

DESIGN DOCUMENT

Project

HOME PRO

Client

AK COMPUTER SOLUTIONS INC.

Course

CMPS 303 Object Oriented System Analysis and Design

Instructor

Ali Moussa

Team: Dynamic Developers

**Gao Liu | Scott Normore | Saksham Ohri | Madhu Madhavan | Eze
Adiele | Liam MacDiarmid**

Table of Contents

1. Purpose of Document.....	2
1.1 Documentation Standards.....	2
2. User Requirements	2
2.1 Business Overview and Objectives	2
2.2 Project Overview	3
3. System Requirements	4
3.1 Fact Finding Methodology	4
3.2 Functional Requirements.....	4
3.3 Use Case Diagram	6
3.4 Use Case Descriptions and Scenarios.....	7
3.5 Non-Functional Requirements	11
3.6 System Interface Requirements.....	12
3.7 Maintainability and Administration Requirements	13
3.8 Usability Requirements	14
4. Diagrams.....	15
4.1 Interaction Sequence Diagram	15
4.2 State Machine Diagram	19
4.3 Activity Diagram.....	20
5. System Design	21
5.1 Layered Architecture.....	21
5.2 Hardware Architecture and Platform.....	22
5.3 Software Platform	23
6. Interaction Model	24
7. Persistence Model.....	50
7. Class Diagram.....	52
7. Interaction Sequence Diagram.....	59
8. Project Management.....	61
8.1 Schedule	61
8.2 Team Configuration	61
8.3 Project Standards and Procedures	62
8.3.1 Project Procedures.....	62
8.3.2 Project Standards.....	62
9. Glossary	64
10. Index	67
11. Appendix A: Data Dictionary	69

1. Purpose of Document

This document aims to help readers have a better understanding of our Home Pro project. Requirement document depicted all the essential information that were gathered for development and explained the project in major parts. First part user requirements stated our clients' needs regarding the new system and defined the objectives and scope of this project. The second part system requirements, we defined the subcategories such as functional requirements, non-functional requirements, system interface requirements, maintainability and administration requirements, and usability requirements. In this document we discuss these categories in detail along with usability requirements that includes use cases and use case diagrams to describe the new system.

In addition to the user and system requirements we discuss the system design process and procedure that we will be implementing in this project. An overview of the layered architecture involved, platforms such as hardware and software that would be used in the development of the project, interaction model that talks about the style, desired user support and system feedback. Persistence model that shows the data management process involved along with conceptual and internal schema

1.1 Documentation Standards

- Times New Roman font is used in this document. Since it improves both screen and print readability, we decided to use this as our primary font.
- We have used Software Ideas Modeler (SIM) tool for creating UML diagrams like Use Case diagram, Sequence Diagram, State Machine and Activity diagrams.
- For scheduling our tasks and managing deliverables we are using Gantt chart and we will create it using Microsoft Projects.

2. User Requirements

2.1 Business Overview and Objectives

The client is AK Computer Solutions, an IT services company. The company provides various IT-related services and goods, and operates primarily within the Calgary, Alberta area. The company's mission is to become a premier provider of IT solutions, services, and goods to businesses and individuals. Their major services include computer repair, emergency disaster recovery, virus malware removal, network solutions, and remote support. They do estimates and deliver repaired products for free. Apart from providing IT related services, our client intends to expand to small home service jobs around Calgary. The client wants to achieve this by creating a web application that will connect customers with contractors quickly and efficiently. This new application would allow them to retain their existing customers and regain new customers by providing access to services that are not related to information technology as well as create a database of all client and technician activities and their monetary transactions.

2.2 Project Overview

2.2.1 Statement of the Problem

Individuals looking for home and other basic services often become overwhelmed with the number of companies offering varying services and solutions. It is inefficient and frustrating for individuals to look up companies one-by-one, contact them for a quote, and go through the whole process of explaining their needs to each company/provider. They have a hard time finding the appropriate service providers for their needs. Another major issue is regarding payment associated with services they need. They prefer to reach an agreement on the amount they are supposed to pay for the required service at the time of finalizing the job order so that they can budget accordingly. To eliminate these concerns and to provide ease of selecting an appropriate technician for the job, we need to create a user-friendly dynamic web application that can be accessed and viewed across multiple browsing platforms. The application should allow our clients, new and existing customers an easy navigable experience in finding a technician to resolve their issue and make necessary payments easily when the job is complete.

The technicians should be able to find these requests from the customers and fulfil them. They should be able to use the application to keep track and manage these jobs. They should also be able to track their performance through revenue, rating, and other metrics.

Lastly, administrators should be able to manage which technicians can fulfill each job, pay the technicians for their work, and refund payments when needed. They should also be able to track the revenue of the company/application and be able to look at each technician's performance data.

2.2.2 Project Scope

The goal for this project is to create a system where clients can book appointments with service technicians, and that an admin can manage these technicians. Broadly, the scope of the project is to develop a web-based platform that connects individuals with service providers ("technicians") who can fulfill their needs. This application is being made for "AK Computer solutions Inc". They requested that this system should be able to operate on mobile devices and should be able to meet their business needs. These needs include being able to pay technicians, track transactions, and provide financial and performance data. We plan on delivering this through a full stack application. This application will include a nice-looking website (that can be viewed dynamically on a mobile device too), a database that can store the necessary data, and a working server application.

The customer will be able to visit the website as a guest (without using login credentials), browse through various services offered and place a request using all their contact information and a brief on the issue they need resolved in a certain time frame. These details are entered into a form and submitted. The technicians on the back end will be able to review these submitted forms and respond to the one they feel they can provide service to. A communication channel is then opened between the technician and the customer concerned to further understand the customer's requirements and submit a quote for the job. The customer then accepts the job and agrees to make the payment once the job is complete. Administrative activities will also be incorporated in this application with access privileges to review transactions, modify data of users like technicians and customers, generate feedback facilities, payment transaction logs, termination of contracts and other metrics like ratings and content management.

To do this task, we are given 8 months (2 semesters) worth of time. For the first semester, we plan on defining the requirements and coming up with the design. This will include interviewing with the client in addition to researching the best technologies and practices to use in our application. We will be conducting multiple meetings with the client in addition to making multiple and accurate diagrams for our

application. In the second semester, we plan on implementing our design along with rigorous testing. This will include coding based on the designs we did during our first semester in addition to testing our application. We will also verify from our client if our application meets their requirements, and make changes as needed.

2.2.3 System Environment

The scope of business of our client is mainly providing IT related services, however, they are planning to develop Home Pro system to expand their business scope. Home Pro is a platform to bridge customers who are living in Calgary and need home services with some third-party technicians. The system will therefore operate in an environment where it is only administered by the client but does not overlap with the client's main operations. The client will administer and host the system on their own hardware, such as an internal server, or use cloud service. For instance, our client is considering using Amazon Web Services (AWS)/Microsoft Azure to deploy this system.

2.2.4 Current System

They currently have a website developed and hosted using 'wix.com', a cloud-based web development service that allows users to create HTML5 websites and mobile sites using online drag and drop tools. They use this website to market their IT solution services. The new web application being created will not be an extension to this website they currently have. This will be a completely new system that will need infrastructure set up both locally in their premises and the cloud.

3. System Requirements

3.1 Fact Finding Methodology

We decided to interview key stakeholders in our client company to find facts and gather requirements. Two main decision makers were contacted via email and communicated the need for an interview. After their approval, we scheduled a mutually agreeable meeting time and date through Microsoft Teams. The interview process ran for about an hour and half and we managed to record the meeting as per client's approval. All questions to be asked were transcribed by our team and their responses were documented. We agreed to meet and collect information as and when they arise with the same decision makers, following the same meeting process. Recording the team meetings is a way of both parties signing off on the agreed terms and conditions laid out during the meeting. An NDA was drafted by our clients and signed by all application developers, legally binding in contract throughout the course of this project and beyond. In addition, we can contact our client through email for further clarifications or meetings if needed.

3.2 Functional Requirements

Information about the user needs of the project was retrieved from the client himself. To do so, we asked questions based on the information needed for this document. We listed most of those questions in a word document and got the answers for them during the interview.

A few of the responses we received are listed below

- The app will have three main dashboards: one for the customer, one for the technician, and one for the admin.
- The customer interface will allow users to book a service, contact the technician, and leave a review.

- The technician interface will allow technicians to look up available jobs, look at currently ongoing jobs, request payment from customers, and to view their personal performance through different metrics. These metrics include revenue, jobs taken, and star rating.
- The admin interface will allow the management of technicians. This will include the addition, removal, and permissions of what jobs a technician can access/view. In addition, the admin will have the ability to add and remove customers. Also, the admin will be able to view past payments and refund any payment. Lastly, the admin will also be able to view financial planning data such as revenue earned, company average performance rating, and individual technician revenue and their star rating.
- The services provided by this company are home technical services. This includes Heating and cooling, plumbing, electrical, painting, handyman services, and appliance repairs.
- Each service will show details including rates, descriptions, and sometimes video.
- Service to be made available only in Calgary
- Images and pictures and logo will be provided by the Client.
- Customers to pay for the services offered through the application's payment gateway.
- The new application will not be linked with their existing application and will have a fresh look and feel
- The application will have 3 main entities, Admin, Technician and Customers
- Admin will have the ability to manage customers and technicians (including the privileges of a technician), pay technicians, refund customers, manage customers and technicians, and view the reports/financial planning data.
- The privileges that a technician has (and the admin can manage) include the types of services/jobs the technician can fulfil. For example, a plumber who is only proficient in plumbing should be able to access the plumbing job, but not a painting job.
- Technicians should be able to accept and take on jobs and update the status of a job (complete, in progress, accepted, payment pending). Also, the technician should be able to look up their job schedule and contact the customers. Lastly, the technician should be able to look at their performance through metrics such as revenue earned, jobs done, and star rating.
- Technician will have certain privileges as allowed by the admin
- Customers will have the ability to post reviews and leave feedback, choose and request services, and track status of a job
- Customers should have multiple ways to pay (debit card, credit card, cash, PayPal)
- Application should have an internal search engine for customers to browse for any product pricing or more information to describe their issue.

3.3 Use Case Diagram

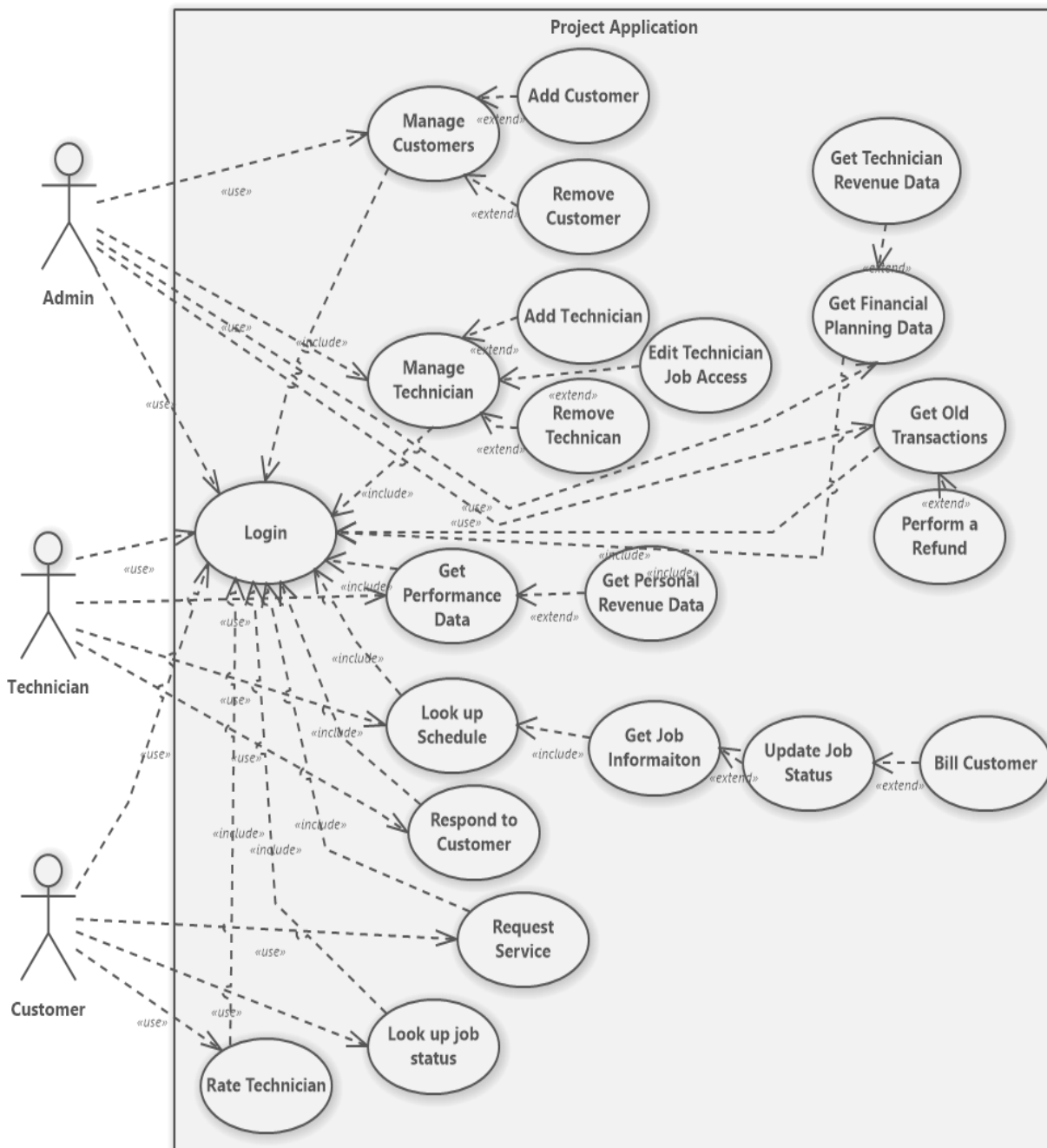
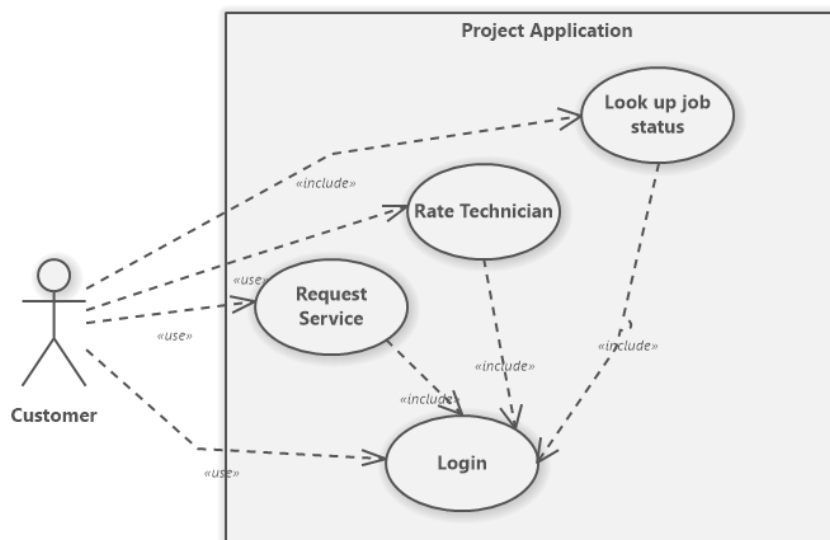


Figure 1. Use Case Diagram

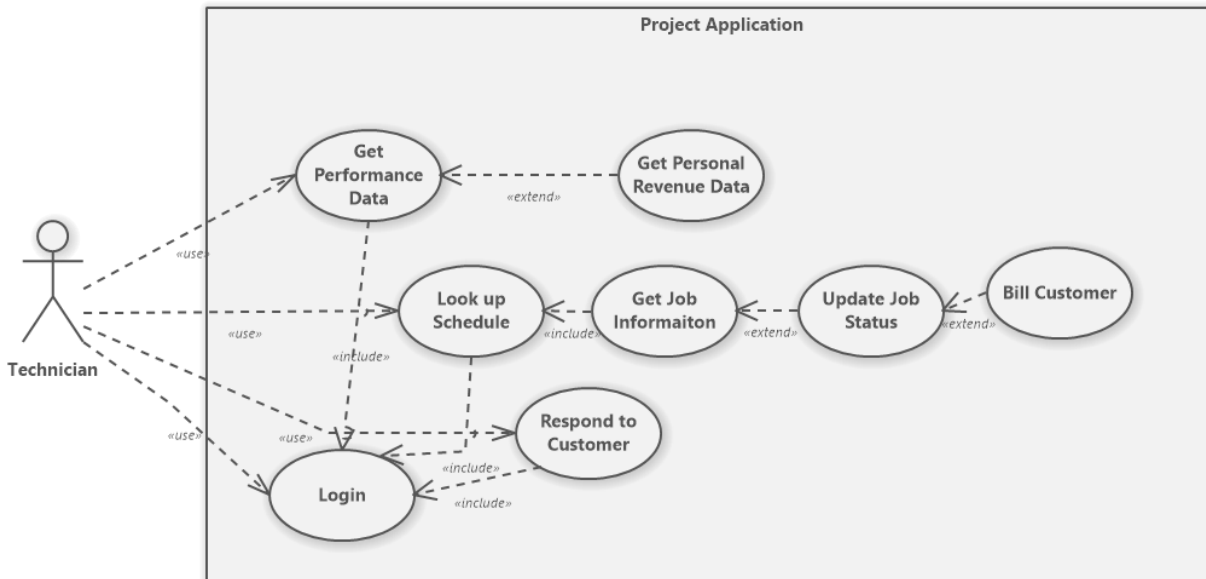
3.4 Use Case Descriptions and Scenarios

- **Application Actors:** Customer, Technician, Administrator.
- **Customer Responsibilities:** The end user that uses the application to get service and solve their household technical problem
- **Technician Responsibilities:** The other user that uses the application to find work
- **Administrator Responsibilities:** Monitors the application, performance of technicians, adds and removes accounts, manages the web application.
- **Customer Use Cases:** Request Service, Rate Technician, look up job Status
- **Technician Use Cases:** Respond to customer, lookup schedule, get job information, bill customer, get performance data, get personal revenue, update job status
- **Administrator Use Cases:** Add Technician, Remove Technician, Add Customer, Remove Customer, manage technician job restrictions and access, pay technician, get technician revenue data, get company revenue data, get past jobs, send refund



Actor: Customer Use case: Request Service	
Starting Interface: Login Page	
Input	Output
1.Username and Password	Customer Landing Page
1.A. Incorrect Credentials	Display error message on login page
2.Select Service Pressed	Page with Services
3.Vent Cleaning Selected	Vent Cleaning Form
4.Submit Form	Schedule page
4A. Incomplete/incorrect Information	Display Error prompts on form
5.Time Selected	Database updated with info. Acknowledgement page sent to user
5.A. Invalid time requested	Display error message, prompt user to try again

Actor: Customer Use Case: Rate Technician	
Starting Interface: Customer Landing Page	
Input	Output
1.Select Past Jobs	Display Past Jobs
2.Select A Past Job	Information on Past Job
3.Select “Rate Pro”	Pro Rating Form Displayed
4.Form Submitted	Rating Updated, Acknowledgement page sent to user
4.A Invalid Data/Missing Data	Display error message, prompt user to try again



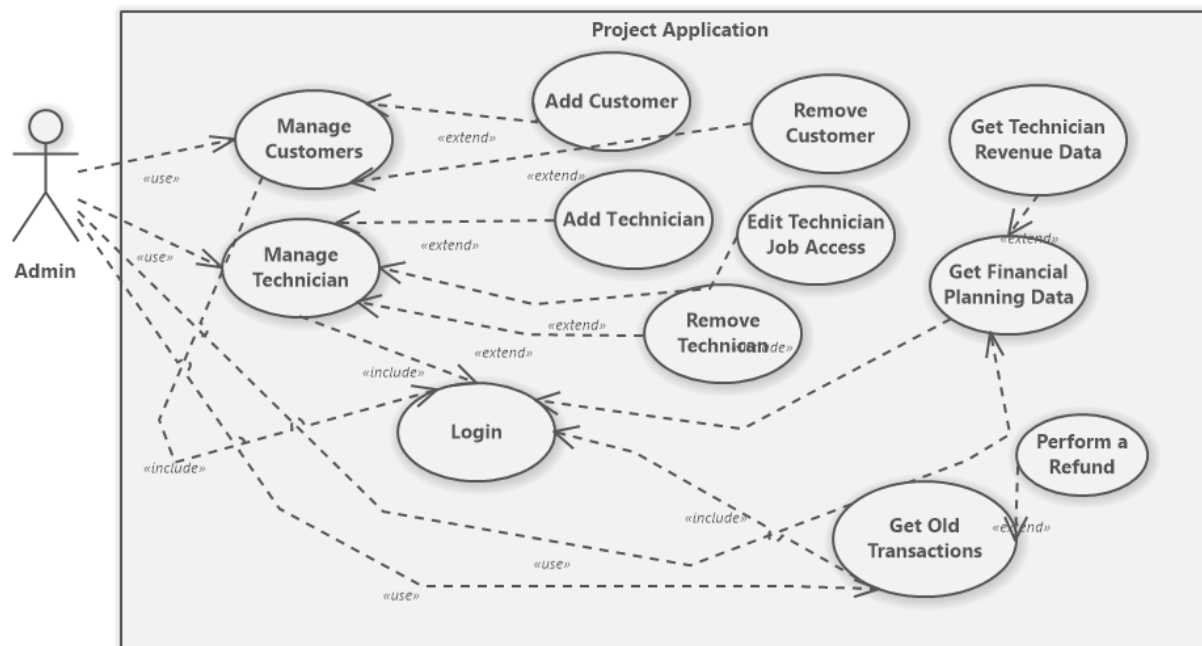
Actor: Technician Use Case: Respond to Customer	
Starting Interface: Login Page	
Input	Output
1.User inputs username and password	Technician landing page
1.A Incorrect Credentials	Display error message on login page
2.Customer Message Alert Pressed	Customer Message Displayed
3.Reply button pressed	Replay Form Displayed
4.Submit button pressed	Message sent back to customer
4.A. Missing Fields Submitted	Error Message displayed. Prompt user to try again

Actor: Technician Use Case: Look Up Schedule/Look up Job information	
Starting Interface: Technician Landing Page	
Input	Output
1.User Selects Job Schedule	Job Calendar Displayed. Show's User's Schedule
2. User Selects a Job from Calendar	More information about the job is displayed (such as address)

Actor: Technician Use Case: Bill Customer	
Starting Interface: Technician Landing Page	
Input	Output
1. User Selects Job Schedule	Job Calendar Displayed. Show's User's Schedule
2. User Selects a Job from Calendar	More information about the job is displayed (such as address)
3. User Selects Job to complete	Confirmation Asked
4. User Confirms	Customer Billed

Actor: Technician Use Case: Look Up Personal Revenue	
Starting Interface: Technician Landing Page	
Input	Output
1. User Selects Personal Performance Page	Personal Performance Page opened
2. User Selects the "revenue" metric	Performance information displayed based on personal revenue

Actor: Admin Use Case: Add Technician to Database	
Starting Interface: Login Page	
Input	Output
1. User enters Credentials	Admin Page Displayed
1.A User enters wrong credentials	Display Error Message on login page
2. User Selects manage technicians	Technicians Page Displayed
3. User Select "Add technician"	New Technician Form Displayed
4. User fills out info, Submit pressed	Technician Added. Acknowledgment displayed
4.A Invalid input or missing fields	Error Message Displayed.



