Verification and Validation Report: Sayyara

Team 3, Tiny Coders
Arkin Modi
Joy Xiao
Leon So
Timothy Choy

March 8, 2023

1 Revision History

Table 1: Revision History

Date	Developer(s)	Change
February 22, 2023	Arkin Modi	Add Automated Testing Section
March 3, 2023	Joy Xiao	Add Tests Evaluations for Legal and Cultural Requirements
March 3, 2023	Joy Xiao	Add Services and Appointments Test Evaluations
March 4, 2023	Leon So	Add Introduction
March 4, 2023	Arkin Modi	Add Tests Evaluation for Security Requirements
March 4, 2023	Arkin Modi	Update Look and Feel Requirements Tests
March 4, 2023	Arkin Modi	Add Unit Testing Summary Table
March 4, 2023	Leon So	Add NFR Tests for Operational and Environmental Requirements
March 4, 2023	Timothy Choy	Add Test Evaluations for Usability and Humanity Requirements
March 4, 2023	Timothy Choy	Add Test Evaluations for Performance Requirements
March 5, 2023	Leon So	Add Functional Requirement Tests for Shop Profile
March 5, 2023	Timothy Choy	Add Test Evaluations for Maintainability and Support Requirements
March 6, 2023	Leon So	Add Functional Requirement Tests for Authentication
March 6, 2023	Leon So	Add Functional Requirement Tests for Employee Management
March 6, 2023	Timothy Choy	Add Functional Requirement Tests for Shop Lookup
March 6, 2023	Arkin Modi	Add Functional Requirement Tests for Quotes
March 7, 2023	Arkin Modi	Add Functional Requirement Tests for Work Orders
March 7, 2023	Arkin Modi	Add Code Coverage Metrics Section
March 7, 2023	Arkin Modi	Add Changes Due to Testing Section
March 7, 2023	Timothy Choy	Add Unit Test Evaluations for Shop
March 8, 2023	Arkin Modi	Add Unit Test Results

2 Symbols, Abbreviations and Acronyms

symbol	description
API	Application Programming Interface
CD	Continuous Deployment
CI	Continuous Integration
CLS	Cumulative Layout Shift
FCP	First Contentful Paint
FID	First Input Delay
HTML	HyperText Markup Language
LCP	Largest Contentful Paint
ms	Milliseconds
PII	Personal Identifiable Information
PWA	Progressive Web Application
s	seconds
SRS	Software Requirements Specification
T	Test
TTFB	Time to First Byte
VnV	Verification and Validation

Contents

List of Tables

List of Figures

3 Introduction

3.1 Project Summary

Sayyara is a Progressive Web Application (PWA) which acts as a single platform for independent auto repair shops and vehicle owners. This platform allows independent auto repair shops and vehicle owners to interact in various ways. Using Sayyara, vehicle owners can search for auto repair shops and services based on a variety of search filters; request quotes for service; book, view, and manage service appointments. On the application, auto repair shop owners have full shop management capabilities such as: adding and managing a list of employees; managing a list of service types and corresponding service appointment availabilities; managing store information such as location, hours of operation, and contact information. Auto repair shop owners and employees will be able to manage quotes, service appointments, and work orders from a single application.

3.2 Purpose of Document

This document outlines the validation and verification process for the application, Sayyara, and a detailed summary of results for the planned tests defined in the VnV Plan.

3.3 Scope of Testing

As defined in the VnV Plan, the tests reported aim to verify and validate functional and nonfunctional requirements listed in the SRS.

In addition, the testing reported in this document aims to validate that the application provides adequate usability and to verify that system is in a functional state for the end users. The testing and validation will aid in ensuring that the product fulfills the system requirements, intended use, and goals of stakeholders.

4 Functional Requirements Evaluation

4.1 Authentication

1. **FRT-BE1-1**

   Control: Manual

   Initial State: Customer with no existing account

   Input: Customer submits sign up form with valid email, password, name, and phone number

   Output: Redirect to vehicle owner landing page upon successful registration

   Test Case Derivation: New customer wants to sign up for a new account with valid sign up information provided

   How test will be performed: The tester will register and sign up for a new customer account using valid sign up information

   Results: Passed

2. **FRT-BE1-2**

   Control: Manual

   Initial State: Shop owner with no existing account

   Input: Shop owner submits sign up form with valid email, password, name, phone number, shop name, shop address, shop phone number, and shop email

   Output: Redirect to shop owner landing page upon successful registration

   Test Case Derivation: New shop owner user wants to sign up for a new account with valid sign up information provided

   How test will be performed: The tester will register and sign up for a new shop owner account using valid sign up information

   Results: Passed

3. **FRT-BE1-3**

   Control: Manual

   Initial State: Employee with no existing account

   Input: Employee submits sign up form with valid email, password, name, phone number, and shop ID

   Output: Redirect to employee landing page upon successful registration

   Test Case Derivation: New employee user wants to sign up for a new account with valid sign up information provided

   How test will be performed: The tester will register and sign up for a new employee account using valid sign up information

   Results: Passed

4. **FRT-BE2-1**

   Control: Manual

   Initial State: Customer with existing account and not logged in

   Input: Customer submits login form with valid email, password, and vehicle information

   Output: Redirect to vehicle owner landing page upon successful login

   Test Case Derivation: Customer wants to login to existing with valid login credentials provided

   How test will be performed: The tester will login to an existing customer account using valid login credentials

   Results: Passed

5. **FRT-BE2-2**

   Control: Manual

Initial State: Shop owner with existing account and not logged in

Input: Shop owner submits login form with valid email and password

Output: Redirect to shop owner landing page upon successful login

Test Case Derivation: Shop owner wants to login to existing with valid login credentials provided

How test will be performed: The tester will login to an existing shop owner account using valid login credentials

Results: Passed

6. **FRT-BE2-3**

   Control: Manual

   Initial State: Employee with existing account and not logged in

   Input: Employee submits login form with valid email and password

   Output: Redirect to employee landing page upon successful login

   Test Case Derivation: Employee wants to login to existing with valid login credentials provided

   How test will be performed: The tester will login to an existing employee account using valid login credentials

   Results: Passed

7. **FRT-BE3-1**

   Control: Manual

   Initial State: Customer with existing account and logged in

   Input: Customer requests to logout

   Output: System logs the user out and transitions to home page

   Test Case Derivation: Customer wants to logout

   How test will be performed: The tester will request to logout on a customer account

   Results: Passed

8. **FRT-BE3-2** Control: Manual

   Initial State: Shop owner with existing account and logged in

   Input: Shop owner requests to logout

   Output: System logs the user out and transitions to home page

   Test Case Derivation: Shop owner wants to logout

   How test will be performed: The tester will request to logout on a shop owner account

   Results: Passed

9. **FRT-BE3-3**

   Control: Manual

   Initial State: Employee with existing account and logged in

   Input: Employee requests to logout

   Output: System logs the user out and transitions to home page

   Test Case Derivation: Employee wants to logout

   How test will be performed: The tester will request to logout on an employee account

