

Binit Shrestha

Development of Location-based Photo Sharing iPhone Application

A case study Breakit

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor's Thesis

20 June 2013

Author(s) Title Number of Pages Date	Binit Shrestha Development of location-based photo sharing iPhone Application: A case study Breakit 44 pages + 7 appendices
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Computer Networks
Instructor(s)	Peter Hjort, Senior Lecturer Sebastian Schroderus , CEO
<p>The primary purpose of the project was to build an outstanding location-based photo sharing iPhone application named Breakit for a start-up company named Eighty-Five Media Oy. The core functions of the application were displaying the photo feed based on the user's current location and the ability to take photo, write some comments and upload to the Breakit server and other social media such as Twitter and Facebook. The main goal of the project was to develop the application and release it globally on Apple Store. On the other hand, the project was an attempt to implement the knowledge of iPhone programming learnt over the years at the university and to learn and practice agile software development in the production environment with a multinational team.</p> <p>For the Breakit server, relatively new technologies such as Node.js, backbone.js, Mongoddb and coffeeScript were used and for the client, iOS frameworks and some third party open source projects for iOS were used. The project was deeply concerned only with the iOS frameworks and other third party open source projects. The integration with Facebook, Twitter, Foursquare and obtaining the relatively accurate location of the user were the core features that the project dealt with. Apple's official documentation, various iOS programming books and online materials were gathered and used during the project.</p> <p>The application was released on Apple Store for global audience. Hence, the iPhone user would benefit by using the application and as for the iOS application developers, start-up workers and the iOS programming students, the project can be a guidebook to develop for location based services, social media integration and Apple Push Notification Service-enabled application.</p>	
Keywords	LBS, PhotoSharing, iOS, FacebookSDK, CoreLocation, Twitter

Contents

1	Introduction	1
2	Location-Based Services And Their Marketing Potential	2
3	Location-Based Photo Sharing Application Breakit	3
3.1	Bridge Between Users And Media	3
3.2	Target Customers	4
3.3	Competitors	4
4	Technologies Behind The Breakit Application	5
4.1	Backend And Frontend Technologies	5
4.1.1	Node.js	5
4.1.2	MongoDB	6
4.1.3	HTML5, CSS3, CoffeeScript	6
4.2	iOS Mobile Client Technologies	7
4.2.1	iOS Software Development Kit	7
4.2.2	iOS Frameworks	9
4.2.3	iOS Third Party Open Source Projects	12
5	Development Of iOS Mobile Client	15
5.1	Analysis And Design Details	15
5.1.1	Users' Perceptions	23
5.1.2	Authentication Workflow	28
5.2	Application Implementation	29
5.2.1	Design Implementation	29
5.2.2	Application Testing	40
6	Result and Discussion	42
7	Conclusion	44
	References	45
	Appendices	
	Appendix 1. Facebook User Authentication Process	
	Appendix 2. Twitter Authentication Overflow	
	Appendix 3. Behavioural Test Cases	

Abbreviations and Terms

Application Programming Interface (API)

An interface implemented by a software program to interact with other software components.

Cascading Style Sheets (CSS)

A stylesheet language used to format a document in a way to enable separation of layout, colors, font that improves accessibility, flexibility and reduces complexity and repetition in the structural contents.

Graphical User Interface (GUI)

A type of user interface that allows users to interact with electronic devices using graphics rather than the text or commands.

Hypertext Pre-Processor (PHP)

A server side scripting language.

Integrated Development Environment (IDE)

A software application that provides source code editor, build automation tools and a debugger for software development.

iPhone Operating System (iOS)

A mobile operating system developed and distributed by Apple Inc.

JavaScript Object Notation (JSON)

Originated from the JavaScript scripting language. Its format is used for serialization and transmission of structured data over a network connection.

Multimedia Messaging Service (MMS)

A service in cellular phone to send and received pictures, sounds clips and text messages.

Must, Should, Could, Would (MoSCoW)

A technique used in software development to reach a defined goal.

Software Development Kit (SDK)

A set of software development tools to create application for software package, software framework.

Structured Query Language (SQL)

A language for accessing and manipulating databases.

User Interface (UI)

A system by which user interact with a machine.

1 Introduction

Recent observation shows that mobile phones and social media have revolutionized the way of communication and the lifestyle of people. An increasing number of mobile phones and networking sites allow people to access the Internet, interact with people virtually and gather information wherever they are and whenever they want to. The other reason behind the popularity of smart phones is that they make it possible for the user to download different applications related to business, entertainment, music, finance and games. Increasing rage towards the smartphone, mobile applications and social media were the motivation for this project.

These days, the mobile phone has been playing an essentially important role in our daily life. Users do not just use mobile phones for communication purposes but also for online banking, playing games, navigation purposes and educational purposes. Due to the mobile phone programming platform, it has been possible to convert a mobile device into a multi-functional device. According to a recent update, 91% of all mobile Internet usage is social-related, such as Facebook, Twitter, Foursquare, Pinterest, Tumblr and Path.[25].

The purpose of this project mainly concerns the study of some of the popular social media platforms such as Facebook, Twitter and Foursquare and a theoretical study of location-based services and their practical implementation on the final product. The goal of this project is to develop a location-based photo sharing application for iPhone with iOS 5 and later. The project attempts to show how location-based services have been useful not only for users but also for service providers.

The scope of this project is focused on creating a location-based application for iPhone. The application was created with the help of application programming interfaces (API) provided by social networking sites combined with software development kits (SDK) such as Facebook SDK3.0, iOS in-built Twitter API and Foursquare API. The created application would provide a flow of photo feeds that are happening around the user's current location rather than time-based feeds as the current giant social media has been providing. Being a startup company, Eight-Five Oy currently pays attention to the iPhone application but will produce an application for iPad, Android and Windows phones in the coming days.

2 Location-Based Services And Their Marketing Potential

Location-based services (LBS) literally mean the services provided according to one's current location. At a specific time and location, a location-based service uses geographical location of a user to provide information via mobile devices. The services provided through this technology traverse a broad range such as entertainment, restaurant finder, buddy tracker and navigation services for mobile marketing and mobile gaming. The technology has been gaining popularity due to the fact that the users are not required to enter location information manually but are automatically tracked. Particularly in today's technical era, location information is important where users access and consume wireless services.

Location based services have leaped off today's technological world, as a result of which people use it on a daily basis, whether it is to 'check-in' at a certain place, to look for a nearby bank or ATMs, or to search for restaurants and gas stations. These days not only the customer but also the marketers do have their reach in location based services. In mobile marketing, the main target of the marketers is to reach consumers at an appropriate place and time for them to become satisfied. For example, if a customer is nearby a Hesburger, and marketers are offering special deals only on that particular day, then the consumer gets a chance to know about the discount offered. That means it increases the chance that the consumer is willing to get the offer. These days, LBS have been used by some of the applications such as Foursquare, Google latitude, yelp and Facebook.

In the previous decade, the scope of location-based services were limited to tracking and navigation in military applications. The usability and demand have grown with respect to time, which results in the daily access of location information and wireless services. Since the recent launch of iPhone's location aware-map, the possibility of LBS growth has highly increased.

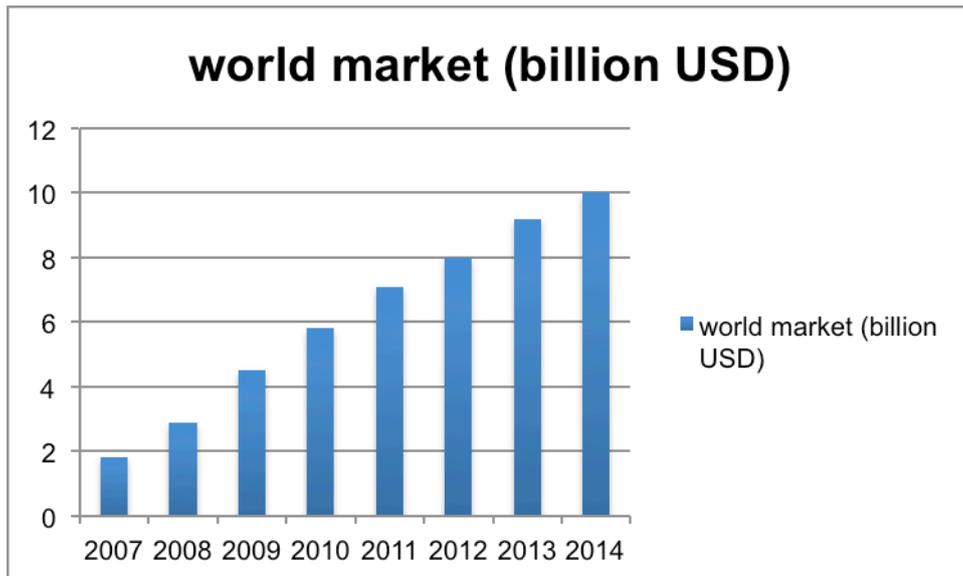


Figure 1. Market Potential. Reprinted from Kushki (2012), [26, 12]

As shown in figure 1, the world market of LBS is targeted to reach 10 billion dollars in 2014. Therefore, the application developers have integrated LBS into their applications to take part in the LBS market.

3 Location-Based Photo Sharing Application Breakit

The application, Breakit, is being developed for the company named Eighty Five Media Oy with six passionate and professional individuals, sharing the common goal of developing a location based photo-sharing application for iOS platform, which aims to change the current trend of sharing digital content. A short description of why and what the Breakit application aims to be is presented below.

3.1 Bridge Between Users and Media

Breakit team is building a mobile application that allows users to share pictures and show their stories with people nearby, instead of just friends and followers. It is an effective and easy way to inform others about current events and news, about the happenings that interest more people than just friends. These events and happenings are very relevant locally, but not necessarily on a larger scale. Breakit is not about sharing within a closed circle or group, but sharing information to nearby users. Breakit is integrated to social media and also co-operates with traditional media such as daily and weekly newspapers allowing photos to be published.

3.2 Target Customers

Breakit customers are the local publication houses such as Helsinki Sanomat and HBL in Finland. The current news trend has proven that the keeping up with the social media is one of the biggest challenges of the traditional media. At the same time, the current approach of getting contents from the user via email and Multimedia Messaging Service (MMS) is not the best approach as it consumes significantly more work in order for the authentication, and copyright issue of the picture. Therefore, Breakit offers all the media the real-time, rated information of what people want to read and share. The users can use the services to share photos and stories to publication media houses easily.

3.3 Competitors

There have been quite many location-based photo-sharing applications available in the market with a different focus compared to the Breakit application. Most of the applications share information within a closed circle of friends or within certain channels or subscribers. Moreover, users have to provide their data such as email addresses and passwords in order to access the services provided by other location-based application. It overrides the possibility of being anonymous and the possibility of missing out a wide range of users to deliver the relevant information at the right time. Every day more than 250 million pictures are being uploaded to the Facebook server alone and yet very few shares are relevant to users at the place they are.[17] Similarly, Twitter gets 177 million of tweets every day from their users. Yet information is not shared to relevant users at the right time. [18] On the contrary, the Breakit application shows the photo feed based on the current location of the user, which makes it possible to get the right information about all the happenings nearby.

4 Technologies Behind The Breakit Application

The Breakit application was built mainly on new technologies both on backend, frontend and on the mobile client. The backend server was installed in Amazon Server. The co-workers created all the supportive scripting for backend and frontend. The lists of major technologies used in the application are briefly described.

4.1 Backend and Frontend Technologies

The Breakit application is a location-based photo sharing application, which emerges the need of the backend server. There have been quite many backend server technologies being used for various purposes in various fields. The nature of Breakit is enormous in a sense that it has to handle all the upload of the stories, photos, comments, user credentials and other user generated contents in a timely manner. Therefore, the technologies below were adapted for the backend server.

4.1.1 Node.js

Node.js is relatively new technology compared to PHP, Ruby, Java, Python and other web technologies. It is a wrapper around the high-performance V8 JavaScript runtime provided by the Google Chrome browser. Since it is built on top of JavaScript, it uses an event-driven, non-blocking I/O model, which makes it lightweight, efficient and perfect for a data-intensive real-time application Breakit. [3]

The introduction of event-driven and non-blocking libraries has made Node.js more robust in manipulation of binary data in the backend. Similarly, the introduction of various external libraries such as Backbone.js and Express has made Node.js extremely extensible with no limits. [4,3-4] Some of the key points that led to adopting Node for the Breakit application are presented below.

- a. It is built on top of JavaScript.
- b. It is an extremely extensive, scalable and high performance web server.
- c. It supports various type of database from traditional SQL to modern MongoDB and memcache such as Redis.

- d. It supports on various linux variant platforms and windows.
- e. It has various APIs available.
- f. It is an open source project and has a huge open community for support. [4,3-4]

4.1.2 MongoDB

MongoDB is a scalable, high-performance, open source document-oriented database, which is written in C++ and developed and supported by 10gen. MongoDB stores the structured data as a JSON-like document with dynamic schemas called BOSON (a binary adaption of JSON) unlike the traditional relational database that stores the data in a table. MongoDB stores the incoming records in the memory. Thus, it is non-blocking and less time consuming to insert data into the database [4]. Some of the features that led to choosing MongoDB for the project are listed below.

