Project Title: System Verification and Validation Plan for Sayyara

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

April 5, 2023

Revision History

Date	Version	Notes
Oct. 29, 2022	Rev 0	Nonfunctional tests for SRS sections 10.2.3-10.2.5
Oct. 31, 2022	Rev 0	Authentication and profile system tests
Nov. 2, 2022	Rev 0	Tests Defined and Usability Survey Questions Added for SRS Sections 10.2.1 and 10.2.2
Nov. 2, 2022	Rev 0	Revision 0
April 5, 2022	Rev 1	Revision 1

Table 1: Revision History

Contents

List of Tables

1 Symbols, Abbreviations and Acronyms

Term	Definition
Automated Testing	Testing that is performed automatically and is done by a tool
Dynamic Testing	Tests that are performed with the actual system running
Functional Testing	Tests that validate the system against the requirements
Manual Testing	Tests that are performed by hand by a user as opposed to automated testing
Record-And-Playback Testing	Tests that are performed through recording a user's activity and then imitating it (form of automated testing)
Static Testing	Tests that are performed without executing the actual system
Stress Testing	Tests that are performed to determine the robustness of the system
Structural Testing	Tests that are performed using the internal design of the system
PR	Pull request
UI	User Interface
DDoS	Distributed Denial-of-service; when multiple systems flood the bandwidth or resources of a targeted system

Table 2: Symbols, Abbreviations, and Acronyms

This document provides the team's plans for the verification and validation of the Sayyara progressive web application. The document also establishes test cases for relevant functional and nonfunctional requirements.

2 General Information

2.1 Summary

The purpose of this VnV plan is to state the team's intentions for the verification and validation of the Sayyara progressive web application as well as document the test cases to evaluate the functional and nonfunctional requirements established in the SRS document. Unit testing will be added at a later date when the application is further along in development.

2.2 Objectives

The purpose of the VnV plan is to ensure that we have sufficient coverage of testing for as many functional and nonfunctional requirements that require testing and can be tested. The VnV plan is also to ensure that the team builds confidence in not only building the correct software, but building the software correctly and to the satisfaction of stakeholders.

2.3 Relevant Documentation

The development plan discusses relevant tools that we plan to use for testing and development of the Sayyara application. Section 4 of this VnV plan expands upon some of these discussed ideas as we have delved further into development and research of more relevant tools and frameworks.
The functional and nonfunctional requirements referenced in section 5 of this VnV plan are cited directly from section 10 of the team's SRS document. The traceability matrices at the end of section 5 map the developed test cases to the relevant requirements from the SRS document.

3 Plan

In order to optimize the build, validation and test time on the final product, the project is split into two separate projects for the backend and frontend respectively. This is a cleaner method, as opposed to one large monolithic web application and allows for the team to split up work more efficiently. This approach also allows for the user and business logic to be decoupled and tested separately.

3.1 Verification and Validation Team

The team is split into two sub-teams responsible for the backend and frontend, and each member is assigned a role with respect to their commitments to the team below.

	Role	
Member	Backend	Frontend
Harsh	Team Lead	Validation Lead
Kai	Test Lead	Implementation Lead
Chris	Implementation Lead	Tester
Ethan	Validation Lead	Developer
Alyssa	Tester	Team Lead
Collin	Developer	Test Lead

Table 3: Verification and Validation Team

3.2 SRS Verification Plan

We intend to utilize the SRS document as a reference to the product features being delivered, which will be tracked using the requirement short name identifier in the issues and PR's of said feature. This will ensure traceability of every PR back to the respective requirements. Moreover, the team will use ad hoc feedback from other reviewers like our project supervisor and their correspondents and the teams' peers.

3.3 Design Verification Plan

Part of the design verification will be done through automated test cases, which will be delivered in conjunction with any features. The repository hosting the project will also enforce strict delivery requirements for the quality and correctness of the code being delivered.

3.4 Verification and Validation Plan Verification Plan

The VnV Verification Plan will involve a discussion among the team and the supervisor that includes the following topics:

- Ensure each task has appropriate description, priority, due dates, and lead developer assigned.

- Ensure that the VnV is integrated into the project management plan used by the supervisor to track progress.

3.5 Implementation Verification Plan

This verification plan serves to address the objectives and requirements defined in the next section for the system tests. This plan serves as a guide for reviewing the internal specifications of the system so that all aspects of the system are accounted for. This plan should be carried out prior to unit testing to make sure the core functionality is present prior to deploying the system.
The plan involves strict requirements for the code being delivered in TypeScript and Python to be linted and have test code for any features being delivered. For example, the PR for a notification system must include the test cases for the same, and those tests must pass to proceed with merging. Furthermore, we intend to have at least two code reviewers per PR, this way the team makes sure that the code is legible and up to the right standard of quality to be delivered into the project.

3.6 Automated Testing and Verification Tools

This project will heavily utilize automated testing and verification tools. While automated testing and verification tools were covered in the Development Plan, we have added testing and verification for the Django backend with unittest and autopep8 for linting. We will also need external geolocation tools such as an address generator and a distance between two addresses calculator. On top of this, we intend to utilize GitHub actions to build Docker images on demand to test, validate and publish the web application.

