Appointment Scheduling Application

CO 328 - Software Engineering

[Appointment scheduling software - Github Repository]

Group 03
E/16/332 - A.A.R.V. Samaraweera
E/16/364 - T.T.N. Suwaris
E/16/377 - I.B.S. Vindula

Introduction & Problem Definition

Many websites and mobile applications create appointments for doctors and government services like driver's license, passport, etc. But small business owners will face many difficulties if they build a platform on their own and manage payments. Customers will not trust a platform if they don't know much about it. Also, customers don't like to book appointments on several platforms. Registering for different platforms wastes time. Several appointments are going on using phone calls and excel sheets or notebooks. This can cause mess and also missing appointments. Also, it is challenging to let customers know if you are going to cancel any scheduled appointment. Also, customers are facing difficulties in letting them know that they are unable to attend. And also, most platforms don't have a way to review their experience with the business owner. It is challenging to collect a database of our customers with their details for marketing purposes. With the ongoing covid-19 situation, people are looking for ways not to gather in places. So, taking appointments before attending is much more critical than ever.

Current applications do not have effective ways to overcome all of these problems. The proposed solution is to develop a mobile app to solve all these problems. This app will enable both business people and customers to ease their work. Any kind of business will be able to use this. First it is planned to build this application for use in this company. Later it will be out to the public with a marketing plan.

Requirement Analysis

Functional Requirements

The appointment scheduling software contains eight subsystems, and for each subsystem, a different functional specification describes each subsystem's functionality. In software engineering, functional requirement is defined as a function of a system, or its component, where the function is a specification of behaviour between inputs and outputs.

Functional Requirements		
Sub-system	**Functions**	**Description**
Application Programming Interface (API) for the database	The interface between the appointment scheduling software and the database	This subsystem handles the data transactions and routes for the data between the database and the application
Appointment Handling subsystem	Getting the Appointments	Get the appointments from users according to the available times and dates and provide them with a bill or report according to the appointment.
	Time management	Manage time accurately and synchronously in order to hold against a high traffic situation.
	Business owners' portal	Getting data such as available dates, times, necessary things from the business owners.
Alerting subsystem	Notify the users about cancellation and updates	If the appointments are cancelled, business owners cannot participate in the scheduled time to broadcast messages to every client. Also, if they want to give an update like bring something, they can notify the clients.

Table 01: Functional requirements for NoFra Appointment Scheduling App

Non Functional Requirements

Non-functional requirements to the requirements that are not directly related to the system functionalities. But they are essential for higher performance, including system properties and constraints. Non-functional requirements place constraints on "How should the software system fulfil the functional requirements?". Also, these requirements are not mandatory, and also, they are also applied to the system as a whole. These are

represented by the letters "URPs+" in the FURPS+. Given below shows the non-functional requirements for an appointment scheduling application. These non-functional requirements will affect the success of the App as they affect the quality. It is necessary to update more non-functional requirements along with the usage by clients to maintain an up-to-date application.

Non-functional Requirements	
FURPS+ Category	**Requirement Description**
Usability	• The user interface should be user friendly and easily accessible • Two-factor authorizations should be implemented when login, • The user interface must be simple but with every option. • Should focus on branding of every business. • Application privacy policy should be shown when entering the App for the first time and in the about section. • User manual, tutorial videos and FAQs should be there in the help section.
Reliability	• Should select a hosting service where uptime is greater (99.9%) to be available 24/7. • The system must be backed up daily, at least on two different servers. • Error log must be added to every warning and error. • The recovery plan should be there for every single failure.
Performance	• The system should be fast and work on any kind of traffic and with any number of users. • Response time should be minimum and should be less than 0.1seconds. • The server speed should be running within 150-200 milliseconds. • The system should support at least 10,000 users concurrently.
Security	• All traffic must be encrypted using an up-to-date protocol such as HTTPS. • Two-factor login and tokens should be used to maximizing security. • System and software components must be safe from known vulnerabilities such as SQL injections etc. • All the passwords must be encrypted. • If the user account on the App remained untouched for a month user will have to log in again when he comes back to the App.
+Design constraint	• The App should be compatible with android 4.0 or never. • Both the API and database should be deployed on the cloud. • Web Server specifications should have at least 500GB SSD, 16GB RAM with eight cores.
+Implementation	• User Interface will be designed using flutter for mobile apps and VueJS for the website. • Backend will be coded using Golang while database will be designed in MongoDB

+Interface	● The users will be connected to the API using the internet, and through the API, they can receive and send data to the database.
+Physical	● Dedicated space in the data centres will be allocated for cloud servers.

Table 02: Non-functional requirements

Software Design

System Capabilities

This system will have the following capabilities.

Interface for Business Owner

- Can insert the business data, available time, and location for giving appointments.
- They are alerting customers if appointments are canceled.

Interface for Customers

- Can get appointments.
- Can check and put reviews.
- Can inform if the customer is unable to attend.

Below use case diagram (figure 1) shows the use case diagram of the appointment scheduling subsystem. A use case diagram creates a pictorial representation of the functional requirements by showing the actors use cases and how they interact.

There are two actors in the appointment scheduling subsystem as business owner and client. The lines show how the actors interact with the use cases by pointing out necessary participation between them. The figure also contains the use cases by pointing out essential involvement between them.

