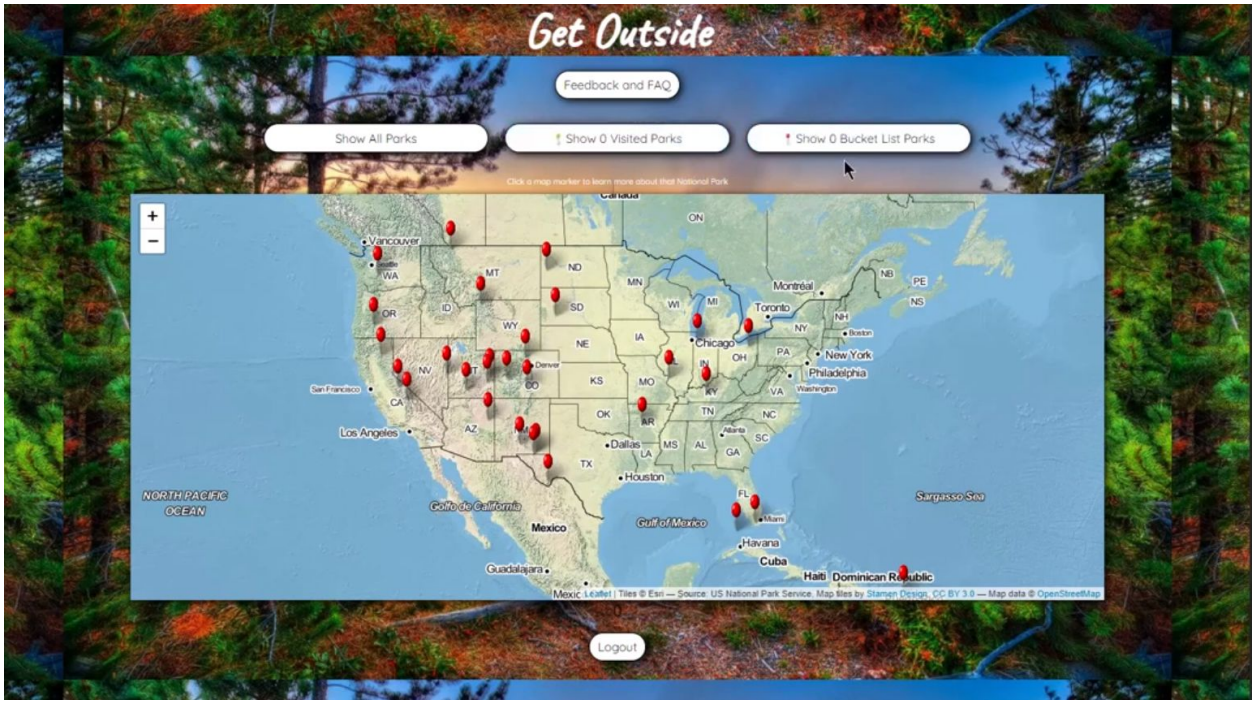
# Get Outside




## Coding Project Final Report

## Prepared By:

**Sanket Shah, Tarush Gupta, Pushti Dalal, Akhil Snehal Modi**
**For CS 440 at**
**University of Illinois at Chicago**

**Fall 2020**

# Table of Contents

# List of Figures

# I Project Description

## 1 Project Overview

The product is a web application that serves as a means to help the user to be able to locate the nearby national parks. The product allows the user to access various information about a national park at one stop. The user can locate a national park in the country and can get the various information about the particular national park such as weather, things to do, etc. The user can also mark the visited national park so that it's in a separate list and can easily differentiate with the unvisited national parks. Users can also mark the national park into the bucket list. Lastly the most important, users can also book the hotels, restaurants, etc. of the nearby national park.

## 2 Project Domain

The application can be used on any web browsers and on any device. Since the application is used on web browsers, it needs to have an internet connection. The application can be used by all users and doesn't have to be the computer expert. Although, users are required to create an account for privacy reasons and because of personalized application ability.

