

Nidhi Lakoul

Calendar Integration with Social Media: Facebook

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Degree Programme in Information Technology

Thesis

28 September 2012

Author(s) Title	Nidhi Lakoul Calendar Integration with Social Media: Facebook
Number of Pages Date	33 pages + 12 appendices 28 September 2012
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Computer Internetworks
Instructor(s)	Peter Hjort, Principal Lecturer
<p>The goal of this project was to develop an application that would integrate a calendar application with the social media, that is, Facebook. The main purpose of this application was to retrieve events from Facebook into the local calendar application and vice versa and also to fetch friends' birthday events into the calendar application. The functional part mainly concerned the core study of the Facebook platform and developers' tools for both Facebook and iPhone.</p> <p>The Facebook connect SDK (Software Development Kit) was identified as the key player in the integration business which provided readymade code to embed into the Facebook application that was created during the project in order to exchange information with iPhone application, that is, the Student Calendar. The Graph API was found to be the core of the Facebook Platform to enable developers to read from and write data into Facebook. All the targets of the project were achieved although there were some complexities within the platform components.</p> <p>Nowadays, integrating applications with social networking sites are becoming popular which is attracting more developers. In such a context, the application developed has a great demand. This application can be useful for students and those who are fond of using Facebook. The application has some important features of Facebook, such as a post-on-wall, view news feed or checking a friend's birthday, so the user does not have to go through the Facebook page for these functions. The procedures that were taken during the application have been documented in detail, so the thesis can be used as a guide for developers, and those who have interest in how integration is done can have a glance at it.</p>	
Keywords	Facebook , integration, graph API, SSO, events

Contents

1	Introduction	1
2	Background	2
2.1	History of Facebook	2
2.1.1	Facebook and its LAMP Server	2
2.1.2	Architecture Overview	4
2.2	Student Calendar	5
2.3	Single Sign-On (SSO)	6
3	Facebook Developer's Platform	7
3.1	Facebook Application Architecture	7
3.2	Facebook Apparatuses	8
3.2.1	Facebook API	8
3.2.2	Facebook Markup Language	9
3.2.3	Facebook Javascript	9
3.2.4	Facebook Query Language	10
3.3	iOS Software Development Kit	10
3.3.1	Xcode	10
3.3.2	Interface Builder	11
3.3.3	iOS Simulator	11
3.3.4	Instruments	11
3.4	iOS Framework	12
4	Development	13
4.1	Facebook iOS SDK	13
4.1.1	Methods	14
4.1.2	Delegation and Protocols	16
4.2	Analysis and Design Details	20
4.2.1	User's Perception	21
4.2.2	Authentication Workflow	22
4.2.3	Application Implementation	24
5	Result and Discussion	31
6	Conclusion	33

Appendices

Appendix 1. Calendar and Facebook Classes

Appendix 2. Event View and Facebook Functionality

Appendix 3. Facebook Functionality

Appendix 4. The iOS Frameworks

1 Introduction

In the present context, the online community has been the most significant part of the Internet world and so are the social networking sites. Merging of social activities with computing enabled the changeover of everyday objects into information appliances. Social networking sites do not just enable users to communicate but also provide virtual community for the people interested in a peculiar subject or in “hanging out” together. It allows a user to connect with family and friends by creating personal information profiles, sending messages, having access to each other's profile, commenting on the status or letting a user to vote on an article. In addition, user can join common-interest user groups, categorize his/her friends into lists such as “relatives”, “close friends” or “people from work”.

Such multi-way activities have been possible through Web 2.0 technologies, a collection of scripting languages and applications. Networking has taken a prominent leap within the last decade, and as an outcome, we are able to integrate our applications with the social networking sites. Badoo, Facebook, Friendster, hi5, and Twitter are some of the popular sites at present. These applications provide their own platforms, which enable us to integrate, for example, a website, a desktop application, or a cell phone application with it.

This final year project mainly focuses on the study of a social media platform, iPhone Operating System, and its developing tools. The integration requires the usage of Application Programming Interfaces (APIs) provided by social networking sites combined with Software Development Kits (SDKs), and they are discussed briefly in this thesis. Such integration improves assessment of opportunities, promotion of products and services indirectly, and easy access of useful events. On the other hand, it reduces the multiple login process and enhances user experience.

The thesis deals with the Facebook application platform, APIs and two-way integration of events from/to a Facebook application to/from a calendar application. Despite the different forms of integration (such as website integration, integration with Android application), the project integrates iOS application with the Facebook application. The goal of the project is to develop an application that would integrate an iOS application

(student calendar) with Facebook application. The produced application would retrieve events, get notifications from the Facebook and also post on Facebook application straight from the synced calendar application.

2 Background

2.1 History of Facebook

As of the recent updates, Facebook has been announced as the world's largest social network having more than 955 million active users. In February 2004, a Harvard student Mark Zuckerberg, who also created Facemash, HOTorNOT.com for Harvard students, founded Facebook. Facebook is a social networking site that helps to get connected with friends, family, and business associates. The social networking giant facilitates the sharing of information through social graph, the digital mapping of people's real-world social connections [1].

Facebook was originally called thefacebook and was renamed Facebook in August of 2005. Initially, Facebook membership was restricted to school, universities and some English- speaking companies in Canada and the United States. However, within some months' time it expanded to include anyone over the age of 13. Later in the year 2007, Facebook launched the platform API (REST) publicly through which third-party developers and websites could control Facebook's user basics.[1]

2.1.1 Facebook and its LAMP Server

It has been a challenging task for Facebook to deal with more than 500 billion page views per month; more than 3 billion photos uploaded every month, and more than 30,000 servers. Despite these, Facebook's site is up and running efficiently, which has been possible due to scalability, simplicity, openness and the technical architecture. Facebook uses a variety of services, tools and programming languages to build up its core infrastructure. Their servers run on LAMP (Linux/Apache/MySQL/PHP) stack with Memcache. [2]

Linux is an open source Unix-like operating system kernel developed by Linus Torvalds in 1991. As the Linux system is an advanced form of Unix OS, almost all Unix programs can be compiled and run in Linux environment. Even though it is open source software, it is very secure due to several reasons such as privileges, social engineering, monoculture effect and large number of developers and testers.[3] Apache is the most popular open source web server in use. It runs about 60 per cent of all websites on the Internet. [4]

In Facebook, there are thousands of millions of data in the server, and without the database it is almost impossible to keep track of them. Facebook uses MySQL to store user data, as it is fast and reliable. *MySQL* is a well-renown open source database that competes well with highly priced servers. The Facebook data is stored in randomly distributed logical nodes, and MySQL uses data storage as in dictionary. For example, search engines or online payment using the Facebook credit are the results of using a database.[5, xiv]

PHP stands for “PHP: Hypertext Preprocessor”. As a programming language, *PHP* is simple to learn, read and write, and debug. PHP scripts are used for different purposes as server-side scripting, command line scripting and writing desktop applications. Also with the help of PHP, a standard HTML page can end up resulting into a dynamic web-page. It has greater advantage compared to Java Server Pages (JSP), Ruby on Rails (RoR) and ASP.NET due to its excellent performance, stability, availability, portability, extendibility and also being open source. [5, xii] Facebook uses a source code transformer named HipHop for PHP, which speeds up the PHP operations, improves efficiency and reduces the CPU usage on the Web servers almost by half, depending on the size of the page, and results in improved performance [6].

Memcache is a memory caching system that is used to speed up dynamic database-driven websites (such as Facebook) by caching data and objects in RAM, reducing the need to search a database for information. This decreases the amount of time it takes between a request for information and the delivery of that data. Facebook had a really slow access to the database, and to enhance the speed of the database, it now uses memcache as a caching layer between the web servers and MySQL servers. [2]

2.1.2 Architecture Overview

According to a presentation by Aditya Agarwal, Director of Engineering, LAMP resolves huge number of problems yet is not perfect. Therefore, the services were written to store the code closer to the data, to make a compiled environment more efficient and to use functionality present only in a certain language. Facebook uses several different languages for its different services. For example, PHP is used for the front-end, Erlang is used for Chat, Java and C++ is also used in several places.[7] Figure 1 illustrates the architecture overview of Facebook, along with its services.

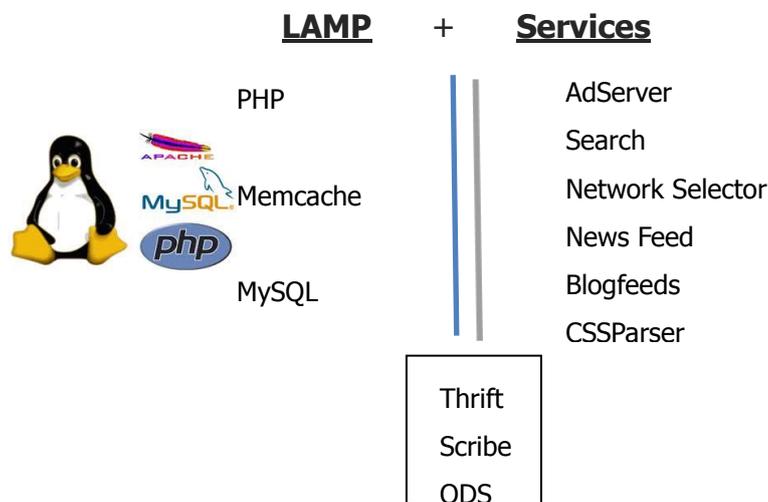


Figure 1. LAMP architecture overview of Facebook. Reprinted from Agarwal (2009) [7]

Thrift is an internally developed framework from Facebook that is used to seamlessly and effortlessly communicate between different programming languages [2]. Similarly, *Scribe* also helps in the performance of Facebook by accumulating streaming log data and scaling them to a huge number of nodes avoiding any network failure. Scribe is available running on each node and accumulates data and sends them to the central server. Scribe also works as a back-up server when the central server is down, and when the server is up, it sends the data back to the server. [8]

Several nodes have multiple data gathered which then needs to be integrated for the surplus operation using data virtualization, data federation, or extract, transfer and load. This will allow operational access to the data for operational reporting, master

data or reference data management. It stores constantly updated data used in the current operation before it gets transmitted for archiving.

