System Verification and Validation Plan for Sayyara Automotive Matcher

Team 27, Kappastone
Tevis Doe, doet
Caitlin Bridel, bridelc
Gilbert Cherrie, cherrieg
Rachel Johnson, johnsr12
Harkeerat Kanwal, kanwalh
Himanshu Aggarwal, aggarwah

November 3, 2022

1 Revision History

Date	Version	Notes
2 November 2022	1.0	First iteration of V&V Plan

Contents

List of Tables

2 Symbols, Abbreviations and Acronyms

Acronym	Description
SRS	Software Requirements Specification
PWA	Progressive Web App

This document is the verification plan for the Sayyara Automotive Matcher. The primary focus of this document is to outline the plan for the testing of the Sayyara Automotive Matcher, ensuring that it meets the requirements detailed in the SRS. This document will go over the general information of the project, provide a section on the general planning for testing, and finally provide concrete tests for both the functional and non-functional requirements, taken from the SRS.

3 General Information

3.1 Summary

The Sayyara Automotive Matcher is a software that is intended to provide customers an easy way to find and meet with mechanics for their automobiles. It will allow shop owners to create their shop in an application, provide services to clients, book appointments with them, and provide them with quotes for their services. Clients will be able to search through shops in the system and send requests for quotes to many shops. Clients will also be able to quickly make appointments with shops for the services they provide that don't need a quote. The bottom line is that the Sayyara Automotive Matcher facilitates communication between clients and shop owners.

3.2 Objectives

The objective of this document is twofold: To provide both current and future developers an easy way to verify and validate both the requirements that have already been implemented in the project, as well as the future requirements, and also to demonstrate to any viewer that the software has reached a point where it can be adequately used as a minimal viable product.

3.3 Relevant Documentation

The relevant documentation for this project includes the Software Requirements Specification for the Sayyara Automotive Matcher.

?

4 Plan

4.1 Verification and Validation Team

To execute the team's verification and validation plan, each team member was given a role. The following table explains each member's responsibilities.

Table 1: Breakdown of team member testing roles

Team Member	**Role(s)**	**Role Description**
Tevis Doe	Back-End Tester	Write back-end tests using PyTest.
Caitlin Bridel	Documentation Reviewer	Review documentation and ensure that it meets standards.
Gilbert Cherrie	Front-End & End-to-End Tester	Write front-end unit tests and end-to-end tests.
Rachel Johnson	User Surveyor	Survey users for feedback on usability.
Harkeerat Kanwal	Front-End & End-to-End Tester	Write front-end unit tests and end-to-end tests.
Himanshu Aggarwal	Back-End Tester	Write back-end tests using PyTest.
Nabil Ibrahim	General Feedback	Will be referenced to provide general feedback as the project progresses. This ensures that our requirements and concept will stay relevant to Sayyara.

4.2 SRS Verification Plan

As this is an industry project, the SRS cannot receive feedback from fellow members of the course. Because of this, the verification of the SRS is more internal than other projects. For feedback from an outside source, the supervisor of this project, Nabil Ibrahim, will be consulted for feedback on the requirements in the SRS. When a requirement is added to or removed from the scope of the project, Mr. Ibrahim will again be consulted, depending on his availability. In addition to consulting the supervisor of this project, before implementing requirements into the system, a few questions will be asked:

1. Is the requirement relevant to the project?
2. Is the requirement impossible to verify?
3. Is the requirement achievable within the first phase of the project?

If the answer to any of these questions is no, then the requirement will be reexamined and removed, improved upon, or left as-is, depending on the discussion. It is hoped that this process will help verify the SRS as a whole.

4.3 Design Verification Plan

According to the Many Worlds interpretation of quantum mechanics, our design exists in a superposition of varying degrees of correctness. It follows that there exists a world in which our design is 100% correct and, given that such a world is guaranteed to exist, verification of our design is not necessary.

