Logo

Description automatically generated

ViroTour Software Test Plan

University of Maryland Global Campus

SWEN 670 : Team A

Spring 2023

Version 3.0

April 4, 2023

Document History

|  |  |  |
| --- | --- | --- |
| Version | Issue Date | Changes |
| 0.1 | 02/08/2023 | Initial Draft. |
| 1.0 | 02/12/2023 | Milestone 2 Submission |
| 2.0 | 03/25/2023 | Milestone 3 Re-submission |
| 3.0 | 04/03/2023 | Milestone 4 Re-submission |

Table of Contents

[1. Software Test Plan Purpose 5](#_Toc131531242)

[1.1 Objective 5](#_Toc131531243)

[1.2 Scope 5](#_Toc131531244)

[1.3 Roles and Responsibilities 6](#_Toc131531245)

[1.4 Definitions, Acronyms and Abbreviations 6](#_Toc131531246)

[2. Functional Success Test Cases 7](#_Toc131531247)

[2.1 Viewing the Image as a Sphere from Hotspots 8](#_Toc131531248)

[2.2 VR View Functionality 8](#_Toc131531249)

[2.3 Transition Between Images Functionality 9](#_Toc131531250)

[2.4 Smooth Transition Between Images 10](#_Toc131531251)

[2.5 Image Zoom and Pan Functionality 11](#_Toc131531252)

[2.6 Search by Text Functionality 11](#_Toc131531253)

[2.7 Image Glow Effect Functionality 12](#_Toc131531254)

[2.8 Hotspot Functionality 13](#_Toc131531255)

[2.8.1 Create Tour Functionality 13](#_Toc131531256)

[2.8.2 Edit Tour Functionality 14](#_Toc131531257)

[2.8.3 Delete Tour Functionality 15](#_Toc131531258)

[3. Non-functional Test Cases 16](#_Toc131531260)

[3.1 ViroTour Homepage Load Time 16](#_Toc131531261)

[3.2 ViroTour Image Display on Tour Page 16](#_Toc131531262)

[3.3 ViroTour Successful Opening 17](#_Toc131531263)

[4. Failure Confirmation Test Cases 18](#_Toc131531265)

[4.1 Hotspot Creation Without Name/Description. 18](#_Toc131531266)

[4.2 Uploading Unsupported Image Format 19](#_Toc131531267)

[5. Aspects not being tested 19](#_Toc131531268)

[6. Test Suite 20](#_Toc131531269)

[7. Test Report 20](#_Toc131531270)

[8. Defect Report 20](#_Toc131531271)

[9. Test Risks 21](#_Toc131531272)

[10. Testing Schedule 21](#_Toc131531273)

[11. Pass/Fail Criteria 23](#_Toc131531274)

[12. Test Environment 23](#_Toc131531275)

[13. Non-Functioning Test 24](#_Toc131531276)

[14. Test Deliverables 24](#_Toc131531277)

# Software Test Plan Purpose

This software test plan (STP) is an essential component of software development projects as it outlines the testing approach, objectives, and activities that are required to validate the software's functionality and quality. Having a well-defined test plan in place helps to ensure that the testing process is systematic, thorough, and efficient. The test plan lays out the scope of testing, including the types of tests that will be performed, such as, manual testing, system testing, and acceptance testing. It also defines the criteria for determining whether the software is ready to be released.

In addition to providing a roadmap for the testing process, this test plan also helps to identify potential risks and issues early in the development cycle. This allows the development team to address these problems before they become more significant and cause delays in the project timeline. The test plan also provides stakeholders with a clear understanding of the testing effort required and the resources needed to complete the testing process successfully. This helps to ensure that adequate resources and budget are allocated for testing, reducing the risk of cost overruns and project delays. Overall, a well-written test plan is crucial for the success of any software development project as it helps to ensure that the end-product meets the specified quality and functionality requirements.

## 1.1 Objective

The objective of this STP is to use the previously completed Software Requirements Specification document to identify product requirements and help determine the most crucial areas of testing importance. Each of the testing axes will include breakdowns of the functionality, code structure, user interface elements, internal/external interfaces, input/output space, physical components, data, platform and environment information, configuration elements, and performance details. Each of these axes will also have estimates that help explain how long each test case will take to execute.

