6-6-2013

Eventify : an android-based planning application

Bryson Lam
Santa Clara University

Jeff Matsunaga
Santa Clara University

Matt Tu
Santa Clara University

Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 7, 2013

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Bryson Lam, Jeff Matsunaga, and Matt Tu

ENTITLED

Eventify: An Android-based Event Planning Application

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER ENGINEERING

JoAnne Holliday
THESIS ADVISOR

DEPARTMENTCHAIR

Eventify: An Android-based Event Planning Application

by

Bryson Lam, Jeff Matsunaga, and Matt Tu

SENIOR DESIGN PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Engineering
School of Engineering
Santa Clara University

Santa Clara, California

June 6, 2013

1. Abstract

One of the common problems with social networking is the ability to plan out an event. Many applications available today simply provide for a broad or narrow audience of invitees. These tools also have the additional problem of being flooded with spam, and content with little interest to its users. To remedy this issue, we devised a new tool, Eventify.

Eventify is an Android based application that potentially will make creating events and bringing groups of people together more feasible. We built this application with two main design constraints in mind: The application runs solely via text messaging, without a centralized database and also without use of the internet. In this paper, we researched into the functionality and the architecture of this application, as well as test results and the development process of the application.

Our implementation of the application shows that creating a social-based system architecture is possible without use of the internet or use of a centralized database, which is unique to our application. We created a self-sustaining application that can run without support or maintenance costs, provided that all involved users have an Android device and/or text messaging capabilities. The application itself is presented in a straightforward manner; functionality of the application is apparent, and there is no excess flashiness in the application. The application relies on file storage within the phone itself to function: the text files created and handled by the application, however, are lightweight and typical use of the application takes up no more than 1MB of space on the device.

Eventify is a lightweight, decentralized application that makes event planning and friendly gatherings less troublesome to handle. While there are solutions that have been created for the purpose of event planning, Eventify was successfully implemented to require no support or maintenance costs and run solely via text messaging.

2. Acknowledgements

We would like to extend our thanks to our faculty advisor, Joanne Holliday. With her advice and guidance, our team was able to successfully produce a working product that achieved certain goals that we aimed to produce.

3. Table of Contents

4. Table of Figures

5. Introduction

5.1: Problem Statement

Event planning is a difficult task that can require extensive communication between all participants. Though social networking provides a good medium for communicating, it also has other features with can become distracting in organizing an event smoothly. In addition to this, reaching those outside of the social network can also become a problem as it usually leaves them out of the group. With the advancements made in technology over the past decade, the tools to coordinate events are much more readily available, however, there are still problems in creating a more mainstream method in doing so.

The method for organizing small gatherings proves to be more trouble than necessary. Communication and planning between two people via text message is simple and easy, but when more people are added to the group, problems begin to arise. Text messaging was originally developed as a form of communication between two people, and with the added functionality of group chats, text messaging became the most efficient way to plan the details of an event within a small group. Although this made the communication aspect easier, agreeing on a date and time was still just as difficult as before. Many factors may limit the ability to effectively plan an event such as a guest failing to respond, difficulty in tracking the messages where guests state their availability, and problems associated with taking everyone's availability and choosing a date and time for the event. As the number of people in the group grows to about 5 or more, using text messages as a means to plan an event falls apart.

When the size of a group becomes too large, text messaging fails as a method of planning an event, and other technologies can be utilized. Virtual social networks have become an integral part of human interaction in the 21st century. Some of these types of social networks recognize that although their technology connects people faster and easier than before, the need for physical human interaction is still necessary. To further their business, these companies created products that help people to plan their events. The most widely used event app for planning medium-to-large-scale events is the Facebook events app. By giving the user abilities, such as updating and viewing the guest-list in real-time, listing the event details, and allowing comments, the app does a great job of assisting in the organization of events. However, people use the app in ways that it was not intended to be used.

Facebook as a social network has become a somewhat impersonal one. Connections can be made between any two people, whether or not they know each other in real-life and the number of acquaintances on a user's friend list vastly outnumbers the amount of friends the user interacts with on a regular basis. As a result, users have realized that because of the way "friends" work on Facebook, the events app can be a method of marketing. Often times, users will have event invites that can be considered spam. Such invites include surveys, contests, and other forms of advertisement. These spam invites are the largest problem with the Facebook events app because it contributes to a decreased user satisfaction of the app.

Although the current solutions of event organization provide the basic functionality needed, these solutions do not solve all the problems associated with event planning. The system that currently handles most of the medium to large-scale events is filled with spam, while the scheme for small-scale events makes the coordination of details challenging. Our solution, Eventify, will be scalable enough to handle events of all sizes, and remedy the problems that exist with the current solutions. The proposed system will:

- Allow the event creator to invite other people to the event
- Allow the event to be either private or open-invite
- Coordinate the time and day of an event based on polls of the guests' availability and preference (if specified by the event creator)
- Coordinate the location of the event based on polls (if specified by the event creator)
- Links to a map of the location
- Coordinate carpooling (if specified by the event creator)
- Provide all guests with real-time updates to the guest-list
- Send notifications when details are changed or when an invite is received

The most important triumphs of this system will be the advantages associated with hosting the solution on a mobile platform. The event creator would be able to access all the features via a smartphone and all the event invites and notifications will be routed based on mobile numbers. Since phone numbers are not treated in the same manner as "friends" on Facebook, communication via phones is more personal than communication through Facebook, so spam or advertisement "events" will not be an issue. Eventify will provide an easy-to-use yet sophisticated solution to address the issues of event planning. The success of the app will be determined by the amount of frequent users and user satisfaction.