# 3 Relationship to Other Documents

This document serves as an overview that provides sufficient information about the product and its domain in order to put the materials being tested and inspected into proper context. This document stands as supplemental to and should not be considered independent of outside knowledge of the specified currencies. The application is user friendly, but some background knowledge is necessary in order to understand the full scope of the product

# 4 Naming Conventions and Definitions.

## 4a Definitions of Key Terms

Below is the list of some of the most important words which include acronyms and abbreviations.

Show All parks - This will show all the national parks in the maps.

Show visited parks - This will show all the parks which are marked as visited.

Show bucket list - This will show all the parks that are marked in the bucket list.

Bookings - This will show nearby hotels, restaurants and things to do.

## 4b UML and Other Notation Used in This Document

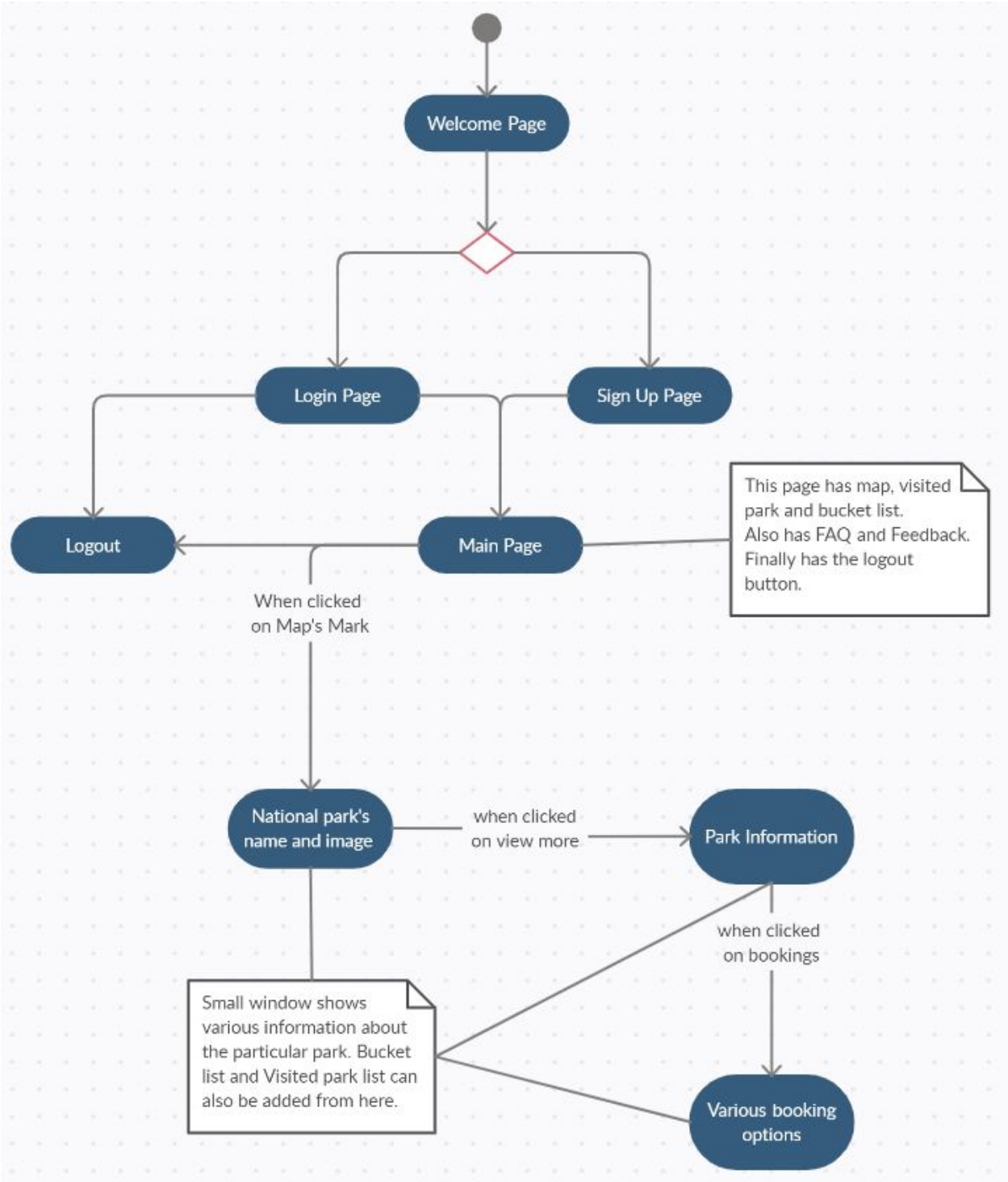



Figure 1 : UML Diagram

### 4c Data Dictionary for Any Included Models

All login and sign up data is stored in the firebase. National Park and Map information is being received from the NPS API.