Automated testing software will be implemented in the future once the software is at a state near release. This software will include Selenium and the built-in Flutter testing packages. The Selenium WebDriver will be used to test the web-based application of ViroTour. The Flutter testing packages will be used to test the mobile device-based application of ViroTour. Both Selenium and Flutter are open source and completely open for the ViroTour project to utilize. This software will greatly improve ViroTour’s ability to run tests throughout the application’s lifetime and help find/document issues/bugs.

## 1.2 Scope

The scope of this STP is to design the test plan for the frontend of ViroTour Application Development. Test Cases are divided into functional and non-functional test cases. Functional test cases test the functionality of the system, according to the specified requirements, and ensure that the system behaves as expected with valid input(s). Non-functional requirements evaluate the systems non-functional characteristics, which can include performance, security, etc. This test plan document will detail both the functional and non-functional test cases that relate to ViroTour’s requirements.

## 1.3 Roles and Responsibilities

The main responsibilities associated with ViroTour’s testing plan include test planning, test case development, test environment setup, test execution, test closure/documentation, and test reporting. Test planning includes defining the scope, objectives, and possible approach to the testing throughout a software project. Test case development includes writing and reviewing test documents to ensure that they are comprehensive enough to cover project requirements. Test environment setup includes configuring the testing environment that will be used by testers to be as closely aligned with production deployments as possible. Test execution includes the running, verifying, and reporting of tests in the test plan. Test closure mainly contains the creation of documentation and archiving test artifacts necessary for future reporting and testing. Test reporting involves the business analyst reviewing the test reports and including the customer when necessary for decision making.

Each of these responsibilities will be assigned to developers, testers, and project managers that align with their abilities. The following roles and responsibilities detail ViroTour’s test plan:

|  |  |  |
| --- | --- | --- |
| Number | Role | Responsibilities |
| 1 | Testers | Test Execution, Test Closure/Documentation |
| 2 | Developers | Test Case Development, Test Environment Setup |
| 3 | Project Manager (PM) | Test Planning |
| 4 | Test Managers | Test Reporting |

Table 1.3 - Roles and Responsibilities

## 1.4 Definitions, Acronyms and Abbreviations

The following are definitions, acronyms and abbreviations that might be mentioned in this document and need more clarity.

* API – Application Programming Interface
* Hotspots - transition points, as in points that indicate a direction of a new image
* Hotspot Information Points - points of interest, information in the form of text and potentially a picture that is docked to a point in a picture
* iOS – iPhone Operating System
* Mbps - megabits per second
* QA – Quality Assurance
* SRS – Software Requirement Specification
* STP – Software Test Plan
* TDD – Technical Design Documentation
* UMGC – University of Maryland Global Campus
* UI - User Interface
* UX – User Experience
* VR – Virtual Reality

# Functional Success Test Cases

Writing test cases for software features provides several benefits in software development. Firstly, it ensures that all aspects of the feature are thoroughly tested, increasing the likelihood that any bugs or issues are caught early in the development process. This can save time and resources compared to finding and fixing problems later in the development cycle or after the software has been released. Secondly, test cases serve as a form of documentation for developers, providing a clear and concise description of how the software should behave in specific and detailed scenarios. This can be useful for future reference and for new team members who may need to maintain or update features. Thirdly, having a comprehensive set of test cases can also help to improve the overall quality of the software by encouraging a thorough and systematic testing approach. This can increase customer confidence in the software and reduce the risk of negative feedback and support requests from future users.

