Term Project Specification Report: Group 5, TechAssist Application

[redacted]

Natibundit Suntraranusorn [redacted]

CSIS3275-001 Software Engineering

[redacted]

December 5, 2023

Table of Contents

1. Project Overview

I. Introduction:

The Tech Assist Application seeks to bridge the gap between individuals facing immediate household or automotive issues and skilled technicians. By leveraging video calls, customers can connect with technicians for guidance on mechanical repairs or consultations. This introduction provides a brief background of the company motivation behind developing the Tech Assist Application.

II. The business case:

In Canada, there are many people who are really in need of mechanical repairs or plumbing because they are too expensive. The fact is that the average cost of hiring a plumber is a few hundred dollars to over a thousand dollars. There is a great demand for affordable mechanical repairs or consultation services such as a need for a short mechanical consultation. At the same time, there are many technicians who have mechanical knowledge and are looking for opportunities to use it. In order to meet these strong demands, we decided to develop an application that connects customers with technicians. In the application, the customers can choose a technician with expertise in a particular field and choose the length of time to get advice on how to fix the problems. It also allows the technicians to utilize their abilities in their spare time and earn money. The cost of payment is determined based on abilities and time they used.

III. Problem statement:

In today's world, individuals often encounter unexpected household or automotive issues such as broken pipes or flat tires, necessitating immediate solutions. The lack of accessible and efficient methods for resolving these problems often results in frustration, wasted time, and potential damage. Furthermore, locating a qualified technician to provide real-time guidance can be a daunting task. Our proposed web-app seeks to address this pressing issue by connecting users facing such predicaments with skilled technicians via video calls. However, several challenges and concerns need to be addressed:

- User Accessibility: Ensuring that users of all technological backgrounds can easily navigate and utilize the platform is paramount. The app should be user-friendly and accessible to a broad audience.
- Technical Reliability: Maintaining a secure and stable video call infrastructure is essential for delivering high-quality real-time assistance. Technical glitches or connection issues could hinder the effectiveness of the service.
- Trust and safety: Establishing a sense of trust between users and technicians is crucial. Users need assurance that technicians are qualified and trustworthy, while technicians need safeguards to protect their personal information.

- Monetization Strategy: Developing a fair and transparent pricing model that accurately charges users based on call duration without creating financial barriers or dissatisfaction is a significant challenge.
- Legal and Compliance: Complying with local and international regulations related to data privacy, payment processing, and liability is vital to prevent legal complications.

To address these concerns, we are diligently working on creating a robust ,user-friendly and secure platform that fosters trust, reliability and efficiency.

IV. The software engineering paradigm:

The team applied Functional Programming Paradigm. It focuses on composing functions and data. There are three parts, which are the client functions, technician functions, and the video calling function. Basically, each function is independent in terms of workflow, so Functional Programming Paradigm should be the best software engineering paradigm for this project. Team members have been assigned one of the functions. After completing each function, it was integrated into the mainstream with integration tests.

2. Requirements

I. Functional Requirements:

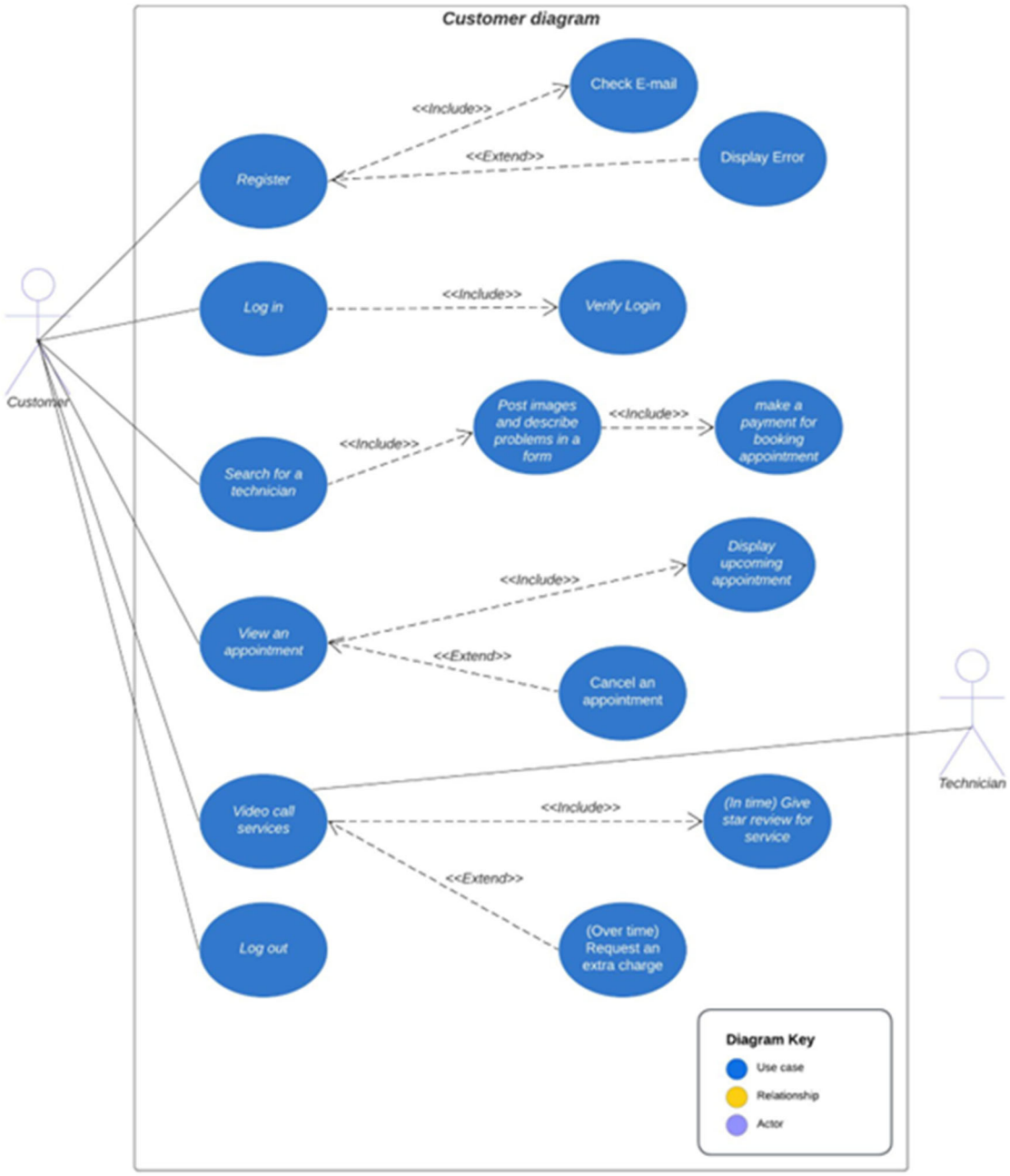
- Customers and technicians can login to their accounts.
- Customers can connect to technicians through a video call.
- Customers can register for the service of a technician for 15, 30, 45, 60 minutes.
- Customers can pay for the services the technician provided before the call.
- Customers can rate the technicians and read the comments of the technicians.
- Technicians can provide their available time and rate.
- Technicians can provide their specialties for customers to find them.

II. Non-Functional Requirements:

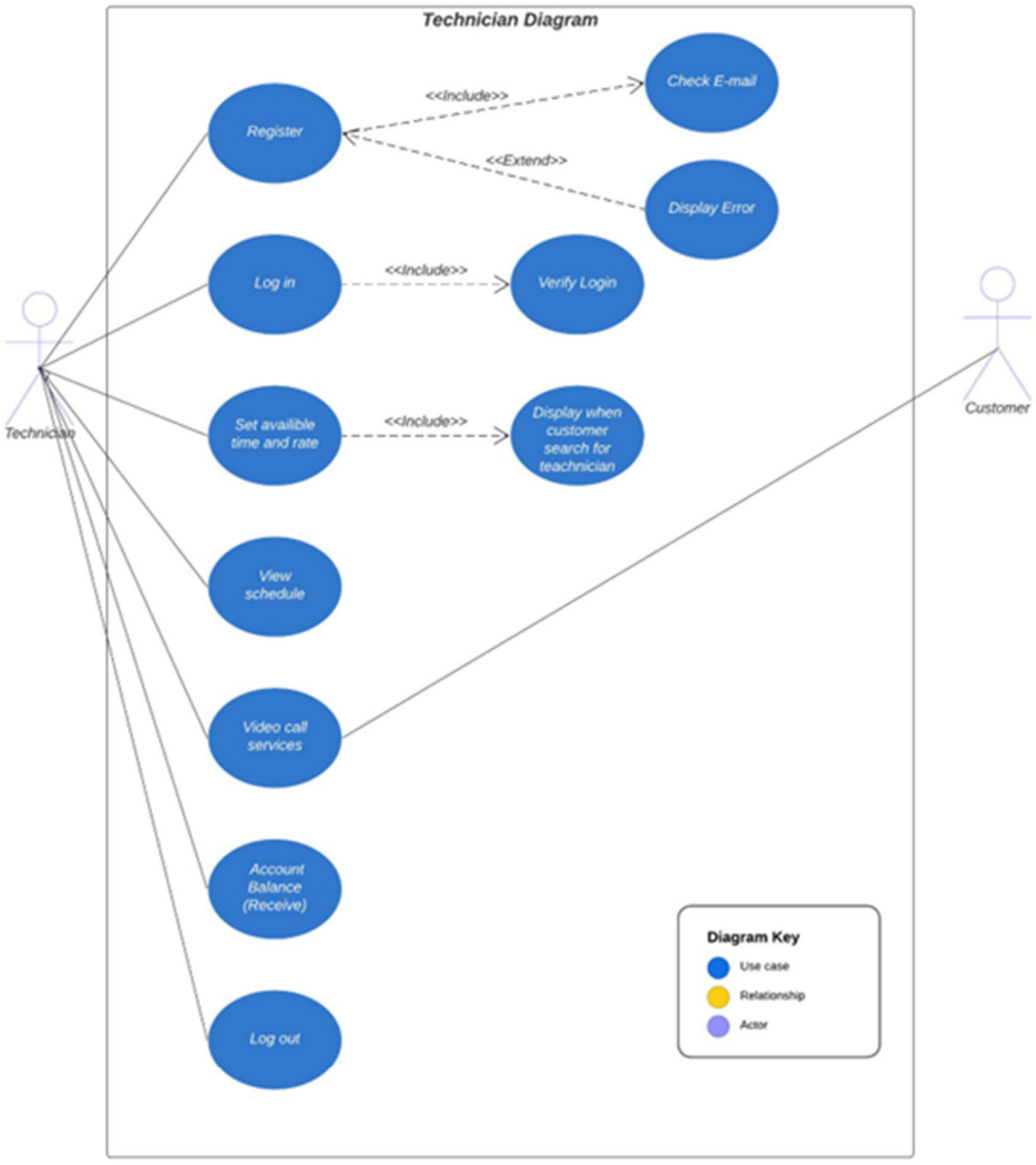
- Users can use the application with the web browser on their phone.
- The application can display correctly on the phone, but it does not have to be RWD.
- The video calling feature should be available and functional on the latest Android and iOS phones.

III. Use cases Diagrams:

The pipe repair service App.



