<table>
<thead>
<tr>
<th>Title</th>
<th>Context-aware life memorandum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Lim, Hiu Lung Allen (林曉龍)</td>
</tr>
<tr>
<td>Citation</td>
<td>Lim, H. L. A. (2012). Context-aware life memorandum (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.</td>
</tr>
<tr>
<td>Issue Date</td>
<td>2012</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2031/6750">http://hdl.handle.net/2031/6750</a></td>
</tr>
<tr>
<td>Rights</td>
<td>This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.</td>
</tr>
</tbody>
</table>
Project Code: 11CS070

Context-Aware Life Memorandum

(Volume 1 of 1)

Student Name : Lim Hiu Lung Allen
Student No. : 
Programme Code : BSCCS

Supervisor : Dr. Ng Tsz Hin, Sam
1st Reader : Dr. Leung Wing Ho, Howard
2nd Reader : Dr. Yu Yuen Tak
Context-aware Life Memorandum

LIM Hiu Lung Allen
Final Year Project Report

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

__________________________________________ Context-Aware Life Memorandum

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: Lim Hiu Lung Allen       Signature: ________________________________

Student ID: ___________________________ Date: 10 April 2012
Abstract

“Life is not what you live, but what you remember”, said by Gabriel Garcia Marquez. Most people tried to write diary when they were young. As they grow up, they may abandon this habit due to busy life style in modern society. However, the value of life recording is inevitably beneficial. This project aims to facilitate people in modern city to record their daily lives with their smart phone and provide a convenient tool for memory exploration.

To achieve the project objective, an iPhone application called iLifeMemo is developed. It is a context-aware application that possesses the following features:

- Automatic recording places visited by user using technologies and algorithm such as GPS positioning, geocoding services, Bayesian Network, etc.
- Innovative life memo exploration using augmented reality.
- Building collective memory based on life memos written by different users so that they can explore famous events happened in the past.
- Fundamental life memo functions which include text editing, photo attachment, tagging and geotagging

As iLifeMemo is developed as an iPhone application, xcode 4.3 is used to develop an iOS 5 mobile application. The application targets iPhone series from 3GS to iPhone 4S with iOS 5.0.1 or above. Since mobile data network are expensive and iLifeMemo is a context-aware application such that many operations are done automatically, the data network will be carefully monitored to avoid unnecessary network overhead. As battery life of mobile device is limited, memory control is carefully performed to prevent unnecessary memory usage as well as memory leakage. Processes that involve large computation power are held in server side to reduce battery consumption in iPhone.
Acknowledgement

This project is accomplished with a plenty of helpful support and assistance from different people. I would like to take this opportunity to give my genuine gratitude to them.

Firstly, I am wholeheartedly thankful to my supervisor, Dr. Ng Tsz Hin, Sam, who spends remarkable amount of time to discuss with me, offering guidance and support. Without his assistance, this project could hardly be completed.

I would also thank my classmates who share their experience and providing support throughout the whole year. Their accompany offers me with confidence and endless energy in working on this project.

I would also thank to staffs who tried their best in providing a good working environment in computer science project laboratory. Their effort must never be missed.
# Table of Contents

1. **Project Introduction** .................................................................................................................................. 10
   1.1. **Background Information** .................................................................................................................... 10
   1.2. **Existing Solutions and Applications** .................................................................................................. 10
       1.2.1. Context-Aware Diary Builder Model ................................................................................................. 10
       1.2.2. Location-Aware Application ............................................................................................................. 10
   1.3. **Incorporating Augmented Reality with Life Memorandum** ................................................................. 11
   1.4. **Collective Memory** ............................................................................................................................ 12
   1.5. **Project Objective** .................................................................................................................................. 12
   1.6. **Project Schedule** .................................................................................................................................. 13

2. **Literature Review** ........................................................................................................................................ 14
   2.1. **Application of Different Technologies** ................................................................................................. 14
       2.1.1. Application of Context-aware Computing .......................................................................................... 14
       2.1.2. Application of Augmented Reality .................................................................................................... 14
   2.2. **Place Recognition in Smartphone** ...................................................................................................... 15
       2.2.1. Accuracy of Different Positioning methods in Smartphone ..................................................................... 15
       2.2.2. Segmentation of GPS Coordinates .................................................................................................... 15
       2.2.3. Place Recognition by Geocoding Service .......................................................................................... 15
       2.2.4. User-defined Place Recognition ......................................................................................................... 16
       2.2.5. Learning-based Place Recognition ..................................................................................................... 16
       2.2.6. Integration of Different Place Recognition Approaches ...................................................................... 16
   2.3. **Comparison with Existing Application** ................................................................................................. 17
       2.3.1. aNote .................................................................................................................................................. 17
       2.3.2. Momento ........................................................................................................................................... 18
       2.3.3. Google+ ............................................................................................................................................. 18
       2.3.4. Summary of Comparison ................................................................................................................... 19
   2.4. **Summary of Literature Review** ............................................................................................................ 19

3. **System Design and Solution** ..................................................................................................................... 20
   3.1. **System Overview** ................................................................................................................................... 20
   3.2. **Object-Oriented Analysis** ..................................................................................................................... 21
       3.2.1. Identifying Actors and their goals ....................................................................................................... 21
       3.2.2. Model the Actions into Use Case ........................................................................................................ 22
       3.2.3. Reasons for Modeling the Actors and Use Cases Mentioned .......................................................... 23
       3.2.4. Use Case Model in UML .................................................................................................................. 24
4. **Detailed Methodologies and Implementation** .......................................................... 50

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.5. Use Case Summary</td>
<td>25</td>
</tr>
<tr>
<td>3.2.6. Main Courses of Events in Use Case</td>
<td>26</td>
</tr>
<tr>
<td>3.2.7. Alternative Courses of Use Cases</td>
<td>31</td>
</tr>
<tr>
<td>3.2.8. Object Modeling</td>
<td>32</td>
</tr>
<tr>
<td>3.2.9. Analysis Class Diagram from Object Modeling</td>
<td>33</td>
</tr>
<tr>
<td>3.3. <strong>Object-Oriented Design</strong></td>
<td>34</td>
</tr>
<tr>
<td>3.3.1. Model-View-Controller pattern</td>
<td>34</td>
</tr>
<tr>
<td>3.3.2. Boundary-Controller-Entity</td>
<td>35</td>
</tr>
<tr>
<td>3.3.3. Modeling Design Classes</td>
<td>35</td>
</tr>
<tr>
<td>3.3.4. High-Level Object Interactions</td>
<td>36</td>
</tr>
<tr>
<td>3.3.5. Design Class Diagram</td>
<td>43</td>
</tr>
<tr>
<td>3.4. <strong>Database Modeling</strong></td>
<td>48</td>
</tr>
<tr>
<td>3.4.1. Normalization</td>
<td>48</td>
</tr>
<tr>
<td>3.4.2. Entity Relationship Diagram</td>
<td>49</td>
</tr>
</tbody>
</table>

4. **Detailed Methodologies and Implementation** .......................................................... 50

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1. Development Tools</td>
<td>50</td>
</tr>
<tr>
<td>4.2. Data Accessing Process in Client-Server Architecture</td>
<td>51</td>
</tr>
<tr>
<td>4.2.1. Data Encapsulation using JSON for Transmission</td>
<td>51</td>
</tr>
<tr>
<td>4.2.2. Mechanism for Data Transmission</td>
<td>52</td>
</tr>
<tr>
<td>4.3. Multiple Asynchronous Web Requests in Client-Server Architecture</td>
<td>53</td>
</tr>
<tr>
<td>4.3.1. Concurrent Access of Data Accessing Objects</td>
<td>53</td>
</tr>
<tr>
<td>4.3.2. Concurrent Retrieval of Images in Web Server</td>
<td>54</td>
</tr>
<tr>
<td>4.4. Identify Current Location of User</td>
<td>56</td>
</tr>
<tr>
<td>4.4.1. CoreLocation Framework</td>
<td>56</td>
</tr>
<tr>
<td>4.4.2. Screening out Unacceptable Location due to Positioning Error</td>
<td>56</td>
</tr>
<tr>
<td>4.5. Resolving Concurrent Access to Location Service in iPhone</td>
<td>57</td>
</tr>
<tr>
<td>4.5.1. Singleton - Location Manager Service in iPhone</td>
<td>57</td>
</tr>
<tr>
<td>4.5.2. Preemptive Multi-Threading of Location Manager</td>
<td>58</td>
</tr>
<tr>
<td>4.6. Display Location Information with MapKit</td>
<td>59</td>
</tr>
<tr>
<td>4.6.1. Mapkit Framework</td>
<td>59</td>
</tr>
<tr>
<td>4.6.2. Adding Annotation on Map View</td>
<td>60</td>
</tr>
<tr>
<td>4.7. Place Recognition</td>
<td>62</td>
</tr>
<tr>
<td>4.7.1. Translating GPS Coordinates with Google Place API</td>
<td>62</td>
</tr>
<tr>
<td>4.7.2. Translating GPS Coordinates to User-Defined Location</td>
<td>64</td>
</tr>
<tr>
<td>4.8. Automatic Recording User Location</td>
<td>65</td>
</tr>
</tbody>
</table>
4.8.1. Factors Hindering Automatic Location Recording ......................................................... 65
4.8.2. Automatic Recording Toggle ......................................................................................... 67
4.8.3. Determination of Closest District .................................................................................. 68
4.8.4. Determining Closest Location with Bayesian Network .................................................. 69
4.8.5. Resolving Incorrect Location Record ............................................................................. 70

4.9. Exploring Personal Life Memo using Augmented Reality Browser ............................... 71
   4.9.1. Geotagged Life Memo ................................................................................................. 71
   4.9.2. Mechanism for Personal Life Memo Exploration ......................................................... 72
   4.9.3. Augmented Reality Browser Engine ............................................................................ 74

4.10. Explore Collective Memory .............................................................................................. 78
   4.10.1. Notion of Collective Memory ..................................................................................... 78
   4.10.2. Analysis of Collective Memory Based on Memo Tags ................................................ 78
   4.10.3. Mechanism for Collective Memory Exploration ........................................................ 79
   4.10.4. Statistical Analysis on Memo Tags ............................................................................ 80

4.11. Performance Considerations ............................................................................................. 81
   4.11.1. User-driven Data Retrieval ....................................................................................... 81
   4.11.2. Image Compression .................................................................................................. 82
   4.11.3. Caching Images to Reduce Network Overhead ......................................................... 83
   4.11.4. Memory Control ...................................................................................................... 83
   4.11.5. Quick Sort of Memo Tags for Counting Process in Statistical Analysis .................... 83

5. Experimental Results and Evaluation .................................................................................. 84
   5.1. Field Test ....................................................................................................................... 84
      5.1.1. Automatic Location Recording ............................................................................... 84
   5.2. Battery Consumption Test ............................................................................................ 88
   5.3. Memory Test ................................................................................................................ 89

6. Discussion and Conclusion .................................................................................................. 90
   6.1. Achievement ................................................................................................................ 90
      6.1.1. Project Achievement ............................................................................................... 90
      6.1.2. Technical Achievement ........................................................................................... 91
      6.1.3. Personal Achievement ............................................................................................ 91
   6.2. Difficulties Encountered ................................................................................................ 91

7. Future Enhancement .......................................................................................................... 92
1. Project Introduction

1.1. Background Information

Keeping a life memorandum gives people astonishing benefits. It helps you to clarify your goals, learn about yourself and strengthen your relationships (Humbert, 2003). Despite there are tons of benefits, writing life memorandum takes time and people may not be willing to spend time on it. For instance, students may be busy with assignments while businessmen may need to work overtime and feel exhausted after duty. Moreover, travelers may spend most of their time to play and go shopping so that no time is left for writing life memo. If there is a technology that can automatically capture people’s daily life, then everyone can benefit from keeping their life memos.

