Travel Advisor Final Report

**Prepared by Ali Dawarsha, Meet Patel, Vedant Majmudar, Joshua Gonzales for use in CS 440
at the
University of Illinois Chicago**

April 2021

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

Travel Advisor is a mobile application developed via the Flutter framework that is designed to assist users in navigating to and within national parks in the United States.

2 Project Domain

The domain for this application are all the national parks located within the United States of America.

3 Relationship to Other Documents

The Project Design Document and Project Requirements Document are closely related to this document, as they were used as guidelines for developing the features that are being tested.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Scenic Spots: Recommendations of places users should visit throughout various national parks.

Weather Advisor: Weather information that is pulled via OpenWeather API and displayed to users, the information pertains to the current scenic spots that are related to the currently selected national park.

MarkerInformation: Class that holds information for the custom markers for Scenic Spots.

MarkerInformation Window: Custom window that displays Scenic Spot information upon clicking a marker within the in-app map.

Home Page: Page of application that holds a list of national parks, which upon clicking will present the user with a multitude of options to choose from.

Scenic Spot Page: Page of application that holds a list of Scenic Spots that pertain to the specific national park that was chosen within the home page.

Setting Page: Page of application that holds the emergency contact button.

Emergency Contact Button: Button that will send out emergency contact information from the application to a database, this contact information will be prefilled by the user and the contact information will also include the current location of the user upon pressing the button.

Firebase: Platform developed by Google for mobile development from which Travel Advisor uses the database and authentication services. The login information is important to store data such as favorites and emergency information. The database is helpful to store different data for each user and also to store National Park information.

OpenWeather: API which provides a way to get the weather information for the Scenic Spots page of the application. Travel Advisor uses the by location function.

4b UML and Other Notation Used in This Document

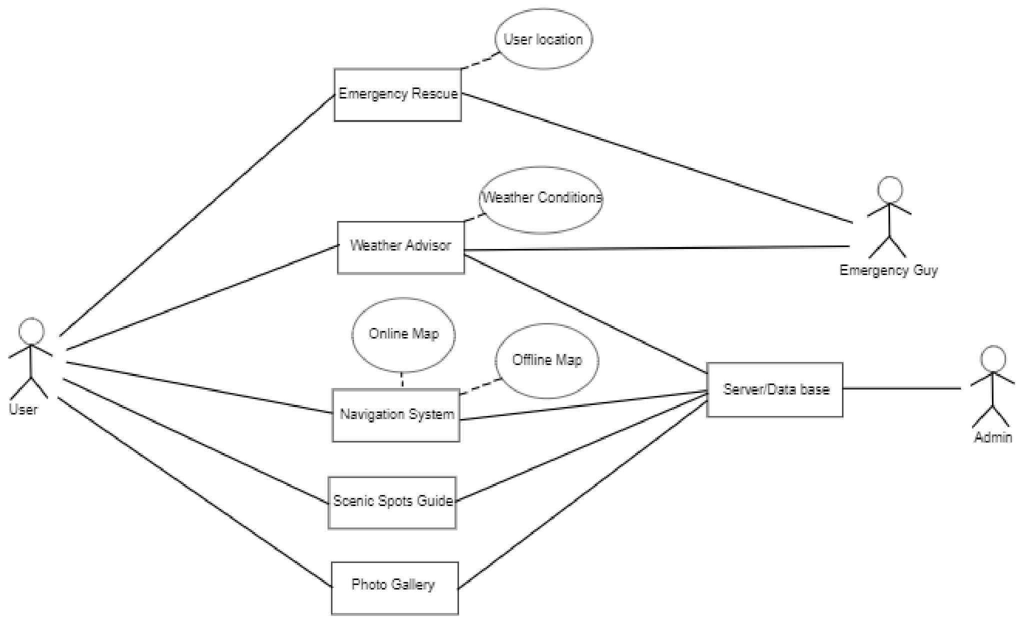


Figure F1

The use case diagram above was used as the foundation for the application's development. Most of the rectangles in the middle of the diagram were turned into pages in the navigation bar. The server/data is proxied through Firebase, while each circle represents different functionalities and methods in the application.