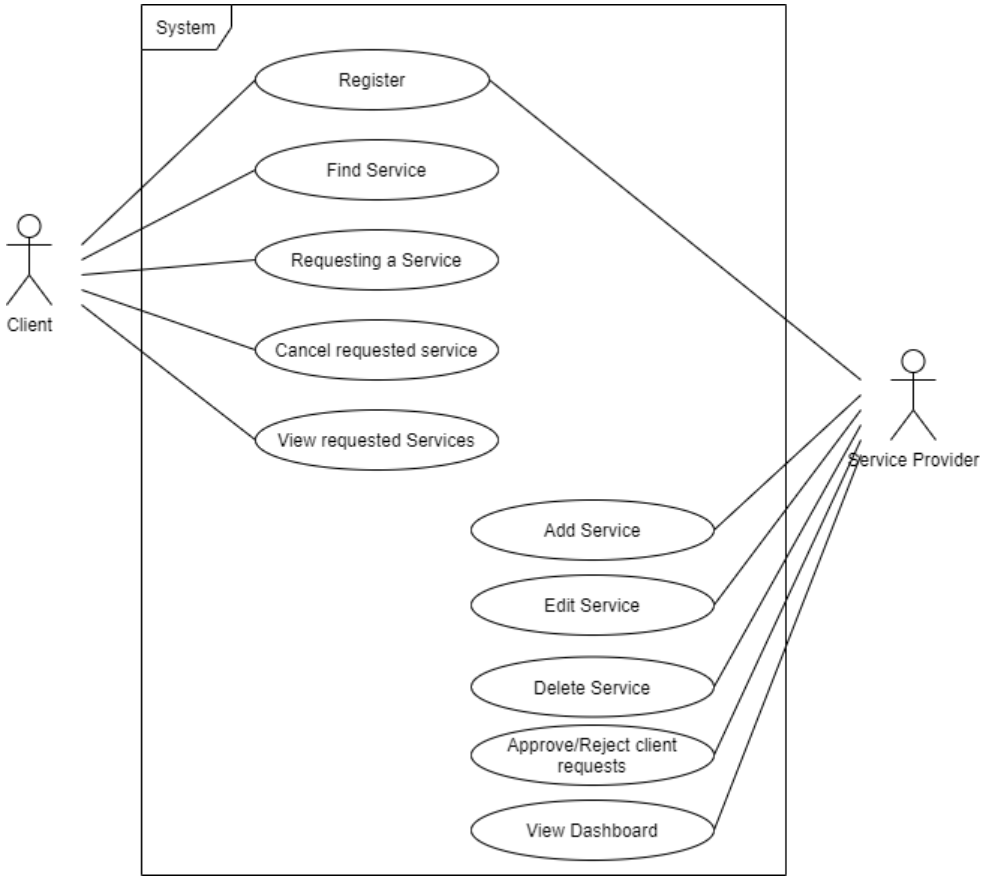


Figure 01: Use case Diagram

Use case descriptions to properly arrange and document the interaction between users and the system thoroughly. Table 4 demonstrates the described use cases for the Appointment Handling Subsystem.

After someone signs up for the first time, they can log in to the App. Then they can select their role whether they are a business owner or a client. Then both of the users will be taken into a dashboard according to their roles. Every transaction will be reviewed. Every subsystem has different use cases which are sometimes connected. We have to focus more on all of these use cases when developing.

Brief Use Case Description		
Use Case	Actors	Description
Search Business	Business clients	Clients can search for a business to place an appointment. They can click on the business to access the business. It will show the business details such as location, opening days etc.
Add new appointment	Business clients	Clients can add an appointment in a time slot. It will ask to fill in the details asked by business owners and show a confirmation message after placing an appointment.
Delete already placed appointment	Business clients	If the business owners allow this option, the clients can delete their already placed appointments if they cannot attend.
View Appointments	Business clients	Clients can view all of their currently placed appointments and check for details.
View free dates	Business clients	Clients will be able to view free dates and time slots through this.
Add dates and times	Business owners	Owners will be able to add dates and times they are planning to get appointments. These times and dates will be shown to clients as the available date and times.
View the current appointments	Business owners	Business owners can view currently placed appointments with their details through this.
Cancel scheduled time slot	Business owners	Owners will be able to cancel the scheduled times if they are unable to attend. Money will be refunded, and a cancellation message will be sent to the respective clients automatically.

Table 04: Brief Use Case Description

Class Diagram :

A class diagram is a diagram used in designing and modeling software to describe classes and their relationships. Classes in a class diagram correspond with classes in the source code. The diagram shows the names and attributes of the classes, connections between the classes, and sometimes also the methods of the classes.

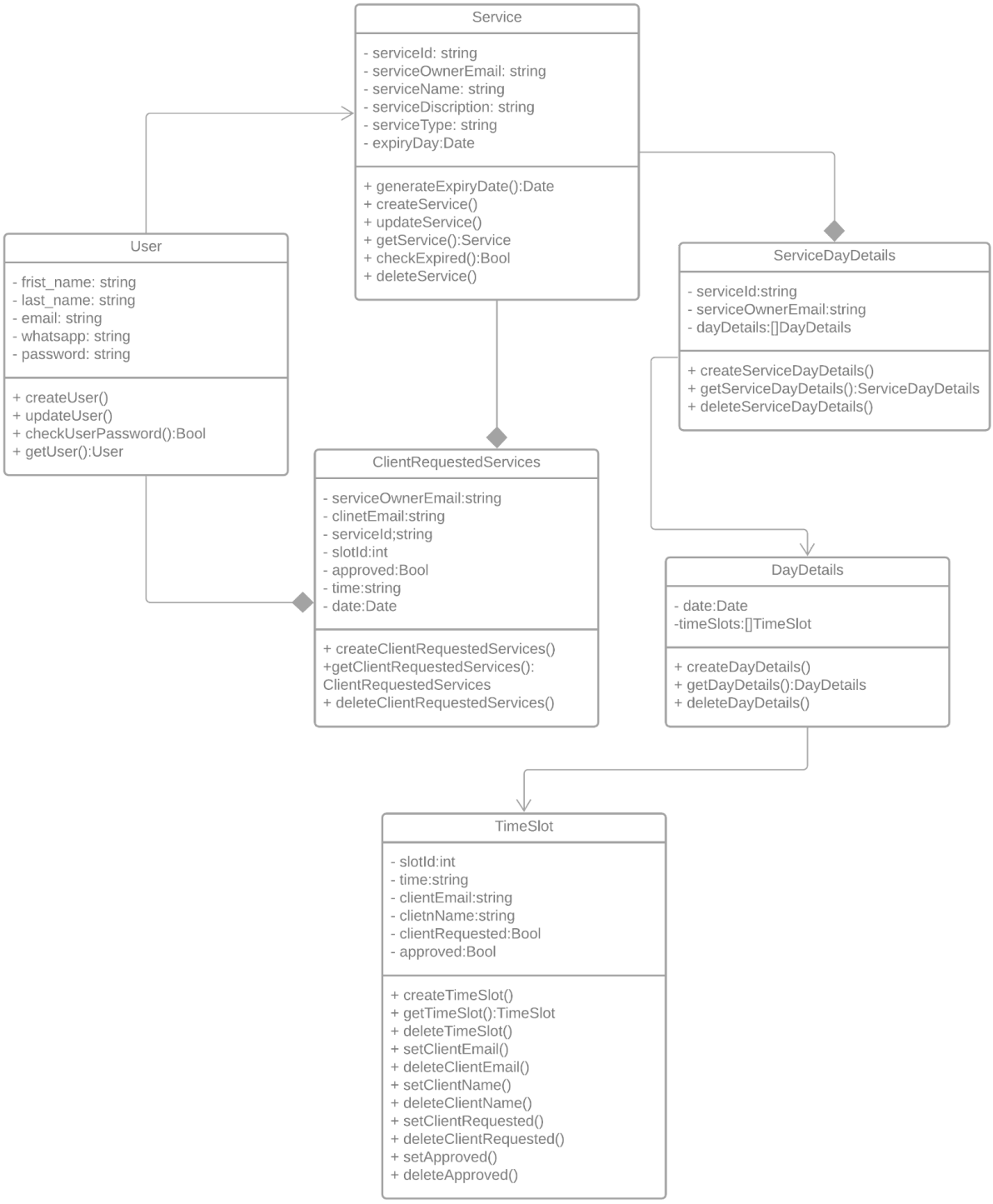


Figure 02 : Class Diagram

Website User Interfaces :

