A PROTOTYPE FOR GROUP CALENDAR ON GOOGLE APPS ENGINE

MS PHAKAWIPHA SUPHISING

MR SUPHAWIT WOTISOMBOONPHUN

A SPACIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIRMENT FOR DEGREE OF BACHELOR OF SCIENCE

IN COMPUTER SCIENCE INTERNATIONAL PROGRAM

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2010

Title	A Prototype for group calendar on Google Apps Engine	
Student	Ms.Phakawipha Suphising	50050255
	Mr.Suphawit Wotisomboonphun	50050258
Degree	Bachelor of Science	
Major	Computer Science	
Academic Year	2010	
Advisor	Mr.Suntana Oudomying	

ABSTRACT

Nowadays many people use the service of online calendar. The online calendar user also makes use of viewing friends' calendar to improve the efficiency of social network activities. It would be even more benefit to users if one can view an integrated calendar of the calendar and anyone. Google App Engine is an emerging platform to allow developers to enhance the application domain of the social network era. With the idea in mind, we propose an application to union users' calendar.

Keyword: Google APIs, CSS, JQuery, JSON, HTML, HTML DOM, JavaScript, PHP and MySQL.

Table of Contents

Table of Contents (cont.)

List of Figures

List of Table

Chapter 1

Introduction

This chapter explains the origin of the problem and the solution we envision. It also covers the scope of this work, the benefits of creating this system and the plan to complete the project.

1.1 Importance and cause of problem

Appointment is important when you need to meet someone at somewhere to work. They need to agree together. Sometime they need some file, video or work for their meeting but we find that there are not much social appointment network on web application.

Development of Appointment system by using Google app Engine to run your web applications on Google's infrastructure and using Google APIs tools such as, Google Calendar APIs, Google map APIs, Google talk APIs and Google Friend connect APIs should allow us to unlock what current social appointment network cannot provide. The application would allow users to share their appointments among friends so that the offer better collaboration in this particular social network era.

1.2 Purpose of the special topic

To create a new Google app engine compatible social network application for managing appointments among social network users.

1.3 Coverage of the special topic

1) Create new web application combining appointment and social network.

2) Implement program with Google APIs to make user convenient to make an appointment.

3) Users can share their calendar among their friends and chat together on web along with making appointment with friend whom they are speaking.

1.4 Expected benefits

1.4.1 Benefit to the system developers

1) Benefit from learning and gaining experience from developing Google APIs tool.

2) Benefit from studying about efficient application design in several aspects for example, system design according to the functional requirements, interface functional design.

1.4.2 Benefit to the users

1) User gets new innovation that is social network appointment

2) User can chat with his friend along with make appointment together

3) User gets the new edged technology, Google APIs providing new technology for easily making appointments

1.5 Implementation procedures

1) Study the current system to analyze the problems and use as guidelines for improvement, redesign, and analyze to implement the improved system for the real world usage.

2) Study about Google App and Google APIs, which is the current technology for the system improvement.

3) Collect related information for the implementation.

4) Analyze and design operating processes and system databases.

5) Implement the system as designed.

6) Test the implemented system and enhance it to perfection, and point out its capacity, limitation, and eliminate a problem regarding the operation.

7) Document the user manual and reference for this special topic study.

1.6 Project Planning

1 Jul - 31 Jul 2010	Study and data collection systems.
16 Jul - 30 Aug 2010	Analysis system and database.
15 Aug - 30 Sep 2010	Developed preliminary program.
1 Oct - 31 Dec 2010	Preliminary testing and debugging program.
1 Jan - 26 Feb 2011	Finalize the application.
27 Feb – 31 Mar 2011	Documentation of special projects.

1.7 Equipment used in making special problems.

1.7.1 Description of the computer equipment.

Minimum computer minimum specification require

1) CPU :Intel Pentium 4 or higher
2) GPU: (graphic card) GMA 900
3) Ram: Up to 256 MB and Up to 500 MHZ
4) Harddisk: Up to 60 GB
5) Operation system require : window 2000, window XP

1.7.2 Details of the program.

1) Google APIs

Google Apps is a service from Google providing independently customizable versions of several Google products under a custom domain name.

2) Operating system Windows 7

Windows 7 is the latest release of Microsoft Windows, a series of operating systems produced by Microsoft for use on personal computers, including home and business desktops, laptops, net books, tablet PCs, and media center PCs.

Chapter 2

Background

We will use Google apps with other recent APIs to customize our web to provide good web that have function from Google apps making a better web. We briefly explain them in this chapter.

2.1 Google Apps Engine

Google Apps is a service from Google providing independently customizable versions of several Google products under a custom domain name. It features several web applications with similar functionality to traditional office suites, including: Gmail, Google Groups, Google Wave, Google Calendar, Talk, Docs and Sites.

Google App Engine lets you run your web applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: You just upload your application, and it's ready to serve your users.

Google App Engine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. App Engine also features a dedicated Python runtime environment, which includes a fast Python interpreter and the Python standard library. The Java and Python runtime environments are built to ensure that your application runs quickly, securely, and without interference from other apps on the system. Developer can serve an app from his own domain name (such as http://www.example.com/) using Google Apps or serving your app using a free name on the appspot.com domain. He can share his application with the world, or limit access to members of his organization. App Engine costs nothing to get started. All applications can use up to 500 MB of storage and enough CPU and bandwidth to support an efficient app serving around 5 million page views a month, absolutely free.

2.2 Google APIs

Google APIs is a web application that developed by Google .The site contains documentation on using Google developer tools and APIs including discussion groups and blogs for developers using Google's developer products

Google APIs provide interesting web application such as Google calendar which user can note thing to do on their calendar and share to other friend and Google friend contact which provide community to user that they can talk together. We will combine these tools to make appointment online system. The system users come to join community. The sociality system user come to make online appointment by powerful tool like Google APIs. It can make great sociality system.

2.2.1Overview APIs tool

Google Calendar APIs is a free time-management web application offered by Google. Users are required to have a Google Account in order to use the software.

Google Maps APIs is a web mapping service application and technology provided by Google that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit, and maps embedded on third-party websites via the Google Maps API. It offers street maps, a route planner for traveling by foot, car, or public transport and an urban business locator for numerous countries around the world.

Google Talk APIs is a freeware Windows web-based application for instant messaging and voice over internet protocol (VOIP) client offered by Google Inc. The first beta version of the program was released on August 24, 2005.

Google Friend Connect APIs is an online service by Google that allows users on the internet to connect with their friends on different websites. Google Friend Connect is an Open Social application offered by Google that started in May 2008. The main focus of Google Friend Connect is to simplify the connection between social and non-social websites and standardize the handling and presentation of social applications and content. It uses a blend of open standards, such as OpenID for sign in, Auth to control data, and OpenSocial for applications.

Google Friend Connect is free but requires approval of the website using it. It requires no knowledge of web programming and enables any website to offer social applications and content from Hi5, Orkut, Plaxo, MySpace, Google Talk, Netlog and other social networks.

In this web appointment social network will use Google Calendar APIs, Google map APIs, Google talk APIs and Google Friend connection APIs.

2.2.2 Google Calendar APIs

We will use this APIs for make appointment. It can share calendar file to other Google account that user want. In this system we will improve to make it share to other easier, more convenient function.

2.2.3 Google map APIs

We will use this APIs to set location where friend make appointment. It is not only show location but it also show how you travel there by walk, bus, car to make friend and member convenient to go to destination where they make appointment.

We will make system keep data of journey to user profile to help user estimate time to go to destination.

2.2.4 Google talk APIs

Use this to let user talk together. This will let user speak about their appointment.

2.2.5 Google Friend connect APIs

Use this to create sociality function such as "Members" that Let users join your site, create profiles, find other users, and invite friends to join and "Social bar" that Let users sign in, view other members, leave comments, and see recent site activity from the top or bottom of your website.

2.3 Google API protocol Basic

Google API protocol Basic explains and show example about how Google APIs get request and manage user's data, including information about what a query looks like, what results look like, and so on. This section describes the Google Data Protocol document format and query syntax.

2.3.1 Requesting a feed or other resource

There is a feed name /myFeed. To see it, send the following request to the server.

Request	Response
GET /myFeed	200 OK <?xml version="1.0"?> <feed xmlns="http://www.w3.org/2005/Atom"> <title>Foo</title> <updated>2006-01-23T16:25:00- 08:00</updated> <id>http://www.example.com/myFeed</id> <author> <name>Jo March</name> </author> <link href="/myFeed" rel="self"/> </feed>

Table 2.1 Requesting a feed.

2.3.2 Inserting a new entry

To create a new entry, send a POST request, and supply the XML representation of the new entry:

Request	Response
POST /myFeed <?xml version="1.0"?> <entry xmlns="http://www.w3.org/2005/Atom"> <author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author> <title type="text">Entry 1</title> <content type="text">This is my entry</content> </entry>	201 CREATED <?xml version="1.0"?> <entry xmlns="http://www.w3.org/2005/Atom"> <id>http://www.example.com/id/1</id> <link rel="edit" href="http://example.com/myFeed/1/1/"/> <updated>2006-01-23T16:26:03- 08:00</updated> <author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author> <title type="text">Entry 1</title> <content type="text">This is my entry</content> </entry>

Table 2.2 Inserting a new entry.

2.3.3 Updating an entry

To update an existing entry, use PUT, with the entry's edit URI (as provided by the server in the previous example, in the <link rel="edit"> element).

If firewall does not allow PUT, then do an HTTP POST and set the method override header as follows:

X-HTTP-Method-Override: PUT

In the following example, we're changing the entry's text from its old value ("This is my entry") to a new value ("This is my first entry."):

Request	**Response**
PUT /myFeed/1/1/ <?xml version="1.0"?> <entry xmlns="http://www.w3.org/2005/Atom"> <id>http://www.example.com/id/1</id> <link rel="edit" href="http://example.com/myFeed/1/1/"/> <updated>2006-01-23T16:28:05- 08:00</updated> <author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author> <title type="text">Entry 1</title> <content type="text">This is my first entry.</content> </entry>	200 OK <?xml version="1.0"?> <entry xmlns="http://www.w3.org/2005/Atom"> <id>http://www.example.com/id/1</id> <link rel="edit" href="http://example.com/myFeed/1/2/"/> <updated>2006-01-23T16:28:05- 08:00</updated> <author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author> <title type="text">Entry 1</title> <content type="text">This is my first entry.</content> </entry>

Table 2.3 Updating an entry.

To see the new entry in context, request the entire resource again:

Request	**Response**
GET /myFeed	200 OK <?xml version="1.0"?> <feed xmlns="http://www.w3.org/2005/Atom"> <title>Foo</title> <updated>2006-01-23T16:28:05-08:00</updated> <id>http://www.example.com/myFeed</id> <author> <name>Jo March</name> </author> <link href="/myFeed" rel="self"/> <entry> <id>http://www.example.com/id/1</id> <link rel="edit" href="http://example.com/myFeed/1/2/"/> <updated>2006-01-23T16:28:05-08:00</updated> <author> <name>Elizabeth Bennet</name> <email>liz@gmail.com</email> </author> <title type="text">Entry 1</title> <content type="text">This is my first entry.</content> </entry> </feed>

Table 2.4 Request the entire resource to show the result of update command.

2.3.4 Deleting an entry

To delete an existing entry, send a DELETE request, using the entry's edit URI If your firewall does not allow DELETE, then do an HTTP POST and set the method override header as follows:

X-HTTP-Method-Override: DELETE

The following example deletes an entry that shows the delete command:

Request -> DELETE /myFeed/1/2/ **Response** -> 200 OK

Do another GET to see that the feed now contains no entries:

Request	**Response**
GET /myFeed	200 OK <?xml version="1.0"?> <feed xmlns="http://www.w3.org/2005/Atom"> <title>Foo</title> <updated>2006-01-23T16:30:11- 08:00</updated> <id>http://www.example.com/myFeed</id> <author> <name>Jo March</name> </author> <link href="/myFeed" rel="self"/> </feed>

Table 2.5 Get data to check if the feed now contains no entries.

