Programmer's Guide

University of Maryland Global Campus
SWEN 670 – Team A
Spring 2023
Version 2.0

April 4, 2023

Document History

Version	Issue Date	Changes
0.1	3/22/2023	Initial Version
1.0	3/25/2023	Milestone 3 Submission
2.0	4/4/2023	Milestone 4 Submission

Contents

1. Introduction

1.1. Purpose

The ViroTour Application Programmer's Guide walks the readers through tool setup, demonstrates the application design, covers repository management, and explains the decision-making process of the development team.

The guide provides information about the syntax and usage of programming constructs, as well as practical examples and tips for using the software or tool effectively. It also provides guidance on best practices, common pitfalls to avoid, and troubleshooting techniques.

1.2. Intended Audience

The ViroTour Application Programmer's Guide is intended for software developers and programmers who are responsible for designing, developing, and maintaining the ViroTour application. This programmer's guide is written in technical language and assumes that the reader has some prior knowledge of programming concepts and terminology.

1.3. Technical Project Stakeholders

The technical stakeholders of the project are listed below:

Stakeholder Name	Project Role
Dr. Mir Assadullah	Client/Professor
Roy Gordon	Project Mentor
Robert Wilson	DevSecOps Mentor
Khoa Nguyen	Project Manager (PM)
Jacob Lynn	Product Owner (PO)
Viet Nguyen	Lead Developer (Dev Lead)
Tilahun Abreha	Business Analyst (BA)
Fedor Menchukov	Software Engineer (Dev)
Jude Ibe	Software Engineer (Dev)
Nicholas Platt	Software Engineer (Dev)
Samson Alemneh	Software Engineer (Dev)
Jeffrey Welch	Software Engineer (Dev)
Shawn Kagwa	Software Engineer (Dev)
Christian Dovel	Software Engineer (Dev)

Table 1.3 - Technical Project Stakeholders

1.4. Project Documents

There are various documents created for this effort to provide the stakeholders, namely the project team, the client, and external users with sufficient information and understanding for the success of the project. These documents are summarized below.

	Document	Version	Date
1	Project Management Plan (PMP)	4.0	4/4/2023
2	Software Requirements Specification (SRS)	4.0	4/4/2023
3	Technical Design Document (TDD)	3.0	4/4/2023
4	Software Test Plan (STP)	2.0	4/4/2023
5	Programmers Guide (PG)	2.0	4/4/2023
6	Project Deployment and Operations Guide (DevOps)	2.0	4/4/2023
7	User Guide (UG)	1.0	4/4/2023
8	Test Report (TR)	1.0	4/4/2023

Table 1.4 - Project Documents

1.5. References

- Using packages. Flutter. (n.d.). Retrieved October 22, 2022, from https://docs.flutter.dev/development/packages:and:plugins/using:packages
- Google. (n.d.). Create and set up your app - play console help. Google. Retrieved October 23, 2022, from https://support.google.com/googleplay/android:developer/answer/9859152?hl=en

1.6. Definitions, Acronyms, and Abbreviations

- API - Application Program Interface
- Flutter - is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.
- GB – Gigabyte
- GPU – Graphical Processing Unit
- Hotspots - transition points, as in points that indicate the 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.
- HTTP - Hypertext Transfer Protocol layers of the network protocol stack.
- IDE - Integrated Development Environment
- iOS - iPhone Operating System
- MB - Megabyte

- OS - Operating System
- REST - Representational State Transfer
- SDK - Software Development Kit
- SRS – Software Requirement Specification
- STP – Software Test Plan
- SWE - Software Engineer
- TDD – Technical Design Documentation
- UC – Use Case (referring to the Use Cases in the Software Requirement Specification)
- UI - User Interface
- UMGC – University of Maryland Global Campus
- UN – United Nations
- URL - Uniform Resource Locators
- USB – Universal Serial Bus
- VR – Virtual Reality
- VT – ViroTour
- WCAG – Web Content Accessibility Guidelines
- Wi-Fi – Not an Anacronym - a facility allowing computers, smartphones, or other devices to connect to the internet or communicate with one another wirelessly within a particular area.

2. Operating Systems and Development Tool Versions

The Operating Systems and Development Tools used in the development of ViroTour are as follows:

Product Name	Version
Windows 10	OS Build 19044
Mac OS Ventura	13.1
Android Studio Giraffe	2022.3.1
GitHub Desktop	3.2 (x64)
Flutter	3.7.2
Dart	2.18
Git	2.37.1

Table 2 - Project Software Tool Versions

3. Development Environment

3.1. Integrated Development Environments (IDE)

The ViroTour application is built with functionality that will provide a central interface which aids in application development. The Integrated Development Environment will consist of the source code editor, a build automation tool, and a debugger. The importance of an IDE in the ViroTour application development process cannot be overstated, as it streamlines the workflow and enhances the productivity of developers. IDEs offer a comprehensive set of tools inside one unified platform, which reduces development time and improves software quality.

3.1.1. Android Studio

Android studio provides a unified environment where applications can be built for android applications. The structured code allows division of projects into units of functionality that can be built independently, tested, and debugged. Considering that Flutter is a versatile software framework designed for creating high-quality, high-performance mobile and desktop applications, which cater to a wide range of operating systems, it is crucial to utilize an IDE that fully supports its capabilities. Android Studio is an excellent choice for developing Flutter applications for the Android platform.

3.2. Emulators and Simulators

For the ViroTour application to be tested in a near native, software-defined environment, emulators, and simulators were used. The emulators will attempt to emulate the actual hardware that will host the ViroTour application in production. On the other hand, the simulators are designed to create an environment that contains all the software variables and configurations that will exist in the ViroTour's actual environment. Under section 4 of the Project Deployment and Operations Guide, different types of emulators and their configuration steps are included.

4. Frameworks and Languages

4.1. Flutter

Flutter is a popular framework for building cross-platform mobile applications with a single codebase. This cross-platform support includes iOS, Android, Web, Desktop (MacOS, Windows, and Linux), and embedded systems. One of the main advantages of Flutter UI is the ability to create beautiful and responsive user interfaces that work across platforms. The framework uses a reactive programming model that promotes fast rendering, beautiful animations, and a smooth user experience. Flutter also provides a feature-rich suite of widgets that can be used to match any application's use cases.

From a developer's perspective, Flutter provides many different benefits. The "hot reload" feature allows developers to make changes to the code and quickly see the results in real-time, which can drastically improve the speed of the development process. Flutter also has a large and supportive community, which offers many open-source packages through the pub.dev repository.

4.1.1. Flutter Doctor

Flutter Doctor is a command-line utility that comes with the Flutter SDK and is used to help developers quickly diagnose and fix issues with the Flutter installation. Running the command 'flutter doctor' conducts a scan and produces a report of your local Flutter environment. This report includes information about your system, installation paths, and any missing or outdated dependencies. Flutter Doctor is an essential tool for developers and helps ensure that their local environment is configured correctly for development.

```
C:\>flutter  doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.3.10, on Microsoft Windows [Version 10.0.22000.1574], locale en-US)
[X] Android toolchain - develop for Android devices
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line for more details.
[√] Chrome - develop for the web
[!] Visual Studio - develop for Windows (Visual Studio Community 2022 17.4.4)
    X Visual Studio is missing necessary components. Please re-run the Visual Studio installer for the "Desktop
      development with C++" workload, and include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
         - If there are multiple build tool versions available, install the latest
        C++ CMake tools for Windows
        Windows 10 SDK
[!] Android Studio (version 2022.1)
    X Unable to find bundled Java version.
[√] VS Code, 64-bit edition (version 1.74.3)
[√] Connected device (4 available)
[√] HTTP Host Availability

! Doctor found issues in 3 categories.
```

Figure 4.1.1 Flutter Doctor Summary

4.1.2. Pub Package Manager

Pub Package Manager is used by Flutter to manage the Dart packages inside the Flutter project. The pubspec.yaml file imports Dart dependencies inside the Flutter project. The pubspec.yaml file keeps all the configuration items for Flutter app.

4.1.3. Hot Reload and Hot Restart

These two features in Flutter help decrease the execution time of an app once it is executed. The features can be used if the program is executed once, and they are faster than the default restart. Hot reload is the easiest and the fastest function that takes place in approximately one second to apply changes, fixes, added features etc. In general, hot reload helps to see the effect of some changes in the code.

A hot restart as compared to a hot reload takes more time, but it takes less time than the full restart function. The way a hot restart works is that it destroys the state of the app and gets the code compiled again and starts from the default state.

A hot reload can be performed using 'ctrl+\' in Windows or using the hot reload button. 'cmd+s' can be used in Mac devices to perform a hot reload. When using the command prompt using flutter run enter 'r' to run. We perform hot restart using ctrl+shift+\.

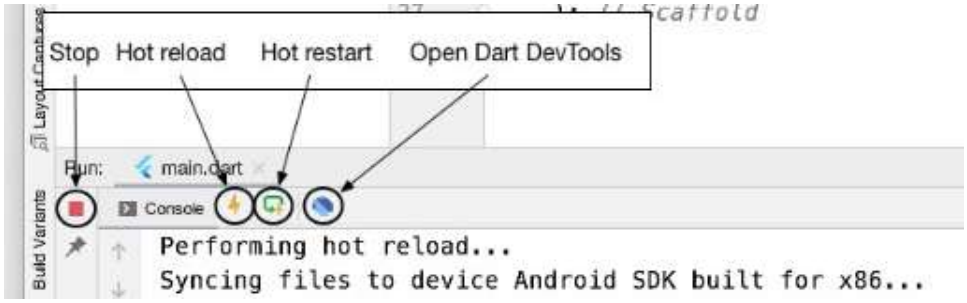

Figure 4.1.3 Hot Reload and Hot Restart Buttons

4.1.4. Flutter Outline

Flutter Outline is a feature in the Android studio that shows the tree structure of the entire code base. Basically, it shows the tree structure of all the widgets which can be very helpful when there is a long file full of code where it is difficult to locate a specific part. The outline makes it easy to locate the specific part of the code that we are looking for.

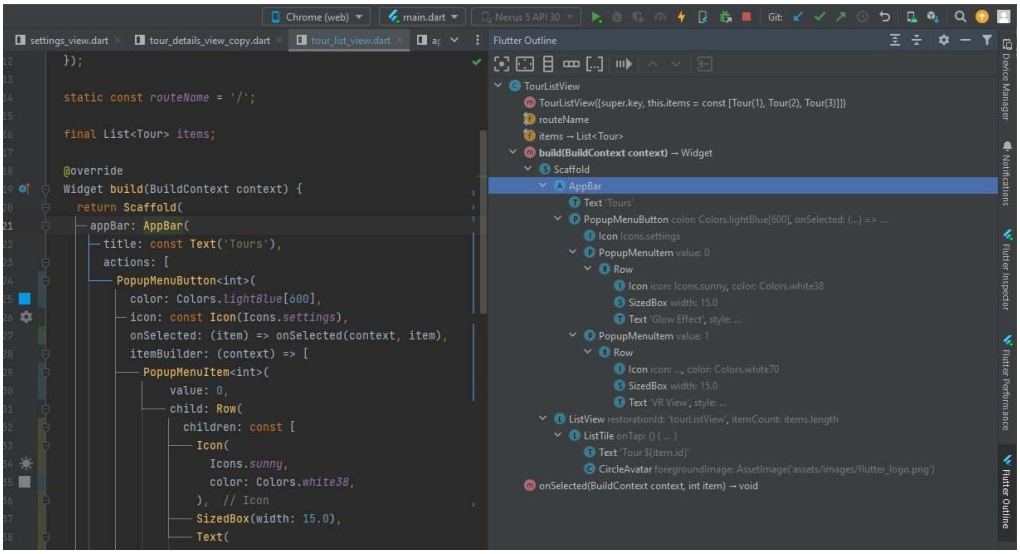

Figure 4.1.4 Flutter Outline

4.1.5. Flutter Inspector

The Flutter Inspector is a very useful tool for visualizing and exploring the Flutter widget trees. Since Flutter framework uses widgets as the building blocks for developing apps, the inspector helps to explore Flutter widget tree to understand existing layouts and to diagnose layout issues (https://docs.flutter.dev/development/tools/devtools/inspector). Flutter Inspector serves a similar purpose as the Chrome developer tool. For Flutter Inspector to work, Emulator of one kind (Android, iOS, or Web) should be running. Flutter Inspector is useful for debugging and inspecting components.