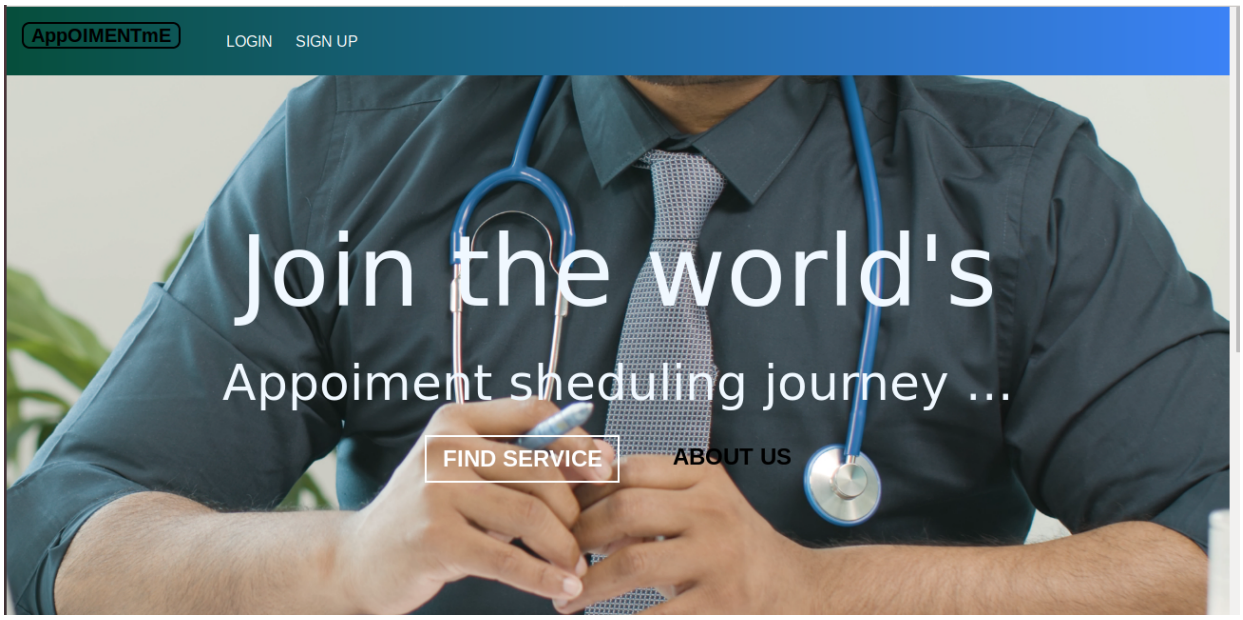


Figure 02 : Home Page

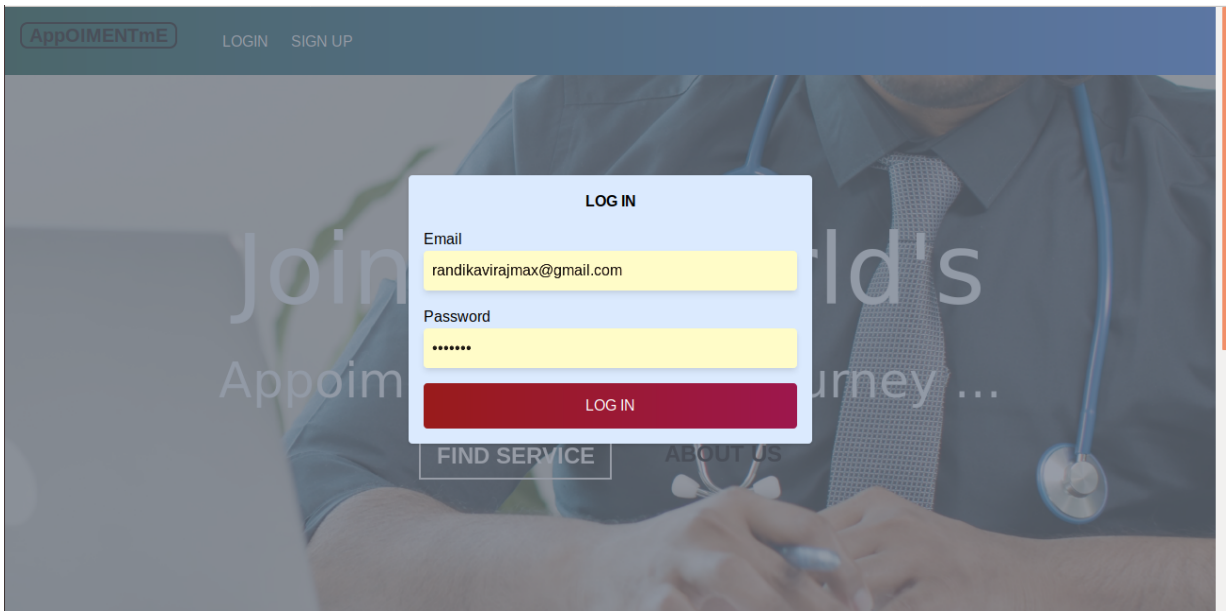


Figure 03 : Logging In

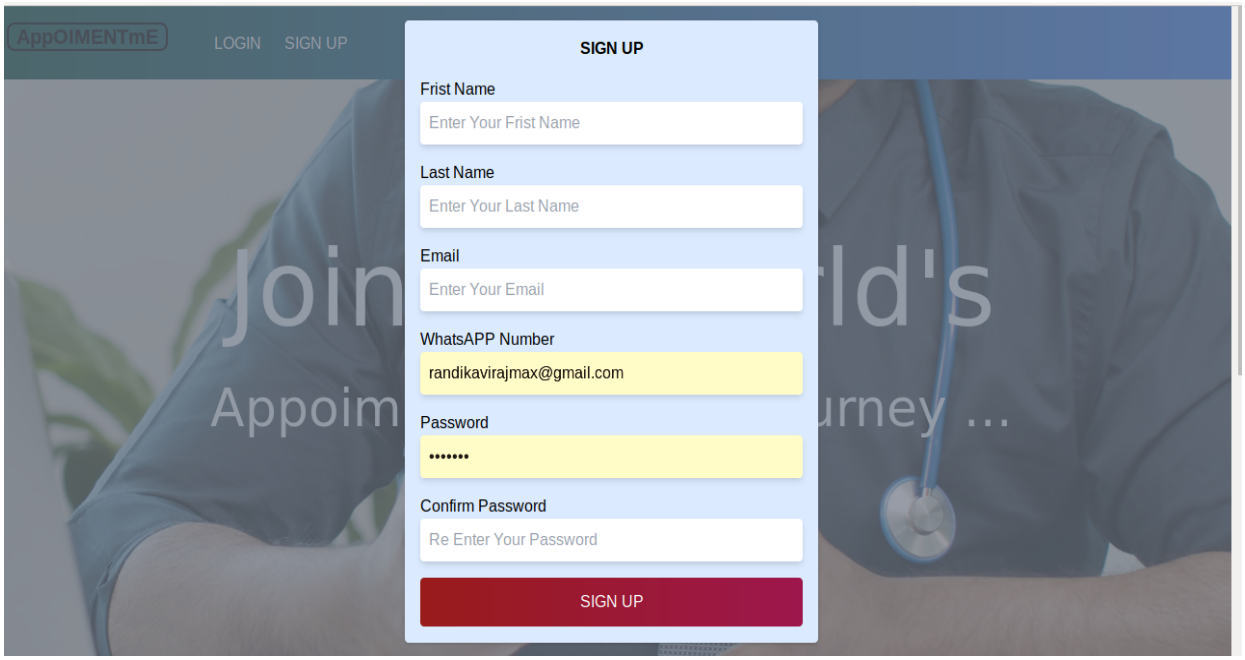