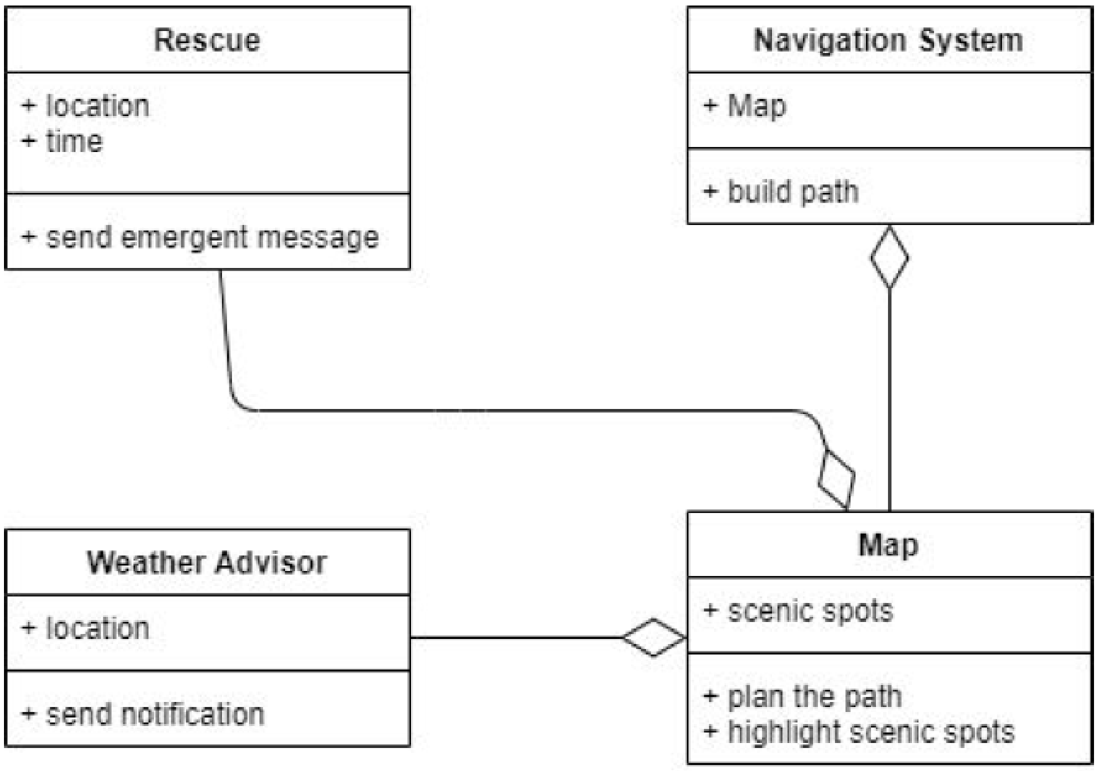


Figure F2

This UML diagram was provided in the original Travel Advisor document [2]. It helped initial mappings of the use case diagram seen before this diagram. The functions were implemented in the way it was described here, but adapted into Flutter's Widget and Builder syntax.

4c Data Dictionary for Any Included Models

In terms as data structures utilized for data, many utilized simple arrays with custom classes in order to achieve desired functionality. In regards to navigation, a class was created known as MarkerInformation that was designed to hold pertinent information for the Scenic Spots located in various National Parks. The MarkerInformation holds the name, image, description, and location within the class. All of which are of type String with no limit imposed, except for the location which is of type LatLng in order to hold the Longitude and Latitude, Latitude having a range of [-90,90] and Longitude having a range of [-180,180).

II Project Deliverables

Our Approach was simple and based on the documentation. We started with going through the documentation and collecting all the features to be implemented. Taking a hard look at this feature we decided Flutter as our platform, so we could reach almost all the users in the current market. While implementing the features we took our owne approach with the implementation process and did put it in the application as we see fit.

1 First Release

We started our project by, deciding to use flutter as our platform to begin with. It was a steep learning curve for the first week, which included setting up a common github page and a common tool like Visual Studio Code and Application supporting tools like android emulator within VS Code. Using the documentation, Meet created five basic pages where we can implement different fractures namely Home Page, Map Page, Favourites Page, Scenic Spot Page. After the ground setup and everyone getting familiar with flutter, Josh started working on Firebase and authentication features for user login and account creation. Later on I added several national parks on the Main page and Meet added favorites features. During this time Ali worked on adding Google Map API in the map page, this also included markers for the scenic spots marker for that National Park.

