Ecolocation

Software Design

Version 2.0

February 22, 2018

Client/Mentor:
Dr. Chris Doughty

Team Members:
Brenden Bernal
Chandler Hayes
Michael Hartzell
Anthony De La Torre

Table of Contents

Introduction

Animals play an indispensable role in providing for the overall health of our ecosystems. Of all the animals that play a role in this, megafauna (large mammals) play a disproportionately more significant role due to their size, range of travel, and longer food passage times. Due to the more substantial impact of megafauna, our project places its focus on these large mammals and seeks to provide information on the effects they have in a specified area specifically mammals greater than 10kg. Our client and mentor Dr. Chris Doughty has been researching in this area, and we are working with him to make this information more accessible and understandable.

Although this information is accessible for existing megafauna, it is not readily available or formatted in a way that is easy to interpret. We are creating a unique mobile application that seeks to bring this information together and explain the data in ways that will educate the user. Furthermore, we will be providing information on megafauna going back thousands of years; mammals that are extinct. This is a unique aspect of our application as range maps and nutrient distribution information on extinct megafauna is not publically available. We will be combining information on current and historic megafauna to provide unique data such as the effect that an extinct mammal could have on our ecosystems should it return.

Users will download the application from the Google Play Store with a phone running Android 2.2 or higher. The users will use the application with an internet connection and will provide a location, either by using the phone's location or specific, user defined, location. These user requirements lead to the application's functional requirements. The application will use the given location to receive information on all mammals that are more than 10kg at the provided location. This information will then be used to calculate the nutrient distribution for all mammals at the location, then generate graphs to display the info. The application will also generate a list to show the user all the mammals in the given location. This list will show an image of the animal (retrieved from a cloud based database) along with both the scientific name and common name for each animal, and the list will be color coded to represent the animals threat level. The application will allow the user to see more information on any animal on the list by clicking on it. The information will include a short description, mass, population, endangered level, and a link to a Wikipedia article for more information.

The purpose of this document is to explain the software design of the application. The first part will include an overview of the application. Then we will describe the general tools that will be used to make the application. Next, an overview of the application architecture will be discussed. Following the architecture, we will discuss the modules for both the front-end and back-end of the app in detail. The final part of this document will be exploring the plans for development going forward.

Implementation Overview

Our approach is to create an Android mobile app that graphically displays how animals impact any given ecosystem globally. It will focus on nutrient distribution (how much nutrients are circulated throughout an ecosystem) to demonstrate the effects animals collectively produce. In later phases, other features will be added such as displaying other ecosystem services (like seed dispersal), providing information on marine animals, and enabling exploration of different time periods. The app will compare animals currently existing in the ecosystem and animals it had during the Pleistocene Era (an era without human impact) to emphasize how larger animals impact ecosystems more and the changes to ecosystems since the Pleistocene Era.

The app will require a set of coordinates for a location and will generate the current animals and Pleistocene Era animals at those coordinates. The effects of these species will be demonstrated by displaying three types of graphs to represent nutrient distribution. These graphs will enable the user to compare differences between current and Pleistocene Era at a local and global scale. The user can learn more information about specific animals and alter the current ecosystem to see how changes can affect the ecosystem.

Implementing this application will require the use of different technologies. The overall project will be written in Android Studio. This allows the app to be built natively for Android devices. The performance of a native app is superior to a cross-platform; a cross-platform application has slower performance because it is a web app wrapped inside of a web browser. Additionally the amount of data it needs to process through and the number of calculations will require the performance of a native app.