6. Main Body

6.1: Requirements

The requirements of the project specify the characteristics of the app. These requirements come in three forms: functional requirements, non-functional requirements and design constraints. Functional requirements define the specific actions that the app must be able to perform, non-functional requirements define what qualities the app must possess, and design constraints are requirements that limit the way the app can designed and implemented. Based on our vision of the application, we have identified the following functional requirements, non-functional requirements and design constraints:

Functional Requirements

The application:

- Must allow users to create a detailed event
- Allows users to invite other people to their event
- Is scalable to handle events of all sizes

Non-Functional Requirements

The application:

- should be easy to navigate through
- Functionality should be clear

Design Constraints

The application:

- Is built for Android 4.1.1 or higher
- Does not use a centralized server or database
- Does not require an extensive user registration

6.2: Conceptual Model

Based on the requirements, we have chosen these pictorial models to represent how the application appears from the user's perspective. This will help to ease understanding of concepts presented in the remainder of this document.

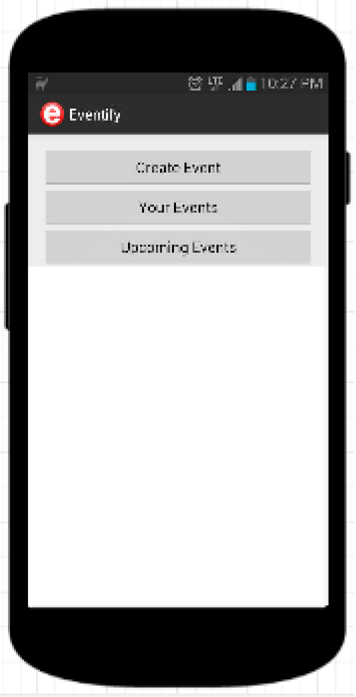
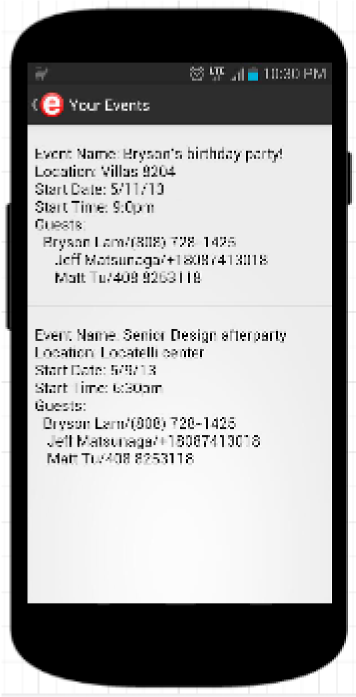


Figure 6.2.1: Home feed page mockup.



Event Name: Bryson's birthday party!
Location: Villas 9204
Start Date: 5/11/13
Start Time: 9:0pm
Guests:
Bryson Lam/(808) 728-1425
Jeff Matsunaga/+18087413018
Matt Tu/408 8253118

Event Name: Senior Design afterparty
Location: Locatelli center
Start Date: 5/9/13
Start Time: 6:30pm
Guests:
Bryson Lam/(808) 728-1425
Jeff Matsunaga/+18087413018
Matt Tu/408 8253118

Figure 6.2.2: Event creation page.

