Verification and Validation Report: Sayyara

Team 31
SFWRENG 4G06
Christopher Andrade
Alyssa Tunney
Kai Zhu
Ethan Vince-Budan
Collin Kan
Harsh Gupta

April 5, 2023

1 Revision History

Date	Version	Notes
March 3, 2023	Rev 0	Added testing results for NFRs
March 5, 2023	Rev 0	Added sections for Unit Testing and Automated Testing
March 8, 2023	Rev 0	Added testing reults for Look/Feel and Usability NFRs

2 Symbols, Abbreviations and Acronyms

symbol	**description**
DDoS	Distributed Denial-of-service; when multiple systems flood the bandwidth or resources of a targeted system
SessionID	A unique number that a web site's server assigns a signed in user for the duration of the user's visit
CSRF Token	Unique, secret value generated server-side that is shared with the client. The client must include this CSRF token when submitting a form, etc.
E2E	End-to-end

Contents

List of Tables

List of Figures

3 Introduction

This document outlines the steps that have been taken to test and verify the requirements of the Sayyara application, and follow the general plans from the VnV Plan document.

4 Testing Methodology

The web application architecture consists of a Django backend and NextJS frontend, deployed on Docker with an Nginx proxy routing the endpoints respectively. For backend unit testing, the pytest framework is utilized to write and execute unit tests for individual components or units of code in isolation. The code coverage of the unit tests is also tracked using the coverage.py.

For frontend unit testing as well as end-to-end testing, Playwright is used. This Node.js library provides a high-level API to interact with web browsers, simulating user interactions and verifying expected behavior. By combining backend unit testing with frontend unit and end-to-end testing, the entire system can be tested comprehensively, from the database to the user interface, helping to identify issues early in the development process and reduce the cost and time required for debugging and fixing issues later on.

5 Functional Requirements Evaluation

5.1 Account Requirements

5.1.1 Registration

1. Id: FR-AR1-1
   Req: AuR1
   Description: registration of new account with valid account details
   Init State:

   - Empty account registration form is displayed.
   - Username TestUser does not exist in account database.

   Input:

   - username: TestUser
   - password: somePassword
   - first name: John

- last name: Doe
- email: sayyara.test@gmail.com
- phone: 5551234567
- role: customer
- click Submit button

Expected Output: Redirect to login page. New user exists in database with all fields matching input.
Actual Output:

- login view is displayed
- TestUser exists in account database
- TestUser.first_name: John
- TestUser.last_name: Doe
- TestUser.email: sayyara.test@gmail.com
- TestUser.phone: 555-123-4567
- role: customer

Pass/Fail: Passed

2. Id: FR-AR1-2
Req: AuR1
Description: attempt registration of duplicate username
Init State:

   - Empty account registration form is displayed.
   - Username TestUser already exists in account database (details match input in test FR-AR1-1).

   Input:

   - username: TestUser
   - password: somePassword
   - first name: John
   - last name: Doe
   - email: sayyara.test@gmail.com
   - phone: 1112223333
   - role: customer
   - click Submit button

   Expected Output: Stay on registration page, display duplicate username error, no change to account database.
   Actual Output:

- no redirection occurred
- registration form displayed duplicate username error
- TestUser.phone: 555-123-4567

Pass/Fail: Passed

3. Id: FR-AR1-3
Req: AuR1
Description: attempt registration using invalid password
Init State:

   - Empty account registration form is displayed.
   - Username TestUser does not exists in account database.

   Input:

   - username: TestUser
   - password: 123
   - first name: John
   - last name: Doe
   - email: sayyara.test@gmail.com
   - phone: 1112223333
   - role: customer
   - click Submit button

   Expected Output: Stay on registration page, display password format error, no change to account database.
   Actual Output:

   - no redirection occurred
   - registration form displayed invalid password format error
   - TestUser does not exist in account database

   Pass/Fail: Passed

5.1.2 Login

1. Id: FR-AR2-1
Req: AuR2
Description: login using valid password and password
Init State:

   - Empty login form is displayed.
   - Customer TestUser with password "somePassword" exists in database

Input:

- username: TestUser
- password: somePassword
- click Login button

Expected Output: Receive access and refresh JWT tokens, redirect to homepage.
Actual Output:

- access token received in API fetch response
- refresh token received in API fetch response
- homepage is displayed with greeting message for TestUser

Pass/Fail: Passed

2. Id: FR-AR2-2A
   Req: AuR2
   Description: attempt login using the wrong password
   Init State:

   - Empty login form is displayed.
   - Customer TestUser with password "somePassword" exists in database

   Input:

   - username: TestUser
   - password: wrongPassword
   - click Login button

   Expected Output: Stay on login page, display invalid username/password error.
   Actual Output:

   - no redirection occurred
   - login form displays username/password error
   - no token received in API response

   Pass/Fail: Passed

3. Id: FR-AR2-2B
   Req: AuR2
   Description: attempt login to non-existent user
   Init State:

   - Empty login form is displayed.
   - TestUser does not exists in account database

Input:

- username: TestUser
- password: somePassword
- click Login button

Expected Output: Stay on login page, display invalid username/password error.
Actual Output:

- no redirection occurred
- login form displays username/password error
- no token received in API response

Pass/Fail: Passed

4. Id: FR-AR2-3
   Req: AuR3
   Description: unauthorized attempt to change password
   Init State:

   - TestUser with password "somePassword" exists in account database
   - command prompt initialized

   Input: CURL –request POST –url http://localhost/api/auth/users/set_password/ –header 'Content-Type: application/json' –data ' "current_password":"password", "new_password":"newPassword"' Expected Output: 401 unauthorized error.
   Actual Output:

   - 401 Unauthorized error

   Pass/Fail: Passed

5.1.3 User Profile

1. Id: FR-PAR3-1
   Req: PAR3
   Description: Editting user profile
   Init State:

   - Profile page is displayed
   - Customer TestUser exists in database
   - Logged in as TestUser

   Input:

   - click Edit button

- last name: Dow
- phone: 555-123-4444
- click Update button

Expected Output: Profile page displays updated information
Actual Output:

- last name: Dow
- phone: 555-123-4444
- all other fields: unchanged

Pass/Fail: Passed

2. Id: FR-PAR3-2
   Req: PAR3
   Description: Editting user profile with invalid details
   Init State:

   - Profile page is displayed
   - Customer TestUser exists in database
   - Logged in as TestUser

   Input:

   - click Edit button
   - first name: <blank>
   - phone: 555
   - click Update button

   Expected Output: Profile page stays in edit mode, displays invalid form fields error
   Actual Output:

   - silently failed. No error was displayed
   - profile page exited edit mode
   - TestUser profile is unchanged

   Pass/Fail: Failed

5.2 Quote Requirements

5.2.1 Summary of Test Cases

Id	Reference Requirement	Summary	Expected Output	Pass/Fail
FR-QR1-1.1	QR1	Vehicle owner sends a message on quote request	Message visible for both vehicle owner and shop owner/employees	PASS
FR-QR1-1.2	QR1	Shop owner sends a message on quote request	Message visible for both vehicle owner and shop owner/employee	PASS
FR-QR2-1.1	QR2, QR3	User selects the "New Quote Request" button	Modal for setting up a new quote is displayed	PASS
FR-QR2-1.2	QR2, QR3	User fills in quote request form with *correct/valid* data and submits the form	A new quote request is created, stored and becomes viewable by selected shops	PASS
FR-QR2-1.3	QR2, QR3	User fills in quote request form with *incorrect/invalid* data and attempts to submit the form	User is shown an error message indicating which part(s) of their input are invalid	PASS

Table 1: Summary of tests for functional requirements from section 10.1.6 of the SRS

5.2.2 Individual Test Cases

1. **FR-QR1-1.1**
   Requirements: QR1
   Description: Vehicle owner sending a message on a quote request
   Initial State/Setup:

   - Authenticated as vehicle owner

- Quote request successfully created

Input:

- Expand quote request and select shop from list of pending responses
- Enter message "Test Message"
- Press "Submit reply" button

Expected output: Message saved to quote request conversation, message visible to both vehicle owner and shop owner/employees
Actual output:

- Vehicle owner's view updates to display new message
- Shop owner/employee's view automatically refreshes in order to display new message

Result: PASS

2. **FR-QR1-1.2**
Requirements: QR1
Description: Shop owner sending a message on a quote request
Initial State/Setup:

- Authenticated as shop owner
- Quote request created and sent to owner's shop

Input:

- Select quote from list
- Enter message "Test Reply"
- Press "Submit reply" button

Expected output: Message saved to quote request conversation, message visible to both vehicle owner and shop owner/employees
Actual output:

- Shop owner's view updates to display new message
- Vehicle owner's view automatically refreshes in order to display new message

Result: PASS

3. **FR-QR2-1.1**
   Requirements: QR2, QR3
   Description: Accessing quote adding functionality
   Initial State/Setup:

   - Authenticated as vehicle owner
   - Navigate to the Quotes page

   Input:

   - Select the "New Quote Request" button

   Expected output: Modal for setting up a new quote is displayed
   Actual output: Modal for setting up a new quote is displayed
   Result: PASS

4. **FR-QR2-1.2**
   Requirements: QR2, QR3
   Description: Creating a quote request (normal input)
   Initial State/Setup:

   - Authenticated as vehicle owner
   - Account has at least 1 associated vehicle
   - Navigate to the Quotes page
   - Select the "New Quote Request" button

   Input:

   - Vehicle for servicing: 1st option in list (vehicle associated with account)
   - Description: "Normal Description"
   - Additional photo information: N/A (**TODO**)
   - Part preference: New
   - Available shops: Every option selected
   - Availability: "Monday 12:00 - 15:00"
   - Click "Submit quote request" button

Expected output:

- New quote request matching the given information is created
- New quote is now visible on the Quotes page
- Modal is automatically closed to prevent re-submission

Actual output:

- Modal is automatically closed
- Quote appears in list for customer
- Quote appears in list for shop owners & employees

Result: PASS

5. **FR-QR2-1.3**
   Requirements: QR2, QR3
   Description: Creating a quote request (abnormal input)
   Initial State/Setup: See FR-QR2-1.2
   Input:

   - Vehicle for servicing: 1st option in list (vehicle associated with account)
   - Description: "Abnormal Description"
   - Additional photo information: N/A (**TODO**)
   - Part preference: Used
   - Available shops: None selected
   - Availability: " " (Empty string)
   - Click "Submit quote request" button

   Expected output: Quote request creation fails, user is notified, modal remains open
   Actual output:

   - Modal window remains open
   - No new quote request is created
   - Message displayed to user indicating the location & type of error

   Result PASS

5.3 Vehicle Requirements

5.3.1 Summary of Test Cases

Id	Reference Requirement	Summary	Expected Output	Pass/Fail
FR-VE1-1.1	PAR5	User selects the "New Vehicle" button	Modal for creating a new vehicle is displayed	PASS
FR-VE1-1.2	PAR5	User adds a vehicle to their account	New vehicle is created and linked to user	PASS
FR-VE1-1.3	PAR5	User removes a vehicle from their account (normal case)	Vehicle deleted and no longer visible on profile	FAIL
FR-VE1-1.4	PAR5	User removes a vehicle from their account (abnormal case)	Error message is displayed, and vehicle remains	FAIL

Table 2: Summary of test cases for Vehicles module

5.3.2 Individual Test Cases

1. **FR-VE1-1.1**
   Requirements: PAR5
   Description: User selects the "New Vehicle" button
   Initial State/Setup:

   - Authenticated as Vehicle owner
   - Profile page is selected
   - No prior vehicles added to account

   Input:

   - Click "Add vehicle" button

Expected output: Modal for creating a new vehicle is displayed
Actual output: Modal for creating a new vehicle is displayed
Result: PASS

2. **FR-VE1-1.2**
   Requirements: PAR5
   Description: User adds a new vehicle to their account
   Initial State/Setup:

   - Authenticated as vehicle owner
   - Profile page is selected

   Input:

   - Click "Add vehicle" button
   - Make: Toyota
   - Model: Corolla
   - Year: 2009
   - Plate number: "ABC 123"
   - VIN: "2HSABC3489AIBF76H"
   - Click "Submit" button

   Expected output:

   - Add vehicle modal is automatically closed to avoid re-submission
   - Vehicle with matching information is added to user account
   - Vehicle is visible on profile page

   Actual output:

   - Vehicle modal is closed automatically
   - Page must be manually refreshed in order for new vehicle to appear on profile

   Result: PASS

3. **FR-VE1-1.3**
   Requirements: PAR5
   Description: User removes a vehicle from their account (normal case)
   Initial state/setup:

- Authenticated as vehicle owner
- At least one vehicle created and associated with current account
- That vehicle must *not* be associated with any quote requests/quotes
- Profile page is selected

Input:

- Click "Remove Vehicle" Button on vehicle component

Expected output: Vehicle is removed from user account, and is no longer visible from profile page Actual output: No method to delete vehicles is currently present on the profile page
Result: FAIL

4. **FR-VE1-1.4**
Requirements: PAR5
Description: User attempts to remove a vehicle from their account, which is attached to at least one quote request, quote or appointment
Initial state/setup:

- Authenticated as vehicle owner
- One vehicle created and associated with current account
- At least one quote request created using current user's vehicle
- Profile page is selected

Input:

- Click "Remove Vehicle" button on vehicle component

Expected output:

- Error message is displayed explaining to user why vehicle could not be removed
- Vehicle remains attached to user account

Actual output: No method to delete vehicles is currently present on the profile page
Result: FAIL

5.4 Appointment Requirements

5.4.1 Summary of Test Cases

Id	Reference Requirement	Summary	Expected Output	Pass/Fail
FR-AP1-1.1	APR1	Shop owners should be able to change their shop's availability settings for appointments on the calendar.	The shop owner should be able to successfully change the availability settings for appointments.	PASS
FR-AP2-1.1	APR2	All types of users should be able to schedule appointments to service a vehicle.	The user should be able to successfully schedule an appointment to service a vehicle.	PASS
FR-AP4-1.1	APR4	All users should be able to optionally include a note in the appointment booking for organizational purposes.	The user should be able to add a note to the appointment booking.	PASS
FR-AP5-1.1	APR5	Customers should be able to interact with a virtual AI assistant to assist the user in scheduling an appointment.	The AI assistant should be able to help the user schedule an appointment.	FAIL

Table 3: Test Cases

5.4.2 Individual Test Cases

1. **FR-AP1-1.1**
   Requirements: APR1

Description: Test that shop owners can change their shop's availability settings for appointments on the calendar.
Initial State/Setup:

- Shop owner is logged into their account

Input:

- Shop owner changes the availability settings for the day
- Inputs:
  - Quote
  - Title
  - Description
  - Time Estimate
  - Price Estimate
  - Message
  - Reservation

Expected output: The shop owner should be able to successfully change the availability settings for appointments.
Actual output: The shop owner is able to successfully change the availability settings for appointments.
Result: PASS

2. **FR-AP2-1.1**
   Requirements: APR2
   Description: Test that shops can schedule appointments to service a vehicle.
   Initial State/Setup:

   - User is logged into their account

   Input:

   - User selects an available appointment time

   Expected output: The user should be able to successfully schedule an appointment to service a vehicle.
   Actual output: The user is able to successfully schedule an appointment to service a vehicle.
   Result: PASS

3. **FR-AP4-1.1**
   Requirements: APR4
   Description: Test that all users can optionally include a note in the appointment booking for organizational purposes.
   Initial State/Setup:

   - User is logged into their account
   - User selects an available appointment time

   Input:

   - User adds a note to the appointment booking
   - Inputs:
     - Appointment
     - Note