3.7 Software Validation Plan

The system will be tested on dummy data in the database that is loaded for every build using a collection of fixtures. This data will be constantly updated as new features get delivered, and may not represent the full spectrum of data we might see in the a real-world scenario. However, it provides a good baseline for the developers to test against until the project goes live.
The supervisor will host review sessions along with the key stakeholders of automotive shop owners to evaluate the software throughout the build process. This is also where the feedback on implementation of specific tasks will be incorporated to improve the next iteration of the product.

4 System Test Description

4.1 Tests for Functional Requirements

The system tests for functional requirements cover the main functionality of the application as outlined in section 10.1 of the SRS.

4.1.1 Authentication

Authentication tests corresponds to the requirements related to user account authentication under section 10.1.1 of the SRS.

Account Creation

1. FR-AR1-1

   Control: Manual, dynamic

   Initial State: Empty account registration form is displayed

   Input: Unused username and valid password of length 8

   Output: Displays successful registration, redirect to login view. New account created in user database

   Test Case Derivation: Valid account name and password should successfully complete account registration.

   How test will be performed: Manually verify that using a previously unused username and valid password redirects to a success display, then verify using pgAdmin that an row with corresponding username is created in the database.

2. FR-AR1-2

   Control: Manual, dynamic

   Initial State: Empty account registration form is displayed

   Input: A username that already exists in the database (e.g. admin) and any password

   Output: Displays username error, no new user created in database

   Test Case Derivation: New accounts should not be created using existing user name.

How test will be performed: See method for FR-AR1-1, verify by inspection that row with username in database is not duplicated or modified.

3. FR-AR1-3

   Control: Manual, dynamic

   Initial State: Empty account registration form is displayed

   Input: An unused username and a set of invalid passwords (length below 7, number or alphabet only, no capitalization)

   Output: Displays password error, no new user created in database

   Test Case Derivation: Invalid password should result in account registration failure and notification.

   How test will be performed: See method for FR-AR1-2.

Login Authentication

1. FR-AR2-1

   Control: Automated, dynamic

   Initial State: Empty login form is displayed

   Input: Valid test username and password existing in database (TEST_USER, TEST_PW)

   Output: Redirects to main user menu, returns authenticated token

   Test Case Derivation: Accessing the system with an existing account name with matching password should result in a successful login.

   How test will be performed: Automatic end-to-end testing confirms that valid input redirects to the correct page, and that an authenticated token is received as a response upon form submission.

2. FR-AR2-2

   Control: Automated, dynamic

   Initial State: Empty login form is displayed

Input: Non-existing username with any password, and valid username with wrong password

Output: Displays login failure

Test Case Derivation: Invalid login information should result in failure. No authenticated token should be provided for access to the rest of the application.

How test will be performed: See FR-AR2-1, ensure that no token is returned.

3. FR-AR2-3

   Control: Automated, dynamic

   Initial State: Backend host listening for API calls

   Input: API calls to profile edit and password update services without authentication token

   Output: Return 401 unauthorized access error

   Test Case Derivation: Unauthenticated users should not be able to access application functionality, especially modifying unauthorized information such as password and account profiles.

   How test will be performed: A list of API calls to profile and password updates are made using CURL or similar commands, verify that all calls are refused with 401 error.

4.1.2 Account Profile

This set of tests cover business and user profile display and editing requirements outlined under sections 10.1.2 and 10.1.3 of the SRS.

Profile Update

1. FR-PAR1-1

   Control: Manual, dynamic

   Initial State: Authenticated as business owner, business profile edit page is displayed with current description an contact information

   Input: new business name, address, phone number, email, description

Output: Redirects to business profile page displaying updated details based on input

Test Case Derivation: The supplied input should update each corresponding displayed business information correctly.

How test will be performed: Manual inspection of the updated business profile to ensure the new information matches input.

2. FR-PAR3-1

   Control: Manual, dynamic

   Initial State: Authenticated as customer, user profile edit page is displayed with current description and contact information

   Input: name, owned vehicle make, model, year, contact information

   Output: Redirects to user profile page displaying updated details based on input

   Test Case Derivation: The supplied input should update each corresponding displayed user information.

   How test will be performed: See FR-PAR1-1, verify new information matches input.

Employee Account Management

1. FR-PAR6-1

   Control: Manual, dynamic

   Initial State: Authenticated as business owner, employee with EMPLOYEE_NAME exists in database, employees list is displayed

   Input: EMPLOYEE_NAME is entered in search field

   Output: Employees list view is filtered to display only the row containing EMPLOYEE_NAME

   Test Case Derivation: Employees list should display all employees by default and filters to show only employees with matching name using the search input field.

   How test will be performed: Manual inspection of the filtered employees list view to confirm only the row containing EMPLOYEE_NAME is displayed.

2. FR-PAR7-1

   Control: Manual, dynamic

   Initial State: Authenticated as business owner, empty create employee form is displayed

   Input: new employee name and email address

   Output: Employees list displaying new employee, email is sent to input address with randomized default password

   Test Case Derivation: Owners should be able to create employee accounts for new employees, and new account login is emailed to the employee email address.

   How test will be performed: Manual inspection of the employees list view and email inbox to confirm successful employee creation, password generation and notification.