The above logic presents us with an issue, however. Though there exists a world with a perfectly correct design, it is unlikely to be the world we find ourselves in. So, how do we verify our design in this more likely scenario? Since our project is very much dependent on our partner, Sayyara's, vision, it is vital that our design is reviewed by our supervisor, Nabil Ibrahim. This review will take place during a meeting with our supervisor in which we describe and walk through our design, while receiving his feedback.

Additionally, the design shall be reviewed by all team members to ensure technical details are correct.

4.4 Verification and Validation Plan Verification Plan

The verification of this verification plan shall be outsourced to independent, third-party auditors, also known as "Teaching Assistants". These auditors shall review this document and evaluate it according to a predetermined set of criteria, eventually producing an evaluation score, also known as a "grade". This "grade" shall be given to the writers of this plan, also known as the "students", to serve as a gauge for the quality of this document.

4.5 Implementation Verification Plan

To test our implementation, the team shall implement all of the unit tests and end-to-end tests listed in this document, in addition to further unit tests discovered to be useful after implementation has been completed.

During development pull requests must be reviewed by at least one other team member before they can be merged into the main branch. This is a means of verification of the implementation via manual code inspection.

4.6 Automated Testing and Verification Tools

The tools we will be using, including testing libraries, frameworks, and continuous integration tools, have already been thoroughly detailed in our development plan. Below is a brief summary, but it is recommended that the reader look at "Section 6 - Technology" of our development plan to read a complete description.

For the front-end, the linter we are using is ESLint. ESLint can also be used to ensure full code coverage. Our front-end testing frameworks will be Jest and Cypress. Jest will be used for unit testing, while Cypress will be used for end-to-end testing.

For the back-end, the linter we are using is pylint. Our back-end testing framework will be pytest.

To facilitate continuous integration, we are using GitHub actions. As an additional tool that was not mentioned in our development plan becuse we had not planed to deploy our project, we are using Vercel to facilitate continuous deployment of both our main branch and all development branches of our GitHub repository.

4.7 Software Validation Plan

There is no external data that can be used for validation. Instead review sessions with our stakeholder shall be conducting to validate our software.

5 System Test Description

5.1 Tests for Functional Requirements

The following subsections cover the various functionalities of the system. Each set of test cases relate to a specific subsystem. Each test case relates to one ore more functional requirements as specified in the SRS.

5.1.1 Shop Owner Registration

1. FT-RT-1

   Control: Automatic

   Initial State: Add Shop Page

   Input: Valid shop details, such as address, availability and contact information is entered.

   Output: The shop is added to the shop owner profile.

   How test will be performed: The test will be performed using the **cypress** library to automatically fill in valid shop registration details.

2. FT-RT-1.1

   Control: Manual

   Initial State: Back end server is running and a shop was added through the automatic testing

   Input: None

   Output: The shop is added to the shop owner profile database.

   How test will be performed: The test will be performed by manually checking the database to verify that the shop was correctly added to the shop owner profile.

3. FT-RT-2

   Control: Automatic

   Initial State: Registration Page

   Input: A valid username, email, and password are entered.

   Output: The user is redirected to the 'Add Shop Page'.

   How test will be performed: The test will be performed using the **cypress** library to automatically fill in valid registration details to verify if the user is redirected to the 'Add Shop Page'.

5.1.2 Employee Account Creation, Edit and Deletion

1. FT-RT-3

   Control: Manual

   Initial State: Invite employee page

   Input: Employee emails are entered.

   Output: The employees receive an email allowing them to sign up.

   How test will be performed: The test will be performed by manually entering in emails to the invite employee page and then manually verifying that the emails were sent and received by the intended accounts.

2. FT-RT-4

   Control: Automatic

   Initial State: Employee email invitation received

   Input: Valid employee registration data including email, name, username and password is entered.

   Output: Employee added to shop page.

   How test will be performed: The test will be performed using the **cypress** library to verify that employees can create an account with valid registration information and join a shop using a given email invitation.

3. FT-RT-5

   Control: Automatic

   Initial State: Shop owner or employee account created and logged in

   Input: Valid shop owner or employee account information different from registration information is entered.

   Output: Shop owner or employee account is updated with new information.

