|  |
| --- |
| CS123- A |
| MapBook |
| CS 123 Final Project |
|  |
| **Christoffer Kho, Nitesh Purswani, Miguel Antonio Ramos, Jose Enrico Tiongson** |
| **12/10/2015** |

|  |
| --- |
|  |

Contents

[I. Context 2](#_Toc437438069)

[II. Team Organization 3](#_Toc437438070)

[III. Sprint Plans 4](#_Toc437438071)

[IMPLEMENTED MINIMUM VIABLE PRODUCT 4](#_Toc437438072)

[UNIMPLEMENTED FEATURES 6](#_Toc437438073)

[IV. Activity Network Diagram 7](#_Toc437438074)

[Dependencies: 8](#_Toc437438075)

[V. Enhanced Relatonal Diagram 11](#_Toc437438076)

[VI. Data Dictionary 12](#_Toc437438077)

[VII. Use Case Diagram 13](#_Toc437438078)

[VIII. Use Case Description 14](#_Toc437438079)

1. Context

Problem Tackled:

Keeping track of landmarks when travelling can’t be immediately done through apps like Google Maps when an internet connection is not available. This is particularly true when travelling to less urban areas, particularly in some provinces in the Philippines and on natural sites such as rivers, mountains and beaches. For example, particular landmarks on the trails on mountains such as trees or waterfalls may not necessarily be available on Google Maps. By using this app, these landmarks can be saved while on the trail, and reviewed on a Google Map once an internet connection is available, in addition to tracked GPS points being sent to OpenStreetMap’s Public GPS Traces. This may be used for future reference when future climbs on that mountain are planned.

This app will make use of GPS, Internet and Google services in Android.

The basic function of the app will be as follows:

1. The user inputs basic information about a location (a Name, an Address and Notes).
2. GPS coordinates of the location is saved when a button is pressed. Alternatively, a path can be made, tracking multiple GPS coordinate as waypoints in a trail.
3. The coordinates and user-inputted information are saved as one row in a table using Android’s built-in SQLite.
4. The user can view previously saved locations through a list. If there is an internet connection available, then the information can be viewed in a Google map.

5) Saved paths can be sent to OSM’s Pubic GPS Traces for others’ use.

6) Locations and paths may be saved in a Parse database for other app users to view.

1. Team Organization

Team Organization Selected: Democratic Team Approach

Rational:

* 1. The members of the groups are, put simply, friends already.
  2. The group works best through delegation, rather than by following a leader.

The group was organized by splitting front-end and back-end tasks in parallel.

ORGANIZATIONAL CHART/ROLE OF EACH MEMBER:

Communication was done via Facebook and personal meetings. A GitHub repository was used to handle work distribution and maintain updated versions of the project.