6.3: Use Cases

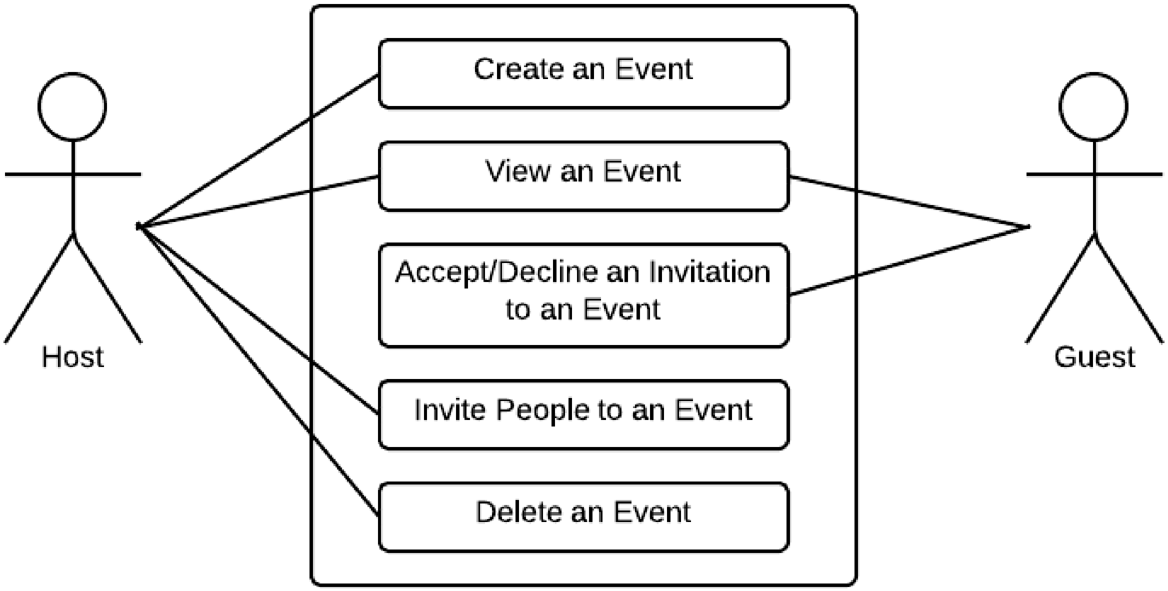

Create an Event
View an Event
Accept/Decline an Invitation to an Event
Invite People to an Event
Delete an Event
Host
Guest

Figure 6.3.1: Use case diagram for hosts and guests.

With the above requirements in mind, we have identified two main user groups: hosts and guests, and five main functions that the users will be able to perform. Hosts will be able to create events, invite people to events, and delete events; guests will be able to accept or decline event invitations, and both types of users should be able to view events that they are associated with.

Creating an Event

Actor: Host

Goal: Create an event

Preconditions: Host is at the app homepage (Figure 3.1)

Postconditions: Host has created an event and is ready to invite guests

Scenario:

1. Host selects the “Create” button from the homepage and is taken to the event creation page (Figure 3.2)
2. Host specifies the event title, location, date, time, and guests to be invited
3. Host selects the “Create Event” button

Exceptions: N/A

View an Event

Actor: Host or Guest

Goal: View the details of an event

Preconditions:

- Event already exists
- User is at the app’s homepage (Figure 3.1)

Postconditions: User has navigated to the desired event page (Figure 3.3)

Scenario:

1. The user navigates to the “Your Events” page if they want to view events that they are the host of, or the “Upcoming Events” page if they are the guest of an event.

Exceptions: N/A

Joining/Declining an Event

Actor: Guest

Goal: Join/Decline an event invitation

Preconditions:

- User is at the app homepage (Figure 3.1)
- User has been invited to an event by a host

Postconditions: User is on the guest list

Scenario:

1. User views the details of the event he/she wants to attend
2. User clicks on the appropriate “Attending,” “Not Attending,” or “Maybe Attending” option.

Exceptions: N/A

Deleting an Event

Actor: Host

Goal: Delete an event

Preconditions: The event that the host wants to delete must exist

Postconditions: The event is deleted and notifications are sent out to all invitees

Scenario:

1. User navigates to the event page that he/she wants to delete (Figure 3.3)
2. User selects the "Delete Event" option
3. User confirms the deletion of the event

Exceptions: N/A

6.4: System Sequence/Activity Diagram

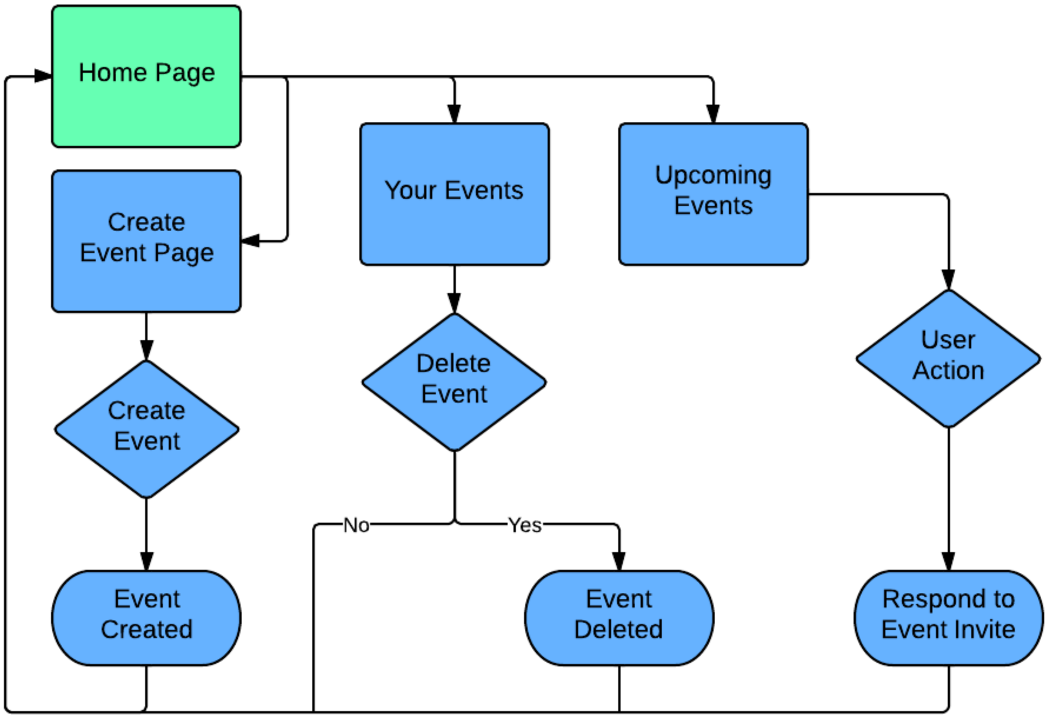


Figure 6.4.1: The app flow of Eventify.