Actor: Admin Use Case: Remove Technician to Database Starting Interface: Admin Page	
Input	Output
1. User Selects manage technicians	Technicians Page Displayed
2. User Select “Remove technician”	Remove Technician Page Displayed
3. User Selects Technician	Confirmation Box displayed
4. Confirmation Filled	Technician Deleted

Actor: Admin Use Case: Add Customer to Database Starting Interface: Admin Page	
Input	Output
1. User Selects manage technicians	Customer Page Displayed
2. User Select “Add technician”	New Customer Form Displayed
3. User fills out info, Submit pressed	Customer Added. Acknowledgment displayed
3.A Invalid input or missing fields	Error Message Displayed.

Actor: Admin Use Case: Remove Customer to Database Starting Interface: Admin Page	
Input	Output
1. User Selects manage technicians	Customers Page Displayed
2. User Select “Remove technician”	Remove Customers Page Displayed
3. User Selects Technician	Confirmation Box displayed
4. Confirmation Filled	Customer Deleted

Actor: Admin Use Case: Manage Technician Access Starting Interface: Admin Page	
Input	Output
1. User Selects manage technicians	Technicians Page Displayed
2. User Select “manage technician”	New Technician Form Displayed
3. User Selects Desired Technician	Current Tech’s Specializations and Access listed
3. User Modifies the data. Submit is pressed	Technician Information updated
3.A Invalid input or missing fields	Error Message Displayed.

Actor: Admin Use Case: Pay Technician Starting Interface: Admin Page	
Input	Output
1. User Selects manage payments	Payments Page Displayed
2. User Select “Pay technician”	Technicians Displayed
3. User Selects Technician	Technician Information displayed, along with how much he is owed
4. User Selects “Pay Amount Due”	Technician is paid the amount
4.A1 User Selects “Modify Amount Due”	Modified Pay Form Displayed
4.A2 User Sends Modified Amount	Technician is paid the modified amount
4.A2.A User inputs invalid data	Error Message Displayed

Actor: Admin Use Case: Get Technician Revenue/Company Revenue Starting Interface: Admin Page	
Input	Output
1. User Selects Financial Planning Page	Financial Planning Page Displayed
2. User Select Revenue	Company Revenue Displayed
3. User Selects “Revenue on Technician”	List of Technicians and their revenue is displayed
4. User Selects desired Technician	Technician revenue information displayed

Actor: Admin Use Case: Get Past Transactions/Refund Service Starting Interface: Admin Page	
Input	Output
1. User Selects Past Payments Page	Past Payments Displayed
2. User Enters Data to Filter By	List Filtered
3. User Selects Desired transaction to refund	More information on transactions displayed
4. Presses “Refund Transaction”	Transaction is refunded

3.5 Non-Functional Requirements

The non-functional requirement for this application is as follows:

- **Secure Logins** – This is an important feature requested by the client because in our application customers are going to be able to use their accounts to pay for these services. Not having secure logins would open these customers for the possibility of paying for services that they did not request. Apart from that, having a secure login will ensure that customers' information is protected from any unauthorized activity on the application.
- **No security flaw that can lead to leak of data** - We are storing Customer Data, and potentially customer payment information. Because of this, our server-side application must be able to safely hold this information with no risk of the information being compromised. Afterall; companies are responsible for the information they decide to keep on record.
- **Secure payments** –With the growth in technology, many applications are now opting for easy and secure online payment options. Payments should not be the cause of a security breach that can compromise the customer’s bank account. In addition, they should be confidential. Applications offering payment options within the application should follow the most secure way of dealing with online payments.
- **Ease of Usability** – Customers should be able to get a hold of these services easier than its competition. If the process of using the application to get a home service took longer than the use of the application would not be desirable. A user-friendly design will be opted throughout the application which will enhance the user experience and will get the job done easily. A complete user guide will be provided to better understand the functionality of the application. All the components of the application will be thoroughly tested to identify any error in the functionality before the user gets their hands on the application. Therefore, providing a quality product to gain trust of the user.
- **Must be able to use without user Account** – Falling in line with ease of use, if the customer does not wish to have an account, they should still be able to book a service with relative ease. A Customer who deems an account to hard or time consuming to make is a lost customer.
- **Displaying Job information according to Technician roles-** For technicians looking for jobs on the application will be provided with a filtered list of jobs matching their expertise and skills. Hence, creating a user-friendly environment throughout the application.

3.6 System Interface Requirements

These are the systems that we are planning to use for the project

- **Server Provider / Web hosting service:** Microsoft Azure
- **Server Application:** Coded in Java (Spring) or JavaScript (Node.js). We are weighing options, but are leaning towards Node.js
- **Database:** Mongo DB or the Oracle database. We are still weighing options here.
- **Frontend Interfaces:** HTML and CSS. We are thinking about using bootstrap can be used here to ensure device responsiveness. We are also considering React.
- **Payment System:** Direct Transfer, Google Pay, Bank API, Direct Payment, etc...

Here is the current idea of how those systems will interact with each other.

Simple Interface/System Diagram

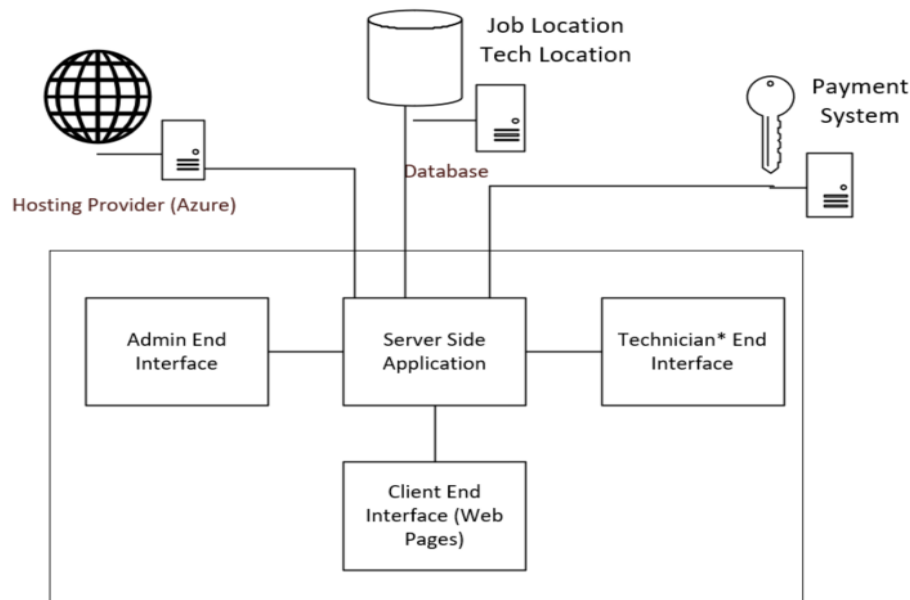


Figure 2. System Interface Diagram

The idea is that the server-side application will handle the different requests from the interfaces. It will fulfil these requests by fetching needed data from the database and using the payment system when needed. The application itself will be hosted through a web hosting service (in this case, Azure).

3.7 Maintainability and Administration Requirements

Maintainability:

- This is considered, inherent to the system design, ensuring the ease and accuracy of maintenance tasks within the project. While this concept emphasizes the importance of timely integration of design, most features for maintainability will be decided during project design as much information is not available now but will become clearer during project creation.
- This will be determined after project completion by ensuring administrators are trained on design application used in writing the program to enable them easily to maintain the program.
- A program structure/design diagram will be provided for the user to enable them trace step by step
- A procedure will be created for resolving or adding to the program when required.
- A helpline will be provided for the client should contact be required.

Administration:

- Admin requirements so far include ability to manage backups.
- Ability to oversee user activities and allocate jobs.
- Makes specific arrangements to stop technicians from refusing requests
- Privilege to access application services belonging to financial services.
- Make payment transactions with respect to technicians and other third-party users
- Modify how the compensation is split up between the technician and the company
- Can modify all user accounts with respect to technicians and customers.
- Add users to user groups, create new roles, assign new roles to users and control user access
- Responds to incidents and problems escalated due to account transactions
- Verify the order request raised by the customer. Admin checks the work and selects or rejects it accordingly.
- Manages promotions and discounts
- Add or remove services offered by the company

3.8 Usability Requirements

- User interface for users must be easy to use, clearly worded text and easy to navigate to other links/sites. (Login Optional)
- User interface should have the option to review services and request more information without having to necessarily create a profile.
- User interface should have review options after a service has been completed.
- The technician interface will have options to accept user jobs request on Jobs.
- The technician interface will be able to respond to user requests for services.
- The technician interface can respond to Admin requests of any kind.
- Admin should be able to restrict the technician's view, so as to make it easier for the technician.
- Admin should have a view of how the traffic flow is to the website.
- Admin page will have access to a list accepted job and technician handling the job, of processed/completed jobs, be able to view jobs that are yet to be completed and to follow through with technicians if job is delayed.
- Admin interface will have an update/edit function, a delete option of the services or products currently being displayed by the company.