1.2. Existing Solutions and Applications

1.2.1. Context-Aware Diary Builder Model

In a novel subject namely pervasive computing, Context-Aware Diary Builder (CADB) model was proposed to minimize the effort of diary writing (Addis Ababa University, 2009). Content of a life memo such as user activities and emotions are detected by sensors and actuators in smart environment to form a diary automatically. With CADB model, people do not need to spend time on manual diary writing which is obviously an optimal solution. Nevertheless, a major shortcoming of the CADB model is smart environment oriented. Apparently, it is impossible to install sensors and actuators everywhere. In other words, the CADB model is infeasible to be implemented with today’s technology limitation.

1.2.2. Location-Aware Application

Nowadays, smartphone like iPhone and Android phones are always equipped with GPS function (Apple Inc., 2010; Samsung Electronic Co. Ltd., 2010). It implies that geographical information of smartphone users can be obtained to form personal location records. For example, location-based service such as Google Latitude can automatically record places visited by users (LaPennna, 2011). However, traveling is only a subset of daily activities which is incomprehensive to be used alone for life recording.
1.3. **Incorporating Augmented Reality with Life Memorandum**

Life memorandum is a memo-alike diary for people to share their memories. It is a proper tool to learn from experience and carry out self-reflection (Bolton, 2005). It is obviously beneficial if we read our life memos. Traditionally, people record their life by writing online diary, blog or using mobile diary application. These services provide searching function so that user can search diary content based on inputted keywords. Nonetheless, the more life memos we have, the higher chances we may forget the memo content and we may not be able to find a keyword for searching. Instead of searching life memo with keywords, it is better to have an “exploration tool” so that user can read the old memos without any manual input. Since augmented reality browser offers a way to search information related to user’s current location (Layar, 2009), it is applicable in this project such that users can find meaningful content from a stack of life memos based on their current locations.

*Fig 1.* Augmented Reality browser application showing information of ATM nearby
1.4. Collective Memory

As stated in previous section, life memos written in the past is valuable and the value is not only personal. By analysis on the content, it is possible to find out collective memory happened in the past. The terminology of collective memory may seem a little bit arbitrary and subjective to different people. Generally speaking, collective memory should be something common and popular among a group of people. It can be an event such as a birthday party, or a controversy issue like the election of Hong Kong Chief Execution Election. This kind of events may be captured by life memos written by different people and is valuable to everyone.

1.5. Project Objective

In general, this project aims to encourage people to build their own life memorandums plus facilitate memory exploration by using a context-aware iPhone application. More specifically, this project has three distinctive goals. First, in order to assist the process of life memory recording, the system is capable of automatic recording places visited by user. The location information is useful to remind user about what should be recorded in that day. Second, in order to provide a convenient life memory exploration tool, an augmented reality browser is developed to search for personal memos with regards to user’s current location. Finally, the project also aims to build up collective memories among users so that they can learn from the past memos written by each other.
1.6. Project Schedule

The project starts from 1st September 2011 and ends at 21st April 2012. In order to perform good time management, required project activities and their durations are identified. Then, the relationship between project activities (e.g. predecessors) is found and critical path of this project is generated automatically using Microsoft Project 2007. To avoid running behind schedule, some project activities such as Design and Implementation is buffered with 5 days delay.

Fig 2. Gantt Chart showing Project Activities and Schedule
2. Literature Review

In this section, I will first discuss different technologies that will be applied in this project. In particular, I will explain automatic life memo generation by using context-aware computing and an adoption of augment reality to facilitate content searching. After that, I will discuss deeply on an issue of place recognition in smartphone to perform automatic user location recording. Finally, I will show a comparison between existing applications and this project.

2.1. Application of Different Technologies

2.1.1. Application of Context-aware Computing

In Dey (2001), the term “context” is introduced to mean different characteristics of a situation that describes an entity. The characteristics include identity (who), activity (what), time (when) and location (where) which are all essential components in a typical life memo. In mobile application, identity and location can be gathered by determining physical location of application users with place recognition algorithms. However, user activities are difficult to be inferred because they often require sensor and actuator as suggested in Context-Aware Diary Builder (CADB) model. Although some context-aware applications such as Nokia Sensation can detect user situations like “in a meeting”, “sleeping” and “watching TV” (Nokia Beta Labs, 2010), they are hardly feasible to apply in life memo since there are uncountable situations in real life. Hence, user activities should be recorded using photos or texts inputted by user.

2.1.2. Application of Augmented Reality

Azuma (1997) defines augmented reality as “merging real and virtual, interactive in real time and registered in 3D”. Apart from creating virtual 3D objects in real environment by using camera, augmented reality browser is another popular application in the field that overlays the real world with digital information. For instance, an iPhone application called Layar permits users to locate a specific target such as ATM and McDonald using camera view. The idea of augmented reality browser can be brought into this project for searching geotagged life memos.
2.2. Place Recognition in Smartphone

2.2.1. Accuracy of Different Positioning methods in Smartphone

Three positioning technologies including Assisted GPS (A-GPS), WiFi positioning and cellular network positioning have been integrated to iPhone product series starting from iPhone 3G. Research reveals that accuracy of fore-mentioned approaches is less accurate than regular GPS device (Zandbergen, 2009). For example, median deviation in accuracy between smartphone and regular GPS device ranges from 8 meters (with A-GPS) to 600 meters (with cellular network positioning). Inevitably, we cannot identify the exact location of a person even with 8 meters deviation. Therefore, some methodologies must be developed to investigate possible locations of a person.

2.2.2. Segmentation of GPS Coordinates

To deal with GPS localization errors in smartphone, segmentation and clustering algorithms can be utilized to classify GPS coordinates into different clusters to ease place recognition (Hightower et al., 2005). In this way, user location can be deduced from a list of related GPS coordinates within the same cluster, which is a suitable method to be adopted in this project.

2.2.3. Place Recognition by Geocoding Service

With a list of GPS coordinates that represents possible locations of a user, place recognition can be achieved by translating the coordinates into street addresses with geocoding services such as Microsoft MapPoint or Google Geocode API. Street addresses are semantically more meaningful than GPS coordinate, yet they are still not intuitive enough. When people indicate a particular place, say City University of Hong Kong, they use the name directly instead of its address. Thus, geocoding service is a promising choice to locate people in district level but is insufficient to be used alone in place recognition.
2.2.4. User-defined Place Recognition

Beside of business name people also indicate different places with their own semantic labels such as “home” or “project lab”. These names can never be detected without previous input by user. Moreover, if place names are only generated by system automatically, they are possibly outdated as local businesses are changeable. As a result, place recognition should also involve user-defined place. These places are either semantically meaningful name to oneself or simply an updated business name.

2.2.5. Learning-based Place Recognition

Different places have various degrees of relevance to different people at different time. For example, a student from City University of Hong Kong may find “Lecture Theatre 1” more relevant than “Heung To Middle School” in a list of nearby places. One may also find “Home” more relevant than “Neway Karaoke” at 8:00am on weekdays. To distinguish the degree of relevance of different places, Bayesian networks can be applied to analyze historical user data with consideration of time, place and category of place. The drawback of learning-based algorithm is that user location history must be collected in advance. It hints that this approach is not applicable to new users.

2.2.6. Integration of Different Place Recognition Approaches

Considering all fore-mentioned approaches, it can be problematic if only a particular approach is employed in place recognition since each of them possesses certain drawbacks and are applicable in different situations. Therefore, an integration of different approaches is necessary to improve the accuracy of place recognition process.
2.3. Comparison with Existing Application

2.3.1. aNote

aNote is an iPhone application that allows user to create mobile diary with paragraphs, photos and maps. Users must input all content manually and it does not have any place recognition feature and diary searching function. In contrary, system being developed in this project can automatically record user locations as well as providing a memory exploration tool named augmented reality browser.

Fig 3. Diary Entries of aNote with embedded photo and map
2.3.2. Momento

Momento is a mobile diary that contains automatic place recognition. The system is capable of finding address of user’s current location but location name must be inputted by user manually. An even worse issue is that location name inputted previously by user cannot be retrieved automatically afterward which leads to inconvenience to application users. On the other hand, since this project handles place recognition using different approaches, it ensures that there must be a list of relevant places shown to user.

2.3.3. Google+

Google+ is a mobile application that allows users to upload and share geotagged memo with their friends. Although place recognition is done to attach location name directly to memo content, the tool does not support automatic life recording which is the most significant issue tackled by this project. Moreover, memo in Google+ are not classified by date and are shown chronologically. There are no ways to efficiently find out memo of a particular place. In this project, life memo are geotagged and can be retrieved based on user’s current location.
2.3.4. Summary of Comparison

The three applications mentioned previously represent typical life memo in mobile market. Among fore-mentioned applications, mobile diary such as aNote and Momento do not possess good place recognition feature. Although social networking tool such as Google+ and Facebook offer excellent place recognition, they does not provide memory exploration tool for user to review meaningful memos written in the past.

<table>
<thead>
<tr>
<th></th>
<th>aNote</th>
<th>Momento</th>
<th>Google+</th>
<th>This project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geotagged memo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Automatic place visited recording</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Personal memo exploration by</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Augmented Reality Browser</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Collective memory exploration</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

2.4. Summary of Literature Review

This project takes advantage of context-aware computing and augmented reality technology to automate life memory recording and exploration. Different place recognition approaches such as GPS coordinate segmentation, geocoding services, web scraping and learning-based algorithm will also be integrated to improve the accuracy of place recognition process. The place identified will be recorded by the system automatically so that user can use the information in memo generation afterwards. Moreover, augmented reality will be adopted to provide an easy way for memo exploration. Finally, this project also builds up collective memories among users so that they can learn from life memos written by each other.
3. System Design and Solution

3.1. System Overview

Basically the system consists of an iPhone application and a back-end server that contains database and web services. This is a client-server architecture such that the iPhone application “iLifeMemo” is connected to a remote web server via wireless network. On one hand, the iPhone application serves as a front-end interface that process user input and display output. Some of the domain logics such as positioning user and photo processing are embedded in the front-end program as the operations involve native functions in iPhone. On the other hand, the back-end server possesses a web service layer that manipulates user data and controls the access to database MSSQL with a list of data accessing objects (DAO). The dispatcher in the below diagram is responsible to direct the request sent by user to the correct DAO in order to retrieve corresponding data.