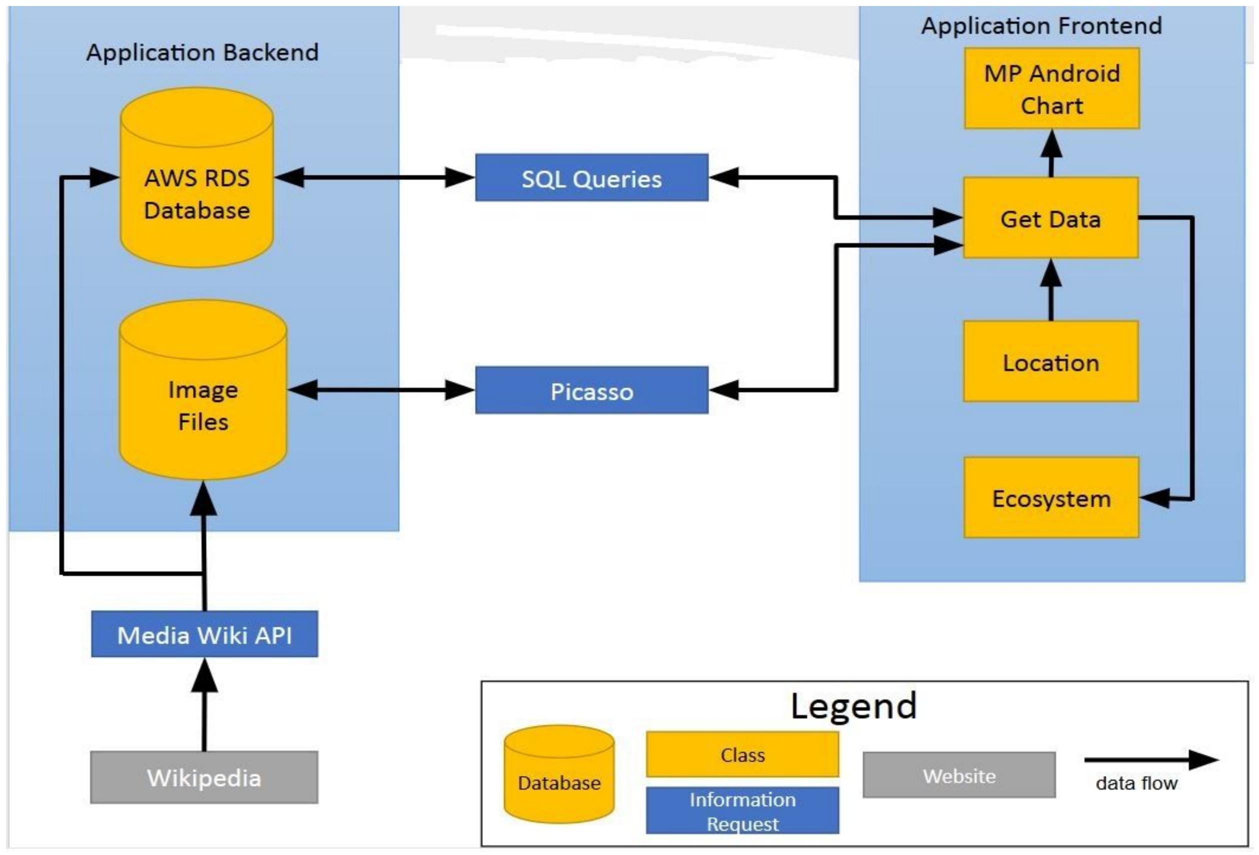
The data on all the current animals and Pleistocene Era animals are from different sources. The data will be copied and transferred to Amazon Web Services (AWS). Having all of the data on the animals in one place simplifies data retrieval. In later phases, more sources on different types of animals (like marine animals) will be added. Additionally, storing this data on the device would be impractical; data on modern and Pleistocene Era animals is roughly 2GB of data.

In addition to the data on the animals, we want to provide images of each species if possible. To be able to access these images, they need to be hosted on a website. This will reduce the amount of space that will be used on AWS and keeps the prices for AWS as low as possible. We will be using Picasso to retrieve the images and display them on the application quickly.

Google Play services will be used for Google Maps API and Google Location API because the information shown for an ecosystem is location dependent. Google Play services is typically used instead of Android’s framework because Google’s framework provides better performance and uses less power. These two frameworks (Google Maps API and Google Location API) will allow the user to either pick their current location or selecting a custom location by choosing a spot on the map or entering coordinates.

Architectural Overview

Our Android application consists of two main components, the backend and the frontend. The figure below shows the layout of each component's key pieces, and the directions that data flows between them.



Application Backend- The backend of our application consists of information stored in two separate locations. The text based information is stored using Amazon Web Services(AWS), and the images are stored as URLs on a website.

- **AWS RDS Database-** The original information is provided by the International Union for the Conservation of Nature (IUCN) and a historic database, supplemented with information from Wikipedia, using the Media Wiki API. This information includes their scientific name, common name, mass, and a brief description. It was then converted into a Relational Database and stored in the cloud using AWS.
- **Image Files-** Our plan is that every animal in the database has an image, but some may be unavailable. The available images were gathered from

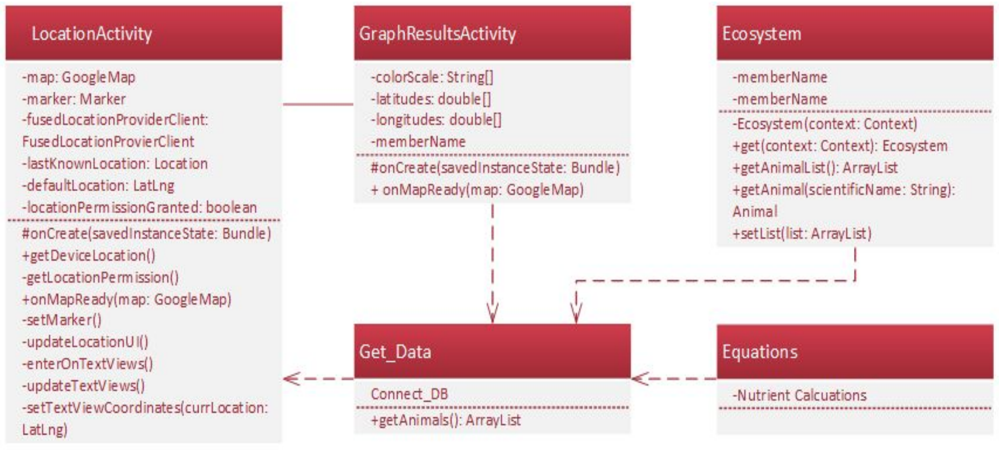
Wikipedia using the Media Wiki API and are stored as jpeg files on a private website where each image has a unique URL.

