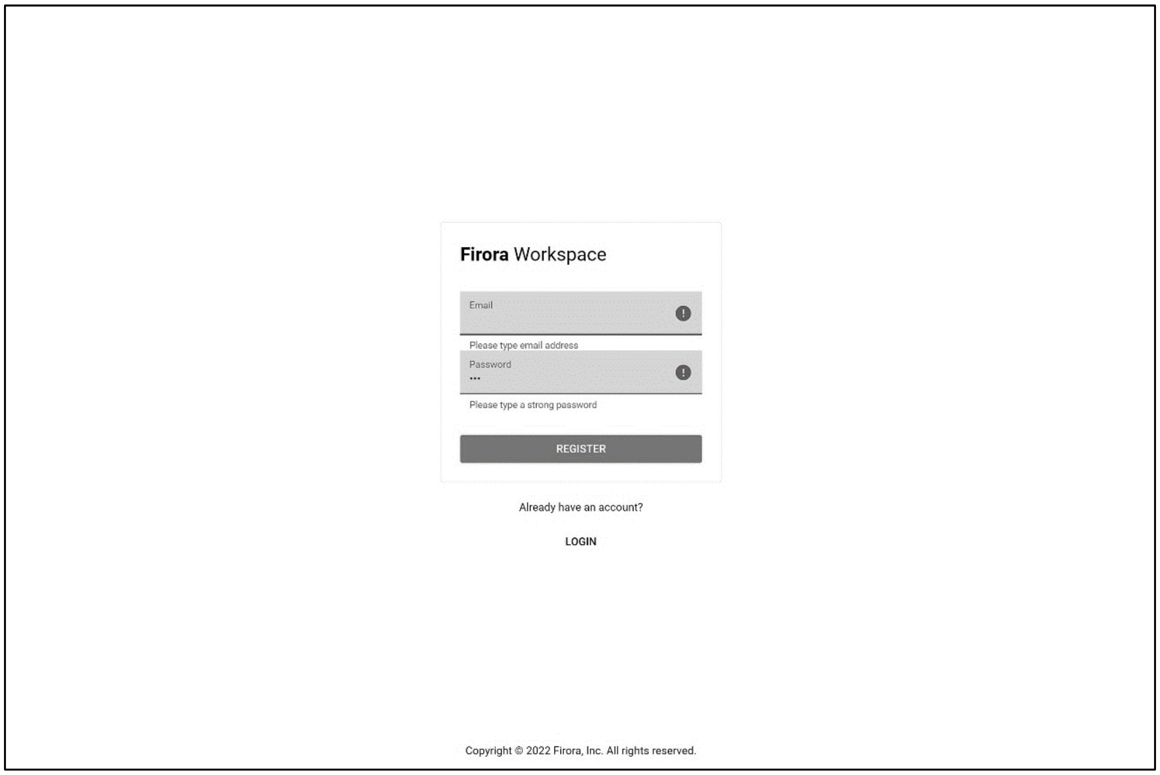

Firora Workspace
Email
Please type email address
Password
•••
Please type a strong password
REGISTER
Already have an account?
LOGIN
Copyright © 2022 Firora, Inc. All rights reserved.

Figure 4.8: Validation example

4.3 Interaction Design and user Experience (Continued)

To sign in and utilize the system, the user must supply valid credentials. This serves as the system's entrance securely.

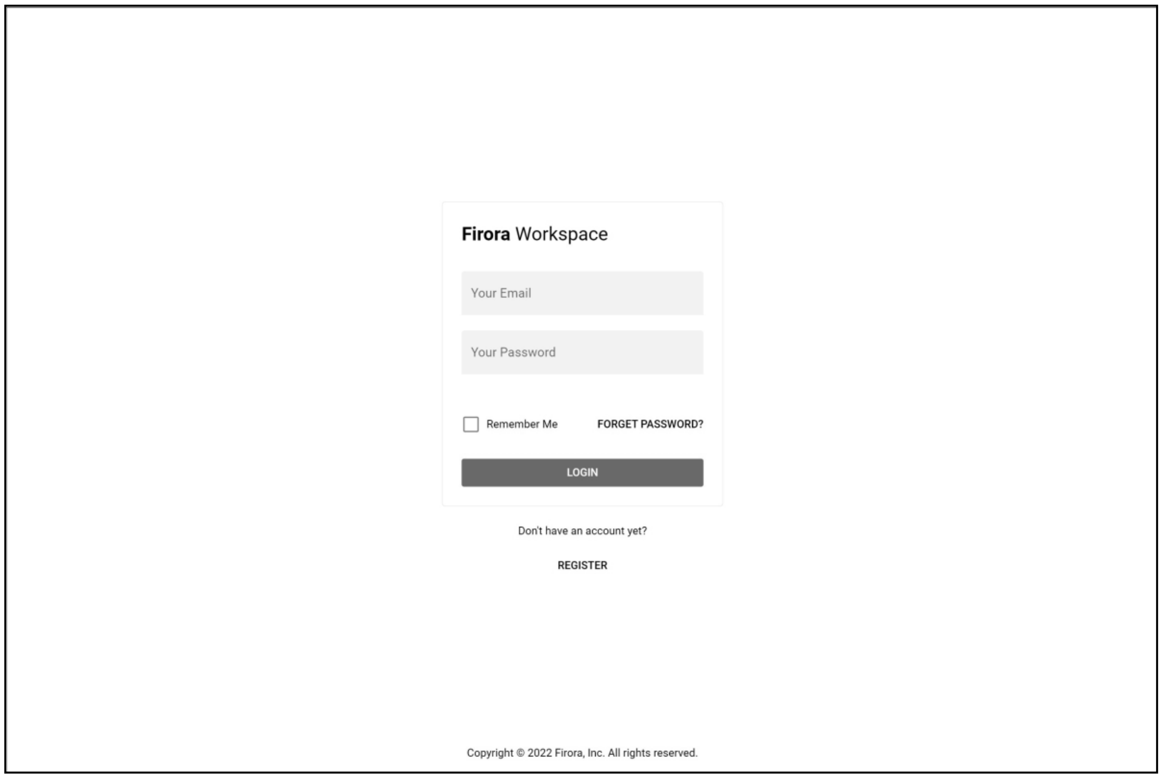
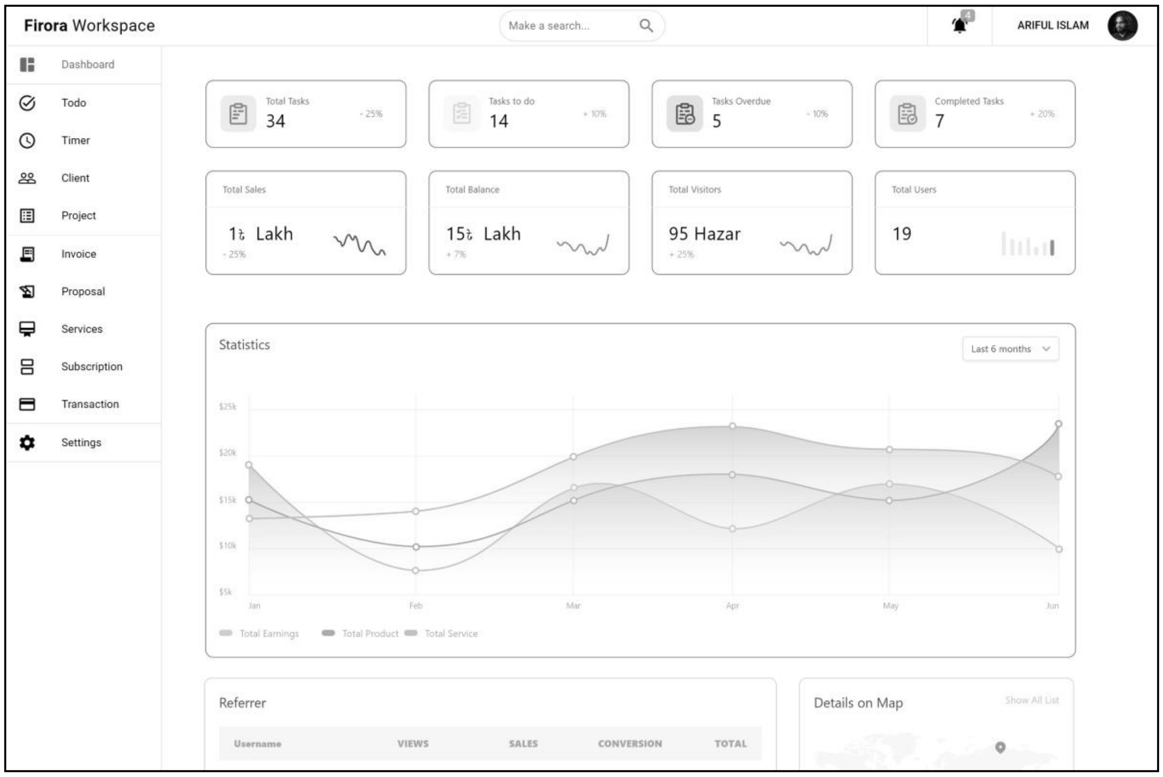