Fig 4. An overview of system architecture
3.2. Object-Oriented Analysis

In this project, object-oriented analysis (OOA) has been conducted to transform system requirements to system features that needed to be implemented. First, requirements are captured using use case modeling. Second, analysis class diagram is created to consolidate the requirements by organizing the identified attributes and classes. After the whole process, system requirements are well studied to ease further system design using object-oriented design (OOD) process.

3.2.1. Identifying Actors and their goals

According to project objectives, related actors and their business objectives are identified in the following table:

<table>
<thead>
<tr>
<th>Actors</th>
<th>Main Business Objectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Add life memo</td>
</tr>
<tr>
<td></td>
<td>Automatic recording place visited</td>
</tr>
<tr>
<td></td>
<td>Explore personal life memo</td>
</tr>
<tr>
<td></td>
<td>Explore collective life memo</td>
</tr>
<tr>
<td>Visitor</td>
<td>Register</td>
</tr>
</tbody>
</table>
3.2.2. Model the Actions into Use Case

Based on business objectives obtained from different actors, 6 use cases are identified:

1. Add life memo
2. Automatic recording place visited
3. Explore personal life memo using augmented reality browser
4. Explore collective life memo
5. Register

Table 3. Summary of valid actors

<table>
<thead>
<tr>
<th>Actors</th>
<th>Goals</th>
<th>Involved Use Case</th>
<th>Primary/Participating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visitor</td>
<td>Register as a member</td>
<td>Register</td>
<td>Primary</td>
</tr>
<tr>
<td>User</td>
<td>Record daily life for future revision</td>
<td>Add life memo</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td>Automatic recording place visited</td>
<td></td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td>Explore personal life memo using augmented reality browser</td>
<td></td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td>Explore collective life memo</td>
<td></td>
<td>Primary</td>
</tr>
</tbody>
</table>
3.2.3. Reasons for Modeling the Actors and Use Cases Mentioned

Visitor – Before using any services provided by the system, it is necessary for people to register an account so that the system can distinguish writers of different life memos. Since life memos can be analyzed to form collective memory among specific group of users, the identity of user is important to permit the system analyzing on the correct memos. Thus, it is a must for all visitors to register an account and they are considered as important actors for use case “Register”.

User – User is important to be modeled since it is a key participant of the system. For instance, user will invoke many significant use cases such as “Add life memo”, “Automatic recording place visited”, “Explore personal life memo using augmented reality browser” and “Explore collective life memo”.

According to the fore-mentioned project objectives, use case “Add life memo” is most basic functionality that must be offered by the system to record daily life of users. Since the project aims at assisting life memory recording, use case “Automatic recording place visited” is an important feature that must be included in the system. Moreover, in order to provide a convenient life memory exploration tool, life memos can be geotagged with GPS coordinate to ease life memo exploration using augmented reality browser. Therefore, “Explore personal life memo using augmented reality browser” is a mandatory use case. In addition, since life memo is a proper tool to learn from experience and carry out self-reflection (Bolton, 2005), it will be much better if users can learn from life memos written by each other. In other words, the system must be able to build collective memory by analyzing life memos written by users. Thus, use case “Explore collective life memo” is required to be modeled.
3.2.4. Use Case Model in UML

An overview of relationships between actors and use cases is revealed in below figure to illustrate major user requirements of this project.

Fig 5. Use-case diagram of the system
### 3.2.5. Use Case Summary

Table 4. Summary of Use Cases

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1: Add life memo</td>
<td>Users can create memo by inputting text content to record their daily life</td>
</tr>
<tr>
<td>UC2: Attach photo</td>
<td>When users are creating or editing a memo, they can attach photo by taking a new photo using iPhone camera or selecting an existing photo from camera rolls</td>
</tr>
<tr>
<td>UC3: Attach location tag</td>
<td>When users are creating or editing a memo, they can attach a location tag to indicate a particular place for that memo</td>
</tr>
<tr>
<td>UC4: Explore personal life memo using augmented reality browser</td>
<td>A memory exploration tool named “Augmented Reality Browser” will be implemented so that user can explore life memos written in the past based on his current location</td>
</tr>
<tr>
<td>UC5: Automatic recording place visited</td>
<td>The system can automatically record place visited by user with compliance to certain predefined rules</td>
</tr>
<tr>
<td>UC6: Explore collective life memo</td>
<td>The system can analyze life memos written by different users to find important events happened in the past. These important events contribute to so-called “collective memory”. Users can explore “collective memory” in form of collective life memos written by each other.</td>
</tr>
<tr>
<td>UC7: Register</td>
<td>Visitor register an account via the system</td>
</tr>
</tbody>
</table>
### 3.2.6. Main Courses of Events in Use Case

Table 5. UC1 Add life memo

<table>
<thead>
<tr>
<th>UC1 Add life memo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primary Actor:</strong> User</td>
</tr>
<tr>
<td><strong>Stakeholders and Interests:</strong></td>
</tr>
<tr>
<td>• User: want to add a new life memo</td>
</tr>
<tr>
<td><strong>Preconditions:</strong></td>
</tr>
<tr>
<td>• N/A</td>
</tr>
<tr>
<td><strong>Success Guarantee (Post-conditions):</strong></td>
</tr>
<tr>
<td>• An new life memo will be saved into database in remote server</td>
</tr>
<tr>
<td><strong>Main Success Scenario:</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actor Action and Intention</th>
<th>System Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: User adds a new life memo by writing some texts</td>
<td></td>
</tr>
<tr>
<td>Extension: 1b. Attaching photo: Include Attach photo</td>
<td>Step 2: The system verifies the texts inputted</td>
</tr>
<tr>
<td>1c. Attaching location tag: Include Attach location tag</td>
<td>Step 3: The system stores the life memo into database in remote server</td>
</tr>
</tbody>
</table>
UC4 Explore personal life memo using augmented reality browser

<table>
<thead>
<tr>
<th>Primary Actor:</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholders and Interests:</td>
<td>• User: want to explore personal life memos that are tagged with his current location</td>
</tr>
<tr>
<td>Preconditions:</td>
<td>• Some personal life memos related to user's current location are stored by the system in advance</td>
</tr>
<tr>
<td>Success Guarantee (Post-conditions):</td>
<td>• A list of life memos that are tagged with user's current location are retrieved</td>
</tr>
<tr>
<td>Main Success Scenario:</td>
<td></td>
</tr>
<tr>
<td>Actor Action and Intention</td>
<td>System Responsibility</td>
</tr>
<tr>
<td>Step 1: User opens the augmented reality browser and holds iPhone camera forward and moves it around slowly</td>
<td></td>
</tr>
<tr>
<td>Step 2: The system examines the current location of user (e.g. GPS coordinate)</td>
<td></td>
</tr>
<tr>
<td>Step 3: The system retrieves related life memos within a particular range</td>
<td></td>
</tr>
<tr>
<td>Step 4: The system displays a list of life memos to user for selection</td>
<td></td>
</tr>
<tr>
<td>Repeat step 1-4 until target memo is found</td>
<td></td>
</tr>
<tr>
<td>Step 5: User selects a particular life memo to view the content</td>
<td></td>
</tr>
</tbody>
</table>
Table 7. UC5 Automatic recording place visited

<table>
<thead>
<tr>
<th>UC5 Automatic recording place visited</th>
</tr>
</thead>
</table>

**Primary Actor:** User

**Stakeholders and Interests:**
- User: want to write life memo with assistance of their location records

**Preconditions:**
- N/A

**Success Guarantee (Post-conditions):**
- A list of location is recorded by the system automatically. User can create life memo based on the locations recorded.

**Main Success Scenario:**

<table>
<thead>
<tr>
<th>Actor Action and Intention</th>
<th>System Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: User turns on automatic check-in function</td>
<td>Step 2: The system examines the current location of user such as his GPS coordinate (longitude and latitude) periodically</td>
</tr>
<tr>
<td>Step 4: User creates a life memo by selecting one of the location record</td>
<td>Step 3: The system records all locations of user into a list</td>
</tr>
<tr>
<td></td>
<td>Step 5: The system stores the life memo into remote server</td>
</tr>
</tbody>
</table>
Table 8. UC6 Explore collective life memo

<table>
<thead>
<tr>
<th>UC6 Explore collective life memo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Primary Actor:</strong> User</td>
</tr>
<tr>
<td><strong>Stakeholders and Interests:</strong></td>
</tr>
<tr>
<td>• User: want to explore collective memory in form of collective life memos written by different users</td>
</tr>
<tr>
<td><strong>Preconditions:</strong></td>
</tr>
<tr>
<td>• Some life memos are stored by the system</td>
</tr>
<tr>
<td><strong>Success Guarantee (Post-conditions):</strong></td>
</tr>
<tr>
<td>• A list of collective life memos is shown to user</td>
</tr>
<tr>
<td><strong>Main Success Scenario:</strong></td>
</tr>
<tr>
<td><strong>Actor Action and Intention</strong></td>
</tr>
<tr>
<td>Step 1: User opens the augmented reality browser and holds iPhone camera forward and moves it around slowly</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
### Table 9. UC7 Register

<table>
<thead>
<tr>
<th><strong>UC7 Register</strong></th>
</tr>
</thead>
</table>

**Primary Actor:** Visitor

**Stakeholders and Interests:**
- Visitor: want to register an account

**Preconditions:**
- N/A

**Success Guarantee (Post-conditions):**
- An unique account is created such that the visitor is able to use the system

**Main Success Scenario:**

<table>
<thead>
<tr>
<th>Actor Action and Intention</th>
<th>System Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: Visitor inputs account name and password</td>
<td></td>
</tr>
<tr>
<td>Step 2: The system verifies the account name is not taken by others and the password consists of correct length and characters</td>
<td></td>
</tr>
<tr>
<td>Step 3: The system creates an account for the user based on the login information inputted</td>
<td></td>
</tr>
<tr>
<td>Step 4: Visitor receives the right to use the system</td>
<td></td>
</tr>
</tbody>
</table>
3.2.7. Alternative Courses of Use Cases

Below are some alternative courses for fore-mentioned use cases.