   Results: Passed

4.2 Appointments

1. **FRT-BE4-1**

   Control: Manual

   Initial State: Customer account which received a quote from an auto shop

   Input: Enters service request information and selects an available appointment time slot

   Output: A new appointment request is created with service details linked to the quote. Shop Owner/Employee Accounts will receive the appointment request

   Test Case Derivation: The appointment will be requested with the details the user entered

   How test will be performed: The tester will go through the appointment booking process through the user interface through quotes

   Results: Passed

2. **FRT-BE4-2**

   Control: Manual

   Initial State: Customer account selects a canned job at an auto shop

   Input: Enters service request information and selects an available appointment time slot

   Output: A new appointment request is created with service details. Shop Owner/Employee Accounts will receive the appointment request

   Test Case Derivation: The appointment will be requested with the canned job details as well as details the user entered

   How test will be performed: The tester will go through the appointment booking process through the user interface by selecting a canned job

   Results: Passed

3. **FRT-BE4-3**

   Control: Manual

   Initial State: Shop Owner/Employee Account with appointment requests

   Input: Accept an appointment request

   Output: Appointment booking displayed for Shop Owner/Employee accounts and Customer account

   Test Case Derivation: The appointment request is accepted and appointment booking is completed

   How test will be performed: The tester will accept an appointment request through the user interface

   Results: Passed

4. **FRT-BE5-1**

   Control: Manual

   Initial State: Customer Account has a service appointment scheduled

   Input: Selects an appointment to edit and selects a new available time slot for the appointment

   Output: A new appointment request is made with the new time slot. Shop Owner/Employee Accounts will receive the appointment request

   Test Case Derivation: The user is able to change their appointment to any of the available time slots displayed

   How test will be performed: The tester will go through the user interface to edit an appointment booking

   Results: Failed. Cannot edit an appointment at the moment but can cancel and reschedule an appointment

5. **FRT-BE5-2**

   Control: Manual

   Initial State: Shop Owner/Employee Account has a service appointment scheduled

   Input: Selects an appointment to edit and selects a new available time slot for the appointment

   Output: The appointment booking is updated to the new time slot

   Test Case Derivation: The user is able to change their appointment to any of the available time slots displayed

   How test will be performed: The tester will go through the user interface to edit an appointment booking time slot

   Results: Failed. Cannot edit an appointment at the moment but can cancel and reschedule an appointment.

6. **FRT-BE5-3**

   Control: Manual

   Initial State: Shop Owner/Employee Account has a service appointment scheduled

   Input: Selects an appointment to edit service details

   Output: The appointment booking is updated with the updated service details

   Test Case Derivation: The user is able to update the appointment details

   How test will be performed: The tester will go through the user interface to edit appointment details

   Results: Passed

7. **FRT-BE6-1**

   Control: Manual

   Initial State: Customer Account has a service appointment scheduled

   Input: Selects an appointment to cancel

   Output: The appointment booking is cancelled and removed from the customer and shop owner/employee schedules

   Test Case Derivation: The appointment should be removed from the calendar when it is cancelled

   How test will be performed: The tester will go through the user interface to cancel an appointment

   Results: Passed

8. **FRT-BE6-2**

   Control: Manual

   Initial State: Shop Owner/Shop Employee has a service appointment scheduled

   Input: Selects an appointment to cancel

   Output: The appointment booking is cancelled and removed from the customer and shop owner/employee schedules

   Test Case Derivation: The appointment should be removed from the calendar when it is cancelled

   How test will be performed: The tester will go through the user interface to cancel an appointment

   Results: Passed

9. **FRT-BE7-1**

   Control: Manual

   Initial State: Shop Owner selects appointment availability

Input: Sets appointment availabilities

Output: The appointment availability is updated in the database

Test Case Derivation: The appointment availability will be saved, and availability time slots to book appointments will be updated

How test will be performed: The tester will go through the user interface to set appointment availabilities

Results: Passed

4.3 Quotes

1. **FRT-BE8-1**

   Control: Manual

   Initial State: Customer Account with no quote requests initiated from the account

   Input: Request past quotes

   Output: A message saying there are no past quotes

   Test Case Derivation: There are no past quotes to display

   How test will be performed: The tester will request for past quotes through the user interface

   Results: Passed

2. **FRT-BE8-2**

   Control: Manual

   Initial State: Customer Account with quotes requests which have been completed and were initiated by the account

   Input: Request all quotes

   Output: A table with all the quotes

   Test Case Derivation: There are past quotes to display

   How test will be performed: The tester will request for past quotes through the user interface

   Results: Passed

3. **FRT-BE8-3**

   Control: Manual

   Initial State: Shop Owner/Employee Account with no quotes requests assigned to the shop

   Input: Request all quotes

   Output: A message saying there are no past quotes

   Test Case Derivation: There are no past quotes to display

How test will be performed: The tester will request for past quotes through the user interface

Results: Passed

4. **FRT-BE8-4**

   Control: Manual

   Initial State: Shop Owner/Employee Account with completed quotes requests that were assigned to the shop

   Input: Request all quotes

   Output: A table with all the quotes

   Test Case Derivation: There are past quotes to display

   How test will be performed: The tester will request for past quotes through the user interface

   Results: Passed

5. **FRT-BE9-1**

   Control: Manual

   Initial State: Customer Account with no quote requests initiated from the account

   Input: A filled in quote request form

   Output: A confirmation messaging indicating that the quote was requested

   Test Case Derivation: The system shall verify that the quote was created successfully

   How test will be performed: The tester will create a quote through the user interface

   Results: Passed

6. **FRT-BE10-1**

   Control: Manual

   Initial State: Customer Account with a completed quote

   Input: A quote's chat history is requested

   Output: The quote's chat history

   Test Case Derivation: There are chat messages to display

   How test will be performed: The tester will view a quote's chat messages through the user interface

   Results: Passed

7. **FRT-BE10-2**

   Control: Manual

Initial State: Shop Owner/Employee Account with a registered shop and a completed quote

Input: A quote's chat history is requested

Output: The quote's chat history

Test Case Derivation: There are chat messages to display

How test will be performed: The tester will view a quote's chat messages through the user interface

Results: Passed

8. **FRT-BE11-1**

   Control: Manual

   Initial State: Customer Account with a created quote

   Input: A chat message

   Output: A chat message is created and attached to the quote

   Test Case Derivation: The system shall send chat messages upon request

   How test will be performed: The tester will send a chat message within a quote through the user interface

   Results: Passed

9. **FRT-BE11-2**

   Control: Manual

   Initial State: Shop Owner/Employee Account with a registered shop and a created quote

   Input: A chat message

   Output: A chat message is created and attached to the quote

   Test Case Derivation: The system shall send chat messages upon request

   How test will be performed: The tester will send a chat message within a quote through the user interface

   Results: Passed

10. **FRT-BE12-1**

    Control: Manual

    Initial State: Shop Owner/Employee Account with a registered shop and a created quote

    Input: An filled in appointment invitation form

    Output: An appointment invitation is sent to the customer

Test Case Derivation: The system shall verify that the appointment invitation was created successfully