3. FR-PAR7-2

   Control: Manual, dynamic

   Initial State: Authenticated as business owner, employee with EMPLOYEE_NAME exists in database, employees list is displayed

   Input: delete employee button and confirm

   Output: View refreshes to display employees list minus deleted employee. Employee removed from authentication and profile database tables.

   Test Case Derivation: Owners should be able to remove employees from their business profile page.

   How test will be performed: Manual inspection of the employees list view to confirm deletion, log in with deleted account information to ensure authentication failure.

4.1.3 Appointments

This set of tests covers appointment requirements as outlined in section 10.1.4 of the SRS.

Change Shop Availability

1. FR-ApR1-1

   Control: Automated, dynamic

   Initial State: Authenticated as Shop Owner.

   Input: New range of times where shop is available for appointments.

   Output: Available hours for scheduling an appointment updated.

   Test Case Derivation: Shop Owners need to be able to change their availability on the fly.

   How test will be performed: Manual entry of new availability hours, and manual verification that these new hours are properly reflected.

Appointment Creation Relevant system requirements include ApR2 and ApR3.

1. FR-ApR3-2

   Control: Automated, dynamic

   Initial State: Authenticated as Shop Employee, quote sent to vehicle owner, appointment for quote not already scheduled.

   Input: Select "Schedule Appointment", choose approved quote, choose available date, TEST_MESSAGE input for appointment notes.

   Output: Appointment added to both vehicle owners and auto shops calendars, quote marked as "scheduled".

   Test Case Derivation: Shop Employees should be able to schedule appointments for vehicle owners.

   How test will be performed: Automatic selection of next available appointment slot, and verification that appointment was made, using API calls.

2. FR-ApR3-3

   Control: Automated, dynamic

   Initial State: Authenticated as Shop Owner, quote sent to vehicle owner, appointment for quote not already scheduled.

   Input: See FR-ApR3-2.

Output: See FR-ApR3-2.

Test Case Derivation: Same as FR-ApR3-2.

How test will be performed: See FR-ApR3-2.

4.1.4 Quotes

Quote Chat Service

1. FR-QR1-1

   Control: Automated, dynamic

   Initial State: Authenticated as vehicle owner, at least one quote request created and sent to at least one shop.

   Input: Send message in chat SAMPLE_MESSAGE.

   Output: User interface updated to show message delivered successfully; message appears visible to shop employees & shop owner.

   Test Case Derivation: Vehicle owners and shop employees may need to communicate about quote requests, clarifying details ect.

   How test will be performed: Automatic delivery of message and verification of message state through API calls.

Creating Quote Request Relevant system requirements include QR2 and QR3.

1. FR-QR2-1

   Control: Manual, dynamic

   Initial State: Authenticated as vehicle owner.

   Input: Select "New Quote Request"; input car details; select automotive shops to send request to.

   Output: New quote request is created and viewable by vehicle owner, and shops which have been sent the request.

   Test Case Derivation: Vehicle owners need to be able to create quote requests and send them to shops.

   How test will be performed: Manual interaction with user interface, and manual inspection of resulting quote request.

4.1.5 Shop Lookup

Search Bar

1. FR-SL-1

   Control: Automatic, Manual

   Initial State: Authenticated as a customer. There is at least one shop registered in the database.

   Input: A search query (string).

   Output: A list of all the shops in the database whose name field contains the search query (case insensitive).

   Test Case Derivation: Must ensure that the search bar can accurately return searching by name so that customers can easily find the details of a certain shop.

   How test will be performed: 20 random shop names will be generated in the database. The shop names will be a combination of randomly selected strings (3-5) from a predetermined set of strings. The search query will be a random sub-string of a randomly selected string from that same set. The query result (list of shops) will then be compared to the true list to see if they are the same.

2. FR-SL-2

   Control: Automatic, Manual

   Initial State: Authenticated as a customer. There is at least one shop registered in the database with geolocation data.

   Input: A search query (string)

   Output: A list of all the shops in the database whose address contains the search query (e.g street name, city, postal code, etc.)

   Test Case Derivation: It is often more useful to search by city or postal code so that customers can discover auto shops that are near them or near a point of interest.

   How test will be performed: 20 shops with random names will be generated with a random address. The street name, postal code, city, and country will be randomly selected from a set of predetermined strings (one set for each field). There will be a separate search query for street name, postal code, city, and

country. The query results (list of shops) will be compared to the true list of shops to see if they are the same.

Filtering

1. FR-SL-3

   Control: Manual, Automated

   Initial State: None

   Input: A valid street address

   Output: A geolocation (coordinates)

   Test Case Derivation: Need to make sure the system can properly parse street addresses and convert them into their corresponding geolocation.

   How test will be performed: Generate an address from an address generator. Get the coordinates from a geolocation API. Compare the geolocation returned by the system matches the correct geolocation.

2. FR-SL-4

   Control: Automatic

   Initial State: Authenticated as a customer. There is at least one shop registered in the database with geolocation data.

   Input: A distance limit in km (integer) and a valid address (string)

   Output: A list of all the shops in the database whose distance from the provided address is less than or equal to the distance limit provided.