    - **Media Wiki Api-** A package is available in most programming languages that allow for searching and information gathering of Wikipedia's articles.

Application Frontend- The front-end consists of a mobile application created with Android Studio. The object oriented nature of Android Studio allows for all of the classes in the frontend to easily communicate with one another while dividing up tasks into several classes.

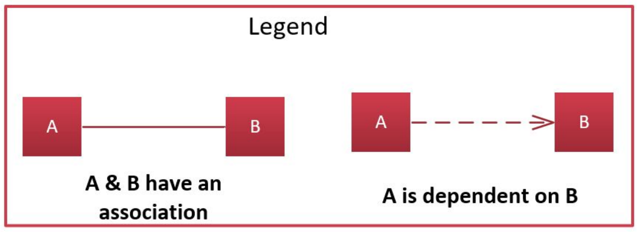
- **LocationActivity-** This class uses the Google Maps API to give users a way to choose different locations. By default, it displays a map of the user's current location. The user has two options: use their current location or use a custom location. To choose a custom location, the user can enter coordinates into an input box or move the marker on the map. The coordinates then sent to the GetData class.
- **GetData-** The coordinates selected by the user are then used in SQL queries to access the data stored in the AWS database. For each animal in the given location, the database returns their information. After the animal information is returned, the masses are used in the mass based scaling equations, and the results are sent to MP Android Chart. Picasso is then used to get the images of each animal. The images and the database information are then sent to the Ecosystem class.
    - **SQL Queries-** Structured queries used to access data stored in a relational database.
    - **Picasso-** An Android Studio package designed to display images from a given URL.
- **MP Android Chart-** The information from Get Data is then used to create a bar graph with three different bars. These are ecosystems impact on the given service in the present, past, and user defined scenarios.
- **Ecosystem-** The animal information from Get Data is then used to create a list of every animal in the given ecosystem along with their pictures. Users can select an animal to view more information on them. The list can also be sorted by things like endangered level and mass.

Module and Interface Descriptions


LocationActivity
-map: GoogleMap
-marker: Marker
-fusedLocationProviderClient: FusedLocationProvierClient
-lastKnownLocation: Location
-defaultLocation: LatLng
-locationPermissionGranted: boolean
#onCreate(savedInstanceState: Bundle)
+getDeviceLocation()
-getLocationPermission()
+onMapReady(map: GoogleMap)
-setMarker()
-updateLocationUI()
-enterOnTextViews()
-updateTextViews()
-setTextViewCoordinates(currLocation: LatLng)
GraphResultsActivity
-colorScale: String[]
-latitudes: double[]
-longitudes: double[]
-memberName
#onCreate(savedInstanceState: Bundle)
+ onMapReady(map: GoogleMap)
Ecosystem
-memberName
-memberName
-Ecosystem(context: Context)
+get(context: Context): Ecosystem
+getAnimalList(): ArrayList
+getAnimal(scientificName: String): Animal
+setList(list: ArrayList)
Get_Data
Connect_DB
+getAnimals(): ArrayList
Equations
-Nutrient Calcuations