Each of the test cases listed in this document were created by determining the most important use cases/requirements of the project and the main features that users will interact with. Testing what users will see is important because it helps to ensure that the user interface (UI) and user experience (UX) of the software are intuitive, visually appealing, and most importantly, functional. This document contains both success test cases and failure test cases. Success test cases focus on ensuring that the software operates as intended. Failure test cases focus on confirming that certain functionalities fail when all requirements are not met. The success test cases covered in this document are as follows:

## 2.1 Viewing the Image as a Sphere from Hotspots

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | View images as a sphere from hotspots test |
| Description | Users can automatically view all existing tour pictures via Hotspots. |
| Requirements | The tour must have images with location hotspots. |
| Prerequisites | The user must have access to ViroTour. |
| Steps | 1. Click on a hotspot. 2. Try to view the image without clicking on a hotspot. 3. Move the cursor within the image.   Zoom in and out using the mouse wheel or pinch gesture, applicable to mobile. |
| Expected Output | 1. The image should be displayed in a sphere. 2. The view should change accordingly, allowing the user to explore.   The image should enlarge or reduce in size, maintaining the sphere. |
| Assumptions | 1. Location hotspots are accurately placed in the virtual tour. 2. There are no issues with ViroTour.   The user has a mouse or a touchscreen device. |

Table 2.1 - Test Plan: Viewing the Image as a Sphere from Hotspots

## 2.2 VR View Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Virtual reality view test |
| Description | Verify that users can view the images using a VR viewer. The user will be able to enjoy a virtual reality experience of tours. |
| Requirements | A tour must be available. |
| Prerequisites | 1. A virtual tour has been created with selected images. 2. The user owns a VR headset. |
| Steps | 1. User selects the VR View button. 2. The tour is started in VR View   Observe if the image is displayed in the VR viewer |
| Expected Output | The current location’s image is displayed in VR View. |
| Assumptions | The user has a connected VR headset which is used to view the images. |

Table 2.2 - Test Plan: VR View Functionality

## 2.3 Transition Between Images Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Transition between images test |
| Description | Verify that users can move from one hotspot to another. |
| Requirements | User selects a locational hotspot.. |
| Prerequisites | User opens application to start tour. |
| Steps | 1. The user selects a hotspot to move to.   The tour moves to the next location’s image. |
| Expected Output | The user can view tour hotspots or images one after another by clicking on the transition points. When a user clicks/taps on a transition point, he should be moved to another location or image. |
| Assumptions | The user selects the images/hotspots manually. |

Table 2.3 - Test Plan: Transition Between Images Functionality

## 2.4 Smooth Transition Between Images

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Smooth transition between images test |
| Description | Verify that the transition from one image to the next image is smooth, transition time should be less than a second and transitioning images should have a blur effect applied on them. |
| Requirements | 1. Transitioning from one image to the next takes the system no longer than one second. 2. Transition between images is smoothed via a blurring effect. |
| Prerequisites | 1. The viewed tour has multiple images.   The images have transitional hotspots |
| Steps | 1. Open the app and navigate to any image in the tour. 2. Measure the time it takes to transition from one image to the next. 3. Repeat step 3 multiple times as you record the time each transition takes.   Check if the blur effect is present during the transitions. |
| Expected Output | Each transition takes no longer than a single second and the blurring effect is present each time. |
| Assumptions | The transition time should be constant even when there are many images in a tour. |

Table 2.4 - Test Plan: Smooth Transition Between Images

## Image Zoom and Pan Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Image Zoom and Pan Test |
| Description | Verify that users can zoom and pan at each hotspot. |
| Requirements | Can only zoom in/out of one image at a time. |
| Prerequisites | User views an image. |
| Steps | 1. User opens the ViroTour Application and begins a tour.   The user zooms in and out on an image using a mouse action, pinch gesture, or the onscreen zoom bar. |
| Expected Output | The user can zoom in and out of a picture, enlarging it or making it smaller. |
| Assumptions | The zoom functionality works on all respective platforms; mobile, desktop, and web applications.  Zoom feature controls behave in a similar manner across all applications. |

Table 2.5 - Test Plan: Image Zoom Functionality

## 2.6 Search by Text Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Text search test |
| Description | Verify that the users can search for text that has been extracted from the tour’s uploaded images. |
| Requirements | The search string of text must be accurately matched with the text displayed in an image’s informational hotspot. |
| Prerequisites | All images have been uploaded and processed. |
| Steps | 1. The user types a search term on the search bar.   The user selects a single item/object in returned results to view the location of the text. |
| Expected Output | The system should return a list of objects if the search term is present in any of the objects.  Tapping one of the objects should take you to the location of the text. |
| Assumptions | 1. The application should have the ability to correct errors that may be present in the search term. An error in the search term should still yield the intended search outcomes which would include spelling errors and other grammar errors.   The application should extract text from images on upload and index them in the database. |