Figure 04 : Sign UP

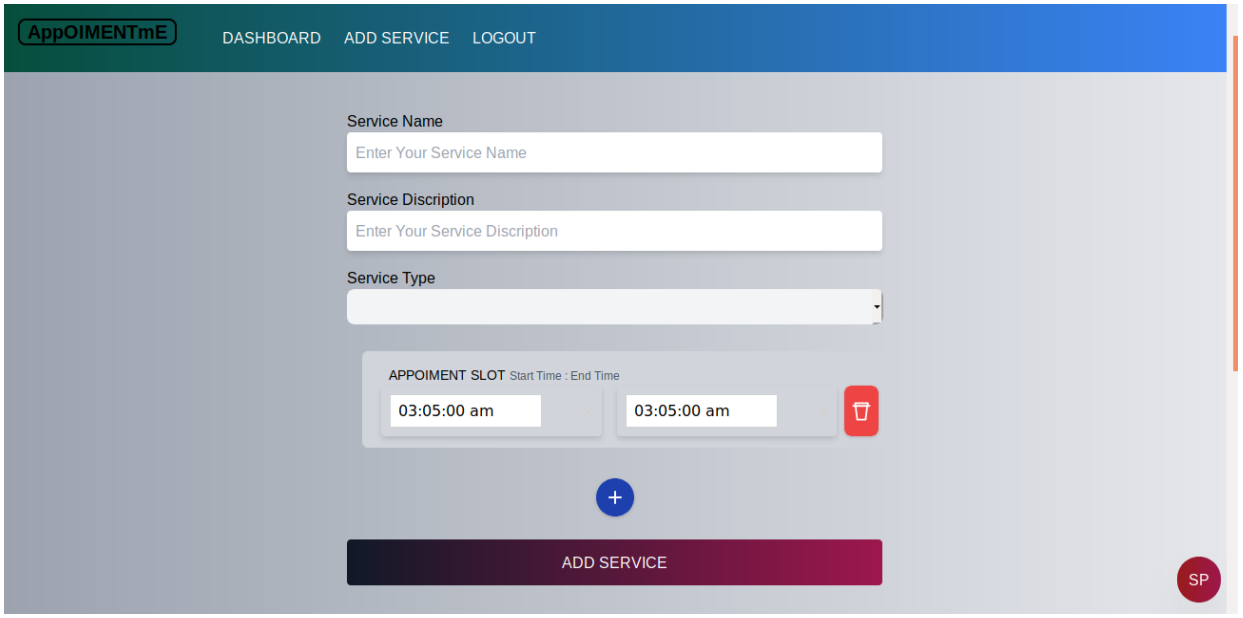


Figure 05 : Add services

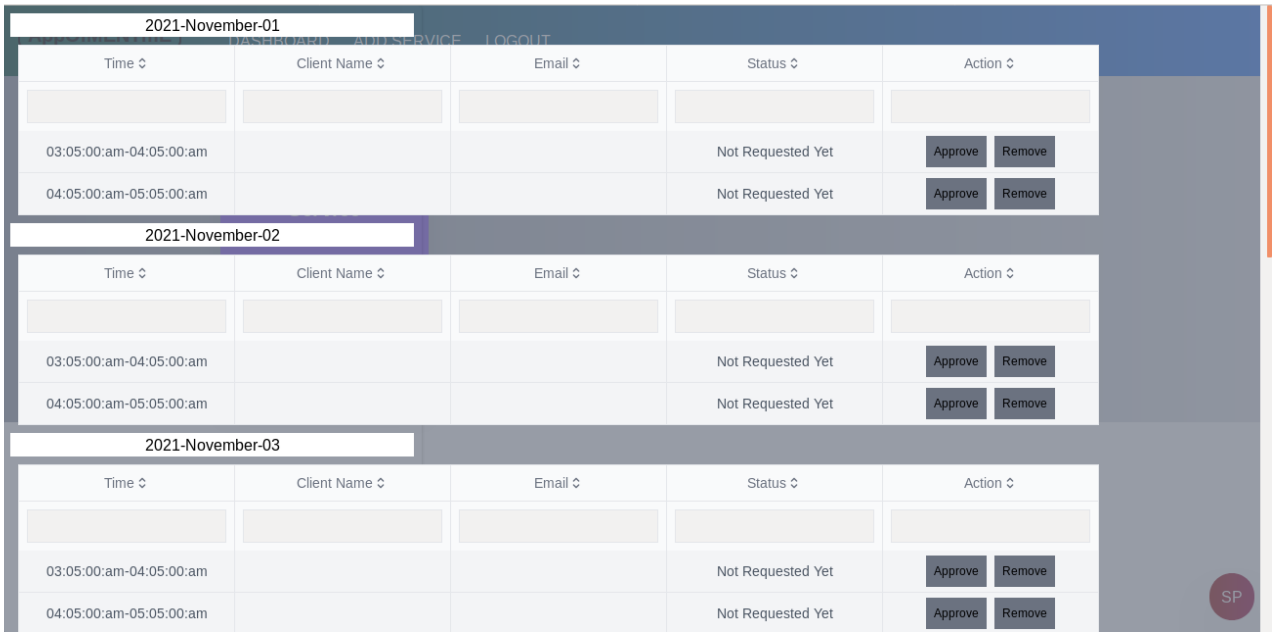


Figure 06 : Service Provider Dashboard