   Expected output: The user should be able to add a note to the appointment booking.
   Actual output: The user is able to add a note to the appointment booking.
   Result: PASS

4. **FR-AP5-1.1**
   Requirements: APR5
   Description: Test that customers can interact with a virtual AI assistant to assist the user in scheduling an appointment.
   Initial State/Setup:

   - User is logged into their account

   Input:

   - User interacts with the AI assistant to schedule an appointment

   Expected output: The AI assistant should be able to help the user schedule an appointment.
   Actual output: Fails with NotImplemented Error
   Result: FAIL

5.5 Shop Lookup Requirements

5.5.1 Summary of Test Cases

Id	Reference Requirement	Summary	Expected Output	Pass/Fail
FR-SL1-1.1	SL1	Customers should be able to search for shops by name.	PASS	
FR-SL2-1.1	SL2	Customers should be able to search for shops by address, postal code, city, country, etc.	PASS	
FR-SL3-1.1	SL3	The system should be able to return a set of coordinates (geolocation) if given a valid address.	PASS	
FR-SL4-1.1	SL4	Customers should be able to filter out shops that they think are too far from thei current location or a spefi-cied address.	PASS	
FR-SL5-1.1	SL5	Customers should be able filter out shop that do not provide the service or services that they are looking for.	FAIL	
FR-SL6-1.1	SL6	Customers should be able to sort the search results alphabetically	FAIL	
FR-SL7-1.1	SL7	Customers should be able to sort the search results from closest to furthest (or vice versa)	FAIL	

Table 4: Test Cases

5.5.2 Individual Test Cases

1. **FR-SL1.1**
   Requirements: SL1
   Description: Test that customers search shop by name.
   Initial State/Setup: Customer account is logged in, there are multiple shops in the database.
   Input: Search query Expected output: List of shops containing the string
   Actual output: Sorted list (alphabetically)
   Result: PASS

2. **FR-SL2.1**
   Requirements: SL2
   Description: Test that customers can sort search results by name.
   Initial State/Setup: Customer account is logged in, there are multiple shops in the database.
   Input: Boolean Expected output. Sorted list (alphabetically)
   Actual output: Sorted list (alphabetically)
   Result: PASS

3. **FR-SL3.1**
   Requirements: SL3
   Description: Test that customers can sort search results by name.
   Initial State/Setup: Customer account is logged in, there are multiple shops in the database.
   Input: Boolean Expected output. Sorted list (alphabetically)
   Actual output: Sorted list (alphabetically)
   Result: PASS

4. **FR-SL4.1**
   Requirements: SL4
   Description: Test that customers can sort search results by name.
   Initial State/Setup: Customer account is logged in, there are multiple shops in the database.
   Input: Boolean Expected output. Sorted list (alphabetically)
   Actual output: Sorted list (alphabetically)
   Result: PASS

5. **FR-SL5.1**
   Requirements: SL5

Description: Test that customers can sort search results by name.
Initial State/Setup: Customer account is logged in, there are multiple shops in the database.
Input: Boolean Expected output. Sorted list (alphabetically)
Actual output: Sorted list (alphabetically)
Result: PASS

6. **FR-SL6.1**
   Requirements: SL6
   Description: Test that customers can sort search results by name.
   Initial State/Setup: Customer account is logged in, there are multiple shops in the database.
   Input: Boolean Expected output. Sorted list (alphabetically)
   Actual output: Sorted list (alphabetically)
   Result: PASS

7. **FR-SL7.1**
   Requirements: SL7
   Description: Test that customers can sort search results by distance.
   Initial State/Setup: Customer account is logged in, there are multiple shops in the database.
   Input: Boolean Expected output. Sorted list (by distance)
   Actual output: Sorted list (by distance)
   Result: PASS

6 Nonfunctional Requirements Evaluation

6.1 Look and Feel

Test ID	Req	Desc	Input	Act. Result	Exp. Result	Pass/Fail
NFR-LFR1-1	LFR1	Looping through all web components that are of the type "NextPage", and checking that no more than 5 "NextPage" web components are rendered inside as children components to avoid overly deep/confusing, interaction levels, with this test outputting the maximum depth found	All "NextPage" web components	3	<= 5	Pass

NFR-LFR2-1	LFR2	Looping through all HTML elements that are rendered in the absolute position (relative to the nearest positioned component; moves along with page scrolling), and checking that each contains a "box-shadow" CSS style	All absolutely positioned HTML elements	true	true	Pass
NFR-LFR3-1	LFR3	Looping through all HTML elements with a class name containing a "panel-" prefix, indicating it is a panel whose visibility can be toggled, and checking if they contain a "transform" CSS property for smooth transitions	All HTML elements with "panel-" styling class prefix	true (no elements found)	true	Pass

NFR-LFR4-1	LFR4	Looping through all HTML elements with a "border" style and checking if they also contain a "border-radius" style for smoothed edges	All HTML elements with a "border" style	true	true	Pass
NFR-LFR5-1	LFR5	Looping through all HTML elements with a "color" style and checking if the corresponding hex code is in a predefined colour palette	All HTML elements with a "color" style	true	true	Pass

Table 5: Look and Feel Tests and Results

6.2 Usability

Test ID	Req	Desc	Input	Act. Result	Exp. Result	Pass/Fail

NFR-UHR1-1	UHR1	A user who has little technical background is asked to perform the common actions of the customer workflow and to rate the easiness of using the UI from 1-10	N/A	9	>= 8	Pass
NFR-UHR1-2	UHR1	A user who has little technical background is asked to perform the common actions of the shop owner/employee workflow and to rate the easiness of using the UI from 1-10	N/A	8	>= 8	Pass
NFR-UHR2-1	UHR2	Query all interactable HTML elements and make sure they have accessibility support through "index" and "role" tags as well as keyboard listeners	Interactable HTML elements (such as buttons)	false (still requiring full implementation, only working in some parts of app)	true	Fail

NFR-UHR3-1	UHR3	A user who has little technical background is asked to guess how many interactable components there are, given different pages of the website, and is tested to see if >= 80 percent accuracy is achieved overall	Various pages of the application	90 percent	>= 80 percent	Pass
NFR-UHR4-1	UHR4	Query all interactable HTML elements and make sure they do not have any double click event handlers	Interactable HTML elements (such as buttons)	true	true	Pass
NFR-UHR5-1	UHR5	Query all HTML "input" tag elements (without a selectable list) and make sure they are tagged as search bars, for authentication/profile, or for chatting purposes	All "input" tag HTML elements without a dropdown list	true	true	Pass

Table 6: Usability Tests and Results

6.3 Performance

6.3.1 System Uptime

References requirement NFR-PR1-1

As of revision 1, the application does not need to be deployed as per our project supervisor Nabil. System uptime testing will not be required for this revision.

6.3.2 System Storage Usage

References requirement NFR-PR2-1

This testing is done to ensure the browser is not storing any unnecessary information and is not storing any information that could lead to harmful behaviour by external users.