1. Sprint Plans

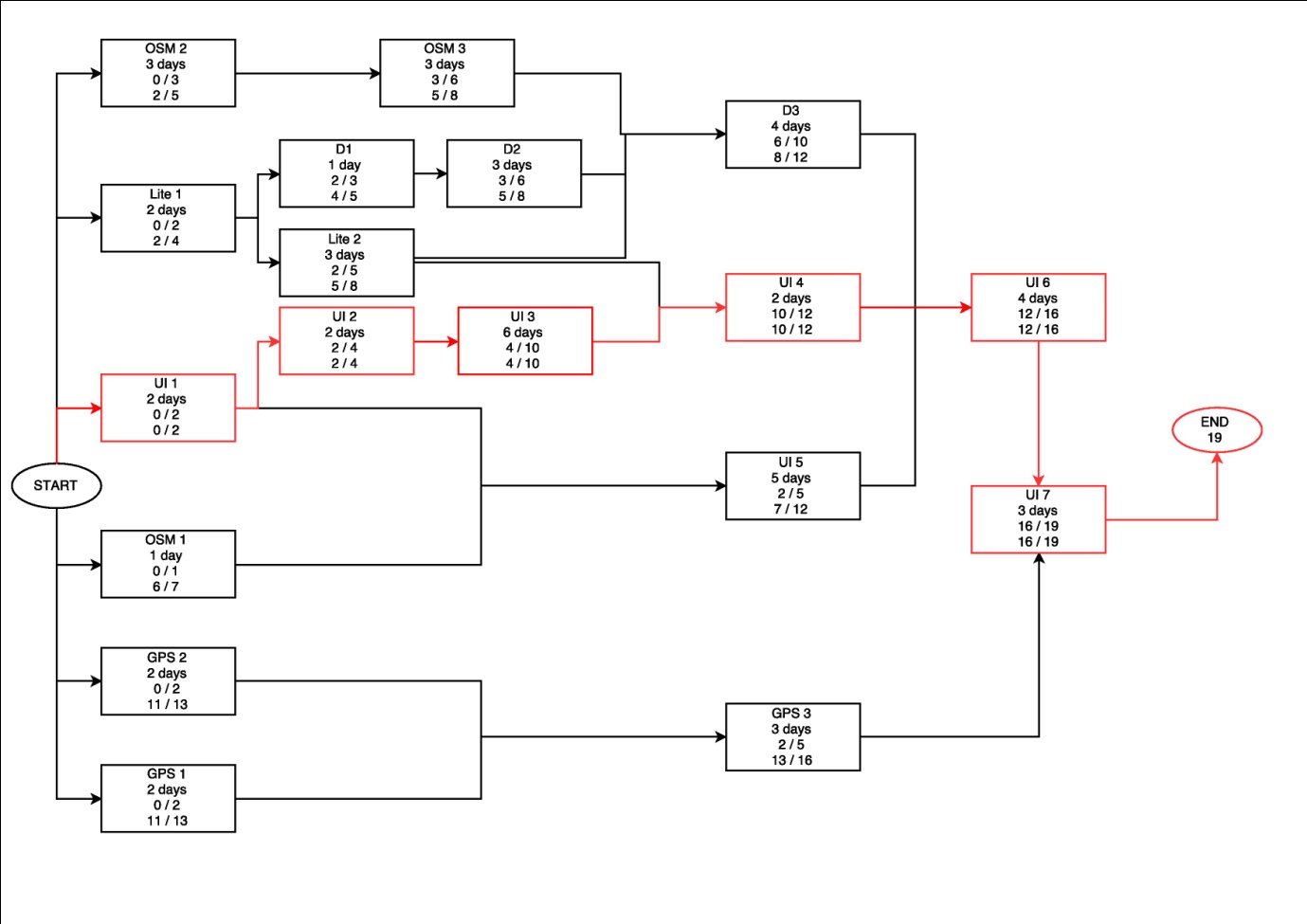
## IMPLEMENTED MINIMUM VIABLE PRODUCT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Stories** | **Module** | **Feature** | **Task** | **Deadline** |
| As a user, I want to be able to save information about a location in my phone. | Database | GPS Database (SQLITE) | Create table(s) | October 1, 2015 |
| Create a database adapter class |
| Add permissions to manifest xml |
| User Interface | Android UI XMLs | Create UI for activities |
| Create events in UI widgets to go from one activity to another |
| Create method stubs for widgets |
| Location information creator | Location text information | Automate GPS/Geolocation saving |
| Implement methods to save inputted information in the creator screen into the database |
| Location logbook | Logbook listview | Preview each location log’s information in the listview |
| As a user, I want to view the locations I have saved on Google Maps | Google Maps | Displaying locations on Google maps | Add markers to the Google map according to a location’s GPS coordinates | October 15, 2015 |
| As a user, I want to track a path starting from a specified start position to an end position | Location Tracker | Constant GPS saving | Continually save GPS coordinates in order to display a series of dots representing a trail in a Google map. |
| Create new markers for the Tracker in the map |
| As an OSM user, I want to use this app to contribute information to OSM | OSM uploader | Sending trail to OSM | Authenticate OSM user (and password) via their API | October 29, 2015 |
| Send a GPX file containing a path to OSM via their API |
| As a user, I want to be able to save my locations and paths to the internet | User database | Saving information to a server | Send user information and locations/paths to a server via Parse webserver | November 26, 2015 |
| Retrieve information from a server | Retrieve user information and locations/paths from a server via Parse webserver |
| Search information from a server | Retrieve other users’ saved paths and locations from a server via Parse webserver |
| As a user, I want to be able to use all of the app’s functions in an easy to use UI | Finalized UI | Implement functions in final UI | Place corresponding functions into finalized UI event triggers (buttons, listviews). | December 2, 2015 |