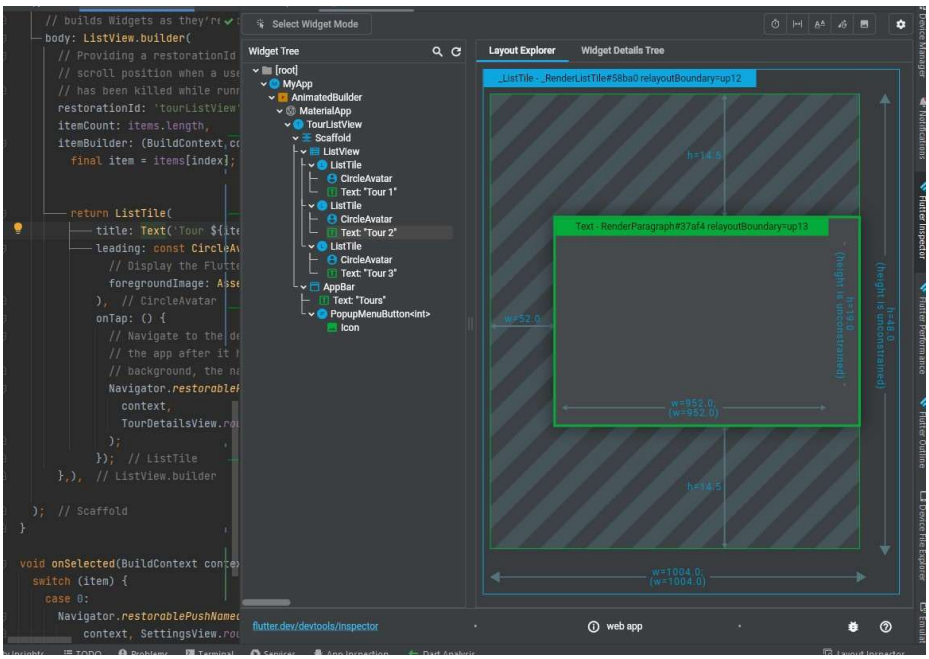

Figure 4.1.5a Flutter Inspector – Layout Explorer

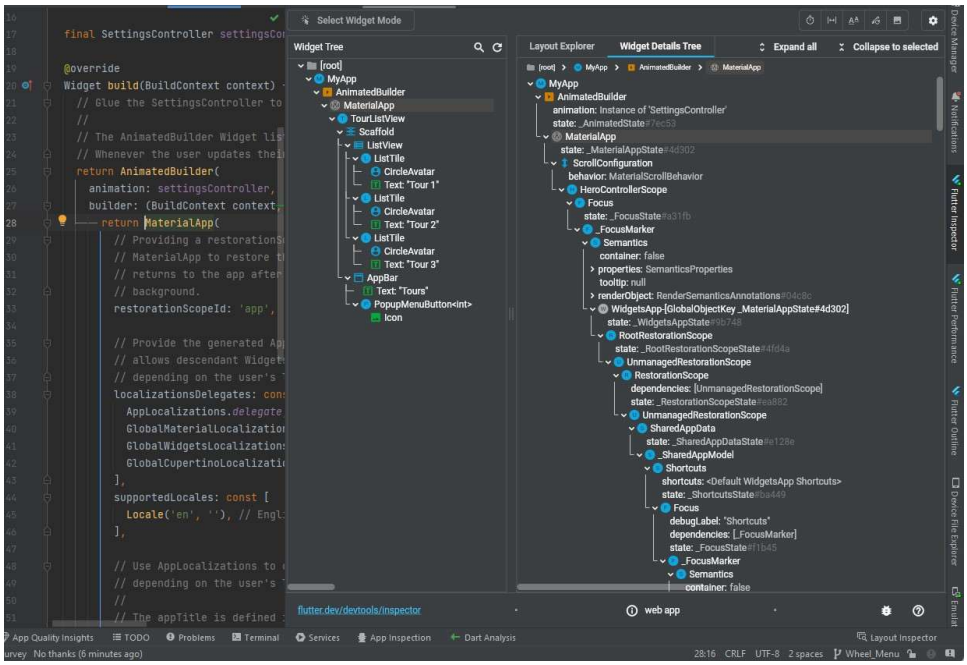

Figure 4.1.5b Flutter Inspector – Detail Tree

4.2. Dart

Dart is a free and open-source mobile development Software Development Kit (SDK) by Google for developing applications for Android and iOS. Being concise, fast, and reliable, Dart supports multiple languages, including Java, JavaScript, and Python. Dart also comes with an extensive package library that makes it easy to build complex applications (https://novateus.com/blog/flutter-vs-dart/).

Dart is a type-safe general-purpose programming language, and it is unique in its ability to compile to multiple targets. It also has a Just-In-Time compiler which compiles the source to machine code on the fly.

4.2.1. Type Safe and Null Safety

The language Dart is “type safe” to mean that a variable’s value always matches a static type which significantly decreases runtime errors in production. Dart also provides null safety in that values cannot be null unless they are explicitly allowed to, a feature that helps cut down runtime exceptions. Dart also has a package manager called ‘pub’ that provides thousands of open-source packages that help developers achieve their goals easily.

5. Development Process

The repository is located on GitHub at https://github.com/umgc/spring2023.git. It can be accessed via the command line or through GitHub Desktop, which is a GUI approach to accessing the repository. GitHub is used for team collaboration providing version control, a place to download the most recent code version, and teams to manage and make changes at a central location.

5.1. GitHub

The development tool utilized by the team is GitHub. This is a web-based, version control and collaborative platform for software development. This tool was used to store the source code for the project and also used to track changes as it happens to the code. The team's work has been done within https://github.com/umgc/spring2023.

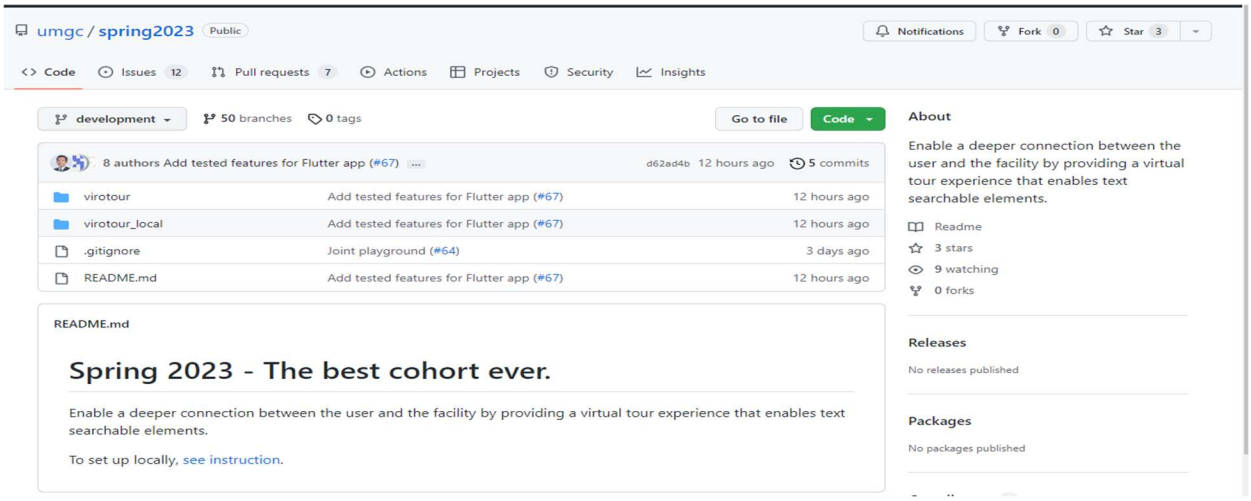

Figure 5.1 - The Project GitHub Interface

5.2. Cloning

The team's repository is located on GitHub at umgc/spring2023: SWEN 670 Spring 2023 cohort (github.com). Navigating to GitHub.com, you will have the option to download a zip archive or clone the repository as shown by the following image. If you only need to access the files for a short time, then downloading the zip file may make the most sense. However, in a development scenario when you'll need to consistently download updated files, cloning a repository will allow you to maintain an updated copy locally, while also pushing updates back to the server.

To clone the repository, ensure you have Git installed on your computer, and copy the link provided. Open a Git Bash or Git Terminal window and enter git clone <enter paste in the URL provided from GitHub>.

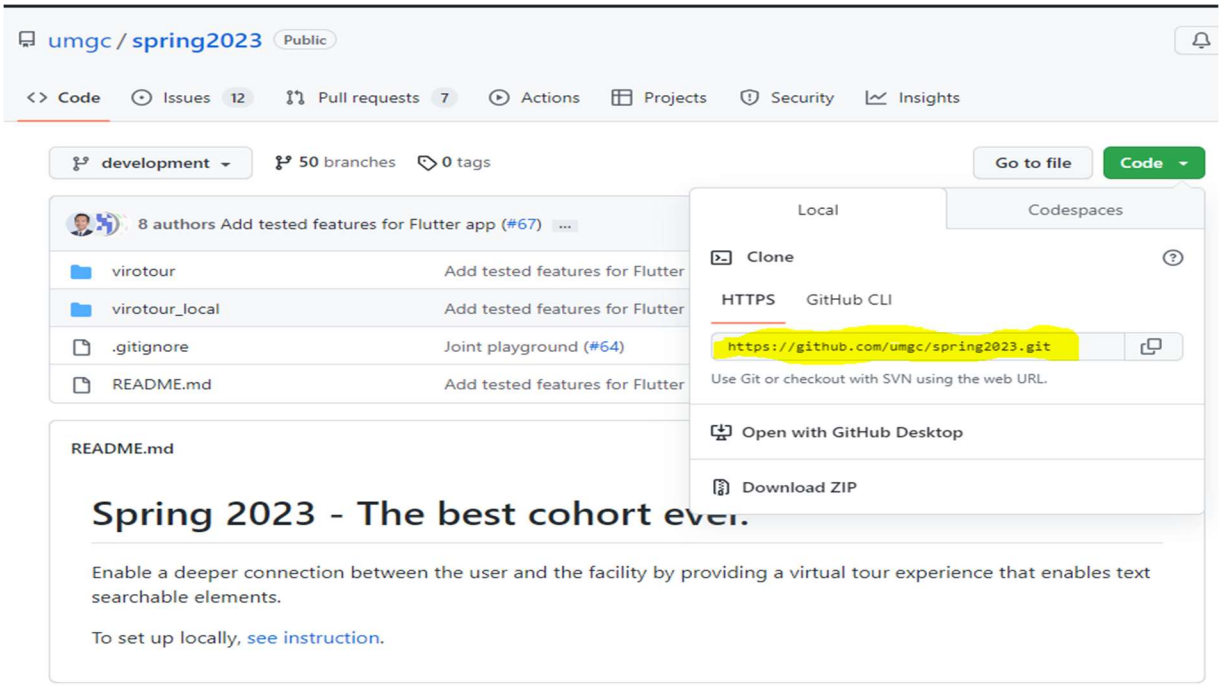

Figure 5.2 - Diagram showing the URL (Highlighted)

5.3. Branching

The team used branches to fix bugs, develop features or safely experiment with new ideas within the contained area of the repository. The reason this is important is that it isolates the main code from any wrong codes that would be detrimental to the whole project.

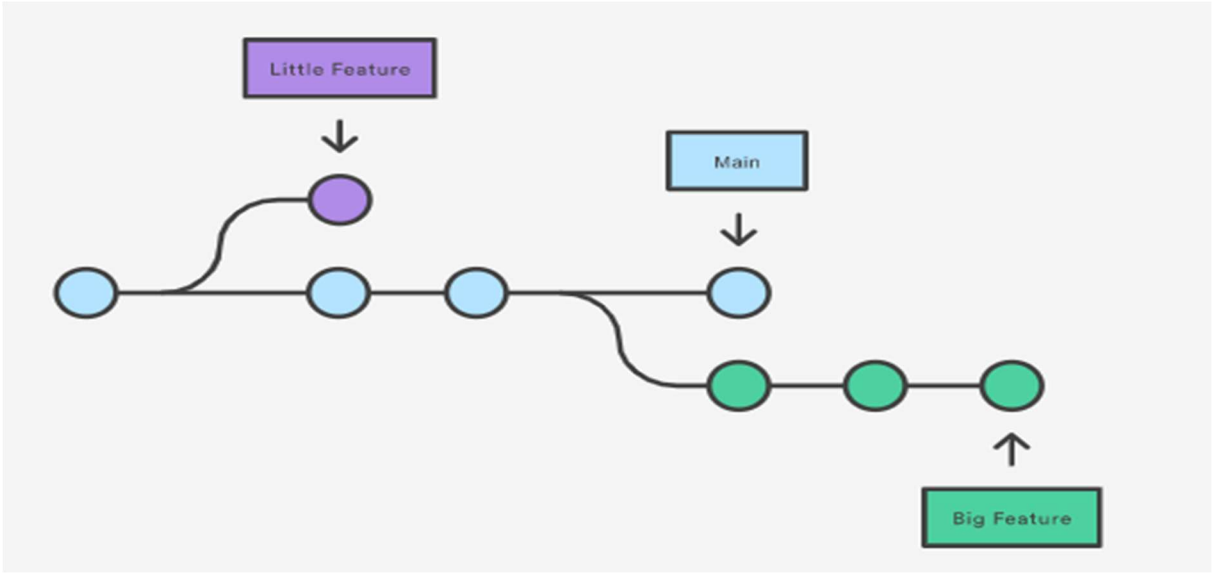

Figure 5.3 - Visuals of a Repository

In the diagram above, a repository can be shown to have two lines that are isolated from the Main code. By developing codes in branches, one can work in parallel and keep the Main branch free from questionable code.