# II Testing

## 5 Items to be Tested

1. Backend:

a. API Calls

i) #1 - NPS API Response:

Retrieves the National Park data ranging from location, description, weather, nearby places, Hotels, Restaurants, activities etc.

b. Persistent Data Storage:

ii) #2 - Firebase Data Storage

Firebase securely stores the signup and login information of the users, it also consists of National Park's Realtime Database.

2. Frontend:

a. Persistent Data:

i) #3 - Bucket List:

Stores the names of the National Parks through the bucket list button which users plan to visit.

b. Data Loading and Displaying:

i) #4 - Map Markers and National Park Information:
This covers the markers on the maps, park information, weather, activities nearby etc.

ii) #5 - Parks visited list and map marker:

One can add visited parks from the map to be displayed in a list and as green markers on the map.

iii) #6 - Hotel Booking and Restaurant Bookings:
Directs users to external website tripadvisor for the specific nearby Hotels and restaurants around the specific National Parks.

iv) #7 - View more and View less button:

View more button displays the complete and detailed information of National Park and surrounding areas whereas the view less button gives a short description and Picture of the National Park.

v) #8 - FAQ's & Feedback Form :
Users' can check FAQ's page for any information regarding the National Parks and the nearby surroundings. Users of the application can submit feedback for the developers.

# 6 Test Specifications:

**ID #1 - NPS API Response:**

**Description:** Test to check the NPS API is being accessed correctly.

**Items covered by this test:** National Park names, location, weather, description, activities information.

**Environmental needs:** Internet connection, node.js installed, API key accessed.

**Intercase Dependencies:** N/A

**Test Procedures:**

- Request the API key to retrieve the information needed.
- Integrate the API key correctly with the code to access the features.
- Verify the data that is being retrieved.

**Input Specification:** The API key desired, mainly that follows the pattern under the data dictionary.

**Output Specification:** A properly formatted JSON object containing all necessary fields which could be further stored in Firebase Database.

**Pass/Fail Criteria:**

- **Pass**: If the response contains all expected data
- **Pass**: If the product handles missing data elegantly
- **Fail**: if the response does not contain all expected data
- **Fail**: If the product does not handle missing data and throws exception

**ID #2 - Application Data Storage:**

**Description:** Test to check the database storage part of the application works correctly.

**Items covered by this test:** User login and signup details along with the National Park real time database in JSON format.

**Environmental needs:** Firebase project setup, firebase access key, integration of firebase in the codebase.

**Intercase Dependencies:** NA

**Test Procedures:**

- Start the application
- Make sure the correct API key for the firebase is being used in the code.
- Login/Sign up as a dummy account then check on the firebase database if the information is being saved.
- Then check for the information being retrieved from the real time database by retrieving the required information through the code.

**Input Specification:** The API key desired, mainly that follows the pattern under the data dictionary.

**Output Specification:** A storage for the user login and signup and correct information being displayed when retrieved from the JSON in the firebase.

**Pass/Fail Criteria:**

- **Pass**: User credentials are securely being captured
- **Pass**:  Correct data is being displayed in the application when retrieved.
- **Fail**: if the response does not contain all expected data
- **Fail**: If the product does not handle missing data and throws exception

**ID #3 Bucket List:**

**Description:** Test to check the bucket list feature works correctly.

**Items covered by this test:** Correct list of the parks marked by the user.

**Environmental needs:** Internet connection, node.js installed, API key accessed.

**Intercase Dependencies:** NA

**Test Procedures:**

- Mark the National Park that you decide to visit in future
- Check if it displays in the list
- Check if it is showing on the map as green marker

**Input Specification:** User selects the specific park name to be stored in the list.

**Output Specification:** Checks if the list is being generated correctly and the parks are being displayed correctly.

**Pass/Fail Criteria:**

• **Pass**: If the response contains all expected marked markers on the map.

