# System Verification and Validation Plan for 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 |
|---|---|---|
| October 26, 2022 | Arkin Modi | Added Software Validation Plan |
| October 27, 2022 | Joy Xiao | Add Testing Team and Design Verification |
| October 27, 2022 | Arkin Modi | Add Automated Testing and Verification Tools |
| October 28, 2022 | Timothy Choy | Add Relevant Documentation |
| October 28, 2022 | Leon So | Add Summary |
| October 28, 2022 | Arkin Modi | Add System Tests for Quotes |
| October 29, 2022 | Leon So | Add Objectives |
| October 29, 2022 | Arkin Modi | Add System Tests for Work Orders |
| October 29, 2022 | Joy Xiao | Add System Tests for Appointments |
| October 30, 2022 | Timothy Choy | Add SRS Verification Plan |
| October 30, 2022 | Leon So | Add Operational and Environmental Tests |
| October 30, 2022 | Arkin Modi | Add Implementation Verification Plan |
| October 30, 2022 | Leon So | Add Usability and Humanity Tests |
| October 30, 2022 | Joy Xiao | Add System Tests for Services |
| October 30, 2022 | Leon So | Add System Tests for Employee Management |
| October 30, 2022 | Arkin Modi | Add Look and Feel System Tests |
| October 30, 2022 | Arkin Modi | Add Security System Tests |
| October 31, 2022 | Timothy Choy | Add System Tests for Shop Lookup |
| October 31, 2022 | Joy Xiao | Add Cultural Requirements Tests |
| October 31, 2022 | Timothy Choy | Add System Tests for Shop Profile |
| October 31, 2022 | Timothy Choy | Add Performance Tests |
| October 31, 2022 | Leon So | Add System Tests for Authentication |
| November 1, 2022 | Timothy Choy | Add System Tests for Employee Profile |
| November 1, 2022 | Timothy Choy | Add Maintainability and Support Tests |
| November 1, 2022 | Timothy Choy | Add VnV Verification Plan |
| November 1, 2022 | Joy Xiao | Add Legal Requirement Tests |
| November 1, 2022 | Arkin Modi | Add Traceability Between Test Cases and Requirements |
| November 2, 2022 | Timothy Choy | Add Introductory Blurbs |
| November 2, 2022 | Arkin Modi | Add reflection |
| November 2, 2022 | Joy Xiao | Add reflection |
| November 2, 2022 | Leon So | Add reflection |
| November 2, 2022 | Timothy Choy | Add reflection |
| March 1, 2023 | Arkin Modi | Add Unit Tests for User Module |
| March 1, 2023 | Arkin Modi | Add Unit Tests for Quotes Module |
| March 1, 2023 | Arkin Modi | Add Unit Tests for Appointments Module |
| March 1, 2023 | Arkin Modi | Add Unit Tests for Work Orders Module |
| March 1, 2023 | Arkin Modi | Add Unit Tests for Services Module |
| March 3, 2023 | Arkin Modi | Add Unit Tests for Nonfunctional Requirements |

| | | |
|---|---|---|
| March 4, 2023 | Joy Xiao | Update Employee Management System Tests and add Unit Tests |
| March 4, 2023 | Timothy Choy | Update Usability and Humanity Tests |
| March 4, 2023 | Arkin Modi | Update Security Requirements Tests |
| March 4, 2023 | Arkin Modi | Add Traceability Between Test Cases and Modules Table |
| March 5, 2023 | Timothy Choy | Update Performance Tests |
| March 5, 2023 | Leon So | Update Shop Profile Tests & Remove Employee Profile Tests |
| March 6, 2023 | Leon So | Update Functional Requirement Tests for Authentication |
| March 6, 2023 | Leon So | Update Functional Requirement Tests for Employee Management |
| March 6, 2023 | Joy Xiao | Remove shop being able to create appointment |
| March 6, 2023 | Timothy Choy | Update Functional Requirement Tests for Shop Lookup |
| March 6, 2023 | Arkin Modi | Update Functional Requirement Tests for Quotes |
| March 6, 2023 | Joy Xiao | Update employee management unit tests |
| March 7, 2023 | Timothy Choy | Update Unit Tests for Shop Module |
| March 7, 2023 | Arkin Modi | Update Functional Requirement Tests for Work Orders |