Frontend

Location



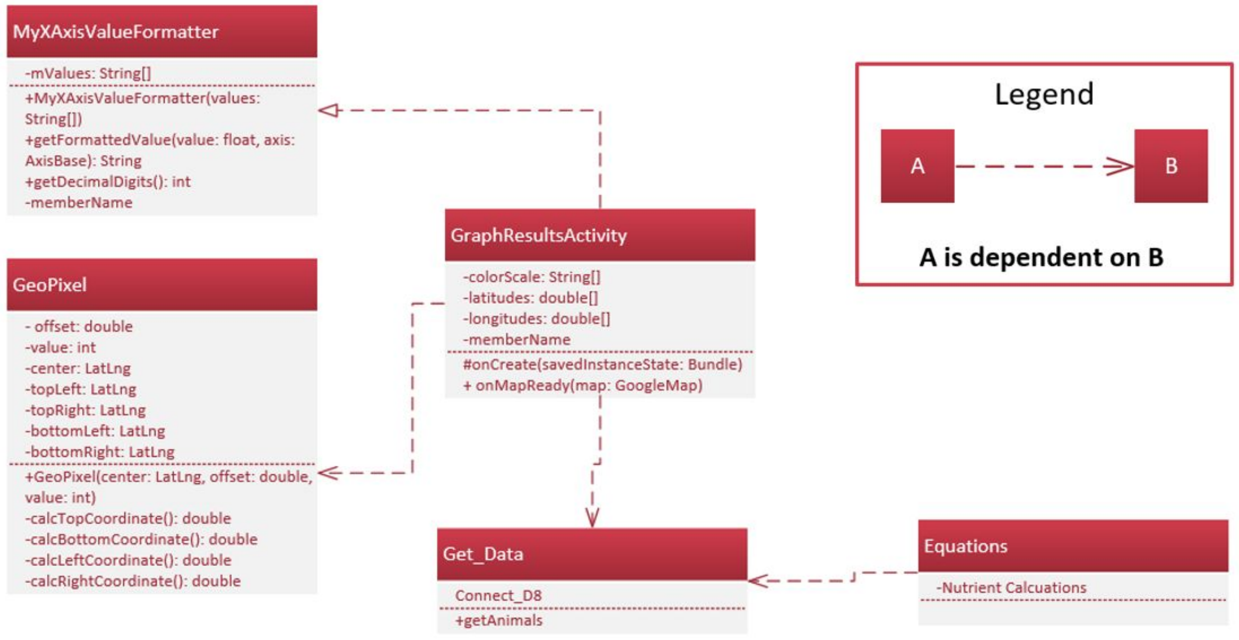
The GetData class is a central point of the application but the results of the LocationActivity class affects the outcome of the rest of the app. Once a location is determined, data can be retrieved from the database by the GetData class. Once the data has been retrieved, the nutrient distribution can be calculated through the equations class. The results of these two classes will be stored in the Ecosystem class so that it can be accessed from other activities that need this information.

This part of the application enables the calculations for the nutrient distribution for a specific ecosystem and in later phases calculating other ecosystem services. The user has two options for selecting a location; the user can select their current location or a custom location. To be able to obtain the user's location, the user must grant permission of using the location services on their phone. The LocationActivity screen will display a Google Map that is centered on the user's current location if they allow access to their location services. If the access is denied, it will show a default location. On the Google

Maps, the user will see a marker showing the currently selected screen and a textbox with location's coordinates. The user has two methods for selecting a custom location. The user can drag the marker to a new location or enter the coordinates.

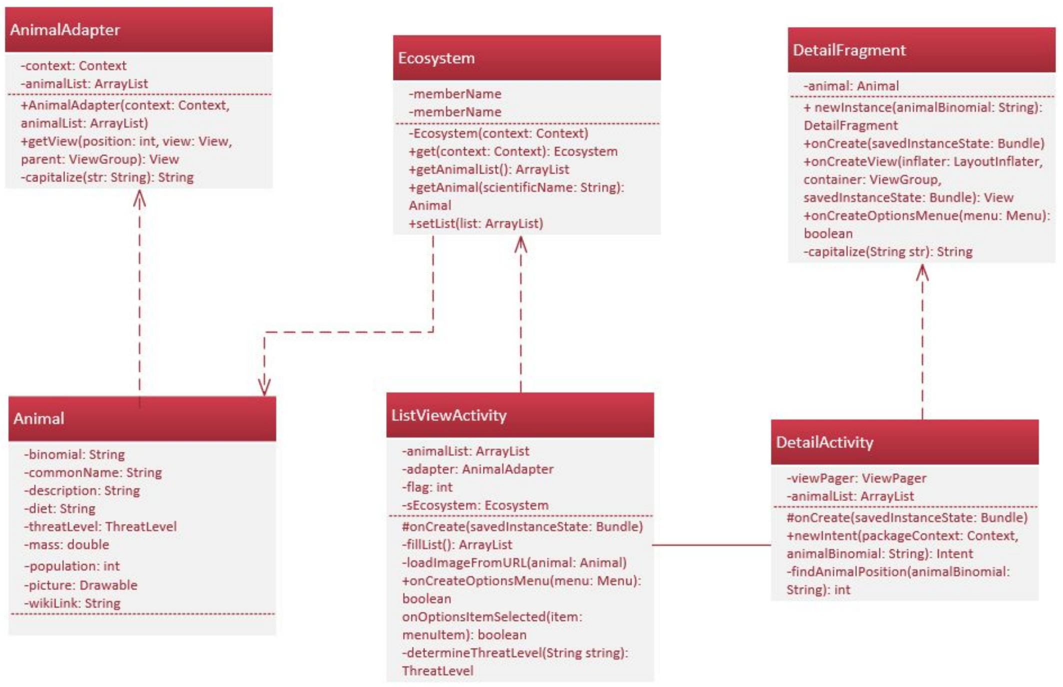
When the user first opens the app, there will be two separate dialog boxes. The first dialog box will ask the user for permission to use location services to get their location. The dialog box will ask the user if the app can turn on their location services if it is not already on. After the user selects a location, it will be sent to the Get_Data class. With the data retrieved from the Get_Data class, the results will be displayed in the GraphResultsActivity.