- a. Document-oriented storage: Storing data in the form of schema rather than in a table
- b. Full index support: An index is a data structure that allows a programmer to quickly locate documents based on values stored in a certain specified field.
- c. Replication: Database replication ensures redundancy, backup and automatic failover.
- d. Auto-Sharding: Sharding allows users to divide a collective document within a database to distribute the collective document across a number of mongoDB instances or shards.
- e. Support of JavaScript on Query: With the standard query commands, the support of JavaScript enhances the read operation in the database.
- f. GridFS: It divides a file into parts or chunks and stores each of those chunks as a separated document that provides the availability of storing files of any size. [5]

4.1.3 HTML5, CSS3, CoffeeScript

Hyper Text Markup Language, in short HTML5, is the core technology being used on the Internet. It structures and presents the content of the World Wide Web for various browsers. In the previous versions of HTML, playing audios and videos were mainly handled by plugins such as Adobe Flash Player, Apple Quick Time but the introduction

of elements such as audio, video and canvas has made it possible to integrate multimedia content without having to restore proprietary plugins and APIs.[5]

The need of HTML5 for the thesis project was providing an interface for various browser users in different platforms. Therefore, a well-formed presentation of the pictures, comments and maps is necessary. To achieve the well-formed presentation Cascading Style Sheets CSS version 3 (CSS3) was used. CSS is a style sheet language that is used for formatting the content written in mark-up language. The most common use of CSS is to style web pages that are written in HTML and XHTML. However, CSS can be applied to XML, SVG and XUL. In the development practice of webpages, the separation of elements such as the layouts, colours and fonts are important and improve the content accessibility and more flexibility with a fewer number of codes.[6]

The nature of the Breakit website is that the contents should be shown according to the users location and can interact with users. This demands the nature of the webpage to be dynamic rather than static. Therefore, the CoffeeScript scripting language was used. The history of CoffeeScript dates back to 2009. It is a programming language that transcompiles (source-to-source) to JavaScript. The advantage of writing CoffeeScript rather than JavaScript is that it compiles non-error-prone (error less) JavaScript codes. Similarly, the CoffeeScript syntax is more human-readable than that of JavaScript. [7]

4.2 Mobile client

The type of the mobile for the project is Apple mobile, supporting from iOS 5 and later. Therefore, the entire mobile client architecture is focused on Objective-C and the requirements are described below.

4.2.1 iOS Software Development Kit

The iOS Software Development Kit was developed by Apple Inc and released in February 2008 to provide a means for worldwide application programmers to develop various types of applications that can be run in iOS devices. The introduction of iOS SDK has not only produced more than 800,000 applications to date but also a huge economy in various forms. Some of the indispensable tools for developing any application for iOS platform are described below. [8]

a. Xcode

Xcode is the integrated Development Environment (IDE) used for developing any application for Apple devices. Objective-C is the main language used by Apple for the OSX and iOS operating system and the APIs such as Cocoa and Cocoa Touch. It is a high-level, object-oriented programming language written in the same fashion as the C programming language with the header file dot h(.h) and its implementation in dot m (.m). The Xcode is powerful enough to find the lexical error in real time. The writing of code is less time consuming in the sense that it provides multiple options with methods and duplication of methods or variables. It seamlessly integrates code editing, user interface design with interface builder, debugging and testing all in the same window. Further more, the availability of getting official documentation makes it more user-friendly and the introduction of Automatic Reference Counting (ARC) has made the platform even more easier to program. [9]

b. Interface Builder

The Interface Builder is the visual tool that developers can use to create all the interfaces the application needs without having to write down a single line of code. All the interfaces are saved as a.nib or .xib file. If the project has been enabled with storyboard, it saves the file under x.storyboard. It has pre-processor directives such as IBOutlet and IBAction to deal with individual controls and methods that respond to events respectively. Unlike in other traditional IDE where everything has to be hard-coded, the Interface Builder allows developers to create appealing interfaces just by dragging the objects and media files from the Library Pane and dropping into the Interface Builder's canvas. [11,9],[12]

c. iOS Simulator

The iOS Simulator is the tool that is used for testing of an application as it simulates the behaviour of real iOS devices. It can simulate three devices (iPhone, iPhone with Retina Display, iPad) with various iOS versions. During the development phase of any application, it is widely recommended to test the performance of the application under iOS Simulator rather than on a real device as it might misconfig-

ure the device system, which might have been caused from the threads or leaks from the application.[11,9],[13]

d. Instruments

The Instrument is a performance, analysis and testing software tool for dynamically tracking and profiling the application code. The use of Instruments in any development phase is that it helps developers to track down the difficult-to-reproduce issues, performance issues, and stress test on parts of the application. The modules such as Allocation, Leaks, Profile and Monitor not only tests the application but also help developers to gain a deeper understanding of the operating system and application flow. [14]

4.2.2 iOS Frameworks

According to Apple Inc, “A framework is a directory that contains a dynamic shared library and the resources (such as header files, images, helper application, and so on) needed to support that library”. [15] The iOS Frameworks are the collection of the software, which runs on top of the operating system. Figure 2 illustrates the iOS technologies as a set of layers.



Figure 2.iOS Technology Overview. Reprinted from Lee (2012) [11, 11]

Figure 2 is a graphical presentation of the iOS architecture. The lowermost layer is Core OS and the topmost is Cocoa Touch for iOS and Cocoa Framework for Mac OS X. The Cocoa Touch framework is used for developing an application for iOS devices

whereas Cocoa is used for developing programs for Mac OSX. The primary difference between these frameworks is that Cocoa Touch implements all the touchscreen functionalities, for example, tap, swipe, pinch, scroll, drag and pan. All the iPhone applications are preferably written in Cocoa Touch Framework as the high-level layer framework provides object-oriented abstraction for the lower-level construct. Cocoa Touch Framework is the key framework to develop any iOS application as it defines the overall application infrastructure and supports high-level features. Some of the technologies that Cocoa Touch provides are presented below.[15]

a. Auto layout

Auto layout is introduced from iOS version 6. The benefit of this feature is that it keeps the layout as defined, commonly known as constraints. [15]

b. Storyboard

Storyboard was introduced from the iOS version 5. It is a GUI part of the iOS project configured with Storyboards. Unlike in Xib files, Storyboard enables developers to design the entire application interfaces in one place and it provides the overall workflow of the application as all the views and view controllers are placed in one place with various types of relationships. [15]

c. Multitasking

The Multitasking feature was introduced in iOS 4. It enables applications to be switched into various states rather than forcing to be closed or halted. The possibility of changing states have made it possible to run an application in the background and do their tasks such as Notification.[15]

d. Apple Push Notification Service

The Apple Push Notification Service was introduced in iOS 3. It allows the application to interact with users with the notification even if the application is running in the background. This service is dependent on the Apple Push Notification Server as the request is relayed from it.[15]

e. Local Notification

The Local Notification feature was introduced in iOS 4. It also alerts the user with a local notification triggered by the application without communicating with

external servers. The benefit of the Local Notification is its nature of being independent and once it is triggered, the system takes care of its delivery.[15]

f. Gesture Recognizer

The Gesture Recognizer was introduced in iOS 3.2. It detects the common types of gestures such as tap, pan, swipe, long pressed, drag and rotation of the view that the Gesture Recognizer is assigned with. [15]

As presented in Figure 2, the Media layer is the second layer in the iOS technology, which is responsible for handling all the multimedia such as audio, video and graphics used in the iOS application. This layer consists of Graphics, Audio, Video and Airplay technology. The Graphics technology is used for creating simple images to high-quality images. Developers can use various available frameworks as per their need. For example, the Quartz framework is used for handling the entire native 2D vector and image-based rendering, and the Assets Library framework is used for accessing photos and videos. Similarly, the Core Animation framework is responsible for animating views and other visual contents. The Graphics technology not only contains the frameworks for images but also the text layout and rendering engine called Core Text. [15]

The Core Services layer consists of the kernel environment and drivers including the basic interfaces of the operating system. It includes all the fundamental system services such as memory management, file system handling and threads. It comprises some of the powerful programming technologies such as Block, Grand Central Dispatch, Core Data and Core Location.[11,12]

At the bottom of the iOS architecture lies Core OS, which is the foundation framework for upper layer frameworks. This layer provides the interfaces to access many low-level features of the operating system. The Core OS comprises the following frameworks.

a. Accelerate Framework

It is responsible for providing an interface to calculate linear algebra, DSP and image processing calculation.

b. Core Bluetooth Framework

It is responsible for providing an interface to interact with the Bluetooth technology.

c. External Accessory Framework

It is responsible for establishing communication with hardware accessories connected to iOS devices.

d. Generic Security Service Framework

It provides a standard set of security-related services to the iOS application.

e. Security Framework

It is responsible for managing digital certificates, public and private keys and trust policies.

4.2.3 iOS Third Party Open Source Projects

The nature of the open source projects is that redistribution and access to the end product from the project is available via the Internet with no cost and void guaranty and warranty. Usually, such projects hold licences such as the Massachusetts Institute of Technology, Berkeley Software Distribution BSD and Apache.[16] Some of the open source projects used in this thesis project are presented below.

a. ASIHTTPRequest

It is a project for iOS client to communicate with a backend server. The Breakit application requires communication with the server at various stages. Therefore, instead of creating the `NSURLConnection`, the available asynchronous block methods of `ASIHTTPRequest` were used. Some of the key features of the project are presented below.

- Provides a straightforward interface for submitting data to and fetching data from webservers.
- The downloaded data can be stored into file or into memory.
- Full support of caching and cookie data.
- Support for digital certificates.
- Provides synchronous/asynchronous request to servers.
- Easy delegation to `UIProgressView` to indicate download or upload progress.
- Easy access to request and response HTTP headers.
- Resume for partial downloads.
- Possibility of canceling delegated object whenever needed.
- As fast as `NSURLConnection`. The requests are implemented in concurrent `NSOperation`.

b. HPGrowingTextView

It is a project for iOS client that resizes the UITextView. The default behaviour of UITextView is that, the height of the frame is defined in advance and cannot be increased or decreased. However, the HPGrowingTextView is the work around with UIView and subclass of UITextView that lets user to define the limit to grow and vice versa. Some of the key features of the project are mentioned below.

- Resize the frame of UITextView as the user input text cannot be fit into a line.
- The height of the UITextView's frame can be limited to certain numbers of lines.

c. EGOPullToRefresh

It is a project for the iOS client that animates the UIActivityIndicator. Some of the key features of the project are presented below.

- Adds the animated view on top of the UITableView, which is visible only when the user pulls down from UITableView's origin.
- Well documented.

d. ZUUIRevealController

It is a project for iOS client that contains a container to hold the UIViewController. It mimics the behaviour of UISplitViewController.

- Holds two ViewControllers (front viewController and rear viewController);
- Has pre-implemented UIGestureRecognizer on both viewControllers.
- Automatic Reference Counting enabled

e. Reachability

The Breakit application requires communication with the server at various stages. Thus, the Reachability class is used to let users know about their Internet connectivity before and during the communication with the server. Some of its features are shown below.

- A project provided by Apple Inc. for checking the Internet connectivity.
- Methods are easier to implement.
- Recommended by Apple.

f. AQGridView

In the Breakit application, the photos are kept in an album if the Foursquare places are chosen. Hence, all the photos belonging to a particular Foursquare place are presented in a grid with three pictures on the row. Some of the features of the project are given below.

- Based around the programming model of UITableView and its associated classes. Thus, simple to use.
- Creating a grid is flexible.
- The grid is selectable with animation.

g. MixPanel

It is a project for the iOS client that keeps track of the user's behaviour on the application. The users generate the contents of the Breakit application that is photos and comments. Therefore, it is necessary to keep track of the user's behaviour on the provided services. By tracking, the Breakit application would know which features of the applications are being used the most and the less and unused function can be removed from the application in the next update. Some of the features of the MixPanel project are listed below.

- Easier to integrate into project as online help and documentation are available.
- Easier methods to be implemented.
- Secure
- The tracked data can be viewed in browsers.
- Helpful for the future planning.

5 Development Of iOS Mobile Client

As described in chapter 3, the Breakit application was built for EightFive Media Oy to provide a new channel to facilitate users to share their content in an easy manner, emphasizing location rather than the user itself. Since the application was developed in the production environment, the software development tools such as SCRUM, MoS-CoW agile development were used.

5.1 Analysis And Design Details

In order to develop any application, the analysis of the features of the application has to be tested and validated in various ways. In order to validate the features of the application, several interviews were conducted with different people in the age group of 13 – 40 years old at different places in Helsinki. Similarly, the company participated in some of the biggest technical events conducted in Helsinki such as Summer of Startup 2012 and Slush 2012. Some the features that were verified with potential users of the application are listed below.

- picture feed: the general appearance of the picture was discussed, such as the dimension of the picture, its position in the application.
- distance between the user and the photo taken: the importance of showing the distance between the user and the photo taken were discussed. Furthermore, the benefit of the providing such data in the LBS application was analysed.
- headline of the picture and time taken: Some text as the headline to reflect the picture was discussed. The size of the headline and its position in the application were also taken into consideration.
- pictures as an album and display as the carousel: The method of multiple pictures taken at the very same place was discussed along with its style of presentation.
- annotated map of the picture taken: The ability to view the place into the map was discussed.
- comment on the picture: The length of a comment for any picture was discussed.

- disliking the photo: The possibility to dislike a picture and its positive and negative impact on the application users were analysed.

There are several design patterns that have to be followed to create any application under the iOS environment. A design pattern is similar to a guide to resolve a particular problem. The most important design patterns are presented below.

a. Model View Controller (MVC)

The Model-View-Controller design pattern is central to any good iOS or Mac app. The Model classes encapsulate all the data needed in the application and define the logic. The View classes are the views that the users interact with. These View classes display the encapsulated data from the Model classes and the only way to access those data are from Controller classes. A Controller class is the collection of views that responds to Model class and View class. The view controller acts as a bridge between Model and the View class.

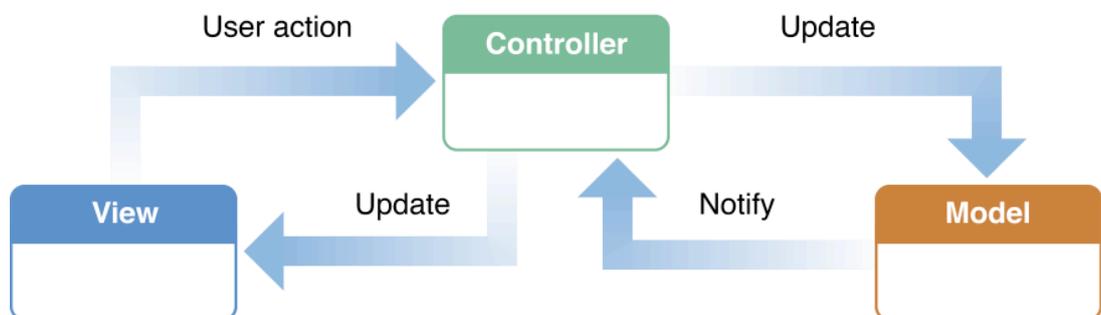


Figure 3. MVC Design Patterns. Reprinted from Apple Inc.[19]

Figure 3 is the graphical representation of the work-flow of MVC Design Pattern. Some of the benefits of using MVC are as follows.