# Contents

# List of Tables



# List of Figures

# 2 Symbols, Abbreviations and Acronyms

| symbol | description |
|---|---|
| CI | Continuous Integration |
| CLS | Cumulative Layout Shift |
| FCP | First Contentful Paint |
| FID | First Input Delay |
| HA | Hazard Analysis |
| HTTP | Hypertext Transfer Protocol |
| LCP | Largest Contentful Paint |
| MG | Module Guide |
| MIS | Module Interface Specification |
| ms | Milliseconds |
| PWA | Progressive Web Application |
| s | Seconds |
| SRS | Software Requirements Specification |
| TTFB | Time to First Byte |
| VnV | Verification and Validation |

This document outlines the verification and validation of Sayyara. Within this document, there will be plans for verifying the SRS document, the design plan, the VnV plan, the implementation plan, and the software plan. This document will also list the methods of testing that we plan to use. Furthermore, this document will list specific system and unit tests as described in our Functional and Nonfunctional requirements in the SRS, and the modules in the MG and MIS.

# 3 General Information

## 3.1 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 Objectives

The objective of the testing outlined in this document is to demonstrate adequate usability and ensure that the system is in a functional state for the end users. The testing and validation will also aid in ensuring that the product fulfills the system requirements, intended use, and goals of stakeholders.

## 3.3 Relevant Documentation

Below are the relevant documents that will be referred to in the VnV Plan.

- Development Plan (Modi, Xiao, So, and Choy, 2022a)
- Hazard Analysis (Modi, Xiao, So, and Choy, 2022b)
- Module Guide (Modi, Xiao, So, and Choy, 2022c)
- Module Interface Specification (Modi, Xiao, So, and Choy, 2022d)
- Software Requirements Specification (Modi, Xiao, So, and Choy, 2022e)

# 4 Plan

The following section will cover our plan to verify and validate certain components of our project. We will include the members responsible for the verification and validation, as well as our plans to verify the SRS, design plan, VnV plan, implementation plan, automated testing and verification tools, and software plan.

## 4.1 Verification and Validation Team

The verification and validation team will consist of the core developers (Joy Xiao, Tim Choy, Leon So, Arkin Modi), as well as the course instructor and TAs.

The developers are responsible for coming up with tests with suitable edge cases to evaluate the correctness of Sayyara. The developers will all be responsible for writing and executing all test cases listed in the document and taking note of the results. The developers will ensure that Sayyara passes all tests after performing the tests and making any necessary updates.

## 4.2 SRS Verification Plan

SRS verification will be done by the core developers of the project, as well as by the TAs of the course. To verify the contents of the SRS, we will compare the requirements stated by the stakeholder, Nabeel, with the requirements stated in the SRS. Furthermore, we will show our SRS to Nabeel for their review. We believe that this is sufficient to verify the SRS, as Nabeel has defined clear requirements for the project.

## 4.3 Design Verification Plan

Design verification will be done by the core developers of the project. The design will also be reviewed by the TAs of the course. The design of the system will be verified by going through the requirements from the SRS and determining whether the outputs correspond with the expected inputs. The verification will also be done by going through the MG and MIS checklist and ensure that all the modules are completed and fulfill the corresponding requirements.

## 4.4 Verification and Validation Plan Verification Plan

The VnV Plan verification will be done by the core developers of the project, and the TAs of the course. To verify the plan, we will run through the requirements stated in the SRS and match every requirement to at least one test. Regarding the unit tests, we will match each module stated in the MG and MIS to the tests written in the VnV plan.

## 4.5 Implementation Verification Plan

Automated integration tests will be created to fulfill the system tests defined in section 5 and automated unit tests will be created to fulfill the tests defined in section 6. Static analysis will be performed through code review by at least one core team member per pull request and automation tools as stated in 4.6. A pull request will cover at most one module.

Manual System Tests defined in section 5 shall take place from January 16, 2023 to January 30, 2023. The areas of responsibility will be divided as described in table 2.

Due to the large scope of this project and the timeline constraints, the team has chosen to rely more heavily on manual testing instead of investing in learning a new end-to-end testing framework and building a large fully featured test suite.