Operational data store (ODS) is a database that is proposed to handle this data from multiple sources in Facebook, converting it into an ODS system, and reorganizing and structuring the data for analysis purposes. [9]

Lastly, the architecture comprises services such as AdServer, Search, Network Selector, News Feed, blogfeeds, CSS Parser and mobile, to form a complete Facebook architecture.

2.2 Student Calendar

A calendar is a system of keeping our days organized into a certain period of time, that is, days, weeks, months and years for various purposes such as social, religious or commercial purposes. The calendar has different display formats, such as daily, weekly and monthly view. Upon the selection of a proper view, the events are shown accordingly. The minimal function that a calendar has is to create an event, edit/delete the event, repetition of the event, and adding reminders to it. The reminder can be set just before the event or it can be a few minutes or hours before the event as the user desires. The events are categorized, for example, under meetings, to-do lists, school events. The calendar also provides the option for recurring events to prevent the user from creating the same event several times.

The student calendar was developed during autumn 2011 for iPhone as a work placement project. The student calendar has the same functionalities as other calendars as it is based on the Gregorian calendar format. The student calendar can create an event, add a description to the event, edit and delete the event, repeat an event, and give a notification on the event. Yet, it is primarily targeted to a student so the extended function to a normal calendar is to synchronize with the exchange server of the school so that he/she gets an updated school schedule and also the journey planner to get from user's current location to the campus. In the right column, it also has the week number that can be very important from the student's point of view. This final year project is being done as an extension of that project.

2.3 Single Sign-On (SSO)

Single sign-on is a process that allows a user to access multiple applications with a single action of user authentication and authorization. It reduces human error since the user does not need to memorize multiple passwords.[10] According to Jan de Clercq[11], SSO is defined as follows:

Single sign-on is the ability for a user to authenticate once to a single authentication authority and then access other protected resources without re-authentication. [11]

Single sign-on has several benefits such as reduced administrative costs, enhanced user experience, and increased security [12]. Nowadays, SSO is also available on iOS and Android platform with the help of which one can log into a mobile application using his/her Facebook identity. Figure 2 clarifies the process involved in single sign-on.



Figure 2. Example of Single Sign-On Using Facebook.

Users may have different identities and password to access the systems and applications they need within a network and they have to keep track of each identity. Due to such a password chaos, people are most likely to forget password and Single Sign-On has made their lives easier by providing a single login process to an application that in turn gives access to multiple applications, also known to be Reduced Sign On (RSO) [13]. Similarly, Helsinki Metropolia University of Applied Sciences can

be taken as an example of SSO. Once a user logs into the tuubi portal, the user gains access to for example winhawille and email automatically.

3 Facebook Developer's Platform

The Facebook platform plays a vital role in its application development. The platform can be used to create an application using Facebook features and also to integrate external application features into Facebook. Facebook launched its platform in May 2007 with few applications, and now it has more than 7 million applications and websites integrated, with more than one million developers and entrepreneurs around the world. It is a collection of application programming interfaces (APIs) and services, which let external brains to interpose new features and contents into the Facebook environment.[14]

3.1 Facebook Application Architecture

There are millions of applications found in Facebook that are not installed straight away onto the Facebook server. Rather, they are located on the developer's server and are called whenever the application URL is requested. This process saves frequent memory consumption especially for sites as Facebook and hi5. Figure 3 below portrays the application architecture of Facebook:

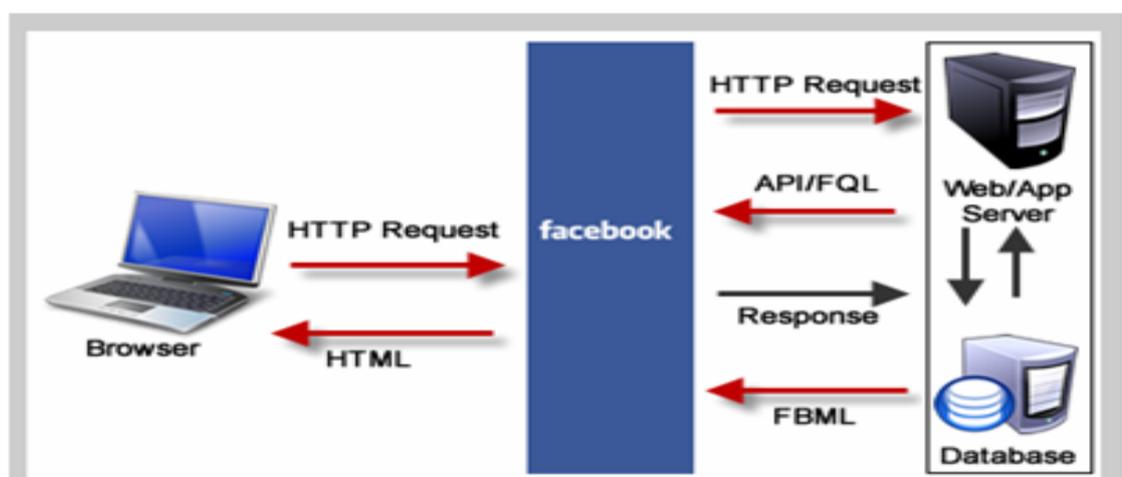


Figure 3. Facebook Application Architecture. Reprinted from Facebook Programming: Facebook Platform (2010) [15]

Literally, Canvas Page is the context or blank space within Facebook that is required to run an application. All the application parts are connected to the canvas and when the action is triggered in the main page (canvas page), the contents on the other application parts are altered accordingly. As stated in figure 3, when the Facebook application URL is requested or HTTP request is done from the canvas, Facebook redirects the request to the developer's own server. The developer's server then processes the request via API or Facebook Query Language (FQL) to get the response back from the server and returns Facebook Markup Language (FBML) for display to the user. Facebook takes the FBML response, presents it within the Facebook canvas, and returns the HTML to the requesting browser.[15]

3.2 Facebook Apparatuses

The Facebook Platform is composed of several core components such as Facebook API, Facebook Markup Language (FBML), Facebook JavaScript (FBJS) and Facebook Query Language (FQL).[15]

3.2.1 Facebook API

The Facebook API is a Web services programming interface through which the members of the social network can build applications by using the social connections and profile information. The Facebook API uses Old REST (Representational State Transfer) API now, but it is clearly stated in the documentation to deploy the Graph API, as Old REST API will be deprecated soon. Hence, the developers are using the Graph API further in the development. Furthermore, REST API will not be discussed in this thesis, as it will not be implemented in future development. The Graph API is more object-oriented (uses friends, user profile, posts) than method-oriented (reading and writing data to and from Facebook).[16]

The Graph API is the fundamental part of the Facebook platform API presenting a consistent view of the social graph along with objects and connections between them. It provides access to objects such as people, events, news feed, likes, tag and friends of a friend, and retrieves/deletes/publishes objects within the Facebook graph.

Developers can access all the public information of any object in the graph simply by using <https://graph.facebook.com/ID> where ID is unique for every object. The Graph API uses OAuth 2.0 for authorization.[17]

The Graph API uses authorization, page login, app login, reading, selection, pictures, paging, dates, introspection, real-time updates, searching, publishing, deleting, analytics, and batch requests to provide information to the developers. Depending upon the application's requirements, the above-mentioned APIs can be used. Likewise, the application of this project deals with multiple users and events; the "batch requests" is used. The application requires permissions, real-time updates along with the batch requests. It uses Event, Group, Message, Photo, Post, and User from the Objects API. [17.]

3.2.2 Facebook Markup Language

FBML stands for 'Facebook Markup Language', and it is a tag-based language used in Facebook application development. It is also known as Facebook's own HTML version with its additional elements.[15] The FBML parser on Facebook server gets the data in FBML form and converts it automatically into HTML, which users' browsers render and display. As HTML, FBML also controls over the displayed output since the Facebook server automatically parses and translates it into HTML, CSS (Cascading Style Sheets) and JavaScript code. [18, 18]

FBML behaves comparatively similar to HTML that it embeds regular HTML in a mixed form. FBML does not have great importance in writing a successful application, however, it can make the application development more efficient.[19,12] Since, FBML does not support JavaScript; FBJS (Facebook JavaScript) can be used instead. The official page states that, FBML is no longer in use after June 2012 but iFrames.[20]

3.2.3 Facebook JavaScript

JavaScript is a widely used scripting language in the web development to add different functionality and communicate with the server. As Facebook does not support

JavaScript elements, there is a limited number of JavaScript codes that can be run on Facebook using Facebook JavaScript (FBJS). Though FBJS is known to be Facebook's JavaScript version, it differs from JavaScript in many ways. The syntax, Document Object Model (DOM) properties and some event-handlers are different in FBJS [6, 18]. FBJS provides AJAX and Facebook dialog objects for developers to take advantage of.

3.2.4 Facebook Query Language

Facebook Query Language (FQL) is an SQL-style language that allows applications to directly explore the Facebook's internal data tables. With FQL, a developer can access the information such as user, friend, group, group_member, event, event_member, photo, album and phototag. FQL queries are more effective as the user can specify the required fields for data extraction. The response format can be specified as either XML or JSON with the format query parameter. In FQL, join query is not allowed as in standard SQL and the other query made should be indexed properly.[17]

3.3 iOS Software Development Kit

The iOS Software Development Kit (SDK), previously iPhone SDK, was released in February 2008 by Apple Inc. to allow developers to create an application for iPhone, iPod Touch and iPad and it is also made possible to test the application in an "iPhone simulator".[21] There are a few but essential application development tools in SDK.

3.3.1 Xcode

Xcode is a developer tools package that includes all the tools needed from writing source code, to debugging and also to designing appealing user interface.[22,3]

Xcode Integrated Development Environment (IDE) is the main tool for application development in an Apple platform. The structure of writing codes in Xcode IDE is similar to the native C language with header files (.h) and implementation of the header files (.m). The manipulation of the file is easier in the sense that the IDE is rich in

functionalities such as documentation. The IDE is very effective to find out the lexical errors, any mistyping and duplication of codes at an instance. The availability of multiple-choice methods at the time of writing code makes the IDE developer-friendly and less time consuming.