Id	Reference Requirement	User Action	Expected Output	Actual Output	Pass/Fail
SSU-1	NFR-PR2-1	User registers an account, signs in, fills in forms on the website, and utilizes search functions	User login info, SessionID, CSRF token saved	The only information saved on the user's browser is their basic login information, their SessionID, and the CSRF token	Pass

Table 7: System storage usage test

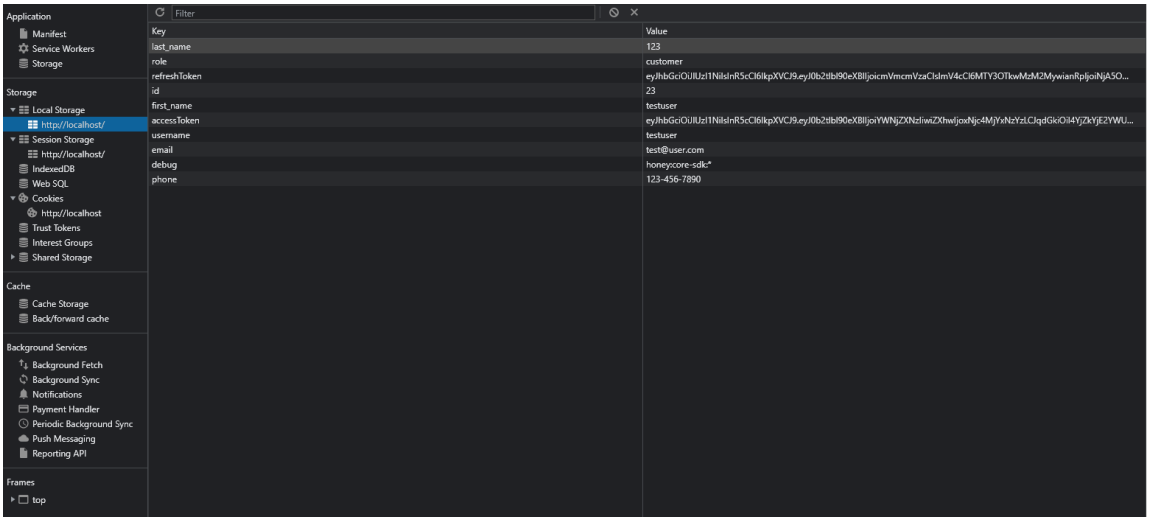

Figure 1: Storage containing information from logged in user

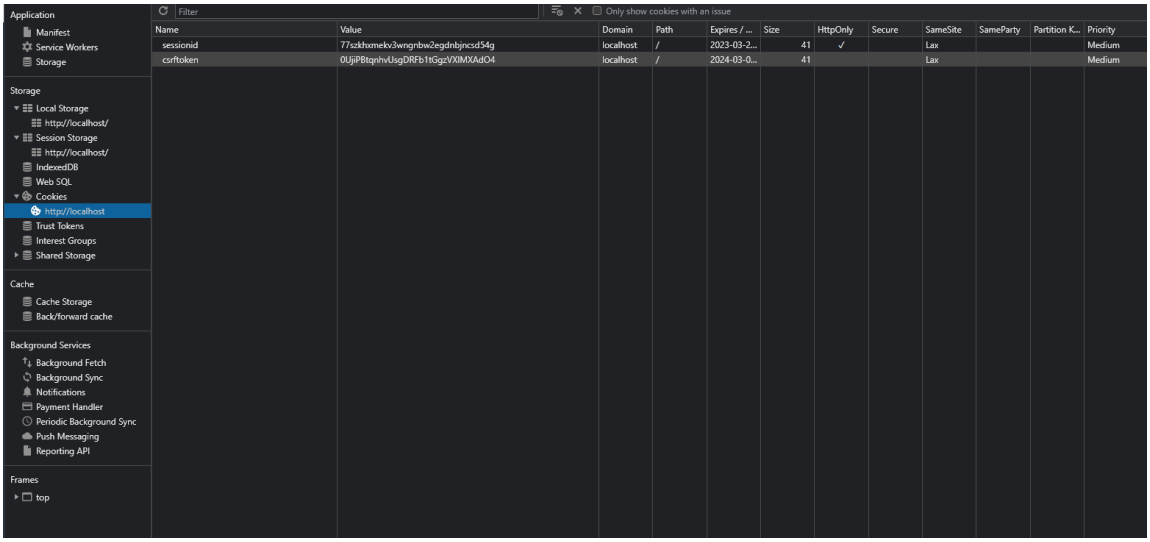

Figure 2: Storage containing saved cookie information, the SessionID and the CSRF token

6.3.3 Database Efficiency

References requirement NFR-PR3-1
Tests for database efficiency were conducted using Silk, a profiling tool for the Django framework. Each time an API request is made, the tool displays how long that request took to retrieve from the database. These results can be used to determine if the team needs to improve efficiency in places where requests are taking an increased amount of time.

After performing various different operations on the Sayyara website, a total of 90 requests to the database was recorded. The average time for a request was 125 ms, which is acceptable and is not noticeable to the end user who is waiting for a request. The SRS document requested that the database requests take between 1 and 5 seconds, so this is even faster than the threshold originally requested and exceeds expectations. While most requests were completed in this amount of time or sooner, some requests took a noticeably longer amount of time, and have been recorded as outliers in the table below.

Id	Reference Requirement	User Action	Output	Improvements Needed
DE-1	NFR-PR3-1	Submitting a new quote to a shop when a customer creates a new quote request	1383ms overall; 8ms spent on queries; 5 queries total	Look into optimizing the process of sending a POST request for quote. Additionally, consider adding a loading indicator to let the user know their request is being processed as there is currently no indicator for this
DE-2	NFR-PR3-1	Registering a new user account	1357ms overall; 4ms spent on queries; 4 queries total	Consider adding a loading indicator to let the user know their request is being processed as there is currently no indicator for this.
DE-3	NFR-PR3-1	The entire database list of vehicle makes is requested even though it is not visible to the user on the frontend	494ms overall; 133ms spent on queries; 66 queries total	Currently the entire list of vehicle makes is being requested from the backend without it being visible to the user. Allow the user to add a new vehicle in a quote request if they choose to do so so this query is not pointless.

Table 8: Outlier requests during database efficiency testing

6.4 Operational and Environmental

6.4.1 Internet Browser Compatibility

The application was tested using 4 of the most popular web browsers, as these are the most likely browsers the general public would use for the application. General usage of the Sayyara application was performed, including creating a new account, signing in, filling out forms such as quote requests and profile modification information, and utilizing the search functions on different pages.

Id	Reference Requirement	Internet Browser	Browser Version	Pass/Fail
IBC-1	NFR-OER1-1	Google Chrome	110.0.5481.178	Pass
IBC-2	NFR-OER1-1	Mozilla Firefox	110.0.1	Pass
IBC-3	NFR-OER1-1	Microsoft Edge	110.0.1587.63	Pass
IBC-4	NFR-OER1-1	Safari	16.3	Pass

Table 9: Internet browser compatibility tests

6.5 Security

6.5.1 Password Encryption

References requirement SR1
Django uses PBKDF2 with SHA56 to hash the passwords when creating a user account. Taking a look at the database entries, we can see that the stored passwords are indeed encrypted and not stored as plain text. An example taken from the database can be seen in the figure below.

Id	Reference Requirement	User Action	Expected Results	Actual Result	Pass/Fail
SR1-1	SR1	Create a test account and check database entry for the value stored in the password field	Encrypted password	Encrypted password	Pass

