National Park Map ++

A Program to Help National Park Visitors Explore

Prepared by
Brian
Omar
Harsh
Emmanuel
***for use in* CS 440**
at the
University of Illinois Chicago

Spring 2020

Table of Contents

I Project Description 6
- 1 Project Overview 6
- 2 The Purpose of the Project 6
  - 2a The User Business or Background of the Project Effort 6
  - 2b Goals of the Project 7
  - 2c Measurement 7
- 3 The Scope of the Work 7
  - 3a The Current Situation 8
  - 3b The Context of the Work 9
  - 3c Work Partitioning 10
  - 3d Competing Products 10
- 4 The Scope of the Product 10
  - 4a Scenario Diagram(s) 11
  - 4b Product Scenario List 12
  - 4c Individual Product Scenarios 12
- 5 Stakeholders 12
  - 5a The Client 12
  - 5b The Customer 12
  - 5c Hands-On Users of the Product 12
  - 5d Maintenance Users and Service Technicians 12
  - 5e Other Stakeholders 13
  - 5f User Participation 13
  - 5g Priorities Assigned to Users 13
- 6 Mandated Constraints 13
  - 6a Solution Constraints 14
  - 6b Implementation Environment of the Current System 14
  - 6c Partner or Collaborative Applications 14
  - 6d Off-the-Shelf Software 15
  - 6e Anticipated Workplace Environment 15
  - 6f Schedule Constraints 15
  - 6g Budget Constraints 15
- 7 Naming Conventions and Definitions 16
  - 7a Definitions of Key Terms 16
  - 7b UML and Other Notation Used in This Document 16
  - 7c Data Dictionary for Any Included Models 17
- 8 Relevant Facts and Assumptions 17
  - 8a Facts 17
  - 8b Assumptions 17

II Requirements 19
9 Product Use Cases 19
9a Use Case Diagrams 19
9b Product Use Case List 21
9c Individual Project Use Case 22
10 Functional Requirements 30
11 Data Requirements 31
12 Performance Requirements 32
12a Speed and Latency Requirements 32
12b Precision or Accuracy Requirements 33
12c Capacity Requirements 34
13 Dependability Requirements 35
13a Reliability Requirements 35
13b Availability Requirements 35
13c Robustness or Fault-Tolerance Requirements 36
13d Safety-Critical Requirements 36
14 Maintainability and Supportability Requirements 37
14a Maintainability Requirements 37
14b Supportability Requirements 38
14c Adaptability Requirements 39
14d Scalability or Extensibility Requirements 39
14e Longevity Requirements 40
15 Security Requirements 41
15a Access Requirements 41
15b Integrity Requirements 41
15c Privacy Requirements 42
15d Audit Requirements 43
15e Immunity Requirements 44
16 Usability and Humanity Requirements 44
16a Ease of Use Requirements 44
16b Personalization and Internationalization Requirements 45
16c Learning Requirements 46
16d Understandability and Politeness Requirements 46
16e Accessibility Requirements 46
16f User Documentation Requirements 47
16g Training Requirements 47
17 Look and Feel Requirements 47
17a Appearance Requirements 47
17b Style Requirements 48
18 Operation and Environmental Requirements 48
18a Expected Physical Environment 48

18b Requirements for Interfacing with Adjacent Systems 49
18c Productization Requirements 50
18d Release Requirements 50
19 Cultural and Political Requirements 51
19a Cultural Requirements 51
19b Political Requirements 51
20 Legal Requirements 52
20a Compliance Requirements 52
20b Standards Requirements 52
21 Requirements Acceptance Tests 53
21a Requirements - Test Correspondence Summary 53
21b Acceptance Test Descriptions 57

III Design 63
22 Design Goals 63
23 Current System Design 63
24 Proposed System Design 64
24a Initial System Analysis and Class Identification 64
24b Dynamic Modelling of Use-Cases 65
24c Proposed System Architecture 65
24d Initial Subsystem Decomposition 66
25 Additional Design Considerations 68
25a Hardware / Software Mapping 68
25b Persistent Data Management 69
25c Access Control and Security 69
25d Global Software Control 70
25e Boundary Conditions 70
25f User Interface 70
25g Application of Design Patterns 71
26 Final System Design 72
27 Object Design 72
27a Packages 72
27b Subsystem I 72
27c Subsystem II 73
27d Subsystem III 74

IV Project Issues 75
28 Open Issues 75
29 Off-the-Shelf Solutions 76
29a Ready-Made Products 76
29b Reusable Components 76
30 New Problems 76

30a Effects on the Current Environment 77
30b Effects on the Installed Systems 77
30c Potential User Problems 77
30d Limitations in the Anticipated Implementation Environment that May Inhibit the New Product 77
30e Follow-Up Problems 78
31 Migration to the New Product 78
32 Risks 78
33 Costs 78
34 Waiting Room 78
35 Ideas for Solutions 79
36 Project Retrospective 79

V Glossary 79

VI References / Bibliography 80

VII Index 81

I Project Description

1 Project Overview

National Park Map ++ is a fun and interactive national park exploration mobile application that allows any visitor to be a click away from obtaining more information about the park and everything in it. The app maintains a connection with a remote database which contains all the information about the national park the user is currently in or querying about. The user will have access to information like the park general information, animals and their statistics and food regimen, land, trees, water and its statistics, trails, etc.

The application will contain multiple ways of accessing this information so that users aren't limited to a single choice for utilizing the app. The user will be able to use the application with or without location services enabled. Likewise, they will also have the option to view the layout of the national park in a flat view (2 dimensional top view), depth view (3 dimensional lateral view for displaying the precise layout), and they will also have the option to use this application through their camera lens (AR view). This gives all the users a full spectrum of options that suits their needs and desires.

2 The Purpose of the Project

2a The User Business or Background of the Project Effort

The purpose of the app would be to help the business aspect of various national parks who will be the client of the application. The idea is to increase the number of visitors by achieving a better and more interactive experience. By providing everything at the hands of the user we would be providing full control and the navigation would be fun using the augmented reality experience.

This application will be most useful for national park visitors who casually visit parks and are curious about the lay of the land, or for those who love observing and learning about nature. The application gives any kind of use the ability to view more information about something in the park with a single click.

The effort of this project is to give each national park visitor a better experience during their visit by allowing them to have all the information at the tip of their fingers. Likewise, this will also give those visitors who are trying to explore the park a better experience since the application can generate a path for them to walk through so they can go through the whole park.

2b Goals of the Project

The goal of this project is so that national park data can constantly be displayed to the users, as well as collected by the users so that the data can always be updated with the latest information about the park. This will allow current, and especially, future users to have all the previous data plus all new incoming data. This can also lead to more thorough analysis by acting on that data to capture future behavior of the animals, trees, plants, etc. Ultimately, this can serve to improve our understanding of wildlife in these environments. In doing so, this will garner the insight necessary to effectively improve organism lifespans.

2c Measurement

The collection of all the national park data will be able to be stored in a parsable fashion. This is so that we can build trends and start to understand our data and how it shapes our real-world results. It would be positive reassurance to see our business practices and community efforts translate into enhanced experiences of our attendees. As users populate our database with activity and requests. These usage statistics will provide some insight to which of our application's tools are more popular. The parameters will provide context for how our tools are being utilized.

Likewise, we will be able to contrast this data with data that is collected by National parks' environmental specialists. This is so that we can assure our tools have a positive influence on organisms' wellbeing. Given organism sustainability is one of our priorities, the average lifespan of various species will be monitored annually. With the assumption that our park staff' practices are effective, there should be an increase in overall average life expectancy. This should occur ideally across the board, however, will be distinguished by its respective species. As expected, if average life expectancy trends negatively, that would indicate a problem for the team to assess.

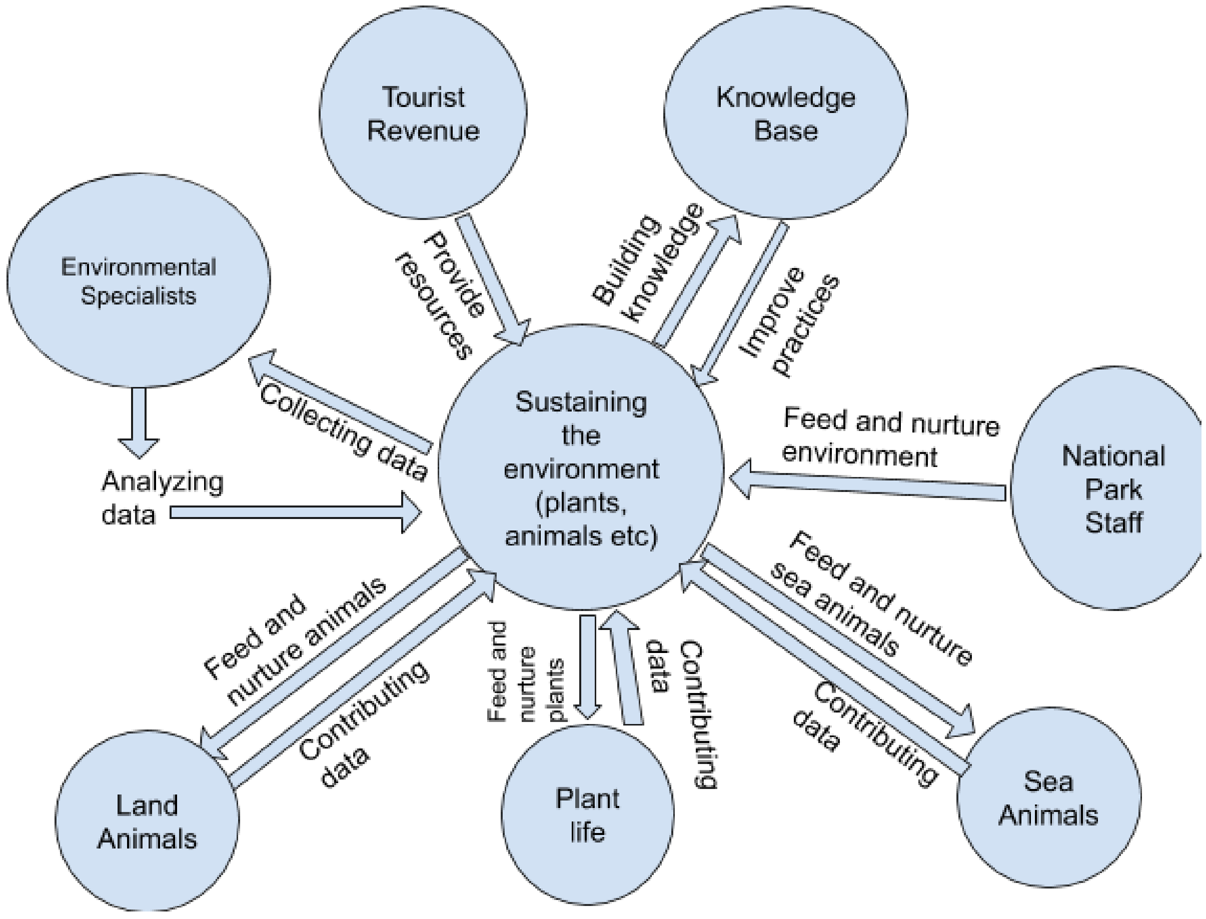
3 The Scope of the Work

Communities are at times put in scenarios where a species needs a dedicated effort to help stimulate their procreation. There are arguments that suggest that humans have contributed to disparities in wildlife populations. This initiative would like to see a reversal of that narrative, with the goal of sustaining our world’s natural inhabitants.

3a The Current Situation

There are more than likely a fair number of individuals who could navigate the National Parks without the assistance of our described companion application. Nevertheless, our application serves purposes that could benefit tourists, as well as end-users. Without the proposed application, users would be subject to paper guides and their own memory to navigate the park. In addition, the National Park themselves would have less data to work with. This long-form data can reveal trends that are simple week by week, or even month by month reporting may not indicate.

3b The Context of the Work



3c Work Partitioning

Event Name	**Input / Output**	**Summary**
Weather station	Weather station readings (in)	Summary
Plant life station	Check the well-being of plants. Water and supply nutrients to all key areas.	Record the health checks on all identified plants in the specified environment
Sea life station	Check the well-being of all sea life and ensure integrity of water habitat(s). Bacteria and microbes need to be kept at a consistent level.	Record the health checks on all identified sea life in the specified environment
Land life station	Check the well-being of all land animals and ensure prosperity of their offspring. Focus on mothers and babies.	Record the health checks on all identified animals in the specified environment

3d Competing Products

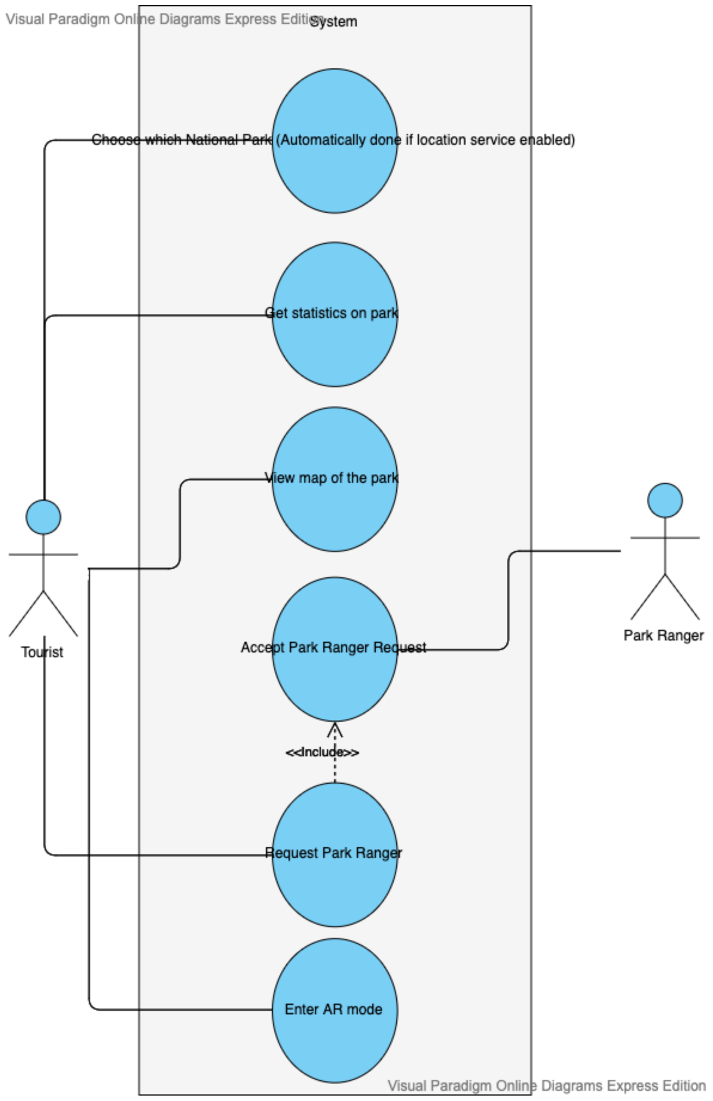
There is no industry standard for a National Park application as described above. This would require an immense amount of data and time/effort; therefore, nobody has seemingly produced a functioning solution. We would be the first ones to break through the niche. Nevertheless, if there were a competing product to ever surface, we would have already established a precedent in regard to this application. Having amassed all our historical data, we would have a strong market advantage and a credible standing in our field.