3.3.2 Interface Builder

Interface builder is a visual tool that enables developers to create interfaces for applications using graphical user interface (GUI), and saves the resulting interface as a .nib or .xib file. 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, Interface Builder helps the programmers to create a GUI just by dragging objects and linking them into its file's owner, making it more developer-friendly.[22,9]

3.3.3 iOS Simulator

IOS simulator is a very useful tool that can be used to test the application without using real devices. Although it simulates the real behavior of an iOS device, it is recommended to test the application on a real device as it does not imitate many of the features and functionalities available on real devices such as camera or playing audio in the background and. However, for the developing phase, it is wise to choose a simulator than using the real device since it reduces the chances of misconfiguration of device system, which might cause from multiple threads and memory leaks.[22,4-7]

3.3.4 Instruments

Instruments is the monitoring software that helps to optimize an application for example, by pinpointing performance issues, memory leaks and unnecessary allocations. It is very useful for tracing memory leaks. Prior to Xcode v4.3, the developer has to take care of all the memory allocation and destruction on their own. Therefore, the Instruments is the tool to avoid performance issues of the application that are caused by unmanaged memory allocation.

3.4 iOS Framework

Since the launch of iPhone in the market, iPhone has become popular among smartphone users by introducing new features as touchscreens, decent music players, third-party applications through App Store and recently Siri application on iPhone 4S. The Objective-C programming language is used for the programming in iPhone, which is an extension to the C language with the concept of Object Oriented Language.

Apple describes the technologies implemented within the iPhone Operating System as different layers of software, each of which runs on top of the operating system. The base layer is the Core OS layer. On top of the Core OS layer is the Core Services layer. Above that is the Media layer and Cocoa Touch is the topmost layer.[23] These OS layers are illustrated in figure 4.



Figure 4. Architecture of the iOS. Reprinted from Lee (2012) [22, 11]

Referring to figure 4, the topmost layer is Cocoa for Mac OS X whereas it is called *Cocoa Touch* in iPhone OS, it is because iOS contains touch events. Being located on the highest level, it allows an abstraction layer to implement several technologies such as multitasking, touch-based input, push notifications, and many high-level system services without any complication, which reduces the amount of work for developers. [23]

The *Media layer* is responsible for the multimedia services that can be used in the iOS application. It contains high-quality graphics, rich-audio experience (play and record

high-quality audio) and video technologies to give a user a good impression about the application.[23]

The *Core Services* layer consists of the kernel environment and drivers including basic interfaces of the operating system. It includes all the fundamental system services such as memory management, file system handling and threads. It provides an abstraction over the services provided in the Core OS layer.[22,12]

The bottom layer of the architecture is the *Core OS*, which is the foundation of the operating system. It deals with the memory management, networking and other operating system tasks.[22,11]

A framework is a software library that provides specific functionalities and different frameworks have their own uses. Hence, the iOS SDK consists of several frameworks and are grouped according to their functionalities in appendix 4.

4 Development

4.1 Facebook iOS Software Development Kit

Facebook provides open-source SDKs for different platforms such as JavaScript SDK, PHP SDK, iOS (Objective-C) SDK and Android (Java) SDK. With the help of these SDKs, developers can build their own applications and integrate with Facebook. Handling authentication and authorization, making API calls and interacting with a user by displaying a dialog is possible due to Facebook SDKs.[27,17] Since, the calendar application was written in Objective-C, the latest Facebook iOS SDK was downloaded from the official site for the thesis. It automatically allows developers to access Facebook platform APIs along with REST API, Graph API, FQL and Dialogs.[28] Some of the popular social applications created in Facebook using SDKs include Zynga, PopCap, foursquare.

4.1.1 Methods

Authentication is a process of examining whether the person or content is, in fact, the same as it is supposed to be. In general, there are three regulatory-approved authentication factors, out of which two factors are required, so that the person or content is authenticated. Hence, authentication is also known as two-factor authentication.[24] Similarly, authentication and authorization in Facebook is done by the OAuth 2.0 protocol.

The authentication methods in Facebook are defined in the header file called Facebook.h and are listed in table 1.

Table 1. Modified from Facebook Developers (2012)[29]

Methods	Parameters	Return Value
initWithAppId:andDelegate:	app_id delegate	Newly initialized Facebook instance
initWithAppId:urlSchemeSuffix:andDelegate:	app_id urlSchemeSuffix delegate	Newly initialized Facebook instance
authorize:	permissions	
extendAccessTokenIfNeeded	None	
handleOpenURL:	url	YES if the URL starts with fb[app_id]; //authorize, NO otherwise
logout:	delegate	
isSessionValid	None	YES if the access_token is valid, NO if it is invalid

- The Facebook iOS SDK authentication methods listed in table 1 are primarily used for following purposes:
- To initialize the Facebook object with the application Id or secret key of the apps.
- To initiate the user authentication and application authorization flow, which is necessary to follow-on API calls.
- For session handling to re-authenticate the user.
- For Facebook Login and Logout and for handling the Single Sign On URL call back. [29]

Dialogs are codes that provide a simple interface to display a dialog, add messages and adjust privacy settings. It can be invoked with either an HTTP request or using

SDKs. Dialogs are built seamlessly, meaning that they could be run on the web as well as the mobile. However, there are various modes of dialog display, namely page, popup and iframe. In case of a mobile, a dialog display varies according to the mobile screen a user has. It could be either touch or WAP.

Facebook currently provides seven dialogs, which are listed below:

- Feed Dialog allows a user to post a story on Wall and to his/her Friends' News Feeds.
- OAuth Dialog is a dialog method that is used for authorizing the applications to Facebook.
- Add Page Tab Dialog allows a user to add an application to the Facebook page which they administer.
- Friends Dialog allows a user to send a friend request to another user.
- Pay Dialog is used for making a purchase using Facebook credits.
- Request Dialog is a dialog to send a request to one or more of their friends.
- Send Dialog is a dialog to send Facebook messages to one or more friends.[30]

The instance methods to generate a Facebook platform dialog are written as follows:

Table 2. Modified from Facebook Developers (2012) [30]

Methods	Parameters
Dialog:andDelegate:	action delegate
Dialog:andParams:andDelegate:	action params delegate

In general, a *request* is a message sent between objects. Facebook requests are also a one to one communication between a sender and a recipient. The request can be either sent to single or multiple recipients at a time.[32] When sending a request to multiple recipients, a user can see only the request from the sender, not all other recipients. A request is further classified into two categories: User-to-User Requests and App-to-User Requests.

The request methods are defined in Facebook.h. The Facebook iOS SDK provides the following request types to connect with the Facebook server.

Legacy REST API has many methods for various tasks, such as Administrative Methods, Login/Auth Methods, Data Retrieval Methods, Publishing Methods, Facebook Connect Methods, Mobile Methods, Dashboard API Methods, Event API Methods, Custom Tags API Methods and Ads Methods however; these methods are on the verge of deprecation. [31]

The *FQL* works with the Graph API. It is the extension of the Graph API since it can combine multiple queries in one method. The structure of the FQL queries is similar to those of Structured Query Languages.[31]

The core concept of the Facebook is the Social Graph, and the *Graph API* provides the mechanism to access the social graph objects, for example people, photos or events along with its connections between multiple objects, for example friend relationships or photo tags. There are 25 different objects with a unique ID in social graph. Therefore, the objects can be accessed via its unique id. For example to get the basic information about self, <https://graph.facebook.com/abcd> can be used where, *abcd* is the unique user ID. [17]

4.1.2 Delegation and Protocols

Delegation is a simple but powerful design in object-oriented programming, where an object is capable of handling a task over to another helper object or in other words, one object acts to behave on behalf of, or in coordination with, another object. The helper object is known to be a delegate. The delegating object always has reference to the other object and whenever the delegating object is about to handle or has just handled the reference object, a message is sent to the other object. For instance, when the user clicks the close box, the window manager sends the delegate a `windowShouldClose:`, and the delegate returns the `BOOL` value about closing of 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 3, and the delegate works as an instance of that object. Hence, with the help of the delegation

it is possible to modify the behavior of an object and it is also not necessary for subclassing. The example is illustrated in Figure 3 [25.]

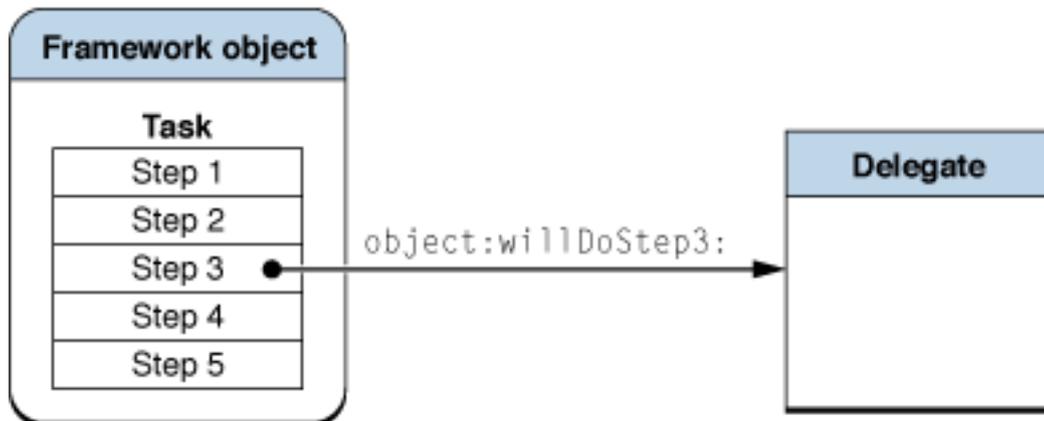


Figure 5. Delegation. Reprinted from Apple official documentation (2010) [25].

A protocol is a set of method declarations that an object agrees to implement in order to communicate with another object. It is a way to influence classes to implement a set of required and/or optional methods. Apple's documentation on protocols states that there are at least three reasons to use protocols [26]:

- "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." [26]

The concept of a protocol is that if a class adopts a protocol, it is necessary to implement all the required methods in the protocols it adopts. Correspondingly, the `FBDialogDelegate`, `FBRequestDelegate` and `FBSessionDelegate` protocol implement the methods that a Facebook object delegate may have. The methods of the protocol are defined in `FBDialog.h`, `FBRequest.h` and `Facebook.h` respectively. A detailed discussion of the protocols is presented below.