The GitHub Branches breakdown has a Master, developer, and Feature. Master is a default name used when the first branch is first created. All repositories have one since the git init command creates it by default. The branch contains the production code, and all development code is merged into master in some time. It requires the DevSecOps and Project Manager (PM) approval before updating this branch.

A developer branch is created from the master branch. It is where all changes made by developers are held before merging back to the master for release, whereas a Feature branch is a supporting branch created from the developer. It is where a new feature is coded and tested by a team member until it is complete and ready for integration. Once satisfied with the work, a pull request is opened to merge the changes in the feature branch into developer.

Two manual approvals are required by team members. This branch is used for deploying the application to the App stores supporting Android and iOS.

5.4. Merging

Once the work on the branch is complete, the next step is to merge the feature branch or bug fix branch into a master branch. Merging takes the changes made on the branch and implements incorporates them into the parent branch.

Steps to Merge Branches in GitHub Desktop

- In GitHub Desktop, click **Current Branch**
- Click Choose a branch **developer.**
- Click the **feature** branch to merge to developer then click **Merge Branch into developer**. (If there is a merge conflict, GitHub will warn the user)
- Click **Push origin** to push your local changes to the remote repository.

5.5. Pull Request

A pull request is the act of notifying the team that you have completed a set of changes and are requesting to incorporate those changes into the master (developer) repository. This process requires that you have already pushed a feature branch to the repository and are now requesting the additional review and sign-off of at least two team members to requesting the target repository to grab changes. In GitHub, once the branch has been pushed, click on create pull requests and select the team members to inform them about changes pushed to a branch in the repository. Once a pull request is opened, changes are discussed and reviewed by team members/developers, and add follow-up commits before merging into the developer branch.

5.5.1. Creating a pull request

- Switch to the branch that you want to create a pull request for.
- Click Create Pull Request.
- On GitHub, confirm that the branch in the base: the drop-down menu, is the branch where you want to merge your changes.
- Type a title and description for your pull request.

6. Code Structure

The structure of the code for the ViroTour application has two categories, the front end, and the back end. The front end of the app was developed using the Flutter framework and Dart programming language. The features of the front end are divided into three general components as Tour Navigation, View Customization, and Hotspots Customization. Under these broad categories, there are specific uses case that are realized by building/coding different features that make up the app. These features include Tour List Page, View Tour, Create Tour, Edit Tour, Search, Hamburger Menu, Wheel Menu, Enable VR, and Glow Effect. The code for the components is placed in different .dart files inside the src directory in the project files. We refer the features below to the Use Cases (UC) in the Software Requirement Specification (SRS).

6.1. Tour Navigation

6.1.1. Tour List

The Tour List is the first page that the user sees after clicking on the ViroTour icon on their device to access the application. This page shows all the tours that are stored in the ViroTour database. The Tour List also includes an option to edit each tour, through an edit icon at the end of each tour's row. The Tour List fetches data from the /api/tours endpoint and maps them into a list of tours, which include the tour id, name, and description. A 'ListTile' is then created for each tour inside the list of tours returned from the 'fetchData' function. The list item includes the tour name and the description, as well as an edit icon that allows the user to change the name and description of the selected tour. When clicking on a list item (a tour), a MaterialPageRoute is used to open the TourDetailsView and passes along the required tour's id, which is used to create the tour.

6.1.2. Hamburger Menu

The hamburger menu is always displayed on the top left corner of the screen. When clicking it, the dropdown menu shows multiple options, including "Create Tour", "Search Tours", "Search Hotspots", and "View Tours". The "Create Tour" option takes the user to the CreateTourView, which allows a user to enter a tour name, description, and upload images they want included in the tour. The "Search Tours" option takes the user to the SearchPage view, which allows users to enter a string of text and any tour name or description that matches the string will be displayed in a list. The "Search Hotspots" option takes the user to the HotspotSearchView, which allows users to enter a string of text and any tour that contains

hotspots that match the string will be displayed in a list. The “View Tours” option will take the user back to the tour list view.

6.1.3. Wheel Menu

The Wheel Menu is always displayed on the top right of the screen. The Wheel Menu, when clicked, opens a screen with three options which allow to enable the VR View, enable Glow Effect, and change Theme. The wheel menu is placed on the appbar using {List<Widget>? actions}. The “Onpressed” in the widget defines the action that is fired when the wheel icon is selected which is to navigation to the “settings_view.dart” file where the code for the navigation is placed. The “Settings” page displays three options, VR View, Glow Effect and change Theme. Two Elevated Buttons with Onpressed actions are used to enable VR View and Glow Effect. While “Change Theme” option uses a dropdown menu that contains three dropdown menu items, System Theme, Light Theme and Dark Theme to change the app theme as needed.

The initial code for the Wheel Menu was written to display a Popup Menu with two menu items, the Glow Effect and VR View when the Wheel Menu button is selected; however, it was later decided to add two Elevated buttons on the dedicated settings page which contains the Theme selection dropdown menu instead of the popup menu.

6.1.4. Create Tour

This page allows users to create a new tour. The new tour contains a tour name, description of the tour, and a list of transitional hotspots (also known as locations) which are comprised of multiple images. Both the tour name and description are strings. Images are selected from the user’s device via the ImagePicker library and its web counterpart available to flutter. These allow image uploads from Google Chrome, Android, and iOS. To the user, this is shown as a “Select Images” button. Multiple images can be selected. If a location has been added with images that the user decides they do not want to include, the user can choose to delete any images already added before the final saving of the tour. When the “Save” button is clicked by the user, the images are sent to the back end for processing. When processing is complete, a notification is provided with the status of the creation. The user is redirected if the tour completion is successful.

6.1.5. Tour Page

The Tour Page is implemented with a Flutter class named TourDetailsView. This class takes in the information of a tour and displays the first 360-degree image if it’s the beginning of a tour or the corresponding image when the users select the next transitional hotspot in the tour.

Within each transitional hotspot or image, the users can see indicators to move to other transitional hotspots that are associated with the current hotspot (UC1.3).

The transition between each image is smooth. (UC1.4). The tour navigation involves several transitions between a series of images that are used in the tour. The transition between images avoids abrupt moves and takes place smoothly to maximize user experience during the tour. Transition to any other adjacent image is shutter free and smooth that takes place in no more than one second. Transitions involve a blurring effect to simulate movement. The transition times will remain constant as changes are made to hotspots, view is adjusted, searches are done, or any other unrelated functionality of the application is being used.

On all supported operating systems: iOS, Android and Chrome, the users can zoom and pan on each 360-degree image (UC1.5). This can be done either by zoom scroll or touch input. Users can also manually zoom in/out of the image using the buttons and can pan the image view around to their preference. Image zoom works the same way in the application across all platforms.

They also can select each informational hotspot displayed on the screen by tapping on their mobile device or clicking on their web browser to see the details of each informational hotspot. Some informational hotspots can have external URLs to redirect the users to another web page to learn more about the object that they are interested in.

6.1.6. View Tour

View the images as a sphere from hotspots. (UC1.1) - User can view images by selecting "View Tour", for a specific tour. In the tour view screen, the user finds all the navigation options and they view the images in a sphere format. To create a panoramic tour navigation possible, a series of images that are sequentially stitched together are used to create a sphere. By making use of associated hotspots, users can navigate through images seamlessly.

The start tour icon on the mobile application starts the selected tour from the list. At this stage all the features of tour navigation and customization will engage. These are the zoom and pane, back, forward and reload buttons, and the glow effect slider bar.

This functionality is a screen navigation function where the user can move back and forth to the different displays opened following the directory location. It is displayed and activated depending on the availability of directory to move back and forth. The reload button reloads all the contents of the current display to include updates and enable ease of navigation through the virtual tour.

6.1.7. VR View

VR View is currently only available on Windows and Android machines and is not yet supported on iOS. To access VR the user can select the "VR View" option from the hamburger menu available in the top left corner upon startup. From there the user can click the "toggle VR" option while viewing the tour in the top left corner of their screen to switch among the three available views- normal, google cardboard and VR stereo mode. Google cardboard view separates the user's view into two windows with a border around them to accommodate the

google cardboard viewing device. The VR stereo mode is made for other viewers that lack the padded edges of google cardboard and thus offer the full view of the panorama without obstruction.

This feature went through a number of development approaches prior to its current implementation which employs the panolens.js library to render a javascript based display inside of a web-container. The reason this feature doesn't work on iOS has to do with the loading/updating of the screen employed by iOS web-browsers. Other approaches were attempted, but did not function as expected. The following list of approaches do not work for various reasons:

- implementing the vr directly through the pannalum library used by the view tour (section 6.1.6): does not work due to abandoned development of implementing VR view with this library.
- implementation through flutter react to render VR natively in flutter: doesn't work due to how flutter handles panoramic images which is very different from the way the back-end server provides them. This solution could potentially be made to work but would require a lot of additional development.

Further the issues with iOS could be resolved using a different, more responsive flutter widget to render the page. As of right now the WebViewX library is used as the container to display the html/javascript required to render the view/display it in VR mode. Potentially a different widget could be employed, but as of now no better alternative has been found. The following is the list of approaches that do not work:

-webView widget: does not render the javascript of the page correctly when the widget is built

-webViewPlus widget: takes an exorbitantly long time to load the images and is thus not viable

6.2. View Customization

6.2.1. Search for Available Text

The Search Function (UC 2.2) allows users to search for a string of text and be given a list of matching texts in the current tour.

This feature will be used to search for text on any existing tours. The feature is made up of a search bar used to enter data and a list view page that shows a list of all items returned by the search query or http call. The process of fetching data from the back end is done through a REST HTTP call. Once presented with the results of their query the user can click on any of the outputs to be taken directly to the tour that holds the specified piece of text with the view centered on the desired text hotspot.

6.2.2. Glow Effect

The mobile application glow effect functionality (UC 2.3) uses a slide bar to gage the amount of lighting in the display to enable the user to choose the level of lighting on a picture or tour. The slider bar is placed on the home screen while starting a tour to customize the tour navigation. This feature can be accessed from the Wheel Menu.

As one of the broad requirements of the software under tour customization, the glow effect customizes the tour experience by changing the lighting of the image as it is dragged from left to right. On the contrary, dragging the slider from right to left decreases the lighting of the image.