4 The Scope of the Product

The scope of the project includes all tourists of national parks. The point of this project is to create an easy to use application to view information pertaining to a specific park. The application will contain an easy to use interface; therefore, every tourist/user will be able to use the application.

The user of the application will be able to pick which park they would like to get more information on. They can either have location services enabled to make this process easier or disable the feature entirely. From there it will show a variety of statistics; such as, but not limited to, animals, trails, weather etc. On top it will also show a map of the park. The user can choose either a 2d view or a 3d view. Finally, the user will then have the option to use AR mode which will give them even more capabilities through the app.

4a Scenario Diagram(s)


Visual Paradigm Online Diagrams Express Edition
System
Choose which National Park (Automatically done if location service enabled)
Get statistics on park
View map of the park
Accept Park Ranger Request
<<Include>>
Request Park Ranger
Enter AR mode
Tourist
Park Ranger
Visual Paradigm Online Diagrams Express Edition

4b Product Scenario List

User (tourist) choosing which national park to view
User (tourist) choosing to view statistics for the previously chosen national park
User (tourist) choosing to view map of the park
User (tourist) choosing to enter AR mode

4c Individual Product Scenarios

The user (tourist) begins by choosing which national park to get information on. Either through location services or automatically. The user can then choose to get statistics on that specific park which will display all the information on that specific park. Then the user can view a map of the national park. Finally, the user can choose to interact with the park in AR mode.

5 Stakeholders

5a The Client

Because we are not creating the application for a specific business, the clients are the tourists of national parks. This app enables them to get information about a specific park conveniently.

5b The Customer

The customer is also a tourist since the app is being designed for their use. The app will be tailored to their use. Such as the easy to understand interface.

5c Hands-On Users of the Product

The hands-on users are the tourists. These are primarily the people who are accessing the application and all the information it stores. The tourist will be able to access the application whenever they would like.

5d Maintenance Users and Service Technicians

The maintenance users will be the developers or others who created the app. The app needs to be available with current information regarding each park. If there is a new attraction for a specific park, the maintainers need to update the application with that data. The users then continue to maintain this information by confirming whether a specific animal, attraction etc. is actually there through the application.

5e Other Stakeholders

Software Developers: Software developers are also going to be in charge of the software architecture along with maintaining the stability of the application.
App functionalities are going to rely on all of the teams coming together to make this functional project.
Marketing needs to ensure people are aware of the application so that it is more widely used. On top of this, the more people who use the application the more accurate it will be because users are the ones who confirm whether specific data pertaining to a park is actually true when they visit per se park.
Researchers are needed to provide initial accurate information for each park. These can be the developers or others within the company.

5f User Participation

The user would be expected to use the application through the development process to listen to their concerns, what they want changed, what they want implemented, what works, what does not, and any other important key features that might benefit them. The users would be allowed to talk with the developers in order to get the best system delivered to them. We plan on sending out surveys to see if they are interested in participating in a beta visit to a few parks, before the scheduled release. This will be essential in getting feedback on the user interface, and if the algorithms are helpful enough.

5g Priorities Assigned to Users

The users would be responsible to report and act to any of the actions that are not functioning properly. The workers for the park will report to their supervisors and supervisors to keep a log of any errors that are occurring in the use of the application while the user is in their park which can then be taken care of as soon as possible by the developing team.

6 Mandated Constraints

6a Solution Constraints

The product is being developed to support a fully functional application to provide to information like the park general information, animals and their statistics and food regimen, land, trees, water and its statistics, trails, etc. We need to make sure that the applications are updated recently with all the updated information otherwise the apps would be outdated. There would be an algorithm which would sync any new map updates, provide 3D views, water spots, recreational spots, and many other amenities.
The system will also need access to location and if that is not provided the application would not be able to function to its full potential. The other main requirement is space on the device since the data would be the backbone of the application it would only be fit to provide the application with all these permissions.

The algorithm will provide the users the nearest National Parks and then the users would select the one they want information on. They can also search for the National Parks and pull out information on the same.

6b Implementation Environment of the Current System

The user would have a seamless and perfect experience with full internet capabilities and having a device with these capabilities would give the users an experience which is the goal of the application. As developers, we will make sure to provide as much information as possible for the emergency response teams to make the trip as safe as possible.

6c Partner or Collaborative Applications

As developers combining functionality with different applications are an advantage when combined like for example adding a functionality of restaurants chains nearby would be a great addition to the application. Another great integration would be a compass which would make following the map easier.

6d Off-the-Shelf Software

We made this application to be a cross platform application; one can use a framework like flutter, which is a Google's UI toolkit for crafting beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. For using the application, the user needs a mobile smartphone and access to the application. We can also use other different frameworks like ionic to develop the application. This is a JavaScript framework that allows for app development for cross-platform applications.

6e Anticipated Workplace Environment

The devices should have an active internet connection, the application also gives 3D view of the national park hence you need to have a device with ample RAM and storage space to use the offline functionality. The device should also be a smartphone, preferably a tablet if the users want amazing experience with the application. The tablets would also be best if it was shock/crack resistant since you might drop it while in the park and you don’t want to be stuck there without the application.

6f Schedule Constraints

There aren’t any short-term schedule constraints since each component of the application is built together and will have them all once the product is finished. Overall, having all the components ready and base data collected, this should take no more than a year. For better data, that will eventually be collected over time. This way the full functionally can be delivered with basic data which will be expanded on as more people visit parks.

6g Budget Constraints

The main funds that must be taken care of is the cloud services that will be used for the project. This is primarily the case for the database(s) that will be storing all the information. The cost to have the database(s) can be high since the project will be managing, potentially, a database for each national park, within each national park there will be animals (and stats), land (and stats), etc. So, there will be a lot of database space to be purchased for the project. Likewise, the data must be accessed very quickly since each visitor will want to have the information about something instantly. Likewise, any sort of automated machine learning services may be

required since there will be data analysis happening as the servers constantly collect data.
The budget should not exceed by a lot which depends on the different services used and their pricing. The budget keeping in mind the developers, servers, security, test should be around 20,000 USD.

7 Naming Conventions and Definitions

7a Definitions of Key Terms

Tourist: A person visiting the national park.

Park Ranger: A worker at the national park.

Trails: All physical paths in a park.

Path: A subset of a trail in a park that a tourist can walk through.

Depth View (or Lateral View): Map layout in the application that represents the layout of the national park as if you were standing on the ground and being able to see all the hills and bumps on the ground, heights of trees, depths of the water, etc.

Flat View (or Top View): Map layout in the application that represents the layout of the national park as if you were in the air looking down.

Augmented Reality (AG): Seeing the objects in front of you through the lens of your camera.

Unified Modeling Language (UML): General purpose programming diagrams used for modeling class and how they are related to each other.

Entity-Relationship Diagram (E-R Diagram): General purpose database diagrams used to show different entity sets that will be stored in the database and the relationships that each one has.

7b UML and Another Notation Used in This Document

There will be many UML diagrams that will contain information on class structures and relationships between objects. Likewise, there will be Entity-Relationship diagrams that will help encapsulate all the data being stored on the database and how they are related to one another. Every diagram used will be labeled with their proper name.

7c Data Dictionary for Any Included Models

Animal: a single living entity that lives within a national park
The database entry for an animal's dangerous level ranges from 1 to 5 which correspond to friendly to harmful respectively.
The database entry for an animal's location ranges from 1 to *n* where *n* is the number of sections the park is partitioned into and that range corresponds to a specific zone.

Tree: a single tree entity that is within the national park
The database entry for a tree contains a numeric value corresponding to the type of wood that composes that tree.

8 Relevant Facts and Assumptions

8a Facts

The national park map and data is obtained by public national park data. The backend servers will pre-populate the map data into a format ready to launch the path generation algorithm.

Object information is inputted by any park rangers. This helps prevent anyone to update object data so that it's not false information.

Tourists can utilize the app while at the park or at home just in case they want to observe the park before they visit.

General data, like number of logged in users or will continuously be tracked for each user

Application map view mode data will be recorded for each user so that we can analyze the different kinds of views that the users prefer for further improvement of other view modes.

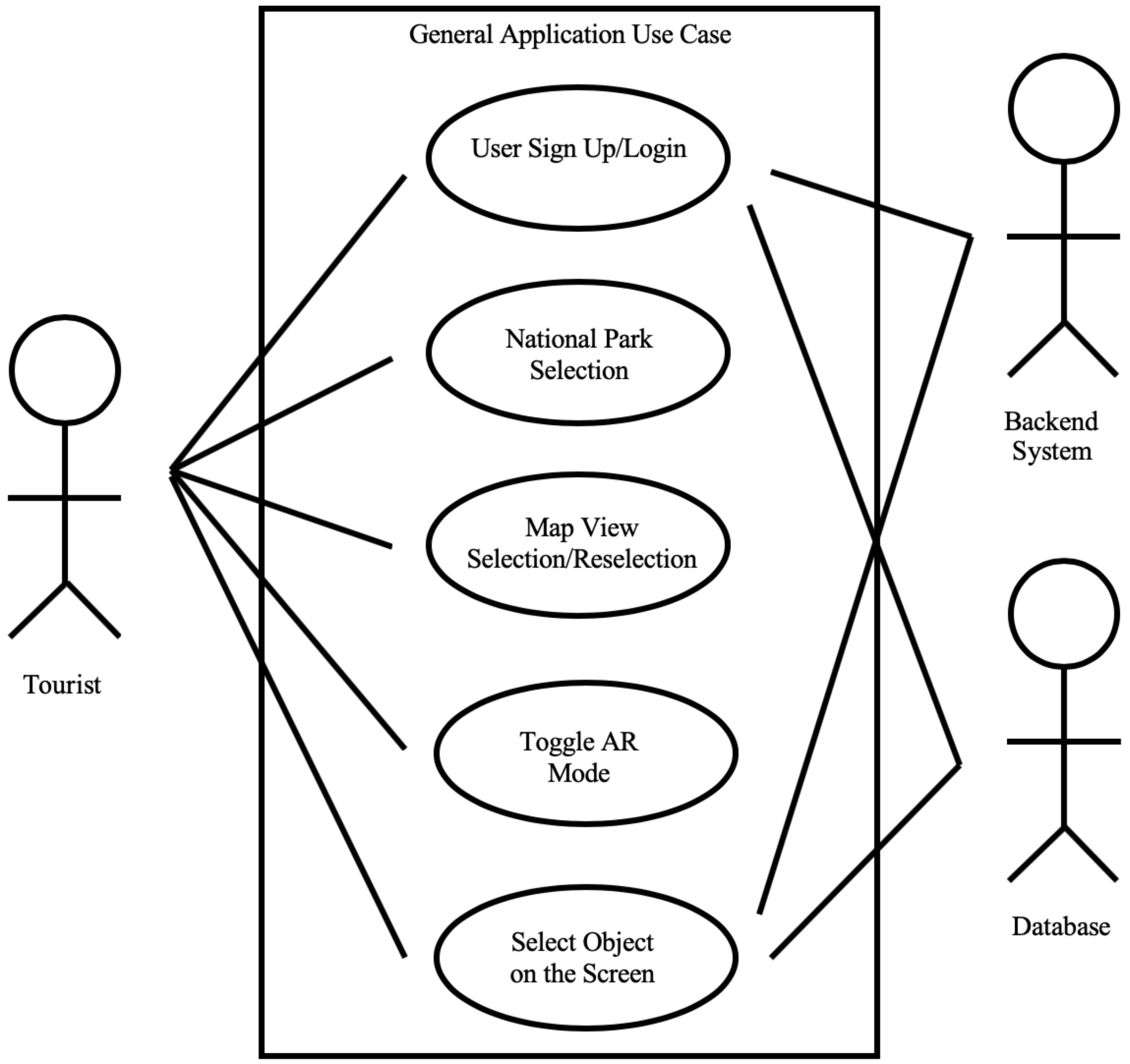
8b Assumptions

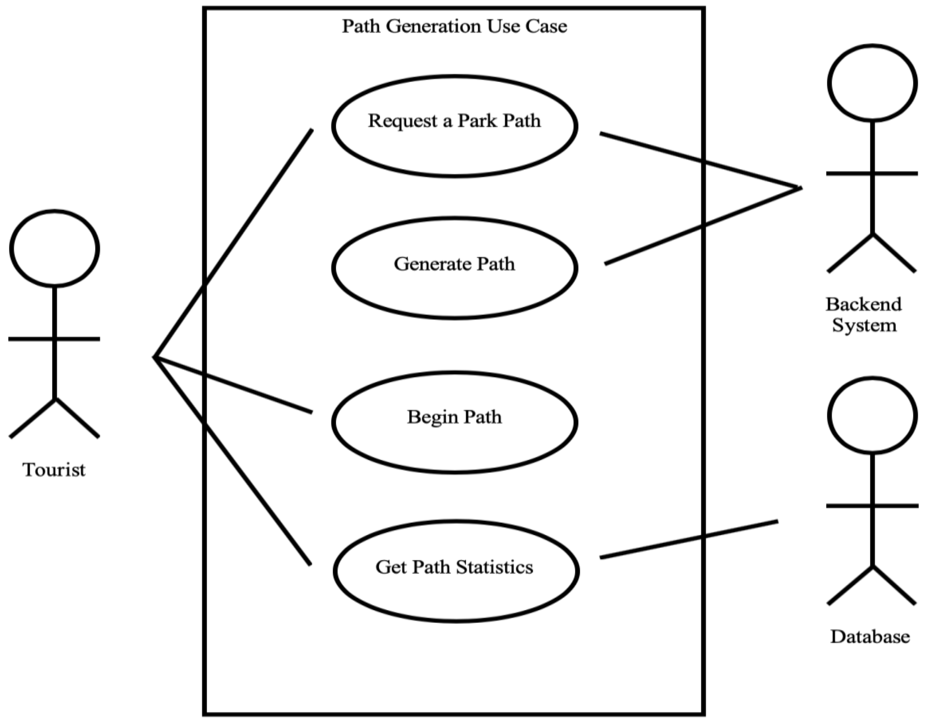
User’s won’t be using a location spoofing 3rd party applications so that they aren’t skewing the data for future users. Likewise, there is the assumption that the park is in operation and not closed for the day or forever. Finally, in order to collect data, there is the assumption that the users must be connected to the internet so that new data can constantly be uploaded to the database.