How test will be performed: The tester will send an appointment invitation from a quote through the user interface

Results: Passed

11. **FRT-BE13-1**

    Control: Manual

    Initial State: Customer Account with a quote that has an appointment invitation

    Input: Accept an appointment request

    Output: The system shall request the user to create an appointment with the shop that responded

    Test Case Derivation: The system shall have the user create an appointment upon accepting an appointment invitation from a quote

    How test will be performed: The tester will accept an appointment invitation from a quote through the user interface

    Results: Passed

4.4 Work Orders

1. **FRT-BE14-1**

    Control: Manual

    Initial State: Shop Owner/Employee Account with a registered shop and a quote

    Input: A new appointment created at the shop from the quote

    Output: A new work order is created with details from the quote

    Test Case Derivation: The system shall automatically create a work order for new appointment with information from the quote

    How test will be performed: The tester will create an appointment through the user interface

    Results: Passed

2. **FRT-BE14-2**

    Control: Manual

    Initial State: Shop Owner/Employee Account with a registered shop

    Input: A new appointment created at the shop

    Output: A new work order is created

    Test Case Derivation: The system shall automatically create a work order for new appointment

How test will be performed: The tester will create an appointment through the user interface

Results: Passed

3. **FRT-BE15-1**

   Control: Manual

   Initial State: Customer Account with a vehicle, an upcoming appointment, and a work order associated to the appointment

   Input: A list of upcoming appointments is requested

   Output: The appointments in a table format with the associated work order attached to each appointment

   Test Case Derivation: The system shall display work order upon request

   How test will be performed: The tester will search for a work order through the user interface

   Results: Passed

4. **FRT-BE15-2**

   Control: Manual

   Initial State: Shop Owner/Employee Account with a registered shop, an upcoming appointment, and a work order associated to the appointment

   Input: A list of upcoming appointments is requested

   Output: The appointments in a table format with the associated work order attached to each appointment

   Test Case Derivation: The system shall display work order upon request

   How test will be performed: The tester will search for a work order through the user interface

   Results: Passed

5. **FRT-BE16-1**

   Control: Manual

   Initial State: Shop Owner/Employee Account with a registered shop, an upcoming appointment, and a work order associated to the appointment

   Input: Request to update a work order with new details

   Output: A work order updated confirmation message

   Test Case Derivation: The system shall update work order upon request

   How test will be performed: The tester will update a work order through the user interface

   Results: Passed

4.5 Employee Management

1. **FRT-BE24-1**

   Control: Manual

   Initial State: Shop Owner has a shop profile

   Input: Shop owner wants to retrieve shop ID to invite employee

   Output: Shop ID is displayed

   Test Case Derivation: The system shall allow the shop owner to retrieve their shop ID to invite their employee

   How test will be performed: The tester will use a shop ID displayed on the shop owner's employee management page and go through the sign up flow as an employee. The tester will verify that the shop ID is displayed and correct.

   Results: Passed

2. **FRT-BE24-2**

   Control: Manual

   Initial State: Shop Owner has a shop profile

   Input: Employee signs up using invalid user sign up information and shop ID

   Output: Message indicating that the inputs are not valid

   Test Case Derivation: The system shall not allow the employee to sign up with the information provided

   How test will be performed: The tester will use invalid user sign up information and invalid shop ID to sign up as an employee

   Results: Passed

3. **FRT-BE25-1**

   Control: Manual

   Initial State: Shop owner account with employees listed under the shop

   Input: The user enters search text to search for an employee

   Output: System displays a list of employees whose name or email matches the search text

   Test Case Derivation: The system shall allow the user to enter search text to search for an employee; the system shall display a list of employees whose name or email matches the search text

   How test will be performed: The tester will use a shop owner account with employees listed under the shop and enter search text to search for an employee

   Results: Passed

4. **FRT-BE26-1**

   Control: Manual

   Initial State: Shop owner account with employees

   Input: The user attempts to view the list of employees

   Output: Table of employees details and employee management options

   Test Case Derivation: The shop owner with employees wants to view the list of employees

   How test will be performed: The tester will use a shop owner account with employees listed under the shop and attempt to view the list of employees

   Results: Passed

5. **FRT-BE27-1**

   Control: Manual

   Initial State: Shop owner account with employees

   Input: The user attempts to remove an employee

   Output: Table of employees details and employee management options updates with employee removed; the remove employee's access should be revoked

   Test Case Derivation: The shop owner with employees wants to remove an employee

   How test will be performed: The tester will use a shop owner account with employees listed under the shop and attempt to remove an employee

   Results: Passed

4.6 Services

1. **FRT-BE28-1**

   Control: Manual

   Initial State: Shop Owner has a shop profile

   Input: Shop Owner adds available services

   Output: Shop services are sent to the database and displayed on the shop profile

   Test Case Derivation: The user can add available auto shop services to their shop profile

   How test will be performed: The tester will add a service to the shop profile and confirms it shows up on the shop profile

   Results: Passed

2. **FRT-BE29-1**

   Control: Manual

Initial State: Shop Owner/Shop Employee searches for auto repair or maintenance services

Input: Enters specific auto service in the search bar

Output: Auto shops with relevant services displayed

Test Case Derivation: Services corresponding to the auto shop will be displayed

How test will be performed: The tester will search for shop services through the user interface and confirms that all the shop's services matching the search criteria are listed

Results: Passed

3. **FRT-BE30-1**

   Control: Manual

   Initial State: Shop Owner with services listed on their shop profile

   Input: Updates the details of a service listed on their profile

   Output: The service is updated with the entered details

   Test Case Derivation: The shop profile will list the service with the updated details

   How test will be performed: The tester will update the details of a shop service through the user interface and confirms that the shop profile is updated

   Results: Passed

4. **FRT-BE31-1**

   Control: Manual

   Initial State: Shop Owner with services listed on their shop profile

   Input: Deletes a service listed on their shop profile

   Output: Service is removed from the shop profile. Other services that were not deleted continue to be listed on the shop profile

   Test Case Derivation: The service that is deleted is removed from the shop profile

   How test will be performed: The tester will delete a shop service through the user interface and confirms that the shop profile is updated

   Results: Passed

4.7 Shop Lookup

1. **FRT-BE32-1**

   Control: Manual

   Initial State: Shop Owner/Employee Account with a registered shop, Customer account exists

Input: The Customer account attempts to view a list of shops using the parameters of shop name, service name, part types and part conditions

Output: A list of shops based on the input parameters

Test Case Derivation: The system shall show a list of shops with the specified parameters provided

How test will be performed: The tester will search for shops, and input various parameters to see if the shop list changes correctly

Results: Passed

4.8 Shop Profile

1. **FRT-BE33-1**

   Control: Manual

   Initial State: Shop Owner/Employee Account with a registered shop

   Input: Request to view shop profile

   Output: Information regarding the shop, which includes the shop's address, phone number, and email

   Test Case Derivation: The system shall display information about the shop upon request