   How test will be performed: The test will be performed using the **cypress** library to verify that shop owners and employees can edit their account information by entering valid data and confirming the account is updated with the new data.

4. FT-RT-6

   Control: Automatic

   Initial State: Employee account created and logged in

   Input: Employee account requests to be deleted.

   Output: Employee account is deleted and removed from shop.

   How test will be performed: The test will be performed using the **cypress** library to verify that employees can delete their account and the account will be removed from the shop it was associated with.

5. FT-RT-7

   Control: Manual

   Initial State: Shop owner logged in and shop has employees assigned to it.

   Input: Shop owner can view employee list and from there they can view employees, add employees, remove employees, search for employees and filter the employee list.

   Output: The shop employee has been added or removed and the employee list has been filtered or returned the search results.

How test will be performed: The test will be performed manually by viewing the employee page on a shop owner account and manually adding an employee, removing an employee, filtering the employee list and searching the employee list.

5.1.3 Shop Owner/Employee Login

1. FT-RT-8

   Control: Automatic

   Initial State: Shop owner or employee account created

   Input: Valid username and password is entered for a shop owner or employee account.

   Output: Shop owner or employee is logged in with access to their information and functionality.

   How test will be performed: The test will be performed using the **cypress** library by entering a valid username and password for a shop owner or employee account and then checking that the user is logged in with access to their relevant data.

2. FT-RT-9

   Control: Automatic

   Initial State: Shop owner or employee account created

   Input: Valid account email or username is entered and then new valid password is entered.

   Output: User account password is changed.

   How test will be performed: The test will be performed using the **cypress** library by entering a valid username or email for a shop owner or employee account then confirming that they receive a password reset email. It will then test that the user can change their password by entering a new valid password and verifying that they can log in with the new password.

5.1.4 Quotes, Appointments and Work Orders

1. FT-RT-10

   Control: Automatic

   Initial State: Submit quote request page

   Input: Quote request data is entered.

   Output: Quote request is submitted and can be viewed by a shop owner for a shop that the quote request was sent to.

   How test will be performed: The test will be performed using the **cypress** library by entering valid quote request data and submitting the quote request to a list of selected shops.

2. FT-RT-11

   Control: Automatic

   Initial State: Quote request is received.

   Input: Quote information is entered and sent to a customer from a shop owner.

   Output: Quote is created and can be viewed by customer.

   How test will be performed: The test will be performed using the **cypress** library by entering valid quote information for a received quote request. Then the test case will verify that the quote has been created and received by the customer.

3. FT-RT-12

   Control: Automatic

   Initial State: Quote is received by customer.

   Input: Appointment is created by customer and shop owner can accept, reject and assign the appointment.

   Output: The appointment has been accepted, rejected or assigned to an employee by a shop owner.

How test will be performed: The test will be performed using the **cypress** library by sending a valid appointment request from a customer and then the shop owner can accept, reject or assign the appointment.

4. FT-RT-13

   Control: Automatic

   Initial State: Quote is accepted by a customer, customer creates appointment and appointment accepted by shop owner.

   Input: Work order is created by automatically obtaining relevant information from the accepted quote.

   Output: Work Order has been created.

   How test will be performed: The test will be performed using the **pytest** library to verify that the work order was correctly created in the database.

5. FT-RT-14

   Control: Automatic

   Initial State: Work order is created.

   Input: Shop owner can send work order to the customer.

   Output: Customer has received and can view the work order.

   How test will be performed: The test will be performed using the **cypress** library by sending a work order to customer and verifying that the customer it was sent to has received the work order and can view it.

6. FT-RT-15

   Control: Automatic

   Initial State: Shop receives appointment request from customer.

   Input: Shop employee can view and accept appointment requests.

   Output: Appointment is created.

How test will be performed: The test will be performed using the **cypress** library to verify that the appointment is created after the employee accepts the appointment request and that employees can view the appointment after its creation.

7. FT-RT-16

   Control: Automatic

   Initial State: Shop is created and shop owner invites employees.

   Input: Shop employees can edit shop availability.

   Output: Shop availability updates to reflect changes made by employees.