Figure 4.9: Login page

4.3 Interaction Design and user Experience (Continued)

This part of the dashboard contains various statistics regarding the user's business. That provides a rapid understanding for business decisions. To make it happen and make it highly helpful, I'm still working on it.



Total Tasks 34 - 25%

Tasks to do 14 + 10%

Tasks Overdue 5 - 10%

Completed Tasks 7 + 20%

Total Sales 1৳ Lakh - 25%

Total Balance 15৳ Lakh + 7%

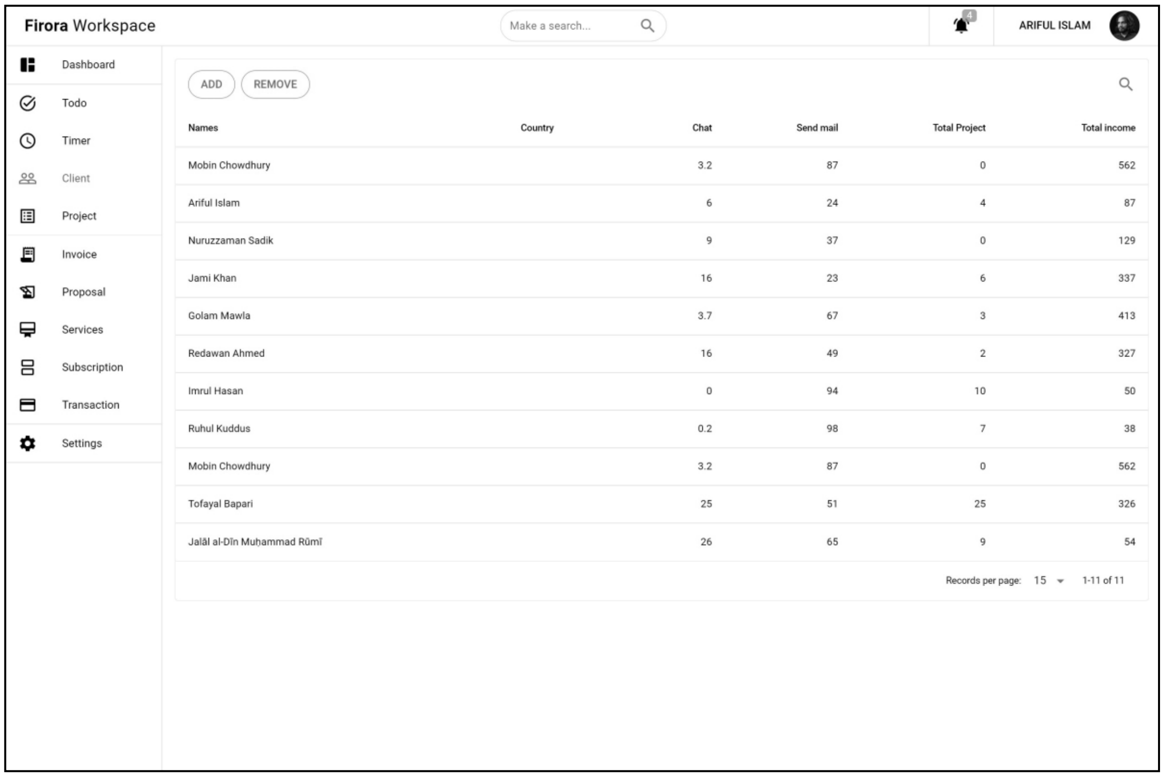
Total Visitors 95 Hazar + 25%

Total Users 19

Figure 4.10: Dashboard

4.3 Interaction Design and user Experience (Continued)

The main consumers of company services are clients. Users can create customer records to keep track of and expand their business. It allows user to add and remove customer and manipulate their records.