2.4 Facebook

Facebook is a social network service and website launched in February 2004 that is operated and privately owned by Facebook, Inc.

Users may create a personal profile, add other users as friends and exchange messages, including automatic notifications when they update their profile. Additionally, users may join common interest user groups, organized by workplace, school, or college, or other characteristics.

2.4.1 Facebook connect

Facebook Connect is a set of APIs from Facebook that enable Facebook members to log onto third-party websites, applications, mobile devices and gaming systems with their Facebook identity. While logged in, users can connect with friends via these mediums and post information and updates to their Facebook profile. Developers can use these services to help their users connect and share with their Facebook friends on and off of Facebook and increase engagement for their website or application.

2.5 Web service

Web services are typically application programming interfaces (API) or **Web APIs** that are accessed via Hypertext Transfer Protocol (HTTP) and executed on a remote system hosting the requested services. Web services tend to fall into one of two camps: big Web services and RESTful Web services.

The W3C defines a "Web service" as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Services Description Language WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards." Figure 2.1 explain how it works.

Object-relational mapping (ORM, O/RM, and O/R mapping) in computer software is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages

available that perform object-relational mapping, although some programmers opt to create their own ORM tools.

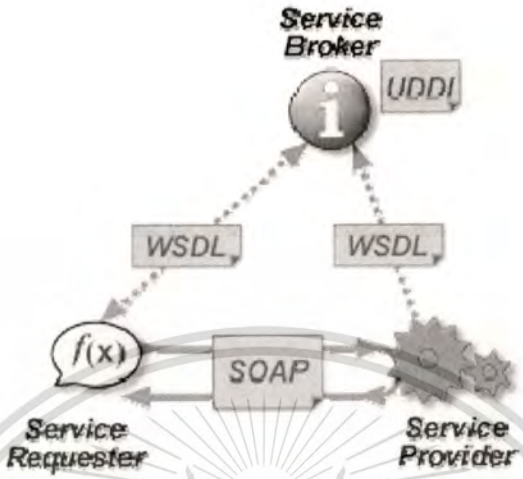


Figure2.1 How web service work.

2.6 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

2.7 DOM

The **Document Object Model (DOM)** is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Aspects of the DOM (such as its "Elements") may be addressed and manipulated within the syntax of the programming language in use. The public interface of a DOM is specified in its application programming interface (API).

2.8 CSS

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language. Its most

common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document, including plain XML, SVG and XUL.

2.9 PHP

PHP is a general-purpose scripting language originally designed for web development to produce dynamic web pages. For this purpose, PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document.

2.10 JavaScript

JavaScript, also known as ECMA Script, is a prototype-based, object-oriented scripting language that is dynamic, weakly typed and has first-class functions. It is also considered a functional programming language like Scheme and OCaml because it has closures and supports higher-order functions.

JavaScript is an implementation of the ECMA Script language standard and is primarily used in the form of client-side JavaScript, implemented as part of a web browser in order to provide enhanced user interfaces and dynamic websites. This enables programmatic access to computational objects within a host environment.

Chapter 3

System Design

This chapter will cover design that we have in this project. These design cover system's functional requirements, system analysis, and application design through our codes. The chapter ends by explaining writing codes using application interfaces.

3.1 A PROTOTYPE FOR GROUP CALENDAR ON GOOGLE APPS ENGINE

Nowadays there are a number of websites offering appointment and calendar service for individual. After we have studied these systems, we believe that some features could be improved. For example, sharing appointments in Google (http://google.com/calendar) requires its users to copy URL of these appointments or use the Iframe embedded that URL for share with friends. We like the WiKalenda interface (http://www.wikalenda.com) but it does not allow users to create their own appointments because its service aims to promote commercial events. With the motivation mentioned, we propose an appointment system that allows users to share their appointments among friends so that the offer better collaboration in this particular social network era.

3.1.1 Functional Requirement

Primary functions that our system would apply to the user Google calendar are the following.

- Retrieving user calendar
- Displaying events details occurring to the user.
- Creating an event with the following attributes - name of event, start time, end time, etc

- Synchronizing the edited event
- Being able to delete an event
- Inviting participants to an event via friends' Google account
- Providing common free time on the calendar for making an appointment among friends
- Managing friends into groups
- Reminding the user and participants involved in an appointment
- Providing a live chat feature among users
- Publishing an appointment on user's facebook wall
- Supporting auxiliary tools to an appointment such as attaching Google map and YouTube video

3.1.2 User case diagram

The analysis of the data in section 3.1.1 provides developers with the design, Use case diagram (as shown in figure 3.1), which consists of the User and Google and Facebook account. Therefore the system is a, self-run, autonomous system. Obviously we need to create our own login page (label no. 1). Our system will manage event via maintain event (label no. 2) directly using APIs mentioned in previous chapter, similar implementation for delete calendar and publish to Facebook(label no.3 and 5 respectively). We also need to implement the union function (label no.4) as there is no such feature offered. We describe the structure of the five modules here so that it will be much easier when explaining the APIs sequence diagrams.

A. Authentication procedure

A.1) First time user will have to select his own Google Account to sign up for AuthToken for access to the system.

A.2) After receiving AuthToken system will request Google to send AuthToken to user to get the right to accessing calendar data.

A.3) Google sends AuthToken to user.

A.4) The user can now access his calendar in the web site.

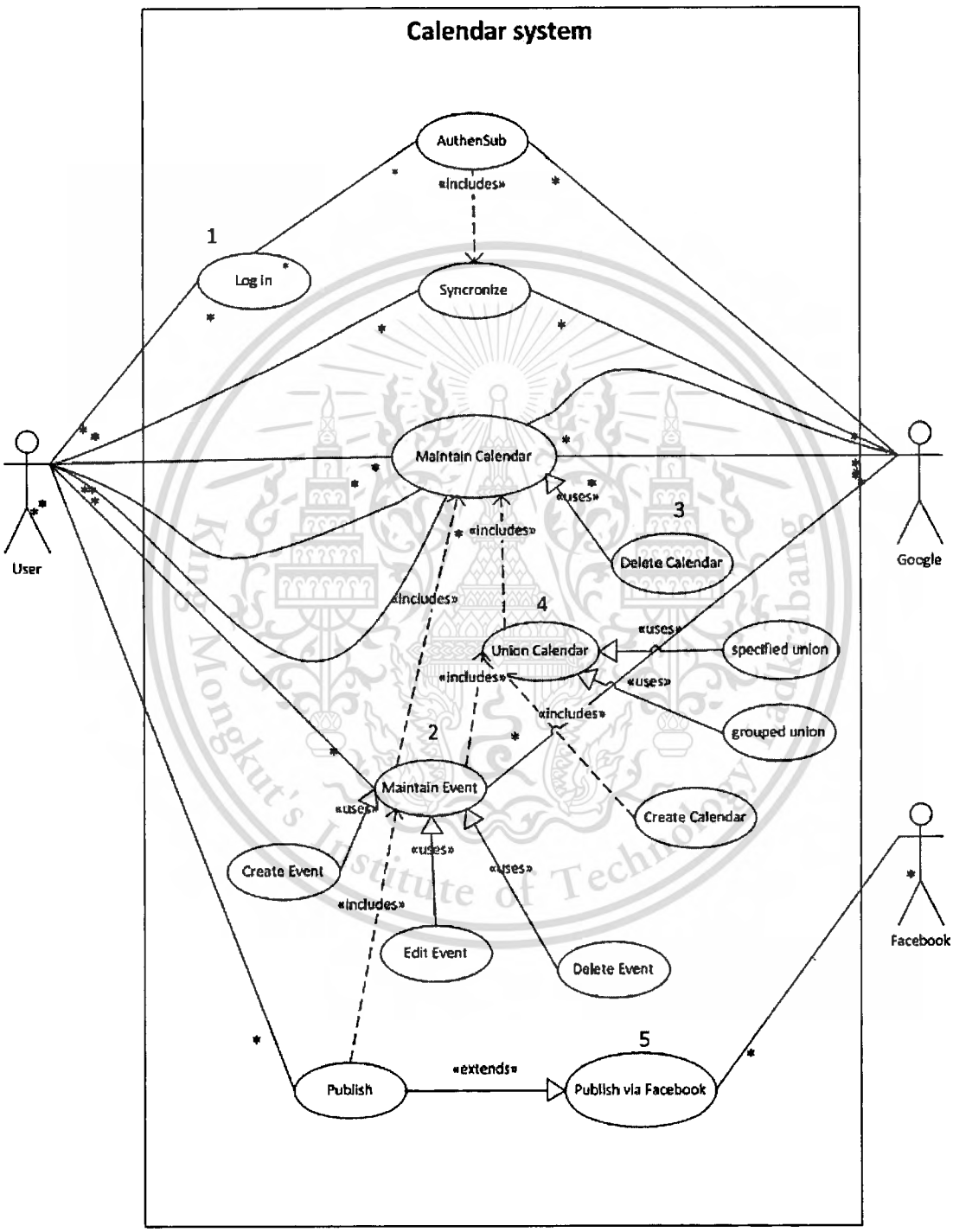


Figure 3.1 System Use case diagram.

B. Event management procedure

B.1) Users can manage and edit the calendar that user has created. To manage event users need to manage the calendar at least one calendar Because of the need to create an event to the calendar to make such activities more. In addition, users can also edit the event that is created by user, delete events have and update the status of participation of users and manage calendar and all activities that occur.

B.2) The website then sends request and command for user data to Google calendar. Google will check request and command to execute with user data.

C. Delete calendar procedure

C.1) User can delete union calendar he would not like to have it by choosing calendar in website then click delete calendar.

C.2) The website then sends request and command for user data to Google calendar. Google will check request and command to execute with user data.

D. Union calendar procedure

D.1) User can union his calendar with friend calendar that he already requested friend to get the right to access friend calendar. As soon as he unions calendar with friend's calendar the website will send command to Google.

D.2) Google receives command from website. It execute command by creating calendar for receiving data from user and user s' friend. When Google receives all data it will fill all event data to calendar that system command Google to create calendar for filling these event.

E. Facebook publish procedure

E.1) The user choose event that he would like to post to Facebook. He chooses then push Facebook button to publish his event calendar to Facebook website.

E.2) Facebook receives command and user's event calendar from this website then executes command to post event to Facebook.

3.2 API examples

Prior to showing you part of import code, we explain the logic of each procedure to show how Google APIs executes in system through each API's sequence diagram. They are tools from Google APIs that we use in this application. Understanding them is vital to successfully completing our application. The developer also modified Calvis system function to make it work more productively and to make whole system work properly.

The figure 3.2 explains how system acquires authentication from Google Calendar API to get the right to use user's calendar data and how Calvis gets user's calendar information to display events on website. The user will set event and calendar id to calvisMonthview to set calvisMonthview to display calendar after it get authentication token from Google Calendar API to get the right to access calendar data.