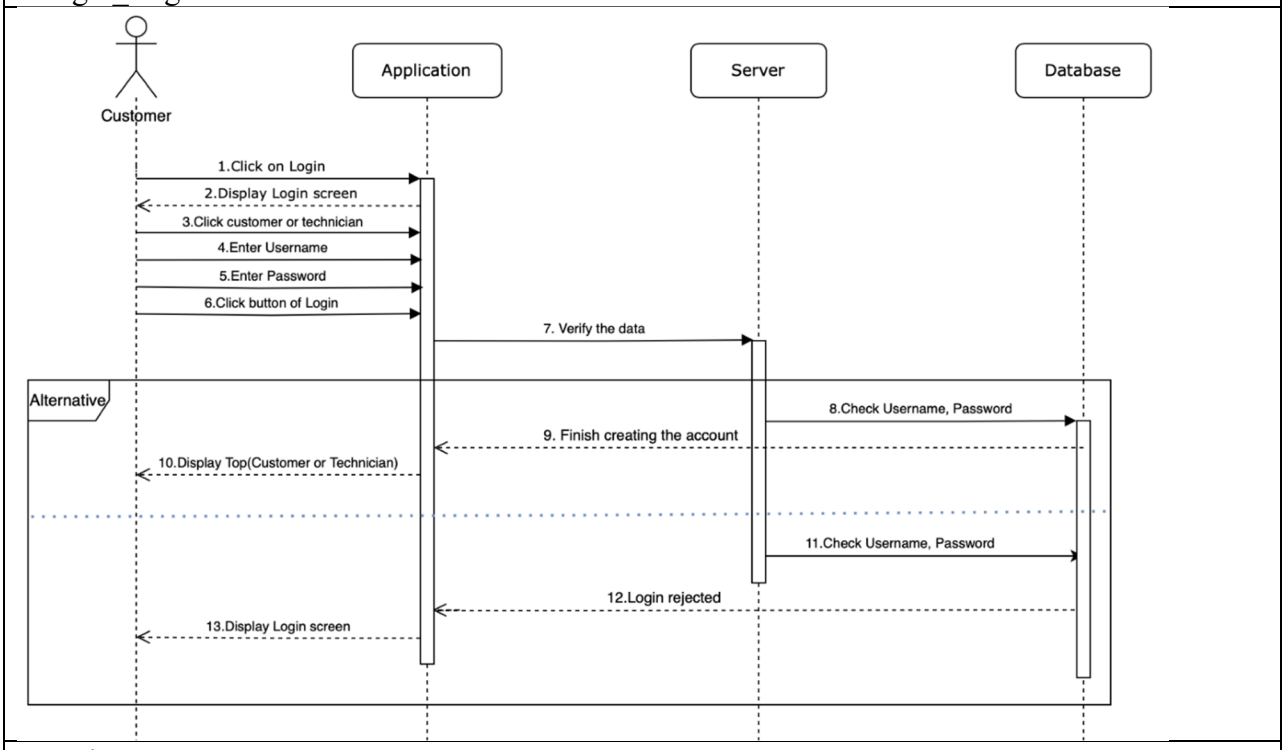
The pipe repair service App.



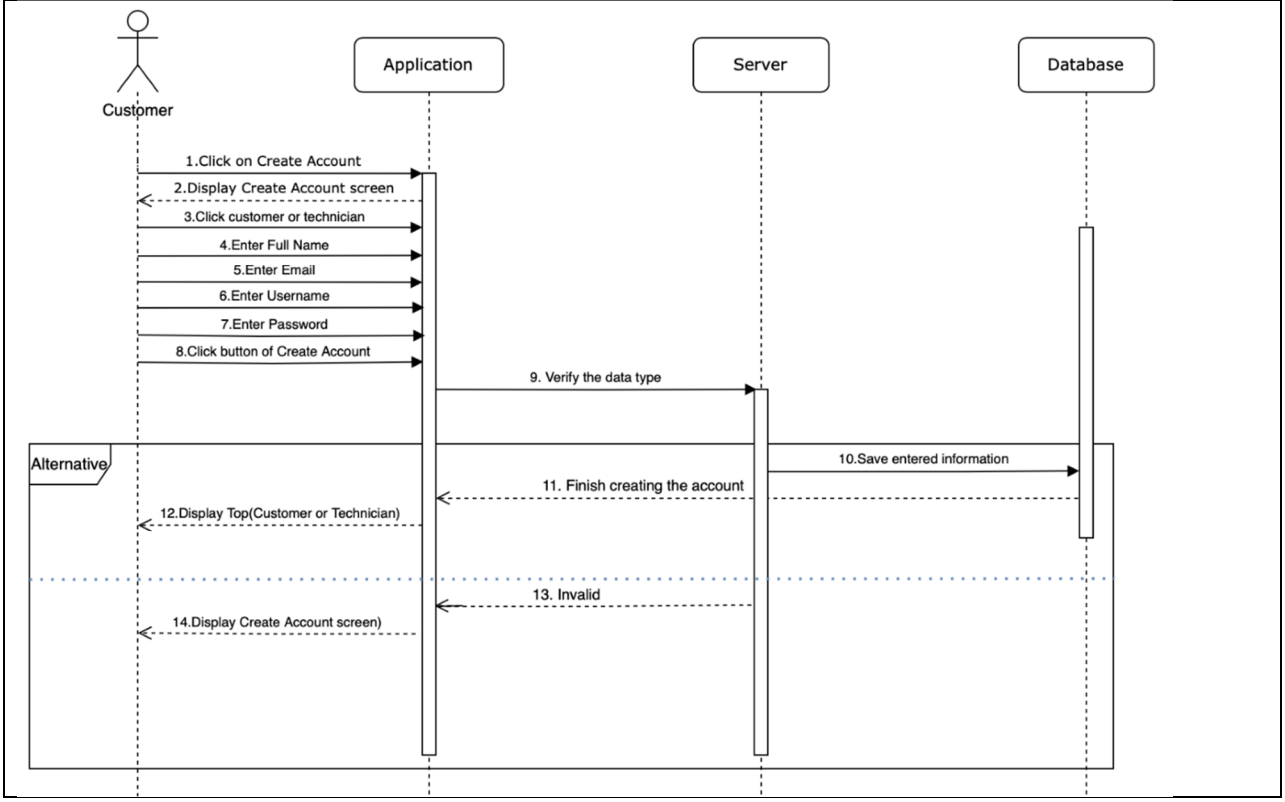
IV. Dynamic Models (Sequence):