   Test Case Derivation: Customers can't or won't travel far for vehicle maintenance. Need to ensure that filtering automotive shops by distance is working correctly so that customers can have a better sense of their options.

   How test will be performed: Random addresses will be generated and the distances from provided address will be calculated with an external API. Compare the list of shops and see if they contain all shops with distance less than or equal to the distance limit.

3. FR-SL-5

   Control: Automatic

Initial State: Authenticated as a customer. There is at least one shop registered in the database with services data.

Input: Service filter(s) (list of services)

Output: A list of all the shops that offer all the services in the service filter list.

Test Case Derivation: Customers need to be able to get an accurate list of shops that offer the types of services that they want.

How test will be performed: Shops with a random list of services will be generated. A list of 3 randomly selected services will be passed as input. Each generated shop will be checked to see if they contain the specified services and will be checked to see if they are contained within the returned list.

Sorting

1. FR-SL-6

   Control: Manual, Automated

   Initial State: Authenticated as a customer. There is at least two shops registered in the database.

   Input: Sort by name ascending or descending (Boolean)

   Output: Sorted list of shops

   Test Case Derivation: Easier for customers to keep track of which shops they've looked at or find a certain shop if they are ordered.

   How test will be performed: Check if the returned list of shops is in alphabetical or reverse alphabetical order.

2. FR-SL-7

   Control: Manual, Automated

   Initial State: Authenticated as a customer. There is at least two shops registered in the database with geolocation data.

   Input: Sort by distance closest or furthest first (Boolean) and a valid address

   Output: Sorted list of shops

   Test Case Derivation: Customers will want to look at the closest shop to them first.

How test will be performed: Check if the returned list of shops is properly sorted by distance.

...

4.2 Tests for Nonfunctional Requirements

The structure of the below tests follow the way the nonfunctional requirements are formatted on the SRS document.

4.2.1 Look and Feel

Minimal Interface

1. NFR-LFR1-1

   Type: Record-and-Playback Testing, Automatic

   Initial State: A page of the website is loaded and visible

   Input/Condition: Any component with an interaction

   Output/Result: A tree of interactions showcasing the depth of each interaction causing a user interface change (without going back to the previous frame), where the level of depth should not exceed 5, and the number of interaction paths taken per page should not exceed 10

   How test will be performed: Using automation tools, we will automate and record every action component being acted upon for every page, where some interactions will lead to other pages with other interactive components (creating a tree), and the tree will be tested by the above given metrics.

Illusion of Depth

1. NFR-LFR2-1

   Type: Automatic

   Initial State: A page of the website is loaded and visible with some popup

   Input/Condition: Some "absolutely" positioned HTML element(s) are visible

   Output/Result: Each absolutely positioned element should contain a "box-shadow" CSS style

How test will be performed: Using automation tools, we will query different page components and check that elements with the "absolute" style should likewise contain a "box-shadow" style, as this is a standard in modern web interface design

Seamless Transitions

1. NFR-LFR3-1

   Type: Automatic

   Initial State: A page of the website is loaded and visible

   Input/Condition: Some panel's visibility is toggled through a button interaction

   Output/Result: Every panel with a visibility toggle should have an associated transform CSS style to give it a smooth animation

   How test will be performed: Using automation tools, we will query for every web component tagged with a "-panel" suffix, which emphasizes the component as a top-level panel, and verify that each contains a transformation style and an animation, if its visibility can be toggled

Smoothed Edges

1. NFR-LFR4-1

   Type: Automatic

   Initial State: A page of the website is loaded and visible

   Input/Condition: Some components with a "border" CSS style attribute are loaded

   Output/Result: Every component with a "border" CSS style should likewise contain a "border-radius" style of value "4px", unless it is an image

   How test will be performed: Using automation tools, we will query for every web component containing a "border" CSS property, and verify that it likewise contains a "border-radius" property of value "4px" if it is not an "img" HTML component

Limited Colour Palette

1. NFR-LFR5-1

   Type: Automatic

   Initial State: A framework for styling is available

   Input/Condition: Some components are loaded in with custom colours

   Output/Result: Every component should NOT contain a colour that is outside the palette defined by a configuration file, where the palette should consist of no more than 6 colours, with any amount of variation in those colours permitted (e.g: "grey" and "light-grey" would be considered 1 colour here if they are under one custom colour category in the configuration)

   How test will be performed: Using automation tools, we will query the stylesheets and web components to verify no colour style from CSS or the given styling framework is used that is not defined in the colour palette configuration file, which contains predefined colour hex values

4.2.2 Usability

Simplicity with Little Technical Background

1. NFR-UHR1-1

   Type: Dynamic, Manual, Survey

   Initial State: The system is running without errors

   Input/Condition: The user will attempt to perform the basic actions of starting the app, searching for a work order, chatting to agree to a quote, scheduling an appointment, and closing the app

   Output/Result: The user will give a rating about how simple it was for each action to occur

   How test will be performed: Using survey questions, we will ask an individual who professes to have little technical background to perform the basic MVP actions of the customer in our application. From there, they will give a rating out of 10 on how easy it was to figure out and interact with the UI to complete the given tasks. The chat systems tests will have one person from the development team act as the chat partner.