4. Diagrams

4.1 Interaction Sequence Diagram

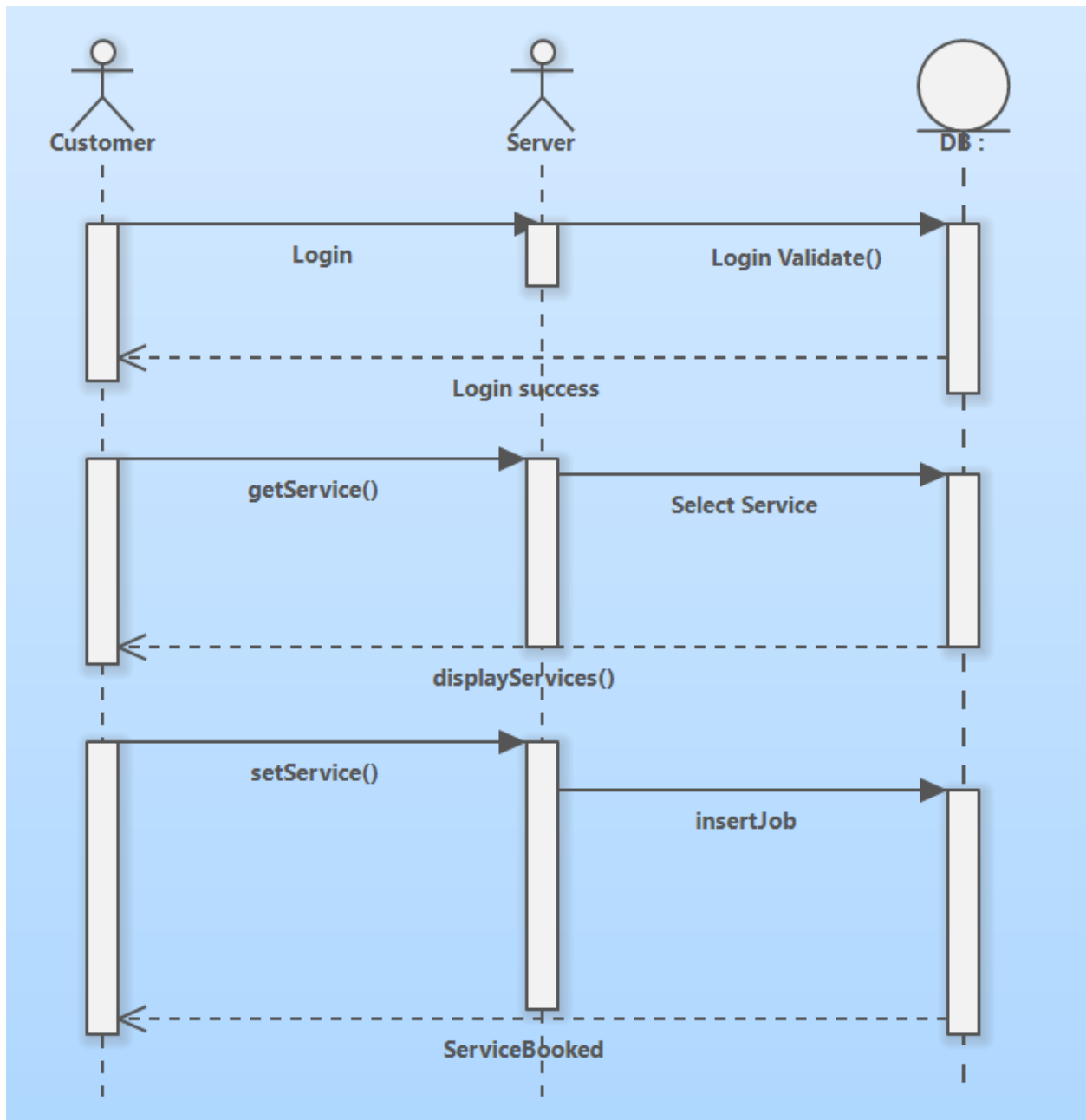


Figure 4. Service Sequence Diagram

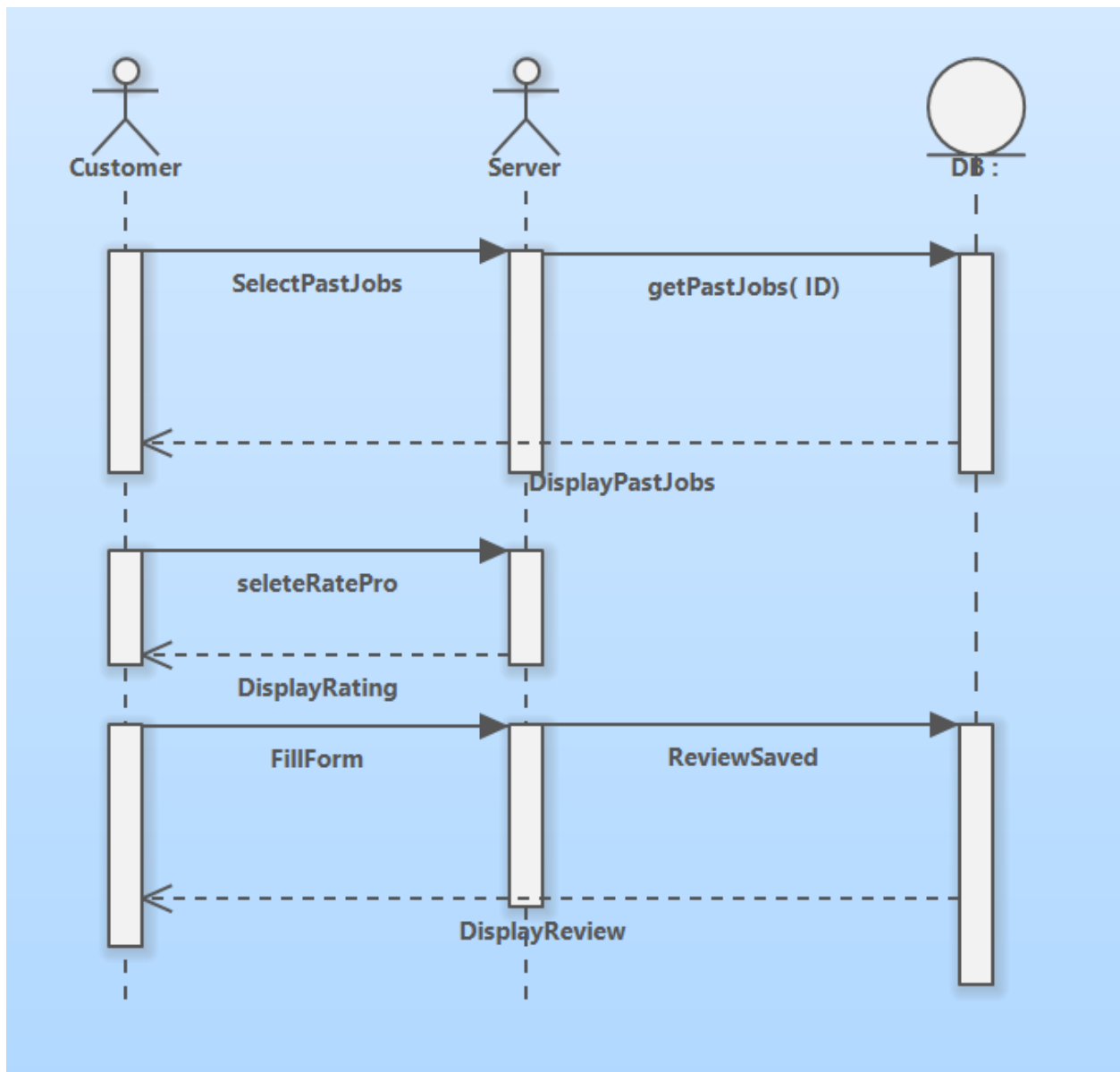


Figure 5. Review Sequence Diagram

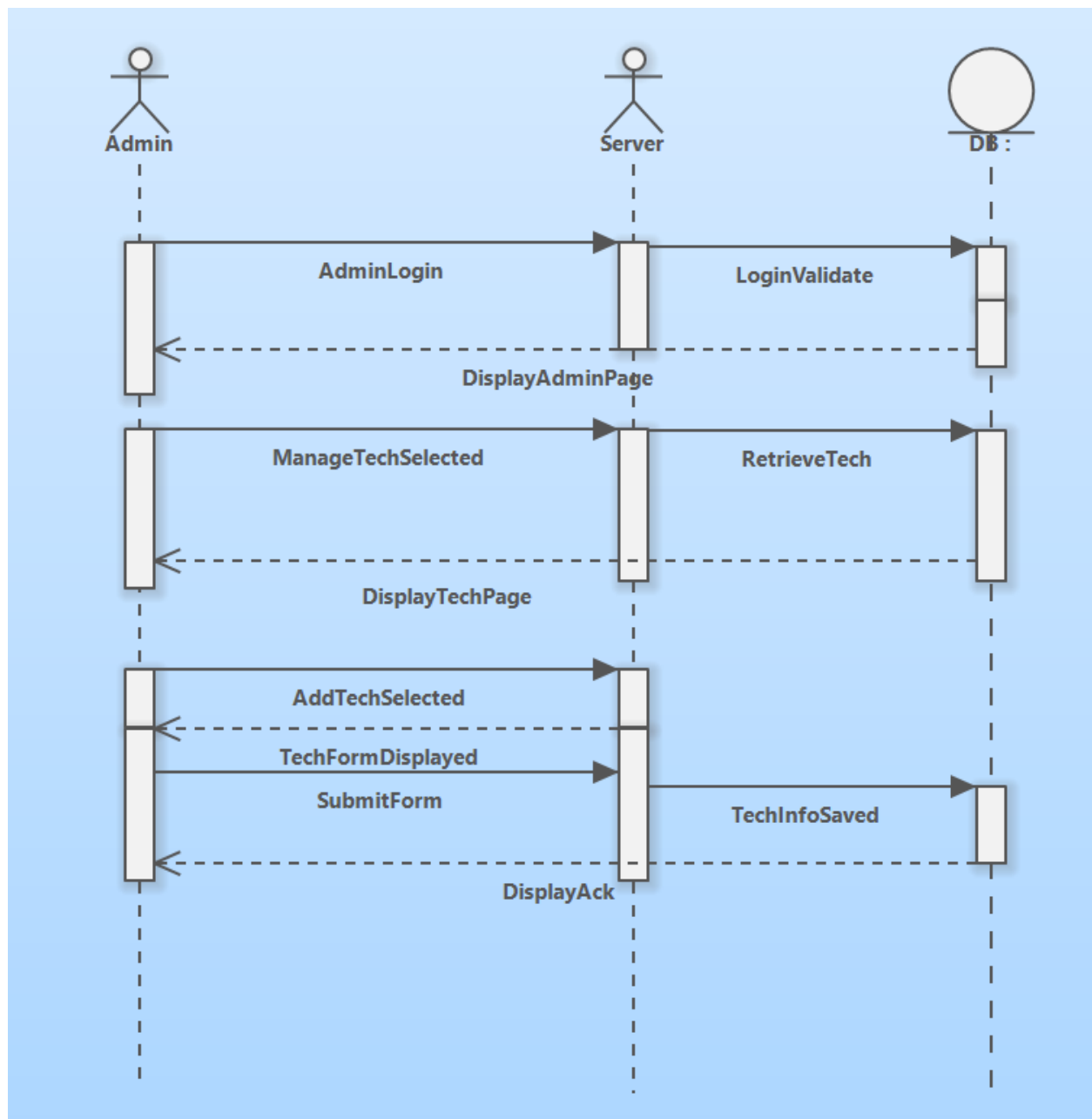


Figure 6. Adding Technician Sequence Diagram

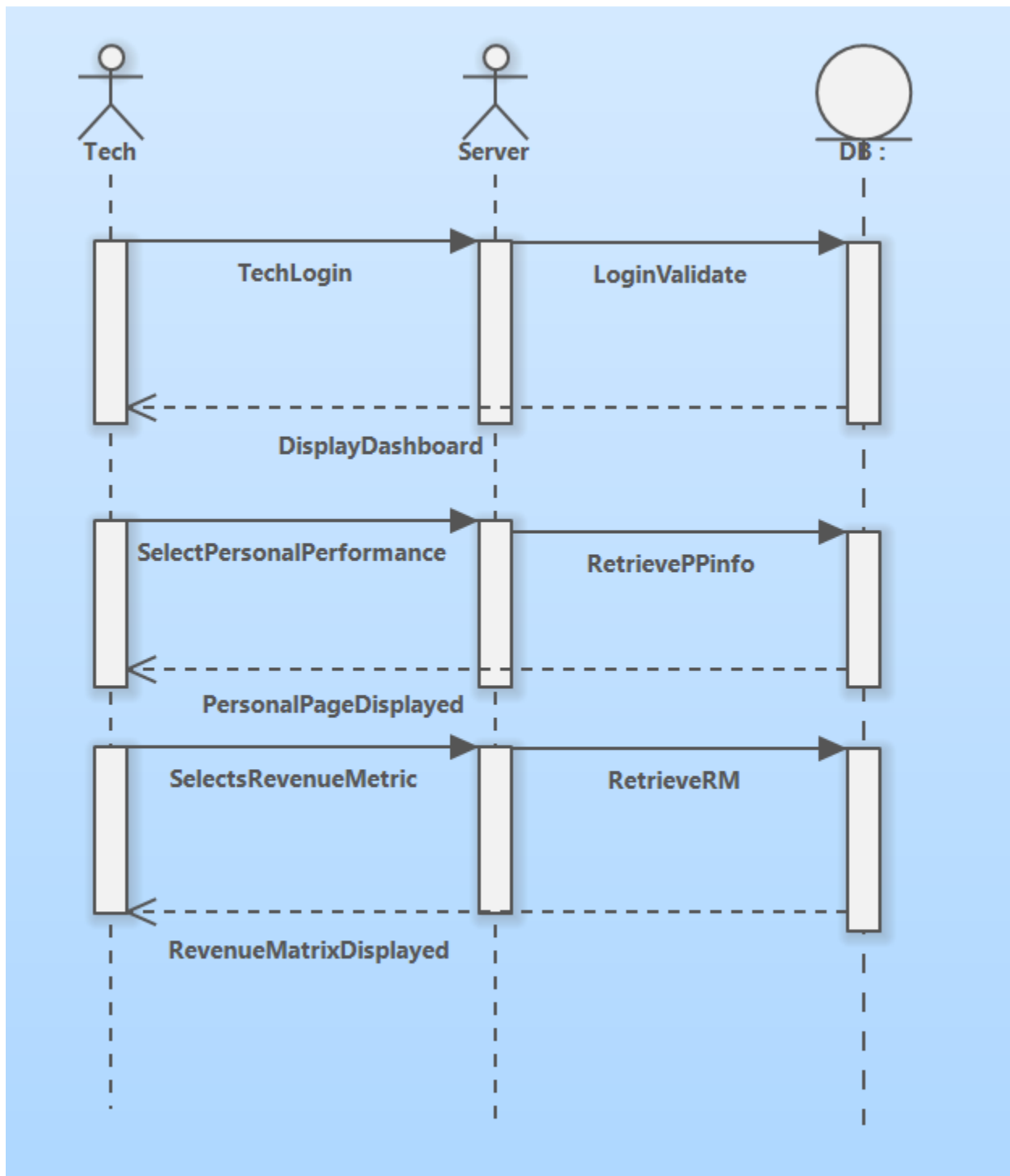


Figure 7. View Personal Revenue Sequence Diagram

4.2 State Machine Diagram

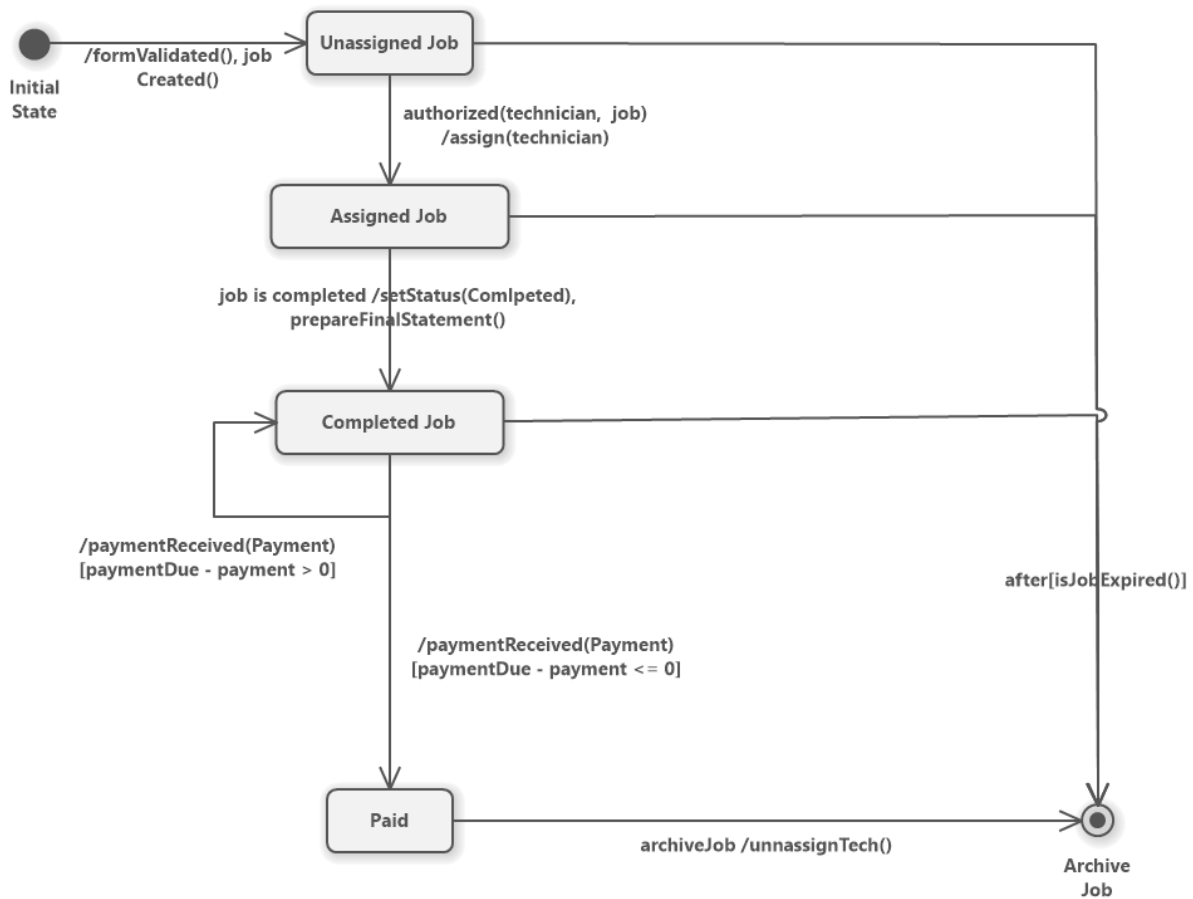


Figure 8. State Machine Diagram for assigned job and payment

This state machine diagram shows how the state of an unsigned job changes with different triggers.

4.3 Activity Diagram

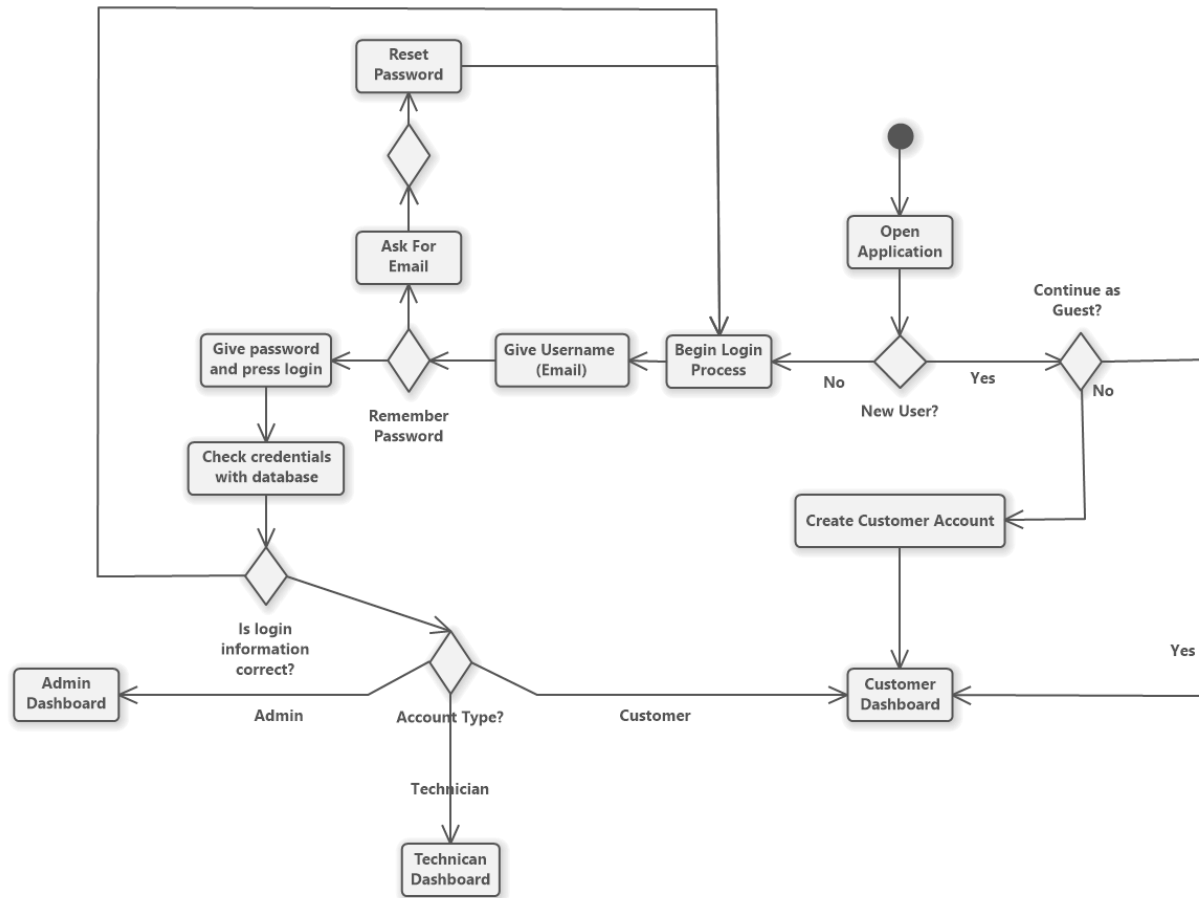


Figure 9. Login Activity Diagram

The activity diagram shows the process that takes place when a user login into the system. The option they choose either to create an account or continue as a guest to browse the application

5. System Design

5.1 Layered Architecture

The layered architecture pattern closely matches the traditional organizational structures found in most companies, making it a natural choice to compare application development efforts. Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., Presentation (front end) back-end/logic and Database layer)

Although the layered architecture pattern does not specify the classes that must exist in the pattern, the application consists of three main standard layers: Presentation, logic and persistence/database. Each level of the layered architecture pattern has a specific role and responsibility within the application. For example, the presentation layer is responsible for handling all user interface and browser communication logic, whereas the logic layer would be responsible for executing specific user requests while the database section will be responsible for storing and persisting all transactions within the application for references and request.

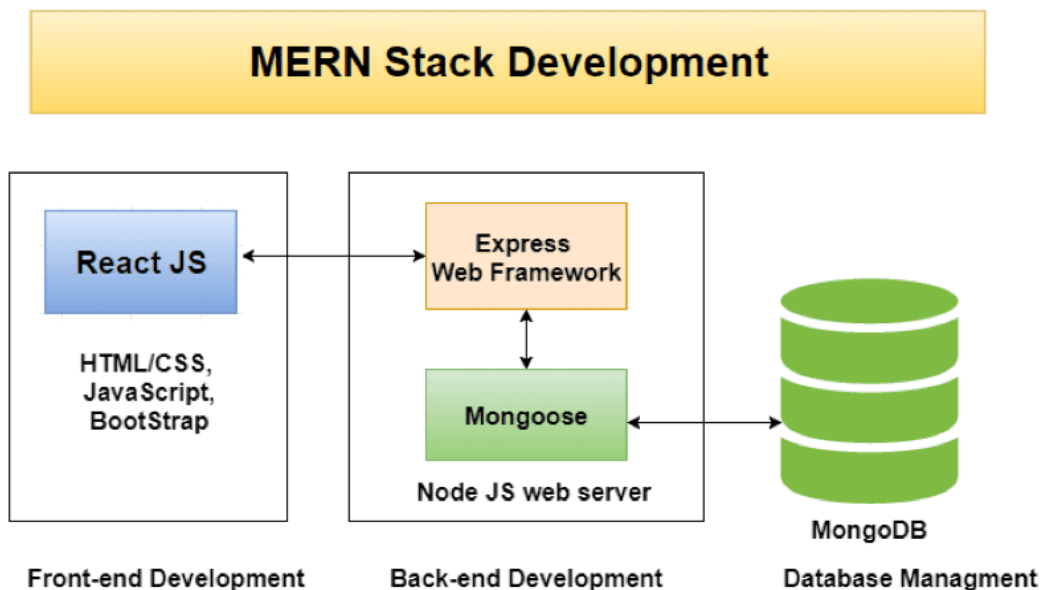


Figure 10. Layered Architecture Model

The application will be a closed layer, and this means that as a request moves from layer to layer, it must go through the layer right below it to get to the next layer after that one. For example, a request originating from the presentation layer must first go through the logic layer before finally hitting the database layer.

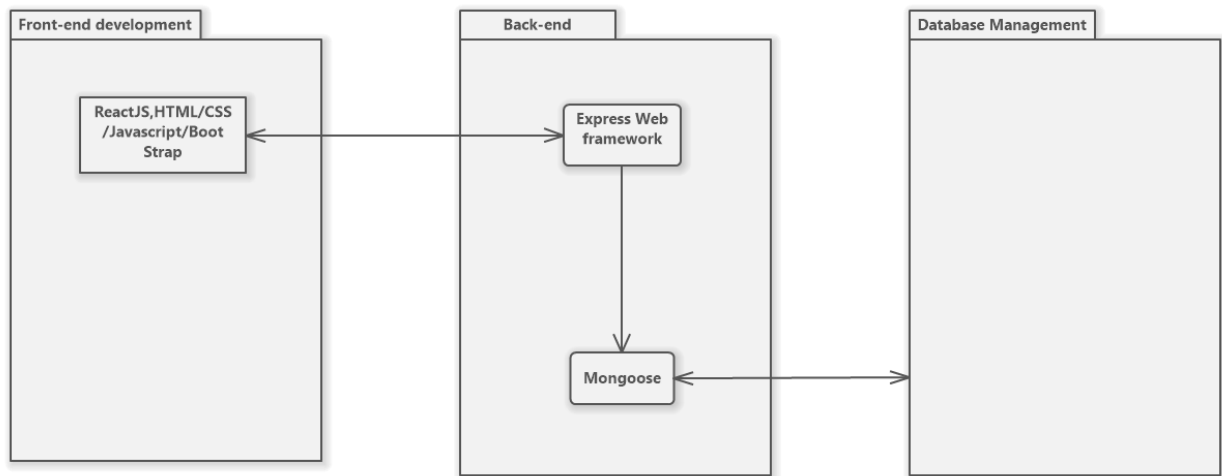


Figure 11. Packaged Layer Architecture

5.2 Hardware Architecture and Platform

The hardware set up we plan to use for our project are PCs with minimum 2.7 GHz Intel Core i5, 8GB Ram around 1 Gb space for project files. Operating system would be Windows 10, Code editor would be Visual Studio Code, Node.JS version ≥ 6.0 and a package manager such as npm. MongoDB would be installed in all the machines involved in the project.

The architecture of our application is based on a typical MVC model. Our client tier (View) will be written in JavaScript, HTML, CSS and Bootstrap using ReactJS as the framework. This level of the architecture is what the user will interact with to access the features of our application.

The Business Logic Tier (Controller) will be written using NodeJS and Express JS, and this tier represents the Application Server that will act as a bridge of communication for the Client Tier and Database Tier. This tier will serve HTML pages to the user's device and accept HTTP requests from the user and follow with the appropriate response.

Our Database Tier (Model) will be hosting MongoDB. This is where we will store all the crucial data, our application needs to function.

The following deployment diagrams explains the set-up for our MERN stack development

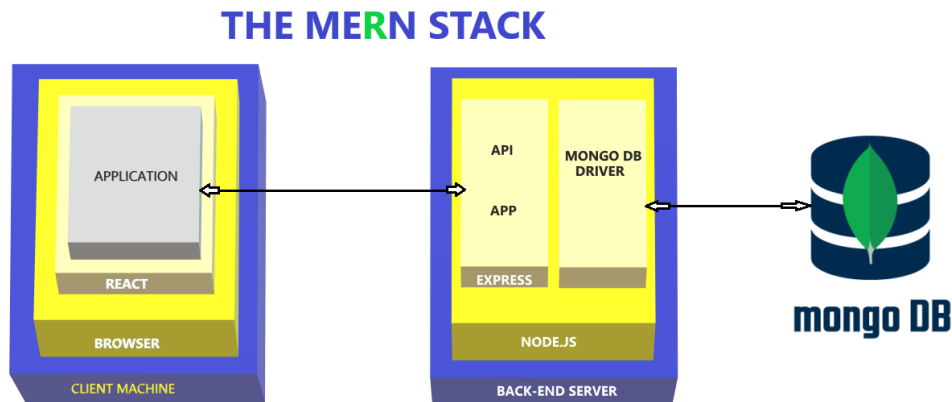


Figure 12. MERN stack development Model

5.3 Software Platform

The software stack that will be used for this project is a set of frameworks and tools used to develop a software product. This set of frameworks and tools are very specifically chosen to work together in creating a well-functioning application

MERN stack consists of MongoDB, Express JS, React, and Node.js as its working components. Here are the details of what each of these components is used for in developing a web application when using MERN stack:

- **MongoDB:** A document-oriented, No-SQL database used to store the application data.
- **NodeJS:** The JavaScript runtime environment. It is used to run JavaScript on a machine rather than in a browser.
- **Express JS:** A framework layered on top of NodeJS, used to build the backend of a site using NodeJS functions and structures. Since NodeJS was not developed to make websites but rather run JavaScript on a machine, Express JS was developed.
- **ReactJS:** A library created by Meta. It is used to build UI components that create the user interface of the single page web application.

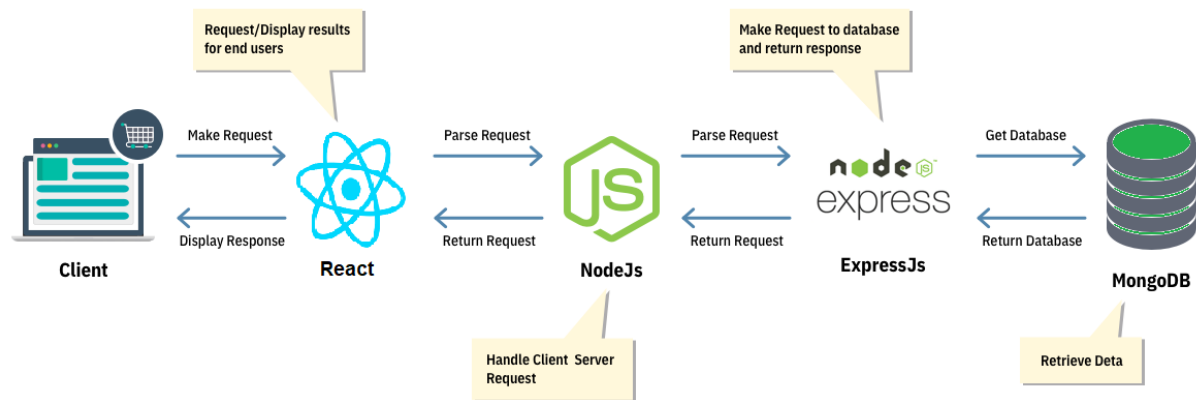


Figure 13. MERN software interaction

As shown in the diagram above, the user interacts with the **ReactJS** UI components at the application front-end residing in the browser. This frontend is served by the application backend residing in a server, through **Express JS** running on top of NodeJS.

Any interaction that causes a data change request is sent to the **NodeJS** based Express server, which grabs data from the **MongoDB** database if required, and returns the data to the frontend of the application, which is then presented to the user.