6.5: Technologies Used

- Android Software Development Kit - Since our application is built for Android devices, we must be developed using the software development kit. Specifically, we used the Eclipse program as it was free and is the universally accepted medium for app development.
- Java: The language used to code Android applications.
- XML: Android applications consist of XML layout files.

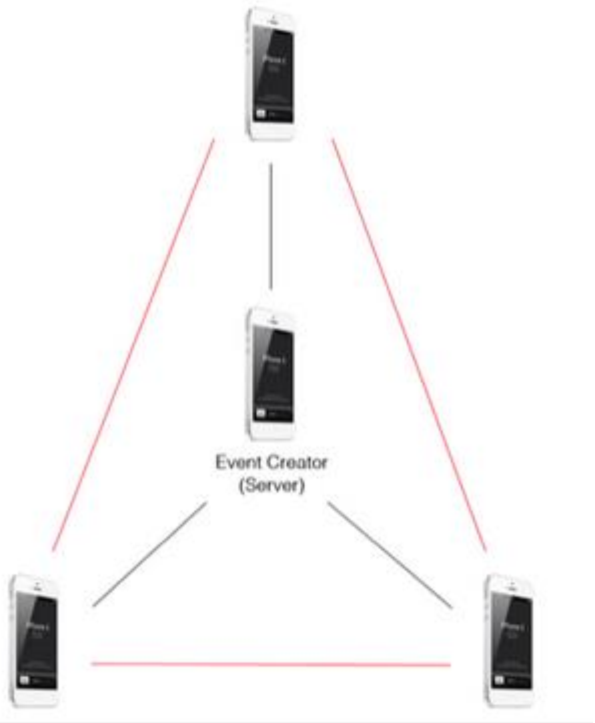


Figure 6.6.1: System Architecture and Communication Diagram.

6.6: System Architecture/Design Rationale

Since our app requires communication between Android devices, we are required to have a network topology. In this topology, each node is an Android device with our application installed, and each link represents the ability for the nodes it connects to communicate. We use each device's phone number as the address of the device, so in order for a node to communicate with another node, it needs to have the phone number associated with the specified node. With all things considered, our network is a peer-to-peer network, but not all nodes will have links to all other nodes.

The rationale behind the peer-to-peer network is that we want each device to be able to be a server and a client. Since any device should be able to create an event, the event creator should become the server for that event; that is, all messages must be routed through the event creator. Additionally, while creating any event, any device should be able to invite any other device to its event provided it knows the other device's phone number. These two features of our app justify the peer-to-peer networking architecture of the app.

To simplify the network aspect of our app, we decided to utilize the cell network to enable communication between devices. By doing this, we can use text messages to send and receive data that is relevant to our app. In addition, by using the cell network, we don't have to worry about phones losing connection or being turned off, as the network already handles these exceptions.

We also decided to store all the event data related to a device in the phone's internal storage. By doing this we eliminate the need for a centralized server thus reducing maintenance costs. The event data is stored in a simple text file only accessible by our app.

6.7: Societal Issues

Ethically, for our project, a major issue for Eventify was the handling of private information from the users. As with most social networking related applications, one issue that always arises is whether or not the user's information is being kept private from commercial companies, hackers, or other users. With our application, there is no centralized server so data is kept locally on the user's phones. However, the data is unencrypted in its current state, and doing so would be preferable for privacy due to the malware that is predominant amongst Android devices. This would definitely be an issue to look towards in future releases.

Socially, our application would assist in bring friends closer together. With our application creating a simple process to invite people to events, big or small, users can easily host parties or get-togethers with each other. Our application aims to simplify the process where other alternatives have failed, thus giving users additional and more efficient options.

Economically, our application has managed to reduce server costs through the use of a decentralized system. This process of using a peer-to-peer network allows users to host and create networks of their own, thus eliminating the need for a data center. At the same time however, many social networks make additional revenue due to data analytics and trends from their users. However, in aiming to create revenue, one possible solution would be to implement ad banners which would not harm the actual use of the application. Another alternate solution would be to allow users to discover nearby events which could be sponsored from other companies, thus giving us a form of revenue.

Regarding health and safety, our application does not encounter many of these issues. The main concern regarding safety would be users encountering strangers online. One way in which we have remedied this problem is by limiting a user's "friend list" to his/her contacts. By doing so, users would already have known these people before the launching of our application creating a safer social environment.

Manufacturing our application would be a relatively simple process. With resources to sell applications like the Google Play store, our application could be distributed easily with many resources to allow us to provide customers with updates and bug fixes.

Sustainability was taken into consideration with our application. We aimed to create a platform for event planning that was not only simple, but scalable as well. We designed Eventify to allow for events of different sizes whether that be from small get-togethers or large scale events like a play or movie screening. Being that our application is built on the Android platform, it also has the ability to provide users with updates through Google Play Store.