Table 10: Encryption test

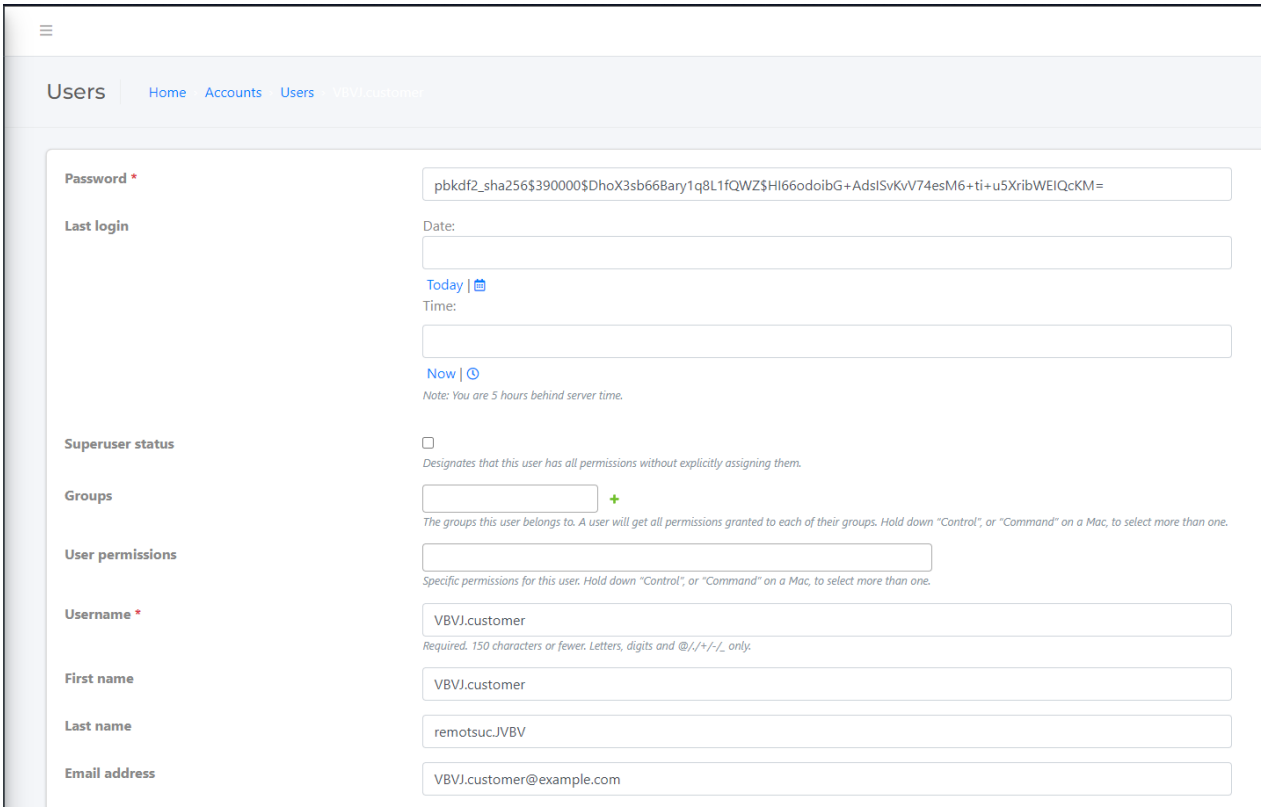

Figure 3: How a user's password is stored within the database.

6.5.2 Securing Personal Information

References requirement SR2
This test is passed by default as the system does not even allow a user other than the one who created the data node (shop, appointment, etc) to perform any CRUD operations on it whatsoever.

6.5.3 Rate limiting

References requirement SR3
Rate limiting database operations was beyond the scope of the system, and this this test (and its corresponding NFR) is not considered nor performed.

6.6 Cultural and Political

6.6.1 Offensive language

References requirement CPR1
There was no trace of any potentially offensive language or profanity to be found within the source code of the project. This was verified by using the search for words function of VSCode and with a list of known offesnvie language.

6.6.2 Other languages

References requirement CPR2
Supporting languages other than English was beyond the scope of the system, and this this test (and its corresponding NFR) is not considered nor performed.

6.6.3 Profanity filter for chat

References requirement CPR3
This feature was not implemented yet and so validating this non functional requirement will come at a later time.

6.7 Compliance

6.7.1 Disclaimer upon registration and survey results

References requirement CR1
As there is not a yet an end user agreement, disclaimer, etc. The test and survey that corresponds with this NFR could not be completed yet. The test and survey

will have to be revisited at a later date when a disclaimer is created or provided by the primary stakeholder.

7 Unit Testing

The following test cases outline the unit tests in simple language. Please do note that the tests are implemented through a combination of backend unit tests.

7.1 Register a new customer

7.1.1 Module

Customer

7.1.2 Initial State

User fills out the registration form and submits the data

7.1.3 Results

A new customer is created with the provided data

7.2 Sign in as customer

7.2.1 Module

Customer

7.2.2 Initial State

User fills out the sign-in form and submits the data

7.2.3 Results

User is authenticated with the provided data

7.3 Search for customers

7.3.1 Module

Customer

7.3.2 Initial State

A **non-customer** is searching for customers

7.3.3 Results

A customer is found by the provided criteria

7.4 View customer profile

7.4.1 Module

Customer

7.4.2 Initial State

A customer has navigated to their profile

7.4.3 Results

The customer sees the relevant data pertaining to their account

7.5 Edit customer profile

7.5.1 Module

Customer

7.5.2 Initial State

A customer is editing their profile information

7.5.3 Results

The customer's profile information is updated

7.6 Register a new employee

7.6.1 Module

Employee

7.6.2 Initial State

The shop owner has already sent an invite for the user. User fills out the registration form and submits the data

7.6.3 Results

A new employee is created with the provided data

7.7 Sign in as employee

7.7.1 Module

Employee

7.7.2 Initial State

User fills out the sign-in form and submits the data

7.7.3 Results

User is authenticated with the provided data

7.8 Search for employees

7.8.1 Module

Employee

7.8.2 Initial State

An employee is searching for other employees

7.8.3 Results

An employee is found by the provided criteria

7.9 View employee profile

7.9.1 Module

Employee

7.9.2 Initial State

An employee has navigated to their profile

7.9.3 Results

The employee sees the relevant data pertaining to their account

7.10 Edit employee profile

7.10.1 Module

Employee

7.10.2 Initial State

An employee is editing their profile information

7.10.3 Results

The employee's profile information is updated

7.11 Delete employee profile

7.11.1 Module

Employee

7.11.2 Initial State

A shop owner is deleting an employee's profile ie, removing them from the shop.

7.11.3 Results

The employee profile is deleted from the database

7.12 Register a new shop

7.12.1 Module

Shop

7.12.2 Initial State

User fills out the registration form and submits the data after signing up as a shop owner.

