LOCATION BASED EDUCATIONAL MOBILE APPLICATION DESIGN AND IMPLEMENTATION

A thesis submitted
to Kent State University in
partial fulfillment of the requirements
for the degree of Master of Science

by
Chang Shu

March 2017

© Copyright
All rights reserved
Except for previously published materials
Thesis written by

Chang Shu

M.S., Kent State University, USA, 2017

B.S., Beifang University for Nationalities, China 2014

Approved by

Dr. Cheng-Chang Lu, Chair, Master Thesis Committee

Dr. Richard E. Ferdig, Members, Master Thesis Committee

Dr. Austin Melton, Members, Master Thesis Committee

Dr. Xiang Lian, Members, Master Thesis Committee

Accepted by

Dr. Jayed Khan, Chair, Department of Computer Science

Dr. Janis Crowther, Dean, College of Arts and Science
TABLE OF CONTENTS

LIST OF FIGURES............................................................................................................. VI

LIST OF TABLES.................................................................................................................. VII

ACKNOWLEDGEMENT......................................................................................................... VIII

CHAPTER 1 INTRODUCTION.............................................................................................. 1

1.1. Location Based Services application ........................................................................ 1

1.2. Requirements of mobile application ........................................................................ 1

1.2.1. Park Map feature .................................................................................................. 1

1.2.2. LBS features ........................................................................................................ 2

1.2.3. Contents related features ...................................................................................... 3

1.3. Overview application design and implements .......................................................... 4

1.4. Overview testing and deployment ............................................................................ 5

CHAPTER 2 REQUIREMENT ANALYSIS ........................................................................... 6

2.1. User functional requirements .................................................................................... 7

2.1.1. Map ....................................................................................................................... 7

2.1.2. Adventure Tracks ................................................................................................. 9

2.1.3. Learn as you Go ................................................................................................. 10

2.1.4. Citizen Science .................................................................................................. 11

2.1.5. Identify ................................................................................................................ 11

2.1.6. My ParkApps ...................................................................................................... 11

2.2. Service functional requirements ................................................................................ 12

2.2.1. User role services ............................................................................................... 12
5.1. Current progress ............................................................................................................. 31
5.2. Future plans .................................................................................................................. 31
5.3. Final words ................................................................................................................... 32
REFERENCES ..................................................................................................................... 33
LIST OF FIGURES

Figure 1 Mobile Application Architecture .......................................................... 6
Figure 2 GeoJSON example .................................................................................. 8
Figure 3 User roles ............................................................................................... 13
Figure 4 Application used third part libraries ...................................................... 17
Figure 5 Application uniform map style ............................................................... 18
Figure 6 ORM used in application ........................................................................ 19
Figure 7 example of GET request result ............................................................... 20
Figure 8 UML mobile application use case diagram ............................................ 22
Figure 9 UML of Class diagrams ......................................................................... 23
Figure 10 UML sequence diagram of content update .......................................... 24
Figure 11 UML sequence diagram of distance calculation ................................... 25
Figure 12 application prototype of Adventure Tracks .......................................... 26
Figure 13 example of Geofence on trails .............................................................. 30
LIST OF TABLES

Table 1 Location service compare ................................................................. 14
ACKNOWLEDGEMENT

I would first like to thank my thesis advisor Dr. Cheng-Chang Lu of the Computer Science at Kent State University. The door to Prof. Lu office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to acknowledge Dr. Rickard Ferdig of the Lifespan Dev & Education Sciences at Kent State University as the second reader of this thesis, and I am gratefully indebted to his/her for his/her very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents and to my girlfriend for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.
Chapter 1

Introduction

1.1. Location Based Services application

The Location-based services (LBS) can be defined as the service that denotes application integrating geographic location with other information, to provide knowledgeable information to users. [1] Applications based on LBS will take advantage of mobile devices’ GPS information to provide the user with more valuable information. Our research project will be based on LBS, to design and implement educational mobile applications. A series of applications will provide useful National Parks locations-based information for park visitors. Additionally, the records will also provide visitors’ feedback and users data for scientists. This thesis will discuss the research project mobile application ParkApps – NE Ohio, which will be applied in Cuyahoga Valley National Park (CVNP) and (Cleveland Metro Parks) CMP.