• **Pass**: If the product handles missing data elegantly

• **Fail**: Application loads with none of the last saved preferences

• **Fail**: If the product does not handle missing data and throws exception

**ID #4 Map Marker and National Park Information:**

**Description:** Tests if the map marker is being displayed so that National Park information could be navigated further.

**Items covered by this test:** Correctly display the correct information of National Park being requested.

**Environmental needs:** Internet connection, node.js installed, API key accessed.

**Intercase Dependencies:** NA

**Test Procedures:**

- Click on the map marker
- Mark the National Park that you decide to visit in future
- Check if it displays in the list
- Check if it is showing on the map as green marker

**Input Specification:** User selects the specific park name to be stored in the list.

**Output Specification:** Checks if the list is being generated correctly and the parks are being displayed correctly.

**Pass/Fail Criteria:**

• **Pass**: If the response contains all expected marked markers on the map.

• **Pass**: If the product handles missing data elegantly

• **Fail**: Application loads with none of the last saved preferences

• **Fail**: If the product does not handle missing data and throws exception

**ID #5 Parks visited list and map marker:**

**Description:** Checks if the user is able to mark the visited park and be able to see the list and the map marker.

**Items covered by this test:** Correctly display the correct information of National Park being marked.

**Environmental needs:** Internet connection, node.js installed, API key accessed.

**Intercase Dependencies:** NA

**Test Procedures:**

- Click on the map marker
- Mark the National Park that you have already visited.
- Check if it displays in the list
- Check if it is showing on the map as green marker

**Input Specification:** User selects the specific park name to be stored in the list.

**Output Specification:** Checks if the list is being generated correctly and the parks are being displayed correctly.

**Pass/Fail Criteria:**

• **Pass**: If the response contains all expected marked markers on the map.

• **Pass**: If the product handles missing data elegantly

• **Fail**: Application loads with none of the last saved preferences

• **Fail**: Application marks the incorrect park.

• **Fail**: If the product does not handle missing data and throws exception

**ID #6 Hotels Bookings and Restaurant Bookings:**

**Description:** Tests if the users are able to book the restaurants and hotels easily.

**Items covered by this test:** Correctly display the correct information of Hotels and Restaurants near the National Park being marked.

**Environmental needs:** Internet connection, node.js installed, API key accessed.

**Intercase Dependencies:** NA

**Test Procedures:**

- Click on the bookings button when you explore a particular National Park.
- Click on the Hotels or Restaurants hyperlink.

**Input Specification:** User selects if they want to book a restaurant/hotel.

**Output Specification:**Check if the link directs you to the correct page of the external website TripAdvisor.

**Pass/Fail Criteria:**

• **Pass**: If the response contains all results and directs correctly to the external website.

• **Pass**: If the product handles missing data elegantly

• **Fail**: Application loads with none of the last saved preferences

• **Fail**: Application displays the incorrect Hotel/Restaurant information

• **Fail**: If the product does not handle missing data and throws exception

**ID #7 View more and View less buttons:**

**Description:** Checks if the user is able to navigate through the application easily by using view more and view less buttons.

**Items covered by this test:** Correctly display the correct information when the user presses view less and view more button accordingly.

**Environmental needs:** Internet connection, node.js installed, API key accessed.

**Intercase Dependencies:** NA

**Test Procedures:**

- Click on the view button to learn more about the National Park such as detail description, weather conditions, directions to the park, activities around the park etc.
- Click on the view less button to go the screen that displays brief information

**Input Specification:** Users have the option to see detailed or brief description of the National Park..

**Output Specification:**Check if both the buttons work as expected.

**Pass/Fail Criteria:**

• **Pass**: If when we press the button view more we are able to see in depth information for the National Park .