Table 2: Manual System Testing Responsibilities

| **Area of Testing** | **Team Member Responsible** |
|---|---|
| Authentication | Leon So |
| Appointments | Joy Xiao |
| Quotes | Arkin Modi |
| Work Orders | Arkin Modi |
| Employee Management | Leon So |
| Services | Joy Xiao |
| Shop Lookup | Timothy Choy |
| Shop Profile | Timothy Choy |
| Employee Profile | Timothy Choy |

## 4.6 Automated Testing and Verification Tools

All tooling and automations that will be used for testing and verification have been documented in the Development Plan under the "Technology" and "Code Standard" sections.

As part of our CI, we will have a pipeline running the automated tests against every commit to the main branch and every pull request opened against the main branch that contains a source code or testing code change. There will also be a pipeline running against every pull request asserting adherence to the documented code style standards. Code coverage reports will be commented directly to the pull request or the commit to the main branch using a pipeline.

## 4.7 Software Validation Plan

The plan for validating the software and the requirements shall be to conduct review session with the stakeholders. These review sessions shall focus on the business events and user flows as defined in the SRS.

# 5 System Test Description

## 5.1 Tests for Functional Requirements

The following system tests are for functional requirements, as defined in the SRS. To ensure all requirements are covered, the subsections for the VnV plan match the subsections in the SRS. Furthermore, each test case will have a label linked to their respective business event.

### 5.1.1 Authentication

This section will contains tests covering the "Authentication" requirements defined in the SRS.

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

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

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

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

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

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

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

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

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

### 5.1.2 Appointments

This section will contain tests covering the "Appointments" requirements defined in the SRS.

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

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

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

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

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

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

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

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

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

### 5.1.3 Quotes

This section will contain tests covering the "Quotes" requirements defined in the SRS.

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

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

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

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

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

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

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

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

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

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

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

### 5.1.4 Work Orders

This section will contain tests covering the "Work Orders" requirements defined in the SRS.

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

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

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

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

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

### 5.1.5 Employee Management

This section will contain tests covering the "Employee Management" requirements defined in the SRS.

1. **FRT-BE17-1**

   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.

2. **FRT-BE17-2**

   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

3. **FRT-BE18-1**

   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

4. **FRT-BE19-1**

   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

5. **FRT-BE20-1**

   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

#### 5.1.6 Services

This section will contain tests covering the "Services" requirements defined in the SRS.

1. **FRT-BE21-1**

   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 on the shop profile

2. **FRT-BE22-1**

   Control: Manual

   Initial State: Shop Owner/Shop Employee searches for a service

   Input: Enters specific auto service in the search bar

   Output: Services matching the search criteria 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

3. **FRT-BE23-1**

   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

4. **FRT-BE24-1**

   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

### 5.1.7 Shop Lookup

This section will contain tests covering the "Shop Lookup" requirements defined in the SRS.

1. **FRT-BE25-1**

   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

### 5.1.8 Shop Profile

This section will contain tests covering the "Shop Profile" requirements defined in the SRS.

1. **FRT-BE26-1**

   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, 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

2. **FRT-BE27-1**

   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

3. **FRT-BE28-1**

   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 profile

## 5.2 Tests for Nonfunctional Requirements

### 5.2.1 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

2. **NFRT-LF2-1**

   Type: Dynamic, Manual

   Initial State: The application is accessible through the Google Chrome 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

3. **NFRT-LF3-1**

   Type: Dynamic, Manual

   Initial State: The application is accessible through the Google Chrome 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

### 5.2.2 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 listed section 5.1 on a macOS desktop/laptop device, a Windows desktop/laptop, an iOS mobile device, and an android mobile device.

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.

3. **NFRT-UH3-1**

   Type: Dynamic, Manual

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

### 5.2.3 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
   - TTFP $<$ 0.5s

- FCP $< 1.8$s

### 5.2.4 Operational and Environmental Requirements

1. **NFRT-OE1-1**

   Type: Dynamic, Manual

   How test will be performed: The testers will complete the manual system tests for functional requirements listed in section 5.1 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 ($390 \times 844$), iPhone SE ($375 \times 667$), and iPad Air ($820 \times 1180$). This set will provide a reasonable variety of different device dimensions.