1.2. Requirements of mobile application

1.2.1. Park Map feature

‘Park Map’ provides online and offline National Parks maps for park visitors. Users will be able to switch maps from online and offline, based on the network conditions at park fields. In case of variability of network states, a monitor program has been designed in the global application environment. This system will be able to monitor networking states in real-time, and notify users to change maps between online and offline. There are multi POIs (Point of Interest)
which contains information including images, place details, amenities, direction information, etc. which could be visualized on park maps. All POIs will be able to update through (Representational state transfer) REST API when devices are online, and maintain persistent display when devices are offline. In addition, maps will provide highlighted walking or cycling trails, which are loaded from external GeoJSON resources.

1.2.2. LBS features

LBS features denote as ‘Adventure Tracks’, ‘Learn as you Go’, and ‘Citizen Science’. These three features will make use of GPS from a mobile device at a high frequency level. ‘Adventure Tracks’ and ‘Citizen Science’ will pull data from REST API, that contains locations coordinate information and locations related information. Data will be pulled from the server, and then cached at a local documented database. Each ‘Adventure tracks’ or ‘Citizen Science’ will locate a specific region. Normally, all POIs will be located around trails. Each POI information basically contains longitude, latitude, and radius. Every time users move into the geofence for each point, the application will push related notifications to them. There are two approaches to implement distance calculation, which are local based polling calculation and server based request calculation. The feature, named ‘Learn as you Go’, will extend POIs locations in a large region of the whole National Park, thus, the location calculation approach will be adjusted accordingly. Device local-based polling calculation method will calculate the distance between users’ real-time locations, and all POIs positions. When users’ motions are faster, the frequency of calculation will increase; if users’ motions are slowing down, the calculation frequency will decrease as well. In the case of users who are standing still, the calculation will reduce to one time per minute. The reason to design this calculation frequency adjustment feature, is to make the batteries work efficiently. If the application continues to use
highly accurate calculation, although the chips of CPU and GPS in mobile device could continue working well, the efficiency of the battery will decrease. Server-based request calculations will simulate distance calculation based on networking and server return results.

The 'Identify' is developed based on LBS. For the 'Identify' feature, if a user creates an observation at a specific location, the application will record the user’s location, and provide the user an interface to take pictures or select pictures from albums. When the user takes or picks photos from albums, the application will upload photos and coordinate information to the server. The server will process uploaded photos, then send photos to iNaturalist, which is a community to allow people to identify objects from the public database. When an object in the database has been identified, the user will get an identified name from the history list.

**1.2.3. Contents related features**

Questions and Badges Rewards systems are associated with previous LBS features. When ‘Adventure Tracks’, ‘Citizen Science’ and ‘Learn as you Go’ features are being used, the application will present different kinds of questions to users. Thus, we have built-in questions and badges management systems in the application. This system will pull questions from server API and manage local storages. Also, this system will be able to read users’ input and give them feedback. If a user reaches some achievement, badges will be assigned to their account. Meanwhile, users will be able to check badges and other information under their account.

The user activity collection system will be working in the background of the application. This feature is mainly focused on anonymous location information collections for park administrators and scientists. This monitor service will keep recording users’ motion coordinates. If there is an Internet connection, the services will send geographic information to backend
server. If the Internet does not connect, the services will temporary save user activity to the local database, and once the connection is established, it will report to server.

1.3. **Overview application design and implements**

This thesis will mainly focus on design and implementation of the mobile application under iOS platform. From the implementation aspect, we used Objective-C as mainly development programming language to build native application. The application integrated with third party libraries to realize rapid development. Third-party libraries include, but are not limited to: AFNetworking, Realm, SDWebimage, MapBox, etc. On the server end, PHP has been used as the development environment, and JSON has been used as the data exchange format through REST API. Through the third-party libraries which are open sources under MIT or BSD licenses, we could integrate them into the application without restrictions. MapBox provides map visualization and map related features, and it provides highly customized map style, and stable performance. What’s more, MapBox provides a completely offline map tiles caching feature, which will provide users with an offline map, when there is no Internet connection. Network library will mainly rely on ‘AFNetworking’, which provides PUT, GET, POST, and DELETE methods. We used ‘Realm’ as the application local database, which could rapidly read and write massive objects. Realm is based on SQLite implementation, and it provides offline, and object-oriented storage interface for developers. We created ‘Point’, ‘Track’, ‘Question’, ‘Badge’, and ‘User’ collections under Realm databases, to meet caching all offline feature requirements. For interface design, we will implement user interface from a design team, and according to their design wireframe we will build a user interface.