Customer view

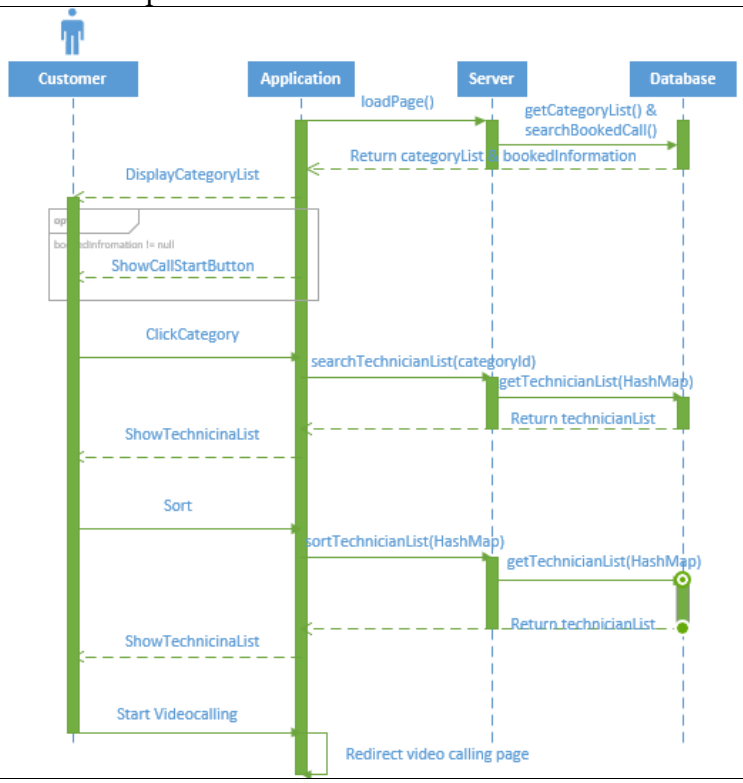
<Login_Login>



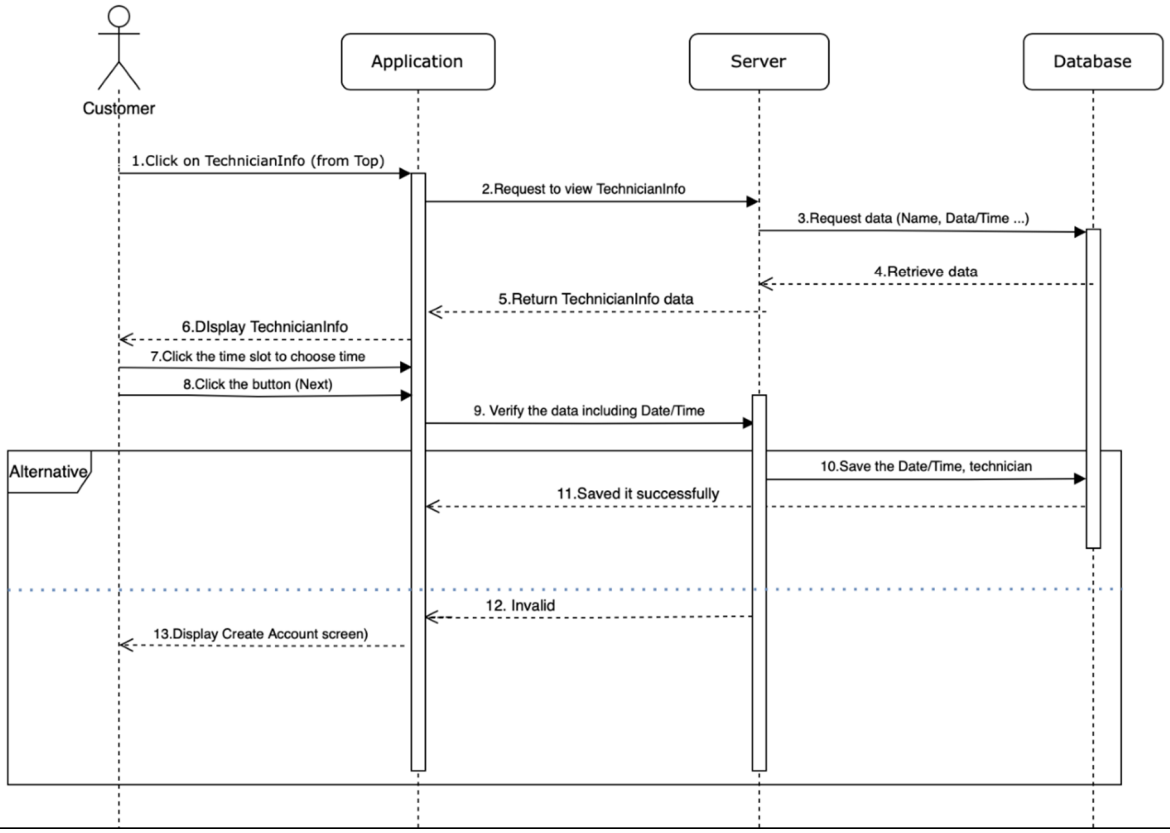
<Login_CreateAccount>



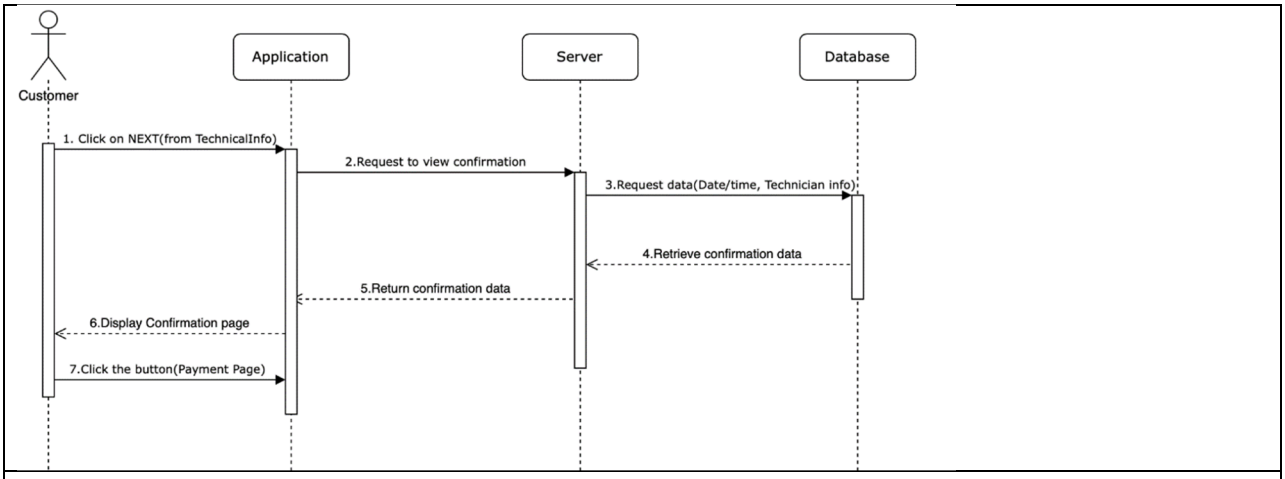
<Customer top>



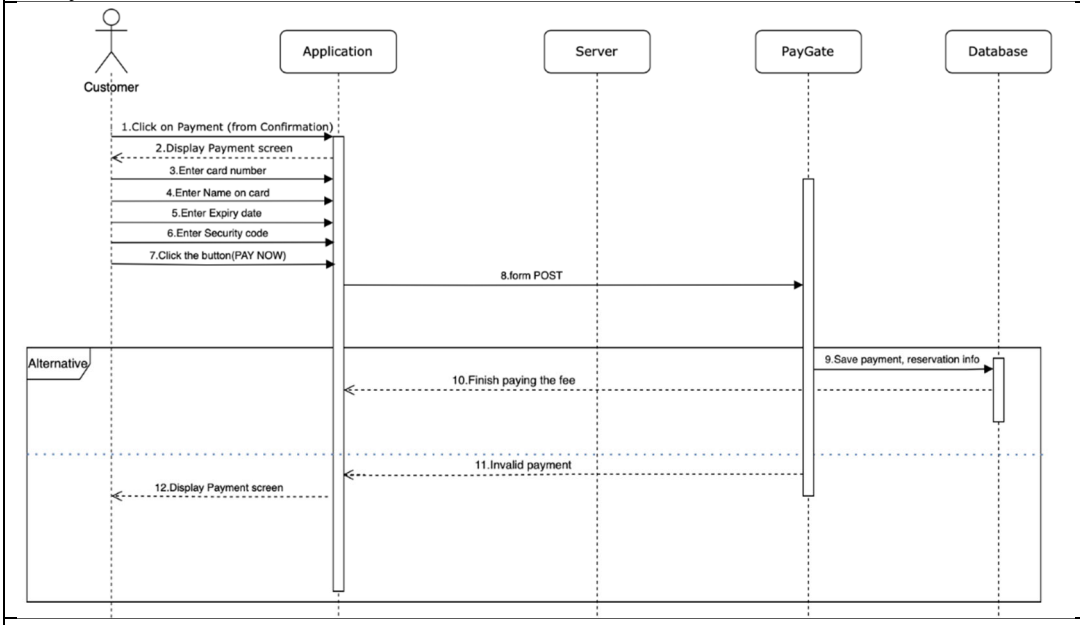
<Technician_Info>



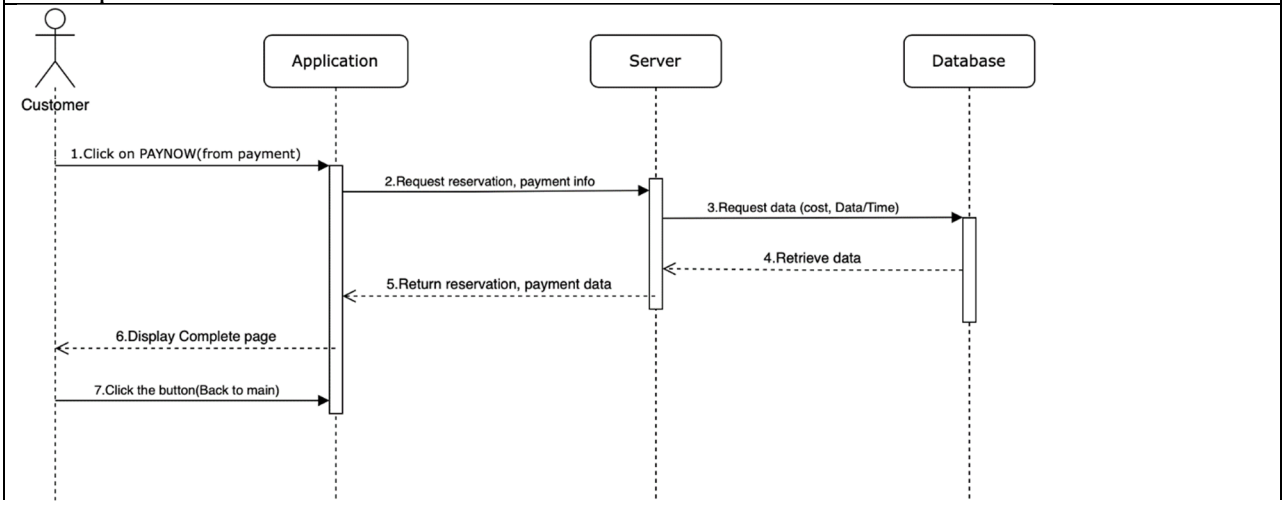
<Confirmation>



<Payment>

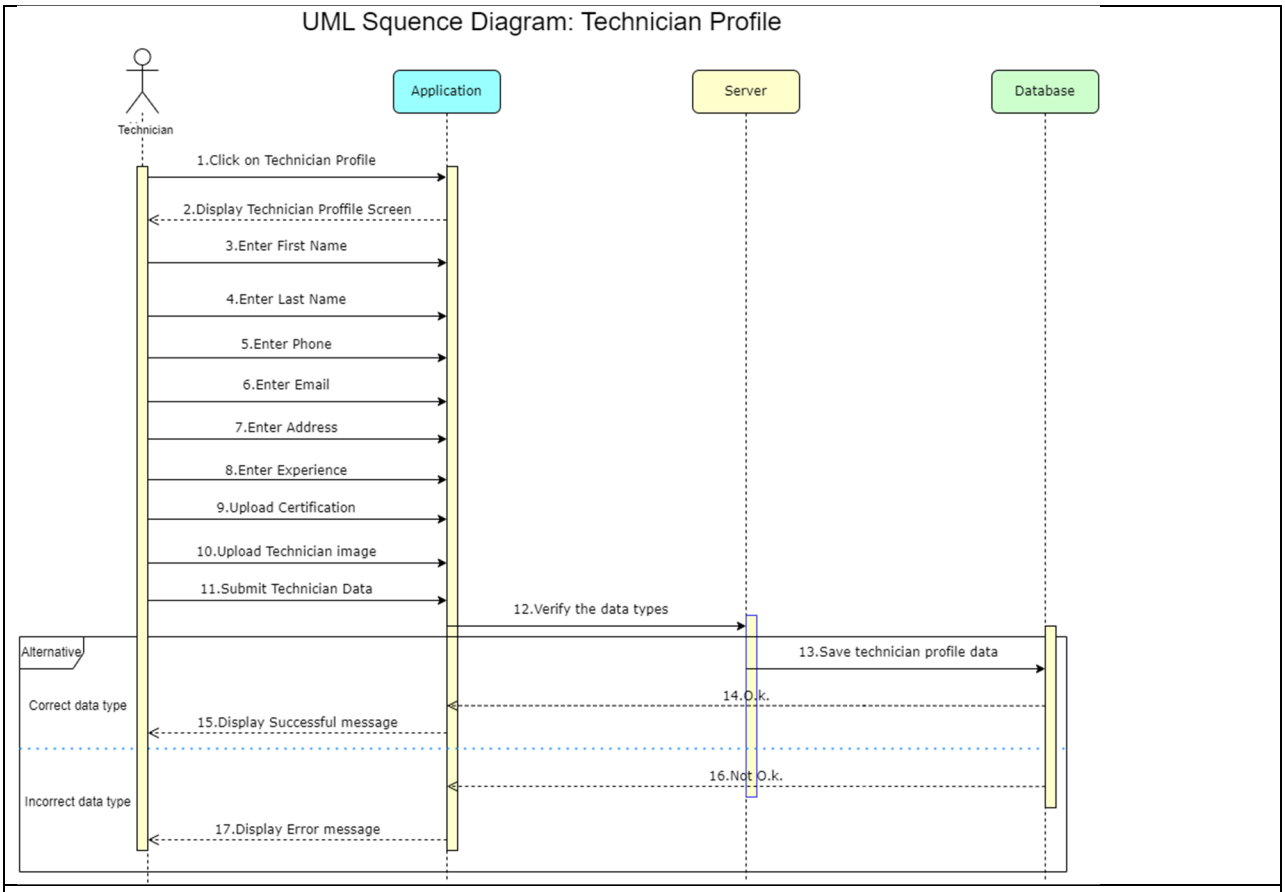


<Complete>

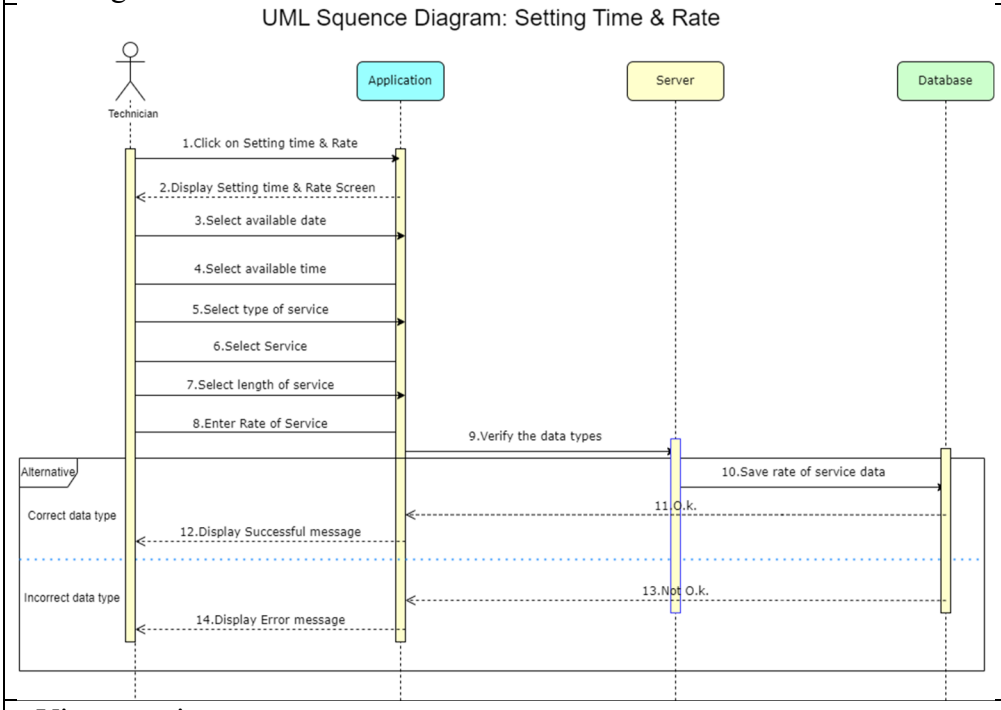


Technician View

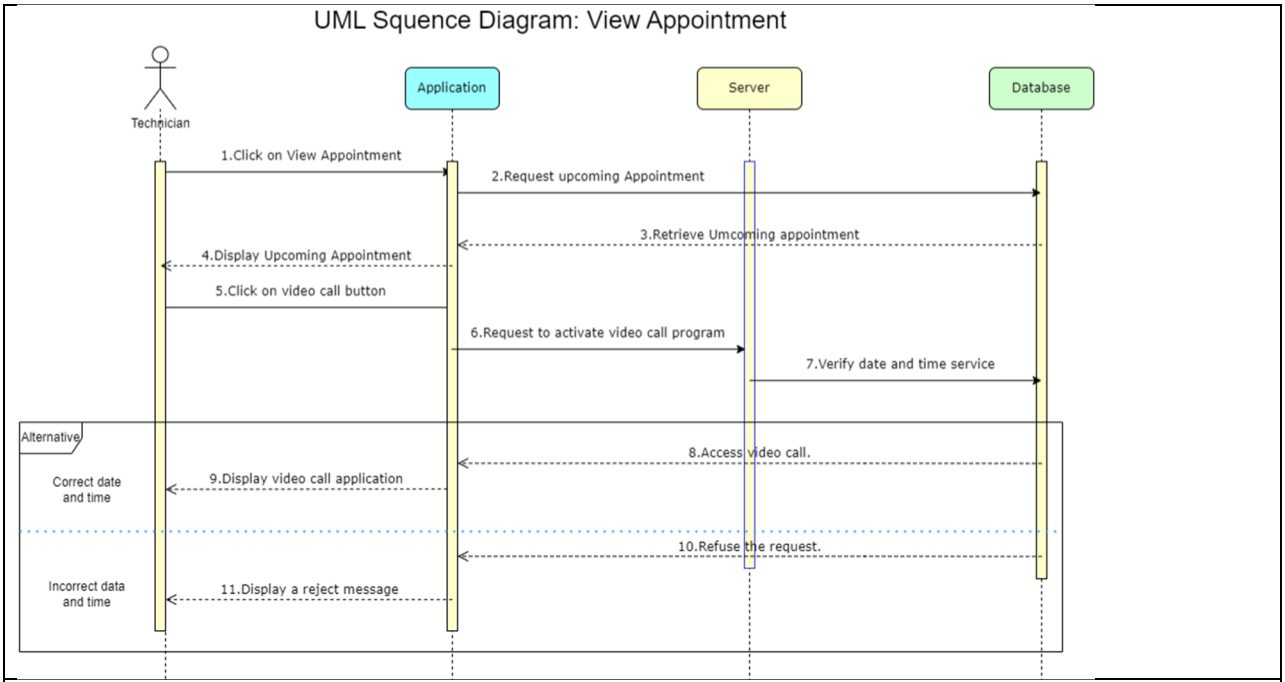
<Technician Profile>



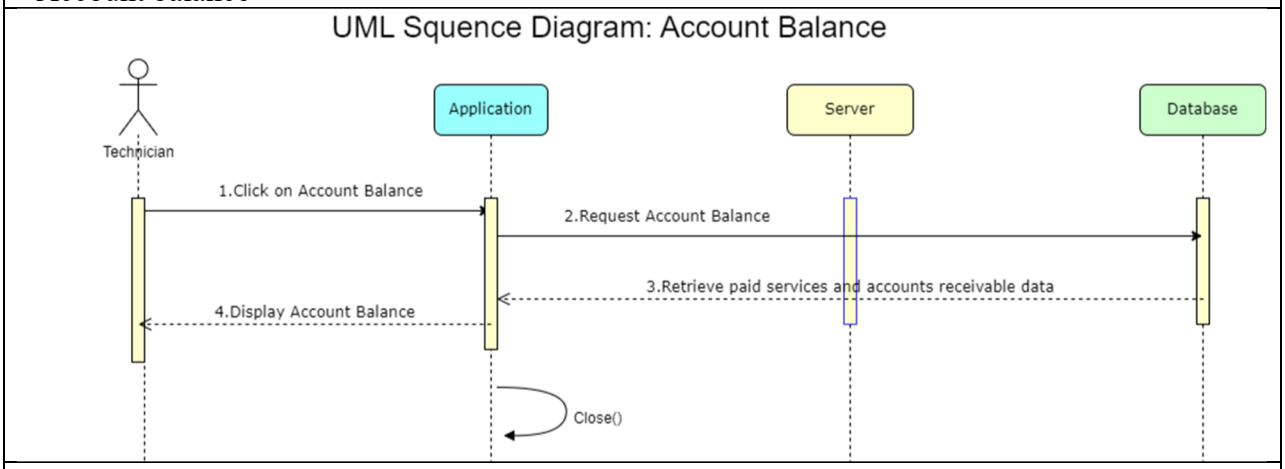
<Setting time and rate>



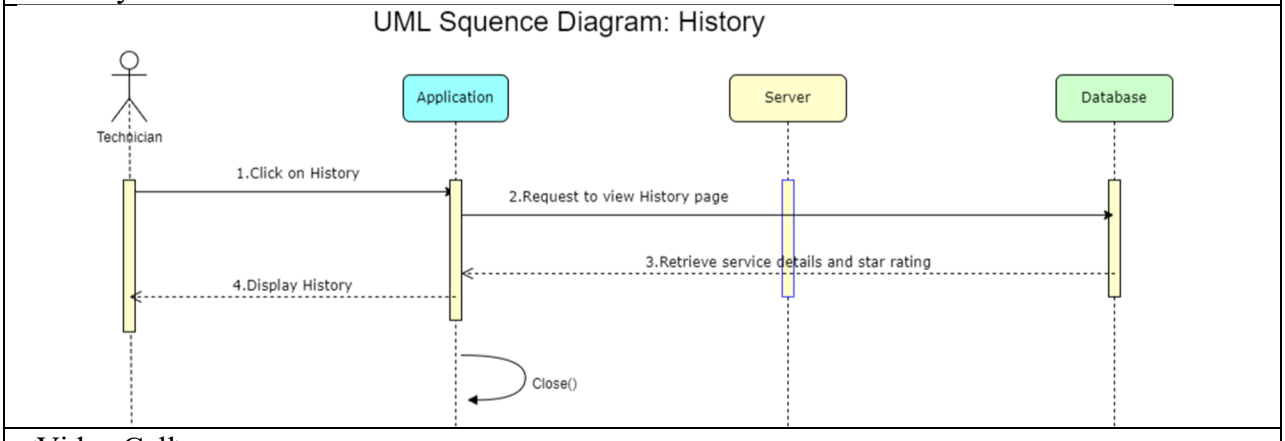
<View appointment>



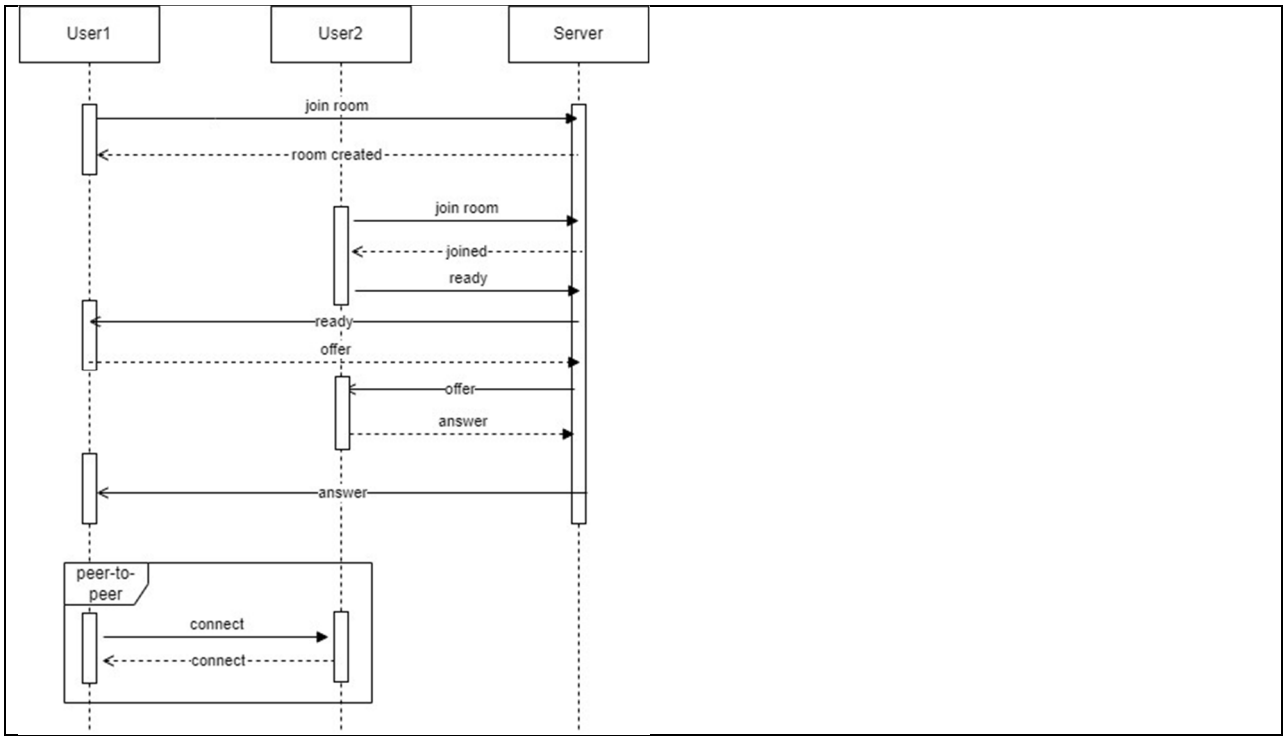
<Account balance>



<History>



<Video Call>



V. Revenue Model:

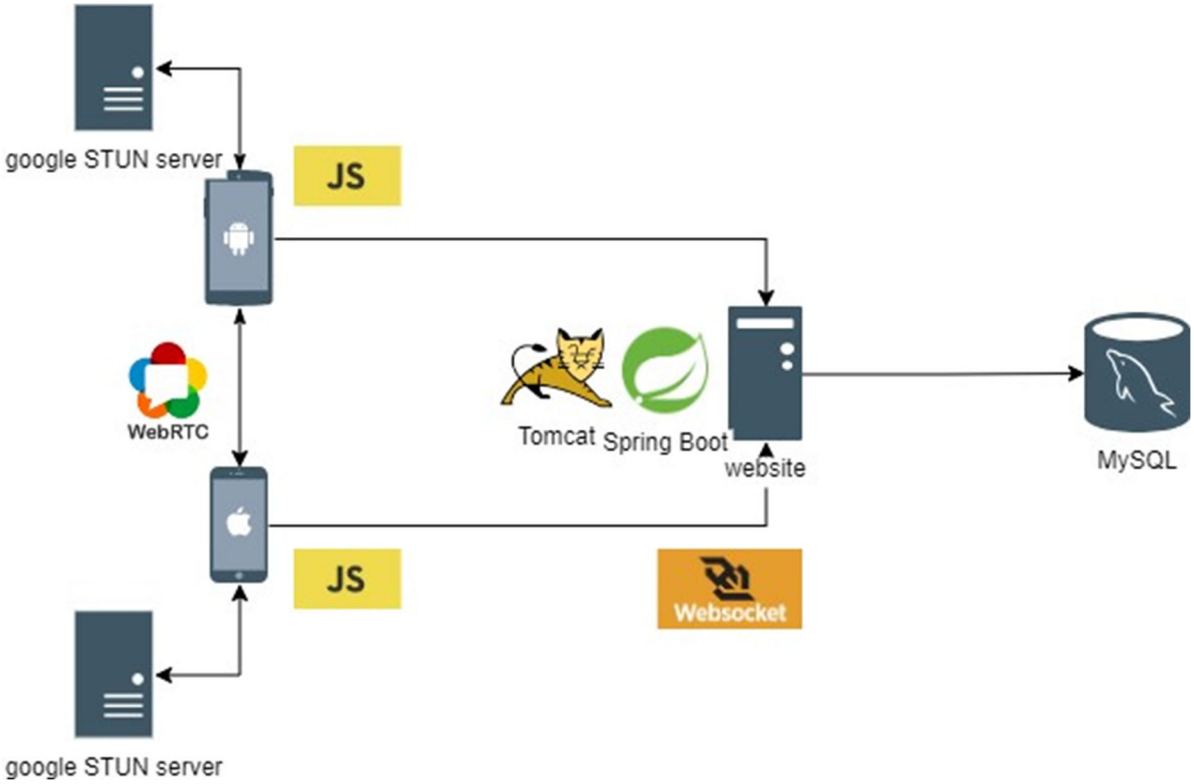
The company gets a profit from 10% of the payment the customer paid. This revenue can be its profit. The customer pays service fees to the company. Then it subtracts 10% of service fees from total as the brokerage fee. In the end, the company pays rewards to the technician. We keep in mind that both customers and technicians could take advantage of money by using this service. So, we set up a fixed brokerage fee of 10% and provide them with clear transactions. And to reduce the cost of maintaining our web application, we focus on developing it for web browsers.

3. Design

I. Design Goals:

The system will provide a lot of customers and technicians with online technical assistant with no bound by place. It will be designed for mobile device friendly to achieve this goal. The advantages of a mobile device are mobility and popularity. The