7.12.3 Results

A new shop is created with the provided data

7.13 Sign in as shop

7.13.1 Module

Shop

7.13.2 Initial State

User fills out the sign-in form and submits the data as a shop owner

7.13.3 Results

User is authenticated with the provided data

7.14 Search for shops

7.14.1 Module

Shop

7.14.2 Initial State

A customer is searching for shops in their area

7.14.3 Results

A shop is found by the provided criteria

7.15 View shop profile

7.15.1 Module

Shop

7.15.2 Initial State

A customer has navigated to a shop's profile

7.15.3 Results

The customer sees the relevant data pertaining to the shop

7.16 Edit shop profile

7.16.1 Module

Shop

7.16.2 Initial State

A shop is editing their profile information

7.16.3 Results

The shop's profile information is updated

7.17 Delete shop profile

7.17.1 Module

Shop

7.17.2 Initial State

A shop is deleting their profile

7.17.3 Results

The shop profile is deleted from the database

7.18 Add an employee to a shop

7.18.1 Module

Shop

7.18.2 Initial State

A shop is adding an employee to their shop

7.18.3 Results

An employee is added to the shop

7.19 Remove an employee from a shop

7.19.1 Module

Shop

7.19.2 Initial State

A shop is removing an employee from their shop

7.19.3 Results

An employee is removed from the shop

7.20 Add a service to a shop

7.20.1 Module

Shop

7.20.2 Initial State

A shop is adding a service to their shop

7.20.3 Results

A service is added to the shop

7.21 Remove a service from a shop

7.21.1 Module

Shop

7.21.2 Initial State

A shop is removing a service from their shop

7.21.3 Results

A service is removed from the shop

7.22 Add a vehicle to a customer's profile

7.22.1 Module

Vehicle

7.22.2 Initial State

A customer is adding a vehicle to their profile

7.22.3 Results

A vehicle is added to the customer profile

7.23 Remove a vehicle from a customer's profile

7.23.1 Module

Vehicle

7.23.2 Initial State

A customer is removing a vehicle from their profile

7.23.3 Results

A vehicle is removed from the customer profile

7.24 Add an appointment to a shop

7.24.1 Module

Appointment

7.24.2 Initial State

A shop is adding an appointment to their schedule

7.24.3 Results

An appointment is added to the shop schedule

7.25 Remove an appointment from a shop

7.25.1 Module

Appointment

7.25.2 Initial State

A shop is removing an appointment from their schedule

7.25.3 Results

An appointment is removed from the shop schedule

7.26 Add an availability to an employee

7.26.1 Module

Availability

7.26.2 Initial State

An employee is adding an availability to their schedule

7.26.3 Results

An availability is added to the employee schedule

7.27 Remove an availability from an employee

7.27.1 Module

Availability

7.27.2 Initial State

An employee is removing an availability from their schedule

7.27.3 Results

An availability is removed from the employee schedule

7.28 Create a work order

7.28.1 Module

Work order

7.28.2 Initial State

A customer is creating a work order

7.28.3 Results

A work order is created

7.29 Edit a work order

7.29.1 Module

Work order

7.29.2 Initial State

A customer is editing a work order

7.29.3 Results

A work order is updated

7.30 Delete a work order

7.30.1 Module

Work order

7.30.2 Initial State

A customer is deleting a work order

7.30.3 Results

A work order is deleted from the database

7.31 Add a service to a work order

7.31.1 Module

Work order

7.31.2 Initial State

A customer is adding a service to a work order

7.31.3 Results

A service is added to the work order

7.32 Remove a service from a work order

7.32.1 Module

Work order

7.32.2 Initial State

A customer is removing a service from a work order

7.32.3 Results

A service is removed from the work order

7.33 Add an employee to a work order

7.33.1 Module

Work order

7.33.2 Initial State

A customer is adding an employee to a work order

7.33.3 Results

An employee is added to the work order

7.34 Remove an employee from a work order

7.34.1 Module

Work order

7.34.2 Initial State

A customer is removing an employee from a work order

7.34.3 Results

An employee is removed from the work order

7.35 Create a conversation

7.35.1 Module

Conversation

7.35.2 Initial State

A customer is creating a conversation

7.35.3 Results

A conversation is created

7.36 Edit a message

7.36.1 Module

Conversation

7.36.2 Initial State

A customer is editing a message

7.36.3 Results

A conversation is updated

7.37 Search for a shop by name

7.37.1 Module

Shop Lookup

7.37.2 Initial State

A customer is searching for a shop

7.37.3 Results

A list of all shops names that contain the query is provided

7.38 Search for a shop by address

7.38.1 Module

Shop Lookup

7.38.2 Initial State

A customer is searching for a shop

7.38.3 Results

A list of all shops addresses that match the query is provided

7.39 Filter shops by services

7.39.1 Module

Shop Lookup

7.39.2 Initial State

A customer is provides a list of desired services

7.39.3 Results

A list of all shops that provide the service is provided

7.40 Filter shops by distance

7.40.1 Module

Shop Lookup

7.40.2 Initial State

A customer is filtering shops that are further than a certain distance in km

7.40.3 Results

A list of all shops names are no further than the specified distance is provided

7.41 Sort shops by name

7.41.1 Module

Shop Lookup

7.41.2 Initial State

A customer is sorting search results alphabetically

7.41.3 Results

A list of shops that match the filter and search ordered alphabetically (or in reverse)

7.42 Sort shops by distance

7.42.1 Module

Shop Lookup

7.42.2 Initial State

A customer is sorting search results from closest to furthest

7.42.3 Results

A list of shops that match the filter and search ordered by distance (or in reverse)

8 Changes Due to Testing

Several initial test failures revealed unimplemented functional requirements such as the lack of password format validity checking. Others exposed bugs including the profile update backend failing to save phone number changes. These missing features and bugs were then implemented and resolved, respectively. Frontend testing revealed an additional bug involving the profile update page failing silently, which will be addressed prior to the next revision.

Performing some of the nonfunctional tests that had to do with compatibility on both web and mobile highlighted some visual issues that will need to be addressed in the next iteration of the application. Additionally, performing the database efficiency testing highlighted that more visual indicators need to be implemented so the user realizes that the page is loading and is still responding. Overall, from nonfunctional testing and stakeholder feedback, there needs to be some visual improvements implemented to ensure the user has the best possible experience while using the Sayyara application.

9 Automated Testing

9.1 System Uptime Stress Testing

As of revision 1 of the project, the application does not need to be deployed as per our project supervisor Nabil. System uptime stress testing will no longer be required for this revision.

9.2 Unit Testing

As mentioned in section 8, the unit testing is managed by automated tests writing in Python and Javascript for automating the given test scenarios.

```
python manage.py test -v 2

Found 96 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
...
...
test_create_service (shop.tests.ServiceTestCase) ... ok
test_delete_service (shop.tests.ServiceTestCase) ... ok
test_get_all_services (shop.tests.ServiceTestCase) ... ok
test_update_service (shop.tests.ServiceTestCase) ... ok
test_base_template_exists (user.tests.UserRegistrationTestCase) ... ok
test_registration_form_get (user.tests.UserRegistrationTestCase) ... ok
test_registration_form_post (user.tests.UserRegistrationTestCase) ... ok
test_valid_logout (user.tests.UserRegistrationTestCase) ... ok
test_get_all_appointments (appointment.tests.AppointmentTestCase) ... ok
test_create_appointment (appointment.tests.AppointmentTestCase) ... ok
test_delete_appointment (appointment.tests.AppointmentTestCase) ... ok
test_update_appointment (appointment.tests.AppointmentTestCase) ... ok
test_create_vehicle (vehicle.tests.VehicleTestCase) ... ok
test_delete_vehicle (vehicle.tests.VehicleTestCase) ... ok
test_get_all_vehicles (vehicle.tests.VehicleTestCase) ... ok
test_update_vehicle (vehicle.tests.VehicleTestCase) ... ok
test_create_quote (quote.tests.QuoteTestCase) ... ok
test_delete_quote (quote.tests.QuoteTestCase) ... ok
test_get_all_quotes (quote.tests.QuoteTestCase) ... ok
test_update_quote (quote.tests.QuoteTestCase) ... ok
test_create_quote_request (quote_request.tests.QuoteRequestTestCase) ... ok
test_delete_quote_request (quote_request.tests.QuoteRequestTestCase) ... ok
test_get_all_quote_requests (quote_request.tests.QuoteRequestTestCase) ... ok
test_update_quote_request (quote_request.tests.QuoteRequestTestCase) ... ok
test_create_conversation (conversation.tests.ConversationTestCase) ... ok
test_delete_conversation (conversation.tests.ConversationTestCase) ... ok
test_get_all_conversations (conversation.tests.ConversationTestCase) ... ok
test_update_conversation (conversation.tests.ConversationTestCase) ... ok
test_create_employee_reservation (employee_reservation.tests.EmployeeReservationTestCase) ... ok
test_delete_employee_reservation (employee_reservation.tests.EmployeeReservationTestCase) ... ok
test_get_all_employee_reservations (employee_reservation.tests.EmployeeReservationTestCase) ... ok
test_update_employee_reservation (employee_reservation.tests.EmployeeReservationTestCase) ... ok
test_create_availabilities (availabilities.tests.AvailabilitiesTestCase) ... ok
test_delete_availabilities (availabilities.tests.AvailabilitiesTestCase) ... ok
test_get_all_availabilities (availabilities.tests.AvailabilitiesTestCase) ... ok
test_update_availabilities (availabilities.tests.AvailabilitiesTestCase) ... ok
test_create_employee (employee.tests.EmployeeTestCase) ... ok
test_delete_employee (employee.tests.EmployeeTestCase) ... ok
test_get_all_employees (employee.tests.EmployeeTestCase) ... ok
test_update_employee (employee.tests.EmployeeTestCase) ... ok
test_create_time_slot (time_slot.tests.TimeSlotTestCase) ... ok
test_delete_time_slot (time_slot.tests.TimeSlotTestCase) ... ok
test_get_all_time_slots (time_slot.tests.TimeSlotTestCase) ... ok
test_update_time_slot (time_slot.tests.TimeSlotTestCase) ... ok
test_create_work_order (work_order.tests.WorkOrderTestCase) ... ok
test_delete_work_order (work_order.tests.WorkOrderTestCase) ... ok
test_get_all_work_orders (work_order.tests.WorkOrderTestCase) ... ok
test_update_work_order (work_order.tests.WorkOrderTestCase) ... ok
test_create_shop_service (shop_service.tests.ShopServiceTestCase) ... ok
test_delete_shop_service (shop_service.tests.ShopServiceTestCase) ... ok
test_get_all_shop_services (shop_service.tests.ShopServiceTestCase) ... ok
test_update_shop_service (shop_service.tests.ShopServiceTestCase) ... ok
test_create_message (message.tests.MessageTestCase) ... ok
test_delete_message (message.tests.MessageTestCase) ... ok
test_get_all_messages (message.tests.MessageTestCase) ... ok
test_update_message (message.tests.MessageTestCase) ... ok
test_create_shop (shop.tests.ShopTestCase) ... ok
test_delete_shop (shop.tests.ShopTestCase) ... ok
test_get_all_shops (shop.tests.ShopTestCase) ... ok
test_update_shop (shop.tests.ShopTestCase) ... ok
test_create_customer (customer.tests.CustomerTestCase) ... ok
test_delete_customer (customer.tests.CustomerTestCase) ... ok
test_get_all_customers (customer.tests.CustomerTestCase) ... ok
test_update_customer (customer.tests.CustomerTestCase) ... ok
test_create_base_account (account.tests.AccountTestCase) ... ok
test_create_new_account (account.tests.AccountTestCase) ... ok
test_get_all_accounts (account.tests.AccountTestCase) ... ok
test_password_setter (account.tests.AccountTestCase) ... ok
```