The `FBDialogDelegate` protocol implements the methods that are declared by the application to handle all the platform dialog callbacks, for example sending applications request to Facebook friends. The following methods are implemented to handle whether the dialog completes successfully, the dialog fails due to an error, user cancellation dialogs and the user clicks on a link in the dialog that could be a browser.

Table 3. FBDialogDelegate. Modified from Facebook Developers (2012) [33]

Methods	Parameters	Return Value
dialogDidComplete:	dialog	NA
dialogDidNotComplete:	dialog	NA
dialogCompleteWithURL:	url	NA
dialogDidNotCompleteWithURL:	url	NA
dialog:didFailWithError:	dialog error	NA
dialog:shouldOpenURLInExternalBrowser:	dialog url	BOOL

Table 3 is a collection of the methods under FBDialogDelegate. These methods are defined in FBDialog.h iOS Facebook SDK. One of the methods provides return value of bool indicating whether to open the external links into the browser or not. The rest of the methods provide no return value and mostly used for checking the successfulness of the dialog.

The *FBRequestDelegate* protocol defines the methods to handle the Graph API and FQL callbacks. Because the methods defined in the protocol are optional, the methods are implemented when they are needed. The delegate object should implement the *FBRequestDelegate* interface to handle request responses such as API call failure/success [14]. If the API call fails, the SDK will callback `request:didFailWithError:` method in the delegate whereas, a successful request will callback `request:didLoad:` in the delegate. The return value to the delegate could be an NSArray or an NSDictionary depending upon the single or multiple results.

Table 4. FBRequestDelegate. Modified from Facebook Developers (2012)[34]

Methods	Parameters
requestLoading:	request
request:didReceiveResponse:	request response
request:didFailWithError:	request error
request:didLoad:	request id
request:didLoadRawResponse:	request data

Table 4 is a list of the methods under `FBRequestDelegate`. The methods are defined in `FBRequest.h` in Facebook iOS SDK. These methods are used for accessing the Social Graph objects. The method `(void)requestLoading:(FBRequest*) request` is called before the application sends the request to the server. Similarly, `(void)request:(FBRequest*)request didReceiveResponse:(NSURLResponse*)response` method is called when the Facebook API request has returned a response. This method provides the raw response of the particular API call. The method `(void)request:(FBRequest*)request didLoadRawResponse:(NSURLResponse*)response` is similar to `(void)request:(FBRequest*)request didReceiveResponse:(NSURLResponse*)response` providing the `NSData` as the call back response. The `(void)request:(FBRequest*)request didLoad:(id)result` method provides the parsed response in the form of the object. The response shall be an array or dictionary depending on the type of the Graph API; for example the Social Graph object `Person` provides a response as an array, whereas the `Friends` object provides the response as a dictionary. During the assessment of the Social Graph object via the Graph API, the occurrence of the error is handled by the method `(void)request:(FBRequest*) request didFailWithError:(NSError*)error` by providing the type of the error along with its respective error code. For example the error in permission leads to get `facebookErrDomain` error 10000. [34]

The `FBSessionDelegate` protocol declares methods to handle session callbacks during the user authentication and permission authorization. All the methods defined here are mandatory. The protocol is used for Facebook user login, logout, access token expiration and extenuation.

Table 5. `FBSessionDelegate`. Modified from Facebook Developers (2012) [35]

Methods	Parameters
<code>fbDidLogin</code>	None
<code>fbDidNotLogin:</code>	Cancelled
<code>fbDidLogout</code>	None
<code>fbDidExtendToken:expiresAt:</code>	accessToken expiresAt
<code>fbSessionInvalidated</code>	None

Table 5 is a collection of the methods under `FBSessionDelegate` Protocol that is defined in `Facebook.h` in iOS Facebook SDK. The method `(void)fbDidLogin` is called

when the user has successfully logged in. In this method, the extension of the access token and its expiry date is extended by calling another method *(void)fbDidExtendToken: (NSString*)accessToken expiresAt :(NSDate*)expiresAt*. When the user cancels the login screen of the Facebook server, the method *(void)fbDidNotLogin* is called to handle the session and the access token accordingly. Similarly, the method *(void)fbSession Invalidated* is called if the accessToken is too old or expired, the application is disabled or uninstalled, the user revoked the applications permission or the user changed the password. Whenever the user logs out from the Facebook application, the method *(void)fbDidLogout* is invoked removing the current access token, key session and the cache from the safari browser. [35]

4.2 Analysis and Design Details

As discussed earlier in section 2.2, the application developed in this project is an extension to the calendar application, and the main goal was to integrate the calendar application with the Facebook application. Many features in the Facebook can be integrated. However, it is not possible to do everything within a certain period of time. Therefore, this application is more focused on dealing with the events that the user has in Facebook (either normal events or birthday events) and events that are created in the calendar application.

The calendar has three tab buttons: the first tab shows the login view to create a user account and save the events in the server, second tab has a calendar view as shown in figure 6, and the last tab shows a list of events that have been created and also fetched from the Facebook account. The synchronization of events helps to remind the user of the events she or he has in Facebook. It is also useful if the user deactivates the Facebook account but can still have those events in his/her local phone. The events are distinguished with different colors, i.e. a green marker shows the event created in the phone and a blue marker shows the events fetched from the Facebook. The Events tab allows a user to access his/her Facebook account to view news feed, post on own/friend's wall, sync events or birthday event to the local calendar and also send the selected events to Facebook. The overview of the calendar application is shown in figure 6:



Figure 6. Calendar view of the application.

In figure 6, only the dates with events have markers and the current data is highlighted. The calendar application also shows the week number in the right column of its user interface.

4.2.1 User's Perception

From the user's point of view, the application has two different approaches. The student calendar is accessible by everyone whereas the extended feature is available only to those who have got a Facebook account. The users with Facebook account can log into the Facebook application by using the Facebook user name and password and access the events feature from their iPhone straight away. The UML diagram, shown in figure 7, shows the background of the application.

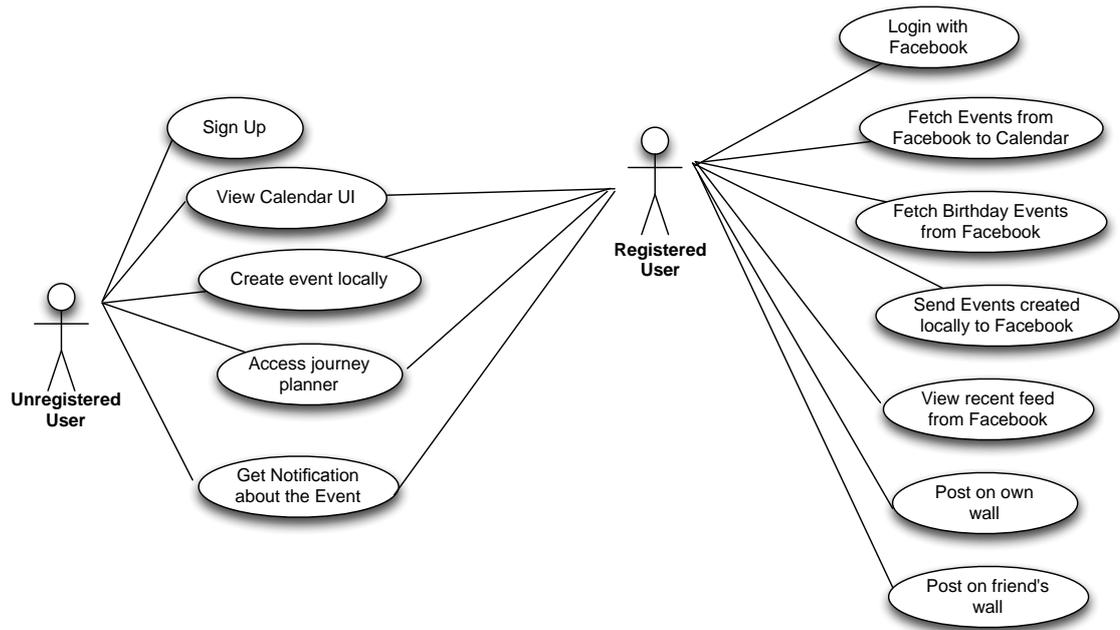


Figure 7. Use Case Diagram for the application.

Initially, the registered user logs into the application using Facebook credentials. A view with the user profile name, picture and a list of the menu items with the option to access events, groups, post on wall and logout appears. Whereas, the unregistered user can access basic features of the calendar such as create event locally, get notifications, and access journey planner.

4.2.2 Authentication Workflow

The following sequence diagram, figure 8 shows the Facebook authentication process of a user to access his/her Facebook resources. In the process of authentication, a Facebook application is used instead of the server to request access to resources controlled by the Facebook user and the Facebook server. An application should be registered by Facebook to have its application ID and secret key. The application behaves as a communication bridge between the calendar application and the Facebook server.

When a user requests to have some protected resources, the calendar application is redirected to the Facebook authorization server with its application ID and the URL, the user should be directed back to the calendar application after the authorization process. The user receives back Request for Permission form. If the user authorizes

the application to get his/her data, Facebook authorization server redirects back to the URL that was stated before along with authorization code.

If the permission is granted, the Facebook application gets the access token for an FB user and can perform authorized requests on behalf of that FB user. The authorized requests are performed including the access token in the Facebook Graph API requests. Figure 8 gives a vision of the authentication work flow.