**UNIMPLEMENTED FEATURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **User Stories** | **Module** | **Feature** | **Task** |
| As a user, I want to access information I saved on the markers in Google Maps | Google Maps | Changing map types and adding information snippets | Provide settings to let the user view the markers in different types of maps (topographical, road, satellite, etc.) Display snippet when a marker is selected. The snippet will contain the text information of the location (name, address, etc.) |
| As a user, I want to save a picture of the location or path I want to keep track of | Picture saving | Saving a picture to the device | Implement permissions and methods to store a picture taken from a camera into the database |
| Sending picture to webserver | Allow the webserver to receive images as part of a location/path row |
| Retrieving pictures from webserver | Allow the webserver to send images as part of a location/path row |

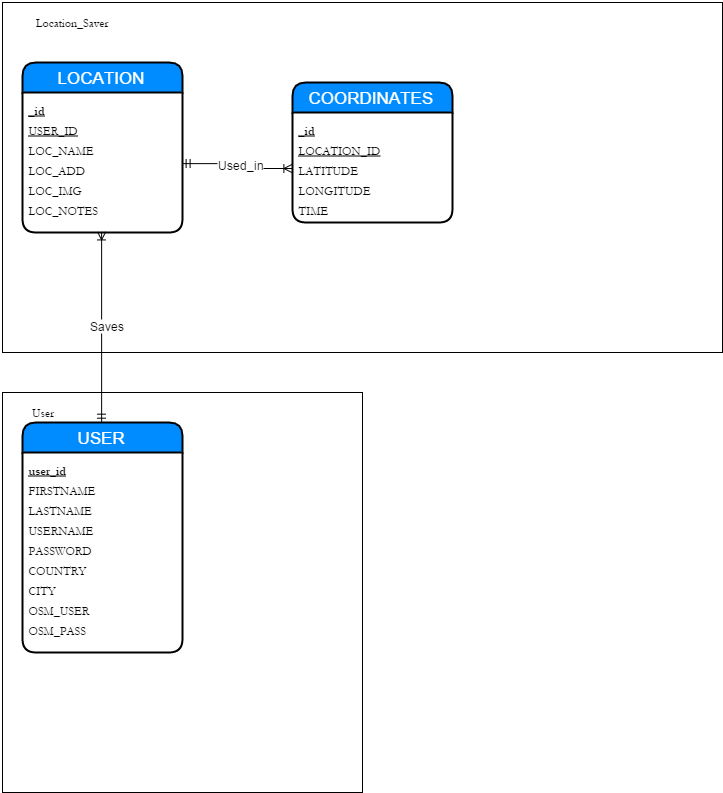
1. Activity Network Diagram



## Dependencies:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task ID | Task | Description | Estimated Duration | Dependencies | Personnel |
| Lite 1 | Design SQLite device database | Create internal database model used for the Android device. This will contain information necessary for browsing through the application without the need for internet connection. E.g. local marker information | 2 days |  | Kitop |
| Lite 2 | Adapt SQLite database | Create Android class that creates and interacts with the SQLite database in Android, even without the UI. This is to prepare for easy integration by the time the UI is finished. | 3 days | Lite 1 | Kitop |
| OSM 1 | Connect to OSM geographic database | Connect with geographic database models in Open Street Map through POST queries and XMLs. | 2 days |  | Rico |
| OSM 2 | Connect to OSM user database | Allow Open Street Map users to login through Android and make a way to collect data for that user. | 3 days |  | Kitop |
| OSM 3 | Send information to OSM user database | Allow Open Street Map users to send their paths through Android. | 3 days | OSM 2 | Kitop |
| D1 | Design online server database | Create database model for the collection of information in the online server, either through OSM/MySQL/Parse. Dynamically, this database should be able to collect ALL GPS points from all users. | 1 day | Lite 1 | Rico |