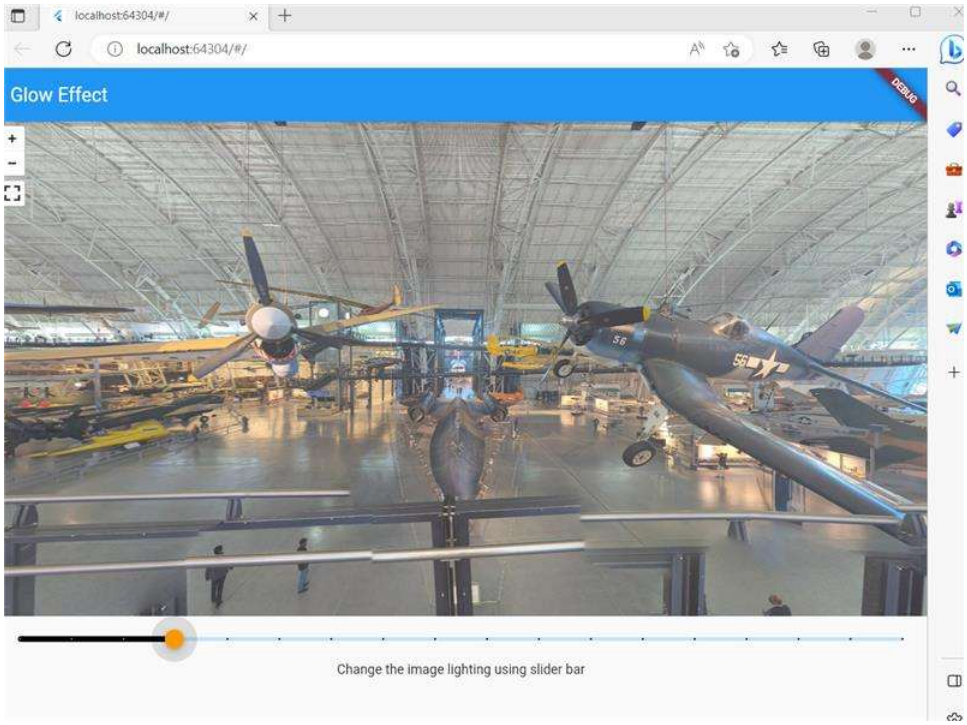

Figure 6.2.2.1 - Image at lower-level lighting on the slider

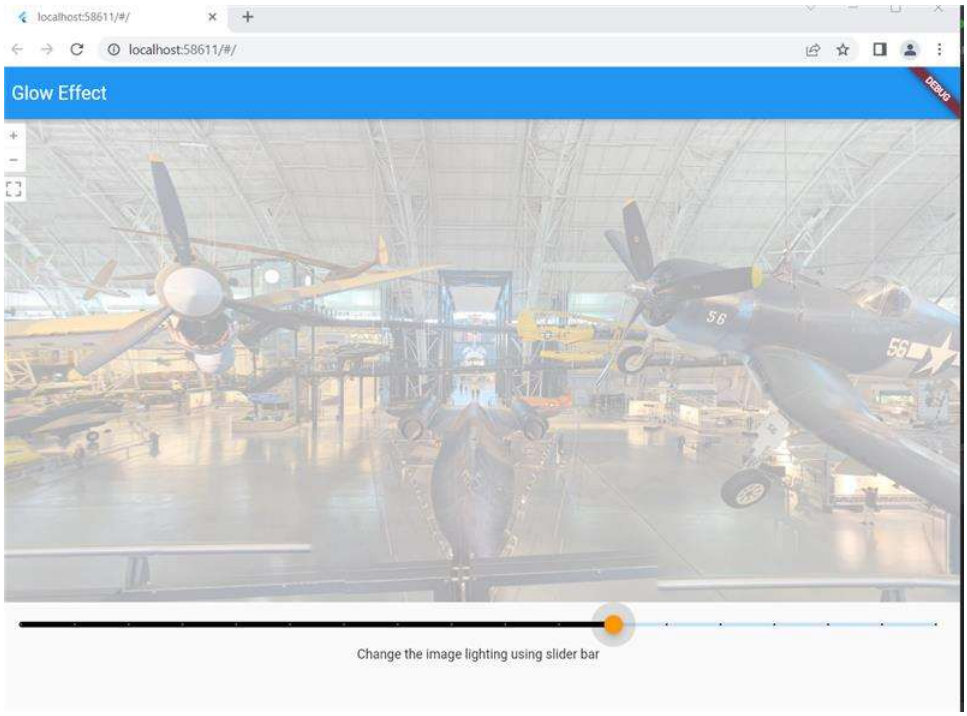

Figure 6.2.2.2 - Image at higher level lighting on the slider

The glow effect Use Case went through different iterations in the development phase to deliver the defined requirement.

1. The initial development was to use ColoredBox with black.withOpacity (0 to 1) to change the lighting of the image place in the Image.assets.

```
SizedBox(
  height: 200,
  child: Stack(
    fit: StackFit.expand,
    children: [
      Image.asset("assets/office.jpg", fit: BoxFit.cover),
      ColoredBox(
        color: Colors.black.withOpacity(lighting),
      ), // ColoredBox
```

Figure 6.2.2.3 - Code moving the Image from original to darker.

This iteration renders output so that as the user moves the slider bar from left to right, the image lighting decreases to darker image. This does not capture the essence of the requirements definition and changes were made to the code.

2. The second iteration developed to make the image brighter from the original by moving the slider from left to right. ColoredBox with white. WithAlpha(0 to 255 with divisions 17) to change the lighting of the image place in the Image.assets.

```
ColoredBox(
    color: Colors.white.withAlpha(lighting.toInt()),
```

Figure 6.2.2.4 - Code moving the Image from original to brighter.

This is recommended by Team B to apply the glow filter in the back end during the image upload process. Team B is currently synchronously developing the back end of ViroTour as a cohesive unit with Team A. Team B will set up a REST API endpoint to apply the 'glow' effect. As soon as the front end can load tours, the parameters that the endpoint would expect are Tour Name, Location ID, and the value for lighting.

3. The glow effect is located under the wheel menu. Once it is clicked, it needs a back button to go back to the tour view. This functionality is added to the glow effect by the wheel menu developer. Clicking on the back button triggers a request to the API to apply/save the glow filter on the image in the back end.
4. In this iteration, the glow effect is customized to fetch image data from the tour view instead of the asset/image directory. The panoramic image displayed by the tour view shall be used for the application of the glow effect. WebviewX is used to refer to the url/location of the image.

```
WebViewX(
  height: 520,
  width: 1600,
  // TODO: replace URL with response from API call GET /tour/<tour_id>/
  initialContent:
  'https://cdn.pannellum.org/2.5/pannellum.htm#panorama=https%3A//i.imgur.com/09CBhdM.jpg&autoLoad=true',
  onPageStarted: (url) {
    // This method is called when the WebView starts loading a new page
    debugPrint('Page started loading: $url');
  },
  onWebViewCreated: (controller) {
    var webviewController = controller;
  },
```

Figure 6.2.2.5 - Code fetching Image from url

6.3. Hotspot Customization

6.3.1. Transitional Hotspot Customization

The search function will list all available tours that contain a given search term in a grid view. A user can enter a search term in the search box and click the find tours tab to go to the tour of choice and start the virtual tour.

Any user can add/edit/delete transitional hotspots associated with an existing tour (UC3.1). These hotspots will be in the place of pre-stitched panoramic images that can be uploaded to an existing tour. These hotspots can be created, existing hotspots edited, and hotspots can be deleted completely. The hotspots once created by the user who created the tour will be visible to all users who access the associated tour. These hotspots will be able to have transition points added directing the user to a new image. That transition point will move to a new image.

7. Project File Structure

7.1. Physical Structure

The project file structure's goal is to organize a projects data and code to make it repeatable and conscience, aiding the development and collaboration of the project as well as future revisions and added features. The file structure will develop over time with the needs of ViroTour. However, this section will separate concerns and establish a consistent, chronological, and descriptive naming scheme.

- Folders and directories are used interchangeably. Folders will be the favored usage.
- File structure, file hierarchy, and file schemas are used interchangeably. File structure will be the favored usage.

7.2. Logical Structure

7.2.1. ViroTour Folder Structure

Android – this folder contains all files necessary to build an application targeted for the Android OS platform.

IOS – this folder contains files that are needed to build the application for an IOS based platform.

Web – the folder contains files that are needed to build the application for web-based applications like web browsers.

MacOS - this folder contains files that are used in building the application for MacOS based operating systems.

Lib - all dart bases user generated source files are stored in this folder. It contains the main.dart file which is the application's execution entry point.

.**idea** - this folder contains configuration files that are related to the Android Studio development environment.

Build - this folder stores all generated build files.

.**dart_tool** - the folder is used to cache files that are specific to the project or development environment.

Assets - the folder is used to store static files like images that are used within the application.

README.md - the file contains important information about the project and is created when placing the project files in a repository like Github.

pubsec.yaml - the file contains all project dependencies.

analysis_options.yaml - the file contains tools that are used to enforce good coding practices for applications, packages, and plugins.

7.2.2. File and Directory Naming

The file and directory naming scheme will use the Flutter naming scheme where all characters are to be in lowercase and words are joined using an underscore.

Example: tour_search.dart

8. User Interface (UI)

8.1. Front End Screens

The user interface of ViroTour application has been developed using Flutter widgets which offer flexibility and speed to the development process. The UI elements in the application facilitate accessing the application easily so that users can accomplish their goal in the application. To help users navigate in the app easily, there are two menu options placed on the appbar at the top right and left corners. Users may also see a back arrow that helps navigate back to the previous screen. The UI screens are presented in the subsections below starting from the app launch icon on users' device.

8.1.1. App Launch Icon

The app launch icon is used to start the ViroTour app from the app launch menu of a mobile device. The app launch icon displays the circular ViroTour logo, and it displays the same way on various models of Android and iOS devices.

Figure 8.1.1 App Launch Icon (Android)

8.1.2. Tour List

When the ViroTour application is launched, the tours that are available to choose from are displayed for the users in the “Tour List” page. Users can select a tour from the list and start navigating. A tour can also be edited by selecting the edit button on the right side of a tour’s row in this view. The screens below depict ViroTour application displayed over multiple platforms, Android, iOS, and Chrome.

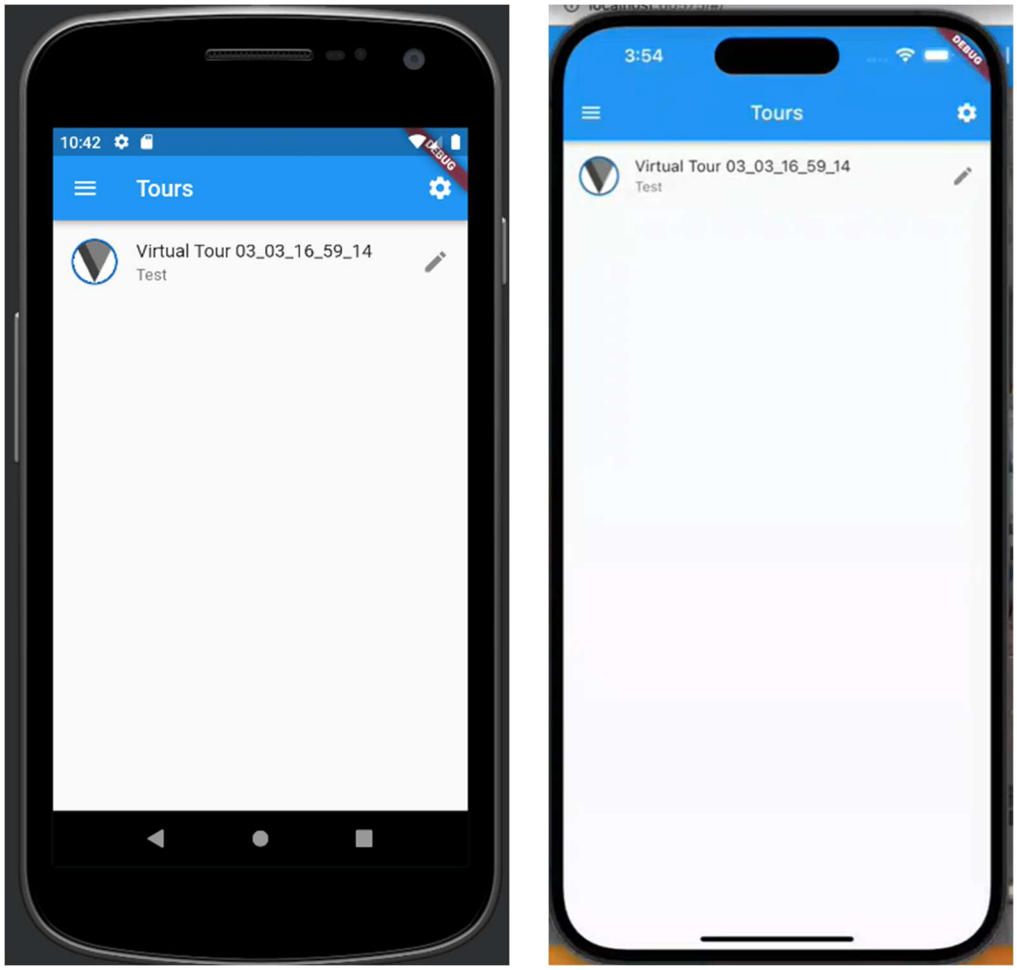

Figure 8.1.2a Tour List Android (left) and iOS (right)

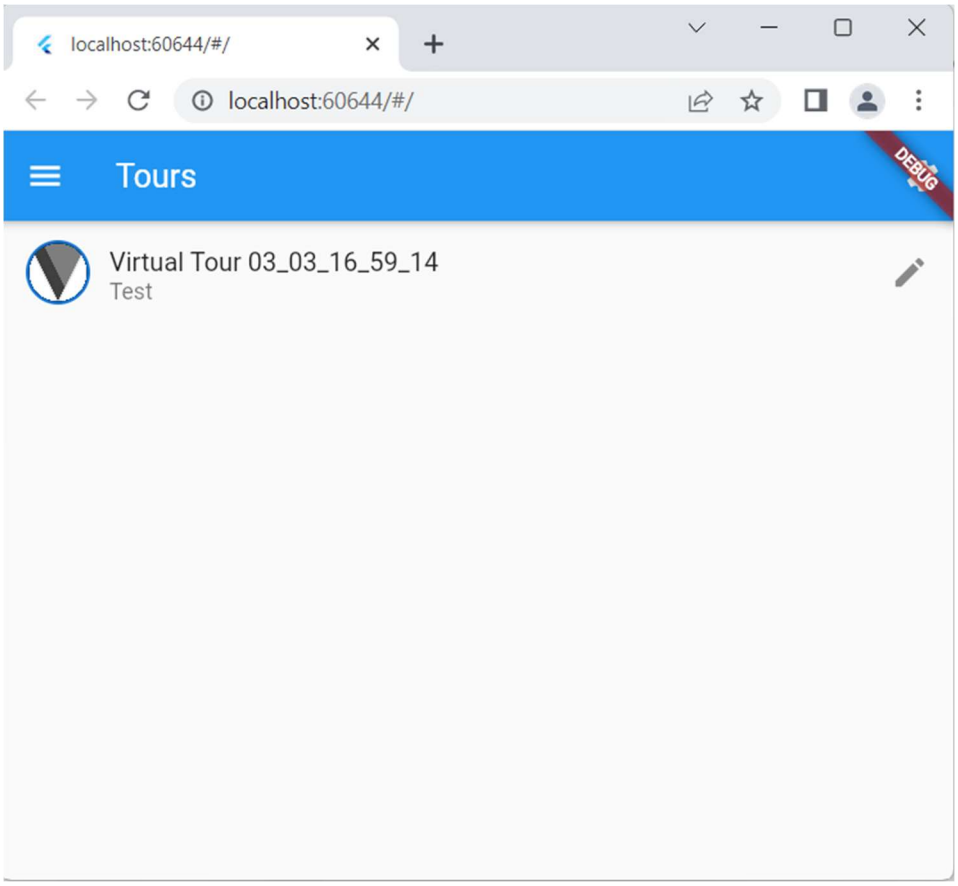

Figure 8.1.2b Tour List (Chrome)

8.1.3. Edit Tour

By selecting the "Edit" button on the right side of a tour from the Tour List view, users can get to the Edit Tour screen. On the "Edit Tour" screen, users can edit the name and description of the tour and they can save the changes, cancel, or delete by making use of the buttons as needed. If a user clicks on the "Delete" button, there will be a popup modal asking the user to confirm, as an added layer of security to help prevent accidental deletion.