- Extensible
- Reliable
- Readable
- Reusable

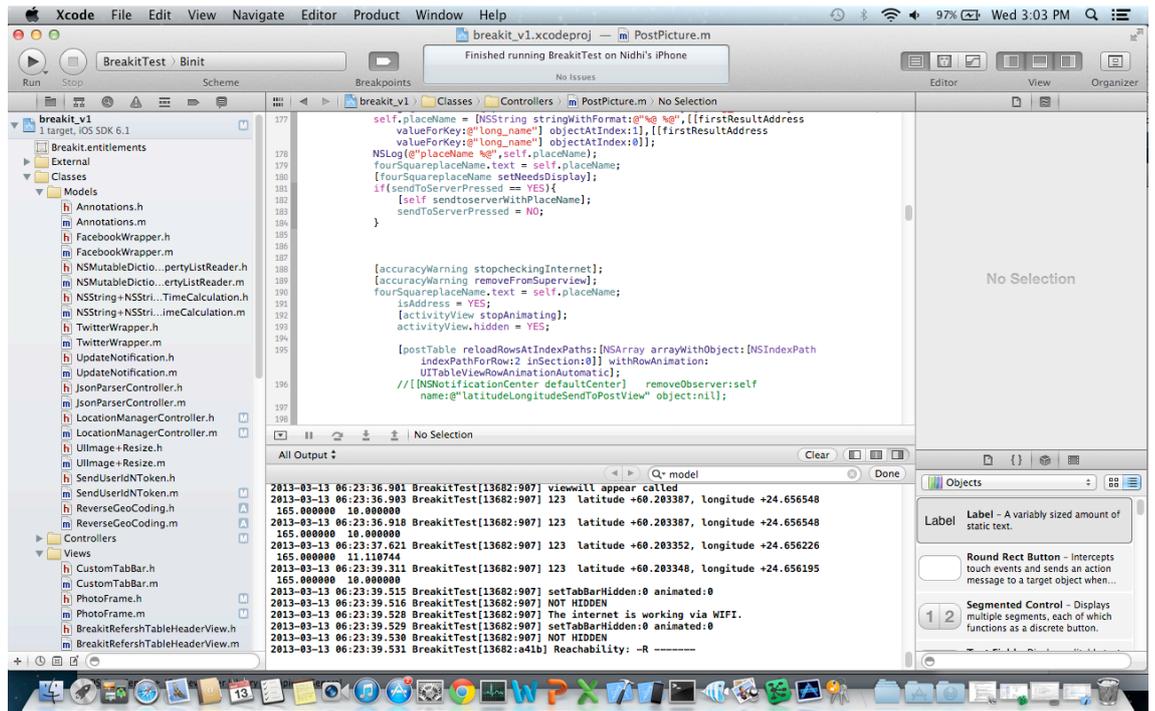


Figure 4. Model View Controller Design Pattern Implementation.

Figure 4 is the implementation of the Model View Controller Design Pattern in the Breakit application. The classes such as Annotation, Facebook and Twitter are used as Model class and BreakitRefershTableHeaderView, PhotoFrame classes are used as view classes.

b. Delegation and Protocol

Another common yet powerful design pattern is Delegation. In Delegation, an object is commonly called a delegate that acts on behalf of, and at the request of, another object, which is typically the framework object. During the execution of the application, at some point the framework object sends a message to its delegate object to respond. The concept can be illustrated as follows.



Figure 5. Delegation. Reprinted from Apple official documentation (2013) [20].

Figure 5 is a graphical representation of the work-flow of Delegation. To elaborate it more, take an example of a click on the close box of the window. As soon as the user clicks the close box, the window manager sends the delegate `windowShouldClose:` and the delegate returns the Boolean value about closing the window. The delegate always acts in coordination with the host object, meaning that the delegate behaves program-specifically. The framework object seeks a delegate to perform step number 3 as shown in figure 5. [20]

A protocol is a set of the declaration of methods that any class can implement to access those declared methods. An instance of a class associated with the protocol calls the methods of the protocol and obtains values returned by the class formally adopting and implementing the protocol. Some of the benefits that can be achieved by adopting a protocol can be listed as follows.

- “To declare methods that others are expected to implement
- To declare the interface to an object while concealing its class
- To capture similarities among classes that are not hierarchically related.”[21]

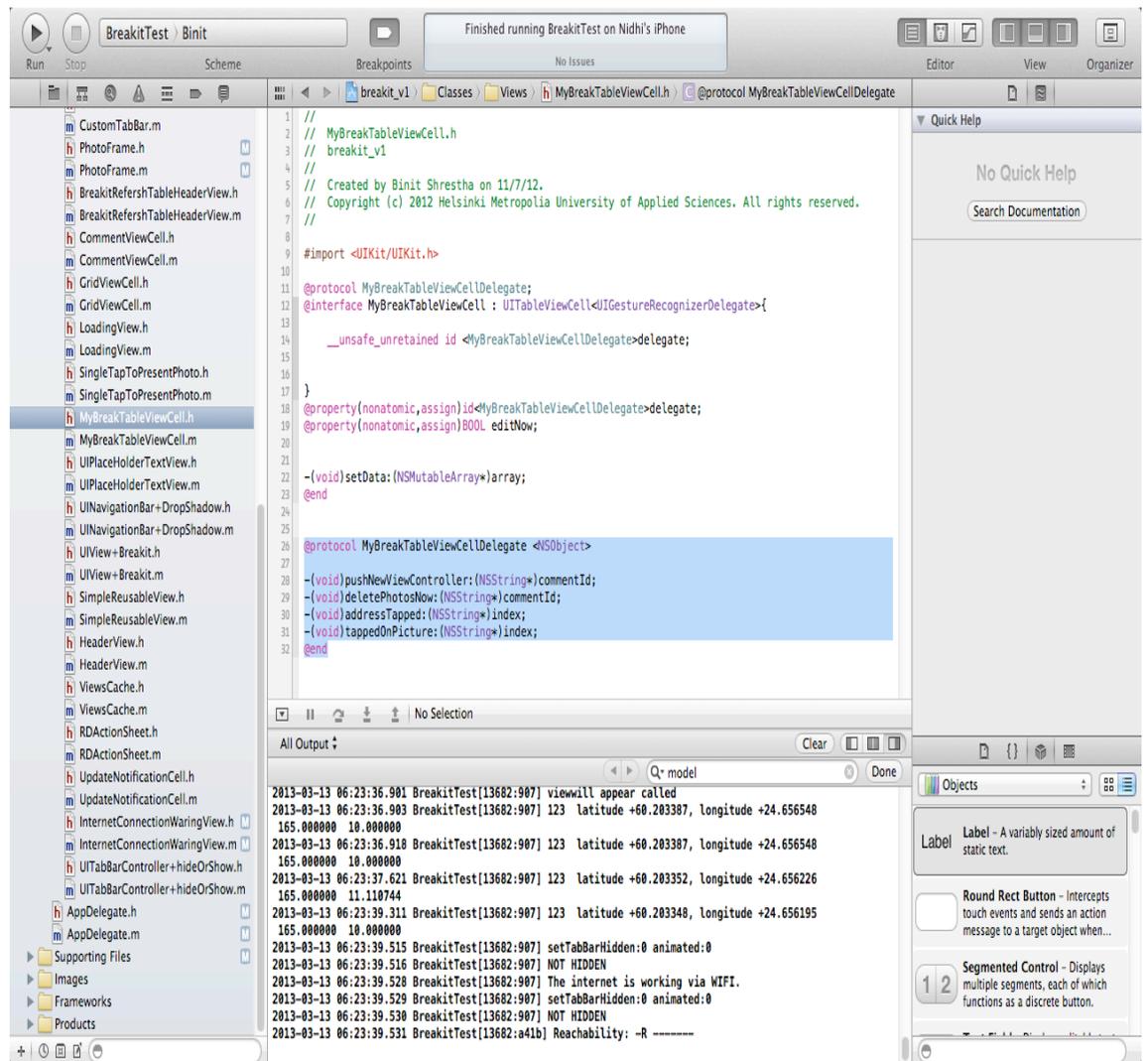


Figure 6. Implementation of Protocol Design Pattern in Breakit application

Figure 6 is a graphical presentation of protocol implementation. In MyBreakTableViewCell class the protocol named MyBreakTableViewCellDelegate is created having four methods and an instance is created named delegate. Whenever the appropriate events such as tap on the picture of the UITableViewCell is invoked, the delegate object will respond to its implemented class object with designated methods.

c. NotificationCenter

A Notification Center is a subsystem of the Foundation system that broadcasts the messages to all objects in the application that are registered with an event in Notification Center. A notification informs the observer that the event has happened or is

about to happen, thus giving the observer an opportunity to respond in an appropriate manner.

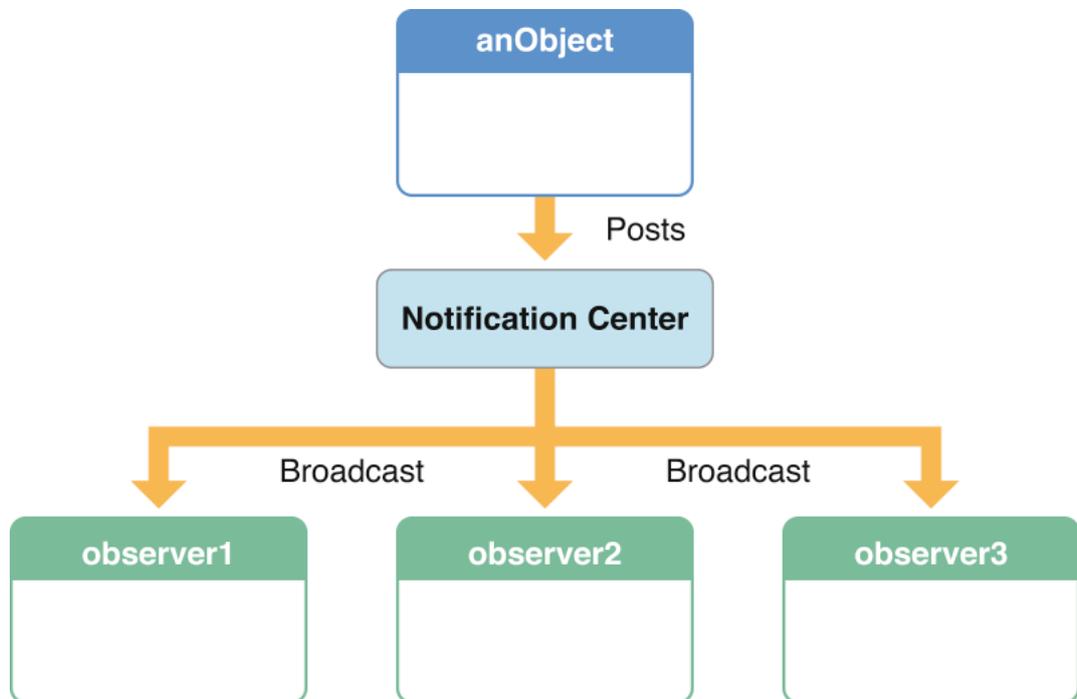


Figure 7. Notification Center Design Pattern. Reprinted from Apple Inc. [19]

Figure 7 is a representation of the Notification Center workflow. Basically, the notification is registered with the name of the notification as an identity along with the dictionary object and the observer event. The dictionary object can be some value or nil value. Whenever, an object post a notification to notification center, it broadcasts the message to enable the observer event that is registered with the notification.

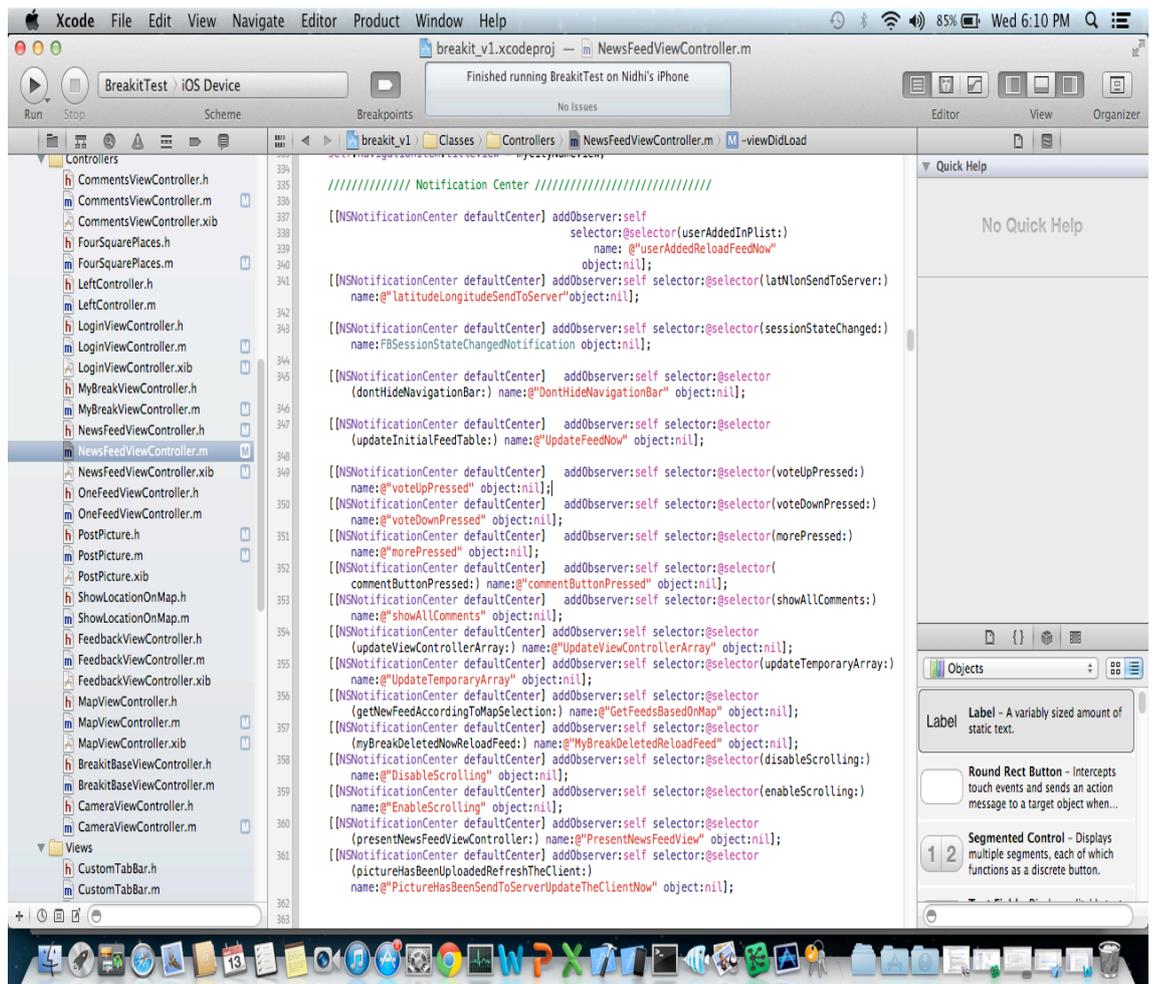


Figure 8. Implementation of Notification Center in Breakit application.