   How test will be performed: The test will be performed using the **cypress** library to verify that changes made to shop availability by employees are visible after changes are saved.

8. FT-RT-17

   Control: Automatic

   Initial State: Shop completes a work order for a customer.

   Input: Customers can submit rework orders.

   Output: Shop receives rework order request from customer.

   How test will be performed: The test will be performed using the **cypress** library to verify that the rework order is visible to the shop after the customer submits the rework order request.

9. FT-RT-18

   Control: Automatic

   Initial State: Quotes are created.

   Input: Shop owners and employees can view, search for, and filter quotes.

   Output: Quotes are displayed when selected or searched for.

How test will be performed: The test will be performed using the **cypress** library to verify that the proper quotes are displayed when selected, filtered, or searched for.

10. FT-RT-19

    Control: Automatic

    Initial State: Work orders are created.

    Input: Shop owners and employees can view, search for, and filter past and upcoming work orders.

    Output: Work orders are displayed when selected or searched for.

    How test will be performed: The test will be performed using the **cypress** library to verify that the correct work orders are displayed when selected, filtered, or searched for.

11. FT-RT-20

    Control: Automatic

    Initial State: Appointment is created.

    Input: Work order is created automatically when an appointment is booked and deleted automatically when appointment is cancelled.

    Output: Both the appointment and work order become visible upon creation and both are no longer visible after cancellation.

    How test will be performed: The test will be performed using the **cypress** library to verify that both the appointment and the work order appear when the appointment is created and both disappear when the appointment is cancelled.

12. FT-RT-21

    Control: Automatic

    Initial State: Work orders are created.

    Input: Shop owners and employees can assign work orders to employees.

    Output: Employees receive work order assignment.

How test will be performed: The test will be performed using the **cypress** library to verify that the correct work order is assigned to the correct employee.

5.1.5 View/Edit Shops and Shop Details

1. FT-RT-22

   Control: Automatic

   Initial State: Shop is created by shop owner

   Input: Shop owner can edit their shop information.

   Output: The shop information has been edited with the new information.

   How test will be performed: The test will be performed using the **cypress** library to edit a shop's information with new information and then testing that the information has been updated with the new information that was inputted.

2. FT-RT-23

   Control: Automatic

   Initial State: Shop and shop profile are created.

   Input: Any user can view a shop profile.

   Output: Shop profile displayed to the user.

   How test will be performed: The test will be performed using the **cypress** library to verify that the shop profile is visible when a user chooses to view it.

3. FT-RT-24

   Control: Automatic

   Initial State: Customers access the system.

   Input: Customers can view, search for, and filter shops.

Output: Shops are displayed to customers when selected and searched for.

How test will be performed: The test will be performed using the **cypress** library to verify that the correct shop is displayed to the customer when selected, filtered, or searched for.

5.1.6 Call and Chat Functionality

1. FT-RT-25

   Control: Automatic and Manual

   Initial State: Shop is created and shop owner invites employees.

   Input: Shop owners and employees can call and chat with customers.

   Output: Shop owners and employees communicate with customers through the call and chat functionality.

   How test will be performed: The test for the chat functionality will be performed using the **cypress** library to verify that shops and customers can send and receive messages to and from each other. The test for the call functionality will be performed manually to ensure that shops and customers can also correspond through audio calls.

5.2 Tests for Nonfunctional Requirements

The following subsections are divided according to non-functional requirement sections in the SRS. Each subsection contains tests that verify relevant non-functional requirements.

5.2.1 Look and Feel

1. NFT-LF-1

   Type: Dynamic, Manual

   Initial State: The application is deployed to a web server.

   Input/Condition: The application is opened in front of the users.

   Output/Result: Within a few minutes, 90% of the users should start using the application without external prompt.

How test will be performed: A sample group of users will be shown the application on individual computers. No external commands will be given. Within a few minutes, at least 90% of the users should start exploring the application. This would reflect if the app is visually appealing.

2. NFT-LF-2

   Type: Dynamic, Manual

   Initial State: The application is deployed to a web server.

   Input/Condition: A sample of users are asked use the application.