• **Pass**: If the product handles missing data elegantly

• **Fail**: Application loads with none of the last saved preferences

• **Fail**: Button do not work as expected

• **Fail**: If the product does not handle missing data and throws exception

**ID #8 FAQ'S and Feedback Form:**

**Description:** Testing if the user is able to check the FAQ page correctly and is able to submit the feedback if required in a hassle free environment.

**Items covered by this test:** Correctly display the FAQ page and feedback form works correctly.

**Environmental needs:** Internet connection, node.js installed, API key accessed.

**Intercase Dependencies:** NA

**Test Procedures:**

- Click on the Feedback and the FAQ button check if its working correctly.
- Click on the FAQ hyperlink page to check if it navigates correctly.
- Check the feedback form if the user is able to submit a feedback.

**Input Specification:** Users have the option to check the FAQ page and fill up the feedback form.

**Output Specification:**Check if the button and both the fields work as expected.

**Pass/Fail Criteria:**

• **Pass**: If when we press the button we are able to see submit the feedback and see the FAQ's

• **Pass**: If the product handles missing data elegantly

• **Fail**: Application loads with none of the last saved preferences

• **Fail**: Button do not work as expected

• **Fail**: If the product does not handle missing data and throws exception

# 7 Test Results

**ID #1 - NPS API Response:**

**Dates(s) of Execution:** 11/27/2020

**Staff Conducting Tests:** Tarush Gupta

**Expected Results:** Handle missing data properly. No exceptions thrown

**Actual Results:** Handled missing data properly. No exceptions were thrown

Test Status: Pass

**ID #2 - Application Data Storage:**

**Dates(s) of Execution:** 11/27/2020

**Staff Conducting Tests:** Sanket Shah

**Expected Results:** The user data is being stored securely on Firebase and the real time database also is accessible.

**Actual Results:** The user data is effectively being stored on the Firebase database. No exceptions were thrown

**Test Status:** Pass

**ID #3 - Bucket List:**

**Dates(s) of Execution:** 11/27/2020

**Staff Conducting Tests:** Sanket Shah

**Expected Results:** The user should be able to store the parks to visit in future correctly through the bucket list button

**Actual Results:** Worked correctly and as expected. No exceptions were thrown

**Test Status:** Pass

**ID #4 - Map Marker and National Park Information:**

**Dates(s) of Execution:** 11/27/2020

**Staff Conducting Tests:** Pushti Dalal

**Expected Results:** The user should be able to see the National Parks marked by a marker and can further successfully navigate to the Park information.

**Actual Results:** The user can efficiently navigate through the application . No exceptions were thrown

**Test Status:** Pass

**ID #5 - Parks Visited List and Map Marker:**

**Dates(s) of Execution:** 11/26/2020

**Staff Conducting Tests:** Pushti Dalal

**Expected Results:** The user should be able to see the National Parks which are already visited and could be displayed as the marker on the map.

**Actual Results:** Worked correctly and as expected. No exceptions were thrown

**Test Status:** Pass

**ID #6 - Hotel Bookings and Restaurant Bookings:**

**Dates(s) of Execution:** 11/25/2020

**Staff Conducting Tests:** Akhil Modi

**Expected Results:** The user should be able to book the nearby Hotels and restaurants easily.

**Actual Results:** Users should be able to book the Hotel and restaurant according to the National Park location. No exceptions were thrown

**Test Status:** Pass

**ID #7 - View More and View Less Buttons:**

**Dates(s) of Execution:** 11/22/2020

**Staff Conducting Tests:** Akhil Modi

**Expected Results:** The user should be able to navigate through the application by using view more and view less button depending on the information needed by the user.

**Actual Results:** Worked correctly and the user was able to navigate through the application as expected. No exceptions were thrown

**Test Status:** Pass

**ID #8 - FAQ's and Feedback Forms:**

**Dates(s) of Execution:** 11/21/2020

**Staff Conducting Tests:** Tarush Gupta

**Expected Results:** User should be able to navigate through the FAQ page correctly and be able to submit the feedback form.

**Actual Results:** Worked correctly and the user was able to submit the feedback form as well as access the FAQ's form as expected. No exceptions were thrown

**Test Status:** Pass

## 8 Regression Testing:

When any new features are added all tests should be able to run again and ensure that they pass. This is done to make sure that the new feature hasn't broken any of the old functionality.