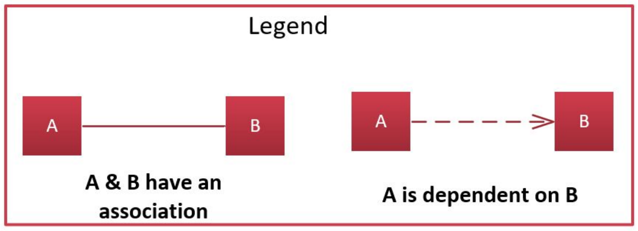
Graph Results



The graph result activity is responsible for generating several graphs for the application. The first is a bar chart. The Chart will show the nutrient distribution for two-contexts: the Pleistocene Era, current ecosystem, a hypothetical ecosystem for the given location.The GraphResultActivity uses MyAxisValueFormatter to format the bar chart. The method uses the Equation method to get the values for nutrient distribution to us for the bar charts. The second graph is a local map that shows the spatial mapping of the nutrient distribution. The method uses GeoPixel to generate. The third graph is a global map of the spatial mapping. The two graphs can show either the current ecosystem or the Pleistocene Era ecosystem.

Animal Details


AnimalAdapter
-context: Context
-animalList: ArrayList
+AnimalAdapter(context: Context, animalList: ArrayList)
+getView(position: int, view: View, parent: ViewGroup): View
-capitalize(str: String): String
Ecosystem
-memberName
-memberName
-Ecosystem(context: Context)
+get(context: Context): Ecosystem
+getAnimalList(): ArrayList
+getAnimal(scientificName: String): Animal
+setList(list: ArrayList)
DetailFragment
-animal: Animal
+ newInstance(animalBinomial: String): DetailFragment
+onCreate(savedInstanceState: Bundle)
+onCreateView(inflater: LayoutInflater, container: ViewGroup, savedInstanceState: Bundle): View
+onCreateOptionsMenue(menu: Menu): boolean
-capitalize(String str): String
Animal
-binomial: String
-commonName: String
-description: String
-diet: String
-threatLevel: ThreatLevel
-mass: double
-population: int
-picture: Drawable
-wikiLink: String
ListViewActivity
-animalList: ArrayList
-adapter: AnimalAdapter
-flag: int
-sEcosystem: Ecosystem
#onCreate(savedInstanceState: Bundle)
-fillList(): ArrayList
-loadImageFromURL(animal: Animal)
+onCreateOptionsMenu(menu: Menu): boolean
onOptionsItemSelected(item: menuItem): boolean
-determineThreatLevel(String string): ThreatLevel
DetailActivity
-viewPager: ViewPager
-animalList: ArrayList
#onCreate(savedInstanceState: Bundle)
+newIntent(packageContext: Context, animalBinomial: String): Intent
-findAnimalPosition(animalBinomial: String): int



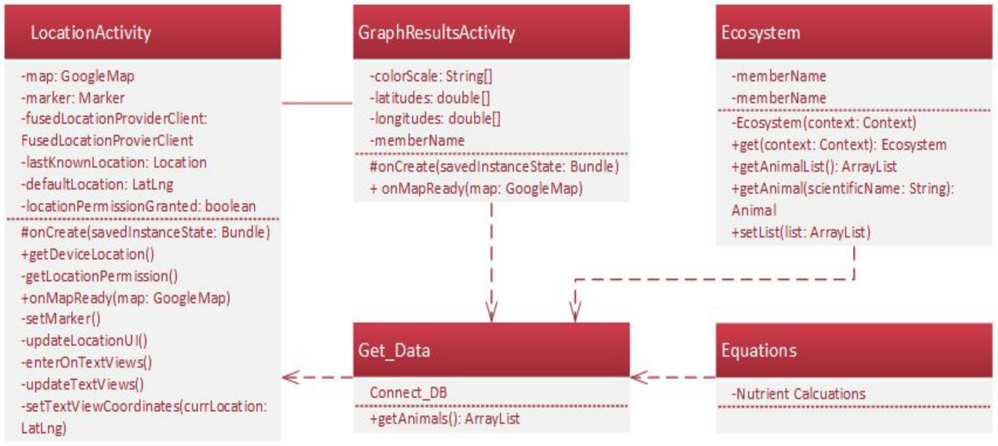
The Animal Details is a combination of the ListViewActivity, and the DetailActivity as both of these classes display information of the animals at the desired location. The ListView activity contains the high level description of all the animals in a list format. The information is pulled from the Ecosystem class which stores all the information from the database query seen in the Amazon Database section below. The Ecosystem class contains the specific information for each animal (name, endangered level, population, etc). Each animal at the chosen location is stored as an instance in the Ecosystem class so each animal instance contains all relevant information for itself. The animals are displayed on a scrollable page with a picture and endangered status. The ListViewActivity class also allows the user to sort the mammals based on certain characteristics such as size, name, and endangered level.