| D2 | Adapt online server database | Create Android class that creates and interacts with the online (OSM/MySQL/Parse) database in Android, even without the UI. This is to prepare for easy integration by the time the UI is finished. This needs not to be really published online with a public domain; because of budget purposes, a local server is sufficient. | 3 days | D1 | Rico, Nesh |
| D3 | Synchronize information between online and device SQLite database | Create Android adapter that synchronizes data between the server and device database whenever device is connected to the internet. | 4 days | Lite 2, OSM 3, D2 | Kitop, Nesh |
| GPS 1 | Collect GPS coordinates | Create Android adapter class that collects GPS coordinates of the device | 2 days |  | Kitop |
| GPS 2 | Establish GPS Path Recording behavior | Establish detailed rules of path recording behavior for implementation. | 2 days |  | Rico |
| GPS 3 | Constant GPS Saving | Create Android class that records GPS coordinates continuously and saves a vector of paths. | 3 days | GPS 1, GPS 2 | Rico, Kitop, Migi |
| UI 1 | Design user interface | Design a layout for the user interface in the Android device. | 2 days |  | Migi |
| UI 2 | Design user experience | Design the behavior of the user interface in the Android device. | 2 days | UI 1 | Migi |
| UI 3 | Implement user interface without backend | Implement prototype user interface with no working map and no working. | 6 days | UI 2 | Migi, Nesh |
| UI 4 | Integrate content with SQLite database | Populate and integrate the content of the UI with the SQLite database. | 2 days | UI 3, Lite 2 | Migi, Kitop |
| UI 5 | Design marker-based map tiling | Design the markers used for the map, such as the marker for current location and the marker for saved locations. | 2 days | UI 1, OSM 1 | Nesh |
| UI 6 | Display the map in the UI | Display the map from OSM data onto the user interface, along with the markers. | 4 days | UI 4, UI 5, D3 | Everyone |
| UI 7 | Implement add marker/path recording function for the UI | Record paths and add markers using the UI and save information to the SQLite database. | 3 days | UI 6, GPS 3 | Everyone |

1. Enhanced Relational Diagram



1. Data Dictionary

|  |  |
| --- | --- |
| TABLE NAME: | COORDINATES |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | LENGTH | PK? | FK |
| \_id | LONG | 10 | Yes | No |
| LOCATION\_ID | LONG | 10 | No | Yes |
| LATITUDE | DOUBLE | 11 | No | No |
| LONGITUDE | DOUBLE | 11 | No | No |
| TIME | STRING | 30 | No | No |