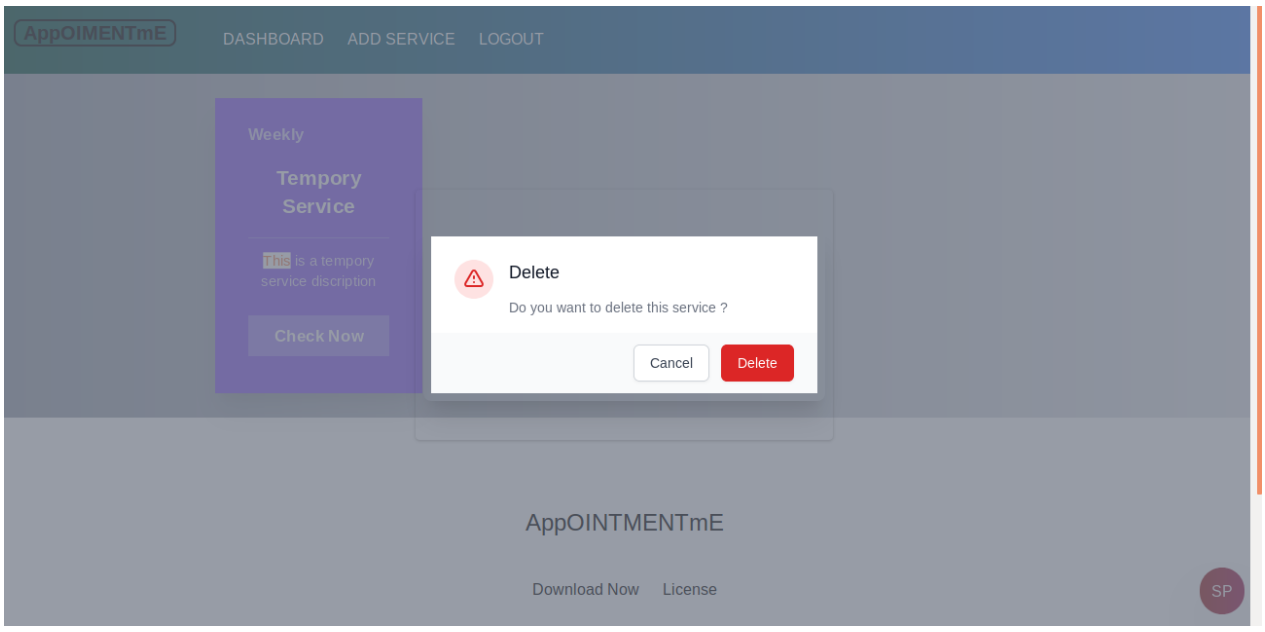


Figure 07 : Deleting Services

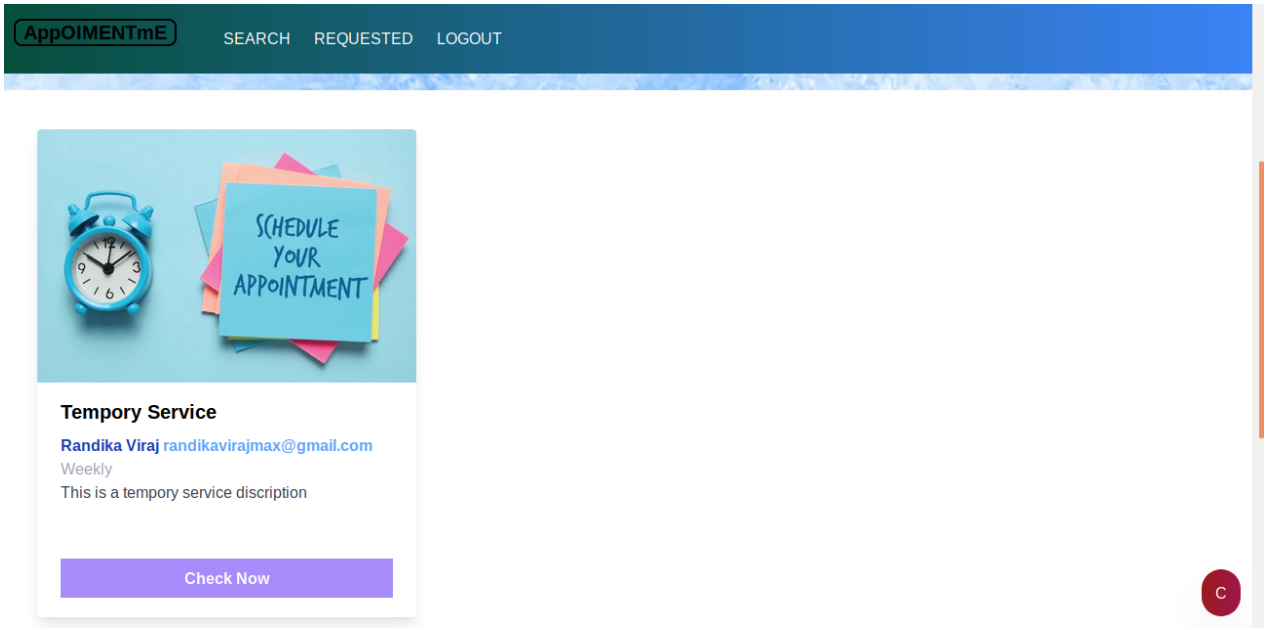


Figure 08 : UI for client to search services