Table 10. Alternative courses of event for UC1

<table>
<thead>
<tr>
<th>UC1 Add life memo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1b. User attaches an photo</strong></td>
</tr>
<tr>
<td>1. User selects a photo from camera roll or by taking a new photo</td>
</tr>
<tr>
<td>2. System attach the photo to the life memo</td>
</tr>
<tr>
<td>3. Go to step 1</td>
</tr>
<tr>
<td><strong>1c. User attaches an location tag</strong></td>
</tr>
<tr>
<td>1. System retrieves a list of nearby places</td>
</tr>
<tr>
<td>2. User selects a place and attaches it to the life memo</td>
</tr>
<tr>
<td>3. Go to step 1</td>
</tr>
<tr>
<td><strong>2b. User inputs empty texts</strong></td>
</tr>
<tr>
<td>1. System asks user to input some texts</td>
</tr>
<tr>
<td>2. Go to step 1</td>
</tr>
</tbody>
</table>

Table 11. Alternative courses of event for UC5

<table>
<thead>
<tr>
<th>UC5 Automatic recording place visited</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4b. User delete the locations recorded</strong></td>
</tr>
<tr>
<td>1. System deletes the location recorded</td>
</tr>
<tr>
<td>2. Go to step 2</td>
</tr>
<tr>
<td><strong>4c. User select other possible locations</strong></td>
</tr>
<tr>
<td>3. System retrieves other possible locations based on the past location of user</td>
</tr>
<tr>
<td>Go to step 2</td>
</tr>
</tbody>
</table>

Table 12. Alternative courses of event for UC7

<table>
<thead>
<tr>
<th>UC7 Register</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1b. Visitor inputs inappropriate login information</strong></td>
</tr>
<tr>
<td>1. System rejects the inputs</td>
</tr>
<tr>
<td>2. Go to step 1</td>
</tr>
</tbody>
</table>
3.2.8. **Object Modeling**

Based on the result of use case modeling, potential objects and attributes can be found by object modeling.

<table>
<thead>
<tr>
<th>Potential List of Objects</th>
<th>Proposed (Yes/No)</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visitor</td>
<td>No</td>
<td>Actor with no attribute to be stored</td>
</tr>
<tr>
<td>User</td>
<td>Yes</td>
<td>“User”</td>
</tr>
<tr>
<td>Memo</td>
<td>Yes</td>
<td>“Memo”</td>
</tr>
<tr>
<td>Texts</td>
<td>No</td>
<td>Attribute of “Memo”</td>
</tr>
<tr>
<td>Photo</td>
<td>No</td>
<td>Attribute of “Memo”</td>
</tr>
<tr>
<td>Location tag</td>
<td>No</td>
<td>Attribute of “Memo”</td>
</tr>
<tr>
<td>Augmented reality browser</td>
<td>No</td>
<td>System component without attribute to be</td>
</tr>
<tr>
<td>Current location of user</td>
<td>No</td>
<td>No need to be stored by the system</td>
</tr>
<tr>
<td>Range</td>
<td>No</td>
<td>Attribute of “SearchSetting”</td>
</tr>
<tr>
<td>Visited place</td>
<td>Yes</td>
<td>“PlaceVisited”</td>
</tr>
<tr>
<td>Location list</td>
<td>Yes</td>
<td>“PendingLocation”</td>
</tr>
<tr>
<td>Important event</td>
<td>No</td>
<td>Vague</td>
</tr>
<tr>
<td>Collective memory</td>
<td>No</td>
<td>Vague</td>
</tr>
<tr>
<td>Account name</td>
<td>No</td>
<td>Attribute of “User”</td>
</tr>
<tr>
<td>Password</td>
<td>No</td>
<td>Attribute of “User”</td>
</tr>
<tr>
<td>Past location</td>
<td>No</td>
<td>Attribute of “PendingLocation”</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional Attributes</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place name</td>
<td>Necessary for indicating the name of a place</td>
</tr>
<tr>
<td>Location</td>
<td>Necessary for indicating the GPS coordinate of a place</td>
</tr>
<tr>
<td>Visited date and time</td>
<td>Necessary for indicating the visited date and time</td>
</tr>
</tbody>
</table>
3.2.9. Analysis Class Diagram from Object Modeling

An overview of relationships between different entity classes is shown in the below class diagram. Identified attributes for different classes are also included.

Fig 6. Analysis class diagram of the system
3.3. Object-Oriented Design

In previous sections, use case modeling and object modeling were done to reveal the system features and basic design by adopting use case diagram and analysis class diagram. The relationship between use cases and actors are also delivered. Nonetheless, there are only entity classes in analysis class diagram which infers for an incomprehensive system design. Furthermore, object-oriented analysis does not provide any detail design strategy such as design pattern and object interaction. As a result, object-oriented design is conducted to assist system implementation systematically using a set of design standards for object communication, object behaviors, etc.

3.3.1. Model-View-Controller pattern

Model-View Controller (MVC) is a software architectural pattern that is highly recommended by Apple Inc. for iPhone application development. It is applied to isolate the domain logic from presentation layer and data access layer. This approach ensures the system will be developed with high maintainability since the actions performed by different classes are constrained by its role: a model represents a data storage, a view represents a presentation interface and a controller is responsible for domain logic operation.

Fig 7. MVC model for life memo processing
3.3.2. **Boundary-Controller-Entity**

As similar to Model-View Controller (MVC) pattern, objects in communication diagram are classified into boundary object, controller object and entity object. Boundary objects are interface that serves system functions to user. Controller objects are responsible for processing domain logic while entity objects represent encapsulated data storage.

According to design standard of this project, boundary classes are restricted to only communicate with controller classes. Any boundary-to-boundary communication or boundary-to-model communication is forbidden to insure the consistence in logical flow and data flow across different use cases. In addition, entity objects in communication diagram represent both database records and encapsulated data objects.

3.3.3. **Modeling Design Classes**

Based on the analysis class diagram released from object-oriented analysis, some new design classes are derived to facilitate the design process in later stage.

**Table 15. Design Class Categorization**

<table>
<thead>
<tr>
<th>Boundary Class</th>
<th>Control Class</th>
<th>Entity Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Memo View</td>
<td>Memo Controller</td>
<td>User</td>
</tr>
<tr>
<td>Attach Location View</td>
<td>Check-in Controller</td>
<td>Friend</td>
</tr>
<tr>
<td>Attach Photo View</td>
<td>Camera Controller</td>
<td>Memo</td>
</tr>
<tr>
<td>Camera</td>
<td>Pending Location</td>
<td>Place</td>
</tr>
<tr>
<td>Search Memo View</td>
<td>Search Memo Controller</td>
<td>Photo in Camera Rolls</td>
</tr>
<tr>
<td>Memo View</td>
<td>Share Memo Controller</td>
<td>Photo</td>
</tr>
<tr>
<td>Pending Location View</td>
<td>User Controller</td>
<td>User</td>
</tr>
<tr>
<td>Register View</td>
<td>User DAO</td>
<td>Pending Place</td>
</tr>
<tr>
<td></td>
<td>Memo DAO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Place DAO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pending Place DAO</td>
<td></td>
</tr>
</tbody>
</table>
3.3.4. High-Level Object Interactions

Fig 8. UC1 Add Life Memo

According to above communication diagram, interaction between different objects for scenario “Add Memo” is illustrated. Basically Add Memo View is a boundary object that accepts inputs from user and displays output to user. The Memo Controller is a controller object that handles logical and arithmetic operation with regard to memo editing. In addition, Memo DAO is a data-accessing object dedicated to control memo record management. It is connected to an entity object Memo that represents a memo record in database.

The figure above illustrated the flow of adding a new text memo in the system. First, user inputs some text in Add Memo View followed by issuing a save operation. The save memo request will pass to Memo DAO via Memo Controller so that Memo DAO will create a memo record in the database. The result of record creation will be returned from Memo DAO back to Memo Controller such that the controller will refresh the Add Memo View and notify user for the result of memo saving.
The above communication diagram illustrates the flow of attaching a photo to a memo. First, user issues an attach photo request in Add Memo View. The request is passed to Memo Controller that further creates a new Attach Photo View to show to the user. After that, user can select the build-in camera in Attach Photo View which triggers a display camera request to Camera Controller. In this way, a boundary object Camera will be created and displayed to the user. With the Camera view, user can take a photo using the iPhone camera. After a photo was taken, the Camera view will send a request to Camera Controller to save the photo in Camera Rolls. After saving the photo in Camera Rolls, the Camera Controller will attach the photo in the memo being editing by signaling Memo Controller. Finally, Memo Controller will update Add Memo View and notify user for the changes.
The above communication diagram illustrates the flow of attaching a location tag to a life memo. First, user issues an attach location command in Add Memo View. After that, the Add Memo View will invoke a Check-in Controller by requesting nearby places. The Check-in Controller will first determine user’s current location, followed by getting the exact place list in database (represented by Place object) via a data accessing object Place DAO. When a list of nearby places is gathered, the Check-in Controller will create an Attach Location View to display a list of nearby places to user. In the meantime, user can check-in a particular place in the Attach Location View. The check-in request will be propagated to Check-in Controller so that it will further signal the Memo Controller to attach a location tag using the check-in place selected. Finally, the Memo Controller will refresh Add Memo View to notify user for the changes.
According to above communication diagram, a user opens an augmented reality browser in Search Memo View which signal a search memo request to Search Memo Controller. The Search Memo Controller will then determine current location of user followed by calling Memo DAO to retrieve nearby memos. Since nearby memos may contain location tag, after Memo DAO gathered the memo record from Memo entity object, it will request Place DAO to get places that matched the location tag in the memos. Similarly, those memos may contain photo attachment so that Memo DAO may also request Photo DAO to get corresponding photos from remote server. After that, Memo DAO will return all nearby life memos back to Search Memo Controller. The Search Memo Controller will signal Search Memo View to refresh and display all nearby memos to user. At the moment, user can select a memo from a list of nearby memos in the Search Memo View. Once a memo is selected the Search Memo View will command the Search Memo Controller to display the memo content. In order to achieve this goal, the Search Memo Controller will create a Memo View Object to display the memo content.
First of all, when user turns on automatic check-in function in Pending Location View, a request will be forward to Check-in Controller to get user’s current location. When user location is identified, the Check-in Controller will signal Place DAO to get nearby places. Place DAO will get nearby places in system database as represented by the Place entity object. Once nearby places are collected, Place DAO will return them to Check-in Controller to determine possible location of the user. After the determination process, Check-in Controller will signal the Pending Location View to update its content in order to show a list of pending locations to the user. In the meantime, user can approve a pending place in the Pending Location View. This triggers a check-in request from Pending Location View to Check-in Controller. Then, the Check-in Controller will command Memo Controller to attach a location tag with the check-in place and save the memo in database via Memo DAO. Finally, the result of check-in approval will be sent back to Pending Location View from Memo DAO via Memo Controller and Check-in Controller.
According to the above communication diagram, user opens an augmented reality browser in Search Memo View which send a search memo request to Search Memo Controller. Then, the Search Memo Controller will determine current location of user and signals Memo DAO to retrieve nearby memos. As similar to other memo retrieval process, life memos that contains location tag or photos will request Place DAO and Photo DAO to get corresponding place and photo respectively. When all nearby life memos are collected by Search Memo Controller, the Search Memo Controller will determine which life memos are able to form collective memory and return the result back to Search Memo View for display. In the meantime, user can select a life memo from the list of collective life memos using Search Memo View. Once a memo is selected the Search Memo View will command the Search Memo Controller to display the memo content by creation of a Memo View.
The communication diagram above illustrates the flow of object interaction in UC7 Register. First, a visitor inputs an account name and password in Register View which will invoke a create user request to the User Controller object. The User Controller will then verify the account name and password to see if the account name is unique and both inputs are non-empty text. If the verification process is passed, the User Controller will signal User DAO to create a user record in the database which is represented by the User entity object. Finally, the result of user creation will be return back from User object to User Controller through User DAO. The User Controller will pass the result to Register View and show the result to visitor.
3.3.5. Design Class Diagram