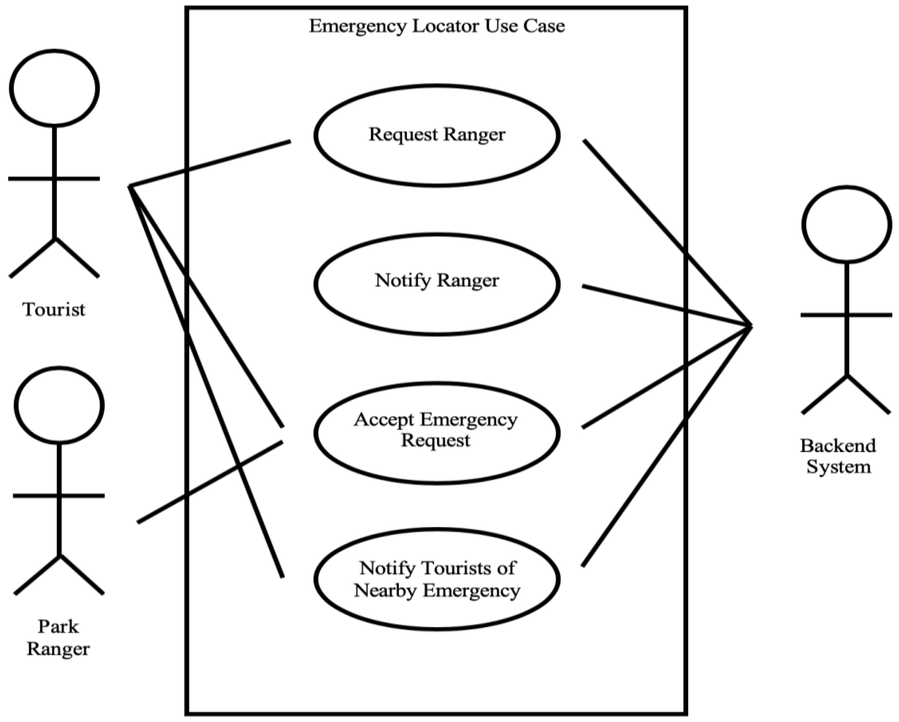
II Requirements

9 Product Use Cases

9a Use Case Diagrams


General Application Use Case
User Sign Up/Login
National Park Selection
Map View Selection/Reselection
Toggle AR Mode
Select Object on the Screen
Tourist
Backend System
Database


Path Generation Use Case
Request a Park Path
Generate Path
Begin Path
Get Path Statistics
Tourist
Backend System
Database


Emergency Locator Use Case
Request Ranger
Notify Ranger
Accept Emergency Request
Notify Tourists of Nearby Emergency
Tourist
Park Ranger
Backend System

9b Product Use Case List

ID	Name	Description
1	Tourist Sign Up/Login	Allows tourists to create an account via email/password, Google account, or Facebook account, or allow them to log back into their existing accounts.
2	National Park Selection	Allows the tourist to select the national park by their location or select it from a list.
3	Map View Selection/Reselection	Allows the tourist to select between the 2D and 3D map view.
4	Toggle AR Mode	Allows the tourist to toggle between the AR Mode.
5	Select Object on the Screen	Allows the tourist to select an object, like an animal or tree, that appears on the screen so they can get more detail about it.
6	Request a Park Path	Allows a tourist to request a path that starts and ends from their location, or a custom destination.
7	Generate Path	Prompts the backend system to take the start and end location requested by the tourist and generates a list of paths that they can take.
8	Begin Path	Allows the tourist to select a path and begin the navigation of it.
9	Get Path Statistics	Generates the path statistics so the tourist can look over and also saves it in the DB so the tourist can access all their paths whenever they want.
10	Request Ranger	Allows the tourist to request a park ranger to visit the tourist’s current location, or a specified location.
11	Notify Ranger	Prompts the backend system to send a notification to all park rangers.
12	Accept Emergency Request	Allows the park ranger to accept or reject the request received.

13	Notify Tourists of Nearby Emergency	Prompts the backend system to notify all the tourists in the park of the emergency request.

9c Individual Project Use Case

Use case ID: 1 Name: Tourist Sign Up/Login Pre-conditions: The tourist must have the application installed and have and active internet connection
Post-conditions: New and existing tourists can login to their account and access the main menu
Initiated by: Any new or existing tourist
Triggering Event: The tourist opens the application
Additional Actors: The backend system and DB
Sequence of Events: 1. The tourist opens the application 2. If the tourist has not signed up before 2.1 Tourist signs up using their credentials 2.2 The backend system will store the new tourist in the DB for future login 3 The tourist uses their login credentials to log into the application 3.1 The backend system will validate the login credentials 4 The main menu screen appears
Alternatives: If the tourist has logged in once, the application auto-logins the tourist Exceptions: N/A

Use case ID: 2 Name: National Park Selection

Pre-conditions: The tourist must have an account to log in with
Post-conditions: The proper national park map is displayed
Initiated by: The tourist
Triggering Event: After the tourist logs into the application
Additional Actors: N/A
Sequence of Events: 1. After the tourist logs into the application 2. Display the national park map in which the player is currently located at with default map view of 2D
Alternatives: If the tourist has denied location tracking permission, then display a list of available national parks so they can choose which one to display Exceptions: N/A

Use case ID: 3 Name: Map View Selection/Reselection Pre-conditions: The proper national park map is currently displayed
Post-conditions: The tourist can select their preferred map view mode
Initiated by: The tourist
Triggering Event: When the tourist clicks the “change view” button
Additional Actors: N/A
Sequence of Events: 1. When the tourist clicks the “change view” button 2. If the current mode is 2D, then switch the map view mode to 3D 3. If the current mode is 3D, then switch the map view mode to 2D

Alternatives: N/A Exceptions: N/A

Use case ID: 4	Name: Toggle AR Mode
Pre-conditions: The proper national park map is currently displayed	
Post-conditions: The tourist will be using their camera as a means of exploring the park	
Initiated by: The tourist	
Triggering Event: When the tourist toggles the "AR Mode" switch	
Additional Actors: N/A	
Sequence of Events: 1. When the tourist toggles the "AR Mode" switch 2. Launch the camera view so the tourist can explore the park through the lens	
Alternatives: If the application doesn't have the camera permissions, then prompt the tourist for it Exceptions: If the tourist has denied camera permissions then tell the tourist they aren't able to access AR Mode	

Use case ID: 5	Name: Select Object on the Screen
Pre-conditions: The proper national park map is displayed, or the tourist is in AR Mode	
Post-conditions: The application will display the information available for the selected object	
Initiated by: The tourist	
Triggering Event: When the tourist clicks on any available object on their map or AR Mode camera screen	
Additional Actors: The backend system and DB	

Sequence of Events: 1. When the tourist clicks on any available object on their map or AR Mode camera screen 2. A request is sent to the backend system to get the object information 3. The backend system queries the DB to retrieve all the recorded information for that object 4. The backend system then returns the results to the application 5. The application will display the information available for the selected object
Alternatives: If there is no information, then the visitor can request a park ranger to go and record basic information about that object Exceptions: N/A

Use case ID: 6 Name: Request a Park Path Pre-conditions: The proper national park map is currently displayed
Post-conditions: The tourist will get a list of paths from the backend system
Initiated by: The tourist
Triggering Event: When the tourist clicks the “Get Path” button
Additional Actors: The backend system
Sequence of Events: 1. When the tourist clicks the “Get Path” button 2. The tourist will input the starting and ending location 2. The application sends a request to the backend system to retrieve a list of available paths to take 3. The backend system generates the paths and returns them to the application 4. The application will display the list of paths the tourist can take
Alternatives: If there are no paths, then return a path that is closest to them

Exceptions: N/A

Use case ID: 7 Name: Generate Path Pre-conditions: The application has made a request to the backend system to retrieve a list of possible paths the tourist can take
Post-conditions: The backend system will have a list of possible paths
Initiated by: The backend system
Triggering Event: When the backend system receives a “GET” request from the application for path generation
Additional Actors: N/A
Sequence of Events: 1. When the backend system gets a “GET” request from the application for path generation 2. Take the starting position and ending position and generate a path 3. For each alternate route that lies within the generated path, generate a modified path and save it into a list
Alternatives: If there are no paths available, return a path that has a starting point close to the requesting starting point Exceptions: N/A

Use case ID: 8 Name: Begin Path Pre-conditions: The tourist has the list of possible paths to take
Post-conditions: The tourist will have the path highlighted on the map or the AR Mode camera
Initiated by: The tourist
Triggering Event: When the tourist clicks on a path from the list
Additional Actors: N/A

Sequence of Events: 1. When the tourist clicks on a path from the list 2. The path is highlighted on the map so the tourist can go through it 3. If the tourist is in AR Mode, then the path will be highlighted through the lens of the camera
Alternatives: If they choose not to select any path, then close the list of paths and display the map once again Exceptions: N/A

Use case ID: 9 Name: Get Path Statistics Pre-conditions: The tourist has finished the path they were on or clicked the "Finish" button
Post-conditions: The application will display the tourist's statistics for the path completed and it's also recorded on the DB
Initiated by: The tourist
Triggering Event: When the tourist has reached the ending position, or they clicked the "Finish" button
Additional Actors: DB
Sequence of Events: 1. When the tourist has reached the ending position, or they clicked the "Finish" button 2. Record the statistics and store them in the DB 3. Display the statistics on the screen for the tourist to see
Alternatives: If the tourist clicked the "Finish" button, then record the statistics accordingly to their ending position Exceptions: N/A

Use case ID: 10 Name: Request Ranger

Pre-conditions: The proper national park map is currently displayed
Post-conditions: The tourist will get a confirmation of their request
Initiated by: The tourist
Triggering Event: When the tourist clicks on the “Request Ranger” emergency button
Additional Actors: The backend system
Sequence of Events: 1. When the tourist clicks on the “Request Ranger” emergency button 2. The application will send a request to the backend system so it can notify the park rangers 3. When successfully sent, the application will notify the tourist that their request has been sent successfully
Alternatives: If there are no park rangers available, then the application should notify the tourist that there are no rangers and they should call 911 Exceptions: N/A

Use case ID: 11 Name: Notify Ranger Pre-conditions: The application has made a request to the backend system to notify the rangers
Post-conditions: All the available park rangers will get notified of the request
Initiated by: The backend system
Triggering Event: When the backend system receives a “GET” request from the application for ranger notification
Additional Actors: N/A
Sequence of Events: 1. When the backend system receives a “GET” request from the application for ranger notification

2. The backend system will notify all the available rangers of the emergency request with its details
Alternatives: N/A Exceptions: N/A

Use case ID: 12 Name: Accept Emergency Request Pre-conditions: The backend system has sent out a notification to all available park rangers
Post-conditions: The tourist will get notified when a ranger is on their way
Initiated by: The park ranger
Triggering Event: When the park ranger accepts the emergency request
Additional Actors: The backend system and the requesting tourist
Sequence of Events: 1. When the park ranger accepts the emergency request 2. The backend system will notify the requesting tourist that there is a ranger on their way 3. The park ranger and the tourist will get an icon on their map of where each person is that updates in real-time
Alternatives: If no park ranger accepts the request, the backend system will re-notify all of the available park rangers once again Exceptions: N/A

Use case ID: 13 Name: Notify Tourists of Nearby Emergency Pre-conditions: The backend system gets a request for a park ranger
Post-conditions: All tourists will get a notification of the emergency
Initiated by: The requesting tourist

Triggering Event: When the backend system receives a "GET" request for ranger notification
Additional Actors: The tourists
Sequence of Events: 1. When the backend system receives a "GET" request for ranger notification 2. Notify all tourists in the park of the current emergency reported
Alternatives: N/A Exceptions: N/A

10 Functional Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
F-1	When the tourists are prompted with the location services request, the application should function regularly regardless of the decision	This allows both kinds of tourists to use the application as intended	The national park map selection will appear after the decision has been made	F-1
F-2	When the tourists are viewing the map, the default view will be the best according to the phone specifications	This allows for tourists with older phones to also enjoy the application	The national park map is selected, it should load to the according view (2D or 3D)	F-2
F-3	When the tourists select an object on the screen, the object information should appear according to what is in the database	This allows the tourists to have information about anything that is around them with a single click	The object information will appear after the user clicks on the object	F-3
F-4	When the tourists request a path around the park, the backend system should provide a	This allows the tourists to have a variety of path options, so they aren't limited to only one path	The list of paths will appear after the user requests a path around the park	F-4

	list of possible paths instead of just a single path			
F-5	When the tourist requests a path, the backend system should be the one to generate the list of paths	This takes the strain off the local application to perform a heavy task just in case the tourist is using a weaker CPU phone	The backend system will return the list of paths to the application when the tourist requests a path	F-5
F-6	When a tourist has reached the end of a path, or they clicked the “Finish” button, the application should report the path statistics to the tourist and the backend system	This allows for the tourist to look at those various statistics when they finish the path and also allows them to view any past paths completed by the tourist	The path statistics will be stored in the DB after the tourist has completed the path or clicked the “Finish” button	F-6
F-7	When a tourist requests a park ranger for some emergency, the backend system should be the central notification deliverer for all park rangers and tourists	This is the easiest way since all the applications in the park are connected to the backend system	All tourists and park rangers will get a notification of the emergency when a tourist requests a park ranger	F-7
F-8	When a park ranger accepts an emergency request from a tourist, the tourist should be notified that there is a park ranger on their way	This is so the tourist is aware that there is indeed someone going to them, just in case they need to at some point dial 911 if no one comes	A notification should appear on the requesting tourists’ screens after the park ranger accepts the request	F-8
F-9	When all available park rangers fail to accept the request, all park rangers will be re-notified once again after some time	This helps the park rangers stay aware that the emergency request is still pending for a park ranger to help	A notification should appear for all available park rangers when one of them fails to accept the emergency request	F-9

11 Data Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
D-1	Tourist passwords must be encrypted	No password should be passed through streams without being encrypted	Passwords must be sent to the backend system encrypted	D-1
D-2	Data for objects around the national park must be created/modified by park rangers	This helps enforce data integrity over objects since park rangers are more knowledgeable and avoids false data from tourists	The data is inputted by park rangers	D-2
D-3	Trails must be pre-populated according to the national park trails and should be updated every month	This helps with the path generation since the layout of the trails are all set ready to launch path generation algorithms	The data will be populated by the system	D-3
D-4	Paths that are generated are stored	This helps future requests to retrieve the generated path if it lies on the same trail	The path won't be generated again, it will be retrieved	D-4
D-5	Emergency requests must store the locations of the emergency	This helps future analysis of locations so tourists can know where emergencies have occurred	Previous emergencies will appear on the tourists maps in the stored locations	D-5

12 Performance Requirements

12a Speed and Latency Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
SL-1	National park map displaying, and icons must be quick	Anytime a tourist logs in or is continuously using the application should see any updates on emergencies, rangers, objects, etc. in a fast manner	Any new item that should appear on a tourist's map must appear no more than 5 seconds from the time it was made available to	SL-1

			the backend system	
SL-2	Path generation should be quick	When a tourist requests a path, they should be able to get their list of paths quickly so they can start going through it	The list should take no more than 10 seconds to generate a path	SL-2
SL-3	Path generation should become much quicker over time	Since D-4 stores the generated paths, future path generations within that area must be retrieved	The list should take no more than the API call delay for a path starting at the same location	SL-3
SL-4	Emergency requests must be sent out immediately	Any emergency must go through the backend system and to park rangers instantaneously since emergencies can sometimes be life threatening	The request should take no more than 3 seconds to reach the park rangers from when the backend system received the request	SL-4
SL-5	Tourists must be notified of any emergency immediately	Any emergency can affect others and every tourist must be notified of it whenever it occurs	The request should take no more than 3 seconds after the backend system receives the request	SL-5