|  |  |
| --- | --- |
| TABLE NAME: | LOCATION |

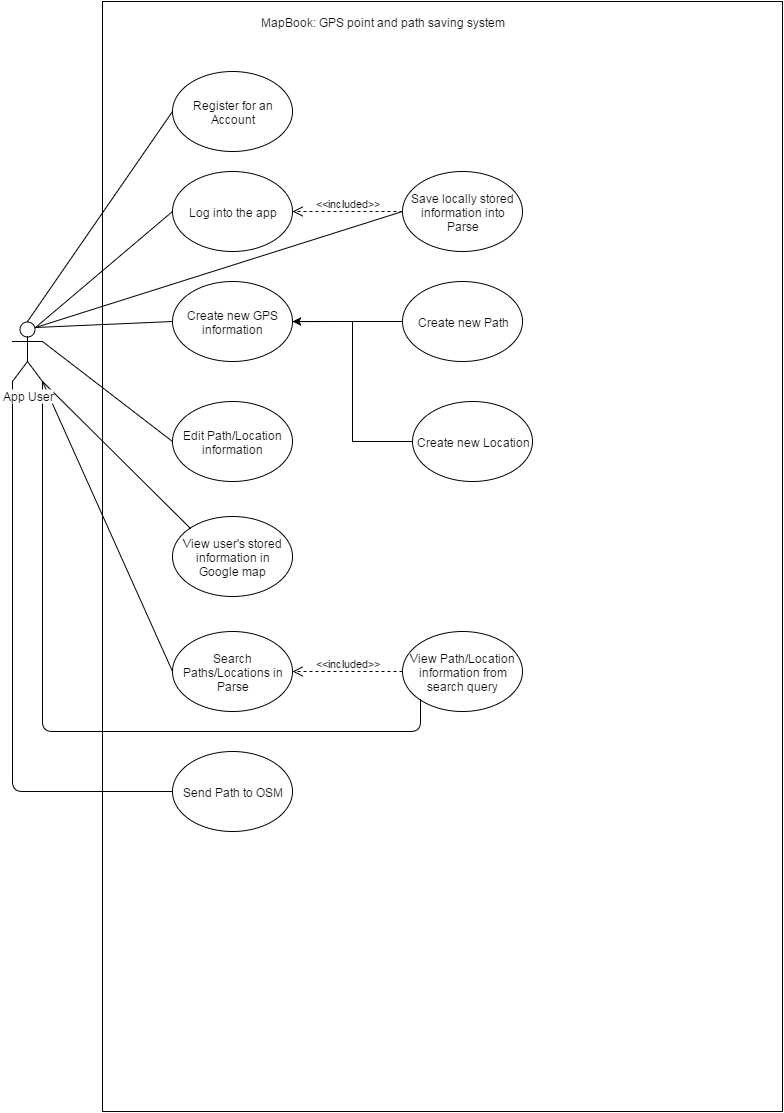
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | LENGTH | PK? | FK |
| \_id | LONG | 10 | Yes | No |
| USER\_ID | LONG | 10 | No | Yes |
| LOC\_NAME | STRING | 20 | No | No |
| LOC\_ADD | STRING | 50 | No | No |
| LOC\_IMG\* | STRING | 20 | No | No |
| LOC\_NOTES | STRING | 20 | No | No |

\*Although picture saving is currently unimplemented, the column dedicated for saving images is still implemented in the database.

|  |  |
| --- | --- |
| TABLE NAME: | USER |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FIELD NAME | DATA TYPE | LENGTH | PK? | FK |
| user\_id | LONG | 10 | Yes | Yes |
| FIRSTNAME | STRING | 16 | No | No |
| LASTNAME | STRING | 16 | No | No |
| USERNAME | STRING | 16 | No | No |
| PASSWORD | STRING | 16 | No | No |
| COUNTRY | STRING | 20 | No | No |
| CITY | STRING | 20 | No | No |
| OSM\_USER | STRING | 16 | No | No |
| OSM\_PASS | STRING | 16 | No | No |

1. Use Case Diagram



1. Use Case Description

Use Case: Register for an Account

Author: Christoffer Kho, Rico Tiongson

Date: November 26, 2015

Last Modified: December 2, 2015

Purpose: To let users have an account for saving information in the Parse webserver.

Overview: The user triggers the register activity by pressing a button in the login screen. The user inputs a username, password, first name, last name, country, city, an OSM username and OSM password in their respective text fields. The user triggers the save button. If the registration is successful, the user is returned to the login screen with a toast message saying that the registration is successful. Else, if the registration failed, a toast message will show a message saying that the registration procedure failed. If at any time the user presses the cancel button, the user will return to the login screen.

Actors: User

Preconditions:

* The device is connected to the internet.

Post conditions:

* The user has a new account that he/she can use to log into the app.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user presses the register button | 2. App checks if user is connected to internet |
| 3. User inputs information |  |
| 4. User presses save button | 5. Information is sent to Parse |
|  | 6. User is returned to the login screen |

Alternative Flow of Events:

* If at any time the user presses the cancel button, the user is returned to the login screen

Exceptional Flow of Events:

* Step 2: If the device is not connected to the internet, the user will not be sent to the registration screen.
* Step 5: If there are exceptions in Parse (such as duplicate user names), the app will tell the user that something went wrong. No new account is created in this case.

Use Case: Log into the app

Author: Christoffer Kho, Rico Tiongson

Date: November 26, 2015

Last Modified: December 2, 2015

Purpose: To let users have save information in the Parse webserver.

Overview: The user inputs a username and password in the respective text fields. If the user presses the login button, the app sends the credentials to parse. If the credentials are valid, the user logs into the app.

Actors: User

Preconditions:

* The device is connected to the internet.
* The user has a registered account.