   Output/Result: After a few minutes using the application, 85% of a representative sample of users should agree that the application has a professional and reliable feel.

   How test will be performed: After being introduced to the application for the first time, a group of individuals representing the target audience will be questioned about whether they feel that the application is professional and trustworthy. It should be considered successful if more than 85% of individuals agree that the application looks professional and trustworthy.

3. NFT-LF-3

   Type: Structural, Static, Manual

   Initial State: The application source code is downloaded onto a computer.

   Input: A team of developers is asked to go through the code and conduct an informal review of the fonts and styles used.

   Output: The developers shall not be able to find inconsistencies in the fonts and styles used throughout the application.

   How test will be performed: A team of developers will be given the source code on a computer and be asked to review the code to find inconsistencies in the fonts and styles used throughout the application. The developers should not be able to find any such inconsistencies.

5.2.2 Usability

1. NFT-UT-1

   Type: Dynamic, Manual

   Initial State: The application is deployed to a web server.

   Input/Condition: A diverse group of users is asked to navigate through the different pages and perform certain actions.

   Output/Result: The actions requested should be easily completed by at least 80% of the users who are 16 years of age or older in the time allotted.

   How test will be performed: A diverse group of users (including users of age 16 and above) will be asked to navigate and perform certain tasks across the different pages of the application. If at least 80% of the users are able to complete the tasks in a given time frame, it should be considered a success.

2. NFT-UT-2

   Type: Dynamic, Manual

   Initial State: The application is deployed and launched on a web server.

   Input/Condition: Users are asked to interpret the meaning of different icons used throughout the application.

   Output/Result: At least 90% of the users are able to recognize all of the icons used.

   How test will be performed: A diverse group of users will be shown the different pages of the application and asked to interpret the meaning of the various icons. It will be considered successful if more than 90% of the users are able to correctly decipher the meaning of all of the icons.

5.2.3 Performance

1. NFT-PF-1

   Type: Dynamic, Automatic

   Initial State: The application is deployed and launched on a web server.

   Input/Condition: Messages are sent from one user to another.

Output/Result: Messages should be received by the receiving user within 2 seconds.

How test will be performed: The **cypress** library will be used to simulate two instances of users. One instance will send a message to another to verify if the messages are received within the desired time. If the messages are received within 2 seconds for 90 percent of the times, and less than 5 seconds for the rest of the times, then the test would be considered successful.

2. NFT-PF-2

   Type: Dynamic, Automatic

   Initial State: The application is deployed and launched on a web server.

   Input/Condition: Different pages of the application are visited.

   Output/Result: The loading time between different pages shall be no more than 5 seconds.

   How test will be performed: The **cypress** library will be used to simulate navigation between different pages. The test will be considered successful if the application takes less than 2 seconds 95% of times, and less than 4 seconds rest of the times.

5.2.4 Operational and Environmental

1. NFT-OE-1

   Type: Dynamic, Manual

   Initial State: The application is deployed on a web server and is made available via a URL.

   Input/Condition: Users from different parts of the world are asked to access the application using the URL.

   Output/Result: The application is accessible regardless of the user's location.

   How test will be performed: Users from around the world will be asked to access the website using the provided URL. If the application is accessible from everywhere, this test would be considered a success.

5.2.5 Maintainability and Support

1. NFT-MS-1

   Type: Dynamic, Manual

   Initial State: The application is deployed on a web server.

   Input/Condition: Install and launch the application on different mobile and desktop devices, and on modern web browsers.

   Output/Result: The application should launch successfully on all devices tested on.

   How test will be performed: The application will be deployed on a web server. It will be accessed via a URL and run on modern web browsers. It will also be installed on different mobile and desktop devices as a Progressive Web App (PWA). This will be used to verify if the application can operate on multiple platforms.

5.2.6 Security

1. NFT-ST-1

   Type: Structural, Static, Manual

   Initial State: The application source code is downloaded onto a computer.

   Input/Condition: A security firm is asked to inspect the code to find potential areas where log reports can leak from.

   Output/Result: The security firm is unable to find any security risks related to log reports.