12b Precision or Accuracy Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
PA-1	Map objects must be accurate to their actual location	Every tourist must be able to navigate to the object and be able to see it in front of them, they shouldn’t be far away from the actual location	The object on the map and physically should be at the same location	PA-1
PA-2	Location must be as accurate as possible to where the tourist is currently at	Each tourist should be able to always walk around and have their map be updated accordingly without much delay	The tourist’s map must be updated to their new location every foot they walk in any direction	PA-2

PA-3	Path generation must be accurate, so tourists always stay on the physical trail and not wander off-road	If there is a dangerous path, the tourist must be on the physical path and not wander anywhere that can harm them	The path on the map and the physical path should always match	PA-3
PA-4	Emergency request location must be accurate	If a tourist is in trouble, the emergency request should give an accurate location and not one that is far from where the tourist is located at	The location of the emergency and the tourist must match when a park ranger arrives	PA-4
PA-5	Emergency location notification to all tourists must be accurate	If there is an emergency, then every tourist must be notified of the location of that emergency, so they know the precise location of the emergency	The notified location and actual location must be the exact same	PA-5

12c Capacity Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
C-1	The backend system must handle ~1 million tourists at the same time	There are that many tourists visiting national parks in a single day and should perform efficiently	There should be no extra delay from that many tourists logged in at the same time	C-1
C-2	The backend system must handle ~1 million path generation requests at the same time	With that many tourists in a day, they should all be able to generate paths while visiting the park	Path generation should have no extra delay from having all the tourists requesting a path	C-2
C-3	The backend system must be able to send an emergency notification to ~1 million tourists at the same time	With that many tourists in a day, a notification should be sent out just in case of an emergency with no delay	Notifications should be sent out with no extra delay to all tourists	C-3

13 Dependability Requirements

13a Reliability Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
RERQ-1	No data will be lost or damaged in the event of an error.	All data is corroborated by remote servers.	Any incomplete data should be resubmitted and verified.	RERQ-1
RERQ-2	Local device data will be retained until uploaded to the cloud.	Data must be retrieved by the remote servers, before it can be discarded locally.	Data cannot be discarded until a duplicate set exists on the cloud.	RERQ-2
RERQ-3	Application data must be consistent across devices.	Data must update in real time, to ensure user experience is up to date and accurate.	Any manipulation of data must be reflected on a local device, without any user intervention.	RERQ-3

13b Availability Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
AVAL-1	Application will be available 24 hours, 7 days a week.	Application resources will be accessible from a mobile device, during and after business hours.	Any request, at any time made to the National Park application will return success.	AVAL-1
AVAL-2	Cloud servers should have 99.9% uptime.	The servers have to be online for the application to function properly.	Utilize a reputable cloud hosting solution that guarantees uptime.	AVAL-2
AVAL-3	Cloud servers will have a quick restart	The servers have to be online for the	There is a short window of 1-3-	AVAL-3

	time.	application to function properly. This is in order to minimize downtime.	minute downtime in the scenario of server(s) reinitializing.	

13c Robustness or Fault-Tolerance Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
TOLR-1	Cloud servers must operate with a load balancer.	This is to mitigate heavy load, evenly distribute traffic across endpoints.	Application must withstand 'heavy' usage, witnessing little to no latency.	TOLR-1
TOLR-2	Cloud servers must have a mirror backup that can be surfaced in the event of main server failure.	This is to alleviate any issues that would arise with a main server error. This mirror will contribute to 99.9% uptime.	Application must be fully functional, despite the main server being offline.	TOLR-2
TOLR-3	Local devices will have limited functionality in the absence of a network connection.	Normal functionality will resume upon reconnection to the cloud servers.	All locally stored data will be retained regardless of an internet connection.	TOLR-3

13d Safety-Critical Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
CRIT-1	Users must be prompted with safety advisory with every initial application launch.	Encounters in a national park can be spontaneous and unpredictable. This alert will convey this.	Users must be aware that the national park is not to be held responsible for injuries and damages.	CRIT-1
CRIT-2	First time users must	This mitigates any	Users should not	CRIT-2

	acknowledge and accept the usage agreement.	scenarios of a user not being informed of important national park information.	be able to begin usage of the application until after the agreement is accepted.	
CRIT-3	All application activities should display a disclaimer about safety and user discretion.	This gives every user the opportunity to reevaluate their decision.	Users will make decisions about their trip based on their own discretion, and not solely from this application.	CRIT-3

14 Maintainability and Supportability Requirements

14a Maintainability Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
MAIN-1	Server maintenance must be able to execute automatically, regardless of user intervention.	This is to allow scheduled maintenance during low traffic time periods.	Maintenance functionality can clean up all garbage data, optimize data	MAIN-1
MAIN-2	Server maintenance procedures must be simple and straightforward to execute.	Development teams adapt over time. In the scenario that staff is revamped, they should be able to utilize maintenance functions without extensive knowledge.	Maintenance must be simple enough to be launched by an end-user who is unfamiliar with the underlying technological processes.	MAIN-2
MAIN-3	Developers must provide thorough documentation of all processes so that they can be communicated and	Original developers will have a deeper understanding of code than a new programmer. The presence of thorough	Code continuation will be easier for unfamiliar coders, as the documentation	MAIN-3

	referenced.	documentation is to bridge that gap.	will be thorough.	

14b Supportability Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
SUPP-1	A helpdesk must be reachable from a support line during hours 8am-6pm.	This provides users with a reliable support desk that can assist various inquiries.	Helpdesk should be easily accessible by the user utilizing various methods.	SUPP-1
SUPP-2	Helpdesk must be reachable from an internet portal or email address.	This provides users with alternative contact methods. Utilizing an online portal will also keep an email history of the support ticket.	A received email should create a respective support ticket in the helpdesk system. Similarly, all helpdesk tickets will send email confirmations to all parties involved.	SUPP-2
SUPP-3	Helpdesk support staff must maintain clear communication with developers.	As changes are made to the code, any necessary changes to support processes must also be considered. This ensures that support staff is always conveying reliable information	Prior to every public release, the support staff should be debriefed on all relevant code amendments and application changes.	SUPP-3

14c Adaptability Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
ADAP-1	Application is supported by Android devices.	The vast majority of smartphone users are iPhone and Android users.	This application will be supported by all devices running Android 4.0.x or later (99.9% of users)	ADAP-1
ADAP-2	Application is supported by iPhone and iWatch devices.	The vast majority of smartphone users are iPhone and Android users.	This application will be supported by all devices running iOS 12 or later (95% of users)	ADAP-2
ADAP-3	Application is built with Flutter engine and packages.	Flutter allows code to be generated for iOS and Android compatible machines. This will assist in adaptability as only a single code base is maintained.	Every change made to the code base will be immediately reflected by an Android build, as well as a functionally identical iOS build.	ADAP-3

14d Scalability or Extensibility Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
EXTE-1	Application design choices must have future expansion in mind.	On average, national parks receive millions of daily attendees. These numbers are counted throughout the day and are not necessarily simultaneously.	Application must be able to withstand one million users in a single day.	EXTE-1
EXTE-2	Fetch national park	This allows for	A structured	EXTE-2

	updates and amendments from cloud servers.	seamless tethering to cloud servers. When there are any changes made, all local application instances will have the ability to parse and process the latest information at ease	format allows for the ability to update information regarding national parks without the need to “update” any local deliverable software.	
EXTE-3	Developers will heavily focus on code optimization as the application and servers continue to adapt in size.	It is important that the program code is heavily scrutinized, so that all design choices are effective and feasible to implement.	The program is capable of maintaining high levels of usage for long durations with no hurdles.	EXTE-3

14e Longevity Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
LONG-1	The expected lifetime of this product is essentially indefinite.	National Parks will be around for the remainder of the foreseeable future, as their presence is crucial to our wildlife.	Application should be developed in a way in which its functionalities are timeless.	LONG-1
LONG-2	The product will be profitable within the first year of launch.	The majority of the costs associated are relevant to the cloud infrastructure required to host the application.	Assuming successful application launch, revenue is collected from day one onward.	LONG-2

15 Security Requirements

15a Access Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
ACRE-1	General application data, national park information will be accessible by all logged in users.	This data is public record and exists in external domains as well.	This data will be administered to all authenticated users of the application.	ACRE-1
ACRE-2	Backend application data, such as usage statistics and productivity data. This data can be accessed by National park staff.	This data is both public and private record. This dataset originates from the cloud servers. This data while only accessible by employees may be plotted for public consumption.	This data can be accessed by employees, managers, supervisors, emergency personnel as well as any other staff member.	ACRE-2
ACRE-3	Infrastructure and design administration tools, System administrator tools, webmaster	This aspect is critical to the application's functionalities. Data in this category have a large impact on the system. Given this, only specialized staff are responsible for working with this.	This data can be accessed by top-level employees such as supervisors, and developers. This data is not tampered with very often as it is system configuration data.	ACRE-3

15b Integrity Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
INRE-1	Users current and prior location should	A data breach could result in the hackers	There should be no location of	INRE-1

	not be stored.	being able to access user's location. Putting the security of users at risk.	users stored anywhere on the backend	
INRE-2	There should be no third parties that have access to user's personal information.	National Park ++ has no right to disclose user's personal information to outside parties.	Disclosed in user's agreement and terms and conditions. Banded by law	INRE-2
INRE-3	Users will only have access to their own information and no other users. Unless, that user agrees to share particular information.	Users should not have access to other users' activities and information unless users agree to share that information this violates user's privacy and security.	Users are not able to retrieve information about other users within the app. Unless that user chooses to make that information public.	INRE-3
INRE-4	The initial data about the national park should be inputted by the developers of the application. All data inputted by users need to be verified by other users.	Having developers input the initial data ensures the data will be correct. By having all user input data verified by other users ensures data integrity	Initial data must be inputted by developers. User inputted data must have an option as to whether it is verified by other users.	INRE-4

15c Privacy Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
PRRE - 1	The application will notify all users of information that will be collected, stored and what that information will be utilized for.	By notifying the users of what information National park ++ protects the user's privacy.	The application will notify users of all information it will store when the user first starts to use it. If the application requires more information, it will	PRRE - 1

			notify them with either a push notification or an in-app message	
PRRE - 2	All laws pertaining to users' privacy are followed.	By following the user privacy laws ensures National park ++ will not have any legal issues.	Team of lawyers and engineers must verify that privacy laws are followed for all user stored information for every country the application is used in.	PRRE - 2
PRRE - 3	The application will notify users if there is a change to the privacy policy of the users and users must agree to that change to continue using the app. .	Users have the right to know what changes are made pertaining to their privacy. Having the users agree to the changes avoids legal issues that may arise otherwise.	Either a push notification or an in-app notification to notify users there is a change to their privacy information. The user must then push to agree to continuing using the app.	PRRE - 3

15d Audit Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
AURE - 1	There will be quarterly audits on all software that is pushed to the application. The main functionality of the app will remain the same throughout.	Quarterly audit ensures the software is up to date and the quality of the program is up to par.	The audits will be conducted by a test team. These audits must happen quarterly or by special request. Team must accomplish them in a timely manner.	AURE - 1

	Or by special request of team executives.			

15e Immunity Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
IMRE - 1	The servers will be high grade secure servers. Run from cloud providers. Equipped with anti-virus, firewall and other security software.	Servers must be running close to 99.999 percent of the time. And be secure from viruses and hackers .	Servers are hosted by reliable Cloud hosting providers. Companies must guarantee servers run 99.999 percent of the time.	IMRE - 1
IMRE -2	Within software ensure their code is resistant from Sql injections. Proper logging is used throughout code. Valuable data is encrypted. Continuous security monitoring	Ensures National Park ++ protects user security	Use of unit tests throughout code to test for vulnerabilities. Continuous security monitoring must be done starting from the product launch.	IMRE -2

16 Usability and Humanity Requirements

16a Ease of Use Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
EAUS - 1	The application should be intuitive. Users should require no training to use the app.	Having an easy to use training free application allows all types of users to use the app and makes the app more attractive as	Tested during a beta phase of the application where users try the app within different parks. Users will	EAUS -1

		there is no learning curve.	rate how easy it is to use the application from a scale of 1 - 10. 1 being very easy, 10 being very hard. Average should be below a 4.0.	
EAUS - 2	The application is simple and easy to remember how things are done	Increases user satisfaction as users don’t need to remember how to do certain actions every time.	Tested during a beta phase of the application where users try the app within different parks. Users will rate how simple it is to navigate to prior actions from a scale of 1 - 10. 1 being very simple, 10 being very hard. Average should be below 4.0.	EAUS - 2

16b Personalization and Internationalization Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
PEIN - 1	Users will be able to customize the application by choosing from the world's current 5 most popular languages (at the time of production) by which the application texts appear.	Having an application that is usable in different languages invites more users to use the application as it will be easier for them to understand.	Have at least 5 languages to choose from which users can choose from.	PEIN -1

16c Learning Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
LERE - 1	The application should be easy for any normal (non-technical) person to use with no training necessary.	Ensures that National Park ++ is intuitive and easy to understand on first use.	A sample of non-technical users must be able to enter a navigation to all functionalities within 10 minutes of first using the application. Tested during beta phase.	LERE -1

16d Understandability and Politeness Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
UPRE - 1	The application will only use words and symbols that are understood by the general population (non-technical people)	This avoids the need to force users to learn more terminology to use the app.	A sample of non-technical users must be asked to use the application for a set amount of time. The users will then be surveyed and asked if they didn't understand and words or symbols throughout the application. No user should not understand a term or symbol. beta phase.	UPRE -1

16e Accessibility Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
ACRE - 1	The application will read out loud texts on the screen so that users with visual disability will be able to use it.	Allows users who have sight disabilities to use the product.	A setting within the application will automatically read out all text on the screen.	ACRE -1

16f User Documentation Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
USDO - 1	A user manual with a list of different actions users can do will be accompanied with the application.	If users are struggling to figure out how to do something, or if that action is even possible.	A user manual will be included within the menu options of the application.	USDO -1

16g Training Requirements

ID	Description	Rationale	Fit Criterion	Acceptance Tests
TRRE - 1	No training is required for this application.	Allows application to be easily accessible to all.	After their first encounter with the product, 99 percent of users should agree that no training is needed to use the application.	TRRE -1

17 Look and Feel Requirements

17a Appearance Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
APRE - 1	The application design should appeal to national park enthusiasts.	The goal of the design is to inspire more national park tourists to use the app. If they like the design, they are more likely to use it.	Before building the UI sample a large portion of National Park attendees and ask if they like the design. 80 percent must agree.	APRE -1
APRE - 2	As with any modern application the UI should be clean and not contain a lot of information per menu option.	This allows the app to be easier to understand and more attractive to users.	Each menu option of the app should be dedicated to exactly one aspect of the app. There should be no overlap.	APRE - 1