   Survey questions are in the Appendix.

2. NFR-UHR1-2

   Type: Dynamic, Manual, Survey

   Initial State: The system is running without errors

   Input/Condition: The user will attempt to perform the basic actions of signing up, logging in, setting their profile, setting service options, chatting to agree to a quote, setting appointment availability, and logging out

   Output/Result: The user will give a rating about how simple it was for each action to occur

   How test will be performed: Using survey questions, we will ask an individual who professes to have little technical background to perform the basic MVP actions of a shop owner/employee in our application. From there, they will give a rating out of 10 on how easy it was to figure out and interact with the UI to complete the given tasks. The chat systems tests will have one person from the development team act as the chat partner.

   Survey questions are in the Appendix.

Accessibility

1. NFR-UHR2-1

   Type: Automatic

   Initial State: All web components are running properly

   Input/Condition: Some components are loaded in that contain interactions

   Output/Result: Every component should have certain HTML and CSS attributes that conform to accessibility standards

   How test will be performed: Using automation tools, we will query different pages' web components and make sure that those which contain interactions have "tabindex" and "role" HTML attributes filled out, as well as have up/down/left/right/enter-key keyboard event listeners, so that users can fully navigate using just the keyboard

Emphasized Interactive Components

1. NFR-UHR3-1

   Type: Manual, Dynamic, Survey

   Initial State: A page is loaded without errors

   Input/Condition: Some web page components are loaded containing both interactive components and non-interactive components

   Output/Result: The user shall give their guess as to how many interactive components there are on the given page

   How test will be performed: Using a survey, we will ask a user to view the pages that make up our application and ask how many interactive components there are. If the user can guess with 80 percent accuracy at the end of the survey, it will be deemed that the interactive components were emphasized enough for the user to easily see them.

   Survey questions are in the Appendix.

Minimizing Clicks Per Interaction

1. NFR-UHR4-1

   Type: Automatic

   Initial State: The system is running with errors

   Input/Condition: All the designed web components are without errors

   Output/Result: Whether any web component has a double click event handler defined

   How test will be performed: Using automation tools, we will verify that no interactive component has any double click functionality, as this could simply be broken down into simpler single click interactions

Limit Input Fields

1. NFR-UHR5-1

   Type: Automatic

   Initial State: The system is running with errors

Input/Condition: All the designed web components are without errors

Output/Result: Whether a web component that is not of a specific type has any input fields

How test will be performed: Using automation tools, we will query the front-end's web components and verify that the only web components with an input field instead of a selectable list are in the authentication, chat, or profile pages, or are tagged with a "input-description" CSS class

4.2.3 Performance

System Uptime

1. NFR-PR1-1

   Type: Dynamic, Automatic, Stress

   Initial State: The system is successfully running without any issues.

   Input/Condition: An overwhelming number of requests to access the system at once will occur.

   Output/Result: The system should successfully remain online throughout these tests and users should continue to access and use the system without encountering any issues.

   How test will be performed: An automated DDoS stress testing application will attempt to crash the application with an overwhelming number of requests.

System Storage Usage

1. NFR-PR2-1

   Type: Structural, Dynamic, Manual

   Initial State: The system is running.

   Input: The tester uses some features on the Sayyara website, signs in, fills out forms, etc.

   Output: There should not be any memory being used by the application aside from the sign-in token.

How test will be performed: Using Google Chrome's 'Inspect Element' resources in the 'Application' tab, the tester can view storage and cache for the website. These features can be used to discern if anything other than the automatic sign-in token is taking up memory or storage and steps can be taken to prevent other features from storing information to ensure the application is using minimal resources.

Database Efficiency

1. NFR-PR3-1

   Type: Structural, Dynamic, Manual

   Initial State: The entire system (frontend, backend, database) is successfully running. The backend code will contain timer debug messages so the tester can determine how long it will take for these database requests to complete to ensure they meet the requirements.

   Input: Manual requests to get and update data in the database will be made on the application by the tester.

   Output: The backend debug messages will output the time it took for the request to complete.

   How test will be performed: The test will be performed manually by the tester inputting information onto the application. The tester will need to evaluate the time it takes for a database request to complete from the debug messages in the back end code. Unit tests for this will be developed in the future to eliminate the need for manual testing and will allow for more coverage.

4.2.4 Operational and Environmental

Internet Browser Compatibility

1. NFR-OER1-1

   Type: Functional, Dynamic, Manual

   Initial State: The system is running.

   Input: The tester will evaluate all features of the Sayyara website on various different web browsers and potentially different versions of these web browsers.

Output: The Sayyara website should function and all features on the website should work without any issues.

How test will be performed: A team member will test the application on the current latest version of each browser and ensure each feature on the application functions according to requirements. Additionally, usability survey questions will be distributed to external testers of the application to gather feedback on potential different browser versions and how they function with the Sayyara website.

4.2.5 Security

1. NFR-SR1-1

   Type: Dynamic, Automated

   Initial State: System is running with no errors

   Input: Passwords

   Output: True or False

   How the test will be performed: Automated tools will be used to create a number of different accounts with varying passwords that match with the system password compliance (e.g. password must contain a letter, number, etc.). The test will check if the database entry for these passwords are properly encrypted (i.e. plain text passwords are not stored in the database). If the encrypted version of the password matches its database entry, then the test case will pass.