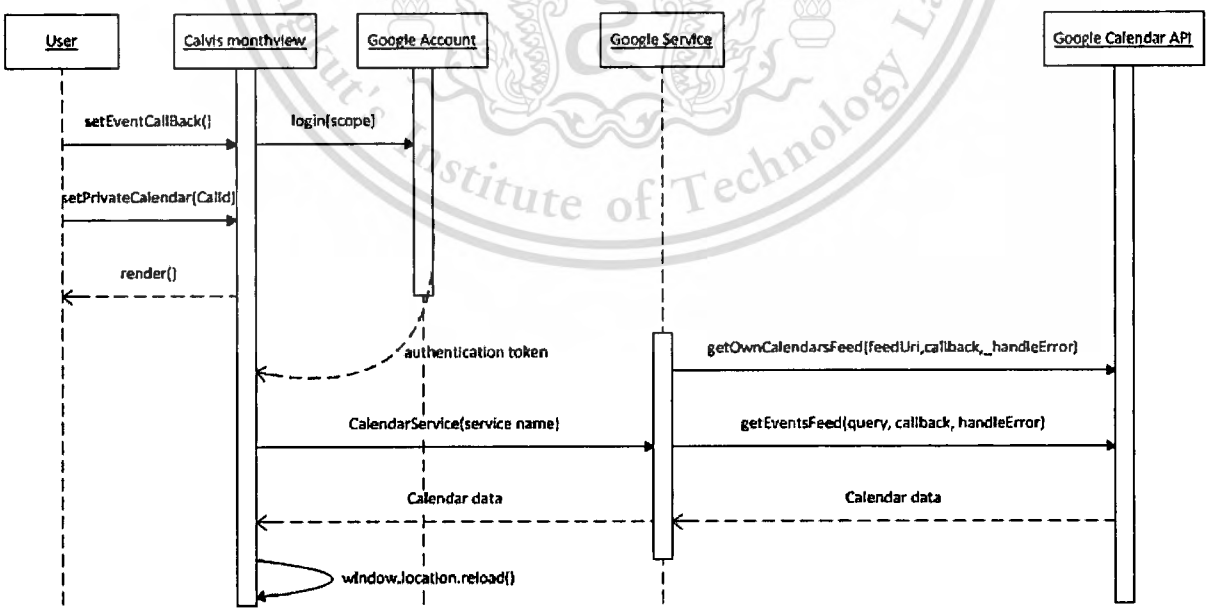


Figure 3.2 The procedure of Authentication from Google calendar.

The calvisMonthview receives calendar id then login to Google account to get authentication token, when it get it user will be able to use calendar data from his' s account. Therefore it sends service to Google Service to request for user's calendar data to display in it. It will then get calendar data to display calendar data by render () function.

The figure 3.3 explains how system create user's calendar event. After user gets authentication token from Google Account, he will now have the right to use his calendar data. He can create event by filling data on web, when he click add event, calvis will send service to Google Service to request Google Calendar API to manage calendar data that user order system to do.

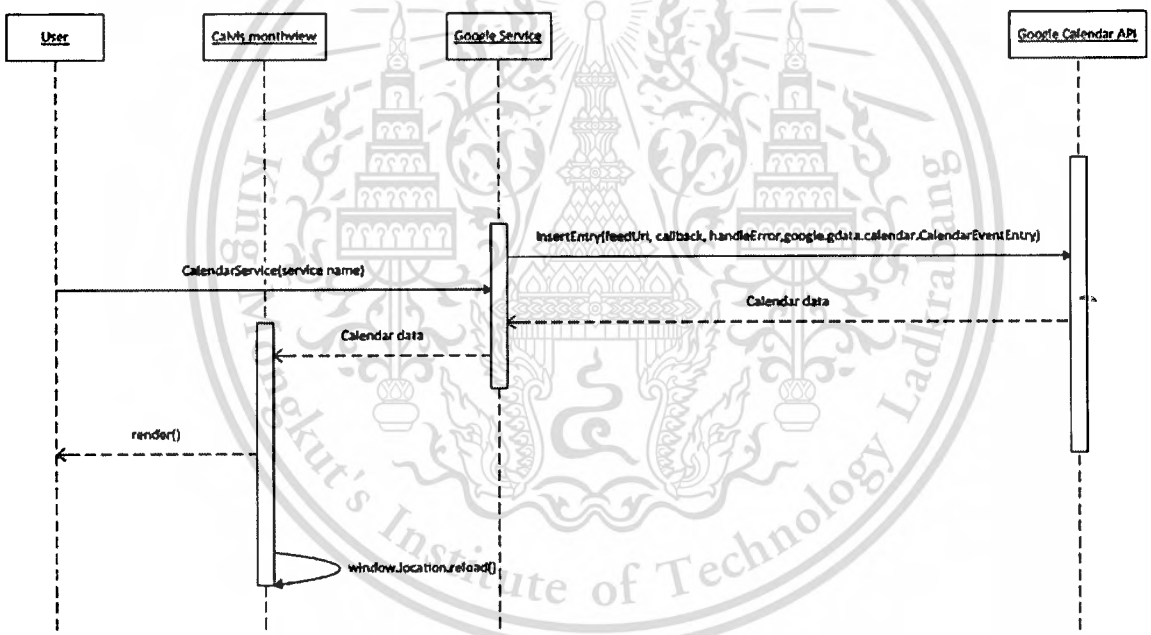


Figure 3.3 The procedure of creating event in user's calendar.

The figure 3.4 explains how system edit user's calendar event. After user gets authentication token from Google_ Account, he will now have the right to use his calendar data. He can edit event by filling data on web, when he click edit event, Calvis will send service to Google Service to request Google Calendar API to manage calendar data that user order system to do.

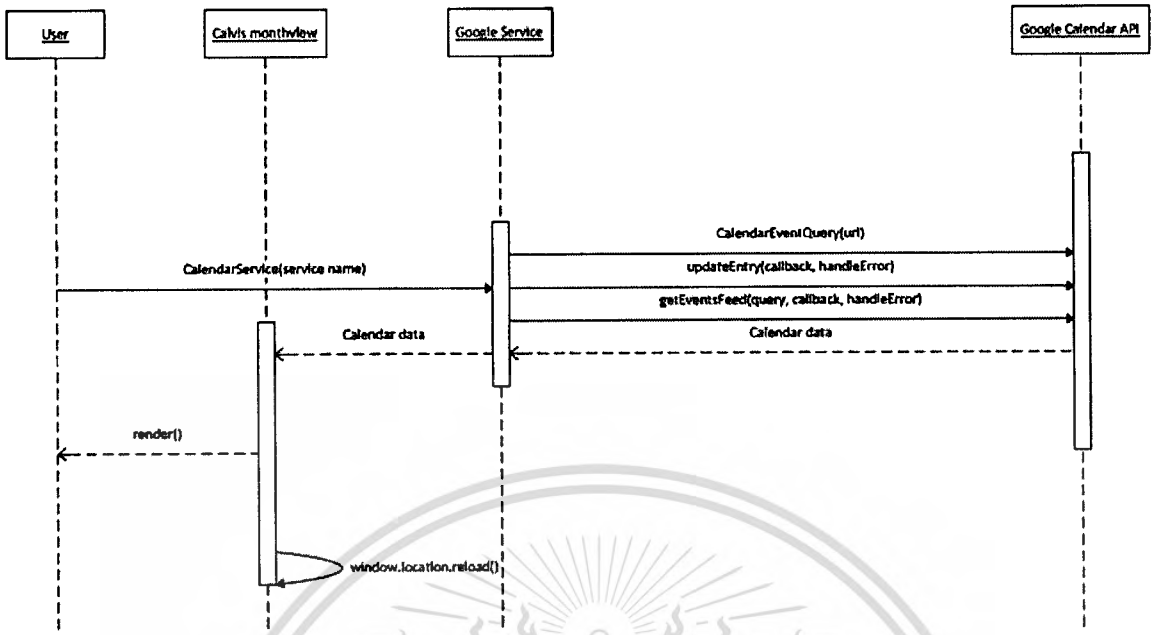


Figure 3.4 The procedure of editing event in user's calendar.

The figure 3.5 explains how system delete user's calendar event .After user gets authentication token from Google Account. He has now the right to use his calendar data. He can delete event by filling data on web, when he click delete, Calvis will send service to Google Service to request Google Calendar API to manage calendar data that user order system to do.

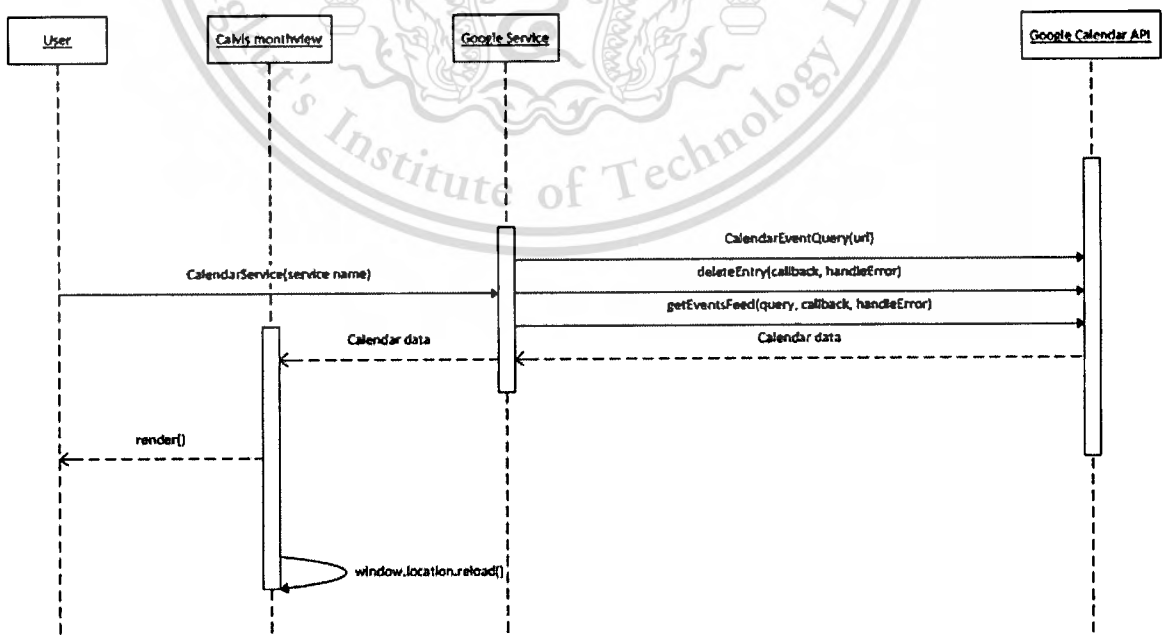


Figure 3.5 The procedure of deleting event in user's calendar.

The figure 3.6 explains how system union user's calendar and friend calendar. The user must invite friend to let Google Calendar send request to have right to access friend calendar. His friend will then get request from his mail. He then decides whether grant him his right to access calendar. If user gets the right to access friend calendar, he can view friend calendar and the friend calendar will be list in the user calendar list.

When user uses union calendar function, the system will create new calendar by sending service to Google Calendar API then user's calendar and friend calendar will be filled in this new calendar. After user's calendar account get new calendar, the Calvis will load all user's calendar to the system.