### 5.2.5 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.

### 5.2.6 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

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 available time slots at the shop 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

### 5.2.7 Cultural Requirements

1. **NFRT-CR1-1**

   Type: Static, Manual

   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.

2. **NFRT-CR2-1**

   Type: Static, Manual

   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.

### 5.2.8 Legal Requirements

1. **NFRT-LR1-1**

   Type: Static, Manual

   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.

## 5.3 Traceability Between Test Cases and Requirements

Table 3: Traceability Between Functional Test Cases and Functional Requirements, BE1 to BE16

| **Test IDs** | **Functional Requirement IDs** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **BE1** | **BE2** | **BE3** | **BE4** | **BE5** | **BE6** | **BE7** | **BE8** | **BE9** | **BE10** | **BE11** | **BE12** | **BE13** | **BE14** | **BE15** | **BE16** |
| **FRT-BE1-1** | X | | | | | | | | | | | | | | | |
| **FRT-BE1-2** | X | | | | | | | | | | | | | | | |
| **FRT-BE1-3** | X | | | | | | | | | | | | | | | |
| **FRT-BE2-1** | | X | | | | | | | | | | | | | | |
| **FRT-BE2-2** | | X | | | | | | | | | | | | | | |
| **FRT-BE2-3** | | X | | | | | | | | | | | | | | |
| **FRT-BE3-1** | | | X | | | | | | | | | | | | | |
| **FRT-BE3-2** | | | X | | | | | | | | | | | | | |
| **FRT-BE3-3** | | | X | | | | | | | | | | | | | |
| **FRT-BE4-1** | | | | X | | | | | | | | | | | | |
| **FRT-BE4-2** | | | | X | | | | | | | | | | | | |
| **FRT-BE4-3** | | | | X | | | | | | | | | | | | |
| **FRT-BE5-1** | | | | | X | | | | | | | | | | | |
| **FRT-BE5-2** | | | | | X | | | | | | | | | | | |
| **FRT-BE5-3** | | | | | X | | | | | | | | | | | |
| **FRT-BE6-1** | | | | | | X | | | | | | | | | | |
| **FRT-BE6-2** | | | | | | X | | | | | | | | | | |
| **FRT-BE7-1** | | | | | | | X | | | | | | | | | |
| **FRT-BE8-1** | | | | | | | | X | | | | | | | | |
| **FRT-BE8-2** | | | | | | | | X | | | | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **FRT-BE8-3** | | | | | | | X | | | | | | | | |
| **FRT-BE8-4** | | | | | | | X | | | | | | | | |
| **FRT-BE9-1** | | | | | | | | X | | | | | | | |
| **FRT-BE10-1** | | | | | | | | | X | | | | | | |
| **FRT-BE10-2** | | | | | | | | | X | | | | | | |
| **FRT-BE11-1** | | | | | | | | | | X | | | | | |
| **FRT-BE11-2** | | | | | | | | | | X | | | | | |
| **FRT-BE12-1** | | | | | | | | | | | X | | | | |
| **FRT-BE13-1** | | | | | | | | | | | | X | | | |
| **FRT-BE14-1** | | | | | | | | | | | | | X | | |
| **FRT-BE14-2** | | | | | | | | | | | | | X | | |
| **FRT-BE15-1** | | | | | | | | | | | | | | X | |
| **FRT-BE15-2** | | | | | | | | | | | | | | X | |
| **FRT-BE16-1** | | | | | | | | | | | | | | | X |

Table 4: Traceability Between Functional Test Cases and Functional Requirements, BE17 to BE28

| **Test IDs** | **Functional Requirement IDs** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **BE17** | **BE18** | **BE19** | **BE20** | **BE21** | **BE21** | **BE23** | **BE24** | **BE25** | **BE26** | **BE27** | **BE28** |
| **FRT-BE17-1** | X | | | | | | | | | | | |
| **FRT-BE17-2** | X | | | | | | | | | | | |
| **FRT-BE18-1** | | X | | | | | | | | | | |
| **FRT-BE19-1** | | | X | | | | | | | | | |
| **FRT-BE20-1** | | | | X | | | | | | | | |
| **FRT-BE21-1** | | | | | X | | | | | | | |
| **FRT-BE22-1** | | | | | | X | | | | | | |
| **FRT-BE23-1** | | | | | | | X | | | | | |
| **FRT-BE24-1** | | | | | | | | X | | | | |
| **FRT-BE25-1** | | | | | | | | | X | | | |
| **FRT-BE26-1** | | | | | | | | | | X | | |
| **FRT-BE27-1** | | | | | | | | | | | X | |
| **FRT-BE28-1** | | | | | | | | | | | | X |