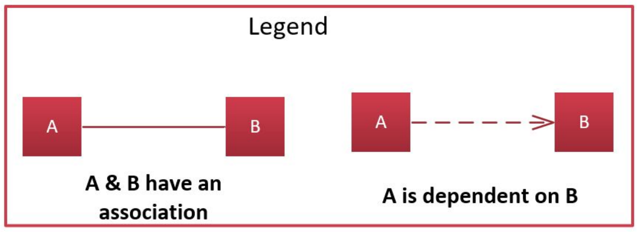
When any of the mammals are clicked, the DetailActivity is called. This activity contains more information on each mammal, including the Wikipedia description and a link to the

Wikipedia page as well as mass and diet. The DetailActivity class utilizes the DetailFragmant to contain the data and allow the screen to be swiped left and right to view the rest of the animals. Each animal is contained in an instance of the fragment which is populated with information from the ListViewActivity when called.

Backend

Amazon Database


LocationActivity
-map: GoogleMap
-marker: Marker
-fusedLocationProviderClient: FusedLocationProvierClient
-lastKnownLocation: Location
-defaultLocation: LatLng
-locationPermissionGranted: boolean
#onCreate(savedInstanceState: Bundle)
+getDeviceLocation()
-getLocationPermission()
+onMapReady(map: GoogleMap)
-setMarker()
-updateLocationUI()
-enterOnTextViews()
-updateTextViews()
-setTextViewCoordinates(currLocation: LatLng)
GraphResultsActivity
-colorScale: String[]
-latitudes: double[]
-longitudes: double[]
-memberName
#onCreate(savedInstanceState: Bundle)
+ onMapReady(map: GoogleMap)
Ecosystem
-memberName
-memberName
-Ecosystem(context: Context)
+get(context: Context): Ecosystem
+getAnimalList(): ArrayList
+getAnimal(scientificName: String): Animal
+setList(list: ArrayList)
Get_Data
Connect_DB
+getAnimals(): ArrayList
Equations
-Nutrient Calcuations

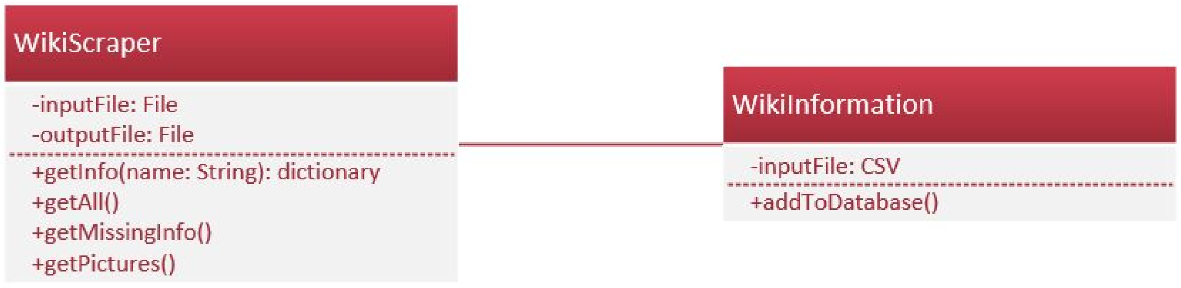


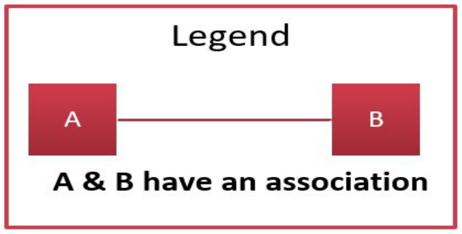
The database is accessed after the user chooses the desired location and the workflow is as follows:

- Coordinates sent to Get_Data class
- Get_Data class inserts coordinates into query and connects to the Amazon database through the Connect_DB function
- The results are sent to the Equations class where the nutrient calculations are performed
- Results from equations are sent to the GraphResultsActivity where graphs are propagated and displayed
- The results from the Database are stored in the Ecosystem class
    - Each animal is a unique instance of the Ecosystem class

The Ecosystem class holds all the data on the animals in the form of an instance of the Animal class. The Animal class is a custom object that contains all the necessary information for each animal. This occurs alongside the calculations after the Get_Data class gets the results from the Amazon database.