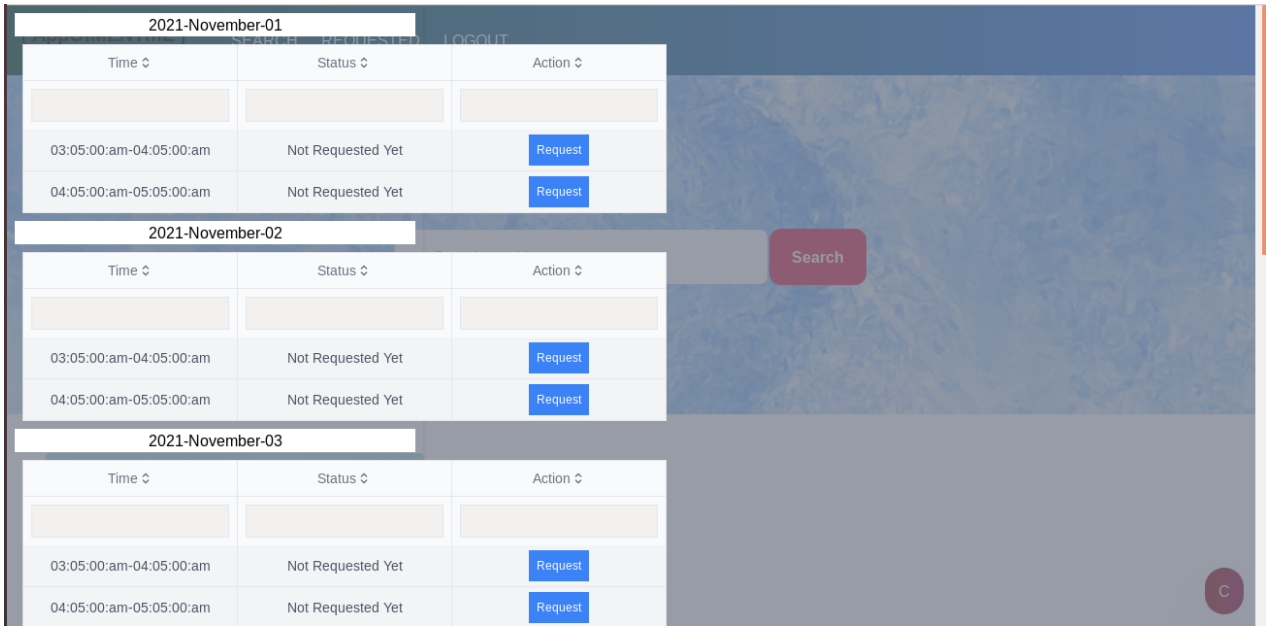


Figure 09 : Client side appointment schedule

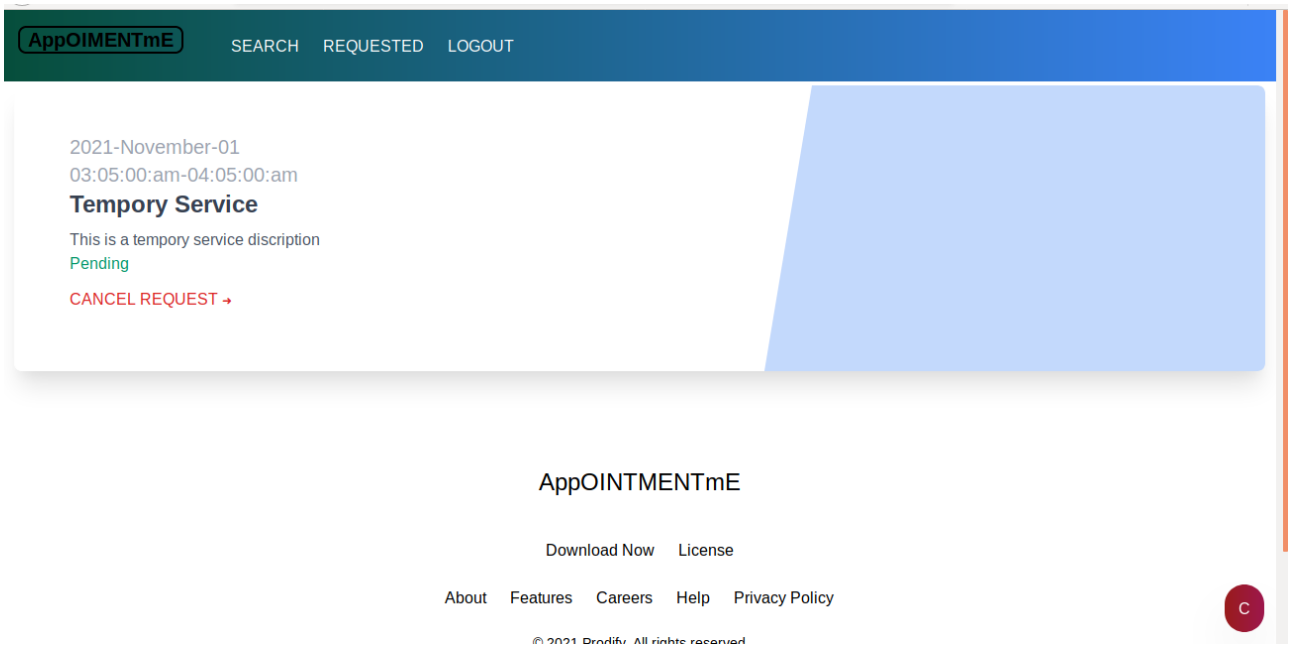


Figure 10 : Client Deleting requested Services

Database Design :