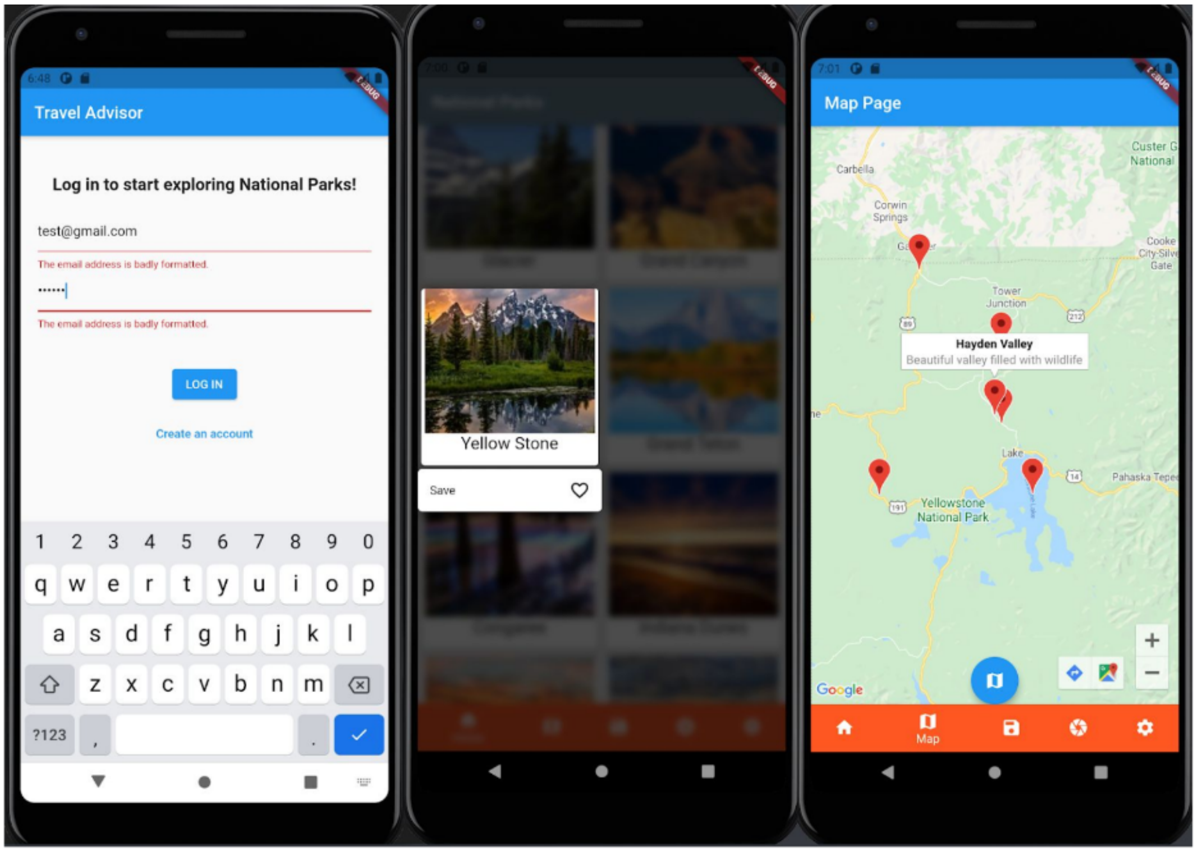
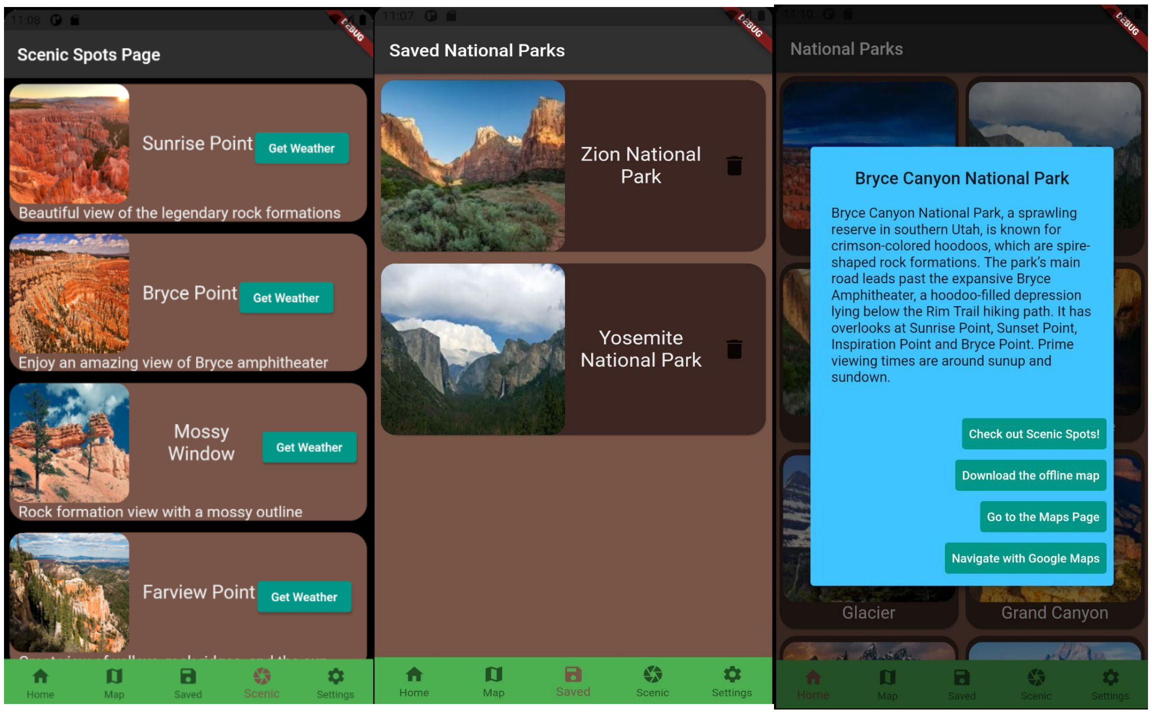