system will make use of them and have a focused function, which is video calling between a customer and a technician. By doing that, the system will be able to get performance optimization and maintainability. The system also will be provided through web browser, so it will be Operation System independent. Many customers and technicians don't need to worry about their mobile device type.

II. The architecture of the system:



- Technology Stack:
  - Frontend:
    - Vanilla JavaScript for dynamic user interfaces.
    - WebSocket for real-time communication.
  - Backend:
    - Spring Boot framework for server-side logic.
    - WebSocket integration for peer-to-peer connection.

        - Communication Protocol:
            - WebRTC for efficient peer-to-peer video calls.
            - WebSocket for real-time communication between devices.
        - Infrastructure Components:
            - Hosting: Docker hosting
            - Database: MySQL for secure and structured data storage.
            - STUN Servers: Utilizing Google's STUN servers for obtaining public IP and device information.

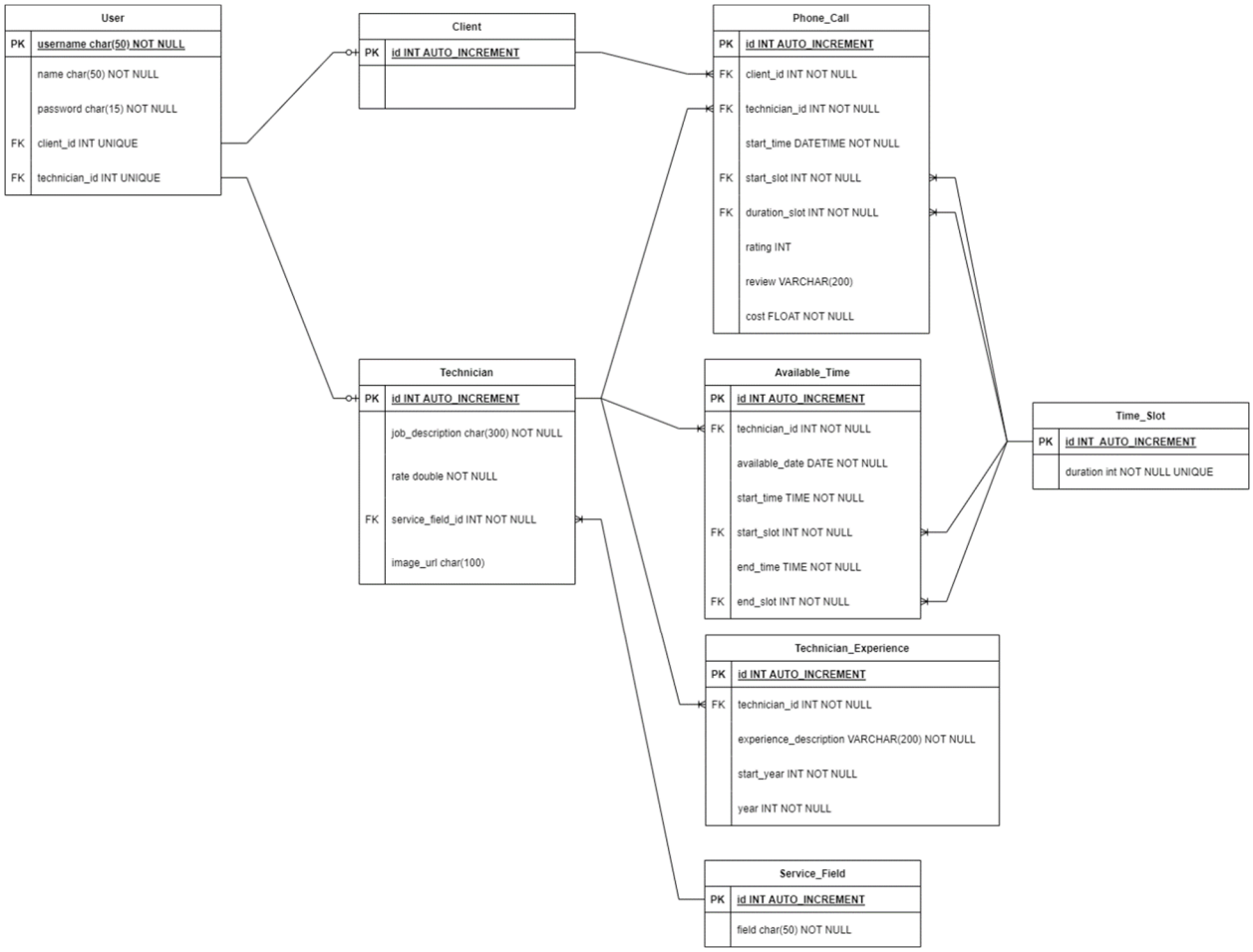
- System Interaction:
    - User devices connect to the website and exchange information using WebSocket.
    - WebSocket facilitates communication between devices to establish a direct peer-to-peer connection.
    - WebRTC enables efficient video calls over the established connection.
    - Information about IPs obtained by using Socket.IO.
- Infrastructure:
    - Hosting Transition: Individual computers with Docker
    - Scalability Planning: Cloud platform scalability for handling multiple connections simultaneously.