6. Interaction Model

Style: direct manipulation (GUI)

Desired user support:

- **Error messages:**
 1. Log in page:
 - a. Situation: email and password do not match
Error message: Credentials do not match, please double check them.
 - b. Situation: email is not entered/format of email is incorrect
Error message: Invalid email address, please re-enter it.
 - c. Situation: password is not entered
Error message: Invalid password, please re-enter it.
 2. Sign up page:
 - a. Situation: first name/last name is missing or does not match pattern (e.g. has special symbols, numbers)
Error message: Invalid name, only alphabetic characters are allowed.
 - b. Situation: phone number is not entered or does not match pattern as the placeholder.
Error message: Invalid phone number, please re-enter it.
 - c. Situation: address is not entered
Error message: Invalid address, please re-enter it.
 - d. Situation: postal code is not entered or does not match pattern as the placeholder.
Error message: Invalid postal code, please re-enter it.
 - e. Situation: password and re-entered password are not same

- Error message: Please double check the second password and re-enter it.
 - f. Situation: email is not entered/format of email is incorrect
Error message: Invalid email address, please re-enter it.
 - g. Situation: password is not entered
Error message: Invalid password, please re-enter it.
- 3. Reset Password page:
 - a. Situation: new password is not entered
Error message: Invalid password, please re-enter it.
 - b. Situation: password and re-entered password are not same
Error message: Please double check the second password and re-enter it.
 - c. Situation: email is not entered/format of email is incorrect
Error message: Invalid email address, please re-enter it.
- 4. Contact Us page:
 - a. Situation: email is not entered/format of email is incorrect
Error message: Invalid email address, please re-enter it.
 - b. Situation: service request # is not entered/does not match pattern (number only with specific length)
Error message: Invalid service request number, please re-enter it.
 - c. Situation: inquiry text area is blank
Error message: Please enter your inquiry description.
- 5. Book Service page
 - a. Situation: address is not entered
Error message: invalid address, please re-enter it.
 - b. Situation: postal code is not entered or does not match pattern as the placeholder.
Error message: Invalid postal code, please re-enter it.
 - c. Situation: date is not selected
Error message: Please select a date.
 - d. Situation: time slot is not selected
Error message: Please select a time.
 - e. Situation: description is blank
Error message: Please write down your problem.
- 6. Manage Profile page
 - a. Situation: first name/last name is missing or does not match pattern (e.g. has special symbols, numbers)
Error message: Invalid name, only alphabetic characters are allowed.
 - b. Situation: phone number is not entered or does not match pattern as the placeholder.
Error message: Invalid phone number, please re-enter it.
 - c. Situation: address is not entered
Error message: Invalid address, please re-enter it.
 - d. Situation: postal code is not entered or does not match pattern as the placeholder.
Error message: Invalid postal code, please re-enter it.
 - e. Situation: email is not entered/format of email is incorrect
Error message: Invalid email address, please re-enter it.
 - f. Situation: password is not entered
Error message: Invalid password, please re-enter it.

- **Required help:**

1. Menu has CONTACT US tab, including information as operation hours and phone number of customer service center. Clients are able to call customer service center for inquiries.
2. There is an inquiry form within CONTACT US page. User can submit an inquiry form for any inquiry.

System feedback:

- After a user sign up a new account, there will be a message says “The account has been created successfully. Thanks for your sign up.”
- After a user book a service, there will be a message says “Success! Your Request is delivered. Service Request #XXXXX. Please check your email for confirmation.”
- After a user submit an inquiry, there will be a message says “Success! We have received your inquiry. Ticket #XXXXX. Please check your email for confirmation.”

Standards:

- Colors: orange (primary), white (secondary)
- Fonts: Malgun Gothic in similight (primary), Arial (secondary)

Interactions:

- Data entry

Home Pro
Delivers trusted repair services

Home About Us Contact Us Log In

WELCOME

Email

Password


☐ Remember Me? [Forgot Password](#)

Log In

Not Registered? [Create An account](#)

Figure 14. Login page

- List of items (calendar and time slot)



Home Pro

Delivers trusted repair services

[< Plumbing](#)
[Log Out](#)
[About Us](#)

Book a Plumber

Enter the address where service is needed

Select date and time for service

April 2022 V

SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

07:30 AM


07	30	AM
08	31	PM
09	32	
10	33	
11	34	
12	35	
01	36	

Brief description of the problem

Send Service Request

Figure 15. Booking Page


- Error warning (missing postal code after click save changes button)



Home Pro

Delivers trusted repair services

Manage Profile



Invalid postal code, please re-enter it





Figure 16. Manage Profile

- Summary report



Home Pro

Delivers trusted repair services

Ongoing Services

Service #	Service Name	Service Request Date	Service Request Time	Technician Name
1772336	Plumbing	12 April, 2022	1400Hr	Michael
1228334	Handyman Services	15 April, 2022	1000Hr	Not assigned yet
1344339	Painting	9 April,2022	0900Hr	Trevor

Back to main page

Any questions or concerns about our services? Please fill out our [contact us](#) form.

Figure 17. Ongoing Services

Screenshots and list of pages

- Landing page

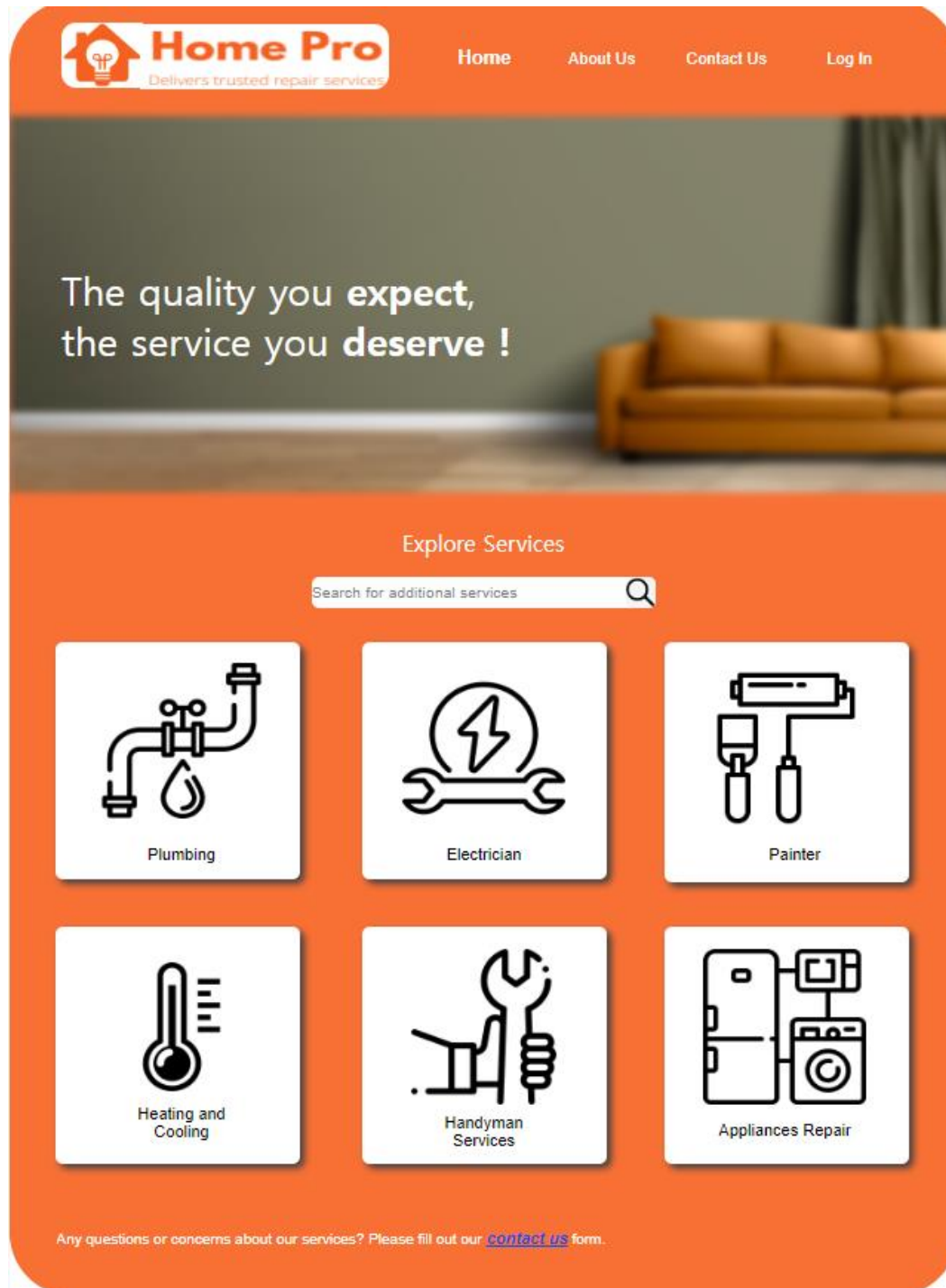
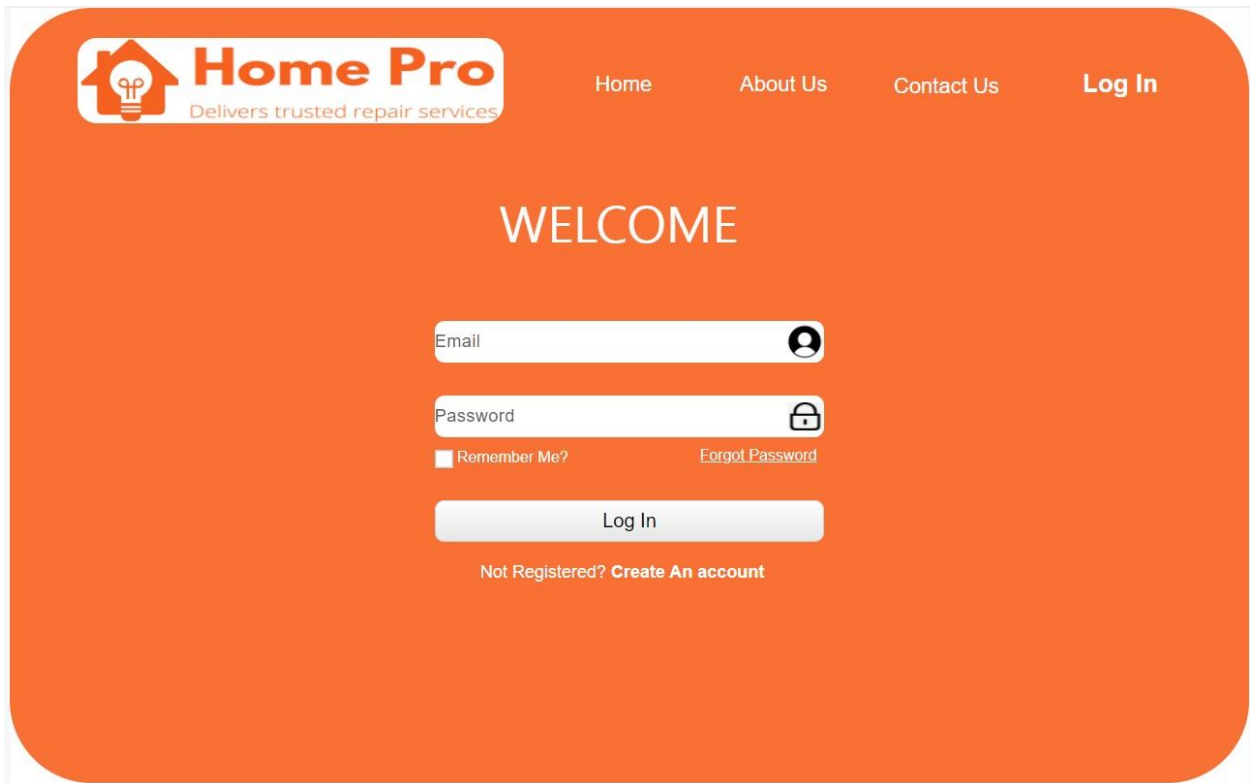


Figure 18. Landing Page

- Login Page



The image shows a login page for 'Home Pro'. The page has an orange background. At the top left is the 'Home Pro' logo, which includes a house icon with a lightbulb inside and the text 'Home Pro' in bold, with 'Delivers trusted repair services' underneath. To the right of the logo are navigation links: 'Home', 'About Us', 'Contact Us', and 'Log In'. In the center of the page, the word 'WELCOME' is displayed in large, white, uppercase letters. Below this, there are two input fields: 'Email' and 'Password'. The 'Email' field has a user icon on the right, and the 'Password' field has a lock icon on the right. Below the 'Password' field, there is a checkbox labeled 'Remember Me?' and a link labeled 'Forgot Password'. Below these fields is a 'Log In' button. At the bottom, there is a link that says 'Not Registered? Create An account'.

Home Pro
Delivers trusted repair services

Home About Us Contact Us Log In

WELCOME

Email

Password


☐ Remember Me? [Forgot Password](#)

Log In

Not Registered? [Create An account](#)


Figure 19. Login Page


- Reset password page




Home Pro
Delivers trusted repair services

Reset Password

Email 

New Password 


Re-enter your password 

Submit

Back to Login

Figure 20. Reset Password Page

- Create a new account page




Home Pro
Delivers trusted repair services


Create an Account


First Name Last Name

Phone Number 999-999-9999

Address Postal Code A1A A1A

Email 

Password 

Re-enter your password 

Submit

Back to Login

Figure 21. New Account Page

- About us page

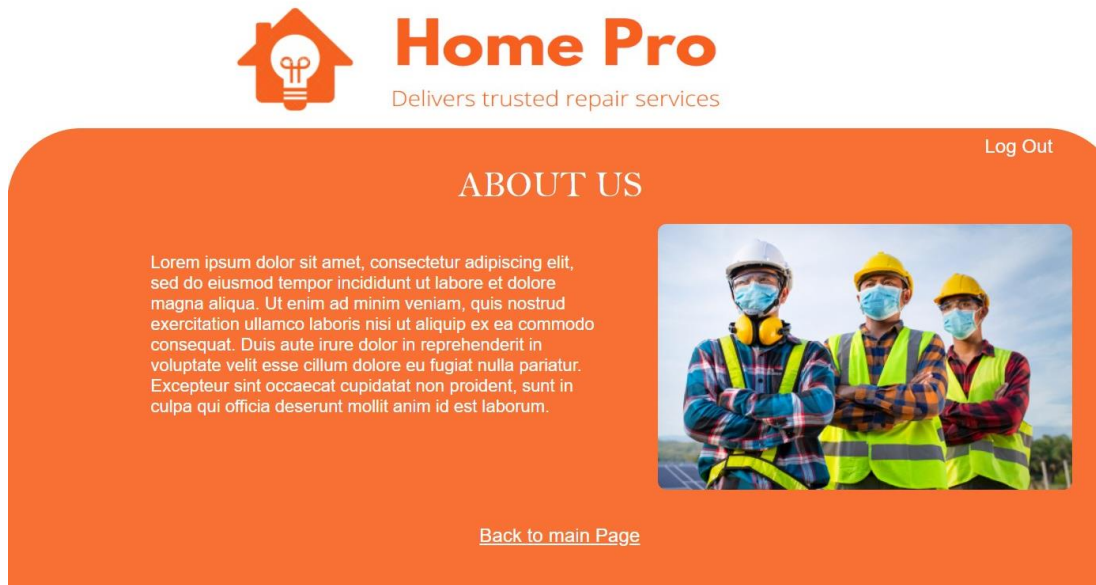



Figure 22. About Us Page

- Contact us page



Home Pro

Delivers trusted repair services

Log Out

Contact Us Form

Please provide us with your valuable feedback about previous or ongoing service.

* Required Fields

* Email


* Service Request #

Technician Name

Service Date

* Inquiry

Submit




Toll-free customer service: 1-800-123-4567

**Hours: Mon-Sat 8 AM - 8 PM MDT
Sun 11 AM - 7 PM MDT**

[Back to main Page](#)

Figure 23. Contact Us Page

- Service introduction page





Home Pro

Delivers trusted repair services

[< Select Service](#)
[Log Out](#)
[About Us](#)

Plumbing





PRICE CHART

\$150 for first hour
And \$80 per hour
Plus material Cost


Lorem ipsum dolor sit amet. Non impedit sint vel placeat tempora in ratione sed quasi porro quo dolorum rerum aut placeat illum qui mollitia minus. Qui cupiditate nobis ad incidunt aliquam sed maiores dolore sed magnam quia non labore nostrum qui enim rerum aut molestiae quia. Et nobis recusandae id repudiandae libero aut molestiae dolores eos totam tempore non neque vitae aut laborum eius. At doloremque incidunt qui sunt consequatur quo iste voluptatem vel doloremque obcaecati. Et rerum voluptas sit inventore illo ex perspiciatis neque et iste fuga. Ut saepe minus ea similique voluptas aut natus minus cum nulla temporibus ad quos tempore. Et aliquam quod 33 dolorem inventore aut laboriosam nobis cum cumque nisi aut quibusdam veritatis aut culpa iste. Vel explicabo quaerat qui neque iste ad vero quia et animi dicta! Rem voluptate voluptatem est nihil maxime qui consequatur dolorum aut earum blanditiis?

Book Now

Any questions or concerns about our services? Please fill out our [contact us](#) form.

Figure 23. Service Page

- Book service page



Home Pro

Delivers trusted repair services

< Plumbing

Log Out
About Us

Book a Plumber

Enter the address where service is needed

Select date and time for service

April 2022 V

SUN	MON	TUE	WED	THU	FRI	SAT
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