Post conditions:

* All paths/locations associated with the user’s account in the Parse database is sent to the device and stored in the SQLite database.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user inputs his/her credentials |  |
| 2. The user presses the login button | 3. App checks if the user is connected to the internet |
|  | 4. App sends credentials to Parse for verification |
|  | 5. App sends user to main screen |
|  |  |

Alternative Flow of Events:

* None.

Exceptional Flow of Events:

* Step 3: If the device is not connected to the internet, the app will tell the user through a toast message that the user must connect to the internet.
* Step 4: If there are exceptions in Parse (such as invalid credentials), the exception in parse is displayed in a toast message.

Use Case: Save locally stored information into Parse

Author: Christoffer Kho

Date: November 30, 2015

Last Modified: December 2, 2015

Purpose: To send locally stored information to Parse.

Overview: Upon logging in, the user is prompted if he/she wants to save locally stored paths and locations to the webserver. If yes is selected, the information is saved in Parse, else, the information is untouched.

Actors: User

Preconditions:

* The device is connected to the internet.
* The user just logged in
* There are stored paths/locations in the device that aren’t associated with any accounts.

Post conditions:

* The locally stored paths/locations are now saved in the Parse webserver

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
|  | 1. App prompts user to save into Parse locally saved information in the device |
| 2. The user presses yes | 4. App sends locally stored paths/locations to Parse |
|  | 5. App toasts a message to the user that the information has been saved successfully. |

Alternative Flow of Events:

* Step 2: If the user presses no, then no information is sent to Parse. The events end here.

Exceptional Flow of Events:

* If at any time the user disconnects from the internet, no information will be sent to parse.

Use Case: Create new Location

Author: Christoffer Kho

Date: October 1, 2015

Last Modified: November 12, 2015

Purpose: To create a Location

Overview: The user inputs a location name, address and notes into the respective text fields. Once the user presses the save button, the current latitude and longitude are obtained via the device’s built in GPS capabilities. The inputted information, the GPS coordinate and the time which the button was pressed is saved into a SQLite database. If the user is logged in, the stored data is automatically sent to Parse in the background.

Actors: User

Preconditions:

* The user is logged in (if the user want to save into Parse immediately)
* The user has GPS enabled in his/her device

Post conditions:

* The location is now saved in the Parse webserver (if logged in)
* The location is now stored locally in the device.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user inputs location information |  |
| 2. The user presses the save button | 3. App gets the current time which the button is pressed |
|  | 4. App gets the current latitude and longitude from the device |
|  | 5. App saves information into the SQLite |
|  | 6. App sends the information to the Parse webserver |
|  | 7. App toasts a message saying that the location was saved successfully. |

Alternative Flow of Events:

* Step 6: If the user is not logged in, this step is skipped.

Exceptional Flow of Events:

* Step 4: If GPS is not enabled in the app, the default latitude/longitude is set to 0.0/0.0
* Step 6: If the user is disconnected from the internet, information is not sent to Parse.

Use Case: Create new Path

Author: Christoffer Kho

Date: October 1, 2015

Last Modified: November 12, 2015

Purpose: To create a Path

Overview: The user inputs a location name, address and notes into the respective text fields. Once the user presses the track button, the user is sent to the Google map. The points being tracked by the application are continually plotted onto the Google map until the user presses the save button. Once the save button is pressed, the inputted information and all the GPS points tracked are stored into the SQLite database. If the user is logged in, the stored data is automatically sent to Parse in the background.

Actors: User

Preconditions:

* The user is logged in (if the user want to save into Parse immediately)
* The user has GPS enabled in his/her device

Post conditions:

* The path is now saved in the Parse webserver (if logged in)
* The path is now stored locally in the device.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user inputs path information |  |
| 2. The user presses the “track a path” button | 3. App sends user to the Google Map |
|  | 4. App continually saves latitude/longitude/time and plots each coordinate in the Google Map. |
| 5. The user presses the save button | 6. App saves information into the SQLite |
|  | 7. App sends the information to the Parse webserver |
|  | 8. App toasts a message saying that the path was saved successfully. |

Alternative Flow of Events:

* Step 7: If the user is not logged in, this step is skipped.