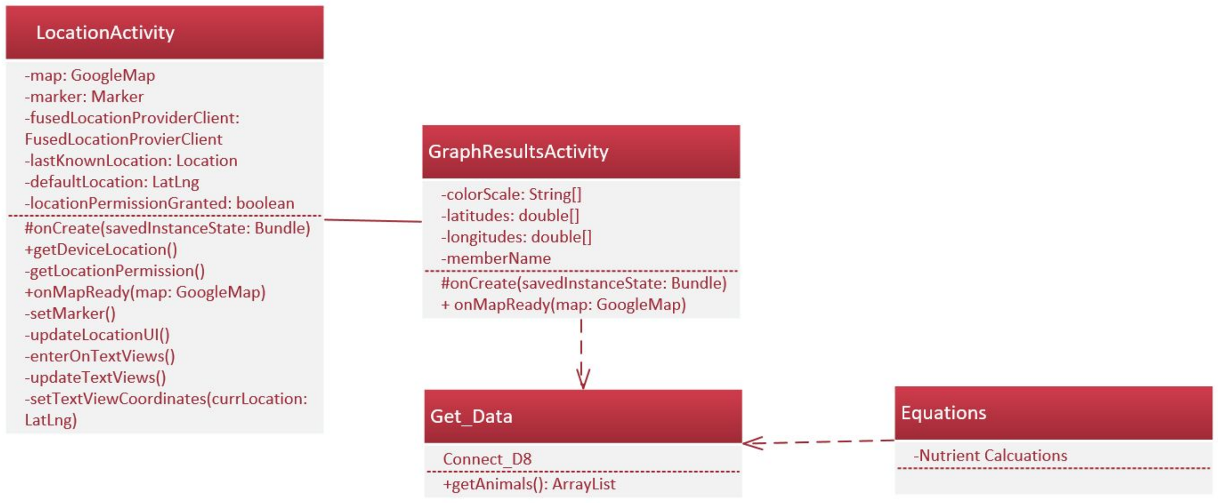
Wikipedia Information

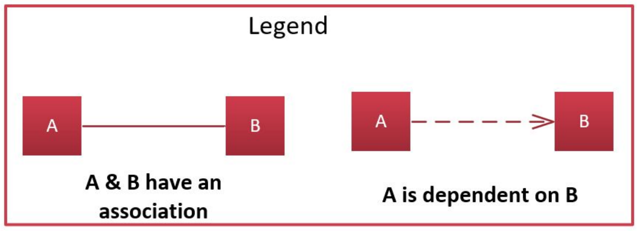




The information gathered from Wikipedia includes an image, their common name(if available), and a brief description. This information was accessed using a single Python class utilizing the Media Wiki API. The names, description, and link were saved to a CSV file, and the images were saved as jpegs. The data from the CSV file will be placed in the database using the WikiInformation class, and the images will be stored on a private website.

Calculations





The Equation class will calculate the nutrient distribution. The LocationActivity will send the user's given a location to Get_Data which will query the AWS database which will send back information on the mammals in the given location. The Equation class will use the mass of the mammals in the given location to calculate the nutrient distribution. The Equation class will then sends its calculations to the GraphResutsActivity which will display the results of the calculations. The Equation class will be called every time the users wishes to manipulate the ecosystem.