07:30 AM

0730AM

0831PM

0932

1033

1134

1235

0136

Brief description of the problem

Send Service Request

Figure 25. Booking Service Page

- Confirmation of booking page

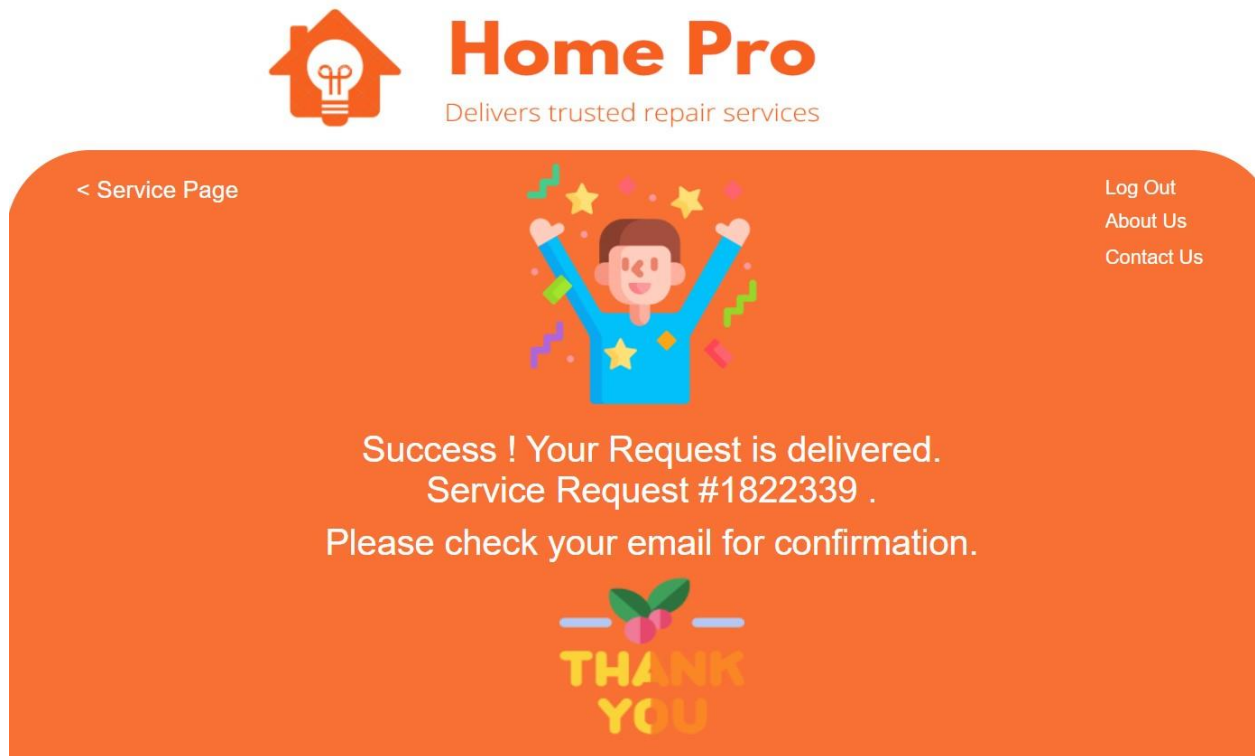


Figure 26. Booking Confirmation Page

- Confirmation of sending inquiry page

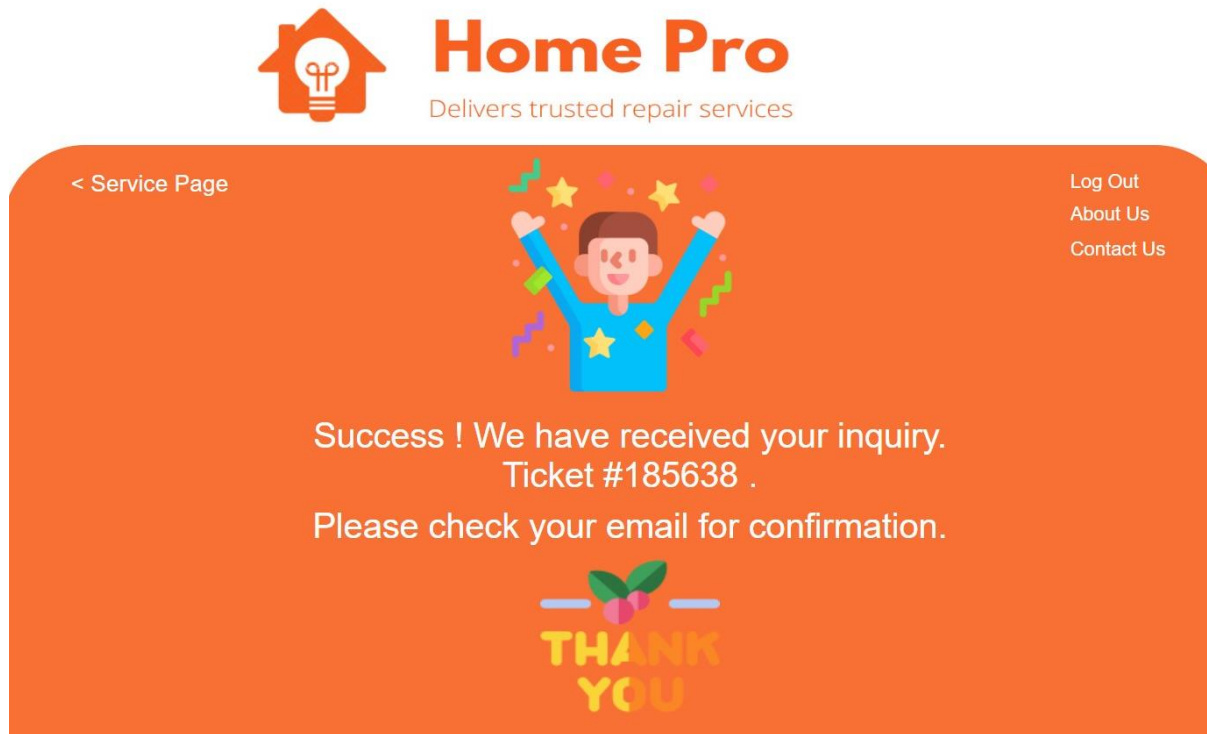


Figure 27. Confirmation of inquiry Page

- Technician's Home page

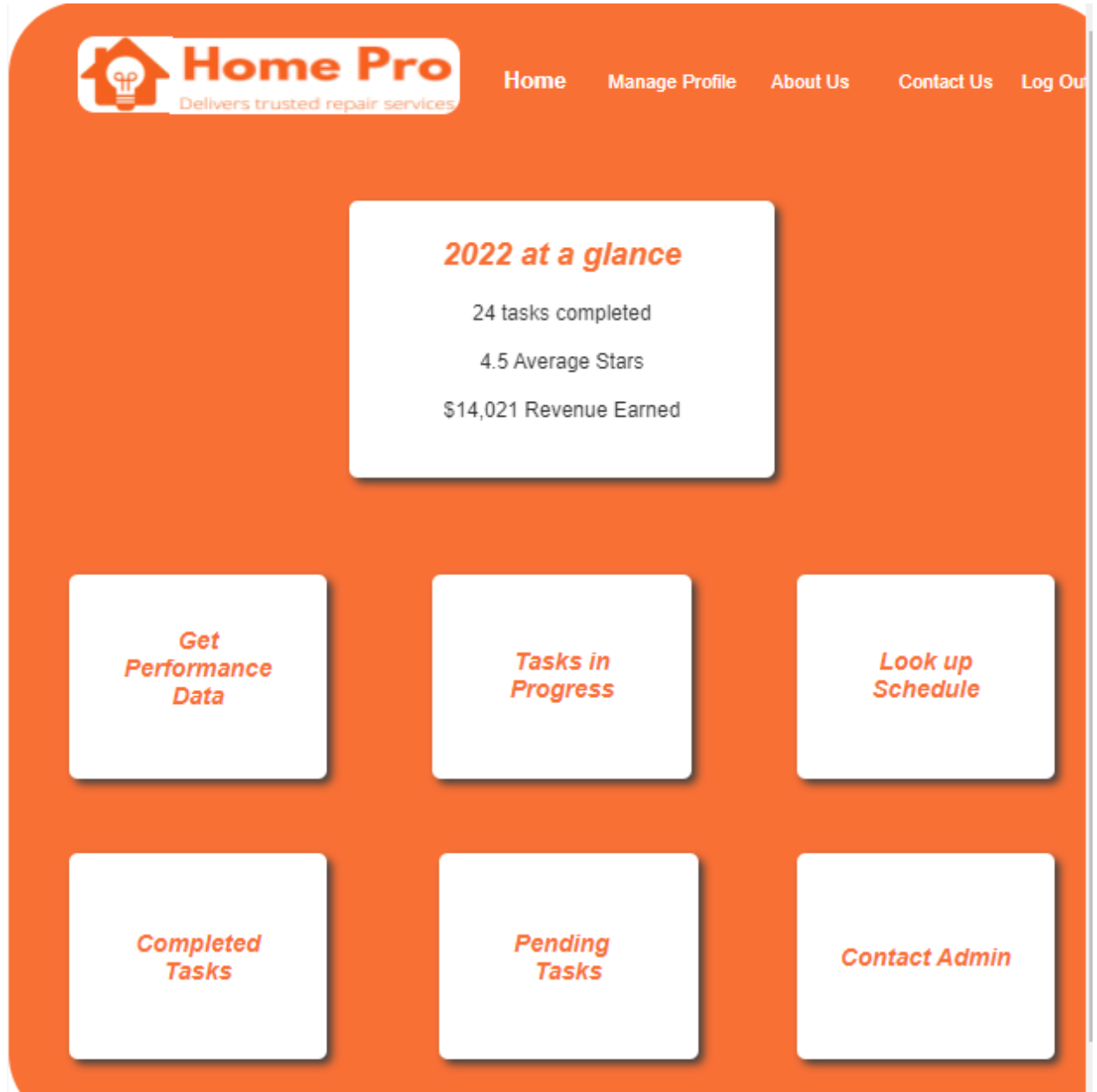


Figure 28. Technician Home Page

- Performance Data Page

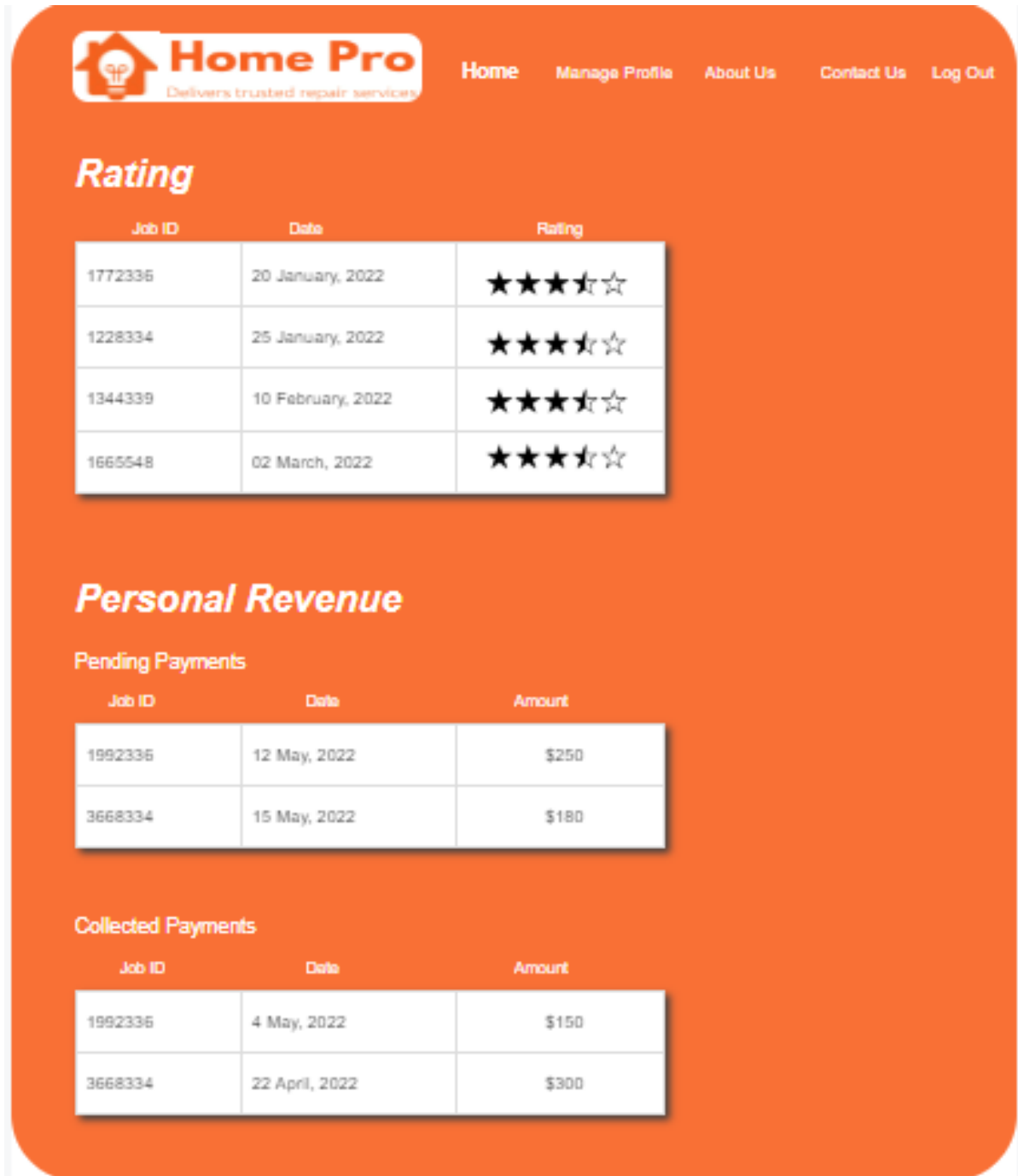


Figure 29. Performace Details Page

- Technician Schedule Page

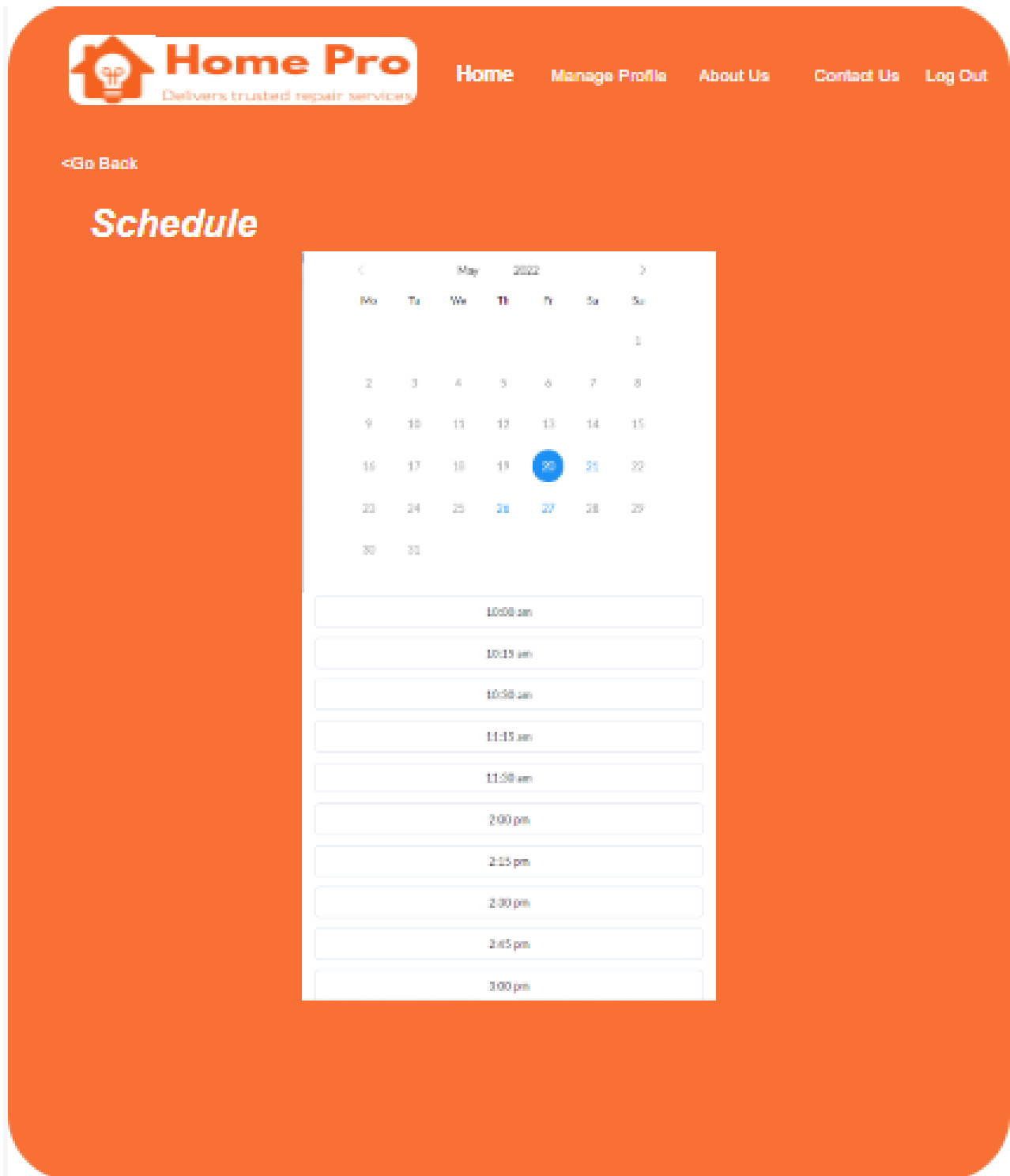


Figure 30. Schedule Page

- Job Details Page

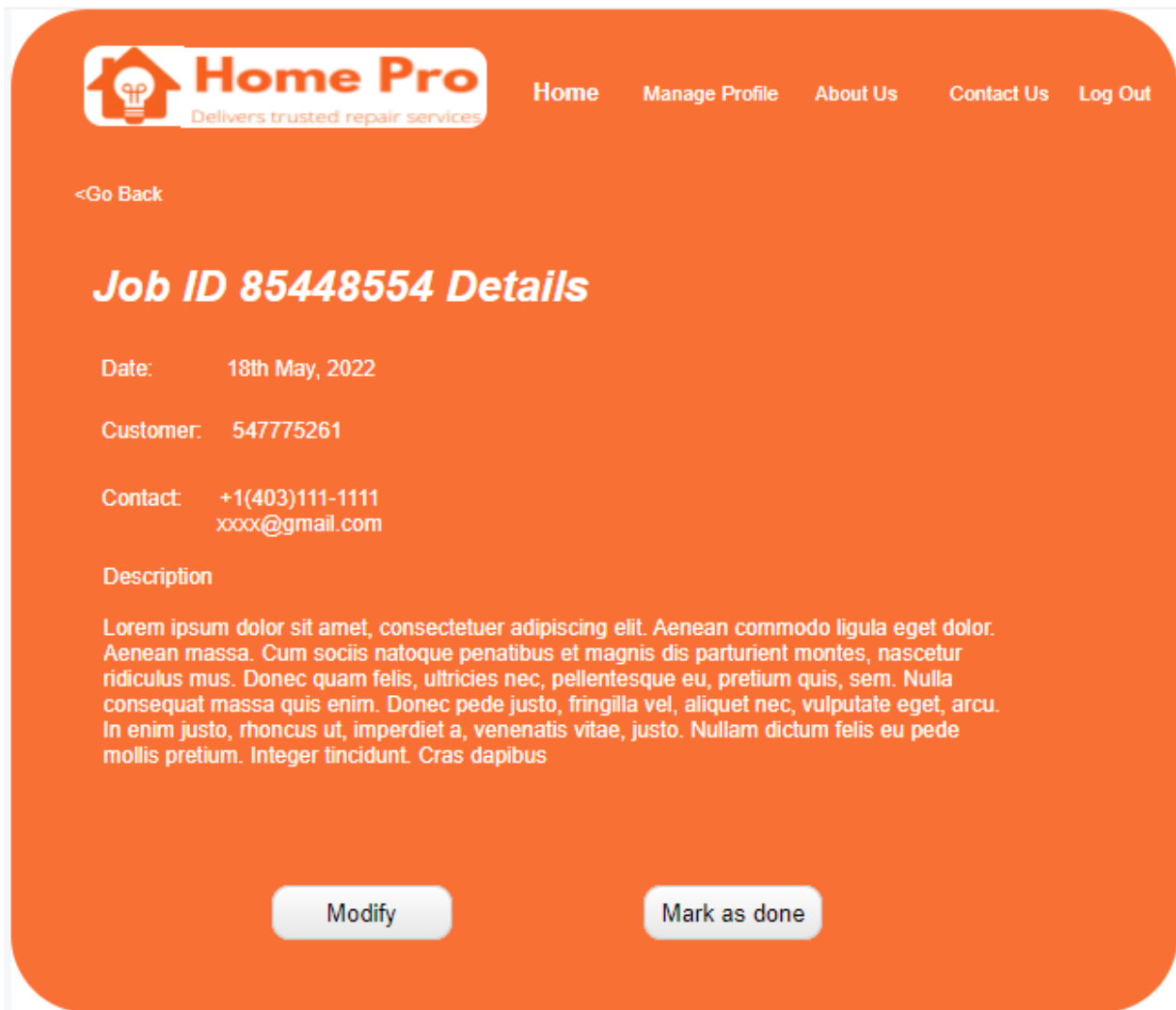



Figure 31. Job details Page

- Payment Information Page

**Home Pro**
Delivers trusted repair services

[Home](#) [Manage Profile](#) [About Us](#) [Contact Us](#) [Log Out](#)

[<Go Back](#)

Job ID 85448554 Payment Details

Working Hours:

Materials (Optional)

Descripton	Quantity	Unit Price	Total Amount

Submit

Figure 32. Payment Information Page

- Admin's Home Page



Figure 33. Admins's home page

- Manage customers Page

Home Pro
Delivers trusted repair services

Manage Profile Log Out

< Back

Manage Customers

Customers List


<input type="checkbox"/>	Customer ID	First Name	Last Name	E-mail
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com

Delete Add a new customer

< Previous 1 2 3 4 5 6 7 8 9 10 Next >

Figure 34. Manage customers page

- Manage Technician Page

**Home Pro**
Delivers trusted repair services

Manage Profile Log Out

< Back

Manage Technicians

Technician List

<input type="checkbox"/>	Technician ID	First Name	Last Name	E-mail	
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com	<button>Edit</button>
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com	<button>Edit</button>
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com	<button>Edit</button>
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com	<button>Edit</button>
<input type="checkbox"/>	123456	Joe	Blow	joe.blow@example.com	<button>Edit</button>

DeleteAdd a new technician

< Previous 1 2 3 4 5 6 7 8 9 10 Next >

Figure 35. Manage Technician Page

- Financial Data Page

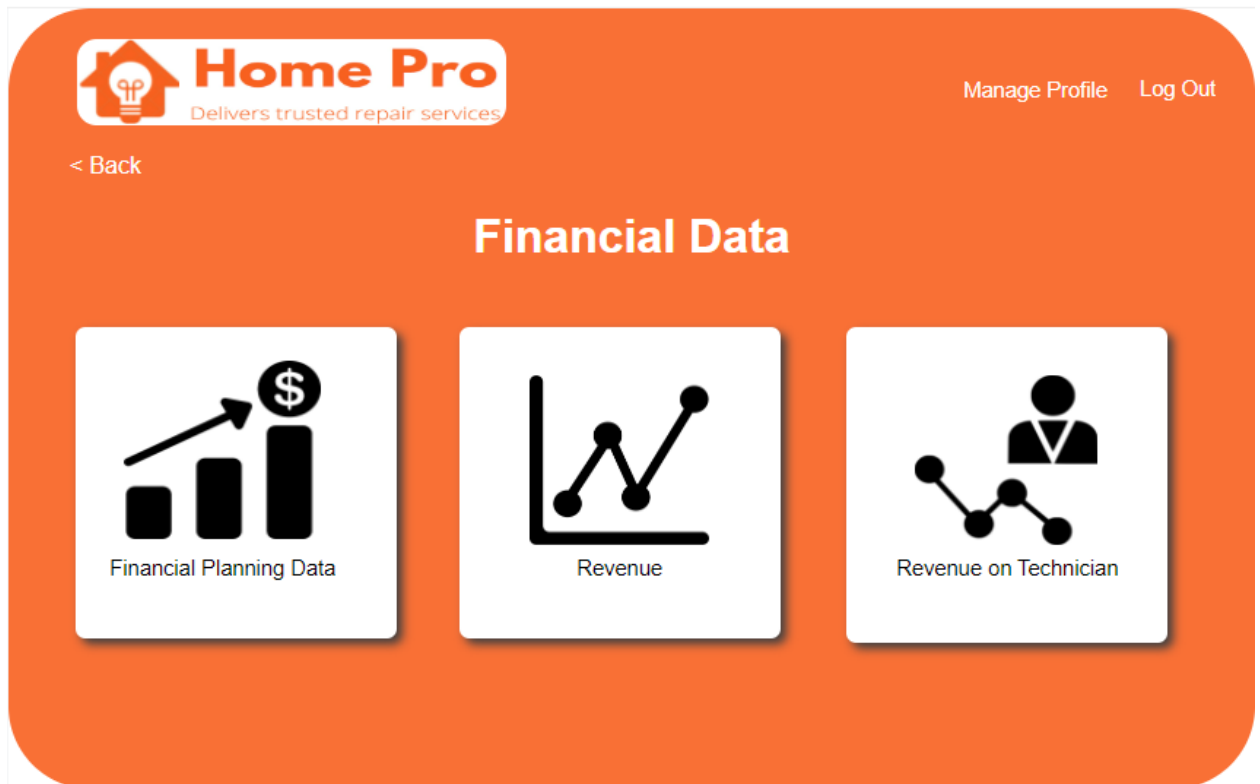



Figure 36. Financial Data page

- Technician Payment Manager Page

**Home Pro**
Delivers trusted repair services

[Manage Profile](#) [Log Out](#)

< Back

Technician Payment Manager


Technician ID Start End

Technician ID	First Name	Last Name	Amount Due	
123456	Joe	Blow	\$1000	Details
123456	Joe	Blow	\$1000	Details
123456	Joe	Blow	\$1000	Details
123456	Joe	Blow	\$1000	Details
123456	Joe	Blow	\$1000	Details

< Previous 1 2 3 4 5 6 7 8 9 10 Next >

Figure 37. Technician Payment Manager

- Job Manager Page

**Home Pro**
Delivers trusted repair services

[Manage Profile](#) [Log Out](#)

[< Back](#)

Job Manager

Job ID

Start

End

Search

Job ID	Date	Technician ID	
123456	June 1, 2022	123456	Details
123456	May 22, 2022	N/A	Details
123456	Apr 2, 2022	123456	Details
123456	Mar 30, 2022	123456	Details
123456	Feb 1, 2022	123456	Details

[< Previous](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [Next >](#)

Figure 38. Job Manager Page

7. Persistence Model

For this system we will be using MongoDB for persistence. MongoDB is a document-based database system (as opposed to a traditional relational database) that does not make use of database tables or SQL. MongoDB stores documents in a structure called a collection, which is analogous to a table in a relational database. One or more collections are stored in a database. MongoDB documents consist of field and value pairs, and values may contain other documents, arrays, or arrays of documents. MongoDB documents are written in BSON, which is a binary representation of JSON. MongoDB records are therefore nearly identical to JSON objects in structure, while also natively allowing a greater number of data types than SQL-based databases, such as Booleans, objects, regular expressions, and JavaScript code. Some of the advantages of using documents and collections of documents over relational tables are:

- Documents already resemble objects and are therefore much simpler to serialize and deserialize. In the case of our system, since we will be using JavaScript throughout, the data contained in the documents are already stored in JSON format and will not require the same serialization and deserialization operations that would be needed when using a relational database.
- Documents can be embedded inside of other documents. This allows for similar functioning to relational databases where tables are related through a foreign key and where data must be joined using joining operations. However, since the related data is simply stored inside the same document, no join operations must be performed, and the system does not need to query multiple tables. This simplifies data lookup and improves performance.