   How test will be performed: The tester will go through the user interface to view a shop profile

   Results: Passed

2. **FRT-BE34-1**

   Control: Manual

   Initial State: User wants to view a shop's profile

   Input: Request to view shop profile

   Output: Information regarding the shop, which includes the shop's address, phone number, email, and hours of operation (if any)

   Test Case Derivation: The system shall display information about the shop upon request

   How test will be performed: The tester will go through the user interface to view a shop profile

   Results: Passed

3. **FRT-BE35-1**

   Control: Manual

   Initial State: Shop Owner wants to add/update hours of operation of their shop

   Input: Request to add and edit hours of operation with valid hours of operation

Output: Updated hours of operation

Test Case Derivation: The system shall allow the shop owner to add/edit their hours of operation

How test will be performed: The tester will go through the user interface to add/edit a shop

Results: Passed

5 Nonfunctional Requirements Evaluation

5.1 Look and Feel Requirements

1. **NFRT-LF1-1**

   Type: Dynamic, Manual

   Initial State: The application is accessible through the Google Chrome web browser

   Input/Condition: The window size is changed

   Output/Result: The application shall adjust and scale to fit the new window size

   How test will be performed: The tester will access the application through Google Chrome on their desktop/laptop and change the windows through the use of Google Chrome DevTools Device Toolbar

   Results: Passed

2. **NFRT-LF2-1**

   Type: Dynamic, Manual

   Initial State: The application is accessible through the web browser

   Input: The application is opened in a full screened web browser window

   Output: All text on the screen shall be readable

   How test will be performed: The tester open the application in a full screened web browser window, navigate to all pages, and judge if all text on the screen is readable from sitting 50 centimeters away from the monitor

   Results: Passed

3. **NFRT-LF3-1**

   Type: Dynamic, Manual

   Initial State: The application is accessible through the web browser and there is a completed work order and quote on the user's account

   Input: The user opens the work order details and the quote details

   Output: All currency shall be rounded to two decimal places

   How test will be performed: The tester will navigate to the work order and quote and verify that all values of currency are rounded to two decimal places

Results: Passed

5.2 Usability and Humanity Requirements

1. **NFRT-UH1-1**

   Type: Dynamic, Manual

   How test will be performed: The testers will complete the manual system tests for functional requirements on a macOS desktop/laptop device, a Windows desktop/laptop, an iOS mobile device, and an android mobile device.

   Results: Passed. Testers were asked to write down which device they used to test to ensure all devices and operating systems were covered.

2. **NFRT-UH2-1**

   Type: Dynamic, Manual

   Initial State: Device is connected to the internet and application is not open.

   Input/Condition: The user launches the application on a web browser.

   Output/Result: The system can be assessed through the web browser.

   How test will be performed: The testers will attempt to launch the application on a web browser, using a device that is connected to the internet.

   Results: Passed. The testers were able to access the application using the web browser while connected to the internet.

3. **NFRT-UH3-1**

   Type: Dynamic, Manual

   Initial State: Device is connected to the internet and application is open.

   Input/Condition: User disconnects from the internet.

   Output/Result: The system notifies the user that there is no network connection.

   How test will be performed: The testers disconnects from the internet while the application is open.

   Results: Passed. Testers were greeted with a toast when they disconnected from the internet.

5.3 Performance Requirements

1. **NFRT-PR1-1**

   Type: Dynamic, Manual

   How test will be performed: A function will be added to the application which logs web metrics of the system. Testers will go through each of the pages and functions, making sure that the metrics meet current web standards as stated in this article. These criteria include:

- LCP < 2.5s
- FID < 100ms
- CLS < 0.1
- TTFB < 0.5s
- FCP < 1.8s

Results: Passed. Testers were able to view logs of their progress and record results which met or exceeded current web standards for the metrics.

5.4 Operational and Environmental Requirements

1. **NFRT-OE-1**

   Type: Dynamic, Manual

   Initial State: The application is accessible through the Google Chrome web browser

   Input: The user navigates across all pages and dialogs, and performs all manual system tests for functional requirements

   Output: All pages and dialogs should be fully operational on the below listed device dimensions

   How test will be performed: The testers will complete the manual system tests for functional requirements on Google Chrome using various device dimension options listed in the Google DevTools Device Toolbar. At minimum, the testers will test with the following dimensions: Responsive, iPhone 12 Pro (390×844), iPhone SE (375×667), and iPad Air (820×1180). This set will provide a reasonable variety of different device dimensions.

   Results: Failed. Manually tested to verify whether the application supports the above device dimensions. Shop lookup page does not work on iPad Air (820×1180) dimensions. Chat window does not work on iPhone SE (375×667) dimensions.

5.5 Maintainability and Support Requirements

1. **NFRT-MS1-1**

   Type: Manual

   How test will be performed: Every document will have to pass a review by every developer before the document is finalized. The review will include running through the system and linking every module to proper documentation.

   Results: Passed. This requirement has been maintained throughout the entire project.

5.6 Security Requirements

1. **NFRT-SR1-1**

   Type: Dynamic, Automatic

Initial State: An account with a created appointment

Input/Condition: An HTTP request to the application's backend to get the appointment by appointment ID

Output/Result: The request is rejected with a 403 Forbidden error message

How test will be performed: Through Jest an HTTP request will be sent to the server

Results: Passed.

2. **NFRT-SR2-1**

   Type: Dynamic, Manual

   Initial State: One Shop Owner account and two Customer accounts with one customer having an appointment at the shop registered under the same Shop Owner account

   Input/Condition: Request available time slots at the Shop Owner's store using the Customer without an appointment's account

   Output/Result: The Customer should only be able to view the time slot taken by the other customer and no other extra information

   How test will be performed: The tester will view the available time slots page as the Customer without an appointment

   Results: Failed. Extra information is returned from the API call that is not needed for these endpoints and exposes data that does not belong to user. None of the data contains PII.

5.7 Cultural Requirements

1. **NFRT-CR1-1**

   Type: Static, Manual

   Initial State: The application is accessible through the web browser

   Input: The user navigates across all pages and dialogs, and performs all end-to-end tests

   Output: There should be no offensive texts or images

   How test will be performed: The testers will go through end-to-end tests for the entire program and will confirm that any texts or images are not offensive.

   Results: Passed. Manually tested to verify that there are no offensive images or text.

2. **NFRT-CR2-1** Type: Static, Manual

   Initial State: The application is accessible through the web browser

   Input: The user navigates across all pages and dialogs, and performs all end-to-end tests

Output: All text should be displayed in English

How test will be performed: The testers will go through end-to-end tests for the entire program and will confirm that the text displayed is in English.

Results: Passed. Checked that everything is written in English in the implementation, documentation, and anything visible to the user.

5.8 Legal Requirements

1. **NFRT-LR1-1**

   Type: Static, Manual

   Input: The user navigates all source code and project documents

   Output: Appropriate credits and/or copyright licenses are used, and MIT License requirements should be adhered to

   How test will be performed: The testers will go through the source code of the project and project documents to determine if any open source resources were used and check that appropriate credits and/or copyright licenses are used.