Travel Advisor
Log in to start exploring National Parks!
test@gmail.com
The email address is badly formatted.
The email address is badly formatted.
LOG IN
Create an account
Yellow Stone
Save
Map Page
Hayden Valley
Beautiful valley filled with wildlife
Yellowstone National Park
Map

Figure F3

2 Second Release

The second release was started with new functionalities, such as Favourites feature, Scenics spots feature, download offline feature, Google Markers, and Emergency feature. The features were created by using Google online Firebase database. Data was sent to the database, some data was retrieved from the database. With the help of the popup menu, a national park has 4 buttons, each button has their own functionality listed above. Click on Scenic spots and it will bring to all the scenic spots of the national park selected. Click on Download map and it will open a pdf of the particular national park and you can either download it or print it before visiting national parks and so on. Lastly, we changed colors to make the app look more fitting to the nature aesthetic and overall more pretty.



Bryce Canyon National Park, a sprawling reserve in southern Utah, is known for crimson-colored hoodoos, which are spire-shaped rock formations. The park's main road leads past the expansive Bryce Amphitheater, a hoodoo-filled depression lying below the Rim Trail hiking path. It has overlooks at Sunrise Point, Sunset Point, Inspiration Point and Bryce Point. Prime viewing times are around sunup and sundown.

Figure F4

3 Comparison with Original Project Design Document

Features	Was in Document	Implemented	Details
Analysing weather data	Yes	Yes	This feature lets users get the weather details about the particular National Park.
Navigation system	Yes	Yes	Lets users use the same application to navigate in the National Park.
Photo Gallery	Yes	No	This Feature lets user to add photo to the app like Instagram
Emergency Rescue system	Yes	Yes	It lets users notify the park rangers and ask for help in case of emergency.
Favourites	No	Yes	This lets the user add any national Park of user choice to the favourite list.
Scenic Spots	Yes	Yes	This user a list of places in a Particular National Park.

III Testing

1 Items to be Tested

Items to be covered are as follows:

MarkerInformation Class MarkerInformation Window Google Maps API Geo-coordinates Dialog box Google Maps API Geo-coordinates	Popup functionality General Button functionality Scenic Spots page Get Weather button Firebase Authentication Email/Password functionality Scenic Spots Accurate Data Retrieval

2 Test Specifications

IDT1 - Population of Scenic Spots

Description: Scenic Spots related to chosen national parks are displayed on the Map Page.