In most cases, denormalized document schemas are the optimal solution when using MongoDB for persistence. This means that multi-document transactions (equivalent to join operations) should be avoided where possible using embedded documents.

For performance reasons, MongoDB only supports BSON documents of up to 16 MB in size. Documents larger than this must be stored using an API.

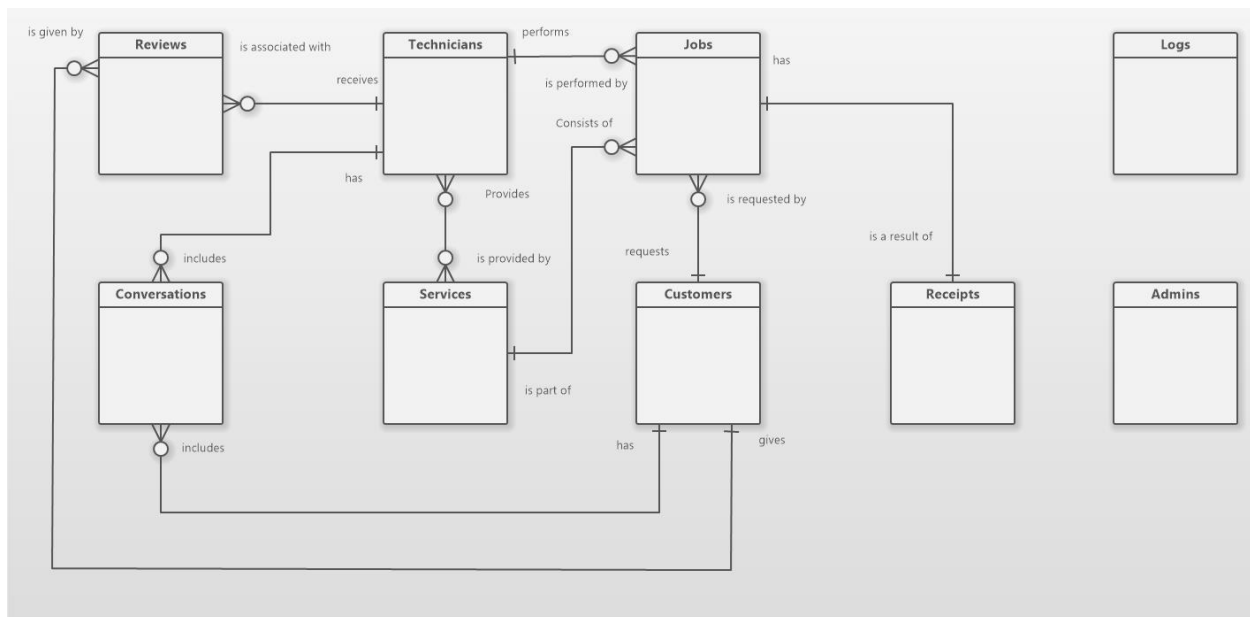


Figure 39. Conceptual ERD

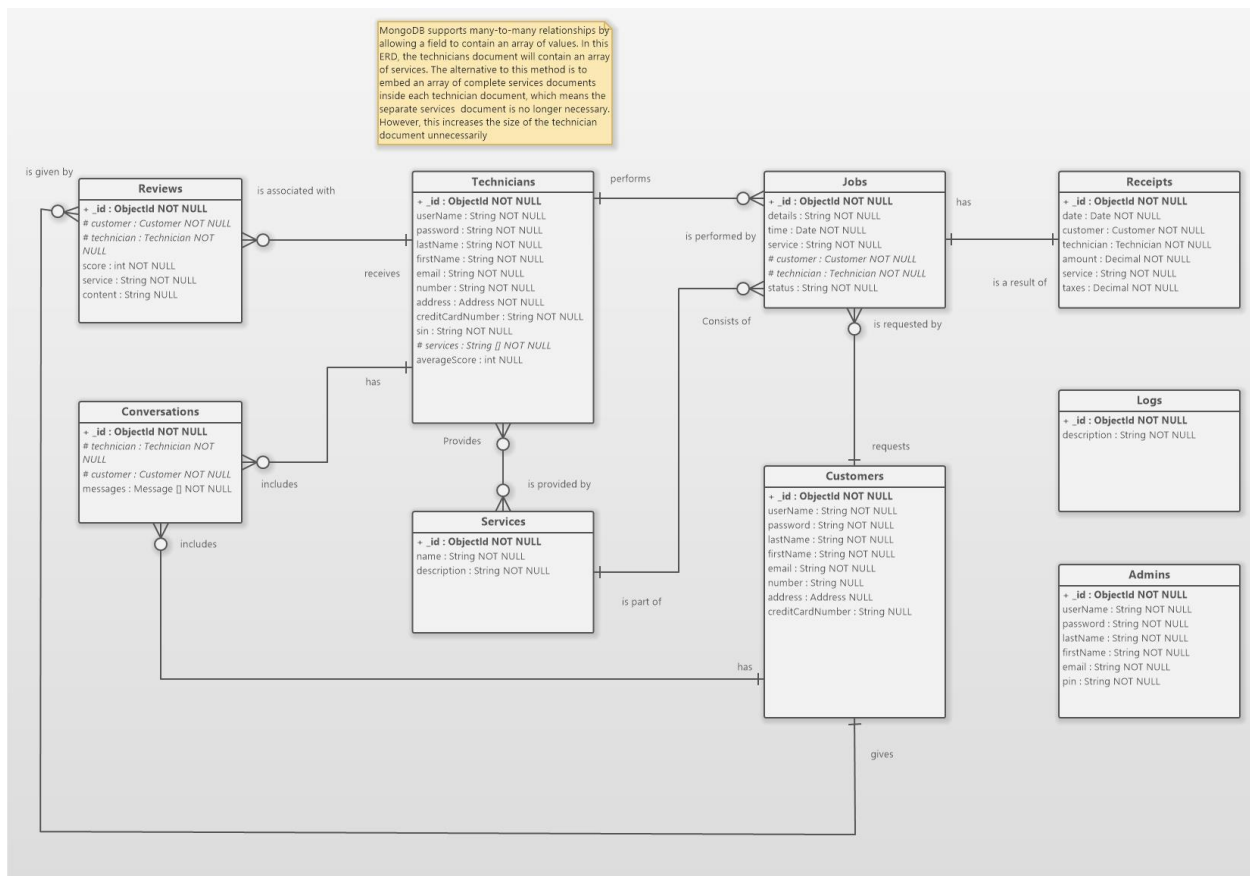


Figure 40.. Physical ERD

7. Class Diagram

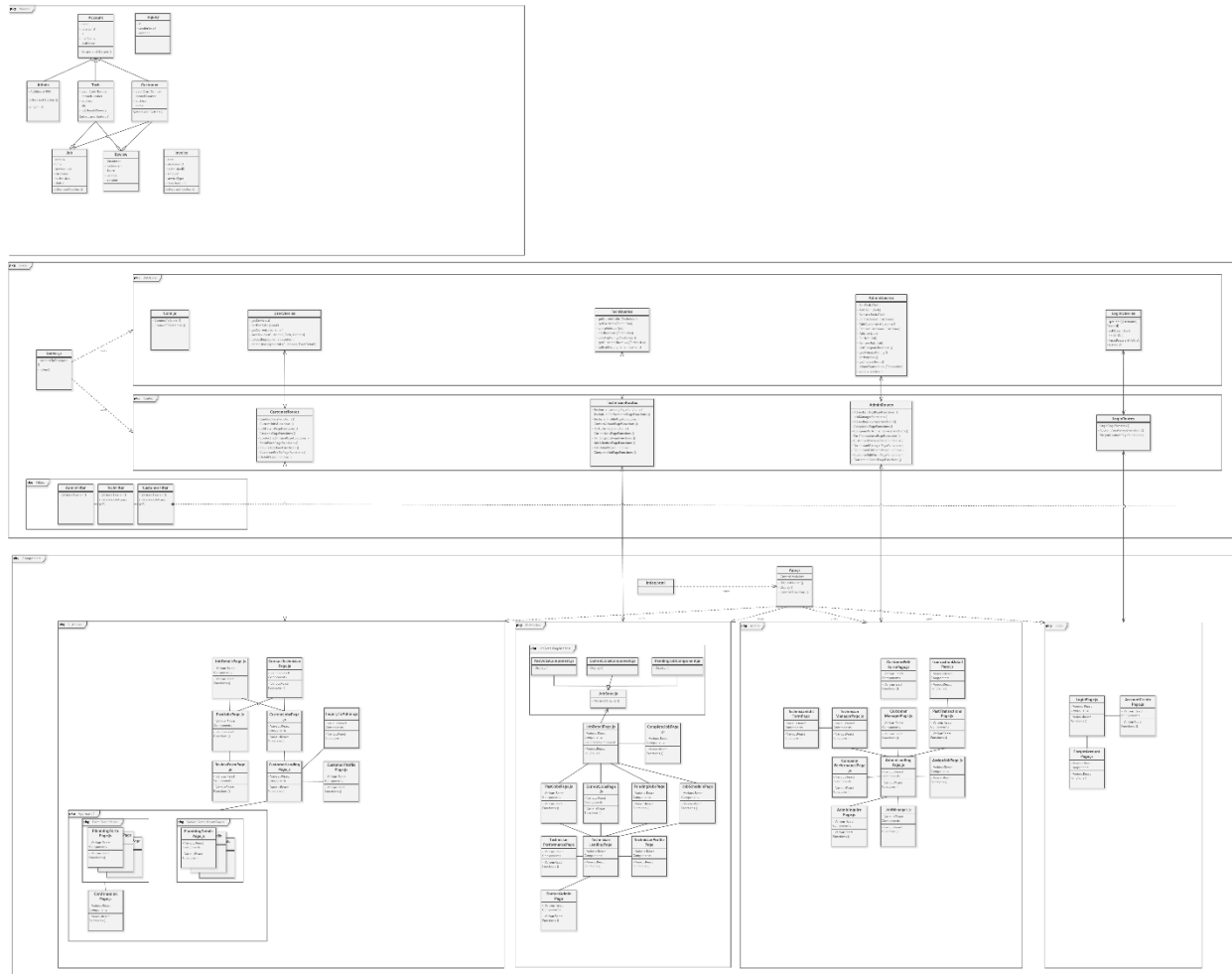


Figure 41.. Complete Class Diagram

Diagram that shows the complete application diagram, highlighting how each of the different parts will play off of each other in addition to where they are located in the directory structure.

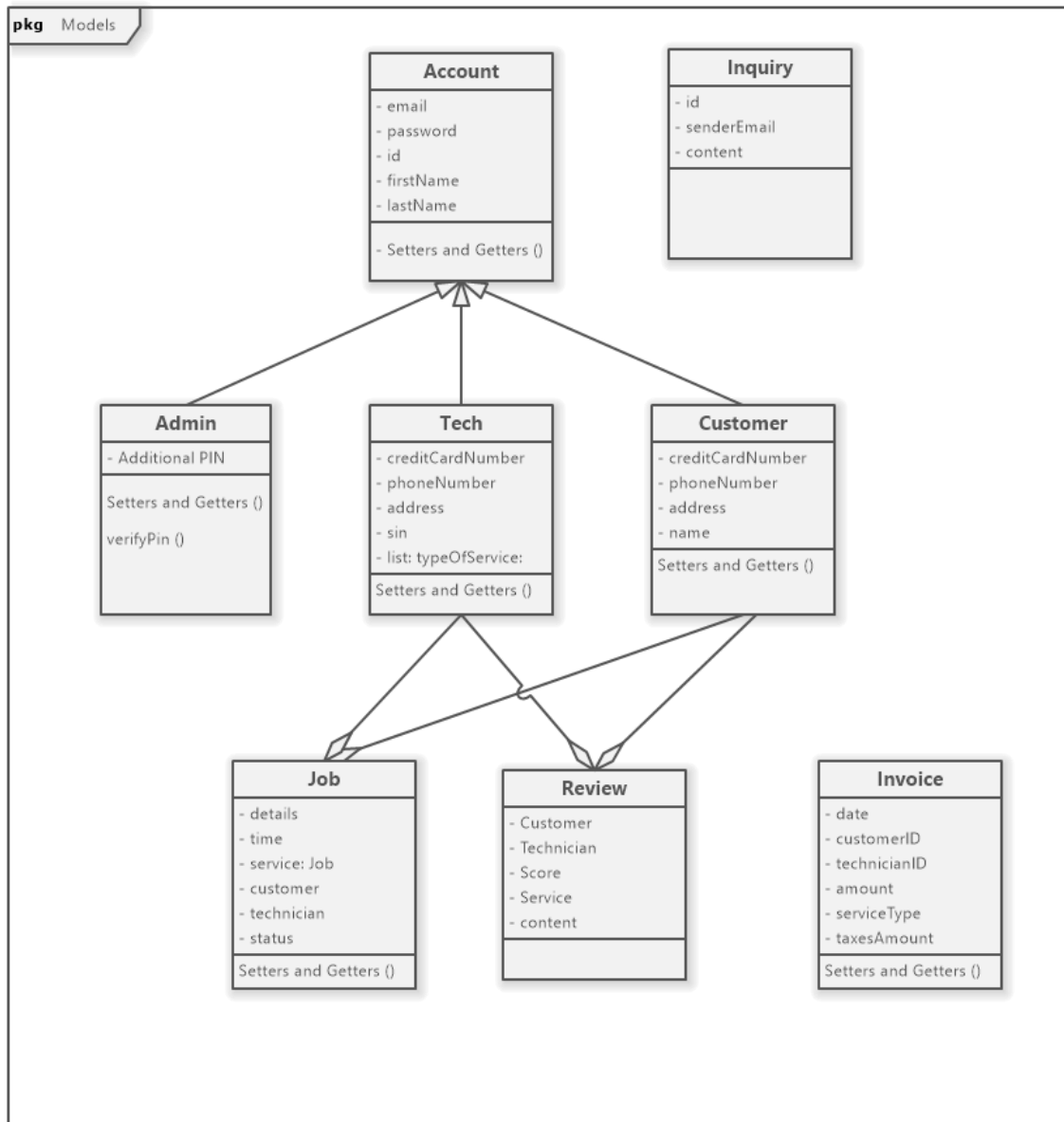


Figure 42.. Models Class Diagram

Diagram that shows the models that will be used by both the server and client side to store and interact with the data. These can also be referred to as domain classes.

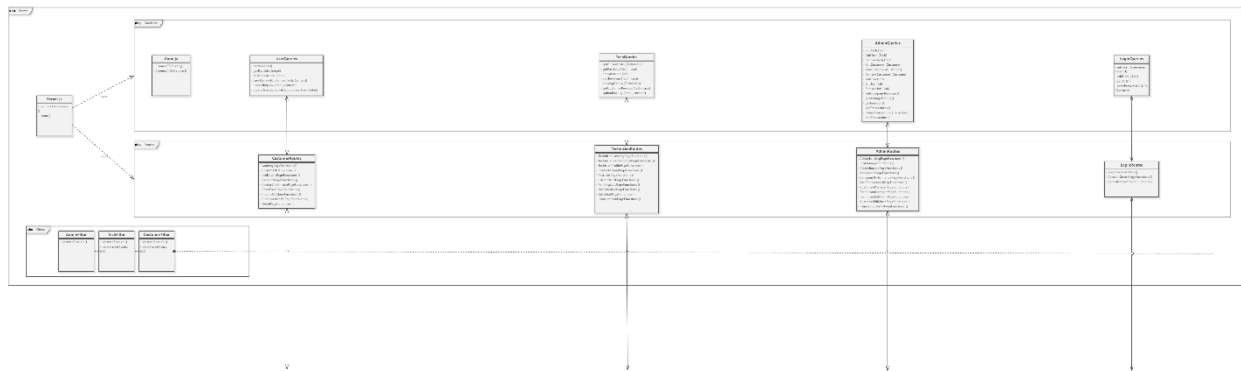


Figure 43. Backend Diagram

Diagram that shows both the Database and Server-side classes of the application. Both the route classes and database classes shown here can be merged or further broken down to suit development needs. Regardless of method, each end point in the application will have a corresponding function in the route classes, which may or may not need to connect to the database. In addition, we will have filter classes that will restrict the endpoints the user can access depending on session privileges. The Server class is used to host the server, and the conn class contains database connection information and functions used by the server.

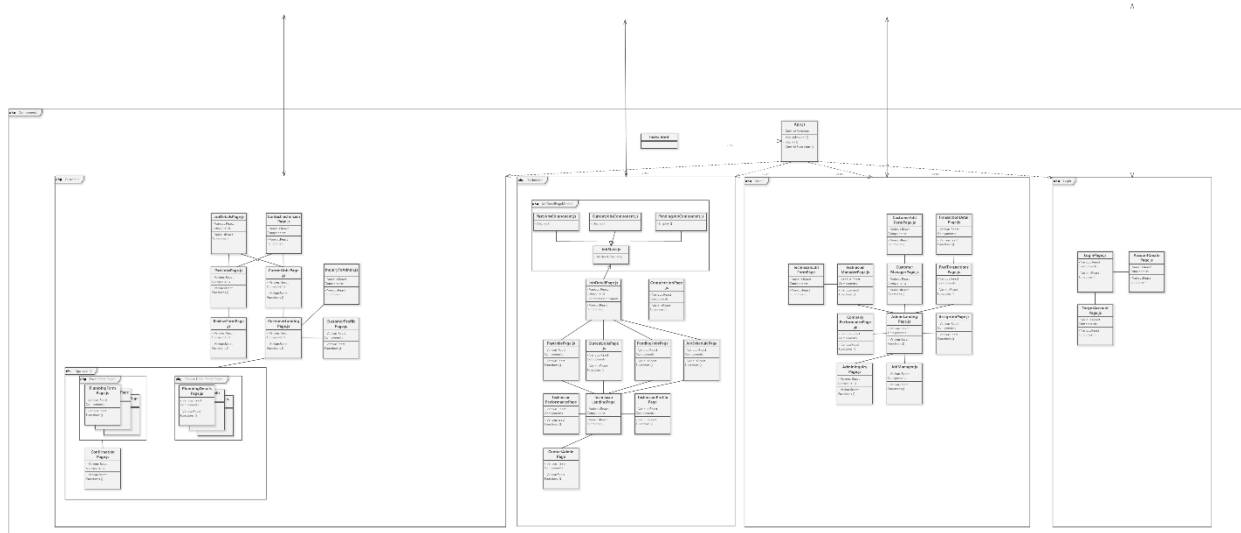


Figure 44.. Front End Diagram

Complete view of the front-end classes, which highlights the four different sets of classes that the user will interact with. Index.html is the single page in our single page application, and app.js controls which React Components will be displayed on the index page at a given time. The page classes are React Components that are swapped into index.js and will have more react and JavaScript components to complete their functionality. The best way to describe it is that the index+app.js combo acts as a stereogram, while the pages are the different slides what can be viewed by the stereogram.