Figure 8 is a graphical presentation of the implementation of Notification Center used in NewsFeedViewController class. The idea behind the use of the Notification Center is that a custom object can define and post its own notifications and another custom object can observe the notification.

- d. View hierarchy: It represents the hierarchy of the views in the application. The topmost view is the window object and other views are kept under it. The views that are added to the content view become subviews of it. The view hierarchy is important in event handling and any drawing in the application. When a window is asked by the system to prepare for the display, all the superviews are asked to render themselves before their subviews.

- e. **Responder chain:** It is a series of objects typically views or view controllers and windows. It is a mechanism for views to cooperate events in appropriate manner. A responder is an object that responds to events and handles them. All the responder objects are eventually inherited from UIResponder under iOS and NSResponder class in OS X. The visible objects of an app are almost always responders such as windows, views and view controllers. The responder chain enables cooperative event handling. Whenever the object in the responder chain cannot handle the event or action, it passed the message to the next responder to the higher-level object if found otherwise the message is discarded by the system.

- f. **View Controllers:** It is a collection of views that acts as the bridge between views and the model of the application. It contains the application logic on how to handle user's inputs. It is one of the fundamental classes for any iOS applications.

- g. **Receptionist:** According to Apple – “In the Receptionist pattern, work that an app is performing is redirected—or “bounced”—from one execution context to another. (An execution context is a dispatch queue or operation queue associated with the main thread or a secondary thread.) You apply the Receptionist pattern primarily in situations where work performed in a secondary queue results in a task that must be performed on the main queue, such as operations updating the user interface.” [19]

- h. **Category**
It is a way to extend a class by adding methods to it. The implementation of category classes avoids the subclassing of an existing class. The Breakit application carries many category classes such as NSString+NSString_DateNTIME, UIImage+Resize. The NSString+NSString_DateNTIME class is shown below.

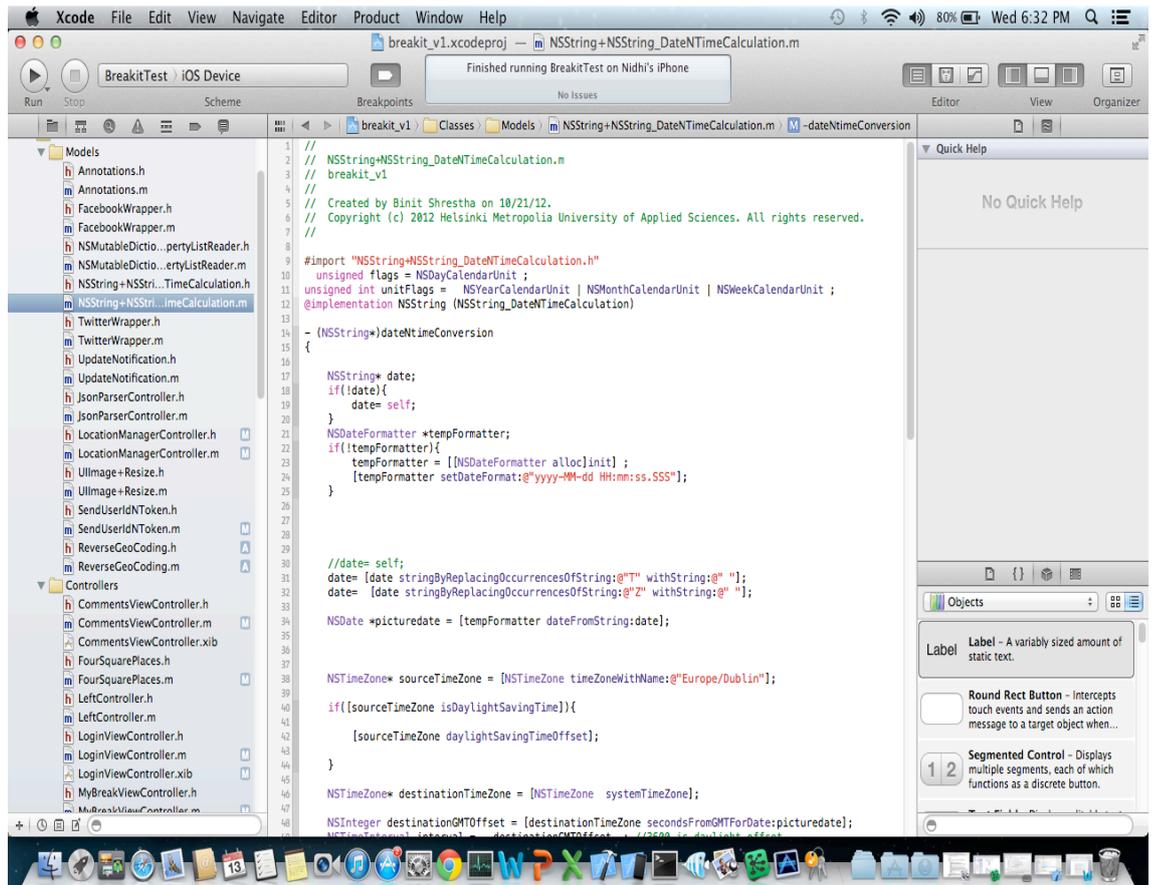


Figure 9. Implementation of Category class for NSString

Figure 9 is the implementation of the Category class for NSString in the application. The NSString+NSDateNTimeCalculation category class converts the NSString into NSDate Format, calculates the time difference between the converted time and current system time and returns the difference time in NSString.

5.1.1 User's Perception

The registration of user with a unique nickname required to access the Breakit application. Along with that, the permission to use users' current location and Apple Push Notification Service permission are vital to run the application. Excluding the permission to use Location Services will result in a blank presentation.

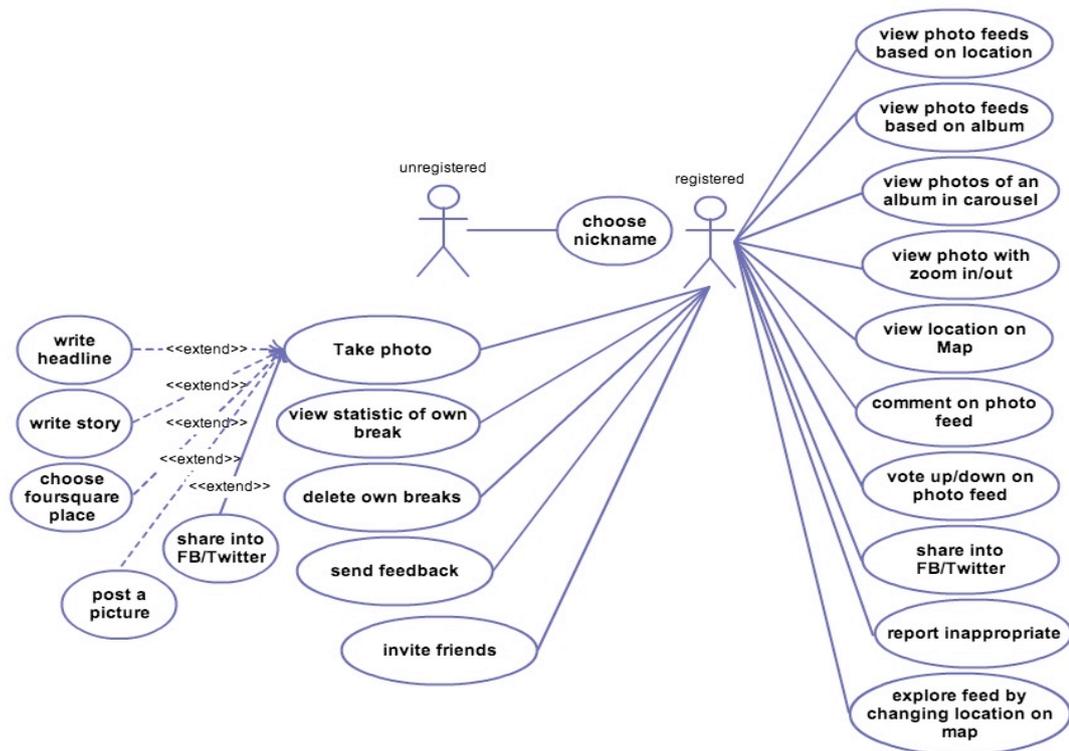


Figure 10. Use Case Diagram for the application.

Figure 10 is the use case diagram of the application showing the services that authorized and unauthorized users can get from the application. The initial launch of the application will provide a view to create a unique nickname for user registration. Along with that the Apple Push Notification Service (APNS) will ask permission to be used in the application. The services that user can get in the application are described below.

- a. View photo feeds based on location: The registered users can view the photo feeds based on the current location. The photo feeds are arranged in a way that the nearby feeds will be shown with respect to its rating and views by other users. The highest rated photo feeds will be shown at first even if there are multiple pictures from the same place.
- b. View photo feeds based on album: The user can view all the photos taken at the particular place with its Foursquare Place Name. For example, the photos

taken at Metropolia University, Lepävaara branch with its FourSquare Place Name will be formed as an album. Thus user can view all the photos along with annotated map by tapping on location name shown in the Photo Feed.

- c. View photos of an Album in a carousel: The user can view all the photos and its respective contents of an album by swiping left and right.
- d. View photos with zooming in/out. A tap on the photo will provide a separate view of photo with the possibility of zooming in or out.
- e. View location of the on the Map: The annotated map is presented in another view as the result of tapping on the location name from Photo feed.
- f. Comment on Photo Feeds: The user can comment on photos shown in the feed.
- g. Vote up/down on Photo: The user have possibility to like or dislike photo by tapping on thumbs up or thumbs down icon on Photo Feed. It will change the icons into percentage of likes and dislike.
- h. Share photo on Facebook/Twitter: The user can share picture link to their Facebook or Twitter timeline.
- i. Report Inappropriate: The user can report any photo as inappropriate content. It will lead to disappearance of the photo from the Photo feed if the report hits the limit set in the backend.
- j. Explore feed by changing location on Map: The user can change their location and get the feed accordingly by tapping on the Explore button kept at the right corner of Photo feed view, which opens the annotated map showing the users current location.
- k. Take Photo: A single tap on the camera icon opens the camera view and picture can be taken. After taking the picture, it will lead to Post view where user can write down the headline for the picture and some comments. At the same time, the address of the picture is also shown with the possibility to change it in-

to foursquare place names. The tap on Add places button opens another view with the list of foursquare places. The selection of foursquare place will take the user back to the Post view with the change in address of the picture. After that, the user can post a photo by tapping on Breakit button. If users like to share the photo their Facebook and Twitter timeline at the same time while sending to Breakit server, the switch buttons of Facebook and Twitter has to be enabled. The successful posting of a picture will lead to the PhotoFeed view with the recent updates.

- l. View statistics of one's Breaks: The user can check the statistic of their own breaks by tapping on dummy user profile icon kept at the bottom of the Photo Feed view. The user can see the total number of views, upvotes, downvotes and comments done in any particular breaks.
- m. Delete one's Breaks: The user can delete their breaks via delete button. The delete button is placed in MyBreakView.
- n. Send Feedback: The user can send the feedback to Breakit team by tapping on feedback row. The possible ways to get to feedback row are tapping on the menu icon kept at the left corner of the navigation bar or by swiping a view with a finger from left to right.
- o. Invite Friends: The user can invite their Facebook friend by tapping on invite friends row.

Figure 11 shows the sequence overflow of how objects interact within a given situation. Though there are several application features that have to be shown in the sequence diagram, only the main feature has been portrayed below.