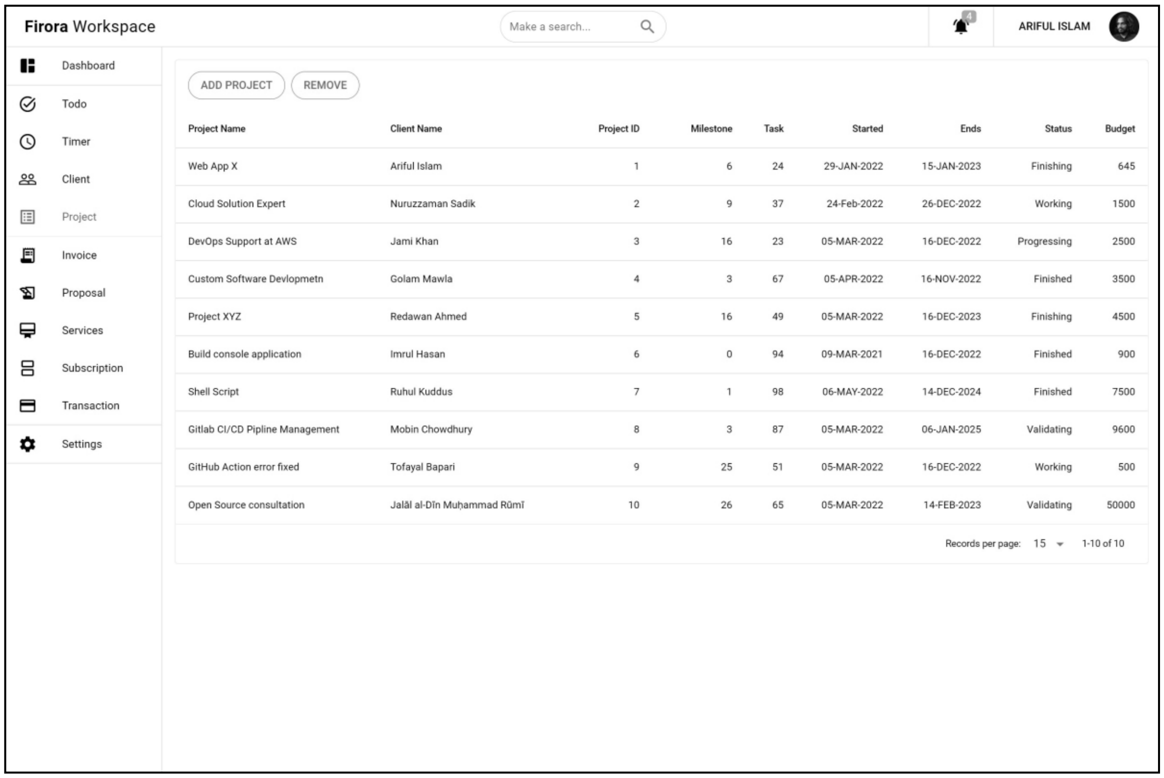
Names	Country	Chat	Send mail	Total Project	Total income
Mobin Chowdhury		3.2	87	0	562
Ariful Islam		6	24	4	87
Nuruzzaman Sadik		9	37	0	129
Jami Khan		16	23	6	337
Golam Mawla		3.7	67	3	413
Redawan Ahmed		16	49	2	327
Imrul Hasan		0	94	10	50
Ruhul Kuddus		0.2	98	7	38
Mobin Chowdhury		3.2	87	0	562
Tofayal Bapari		25	51	25	326
Jalāl al-Dīn Muḥammad Rūmī		26	65	9	54

Records per page: 15 1-11 of 11

Figure 4.11: Client page

4.3 Interaction Design and user Experience (Continued)

Project page allow user to add project and each project assigned to a customer using system. User can add milestone, list of tasked related to the project, budget for the project and the project timeline.



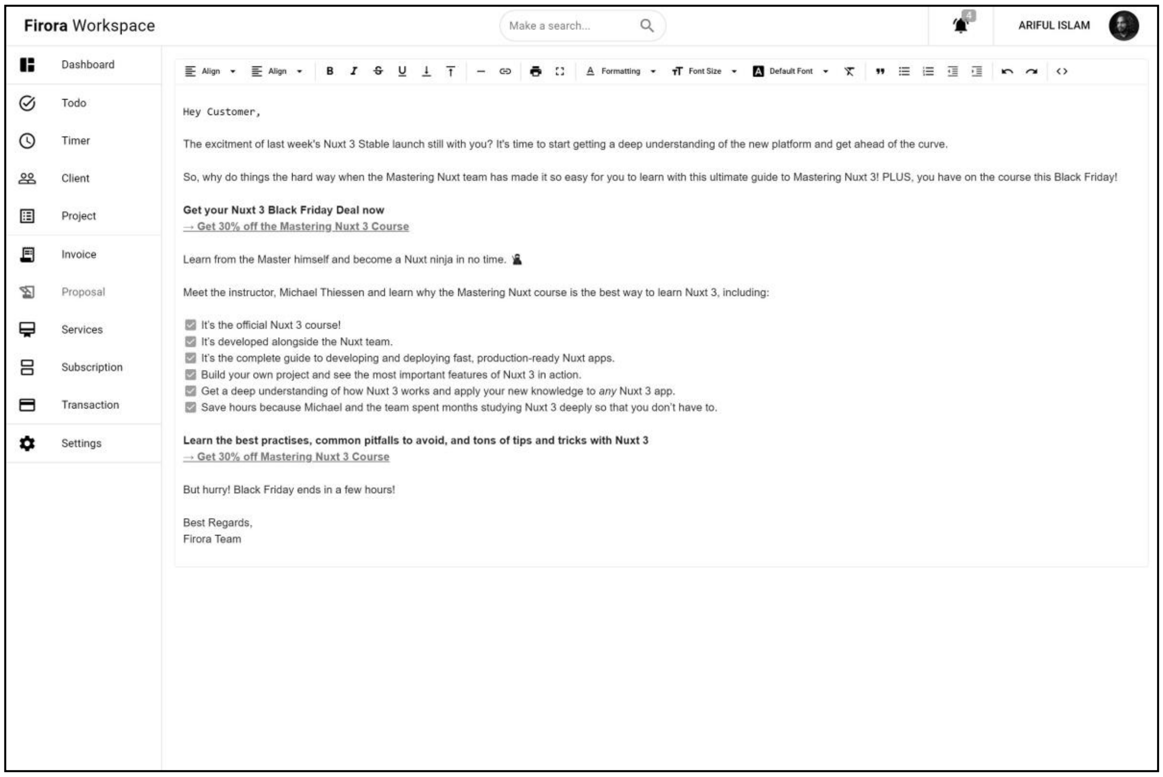
Project Name	Client Name	Project ID	Milestone	Task	Started	Ends	Status	Budget
Web App X	Ariful Islam	1	6	24	29-JAN-2022	15-JAN-2023	Finishing	645
Cloud Solution Expert	Nuruzzaman Sadik	2	9	37	24-Feb-2022	26-DEC-2022	Working	1500
DevOps Support at AWS	Jami Khan	3	16	23	05-MAR-2022	16-DEC-2022	Progressing	2500
Custom Software Devlopmetn	Golam Mawla	4	3	67	05-APR-2022	16-NOV-2022	Finished	3500
Project XYZ	Redawan Ahmed	5	16	49	05-MAR-2022	16-DEC-2023	Finishing	4500
Build console application	Imrul Hasan	6	0	94	09-MAR-2021	16-DEC-2022	Finished	900
Shell Script	Ruhul Kuddus	7	1	98	06-MAY-2022	14-DEC-2024	Finished	7500
Gitlab CI/CD Pipline Management	Mobin Chowdhury	8	3	87	05-MAR-2022	06-JAN-2025	Validating	9600
GitHub Action error fixed	Tofayal Bapari	9	25	51	05-MAR-2022	16-DEC-2022	Working	500
Open Source consultation	Jalāl al-Dīn Muḥammad Rūmī	10	26	65	05-MAR-2022	14-FEB-2023	Validating	50000