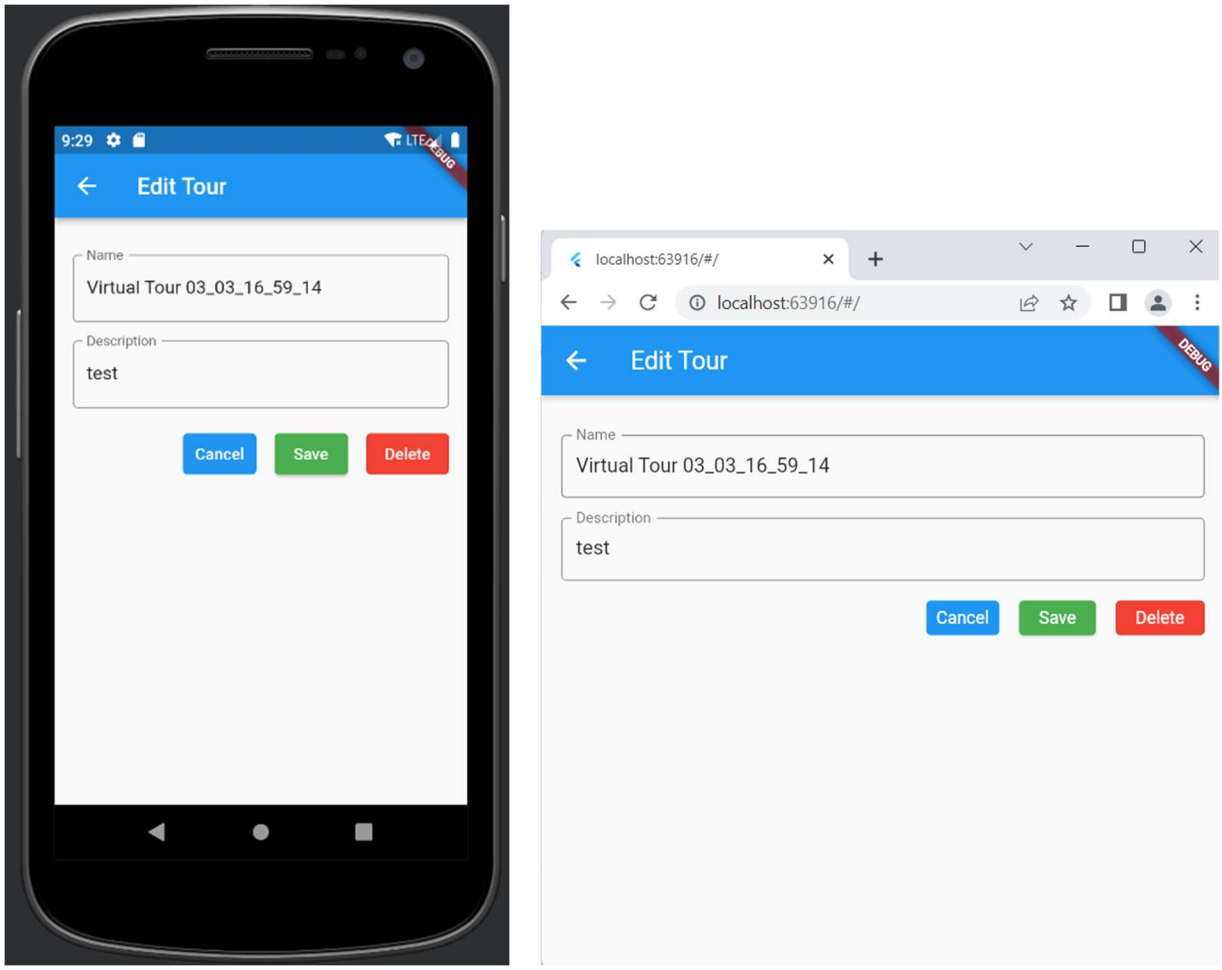

Figure 8.1.3 Edit Tour, Android (left), Chrome (right)

8.1.4. Tour Page

This page will be displayed after the user selects a tour to view. This screen displays 360 images and allows users to move to other images through transitioning between hotspots. In addition, it also allows users to zoom in and zoom out while navigating the tour. Users can also select to navigate the tour in a full screen mode by tapping or clicking on the full screen mode button on the left side of the screen.

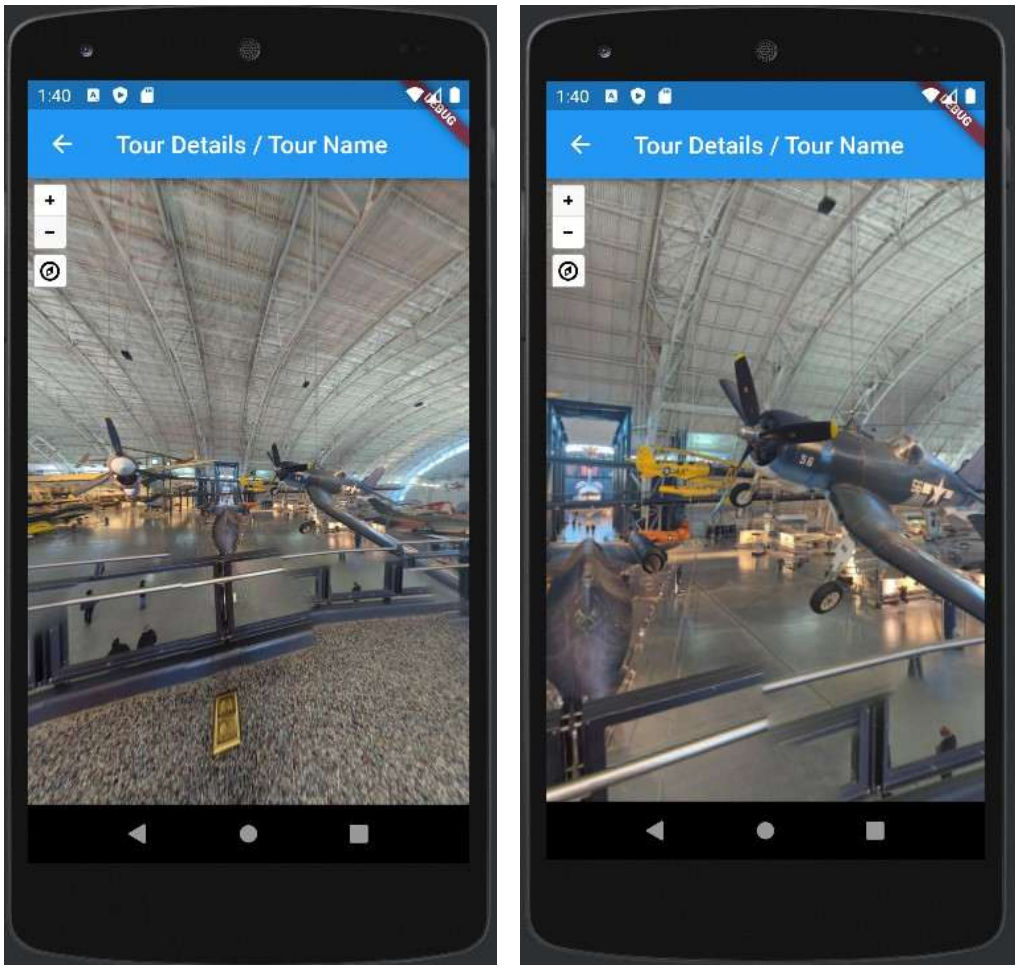

Figure 8.1.4a Tour Page Regular view (left) and zoomed in view (right)

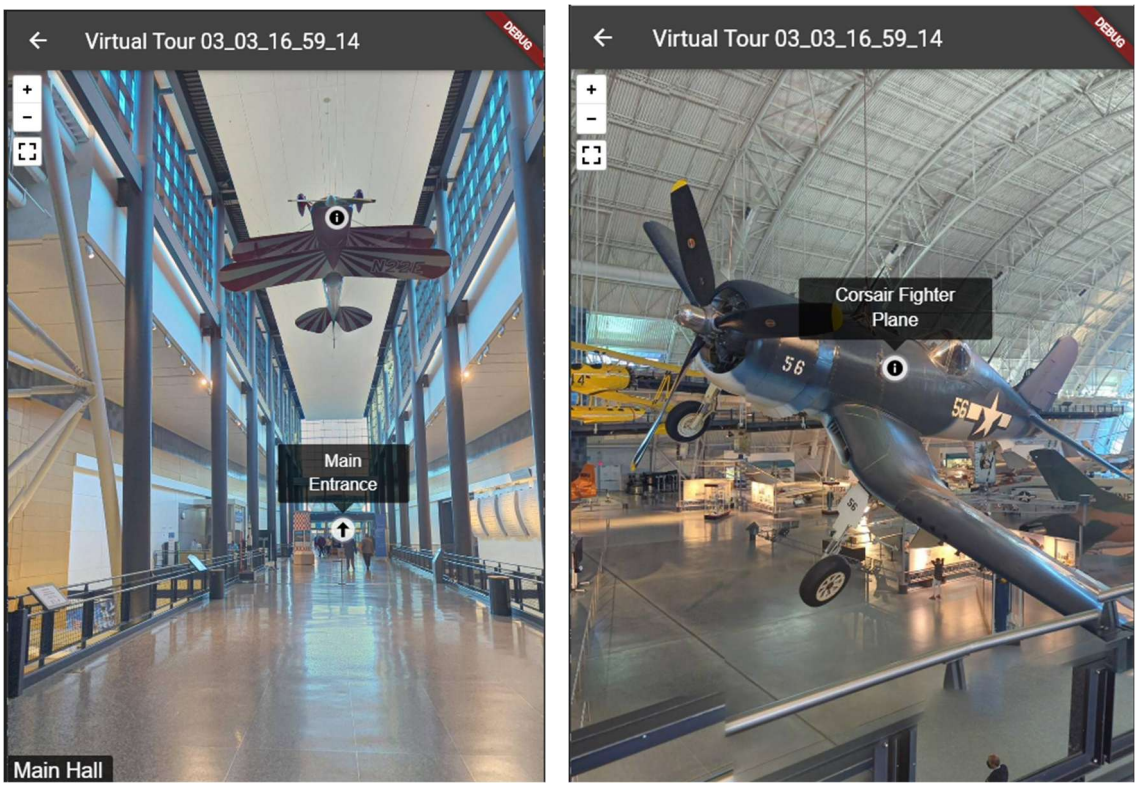

Figure 8.1.4b Tour Page – Navigation Hotspot(left) and Information Hotspot (right)

8.1.5. Hamburger Menu

The Hamburger menu is displayed on the top left corner of the application screens to easily switch between activities. From the Hamburger menu, users can select “View Tours”, “Create Tour”, “Search Tours”, and “Search Hotspots”. If the user selects the “View Tours” option, they will be taken to the Tour List view. If the user selects the “Create Tour” option, they will be taken to the Create Tour view. If the user selects the “Search Tours” option, they will be taken to the Tour Search View, which allows users to search for a specific tour name or description. Finally, if the user selects the “Search Hotspots” option, they will be taken to the Hotspot Search View, which allows them to search for specific hotspots that are inside a tour.