Table 5: Traceability Between Nonfunctional Test Cases and Nonfunctional Requirements

| **Test IDs** | **Functional Requirement IDs** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **LF1** | **LF2** | **LF3** | **UH1** | **UH2** | **UH3** | **PR1** | **OE1** | **MS1** | **SR1** | **SR2** | **CR1** | **CR2** | **LR1** |
| **NFRT-LF1-1** | X | | | | | | | | | | | | | |
| **NFRT-LF2-1** | | X | | | | | | | | | | | | |
| **NFRT-LF3-1** | | | X | | | | | | | | | | | |
| **NFRT-UH1-1** | | | | X | | | | | | | | | | |
| **NFRT-UH2-1** | | | | | X | | | | | | | | | |
| **NFRT-UH3-1** | | | | | | X | | | | | | | | |
| **NFRT-PR1-1** | | | | | | | X | | | | | | | |
| **NFRT-OE1-1** | | | | | | | | X | | | | | | |
| **NFRT-MS1-1** | | | | | | | | | X | | | | | |
| **NFRT-SR1-1** | | | | | | | | | | X | | | | |
| **NFRT-SR2-1** | | | | | | | | | | | X | | | |
| **NFRT-CR1-1** | | | | | | | | | | | | X | | |
| **NFRT-CR2-1** | | | | | | | | | | | | | X | |
| **NFRT-LR1-1** | | | | | | | | | | | | | | X |

# 6 Unit Test Description

## 6.1 Unit Testing Scope

All modules defined in the Module Interface Specification are within the scope testing with exception for the Database Driver Module. This module is largely composed of generated types and logic from a third party library, Prisma. It assumed that Prisma's logic is correct and therefore shall not be tested.

## 6.2 Tests for Functional Requirements

### 6.2.1 Users Module

This section will contain tests covering the "Users Module" defined in the MIS. These tests were chosen based on common user flows and internal execution patterns.

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

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

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

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

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

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

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

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

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

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

### 6.2.2 Quotes Module

This section will contain tests covering the "Quotes Module" defined in the MIS. These tests were chosen based on common user flows.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 6.2.3 Appointments Module

This section will contain tests covering the "Appointments Module" defined in the MIS. These tests were chosen based on common user flows.

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

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

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

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

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

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

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

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

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

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

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

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"

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

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

### 6.2.4 Work Orders Module

This section will contain tests covering the "Work Orders Module" defined in the MIS. These tests were chosen based on common user flows.

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

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

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

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

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

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

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

### 6.2.5 Employee Management Module

This section will contain tests covering the "Employee Management" requirements defined in the SRS. These tests were chosen based on common user flows.

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

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

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

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

### 6.2.6 Services Module

This section will contain tests covering the "Services Module" defined in the MIS. These tests were chosen based on common user flows.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 6.2.7 Shop Module

This section will contain tests covering the "Shop Module" defined in the MIS. These tests were chosen based on common user flows.

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

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

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

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

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

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

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

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

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

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

## 6.3 Tests for Nonfunctional Requirements

The application will not have nonfunctional requirements tested for through the use of unit testing. All nonfunctional requirements will be verified through the system tests documented in section 5.2.