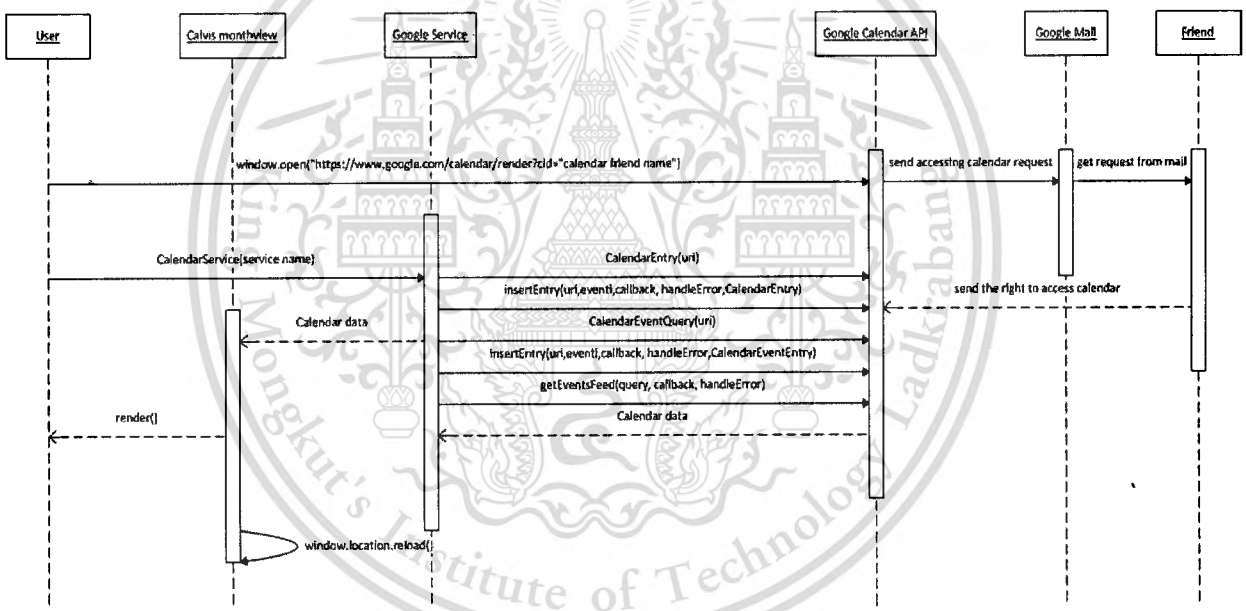


Figure 3.6 The union calendar procedure.

The figure 3.7 shows how system delete user calendar. The user will choose calendar to delete then the system will send service to Google Calendar to request it to delete calendar. To request it the system uses getOwnCalendarsFeed(uri,callback,_handleError),which uses uri attribute to specify user's calendar feed, to specify calendar that user want to delete and use deleteEntry(callback,_handleError) to delete calendar.

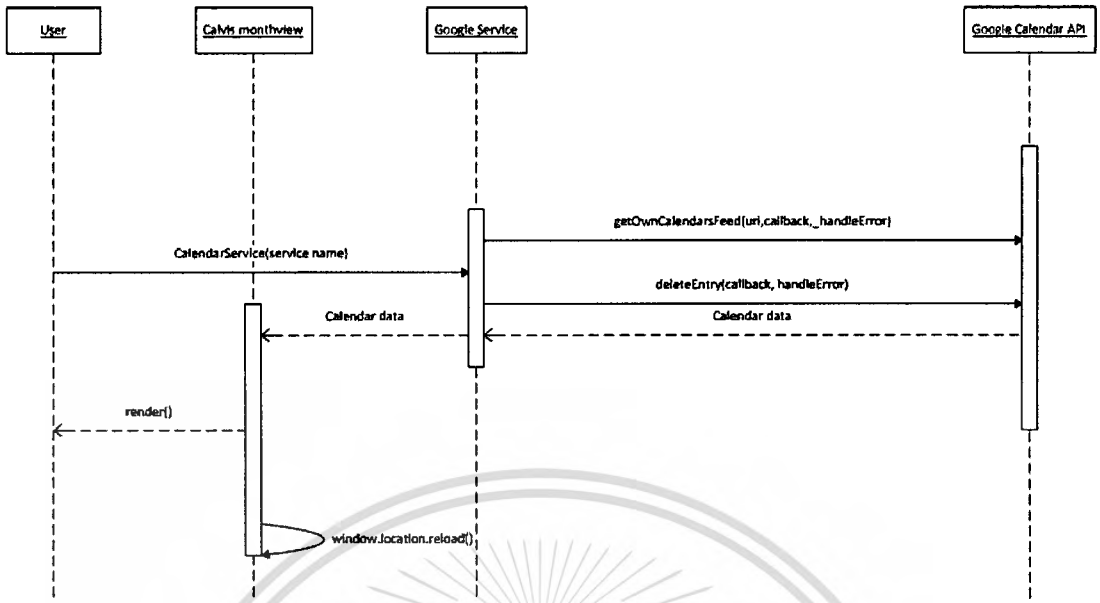


Figure 3.7 The procedure of delete calendar.

To present the system features, the following are activities is created.

1. Google calendar owner activates from his Google calendar page for allowing the application to retrieve his calendar after he logs in to the system.
2. In order to show our system feature, the user makes a request to his friends for permissions to retrieve the friends' calendars, if his friends have never been added to any group. The grants are completed via confirmation emails sent to his friend's.
3. Assign the friend to a group.
4. User can now see the union of his calendar and his friends in the new calendar that the system provides.
5. In case of adding an event for sharing an appointment, the process follows regular. Google's event
   - The user creates an event and asks his friends to join the event. Upon his acceptation, he becomes a participant. The event is included to his calendar automatically to let friend can make appointment together.

3.3 Coding

In this section we explain part of our code which is part of our contribution to this project.

3.3.1 Log in and log out function

Function in figure 3.7 is used for authenticating a user can login into the system. When user log-in into system Google then show log in page to log in into user's Google account. Google also ask user if he would like this website accessing his calendar data. After he accept to let website access his calendar the Google then sends Authtoken to allow website to manage Calendar data for user. Figure 3.8 shows the function used to log-out to the system

```
1  function login()
2  {
3          // authenticate the user through Google Authsub
4          if ( ! google.accounts.user.checkLogin("http://www.google.com/calendar/feeds/")) {
5                  if (google.accounts.user.login("http://www.google.com/calendar/feeds/"))
6                  {}
7          }
8  }
```

Figure 3.8 Function log in.

```
1  function logout()
2  {
3  google.accounts.user.logout();
4  document.getElementById("btnLogin").disabled = false;
5                  document.getElementById("btnClearCookies").disabled = true;
6                  window.location.reload();
7  }
```

Figure 3.9 Function log out.

This code, in figure 3.10, is most important because the system have to load function from Google API. The program will use function that load from above to use to manage calendar data for user. This code loads Google calendar API function that most function in the website need.

```
1  function loaded(){}
2  // load gdata, just calendar package
3  google.load('gdata', '2', {packages:['calendar']});
4  google.setOnLoadCallback(loaded);
```

Figure 3.10 Load GData to use code from Google calendar API.

The program keeps calendar id of the last user who logged in to the system. There is no time out. Thus, in any case whether the user want to change the login id or he has not logged in to the system, the program simply load Google login page shown in figure 3.11.

```
1  if ( ! google.accounts.user.checkLogin("http://www.google.com/calendar/feeds/")) {
2  if (google.accounts.user.login("http://www.google.com/calendar/feeds/")){}
```

Figure 3.11 Loading Google login page.

Once the calendar id is obtained, Google calendar visualization module calvis, in line 1 below, handles displaying calendar on the.screen. Date components as well as monthly view or weekly view are provided. Simply customize the calendar style to make the program harmoniously integrated to our application. Calvis is also responsible for displaying the event detail dialog.

```
1  var calendar = new calvis.Calendar();
2   // set the CSS IDs for various visual components for the calendar container
3   calendar.setCalendarBody('calendarBodyDiv');
4   calendar.setNavControl('navControlDiv');
5   calendar.setViewControl('viewControlDiv');
6   calendar.setStatusControl('statusControlDiv');
7   calendar.setEventCallback('click', displayEvent);
8  calendar.setPrivateCalendar(calId);
9   calendar.setDefaultView('month');
10  // display the calendar
11  calendar.render();
```

Figure 3.12 Displaying the user's calendar.

3.3.2 Request a friend function

The application must make a request to his friends for permission to retrieve the friend's calendar. Line 2 of the below code in figure 3.12, loads our Google calendar with the pop-up dialog for the user to send the request. The program keeps requesting the invitee's calendar. Only after the invitee granting, the program may successfully retrieve the invitee's calendar.

```
1  var r = document.getElementById('createCalendarLink');
2  window.open("https://www.google.com/calendar/render?cid="+r.value);
```

Figure 3.13 Requesting a friend's calendar.

3.3.3 Assign the friend to a group function

The insertion is done regardless whether the invitee grants the user's request or not, shown in line 2. The only limitation to deploying our application of Google app engine is that we have not found any technique to deploy a database on it – not even a text file. Therefore we need to connect to a dedicate database for now.

```
1
2  $sql="INSERT INTO friend (username, email, cal_ID, groupID)
3  VALUES('$_POST[hiddenusername]','$_POST[hiddencalID]','$_POST[addFriendGroup]','$_POST[grou
4  pID]')";
```

Figure 3.14 Assigning a friend id to a group.

3.3.4 Union user's calendar and his friends'

When the user executes the calendar union, he could union his calendar with all members in the group or union his calendar to a subset members of the group. Both operations share the same program logic. The group union is presented here. Basically, there are two ways to implement this operation. We could empty the user's calendar or we could create new calendar every time the request is made. We choose the later technique. We believe that disposing the calendar after viewing is done is simpler than keeping track calendar id of each possible union we might generate. Calendar Union Concept is present as followed.

1. In Google calendar, a user may have any number of (auto generated id) calendars. We exploit the one-user-many-calendar-fact and create a new calendar for each group.

2. We construct a union calendar of all participants by iteratively adding each event from each friend's calendar and self.

3. For any change, we reset the calendar and refreshing the calendar by performing union method again.

Figure 3.15 explains the above logic. Line 53 shows how calendar is created. The component in line 57 controls how many calendar the program needs to process. The insert_eventGroup1 function in line 58 keeps parsing each event belonged to a calendar. By calling insertEntry method, line 65, an extracted event contained in evt2 is inserted into the user's calendar URL. The function works with xml format. Once all events are processed, the calendar being displayed is then automatically updated.

```
51          function joinGroupedCalendar(){
52          createNewGroupedCalendar ('http://www.google.com/calendar/feeds/default/owncalendars/full',
53          function (strCreatedCalURI) {
54                  insert_eventGroup1('http://www.google.com/calendar/feeds/default/private/full',
55          strCreatedCalURI , username);
56          var joinGroupCalendar = document.getElementById('unioncalendar');
57          for(j=0;j<joinGroupCalendar.options.length;j++)  {
58      insert_eventGroup1('http://www.google.com/calendar/feeds/'+joinGroupCalendar.options[j].value+
59    '/private/full', strCreatedCalURI, joinGroupCalendar.options[j].text);    } });    }
60          function insert_eventGroup1(uri, calendarID, importedCalName){
61          _service.getEventsFeed(myQuery,  function (root) {
62                  if (root.feed) {
63                  var entries = root.feed.getEntries();  for ( r in entries ){ var entry = entries[r];
64                  ...
65          _service.insertEntry('http://www.google.com/calendar/feeds/' + calendarID + '/private/full', evt2,
66  function(entry){}, _handleError,  google.gdata.calendar.CalendarEventEntry);}   } }, _handleError);
67  }
```

Figure 3.15 Union calendars

3.3.5 Event management and calendar management functions

Prior to showing the code for managing events, we show the structure of a Google event. A Google event is composed of the following tags.

Attribute tag

- Title – Heading of event.
- Date: Start Time - It is initial event, users can modify the date and time by clicking the button.
- Date: End Time - It is end of the event, users can modify the date and time by clicking the button.
- Content - Description of the event the user can add details to the event.
- Location - Locate the birthplace of the event.
- Participant - Identify participants.
- Permissions management event - As the authorization to manage event for participants are as follows.
    - Modify event - Participants can edit the event.
    - Invite other - Participants can invite others participate this activity.
    - See guest list - Participants can see a list of all participants.
- Reminder - Notification program will alert users to the event prior to the event as a user-defined time.