Records per page: 15 1-10 of 10

Figure 4.12: Project page

4.3 Interaction Design and user Experience (Continued)

The user will be able to create, transmit, and monitor the proposal's status. This is done to attract potential clients to products and services.



Hey Customer,

The excitment of last week's Nuxt 3 Stable launch still with you? It's time to start getting a deep understanding of the new platform and get ahead of the curve.

So, why do things the hard way when the Mastering Nuxt team has made it so easy for you to learn with this ultimate guide to Mastering Nuxt 3! PLUS, you have on the course this Black Friday!

Get your Nuxt 3 Black Friday Deal now
→ Get 30% off the Mastering Nuxt 3 Course

Learn from the Master himself and become a Nuxt ninja in no time.

Meet the instructor, Michael Thiessen and learn why the Mastering Nuxt course is the best way to learn Nuxt 3, including:

✓ It's the official Nuxt 3 course!
✓ It's developed alongside the Nuxt team.
✓ It's the complete guide to developing and deploying fast, production-ready Nuxt apps.
✓ Build your own project and see the most important features of Nuxt 3 in action.
✓ Get a deep understanding of how Nuxt 3 works and apply your new knowledge to *any* Nuxt 3 app.
✓ Save hours because Michael and the team spent months studying Nuxt 3 deeply so that you don't have to.

Learn the best practises, common pitfalls to avoid, and tons of tips and tricks with Nuxt 3
→ Get 30% off Mastering Nuxt 3 Course

But hurry! Black Friday ends in a few hours!

Best Regards,
Firora Team

Figure 4.13: Proposal page

4.3 Interaction Design and user Experience (Continued)

The service functionality on the system allows the user to create the services they want to sell to their customer. The system allows an elegant way to share their services with their customers.

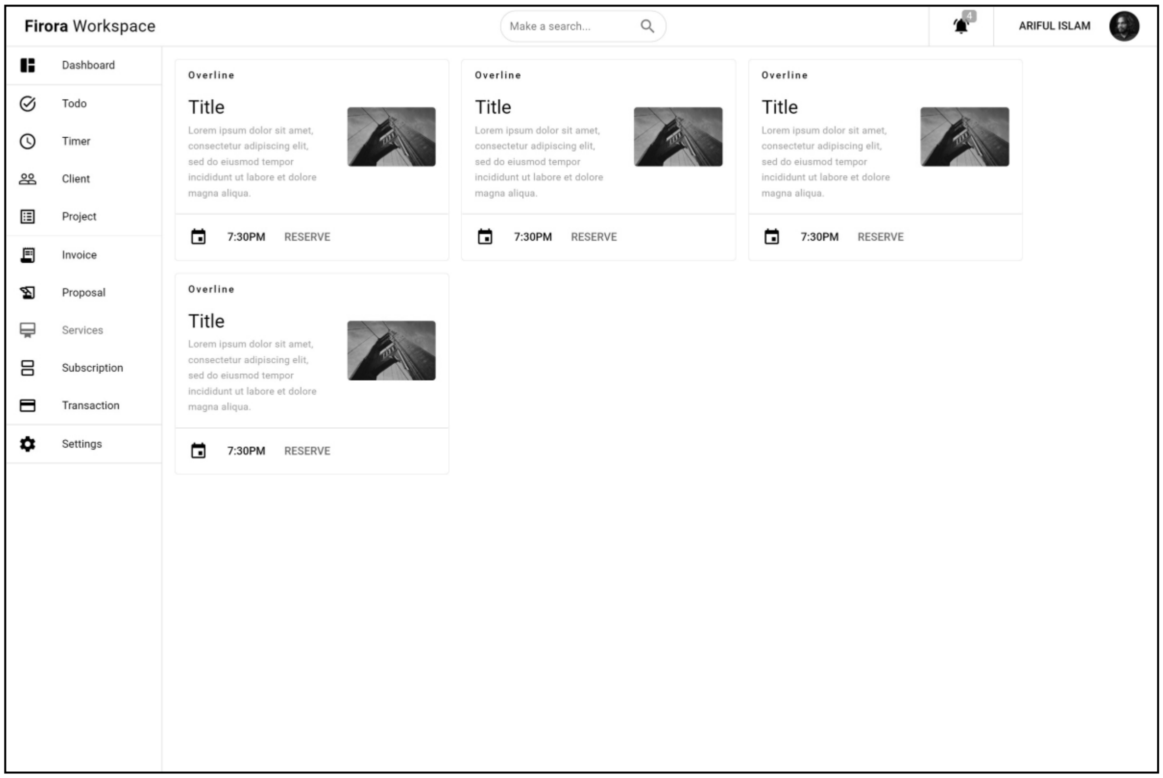


Figure 4.14: Services

4.3 Interaction Design and user Experience (Continued)

Under the settings page user will get to setup his business details. That will allow the software to rebranded for each different companies. Allowing so user will get great way to promote his own brand.