Items covered by this test: MarkerInformation Class, MarkerInformation Window, Google Maps API, Geo-coordinates

Requirements addressed by this test: Scenic Spots functionality, Main Page and Map Page

Environmental needs: GPS is on, application needs to have cellular service or internet connection, a national park is chosen via the main page

Intercase Dependencies: NA

Test Procedures: The application must be running, User then logs in, navigates to the home page, clicks on a national park, and then clicks "Go To Map Page".

Input Specification: Different National Parks: Yosemite, Bryce Canyon, Zion National Park, Death Valley

Output Specifications: Map page is opened properly

Pass/Fail Criteria: Map Page is centered on chosen national park and map is populated with related Scenic Spots

IDT2 - Google Map Navigation

Description: The dialog box in the homepage provides a way to switch activities to find directions to the National Park using Google Maps.

Items covered by this test: Dialog box, Google Maps API, Geo-coordinates

Requirements addressed by this test: GPS is on, route is planned, and destination is selected

Environmental needs: GPS is on, application needs to have cellular service or internet connection, start navigation button is pressed

Intercase Dependencies: NA

Test Procedures: Open Dialog box, press the "Navigate with Google Maps" button

Input Specification: Different National Parks: Yosemite, Bryce Canyon, Zion National Park, Death Valley

Output Specifications: Google Maps is opened.

Pass/Fail Criteria: Google Maps needs to be opened from the application's button press, then have the navigation set to the National Park which was input in the search bar and map itself.

IDT3 - Collecting data before sending to database

Description: The function I am testing the function that collects the data of the national park and sends to the database.

Items covered by this test: Scenic Spots Accurate Data Retrieval

Requirements addressed by this test: Give permission to the app to allow the app to use location.

Environmental needs: Connect to the Cellular and Login to the app.

Intercase Dependencies: NA

Test Procedures: In order to run the test, the app needs to be running itself. Users must be able to click on any national park and click on a scenic spot button.

Input Specification: The function that handles the collection of the data must take 5 different inputs. The inputs are name, latitude and longitude, image and description of a particular national park.

Output Specifications: We see an update in the firebase database, under than database name Scenics Spots

Pass/Fail Criteria: Homepage is centered when selecting a national park.

IDT4 - Pop Testing

Description: In this test we need to check each and every national park on the Home Page has a Popup, counting general information about it and four buttons on it with their individual functionality.

Items covered by this test: Popup functionality, and general button functionality

Requirements addressed by this test: Successful full deploying of popup and functionality of buttons in popup

Environmental needs: Connect to the Cellular and Login to the app.

Intercase Dependencies: Population of Scenic Spots and Mape Page.

Test Procedures: Manually click on each and every national park for popup and the buttons in the popups are functional.

Input Specification: There are no inputs specification as it is manual testing, as it is a manual testing.

Output Specifications: The output is a visual function button on the popup window. With the changing of the page it should also close the popup window.

Pass/Fail Criteria: Passing for each button to redirect to a different page is critical which is fulfilled. Although the closing of the popup window is flailing but not the priority.

IDT5 - Passing data to OpenWeather API function

Description: Test the getWeather function which gets the name of the scenic spot, weather, temperature, and date from the coordinate parameter.

Items covered by this test: Scenic spots page, “Get Weather” button

Requirements addressed by this test: Weather advisory for each scenic spot

Environmental needs: Connect to the Cellular and Login to the app.

Intercase Dependencies: Population of Scenic Spots

Test Procedures: In the Scenic Spots page, the user clicks the get weather button on a scenic spot which calls the OpenWeather API

Input Specification: The function that handles getting the weather needs the coordinates

Output Specifications: The function gets the coordinates, finds the location's name, weather, temperature, and date from a JSON and converts it to string data the application will use

Pass/Fail Criteria: Pass if each scenic spot card is updated with that specific scenic spot's weather and name. Fail otherwise.

IDT6 - Authentication

Description: Test the authentication which allows for login and sign-up.

Items covered by this test: Firebase Authentication, email and password functionality

Requirements addressed by this test: Should be logged out of any existing account on the application.

Environmental needs: Connect to the Cellular data.

Intercase Dependencies: NA

Test Procedures: Open the application, select login or sign-up then provide the email and password.

Input Specification: Email and password

Output Specifications: The function gets the email and password, then makes sure that the user exists and the password provided is correct, or to check if the user does not exist if the sign-up button is pressed.