We start with creating an event. The process follows regular Google's event by calling create event entry in line 6 and insert it to the calendar in line 12, in figure 3.16. The evt2 is in

JSON format. The user creates an event and may ask his friends to join the event. Upon his friend's acceptation, **he becomes** a participant. The event is updated to his calendar automatically.

```
1
2   function testInsert() {
3           // fetch properties
4           var _service = new google.gdata.calendar.CalendarService('gdataservice');
5           var addParticipant = document.getElementById('addParticipant');
6           var evt2 = new google.gdata.calendar.CalendarEventEntry({ ...});
7                   ...
8                   if(addParticipant.value!='')evt2.addParticipant(new google.gdata.Who({
9                                                   rel: google.gdata.Who.REL_MESSAGE_BCC,
10                                                  email: addParticipant.value  }) );
11                  ...
12  _service.insertEntry('http://www.google.com/calendar/feeds/default/private/full', evt2,
13  function(entry){window.location.reload();},_handleError,
14  google.gdata.calendar.CalendarEventEntry);
15          return false;
16  }
```

Figure 3.16 Create event

This function is used to create event to user's calendar. It has to use service variable to send request and command to manage user's data. This function also receive event parameter from user to create event with parameter that user give it ,then set into variable named evt2 and add this event by function insertEntry(). This function is used to insert any Google calendar entry, such as calendar entry and calendar event entry, into Google calendar user's account. It uses evt2 that is event that system is going to insert , function callback that is function that surely work after insertEntry() function finish its task, function _handleError to check error , and entry variable to set which entry developer would like to insert.

The edit event function in figure 3.17 is used to edit event to user's calendar. It has to use service variable to send request and command to manage user's data. User can edit event that he chooses. This function also receive event parameter from user to create event with parameter that

user give it ,then set into variable named evt2 and edit this event by function updateEntry(). This function is used to edit any Google calendar entry, such as calendar entry and calendar event entry, into Google calendar user's account. It uses evt2 that is event that system is going to edit , function callback that is function that surely work after updateEntry() function finish its task, function _handleError to check error , and entry variable to set which entry developer would like to edit. The callback function in line 12 notifies its caller for update completion.

```
1  function go_findedit(value){
2          var _service = new google.gdata.calendar.CalendarService('gdataservice');
3          // authenticate the user through Google Authsub
4          var _scope = "http://www.google.com/calendar/feeds/";
5  ...
6                  _find('http://www.google.com/calendar/feeds/default/private/full',
7                          function(entry){
8                                  if ( entry ){
9                                          if(entry.getId().getValue() == value){...}
10         entry.updateEntry(
11         function()
12                 {alert('event edited'); window.location.reload();}
13                         , _handleError);
14                                  }
15                                  } });                    }
```

Figure 3.17 edit event to calendar user

The delete event function is shown in figure 3.18. It has to use service variable to send request and command to manage user's data. User can delete event that he chooses. It uses evt2 that is event that system is going to delete , function callback that is function that surely work after deleteEntry() function finish its task, function _handleError to check error , and entry variable to set which entry developer would like to delete.

```
1  function go_finddelete(value)          {
2          var _service = new google.gdata.calendar.CalendarService('gdataservice');
3          // authenticate the user through Google Authsub
```

```
4           var _scope = "http://www.google.com/calendar/feeds/";
5                   _find('http://www.google.com/calendar/feeds/default/private/full',
6                           function(entry){
7                                   if ( entry ) {
8                                           // submit changes
9                                           if(entry.getId().getValue() == value)
10                                          entry.deleteEntry(function(){
11                                              alert('event deleted'); window.location.reload();},
12                                          _handleError);
13                                  }
14                          });
15  }
```

Figure 3.18 delete event to calendar user

The delete calendar function is shown in figure 3.19. It deletes by receiving calendar name that user choose. If user choose and click delete it, it will delete calendar by finding calendar in the user's account then delete by deleteEntry () function.

```
1   function _handleError(e){}
2   function deleteCalendar()   {
3           var deleteCalList = document.getElementById('deleteCalList');
4           var _service = new google.gdata.calendar.CalendarService('gdataservice');
5       service.getOwnCalendarsFeed('http://www.google.com/calendar/feeds/default/owncalendars/full',
6                   function(feedRoot)  {
7                           var entries = feedRoot.feed.getEntries();
8                           for(r in entries)  {
9           var entry = entries[r];varcal_title = entry.getTitle().getText();  if(cal_title ==
10  deleteCalList.value)
11                  entry.deleteEntry( function(entry){window.location.reload();}, _handleError);
12                          }   },_handleError);
13  }
```

Figure 3.19 Function delete calendar

3.3.6 Social network function example

This function in figure 3.20 is used for Facebook function. In this time the program uses only window.fbAsyncInit = function () {...} to configure important value and parameter and function showStream (event) to publish event that user choose to Facebook wall. The function showStream (event) receives event that user choose then post to Facebook by streamPublish () function.

```
1  window.fbAsyncInit = function() {
2  FB.init({appId: '135997223138356', status: true, cookie: true, xfbml: true});
3              /* All the events registered */
4  FB.Event.subscribe('auth.login', function(response) {    login();    });
5  FB.Event.subscribe('auth.logout', function(response) {logout();});
6  FB.getLoginStatus(function(response) {  if (response.session) login();});};
7          //stream publish method
8          function streamPublish(name, description, hrefTitle, hrefLink, userPrompt){
9  FB.ui({
10                 method: 'stream.publish',  message: '',
11                 attachment: {name: name, caption: '',    description: (description),   href: hrefLink   },
12 action_links: [   { text: hrefTitle, href: hrefLink }  ],
13 user_prompt_message: userPrompt},
14                 function(response) {});
15         }
16         function showStream(event){
17 FB.api('/me', function(response) {
18 streamPublish(response.name, 'Title: '+event.getTitle().getText()+'\n'+
19 'Date:'+event.getTitle().getText()+'\n'+'Description:'+event.getContent().getText()+'\n'+
20 'Where:'+event.getLocations()[0].getValueString()+'\n'+'Participant:'+
21 event.getParticipants()[0].  getEmail()+'\n', 'hrefTitle', 'http://thinkdiff.net', "Share thinkdiff.net"  );  });
22         }
23         function fqlQuery(){
24 FB.api('/me', function(response) {
25 var query = FB.Data.query('select name, hometown_location, sex, pic_square from user where uid={0}',
26 response.id);
```

```
27  query.wait(function(rows) {
28  document.getElementById('name').innerHTML =
29   'Your name: ' + rows[0].name + "<br />" + '<imgsrc="' + rows[0].pic_square + '" alt="" />' + "<br />";
30  });          });
31        }
```

Figure 3.20 Facebook function

3.4 User interface requirement

Below information summarizes our design the application user interfaces. We created six tabs for operating our application with the default view space on the left hand side of the screen. The important script files involved for each section is also listed.

Workspace/Tab name	Features	Script file involved
Calendar	Show calendar event, edit and delete calendar event	calvis.js, calvisLoad.js, calvis-core.js, Dialog.js
View	Use to choose calendar in list	calvisLoad.js
Event	Use to create new calendar in user's calendar	createEditDeleteEvent.js
Union	Union group and specific union	joinCalendar.js
Request	Request friend	inviteFriend.js
Assign	Manage group	insert.php
Misc.	Delete union calendar	deleteCalendar.js

Table 3.1 User Inteface details

Chapter 4

Implementation

In this special project we create web application "APPOINTMENT SYSTEM ON GOOGLE APPS" This chapter will guide you how to use the system.

4.1 Overview

Developing appointment system on Google App Engine is program used service of Google calendar and applied together with social network in develop appointment system by people who are interested can activated login Gmail account for permission this site and retrieve event information from Google calendar. The program will display user's appointment such as Title Description, time, place and participant. User can create edit event invite friend to share event that user can specify the location or place by Google map. As well as users can select the activities required to publish such activities out

To interact with social network such as Facebook, the activity publishing is displayed in the user's wall on Facebook. The detail information published includes details of the event, date the start of activities and the place of events.

To present the system features, the following are activities is created.

6. Google calendar owner activate from his Google calendar page for allowing the application to retrieve his calendar.
7. The application must make a request to his friends for permission to retrieve the friend's calendar - The grant is done via confirmation email to the friend's email address.
8. Assign the friend to a group.

9. User can now see the union of his calendar and his friends.
10. In case of adding an event for sharing an appointment, the process follows regular. Google's event
    - The user creates an event and asks his friends to join the event. Upon his acceptation, he becomes a participant. The event is included to his calendar automatically.
11. Other features are implemented independently

4.2 User Interface

4.2.1 Login and default page

In access the service and manage the activities of the appointment system. Latest, Gmail login is placed in the drop down component as shown in figure 4.1. The select button is for retrieving the user's calendar from Google. If the user id is not available in the component, the user is required to log in first via the login button below the default view area.

Figure 4.1 The login page.

As mentioned above, the user is required to log in to his Gmail account, shown in figure 4.2. Figure 4.3 shows the granting page from Google for allowing our application to retrieve the user's calendar.

Figure 4.2 The login used Gmail account.

Figure 4.3 illustrate account permission to Google Calendar for user grant our website.

After logging in, the acquired calendar data and activities is shown by the default page is view tab. The view tab default layout is a month view, as shown in figure 4.4. In the case that is a week view is selected, as shown in figure 4.5. Both views display events in seven columns representing each day in a week. At the above right of the calendar is a control to switch the view mentioned while at the above left of the calendar is a control for changing date.

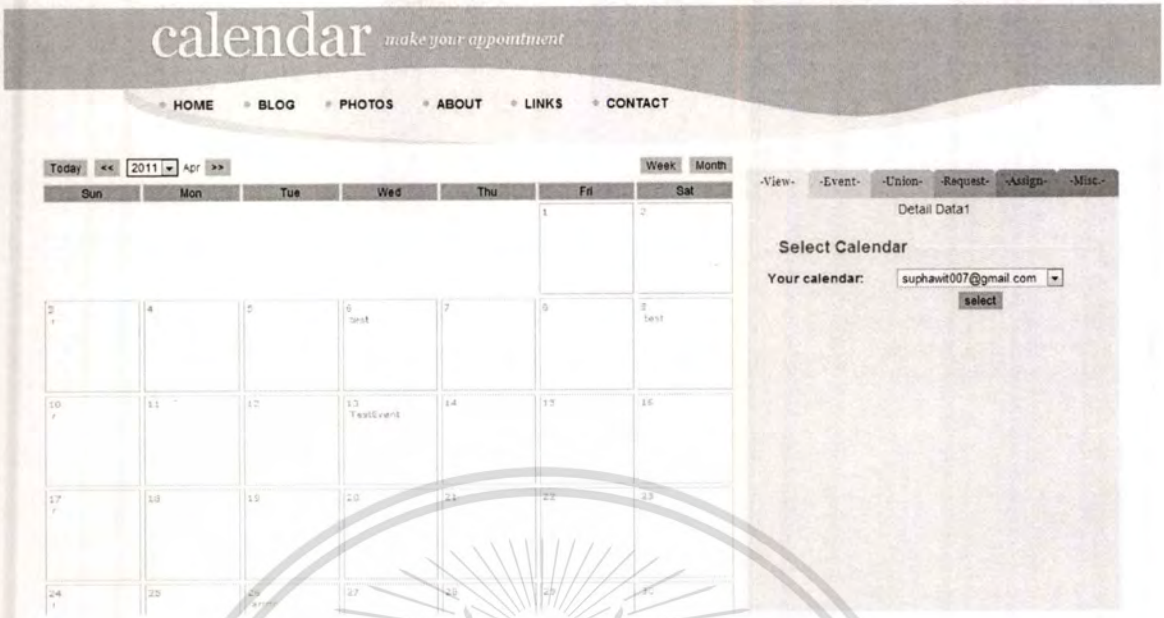


Figure 4.4 this screen will show all user' event in month view.