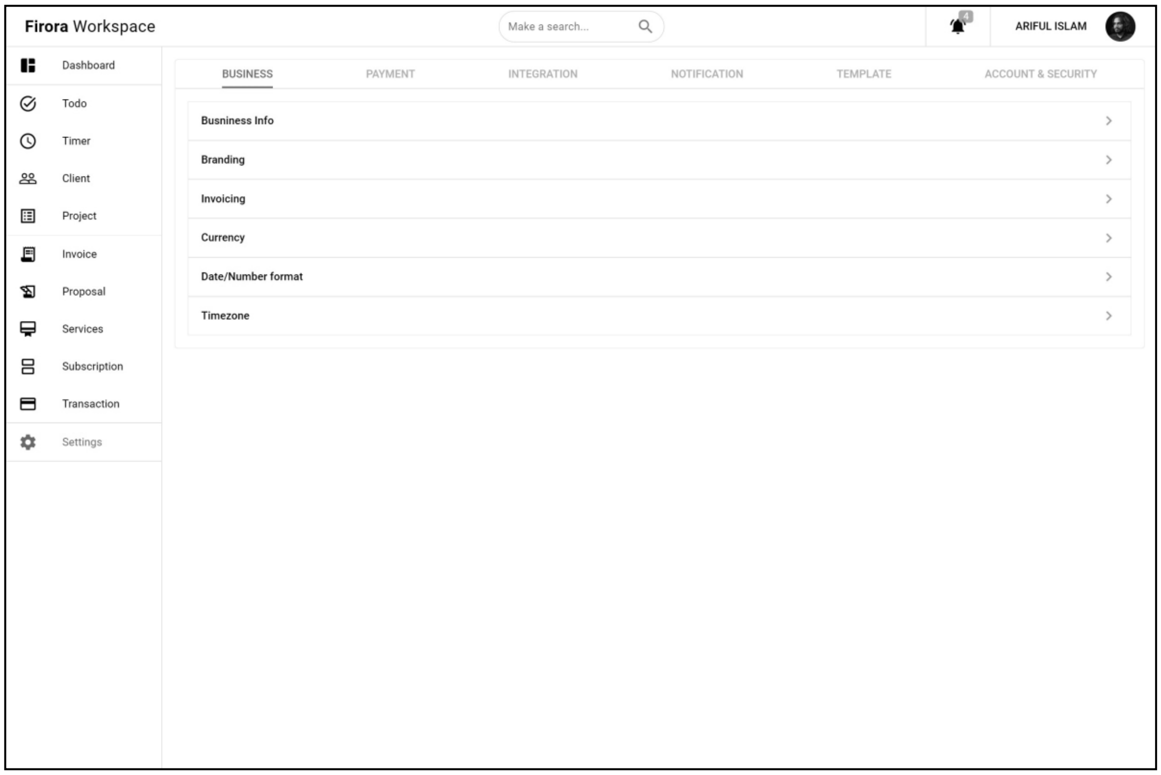
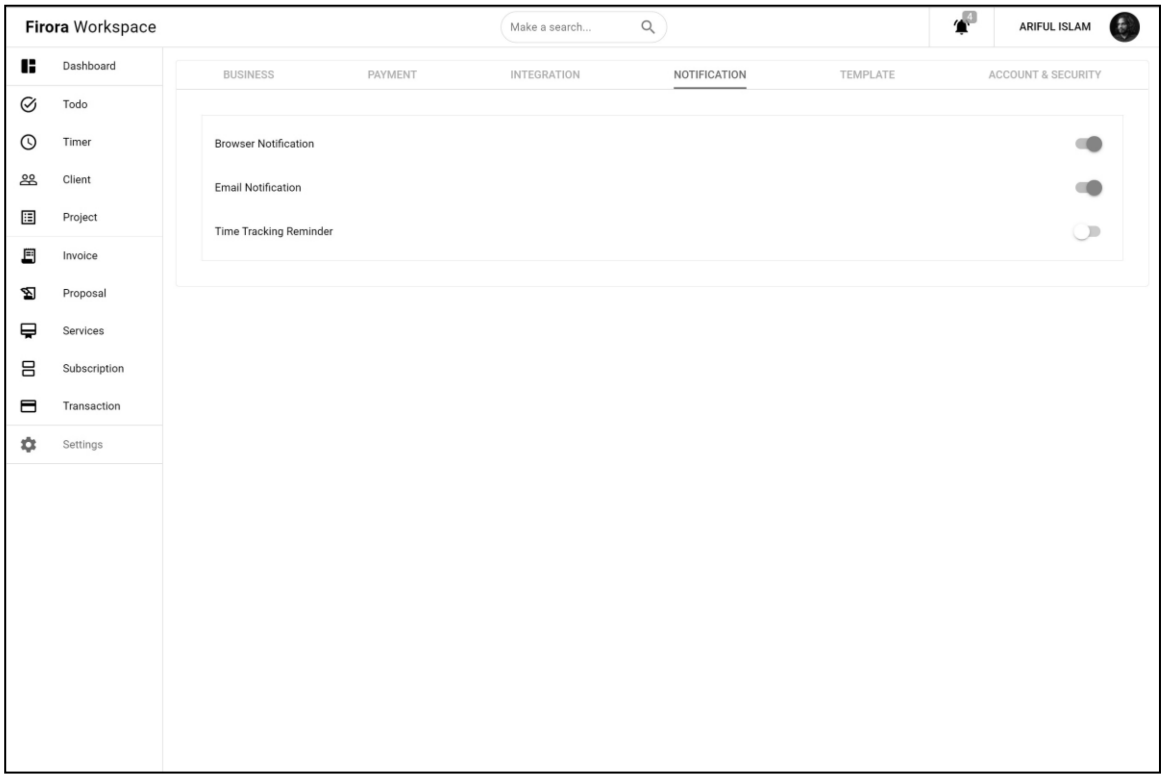


Figure 4.15: Settings – Business Tab

4.3 Interaction Design and user Experience (Continued)

User will receive notification for certain activity regarding his/her concern. From the setting page under the “Notification” tab they can turn on/off what they like get notified for and what not.



Browser Notification ●
Email Notification ●
Time Tracking Reminder ○

Figure 4.16: Settings – Notification Tab

4.3 Interaction Design and user Experience (Continued)

By clicking on the user avatar user can open a pop-up like modal that has the logout option at the end of option list. By clicking on the "Logout" option user can safely logged out and protect from unwanted user accessing his/her account.