Table 2.6 - Test Plan: Search by Text Functionality

## 2.7 Image Glow Effect Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Image glow test |
| Description | Verify that users can add a glow effect to the view. |
| Requirements | The user can customize the tour lighting effect by activating the glow effect functionality on the interface. Use view\_tour\_vr branch when testing as this feature has not been merged to the main branch. |
| Prerequisites | The tour is opened and positioned at the first hotspot. |
| Steps | 1. Open a tour and be positioned at the first hotspot. 2. Click on the wheel menu and select glow effect.   Move the slider on the bar from left to right to apply varying degrees of brightness. |
| Expected Output | The brightness level on a tour can be adjusted to the specified intensity. |
| Assumptions | 1. A tour has been opened and the first image has loaded. |

Table 2.7 - Test Plan: Image Glow Effect Functionality

## 2.8 Hotspot Functionality

### 2.8.1 Create Tour Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Tour creation test |
| Description | Verify that users can create a tour |
| Requirements | User logs into the application and navigates to the Hamburger Menu. |
| Prerequisites | The application is open. |
| Steps | 1. Click on the hamburger menu to open a dropdown list of menu options. 2. Select the create tour option to open the create tour screen. 3. Fill in the name and the description in the input fields. 4. Upload tour images using the select images button.   Click save to upload to create a tour or cancel to cancel the create tour process. |
| Expected Output | 1. A new tour is created.   The tour is visible in the tours list view.. |
| Assumptions | 1. The database is correctly implemented, and data can be saved and retrieved.   Anyone who has access to the application can refresh to view the newly added tour. |

Table 2.8.1 - Test Plan: Create Tour Functionality

### 2.8.2 Edit Tour Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Editing Existing Tour test. |
| Description | Verify that a user can edit textual information related to a tour. |
| Requirements | User logs into the application and navigates to the Hamburger Menu. |
| Prerequisites | There is at least one available tour. |
| Steps | 1. The process of editing a tour consists of changing associated descriptive information (name and description) of a tour. Open the ViroTour Application. This should open a view with a list of all tours. 2. Open a tour you wish to edit. 3. Click edit.   Make the desired changes to the name and/or description.  Click save to update the information. |
| Expected Output | The changes made to the tours are visible to people viewing the tour. |
| Assumptions | 1. The database is correctly implemented, and data can be saved and retrieved.   Anyone who has access to the application can refresh to view the changes made |

Table 2.8.2 - Test Plan: Edit Tour Functionality

### 2.8.3 Delete Tour Functionality

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Deleting Existing Tour test |
| Description | Verify that a user can delete a tour |
| Requirements | User logs into the application and navigates to the Hamburger Menu |
| Prerequisites | There is at least one available tour. |
| Steps | 1. Navigate to the tour list page. Select a tour by clicking the edit icon on the right of the tour. 2. Click the red “Delete” button. A pop-up will be shown for the user to verify that they would like to delete the tour. 3. Select “Delete” in the pop-up.   A message will display any necessary information that was received from the server upon deletion of the tour. |
| Expected Output | The tour is removed from the tour list. |
| Assumptions | 1. The database is correctly implemented, and data can be deleted.   Anyone who has access to the application can refresh to view the changes made. |

Table 2.8.3 - Test Plan: Delete Tour Functionality

# Non-functional Test Cases

## 3.1 ViroTour Homepage Load Time

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | ViroTour homepage load time test |
| Description | The load time of the ViroTour homepage is very important for the overall impression users will have of the software. |
| Requirements | The homepage loads within 5 seconds on internet speed that exceeds 10 Mbps. |
| Prerequisites | The user opens the ViroTour application in any web browser or mobile device. |
| Steps | 1. The user opens a web browser and inputs the URL of ViroTour or opens the ViroTour application on a mobile device.   The ViroTour homepage renders completely. |
| Expected Output | The ViroTour homepage should render completely in under 5 seconds. |
| Assumptions | The user must be using a Chromium-based browser (Google Chrome/Microsoft Edge), or a mobile device (iOS/Android). |