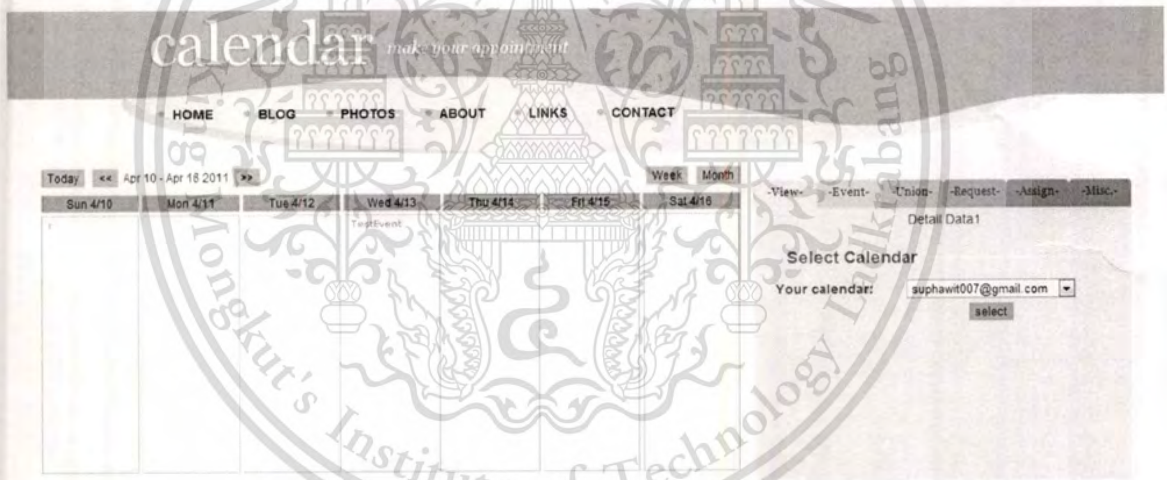


Figure 4.5 this screen will show all user' event in week view.

A user may click to select the event to view the event details. The event window, shown in figure 4.6, contains the following information events title, date, and start time, end time, place of the event, participants, and participant's right and other information. Three operations are provided at the end of the event details –namely published to Facebook, edit the event and delete the event. There is a map for the place of the event and related YouTube video embedded.

Figure 4.6 this screen will show all detail of each event.

4.2.2 Request Friends

To view a user's friend's calendar, he must be granted by his friend. Supply the friend's email in request tab, shown in figure 4.7, for sending the request. The confirmation on sending the request provided by API is shown in figure 4.8 and 4.9.

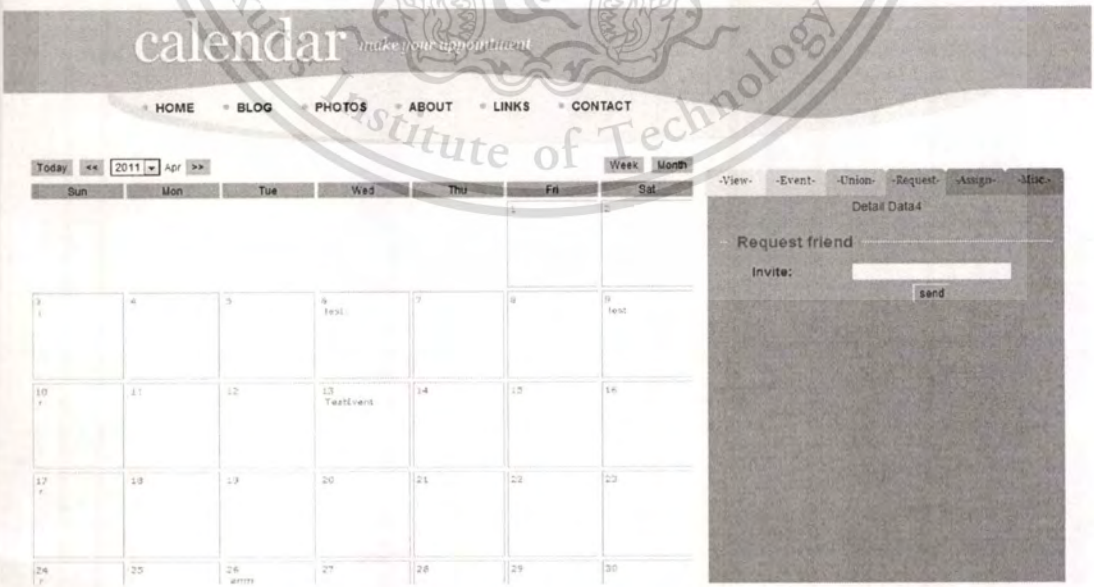
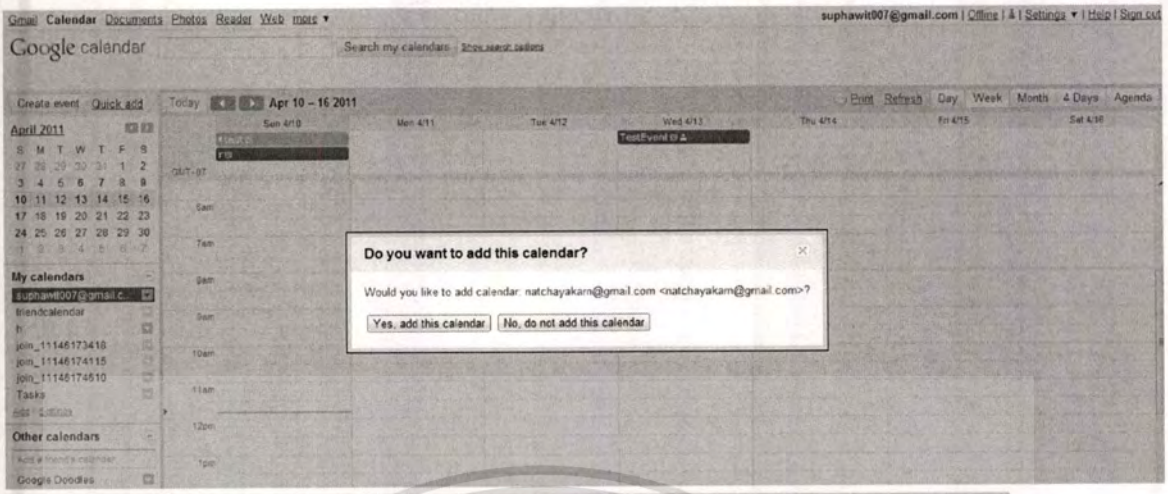


Figure 4.7 This screen show tab request friend.



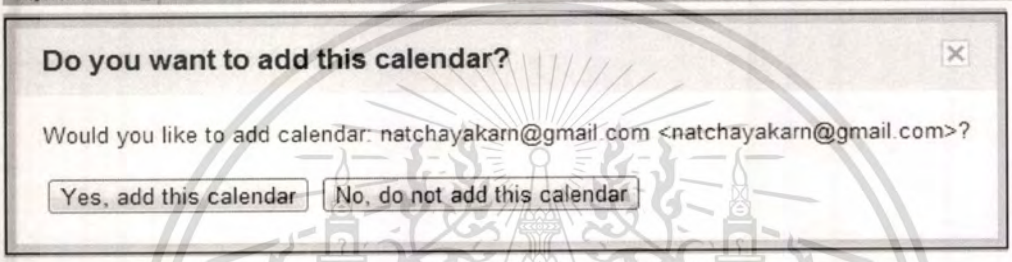


Figure 4.8 Confirmation dialog prior to sending request to friend.

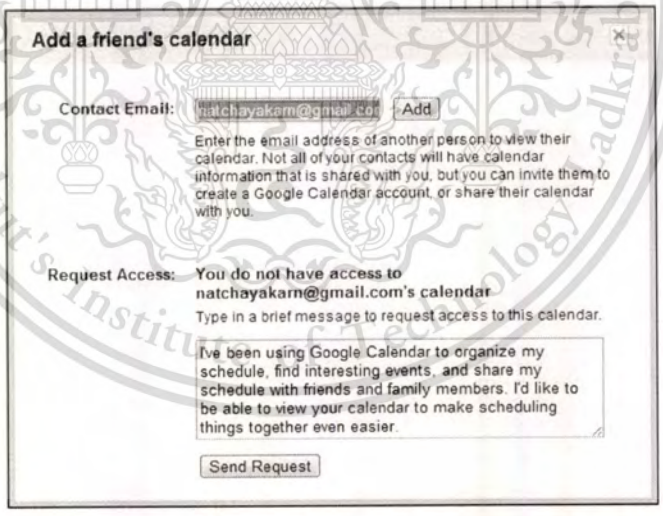


Figure 4.9 Confirmation dialog for sending request to friend.

4.2.3 Assign a friend to a group

For viewing multiple calendars in one screen, one of the main objectives of our application, the user can assign a group to a friend in assign tab, shown in figure 4.10. To perform the task, simply supply the friend's email and the group name. Recall that calendar owner can be belonged to more than one group. Figure 4.11 shows calendars the application recognizes so that the user can assign to group. The user name is for displaying which principle calendar is.

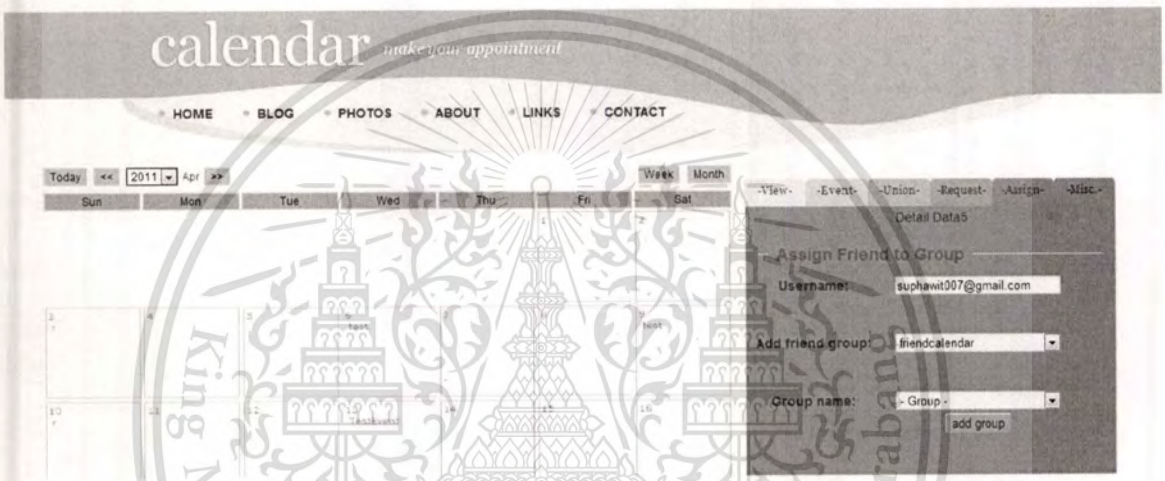


Figure 4.10 This screen show assign friend to group.