## 6.4 Traceability Between Test Cases and Modules

Table 6: Traceability Between Test Cases and Modules

| Test IDs | Module IDs | | | | | | |
|---|---|---|---|---|---|---|---|
| | M3 | M4 | M5 | M6 | M7 | M8 | M9 |
| **FRT-M3-1** | X | | | | | | |
| **FRT-M3-2** | X | | | | | | |
| **FRT-M3-3** | X | | | | | | |
| **FRT-M3-4** | X | | | | | | |
| **FRT-M3-5** | X | | | | | | |
| **FRT-M3-6** | X | | | | | | |
| **FRT-M3-7** | X | | | | | | |
| **FRT-M3-8** | X | | | | | | |
| **FRT-M3-9** | X | | | | | | |
| **FRT-M3-10** | X | | | | | | |
| **FRT-M4-1** | | X | | | | | |
| **FRT-M4-2** | | X | | | | | |
| **FRT-M4-3** | | X | | | | | |
| **FRT-M4-4** | | X | | | | | |
| **FRT-M4-5** | | X | | | | | |
| **FRT-M4-6** | | X | | | | | |
| **FRT-M4-7** | | X | | | | | |
| **FRT-M4-8** | | X | | | | | |
| **FRT-M4-9** | | X | | | | | |
| **FRT-M4-10** | | X | | | | | |
| **FRT-M4-11** | | X | | | | | |
| **FRT-M4-12** | | X | | | | | |
| **FRT-M4-13** | | X | | | | | |
| **FRT-M4-14** | | X | | | | | |
| **FRT-M4-15** | | X | | | | | |
| **FRT-M4-16** | | X | | | | | |
| **FRT-M4-17** | | X | | | | | |
| **FRT-M5-1** | | | X | | | | |
| **FRT-M5-2** | | | X | | | | |
| **FRT-M5-3** | | | X | | | | |
| **FRT-M5-4** | | | X | | | | |
| **FRT-M5-5** | | | X | | | | |
| **FRT-M5-6** | | | X | | | | |
| **FRT-M5-7** | | | X | | | | |
| **FRT-M5-8** | | | X | | | | |

| | | | | | |
|---|---|---|---|---|---|
| **FRT-M5-9** | | X | | | |
| **FRT-M5-10** | | X | | | |
| **FRT-M5-11** | | X | | | |
| **FRT-M5-12** | | X | | | |
| **FRT-M5-13** | | X | | | |
| **FRT-M5-14** | | X | | | |
| **FRT-M6-1** | | | X | | |
| **FRT-M6-2** | | | X | | |
| **FRT-M6-3** | | | X | | |
| **FRT-M6-4** | | | X | | |
| **FRT-M6-5** | | | X | | |
| **FRT-M6-6** | | | X | | |
| **FRT-M6-7** | | | X | | |
| **FRT-M7-1** | | | | X | |
| **FRT-M7-2** | | | | X | |
| **FRT-M7-3** | | | | X | |
| **FRT-M7-4** | | | | X | |
| **FRT-M8-1** | | | | | X |
| **FRT-M8-2** | | | | | X |
| **FRT-M8-3** | | | | | X |
| **FRT-M8-4** | | | | | X |
| **FRT-M8-5** | | | | | X |
| **FRT-M8-6** | | | | | X |
| **FRT-M8-7** | | | | | X |
| **FRT-M8-8** | | | | | X |
| **FRT-M8-9** | | | | | X |
| **FRT-M8-10** | | | | | X |
| **FRT-M8-11** | | | | | X |
| **FRT-M8-12** | | | | | X |
| **FRT-M8-13** | | | | | X |
| **FRT-M8-14** | | | | | X |
| **FRT-M8-15** | | | | | X |
| **FRT-M9-1** | | | | | X |
| **FRT-M9-2** | | | | | X |
| **FRT-M9-3** | | | | | X |
| **FRT-M9-4** | | | | | X |
| **FRT-M9-5** | | | | | X |
| **FRT-M9-6** | | | | | X |
| **FRT-M9-7** | | | | | X |
| **FRT-M9-8** | | | | | X |

| | |
|---|---|
| **FRT-M9-9** | X |
| **FRT-M9-10** | X |

# References


Arkin Modi, Joy Xiao, Leon So, and Timothy Choy. Development plan. https://github.com/arkinmodi/project-sayyara/blob/main/docs/DevelopmentPlan/DevelopmentPlan.pdf, 2022a.

Arkin Modi, Joy Xiao, Leon So, and Timothy Choy. Hazard analysis. https://github.com/arkinmodi/project-sayyara/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf, 2022b.

Arkin Modi, Joy Xiao, Leon So, and Timothy Choy. Module guide. https://github.com/arkinmodi/project-sayyara/blob/main/docs/MG/MG.pdf, 2022c.