Figure 4: Unit Tests run using pytest through Django framework

9.3 End-To-End Testing

Also known as user acceptance testing or E2E testing is also deployed in an automated environment simulating different user agents and user-scenarios to further validate and verify functional requirements down to the user-interface level. The following is a subset of all the E2E tests that cover the requirements given. Note that the steps are implemented in code using javascript and the playwright framework.

- **Customer Module**

  1. A user can register as a customer on the web app.
     - Select the "Sign up" button
     - Fill out the form to create the account
     - Confirm your email by clicking the link sent by email.
     - Log in using the credentials.
  2. A user can create a quote request.
     - Log in as a customer.
     - Select the "Post a quote request" button.
     - Fill out the form.
     - Click "Save".
  3. A user can create a vehicle.
     - Log in as a customer.
     - Select the "Profile" tab.
     - Select the "New Vehicle" button.
     - Fill out the form.
     - Click "Create".

- **Shop Owner Module**

  1. A user can register as a shop owner on the web app.
     - Select the "Sign up" button
     - Fill out the form to create the account
     - Confirm your email by clicking the link sent by email.
     - Log in using the credentials.
  2. A user can create a shop on the web app.

- Log in as a shop owner.
- Select the "Shops" tab.
- Select the "New Shop" button.
- Fill out the form.
- Click "Create".

3. A user can create an appointment on the web app.
   - Log in as a shop owner.
   - Select the "Shops" tab.
   - Select the "New Appointment" button.
   - Fill out the form.
   - Click "Save".
4. A user can create a quote on the web app.
   - Log in as a shop owner.
   - Select the "Shops" tab.
   - Select the "New Quote" button.
   - Fill out the form.
   - Click "Create".

- **Employee Module**

1. A user can register as a shop employee on the web app.
   - Select the "Sign up" button
   - Fill out the form to create the account
   - Confirm your email by clicking the link sent by email.
   - Log in using the credentials.
2. A user can create a shop employee reservation on the web app.
   - Log in as a shop employee.
   - Select the "Shops" tab.
   - Select the "New Shop Employee Reservation" button.
   - Fill out the form.
   - Click "Save".

- **Messaging Module**

1. A user can message other users using the web app.
   - Log in as a customer.
   - Select the "Conversations" tab.
   - Select the "New Conversation" button.
   - Fill out the form.
   - Click "Create".
   - The message should be received.

- **Shop Search Module**

1. A user can search the web app for shops based on location.
   - Log in as a customer.
   - Enter a location in the "Search" field.
   - Shops matching the search should be displayed.

- **Quotes Module**

1. A user can post a quote request on the web app.
   - Log in as a customer.
   - Select the "Quote requests" tab.
   - Select the "New Quote request" button.
   - Fill out the form.
   - Click "Create".
2. A user can post a quote on the web app.
   - Log in as a shop owner.
   - Select the "Quote requests" tab.
   - Select the "New Quote" button.
   - Fill out the form.
   - Click "Create".

- **Reservations Module**

1. A user can create a reservation on the web app.
   - Log in as a customer.
   - Select the "Shops" tab.

- Select the "New Reservation" button.
- Fill out the form.
- Click "Save".

2. A user can create a shop employee reservation on the web app.
   - Log in as a shop employee.
   - Select the "Shops" tab.
   - Select the "New Shop Employee Reservation" button.
   - Fill out the form.
   - Click "Save".

- **Shops Module**

1. A user can create a shop on the web app.
   - Log in as a shop owner.
   - Select the "Shops" tab.
   - Select the "New Shop" button.
   - Fill out the form.
   - Click "Create".

- **Vehicles Module**

1. A user can create a vehicle on the web app.
   - Log in as a customer.
   - Select the "Vehicles" tab.
   - Select the "New Vehicle" button.
   - Fill out the form.
   - Click "Save".

- **Work Orders Module**

1. A user can create a work order on the web app.
   - Log in as a customer.
   - Select the "Work Orders" tab.
   - Select the "New Work Order" button.
   - Fill out the form.
   - Click "Create".

- **Appointments Module**

    1. A user can create an appointment on the web app.
        - Log in as a shop owner.
        - Select the "Appointments" tab.
        - Select the "New Appointment" button.
        - Fill out the form.
        - Click "Save".

10 Trace to Requirements

FR	Tests	FR	Tests
AuR1	FR-AR1-1, FR-AR1-2, FR-AR1-3	AuR2	FR-AR2-1, FR-AR2-2A, FR-AR2-2B
AuR3	FR-AR2-3	SR1	
SR2		SR3	
PAR1		PAR2	
PAR3	FR-PAR3-1, FR-PAR3-2	PAR4	
PAR5	FR-VE1-1.1, FR-VE1-1.2, FR-VE1-1.3, FR-VE1-1.4	PAR6	
PAR7		PAR8	
PAR9		PAR10	
PAR11		ApR1	FR-APR1-1.1
ApR2	FR-ARP2-1.1	ApR3	FR-APR3-1.1
ApR4	FR-APR4-1.1	ApR5	FR-APR5-1.1
WOR1		WOR2	
WOR3		QR1	FR-QR1-1.1, FR-QR1-1.2
QR2	FR-QR2-1.1, FR-QR2-1.2, FR-QR2-1.3	QR3	FR-QR2-1.1, FR-QR2-1.2, FR-QR2-1.3

Table 11: Traceability between Tests and Functional Requirements

NFR	**Tests**	**NFR**	**Tests**
LFR1	NFR-LFR1-1	LFR2	NFR-LFR2-1
LFR3	NFR-LFR3-1	LFR4	NFR-LFR4-1
LFR5	NFR-LFR5-1	UHR1	NFR-UHR1-1, NFR-UHR1-2
UHR2	NFR-UHR2-1	UHR3	NFR-UHR3-1
UHR4	NFR-UHR4-1	UHR5	NFR-UHR5-1
PR1	NFR-PR1-1	PR2	SSU-1
PR3	DE-1, DE-2, DE-3	OER1	IBC-1, IBC-2, IBC-3, IBC-4
MSR1			
SR1	Section 6.5.1	SR2	Section 6.5.2
SR3	Section 6.5.3	CPR1	Section 6.6.1
CPR2	Section 6.6.2	CPR3	Section 6.6.3
CR1	Section 6.7.1		