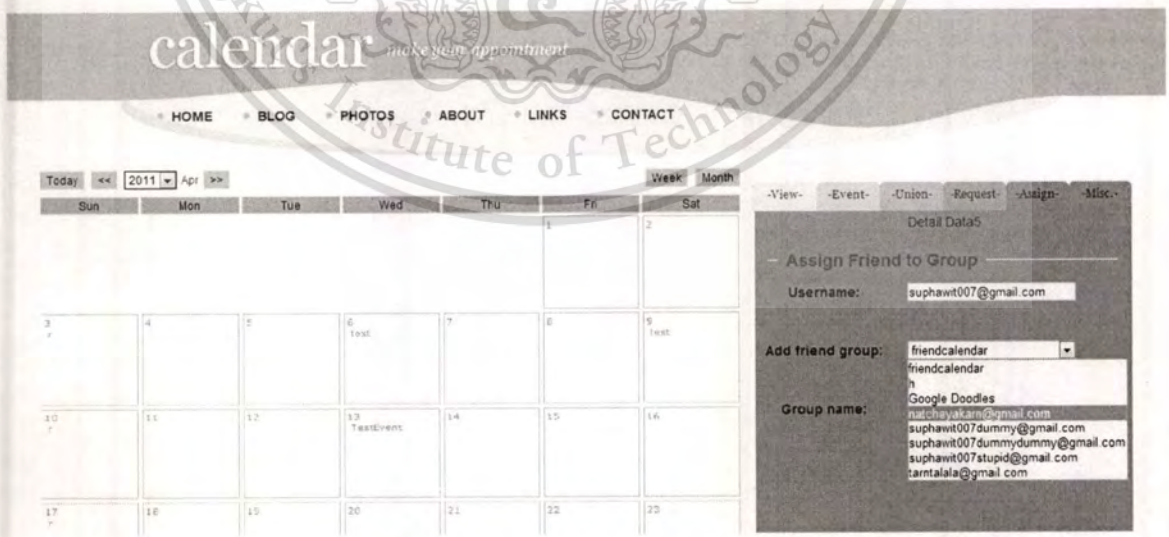


Figure 4.11 This figure show list of calendar 'friend in group.

4.2.4 Display the union calendar

Union all member calendars allow us to see the common free time. Figure 4.12 shows selecting a group for union and click submit. User may view the union result shown in figure 4.14 in the default view tab.

When users want to view the list friend in the group, the user can see it in union calendar component shown in figure 4.13.

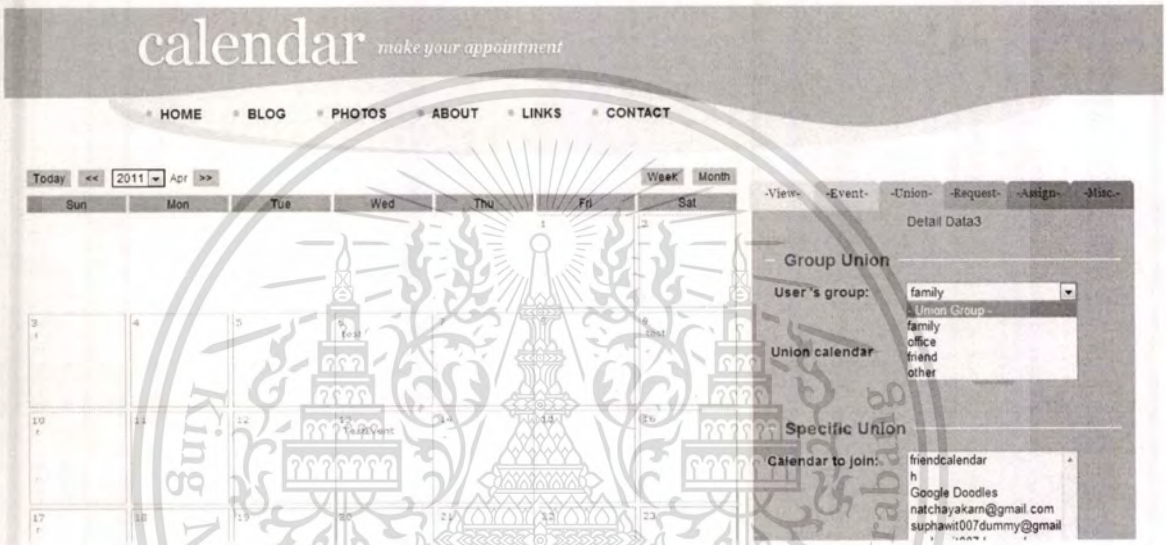


Figure 4.12 This screen show list of all group name.

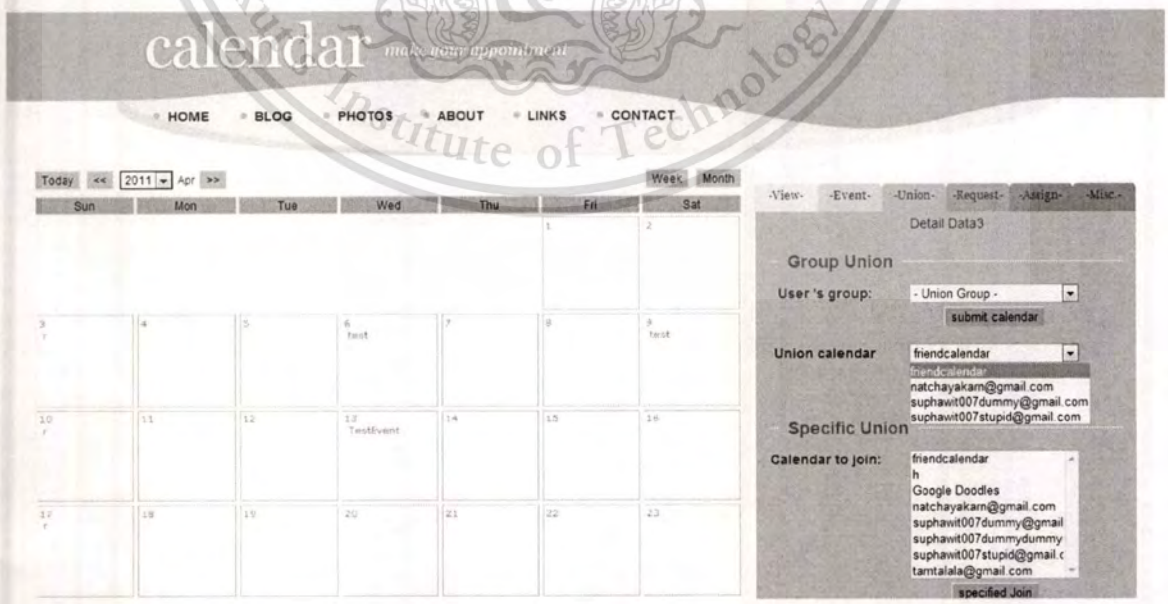


Figure 4.13 This screen show list of friend in group.

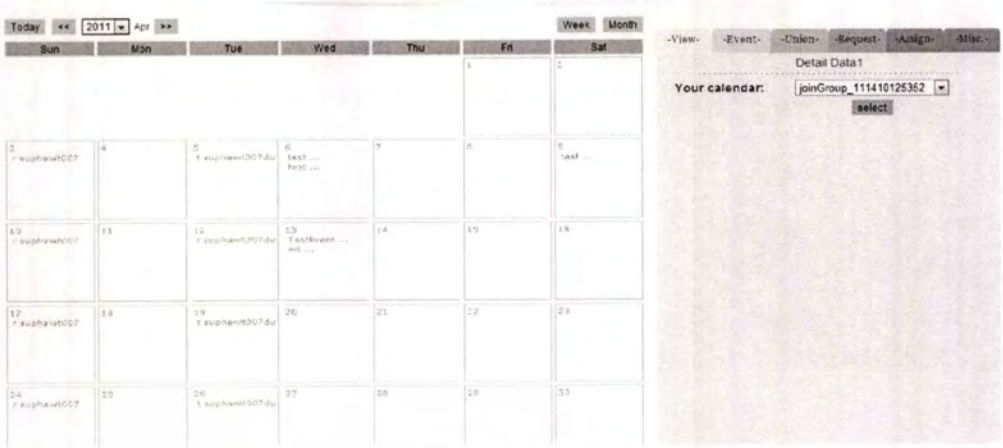


Figure 4.14 This figure show result of search for free time in group friend.

4.2.5 Managing an event

When the user wants to create an event, press -event- tab, the program will display fields for creating an event in Figure 4.15. In the event a user must specify the title of the activity data times the start and end of activity to be create an event which the user can press the add event to confirm information in the Create event. The confirmation process includes validating input for example participant email format.

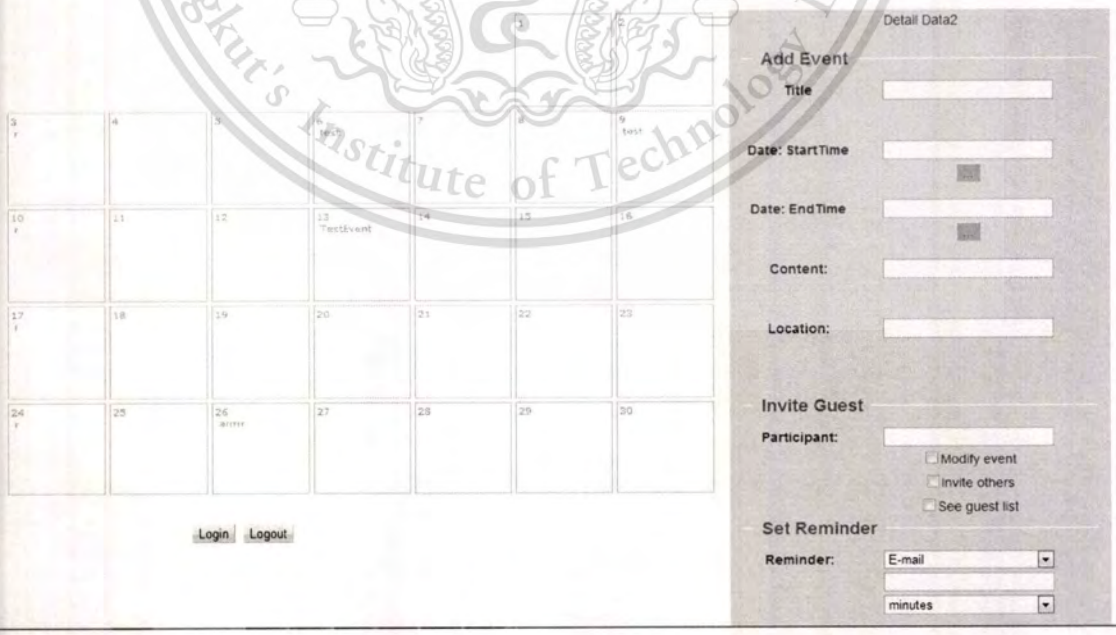


Figure 4.15 This figure shows the specific information used to create an event.

Users can edit an event by clicking the event for displaying the event details, shown in the Figure 4.16. Simply press delete event button to immediately delete the event. In figure 4.17, the user can edit it all except participant. Press Edit event button to confirm the edit event.

Figure 4.16 This figure display details in the event that day to delete the event

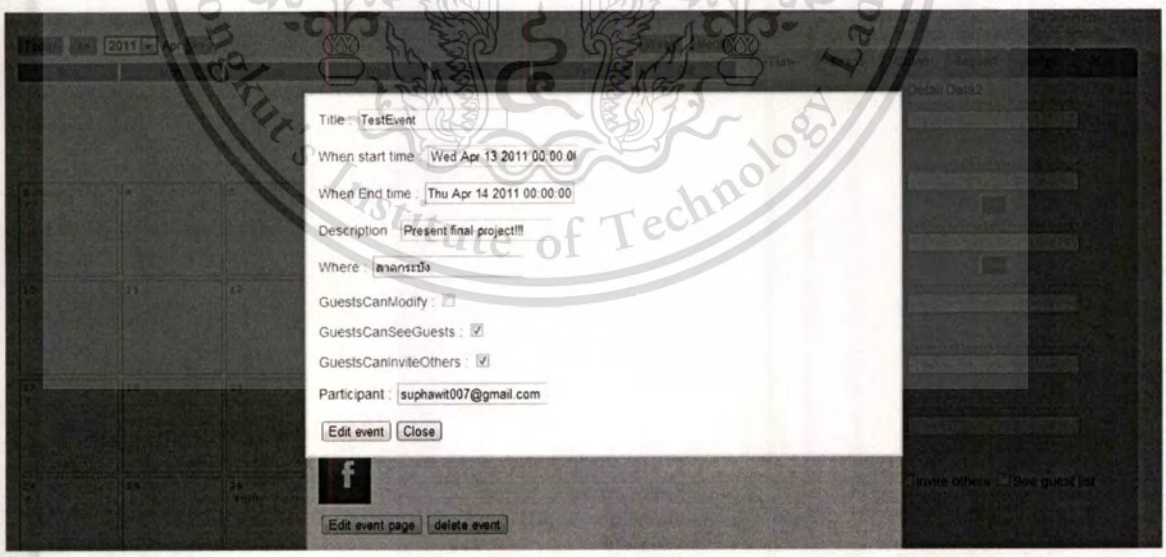


Figure 4.17 This Figure display the specific information used to modify the event.