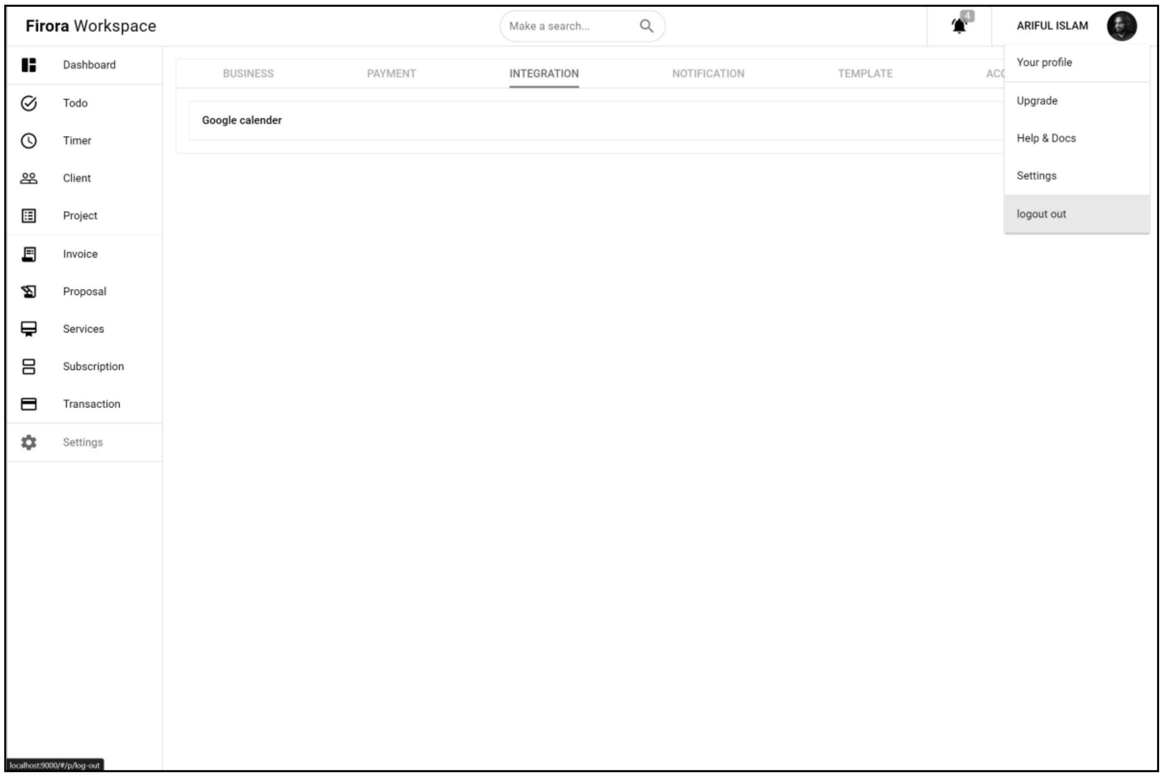


Figure 4.17: Logout modal

4.4 Implementation Requirements

Implementation requirement during development

- NodeJS 18 LTS
- Docker
- Postgres
- Redis
- VS Code (or similar like)
- Node packages for **frontend**

```
"devDependencies": {
  "@intlify/vite-plugin-vue-i18n": "^3.3.1",
  "@quasar/app-vite": "^1.0.0",
  "autoprefixer": "^10.4.2",
  "eslint": "^8.10.0",
  "eslint-config-prettier": "^8.1.0",
  "eslint-plugin-vue": "^9.0.0",
  "postcss": "^8.4.14",
  "prettier": "^2.5.1"
},
"engines": {
  "node": "^18 || ^16 || ^14.19",
  "npm": ">= 6.13.4",
  "yarn": ">= 1.21.1"
}
```

- Node packages for **backen**

```
"devDependencies":  {
  "@types/bcryptjs ":  "^2.4.1",
  "@types/config ": "^3.3.0",
  "@types/cookie-parser":  "^1.4.4",
  "@types/cors": "^2.8.12",
  "@types/express": "^4.17.14",
  "@types/html-to-text": "^8.1.1",
  "@types/jsonwebtoken": "^8.5.9",
  "@types/morgan": "^1.9.3",
  "@types/node": "^18.11.9",
  "@types/nodemailer": "^6.4.6",
  "@types/pug": "^2.0.6",
  "morgan": "^1.10.0",
  "prisma": "^4.6.1",
  "typescript": "^4.9.3"
}
```

Implementation requirement during deployment

- GitLab
- Cloud (like AWS, Azure, GCP)
- Domain
- Build and containerize **frontend**

```
#Dockerfile

FROM node:lts-hydrogen


RUN npm i --global http-server
WORKDIR /app

COPY pack*.json ./
RUN npm i

COPY . .
RUN npm run build


EXPOSE 80

CMD [ "http-server", "dist" ]
```

- Build and containerize **backend**

```
FROM node:lts-hydrogen as base

WORKDIR /app

COPY package.json yarn.lock ./

RUN rm -rf node_modules && yarn install --frozen-lockfile &&
yarn cache clean

COPY . .

EXPOSE 365

CMD ["node", "./app.js"]
```

- Gitlab CI file (auto build)

```
docker-build:
  image: docker:latest
  stage: build
  services:
    - docker:dind
  before_script:
    - docker login -u "$CI-REGISTRY-USER" -p "$CI-REGISTRY-
PASSWORD" $CI-REGISTRY
  # Default branch leaves tag empty (= stable tag)
  # All other branches are tagged with the escaped branch name
(commit ref slug)
  script:
    - |
       if [[ "$CI-COMMIT-BRANCH" == "$CI-DEFAULT-BRANCH" ]];
then
        tag="stable"
        echo "Working on the master branch '$CI-DEFAULT-
BRANCH': tag = 'stable'"
      else
        tag=":$CI-COMMIT-REF-SLUG"
        echo "Running on branch '$CI-COMMIT-BRANCH': tag =
$tag"
      fi
    - docker build --pull -t "$CI-REGISTRY-IMAGE${tag}".
    - docker push "$CI-REGISTRY-IMAGE${tag}"
  rules:
    - if: $CI-COMMIT-BRANCH
      exists:
        - Dockerfile
```

CHAPTER 5

Implementation and Testing

5.1 Implementation of Database

Steps need to implement database:

- Install git
- Clone backend repo
- Install docker
- Install NodeJS 18 LTS
- $ yarn install # (inside project directory)
- $ npx prisma migrate dev --name update # (to migrate the schema to database)
- Docker compose file for database

```
version: "3.6"
services:
  database:
    image: postgres:alpine3.17
    container_name: psql-DB
    ports:
      - "5432:5432"
    volumes:
      - psql-DB:/data/postgres

  caching:
    image: redis:alpine3.17
    container_name: redis-caching
    ports:
      - "6379:6379"
    volumes:
      - redis-DB:/data
volumes:
  psql-DB:
  redis-DB:
```

- $ docker compose –f db-dev.yaml up -d

5.2 Implementation of Front-end Design

Steps need to implement frontend:

- Install git
- Clone backend repo
- Install NodeJS 18 LTS
- $ npm install # (inside project directory)
- $ npx quasar dev
- Output will look like:

```
➜  frontend git:(master) ✗ npx quasar dev

 » Reported at............ 12/3/2022 9:14:02 PM
 »     App     dir................     /home/ariful/firora-
workspace/frontend
 » App URL................ http://localhost:9000/
 » Dev mode............... spa

 App • Opening default browser at http://localhost:9000/
```

5.3 Testing Implementation

Testing database implementation, Docker engine and compose must be present on.

- $ docker ps

```
➜  server git:(master) ✗ dock ps -a
CONTAINER ID   IMAGE             CREATED      STATUS
PORTS                   NAMES
89dbe4d99db0   postgres:alpine3.17   29 days ago   Up 59
minutes   127.0.0.1:5432->5432/tcp   psql-DB
62dba292b0ab   redis:alpine3.17      25 days ago   Up 19
minutes   127.0.0.1:6379->6379/tcp   redis-caching
```

- $ npx prisma studio

```
➜  server git:(master) ✗ npx prisma studio
Environment variables loaded from .env
Prisma schema loaded from prisma/schema.prisma
Prisma Studio is up on http://localhost:5555
```

5.4 Test Result and Reports

Testing API endpoint with Linux curl program. Software like postman can be also used.

Table 5.1: Registration

Test Case ID	TC-01		
Test Case Title	Registration endpoint		
Tested by	Ariful Islam	Execution date: 05-NOV-22	
Description	User must provide password longer than 8 character and valid email		
Command		Expected	Status
curl --request POST \ --url http://127.0.0.1:365/api/auth/register \ --header 'Content-Type: application/json' \ --data '{ "name":"Ariful Islam", "email":"arifulislamat@gmail.com", "password":"SuperPass", "passwordConfirm": "SuperPass"}'		Successful registration and feedback for successful registration.	Passed

Table 5.2: Login

Test Case ID	TC-02		
Test Case Title	Login endpoint		
Tested by	Ariful Islam	Execution date: 06-NOV-22	
Description	User must pre-register on the system		
Command		Expected	Status
curl --request POST \ --url http://127.0.0.1:365/api/auth/login \ --header 'Content-Type: application/json' \ --data '{ "email":"ariful@firora.com", "password":"SuperPass"}'		Successful login and welcome message	Passed

Table 5.3: Forgot Password

Test Case ID	TC-03		
Test Case Title	Forgot password endpoint		
Tested by	Ariful Islam	Execution date: 06-NOV-22	
Description	User must pre-register on the system		
Command		Expected	Status
curl --request POST \ --url http://127.0.0.1:365/api/auth/forgotpassword \ --header 'Content-Type: application/json' \ --data '{ "email":"ariful@firora.com" }'		Successful execution and received a reset link via email	Passed

Table 5.4: Login Validation

Test Case ID	TC-04		
Test Case Title	Login endpoint		
Tested by	Ariful Islam	Execution date: 26-NOV-22	
Description	User must pre-register on the system		
Command		Expected	Status
curl --request POST \ --url http://127.0.0.1:365/api/auth/login \ --header 'Content-Type: application/json' \ --data '{ "email":"ariful@firora.com", "password":"Super pass" }'		Showing warning for using invalid access credential	Passed

CHAPTER 6
Impact on Society, Environment and Sustainability

6.1 Impact on Society

In 2019, the covid19 epidemic swept the globe. Since 2020, remote work has grown significantly. The majority of workers think that hybrid work may help to achieve a harmonious balance between home and professional lives. According to the study, this tendency is not going away, and researchers still think that hybrid work will be the most popular type of employment in the future.

Table 6.1: Growth in remote workers

Timeline	No remote worker on their team	Fully remote team	Share of their workers remote
Pre-COVID	46%	2.3%	13.2%
Post-COVID	6%	20%	56% to 74%

This radical transition to remote teams is a fresh experiment that represents a totally different way of working for the great majority of enterprises. Video conversations have taken the role of in-person meetings, Remote desk software replacing the IT service, and sending a fast Slack message has taken the place of dropping by someone's desk or office.

It is not surprising that people have had to change how they collaborate despite being separated by distance; our poll shows that remote work is effective. Working remotely has gone better than predicted for 57% of recruiting managers, and it has gone as expected for another 36%. It went worse than predicted for only approximately one in ten people.

There are several types of software for various types of remote work, more are on the way. However, In this new age of work-life, I want to mark my contribution by creating an all-in-one software solution that will benefit the vast majority of users and enable them to achieve their goals.

6.2 Impact on Environment

The ecology will be impacted by remote labor, without a doubt. I think it will have primarily good effects. Consider how much less energy will be needed to maintain the massive company infrastructure and how little daily transportation will be needed. We can combat greenhouse gases and their effects by reducing electricity use.

Right about now, you're probably wondering what the hell this has to do with the software. Although there is no direct correlation, creating a better workplace where work can be done remotely would undoubtedly increase the number of individuals who choose to work entirely or partially remotely. And the environment will undoubtedly be impacted by it.

6.3 Ethical Aspects

The internet not only enables people to perform feats of inconceivable strength, but it also facilitates fraud. Since there are various ethical considerations while conducting business remotely, I am paying close attention as I design the system.

For example, imagine to have confidential data of your business that should be only accessed by workers. However, one of your employees hired a stranger to perform a task for him, and as a result, the stranger has access to this information. Therefore, the ethical component is absolutely crucial to our undertaking.

Fortunately, we live in a time when OpenAI and similar big projects are there to assist you. Although we can use all the conventional security measures to make sure that only authorized users can access our program, there will always be some who find a way to get around the security measures. Because of this, we may employ artificial intelligence and machine learning to create a safe and intelligent system.