   How test will be performed: A security firm will be asked to inspect the code and find security risks where log reports can get accessed from someone other than the developers or other members of the company. The test will be considered successful if the security firm can certify that only company employees and developers have access to the internal bug and crash log reports.

2. NFT-ST-2

   Type: Structural, Static, Manual

Initial State: The application source code is downloaded onto a computer.

Input/Condition: A security firm is asked to inspect the code to determine if there is a flaw that allows users to directly access the database.

Output/Result: The security firm is unable to find any security risks related to database access.

How test will be performed: A security firm will be asked to inspect the code to see if there is a vulnerability that enables users to access the database directly. The test will be considered successful if the security firm can certify that the database cannot be accessed by the users directly.

3. NFT-ST-3

   Type: Dynamic, Automatic

   Initial State: The application is deployed on a web server.

   Input/Condition: The application is launched.

   Output/Result: A notification is displayed to notify the user that the application uses cookies.

   How test will be performed: The **cypress** library will be used to open the application and verify if a notification is shown related to storing of cookies.

5.3 Traceability Between Test Cases and Requirements

The following tables show the traceability between functional and non-functional requirements and the previously defined test cases.

Table 2: Traceability Table for Functional Test Cases and Requirements

Test Case ID	**Related Requirement(s)**
FT-RT-1	FR-1
FT-RT-1.1	FR-1
FT-RT-2	FR-2
FT-RT-3	FR-3
FT-RT-4	FR-4
FT-RT-5	FR-6
FT-RT-6	FR-7
FT-RT-7	FR-5, FR-13, FR-14
FT-RT-8	FR-8
FT-RT-9	FR-9
FT-RT-10	FR-29
FT-RT-11	FR-10, FR-11
FT-RT-12	FR-12

Test Case ID	**Related Requirement(s)**
FT-RT-13	FR-15
FT-RT-14	FR-16
FT-RT-15	FR-18, FR-28
FT-RT-16	FR-19
FT-RT-17	FR-21
FT-RT-18	FR-22
FT-RT-19	FR-23
FT-RT-20	FR-25, FR-26, FR-28
FT-RT-21	FR-27
FT-RT-22	FR-17
FT-RT-23	FR-20
FT-RT-24	FR-30
FT-RT-25	FR-24

Table 3: Traceability Table for Non-Functional Test Cases and Requirements

Test Case ID	**Related Requirement(s)**
NFT-LF-1	NFR-1
NFT-LF-2	NFR-4
NFT-LF-3	NFR-5
NFT-UT-1	NFR-6, NFR-7
NFT-UT-2	NFR-10
NFT-PF-1	NFR-11
NFT-PF-2	NFR-12
NFT-OE-1	NFR-19
NFT-MS-1	NFR-21
NFT-ST-1	NFR-22
NFT-ST-2	NFR-23
NFT-ST-3	NFR-25

6 Non-dynamic Test Description

This section will go over the any non-dynamic tests that the team plans to use to verify the software.

6.1 Code Review

The utilization of version control alongside code review will be used to systematically find faults in the software before it is integrated into the main branch. At least 1 approval of any modifications to the codebase is required

to be gotten before the modifications can be merged. In this way it is hoped that any bugs with code are found before they are merged (even if that notion is unrealistic).

6.2 Document Review

Document review will be carried out as required on documents at fixed times, at milestones of the project. They will be carried on each document as follows:

1. Upon completion of revision 0.
2. Upon the completion of any related document.
3. Before work is carried out to create the final project.
4. Before the deadline for the final documentation.

Document review may or may not include the supervisor depending on the relevance to the supervisor, and/or the availability of the supervisor. In the case the project supervisor is available, feedback will be elicited from him in meetings to ensure that documents remain relevant to the scope of the project. In the case that the supervisor is not available, internal review will be carried out on each document to ensure it aligns with the current state of the project.