    - Resource Optimization: Using containers to host the website, optimizing resource utilization.

III. The hardware and software configuration of the system:

- Server Configuration:
    - Hosting Environment: Initially on free resources or individual computers, later on cloud platforms.
    - Cloud Platform: Azure for efficient scaling and resource management.
    - Containerization: Utilizing containers for hosting the website, enhancing resource optimization.
    - Scalability Measures: Scaling resources dynamically to handle increased connections.
- Client-Side Configuration:
    - Device Compatibility: Ensuring compatibility with various devices for a seamless user experience.
    - Browser Support: Full compatibility with major browsers for widespread accessibility.
- Security Measures:
    - WebRTC Security: Implementing secure peer-to-peer connections through WebRTC.

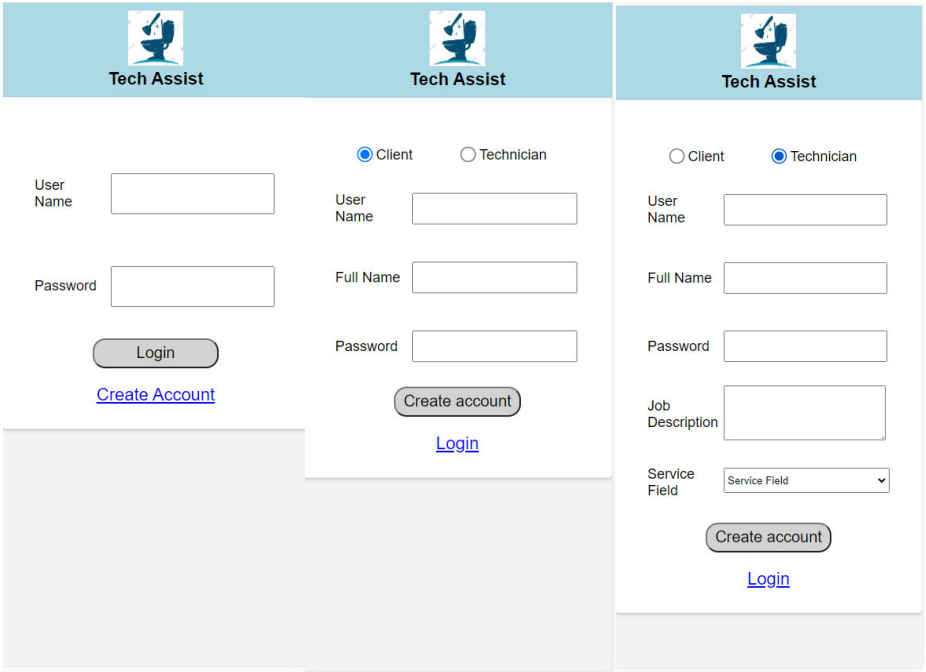
        - Data Encryption: Industry-standard encryption protocols for secure data transmission.
        - User Authentication: Secure authentication mechanisms for user accounts.
    - Database Configuration:
        - Database Type: MySQL, an open-source and relational database.
        - Scaling Consideration: Current design doesn't necessitate database scaling.
        - Future Optimization: CDN consideration for decreased latency in the future.
    - Documentation and Compliance:
        - Technical Documentation: Comprehensive documentation of the system's architecture.
        - Regulatory Compliance: Adherence to data protection and privacy regulations.