   Results: Passed. Checked the MIT License requirements and that the project adhered to all the requirements.

6 Unit Testing

A summary of all the test results can be found in table **??**.

Table 2: Unit Tests Results Summary

✓ = Pass, X = Fail					
Test ID	**Result**	**Test ID**	**Result**	**Test ID**	**Result**
FRT-M3-1	✓	FRT-M3-2	✓	FRT-M3-3	✓
FRT-M3-4	✓	FRT-M3-5	✓	FRT-M3-6	✓
FRT-M3-7	✓	FRT-M3-8	✓	FRT-M3-9	✓
FRT-M3-10	✓	FRT-M4-1	✓	FRT-M4-2	✓
FRT-M4-3	✓	FRT-M4-4	✓	FRT-M4-5	✓
FRT-M3-6	✓	FRT-M4-7	✓	FRT-M4-8	✓
FRT-M4-9	✓	FRT-M4-10	✓	FRT-M4-11	✓
FRT-M4-12	✓	FRT-M4-13	✓	FRT-M4-14	✓
FRT-M4-15	✓	FRT-M4-16	✓	FRT-M4-17	✓
FRT-M5-1	✓	FRT-M5-2	✓	FRT-M5-3	✓
FRT-M5-4	✓	FRT-M5-5	✓	FRT-M5-6	✓
FRT-M5-7	✓	FRT-M5-8	✓	FRT-M5-9	✓
FRT-M5-10	✓	FRT-M5-11	✓	FRT-M5-12	✓
FRT-M5-13	✓	FRT-M5-14	✓	FRT-M6-1	✓
FRT-M6-2	✓	FRT-M6-3	✓	FRT-M6-4	✓
FRT-M6-5	✓	FRT-M6-6	✓	FRT-M6-7	✓
FRT-M7-1	✓	FRT-M7-2	✓	FRT-M7-3	✓
FRT-M7-4	✓	FRT-M7-5	✓	FRT-M7-6	✓
FRT-M8-1	✓	FRT-M8-2	✓	FRT-M8-3	✓
FRT-M8-4	✓	FRT-M8-5	✓	FRT-M8-6	✓
FRT-M8-7	✓	FRT-M8-8	✓	FRT-M8-9	✓
FRT-M8-10	✓	FRT-M8-11	✓	FRT-M8-12	✓
FRT-M8-13	✓	FRT-M8-14	✓	FRT-M8-15	✓
FRT-M9-1	✓	FRT-M9-2	✓	FRT-M9-3	✓

FRT-M9-4	✓	FRT-M9-5	✓	FRT-M9-6	✓
FRT-M9-7	✓	FRT-M9-8	✓	FRT-M9-9	✓
FRT-M9-10	✓				

6.1 User Module

1. **FRT-M3-1**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: N/A

   Input: Create customer request received with valid information

   Output: Customer and Vehicle are created

   Test Case Derivation: New customer wants to sign up for a new account with valid sign up information provided

   How test will be performed: Jest will send the create customer request with valid information

   Result: Passed

2. **FRT-M3-2**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: N/A

   Input: Create customer request received with invalid information

   Output: Request is rejected

   Test Case Derivation: New customer wants to sign up for a new account but not all information is provided

   How test will be performed: Jest will send the create customer request with invalid information

   Result: Passed

3. **FRT-M3-3**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: N/A

   Input: Create shop owner request received with valid information

   Output: Shop Owner and Shop are created

   Test Case Derivation: New shop owner wants to sign up for a new account with valid sign up information provided

   How test will be performed: Jest will send the create shop owner request with valid information

Result: Passed

4. **FRT-M3-4**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: N/A

   Input: Create shop owner request received with invalid information

   Output: Request is rejected

   Test Case Derivation: New shop owner wants to sign up for a new account but not all information is provided

   How test will be performed: Jest will send the create shop owner request with invalid information

   Result: Passed

5. **FRT-M3-5**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop exists

   Input: Create employee request received with valid information

   Output: Employee is created and register to given shop

   Test Case Derivation: New employee wants to sign up for a new account with valid sign up information provided

   How test will be performed: Jest will send the create employee request with valid information

   Result: Passed

6. **FRT-M3-6**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop exists

   Input: Create employee request received with invalid information

   Output: Request is rejected

   Test Case Derivation: New employee wants to sign up for a new account but not all information is provided

   How test will be performed: Jest will send the create employee request with invalid information

   Result: Passed

7. **FRT-M3-7**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A user exists

Input: Get user by an email address for an existing user

Output: The user

Test Case Derivation: Need to look up user data by the user provided email address

How test will be performed: Jest will call the internal command to get a user by email address

Result: Passed

8. **FRT-M3-8**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A user exists

   Input: Get user by an email address for a user that does not exist

   Output: None

   Test Case Derivation: Need to look up user data by the user provided email address

   How test will be performed: Jest will call the internal command to get a user by email address

   Result: Passed

9. **FRT-M3-9**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A user exists

   Input: Authorize a user with a valid email and password

   Output: The user's data

   Test Case Derivation: A user wants to authorize themselves in order to view their data (e.g., appointments, work orders, etc.)

   How test will be performed: Jest will call the internal command to authorize a user

   Result: Passed

10. **FRT-M3-10**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A user exists

    Input: Authorize a user with an invalid email and password

    Output: A unauthorized error message

    Test Case Derivation: A user wants to authorize themselves in order to view their data (e.g., appointments, work orders, etc.)

    How test will be performed: Jest will call the internal command to authorize a user

    Result: Passed

6.2 Quotes Module

1. **FRT-M4-1**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A customer, a vehicle, a shop, a shop owner, and a "CUSTOM" service exists

   Input: Create quote request with valid information

   Output: A quote is created

   Test Case Derivation: A customer wants to request a quote from a shop

   How test will be performed: Jest will send a create quote request with valid information

   Result: Passed

2. **FRT-M4-2**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A customer, a vehicle, a shop, a shop owner, and a "CUSTOM" service exists

   Input: Create quote request with invalid information

   Output: Request is rejected

   Test Case Derivation: A customer wants to request a quote from a shop

   How test will be performed: Jest will send a create quote request with invalid information

   Result: Passed

3. **FRT-M4-3**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A quote exists

   Input: Update quote request with a valid ID and valid information

   Output: The quote is updated

   Test Case Derivation: A shop owner/employee wants to invite the customer to book an appointment and need to prepopulate the booking with service information

   How test will be performed: Jest will send an update quote request with a valid ID and valid information

   Result: Passed

4. **FRT-M4-4**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A quote exists

Input: Update quote request with an invalid ID and valid information

Output: Request is rejected

Test Case Derivation: A shop owner/employee wants to invite the customer to book an appointment and need to prepopulate the booking with service information

How test will be performed: Jest will send an update quote request with an invalid ID and valid information

Result: Passed

5. **FRT-M4-5**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A quote exists

   Input: Update quote request with a valid ID and invalid information

   Output: Request is rejected

   Test Case Derivation: A shop owner/employee wants to invite the customer to book an appointment and need to prepopulate the booking with service information