17b Style Requirements

ID	**Description**	**Rationale**	**Fit Criterion**	**Acceptance Tests**
STRE - 1	The style of the application should appear adventurous	By having an adventurous appearance, it will inspire users to go to national parks and use the application more.	After their first encounter with the product, 80 percent of national park attendees should agree the app inspires them to be more adventurous. Tested during beta phase.	STRE -1

18 Operation and Environmental Requirements

18a Expected Physical Environment

Name - ID	Description	Rationale	Fit Criterion	Acceptance

				Tests
EPE - 1	The product can be used by park visitors that are trekking.	Park Visitors are trying to navigate the park and the maps would provide them with access to their surroundings with walking directions.	The visitors should be able to use the gps location on their device to make the map work and use it to navigate.	EPE -1
EPE - 2	The product can be used by park visitors that are driving.	Park visitors are trying to navigate the park and the maps would provide them with access to their surroundings and the app would provide them with driving directions.	The visitors should be able to use the gps location on their device to make the map work and use it to navigate. For this the user might need to put the phone in the car while driving so it should have sensors needed by maps to provide driving directions.	EPE-2
EPE - 3	The product can be used by park authorities that are driving or on foot for official reasons.	Park authorities may have to use the app for maintenance work in the park.	The product shall be updated in real-time with any maintenance work by the park authorities	EPE-3

18b Requirements for Interfacing with Adjacent Systems

NAME - ID	Description	Rationale	Fit Criterion	Acceptance Tests
IAS-1	The product shall be used mainly on cellular and tablet devices	It is not advisable to go to a park and use a laptop for navigation, hence the product shall support cellular and tablet devices	The product must be able to function using these devices and internet speed minimum to be 10Mb/s since we need to provide a map.	IAS-1

18c Productization Requirements

NAME - ID	Description	Rationale	Fit Criterion	Acceptance Tests
PRO-1	The device that the product is used on should have internet and Wi-Fi capabilities	The product will have to have a cloud-based database which will require internet connectivity for functionality	The product shall be used on different devices such as tablets, browsers and mobile phones.	PRO-1

18d Release Requirements

NAME - ID	Description	Rationale	Fit Criterion	Acceptance Tests
R-1	The regular and security update will be provided to the users monthly.	The update will ensure up to date security and better functionality among the users.	The effort would not be drastic and won’t be affecting the budget a lot.	R-1
R-2	Each release shall not cause the previous release to fail	The update should be building on the previous release and not causing the product to	Each release should be tested with the previous release to ensure smooth transition	R-2

		break or become obsolete		
R-3	A big release will be planned every 6 months.	This release will aim at new and creative changes to the product and make the product more efficient.	The release will be scheduled and practice AGILE methodology.	R-3

19 Cultural and Political Requirements

19a Cultural Requirements

NAME -ID	Description	Rationale	Fit Criterion	Acceptance Tests
CUL-1	The product shall not be offensive to any religious or ethnic groups.	The product is a means to bring communities closer and hence it shall not be offensive in any way.	Park authorities will ensure that these requirements are met.	CUL-1
CUL-2	The product shall not be offensive against a person with disability.	The product may provide special assistance to people with disability but cannot be offensive against them.	Park authorities will ensure that these requirements are met.	CUL-2

19b Political Requirements

NAME - ID	Description	Rationale	Fit Criterion	Acceptance Tests
POL-1	The product will provide super user access to park authorities to edit things related to their park	The product is designed to provide users with the best experience the park can cater and to provide	There will be one single login for each park and that can be used by the park's developing	POL-1

		that they need access to everything.	team.	

20 Legal Requirements

20a Compliance Requirements

NAME - ID	Description	Rationale	Fit Criterion	Acceptance Tests
COR-1	The product doesn’t need personal information like medical details, address and payment details and will not store it.	The product won’t require personal information to achieve best results and will not try to ask the user for it. It will just store names.	No personal data will be stored by the product	COR-1
COR-2	The developer or anyone working on the developing team will have no access to any data since it will all be encrypted	Encrypted data would be passed and only the user will have a decryption key that will be their password through which all the data will be encrypted	Tests will make sure there are no keys that are stored with encrypting it except names.	COR-2

20b Standards Requirements

NAME - ID	Description	Rationale	Fit Criterion	Acceptance Tests
SR-1	The product will be using Azure database firewall to prevent any types DDoS attacks	The product needs to be secure to prevent itself from any kind of data theft and not leak its user information	Have a firewall and tested against DDos attacks.	SR-1

SR-2	The product will be also required to encrypt any user provided data using the database to prevent any type of leaks	Encryption would provide an extra layer of security which would not let the database store personal info preventing major investment in the security sector.	The backend will not have any personal information in its original format and should be encrypted, this could be checked by having the database show tables.	SR-2
SR-3	The product should not store any personal information during the integration of any third-party app and just try to store the data locally on the device.	Since the product might need to include some integrations it needs to make sure not to store any of the personal information again to reduce the cost of security aspect of the product.	The product should fetch data from third party apps and not store them	SR-3

21 Requirements Acceptance Tests

21a Requirements - Test Correspondence Summary

	Requirements														
Test	F-1	F-2	F-3	F-4	F-5	F-6	F-7	F-8	F-9	D-1	D-2	D-3	D-4	D-5	SL-1
F-1	X														
F-2		X													
F-3			X												
F-4				X											
F-5					X										
F-6						X									
F-7							X								
F-8								X							
F-9									X						
D-1										X					
D-2											X				
D-3												X			
D-4													X		
D-5														X	
SL-1															X

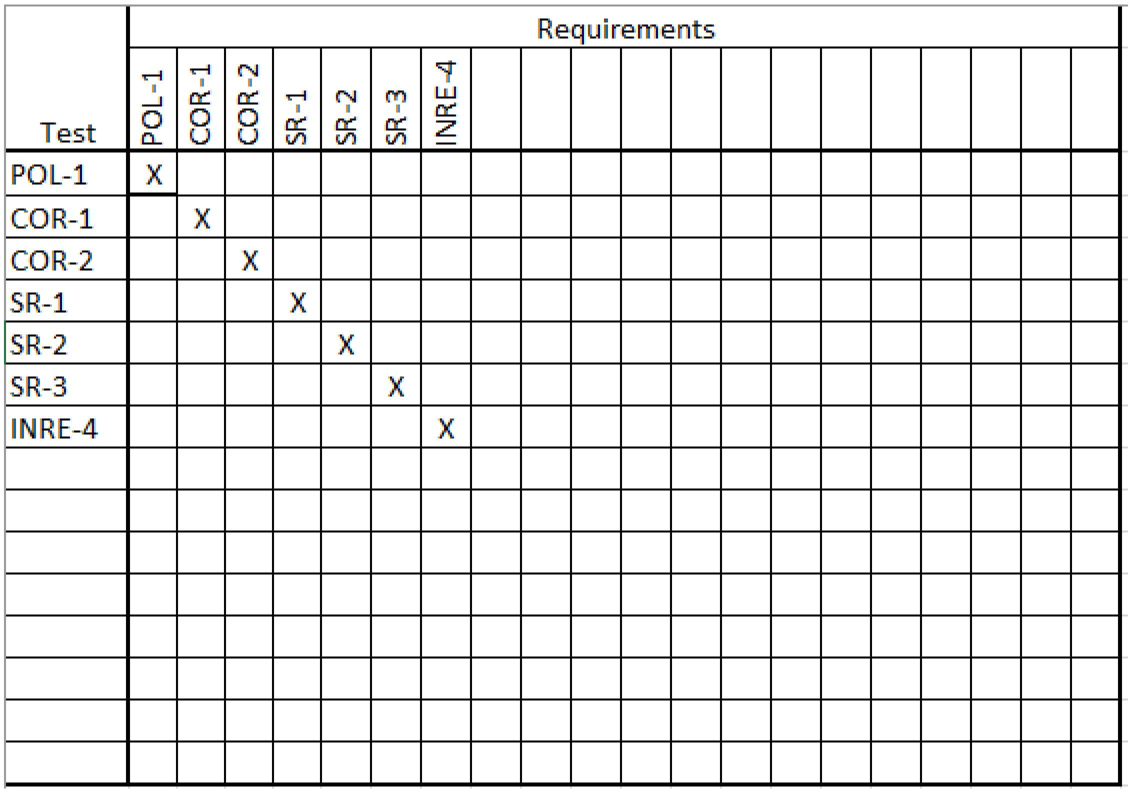
	Requirements											
Test	SL-2	SL-3	SL-4	SL-5	PA-1	PA-2	PA-3	PA-4	PA-5	C-1	C-2	C-3
SL-2	X											
SL-3		X										
SL-4			X									
SL-5				X								
PA-1					X							
PA-2						X						
PA-3							X					
PA-4								X				
PA-5									X			
C-1										X		
C-2											X	
C-3												X

	Requirements														
Test	RERQ-1	RERQ-2	RERQ-3	AVAL-1	AVAL-2	AVAL-3	TOLR-1	TOLR-2	TOLR-3	CRIT-1	CRIT-2	CRIT-3	MAIN-1	MAIN-2	MAIN-3
RERQ-1	X														
RERQ-2		X													
RERQ-3			X												
AVAL-1				X											
AVAL-2					X										
AVAL-3						X									
TOLR-1							X								
TOLR-2								X							
TOLR-3									X						
CRIT-1										X					
CRIT-2											X				
CRIT-3												X			
MAIN-1													X		
MAIN-2														X	
MAIN-3															X

	Requirements														
Test	SUPP-1	SUPP-2	SUPP-3	ADAP-1	ADAP-2	ADAP-3	EXTE-1	EXTE-2	EXTE-3	LONG-1	LONG-2	ACRE-1	ACRE-2	ACRE-3	
SUPP-1	X														
SUPP-2		X													
SUPP-3			X												
ADAP-1				X											
ADAP-2					X										
ADAP-3						X									
EXTE-1							X								
EXTE-2								X							
EXTE-3									X						
LONG-1										X					
LONG-2											X				
ACRE-1												X			
ACRE-2													X		
ACRE-3														X	

Test	Requirements														
	INRE-1	INRE-2	INRE-3	PRRE-1	PRRE-2	PRRE-3	AURE-1	IMRE-1	IMRE-2	EAUS-1	EAUS-2	PEIN-1	LERE-1	UPRE-1	ACRE-1
INRE-1	X														
INRE-2		X													
INRE-3			X												
PRRE-1				X											
PRRE-2					X										
PRRE-3						X									
AURE-1							X								
IMRE-1								X							
IMRE-2									X						
EAUS-1										X					
EAUS-2											X				
PEIN-1												X			
LERE-1													X		
UPRE-1														X	
ACRE-1															X

Test	Requirements														
	USDO-1	TRRE-1	APRE-1	APRE-2	STRE-1	EPE-1	EPE-2	EPE-3	IAS-1	PRO-1	R-1	R-2	R-3	CUL-1	CUL-2
USDO-1	X														
TRRE-1		X													
APRE-1			X												
APRE-2				X											
STRE-1					X										
EPE-1						X									
EPE-2							X								
EPE-3								X							
IAS-1									X						
PRO-1										X					
R-1											X				
R-2												X			
R-3													X		
CUL-1														X	
CUL-2															X



	Requirements																			
Test	POL-1	COR-1	COR-2	SR-1	SR-2	SR-3	INRE-4													
POL-1	X																			
COR-1		X																		
COR-2			X																	
SR-1				X																
SR-2					X															
SR-3						X														
INRE-4							X													

21b Acceptance Test Descriptions

F-1 : The national park map selection will appear after the decision has been made

F-2: The national park map is selected; it should load to the according view (2D or 3D)

F-3: The object information will appear after the user clicks on the object

F-4: The list of paths will appear after the user requests a path around the park

F-5: The backend system will return the list of paths to the application when the tourist requests a path

F-6: The path statistics will be stored in the DB after the tourist has completed the path or clicked the “Finish” button

F-7: All tourists and park rangers will get a notification of the emergency when a tourist requests a park ranger

F-8: A notification should appear on the requesting tourists’ screens after the park ranger accepts the request

F-9: A notification should appear for all available park rangers when one of them fails to accept the emergency request

D-1: Passwords must be sent to the backend system encrypted

D-2: The data is inputted by park rangers

D-3: The data will be populated by the system

D-4: The path won't be generated again, it will be retrieved

D-5: Previous emergencies will appear on the tourists maps in the stored locations

SL-1: Any new item that should appear on a tourist's map must appear no more than 5 seconds from the time it was made available to the backend system

SL-2: The list should take no more than 10 seconds to generate a path

SL-3: The list should take no more than the API call delay for a path starting at the same location

SL-4: The request should take no more than 3 seconds to reach the park rangers from when the backend system received the request

SL-5: The request should take no more than 3 seconds after the backend system receives the request

PA-1: The object on the map and physically should be at the same location

PA-2: The tourist's map must be updated to their new location every foot they walk in any direction

PA-3: The path on the map and the physical path should always match

PA-4: The location of the emergency and the tourist must match when a park ranger arrives

PA-5: The notified location and actual location must be the exact same

C-1: There should be no extra delay from that many tourists logged in at the same time

C-2: Path generation should have no extra delay from having all the tourists requesting a path

C-3: Notifications should be sent out with no extra delay to all tourists

RERQ-1: Any incomplete data should be resubmitted and verified.

RERQ-2: Data cannot be discarded until a duplicate set exists on the cloud.

RERQ-3: Any manipulation of data must be reflected on a local device, without any user intervention.

AVAL-1: Any request, at any time made to the National Park application will return success.

AVAL-2: Utilize a reputable cloud hosting solution that guarantees uptime.

AVAL-3: There is a short window of 1-3-minute downtime in the scenario of server(s) reinitializing.
TOLR-1: Application must withstand 'heavy' usage, witnessing little to no latency.

TOLR-2: Application must be fully functional, despite the main server being offline.

TOLR-3: All locally stored data will be retained regardless of an internet connection.

CRIT-1: Users must be aware that the national park is not to be held responsible for injuries and damages.

CRIT-2: Users should not be able to begin usage of the application until after the agreement is accepted.

CRIT-3: Users will make decisions about their trip based on their own discretion, and not solely from this application.

MAIN-1: Maintenance functionality can clean up all garbage data, optimize data

MAIN-2: Maintenance must be simple enough to be launched by an end-user who is unfamiliar with the underlying technological processes.

MAIN-3: Code continuation will be easier for unfamiliar coders, as the documentation will be thorough

SUPP-1: Helpdesk should be easily accessible by the user utilizing various methods.

SUPP-2: A received email should create a respective support ticket in the helpdesk system. Similarly, all helpdesk tickets will send email confirmations to all parties involved.

SUPP-3: Prior to every public release, the support staff should be debriefed on all relevant code amendments and application changes.

ADAP-1: This application will be supported by all devices running Android 4.0.x or later (99.9% of users)

ADAP-2: This application will be supported by all devices running iOS 12 or later (95% of users)

ADAP-3: Every change made to the code base will be immediately reflected by an Android build, as well as a functionally identical iOS build.

EXTE-1: Application must be able to withstand one million users in a single day.