4.2.6 Other social network features

Figure 4.18, user can talk with friends and leave comments about this event by using live chat.

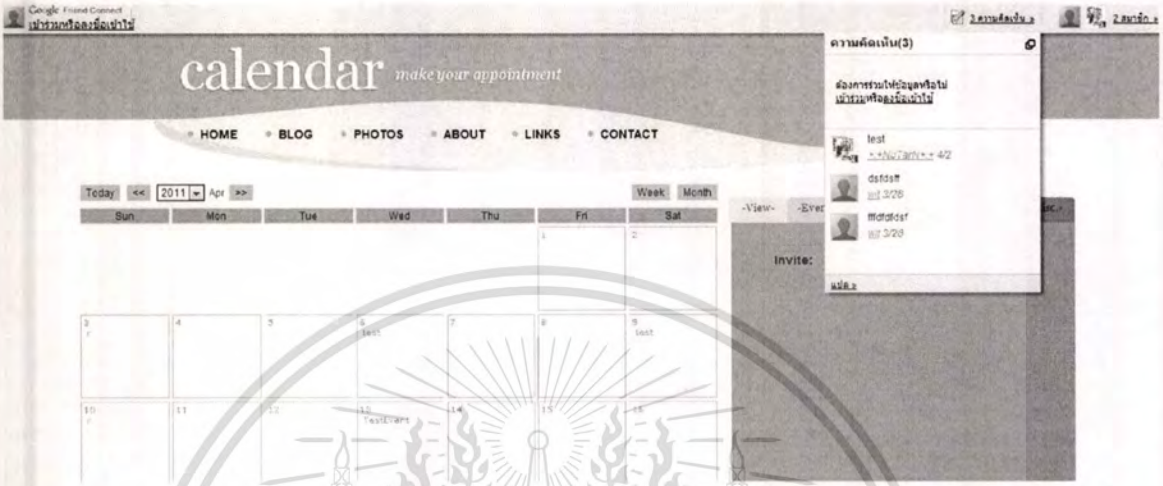


Figure 4.18 This figure show all comment on live chat bar.

Figure 4.19 shows the Facebook authentication dialog after publishing on Facebook image is clicked. The user is allowed to add any comment to the event, shown in figure 4.20, prior to the application post the event to the user's wall shown in figure 4.21.

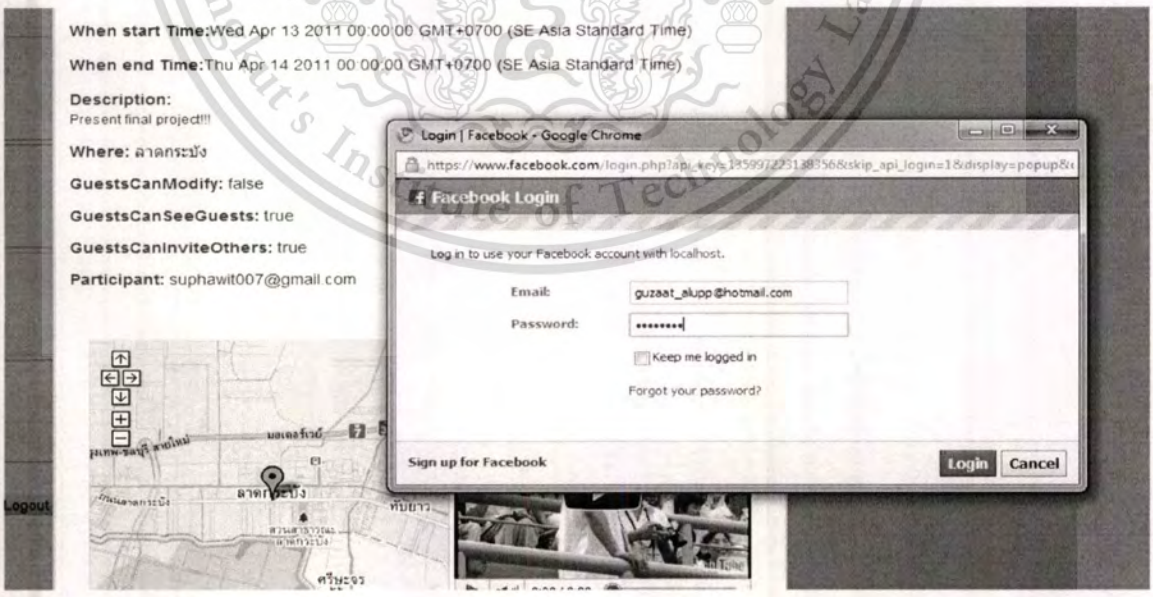


Figure 4.19 This figure show process of identification with facebook.

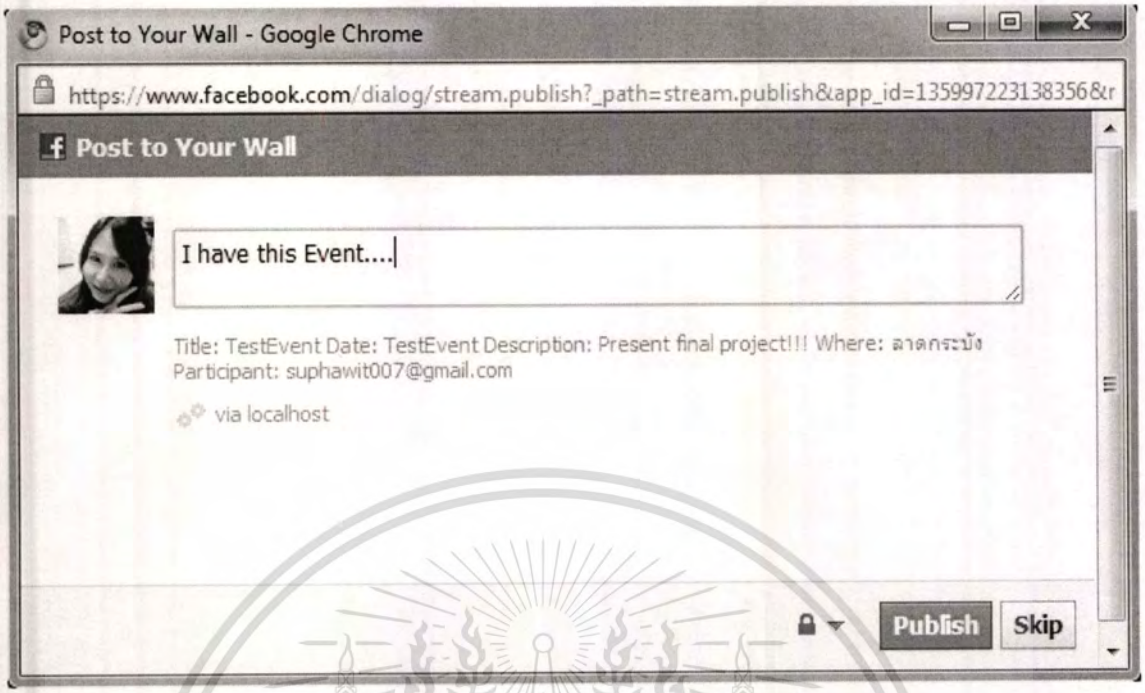


Figure 4.20 This fiqure show the user add comment of each event.

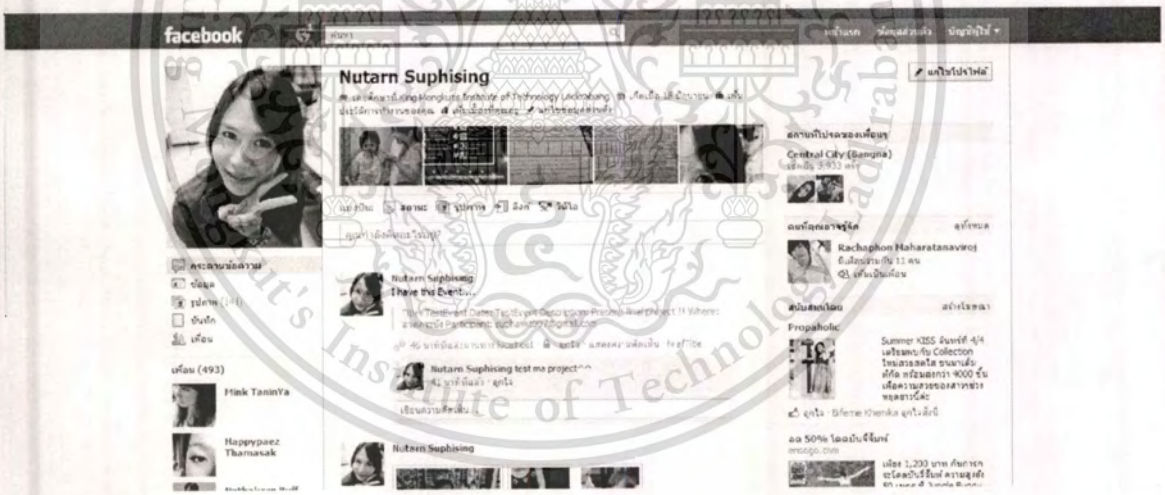


Figure 4.21 This figure display events that are published to facebook.

Chapter 5

Conclusion

We have achieved in implementing the system we envisioned. By using Google calendar APIs, our program has abilities to manage Google calendar appointments. It also contains social network features. The program can exchange appointment information with Google calendar by creating, editing, deleting and inviting others to participate at appointments. It also has ability to manage friend group to find the free time of all friend in group for the convenience of user to make appointments. Moreover, it can publish activities to social network such as, Facebook.

The only major pitfall which prevents us for actually deploying our system on Google app engine or any free online service is that we could not find a database service on such platform. We would not be able to hide our disappointment to accept that coordinating the group management on a dedicated platform is acceptable.

Other improvements worth discussion are as followed. Firstly, the user authority for viewing classes of event's privilege should be implemented. To be specific, we believe that in reality a user prefers to set a certain appointment to be private – be able to view its detail himself only. An event, in some cases, is meant for only participants to be able to view its detail. Since it is not in our scope, we briefly tested the tag but did not get the result we had expected. Because creating a new tag is not allowed, we, thus, would need to do further study on the issue.

Other issues that could improve our system are as followed. We have not tested whether it is possible for a user to invite more participants after the first invitation is made. Revoking the grant was not considered when designing the system. We believe this depends on whether Google calendar has such an API or not. Another feature is working on a repeatable event. This includes deleting it as well.

Displaying overlap events is also challenging. Had we been able to display events in color codes, the program would be much easier for viewing.

Reference

http://code.google.com/intl/th-TH/apis/calendar/data/1.0/developers_guide_js.html

http://code.google.com/intl/th-TH/apis/calendar/data/1.0/reference.html

http://code.google.com/intl/th-TH/appengine/docs/whatisgoogleappengine.html

http://www.w3schools.com/html/default.asp

http://www.w3schools.com/js/default.asp

http://www.w3schools.com/sql/default.asp

http://www.w3schools.com/php/default.asp