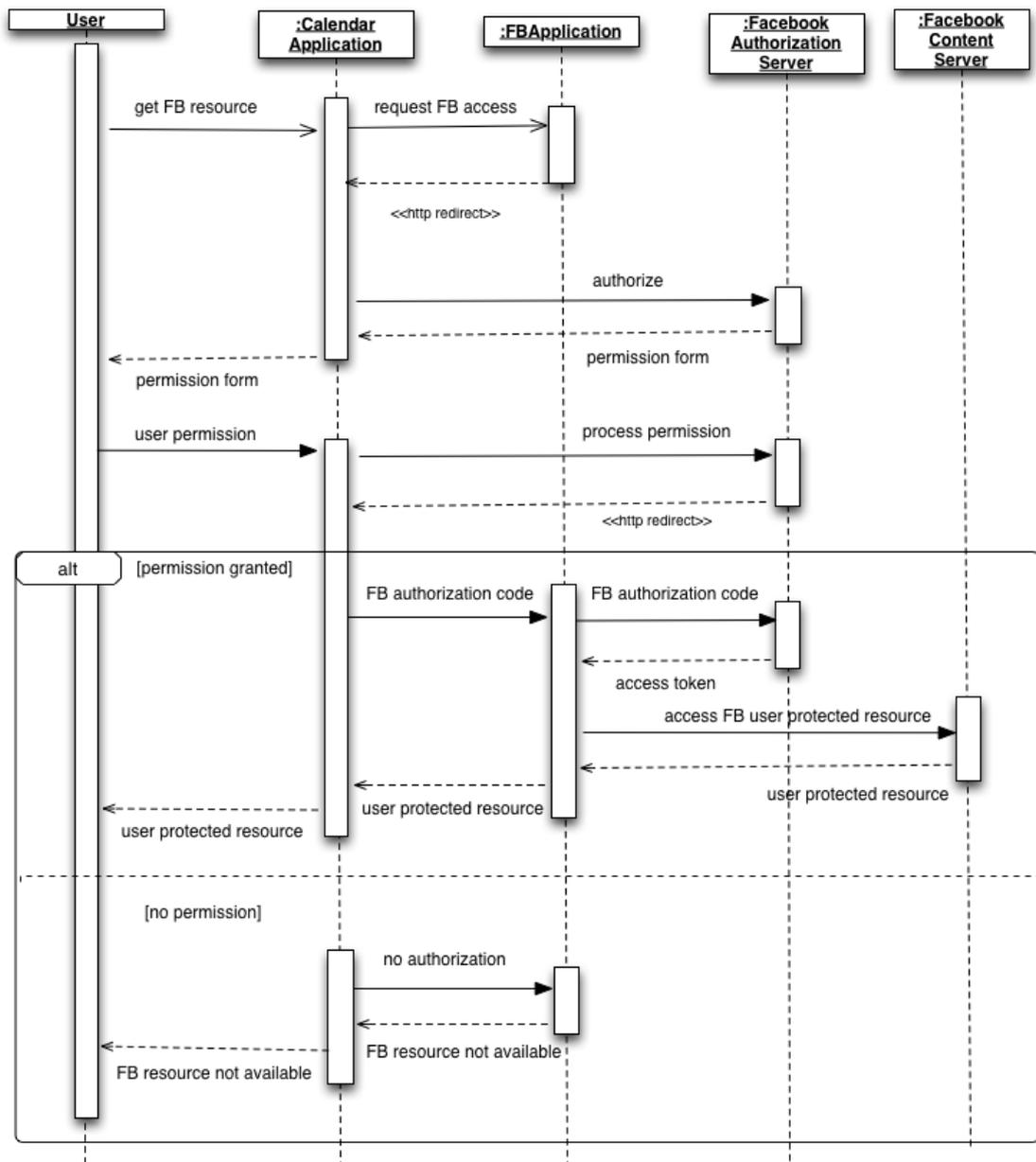


Figure 8. Facebook authentication overflow (sequence diagram).

As described in figure 8, if the user does not authorize the application, Facebook issues redirect requests to the URL specified beforehand, and add the error reason parameter to notify the application that the authorization was denied.

4.2.3 Application Implementation

The application was divided into two parts during the development phase. The first phase of the application consisted of developing the fully functional calendar where users can create an event of their choice on a particular day, display the list of the events that were created and present the event in detailed form once the event is selected. The functionalities of the calendar such as mark the event dates, current day, day selection and possibilities of creating an event from different approaches, such as tap on the plus button, double tap on the dates were focused in the first phase.

First Phase - Calendric Functionalities

Some of the important classes that are needed to populate calendric functionalities are presented below in figure 9.

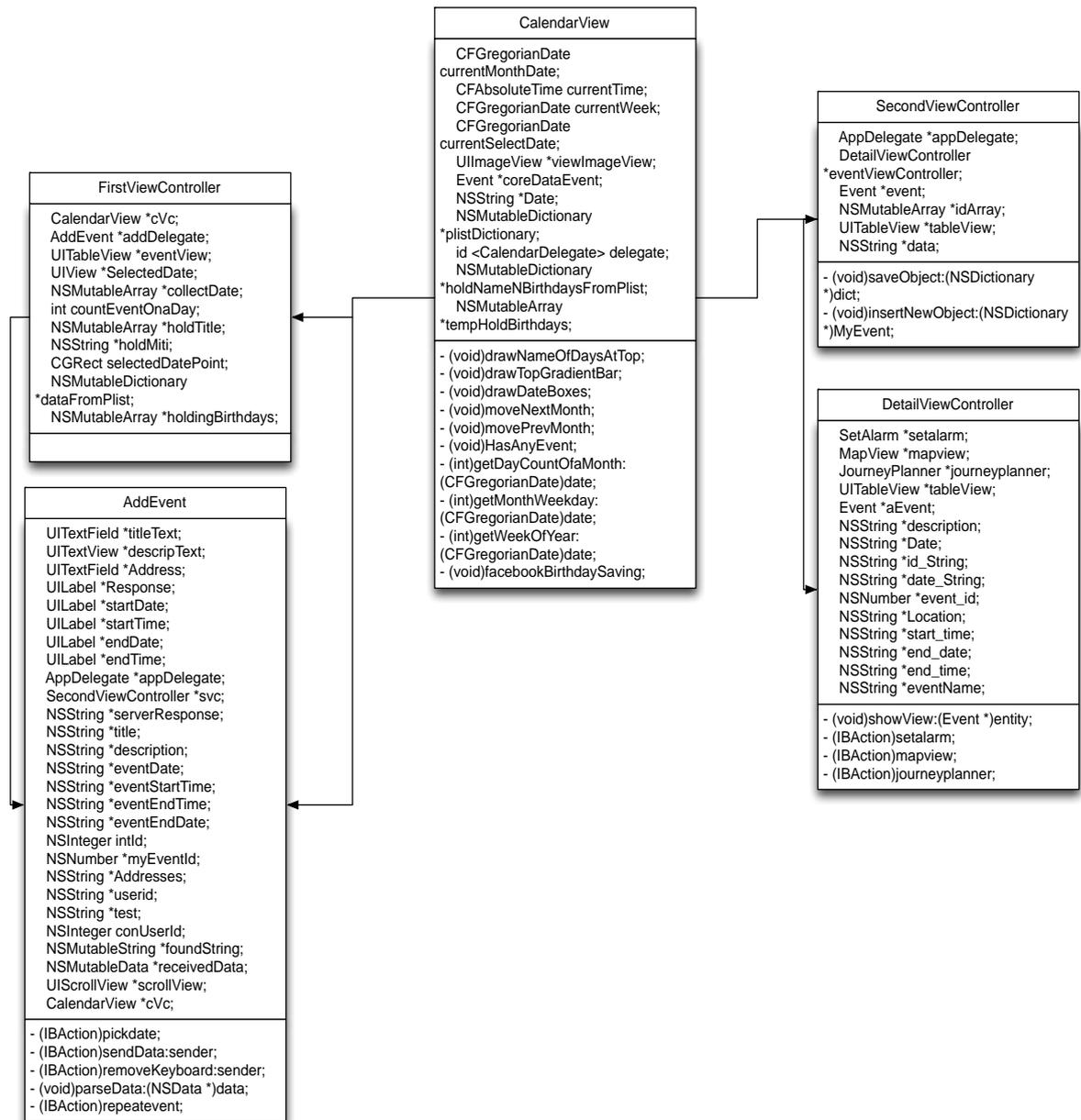


Figure 9. Calendric Classes

Figure 9 is the accumulation of some of the important classes used for generating calendric functionalities. The class such as AppDelegate and its application life cycle describing methods were not taken into the presentation of the calendric classes, as it is similar for all the iOS development. One of the important classes in figure 9 is CalendarView class, which is a subclass of UIView class. The primary goal of this class is to create calendar dates in a systematic manner and present them in its optimal position. In order to present the calendar, the empty view is created on top of the default view of the FirstViewController and overridden by CalendarView. All the presentation in the CalendarView was created using the frameworks such as CoreGraphics and Quartz. Methods such as

(int)getDayCountOfaMonth:(CFGregorianDate)date, *(int)getMonthWeekday:(CFGregorianDate) date*, *(int)getWeekOfYear:(CFGregorianDate)date* were used to get the current day of the month, total number of days in a month and the week number of the year respectively. Similarly, the methods such as *(void)drawNameOfDaysAtTop*, *(void)drawTopGradientBar*, *(void)drawDateBoxes* were used to create days of the month, the Title bar of the calendar and the grid respectively. In order to distinguish the dates with the event from the normal dates, the method *(void)HasAnyEvent* was implemented in a way that the dates were extracted from the persistent storage and a mark was placed on the dates. For the back and forth functionality, the methods *(void)moveNextMonth* and *movePrevMonth* were implemented accordingly.

All the gesture recognition functionalities, such as swipe left, right, single tap and double tap, were implemented in the FirstViewController class. The AddEvent viewController class was invoked from FirstViewController class by tapping on the plus button or by tapping on the dates twice. The AddEvent class lets the user to create an event, and once the event is created, it is sent to the server and saved into the core data for persistence use. The method *(IBAction)pickdate* was used to select the event date and time from the pickerView. Some of the important methods of the AddEvent class are *(void)parseData:(NSData*)data* and *(IBAction)sendData: sender* , which parse the created event data into xml format and send it to the server respectively. At the same time, the created event was saved into persistent data storage as well.

The SecondViewController class was used to present the list of created events. It has two buttons, named Social media and update, in order to get connected with the Facebook server and update the event from the client server. Once the user gets the list of the events, tapping on the particular event invokes the DetailViewController class presenting the event in detailed format. In the DetailViewController class the methods such as *(IBAction)setalarm* , *(IBAction)mapview*, *(IBAction)journeyplanner* were used for setting the alarm for the event, searching for the location and presenting the map of the searched location.