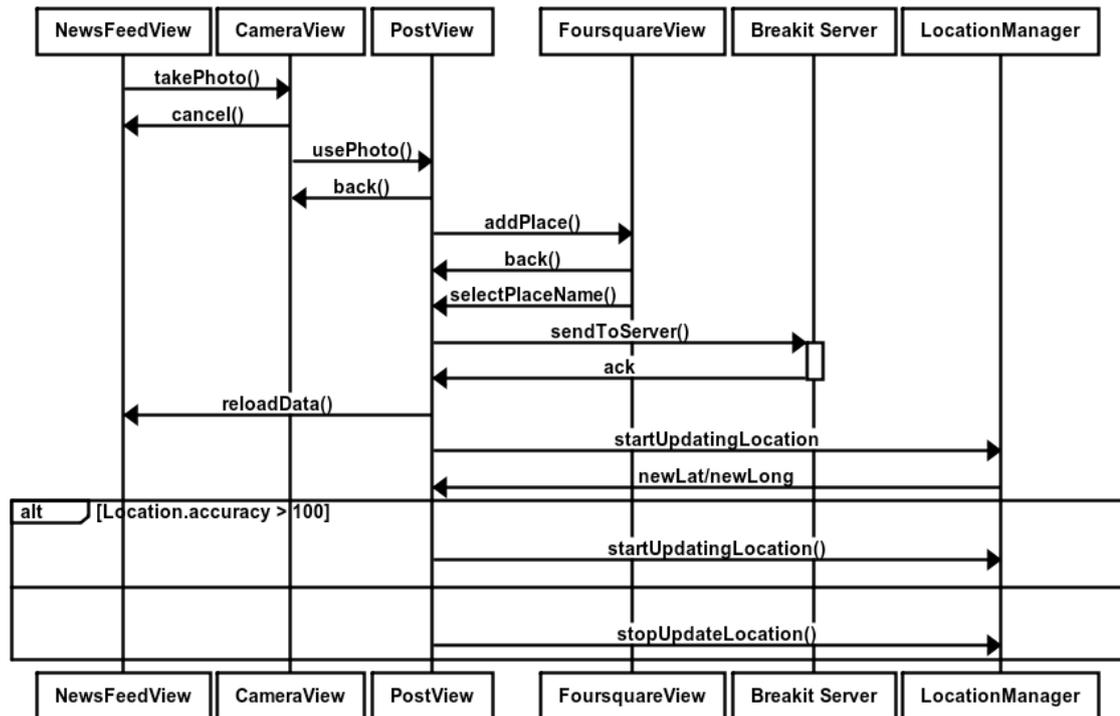


Figure 11. Sequence Diagram of photo feed feature

Figure 11 is the sequence diagram of a photo feed feature. It shows how the user can post a picture to the Breakit server. The takePhoto method invokes the camera view where user can either take a picture or cancel the camera view. After taking the picture, the startUpdatingLocation method of the LocationManager class is invoked for relatively new latitude and longitude values which are used for reverse geocoding. If the vertical accuracy of the newly received latitude and longitude value is greater than 100, the startUpdatingLocation method is again invoked and the cycle repeats until the vertical accuracy doesn't come below hundred. The user can use the foursquare places instead the reversed geo coded location. The addPlaces method invokes the foursquare view where the currently available places of the given latitude and longitude are shown which are provided by foursquare API. The user can select the place just by tapping on the name of the given list. After having the place name and adding headline to the picture, user can post it to Breakit server by tapping on the Breakit button, which invokes the sendToServer method that sends the headline, comment, latitude, longitude, usernickname, userId, foursquareId and placename to the server. Upon the successful response from the server, the current view in the navigation stack is popped out and NewsFeedView's tableview is reloaded and shown. If not, the error description is shown in the alertview.

5.1.2 Authentication Workflow

In the process of authentication, the user ID is being extracted from property list and sent to the Breakit server as a POST method. The server then verifies the user ID and sends back the acknowledgement. The detailed procedure is shown below.

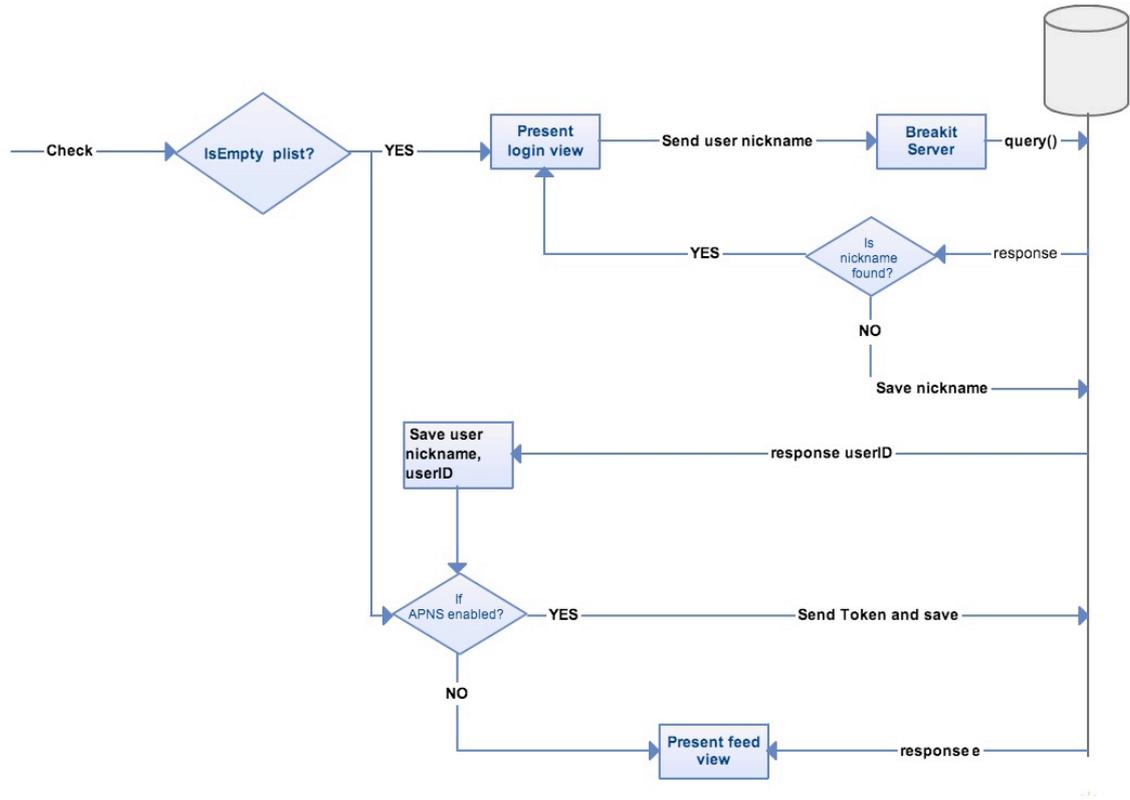


Figure 12. Breakit user authentication flow

Figure 12 is a graphical presentation of user authentication. From the user's point of view, the user authentication flow can be divided into three aspects.

a. New User Creation without APNS enabled:

Whenever the application is launched, it checks the status of the property list (a file where the user information is stored). If the property list is empty, it opens the Login View to the user. The Login view is the view where users have to come up with unique nickname. A nickname can contain alphanumeric characters except the white spaces, arithmetic operators and special characters. Upon the successful nickname creation, the " Let me in " button will appear in the view. By pressing the " Let me in " button, it sends the nickname to server as a

post method. The server makes a query to database to check its existence and sends back the acknowledgement to the client. If the user's nickname is not found in the database, it saves the nickname along with automatically generated user ID and sends the user ID as a successful response. In the case of nickname found in the database, it sends the failure response and the users have to repeat the procedure of creating new unique name again. Upon, the successful response, the application stores the user ID and nickname to the property list and the present the Photo view.

b. New User Creation with APNS enabled:

Once, the user gets the successful response along with `userId`, the server asks the client to send the device token. The client sends the device token and upon the successful acknowledgement it stores the device token string in the property list and presents the Photo view.

c. User login:

The normal login flow begins with checking of property list being empty. If property list is found empty it opens the Login view. If the property list is not empty and the user has enabled APNS then sends the device token to server and the opens the Photo Feed view.

5.2 Application Implementation

5.2.1 Design Implementation

The implementation of an application has to be followed with the designed pattern guided by the Apple documentation. Hence, the application is logically and by its functionality divided into Model, View and Controllers.

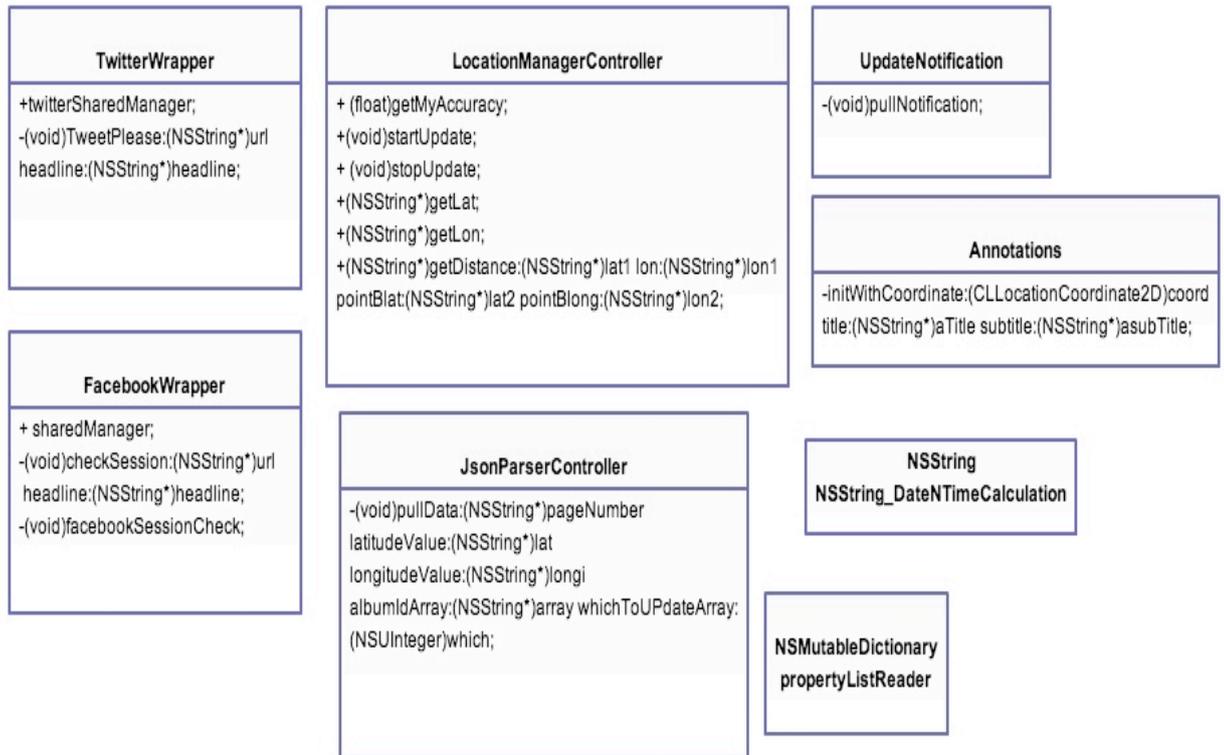


Figure 13. Model Classes of Breakit application

Figure 13 is the presentation of Model classes used in the application. One of the most important requirements to use application is enabling Location Service. The relatively new latitude and longitude of the user's current location is handled in LocationManagerController class.

```

+ (void)startUpdate:(NSUInteger)locationWhatFor{
    locationUpdateForWhat = locationWhatFor;
    counter = 0;
    hasLocation = false;
    hasError = false;
    if (!loc) {
        loc = [LocationManagerController alloc];
        [loc initLoc];
    } [locationManager startUpdatingLocation];
}

#if __IPHONE_6_0
-(void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations{
    CLLocation* location = [locations lastObject];
    CLLocation* oldlocation = [locations objectAtIndex:0];
    NSDate* eventDate = location.timestamp;
    NSTimeInterval howRecent = [eventDate timeIntervalSinceNow];
}

```

```

if (abs(howRecent) <5.0) {
    if (location.coordinate.latitude != oldlocation.coordinate.latitude ) {
        lat = [NSString stringWithFormat:@"%f", location.coordinate.latitude];
        lon = [NSString stringWithFormat:@"%f", location.coordinate.longitude];
    }

    if(locationUpdateForWhat == 0){
        [[NSNotificationCenter defaultCenter] postNotificationName:@"latitudeLongitudeSendToServer" object:nil];
    }
    myAccuracy = location.horizontalAccuracy;
    if(locationUpdateForWhat == 1){
        counter++;
        if ((location.horizontalAccuracy<=
100&&location.verticalAccuracy<= 25) || counter>5) {
            [[NSNotificationCenter defaultCenter] postNotificationName:@"latitudeLongitudeSendToPostView" object:nil];
        }
    }
}
}
#endif

```

Listing 1. Getting latitude and longitude from the LocationManagerController class (Objective-C code).

As shown in listing 1, the invocation of class method `+(void)startUpdate:(NSUInteger)locationWhatFor` will initialize the core location manager object and start invoking `startUpdatingLocation` delegated method. The logic to get relatively new and accurate latitude and longitude is implemented in the delegated method `locationManager:didUpdateManager:`. Whenever the `locationManager:didUpdateManager:` is invoked, it provides the latitude and longitude values. To ensure the application gets recent latitude and longitude, the threshold of 5 second is set to prevent of using values that are 5 seconds old. The accuracy is the key point on getting the latitude and longitude, which varies on the time the core location manager object is let running and the distance between the mobile communication service provider's towers or the WiFi routers.

When a users, takes a photo from the application, the accuracy of the latitude and longitude of the user's current location is the key figure to determine the reverse geocoded address (an address name or place name which is retrieved from latitude and longitude value). In order to provide the best nearby location as possible, the flag value as 1 is used to get at least less than equal to 100 meters of horizontal accuracy and 25 is

set with the counter value. The need of the counter value is such that the CoreLocation Framework itself handles the invocation of the delegated method and thus there is no guarantees of immediate invocation of the delegate method. Once the counter limit hits 5 or the accuracy is less than equal to 100 and 25, it triggers the notification to an other class.

Another important model class is the NSString category class (NSString+NSString_DateNTIMECalculation) that converts the NSString into NSDate and provides the difference between the converted NSDate and the system date. The class JsonParserController takes care of the communication between the server and the client.

Whenever the application is launched or pulled to refresh new updates, the location manager is called for relatively new latitude and longitude. After getting the latitude and longitude it triggers the JsonParserController class methods to send the latitude and longitude to server to get the nearby feed. Upon, the successful communication, it converts the server replied data into object and conveys to other classes. The NSString category class takes all the data related with date and provides its differences with the current system time, which is shown in the NewsFeedController.