Environmental Impact of our application is fairly low due to it being based in software rather than hardware. The maintenance of our application is fairly low as well since we do not use a centralized server. One way in which we can prevent harm to the environment is by making our code as efficient as possible to ensure that resources such as battery life are not used wastefully.

Usability of our application is extremely straightforward. All functions in the application are clearly labeled for ease of use, and the application itself is presented in a minimalistic manner. Confirmation pages are generated as needed, and the application gives a Toast notification whenever an update to the guest list is completed.

Lifelong Learning has definitely been acquired from our project. Through this experience, our group has undergone the process of software development, meeting deadlines, and managing projects, all of which are important processes for software engineers. In addition to this, collaboration is an important factor in lifelong learning. Through this project, many decisions were made in which we worked together towards the best possible solution.

Compassion was accomplished with our application by creating a solution to a problem that existed within social networks. We aimed to bring people closer together, as well as attempt to bring a standard towards event planning. Social networks still have many issues in which we can connect people closer together.

Political issues do not play a role in our application. We designed Eventify for the purpose of bringing individuals together, and we designed it solely with functionality and usability in mind.

6.8: Project Risks

The risk table shows the problems that the team might run into while developing the app. It shows the probability of each problem occurring, how severe it will be, the impact it will have on the development of our app and a mitigation strategy.

Risks	Consequences	Probability	Severity	Impact	Mitigation Strategy
Unfamiliarity with the programming language	Less development time	0.7	9.0	6.3	Spend time in the beginning learning programming languages
Failure to note changes	Unknowingly overwriting changes	0.6	5.0	3.0	- Commented code - Group meetings
Unfamiliarity with the SDK	Less development time	0.5	5.0	2.5	Spend time in the beginning working with the SDK
Sickness	Less productivity	0.4	6.0	2.4	- Redistribute tasks
Communicatio n	Results in confusion in development of product	0.4	5.0	2.0	- Group meetings - Commented code
Time	Features will not be implemented	0.3	5.0	1.5	Complete all tasks on-time based on the Gantt charts

6.9: Test Plan

Prototyping (High and low fidelity)

Prototypes of the user interface allow us to develop a rough idea of what the application will look like. This allows us to test the capabilities and ease of use of the application without sinking copious amounts of time and effort into the actual development of the application.

Heuristic Analysis

Will be performed upon completion of prototypes of the application. Heuristic analyses ensure that the user interface of an application can be used effectively.

Host-Client Model Testing

We will test communication between Android to ensure the efficiency and correctness of the communication.

App Testing

Lastly, we will test all components of our app to ensure that they all function as specified.

6.10: Test Results

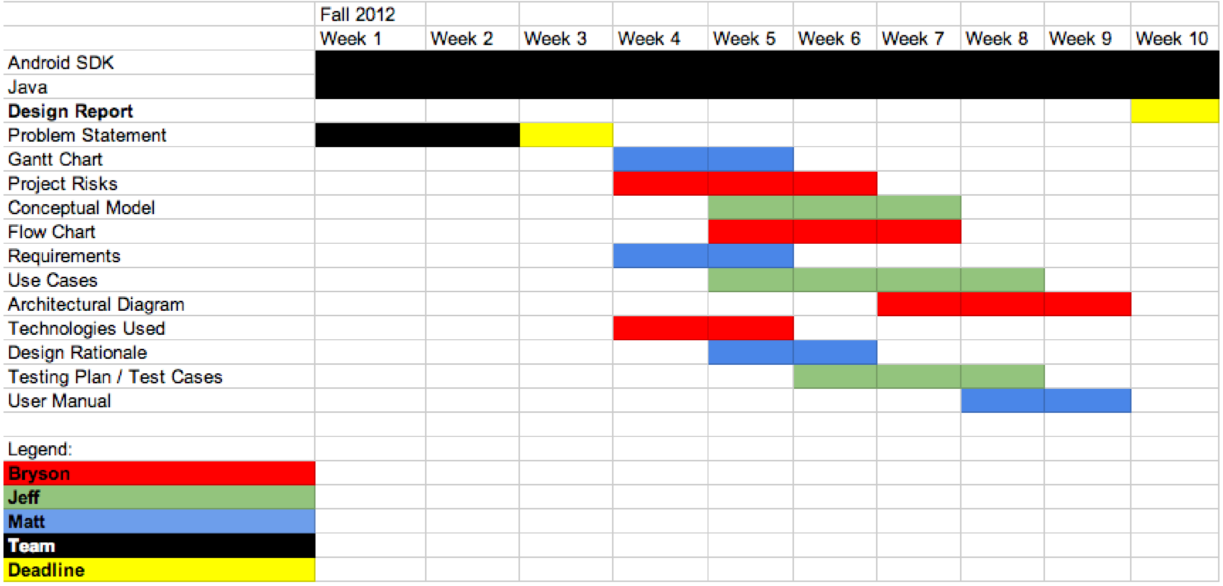
- Inviting other Eventify users to an event works as intended
- Inviting users without the application results in an SMS being sent to the user due to the design of the application

Currently Known Bugs after Version 1.0

- Event cannot be made with zero invitees
- “Date Picker” does not allow for only dates in the future
- “Clear Data” button for debugging still implemented