Table 12: Traceability between Tests and Non-functional requirements

11 Trace to Modules

Module	Tests	Module	Tests
FE1	FR-AR2-1, FR-AR2-2A, FR-AR2-2B	BE1	FR-AR1-1, FR-AR1-2, FR-AR1-3, FR-PAR3-1, FR-PAR3-2
FE2	FR-VE1-1.1	BE2	FR-VE1-1.2, FR-VE1-1.3, FR-VE1-1.4
FE3	NFR-LFR1-1, NFR-UHR1-1	BE3	FR-AP1-1.1, FR-AP2-1.1
FE4		BE4	FR-AP1-1.1, FR-AP2-1.1
FE5		BE5	N/A
FE6	NFR-LFR1-1	BE6	FR-AP1-1.1, FR-AP2-1.1
FE7	NFR-LFR1-1	BE7	FR-VE1-1.1, FR-VE1-1.2
FE8	NFR-LFR1-1	BE8	FR-QR1-1.1, FR-QR1-1.2, FR-QR2-1.1, FR-QR2-1.2
FE9	FR-QR1-1.1, FR-QR2-1.1, FR-QR2-1.2, FR-QR2-1.3	BE9	FR-QR1-1.1, FR-QR2-1.1
FE10	FR-QR1-1.2, FR-QR2-1.2, FR-QR2-1.3	BE10	FR-AP2-1.1, FR-AP4-1.1
FE11		BE11	N/A
FE12	N/A		
FE13	FR-AP1-1.1, FR-AP2-1.1, FR-AP4-1.1		
FE14			

Table 13: Traceability between Tests and Modules

12 Code Coverage Metrics

```
>>> coverage.py json --pretty-print --show-contexts --rcfile=.coveragerc
Generating the json report with coverage.py

Name                       Stmts   Miss  Cover   Context
-----------------------------------------------------------
customer/models.py           106     40    62%
customer/tests.py             58      0   100%
customer/views.py             71      0   100%
employee/models.py            83     24    71%
employee/tests.py             55      0   100%
employee/views.py             70      0   100%
message/models.py            117     36    69%
message/tests.py              76      0   100%
message/views.py              72      0   100%
quote/models.py              105     38    64%
quote/tests.py                55      0   100%
quote/views.py                69      0   100%
quote_request/models.py      101     46    54%
quote_request/tests.py        55      0   100%
quote_request/views.py        71      0   100%
shop/models.py               234     63    73%
shop/tests.py                144      0   100%
shop/views.py                195      0   100%
shop_service/models.py        99     39    61%
shop_service/tests.py         55      0   100%
shop_service/views.py         70      0   100%
time_slot/models.py           85     25    71%
time_slot/tests.py            54      0   100%
time_slot/views.py            67      0   100%
vehicle/models.py            113     44    61%
vehicle/tests.py              57      0   100%
vehicle/views.py              76      0   100%
work_order/models.py          85     23    73%
work_order/tests.py           55      0   100%
work_order/views.py           67      0   100%
-----------------------------------------------------------
TOTAL                       2077    547    74%
```

Figure 5: Code coverage of the backend using coverage.py

Appendix — 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.)

   Reflections

- Harsh: The VnV Plan for the Sayyara project stayed similar to the activities that were actually conducted for VnV, with the exception of the scope of user testing. The changes to the original VnV Plan came about in response to feedback from stakeholders, which indicated that the user interface was not as intuitive or user-friendly as desired. In future projects, it may be possible to anticipate the need to adjust the scope of VnV activities, depending on the nature of the product and feedback from stakeholders.

- Alyssa: In terms of the nonfunctional requirement testing, for the most part, the VnV Plan stayed the same. What changed were some of the methods of testing these requirements. For example, for database efficiency testing, when writing the VnV Plan, it was undetermined what would be used to test this. After doing research while testing, I was able to find a dedicated profiling tool for Django that performed the exact tests I needed for the VnV Report. This change occurred because I was looking for a simple but effective solution that was catered to the exact framework we are using for the backend. Otherwise, I would've had to have spend time manually querying database entries and writing debug messages to determine how long they took. Changes in future projects would depend on the requirements of the project, as well as feedback from stakeholders on whether or not this type of VnV would be necessary to ensure the product meets the requirements.

- Collin: For the most part, the VnV Plan was quite similar to the activites that were ultimately performed. However, there were a few differences between the

VnV Plan and the activities that were actually conducted. For some non function requirements testing, namely the compliance and security, and political ones, the test plans that were proposed were either unnecessary, did not need to be automated (i.e. manually testing would have sufficed), was beyond the scope of what is reasonable given the time frame, or the features were simply not implemented. This resulted in the initial VnV plan requiring more work and effort than required and would not really improve the validation an verification of the system, so changes to the some of the testing methods had to be made. In future projects, changes to non-functional requirements VnV could possibly be anticipated, as the scope of the project can always change, some features deemed unnecessary or unneeded.

- Chris: The VnV Plan and VnV Report are not much different, except a few key areas. Firstly, some new testing frameworks were added to the frontend to better facilitate user interaction testing. These frameworks were not considered before, because at the time of the VnV Plan creation, development was still in its beginning stages, and as such there was not much yet to test; much of it was anticipation of modules and planning. However, after creating the user interface, the team noticed a need for a tool that can more quickly test clicks and interactions on the website. Additionally, there were some changes to non-functional requirement testing methodologies. For example, test NFR-LR1-1 was planned to be executed by using a tool that can visualize an interaction tree, but no such framework could be found in time that was compatible with our project. This was a mistake on my part, as I should have done more research to verify if conducting such a test was even possible. Overall, the changes made were mostly in pursuit of finding more efficient and easier methods of testing. These changes could've been anticipated if more research was done on testing frameworks, and if some practice tests were written first to gauge the difficulty in writing them, or if we sought out more experienced advice. As someone who hasn't done much test development, it was a vital learning experience for me, learning the importance of research, efficient tools, and not underestimating the difficulty of testing scope. Most of the rest of the VnV Plan was accurate, due to the team's experience with some of the initial testing frameworks we chose, as well as due to the team's concrete planning of what all the modules will be and how they would precisely interact.

- Kai: The VnV plan was devised prior to the design documents and most of the implementation, and as such did not have as much specific details on test cases and specific features to test for, compared to those outlined in the VnV report.

A major source of deviation was the relatively large breadth of the features in the application, and the uncertainties in the depth for the implementation of each. Ideally, the implementation should follow the SRS as closely as possible, and so the VnV should readily verify the SRS regardless of implementation details. In practice, due to time constraints and uneven familiarity with some of the tech stacks, changes were made to facilitate rapid implementation that cover as much of the requirement as possible. In spite of the differences, following rev0 implementations we were able to devise more time-feasible test cases with a small number of both success and failure scenarios for each major requirement, both manual and automatic, for the frontend and the backend, as documented in the VnV report. The deviations served as a good example of what to expect in the future, and highlights the advantages of a test-driven development process, in which tests and development can be coupled more closely.

- Ethan: There are two main differences I see between the VnV plan and the actual VnV work that took place:

    1. The number of tests planned was much lower than the number of tests performed. The VnV plan does a decent job of outlining the broad categories of tests which need to be performed, but within each of those categories is around 2-4 individual tests that need to be run: at least one for both normal and abnormal inputs, in addition to any boundary values. This change should be quite easy to anticipate in future projects as these types of test cases are quite commonly used.
    2. The control method of most frontend testing was manual, while the backend was mostly automatic. This change was made as the team discovered that manually testing the frontend portions of the system was much more time efficient than writing automatic tests, whereas the backend was much easier to automate. Running frontend manual tests allows for checking many aspects of the design at once such as errors rendering, layout problems, text formatting, color/contrast issues, etc. all in a much shorter time. Future projects might have a different approach to this problem (especially if the frontend framework or architecture is different), so anticipating which control method to use may still be difficult. Of course, this experience should mean more thought goes towards the problem next time.