EXTE-2: A structured format allows for the ability to update information regarding national parks without the need to “update” any local deliverable software.

EXTE-3: The program is capable of maintaining high levels of usage for long durations with no hurdles.

LONG-1: Application should be developed in a way in which its functionalities are timeless.

LONG-2: Assuming successful application launch, revenue is collected from day one onward.
ACRE-1: This data will be administered to all authenticated users of the application.

ACRE-2: This data can be accessed by employees, managers, supervisors, emergency personnel as well as any other staff member.

ACRE-3: This data can be accessed by top-level employees such as supervisors, and developers. This data is not tampered with very often as it is system configuration data.

INRE-1: There should be no location of users stored anywhere on the backend

INRE-2: Disclosed in users' agreement and terms and conditions. Binded by law

INRE-3: Users are not able to retrieve information about other users within the app. Unless that user chooses to make that information public.

PRRE-1: The application will notify users of all information it will store when the user first starts to use it. If the application requires more information, it will notify them with either a push notification or an in-app message

PRRE-2: Team of lawyers and engineers must verify that privacy laws are followed for all user stored information for every country the application is used in.
PRRE-3: Either a push notification or an in-app notification to notify users there is a change to their privacy information. The user must then push to agree to continuing using the app.

AURE-1: The audits will be conducted by a test team. These audits must happen quarterly or by special request. Team must accomplish them in a timely manner.

IMRE-1: Servers are hosted by reliable Cloud hosting providers. Companies must guarantee servers run 99.999 percent of the time.

IMRE-2: Use of unit tests throughout code to test for vulnerabilities.
Continuous security monitoring must be done starting from the product launch.

EAUS-1: Tested during a beta phase of the application where users try the app within different parks. Users will rate how easy it is to use the application from a scale of 1 - 10. 1 being very easy, 10 being very hard. Average should be below a 4.0

EAUS-2: Tested during a beta phase of the application where users try the app within different parks. Users will rate how simple it is to re-navigate to prior actions from a scale of 1 - 10. 1 being very simple, 10 being very hard. Average should be below 4.0.

PEIN-1: Have at least 5 languages to choose from which users can choose from.

LERE-1: A sample of non-technical users must be able to enter a navigation to all functionalities within 10 minutes of first using the application. Tested during beta phase.

UPRE-1: A sample of non-technical users must be asked to use the application for a set amount of time.

ACRE-1: A setting within the application will automatically read out all text on the screen.

USDO-1: A user manual will be included within the menu options of the application.

TRRE-1: After their first encounter with the product, 99 percent of users should agree that no training is needed to use the application

APRE-1: Before building the UI sample a large portion of National Park attendees and ask if they like the design. 80 percent must agree.

APRE-2: Each menu option of the app should be dedicated to exactly one aspect of the app. There should be no overlap.

STRE-1: After their first encounter with the product, 80 percent of national park attendees should agree the app inspires them to be more adventurous. Tested during beta phase

EPE-1: The visitors should be able to use the gps location on their device to make the map work and use it to navigate.

III Design

22 Design Goals

Seeing that there are hundreds of thousands of visitors on a daily basis throughout all the national parks, our overall system design goal is to have the capacity and versatility to support all these visitors being connected at once.

Providing a powerful database to support high volume complex queries is essential for fast data retrieval for any visitor. The backend system will be chosen with high computing power and fast requests in mind. With all the visitors requesting to generate a path through the national park, the backend system should be able to compute these paths and return to the visitors in a timely manner.

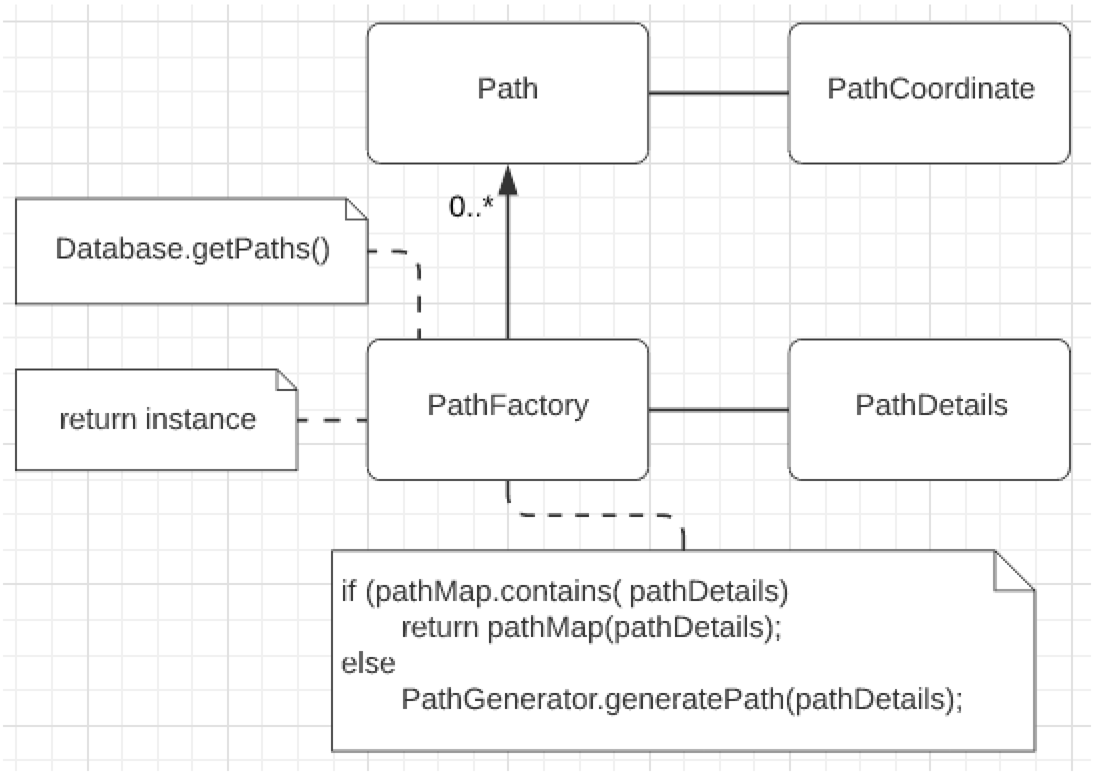
Paring the backend system and database with a good, clean, and responsive mobile application UI ensures that the visitors have the best experience when utilizing the application. Having a minimal clean UI gives the best user experience for all the visitors.

23 Current System Design

There is no current system design. The proposed system design will help replace the traditional paper maps given to the visitors when they arrive at each national park. Likewise, the proposed system design will help give the visitors a single place to view the overall national park map and interact with their surroundings as they explore.

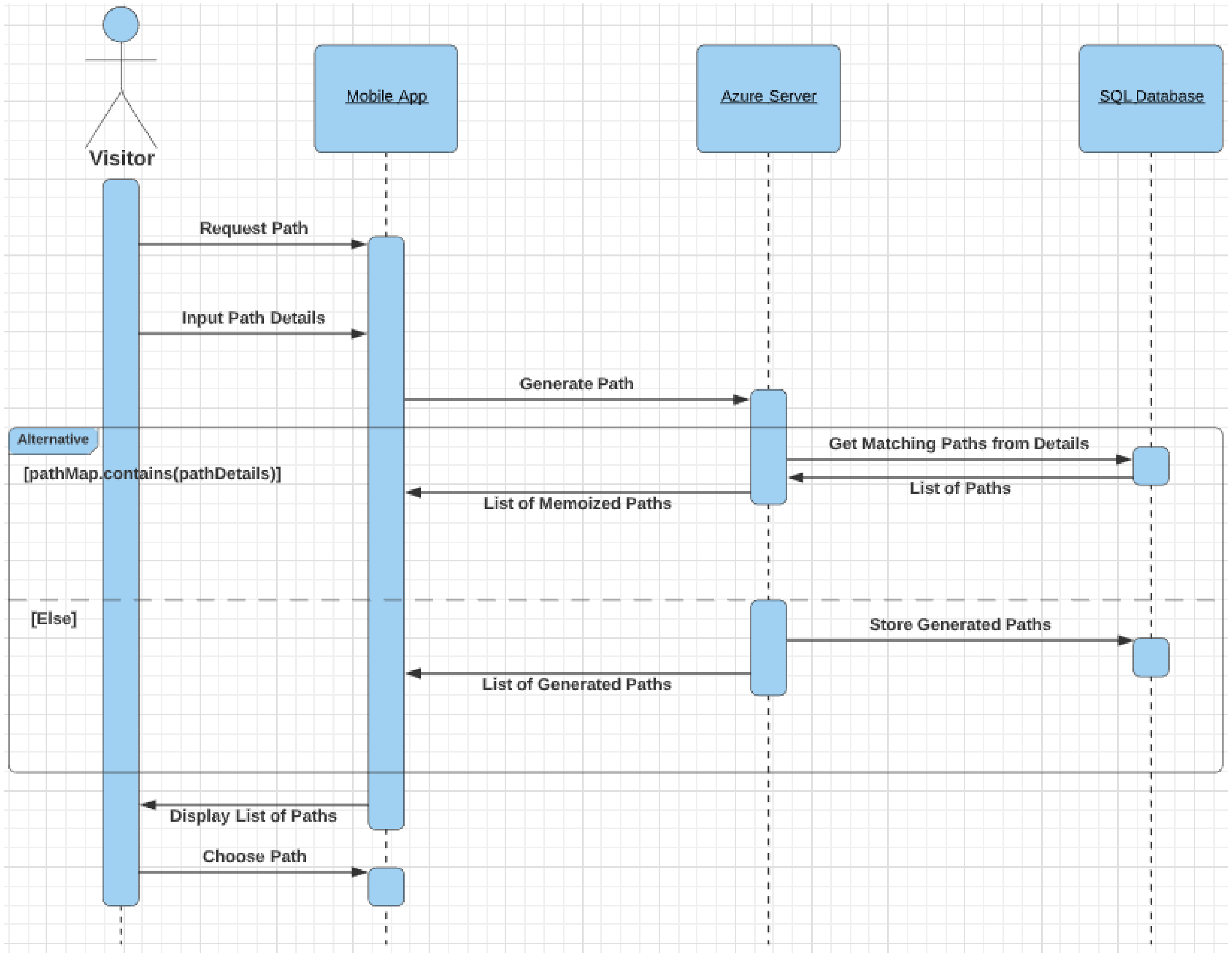
24 Proposed System Design

24a Initial System Analysis and Class Identification



```
if (pathMap.contains( pathDetails)
        return pathMap(pathDetails);
else
        PathGenerator.generatePath(pathDetails);
```

24b Dynamic Modelling of Use-Cases



24c Proposed System Architecture

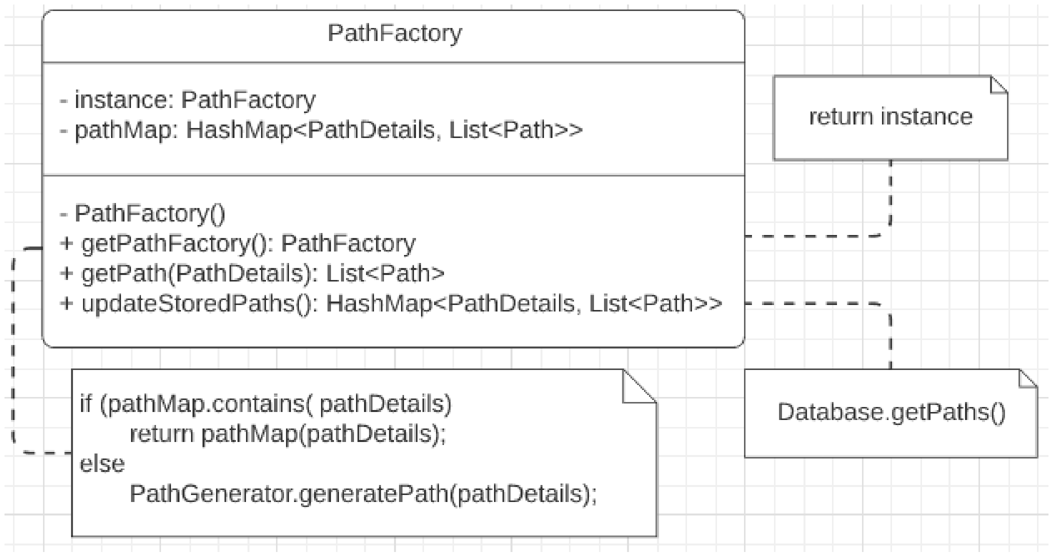
Frontend: Flutter, is a very easy to use and powerful Google's UI toolkit for crafting beautiful, natively compiled applications for mobile and desktop from a single codebase. It is an open source mobile UI framework created by Google and released in May 2017. It allows you to create native mobile applications with only one codebase, this allows you to create an application for both Android and iOS from the same codebase.

Backend: Azure is a reliable and consistent cloud service provider from Microsoft. Azure functions, a service within Azure, will automatically scale the server to the needs of users. Using the Azure functions framework, developers can easily create HTTP endpoints in Java. On top of this, developers can then monitor the status of their API's examining how many of each status code was produced per each function.

Database: The database of choice for this system is a full **MS SQL** server and database. This allows the developers to manage and maintain the server alongside the backend through the azure portal. Likewise, the versatility and dynamic features in computational cores for the database and storage helps with the high volume of simultaneous queries and their speeds. Likewise, having this database allows for T-SQL queries for any backtracking needed in case of errors.

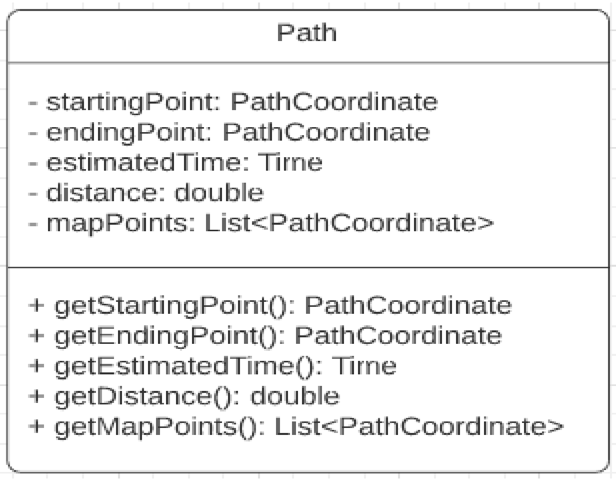
24d Initial Subsystem Decomposition

PathFactory: Singleton factory class used to retrieve a list of previously generated paths that match all the criteria in the PathDetails that was passed into the getPath method. This helps with complex computations since the same requested path has been generated before. With the growing complexity of the paths, the path generation algorithm can take a lot of time to complete, so there will be a trade-off since the amortized time complexity will decrease dramatically while the space complexity will increase. This class contains a HashMap from PathDetails to a list of Path(s) due to the fact that this is implementing the Flyweight design pattern. Likewise, since we store the previously generated paths, the class also implements the Singleton design pattern so there isn't more than one collection of the previously generated paths.