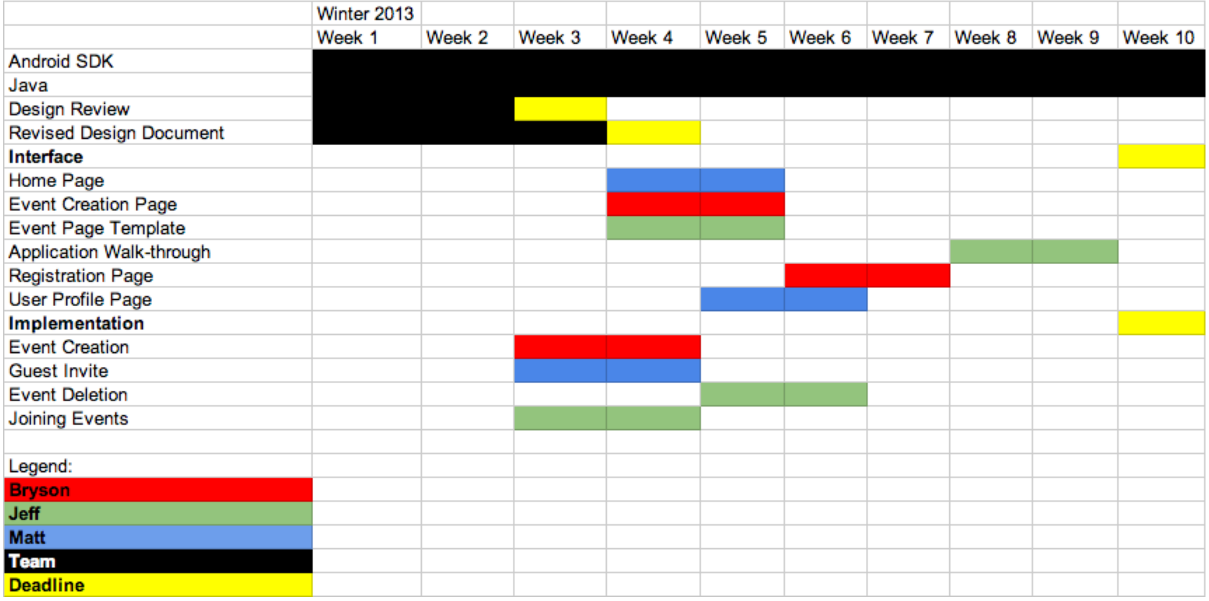
6.11: Development Timeline

With the features and requirements we have chosen to implement, we have developed the following project timelines in order to ensure that the project development remains on track.



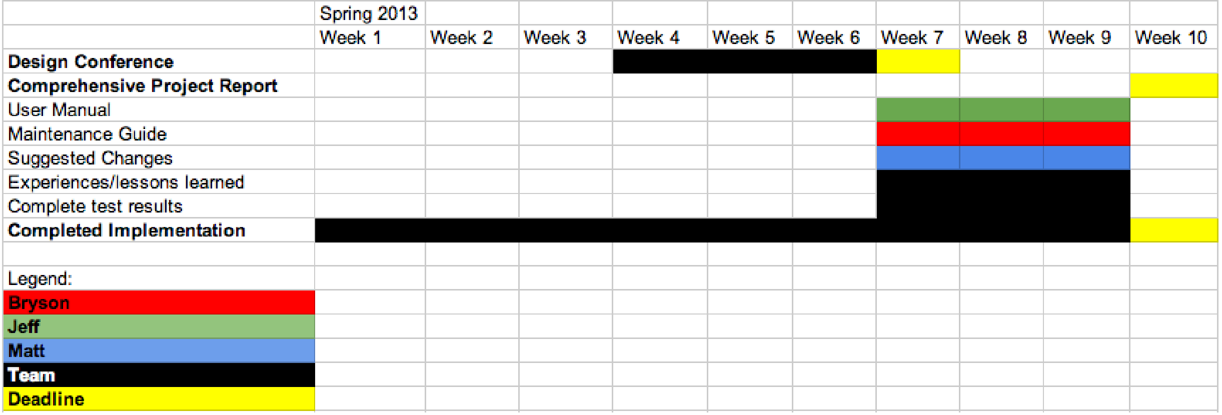
	Fall 2012									
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Android SDK	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Java	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Design Report										🟨
Problem Statement	⬛	⬛	🟨							
Gantt Chart				🟦	🟦					
Project Risks				🟥	🟥	🟥				
Conceptual Model					🟩	🟩	🟩			
Flow Chart					🟥	🟥	🟥			
Requirements				🟦	🟦					
Use Cases					🟩	🟩	🟩	🟩		
Architectural Diagram							🟥	🟥	🟥	
Technologies Used				🟥	🟥					
Design Rationale					🟦	🟦				
Testing Plan / Test Cases						🟩	🟩	🟩		
User Manual								🟦	🟦	
Legend:										
🟥 **Bryson**										
🟩 **Jeff**										
🟦 **Matt**										
⬛ **Team**										
🟨 **Deadline**										

Figure 6.11.1: Gantt Chart for Fall Quarter.