# III Inspection

## 9 Items to be Inspected

The following pieces of code were submitted by the following team members to be inspected:

1. Sanket Shah: Buttons.js, Home.js
2. Tarush Gupta: LandingPage.js, ControlForm.js
3. Pushti Dalal: ParkMap.js, ParkList.js
4. Akhil Snehal Modi: App.js, Map.js, redirect.js

## 10 Inspection Procedures

We used the following checklist for doing the code inspections:

https://muthuks.medium.com/here-is-the-checklist-for-reviewing-your-own-react-code-17c03761ac38 [4]

We held a couple of meetings while doing the code inspections. For the very first meeting, all the team members took chances to share their code with the rest of the team and give a brief explanation of what the code is supposed to do. Then, all the team members performed the code inspection on their own times using

the checklist provided above. For the next meeting, all the team members gathered up to discuss their results for the codes they had just inspected. All the team members, one by one, explained the information they gained from the code, and discussed the possible changes, while other members took note of it. The final meeting was held to discuss any follow-up questions.

# 11 Inspection Results

1. Sanket Shah: Buttons.js, Home.js
   a. Tarush Gupta's Inspection (11/20/2020 - 02:30 pm):
      The code was written in such a manner that it was easily readable and understandable, hence making the logic of the code easy to follow. The functions and variables had meaningful names and they each performed their own task. The code also included a generous amount of comments that explained what the code exactly did, which made following the code really easy. One improvement that could be done is that there are a few getters and setters that are defined but not being used in the code. So that can be eliminated.

   b. Pushti Dalal's Inspection (11/23/2020 - 04:45 pm):
      The code written was very readable and well-spaced out. Lots of comments made the code very easy to understand and easy to be followed throughout. The helper classes that were written to read and write the files made the code look a bit cleaner and a well-thought out implementation. Some variable names included numbers in them which made it a little difficult to understand what that variable is defined for. So that could be an area of improvement.

   c. Akhil Modi's Inspection (11/21/2020 - 06:30 pm):

The code was constructed very well. The functions implemented throughout the code were small and performed exactly one functionality each, hence making the code easy to read, understand and debug. The errors and exceptions were also taken care of quite nicely. Only one area of improvement I would suggest would be to log the errors or exceptions to the console so that the code could easily be fixed when needed.

2. Tarush Gupta: LandingPage.js, ControlForm.js

   a. Sanket Shah's Inspection (11/24/2020 - 04:00 pm)
   There was nothing too much to be concerned about in the code provided. It was a very well structured code which was easy to follow. One part of improvement I would suggest would be to check whether the file input was null or not in the constructor, as it could be a potential cause of problems which might later crash the web-app.

   b. Pushti Dalal's Inspection (11/23/2020 - 05:20 pm):
   The code was very well commented and easy to understand. The programmer did a good job in making the control form, which had all the important buttons which would help you navigate throughout the web-app. One thing I would suggest would be to use the string modification functionalities of react, rather than using the '+' operator, which would help speed up the application and use less memory.

   c. Akhil Modi's Inspection (11/24/2020 - 08:45 pm):
   The code was very well implemented and well-structured but I would like to point towards a few things. The code did a very good use of exceptions and exception handling, but also printing the stack trace of the errors would be really helpful when trying to debug the unwanted

conditions. Also, a few strings which are constant can be declared in the class scope, to avoid it being modified later and causing any errors.

3. Pushti Dalal: ParkMap.js, ParkList.js

   a. Sanket Shah's Inspection (11/25/2020 - 12:20 pm):
   The code was written very well overall, and was also very well-structured. It was easy to connect all the dots and understand exactly what was going on. One area of improvement would be make less use of nested loops and conditions as it would make the debugging process cumbersome.

   b. Tarush Gupta's Inspection (11/23/2020 - 11:15 am):
   The code was very well written and very straight-forward in terms of understanding as a reviewer. The way of implementing the interaction of the web-app with the NPS API was seamless, and very-well structured and implemented. The only thing I would suggest would be to add more comments in the code which help the other developers to understand the code.

   c. Akhil Modi's Inspection (11/26/2020 - 11:15 pm):
   This piece of code was very well structured and written but there are a few tweaks which I would like to point out. First, there were a few temporary variables defined, which were initialized from other variables. This could be avoided as the variable s declared earlier could be used. Secondly those class variables could be used directly, which would use less memory and hence, making the web-app use experience more smooth.