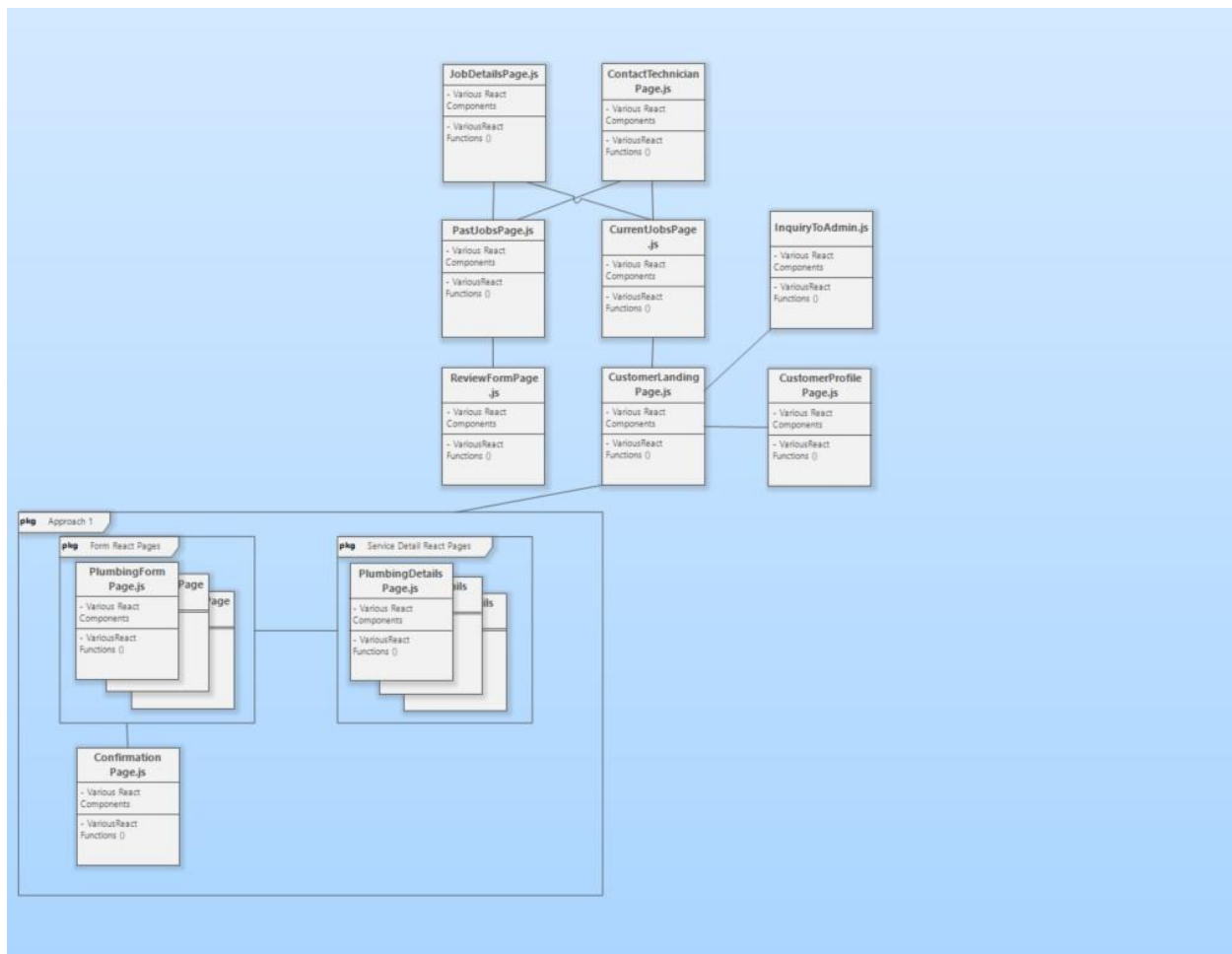


Figure 45. Customer Front-End Classes

These are the JavaScript classes that will render the front end for the customer. These are the base classes we feel that we will need to encapsulate the needs of the user. We are considering the use of a design pattern for the service pages, but until further research on JavaScript objects are done, we feel like this is a great plan of attack.

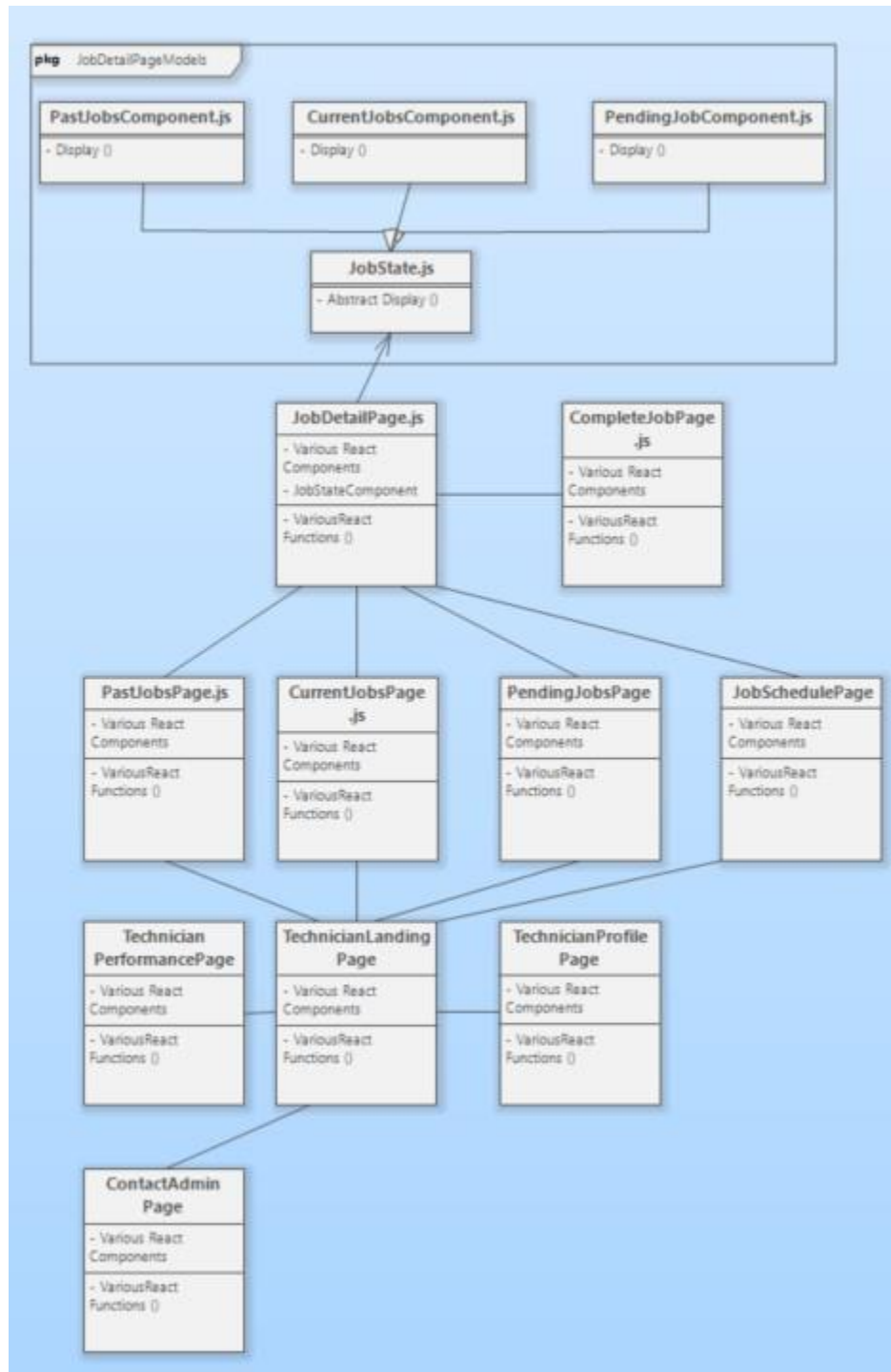


Figure 46. Technician Front-End Classes

These are the JavaScript classes that will render the front end for the Technician. These are the base classes we feel that we will need to encapsulate the needs of the user. Note the design pattern regarding state. Depending on the job state, the corresponding state component will display the page

differently. If the use of a design pattern brings too much overhead, then we will have separate pages for this functionality.

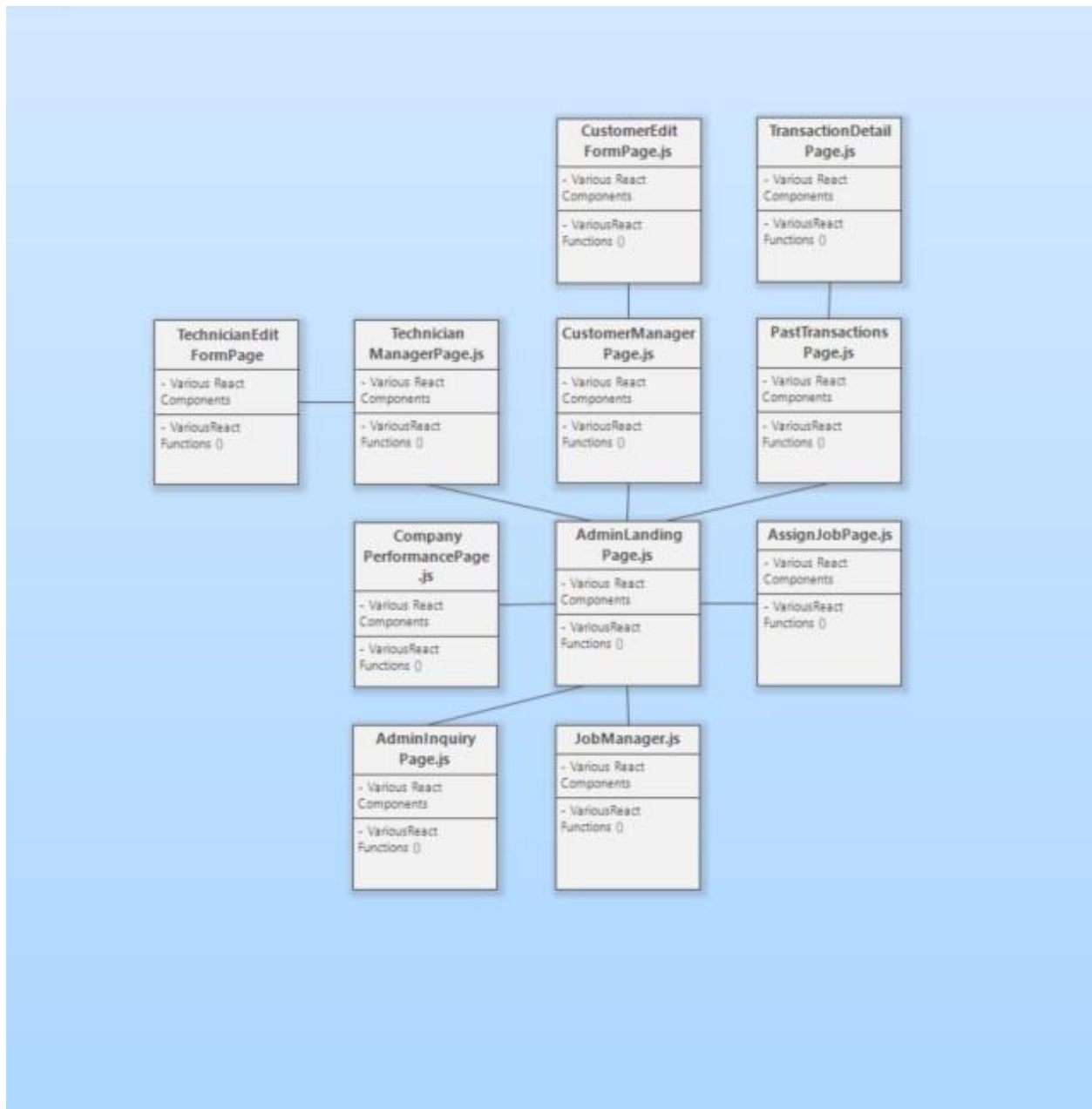


Figure 47.. Admin Front-End Classes

These are the JavaScript classes that will render the front end for the Admin. These are the base classes we feel that we will need to encapsulate the needs of the user.



Figure 48.. Login Front-End Diagram

Before the user is logged in, they are neither an admin, technician, nor customer. These classes represent the front-end interfaces and functionalities available to these users.

7. Interaction Sequence Diagram

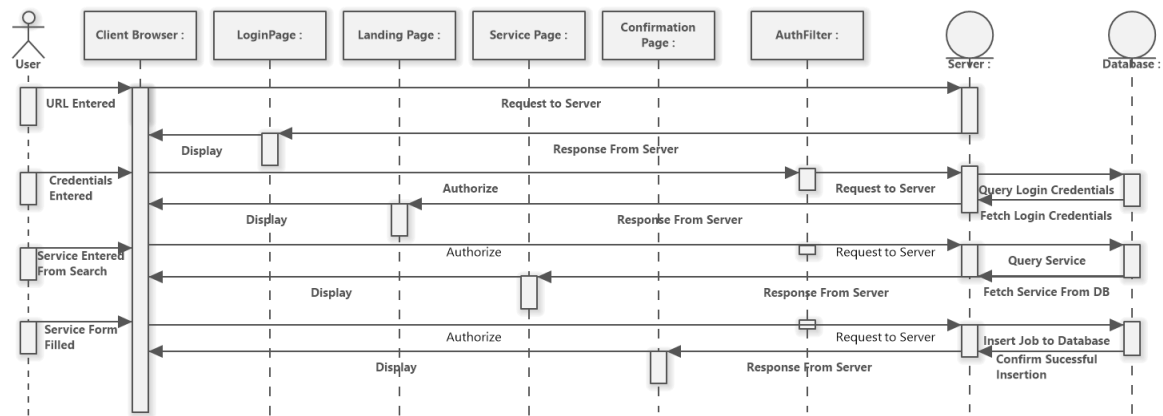


Figure 49. Service Sequence Diagram

Service sequence diagram shows interaction that how customer can log in, view and select service that he needs with the server and database.

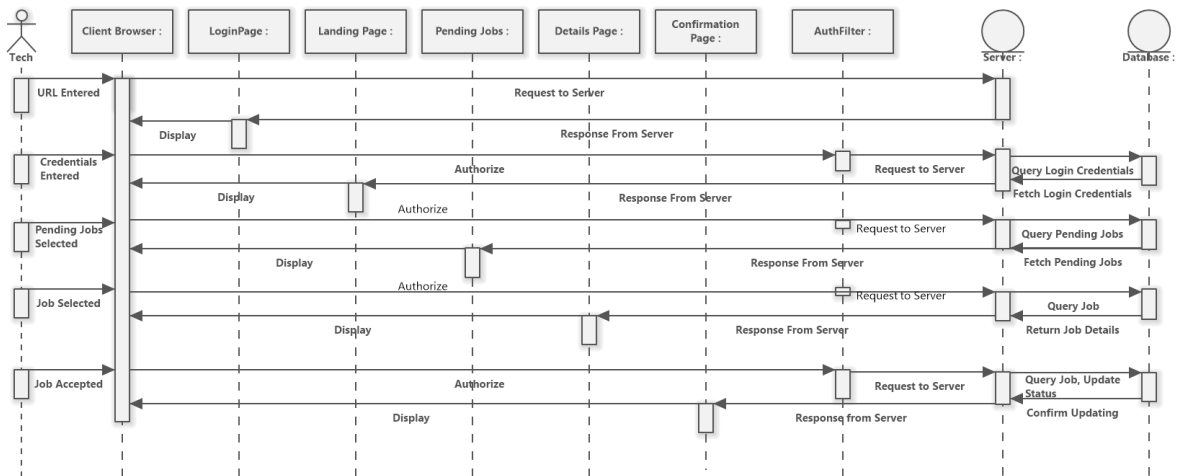


Figure 50. Technician Job Acceptance Diagram

Sequence that a technician will undertake to accept a new job request that has been uploaded to the data base.

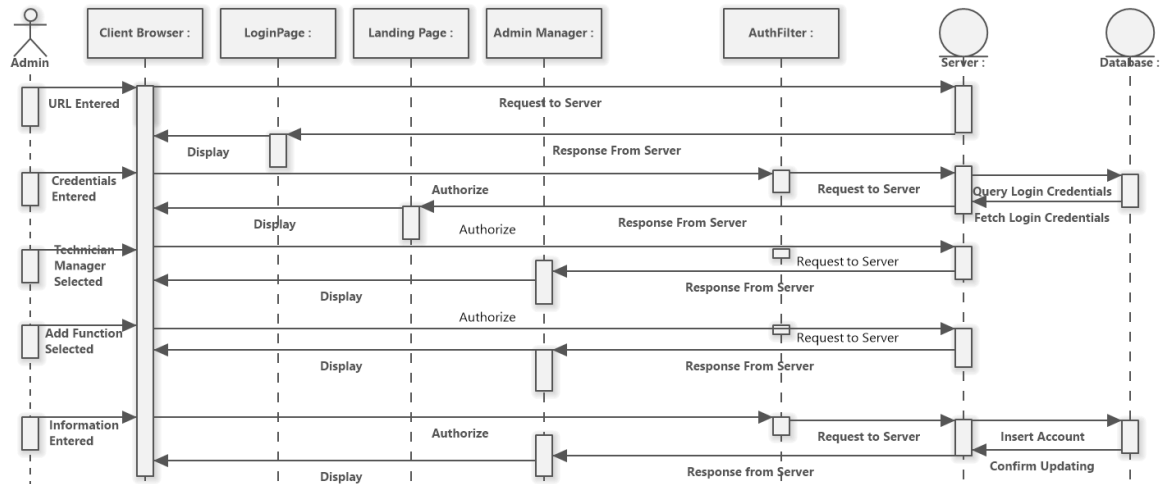


Figure 51. Adding Technician Sequence Diagram

Adding technician sequence diagram shows interactions that how admin can log in, view, and add a technician to the data base.

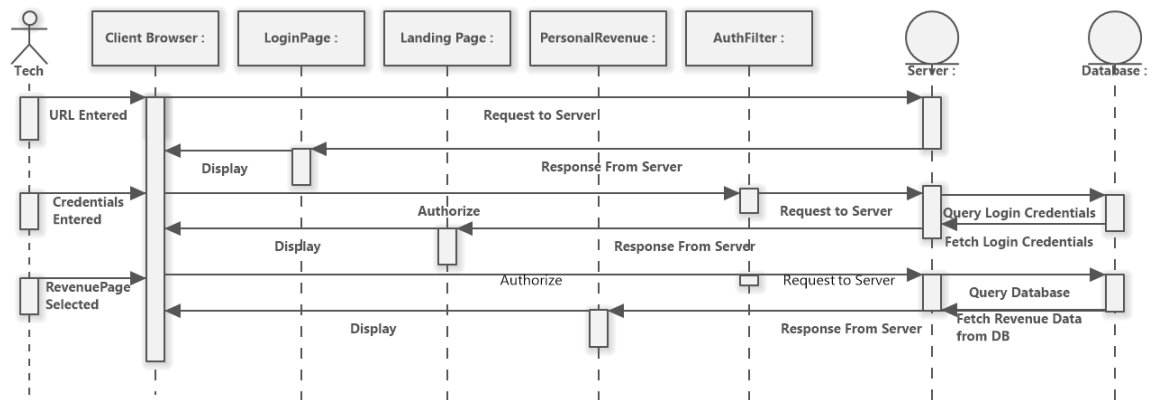


Figure 52. View Personal Revenue Sequence Diagram

View personal revenue sequence diagram shows interactions that how the technician can log in, view his information and revenue with server and database.

8. Project Management

8.1 Schedule

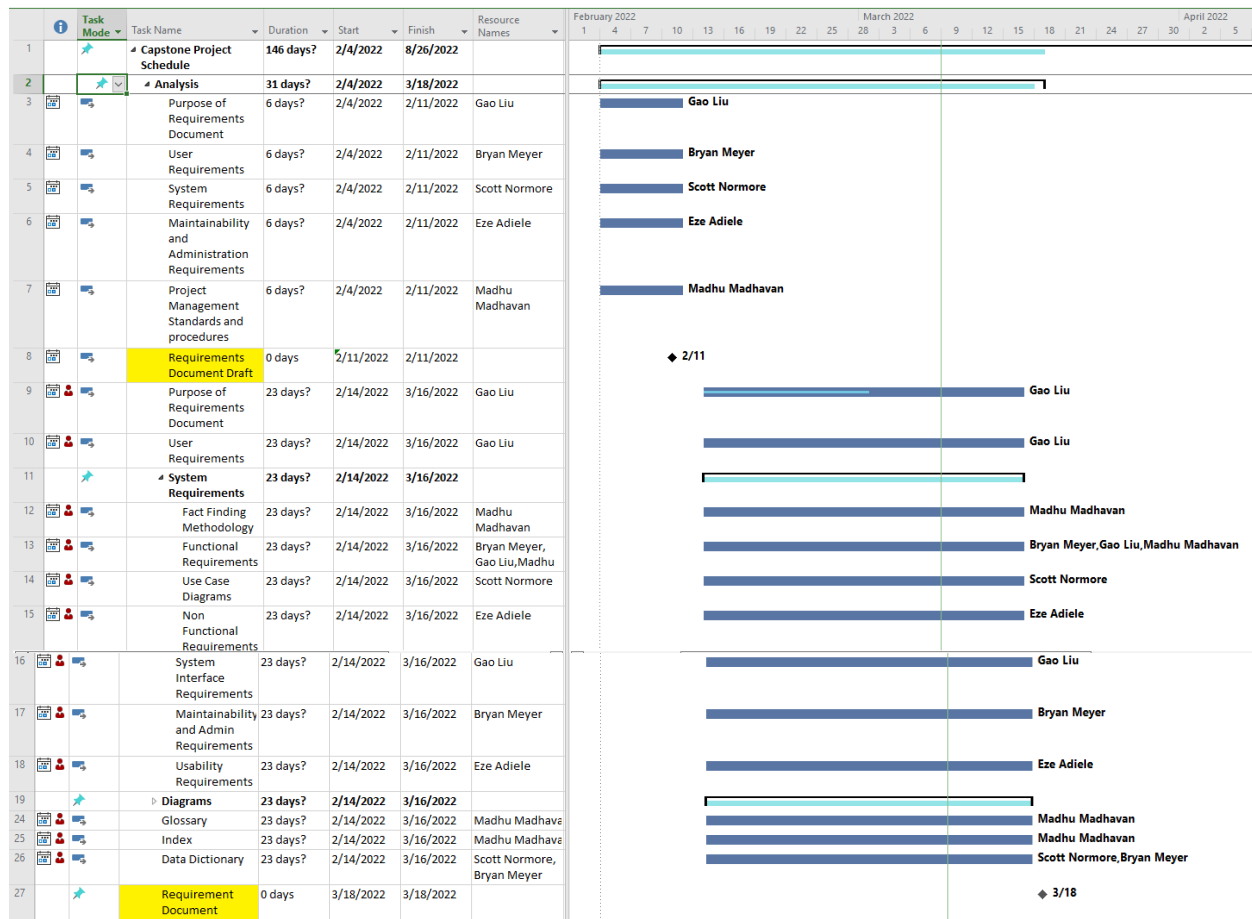


Figure 53. GANTT Chart Schedule

More Schedule tasks will be added as we progress through the project and assign ownerships accordingly.