2. NFR-SR2-1

   Type: Dynamic, Automated

   Initial State: System is running with no errors

   Input: Two test accounts, create and delete operations

   Output: Error message (you do not have access)

   How the test will be performed: Two internal test accounts will be set up. An automated tool will run create operations on the first account (creating shops, appointments, etc.). The second account will attempt to request a delete operation through the API on the newly created data nodes. The system

should return an error message to the tool. If none of the created data nodes were deleted, then the test case will pass.

3. NFR-SR3-1

   Type: Automated

   Initial State: System is running with no errors

   Input: Any database operation (CRUD), number of seconds between operations)

   Output: Temporary timeout (rate limit)

   How the test will be performed: Automated tools will be given a list of valid CRUD database operations that a typical user will be expected to be able to perform. The automated tool will run one random valid operation every X amount of seconds. If the number of operations exceeds the allowed amount within 15 seconds, the system should return some message that states that there will be a timeout. The test will pass if the timeout is given after the allowed number of operations are exceeded.

4.2.6 Cultural and Political

1. NFR-CPR1-1

   Type: Static, Automated

   Initial State: Front-end code can be accessed

   Input: Potentially offensive language

   Output: Locations of potentially offensive language (file, line, etc.)

   How the test will be performed: Automated tools will be used on the source code for the front-end of the system. A list of potentially offensive language will be passed to the automated tool. If the testing tool finds a match for any of the words or sentences in the code, all instances of the use will be returned as an output, this includes the file name, the file path, and the line number.

2. NFR-CPR2-1

   Type: Static, Manual

Initial State: System is translated and system is running with no errors

Input: Tester navigates the website

Output: Locations of any mistranslated sections or components

How the test will be performed: Native speakers of the translated language will be asked to navigate the web-app and be tasked with finding and mistranslated or untranslated components of the website. They will also report if any components are coherent and can be easily read by other native speakers.

3. NFR-CPR3-1

Type: Dynamic, Automated

Initial State: Chat system is running without errors

Input: Profanity

Output: Censored words

How the test will be performed: Automated tools will be given a list of known profanity and will slowly send these words in the chat system to an internal testing account. If the word is properly displayed as some number of asterisks (e.g. ***), then the test cases is considered a pass.

4.2.7 Compliance

1. NFR-CR1-1

Type: Static, Manual

Initial State: The system is running without errors.

Input: Web component with disclaimer is loaded during user sign up.

Output: Disclaimer should be emphasized to the user.

How the test will be performed: Tester will go through the account sign up process and ensure the disclaimer is shown and is large enough to be read easily and quickly. Tester will also ensure the box to click "I agree" is functioning correctly (i.e. user cannot create an account unless the box is checked).

2. NFR-CR1-2

Type: Manual, Survey

Initial State: The system is running without errors.

Input: Web component with disclaimer is loaded during user sign up.

Output: The user will answer whether or not they have seen and understood the disclaimer.

How the test will be performed: Users will be asked a short series of questions about the Sayyara product disclaimer. If at least 75% of users answer favourably, then the component will be deemed successful.

Survey questions can be found in the Appendix.

4.3 Traceability Between Test Cases and Requirements

Functional Requirements FR- removed from beginning of test case names to preserve space

Authentication and Account Profile Requirements
Covering sections 5.1.1, 5.2.2

	AR1-1	AR1-2	AR1-3	AR2-1	AR2-2	AR2-3	PAR1-1	PAR3-1	PAR6-1	PAR7-1	PAR7-2
AuR1	X	X	X								
AuR2				X	X	X					
AuR3						X					
SR1							X				
SR2							X				
PAR1							X				
PAR3								X			
PAR6									X		X
PAR7										X	X

Table 4: Traceability Matrix between Test Cases and Functional Requirements

Appointment and Quote Requirements
Covering sections 5.1.3, 5.1.4, 5.1.5, 5.1.6, 5.1.7

	ApR1-1	ApR3-2	ApR3-3	QR1-1	QR2-1
ApR1	X				
ApR2		X	X		
ApR3		X			
QR1				X	
QR3					X

Table 5: Traceability Matrix between Test Cases and Functional Requirements

Shop Lookup Requirements
Covering section 5.1.7

	ShL1-1	ShL1-2	ShL2-1	ApR3-2
SL1	X			
SL2		X		
SL3			X	
SL4				
SL5			X	
SL6				
SL7			X	X

Table 6: Traceability Matrix between Test Cases and Functional Requirements

Non-Functional Requirements
NFR- removed from beginning of test case names to preserve space

Look and Feel and Usability and Humanity Nonfunctional Requirements Covering sections 5.2.1, 5.2.2

	LFR1-1	LFR2-1	LFR3-1	LFR4-1	LFR5-1	UHR1-1	UHR1-2	UHR2-1	UHR3-1	UHR4-1	UHR5-1
LFR1	X										
LFR2		X									
LFR3			X								
LFR4				X							
LFR5					X						
UHR1	X					X	X				
UHR2	X							X		X	
UHR3									X		
UHR4	X							X		X	
UHR5	X								X	X	X