4. Akhil Snehal Modi: App.js, Map.js, redirect.js

   a. Sanket Shah's Inspection (11/24/2020 - 07:30 pm):
   This code was written very well and it was self-explanatory. The naming conventions used for the variable names and function names was great. My only suggestion would be to group the similar lines of code more closely and use some more spaces to distinguish between different types of code, hence making it more readable and easy-to-find.

   b. Tarush Gupta's Inspection (11/22/2020 - 05:00 pm):
   Great use of naming conventions throughout the code, making it understandable by just looking at it. Code was also formatted correctly. The way of writing the redirecting code and the code to connect all other parts of the code was done very seamlessly. One area of improvement would be to write a little simple code, as some of the code written here was complex.

   c. Pushti Dalal's Inspection (11/24/2020 - 01:00 pm):
   This code handles the integration of maps from the NPS API to the web-app and shows all the parks using pins on the map. It was implemented very well, and in a very efficient way. The code was very well structured, but adding a few comments won't hurt.

# IV Recommendations and Conclusions

All current features have been properly tested and inspected. All features work as expected. When new features are added, tests will need to be rerun.App needs to be updated on regular basis to check that the database is updated and ready to use

# V Project Issues

## 12 Open Issues

There are some factors that stay to be managed to completely upgrade the item. Even though there were no issues that inalienably impeded the application's usefulness, there were issues that hindered execution. The application was planned without the utilization of multithreading. The issue with not having done this, is that long API calls that set aside longer effort to get a reaction can hinder the fundamental UI string and cause the program to hang and get inert. This was very little of an issue since the greater part of the calls got a reaction rapidly. This would be a more obvious issue if the API calls got greater or different API calls were made on a solitary client activity. Moreover, if the client had a slower web association the client would see the application hanging frequently when making clicks.

## 13 Waiting Rooms

There are certain implementations which have yet to be followed up with which are worth looking into. Although the application has an impressive amount of functionality in the given time, there are some features which could be improved upon. One open ended issue that we had yet to resolve was the in-application booking system. This would have been a significant addition since the application would then be completely self-sufficient without need of external resources. The communication system connected to the national park's authorities is a very essential part for one' safety.

# 14 Ideas for Solutions

The app was initially meant to be a solution for the tourists providing them a safe and secure place to explore. Although the application does a good job with the features that have been implemented, it lacks the implementation of its own booking system along with the communication system. It is important for the product to add this functionality to compete with other products.

The application needs to properly upgrade to keep the information updated in the database and minimize the API calls. This is possible by making all the API calls in an additional thread using the concurrency class. This way while the user is waiting for the data to be downloaded, they can still make other clicks on the UI, as well as cancel the application as needed. This will also solve the problem of the UI being unresponsive for a long API call.

# 15 Project Prospective

Over the course of the project, many organizations were needed to meet the objectives for the final product. To meet weekly objectives the activities were distributed in a fair and efficient manner. During the last few months, the group members continuously divided into pairs of two people. of these groups would have a focus on the front-end or the back end of the application by using pair programming technique. Dividing the tasks into small tasks helped to accelerate the process. If given more time, we would have improved the app by adding few important features. We chose the product based on that fact that it was interesting. In the case that we devoted a bit more time to research before starting on implementation, perhaps we could have come up with even more unique ideas.

# VI Glossary

Show All parks - This will show all the national parks in the maps.

Show visited parks - This will show all the parks which are marked as visited.

Show bucket list - This will show all the parks that are marked in the bucket list.

Bookings - This will show nearby hotels, restaurants and things to do

# VII Reference/Bibliography


[1] Robertson and Robertson, Mastering the Requirements Process.

[2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] Muthu Kumar, “Here is the checklist for reviewing your own React code”, Medium, 02 December 2017, [Online]

# VIII Index