1.4. **Overview testing and deployment**

There are two stages of testing procedures for each version for the application. The first stage is functional testing through Xcode unit test class. The second stage is Non-Functional test, including performance testing, usability testing and compatibility testing. Function testing will cover all requirements during the development stage. When all test cases pass, an application with specific build version will be submit to TestFlight which is testing platform provided by Apple. Then the internal test team will check applications’ usability and compatibility. This application is designed to work in all versions after iOS 9.0, and all screen sizes of iPhone devices are supported. After internal testing, we have the option of beta testing for small publics or we can release our latest version. At the same time, we have created another test version for new features development that does not have any effect on main branch version.

When there is a build version that needs to be released to the public, it will be released through iTunes Connection. At back end of iTunes Connection, there are false Fig information submitted from user’s devices, download number and download region. iTunes Connection platform provides us roll back function, if some latest features are not working as expecting, we will roll back to the stable version. At same time, our website (parkapps.kent.edu) provides a portal for scientists to monitor user’s activity.

At last, learning information will be managed via web based system. Users information and user’s activity will be analyzed via web based system and exported to Excel sheets. The mobile application live version and development version will both obtain the dynamic contents from web system and send user’s activity.
Chapter 2

Requirement analysis

In this chapter, the whole application requirements will be introduced completely. Requirement analysis stems from functional requirements and non-functional requirements. Functional requirements analysis will cover all features including iOS mobile application and they can be categorized into two main types: 1. User functional requirements, 2. Services functional requirements.

The whole application architecture is shown in Error! Reference source not found.. Mobile application will be built based on this architecture. Base service includes: location services, database services, and server side services. Middleware controllers will construct different features by accessing different base services.

Figure 1 Mobile Application Architecture
2.1. User functional requirements

In this section, we will discuss user functional requirements. User functional requirements will mainly focus on user interaction with application and the application could provide some specifics features for users.

2.1.1. Map

Requirement in Map features including:

1. Showing current user’s real position.
2. Highlighting walking trails in park.
3. Switching offline or online map tiles.
4. Updating latest POIs on map.

Whenever users use map features, they need to locate their position at the park. Therefore, we need to update location information during the constant time periods; type one location service has been used in the map features. Map interface has been displayed below, it keeps updating a user location when the users stay in the same interface. In addition, when location button has been double tapped, it will rotate automatically on the map and show the directions.

Park trails data is presented in GeoJSON file, file format example is presented in Figure 2 GeoJSON example. A geometry is a GeoJSON object where the type member's value is one of the following strings: "Point", "MultiPoint", "LineString", "MultiLineString", "Polygon", "MultiPolygon", or "GeometryCollection". All trails will be represented in “LineString” object. [2]
In each geometry object, coordinates array will contain a series of points to drawing trails. Trails resources file will be imported to memory when users touch the trail button. After trails resources files have been loaded to memory the first time, it will remain in memory until memory warning is received.

```json
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "feature",
      "geometry": {
        "type": "Point",
        "coordinates": [102.0, 0.5]
      },
      "properties": {
        "prop0": "value0"
      }
    },
    {
      "type": "feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": 0.0
      }
    },
    {
      "type": "feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
          [100.0, 1.0], [100.0, 0.0]
        ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": {"this": "that"}
      }
    }
  ]
}
```

**Figure 2 GeoJSON example**

Online and offline map switch is another significant feature under map features. The application could provide an accessible and stable map to users in cases of no cellular data or unstable signal in some part of park. We applied MapBox iOS SDK as a main map visualization component. MapBox supports loading static tile files as key map. Therefore, we use map tile cache tool, named “tiles,” to package all tiles into a tile file. File size shows an exponent increase with map zoom level, we are going to limit cache tiles level to threshold 13 (1 - 20). For example, at level 13, map file size (less than 30Mb) is balanced with map display details.
POIs update features include two phases at background. Phase 1 is the program checks Internet reachability at background. Phase 2 is according to the result of reachability: if network is connected, program pulls latest POIs using GET method from server API, then documents results to database. If network is not reachable, the program accesses local database then loads previous POIs data.