   How test will be performed: Jest will send an update quote request with a valid ID and invalid information

   Result: Passed

6. **FRT-M4-6**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A quote exists

   Input: Get quote request with a valid quote ID

   Output: Quote data is returned

   Test Case Derivation: A shop owner/employee wants to view the quote

   How test will be performed: Jest will send a get quote request with a valid quote ID

   Result: Passed

7. **FRT-M4-7**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A quote exists

   Input: Get quote request with an invalid quote ID

   Output: Request is rejected

   Test Case Derivation: A shop owner/employee wants to view the quote

   How test will be performed: Jest will send a get quote request with an invalid quote ID

Result: Passed

8. **FRT-M4-8**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A quote exists

   Input: Get quote request with a valid customer ID

   Output: List of quotes is returned

   Test Case Derivation: A shop owner/employee wants to view the quote

   How test will be performed: Jest will send a get quote request with a valid customer ID

   Result: Passed

9. **FRT-M4-9**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A quote exists

   Input: Get quote request with an invalid customer ID

   Output: Empty list

   Test Case Derivation: A shop owner/employee wants to view the quote

   How test will be performed: Jest will send a get quote request with an invalid customer ID

   Result: Passed

10. **FRT-M4-10**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

    Input: Get quote request with a valid shop ID

    Output: List of quotes is returned

    Test Case Derivation: A shop owner/employee wants to view the quote

    How test will be performed: Jest will send a get quote request with a valid shop ID

    Result: Passed

11. **FRT-M4-11**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

    Input: Get quote request with an invalid shop ID

    Output: Empty list

    Test Case Derivation: A shop owner/employee wants to view the quote

How test will be performed: Jest will send a get quote request with an invalid shop ID

Result: Passed

12. **FRT-M4-12**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

    Input: Delete quote request with a valid quote ID

    Output: Quote and Chat Messages are deleted

    Test Case Derivation: A user want to delete a quote and all associated chat messages

    How test will be performed: Jest will send a delete quote request with a valid quote ID

    Result: Passed

13. **FRT-M4-13**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

    Input: Delete quote request with an invalid quote ID

    Output: Request is rejected

    Test Case Derivation: A shop owner/employee wants to view the quote

    How test will be performed: Jest will send a delete quote request with an invalid quote ID

    Result: Passed

14. **FRT-M4-14**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

    Input: Create a chat message request with valid information

    Output: A chat message is created

    Test Case Derivation: A user wants to send a chat message

    How test will be performed: Jest will send a create chat message request with valid information

    Result: Passed

15. **FRT-M4-15**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

Input: Create a chat message request with invalid information

Output: Request is rejected

Test Case Derivation: A user wants to send a chat message

How test will be performed: Jest will send a create chat message request with invalid information

Result: Passed

16. **FRT-M4-16**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

    Input: Get chat messages request with a valid quote ID

    Output: List of chat messages is returned

    Test Case Derivation: A user want to see the previous chat messages

    How test will be performed: Jest will send a get chat messages request with a valid quote ID

    Result: Passed

17. **FRT-M4-17**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A quote exists

    Input: Get chat messages request with an invalid quote ID

    Output: Empty list

    Test Case Derivation: A user want to see the previous chat messages

    How test will be performed: Jest will send a get chat messages request with an invalid quote ID

    Result: Passed

6.3 Appointments Module

1. **FRT-M5-1**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A customer, vehicle, shop, shop owner, and service exist

    Input: Create appointment request with valid information

    Output: An appointment and work order is created

    Test Case Derivation: A customer wants to create an appointment at a shop

    How test will be performed: Jest will send a create appointment request with valid information

Result: Passed

2. **FRT-M5-2**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A customer, vehicle, shop, shop owner, and service exist

   Input: Create appointment request with missing information

   Output: Request is rejected

   Test Case Derivation: A customer wants to create an appointment at a shop

   How test will be performed: Jest will send a create appointment request with missing information

   Result: Passed

3. **FRT-M5-3**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A customer, vehicle, shop, shop owner, and service exist

   Input: Create appointment request with an end time before the start time

   Output: Request is rejected

   Test Case Derivation: A customer wants to create an appointment at a shop

   How test will be performed: Jest will send a create appointment request with invalid timing information

   Result: Passed

4. **FRT-M5-4**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: An appointment exists

   Input: Get appointment request with a valid appointment ID

   Output: Appointment data is returned

   Test Case Derivation: A user wants to view an appointment

   How test will be performed: Jest will send a get appointment request with a valid appointment ID

   Result: Passed

5. **FRT-M5-5**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: An appointment exists

   Input: Get appointment request with an invalid appointment ID

   Output: Request is rejected

Test Case Derivation: A user wants to view an appointment

How test will be performed: Jest will send a get appointment request with an invalid appointment ID

Result: Passed

6. **FRT-M5-6**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: An appointment exists

   Input: Get appointments request with a valid shop ID

   Output: List of appointments is returned

   Test Case Derivation: A shop owner/employee wants to view all appointments as their shop

   How test will be performed: Jest will send a get appointments request with a valid shop ID

   Result: Passed

7. **FRT-M5-7**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: An appointment exists

   Input: Get appointments request with an invalid shop ID

   Output: Empty list

   Test Case Derivation: A shop owner/employee wants to view all appointments as their shop

   How test will be performed: Jest will send a get appointments request with an invalid shop ID

   Result: Passed

8. **FRT-M5-8**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: An appointment exists

   Input: Update an appointment request with a valid appointment ID and valid information

   Output: The appointment is updated

   Test Case Derivation: A user wants to update the appointment

   How test will be performed: Jest will send an update appointment request with a valid appointment ID and valid information

   Result: Passed

9. **FRT-M5-9**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: An appointment exists

   Input: Update an appointment request with an invalid appointment ID and valid information

   Output: Request is rejected

   Test Case Derivation: A user wants to update the appointment

   How test will be performed: Jest will send an update appointment request with an invalid appointment ID and valid information

   Result: Passed

10. **FRT-M5-10**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: An appointment exists

    Input: Update an appointment request with a valid appointment ID and invalid information

    Output: Request is rejected

    Test Case Derivation: A user wants to update the appointment

    How test will be performed: Jest will send an update appointment request with a valid appointment ID and invalid information

    Result: Passed

11. **FRT-M5-11**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: An appointment exists

    Input: Update an appointment request with a valid appointment ID and with an end time before the start time

    Output: Request is rejected

    Test Case Derivation: A user wants to update the appointment

    How test will be performed: Jest will send an update appointment request with a valid appointment ID and invalid timing information

    Result: Passed

12. **FRT-M5-12**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: Multiple appointment exists with some having overlapping time slots

Input: Update an appointment request with a valid appointment ID and update status from "PENDING_APPROVAL" to "ACCEPTED"

Output: The appointment is updated and all other overlapping appointment are updated to "REJECTED"