Table 3.1 - Test Plan: ViroTour Homepage Load Time

## 3.2 ViroTour Image Display on Tour Page

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | ViroTour images display on tour page test |
| Description | Verify the image is displayed for each location in a tour. |
| Requirements | The user should see the images displayed as a sphere at each location. |
| Prerequisites | The user opens ViroTour app on their device and select a tour. |
| Steps | 1. The user opens the app. 2. The user selects a tour. |
| Expected Output | The user sees the images displayed as a sphere at each location of the tour. |
| Assumptions | The user must be using a Chromium-based browser (Google Chrome/Microsoft Edge), or a mobile device (iOS/Android). |

Table 3.2 - Test Plan: ViroTour Image Display on Tour Page

## 3.3 ViroTour Successful Opening

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Virtual tour opening test |
| Description | Opening a virtual tour from the homepage allows users to enter a virtual tour’s images. |
| Requirements | User selects one of the virtual tours from View Tours menu |
| Prerequisites | Successful loading of ViroTour home page. |
| Steps | 1. User selects *View Tours* from Hamburger menu. 2. ViroTour displays list of tours. 3. User selects desired tour. 4. ViroTour displays the selected tour. |
| Expected Output | ViroTour displays a tour matching the selection from the menu |
| Assumptions | 1. Home page is loaded to allow user to display screen items. |

Table 3.3 - Test Plan: ViroTour Successful Opening

# Failure Confirmation Test Cases

Testing for failures is an important part of software testing because it helps ensure that the system is robust, reliable, and can handle unexpected conditions. The following test cases focus on expected failures:

## 4.1 Hotspot Creation Without Name/Description.

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Hotspot creation failure test |
| Description | When creating a hotspot, both a name and description is required. |
| Requirements | User attempts to create a hotspot without inputting a name/description. |
| Prerequisites | Successful loading of ViroTour home page. |
| Steps | 1. User opens the create hotspot UI element  2. User does not put in a hotspot name, description, or both.  3. User clicks on the create hotspot button. |
| Expected Output | ViroTour displays a message indicating that the creation failed because a hotspot requires both a name and description. |
| Assumptions | 1. Tour page is loaded to allow user to create a new hotspot. |

Table 4.1 - Failed Test Case: Hotspot Creation Without Name/Description.

## 4.2 Uploading Unsupported Image Format

|  |  |
| --- | --- |
| **Item** | **Description** |
| Name | Image format test |
| Description | ViroTour only supports uploading images in PNG or JPEG formats. Any other image formats should not be accepted by the front end. |
| Requirements | User attempts to upload an unsupported image file in an unsupported format. |
| Prerequisites | Successful loading of ViroTour home page. |
| Steps | 1. User clicks on the upload image button.  2. User uploads an unsupported image format (HEIF/BMP). |
| Expected Output | ViroTour displays a message saying that the image uploaded is in a format that is not supported by the system and gives a short message about the supported formats. |
| Assumptions | 1. Homepage loaded successfully for user to click upload button. |

Table 4.2 - Failed Test Case: Uploading Unsupported Image Format

# 5. Aspects not being tested

The following items are currently outside of the scope of the project or not the priority for the current cycle of development and are therefore not included in the testing. These items may either be outsourced to a different team or may be included in future iterations of development.

|  |  |
| --- | --- |
| **Item excluded from testing** | **Reason** |
| Any load time associated with server-side processing (building a tour, retrieving a tour, etc.) | This is outside of scope for team A and will be tested by team B |
| Issues with filling a screen up with too many hotspots | Not currently a priority, but may be addressed in future development |
| Responsiveness of menu on all mobile devices | This should be handled by flutter. We lack the infrastructure to test on all kinds of devices so this testing will have to either be outsourced or wait for a future development cycle. |
| Resistance to SQL injection or other attacks aimed at the back end | Currently not a priority but will likely need to be addressed in future development. |

*Table 5 – Items not Being Tested*

# 6. Test Suite

Test suites are made up of test cases. Typically, large applications have many different test suites for each of the parts of the application. ViroTour is a medium-sized application and does not require more than one test suite.