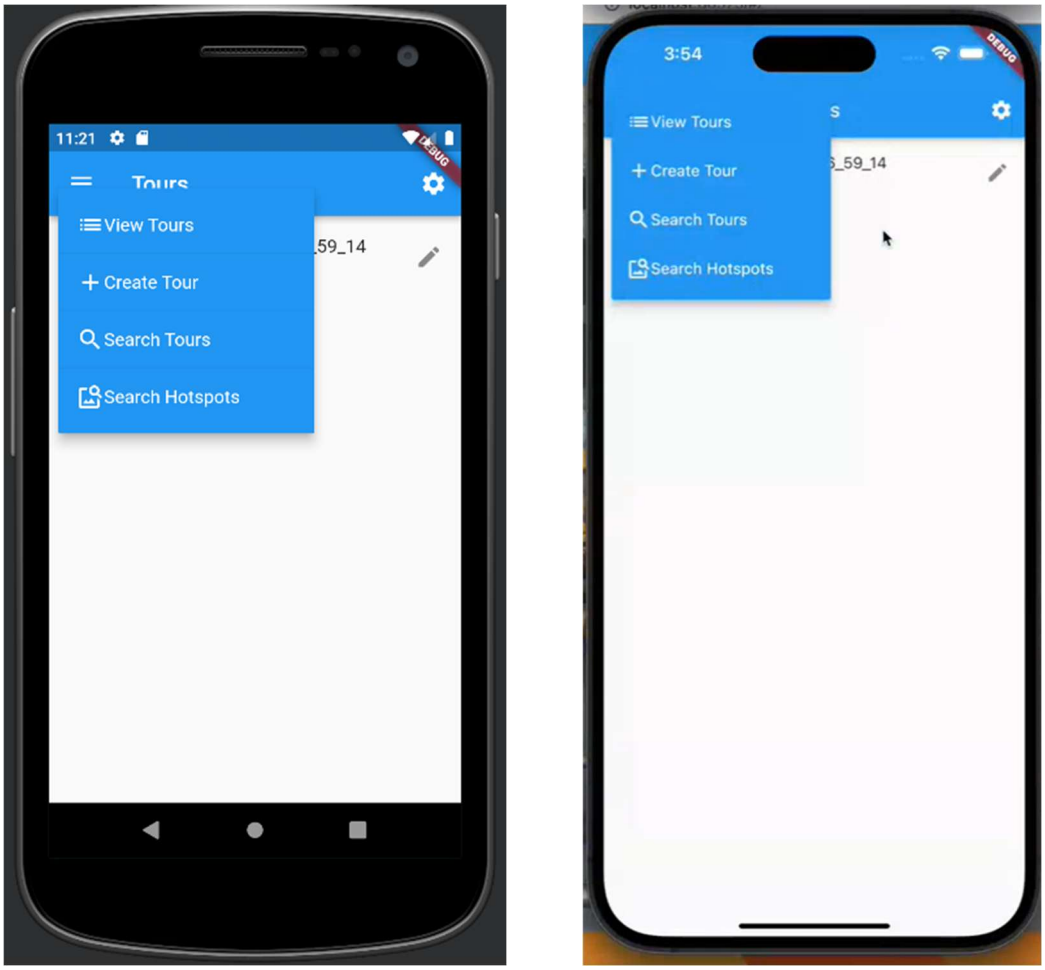

Figure 3.1.5 Hamburger Menu, Android (left), iOS (right)

8.1.6. Wheel Menu

The Wheel menu is displayed at the top right corner of the application. This menu displays the Settings screen when selected. The settings screen contains three options Theme Selection Dropdown, “Glow Effect”, and “Share”. The system theme selection dropdown provides options to select and apply either a system theme, light them, or dark theme. The system theme will use the operating system’s theme of choice. The glow effect button displays the glow effect adjustment bar and allows user to apply glow filter to selected pictures.

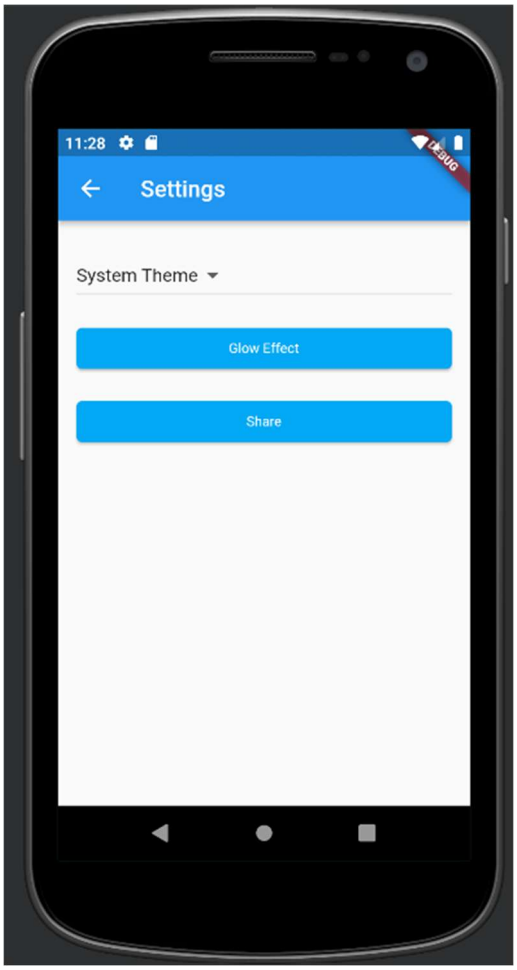

Figure 8.1.6 Wheel Menu

8.1.7. Create Tour

When user selects “Create Tour” from the Hamburger menu, to add a new tour, the form to add tour information will be displayed in the screen requesting user to add “Tour Name”, “Description” and select images. These text input and image selection are required fields for the tour to be created. The “Select Image” button allows user to select and upload images that will be processed for the new tour. When the user is done adding the data, they will click the “Save” button to submit the information and create the tour. The user can also select the “Cancel” button to discard tour creation and return to the Tour List view.

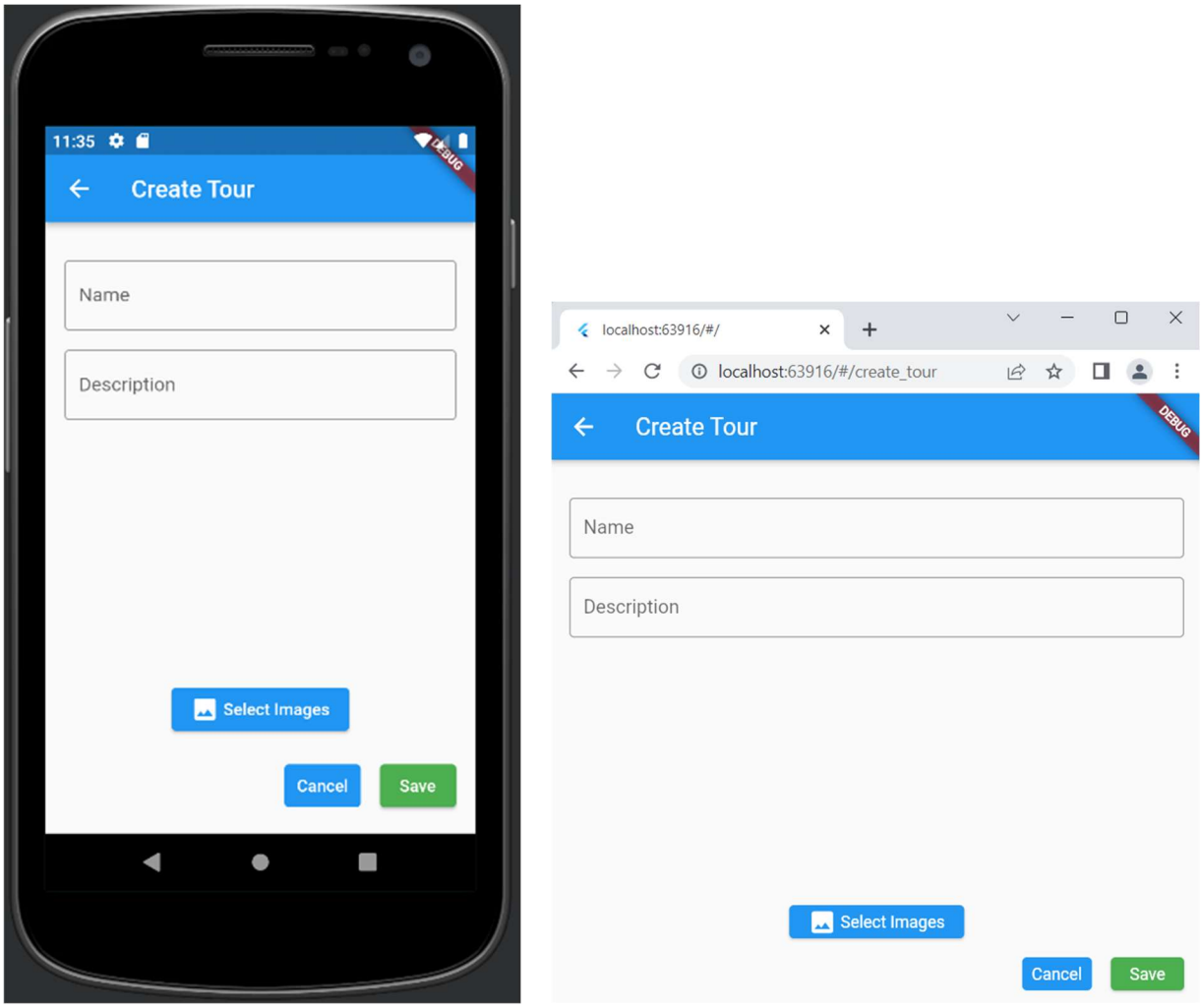

Figure 8.1.7 Create Tour Screen Android (left), Chrome (right)

8.1.8. Search

From the Hamburger menu, users can access two types of searches “Search Tours” and “Search Hotspots”

8.1.8.1. Search Tours

By selecting “Search Tours”, users can enter the search key term related to the tour they are searching for, and press enter to get the results. The search returns the list of tours related to the search key word(s) entered. The search text entered will be checked against all tour names and descriptions. Any tour that matches the search text will be displayed in a list below the text input box. Users can select the tour from the list to view tour details.

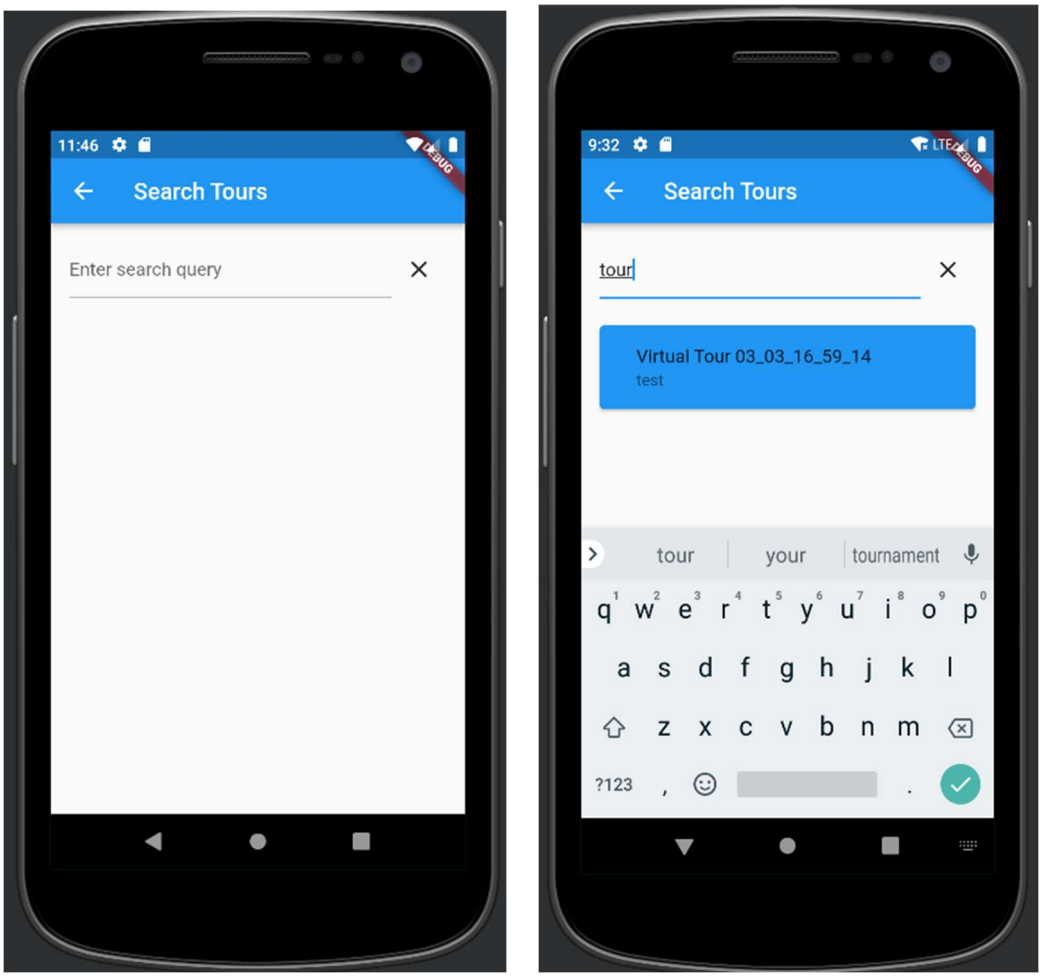

Figure 8.1.8.1 Search Tours input (left), Search Tours result(right)

8.1.8.2. Search Hotspots

By selecting “Search Hotspots”, users can search for Hotspots by entering search term(s). Once users submit the search request, the search hotspot result matching the request shall be returned in a list. Users can navigate to the hotspot by selecting the retuned search result.