Design class diagrams are composed by extracting functional calls declared in communication diagrams in previous section. There are several relationships between different classes. Basically, an unidirectional association implies that a class uses another class but not vice versa. For instance, only MemoDAO class will operate on Memo class to manage memo data. In contrary, a bidirectional association between two classes means that both of the classes make use of each other during their operation. For example, AddMemoView class will translate user request into corresponding functional call in MemoController class. After process user request, MemoController class will inform AddMemoView to refresh its screen data. Moreover, shared aggregation is used between User class and Memo class which indicates that memo objects are potentially aggregated by User class and/or other classes. Finally, association class Friend is held by User class itself to store extra information for association between two users.

Fig 15. Design class diagram for UC1 Add Life Memo
Fig 16. Design class diagram for UC2 Attach Photo
Fig 17. Design class diagram for UC3 Attach Location Tag

Fig 18. Design class diagram for UC4 Explore Personal Life Memo using Augmented Reality Browser
Fig 19. Design class diagram for UC5 Automatic Record Place Visited

Fig 20. Design class diagram for UC6 Explore Collective Life Memo
Fig 21. Design class diagram for UC7 Register
3.4. Database Modeling

3.4.1. Normalization

The fundamental task in database design is to come up with a relational scheme. Database normalization can effectively reduce the risk of getting data inconsistency. Nonetheless, the higher level the normalization, the more effort should be paid on data retrieval process since there will be more sub tables which lead to more table-joining operations. To explain decisions made in a tradeoff between minimizing data redundancy and table processing, below shows a list of normalization level considered during database design process in this project.

Definitions:

- 1 NF requires primary key.
- 2 NF requires no non-prime attribute dependent on candidate key.
- 3 NF requires non-prime attribute non-transitively dependent on the whole primary key.
- BCNF requires every non-trivial functional dependency X→Y, X is a super key.
- 4NF requires non-trivial multivalued dependency X →→ Y, X is a super key.

This project adopts normalization up to 4NF. As Microsoft SQL server is used in server side, the server must be eligible to handle complicated database operations such as table joining since MSSQL possesses good query optimizer. Below shows sample tables meeting 4NF.

Table 16. Memo Table

<table>
<thead>
<tr>
<th>Memo ID</th>
<th>Content</th>
<th>Time</th>
<th>Place ID</th>
<th>Photo Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Eating with Jason in ...</td>
<td>2012-02-03 12:02:43</td>
<td>3</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>Writing a letter to ...</td>
<td>2012-02-03 14:22:56</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 17. Memo-Tag Table

<table>
<thead>
<tr>
<th>Memo ID</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Food</td>
</tr>
<tr>
<td>1</td>
<td>Festival Walk</td>
</tr>
</tbody>
</table>
3.4.2. Entity Relationship Diagram

ER diagram is utilized to demonstrate the relationship between system entities. In below diagram, there are four major entities User, Memo, Place and Pending Place. Each entity possesses a primary key and some attributes while foreign key is not shown. Since Memo, Place and Pending Place are all dependent on User, they are designed as weak entity. Moreover, there is one multi-value attribute “tag” in Memo entity. While in actual implementation, the tag will be separated into a sub table since multi-value column is forbidden by 1NF.

Fig 22. Entity Relationship Diagram of the System
4. Detailed Methodologies and Implementation

4.1. Development Tools

Since this project involves development of iPhone application, the Integrated Development Environment (IDE) being used is an official tool released from Apple Inc. named Xcode 4.3. It is a versatile tool which includes an interface builder and an user-friendly editor for user interface development and coding Objective-C respectively. An additional tool from Xcode package named “Instruments” further offers developer a way to conduct automated testing and performance evaluation.

Table 18. Summary of all tools being used in this project

<table>
<thead>
<tr>
<th>Development Tool</th>
<th>Tool Name and Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Schedule tool</td>
<td>Microsoft Project 2010</td>
</tr>
<tr>
<td>Documentation tool</td>
<td>Microsoft Word 2010</td>
</tr>
<tr>
<td></td>
<td>Microsoft Excel 2010</td>
</tr>
<tr>
<td>System Design tool</td>
<td>ArgoUML 0.28.1 (Mac)</td>
</tr>
<tr>
<td>Development tool</td>
<td>iOS IDE: Xcode version 4.3</td>
</tr>
<tr>
<td></td>
<td>Java IDE: Netbeans 7.0.1</td>
</tr>
<tr>
<td></td>
<td>Web Server: Apache Tomcat</td>
</tr>
<tr>
<td></td>
<td>Database: MSSQL</td>
</tr>
<tr>
<td>Version Control tool</td>
<td>Google Document (cloud service)</td>
</tr>
<tr>
<td></td>
<td>Xcode built-in SVN</td>
</tr>
<tr>
<td>Presentation tool</td>
<td>Microsoft Powerpoint 2010</td>
</tr>
</tbody>
</table>
4.2. Data Accessing Process in Client-Server Architecture

In this project, Microsoft SQL Server (MSSQL) is utilized to store user data in a server machine. Since there is no connector for relational database that is suitable for iOS development, a middle layer must be implemented to handle data processing between data accessing objects (DAO) in client-side and database in server side. The middle layer is developed using a set of web services written in Java and stays server side. Recalled below system architecture, the middle layer is implemented as a set of Data Accessing Objects (web services) that stays behind a dispatcher and in front of MSSQL.

Fig 23. An overview of system architecture

4.2.1. Data Encapsulation using JSON for Transmission

In order to transmit data in a round trip between client and server, there must be a common data format that is readable by both server side and client side programs. In this project, since iPhone application can only make web request, JSON format is used to encapsulate database records into plain text so that the records can be read at client side program.
4.2.2. Mechanism for Data Transmission

In real practice, a JSON parser called “Maven JSon-lib” is used to translate Java beans into JSON format (Maven Json-lib, 2006). Besides, there is also an open source JSON parser namely “SBJson” used in client side to decode received data (in JSON format) back to readable objects in Objective-C (SBJson, 2009).

According to the below figure, client side DAO are responsible to translate user request into web request. The dispatcher in web server receives the request and invokes corresponding DAO to perform operations on MSSQL. After database operation, the server side DAO returns data in JSON format back to client. Finally, the client side DAO will decode the data back to readable objects in Objective-C.

Fig 24. A typical workflow of data transmission
4.3. Multiple Asynchronous Web Requests in Client-Server Architecture

4.3.1. Concurrent Access of Data Accessing Objects

Based on fore-mention system design, data transmission process is centralized by a set of data accessing objects (DAO) so as to improve maintainability and reusability. However, since it is possible to send multiple asynchronous web requests by one DAO, the asynchronous web request does not guarantee the same ordering of web responses. As the client side DAO need to decode received data (which is in JSON format), it is necessary to know the ordering of web response. Therefore, instead of solely return data back from the server, meta info is inserted into web response to tell the client side DAO about the action should be performed. For instance, the below code segment shows that meta information is inserted to tell the client side DAO that “readMemos” should be performed after receiving the web response.

Fig 25. Meta Information is Inserted in Web Response

```java
JSONArray result = new JSONArray();
MetaInfo meta = new MetaInfo();
meta.setAction("readMemos");
JSONObject metaJSON = JSONObject.fromObject(meta);
result.add(metaJSON);
while (rs.next()) {
    Memo memo = new Memo();
    memo.setId(rs.getInt("id"));
    memo.setUsr_id(rs.getString("usr_id"));
    memo.setTime(rs.getString("time"));
    memo.setContent(rs.getString("content"));
    if (rs.getInt("place_id") != 0)
    {
        Place place = new Place();
        place.setId(rs.getInt("place_id"));
        place.setName(rs.getString("name"));
        place.setLatitude(rs.getDouble("latitude"));
        place.setLongitude(rs.getDouble("longitude"));
        place.setAuto_checkin(rs.getInt("auto_checkin"));
        memo.setPlace(place);
    }
    Memo memo = new Memo();
    memo.setId(rs.getInt("id"));
    memo.setUsr_id(rs.getString("usr_id"));
    memo.setTime(rs.getString("time"));
    memo.setContent(rs.getString("content"));
    if (rs.getInt("place_id") != 0)
    {
        Place place = new Place();
        place.setId(rs.getInt("place_id"));
        place.setName(rs.getString("name"));
        place.setLatitude(rs.getDouble("latitude"));
        place.setLongitude(rs.getDouble("longitude"));
        place.setAuto_checkin(rs.getInt("auto_checkin"));
        memo.setPlace(place);
    }
    result.add(jsonObject);
}
out.println(result.toString());
```
4.3.2. Concurrent Retrieval of Images in Web Server

There are several circumstances that multiple images must be retrieved from server at the same time. For instance, the below figure shows a life memo list view which contains multiple user images. When the list view is loading, multiple user images should be collected from remote server at a time. Since every web response can only return one image, there must be certain mechanism to handle concurrent retrieval of images.

Fig 26. Concurrent retrieval of User Image in Memo List View
To resolve the problem, a dedicated class “WebImageView” is created to handle image retrieval. Multiple WebImageView objects can be created simultaneously to retrieve user image or memo image from local cache or remote server. Image will be retrieved from remote server only if there is no cached copy. The below code segment reveals two private function getMemoImage and getProfileImage for performing the fore-mentioned tasks.

Fig 27. WebImageView Class for Concurrent Image Retrieval

```objc
@interface WebImageView : UIView {
    UIImage *image;
    NSMutableData *responseData;
    NSString *action;
    iLifeMemoAppDelegate *d;
}

- (void)getMemoImage:(NSString *)usrID photoPath:(NSString *)photoPath;
- (void)getProfileImage:(NSString *)usrID;
@end
```
4.4. Identify Current Location of User

4.4.1. CoreLocation Framework

In order to locate the position of a user, it is necessary to access a powerful framework namely CoreLocation provided by Xcode. This framework provides a CLLocationManager class which contains vital positioning information such as current GPS coordinate, horizontal accuracy, vertical accuracy as well as current altitude of iPhone user.