# 

# 7. Test Report

Test reports will be created at the completion of each test suite’s run. The test report includes the test case names, outcomes, and any additional comments that may better explain the outcome of each test. These reports help developers document and understand the issues throughout the development cycle.

# 8. Defect Report

This Section will list any identified defects. The QA analyst should be the one who writes and executes unit test. The analyst completes the test case document with the errors found and ensuring the problem can be reproducible. The test cases go back to the developer team to be fixed. After the problem is fixed, the QA team will rerun the unit tests as part of regression testing, running not only one test case that identified the problem but all test cases to ensure the fix did not cause other issues.

# 9. Test Risks

Software testing always has some risks attached. These risks can include incomplete testing, human error, integration/compatibility issues, testing environment problems, and time/cost constraints. Each risk and its cause should be identified, the likelihood of it happening should be addressed. A risk management plan should be created. The risk management process needs a schedule to determine how often and when risk activities should occur throughout the project. Qualitative risk analysis qualifies the risks that have been identified in the project. Not all risks are worth responding to, but some demand attention. Qualitative analysis is a subjective approach to organizing and prioritizing risks. Identified risks can be rated according to probability and potential impact. An assessment has been made and these are the main testing risks associated with the ViroTour project:

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Risk | Impact | Mitigation Plan |
| 1 | Changing requirements | Medium | Choosing a software development framework, where progress is shown to the customer in increments and feedback is given often, can help mitigate the impact of changing requirements. |
| 2 | Insufficient test coverage | Low | Weekly meetings for the testing team can help improve planning and coordination in creating new tests. |
| 3 | Deployment environment changes | Medium | Weekly meetings between the development team and test team can give increased warning for any system-altering changes and test updates can be made ahead of time. |
| 4 | Human error | Low | The testing repository will have a mandatory two-person review on any testing code that is merged into the main branch. |
| 5 | False positive and/or false negative test results | Low | Any time a test fails/passes in an unexpected way, the test team will conduct a group review to find the root cause and fix any issue as soon as possible. |

Table 9 – Test Risks Matrix

# 10. Testing Schedule

Typical software development projects have multiple stages. The first stage is unit testing, followed by integration testing, system testing, and finally, acceptance testing. Each phase is unique and focuses on different parts of the overall system. Unit testing verifies the functionality of individual components of the software. For ViroTour’s case, this could include testing the hotspot creation/edit/deletion feature. Integration testing focuses on how each individual feature works with each other and determines if there are any compatibility problems.

System testing is more rigorous and tries to confirm whether or not the software meets the requirements and specifications defined in the project’s early planning documents. The final step, acceptance testing, is conducted with the help of users and the customer, to determine if the software meets their expectations and needs before full release to production. ViroTour will utilize each of these testing steps to create a uniform and repeatable testing schedule. This will also provide quality assurance, improved testing efficiency, and an overall better experience for the user.