Exceptional Flow of Events:

* Step 7: If the user is disconnected from the internet, information is not sent to Parse.

Use Case: Edit Path/Location information

Author: Christoffer Kho

Date: October 16, 2015

Last Modified: November 26, 2015

Purpose: Change the information of a Path/Location

Overview: The user clicks on a row in the listview displaying all the currently stored paths/locations. The information associated with the row is displayed in text fields. The user may change any of the saved information (location name, address, notes) in the text fields. If the user presses save, the changes are updated in the SQLite database. If the user is logged in, the changes are updated in Parse. If the user presses cancel any time, the user is returned to the listview.

Actors: User

Preconditions:

* The user is logged in (if the user want to save into Parse immediately)
* There are currently stored paths/locations in the device

Post conditions:

* The path is now updated in the Parse webserver (if logged in)
* The path is now updated locally in the device.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user selects a row in the listview | 2. App displays selected row’s information in textfields. |
| 3. User changes information from the textfields | 4. App continually saves latitude/longitude/time and plots each coordinate in the Google Map. |
| 5. The user presses the “save changes” button | 6. App updates information into the SQLite |
|  | 7. App sends the updated information to the Parse webserver |
|  | 8. App toasts a message saying that the information was updated successfully. |

Alternative Flow of Events:

* If at any time the user presses the cancel button, the user is returned to the listview.
* Step 5: If the user presses the “save changes” button with no changes made, the app will toast a message saying that no changes were made.
* Step 7: If the user is not logged in, this step is skipped.

Exceptional Flow of Events:

* Step 7: If the user is disconnected from the internet, information is not sent to Parse.

Use Case: View user’s stored information in Google map

Author: Christoffer Kho

Date: October 1, 2015

Last Modified: November 12, 2015

Purpose: View coordinates in the map

Overview: The user on the map button in the viewflipper. The app then displays all currently saved paths/locations in the SQLite database.

Actors: User

Preconditions:

* The user is connected to the internet

Post conditions:

* None.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user presses the map button in the viewflipper | 2. App plots all currently stored paths/locations in the Google Map |

Alternative Flow of Events:

* None.

Exceptional Flow of Events:

* Step 2: If the user is not connected to the internet, the app will still plot points on the Google map, but the map tiles will not be seen (or at least may appear blurry) because they must be downloaded while connected to the internet.

Use Case: Send Path to OSM

Author: Christoffer Kho

Date: October 16, 2015

Last Modified: October 30, 2015

Purpose: Sends a .gpx file created from a Path for OSM Public GPS Traces

Overview: The user clicks on a row in the listview displaying all the currently stored paths/locations. The user presses the “Send to OSM” button. The app obtains information from the SQLite database and stores the information into a .gpx file in the device’s memory. The .gpx file is sent to OSM via their API.

Actors: User

Preconditions:

* The user is connected to the internet
* The user has a path (meaning multiple GPS points)

Post conditions:

* The .gpx file can be viewed in OSM’s Public GPS Traces

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user selects a path in the listview |  |
| 2. The user presses the “send to osm” button | 3. App saves the path’s information in a .gpx file in the device’s memory |
|  | 4. App sends the .gpx file to OSM Public GPS Traces via OSM’s API. |

Alternative Flow of Events:

* None.

Exceptional Flow of Events:

* Step 1 and 2: If the user selects a location (1 GPS coordinate only) in the listview and presses the “send to OSM” button, nothing will be sent.

Use Case: Search Paths/Locations in Parse

Author: Christoffer Kho

Date: November 13, 2015

Last Modified: December 2, 2015

Purpose: Searches paths/locations of other users in the Parse webserver

Overview: The user inputs a search text in the text field. The user selects whether the search will filter through addresses or location names. When the user presses the search button, the app sends query requests to Parse. Once parse returns the search query, the contents of the search query are displayed in rows in a listview.