Second Phase – Facebook Integration

For the second phase of the application, the iOS Facebook SDK was downloaded from the office site. The static library for the SDK was configured as described in the official documentation provided by Facebook developers. Some of the important classes are presented in figure 10.

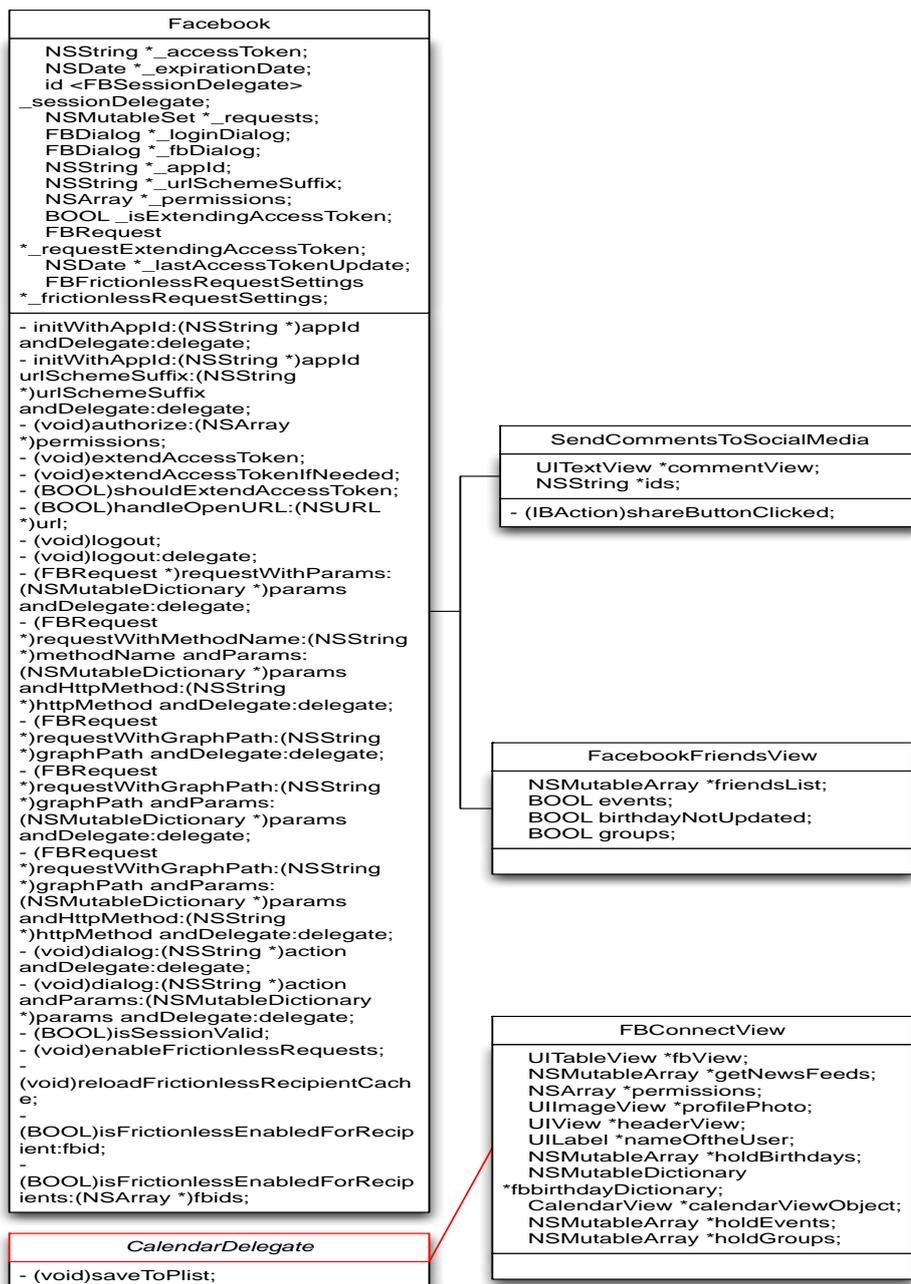


Figure 10. Facebook functional classes

Figure 10 is a presentation of some of the important classes used for integrating Facebook into the calendar application. The iOS Facebook SDK provided all the

required methods to communicate with the Facebook server. At the beginning of the second phase, the static library of Facebook SDK was configured to work with automatic reference counting (ARC). Since, the back-end for the calendar application was out of scope, an application was created in Facebook developer under the name, Calendar. The calendar application registered in Facebook developers provides the access token and all the required permission on behalf of the user.

In order to work with the Facebook server, some changes were made in property list (.plist) file that handles configuration of the application. A new row named URL types was created with an item, URL Schemes, which also contains a single value, that is, fb followed by app ID, which is 15 digits in length. This was done to enable SSO support for the application.

The creation of the Facebook object and initialization with the Facebook Calendar Application id along with the delegate was defined by the method `(id)initWithAppld:(NSString *)appld andDelegate:(id<FBSessionDelegate>)delegate`. The method was implemented in AppDelegate of the Calendar application. The FBConnectView class was invoked from the button named Social Media, which lies in the SecondViewController class. The initial work of the FBConnectView is to check the session of the application, which is carried out by the method `(BOOL)isSessionValid`. For the first time the session is invalid, and therefore the method `(void)authorize:(NSArray *)permissions` had to be called along with the required permission such as `@ "offline_access", @ "publish_stream", @ "friends_birthday", @ "friends_likes", @ "user_likes", @ "friends_events", @ "user_groups", @ "friends_groups", @ "read_stream"`.

The authorize method invoked the method `(BOOL) application:(UIApplication*)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation` returning the valid URL to the Facebook Calendar application from which the access token could be achieved. The session delegate method called the `fbDidLogin` method where the achieved access token was stored into the Facebook accessToken variable along with the expiration date. The required request methods were implemented as per the need of the information. As per the project, the private methods such as `(void) getMyBasic`

Information and *(void)getNewsFeed* methods were called for users' basic information and home feeds.

Buttons such as *PostOnWall*, *PostOnFriendWall*, *AppsRequest*, *GetFBBirthday*, *Sync Events* and *Get Groups* were created programmatically. The behavior of the buttons can be described as follows:

PostOnWall button invokes the method *(void)dialog:(NSString *)action andDelegate:(id<FBDialogDelegate>)delegate* to write something on the user's wall. The action section of the method holds "me" as the identifier.

PostOnFriendWall button invokes the method *(FBRequest*)requestWithGraphPath:(NSString*)graphPath andDelegate:(id<FBRequestDelegate>) delegate* in which the *graphPath* holds the identifier "me/friends" that gets the list of the user's current friends. For displaying the friends' list, the *FacebookFriendsView* *ViewController* class is pushed on top of the view stack. Tapping on the friends list invokes the *SendCommentsToSocialMedia* *ViewController* class that lets the user to write on the wall of the selected users.

Tapping on the button *AppsRequest* invokes the method *(FBRequest*)requestWithGraphPath:(NSString*)graphPath andParams: (NSMutableDictionary *) params andDelegate:(id <FBRequestDelegate>)delegate* , in which, the *graphPath* holds the identifier "appsRequest" and *Params* holds some dictionary objects. This method enables a browser and presents the list of friends. By tapping on the name list, the application request can be sent to multiple friends at a time.

Tapping on the button *Get Fbbirthday* invokes the *FacebookFriendsView* *viewController* and is pushed on top of the view stack. The Facebook API call method *(FBRequest*)requestWithGrapPath:(NSString*)graphPath andDelegate: (id<FBRequestDelegate>)delegate* is invoked in which the *graphPath* holds the identifier "me/friends" which gets the list of friends data. Before pushing the *FacebookFriendsView* *viewController*, the method *(void)facebookBirthdaySaving* of the *CalendarView* of the *FirstViewController* class is called, which calls its delegate method *(void)saveToPlist* where the list of friends data is saved into the plist named *FBbirthdayInfo.plist*.

Tapping on the button *Sync Events* invokes the `FacebookFriendsView` viewController and is pushed on top of the view stack. The method `(FBRequest*) requestWithGraphPath:(NSString*) graphPath andDelegate: (id<FBRequestDelegate>) delegate` is invoked in which the `graphPath` holds “me/events” identifier, which indicates all the events created by the user and the event that the user has replied as attending in the Facebook. The data of the events is separated and stored in the core data in the `FacebookFriendsView` viewController and shown in the events list. The data insertion in the core data is done in the way that multiple data is avoided by using the `NSPredicate` that checks the title of the event before storing it into the persistent store.

Tapping on the button *Get Groups* the Facebook API call method `(FBRequest*) requestWithGrapPath:(NSString*) graphPath andDelegate:(id<FBRequestDelegate>) delegate` is invoked in which the `graphPath` holds the identifier “me/groups”, which gives the list of the groups that the user has created and is a member of. After that, the `FacebookFriendsView` viewController is pushed on top of the view stack that presents the list of the groups.

5 Result and Discussion

Learning about social media platform, iOS and its developing tools was also a part of the thesis. It was achieved by gathering materials from different reliable sources, online tutorials and the official sites of Apple and Facebook (www.developers.apple.com and www.developers.facebook.com). Moreover, guidance from the supervisor was admirable during the project. Also, the goal set as in the introduction to integrate a student calendar with Facebook was met.

The thesis project produced all the basic features that are indispensable for integration with Facebook, such as exporting the events created in the calendar application into user's Facebook account and vice versa. Similarly, importing the birthdays of the user's Facebook friends into calendar application was attained. Some of the Facebook functionalities such as writing into user's wall and friend's wall were also achieved making the calendar application more feature-rich. The integration with Facebook made the calendar more user-friendly and efficient in the sense that it avoids the necessity to log into Facebook account, into native Facebook application or into browser to view the events, birthdays, groups, news feeds, writing into timeline and friends wall.