Table 7: Traceability Matrix 1 between Test Cases and Nonfunctional Requirements

Performance, Operational and Environmental, and Maintainability and Support Requirements Covering sections 5.2.3, 5.2.4, 5.2.5

	PR1-1	PR2-1	PR3-1	OER1-1
PR1	X			
PR2		X		
PR3			X	
OER1			X	
OER2			X	
OER3				
MSR1				
SR1	X			
SR3	X			

Table 8: Traceability Matrix 2 between Test Cases and Nonfunctional Requirements

Security, Cultural and Political, and Compliance Nonfunctional Requirements Covering sections 5.2.6, 5.2.7, 5.2.8

	SR1-1	SR2-1	SR3-1	CRP1-1	CPR2-1	CPR3-1	CR1-1	CR1-2
SR1	X							
SR2		X						
SR3			X					
CPR1				X				
CPR2					X			
CPR3						X		
CR1							X	X

Table 9: Traceability Matrix 3 between Test Cases and Nonfunctional Requirements

5 Unit Test Description

This section describes the unit testing plan for the given list of unit tests. The tests are organized by module and include the initial state and expected results for each test.

5.1 Customer Module

1. **Test Case:** Register a new customer
   Initial State: User fills out the registration form and submits the data
   Expected Results: A new customer is created with the provided data

2. **Test Case:** Sign in as customer
   Initial State: User fills out the sign-in form and submits the data
   Expected Results: User is authenticated with the provided data

3. **Test Case:** Search for customers
   Initial State: A non-customer is searching for customers
   Expected Results: A customer is found by the provided criteria

4. **Test Case:** View customer profile
   Initial State: A customer has navigated to their profile
   Expected Results: The customer sees the relevant data pertaining to their account

5. **Test Case:** Edit customer profile
   Initial State: A customer is editing their profile information
   Expected Results: The customer's profile information is updated

5.2 Employee Module

1. **Test Case:** Register a new employee
   Initial State: The shop owner has already sent an invite for the user. User fills out the registration form and submits the data
   Expected Results: A new employee is created with the provided data

2. **Test Case:** Sign in as employee
   Initial State: User fills out the sign-in form and submits the data
   Expected Results: User is authenticated with the provided data

3. **Test Case:** Search for employees
   Initial State: An employee is searching for other employees
   Expected Results: An employee is found by the provided criteria

4. **Test Case:** View employee profile
   Initial State: An employee has navigated to their profile
   Expected Results: The employee sees the relevant data pertaining to their account

5. **Test Case:** Edit employee profile
   Initial State: An employee is editing their profile information
   Expected Results: The employee's profile information is updated

6. **Test Case:** Delete employee profile
   Initial State: A shop owner is deleting an employee's profile ie, removing them from the shop.
   Expected Results: The employee profile is deleted from the database

5.3 Shop Module

1. **Test Case:** Register a new shop
   Initial State: User fills out the registration form and submits the data after signing up as a shop owner.
   Expected Results: A new shop is created with the provided data

2. **Test Case:** Sign in as shop
   Initial State: User fills out the sign-in form and submits the data as a shop owner
   Expected Results: User is authenticated with the provided data

3. **Test Case:** Search for shops
   Initial State: A customer is searching for shops in their area
   Expected Results: A shop is found by the provided criteria

4. **Test Case:** View shop profile
   Initial State: A customer has navigated to a shop's profile
   Expected Results: The customer sees the relevant data pertaining to the shop

5. **Test Case:** Edit shop profile
   Initial State: A shop is editing their profile information
   Expected Results: The shop's profile information is updated

6. **Test Case:** Delete shop profile
   Initial State: A shop is deleting their profile
   Expected Results: The shop profile is deleted from the database

7. **Test Case:** Add an employee to a shop
   Initial State: A shop is adding an employee to their shop
   Expected Results: An employee is added to the shop

8. **Test Case:** Remove an employee from a shop
   Initial State: A shop is removing an employee from their shop
   Expected Results: An employee is removed from the shop

9. **Test Case:** Add a service to a shop
   Initial State: A shop is adding a service to their shop
   Expected Results: A service is added to the shop

10. **Test Case:** Remove a service from a shop
    Initial State: A shop is removing a service from their shop
    Expected Results: A service is removed from the shop

5.4 Vehicle Module

1. **Test Case:** Add a vehicle to a customer's profile
   Initial State: A customer is adding a vehicle to their profile
   Expected Results: A vehicle is added to the customer profile

2. **Test Case:** Remove a vehicle from a customer's profile
   Initial State: A customer is removing a vehicle from their profile
   Expected Results: A vehicle is removed from the customer profile

5.5 Appointment Module

1. **Test Case:** Add an appointment to a shop
   Initial State: A shop is adding an appointment to their schedule
   Expected Results: An appointment is added to the shop schedule

2. **Test Case:** Remove an appointment from a shop
   Initial State: A shop is removing an appointment from their schedule
   Expected Results: An appointment is removed from the shop schedule

5.6 Availability Module

1. **Test Case:** Add an availability to an employee
   Initial State: An employee is adding an availability to their schedule
   Expected Results: An availability is added to the employee schedule

2. **Test Case:** Remove an availability from an employee
   Initial State: An employee is removing an availability from their schedule
   Expected Results: An availability is removed from the employee schedule

References

6 Appendix

6.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

TEST_USER: valid and existing user name ("test_user")

TEST_PW: valid password for TEST_USER ("Pass451!")