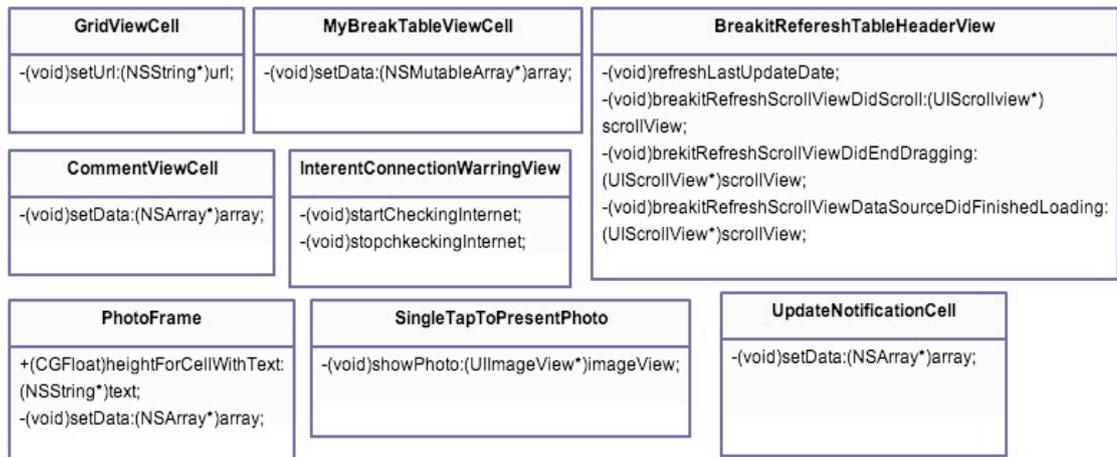


Figure 14 View classes of Breakit application

Figure 14 is the collection of the view classes that works on top of the viewcontrollers default view. The general purpose of the presented view classes can be described as follows.

- a. GridTableViewCell is a class responsible for creating the grid view for the pictures associated with the album. This view is invoked when users taps on the place name from NewsFeedViewController.
- b. MyBreakTableViewCell is a subclass of UITableViewCell that is responsible for creating the cell to present the statistic of the users own break. It is invoked when user taps on the user icon presented on NewsFeedViewController.
- c. BreakitRefreshTableViewHeader is a view that is responsible for animating the UIActivityIndicator. It is invoked when users pulls down the upper section of the table in NewsFeedViewController.
- d. CommentTableViewCell is a subclass of UITableViewCell that is responsible to present all the comments made on the particular photo by the users.
- e. InternetWarningView is responsible for checking the Internet connectivity and arrival of new notification to the users.
- f. PhotoFrame is a subclass of UITableViewCell that is responsible for presenting photo and its associated information such as comments, number of upvotes, downvotes, headline, distance, time and address.
- g. SingleTapToPhoto is responsible for presenting a photo with zoom in and out functionality.
- h. UpdateNotificationCell is also a subclass of UITableViewCell that is responsible for presenting newly arrival notification on LeftViewController.

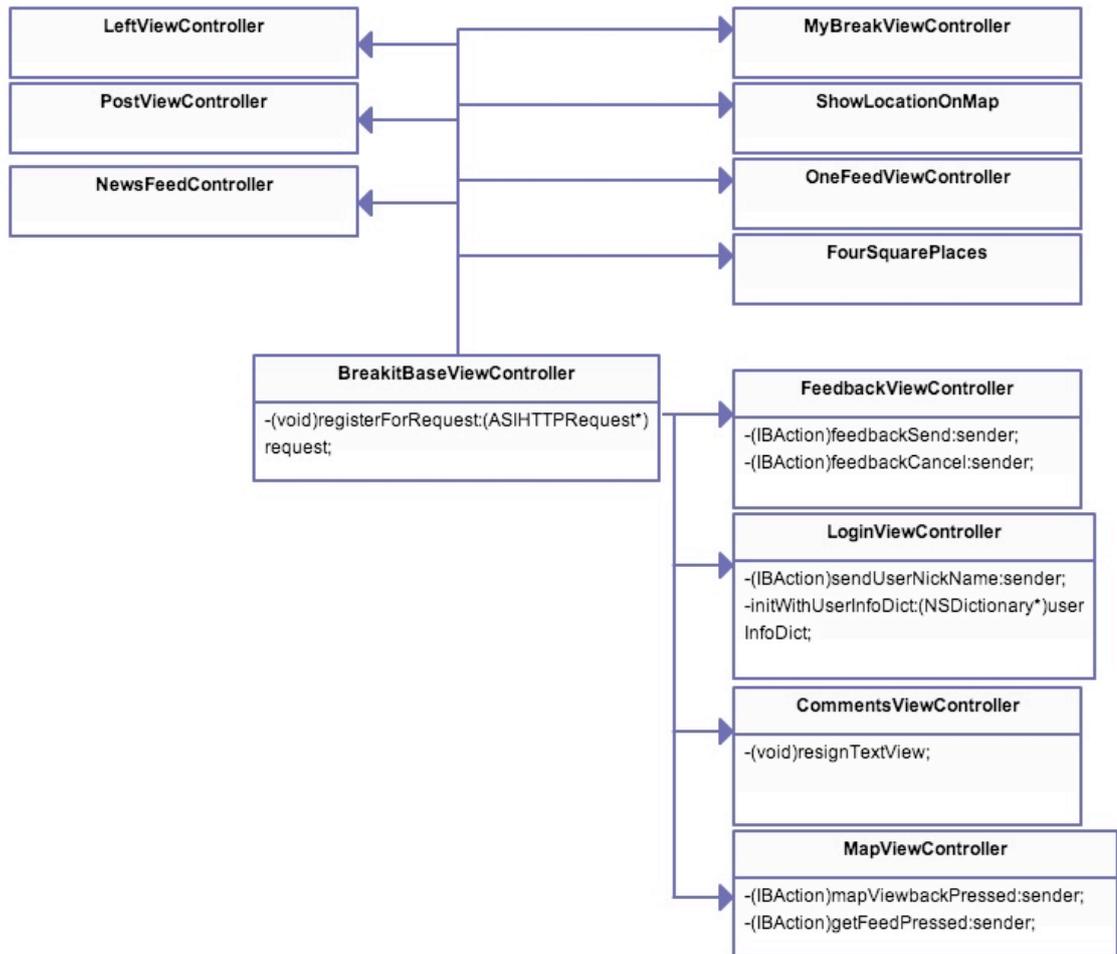


Figure 15. Controller classes of Breakit application

Figure 15 is a collection of the ViewController classes used in the application. All the ViewController classes are sub classed from a BreakitBaseViewController class as all the view controllers have some functionality to communicate with the server. The delegated request of ASIHTTPRequest is handled in this class to avoid crashes that might happen when the delegated request is not in the memory. The implementation of controller classes can be described with the help of use cases and its graphical presentation.

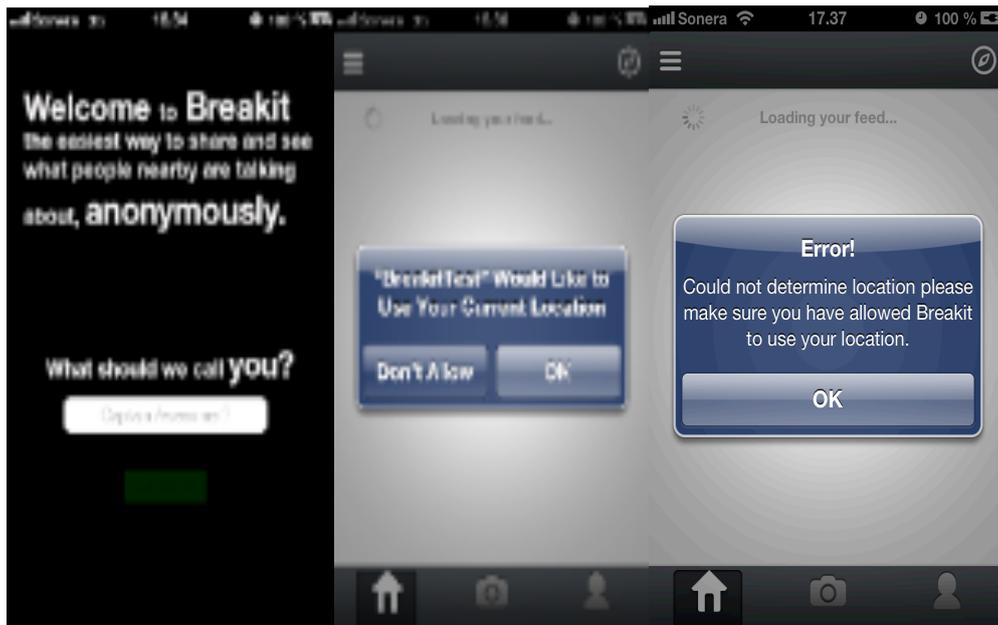


Figure 16. The initial view for a new user creation and the permission for Location Service.

Figure 16 is a collection of views when the application is launched for the first time. During the application launch, the execution flow of the application checks the property list with the help of model class `NSMutableDictionaryPropertyListReader`. If the property list is found empty the `LoginViewController` is presented. The user sends the nickname to the server for uniqueness and if the similar name is not found in database, the server stores the nickname into database and sends a response with `userId` and asks to send the device token if Apple Push Notification is enabled. The client stores the `userId`, `username` and device token into its property list and sends the device token to the server. Upon, the successful response from the server, the application loads the `NewsFeedViewController` to show the photofeeds. The Location Service popup is shown and the user has to allow getting the current location information. In the case of not allowing Location Service another Error popup would be shown and user will have to allow it from the iPhone setting panel. Until, the user enables the Location Service, there will be a blank no photo feeds shown.



Figure 17. PhotoFeed from NewsFeedViewController and LeftViewController with Notifications

As shown in figure 17, the NewsFeedViewController class is responsible for presenting the photo with its associated information. As the LocationManagerController class provides the current latitude and longitude of the user, the JsonParsorController class communicates with the server and gets all the nearby objects. These objects are used in the PhotoFrame class to populate table of photo with its associated data. Category class NSString+NSString_DateNTIMECalculation is handling the time information and the address of the picture and distance is calculated in LocationManagerController class.

The interaction on the left corner button leads to the picture shown at right hand side in figure 17. As soon as the button is pressed, the UpdateNotification class is called to get all the notification that the user has. The `-(void)pullNotification;` method communicates with the server by providing its userId and gets all the notifications replied if any exists. The replied objects are used by UpdateNotificationCell class to populate the notifications.

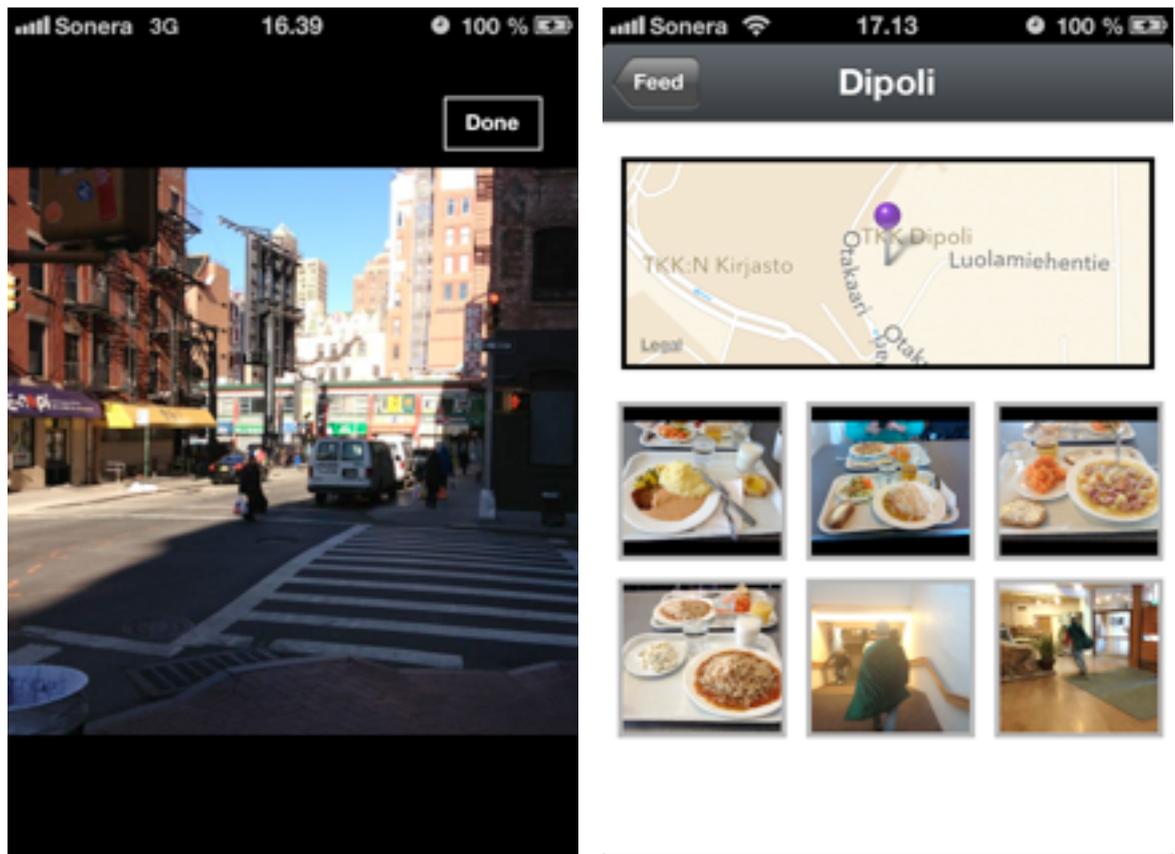


Figure 18. Use of SingleTapToPresentPhoto and ShowLocationOnMap class.

When the user taps on the photo, the view as shown in figure 18 on the left would appear. All the images are presented with the UITapGestureRecognizer. Hence, the single tap on the photo will call the `-(void)showPhoto:(UIImageView*)imageView` methods of SingleTapToPresentPhoto and the zoom in/out enabled view is added on top of the current view. The done button is linked with protocol method `-(void)removePhotoNshowTabbarController;` which removes the added view.

The second picture shown in figure 18 can be achieved by tapping on the place name on the main feed. The Annotation class is used to present the annotated pin on the map and the GridViewCell class is used to populated the grid. The user can view all the photos that are taken at the particular place by any users. The selection of any picture would add the OneFeedViewController view on to the navigation stack.