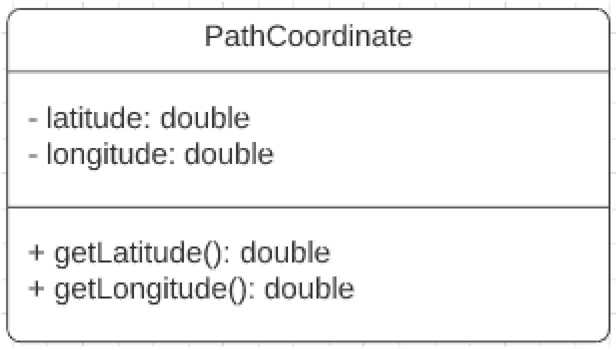

PathFactory
- instance: PathFactory
- pathMap: HashMap<PathDetails, List<Path>>
- PathFactory()
+ getPathFactory(): PathFactory
+ getPath(PathDetails): List<Path>
+ updateStoredPaths(): HashMap<PathDetails, List<Path>>
return instance
Database.getPaths()

```
if (pathMap.contains( pathDetails)
      return pathMap(pathDetails);
else
      PathGenerator.generatePath(pathDetails);
```

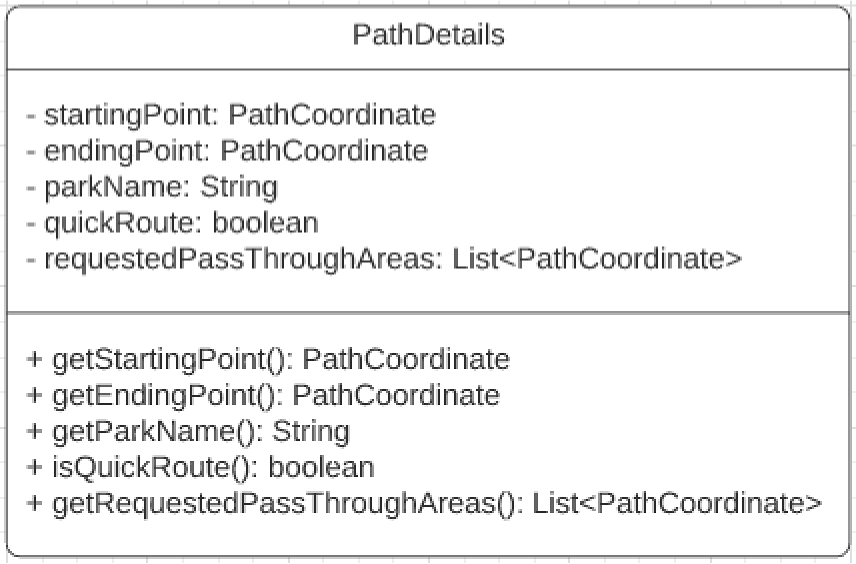
Path: Important class used to store information about generated paths like the starting and ending point, estimated time, distance, and most importantly the actual map coordinates to display on the application's map for the user to see. This path was either stored and retrieved from the SQL database to avoid recomputations, or it's a completely new path which was generated. When the path gets generated, the backend will store this path on the SQL database for future use.


Path
- startingPoint: PathCoordinate
- endingPoint: PathCoordinate
- estimatedTime: Time
- distance: double
- mapPoints: List<PathCoordinate>
+ getStartingPoint(): PathCoordinate
+ getEndingPoint(): PathCoordinate
+ getEstimatedTime(): Time
+ getDistance(): double
+ getMapPoints(): List<PathCoordinate>

PathCoordinate: Information class used to store the latitude and longitude of a point that goes on a path. When a user requests a path, the backend will create a PathCoordinate for the encrypted starting and ending point of their path. This allows for the decoupling of a Path from its starting and ending coordinates since no path can be generated without them.



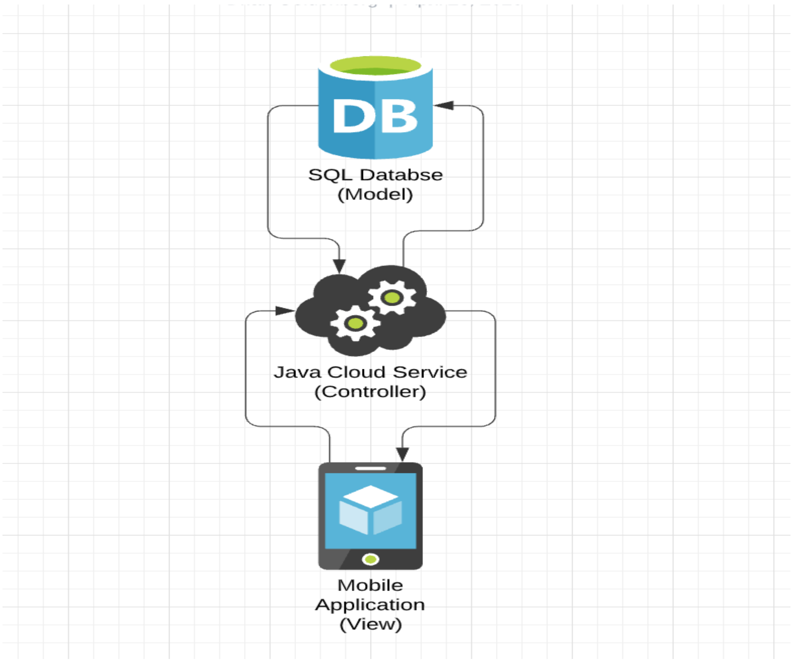
PathDetails: Information class used to store the requested path details. These details include the starting and ending points the user wants to have for their path, the park name, whether it's a quick route, and finally the areas of the park they wish to pass through when generating the path. This allows the user to personalize their path and also provides a basis for the Flyweight design pattern so the flyweight factory can have a means to map the generated paths.



```
PathDetails
- startingPoint: PathCoordinate
- endingPoint: PathCoordinate
- parkName: String
- quickRoute: boolean
- requestedPassThroughAreas: List<PathCoordinate>
+ getStartingPoint(): PathCoordinate
+ getEndingPoint(): PathCoordinate
+ getParkName(): String
+ isQuickRoute(): boolean
+ getRequestedPassThroughAreas(): List<PathCoordinate>
```

25 Additional Design Considerations

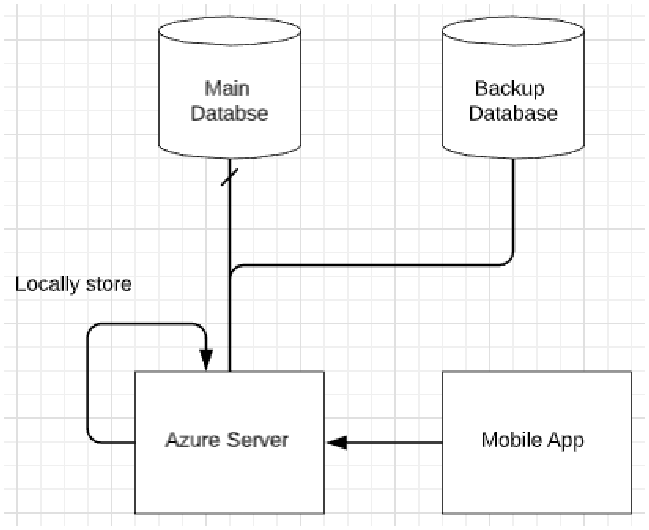
25a Hardware / Software Mapping



The main system design will implement the model view controller design pattern. Where the mobile application will be the view, the azure app service as the controller and MS SQL database as the model. This design pattern allows us to create scalable code that can be extended to other devices easily as needed and is modular allowing for each developer to work on different systems at the same time.

25b Persistent Data Management

All the data that is used throughout the application for all users is stored in the main SQL database. When the Azure server alters the data from that main SQL database, then the server also updates the backup SQL database. This is done instead of backing up periodically because it avoids replicating everything in the database every time. In the event that the main database fails, then the backup is ready to be taken over in place of the main database since it has all the updated data ready to go. Furthermore, even that database can also fail; so, in that event the Azure server will keep a local copy of any new updates that should take place once any of the databases are back online. The server will also hold the necessary data for path creation since it's the main feature and also already has a copy of previously stored paths through the PathFactory class.



25c Access Control and Security

This application utilizes AES encryption in order to protect sensitive information. This is in order to reduce the potential for data to be intercepted and subsequently decoded. AES encryption utilizes a 256-bit hashing algorithm paired with a private key to encode and decode data. This private key is unique, as no separate two keys could be used to generate the same information. Furthermore, AES is a reliable encryption method, as brute force techniques with modern machines would take an unreasonable amount of time to crack. In addition, our security mechanism involves multiple encryption/decryption rounds to further obfuscate the data to potential attackers. Data is encrypted locally before being transmitted across the network.

25d Global Software Control

The software should be able to work globally. The control of the system does not change based on the location of the user but rather the role of the user.

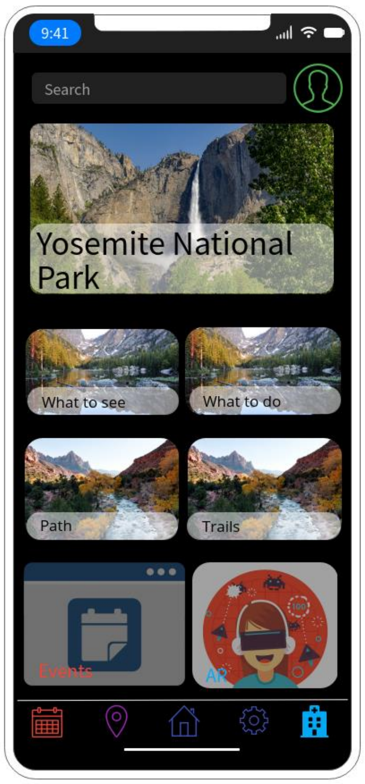
25e Boundary Conditions

The startup and shutdown of the backend and SQL system is not a major concern as the system will rarely be off. This is an abnormal occurrence. However, data creation between the backend and SQL database needs to be close to instant as this operation will happen relatively often compared to other boundary conditions and is considered a normal occurrence.

25f User Interface

User Interface (UI) design is a very integral process of application development. It is the process of making interfaces with a focus on looks and style. It is very important for the success of the application since it is what allows the user to communicate with the application. The user interface would follow general UI design principles that are having a consistent design, simple and clear interface, well labeled icons. It should also focus on color, layout, brightness and contrast providing users with seamless experience.

A sample UI design and wireframe are provided below:



The UI follows a consistent design and has options for every service provided. The bottom bar contains an emergency locator, map, home, calendar and settings for better accessibility.

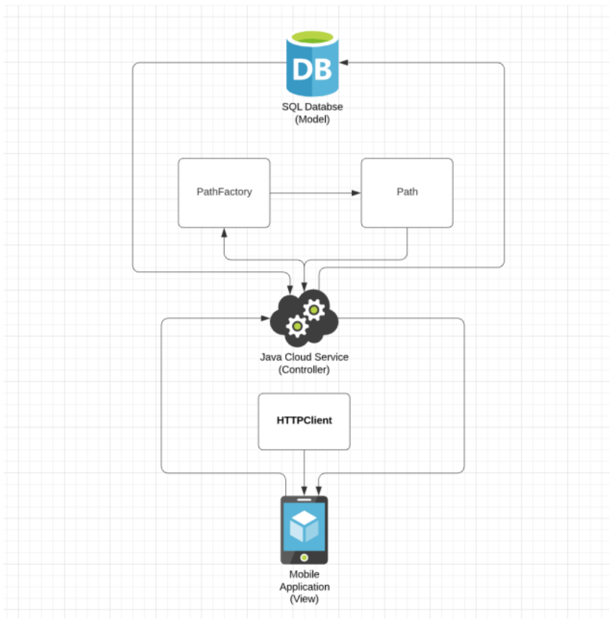
25g Application of Design Patterns

Model-View-Controller: Main design pattern that connects all the systems together. In the case of the SQL Database, it's where all the data is stored which acts as the Model. The Azure server will act on that data and manipulate it to the need of the users which acts as the Controller. Finally, the Mobile Application will be in charge of receiving that data from the Azure server and displaying it for the user to see which acts as the View.

Flyweight: Another major design pattern used in our system is the flyweight design pattern. This is a very important design pattern since it helps with reducing computation power and enforces the requirement that previously generated paths are stored for later retrieval. The PathFactory is the driver for Path retrieval since it stores a HashMap from PathDetails to a List of Paths. When the Azure server receives an incoming request for a path to be generated, the PathFactory will already have the HashMap populated from the SQL database and will check if the list of Paths matching the incoming details has been generated before. If it has, then the system returns the stored list to save computational power, else then generate the list of paths, store them, and return them to the user.

Singleton: A minor, yet equally important, design pattern that we use in the Azure server is the Singleton design pattern. This pattern is used in the PathFactory class for the main reason that there should be only one PathFactory instance. The main reason that the PathFactory needs to be a single instance is because of the fact that previously computed paths are stored in the PathFactory; so, we must keep one set of paths to save computing power, storage, and reduce redundancy.

26 Final System Design



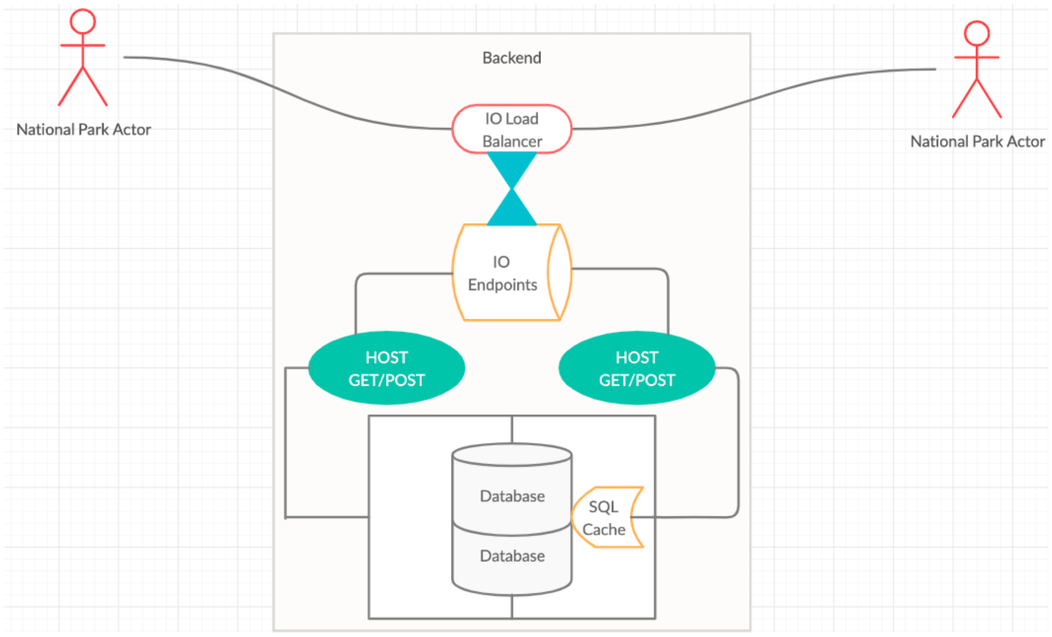
27 Object Design

27a Packages

Each component and class stays contained within each subsystem. Therefore, there is no need to create and package.