Milestone 3 will have five sprints where the team develop and test the product simultaneously. During Sprint 1, the team will spend two weeks on developing the product (demo). Entering Sprint 2, testing will start immediately after the demo is ready starting with Unit Test for all components. The team continues to work on the product for Sprint 3 & 4 and performing Integration Test, System Test, and User Acceptance Test. Sprint 5 is to complete all testing and the Test Report.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MS3 Sprints** | **Work Area** | **Testing** | **Duration (days)** | **Start** | **Finish** |
| Sprint 1 | Develop | Developing Product and first demo | 14 | 2/12 | 2/26 |
| Sprint 2 | Develop & Test | Develop & Execute Unit Test for Component 1 | 7 | 2/26 | 3/5 |
| Sprint 2 | Develop & Test | Develop & Execute Unit Test for Component 2 | 7 | 2/26 | 3/5 |
| Sprint 2 | Develop & Test | Develop & Execute Unit Test for Component 3 | 7 | 2/26 | 3/5 |
| Sprint 3 | Develop & Test | Develop & Execute Integration Test for Component 1 | 7 | 3/5 | 3/12 |
| Sprint 3 | Develop & Test | Develop & Execute Integration Test for Component 2 | 7 | 3/5 | 3/12 |
| Sprint 3 | Develop & Test | Develop & Execute Integration Test for Component 3 | 7 | 3/5 | 3/12 |
| Sprint 4 | Develop & Test | Develop & Execute System Test for Component 1 | 7 | 3/12 | 3/19 |
| Sprint 4 | Develop & Test | Develop & Execute System Test for Component 2 | 7 | 3/12 | 3/19 |
| Sprint 4 | Develop & Test | Develop & Execute System Test for Component 3 | 7 | 3/12 | 3/19 |
| Sprint 4 | Develop & Test | Develop & Execute User Acceptance Testing | 7 | 3/12 | 3/19 |
| Sprint 5 | Test | Test Report | 33 | 2/26 | 3/31 |

Table 10 – Develop and Test Schedule

# 11. Pass/Fail Criteria

ViroTour’s pass/fail criteria will be broken up into the following sections:

|  |  |
| --- | --- |
| **Pass Criteria** | **Description** |
| Functionality | The software performs as expected and meets all functional requirements covered in earlier planning documents. |
| Performance | The software performs at the optimum level describes by earlier planning documents. UI response times, overall load time, and image processing time will all be considered for ViroTour. |
| Security | The software must protect against simple and complex attacks. These attacks can come in the form of script abuse or data injection. |
| Compatibility | The software must be compatible with the targeted environments (Browser, iOS, and Android). |
| **Fail Criteria** | **Description** |
| Bugs | The software must not contain any critical bugs that prevent the overall functionality of the system. |
| Performance | The software must meet all performance goals determined in previous planning documents. These include UI response times, overall load time, and image processing time. |
| Security Vulnerabilities | The software must not have any critical security vulnerabilities that could lead to exploitation or attacks that damage the integrity of the application. |

Table 11a and 11b – Pass/Fail Criteria

# 12. Test Environment

Application functionality on desktop computers and mobile phones

ViroTour comes with mobile-device based application and web-based application. The applications will be tested separately by making use of suitable testing environment for both. To support the testing effort in the project, the packages that will be used are Selenium WebDriver and Flutter testing packages. Selenium is an open-source testing tool that allows testing of web applications across different browsers. Among Selenium suite of software, Selenium WebDriver will be used to test the web-based application of ViroTour. Similarly, Flutter testing packages which also are open-source, will be used to test the mobile-device based application of ViroTour.

# 13. Non-Functioning Test

* Security Testing- Ensure that the application has no loopholes which could lead to any data loss or threats. It is one of the important aspects of non-functional testing and if not performed properly, it can lead to security threats.
* Load Testing- Evaluates whether the system’s performance is as expected under normal and expected conditions.
* Stress Testing- Evaluates whether the system’s performance is as expected when it is low on resources.
* Recovery Testing- Evaluates that the application terminates gracefully in case of any failure and the data is recovered appropriately from any hardware and software failures.
* Scalability testing- Verify if the application is capable enough to handle increased traffic, number of transactions, data volume, etc. The system should work as expected when the volume of data or change in the size of data is done.
* Disaster Recovery Testing- Verify the success rate of recovery of an application or system if any critical failure happens and whether the system is able to restore the data and application or the system could cope up easily to return the way it was working earlier i.e., from the operational front.
* Maintainability Testing- Once the application goes live, then there are chances for an issue to come up in the live environment or the customer may want an enhancement for the application which is already live. In this case, maintenance testing team is available to test the above scenarios mentioned. Once the application goes live it still needs maintenance for which the maintenance testing team works.

# 14. Test Deliverables

End end product of the entire testing process conducted by the testing team will be gathered into a Test Report. The key deliverables contained within are the following:

* Executive Summary - Which describes overall activities of the mobile application tests.
* Requirements Traceability Matrix
* Test Strategy
* Defect Summary
* Test Cases
* Test Summary Report.