8.2 Team Configuration

Members	Roles	Reporting	Contact Information
Gao Liu		Ali Moussa	gao.liu@edu.sait.ca
Scott Normore		Ali Moussa	scott.normore@edu.sait.ca
Bryan Meyer		Ali Moussa	bryan.meyer@edu.sait.ca
Eze Adiele		Ali Moussa	eze.adiele@edu.sait.ca
Madhu Madhavan		Ali Moussa	madhu.madhavan@edu.sait.ca

Our roles tend to change on every task, so we have not designated any roles in this project for now.

8.3 Project Standards and Procedures

8.3.1 Project Procedures

- The project will be broken down into tasks and it has been decided that two of us from the work group will lead every task. They will be responsible for scheduling that task, communicating with all stakeholders, and lead the rest of the group to finish the task on time. They would take the ownership of their task deliverables. By doing this we are creating an opportunity for all group members to gain experience in managing a task or part of a project. Project dependencies will be charted down in the Gantt chart project scheduler and made available to all stakeholders at any given time.
- Team progress will be monitored on a granular, task-by-task level and on a bigger-picture level using visual Gantt chart which represent the percentage of project work completed, relative to the available time.
- Over and above the allotted work periods during CMPS-303 and PROJ-304, we have already scheduled Tuesday and Wednesday evenings (6pm - 7:30pm) to work on this project. We will be using Microsoft Teams to conduct these meetings. Minutes of the meeting will be documented along with video recordings for reference.

8.3.2 Project Standards

- Web applications must meet the following general standards:
 - Must be easy and intuitive to use for the target audience.
 - Must function in a logical manner for the target audience.
 - Must use styles that are consistent throughout the application, including:
 - The use of capitalization (e.g., title case vs. sentence case).
 - The use of punctuation (e.g., consistent use/no use of colons on labels).
 - Error messages must appear in a consistent location and style.
 - Consistent use of any web document notations (e.g., PDF, DOC, etc.).
 - Layout/spacing (e.g., the space between a field label and input control).
- Must adhere to industry best practices.
- Web applications require the use of the following (or higher) technologies:
 - **Framework:** JDK 15
 - **Development Environment:** Eclipse, Netbeans, IntelliJ, Visual Studio
 - **Server-Side Tech:** Java 1.8, NodeJS,
 - **Front-End Tech:** HTML, Bootstrap, CSS, AJAX, React
 - **Database:** Oracle SQL 12c, MongoDB
 - **Web Server:** Tomcat Live Server (Testing)
- For optimum performance, Web applications must function and display properly in browser versions such as Mozilla Firefox, Apple Safari, Google Chrome, and Microsoft Edge (based on Chromium). Web applications must be thoroughly tested in all required browser versions.

- The web application must be responsive. The application must resize correctly and be functional on mobile devices.
- Every web page in the application must have one or more links or control buttons that allow a user to navigate back and forth within the application without having to use the back button or other browser navigation functionality.
- Form fields must be validated to ensure required fields are completed, numeric fields have numeric data, and data input is properly formatted (e.g., date, email address).
- Code exceptions must be handled in a user-friendly manner by displaying a custom error page that does not display information such as database object names or source code.
- The HTML code in all web applications must be valid via a reputable validation technique, such as W3C or by using the HTML Validator Firefox.
- Textbox input controls in a web form must have properties set for display width and maximum input characters.
- The use of JavaScript is allowed for client-side data validation and manipulation as long as the script is invoked as a result of a user action (i.e., button selection, dropdown selection, movement to another form field, etc.).
- AJAX elements can be included as long as there is an equivalent non-AJAX alternative that produces the same results or provides the same functionality.

9. Glossary

- **JDK 15** JDK 15 is the open-source reference implementation of version 15 of the Java SE Platform
- **Eclipse** Eclipse is an integrated development environment used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment.
- **Netbeans** NetBeans is an integrated development environment for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris.
- **IntelliJ** IntelliJ IDEA is an integrated development environment written in Java for developing computer software. It is developed by JetBrains and is available as an Apache 2 Licensed community edition, and in a proprietary commercial edition.
- **Visual Studio** Microsoft Visual Studio is an integrated development environment from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps
- **NodeJS** Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.
- **HTML** The Hyper Text Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser.

- Bootstrap Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

- CSS Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.

- AJAX AJAX is a set of web development techniques that uses various web technologies on the client-side to create asynchronous web applications. With Ajax, web applications can send and retrieve data from a server asynchronously without interfering with the display and behavior of the existing page.

- SQL12c Structured Query Language (SQL) is the set of statements with which all programs and users access data in an Oracle Database. Application programs and Oracle tools often allow users access to the database without using SQL directly, but these applications in turn must use SQL when executing the user's request.

- MongoDB MongoDB is a cross-platform modern database system. It can be manageable in the cloud. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas

- Tomcat Tomcat Live Server is an open-source Java servlet and Java Server Page container that lets developers implement an array of enterprise Java applications. Tomcat also runs a HTTP web server environment in which Java code can run.

- Use case A use case diagram is a graphical depiction of a user's possible interactions with system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well

- **Class Diagram** The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code.
- **Activity Diagram** Activity Diagram is a behavioral diagram that depicts the behavior of a system. It portrays the control flow from start to finish of an activity, showing various decision making along the execution path of the activity.
- **Sequence Diagram** A diagram that depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of scenario.
- **State Machine** Any device that stores the status of an object at a given time and can change status or cause other actions based on the input it receives. States of an object refer to the different combinations of information that an object can hold, not how the object behaves. State Machine Diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that shows transitions of states of an object between various actions and conditions.
- **NDA** Non-Disclosure Agreement signed between the Client and Developers
- **NPM** npm is a package manager for the JavaScript programming language maintained by npm, Inc. npm is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry
- **EXPRESS JS** Express.js, or simply Express, is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js
- **REACT** React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies.

10. Index

Figure 1: Use Case Diagram

Figure 2: System Interface Diagram

Figure 3: Problem Domain Class Diagram

Figure 4: Service Sequence Diagram

Figure 5: Review Sequence Diagram

Figure 6: Adding Technician Sequence Diagram

Figure 7: View Personal Revenue Sequence Diagram

Figure 8: State Machine Diagram

Figure 8: Activity Diagram

Figure 9: Login Activity Diagram

Figure 10: Layered Architecture Model

Figure 11. Packaged Layer Architecture

Figure 12. MERN stack development Model

Figure 13. MERN software interaction

Figure 14. Welcome page

Figure 15. Booking Page

Figure 16. Manage Profile

Figure 17. Ongoing Services

Figure 18. Landing Page

Figure 19. Login Page

Figure 20. Reset Password Page

Figure 21. New Account Page

Figure 22. About Us Page

Figure 23. Contact Us Page

Figure 24. Service Page

Figure 25. Booking Service Page

Figure 26. Booking Confirmation Page

Figure 27. Confirmation of inquiry Page

Figure 28. Technician Home Page

Figure 29. Performance Detail Page

Figure 30. Schedule Page

Figure 31. Job details Page

Figure 32. Payment Information Page

Figure 33. Admin's Home Page

Figure 34. Manage Customer's Page

Figure 35. Manage Technician Page

Figure 36. Financial Data Page

Figure. 37 Technician Payment Page

Figure 38. Job Manager Page

Figure 39. Conceptual ERD

Figure 40. Physical ERD

Figure 41. Complete Class Diagram

Figure 42. Models Class Diagram

Figure 43. Backend Diagram

Figure 44: Front End Diagram

Figure 45. Customer Front-End Classes

Figure 46. Technician Front-End Classes

Figure 47. Admin Front-End Classes

Figure 48. Login Front-End Diagram

Figure 49. Service Sequence Diagram

Figure 50. Technician Job Acceptance Diagram

Figure 51. Adding Technician Sequence Diagram

Figure 52. View Personal Revenue Sequence Diagram

Figure 53. GANTT Chart Schedule

11. Appendix A: Data Dictionary

Models

Account	
Class Description	An abstract model class used by the server to handle account information from the database. This class will be used to handle the login functionality.
Operations	Getters and Setters for its attributes
Attributes	Username: String Password: String ID: Int String: firstName String: lastName
Class Association	Admin, Tech, Customer, All Route classes, All Database classes, All Filters, LoginPage, AccountCreatePage, ForgotAccountPage.

Admin	
Class Description	A model class used by the server to handle admin information from the database. This includes all the information from account in addition to admin Specific information.
Operations	Getters and Setters for its attributes, Verify Pin
Attributes	Attributes from super class, PIN
Class Association	Account, AdminRoutes, LoginRoutes, Admin Filters, Inquiry

Customer	
Class Description	A model class used by the server to handle customer information from the database. Some customers do not have account information but have information that must be tracked.
Operations	Getters and Setters for its attributes
Attributes	Attributes from super class, CreditCardNumber: Number PhoneNumber: String

	Address: String Email: String
Class Association	Account, Customer Routes, Technician Routes, Admin Routes, Job, Review, All Customer Pages, JobDetailPage, TechnicianPerformancePage, Customer Manager Page

Tech	
Class Description	A model class used by the server to handle technician information from the database. This includes all the information from account in addition to technician Specific information.
Operations	Getters and Setters for its attributes
Attributes	Attributes from super class, CreditCardNumber: Number PhoneNumber: String Address: String Email: String SIN: int typeOfService: String[] – Contains the permissions of the technician.
Class Association	Account, Customer Routes, Technician Routes, Admin Routes, Job, Review, JobDetailPage,

Job	
Class Description	Class representing the jobs created by the customers and that are assigned to technicians
Operations	Getters and Setters
Attributes	Details: String Time: Date Service: String Customer: Customer Technician: Tech Status: String
Class Association	Customer, Technician, Customer Routes, Technician Routes, Amin Routes

Review	
Class Description	Holds and manages customer reviews of technicians
Operations	Setters and Getters
Attributes	Customer: Customer Technician: Tech Score: int

	Service: String Content: String
Class Association	Technician, Customer, Customer Routes, Technician Routes, Admin Routes, LoginPage, All Admin Pages

Log	
Class Description	Class representing error or other logs
Operations	Setters and Getters
Attributes	Description: String
Class Association	Any class that can output an error

Invoice	
Class Description	Manages receipts for jobs completed by technicians
Operations	
Attributes	Date: Date Customer: Customer Technician: Tech Amount: double ServiceType: String TaxesAmount: double
Class Association	Technician, Customer, Admin Routes, Invoice Manager

Inquiry	
Class Description	Class representing messages between admins and customers/technicians
Operations	Setters and Getters, Display
Attributes	Content: String Technician: Tech Customer: Customer
Class Association	Tech, Customer, Admin, AdminInquiryPage, ContactAdminPage

Server

Server.js	
Class Description	Class that establishes connection to database using Conn.js and starts the server.
Operations	connectToDatabase() listen()
Attributes	
Class Association	Conn.js, Database and Route Classes.

Routes

LoginRoutes	
Class Description	Server-side routes that handle user-less endpoints that handle the functionality of logging in, account creation, and account retrieval of the application. Functions here will insert new users or create new login sessions.
Operations	LoginPageFunctions AccountCreatePageFunctions ForgotAccountPageFunctions
Attributes	
Class Association	Account, All Login Pages, Login Queries

AdminRoutes	
Class Description	Server-side routes that handle the admin endpoints. Handles the functionality of all admin interfaces. Functions here are for managing accounts, payments, financial information, and assigning/managing jobs.
Operations	AdminLanfingPageFunctions JobManagerFunctions, AdminInquiryPageFunctions AssignJobsPageFunctions CompanyPerformancePageFunctions CustomerManagerPageFunctions TechnicianManagerPageFunctions TechnicanEditFormPageFunctions CustomerEditFormPageFunctions TransactionDetailPageFunctions
Attributes	
Class Association	Admin Queries, All admin Pages, All Models

TechnicianRoutes	
Class Description	Server-side routes that handle the Technician endpoints. Handles the functionality of all Technician interfaces. Functions here are for technicians managing their own technician jobs and performance information
Operations	TechnicianLandingPageFunctions TechnicianPerformancePageFunctions TechnicianProfilePageFunctions PastJobsPageFunctions CurrentJobsPageFunctions JobSchedulePageFunctions JobDetailPageFunctions CompleteJobPageFunctions
Attributes	
Class Association	TechQueries, All Technicina Pages, Customer, Jobs, Techs, Reviews and invoice models

CustomerRoutes	
Class Description	Server-side routes that handle the customer endpoints. Handles the functionality of all the customer interfaces. Functions here are for customer to book a job, view past jobs, and leave a review.
Operations	LandingPageFunctions CurrentJobsFunctions JobDetailsPageFunctions PastJobFunctions ContactTechPageFunctions ReviewFormFunctions InquiryToAdminFunctions CustomerProfilePageFunctions DetailPageFunctions
Attributes	
Class Association	Customer Queries, All Customer Pages, Customer, Technicians, Job, and Inquiry Models.

Database

Conn.js	
Class Description	Class that contains the required functions to connect to the database
Operations	ConnectToDataBase()
Attributes	Database information
Class Association	All query classes, Server.js

Customer Queries	
Class Description	Queries that are used by the customer endpoints are contained within this class
Operations	getServices() getCurrentJobs() getCurrentJobs(email) saveReview(Customer, Tech, Content) uploadInquiry(email, content) createUnassignedJob(Customer, Formdetails)
Attributes	
Class Association	Customer Routes, Conn.js, Model Classes

Technician Queries	
Class Description	Queries that are used by the Technician endpoints are contained within this class.
Operations	getCurrentJobs(technician) getPastJobs(technician) completeJob(job) calcRevenue(technician) calcAvgRating(technician) getCustomerReviews(technicians) uploadInquiry(email, content)
Attributes	
Class Association	Customer Routes, Conn.js, Model Classes

Admin Queries	
Class Description	Queries that are used by the Admin endpoints are contained within this class.
Operations	EditTech(tech) AddTech(tech) RemoveTech(tech) EditCustomer(customer) AddCustomer(customer) RemoveCustomer(customer) AddJob(job) removeJob(job) getCompanyRevenue() getAverageRating() getInquirys() getTransactions() refundTransaction(invoice) editTransaction(invoice)
Attributes	
Class Association	Admin Routes, Conn.js, Model Classes

Login Queries	
Class Description	Queries that are used by the user-less endpoints are contained within this class.
Operations	getUser(email, password) addUser(user) setUUID(user) resetPassword(UUID, password)
Attributes	
Class Association	Login Routes, Conn.js, Model Classes

FrontEnd Components

Index.html	
Class Description	The single page in our single page application. Is used to display the component chosen by App.js
Operations	
Attributes	
Class Association	App.js, all Page Classes.

App.js	
Class Description	The class responsible for choosing which React component page to render.
Operations	Control Variables
Attributes	DisplayNavbar() Display(page) VariousControlFunctions
Class Association	App.js, all Page Classes.

Customer

CustomerLandingPage	
Class Description	The React Component Class responsible for rendering the Customer Landing page to the user. Plan on including a search function along with navigation options to the other Customer Classes.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	CurrentJobs, Inquiry to Admin, Customer Profile, All Service Pages, CustomerRoutes, App.js

CustomerProfilePage	
Class Description	The React Component Class responsible for rendering the Customer profile page. Here will have the functionality to change customer details.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	Customer Landing, CustomerRoutes. App.js

InquiryToAdminPage	
Class Description	The React Component Class responsible for customer to admin inquires.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	Customer Landing, CustomerRoutes, App.js

CurrentJobsPage	
Class Description	The React Component Class responsible for showing ongoing jobs
Operations	Various javascript functions
Attributes	Various React Components

	Various Control Variables
Class Association	Customer Landing, CustomerRoutes, App.js

ContactTechnicianPage	
Class Description	The React Component Class responsible for customer to contact the technician
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	CurrentJobsPage, PastJobsPage, CustomerRoutes, App.js

JobDetailsPage	
Class Description	The React Component Class responsible for displaying the details of a job
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	CurrentJobsPage, Past Jobs Page, CustomerRoutes, App.js

PastJobsPage	
Class Description	The React Component Class responsible for showing past jobs
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	CurrentJobsPage, CustomerRoutes, App.js

ReviewFormPage	
Class Description	The React Component Class responsible allowing a customer to edit or leave a review on a job.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	PastJobsPage, CustomerRoutes, App.js

ServiceDetailPages(several)	
Class Description	The React Component Class responsible for showing the details of a given job
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	Customer Landing, CustomerRoutes, App.js

ServiceFormPages(several)	
----------------------------------	--

Class Description	The React Component Class responsible for handling the submission of information on a customer Job.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	Customer Landing, CustomerRoutes, App.js

Technician

Technician Landing Page	
Class Description	The React Component Class responsible for displaying the interface that will handle the technician's navigation to other pages. In addition, we plan on having an alert box to update the technician on new jobs, or customer notifications.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	ContactAdmin, TechPerformancePage, TechnicianProfilePage, PastJobsPage, CurrentJobsPage, PendingJobsPage, JobSchedulePage, TechnicianRoutes, App.js

Contact Admin Page	
Class Description	The React Component Class responsible for displaying the interface that allows a tech to contact the admin via inquiry.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	TechnicianLandingPage, TechnicianRoutes, App.js

Technician Performance Page	
Class Description	The React Component Class responsible for displaying the interface that allows a tech to view performance data
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	TechnicianLandingPage, TechnicianRoutes, App.js

Technician Profile Page

Class Description	The React Component Class responsible for displaying the interface that allows a tech to view performance data
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	TechnicianLandingPage, TechnicianRoutes, App.js

JobSchedulePage	
Class Description	The React Component Class responsible for displaying the interface allows a tech to see their assigned jobs in a calander format
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	TechnicianLandingPage, TechnicianRoutes, App.js

CompleteJobPage	
Class Description	The React Component Class responsible for displaying the interface allows a tech to complete a job and charge payment.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	JobDetailPage, TechnicianRoutes, App.js

JobState	
Class Description	The React Component Class that is a subcomponent for the Job Detail Page that will be different depending on job state.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	JobDetailPage

Admin

AdminLandingPage	
Class Description	The React Component Class responsible for displaying the interface allows an admin to navigate their interfaces. Also may contain other functionalities such as inquiry alerts.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	JobManager,AdminInquiryPage,CompanyPerformancePage,AssignJobPage, CustomerManagerPage,PastTransactionsPage,TechnicianManagerPage, AdminRoutes,App.js

AdminInquiryPage	
-------------------------	--

Class Description	The React Component Class responsible for displaying the interface allows an admin to respond to inquiries.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	AdminLandingPage, AdminRoutes,App.js

CompanyPerformancePage	
Class Description	The React Component Class responsible for displaying the interface allows an admin to view financial and performance data regarding to the company
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	AdminLandingPage, AdminRoutes,App.js

JobManagerPage	
Class Description	The React Component Class responsible for displaying the interface allows an admin manage the ongoing jobs in the system.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	AdminLandingPage, AdminRoutes,App.js

AssignJobPage	
Class Description	The React Component Class responsible for displaying the interface allows an admin to directly assign a tech to a job
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	AdminLandingPage, AdminRoutes,App.js

TechnicianManagerPage	
Class Description	The React Component Class responsible for displaying the interface that allows an admin to manage the properties of a technician
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	AdminLandingPage, AdminRoutes,App.js

CustomerManagerPage	
Class Description	The React Component Class responsible for displaying the interface that allows an admin to manage the properties of a customer
Operations	Various javascript functions
Attributes	Various React Components

	Various Control Variables
Class Association	AdminLandingPage, AdminRoutes,App.js

PastTransactionPage	
Class Description	The React Component Class responsible for displaying the interface that allows an admin to manage and view past transactions
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	AdminLandingPage, AdminRoutes,App.js

Login

LoginPage	
Class Description	The React Component Class responsible for displaying the interface that allows one to login to the application
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	ForgotPassword, CreateAccountPage, LoginRoutes,App.js

AccountCreatePage	
Class Description	The React Component Class responsible for displaying the interface that allows one to create an account
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	LoginPage, LoginRoutes,App.js

ForgotAccountPage	
Class Description	The React Component Class responsible for displaying the interface that allows one to recover a forgotten password.
Operations	Various javascript functions
Attributes	Various React Components Various Control Variables
Class Association	LoginPage, LoginRoutes,App.js