Test Case Derivation: A shop owner/employee wants to accept an appointment

How test will be performed: Jest will send an update appointment request with a valid appointment ID and status as "ACCEPTED"

Result: Passed

13. **FRT-M5-13**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: An appointment exists

    Input: Delete appointment request with a valid appointment ID

    Output: The appointment and work order should be deleted

    Test Case Derivation: A user wants to delete an appointment

    How test will be performed: Jest will send a delete appointment request with a valid appointment ID

    Result: Passed

14. **FRT-M5-14**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: An appointment exists

    Input: Delete appointment request with an invalid appointment ID

    Output: Request is rejected

    Test Case Derivation: A user wants to delete an appointment

    How test will be performed: Jest will send a delete appointment request with a valid appointment ID

    Result: Passed

6.4 Work Orders Module

1. **FRT-M6-1**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A work order exists

    Input: Get work order request with a valid work order ID

    Output: Work order data

    Test Case Derivation: A user wants to view the work order

How test will be performed: Jest will send a get work order request with a valid work order ID

Result: Passed

2. **FRT-M6-2**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A work order exists

   Input: Get work order request with an invalid work order ID

   Output: Request is rejected

   Test Case Derivation: A user wants to view the work order

   How test will be performed: Jest will send a get work order request with an invalid work order ID

   Result: Passed

3. **FRT-M6-3**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A work order exists

   Input: Get work order request with a valid shop ID

   Output: List of work orders

   Test Case Derivation: A shop owner/employee wants to view all the work orders for their shop

   How test will be performed: Jest will send a get work orders request with a valid shop ID

   Result: Passed

4. **FRT-M6-4**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A work order exists

   Input: Get work order request with an invalid shop ID

   Output: Request is rejected

   Test Case Derivation: A shop owner/employee wants to view all the work orders for their shop

   How test will be performed: Jest will send a get work order request with an invalid shop ID

   Result: Passed

5. **FRT-M6-5**

   Type: Functional, Dynamic, Unit, Automated

Initial State: A work order exists

Input: Update work order request with a valid work order ID and valid information

Output: Update the work order

Test Case Derivation: A shop owner/employee wants to update the work order

How test will be performed: Jest will send an update work order request with a valid work order ID and valid information

Result: Passed

6. **FRT-M6-6**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A work order exists

   Input: Update work order request with an invalid work order ID and valid information

   Output: Request is rejected

   Test Case Derivation: A shop owner/employee wants to update the work order

   How test will be performed: Jest will send an update work order request with an invalid work order ID and valid information

   Result: Passed

7. **FRT-M6-7**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A work order exists

   Input: Update work order request with a valid work order ID and invalid information

   Output: Request is rejected

   Test Case Derivation: A shop owner/employee wants to update the work order

   How test will be performed: Jest will send an update work order request with a valid work order ID and invalid information

   Result: Passed

6.5 Employee Management Module

1. **FRT-M7-1**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop account exists with employees

   Input: Get employees with a valid shop ID

   Output: All employees associated with the shop

Test Case Derivation: A shop owner with employees wants to view the list of employees

How test will be performed: Jest will send a get employees request with a valid shop ID

Result: Passed

2. **FRT-M7-2**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop account exists with employees

   Input: Get employees with an invalid shop ID

   Output: Empty list

   Test Case Derivation: A shop owner with employees wants to view the list of employees

   How test will be performed: Jest will send a get employees request with an invalid shop ID

   Result: Passed

3. **FRT-M7-3**

   Control: Manual

   Initial State: Shop owner account with employees

   Input: Update employee status from "ACTIVE" to "SUSPENDED" for a given employee ID

   Output: No operation

   Test Case Derivation: The shop owner with employees wants to remove an employee

   How test will be performed: Jest will send a delete employee request with a valid employee ID

   Result: Passed

4. **FRT-M7-4**

   Control: Manual

   Initial State: Shop owner account with employees

   Input: Update employee status from "ACTIVE" to "SUSPENDED" for a given employee ID

   Output: Request is rejected

   Test Case Derivation: The shop owner with employees wants to remove an employee

   How test will be performed: Jest will send a delete employee request with an invalid employee ID

   Result: Passed

6.6 Services Module

1. **FRT-M8-1**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop and a shop owner exist

   Input: Create service request with valid information

   Output: Create the service

   Test Case Derivation: A shop owner wants to create a service at their shop

   How test will be performed: Jest will send a create service request with valid information

   Result: Passed

2. **FRT-M8-2**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop and a shop owner exist

   Input: Create service request with invalid information

   Output: Request is rejected

   Test Case Derivation: A shop owner wants to create a service at their shop

   How test will be performed: Jest will send a create service request with invalid information

   Result: Passed

3. **FRT-M8-3**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A service exists

   Input: Get service request with a valid service ID

   Output: The service data

   Test Case Derivation: A user wants to view a service

   How test will be performed: Jest will send a get service request with a valid service ID

   Result: Passed

4. **FRT-M8-4**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A service exists

   Input: Get service request with an invalid service ID

   Output: Request is rejected

Test Case Derivation: A user wants to view a service

How test will be performed: Jest will send a get service request with an invalid service ID

Result: Passed

5. **FRT-M8-5**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A service exists

   Input: Get services request with a valid shop ID

   Output: List of services

   Test Case Derivation: A user wants to view the services at a shop

   How test will be performed: Jest will send a get services request with a valid shop ID

   Result: Passed

6. **FRT-M8-6**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A service exists

   Input: Get services request with an invalid shop ID

   Output: Empty list

   Test Case Derivation: A user wants to view the services at a shop

   How test will be performed: Jest will send a get services request with an invalid shop ID

   Result: Passed

7. **FRT-M8-7**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A “CANNED” service exists

   Input: Get “CANNED” services request with a valid shop ID

   Output: List of “CANNED” services

   Test Case Derivation: A user wants to view the “CANNED” services at a shop

   How test will be performed: Jest will send a get “CANNED” services request with a valid shop ID

   Result: Passed

8. **FRT-M8-8**

   Type: Functional, Dynamic, Unit, Automated

Initial State: A “CANNED” service exists

Input: Get “CANNED” services request with an invalid shop ID

Output: Request is rejected

Test Case Derivation: A user wants to view the “CANNED” services at a shop

How test will be performed: Jest will send a get “CANNED” services request with an invalid shop ID

Result: Passed

9. **FRT-M8-9**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A “CUSTOM” service exists

   Input: Get “CUSTOM” services request with a valid shop ID

   Output: List of “CUSTOM” services

   Test Case Derivation: A user wants to view the “CUSTOM” services at a shop

   How test will be performed: Jest will send a get “CUSTOM” services request with a valid shop ID

   Result: Passed

10. **FRT-M8-10**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A “CUSTOM” service exists

    Input: Get “CUSTOM” services request with an invalid shop ID

    Output: Request is rejected

    Test Case Derivation: A user wants to view the “CUSTOM” services at a shop

    How test will be performed: Jest will send a get “CUSTOM” services request with an invalid shop ID

    Result: Passed

11. **FRT-M8-11**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A service exists