By implementing a CLLocationManagerDelegate class, GPS coordinate of user can be collected periodically via a delegate function called “didUpdateToLocation”. In this project, the delegate function is very important because the GPS coordinate of user contributes to the most basic ingredient for automatic location recording feature in the system.

4.4.2. Screening out Unacceptable Location due to Positioning Error

Due to the limitation of positioning accuracy in iPhone, it is necessary to screen out unacceptable location detected. To achieve the purpose, the system predefined an acceptable range of horizontal accuracy. The initial range of horizontal accuracy is set to bigger than 0 since it must be impossible to be negative value.

Fig 28. Code segment for checking horizontal accuracy detected

```
- (BOOL) isValidLocation:(CLLocation *) location {
    if (location.horizontalAccuracy < 0) {
        return NO;
    }
    // valid location
    return YES;
}

#pragma mark CLLocationManagerDelegate
-(void) locationManager:(CLLocationManager *) manager
didUpdateToLocation:(CLLocation *)newLocation
fromLocation:(CLLocation *) oldLocation
{
    if (! [self isValidLocation: newLocation]) {
        NSLog(@"bad position");
    }
    return;
}
```
4.5. **Resolving Concurrent Access to Location Service in iPhone**

4.5.1. **Singleton - Location Manager Service in iPhone**

As discussed in previous section, the CCLocationManager class provided by CoreLocation framework permits the system to access to location service in iPhone. The CCLocationManager objects, however, cannot access to location service in iPhone simultaneously. In this project, since user can manually attach a location tag to a life memo while the system will automatically records user location in periodic manner, it is possible to cause corruption if two CCLocationManager objects access to location service at the same time. As a result, CCLocationManager must be implemented with Singleton Pattern to centralize location processing into only one CCLocationManager object. On top of singleton pattern, there must be a multi-threading control mechanism to handle simultaneous access of the singleton object.

Fig 29. Singleton Pattern of CCLocationManager Object
4.5.2. Preemptive Multi-Threading of Location Manager

When user perform manual task such as attaching a location tag to a life memo, the system should display a list of nearby places for selection. Since the system access location service layer periodically to record user locations automatically, this automated process must be put to sleep in order to release the location service for the manual task. In this case, we can summarize that automatic running thread is always preempted by manual process thread to insure the user getting real-time access to location service without any delay.

Fig 30. Preemptive multi-threading approach used in location processing
4.6. Display Location Information with MapKit

4.6.1. Mapkit Framework

The “MapKit” framework provided by Xcode offers a list of services related to map view processing. The framework adopts Google Map API to provide map grids and corresponding map information such as place names and street names. A class “MKMapView” provided by the framework is used to control the actual map view shown on the screen. There are some properties that can be set, including the center of the map, zooming level and type of map view (i.e. map or satellite view), etc.

Fig 31. Map view of user’s current location (left: map view, right: satellite view)
4.6.2. Adding Annotation on Map View

In addition to zooming and positioning the map view, we can also add pins and annotations into a map to provide extra information to user. In this project, the most obvious usage of pinning a map is to display the name of a place. For instance, if a user turns on automatic location recording function, once he walked into CS Lab, the iPhone application will automatically record his location in a remote database. If he views this record in a map, there will be a pin showing the location of CS Lab as in the following figure.

Fig 32. Map view showing the location of CS Lab
Fig 33. Code segment for adding a pin in the map

```swift
// add map view object
CLLocationCoordinate2D annotationCoord;
annotationCoord.latitude = 22.3363;
annotationCoord.longitude = 114.174;

MKPointAnnotation *annotationPoint = [[MKPointAnnotation alloc] init];
annotationPoint.coordinate = annotationCoord;
annotationPoint.title = @"CS Lab";
annotationPoint.subtitle = @"CS Lab - CityU";
[mapView addAnnotation:annotationPoint];
```

As shown in the above figure, class “MKPointAnnotation” is used to attach annotation such as title and subtitle in a map view based on predefined coordinate. Indeed, the above code segment only demonstrates the brief idea of how an annotation is created. In actual implementation, the annotation will be generated dynamically based on the place requested by user. The title and subtitle of the annotation will also be initialized according to location detail stored in system database.
4.7. Place Recognition

Before discussion on how to perform automatic recording user location, it is important to state the methodology used in place recognition.

4.7.1. Translating GPS Coordinates with Google Place API

Google Place API can translate GPS coordinate into corresponding city, district, street address or the name of a place. In initial stage of system release, there will be very little location information stored by the system. Google Place API becomes a crucial component in place recognition as it provides an acceptable number of location information. As discussed in literature review, people usually do not mention a location with street address, therefore, only district and place name will be utilized in the system.

In order to access to Google Place API, a Google API account must be registered. After registration, Google will offer you a unique key for accessing all kind of Google API service.

Fig 34. API Key offered by Google to access Google Place API

API Access

To prevent abuse, Google places limits on API requests. Using a valid OAuth token or API key allows you to connecting requests back to your project.

Authorized API Access

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private. Learn more

Create an OAuth 2.0 client ID...

Simple API Access

Use API keys to identify your project when you do not need to access user data. Learn more

Key for browser apps (with referers)

API key: ATzeSyAB6CF-□□□□□□□□□□□
Referers: Any referer allowed
Activated on: Feb 16, 2012 4:57 AM
Activated by: §□□□@gmail.com – you

Create new Server key... Create new Browser key...
As stated in Google Place API documentation, the API can be accessed via a HTTP request. There are some mandatory parameters to be setup in the HTTP request, including API key, radius, location (GPS coordinate in form of latitude and longitude) and sensor type. In this project, the radius is set to retrieve location information within 500 meters to restrict the number of resulting locations. Besides, the location parameter will be configured based on the location of user and the sensor parameter is set to “true” to indicate the request is sent from a GPS device (i.e. iPhone).

Fig 35. Format of making a HTTP request to Google Place API

**Place Search Requests**

A Place Search request is an HTTP URL of the following form:

```
https://maps.googleapis.com/maps/api/place/search/output?parameters
```

where `output` may be either of the following values:

- `json` (recommended) indicates output in JavaScript Object Notation (JSON)
- `xml` indicates output as XML
4.7.2. Translating GPS Coordinates to User-Defined Location

Apart from business name people also indicate different places with their own semantic labels such as “home” or “project lab”. To implementing this feature, the system can provide an interface that allow user to input the name of a place and get the GPS coordinate of user automatically to represent the location of that place. In this way, when automatic location recording function is activated, the system can retrieve user-defined places nearby from system database.

Fig 36. Creating User Defined Place during Place Management
4.8. **Automatic Recording User Location**

According to the project objective, in order to assist the process of life memory recording, the system should be capable of automatic recording daily places that are visited by a user. Since the location records are used to remind user about what should be written in that day, the accuracy and correctness of location records are very important. The below sections will explain the rationale of methodologies used to insure the accuracy and correctness of automatic location recording.

4.8.1. **Factors Hindering Automatic Location Recording**

An ideal case for automatic location recording is to directly store every place user has visited during the day. However, there are several factors hindering the process which make it impossible to be done in real practice.

Assume a user locates in somewhere such that nearby places are already collected by the system. To determine which place a user is locating, the most obvious approach is to choose a place with minimal distance from the current location of user. Nonetheless, as discussed in literature review section, the median deviation in GPS positioning accuracy of smartphone is up to 600 meters (Zandbergen, 2009). The positioning accuracy can be even more unstable if a user locates indoor or surrounded by crowded buildings. Therefore, user location detected by smartphone may not be the exact location of the user. It further implies that a place with minimal distance from user location detected may not be the actual place a user is locating.
According to the above figure, a user stays in Mong Man Wai Building (solid circle) may be misinterpreted as nearby Innocentre (hollow circle). In this case, the place with minimum distance from the detected location is Innocentre but not Mong Man Wai Building. Hence, the system fails to reason the actual location of user.
4.8.2. Automatic Recording Toggle

Since Google Latitude also possesses automatic location recording feature, it is a good idea to reference their approach. Basically, before the system can decide which place user is locating, user has to turn on an “automatic recording toggle” for some places. In this project, once user has manually attached a location tag to a life memo, he can decide whether the “automatic recording toggle” for that location should be turn on or off. If the toggle is turned on, the system will automatically record this location once user is right there. With this mechanism, the system does not need to consider every place nearby user location, which makes the place reasoning process much more accurate.

Fig 38. Setting Automatic Recording Toggle for City University of Hong Kong
Take the below table as an example. When user locates at Kowloon Tong, the system identifies 4 nearby places. The user has manually added location tag to life memo using City University of Hong Kong and Festival Walk before. Moreover, he only turns on “automatic recording toggle” for City University of Hong Kong so that the system will determine his location as City University of Hong Kong but not others.

<table>
<thead>
<tr>
<th>Nearby Places</th>
<th>Automatic Recording Toggle</th>
</tr>
</thead>
<tbody>
<tr>
<td>City University of Hong Kong</td>
<td>On</td>
</tr>
<tr>
<td>Innocentre</td>
<td>N/A</td>
</tr>
<tr>
<td>Heung To Middle School</td>
<td>N/A</td>
</tr>
<tr>
<td>Festival Walk</td>
<td>Off</td>
</tr>
</tbody>
</table>

4.8.3. Determination of Closest District

As discussed in previous section, “automatic recording toggle” can effectively improve place-reasoning process by minimizing number of place considered. However, it does not mention what should be done if there are no nearby places with “automatic recording toggle” set to on. To cope with this circumstance, the system passes GPS coordinate of a user to Google Place API to get closest district in return. For instance, when a user arrive Festival Walk but he does not turn on any “automatic recording toggle”, the system determines closest district of Festival Walk (which is Kowloon Tong) and store the district as location record. Although this approach is not specific enough to tell which place user has visited but it ensures that location recorded is correct. As stated before, in order to make automatic location recording useful, the correctness of location record is more important.
4.8.4. Determining Closest Location with Bayesian Network

With reference to previous methodologies used in automatic location recording, if there are multiple places with “automatic recording toggle” turned on, the system should select the most likely location. The process cannot simply done by calculating the minimal distance between a place and the detected user location since the detected user location can be wrong. As discussed in literature review section, different places may have different degree of relevance to different people. To determine which place is more likely to be the actual location of user, Bayesian Network is adopted to select a place with largest probability.

Basically, the Bayesian Network comprises for 3 components:

- Is Weekend - a Boolean factor that determines whether the place is visited in weekend
- Hour – an integer factor ranged from 0-23 to determine the time a place is visited
- Happen – a Boolean factor that determines whether a user visits that place with a given time slot

By using Bayesian network, we can determine whether a user is at a particular place in a particular moment. This is done by calculating the probability $P(\text{happens})$. For example, the below table reveals the probability of a user being at school in weekdays.