6.3 Walkthroughs

Informal walkthroughs will be performed, as needed, when a given team member has issues thinking of test cases for a given component. As such, the primary purpose of walkthroughs for this project is to generate test cases, and figure out expected outputs to said test cases. The given team member and at least one other team member will group together, inspect a piece of code, and brainstorm different test cases. Once a valid test case is thought up, the group will draft different outputs to different inputs, making sure to include normal inputs, fringe cases, etc. Test cases generated through this method will then be turned into dynamic tests that will be run by the continuous integration portion of this project on all new changes to the system.

7 Appendix

7.1 Usability Survey Questions

The following survey will be filled out by members of the survey group to validate the system's usability.

Sayyara Automotive Matcher Survey

Please fill out the survey after using the application.

Time spent using software:

Device/OS used:

Provide a rating on a scale from 1 to 5 (where 1 represents a poor review and 5 represents a strong review) in each of these questions by filling in the number of circles corresponding to your desired rating.

How easy is the app to use?	○	○	○	○	○
	1	2	3	4	5
How appealing is the app visually?	○	○	○	○	○
	1	2	3	4	5

Sayyara Automotive Matcher Survey

Continued.

How professional and trustworthy does the app look?	○	○	○	○	○
	1	2	3	4	5
How responsive is the app?	○	○	○	○	○
	1	2	3	4	5
How easy are the icons to recognize?	○	○	○	○	○
	1	2	3	4	5

Appendix — Reflection

The successful completion of this verification and validation plan is dependent on the team's collective acquisition of necessary testing skills. The skills and/or knowledge that each team member will need are listed in Table A1.

Table A1: Team Member Required Knowledge List

Team Member	**Required Knowledge**
Tevis Doe	• End-to-End Testing • User Surveying • Jest Testing Framework
Caitlin Bridel	• Jest testing framework • Cypress testing framework
Gilbert Cherrie	• End-to-End Testing • User Surveying

Continued on next page

Table A1 – continued from previous page

Team Member	Required Knowledge
Rachel Johnson	• Jest testing framework • Cypress testing framework
Harkeerat Kanwal	• Jest testing framework • Cypress testing framework
Himanshu Aggarwal	• Pytest testing library • Cypress testing framework

Approaches

A possible approach to understanding the topic of End-to-End testing is to read online guides such as guru99 and browserstack. These websites provide an overview of relevant concepts, necessary steps and best practices. Another strategy is to watch tutorial Youtube videos that explain the topic. Both approaches are free but quality Youtube videos are less readily available.

To understand how to effectively survey users, team members can use online content such as Youtube videos and articles that define strategies to survey users. This approach is free and widely available online. Another strategy would be to learn through trial and error by creating small surveys and surveying friends and family to discover effective strategies. This method could provide hands-on, real-world experience but may be time consuming.

To get familiar with Jest, Pytest and Cypress testing frameworks, team

members can watch tutorial Youtube videos. These videos will provide an overarching crash course on the testing framework to allow team members to understand concepts and practices. Team members can also follow along on their own machines to further understand and have hands on learning. This approach would be free and easily accessible, but may not be as in-depth as other approaches. Another approach would be to take a Udemy course on the testing frameworks. This method will provide more hands-on learning but may also be much more time consuming. Finally, the team can use online documentation and tutorial websites to review the framework and get a better understanding of syntax and methods. This approach would require the team to dig through a lot of content and does not provide hands-on examples, but would be free and easily accessible.

Chosen Strategies

To gain knowledge in end-to-end testing, the team has decided to use online written guides on websites such as browserstack and guru99. This will allow them to move at their own pace and take notes on the subject. It is free to access and the sites provide high quality explanations. If the team requires further knowledge, they will look to Youtube to fill in the gaps.

To understand user surveying, the team will use online content such as videos and articles to discover best practices and apply it to the usability survey being used for this project. This is the least time-consuming approach and provides the best way to gain knowledge quickly.

To learn specific testing frameworks such as Jest, Pytest and Cypress, the team has chosen to use Youtube tutorials. Through research, the team was able to find a full and in-depth crash course on jest and cypress that will provide all the necessary knowledge with no cost to the team. To supplement our learning, the team will also reference official documentation for the frameworks to further understand syntax and specific methods we will be using.