2.1.2. Adventure Tracks

Adventure Tracks function requires application push notification to user at real-time based on their location. As expect, this feature will be activated when users tour at specific trails in park. Each Adventure Track will correspond to a specific trail. Managers predefine a series of POIs with different geofence along trails, and each POIs corresponds to related information or a type of questions. When users enter POIs geofence, the application will give users notifications to check information or answer questions. Therefore, we need to use high accurate GPS location to calculate user real-time relative distances with POIs.

Location distance calculation service will be used in Adventure Tracks, next chapter we will discuss about this service optimization. Within Adventure Tracks function, here are listed related features:

1. Pull update tracks list from server and cache to local database.
2. Pull update questions from server and cache to local filesystem.
3. Two type Adventure Tracks display approaches (Map or List).
4. Display and response different type of questions.

In updating tracks list features, we design RESTful API at server side for mobile application accessing tracks list. According to different track’s id, mobile application will access each separate track contents through another API. After downloading tracks list, application will
store tracks list into database. According to different user roles, tracks list will be different. Admin will see approved and unapproved tracks, however, the general user will only see approved tracks.

After application obtains latest tracks list, it will download each track details respectively. Each track contains a set of questions or information, and they are represented by JSON format. Different question type will be represented in different JSON format, thus questions will not be fit for keeping in relational database. Questions JSON object will directly be saved to iOS file system. Questions will be displayed in different users interface according to question types.

The two types of user interface are provided to show questions in Adventure Tracks correspondingly. Each Adventure Tracks questions will be displayed on map or on list which is determined by the learning information.

2.1.3. Learn as you Go

As we have explored Adventure Tracks function requirement, learn as you Go provides users large regional scale function in a national park. Adventure Track normally provides users a specific trail for user touring and push notification, however learn as you Go will expend this ability to the whole park area. Content manager will place POIs to whole parks areas, and each POIs will correspond to a related question. When a user is walking through POIs, application will give the user notification. Learn as you Go function contains the following features:

1. Pull update questions from server and cache to file system.

2. Display and response different type of questions.
2.1.4. Citizen Science

Citizen Science function will provide similar Adventure Tracks functions, but project creators will become citizen science contributors. Each Citizen Science project will contain a series of questions. Questions could be displayed on map or listed as different task. Each project’s purpose can be information collection or providing survey for park visitor or target people. The manager will be able to see all Citizen Science projects even the unapproved. Citizen Science contributors will be able to check their own projects.

2.1.5. Identify

Identify function will provide ability to identify objects within photos which are captured by users. Identify function will work with backend server and iNaturalist which provides social community to identify unknown plants or animals. When users are taking photos, this function will record users’ real-time location. Users will be able to retake photos or choose photos from albums. When users submit photos to server, GPS coordinates will be submitted with photos. When the server get users’ uploaded photos, the server will post users’ photos to iNaturalist and wait to be identified. When users’ photos have been identified, users will be able to check identify results.

2.1.6. My ParkApps

My ParkApps provides portal for users to check their own account status. Under this function, users will be able to check their basic information, create field notebooks, record POIs on map and check badges. This feature is only accessible for users who have signed up. If users didn’t sign in, application will give them direction to sign in.
2.2. **Service functional requirements**

Services functional requirements mainly provides support service modules for the application. Some user functions in the application are based on service modules support, users are not aware of these modules running in background.

2.2.1. **User role services**

User roles service provides application ability that separate functions by different user roles. When users first download the application, users will be considered as non-registered public users, all guest features will be provided. Once users sign in, the application will get user roles from the server, correspondingly grant access to high level features for users. All user roles will be managed on web portal and only can be modified by root admin. User roles are defined as these levels:

- **Root administrator:** top user level, currently only one instance account is root administrator. Under this user role, they will be able to check all user information and modify others users’ roles. Root administrator also will be able to modify POIs, adventures tracks data, and badges data.