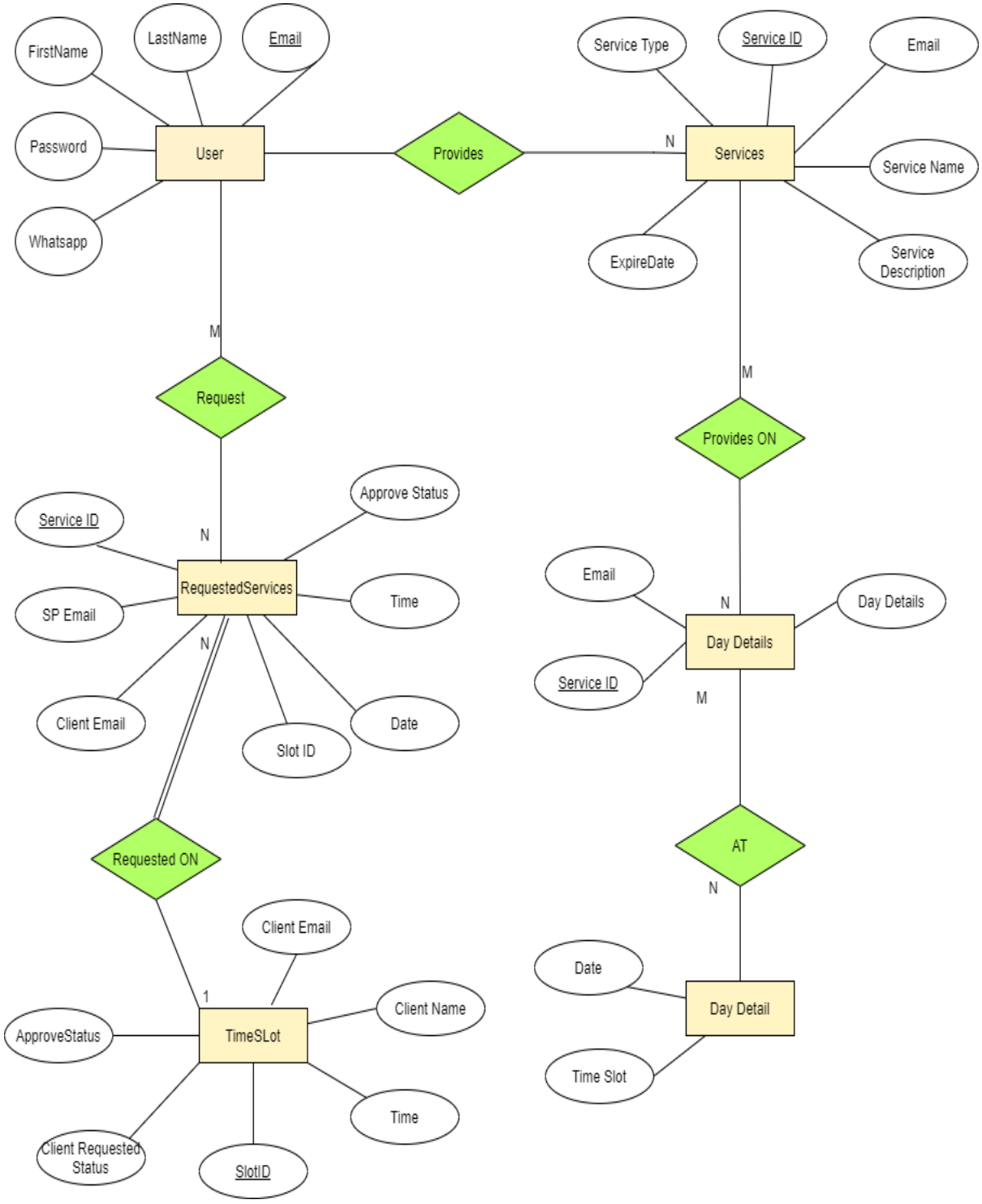


Figure 11 : ER diagram

Technology Justification

Website Frontend - Vue Js

- Simplicity
  The basic idea behind the development of Vue.js is to achieve good results with as little effort as possible so that the user can code using only a few lines. Vue.js is also great for working with components as it needs relatively little overhead because single-file components can store all the codes such as HTML, CSS, and JavaScript in a single file.
- Integration
  Can integrate Vue.js into other frameworks such as React enabling them to customize the project according to their respective requirements.
- Customization
  Vue.js is a great tool for developers because all of its functions are readily accessible. For ease of use, developers can easily name the function as they like. Every segment can have separate functions, which makes it easier to customize the application according to the individual requirements
- Few restrictions
  The design of Vue.js offers fewer restrictions and greater flexibility to complete the project. The core library focuses on the 'View' part; which combined with the modular approach, and the use of various libraries allow programmers to solve problems in different ways.

Mobile App Frontend - Flutter

- Flutter enables you to make instant changes in the app which is a god-sent when it comes to fixing bugs.
- Flutter-based apps are very smooth in their performance which makes for great UX.
- With a single code base, quality assurance and testing usually takes much less time.
- Developing in Flutter is very fast and efficient.

Backend - Golang

- Go takes static typing from C/C++ and aims for code readability similar to that of Python and JS.
- Unlike its predecessor, C, and competitor languages, Go promises development simplification and faster software.
- To be executed and faster , code written by developers has to be translated into machine code that processors understand.Go is faster it can translate its source code to machine code in its own compiler, which makes it faster than others.
- Concurrent programming is one of Go's strongest benefits. It has Goroutines, light-weight functions that can run independently at the same time, and uses a channel-based approach to concurrency.
- Golang has a very specific error syntax that includes errors as values in the code. It simplifies the bug identification process, which saves development time.

Database - MongoDB

- Performance of MongoDB is much higher than compared to any relational database.
- MongoDB is a schema-less NoSQL database. We do not need to design the schema of the database when we are using MongoDB. This saves a lot of time. Thus, the code being written defines the schema.
- It is very easy to set-up and install MongoDB.
- The document query language supported by MongoDB is very simple as compared to SQL queries.
- There is no need for mapping of application objects to database objects in MongoDB.
- No complex joins are needed in MongoDB. There is no relationship among data in MongoDB.

Testing

We unit tested all vue components to verify their functionality. We pass the props and the data to the components and we check for the existence of the components after render the component.

Tested Components:

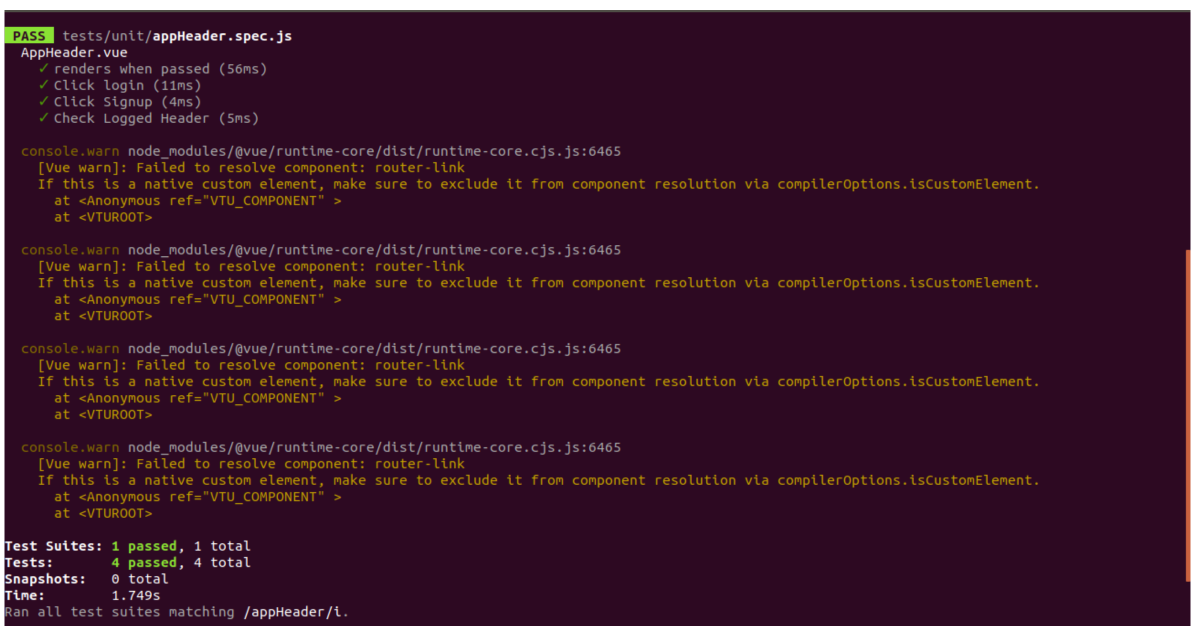
- Alert Component

```
> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit "alert"

 PASS  tests/unit/alert.spec.js
  Alert.vue
    ✓ renders props  when passed (17ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        9.539s
Ran all test suites matching /alert/i.
```