	Winter 2013									
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Android SDK	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Java	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Design Review	⬛	⬛	🟨							
Revised Design Document	⬛	⬛	⬛	🟨						
Interface										🟨
Home Page				🟦	🟦					
Event Creation Page				🟥	🟥					
Event Page Template				🟩	🟩					
Application Walk-through								🟩	🟩	
Registration Page						🟥	🟥			
User Profile Page					🟦	🟦				
Implementation										🟨
Event Creation			🟥	🟥						
Guest Invite			🟦	🟦						
Event Deletion					🟩	🟩				
Joining Events			🟩	🟩						
Legend:										
🟥 **Bryson**										
🟩 **Jeff**										
🟦 **Matt**										
⬛ **Team**										
🟨 **Deadline**										

Figure 6.11.2: Gantt Chart for Winter Quarter.



	Spring 2013									
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Design Conference				⬛	⬛	⬛	🟨			
Comprehensive Project Report										🟨
User Manual							🟩	🟩	🟩	
Maintenance Guide							🟥	🟥	🟥	
Suggested Changes							🟦	🟦	🟦	
Experiences/lessons learned							⬛	⬛	⬛	
Complete test results							⬛	⬛	⬛	
Completed Implementation	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	🟨
Legend:										
🟥 **Bryson**										
🟩 **Jeff**										
🟦 **Matt**										
⬛ **Team**										
🟨 **Deadline**										

Figure 6.11.3: Gantt Chart for Spring Quarter.

6.12: User Manual

On the first launch of Eventify, user "registration" will already be completed. This is due to the nature of Eventify accessing your contacts in order to build a friend list. The app will begin at the home page in which a user can choose to create an event, view their own events, or view any upcoming events that they were invited to.

If a user chooses to create a new event, they will be taken to a screen in which they can input an event name, date, time, and location for their event. In the next page, the user can select who they wish to invite. This list populates based on the users current contacts. Upon continuing, this information must be confirmed before the event's final creation.

Once a user has an event of their own, they can view it under the "Your Events" page. From here a user can touch and hold down an event to delete an event. Current implementation does not allow for the modification of events, but still do so in future releases.

If a user is invited to an event by another user the information will appear inside the "Upcoming Events" page. From here, a user can choose to respond to an event with "Going" or "Not Going". If a user chooses "Going", it will notify the event host of their attendance. If a user chooses "Not Going", then the event will be deleted off the user's upcoming events, and it will also reflect as such on the event host's event.

7. Conclusions

Eventify is now a skeletal backbone of what we hoped to achieve when we envisioned this project. In its current state, the application supports event creation, event details, and deletion of events. However, part of what our group wanted to take away from this project is a proof of concept: from the start, we wanted to avoid the use of a centralized database that would handle event details, mainly to cut down on maintenance and support costs. In turn, this constraint that we imposed upon the application led into our second main constraint of having the application run without internet dependence. While there are some pros and cons to our method, we simply should comply towards the standards set in place. Attempting to try something beyond the norm can lead to better understanding of how we can utilize our options to their full potential.

8. Appendix

8.1: Eventify Application Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.eventify"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.READ_SMS" />
    <uses-permission android:name="android.permission.WRITE_SMS" />
    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="17" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <receiver android:name="com.eventify.SMSReceiver">
            <intent-filter android:priority="2147483647"
```

```
android:exported="true" android:enabled="true">
                  <action
android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
      </receiver>
        <activity
            android:name="com.eventify.MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.eventify.CreateActivity"
            android:label="@string/title_activity_create"
                  android:parentActivityName="com.eventify.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.eventify.MainActivity" />
        </activity>
            <activity
            android:name="com.eventify.InviteGuests"
            android:label="@string/title_activity_invite_guests"
            android:parentActivityName="com.eventify.CreateActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.eventify.CreateActivity" />
        </activity>
            <activity
            android:name="com.eventify.CreateConfirm"
            android:label="@string/title_activity_confirm"
            android:parentActivityName="com.eventify.InviteGuests" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.eventify.InviteGuests" />
        </activity>
            <activity
            android:name="com.eventify.YourEventsActivity"
            android:label="@string/YourEvents"
            android:parentActivityName="com.eventify.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.eventify.MainActivity" />
        </activity>
            <activity
            android:name="com.eventify.UpcomingEventsActivity"
            android:label="@string/UpcomingEvents"
            android:parentActivityName="com.eventify.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.eventify.MainActivity" />
        </activity>
    </application>
</manifest>
```

8.2: Eventify Source Code (Java)

8.2a: CreateActivity.java

```
package com.eventify;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Typeface;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;

public class CreateActivity extends Activity {
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.createactivity);
		((TextView)findViewById(R.id.EventTitle)).setTypeface(null, 
Typeface.BOLD);
		((TextView)findViewById(R.id.LocationTitle)).setTypeface(null, 
Typeface.BOLD);
		((TextView)findViewById(R.id.DateTitle)).setTypeface(null, 
Typeface.BOLD);
		((TextView)findViewById(R.id.TimeTitle)).setTypeface(null, 
Typeface.BOLD);
	}
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		getMenuInflater().inflate(R.menu.create, menu);
		return true;
	}
	//Takes user to invite guests page
	public void toInviteGuests(View view) {
		DatePicker datePicker = 
(DatePicker)findViewById(R.id.datePicker1);
		TimePicker timePicker = 
(TimePicker)findViewById(R.id.timePicker1);