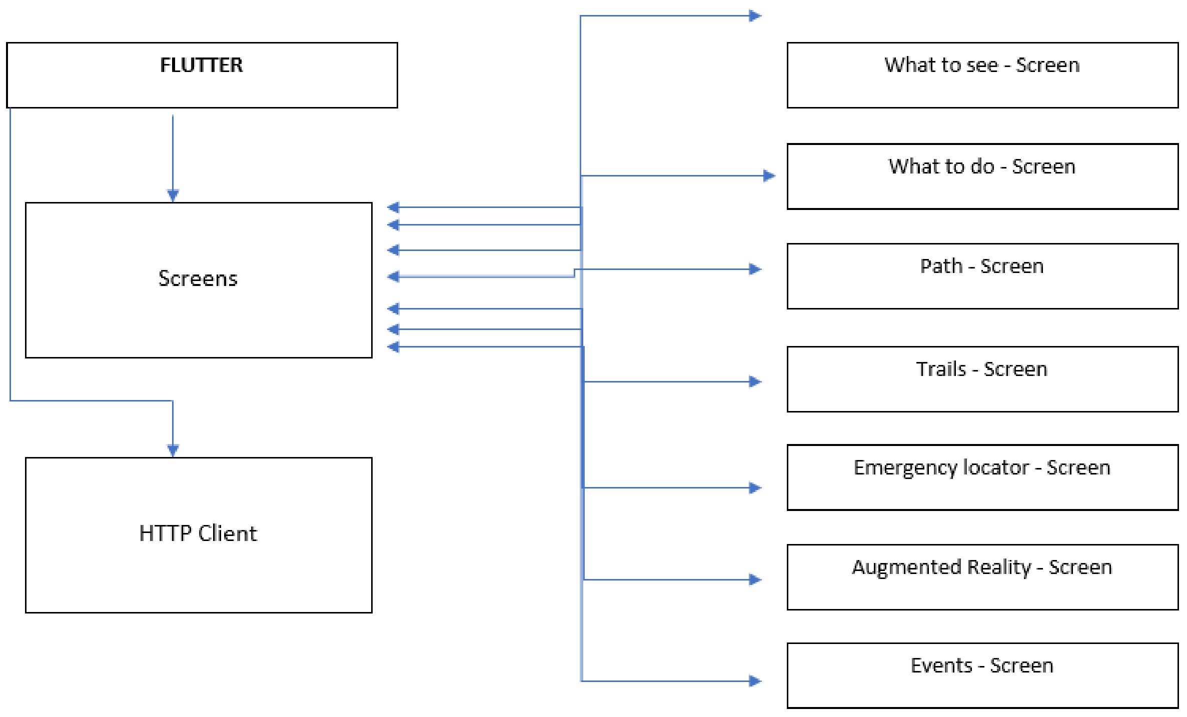
27b Subsystem I

The backend system is being built on Azure. Requests will first hit a load balancer which will direct requests to servers with less traffic. It will then hit the server with the HTTP request. If the SQL database needs to be queried it will first check the SQL cache to see if the data is readily available. This will reduce load against our SQL database. If the data is not there it will then query the database itself and return the results back as a HTTP message.



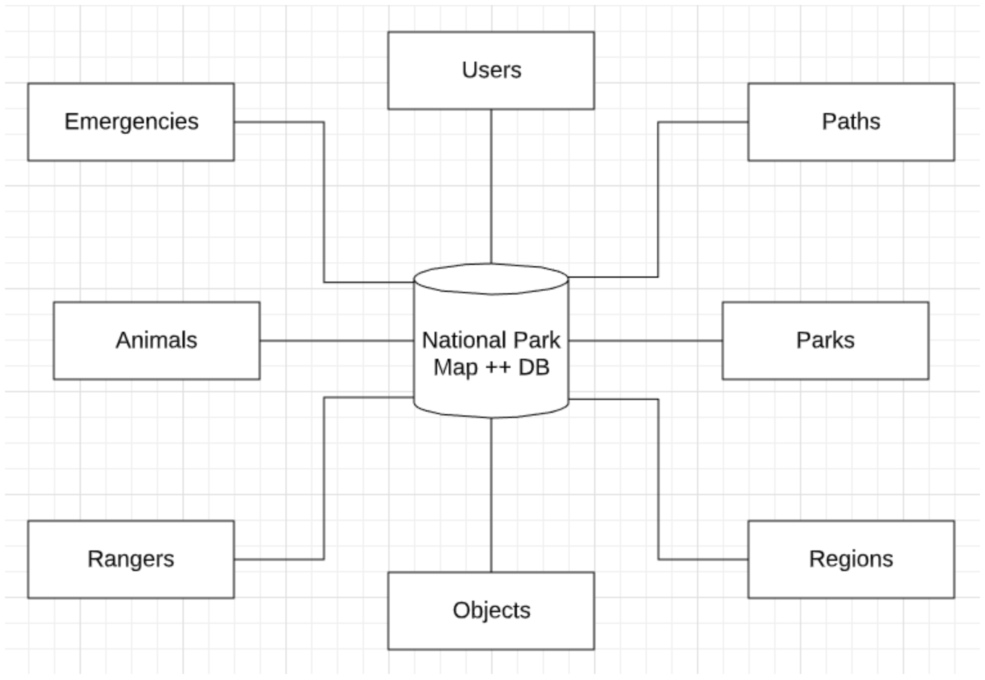
27c Subsystem II

The frontend can be built using various tools that can help achieve these requirements. Below is a diagram of a simple flutter application.



27d Subsystem III

The SQL Database design in terms of the tables to be implemented in the final product. These tables are also included in the backup database in case of an emergency



IV Project Issues

28 Open Issues

As the growing number of users constantly use the application, the single Azure server can become overloaded with requests that can cause a system failure. The case where the Azure server fails, the application has no sort of functionality since it primarily depends on the data received from the Azure server. So, the main issue is the fact that the Mobile Application will have no connection to the backend or the database.

Another major issue is the battery consumption of the application. Map and navigation applications are notorious for consuming the battery life of nearly every smartphone, whether they are old or brand new. The application should be able to handle battery consumption efficiently, so the users don’t suffer from low battery issues, slower phones, very hot phones, and to overall give them a good experience while using the application.

Finally, another open issue is the safety of the application while using the object selection features. When exploring the national park, there are lots of people looking only at their phones to navigate. The safety concern in mind is when explorers, like little children, wander off to areas that aren’t safe to be in just so they can get more information about an object on their screen. This is definitely a major safety issue that must be warned about when using the application.

29 Off-the-Shelf Solutions

There are several products in mind that help with the development of the new solution. Mainly, Google Maps API will be used for the map portion of the application. If the algorithm chosen for the path generation is a reduction to the well-known CNF SAT NP-Hard problem, then another library that can be used is a high-performance SAT Solver such as Z3-Prover [1]. The Flutter will also incorporate libraries for the HTTP Client and statistical graphs as part of the application UI.

29a Ready-Made Products

Google Maps API: The main ready-made product being used for the application is the Google Maps API since a primary feature of the application is its map. This is a very clean product that can be manipulated and customized to the need of the application. This also helps with the path direction for a given list of requested paths so the user can have the freedom to choose within a clean and responsive map UI.

29b Reusable Components

Z3-Prover: If the algorithm designed for the path generation from the starting point to the ending point is reduced to CNF SAT (it's possible and straight forward), then the library to use is the high-performance SAT Solver Z3 [1]. This solver is easy to use and very efficient to solve the problem (find the paths).

Dio: HTTP Client library available for the Flutter frontend called Dio [2]. This client is very simple to use and provides a wide range of configurations. Some of the useful configurations ready to be setup are the timeouts for connecting to the server or for retrieving data and the base URL for specifying the prefix URL in each HTTP request.

Syncfusion Flutter Charts: Statistics library for Flutter to display all the path information for completed paths called Syncfusion Flutter Charts [3]. This library provides clean and responsive graph widgets for displaying information. When a path has been completed by a user, then the Mobile Application will display the speeds throughout the interval of distances throughout the path.

30 New Problems

30a Effects on the Current Environment

With the use of this application park rangers will easily be able to receive requests from tourists if they are in need of any help and be able to act quicker than ever before. This creates more safety to the tourists. It also allows park rangers to do their job even easier.

The wildlife will be better protected because of this app. The app navigates tourists to avoid animals and other wildlife that shouldn't be provoked. This allows not only the safety of the tourists but also the wildlife within the park.

30b Effects on the Installed Systems

Since users might already have outdated systems the application can be built for all these outdated devices and should try to make the applications keeping the current modern devices in mind, so that the application would require minimal changes while migrating to new systems during updates.

30c Potential User Problems

The main user problem that arises from the application is the fact that it can drain the phone's battery life through extensive usage of the application. This will cause their phones to lose power at a much faster rate than normal and also cause their phones to become overheated as a secondary result.

Another user problem that can come from the use of the application is the fact that there has to constantly be an active internet connection to operate as intended. If the user doesn't have data, then the application won't be able to connect to the Azure server. Likewise, if the user has a limited data plan, then they will be consuming their data at a much faster rate than normal.

30d Limitations in the Anticipated Implementation Environment that May Inhibit the New Product

There are various issues that can significantly affect the functioning of the application like animal sightings, weather and various other incidents which can't be predicted by the application. The user would have to use other third-party applications for information on these incidents.

30e Follow-Up Problems

The application needs to provide special assistance to people with disabilities, and for those users the app can have a dictation feature letting users know about options and help navigate the app. The user will already be provided with directions through the map since it also provides audio instructions that could help the users.

31 Migration to the New Product

There are no previous products that must be migrated from for this new product.

32 Risks

One major risk of this project is the incapability for the Azure server and SQL database not being able to handle the hundreds of thousands of simultaneous user connections. This is the biggest downfall since the only focus group the product has in mind are national park tourists. So, if the servers can't handle the overloaded connections, then the product won't perform efficiently and not give a good user experience.

Another major risk is the possibility of having inadequate engineers writing the path generation algorithm. If the team is unable to create an efficient algorithm for path generation, then the product will take too long to finish computing the paths and provide a bad user experience. If this is the case, then there will need to be a reallocation of funds to hire a professional consultant in this area or even faster servers that will hopefully make up the inefficiency of the algorithm in computational power.

33 Costs

The primary operating costs of this application will derive from the cloud-hosting services. Microsoft provides an online calculator tool to determine the monthly costs of certain network configurations. As of 2020, our configuration calls for roughly $7,000 in monthly server and maintenance costs.

34 Waiting Room

Content:
Users will be able to acquire points for including new information about the park.

Motivation:

Encourages users to continue to add new information about the park. This creates a better Experience for other users as there is more information to be found about each park.

Consideration:
This particular feature is relatively easy to implement, and the benefit can be vast. It should be given high priority in the next release.

35 Ideas for Solutions

Content:
A potential solution to creating a reliable backend is to ensure it can handle many users. In order to do this, the developers must take full advantage of the computing resources. The use of multi-threaded code is necessary.

Motivation:
Multithreaded code allows for more users to be on the service at once which reduces the risk of downtime of the server because it cannot handle so many users.

Consideration:
This can be easily achieved by writing immutable code and avoiding side effects in the backend. Since immutable code is inherently thread safe this allows easier implementation

36 Project Retrospective

Overall the main purpose of National Park Map ++ is to provide the visitors of national parks with a memorable and adventurous experience, but at the same time not compromising with their security. The application would provide the visitors with different paths, emergency information and various itineraries for different national parks. It will also implement and provide an augmented reality experience to help navigate the national park. Since National Park ++ doesn’t require any personal information other name and basic details, it will be significantly saving costs on security issues and pay more attention to the app development.

V Glossary

MVC: MVC stands for Model, View, and Controller. MVC separates an application into three components - Model, View, and Controller. Model represents the data; View is the User Interface and Controller is the request handler.

DDoS: A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic.

Load Balancer: A device that adequately distributes traffic across a network to prevent overwhelming of any particular zone.

HTTP: Hyper-Test Transfer Protocol, a global standard in web processing.

CNF SAT: A type of problem which determines if there exists an implementation that satisfies a specific Boolean formula.

Azure Server: It's a public cloud computing platform used as a server for many different applications

Flutter: It is an open source mobile UI framework created by Google and released in May 2017.

SQL: Structured query language, commonly used in programming to manipulate data stored in relational databases.

AES: AES has been adopted by the U.S. government and is now used worldwide. It supersedes the Data Encryption Standard (DES), which was published in 1977. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data[4].

API: (Application Programming Interface) defines interactions between multiple software services.

VI References / Bibliography

[1] "An Efficient Theorem Prover." Z3, z3prover.github.io/api/html/index.html.

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

M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[2] "Dio." Dio - Dart API Docs, pub.dev/documentation/dio/latest/.

[3] “Overview.” Overview of Syncfusion Flutter Charts, help.syncfusion.com/flutter/chart/overview.

[4] “Advanced Encryption Standard.” Wikipedia, Wikimedia Foundation, 2 May 2020, en.wikipedia.org/wiki/Advanced_Encryption_Standard.

VII Index

A

Acceptance Tests 4, 30, 49, 50, 51, 53, 54
Accuracy 33
AES 70, 81
Algorithm 13, 14, 17, 67, 70, 77, 79, 81
Augmented Reality 6, 16, 80
Azure 53, 66, 70, 72, 73, 76, 78, 79, 81

B

Backend 17, 28, 29, 30, 59, 67, 68, 76, 80
Budget 16, 51

C

Capacity 3, 34
Client 2, 12, 77
Collaborative 2, 14
Connection 6, 15, 22, 37, 60, 76, 78
Constraints 2, 13, 14, 15
Customer 2, 12

D

Database 6, 7, 15, 30, 51, 64, 70, 73, 79
Dio 77, 81

E

Entity-Relationship Diagram 16

F

Flutter 15, 74, 82
Flyweight 67, 69, 72
Frontend 66
Functional Requirements 3, 30

G

Google Maps API 77

H

HTTP 66, 73, 77, 81

L

Land Life Station 10
Load Balancer 37, 73
Location Services 6, 10, 12, 30

M

Model View Controller 70, 72

P

Parameters 7
PathCoordinate 68
PathDetails 67, 69, 72
PathFactory 67, 70, 72, 73
Paths 16, 21, 30, 58, 64, 70, 77, 79, 80
Plant Life Station 10
Precision 3, 33

R

Rationale 30, 40, 47, 48, 49, 51, 52, 53
Remote 6, 36

S

Scenario 2, 11, 12
Sea Life Station 10
Singleton 67, 73
SQL 67, 68, 70, 71, 72, 73, 75, 79, 81

Stakeholders 2, 12, 13
Syncfusion Flutter Charts 77, 82
System Design 64, 70

U

UML 2, 16, 81
Use Cases 3, 19
User Interface 4, 71, 80

W

Weather station 10

Z

Z3-Prover 77