- AppHeader Component



```
 PASS  tests/unit/appHeader.spec.js
  AppHeader.vue
    ✓ renders when passed (56ms)
    ✓ Click login (11ms)
    ✓ Click Signup (4ms)
    ✓ Check Logged Header (5ms)

  console.warn node_modules/@vue/runtime-core/dist/runtime-core.cjs.js:6465
    [Vue warn]: Failed to resolve component: router-link
    If this is a native custom element, make sure to exclude it from component resolution via compilerOptions.isCustomElement.
      at <Anonymous ref="VTU_COMPONENT" >
      at <VTUROOT>

  console.warn node_modules/@vue/runtime-core/dist/runtime-core.cjs.js:6465
    [Vue warn]: Failed to resolve component: router-link
    If this is a native custom element, make sure to exclude it from component resolution via compilerOptions.isCustomElement.
      at <Anonymous ref="VTU_COMPONENT" >
      at <VTUROOT>

  console.warn node_modules/@vue/runtime-core/dist/runtime-core.cjs.js:6465
    [Vue warn]: Failed to resolve component: router-link
    If this is a native custom element, make sure to exclude it from component resolution via compilerOptions.isCustomElement.
      at <Anonymous ref="VTU_COMPONENT" >
      at <VTUROOT>

  console.warn node_modules/@vue/runtime-core/dist/runtime-core.cjs.js:6465
    [Vue warn]: Failed to resolve component: router-link
    If this is a native custom element, make sure to exclude it from component resolution via compilerOptions.isCustomElement.
      at <Anonymous ref="VTU_COMPONENT" >
      at <VTUROOT>

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        1.749s
Ran all test suites matching /appHeader/i.
```

- Appointment Card component

```
appoimentCard

> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit "appoimentCard"

 PASS  tests/unit/appoimentCard.spec.js
  AppoimentCard.vue
    ✓ renders props   when passed (34ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.235s
Ran all test suites matching /appoimentCard/i.
```

- Appointment slot component

```
appoimentSlot

> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit "appoimentSlot"

 PASS  tests/unit/appoimentSlot.spec.js
 PASS  tests/unit/searchAppoimentSlot.spec.js

Test Suites: 2 passed, 2 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        2.02s
Ran all test suites matching /appoimentSlot/i.
```

- Blue Modal

```
bluemoadal

> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit "bluemoadal"

 PASS  tests/unit/bluemoadal.spec.js
  BlueModal.vue
    ✓ renders props.eventName and props.customClases when passed (17ms)
    ✓ Emits an event when clicked background of BlueModal (11ms)
    ✓ Test BlueModal slot contain (16ms)

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        1.293s
Ran all test suites matching /bluemoadal/i.
```

- Card

```
card

> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit "card"

 PASS  tests/unit/appoimentCard.spec.js
 PASS  tests/unit/card.spec.js
 PASS  tests/unit/clientRequestedCard.spec.js

Test Suites: 3 passed, 3 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        2.265s
Ran all test suites matching /card/i.
```

- Client Requested card

```
 PASS  tests/unit/clientRequestedCard.spec.js
  ClientRequestedCard.vue
    ✓ renders props approved=false when passed (33ms)
    ✓ renders props approved=true when passed (6ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.945s, estimated 2s
Ran all test suites matching /clientRequestedCard/i.
```

- Floating button

```
floatingButton

> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit "floatingButton"

 PASS  tests/unit/floatingButton.spec.js
  FloatingButton.vue
    ✓ renders when passed (29ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.553s
Ran all test suites matching /floatingButton/i.
```

- Search appointment slot

```
searchAppoimentSlot

> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit "searchAppoimentSlot"

 PASS  tests/unit/searchAppoimentSlot.spec.js
  SearchAppoimentSlot.vue
    ✓ renders props clientRequested true  when passed (22ms)
    ✓ renders props clientRequested false  when passed (4ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.907s, estimated 2s
Ran all test suites matching /searchAppoimentSlot/i.
```

Overall Test summary :

```
(base) randika@randika-Aspire-E5-576:~/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app$ npm run test:unit

> appointment-scheduling-app@0.1.0 test:unit /home/randika/Desktop/e16-CO328-Appointment-Scheduling-Software/webapp/appointment-scheduling-app
> vue-cli-service test:unit

 PASS  tests/unit/clientRequestedCard.spec.js
 PASS  tests/unit/appoimentSlot.spec.js
 PASS  tests/unit/appHeader.spec.js
 PASS  tests/unit/bluemoadal.spec.js
 PASS  tests/unit/searchAppoimentSlot.spec.js
 PASS  tests/unit/appoimentCard.spec.js
 PASS  tests/unit/alert.spec.js
 PASS  tests/unit/floatingButton.spec.js
 PASS  tests/unit/card.spec.js

Test Suites: 9 passed, 9 total
Tests:       18 passed, 18 total
Snapshots:   0 total
Time:        9.485s
Ran all test suites.
```

Figure 12 : Test Summary

Use of Tools

- For development containers -Docker Mongodb instances
- Version Control - Git
- Diagram design - Draw io, Lucid app
- Vue Development kit and Vuejs dev tools for chrome
- Thunder Client Vs extension dev support and Postman
- VS code editor
- Mongodb compass