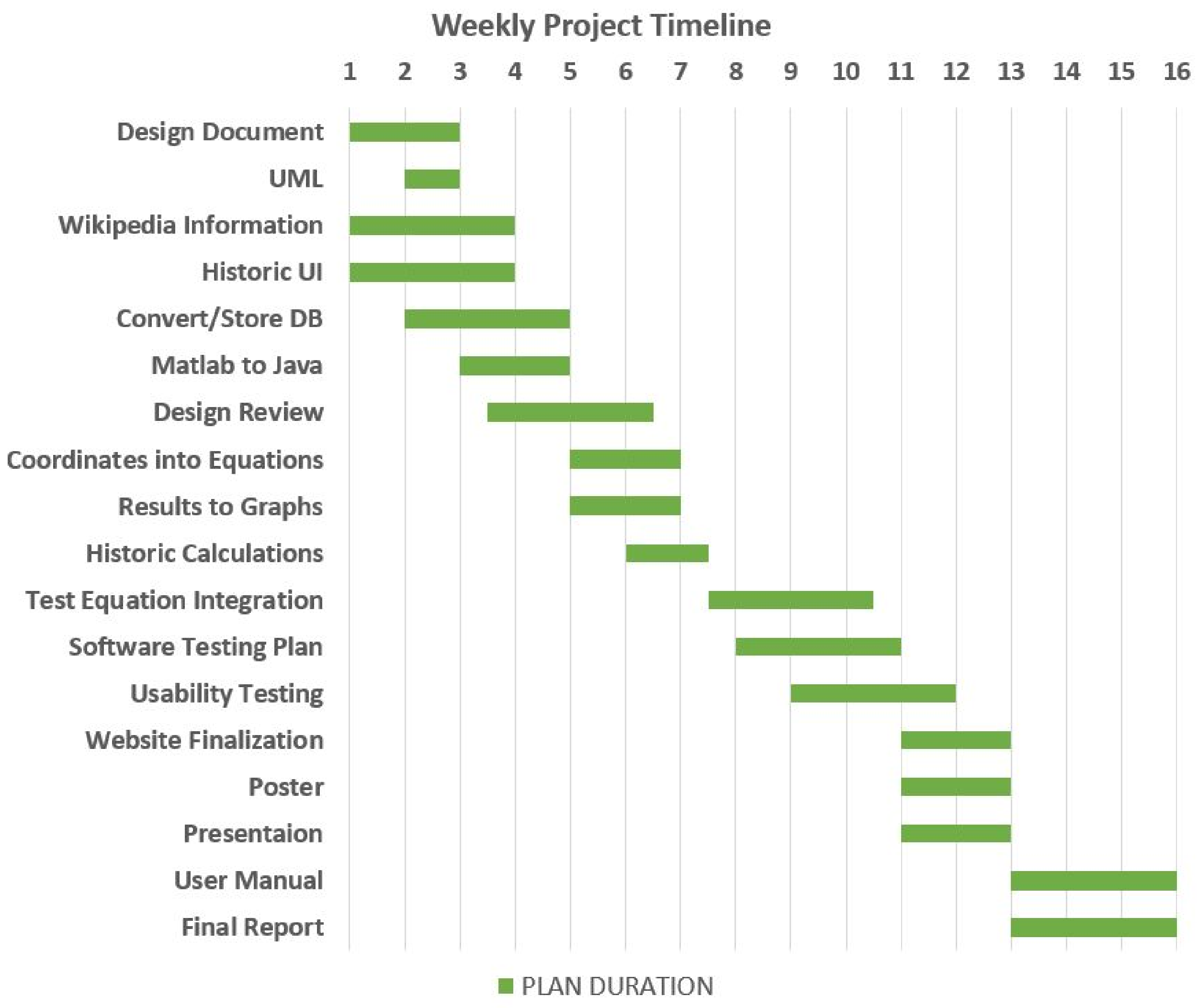
Implementation Plan

Our application has been broken into steps to divide the work better and to split the project up into manageable segments. At the start of the semester, we are going to begin by converting and storing all of the necessary data for the project. All of the information for the mammals must be retrieved from Wikipedia, and the data and photos must be stored. The spatial data must be converted to a format that is usable with Android Studio and is quick to access. Alongside the data gathering, we need to convert the calculations from MATLAB code to some format compatible with Android Studio. We will be testing multiple methods for doing this, including line by line conversion or using a program to convert MATLAB to python. Whichever method produces the best results will be the one we ultimately use.

Once we have these major steps accomplished the integration begins. Our application must integrate the location, calculations, graphs, and data together. Each of these steps will be done separately by a different team member, and once these steps are accomplished, they will be combined for the final integration phase. Once all the parts are combined, we must thoroughly test the code to make sure the calculations are correct. During the testing phase, we will also integrate the historic code as we know the parts are working as they should.

After the code is integrated successfully the usability testing begins. This will involve multiple testers with varying levels of technical skill using our application and trying to accomplish specific tasks we design for them while we record the their progress. We will be able to use the recordings to identify any sections of our application that gave the users problems. This can help us identify confusing instructions or unclear layout of the application. We will also get feedback from the user that will help us ensure a smoother user experience. While the testing is going on, we will continually improve the **User Interface** and the **User Experience** along with fixing any bugs. After thorough testing and modification, we should arrive at the final product in time for the capstone presentation.

Gantt Chart


Weekly Project Timeline
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Design Document
UML
Wikipedia Information
Historic UI
Convert/Store DB
Matlab to Java
Design Review
Coordinates into Equations
Results to Graphs
Historic Calculations
Test Equation Integration
Software Testing Plan
Usability Testing
Website Finalization
Poster
Presentaion
User Manual
Final Report
🟩 PLAN DURATION

Conclusion

Learning about the impact animals have on the overall health of their ecosystems is a difficult process. By making a mobile app that simplifies this process we can enable a wide array of people to understand the crucial role animals have in ecosystems. The app will enable users to explore different ecosystems over the planet. It will extract data from multiple sources and ecological equations to demonstrate how animals affect their ecosystems. Comparisons between current ecosystems and Pleistocene Era (time before human impact) ecosystems will be shown to emphasize this point. Furthermore, the application will allow users to learn more about specific species.

Module Summary

The application has two main components: the front-end and the back-end:

The **front-end** consists of obtaining a location the user wants to learn about through the use of Google Maps and Google Location APIs, displaying the nutrient distribution on different graphs, and calculating the information that needs to be retrieved for a specific location. The front end has three main components:

- Location
- Graph Results
- Animal Details

The **back-end** contains getting and storing all the data on each animal. Most of the data will be from the IUCN (International Union for Conservation of Nature) database and a private historic database. Additional information such as a description and picture will be taken from Wikipedia.This data will be stored in one database to allow the user to obtain this information from one source

- Amazon Database
- Wikipedia Information
- Calculations

References

Anon. 21AD. A beginner's guide to Visio. (21AD). Retrieved February 21, 2018 from https://support.office.com/en-us/article/a-beginner-s-guide-to-visio-bc1605de-d9f3-4c3a-970c-19876386047c?ui=en-US&rs=en-US&ad=US