<table>
<thead>
<tr>
<th>Time</th>
<th>0 - 6 am</th>
<th>7 - 8 am</th>
<th>9 am - 1 pm</th>
<th>2 - 5 pm</th>
<th>6 – 7 pm</th>
<th>8 – 9 pm</th>
<th>10 – 11 pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(\text{happens})$</td>
<td>0.1</td>
<td>0.4</td>
<td>0.8</td>
<td>0.8</td>
<td>0.6</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>
4.8.5. Resolving Incorrect Location Record

All in all, different methodologies are developed to increase the quality of automatic location recording feature. Nevertheless, it is still possible to fail the process. For instance, when “automatic recording toggle” of multiple locations is set to on and the system not yet records user behavior pattern (i.e. probability of visiting a particular place), Bayesian Network can hardly be usable in this situation. Therefore, an error resolution mechanism must be implemented. Basically, the system will store user location and perform operation similar to adding a location tag manually by user when he is creating a new life memo. According to the below figures, City University of Hong Kong is automatic recorded by the system. If this is an incorrect record, user can click the Other Places button to retrieve nearby locations.

Fig 39. Correcting Location Record by Clicking Other Places Button
4.9. Exploring Personal Life Memo using Augmented Reality Browser

4.9.1. Geotagged Life Memo

As stated in project objective, in order to provide a convenient life memory exploration tool, an augmented reality browser should be built to explore personal life memos with respect to user location. To meet the objective, life memos in the system should contain location information. This is achieved in two ways. First, when user is writing a new life memo, he can attach a location tag to indicate the location of this memo. Second, as the system automatically records user location, user can create life memo using the automatic location record as well. In these ways, life memos can be classified based on their attached location information.

Fig 40. Attaching Location Tag “Festival” in a Life Memo during Life Memo Creation
4.9.2. Mechanism for Personal Life Memo Exploration

First, when user activates the augmented reality browser (AR browser), the system will start the camera of iPhone and collect all personal life memos nearby current location of the user. Then, the life memos will be positioned based on their location. When user move the camera around such that some life memos are within the investigation area of AR browser, the life memos will be revealed on the camera view, showing the memo author and content.

Fig 41. Exploring Personal Life Memos with Augmented Reality Browser
The below figure demonstrate an idea of investigation area of AR browser. As the system needs to retrieve nearby life memos, only those within 500 meters will be considered. Moreover, life memo will only be shown on screen when user positions his iPhone in a correct direction with the life memos.

Fig 42. Investigation area of Augmented Reality Browser

```cpp
// Locate memos within 500 meters
NSMutableArray *tmpMemos = [[NSMutableArray alloc] init];
for (Memo *memo in memos) {
    CLLocationCoordinate2D coord = [LocationController sharedInstance].coord;
    CLLocationCoordinate2D userLoc = CLLocationCoordinate2DMake(latitude, longitude);
    if ([userLoc distanceFromLocation:coord] < 500) { // within 500 meters
        [tmpMemos addObject:memo];
    }
}
memos = tmpMemos;
```
4.9.3. Augmented Reality Browser Engine

In order to overlaying information over a camera view of iPhone, we must use an augmented reality browser (AR browser) engine. There is an open-source AR browser engine named iPhone ARKit which is written in Objective-C and is applicable in this project (iPhoneARTKit, 2010). Indeed, it is a lightweight AR browser engine that consists of only four classes. Although the classes already implement fundamental features of an AR browser, certain customization is still required to make it more adaptive to this project.

Firstly, the ARCoordinate class acts as a data model that stores all information of a life memo being displayed on the AR browser. In this project, the ARCoordinate class is tailor-made to encapsulate the information of a life memo such as author, content as well as memo ID for further retrieval of the life memos in a table view. The class also contains positioning attributes such as radial distance, azimuth and inclination to control the location of a life memo.

Fig 44. Code segment showing the attributes in ARCoordinate class

```
@interface ARCoordinate : NSObject {
   // Attributes to be used for positioning
   double radialDistance;
   double inclination;
   double azimuth;

   // Attributes to be display on AR browser
   NSString *title;
   NSString *subtitle;
   NSString *memoID;
}

- (NSUInteger)hash;
- (BOOL)isEqual:(id)other;
- (BOOL)isEqualToCoordinate:(ARCoordinate *)otherCoordinate;

+ (ARCoordinate *)coordinateWithRadialDistance:(double)newRadialDistance inclination:(double)newInclination azimuth:(double)newAzimuth;
```
Secondly, the ARGeoCoordinate class is a subclass of ARCoordinate class which possess an additional attribute to store GPS coordinate of the life memo being displayed. It also contains functions to determine the angle of a life memo, calibrating the location of a life memo, as well as a constructor to setup initial GPS coordinate of the life memo.

Fig 45. Code segment showing the ARGeoCoordinate class

```swift
@interface ARGeoCoordinate : ARCoordinate {
    CLLocation *geoLocation;
}

@property (nonatomic, strong) CLLocation *geoLocation;
-(float)angleFromCoordinate:(CLLocationCoordinate2D)first toCoordinate:(CLLocationCoordinate2D)second;
+(ARGeoCoordinate *)coordinateWithLocation:(CLLocation *)location;
-(void)calibrateUsingOrigin:(CLLocation *)origin;
+(ARGeoCoordinate *)coordinateWithLocation:(CLLocation *)location fromOrigin:(CLLocation *)origin;
@end
```
Thirdly, the ARViewController is a controller class that implements core logic of AR browser. It makes use of CCLocationManager class to access the location service in iPhone. Before the AR browser determines whether some life memos are within investigation range, it is mandatory to get the current location of user first because the system need to retrieve nearby life memos from database. Hence, CCLocationManager is utilized by ARViewController to get user location. Moreover, the UIAccelerometer class is also adopted to access the accelerometer in iPhone. When user activates the AR browser, he needs to move the camera around to explore personal life memos. UIAccelerometer class is adopted to capture the motion of iPhone so as to determine whether a particular life memo should be overlaid on the camera view.

Fig 46. Attributes in ARViewController (e.g. location manager, accelerometer manager)

```objc
@interface ARViewController : UIViewController <UIAccelerometerDelegate,
    CLErrorManagerDelegate> {
    CLErrorManager *locationManager;
    UIAccelerometer *accelerometerManager;

    ARCoordinate *centerCoordinate;
    UIImagePickerController *cameraController;

    NSObject<ARViewDelegate> *__unsafe_unretained delegate;
    NSObject<CLLocationManagerDelegate> *__unsafe_unretained locationDelegate;
    NSObject<UIAccelerometerDelegate> *__unsafe_unretained accelerometerDelegate;
}
```

Indeed, the functionalities of ARViewController are very solid. It provides function to maintain ARCoordinate so that the system can add / remove life memos from the AR browser dynamically. However, it does not provide a toggle mechanism to start / stop the AR browser since it is intentionally designed as a standalone mobile application. In order to incorporate the open-source AR browser engine into this project, it is necessary to implement a toggle mechanism to control the access to different resources such as camera, location manager and accelerometer of iPhone. Apparently, all these resources can only be accessed by one requestor at a time, there must be careful management to ensure the resources are released after use.
Fig 47. Code segment showing the functions for adding or removing life memo

```swift
// Adding coordinates to the underlying data model.
- (void)addCoordinate:(ARCoordinate *)coordinate;
- (void)addCoordinate:(ARCoordinate *)coordinate animated:(BOOL)animated;

- (void)addCoordinates:(NSArray *)newCoordinates;

// Removing coordinates
- (void)removeCoordinate:(ARCoordinate *)coordinate;
- (void)removeCoordinate:(ARCoordinate *)coordinate animated:(BOOL)animated;

- (void)removeCoordinates:(NSArray *)coordinates;
```

Fig 48. Code segment showing the release of resources when AR browser is stopped

```swift
// Release location manager, accelerometer and inactivate timer
- (void)stopListening {
    self.locationManager.delegate = nil;
    self.locationManager = nil;
    self.accelerometerManager.delegate = nil;
    self.accelerometerManager = nil;
    self.centerCoordinate = nil;
    [_updateTimer invalidate];
    _updateTimer = nil;
}
```

Finally, the ARGeoViewController class is a subclass of ARViewController. It encapsulates all functionalities provided by the open-source AR browser engine as a whole. In this project, the system will create only one instance of ARGeoViewController when user wants to explore life memos they have written in the past.
4.10. Explore Collective Memory

4.10.1. Notion of Collective Memory
As stated in project objective, this project aims to build up collective memories among users so that they can learn from the life memos written in the past. Nevertheless, the notion of collective memory may seem a little bit arbitrary and subjective to different people. Generally speaking, collective memory should be something common and popular among a group of people. It can be an event such as a birthday party, or a controversy issue like the election of Hong Kong Chief Execution Election.

4.10.2. Analysis of Collective Memory Based on Memo Tags
When user creates a new life memo, he can attach descriptive tags to the memo. These tags are some keywords that have semantic meaning about the memo content. Thus, memo tags can be analyzed to find most popular keywords within a particular period. Nonetheless, the quality of memo tag analysis depends solely on the quality of tag content. If user input wrong or meaningless tags, the quality of collective memory exploration may be seriously degraded.

Fig 49. Adding Tag during Life Memo Creation
4.10.3. Mechanism for Collective Memory Exploration

As similar to personal life memo exploration, user should activate the augmented reality browser (AR browser) to start collective memory exploration. The system will retrieve life memos that represent collective memory about the current location of user and display them on a camera view. All life memos will be positioned based on their location so that user can move his iPhone around to investigate where are the collective memory. Once there are some life memos within the investigation area of the AR browser, the life memos will be revealed on the camera view.

Fig 50. Exploring Life Memos that Represent Collective Memory in AR Browser
4.10.4. Statistical Analysis on Memo Tags

As stated in previous section, collective memory should be something common and popular among a group of people. To find the popularity and similarity between different life memos, a statistical approach can be done to select life memos based on their content. This is achieved by examining the popularity of memo tags.

Fig 51. Statistical Distribution of Tag Occurrence in Nearby Life Memos

Distribution of Tag Count

Assumed there are totally 100 tags stored in the system and each tags has been count for their occurrence in life memos. The x-axis represents a tag number and the y-axis represents no. of occurrence of a tag. To determine which tag is possibly a keyword for collective memory, the tags with highest occurrence will be used to retrieve corresponding life memos. In other words, the system is able to select life memos based on trending keywords. However, if the sample size is too small or the difference between tags is not large, then this method is no longer meaningful. In such circumstance, all previous life memos around that place will be shown.
4.11. Performance Considerations

4.11.1. User-driven Data Retrieval

Nowadays, 3G network connection is relatively expensive. Network traffic must be carefully monitored to reduce unnecessary data usage. In this project, unless the local cached data is out-of-date, data retrieval process is user-driven to minimize network consumption.