Pass/Fail Criteria: Pass if each the user enters the correct information and is logged in, or if the user's account is added to the database if sign-up

3 Test Results

IDT1 - Population of Scenic Spots

Date(s) of Execution: 3/20/2021, 4/01/2021

Staff conducting tests: Ali Darawsha

Expected Results: Maps page is opened upon pressing national park located within homepage, national park is then displayed with custom scenic spots.

Actual Results: Expected behavior, map page is populated with scenic spots

Test Status: Pass

<u>IDT2 - Google Map Navigation</u>

Date(s) of Execution: 3/20/2021, 4/01/2021

Staff conducting tests: Ali Darawsha

Expected Results: Google Maps is opened from the application's button press, then the navigation is set to the National Park which is populated in the search bar and map itself.

Actual Results: Initial execution: Coordinates were popping up in the search bar and the National Park information was not displaying properly. In the later execution, the function was changed and the National Park correctly displayed and the user was able to find directions and more information typically in a Google Maps search.

Test Status: Fail initially, pass in the end.

<u>IDT3 - Collecting data before sending to the database.</u>

Date(s) of Execution: 3/19/2021 to 3/20/2021

Staff conducting tests: Meet Patel

Expected Results: Meet assumed that test to pass due to the inputs for the functions are collected from 5 arrays, each array associated with each input.

Actual Results: The results are either pass or fail because the function depends on the array and how many elements in the array. If all the arrays have the same number of elements then the result is passed. On the other hand if one array's elements are smaller than the other, the test would fail.

Test Status: Both passed and failed.

<u>IDT4 - Pop Testing</u>

Date(s) of Execution: 3/20/2021 - 4/01/2021

Staff conducting tests: Vedant Majmudar

Expected Results: The popup should close and application should be redirected to the Map Page on selecting the Map page button.

Actual Results: A popup is not closing, we have to manually close it by hitting outside the popup window.

Test Status: Failed partially**.**

IDT5 - Passing data to OpenWeather API function

Date(s) of Execution: 3/30/2021

Staff conducting tests: Joshua Gonzales

Expected Results: Weather, temperature, date, and advisory for the scenic spot should pop up in the specific scenic spot card the user clicked on.

Actual Results: The weather information pops up for every card, with the information detailing the entire park instead.

Test Status: Fail

IDT6 - Authentication

Date(s) of Execution: 2/12/2021

Staff conducting tests: Joshua Gonzales

Expected Results: User is logged in if correct information is provided, provided with an error message if information provided is incorrect. For the sign-up, the account is added to the database.

Actual Results: User is logged in correctly, sign-up works as account is added to the database. However, if a user enters incorrect information, they have to press the login button twice in order to log in with the correct information.

Test Status: Pass

4 Regression Testing

Test ID is subject to regression testing, this is mostly due to the fact that database verification and correct login behaviour is paramount to users experiencing the rest of the applications functionalities.

IV Inspection

1 Items to be Inspected

1) Login Functionality & Scenic Spots Functionality - Josh (either one)
2) Home Page Functionality - Vedant
3) Map Page Functionality - Ali
4) Setting page, Favorites, & Emergency Functionality - Meet

2 Inspection Procedures

The documentation we used to inspect the code:

1. Ali's Coding Inspections:
    a. Electronic meeting with Meet (04/02/2021), Settings Page Functionality
    b. Electronic meeting with Josh (04/02/2021), Favorites Page and Firebase Functionality
    c. Electronic meeting with Vedant (04/02/2021), Home Page Functionality
2. Josh's Coding Inspections:
    a. Electronic meeting with Ali (04/02/2021), Map page for populating scenic spots
    b. Electronic meeting with Meet (04/02/2021), Favorites page for adding data to real-time database on Firebase
    c. Electronic meeting with Vedant (04/02/2021), Home page for dialog box
3. Vedant's Coding Inspections:
    a. Electronic meeting with Ali (04/02/2021), MapPage.dart
    b. Electronic meeting with Meet (04/02/2021), SettingsPage.dart
    c. Electronic meeting with Josh (04/02/2021), AuthService.dart
4. Meet's Coding Inspections:
    a. Electronic meeting with Josh, on (03/26/2021) Implementing openWeather API
    b. Electronic meeting with Ali, on (03/020/2021) Implementing Google Maps Markers
    c. Electronic meeting with Vedant, on (04/02/2021) Implementing Downloads Map of National Parks