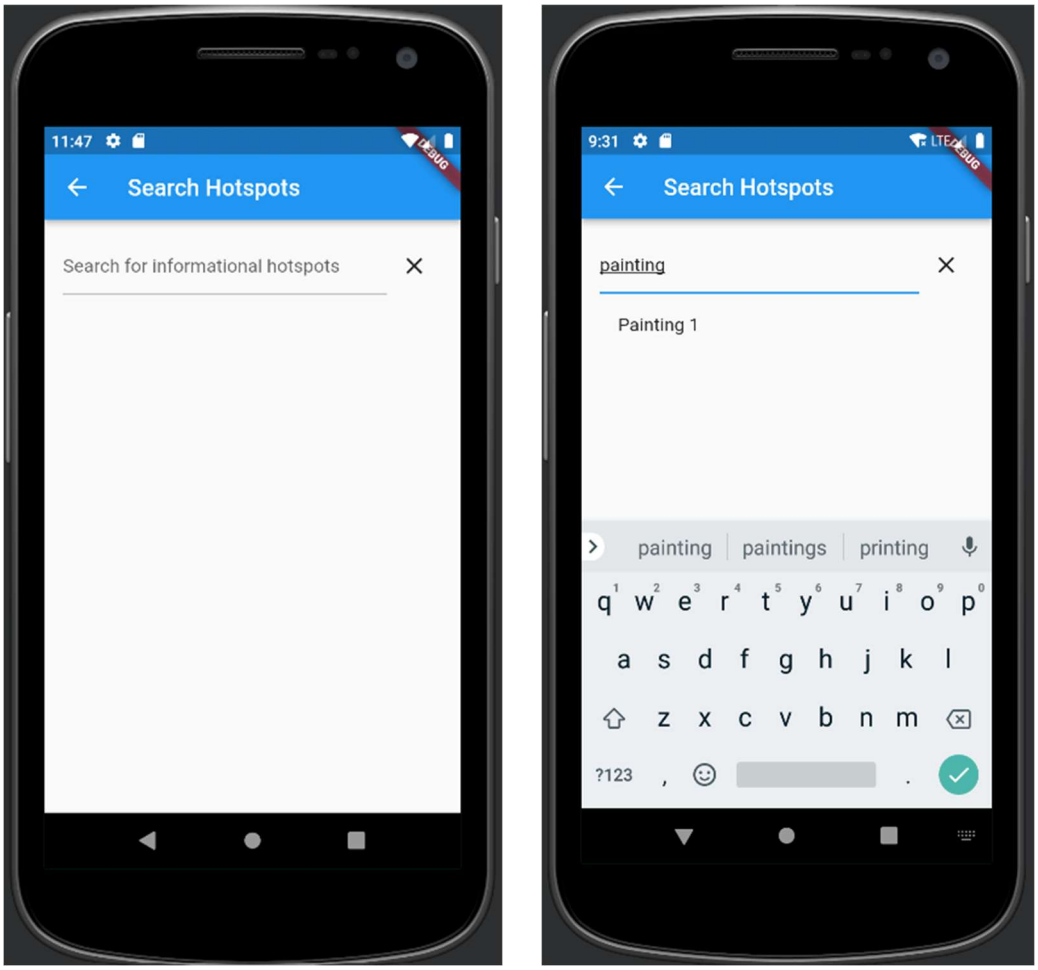

Figure 8.1.8.2 Search Hotspot input (left), Search Hotspot result(right)

8.1.9. Glow Effect

The feature is used to adjust lighting of a selected picture from a tour. By selecting “Glow Effect” from the Wheel Menu, users can access the Glow Effect slider bar to adjust the lighting as needed. The lighting increases when the button slides from left to right and decreases as it moves from right to left.

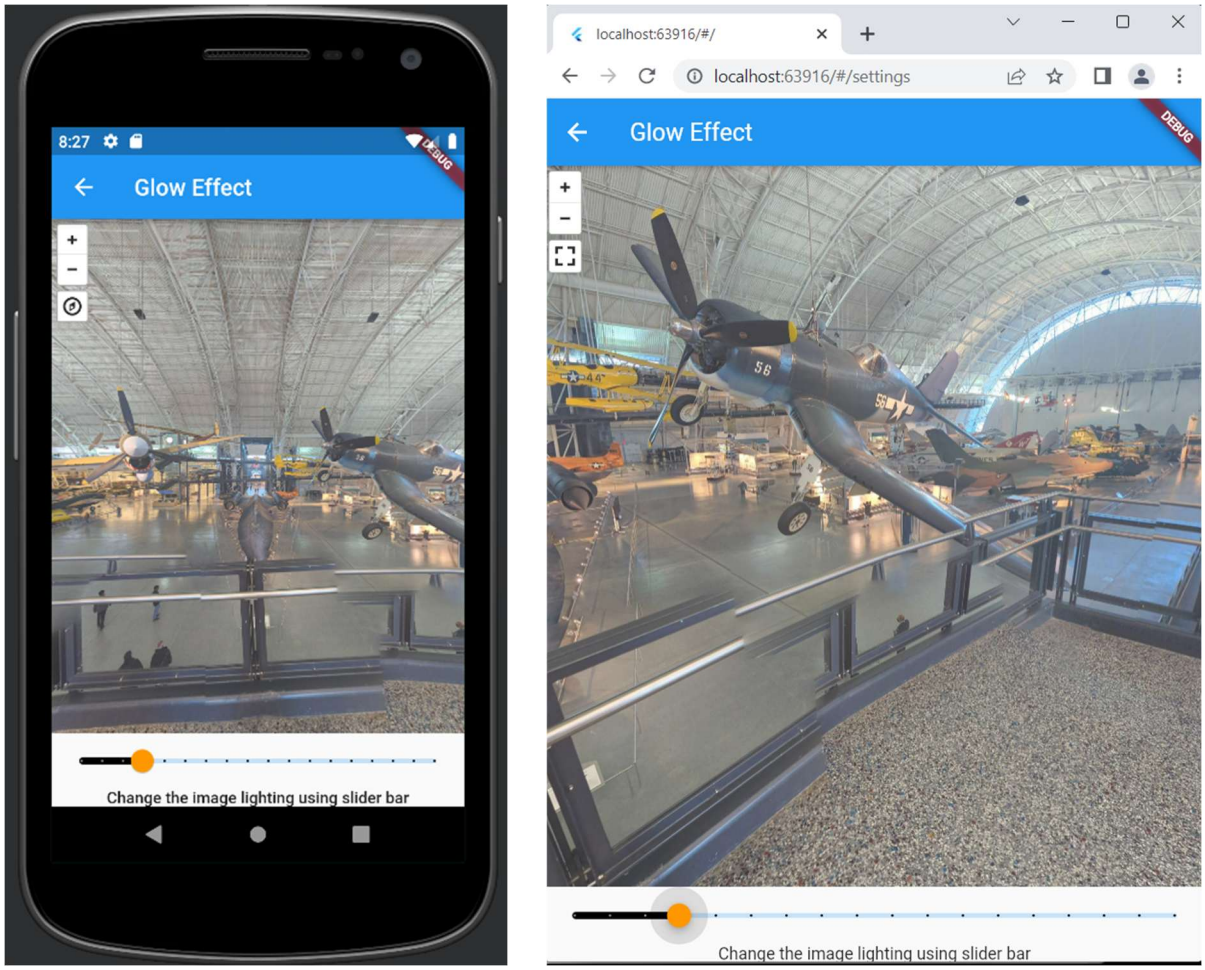

Figure 8.1.9 Glow Effect Android (left), Chrome (right)

8.1.10. VR View

The VR View is selected by clicking on the 'Toggle VR' button from the tour detail screen. By enabling VR View, users can experience viewing the tour using a VR viewer. The VR View also has "settings" and "full screen" control options at the bottom right-side of the screen. From the settings menu, users can select Mode (Normal, Cardboard, Stereoscopic) and Control options (Mouse, Sensor).