IV. The database design:

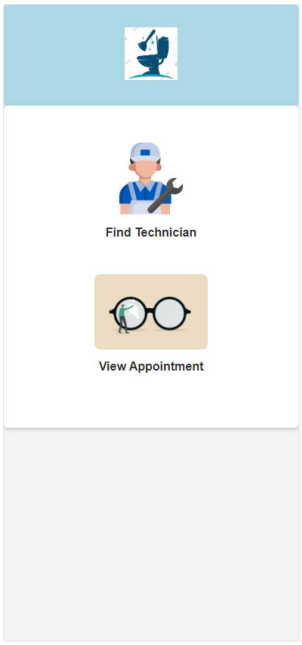


V. The Interface design:

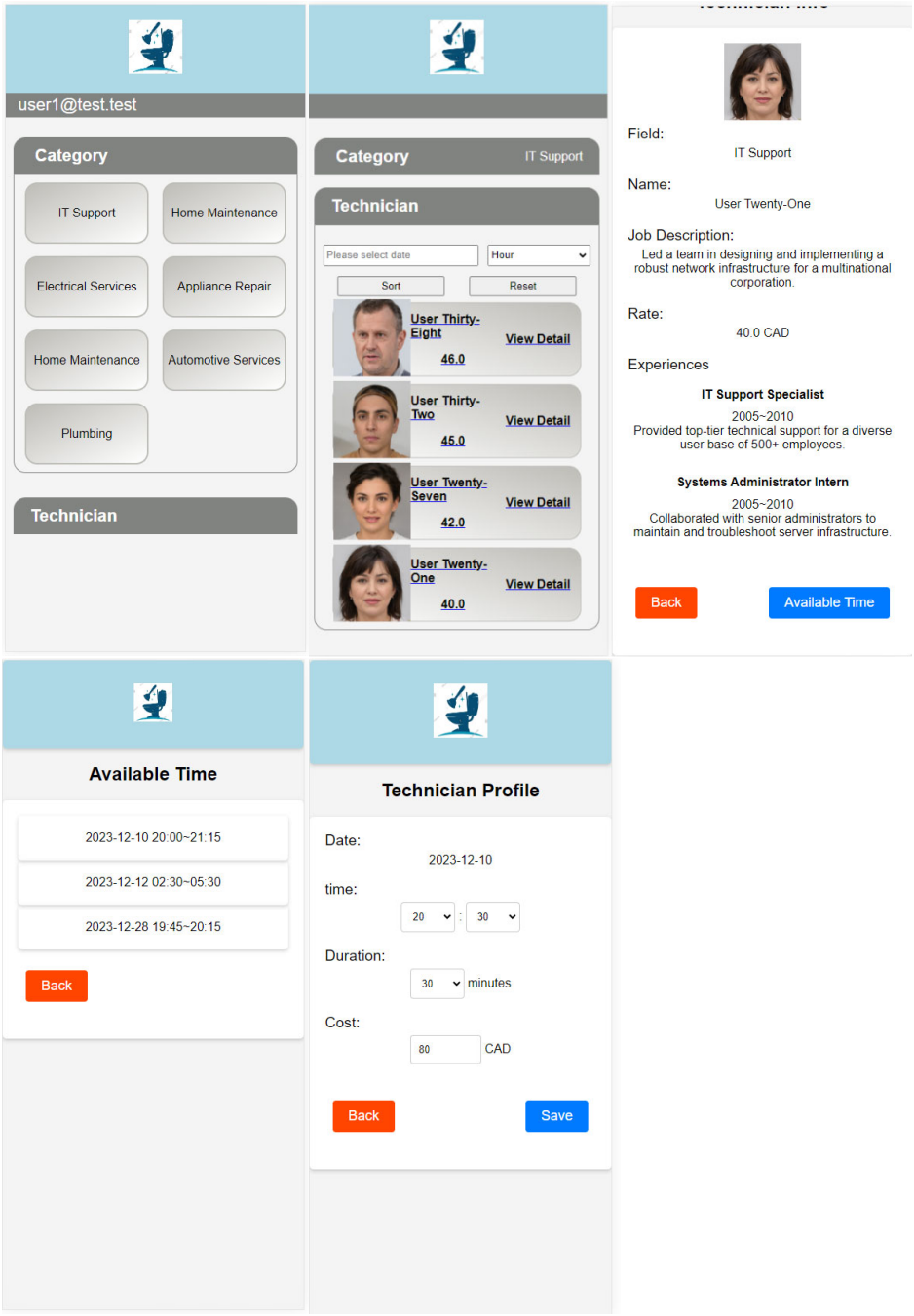
1. Login and register



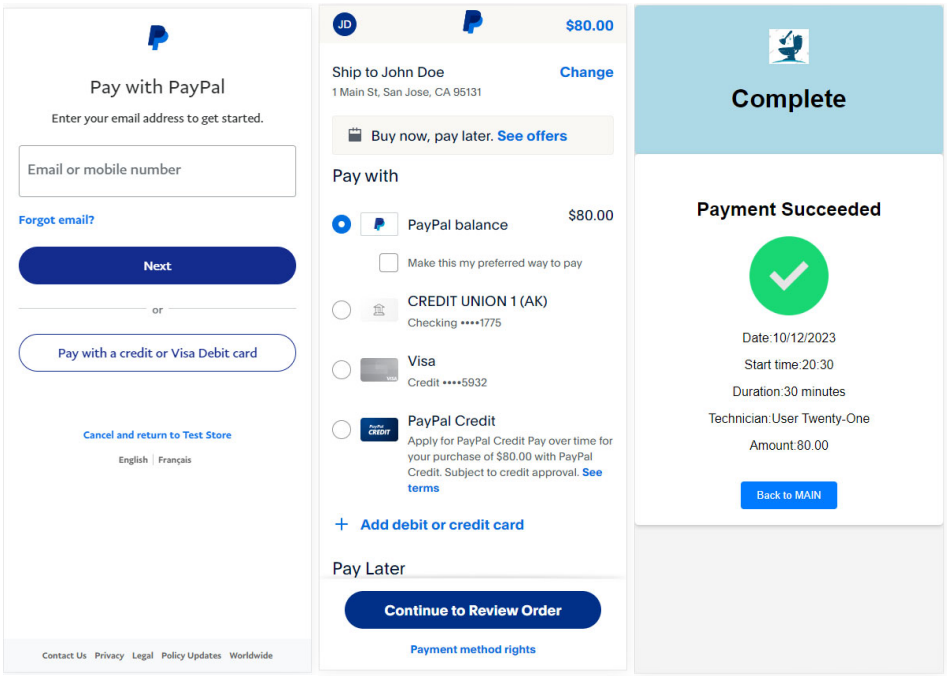
2. Client - Home



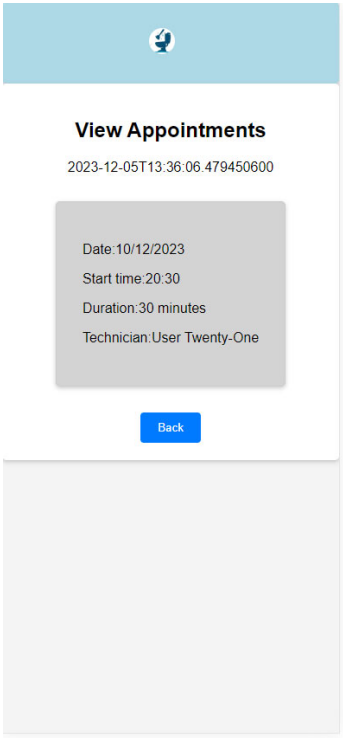
3. Client – Appointment process



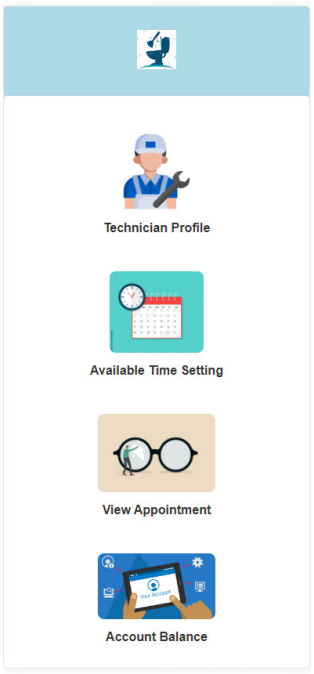
4. Payment Process



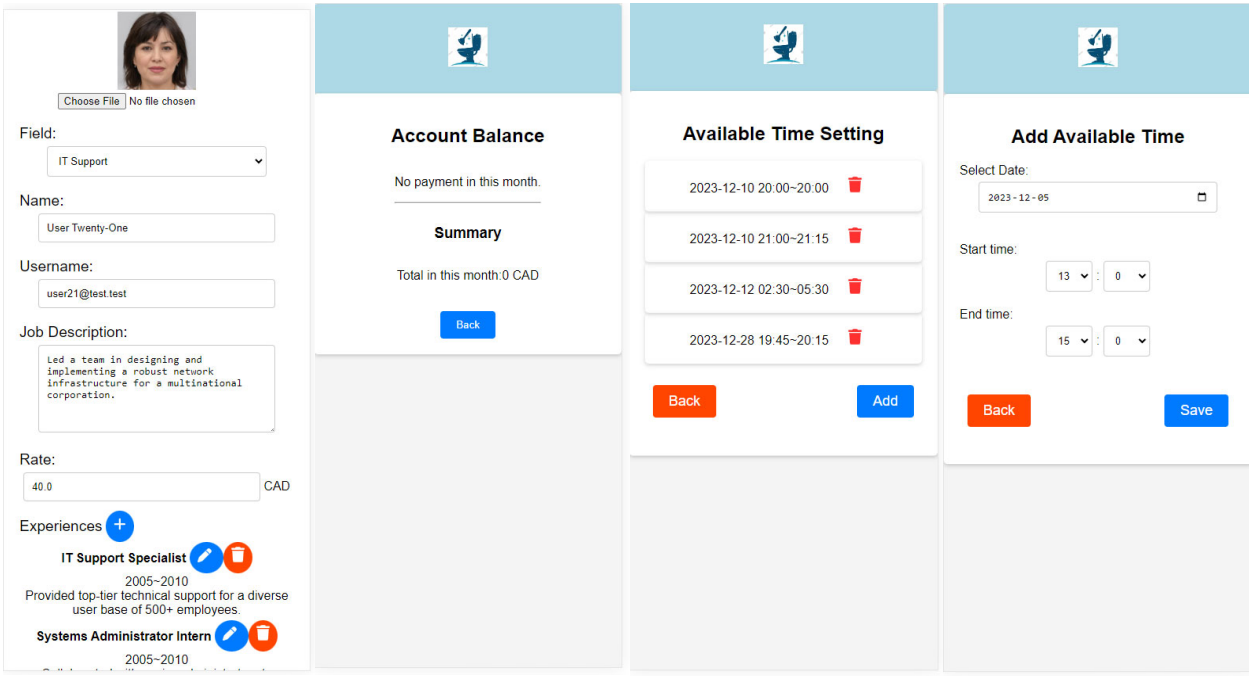
5. Client – View appointments



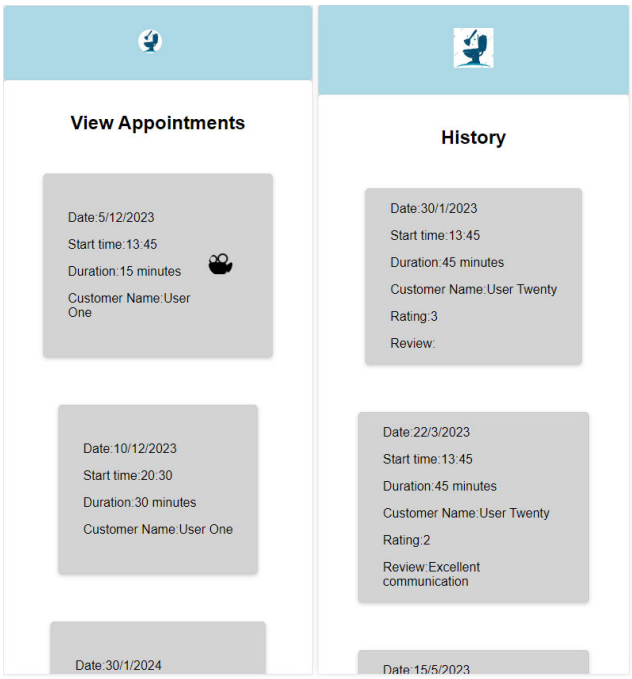
6. Technician – Home



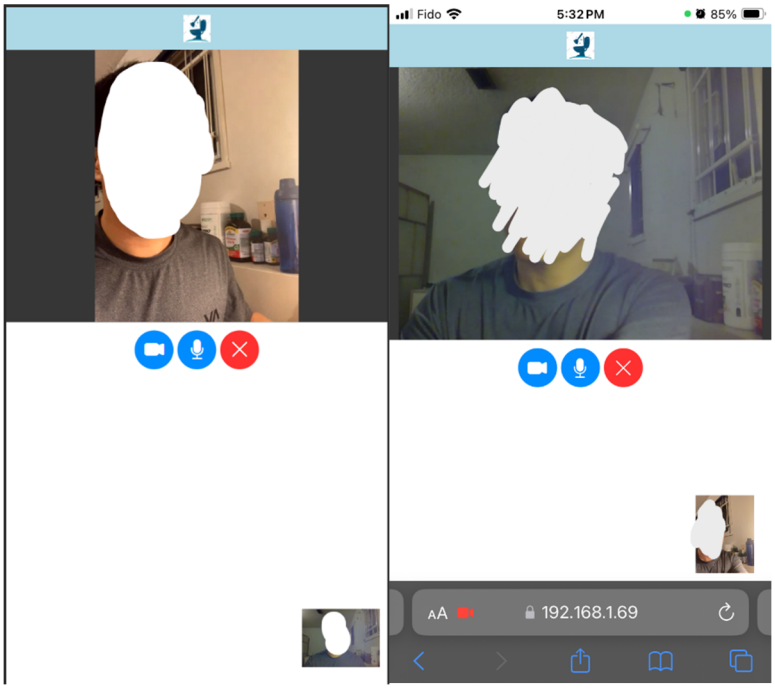
7. Technician – Profile, Account balance, Add available time.



8. Technician – View Appointments, History



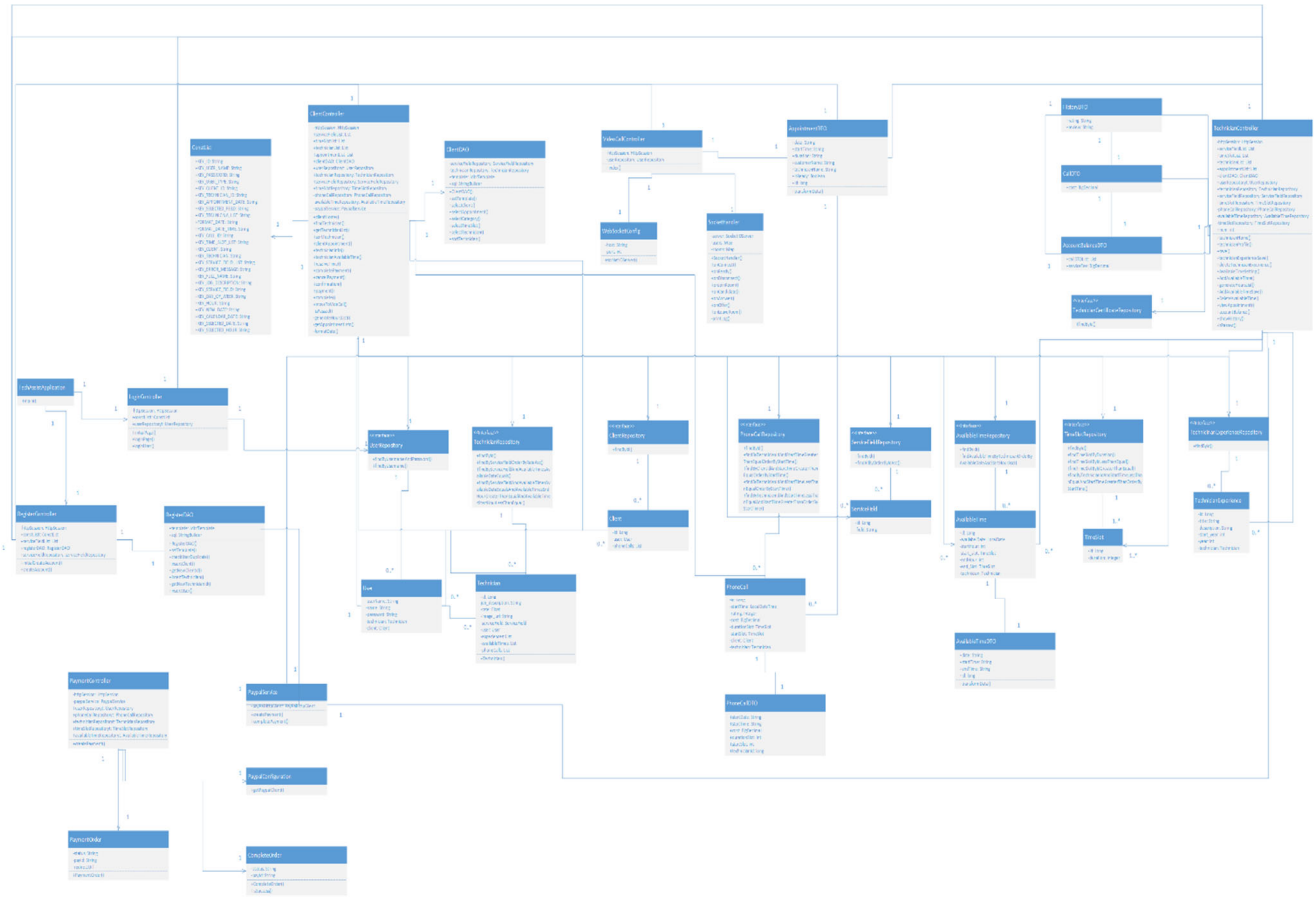
9. **Video Calling**



VI. A detail description on how the non-functional requirement will be achieved:

Nonfunctional requirements are performance, provided on web browser, video calling function works on latest mobile OS First, in terms of performance, focus on video calling and optimize the system for that, we can except good performance. Then, we will pay attention to the browser condition, it would include the older version of browser. Especially, Google chrome, Microsoft Edge, Firefox, and safari, which have a major share in the market, we will test the system on these browsers. Video calling function will make use of default components on the device, so we will make sure it works well in a variety of devices.

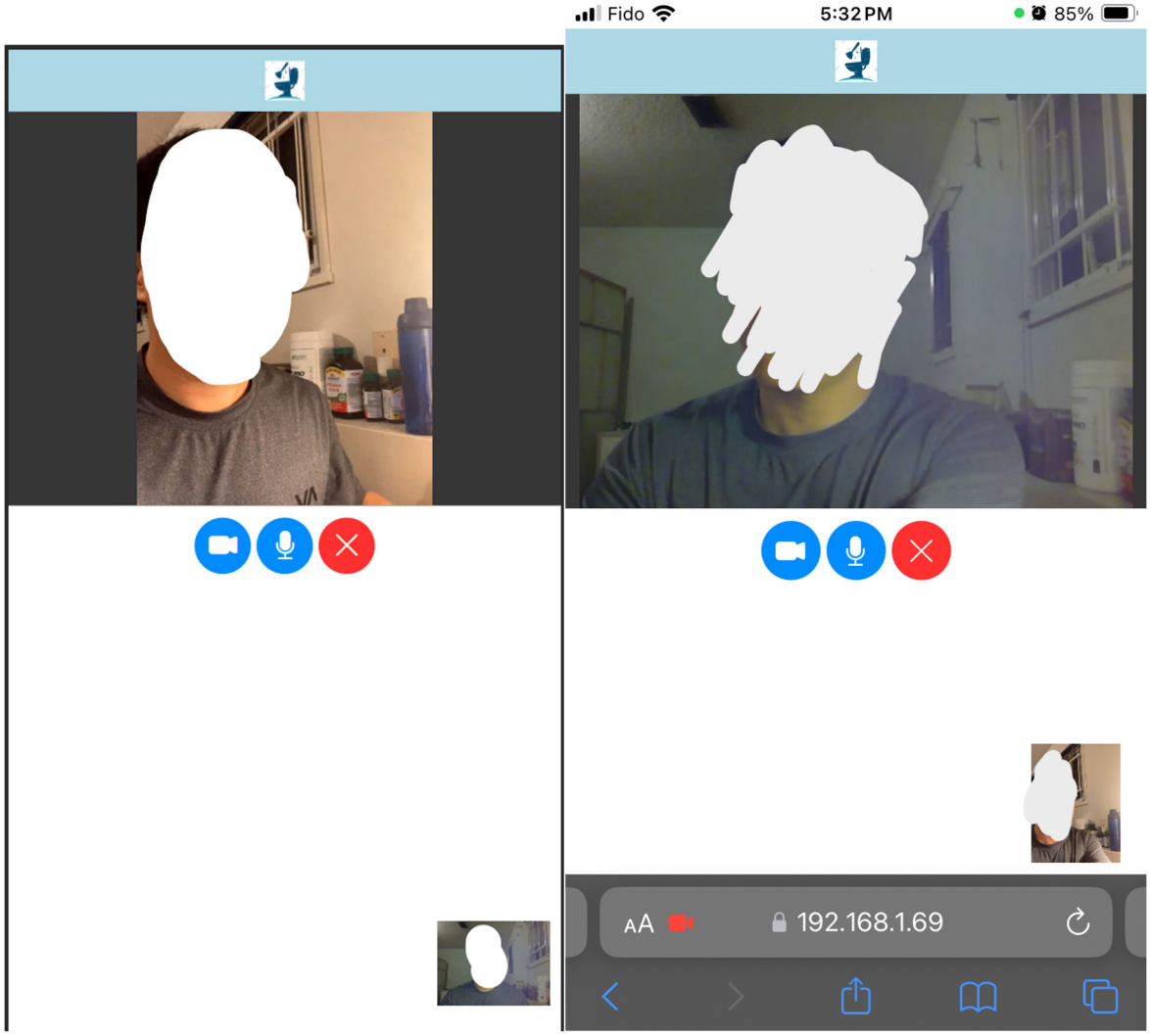
VII. UML diagrams:



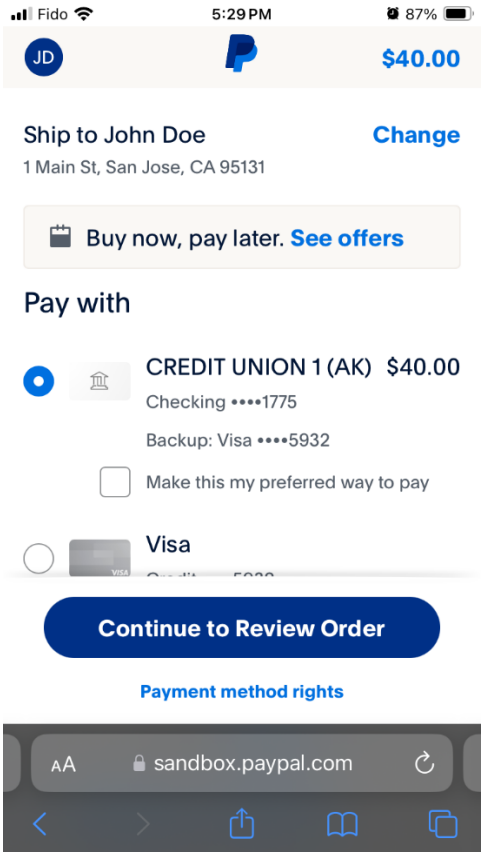
4. Implementation

I. Coding

The code of the project is on GitHub: https://github.com/sungfuchiu/3275GroupProject
Using this GitHub repository to implement video calling function:
https://github.com/gurkanucar/spring-boot-webrtc-peer2peer