3 Inspection Results

1. Ali's Coding Inspections:
    a. Inspecting Meet's Code:

       The structure of the code is straightforward, the global variables used for the settings page are declared in the beginning, directly after the class declaration. Camel case is properly used when necessary and there is even usage of regular expressions in order to parse email entries for validity. The flow is efficient and simple to understand.

    b. Inspecting Josh's Code:

       The code looked over is located within the LoginPage.dart and ScenicPages.dart, as Josh's code expanded over these facets of the application. The loginPage utilizes the Firebase API to create and log in to accounts, the code is simple to understand with minimal API calls made. Global variables are included within the beginning of the class declaration and camel case is used properly when necessary. The Scenic Pages usage of the OpenWeather API was also seamless and did not add any convolution to the creation of the code, the flow was understandable and overall seems properly written.

c. Inspecting Vedant's Code:

This code spanned over the HomePage.dart and utilized many resources that had to be imported in order to be displayed within the page. The HomePage acts as a central hub in which the user will use to traverse the application. As such, the HomePage is filled with the bulk of the information regarding national parks, and this information is properly saved and sifted through via switch cases. The code even utilizes the Firebase database to upload information to the database when it is deemed necessary.

2. Meet's Coding Inspections:
   a. Inspecting Ali's Code:

   The functionality of Google Markers is written on the MapPage.dart file. The variable follows both camel case and snake case. Without comment dart code is always hard to follow no matter who wrote the code. Beginning of the MapPage.dart has clear comments, but when it's time to create Markers comments are not there. In order to get specific markers he has a Map data structure and a switch case. Instead of saving data inside the data structure, first he creates objects and stores data in those objects of Map and then saves it into Map Data Structure. The data are the National Park name and their description.

   b. Inspecting Josh's Code:

   The functionality that is on Scenic spots page is fascinating as it uses the openWeather API which gives live weather information of any place. The variables follow camel casing and use final as data type which means the variables are forbidden to be modified. Since implementing weather api was challenging there was a source that helped him and he gave credit at the top of the Weather.dart page. The code is easy to follow because there are single sentence comments explaining any code that might be hard to read.

   c. Inspecting Vedant's Code:

   One of the functionality that is on Homepage is called Downloding Map Page of a given National Park. The code is easy to understand because the function name is given appropriately, Function does as intended. The variables follow the camel case and comments have added as well.

3. Josh’s Coding Inspections:
    a. Inspecting Ali’s Code:

        The code for the Map Page initially looks good as there are comments throughout for the functions borrowed from the Google Markers API. The variables and functions follow the camel case formatting typically found in Java, which translates well into Flutter. There could be work in setting comments further within the building of the Widget though, because with Flutter’s syntax, things can get jumbled up easily. The overall indentation for the different functions, switch cases, and mappings all seem very well done. The use of the Map data structure also makes sense, which helps make the code easier to read and understand for other developers.

    b. Inspecting Meet’s Code:

        The code Meet presented for a coding inspection involved the adding data to the real-time database on Firebase from the Home page. This code looks good as it follows good indent formatting and comments to what the functions being called are doing. This code bleeds into the Save functionality which also works as intended and follows camelcase.

    c. Inspecting Vedant’s Code:

        Vedant’s code for developing the Home page and its alert dialog box follows camelcase and correctly follows the camelcase formatting we had for naming the assets. There was also clear effort in attempting to format the UI as well, with adjustments to shape, colors, and dimensions. The coding for the text and backgrounds also help fit the theme of the application with a focus on a more nature feel.

4. Vedant’s Coding Inspections:
    a. Inspecting Ali’s Code:

        The code in the MapPage.dart is impleming the map functionality of the application. The code is clearly indented, readable and understandable, thus making it understand and follow incase we have to debug it. This code is following the camelCase coding style in the for naming, which makes it easy to read names and keep in understanding what the variable does. The only thing that can be improved is to add some amount of comments, so every programmer can follow it.

    b. Inspecting Meet’s Code:

        The code is SettingsPage.dart server the function of make a functional setting page with an emergency button. The code in this page is also following CamelCase coding style which makes it easy to understand the name of the variables. The code in this was very readable and well-spaced out. The only thing that can be improved is to add some comments which can make it easy for any programer to follow it!

c. Inspecting Josh’s Code:

The code in this file serves the purpose of creating a login page when the application is started. The naming convention used in this page is camelCase, and easy to follow. This piece of code is well indetade and easy to understand. Also, this consists of many functions, which makes it easy to follow and keep track! The only thing to improve is to add some more comments on this page otherwise it's good enough for any programmer to understand.