- **Park administrator:** Under this role, users will be able to create and modify in park region adventure tracks, and approve citizen sciences projects. Users also will be able to create POIs and badges data.

- **Citizen Scientist:** users will be able to only create their own citizen sciences projects and record their points.
• Public registered and non-registered user: all public users will be able to use map features, and explore adventure tracks and citizen sciences. Signed up users will be able to access their own account and contribute their POI data to server.

User role features can be demonstrated by the diagram below:

![User roles diagram](image)

**Figure 3 User roles**

### 2.2.2. Question contents services

Question interpretation and management is one of the service requirements in the application. Questions contents services will manage different type of questions, including: open question, multiple choice question, single choice question, match question, order question, text/image question. When different questions are needed to be displayed to users, this service will query related question contents then build user interface for other functions. After users make choices, questions services will evaluate users’ choices and send users different feedback. Then in the background, question services will keep user answers recorded for sending back to server.
2.2.3. Location distance calculation services

Based on application features requirements, we defined 3 types of location based services for different scenarios:

1. Constant time location report services.
2. Significant distance change location report services.
3. High frequent distance calculation services

We will discuss different location service types usage in different contexts. Each type of location service will perform different location accuracy, time latency and battery life efficiency. All three type of location services have been shown in the below table.

<table>
<thead>
<tr>
<th>Type</th>
<th>Accurate</th>
<th>Latency</th>
<th>Battery efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>3</td>
<td>Best</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 1 Location service compare

Background user location report will use type 1 services, since the application required to be able to constantly report user’s location, so we need stable service to constantly report location.

2.2.4. Anonymous information collection service

Requirement for record user activity on answering questions is for scientist analyze questions quality and make impact on educational content. Thus, we will mark and modify JSON structure after user answered questions, then save new JSON object into file system. When user exit or quit application, we will send user activity report to server side.
2.3. Non-functional requirements

Application needs to run on iOS platform, system version after 9.0 will be supported. iPhone/iPod touch/iPad devices will be supported, supporting screen size include 3.5, 4, 4.7, 5.5. This application is only working on portrait device orientations. Specifically, iPod touch doesn’t cover GPS module. If user choose iPod touch as testing device, they need external GPS module to acquire real-time location.
Chapter 3

Chosen technologies

3.1. Platform

iOS mobile application will can be implemented in Objective-C or Swift programming language. In our research project, we have chosen Objective-C as development language, there are two reasons: 1. Lots of stable third parties libraries are using Objective-C as implementation language. Swift and Objective-C mix in one project will cause compiler performance degrades. Therefore, only Objective-C will be used in project. 2. Swift is a kind of interpretation language developed by Apple. Swift is under rapid development in recent years, in case language syntax confliction in future development or maintenance, we will choose stable Objective-C as development language.

Due different devices with needs different screen resolutions, Auto Layout will be used in application. Auto Layout system that lets application will be able to arrange UI components in different screen resolutions automatically. Using Auto Layout in application, we can create dynamic and versatile UIs without concern about different size of devices.

3.2. Third party libraries

In application, most of UI components are constructed by CocoaTouch frame, some parts are based on third party libraries. Question display part will base on ‘SCLAlertView’, which
provide well design and functional interface for question display and options. All third party libraries used in application is listed in diagrams below.

![Podfile content]

**Figure 4 Application used third part libraries**

### 3.3. Map

Map display for user will be a significant part in application, and Apple provide MapKit in native framework for developer. However, MapKit is not sufficient for application requirements, offline map and highly customization map features style are not supported. Therefore, MapBox will be introduced into application, which is map framework provides
complete offline map and highly customization component. In application, we will use uniform map style and reuse all map feature base on MapBox framework and all map features will share offline data in local file system. MapBox bright map style will be show in below diagrams, application will take this map style as uniform map style.