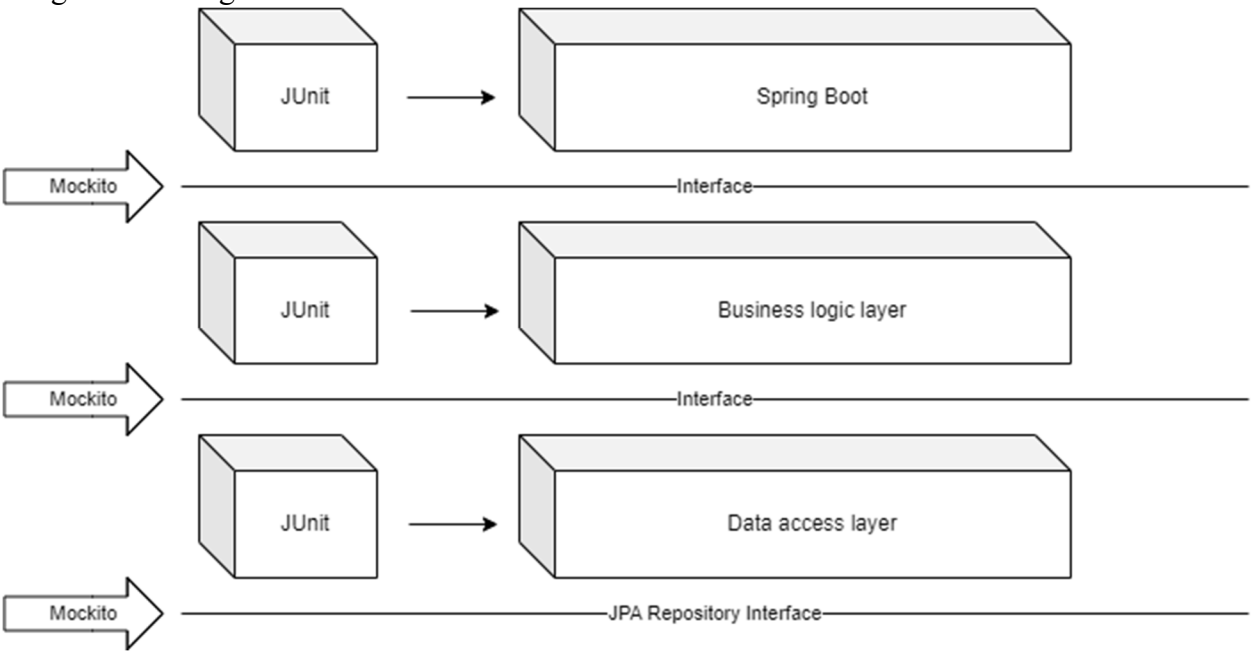
Using PayPal sandbox to demonstrate payment functionality:
https://developer.paypal.com/tools/sandbox/



II. Testing Designs

Our testing design involves the division of our application into three distinct layers. Spring Boot will be responsible for managing flow control, while a dedicated layer will handle business logic, and the final layer will manage data access through interaction with the JPA repository. Each layer will communicate with interfaces from other layers, enabling the injection of dummy objects into these interfaces. For the purpose of unit testing, we will leverage JUnit and Mockito to craft our test code. Once individual units are tested successfully, we will proceed with

integration testing for each use case.



III. Deployment (Windows)

1. Deployment on local

The deployment process presents a significant challenge due to the integration of a video calling function. To enable this feature, a secure HTTPS connection is imperative, as modern browsers restrict web applications from accessing the web camera without it. Considering the ongoing testing phase, acquiring a commercial SSL certificate is not desirable. Instead, a self-signed certificate is created, and the IP is set to the local computer's IP. Additionally, the implementation involves utilizing socketIO to facilitate connection between two devices, necessitating knowledge of the server's IP. The application employs WebRPC for video calling, offering various approaches for peer-to-peer connections. Opting for the simplest method, the application supports video calls only when both devices connect to the same smaller WIFI, such as a personal hotspot on a mobile phone.

A. Get local IP

We can get our IP by calling ipconfig in command line.

B. Get SSL certificate

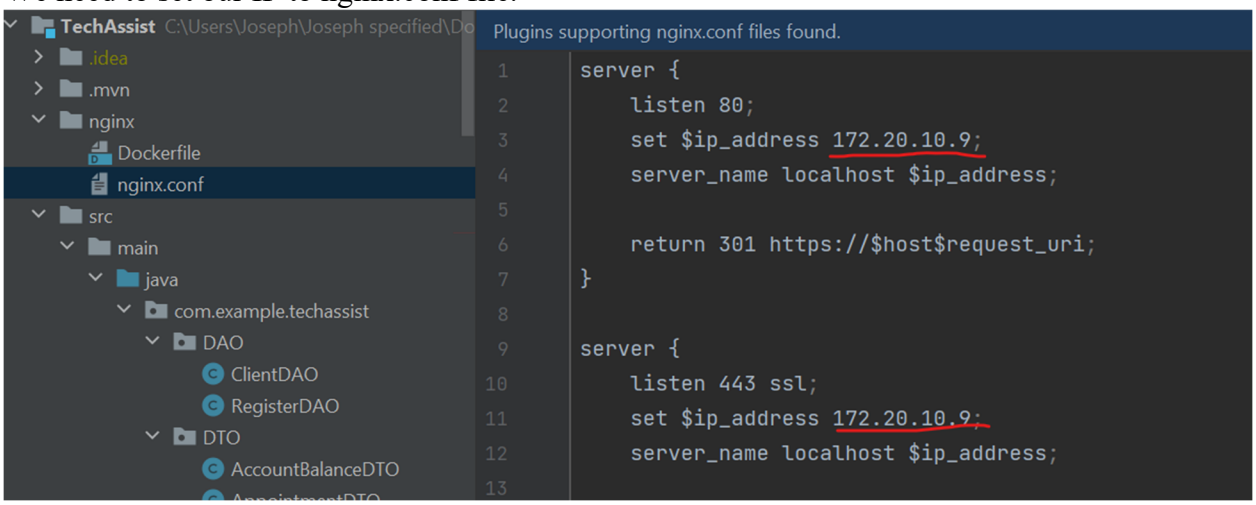
We can use OpenSSL downloaded with Git and using following command to create certificate.

```
mkdir ssl && C:\Program Files\Git\usr\bin\openssl.exe req -x509 -nodes -days 365 -newkey rsa:2048 -keyout ssl/private_key.pem -out ssl/certificate.pem -subj
```

```
"//C=US//ST=California//L=San Francisco//O=MyOrganization//OU=MyDepartment//CN=<YOUR_LOCAL_IP>"
```

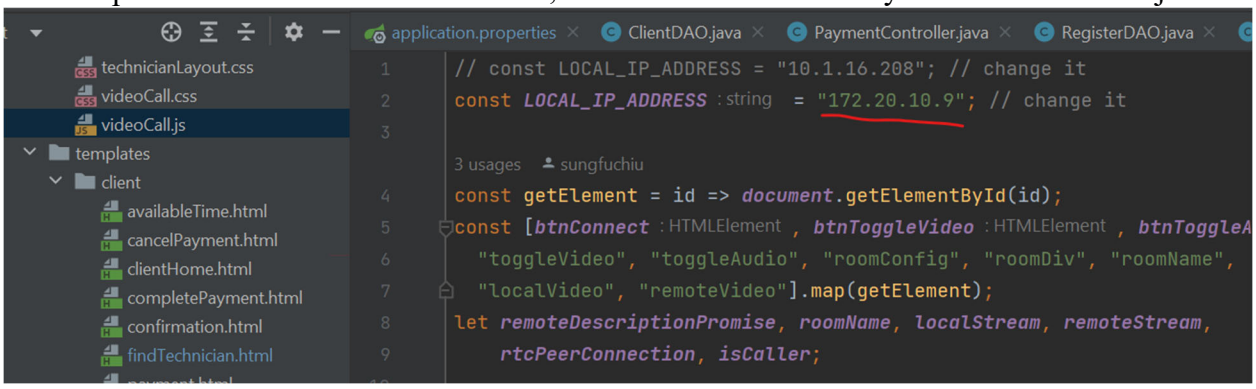
C. Set IP for nginx configuration

We need to set our certificate for users to create HTTPS connection with out website and we need to use socketIO to help two devices connect with each other so we host nginx.
We need to set our IP to nginx.conf file.



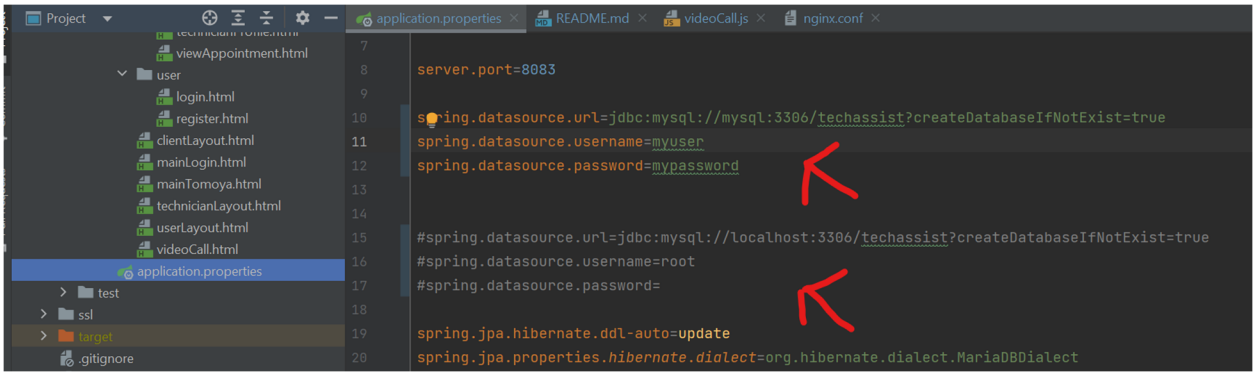
D. Set IP for socketIO in JS

In order to connect to web servers from client's device through socketIO, we need to use JavaScript to communicate with the server, so we also need to modify the IP in videoCall.js.



E. Application properties settings

Comment out the original data source which is used for local environment, and uncomment the data source below it which is used to connect to MySQL in docker.



F. Build the project in docker

The last step is building the application in docker.

```
docker-compose up -d --build
```

While the application finishes building, we can connect to the application through the URL https://<YOUR_LOCAL_IP>/login.

G. Insert data into database

We didn't add dummy data into MySQL through JPA. We insert data into MySQL through SQL scripts, so we need to connect to MySQL in Docker with the following command.

```
mysql -P 8083 --protocol=tcp -u root -p
```

Then, we can key in the password root while the cmd quotes. When we get into the database, we can execute create_database.sql in \SQLscript folder.

2. Deployment on cloud

Deployment on the cloud follows a similar process, given the use of Docker. However, it requires the purchase of a domain name and SSL. Additionally, ensure completion of WebRTC and socket IO configurations for cloud deployment.