EMPLOYEE_NAME: existing employee name ("Test Employee")

TEST_MESSAGE: Sample text for message field ("This is some text.")

SAMPLE_NAME: Example name for a customer ("John Doe")

SAMPLE_DATE: Example date in YYYY-MM-DD format ("2022-01-13")

6.2 Usability Survey Questions

Usability questions for browser and browser versions

1. What internet browser are you using to access Sayyara?
2. What version of browser are you using to access Sayyara? (This can typically be found by going to Settings - About, or Help - About). Enter N/A if you are unsure.

Usability questions for a user acting as a customer. Specifically, asking for a 1-10 rating per the following interactions after using the app independently:

1. How easy was it to start the app as a customer?
2. How easy was it to search for a work order?
3. How easy was it to start a chat?
4. how easy was it to agree to a quote?
5. How easy was it to schedule an appointment with a shop?

6. How easy was it to close the app?

Usability questions for a user acting as a shop owner/employee. Specifically, asking for a 1-10 rating per the following interactions after using the app independently:

1. How easy was it to sign up for an account?
2. How easy was it to log in to your account?
3. How easy was it to setup your profile?
4. How easy was it to set service options?
5. How easy was it to use the chat system to agree to a quote?
6. How easy was it to set your appointment availability?
7. How easy was it to log out of your account?

Usability questions for a user looking at a page (where every page will be shown to the user one at a time) and asked the following question:

1. How many interactive components do you see, such as a button, dropdown, checkbox, etc?

Compliance questions for a user after going through the sign up process will be asked the following questions:

1. Have you properly read the Sayyara disclaimer?
2. Was the disclaimer emphasized enough for you during sign up?
3. Can you briefly summarize what you understand from the disclaimer (if you answered YES to Q1)?

Appendix — 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.

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?

Alyssa

1. There are quite a few testing methods that the team would like to implement to ensure sufficient testing coverage is acquired and one of the testing methods that I would like to learn more about and focus my attention on is static testing. I am vaguely familiar with the definition of static testing but have not had the opportunity to use it in any of my past projects.

2. In order to acquiring more knowledge about static testing, I will be inquiring my notes from previous school courses regarding testing as well as performing my own research into static testing. Additionally, in order to get hands-on experience with static testing, I will reference some previous projects I have worked on and will write a few static tests for the system in order to get a better understanding of how these tests should behave compared to already existing test cases that use a different type of testing.

Harsh

1. The team was eager to learn and use Docker in the project, which allows for the team to build a platform-agnostic application that can be tested in a whole host of different environments.

2. For Docker, the team mainly utilized the docker documentation as well as implementation examples from other projects to help learn the skill reliably.

Kai

1. The team needs to collectively investigate the various testing tools suitable for the tech stack used in the project. I am specifically interested expand my experience in testing by learning static testing for Django using Coverage.py for code coverage, and Cypress for dynamic end-to-end testing of the frontend.

2. For both of the testing tools, my approach to learning involves a combination of reading and doing. Reading the tool documentation and tutorials provide a sense of the use cases and basic usage of the tools. Applying the basic usage on a simple project, then incrementally exploring the tool features offer more practical hands-on experience in using the tools and testing methods.

Ethan

1. Automated testing for the front-end of the app written in React is something that I have never done before, but will be an essential part of verifying the user interface of the system. From some preliminary research it seems the standard tool for this is Jest. So, I will need to learn how to use this testing framework in order to complete the v&v plan.

2. The two possible approaches I thought of were (1) Learning primarily from documentation and tutorial videos, and (2) Learning through using the tool on a small example project. I believe the best way in this case will be to use a combination of the two approaches: create a small example project to test on, and use the documentation/tutorial videos to support my experimentation.

Collin

1. I have some experience with automated testing from my previous co-ops, however I have never worked with any tools for static and front-end testing. There are a significant amount of requirements that will require a form of static testing and I am interested in expanding my knowledge in that area in order to properly test this system. I will also need to learn about Django as this testing tool is a good option for out testing purposes.

2. I will read review past course material (e.g. 3RA3) and also read through articles that pertain to static testing. I will also read official tool documentation and watch tutorials on using Django and follow along in order to get first hand experience using the testing tools.

Christopher

1. I have experience from my previous internship doing unit testing on the front-end with frameworks like Jest. I'm quite familiar with querying generated web elements and verifying their properties are of what is desired. However, I am not familiar with testing any backend functionality, such as API routes. I've also never used a large fully-fledged testing framework like Cypress, so this will be something I'll need to learn.

2. There are many projects I am following, as well as projects I've worked on partially, that have backend systems with testing. I've also worked on projects that switched from Cypress to using purely Jest. Using these past experiences, I can look back at the code and find patterns and correlations from the type of testing I did to the testing I didn't do that I need to learn. I can also use many basic openly available projects that I'm following to read and learn their code to see how Cypress and some Django test cases were built.