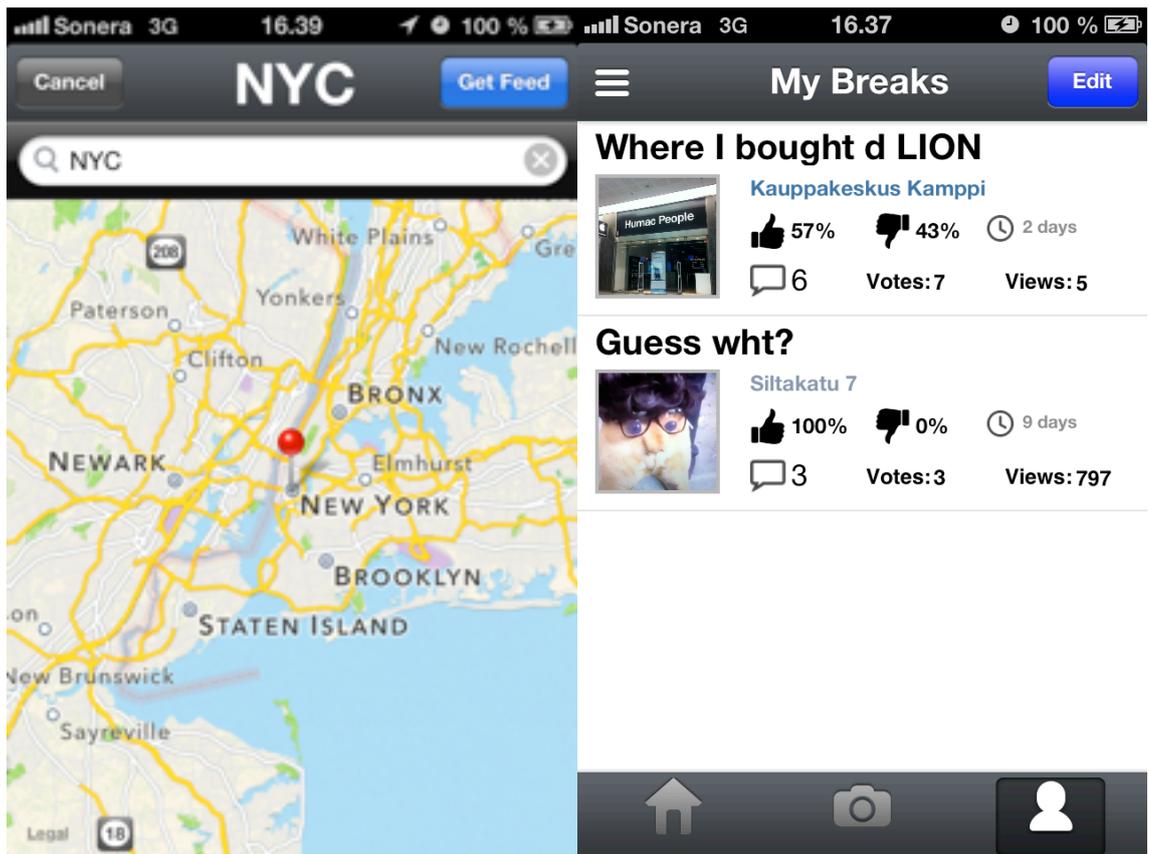


Figure 19. Use of MapViewController and MyBreakViewController class

As shown in figure 19, the MapViewController will be presented if the user taps on the right corner bar in NewsFeedViewController. The user's current location is acquired with LocationManagerController class and Annotation class is used to provide annotated map. The user can search the place by using the search bar or simply tap at any place on the map to move the annotation pin and get the city name of annotated place. The tap on the GetFeed button will take back to the PhotoFeed view with the photos nearby the chosen place in MapViewController.

When the user taps on the dummy profile icon on the tabbarcontroller, it invokes the MyBreakViewController class. During the view load into navigation stack, it sends a request to the server to get the all the information about the photos that users have sent to the server. Upon, the successful response with the objects from the server, it sends the replied objects to MyBreakTableViewCell class to populate the picture and its associated data. The user can delete the photo by tapping on the edit button.

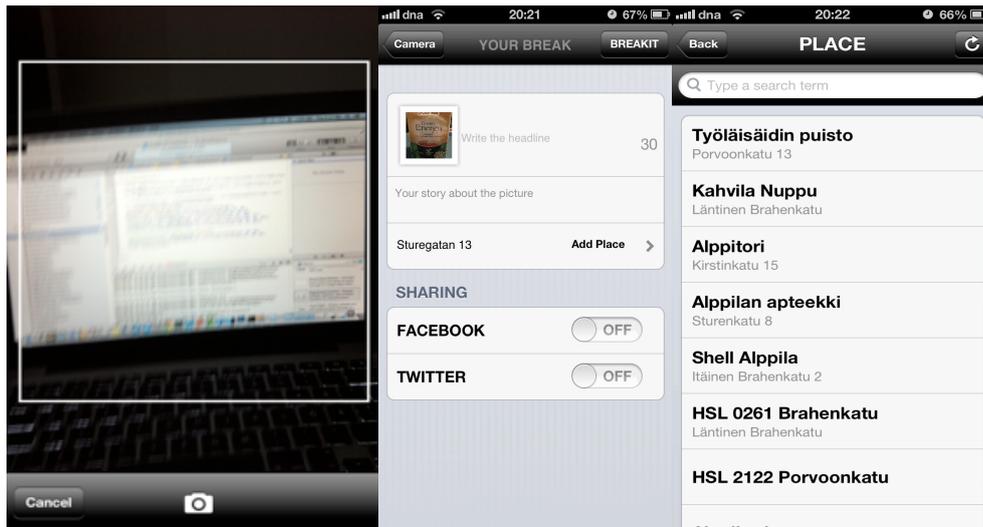


Figure 20. Sending Photo to Breakit Server

As shown in figure 20, the UIImagePickerController controller's capture image view is added into the navigation view stack. The white rectangle box is the indication of the higher and lower limit of the picture that user can take. In the UIImagePickerController object, the temporary view is added as an overView to provide toolbar, consisting camera and cancel button and the white rectangle frame to let user know the pictures above and below the frame will be cropped out. The user can take a picture by tapping on the camera icon. As soon as the image is taken, the pickerView is dismissed and Post-ViewControlller's view is added to the view stack.

In the PostViewControlller, a user has to write something as the headline to post the picture to server. The empty headline will show an Error popup. During the process of the view load into navigation stack, the LocationManagerControlller class's object is called to get the latitude and longitude with the flag 2, which forces the location manger to provide location information with respect to its accuracy limit set in the corelocation delegated method. If the locationManger object couldn't get the desired accuracy, the counter is increased and posts the notification to the PostViewControlller class when the counter value is 5. In this way, a backup method is applied to ensure the more accurate location information extraction from the LocationMangerControlller class. The latitude and longitude is then used in Google map API to get the address name. The user can choose the Foursquare place name by tapping on the Add Place button. The tap on the Add place button invokes the FoursquarePlaces class which provides the nearby Foursquare places by using its API. The selection of the foursquare places leads back to the PostViewControlller's view. After having the address or place name

selected and investigation is carried out. The blue selection shows that the application requires much more resources and time to open the camera. Similarly, the real memory use of the application can be tested as shown in figure 22.



Figure 22. Activity Monitor Test of the application

Figure 22 is a graphical presentation of the Activity Monitor Test of the application. The Activity Monitor test helps testers to see the exact memory amount the application uses along with four top memory consumer applications. Similarly, the memory allocation and leaks can be detected as shown below.

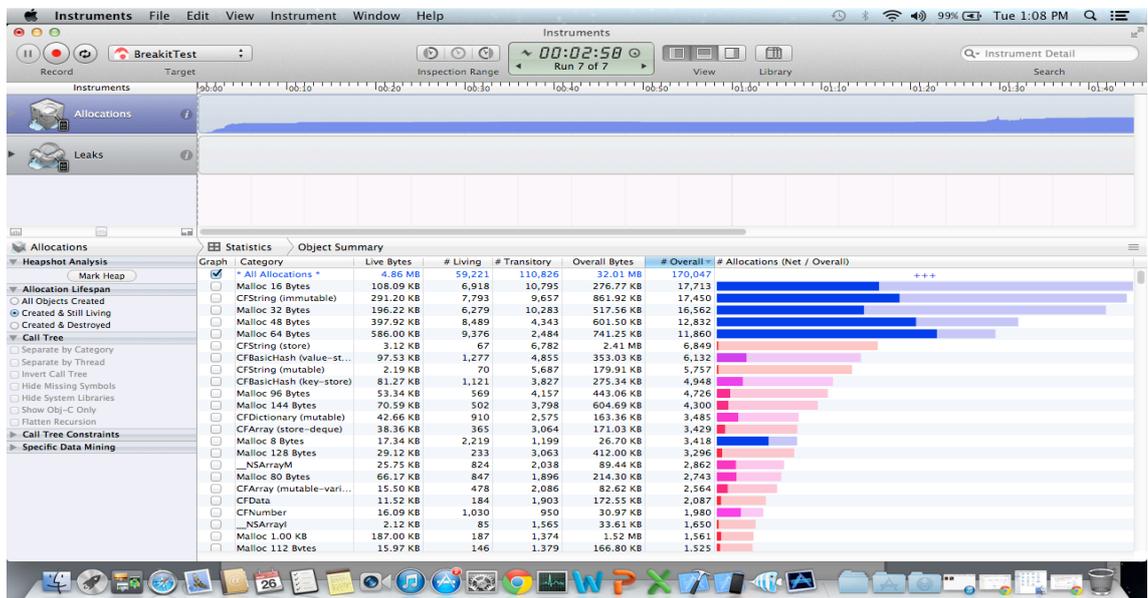


Figure 23. Memory Allocation and Leaks test of the application

Figure 23 is a graphical presentation of memory allocation and leaks test of the application. This particular test helps developers to measure heap memory and all the

memory allocation especially, the Live Byte of the application. All the memory related issues such as unexpected growth of the heap-size, memory leaks can be tracked in the test.

6 Result and Discussion

The aim of the project was to create a location based photo-sharing application for iPhone and submit it to Appstore, which was achieved by releasing the application on 26th February 2013. Besides that, working in a production environment helped to understand and implement Agile Software development methodologies. The project was an opportunity to create and experience a learning journey from a raw concept to produce an end product for iPhone users.

During the design implementation phase, the project went through various stages. Initially, the project was set up as a hybrid application. Hence, the Cordova project was integrated and basic photo feed view was created. However it was soon noticed that the performance of the application was very slow and there was a problem with the location service popup being displayed twice. Besides that, the computation time for presenting photo feed on a WebView was slower compared to presenting it in a UITableView. Also links separation in WebView was difficult, which led to changing the architecture of the project into the iOS native architecture.

As the project moved on with native architecture, some technical problems soon appeared, such as mismatch of the data format stored in MongoDB with NSDate. The server-generated data strings had to be converted into NSDate format for date and time calculation. Another key point to be noted was the time zone difference between the server and the user. For example, a user in Finland and a user in Nepal would have different time, as the server was located in Ireland. The category class of NSString was created and the date from the server was converted into the system date by comparing with the Greenwich Mean Time for time calculation.

The current version of application serves a few sets of services and lacks many features and services such as an appealing view design, services such as follow places, user account setting and profile view, deleting user's own comments, revoking the like/dislike and sharing with other social media such as Instagram, Foursquare and

Tumblr and Google+. The application runs only on iOS 5.1 and later versions of iOS, thus the compatibility with the older version of iOS such as iOS 4.2.3 needs to be fixed.

The application provides users with a better way to know what is happening around as it provides the feeds based on their current location rather than the feeds from closed circle of friends and subscribed channels. The presentation of the photo is sorted with the algorithm that accounts for the distance, time and the user's voting. The recent and nearest pictures are ranked with the user's voting that enables users to see the most rated happenings at recent time nearby. Thus, the application can be used as a location based marketing tool. For example, a restaurant uploads the dish of the day to the Breakit server. The nearby Breakit users would get the photo feed immediately.

7 Conclusion

The innovation of recent wireless technologies and location-based services has made our lives unbelievably sophisticated. Due to the availability of the development platform, creating a mobile application has been possible. The increasing rage of mobile phones and attraction towards iPhone has encouraged application developers to come up with new applications.

The goal of the project was to create a location-based application for iPhone that would basically be used for photo sharing among the users. The application has several functionalities such as taking a picture, posting picture feeds, commenting on a feed, up/down voting a feed, sharing a feed via social networking sites, viewing feeds on the basis of users current location, viewing the notification, inviting Facebook friends and view statistics of user's own feeds.

Since the launch of the application, the number of downloads of the application is increasing every day with more daily users, which reflects the achievement of the goal set for the project. However, to make the application even more appealing and useful, the functionalities have to be extended according to users' feedback and need. Since the application is launched for the global audience, localization has to be done to make a user-friendly application. Similarly, supporting the iPad version is another optimization work to be carried out in the future. Last but not the least, Android, Windows, Research In Motion and Jolla version of the application can be developed to cover all the whole mobile users in the market.