Fig 52. Pull Down to Update Screen Data
4.11.2. Image Compression

A typical photo taken in iPhone ranged from 1~3 MB. Sending a native iPhone photo without compression causes serious network overhead. Therefore, images are compressed as much as possible before sending to remote server. For instance, the below code segment demonstrates a logical flow of image compression of user profile picture. First, maximum scale is calculated to ensure the resized image will not be too small. After scaling the image based on calculated scale, the image is transformed into a square image so that non-displayed part will be cropped.

Fig 53. Code Segment Showing Image Compression of Profile Picture

```swift
- (void)imagePickerController:(UIImagePickerController *)picker didFinishPickingMediaWithInfo:(NSDictionary *)[sender result](id result) {
    NSString *mediaType = [info objectForKey:UIImagePickerControllerMediaType];
    if ([mediaType isEqualToString:NSHexString]) {
        UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];

        double scale = 200/[[image size] width];
        if (scale<200) {
            scale = 200/[[image size] height];
        }

        // resize image
        image = [self scaleImage:image toSize:CGSizeMake([[image size] width]*scale, [[image size] height]*scale)];

        // convert into square profile image
        int padW = ([image size].width-160)/2;
        int padH = ([image size].height-160)/2;
        CGRect requiredBounds = CGRectMake(padW, padW, 160, 160);
        CGImageRef imageRef = CGImageCreateWithImageInRect([image CGimage], requiredBounds);
        UIImage *croppedImage = [UIImage imageWithCGImage:imageRef];
        CGImageRelease(imageRef);

        imageView.image = croppedImage;
    }
}
```

Fig 54. Profile Image Processing (Compression & Cropping)
4.11.3. Caching Images to Reduce Network Overhead

As images are relatively larger in data size than texts, it is a must to cache image data so that unnecessary image transmission between client and server is prohibited. With reference to below figure, when the image is received from web response, it will be cached into a session variable “memoImageCache” for further retrieval.

Fig 55. Code Segment Showing Image Caching

```c
- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    NSString *responseString = [[NSString alloc] initWithData:responseData encoding:NSUTF8StringEncoding];
    UIImageView *imageView = [[UIImageView alloc] initWithData:[UIImage alloc] initWithData:responseData];
    imageView.frame = self.bounds;
    imageView.contentMode = UIViewContentModeScaleAspectFill;
    [self oddSubView:imageView];
    if ([action isEqualToNSString:@"getMemoImage"])
    {
        [NSNotificationCenter defaultCenter] postNotificationName:@"getMemoImageDone" object:nil];
    } else {
        if ([imageView.image.size.height!=0]) { // if photo exists
            [d.memoImageCache setValue:imageView.image forKey:action];
        }
        [NSNotificationCenter defaultCenter] postNotificationName:@"getProfileImageDone" object:nil];
    }
    responseString = nil;
}
```

4.11.4. Memory Control

Memory control is a vital operation in mobile application development due to limited memory capacity of mobile devices. Potential memory leakage may lead to system crashing. In order to perform good memory control, the latest Automatic Reference Counting (ARC) mode introduced in iOS5 is adopted so that unused objects will be destroyed by the system automatically. If there are certain objects retained in memory by developer, these objects must be release immediately after finishing their operation. Therefore, no memory leakage should appear in the system.

4.11.5. Quick Sort of Memo Tags for Counting Process in Statistical Analysis

To perform statistical analysis, all memo tags will be counted programmatically to form a statistical distribution. Considering if the no. of life memo is very large, the counting process can be very slow. Therefore, quick-sort must be applied to ensure the operation can be finished in $O(n \log n)$ time.
5. **Experimental Results and Evaluation**

5.1. **Field Test**

5.1.1. **Automatic Location Recording**

To test automatic location recording feature of the system, a tester turned on the automatic recording toggle for some places and went to those places within a day. He kept the application running in background. The experimental result shows that the system is capable of recording every place he visited. At night, the tester went to Ngau Tau Kok such that he did not set automatic recording toggle for any places nearby. In this case, the district Ngau Tau Kok is recorded by the system. Based on the field test, the system works exactly as what it is intended. With a set of location records, the tester commented that he finds this feature useful as he can simply create life memos with each location record and does not need to write a whole piece of diary.

Fig 56. Using Location Record to Write Life Memos
5.1.2. Exploration of Personal Life Memo

With a set of personal life memos stored in the system, a tester turned on the augmented reality browser (AR browser) to explore the old life memos written in the past. As the tester conducted the field test in project laboratory, all life memos retrieved by AR browser is limited within 500 meters. As shown in below figures, three life memos within this range is retrieved and display in the AR browser.

Fig 57. Exploring Personal Life Memos with Augmented Reality Browser
Although the system is capable of helping user in personal memory exploration, the detected location of user is not always accurate which may lead to wrong direction of life memos. For example, a life memo which contains a location tag of Festival Walk should be displayed when user faces north, but he found the memo when he faced north-east since the detected user location is different from his actual location. Although the direction may not be correct, the AR browser is still useful in find peripheral life memos. Therefore, the exploration of life memory is still achieved.

Fig 58. Incorrect Direction of Augmented Reality Browser
5.1.3. Exploration of Collective Life Memos

To test collective memory exploration, it is necessary to setup a lot of life memos in a particular location. In this test, a tester turned on the augmented reality browser (AR browser) to explore the life memos written in Kowloon Tong. As there are many life memos with a tag of “FYP”, all life memos retrieved are concerned about working on final year project. Moreover, the tester also discover that most of these life memos are tagged in Project Lab, which probably implies that most of the system users are year 4 students working on final year project in the project lab. The test proves that the system is capable of performing collective memory exploration.

Fig 59. Exploring Life Memos that Represent Collective Memory in AR Browser
5.2. Battery Consumption Test

Since the system is developed in Xcode 4.3 which targets iOS 5.0 or above, battery consumption issue is a prolonged problem of iOS 5. Thus, instead of showing the battery consumption rate during the usage of the system, the battery consumption test is conducted to contrast the battery consumption rate with and without using the system. To perform the test, the system was run with automatic location recording feature for 5 hours. After recording the battery consumption, the system is then recharged until 100% and idle for 5 hours. The below figure explains the result.

Fig 60. Comparison of battery consumption in 5 hours

According to the result, the system under normal operations (e.g. write life memos, uploading photos, automatic location recording, running augmented reality browser, etc.) has roughly double battery consumption as the iPhone is idle. If the user only uses automatic location recording function, the battery consumption rate is only 1.5 times as an idle iPhone. If the user does nothing and keep the system idle, the battery consumption is roughly the same as an idle iPhone. In overall, the battery consumption is acceptable.
5.3. Memory Test

Due to limited processing power and battery life of mobile phone, memory control is a crucial task in mobile application development. Potential memory leakage may lead to abnormal user experience and even system crashing. Therefore, memory testing must be done to insure the feasibility of the system in practical use.

There is a memory monitor tool in the Xcode package provided by Apple Inc. The tool can monitor memory usage and detect any memory leakages. The memory leakage test is conducted by running every function in the system for several times. The result is shown in the below figure. Basically, the first row “Allocations” shows memory consumed at a particular period. The second row “Leaks” reveals any potential memory leakage. As seen in the below figure, there is no memory leak during the memory test. Therefore, memory control has been well achieved.

Fig 61. Memory Testing using Instruments
6. Discussion and Conclusion

6.1. Achievement

6.1.1. Project Achievement

“Life is not what you live, but what you remember”, said by Gabriel Garcia Marquez. This project successfully re-invents the way people used to record their daily life. As stated in project introduction, this project aims to facilitate people in life memory recording and memory exploration. The mobile application iLifeMemo provides an accurate automatic location recording function so that user can record their life based on a list of location record. Moreover, instead of reading tons of life memo one by one to review their living in the past, an augmented reality browser is developed as an memory exploration tool such that user can easily search life memos based on their current location. Last but not the least, the AR browser can be used to explore collective memory in form of life memos written by different users. Therefore, this project has achieved three levels of life memory management as shown in below figure.

Fig 62. Three Levels of Memory Management Achieved
6.1.2. Technical Achievement

Instead of making a simple mobile application, this project adopts an innovative technology called augment reality to overlay information into real life situation. The re-engineered AR browser can be plugged into other application so that developers can add image and text using the altered open-source AR browser.

Moreover, positioning in mobile device is inaccurate due to poor GPS receiver embedded. This project has implemented different methodologies to increase the accuracy in determining user location. The methodologies are actually reusable for other applications which requires accurate user location positioning.

6.1.3. Personal Achievement

Within the project year, I have successfully gone through a completed software engineering life cycle. Indeed, before working on this project, I have no background knowledge about mobile application development. The development of iLifeMemo brings me a lot of new notions that should be considered during mobile application development. For instance, generally a good mobile application should carefully monitor wireless data usage, perform good memory control, battery control and provide only mandatory information to user since the screen of mobile device is limited. I also learn to make location-based service using GPS positioning in iOS 5.0. More importantly, I also experience a whole cycle of working on a computer science project especially the way of conducting literature review.

6.2. Difficulties Encountered

There are many difficulties encountered during the project and most of them are mentioned in methodology section. For instance, multiple asynchronous web request in mobile client-server architecture, user location reasoning, multithreading of location manager, place recognition, concurrent access to multiple images in remote server, data caching to reduce network traffic, etc. Fortunately, all difficulties encountered are resolved at the end.
7. **Future Enhancement**

7.1. **Network Disconnection**

Although the field test conducted proves the success in system functionalities, the result is dependent on a normal situation with stable network connection. Indeed, the testing iPhone with 3G network was occasionally disconnected due to unstable wireless network. Since the system is developed in a client-server architecture, network disconnection may lead to the termination of callback event.

For instance, a new life memo written by user will be sent to a remote server, then, the server will store the memo and return the latest set of life memos back to user. If the network is disconnected after sending the new memo to remote server, user may not receive the latest set of life memos. To tackle this problem, there is a drag-down refresh feature provided so that user can refresh all cached content manually. A better enhancement can be done to detect system disconnection and get the missing callback event automatically.

Fig 63. Callback Event Termination (Step 7 & 8) due to Network Disconnection
7.2. Data Integrity

As an extended problem of network disconnection, sometime there will be data integrity problem due to disconnected network. For instance, when user is uploading an image, network disconnection may cause a bad image received in remote server. Similarly, when user requests for an image that are embedded in a life memo, the received image may not be intact if the wireless network is unstable. Currently the system does not handle any data integrity problem. This is a crucial improvement with high priority.

Fig 64. Callback Event Termination (Step 14 & 15) causing corrupted image
8. References