V Recommendations and Conclusions

The items inspected passed their testing and inspection processes. Repeated testing of certain functionalities may be deemed necessary, as it pertains to system performance and maintenance.

VI Project Issues

1 Open Issues

1. The check weather function only works for the entire national park instead of at each scenic spot, which is important because there are parks that can be hundreds or thousands of acres big and have different weathers at different points. After trying to debug the system, each weather component of each scenic spot card gets the last coordinate passed into the weather function and applies this to all cards.
2. The use of SQL has not yet been implemented as Firebase was the main way of passing data for this prototype so far. Using SQL may help in the long run as it will be easier to store data for all the hundreds of U.S. national parks and manipulate them within Flutter in the future.
3. The login system causes issues when users enter wrong authentication information, users are unable to login after that. Even if users enter correct information they are still unable to login.
4. The popup feature closing issue. The popup is supposed to redirect the user to the other page on hitting a button on it and close itself. But it is not closing and the use has to click outside the popup and in order to close the popup.

2 Waiting Room

1) Adding more National Parks into the homepage where currently it only has 12. We want to expand our app to all the national parks in the US.
2) Adding Emergency calling functionality to the settings page could help future save lies in the national Parks.

3) A search bar on the HomePage where all the national parks are located, for quick search of national parks.

3 Ideas for Solutions

1. Using SQL and learning about how it works within Flutter would be helpful to get all the data the user would like to know about and explore within the application, rather than having to go out to different sources to discover what they need.
2. Add popup closing function on button click, which will close the popup and conduct its functionality.
3. Adding proper tab traversal via the navigation tab within Flutter.

4 Project Retrospective

Regular weekly meetings helped keep up work ethic, and each of us felt personally responsible for our sections of the application. Splitting the application into sections to then distribute amongst the group was a great foundational step we took in the beginning which ultimately led to smooth feature integration once we came together. Jira helped maintain and tackle issues and provided valuable visual feedback when working on the project. Being in an online platform also helped with flexibility, as the group did not have to physically be present to actually meet, making meetings much more frequent than would have been otherwise. The project helped all of us by essentially mimicking the environment software engineers experience when tackling a project involving other team members.

VII Glossary

National Park: This application's main focus; a scenic or historically important area of countryside protected by the federal government for the enjoyment of the general public or preservation of wildlife.

Home Page: Default tab where the user can sift through national parks and find what they are interested in.

Navigation System: The application comes with a navigation system containing a map of each location, seven days weather forecasts, sunset/sunrise times, facilities available at the park and links to trail maps, campground maps and more.

Weather Advisor: The system will display the forecasted weather in the map with different areas in the park, like fog in the north of the park, or rain in the south of the park.

Photo Gallery: It is not just a photo gallery but more like a web community where pros and fans can share their photos and stories pertaining to the park with one another. This feature helps to find others interested in the park and develop a network between users.

Emergency Rescue: The emergency rescue feature would consist of a very minimal UI with a button. When the button is pressed, the devices' current GPS coordinates are recorded and immediately relayed to rescue teams. There is a high likelihood the device could run out of battery so this must be instantaneous.

Scenic Spots Guide: The scenic spots guide is a map that has a list of scenic spots and animals which are contained within the selected park. Users can select each of the scenic spots or animals to acquire more information about them.

VIII References / Bibliography

[1] "NPS.gov Homepage (U.S. National Park Service)." *National Parks Service*, U.S. Department of the Interior, www.nps.gov/index.htm.

[2] Hongcheng Wu, Spancer Guo, Jiajie Lin, Zachary Flebbe. "Travel Advisor Project Report." *CS440 at the University of Illinois Chicago,* 2020, pp. 1-46.

[3] "National Park" List of National Park for US, wikipedia https://en.wikipedia.org/wiki/List_of_national_parks.

[4] "Firebase Authentication." *Google*, Google, firebase.google.com/docs/auth.

IX Index