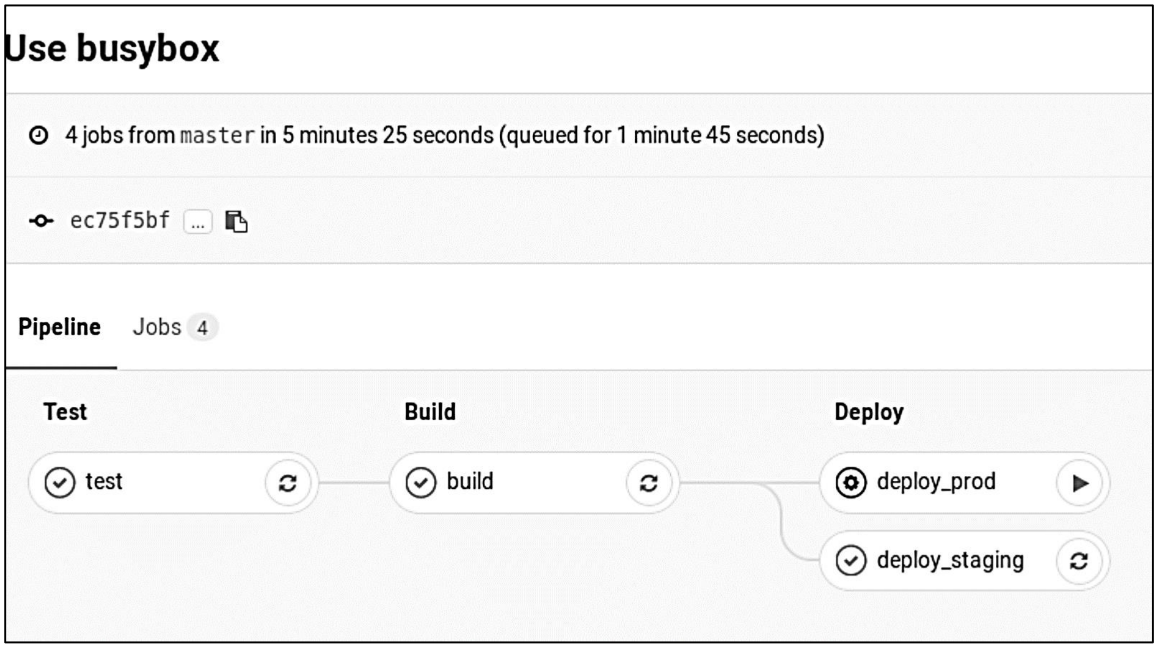
6.4 Sustainability Plan

I adhere to a particular mindset and set of technologies in order to achieve sustainable software development. Below, I provide a summary of my mentality.

Agile is a development process with a set of guidelines for giving customers better software. In the long run, I'm attempting to adhere to Agile development principles, and in the near term, XP.

GitLab is DevOps software that's comes with CI/CD and Project management tools which I think fare sufficient for my project development needs.

To automatically produce Docker images based on merges on the master branch and deploy them on a Kubernetes cluster, I will develop a CI/CD pipeline. Two Kubernetes cluster environments—one for staging and the other for production—will exist. Prior to being released on the production environment, each release will be tested on the staging environment.


Use busybox

4 jobs from master in 5 minutes 25 seconds (queued for 1 minute 45 seconds)

ec75f5bf

Figure 6.1: CI/CD Pipeline example

CHAPTER 7
Conclusion and Future Scope

7.1 Discussion and Conclusion

Well the development process was quite challenging and overwhelming. Nevertheless, I was able to learn so much about programing and development in general. Learning these things was enjoyable, but it also caused irritation when your code didn't function or when it did work but you had no idea why.

I'm attempting to create a program that might revolutionize how small and medium-sized businesses function in the modern era when most activities take place online because the previous solution wasn't created with the need for remote workers in mind.

A wise man would tell you that it all depends on how dedicated and focused you are. But I will add that you require incredible reason and a clear purpose. When you have those, then never loss hope it’s just matter of time when your dream comes true.

7.2 Scope for Further Developments

I am now working on a prototype; when it is released to the client, I will undoubtedly need to add a number of features and code that adheres to industry standards. There will be more requirements and obstacles as the development process expands. However, I will only describe a handful of the things that come to me right now.

- A cross-platform mobile client
- A cross-platform desktop client
- Chat features
- Video conference features
- Email inbox features
- Attendance system for employee
- A Reach Calendar
- Predicting futures with ML and AI

REFERENCES

[1] Jay Mulki, Fleura bardhi, Felicia Lassk and Janye Nanavaty-Dahl “Set Up Remote Workers to Thrive” Researchgate Vol. 51 No. 1, 63-69, Fall 2009

[2] Laker, B., Godley, W., Patel, C. and Cobb, D. “How to monitor remote workers — ethically” MIT Sloan Management Review. ISSN 1532-9194

[3] Yang, L., Holtz, D., Jaffe, S. et al. “The effects of remote work on collaboration among information workers” Nat Hum Behav 6, 164, October 2021

[4] Veronica Popovici, Alina - Lavinia Popovici “Remote Work Revolution: Current Opportunities and Challenges for Organizations” Ovidius University Annals Vol. XX, Issue 1, January 2020

[5] Ausarbeitungen zum Seminar “Rich Internet Applications w/HTML and Javascript” Carl von Ossietzky Universität Oldenburg Vol. X, No. X, 01-33, February 2017

[6] Wikipedia, available at <<https://en.wikipedia.org/wiki/Workspace>>, last accessed on 26-04-2021at 02:00 AM

[7] Salesforce, available at <<https://www.oracle.com/erp/what-is-erp/>>, last accessed on 19-09-2022 at 03:35 AM

[8] Express, available at <<https://expressjs.com/en/4x/api.html>>, last accessed on 09-10-2022 at 09:45 AM

[9] Vue.js, available at <<https://vuejs.org/guide/introduction.html>>, last accessed on 07-01-2023 at 11:00 PM

[10] Docker, available at <<https://docs.docker.com/desktop/>>, last accessed on 11-11-2022 at 09:26 PM

[11] PostgreSQL, available at <<https://www.postgresql.org/docs/>>, last accessed on 05-01-2023 at 12:39 PM

[12] JWT, available at <<https://jwt.io/introduction>>, last accessed on 08-07-2022 at 09:46 PM

A WEB-BASED MANAGEMENT SYSTEM FOR REMOTE WORKERS FIRORA WORKSPACE

ORIGINALITY REPORT

10%	10%	1%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	5%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	3%
3	forum.gitlab.com Internet Source	1%
4	kandi.openweaver.com Internet Source	1%

Exclude quotes	Off	Exclude matches	< 1%
Exclude bibliography	On		