References

- 1 Webopedia. Social-Media. [online]; 2012
URL: http://www.webopedia.com/TERM/S/social_media.html. Accessed 13 December 2012
- 2 Microsoft. Glossary.[online];2012
URL: <http://msdn.microsoft.com/en-us/library/ff359117.aspx> Accessed 13 December 2012
- 3 Nodejs.Node.[online];2012
URL: <http://Nodejs.org>. Accessed 14 December 2012
- 4 Tom Hughes-Croucher& Mike Wilson. Node Up and Running. O'Reilly Media, Inc. 1005 Gravenstein Highway North, Sebastopol. 2012
- 5 MongoDB.Documentation.[online];2012
URL:<http://www.mongodb.org/>. Accessed 18 December 2012
- 6 Wikipedia. HTML5.[online];2012
URL:<http://en.wikipedia.org/wiki/HTML5>. Accessed 20 December 2012
- 7 Wikipedia. Cascading Style Sheet.[online]; 2012
URL: http://en.wikipedia.org/wiki/Cascading_Style_Sheets. Accessed on 26 December 2012
- 8 Wikipedia. CoffeeScript.[online];2012
URL: <http://en.wikipedia.org/wiki/CoffeeScript>. Accessed on 26 December
- 9 Wikipedia. IOS SDK. [online]; 2013
URL: http://en.wikipedia.org/wiki/IOS_SDK. Accessed on 5 January 2013
- 10 Apple Inc. Xcode.[online];2013
URL: <https://developer.apple.com/xcode/>. Accessed on 2 February 2013
- 11 Lee W. Beginning iOS 5 Application Development. Indianapolis, Indiana: John Wiley & Sons; 2012.
- 12 Apple Inc. iOS Developer Library. [online]; 2013
URL:https://developer.apple.com/library/ios/#documentation/ToolsLanguages/Conceptual/Xcode_User_Guide/030-Edit_User_Interfaces/edit_user_interface.html#//apple_ref/doc/uid/TP40010215-CH6-SW41. Accessed on 4 February 2013
- 13 Apple Inc. Tools Workflow Guide for iOS. [online]; 2013
URL:http://developer.apple.com/library/ios/#documentation/Xcode/Conceptual/ios_development_workflow/25-Using_iOS_Simulator/ios_simulator_application.html. Accessed on 5 February 2013

- 14 Apple Inc. Instrument User Guide. [online]; 2013
URL:<http://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html>. Accessed on 20 February 2013
- 15 Apple Inc. iOS Technology Overview. [online]; 2013
URL:<http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf>. Accessed on 25 February 2013
- 16 Wikipedia. Open Source.[online]; 2013
URL: http://en.wikipedia.org/wiki/Open_source. Accessed on 26 February 2013
- 17 Mashable. Facebook-infographic. [online]; 2011
URL: <http://mashable.com/2011/10/21/facebook-infographic/>. Accessed on 27 February 2013
- 18 Twitter Inc. Numbers. [online]; 2011
URL: <http://blog.twitter.com/2011/03/numbers.html>. Accessed on 27 February 2013
- 19 Apple Inc. Streamline Your App with Design Patterns. [online]; 2013
URL:<http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMap-iOS/chapters/StreamlineYourAppswithDesignPatterns/StreamlineYourApps/StreamlineYourApps.html>. Accessed on 12 March 2013
- 20 Apple Inc. Cocoa Fundamentals Guide.[online]; 2013
URL:<http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/CocoaFundamentals/Introduction/Introduction.html>. Accessed on 13 March 2013
- 21 Mac OSX Developer Library. Protocols. [online]; 2013
URL:http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/ObjectiveC/Chapters/ocProtocols.html#//apple_ref/doc/uid/TP30001163-CH15-TPXREF144. Accessed on 13 March 2013
- 22 Wikipedia. History of Facebook.[online] 2013
URL: http://en.wikipedia.org/wiki/History_of_Facebook. Accessed on 13 March 2013
- 23 Wikipedia. Twitter.[online]2013
URL: <http://en.wikipedia.org/wiki/Twitter>. Accessed on 13 March 2013
- 24 Wikipedia. Software Testing. [online]2013
URL: http://en.wikipedia.org/wiki/Software_testing. Accessed on 28 March 2013
- 25 Haselhurst Michael. 10 Interesting Mobile Phone Facts. [online] 2012
URL: <http://geeky.info/10-interesting-mobile-phone-facts>. Accessed on 28 March 2013
- 26 A.Kushki, K.N.Plataniotis, A.N.Venetsanopoulos, WLAN Positioning System. Cambridge University Press. TheEdinburghBuilding,CambridgeCB28RU,UK 2012

Appendix 1: Facebook User Authentication Process

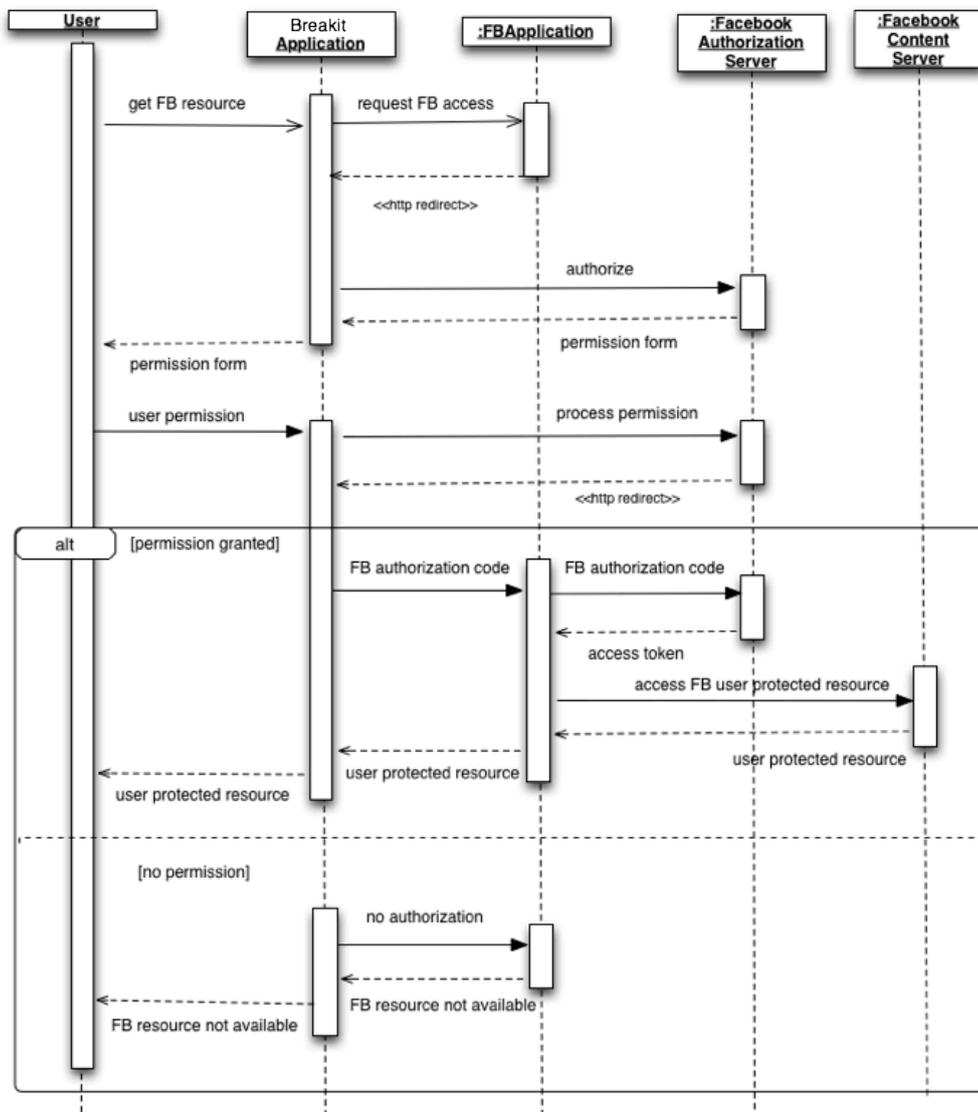


Figure 24 Facebook user authentication process.

Figure 24 shows the whole process of user authentication of a Breakit user to their Facebook account via Breakit application created under Facebook Dev Center.

Appendix 2: Twitter authentication Overflow

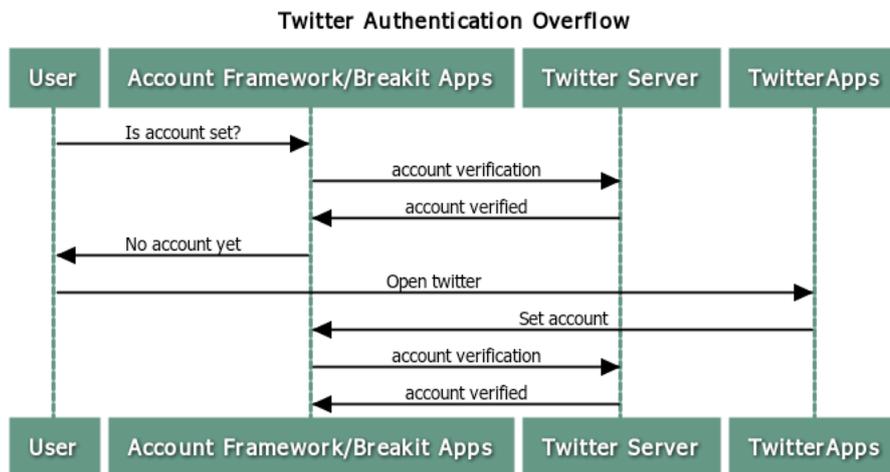


Figure 25. Twitter authentication overflow.

Figure 25 shows the authentication flow of Breakit user to tweet to their respective Twitter timeline.

Test Case Name:**Breakit****Description:**Functionalities and behaviour
of the Breakit application**Overall Pass Criteria:**

no errors, expected behaviour

Step	Action/Event	Expected Result	Pass	Fail	Comments
1	tap the program icon on the menu	Program should check network connection	✓		application needs to have WiFi or 3G connection
2		user registration	✓		asks user for the registration at the very first time
3		check uniqueness of nickname	✓		user can have unique nicknames only
4		location service enable	✓		provides current latitude and longitude
5		location service disable for Breakit		✓	shows error message and provides blank view
6	no login view after registration	application should show Newsfeedview	✓		property list is not empty
7	display of city name in Newsfeed	should show city name of current user	✓		shows city name of the user where s/he opens the application
8	pressed on Explore	should open map with annotated pin	✓		shows the map with users current location via annotation
9	search places on map	should let users to search and place the annotation on	✓		annotated pin changes according to search places

		searched place			
10	cancel on Map	should return to NewsFeed View	✓		shows newsfeed without reloading the data
11	tap on GetFeed on Map	should return to News-feed View	✓		reloads the feed according to searched places
12	tap on Menu icon from NewsFeed	should open the Menu	✓		opens the Leftview
13	tap on Feed	return to NewsFeed	✓		checks for the new item and reloads the tableview of newsfeed
14	tap on Feedback button	opens feedback view	✓		opens the feedback view to write down and send the feedback to breakit server
15	tap on Cancel button on Feedback View	return to Menu Viewq	✓		returns to Leftview
16	tap on send button on Feedback View	if not empty the uitextview, send the content to server	✓		sends feedback to breakit mail address
17	shows notification in Menu	shows the notification ie the comments done by other users to the feed	✓		shows comments, user nickname and time
18	tap on the notification in Menu	shows the respective photo feed in another view	✓		it shows the notification in another view
19	tap on the profile icon	shows the statistics of own feed	✓		shows statistics of own feed ie all the pictuers and its views, like/dislikes received
20	tap on the picture in MyBreak view	open the respective picture in another view	✓		opens the feed as onefeedview
21	tap on MyBreak button	leads to My break View	✓		returns to My Break View
22	tap on the share icon in oneFeedView	opens the uialertview	✓		shows the option to share or report
23	tap on the facebook in uialertview	sends the respective picture to users facebook timeline	✓		sends headline and picture link to users facebook time

					line
24	tap on twitter in uial- ertview	sends the respective picture to user twitter	✓		sends headline and picture link to us- ers twitter as tweet
25	tap on report inappropri- ate	send the inappropriate information of the pic- ture to server	✓		sends userId, pic- tureid to the server as inappropriate picture report
26	tap on like/dislike	increment/decrement like/dislike	✓		sends the voteup/votedown value to server and reloads the data
27	tap on the picture	opens the picture in another view	✓		opens the picture in another view where user can zoom in/out the picture
28	tap on Edit button on My Break view	presents the delete button	✓		presents the delete button on every row of the table
29	tap on the delete button on the row	deletes the picture	✓		sends requests to delete picture and upon sucessful reply reloads the table
30					in the case of serv- er error show the error description as a pop up
31	tap on the comment but- ton on MyBreak view	opens the comment view	✓		opens the com- ment view showing all the comments that the photo has
32	scrolling to the end of the MyBreak view	shows loading more text	✓		connects to server for more pictures
33	scrolling to the end of the MyBreak view	shows no more to load text	✓		connects to server and gets reply as no pictures
34	tap on the camera icon	opens camera view	✓		opens camera
35	cancel on camera view	shows newsfeed view	✓		returns to news- feed view
36	tap on camera icon on camera view	takes the photo	✓		it takes the photo and presents post-

					view
37	tap on camera icon on postview	takes back to camera view	✓		returns to camera view where user have to take a picture
38	tap on the write headline view	presents the keyboard to type	✓		presents the keyboard
39	tap on story view	presents the keyboard to type	✓		presents the keyboard to type
40	tap on addplaces button	shows the foursquare places	✓		shows the four-square places with places provided from foursquare API
41	tap on search bar in four-square	hides the current suggested place and let users to write in searchbar	✓		let users to write down the place name
42	tap on the search in keyboard	searches the placename in foursquare	✓		searches the placename in four-square if found presents else show error
43	tap on the shown placename	the name is selected and returns to post-view	✓		returns to post-view by selecting placename and foursquare id given from API
44	tap on breakit button	change the color, lock the button and send the data to server	✓		it locks down the button, changes color red to gray and send the data to server
45					upon the successful reply, it dismisses the current view and return to Newsfeed view
46	double tap on Newsfeed view icon	scrolls to top	✓		it leads the scrolling to the top of the view
47	left/right swipe on the picture	shows the next picture of an album if it has	✓		shows the picture of an ablum in a carousel fashion

48	left/right swipe on the picture	shows all the respective data of the photo	✓		changes the data not only the picture
49	pull down to refresh	connects to the server for recent updates	✓		reloads the data with new items if it gets from the server
50	No internet availability	show the warning text about the internet under the navigation bar	✓		whenever the internet seems to have down, the warning view is shown under the bar
51	scrolling up/down	hides and shows navigation and tabbar	✓		the movement of the scroll determines the appearance or disappearance of bars
52	tap on the album name	shows all the pictures along with map in another view	✓		shows pictures in another view in a grid layout(3x3)
53	tap on the picture in a grid view	opens the picture as one feed view	✓		shows pictures with its all associated data, similar to main feed
54	tap on the annotation on map	display the name of the place	✓		show the name of the place and tap on the discloser button opens google map
55	tap on the address name in the newsfeed view	opens the map	✓		show the address with pin in the map