During the implementation phase of the project, several problems were faced and eventually resolved with the help of various tools such as the official documentation of Facebook, the official documentation of Apple, a demo application of Facebook (Hackbook for iOS), and ARC version of Facebook iOS SDK. The implementation phase went through some technical problems. For example, the birthday format of the persistence data model was completely different than the birthday formats pulled from Facebook. As an alternative, a property list was created to handle birthdates. As for the event dates, they were changed into the application persistence data model and handled accordingly.

In software development, the achieved product has always room for further development. As in the case of the calendar application, there are several functionalities to be fixed and further developed. Currently, the user can create an event but in practice, the calendar application should also allow the user to delete the

events that are not needed any more. Since the application synchronizes with Facebook events, it should update the event changes accordingly. Rather, it duplicates the event in the calendar application. However, the application should not delete birthday events even though the user deactivates his/her Facebook account. It was also noticed that the event created from the calendar application, when sent to Facebook twice, was created as two events. These features could be further studied in the future to develop a fully functional application.

The application works on the offline mode, but some of the work for online access was done. Since, the project focused on social media integration, the online user management and group were not taken into consideration thoroughly. Hence, the user management and group handling of the user and integration with several social media sites, such as Twitter or Google plus could be taken into consideration in further development. As for the calendar functionalities, various views, such as a day view, week view and month view can be developed. The user interface design of the application was not tested thoroughly. Some work for improving the appearance of the application could be done in the future.

The application was tested mostly on a simulator rather than on real devices. Therefore, testing on different versions of iOS devices should be done for reliability and applicability of the application. Some of the Facebook functionalities, such as photo sharing and video update could be considered since most of the functionalities are already integrated, which will make the calendar application vivacious.

6 Conclusion

The goal of this project was to create an application that would integrate the calendar application with the Facebook application, especially for handling events, getting notified about the event and being able to update the status from the calendar application. The goal of the project was met with additional Facebook functions in the calendar application, such as post on one's own wall, post on a friend's wall, view recent feeds and one can also send a calendar application request to friends from the application.

The application was developed in Xcode, which means the application is limited within the iOS devices. The calendar application can be solely developed using the Xcode IDE and to further work with Facebook, it requires the Facebook iOS SDK. The Xcode IDE is very helpful in unit testing and finding leaks in the process of application development. Although the application was tested in an iPhone simulator and an iPhone device, the application still needs to be thoroughly tested.

The integrated Facebook features were implemented in the Calendar application in the way that it is easy to deploy into various other applications as per the developer's need. Hence, the application can be taken as a reference project to integrate Facebook into various applications. The calendar application is demanding and frequently used by users; therefore, such an application should be developed in different platforms such as Android, Windows Phone, Symbian and Blackberry. The application can be developed for the tablet version of iOS and Android as well.

References

1. Wikipedia. Facebook. [online]; 2012.
URL: <http://en.wikipedia.org/wiki/Facebook>. Accessed 28 August 2012
2. Pingdom. Exploring the software behind Facebook, the world's largest site. [online]; 2010.
URL: <http://royal.pingdom.com/2010/06/18/the-software-behind-facebook>. Accessed 9 May 2012
3. PCWorld. Why Linux Is More Secure Than Windows. [online]; 2010.
URL: http://www.pcworld.com/businesscenter/article/202452/why_linux_is_more_secure_than_windows.html. Accessed 29 August 2012
4. LogicMethodIT. LAMP (Linux, Apache, MySQL,PHP) [online]; 2011
URL: <http://www.logical-software-development.com/capabilities/lamp>. Accessed 29 August 2012
5. Ullman L. PHP 6 and MySQL 5 For Dynamic Websites. Berkeley, CA: Peachpit Press; 2008.
6. Facebook Developers. HipHop for PHP: Move Fast. [online]; 2012.
URL: <http://developers.facebook.com/blog/post/358>. Accessed 22 May 2012
7. Agarwal A. Facebook's Architecture [online]; 2009
URL: <http://www.infoq.com/resource/presentations/Facebook-Software-Stack>. Accessed 29 August 2012
8. Github. Scribe. [online]; 2010.
URL: <https://github.com/facebook/scribe/wiki>. Accessed 9 May 2012
9. Facebook. Operational data store. [online]; 2012.
URL: <https://www.facebook.com/pages/Operational-data-store/129170837168602>. Accessed 10 May 2012
10. The Open Group. Single Sign-On. [online]; 2010.
URL: <http://www.opengroup.org/security/sso>. Accessed 19 August 2012
11. Clarcq J. Single Sign-On Architectures, Infrastructure Security, International Conference, InfraSec 2002, Bristol, UK, October 2002 Proceedings S.40-58.

12. How to implement Single Sign-On with Force.com. [online]. Salesforce.com,Inc; 2000-2011.

URL:http://wiki.developerforce.com/page/How_to_Implement_Single_Sign-On_with_Force.com. Accessed 7 May 2012
13. Huntington Ventures. Single Sign-On [online]; 2006.

URL:<http://www.authenticationworld.com/Single-Sign-On-Authentication>. Accessed 14 April 2012
14. Wikipedia. Facebook Platform. [online]; 2012.

URL: http://en.wikipedia.org/wiki/Facebook_Platform. Accessed 28 August 2012
15. PHPEveryDay. Facebook Programming: Facebook Platform Step By Step Tutorial-PHPEveryDay. [online]; 2010.

URL:<http://www.phpeveryday.com/articles/Facebook-Programming-Facebook-Platform-P845.html>. Accessed 11 April 2012
16. Ortiz C. Introduction to Facebook APIs.[online]. IBM; 16 December 2010.

URL:<http://www.ibm.com/developerworks/library/x-androidfacebookapi>. Accessed 9 May 2012
17. Facebook Developers. Graph API.[online]; 2012.

URL:<https://developers.facebook.com/docs/reference/api>. Accessed 22 May 2012
18. Maver J and Popp C. Essential Facebook Development. Boston, MA: Pearson Education; 2010.
19. Stay J. FBML Essentials. Sebastopol, CA 95472: O'Reilly Media; 2008.
20. Purdy D. Moving to a Modern Platform. [online]. Facebook; 30 September 2011.

URL:<https://developers.facebook.com/blog/post/568>. Accessed 29 August 2012
21. Wikipedia. iOS SDK. [online]; 2012.

URL:http://en.wikipedia.org/wiki/IOS_SDK. Accessed 29 August 2012
22. Lee W. Beginning iOS 5 Application Development. Indianapolis, Indiana: John Wiley & Sons; 2012.
23. Apple Inc. iOS Technology Overview. [online]; 2011.

URL:http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1. Accessed 30 August 2012
24. Wikipedia. Authentication. [online]; 2012.

URL:<http://en.wikipedia.org/wiki/Authentication>. Accessed 29 August 2012

25. Apple Inc. Cocoa Fundamentals Guide. [online]; 2010.

URL:<http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/CocoaFundamentals/CocoaDesignPatterns/CocoaDesignPatterns.html>. Accessed 30 August 2012

26. Mac OSX Developer Library. Protocols. [online]; 2010.

URL:http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/ObjectiveC/Chapters/ocProtocols.html#//apple_ref/doc/uid/TP30001163-CH15-TPXREF144. Accessed 29 August 2012

27. Dannen C and White C. Beginning iOS Apps with Facebook and Twitter APIs for iPhone, iPad, and iPod touch. New York City: Apress Inc; 2011.

28. Facebook Developers. Making Your iOS Apps Social. [online]; 2010.

URL:<https://developers.facebook.com/blog/post/400>. Accessed 28 June 2012

29. Facebook Developers. Authentication. [online]; 2012.

URL:<https://developers.facebook.com/docs/reference/iossdk/authentication>. Accessed 1 September 2012

30. Facebook Developers. Dialog. [online]; 2012.

URL:<https://developers.facebook.com/docs/reference/iossdk/dialog>. Accessed 1 September 2012

31. Facebook Developers. Request. [online]; 2012.

URL:<https://developers.facebook.com/docs/reference/iossdk/request>. Accessed 1 September 2012

32. Facebook Developers. Requests Dialog. [online]; 2012.

URL:<https://developers.facebook.com/docs/reference/dialogs/requests>. Accessed 1 September 2012

33. Facebook Developers. FBDialogDelegate. [online]; 2012.

URL:<https://developers.facebook.com/docs/reference/iossdk/FBDialogDelegate>. Accessed 30 August 2012

34. Facebook Developers. FBRequestDelegate. [online]; 2012.

URL:<https://developers.facebook.com/docs/reference/iossdk/FBRequestDelegate>. Accessed 30 August 2012

35. Facebook Developers. FBSessionDelegate. [online]; 2012.

URL:<https://developers.facebook.com/docs/reference/iossdk/FBSessionDelegate>. Accessed 30 August 2012

Appendix 2: Event View and Facebook Functionality



Figure 12. Event view before Facebook integration

Figure 12 is the screenshot of event view before the Facebook synchronization. The user can tap on the SocialMedia button in order to sync with Facebook. The application asks to input the user's credentials to connect into their respective account if not logged on beforehand.

Figure 13 is the Facebook user authentication view and Facebook application authorization view.



Figure 13. The Facebook user credential input view.

By providing the credentials, it takes user to a new view and asks user to authorize the permissions that the application needs to take care of in order to link Facebook application with the iPhone application. By tapping on okay button on the Calendar view, leads user to the Facebook view that is present below.