    Input: Update a service request with a valid service ID and valid information

    Output: The service is updated

    Test Case Derivation: A shop owner wants to update the details of their service

    How test will be performed: Jest will send an update service request with a valid service ID and valid information

Result: Passed

12. **FRT-M8-12**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A service exists

    Input: Update a service request with an invalid service ID and valid information

    Output: Request is rejected

    Test Case Derivation: A shop owner wants to update the details of their service

    How test will be performed: Jest will send an update service request with an invalid service ID and valid information

    Result: Passed

13. **FRT-M8-13**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A service exists

    Input: Update a service request with a valid service ID and invalid information

    Output: Request is rejected

    Test Case Derivation: A shop owner wants to update the details of their service

    How test will be performed: Jest will send an update service request with a valid service ID and invalid information

    Result: Passed

14. **FRT-M8-14**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A service exists

    Input: Delete a service request with a valid service ID

    Output: The service is deleted

    Test Case Derivation: A shop owner wants to delete their service

    How test will be performed: Jest will send a delete service request with a valid service ID

    Result: Passed

15. **FRT-M8-15**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: A service exists

    Input: Delete a service request with an invalid service ID

    Output: No operation

Test Case Derivation: A shop owner wants to delete their service

How test will be performed: Jest will send a delete service request with an invalid service ID

Result: Passed

6.7 Shop Module

1. **FRT-M9-1**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop exists

   Input: Get shop request with a valid shop ID

   Output: Shop data

   Test Case Derivation: A user wants to view the shop

   How test will be performed: Jest will send a get shop request with a valid shop ID

   Result: Passed

2. **FRT-M9-2**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop exists

   Input: Get shop request with an invalid shop ID

   Output: Request is rejected

   Test Case Derivation: A user wants to view the shop

   How test will be performed: Jest will send a get shop request with an invalid shop ID

   Result: Passed

3. **FRT-M9-3**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop exists

   Input: Update shop request with a valid shop ID and valid information

   Output: Update the shop information

   Test Case Derivation: A shop owner wants to update the shop data information

   How test will be performed: Jest will send an update shop request with a valid shop ID and valid information

   Result: Passed

4. **FRT-M9-4**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop exists

   Input: Update shop request with an invalid shop ID and valid information

   Output: Request is rejected

   Test Case Derivation: A shop owner/employee wants to update the shop

   How test will be performed: Jest will send an update shop request with an invalid shop ID and valid information

   Result: Passed

5. **FRT-M9-5**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: A shop exists

   Input: Update shop request with a valid shop ID and invalid information

   Output: Request is rejected

   Test Case Derivation: A shop owner/employee wants to update the shop

   How test will be performed: Jest will send an update shop request with a valid shop ID and invalid information

   Result: Passed

6. **FRT-M9-6**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: Multiple shops exist

   Input: Get shop request with a valid shop name

   Output: A shop that contains that shop name

   Test Case Derivation: A user wants to search for auto repair shops

   How test will be performed: Jest will send a get shop request with a name that exists in the list of shops

   Result: Passed

7. **FRT-M9-7**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: Multiple shops exist

   Input: Get shop request with a phrase that exists in multiple shop names

   Output: A list of shops that contain that phrase

   Test Case Derivation: A user wants to search for auto repair shops

How test will be performed: Jest will send a get shop request with a phrase that exists in the list of shops

Result: Passed

8. **FRT-M9-8**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: Multiple shops exist

   Input: Get shop request with a shop name that does not exist in the list of shop names

   Output: An empty list

   Test Case Derivation: A user wants to search for auto repair shops

   How test will be performed: Jest will send a get shop request with a name that does not exist in the list of shops

   Result: Passed

9. **FRT-M9-9**

   Type: Functional, Dynamic, Unit, Automated

   Initial State: Multiple shops exist, shops have services

   Input: Get shop request with a service name

   Output: A list of shops that contain that service

   Test Case Derivation: A user wants to search for auto repair shops

   How test will be performed: Jest will send a get shop request with a service that exists in the list of shops

   Result: Passed

10. **FRT-M9-10**

    Type: Functional, Dynamic, Unit, Automated

    Initial State: Multiple shops exist, shops have services

    Input: Get shop request with a service name that does not exist in any shop

    Output: An empty list

    Test Case Derivation: A user wants to search for auto repair shops

    How test will be performed: Jest will send a get shop request with a service that does not exist in the list of shops

    Result: Passed

7 Changes Due to Testing

During testing there were a few changes that were needed to be made to bring the application into alignment with the requirements defined in the SRS and the plan defined in the VnV Plan. Test cases "FRT-BE5-1", "FRT-BE5-2", "NFRT-OE-1", and "NFRT-SR2-1" have failed and changes will be made to address them. Additionally, there were comments made regarding the lack of the ability to edit and cancel appointments in the user interface during the Revision 0 Demonstration. This feedback will also be addressed. All changes are planned to be completed before the Revision 1 Demonstration.

8 Automated Testing

All automated testing was carried out by Jest, a JavaScript testing framework. Each module has its own corresponding test file located within the "test" folder. The "test" folder has the same structure as the "src" folder, creating a one-to-one mapping of a module and its corresponding test. For example, the module "src/server/services/userService.ts" has its tests located in "test/server/services/userService.test.ts". There are two Jest test runs, with the only difference being whether the testing environment is connected to a real database or not (i.e., whether the database needs to be seeded or mocked in each test case). As part of the CI/CD pipeline, the full test suite is run and information regarding pass/fail and code coverage is reported on every pull request to the main branch and to every push to the main branch.

9 Trace to Requirements

10 Trace to Modules

11 Code Coverage Metrics

Code coverage was calculated using Jest's built-in code coverage instrumentation. A tool by the name of "nyc" was used to merge the code coverage reports generated from the multiple runs and output a human-readable HTML report. The code coverage report is documented in Table 3. Code that was not checked by the automated testing was either tested manually or outside the scope of the requirements.

Table 3: Code Coverage Report

Package	**Line Rate**	**Branch Rate**
pages.api.appointment	79%	70%
pages.api.customer.[id]	19%	18%
pages.api.employee	53%	18%
pages.api.quotes	77%	33%
pages.api.quotes.[id]	73%	54%
pages.api.service	74%	61%
pages.api.shop	73%	71%
pages.api.shop.[id]	61%	43%
pages.api.shop.[id].services	65%	71%
pages.api.user.register	76%	42%
pages.api.vehicle	0%	0%
pages.api.work-order	65%	47%
server.services	92%	72%
Summary	67% (616 / 914)	51% (171 / 338)

12 Appendix

12.1 Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Reflection. Please answer the following question:

1. In what ways was the Verification and Validation (VnV) Plan different from the activities that were actually conducted for VnV? If there were differences, what changes required the modification in the plan? Why did these changes occur? Would you be able to anticipate these changes in future projects? If there weren't any differences, how was your team able to clearly predict a feasible amount of effort and the right tasks needed to build the evidence that demonstrates the required quality? (It is expected that most teams will have had to deviate from their original VnV Plan.)