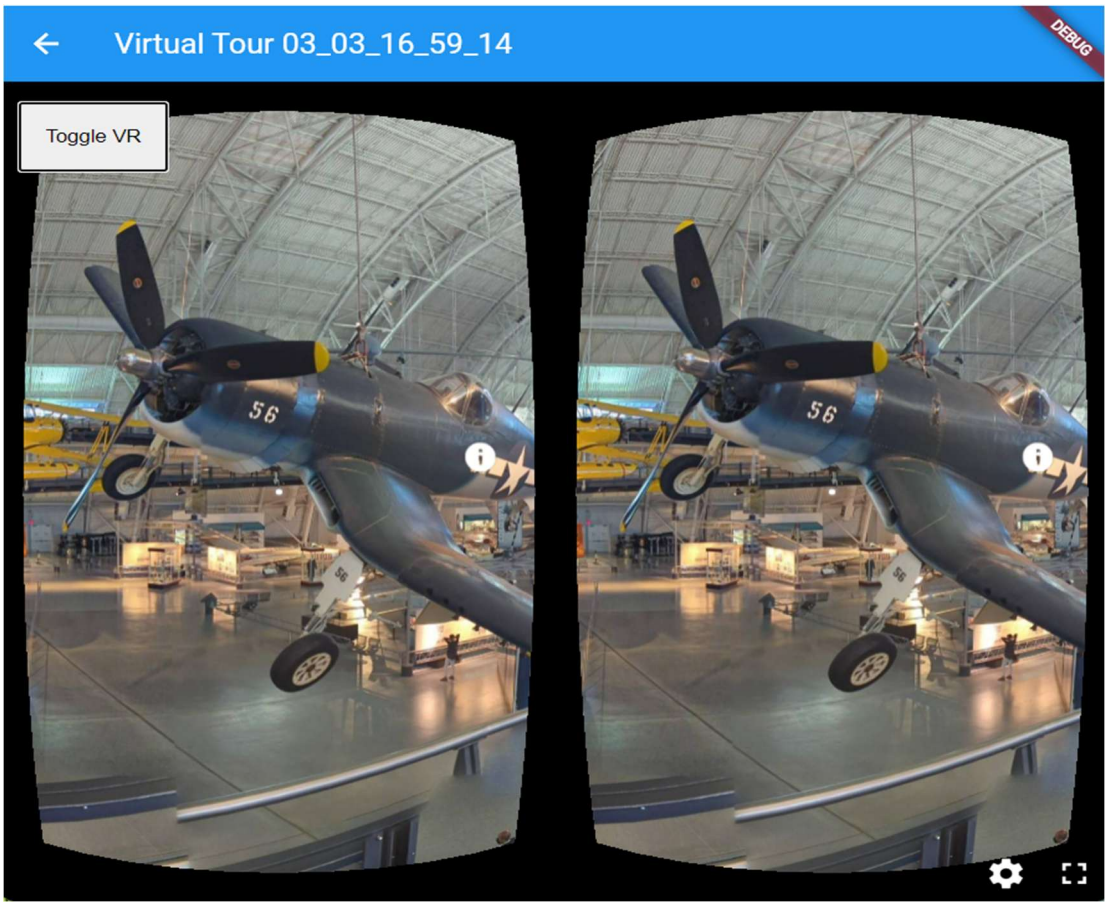

Figure 8.1.10a Google Cardboard VR View

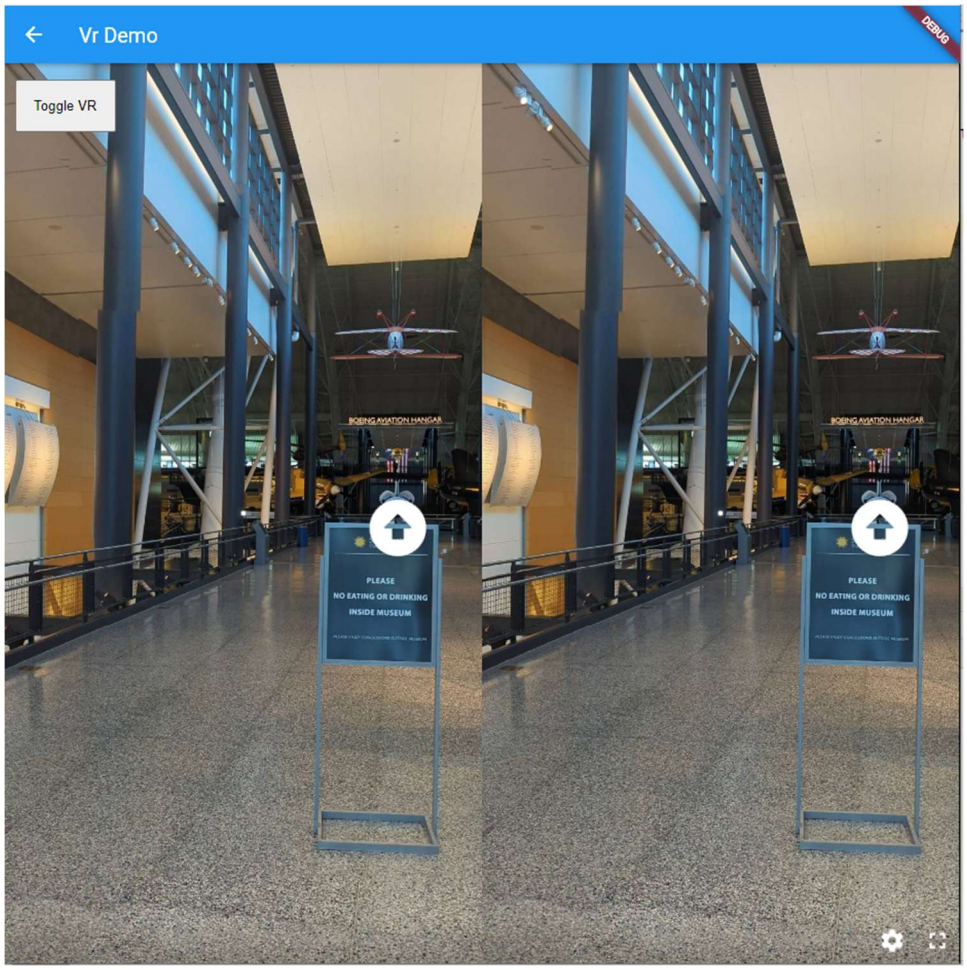

Figure 6.1.7.2 - VR Stereo View

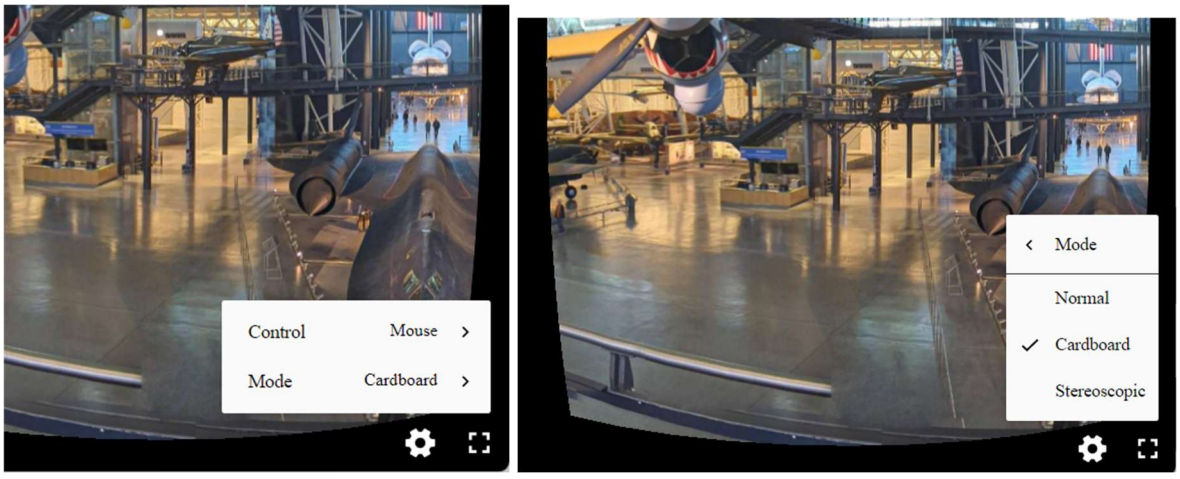

Figure 8.1.10c VR View Controls

8.1.11. Theme Selection Dropdown

The Theme selection dropdown is located on the Settings screen in the wheel menu. The Theme Selection dropdown presents three theme options to choose from, System Theme, Light Theme and Dark Theme. While System theme matches the operating system's theme, Dark Theme converts the theme to dark mode and Light Theme converts the theme to light mode.

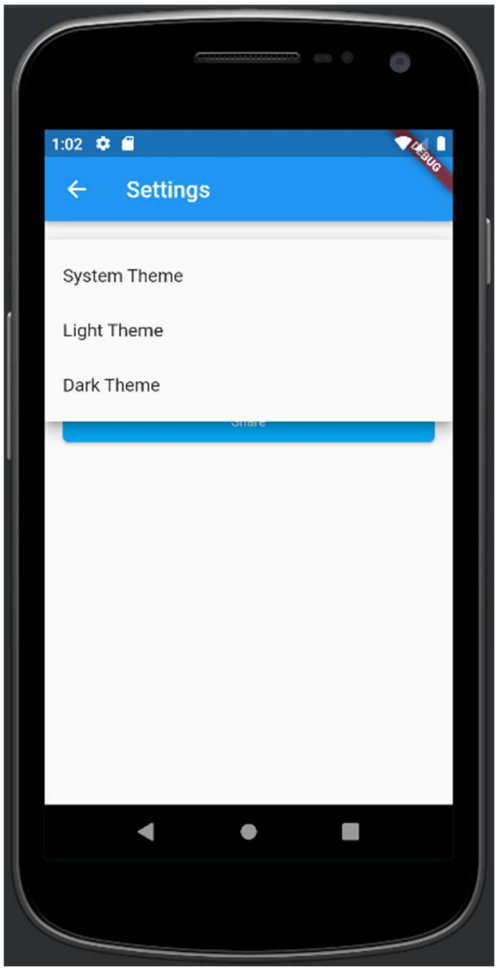

Figure 8.1.13a Theme Selection dropdown

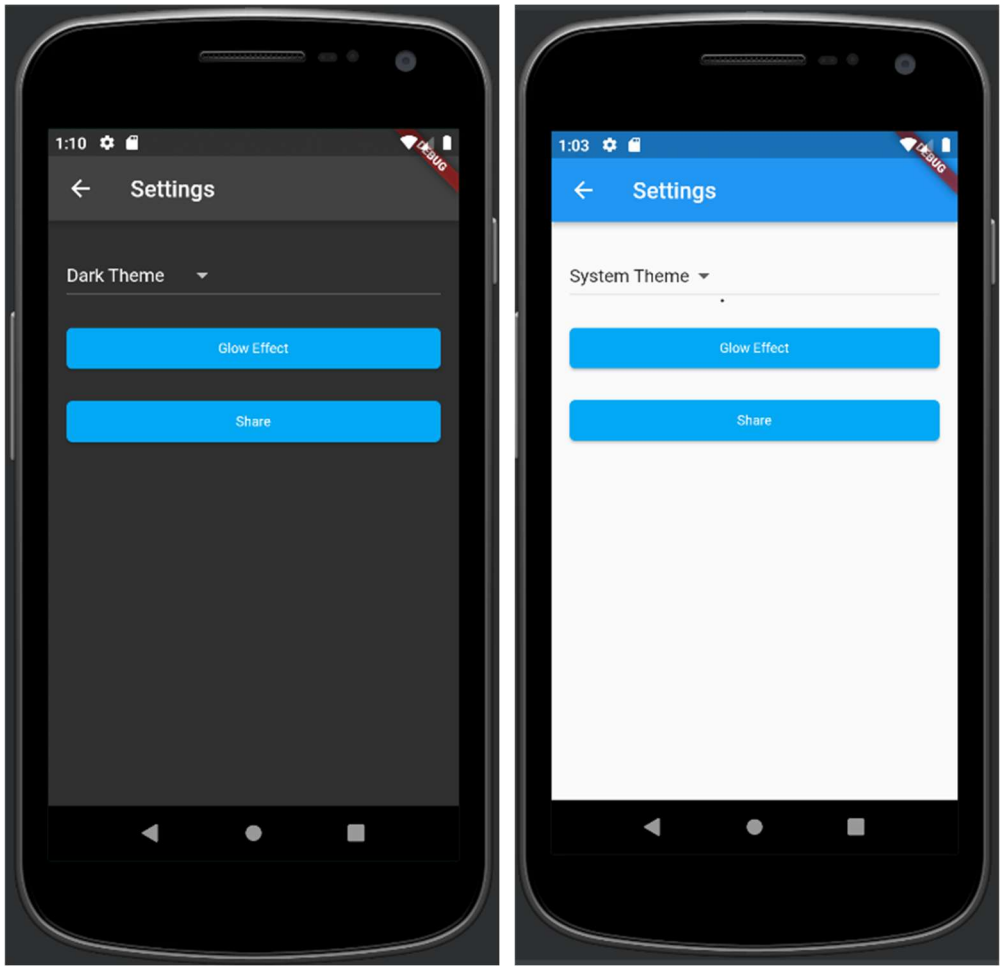

Figure 8.1.13b Dark mode (left) and Light Mode (right)

9. Accessibility

This section explains the steps ViroTour takes to meet accessibility and how to implement them. Accessibility ensures the app has support for the broadest range of users. Failure to meet the specifications or poor design would create barriers to people of all ages and disabilities. The UN Convention on the Rights of Persons with Disabilities states the moral and legal imperative to ensure universal access to information systems; countries around the world enforce accessibility as a requirement; and companies recognize the business advantages of maximizing access to their services. According to web content accessibility guidelines (WCAG) Principles - at the top are four principles that provide the foundation for Web accessibility: perceivable, operable, understandable, and robust.

The following will serve as a checklist to ensure ViroTour meets minimum accessibility requirements:

- Large fonts -Render text widgets with user-specified font sizes

- Screen readers - Communicate spoken feedback about UI contents. (Compatibility with desktop and mobile screen reading).
- Sufficient contrast - Render widgets with colors that have sufficient contrast.
- Alternative text for images – meant to convey the "why" of the image as it relates to the content of a document or webpage. It is read aloud to users by screen reader software, and it is indexed by search engines. It also displays on the page if the image fails to load, as in this example of a missing image.

10. Data and Back End

The main responsibility of this development team does not cover the Data Tier of the application. Therefore, the database selection and back-end architecture are handled by the other ViroTour team.

At a high level, the application uses two types of data storage: a blob storage that stores the uploaded images and the processed images and a NoSQL database that stores user and tour metadata.

The Presentation Tier will relate to the blob storage to handle the creation of the tours or hotspot customization. That allows users to upload their images to the application's storage. The Presentation Tier will take users' input and transform them into API requests. It then will send the requests to the back end. Once the back end responds, the Presentation Tier will parse the returned JSON objects and display them to the users.

11. Publishing

11.1. Google Play Store

As a follow-on feature The ViroTour application can be added to Google Play store as the design for the application is intended for mobile usage.

To add a Flutter app to the Google Play Store, you need to follow these steps:

1. Create a Google Play Console account: If you don't already have one, create a Google Play Console account by going to https://play.google.com/console/signup and following the steps.

2. Generate a signing key: You need to generate a signing key to sign your app. Open a command prompt or terminal window and run the following command:

   keytool -genkey -v -keystore <keystore_name>.keystore -alias <alias_name> -keyalg RSA -keysize 2048 -validity 10000

   Replace <keystore_name> with the name you want to give your keystore and <alias_name> with the name you want to give your key. Follow the prompts to enter the

other required information.

3. Configure your app: Open your Flutter app in Android Studio or Visual Studio Code and go to the android/app/build.gradle file. Update the minSdkVersion and targetSdkVersion to the values required by the Google Play Store.

4. Build your app: In Android Studio or Visual Studio Code, go to the Build menu and select Flutter Build APK. This will generate a signed APK for your app.

5. Upload your app: In the Google Play Console, create a new app and fill in the required information. Then, go to the Release section and upload your signed APK.

6. Publish your app: Once you've uploaded your APK, you can review the release details and publish your app on the Google Play Store.

7. Note: Before uploading your app to the Google Play Store, make sure to test it thoroughly and ensure that it meets all the requirements specified by the Google Play Store.

11.2. Apple App Store

As a follow-on feature The ViroTour application can be added to Apple App store as the design for the application is intended for mobile usage.

1. Register as an Apple Developer: First, you need to register as an Apple developer. This requires an Apple ID and a one-time fee of $99.

2. Create an App ID: Once you are registered as an Apple Developer, you need to create an App ID. The App ID is a unique identifier that is used to identify your app in the Apple ecosystem.

3. Generate a Distribution Certificate: In order to publish your app on the App Store, you need to generate a distribution certificate. This certificate is used to sign your app and verify that it was created by you.

4. Create a Provisioning Profile: A provisioning profile is used to link your app with your distribution certificate. You need to create a provisioning profile for each app that you want to publish on the App Store.

5. Prepare Your App for Submission: Before submitting your app, you need to make sure that it meets all the App Store guidelines and requirements. This includes adding App Store icons, screenshots, descriptions, and metadata.

6. Upload Your App to App Store Connect: App Store Connect is Apple's web-based platform for managing and submitting apps to the App Store. You need to upload your app to App Store Connect and provide all the necessary information, such as pricing, categories, and keywords.

7. Submit Your App for Review: Once you have completed all the necessary steps, you can submit your app for review. Apple will review your app to make sure that it meets all the guidelines and requirements. If your app is approved, it will be available on the App Store for users to download.