![MapBox Bright Map Style](image)

3.4. Database

Realm is third party database library used in application, although Apple provide Core Data storage framework. Compare with Core Data framework, Realm framework is lighter and more concise. We defined Points, Tracks, User, Badge, Images tables in database. Corresponding we created ORM (Object relational mapping) in development project. ORM will
map programming object to database object, which provides convenient methods CRUD (create read update delete) with database. In below diagrams shows all ORM will be used in project.

![Diagram of ORM used in application](image.png)

**Figure 6 ORM used in application**

### 3.5. Networking

When application needs communication with server, iOS native framework provides a set of methods to access RESTful API. However, those native methods are not concise and not reusable for similar functions. Therefore, AFNetworking is primary networking framework used in application. This framework encapsulates GET/POST/PUT/DELETE networking operations
for developers, so we will reuse this framework in everywhere in application. In additional, some similar function will share same networking operations, so we will encapsulate networking framework into business logic. Below diagram is an example of server API GET adventure tracks data in Postman.

![Example of GET request result](image)

**Figure 7** example of GET request result
Chapter 4

Application implementation

4.1. Application design

In this chapter, we will begin discussing details of application design and implementation. In application design part, UML (Unified Modeling Language) will be used to visualize application system design process. UML is a kind of diagram enabling to specify, visualize, construct and document artifacts of a software system.
4.1.1. UML diagrams

4.1.2. Use case diagrams

Figure 8 UML mobile application use case diagram

4.1.3. Class diagrams

Class diagrams will present application class relationship with each other. In class diagrams, UI components or application service are abstracted into UML class. Each component relation will be represented on the UML diagrams.
Sequence diagrams visualize objects communication in a sequence of messages. In the diagrams of below will visualize to major use case in application. First case is that representing content data will be request from server, then update in local database and finally update UI components. Second case shows how distance calculation working in adventure tracks function.
Figure 10 UML sequence diagram of content update
4.2. Application prototypes

Application prototype guides user interface programming and directly business logic in application. In addition, prototype gives controllers navigation guide in application which means exactly behaviors in application interface. According different behaviors in application different UI components will be used in each controller. Base on prototype navigation methods, Navigation Controller as main navigation methods. Navigation Controller will let a set of controllers jump from root controller to each individual function, and give intuitional back/forward operation for user.
Adventure Tracks list wireframe demonstrates how list data be presented on the controller. UITableView is common way to display list data on controller for user. User tap on each individual cell in table view will give user details of this track. In addition, UITableView are nested in navigation view controller.

In the track details controller, a map view will take responsibility to demonstrate all points, when user approaching or tapping on point it will pop up information. On top right corner menu button will lead user to see all question list.

![Application prototype of Adventure Tracks](image)

Figure 12 application prototype of Adventure Tracks

4.3. Used technologies

4.3.1. Development environment

iOS mobile application will be developed in macOS with Xcode IDE (Integrated development environment). Server side application will be developed in PHP and deploy in Apache web server. Third party library will be managed by Cocoapod which provides reliable
third party libraries dependency. Under development testing process, iPhone simulator will be used. Simulator provides real iPhone or iPod touch application environment, but running on macOS. Simulator can simulate multiple devices, therefore under development we will be able test mobile application on diversity screen size without real devices. When we need testing location related functions, simulator will provide controllable location simulation. Simulator provides function that allow developer input virtual GPS location by put GPX file as GPS signal input. Postman is a Chrome app which provides full REST API testing. When mobile application communicates with server application, Postman will be used to test API content correction and reachability. TestFlight is Apple test platform for application development team, it provides internal test, Beta test for different application version. In our development, TestFlight platform to distribute our latest developing application to test user.

4.3.2. Used design pattern

4.3.2.1. MVC

MVC stands for Model/View/Controller which is a common design pattern used in programming. Model represents an object to carry data, sometimes model will be able update controller when data changed. View visualized data on application interface and itself can be a kind of class. Controller interact with model and view, controller will control to update model data when view has some change, reversely when model data changed controller will update view as well.

MVC pattern is prefer fit for mobile application development. This pattern make model are reusable, and their interface tend to better defined. Application with MVC is more easily to extent new function without changing related function codes.
4.3.2.2. Singleton

Singleton pattern is kind of simple design pattern. This design pattern provides best way to create single instance object in application. LocationService and DatabaseManager class are using singleton design pattern under global environment. So whenever other class needs instance of this two class, there will only one copy existing in system. Only one class instance in memory guarantee that other class get same instance and get same data, for instance same GPS coordinates, or same database reference. Only one class instance guarantee related classes get same value real-time data instead inconsistent data.