Arkin Modi, Joy Xiao, Leon So, and Timothy Choy. Module interface specification. https://github.com/arkinmodi/project-sayyara/blob/main/docs/MIS/MIS.pdf, 2022d.

Arkin Modi, Joy Xiao, Leon So, and Timothy Choy. Software requirements specification. https://github.com/arkinmodi/project-sayyara/blob/main/docs/SRS/SRS.pdf, 2022e.

# 7 Appendix

## 7.1 Reflection

**The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:**

1. **What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.**

   The team members will need to collectively acquire a set of unique skills to successfully complete the verification and validation of Sayyara. Some skills and knowledge outlined below are critical not only for the delivery of the project milestones, but also in the ensuring the application is delivered in a fully functional state that satisfies both the requirements and goals for this system. These skills will also contribute to each individual team member's success moving forward in their software engineering careers.

   1. Domain knowledge and skills specific to testing of PWAs
   2. Acquire knowledge and skills related to testing tools: Jest, React Testing Library
   3. Gain an understanding of language specific type checking (i.e., TypeScript)
   4. Familiarize with Code Review process and best practices
   5. Develop a better understanding of dynamic and static testing methods pertaining to testing a full stack progressive web application
   6. Acquire knowledge relating to the best industry practices and methodologies for manual integration testing of progressive web applications

2. **For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?**

   **Arkin Modi**

   Chosen Skills: 1., 2., 3., 5.

   All of Arkin's chosen skill pertain to domain knowledge and best practices. These chosen skills are areas of passion for Arkin as he has a deep interesting in learning about best practices and how he can improve his day-to-day development skills. For skills based on tools, Arkin will be relying on YouTube tutorials and the tool specific documentation to fill in gaps in his knowledge. For the skill relating to more abstract topics (e.g., testing best practices), he will be utilizing YouTube videos and online articles discussing the many ways these topics can be approached. These methods

are what have work best for Arkin in the past, and he already has an understanding on where to look for the missing information.

**Joy Xiao**

Chosen Skills: 1., 2.

Joy has chosen to learn the domain knowledge and skills for testing PWAs. Learning more about how to test PWAs is essential to the project because it is one of the main features of our application. This will be done by accessing various online resources by searching how to test PWAs and looking through the documentation and any informational videos to learn how it works from end to end. Joy will also acquire knowledge and skills related to testing tools such as Jest and the React Testing library by looking through official documentation from these resources and supplementing by any video demonstration resources when needed. By learning each of these areas, it would be very beneficial for performing the testing of our project as well as skills for future career growth as many of the concepts can be used for many different situations.

**Leon So**

Chosen Skills: 1., 2., 5., 6.

Leon has chosen to improve and develop skills related to the testing of PWAs. In specific, Leon will need to acquire skills related to the testing tools and technologies used for this project. This can be done by referring to online resources and documentation related to testing Progressive Web Applications and the technologies used. To develop a better understanding of dynamic and static testing methods pertaining to testing a full stack progressive web application, Leon will refer to course and academic materials and fill any PWA specific gaps with research from online sources. Similarly, for industry best practices and methodologies, Leon will seek guidance from official documentation published by various industry companies, institutions, and sources with known experience working with PWAs. Each of the above development areas will contribute to building a set of knowledge, skills, and understanding that Leon can directly apply to his future work and career. These identified knowledge and skill areas will also allow Leon to contribute effectively to the overall success of the project.

**Timothy Choy**

Chosen Skills: 1., 2., 4., 6.

Timothy has chosen to pursue the skills stated in the list above. He made this choice because he does not have a lot of expertise with testing PWAs as well as TypeScript in general, and would like to add these testing skills and tools to his toolbox. Furthermore, he wants to gain knowledge of how the industry tests, both through code reviews and manual testing, and would like to replicate how it is done in a professional setting in this project. To approach acquiring these skills and knowledge, Timothy will do research on PWA testing and the tools used for testing, such as Jest and the React testing library. Research will be done through reading documentation and watching video tutorials. To get a sense of how the industry performs code reviews and manual testing, Timothy will reach out to

project members who have had more experience in a professional setting (through internships), as well as reach out to his coworkers at his past internship for ideas.