Actors: User

Preconditions:

* The user is connected to the internet

Post conditions:

* The contents of the search query are displayed in a listview.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user inputs a search text |  |
| 2. The user selects a radio button to filter the search query to location name |  |
| 3. The user presses the search button | 4. App sends the search request to Parse |
|  | 5. Once the app receives the search query from Parse, the information in the search query is dispayed in a listview |

Alternative Flow of Events:

* Step 2: The user can select a radio button to filter the search query to addresses as well

Exceptional Flow of Events:

* Step 5: If no results are found, then the listview is not populated.

Use Case: View Path/Location information from search query

Author: Christoffer Kho

Date: November 13, 2015

Last Modified: December 2, 2015

Purpose: Displays information of a selected path/location from a search query

Overview: The user selects a row from the search listview. The row’s location name, address, notes and GPS coordinates are displayed. The GPS coordinates are plotted in a Google map.

Actors: User

Preconditions:

* The user is connected to the internet

Post conditions:

* None.

Typical Course of Events:

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. The user selects a row in the search listview | 2. App retrieves information from the row |
|  | 3. App displays retrieved information from the row in text views. |
|  | 4. App plots the row’s GPS coordinates in a Google map. |

Alternative Flow of Events:

* None.

Exceptional Flow of Events:

* None.

1. Lessons Learned

Overall Learnings:

* There has to be constant communication in order for a team to get things done as quickly as possible, and so that members of the team are always reminded of their requirements.
* It is important to know the strengths and weaknesses of each member so that the team can adjust the workload, and be able to set their goals appropriately
* Specifications can change anytime.
* Development is best done when you don’t have a lot of other requirements to do.
* It’s good to clarify with your “employer” about what they want in the product, as well as whatever changes you want to make with said product so that there is a better understanding on what the end goal is.
* Assigning tasks to members of the group based on skill set makes the project much easier to work on. Not to say that the project was anything but difficult.
* Estimated completion time is never precise due to unforeseen obstacles.
* Managing our schedules to work on the project was more difficult than expected. Everyone had many other requirements and classes at certain times so working together at the same time became very difficult.
* Plugging in UI to a working code base does not always go as planned.
* Weekly updates within the team about the progress of the project goes a long way
* Collaborative tools such as Google docs and Github makes life much easier.

Comments/Thoughts:

Christoffer Kho:

Making software with corresponding documentation is hard work. Up until now, making programs through code-and-fix was the only thing I did. It worked for the most part, but for this project, I think it just wouldn’t be enough.

This app has several components. It has a solely GPS component which is dedicated to saving GPS coordinates. This component had to be connected to a user system somehow in order to allow the sharing of information. Saved paths had to be properly, and flawlessly, translated into a proper .gpx file in order to be sent as a multipart-data form to OSM’s API. There are several thing that could have gone horribly wrong in this project if it were not implemented through sprints and constant consultations with our professor.

With all that said, however, I don’t think this project could have been done better given the circumstances of each member. Everyone had other requirement to meet for other classes. During the November 12 sprint, for example, I was more focused on finishing a paper which had a deadline on November 9. That left me with around 4 day to work on the components that needed to be delivered for that particular sprint. Somehow, it was done.

I would personally love to see this project be improved for particular functions. The searching function can allow users to download .gpx files from other users and store information from it into their SQLite database. The Google map can be replaced with an OSM map, and could allow for direct contributions to OSM if the user wants.

Migi Ramos:

Though in the end the product was able to meet the projected base product, it is sad to see that it could have been improved upon much more given more time. Unfortunately, we did have other classes and requirements to attend to as well, and such we weren’t able to devote all our time for the betterment of our product, else we fail in other classes. It also didn’t help that our knowledge in regards to coding for android is fairly limited and that only about half our group had experience with it beforehand, though it still isn’t too much. Hopefully there would come a day where this project would be once again picked up and improved upon, whether it be us or some other team.

Nitesh Purswani:

After this class and working on the CS123 Final Project, I have a newfound respect for software engineers. I used to think that programmers and software engineers were interchangeable, but creating reusable and scalable code that has to reach a certain deadline is much much harder than just coding something out of a whim. I hope to see this app updated with new features as well as an animated UI (not added due to time constraint). The app itself has a lot of promise and hopefully we will pick up on it and get to actually release the product publicly as well as bring it to other platforms such as iOS.