Appendix 3: Facebook Functionality

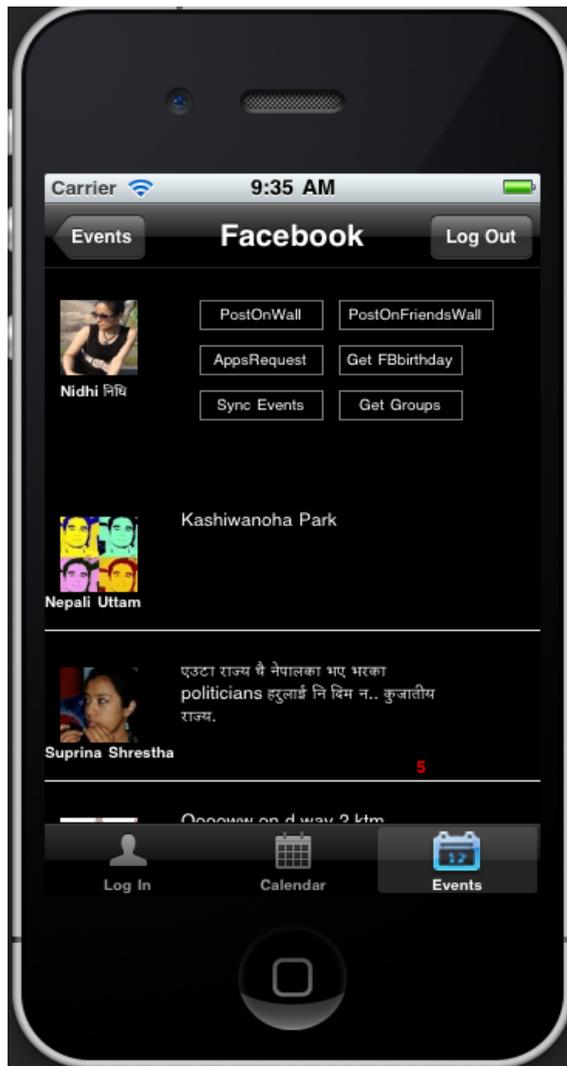


Figure 14. The Facebook main view.

Figure 14 is the screen shot of the Facebook main view consists the Facebook functional buttons along with some of the recent home feeds of the user. By tapping on PostOnWall button, the user can write on their Facebook wall that is shown below.

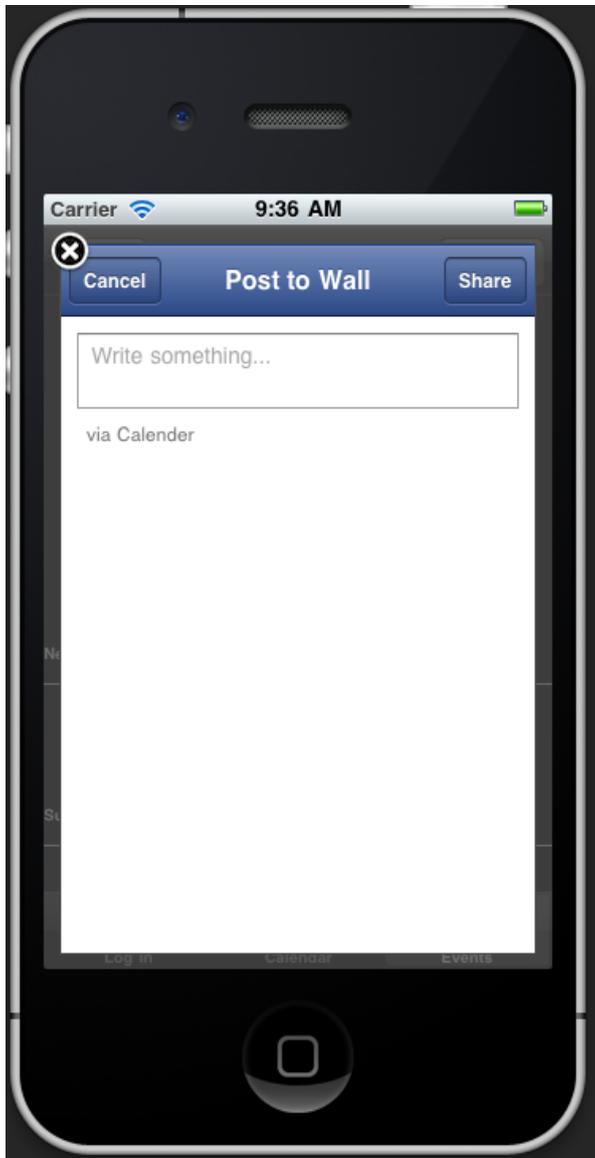


Figure 15. The Post To Wall view.

Figure 15 is the popup view to write something into the users' wall. The share button allows sharing the contents into Facebook and cancel button dismisses the view. Similarly, the user can tap on the PostOnFriendsWall button to write on the friends' wall. The list of the friends is populated in the ascending order. The user can tap on the friends name to write something on their wall. The Facebook Friends view and Write on Wall view are shown as in figure 16 below.



Figure 16. Write on Friends' Wall view.

Figure 16 is the Views to write on friends' wall. After the user taps on PostOn-FriendsWall button, the list of friends view is presented. By selecting the friend, Write on Wall view is shown to the user to write down the content that to be written in friends wall. The Share it button is to share the contents and dismissing the Write on Wall view.

Similarly, the user can send the application request to their friends. The application request can be sent by tapping on AppsRequest button and the Send Request view is shown below

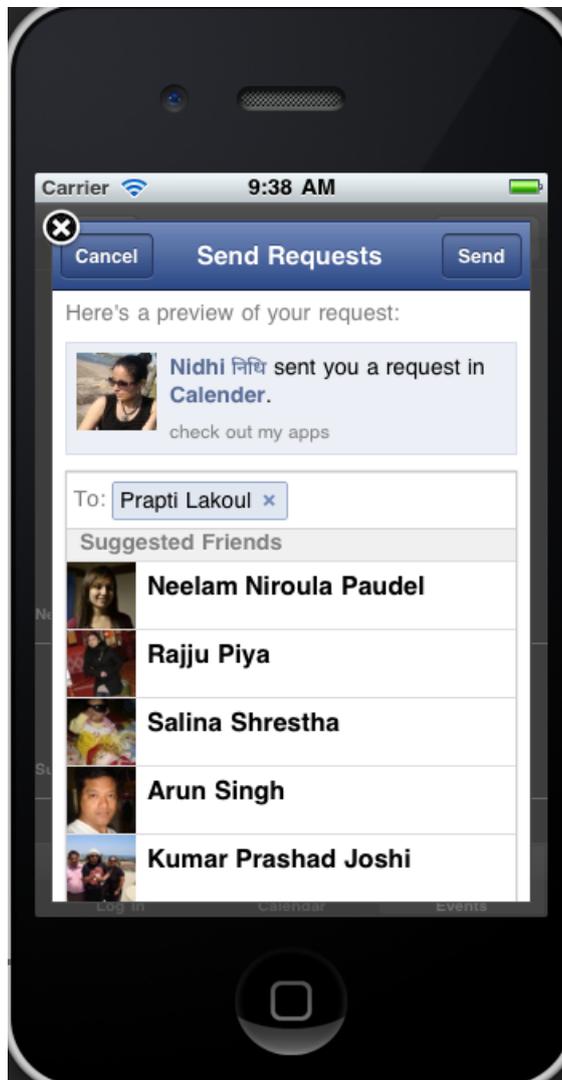


Figure 17. The Application Request View.

Figure 17 is the application request view to the friends. In the figure, the user has selected one of the friends from the suggested Friends list. In the same way multiple friends can be selected to send the request at a single instance. The user can get the birthdays of the friends by tapping on the Get FBbirthday button.



Figure 18. The Facebook birthdays extraction process.

Figure 18 is the process of getting Facebook birthdays into the Calendar application. The tap on the Get FBbirthday button presents the alert message as presented in the left figure. The tap on the cancel button leads to populate FB Friends view along with another alert view, notifying the user that the friends' birthdays are not imported to the Calendar application. By tapping on Get button of the initial alert message, the user can import all the available friends birthdates into the Calendar application.

In the similar manner, the user can import the Facebook events by tapping on the Sync Event buttons. The process of event importation is shown below.



Figure 19. The Facebook Events View.

Figure 19 is the list of Facebook events in a table view. Once, the user taps on the Sync Events button, the Facebook events are pulled into the application persistence storage and the list of the events are populated in a table view. In order to avoid the multiple entries of events into the persistence storage, the events title are checked into the persistence before fetching them permanently. In this way, the redundancy is handled and for user's convenience, the alert message is shown as given in the right figure above.

The Facebook groups of the user can be viewed with the help of GetGroups button. The GetGroups view is given below.



Figure 20. The Facebook Groups list view.

Figure 20 is shows the list of Facebook groups the user has been member of or created. After fetching the list of the groups from the Facebook server, user can write into the wall by selecting the group that populates the Write on Wall view.



Figure 21. Exporting Events to Facebook.

Figure 21 is the detail view of an event, which has EventToFB button on its topmost right corner. The button EventToFB helps to export event from calendar application to the Facebook server. After tapping on the button, the application asks user to confirm whether he/she wants to send the event to Facebook server as in figure on the right hand side.

Appendix 4: The iOS Frameworks

FRAMEWORK	Purpose
Accelerate	Accelerating math functions
AddressBook	Accessing user's contacts
AddressBookUI	Displaying Addressbook
AssetsLibrary	Accessing user's photos and videos
AudioToolbox	Audio data streams; playing and recording audio
AudioUnit	Audio units
AVFoundation	Objective-C interfaces for audio playback and recording
CFNetwork	WiFi and cellular networking
CoreAudio	Core audio classes
CoreData	Object-oriented persistent data storage
CoreFoundation	Similar to Foundation framework, but lower level (don't use unless you absolutely must)
CoreGraphics	Quartz 2D
CoreLocation	User's location/GPS
CoreMedia	Low-level audio and video routines
CoreMotion	Accelerometer and gyro functions
CoreTelephony	Telephony functions and routines
CoreText	Advanced text layout and rendering
CoreVideo	Pipeline model for digital video
EventKit	Accessing user's calendar
EventKitUI	Displaying standard system calendar
ExternalAccessory	Hardware accessory communication interfaces
Foundation	Cocoa foundation layer
GameKit	Peer-to-peer connectivity
iAd	Displaying advertisements
ImageIO	Reading and writing image data
IOKit	Low-level library for developing iPhone hardware attachments
MapKit	Embedding map in application and geocoding coordinates
MediaPlayer	Video playback
MessageUI	Composing e-mail messages
OpenAL	Positional audio library
OpenGLES	Embedded OpenGL (2-D and 3-D graphics rendering)
QuartzCore	Core animation
QuickLook	Previewing files
Security	Certificates, keys, and trust policies
StoreKit	In App purchasing
SystemConfiguration	Network configuration
UIKit	iOS user interface layer