4.3.2.3. Delegate

Delegation is a kind of simple design pattern which provides one object can behalf of another object. Delegate object will be inserted into delegating object. When delegating object needs some functions, it doesn’t need implement by inside, delegate object will hand over functions. This design pattern is widely used in application. For instance, when application service sending GPS coordinates to other class, there will be a delegate object receive GPS message and perform different actions. Even in UI component parts delegate are still widely used. UITableView are designed in delegation mode, UITableViewDataSource is a delegate needs to be implemented which will determine how much rows and sections in table.

4.3.3. Distance calculation algorithm

Geofence is a virtual geographic boundary, defined by GPS technology, that enables software to trigger a response when a mobile device enters or leaves a particular area. Each point has its own individual geofence radius. In diagram below, a set of points place on map, each point with different radius geofence.
The purpose of calculation distance is to determine user enter or leave each geofence. Weak GPS signal will cause user location bounces at some region. Therefore, in this algorithm will pick best GPS signal from all GPS signals list. Type 1 location report method will only pick up coordinates from constant period. Type 2 location report method only update location information when location change significate distance such as 500 meters or more. This function is for save more battery power when application doesn’t need continuous tracking.

Type 3 is combination of Type 1 and Type 2. When user in motion, location service will force be set to Type 1 and calculation distance with each POI. When distance is smaller than one of radius, there will be one POI in range. When function needs left geofence information, hashmap or hashset will be used to record user location and POI statues. When user motion continuous, application will keep using Typ1. Under this mode battery consumption will keeping at same rate.
When user motion slow down or standstill, application will detect this behavior by calculating distance between user location displacement. When displacement is lower than threshold, calculation will be paused and will be resumed displacement higher than threshold. In application, the threshold will be defined as 10 meter within 20 seconds.

Server side calculation is a kind of assist method that move calculation from local to server side to reduce CPU work load. This requirement is for user who low frequency uses and reduce battery consumption. Switching to server side calculation will rely on consistent communication with server. Through simply send user location and track id, server side will give user current in range questions and information.
Chapter 5

Conclusion

5.1. Current progress

iOS mobile application ParkApps NE Ohio has been released on AppStore, some features are still in progress. Application has been duplicated two separate version live version and test version. Live version is for public user using, test version is for internal team developing new feature or update existing features.

On user side, current live version completed features included: Map, Adventure Track, Learn as you Go, Identify and My ParkApps. Completed features user interface and business logic are finalized unless new research requirements came up. Citizen Science is under developing, developing branch version is available on TestFlight. On research side, current live version will be able to collection user answer question status and user position report anonymously. Due application will update content in Adventure Tracks and Citizen Science, therefore user will get latest content in application.

5.2. Future plans

Offline map currently is manually update by updating offline package file. Ideally is when application connect to Internet, map service will begin caching latest map tiles then
package latest tiles into offline package. To implement this feature, we will deeply customize MapBox framework and make big impact on map related features.

Currently in mobile application, images display and cache system is isolated in separated function such as Adventure Tracks, POIs images, Badges Images. When business logic needs changing and related with image display components, all related images codes will be impacted. Therefore, in future we need uniform images upload and cache system built in application. Uniform images upload and cache system will benefit for developer rapid coding, and make application more robustness.

User answer questions in application statics feature. Currently, all user answer questions data will be upload to server side to be analyzed. Application will not be able to give real-time question answering status to user. By adding visualization components libraries in application and adding question answering statistics service, will able give user real-time in application question answering dashboard.

5.3. **Final words**

In this research project, we implement mobile application including iOS and Android version and web-based server end management system. Mobile application now able visualized most of educational content and represent correct information for park users. At application background, it will be able to collection user information. On web-based server backend, managers will be able analysis user activity in park and do research on users question answers. The whole system is keeping update according new research requirements or HCI (Human-computer interaction) design.
REFERENCES