		String eventTitle = 
((EditText)findViewById(R.id.EventTitleInput)).getText().toString();
		String location = 
((EditText)findViewById(R.id.LocationInput)).getText().toString();
		String date = Integer.toString(datePicker.getMonth() + 1) + "/" 
+ Integer.toString(datePicker.getDayOfMonth()) + "/" + 
Integer.toString(datePicker.getYear() - 2000);
		int hour = timePicker.getCurrentHour();
```

```
            String partOfDay;
            if (hour >= 12) {
                  partOfDay = "pm";
                  if (hour > 12)
                        hour -= 12;
            }
            else {
                  partOfDay = "am";
                  if (hour == 0)
                        hour = 12;
            }
            String time = Integer.toString(hour) + ":" +
Integer.toString(timePicker.getCurrentMinute()) + partOfDay;
            if (eventTitle.equals("") && location.equals(""))
                  Toast.makeText(getApplicationContext(),"Please specify an
event name and location",Toast.LENGTH_SHORT).show();
            else if (eventTitle.equals(""))
                  Toast.makeText(getApplicationContext(),"Please specify an
event name",Toast.LENGTH_SHORT).show();
            else if (location.equals(""))
                  Toast.makeText(getApplicationContext(),"Please specify a
location",Toast.LENGTH_SHORT).show();
            else {
                  Intent intent = new Intent(this, InviteGuests.class);
                  intent.putExtra("eventName", eventTitle);
                  intent.putExtra("location", location);
                  intent.putExtra("startDate", date);
                  intent.putExtra("startTime", time);
                  startActivity(intent);
            }
      }
}
```

8.2b: CreateConfirm.java

```
package com.eventify;

import java.util.ArrayList;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class CreateConfirm extends Activity {
      private String _eventName, _location, _startDate, _startTime;
      private ArrayList<String> _guests;

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_create_confirm);
            _eventName = getIntent().getExtras().getString("eventName");
```

```
        _location = getIntent().getExtras().getString("location");
        _startDate = getIntent().getExtras().getString("startDate");
        _startTime = getIntent().getExtras().getString("startTime");
        _guests = getIntent().getStringArrayListExtra("guests");
        TextView text = (TextView)findViewById(R.id.textView1);
        String string = "Event Name: " + _eventName + "\nLocation: " +
_location + "\nStart Date: " + _startDate + "\nStart Time: "
                  + _startTime + "\nGuests:\n";
        for (String guest:_guests)
                  string += "\t" + guest.split("/")[0] + "\n";
        text.setText(string);
      }
      public void finishCreate(View view){
            EventData event = new EventData(_eventName, "Me", _startDate,
_startTime, _location, "Attending", _guests);
            EventData.addEvent(event, getApplicationContext());
            ArrayList<String> numbers = new ArrayList<String>();
            for (String guest:_guests)
                  numbers.add(guest.split("/")[1]);
          String message = "EventifyMessage:EventInvite::" + _eventName +
"::" + _startDate + "::" + _startTime + "::" + _location + "::";
          for (int i=0;i<_guests.size();i++)
            if (i==0)
                  message += _guests.get(i);
            else
                  message += "," + _guests.get(i);
          EventData.sendSMS(getApplicationContext(), numbers, message);
            Intent intent = new Intent(this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
      }
}
```

8.2c: EventData.java

```
package com.eventify;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.Arrays;

import android.content.Context;
import android.telephony.SmsManager;
import android.widget.Toast;

public class EventData {
      private String _name, _from, _startDate, _startTime, _location,
_status;
      private ArrayList<String> _guests;

      public EventData(String name, String from, String startDate, String
startTime, String location, String status, ArrayList<String> guests) {
            _name = name;
            _from = from;
            _startDate = startDate;
            _startTime = startTime;
            _location = location;
            _guests = guests;
            _status = status;
      }

      /*
       * Section: Accessor Methods
       * These methods will return the specified properties of a particular
event.
       */
      public String getName() { return _name; }
      public String getFrom() { return _from; }
      public String getStartDate() { return _startDate; }
      public String getStartTime() { return _startTime; }
      public String getLocation() { return _location; }
      public String getStatus() { return _status; }
      public ArrayList<String> getGuests() { return _guests; }
      public void setAttending() { _status = "Attending"; }
      public void setMaybe() { _status = "Maybe"; }
      public void setGuests(ArrayList<String> guests) { _guests = guests; }

/*
 * Section: Class Methods
 * These methods will update the application's file. The file is used for
 * persistent data about all the events that the user is involved in.
 */
      /*
       * Method: public static void writeEvents(ArrayList<EventData> events,
Context context);
       *
       * This method writes all of the event data passed into it to the
file.
       */
      public static void writeEvents(ArrayList<EventData> events, Context
context) {
            String text = "";
            int iter;

            for (EventData data:events) {
                  text += "Begin Event\nEvent Name:" + data.getName() +
"\nOwner:" + data.getFrom() + "\nStart Date:" + data.getStartDate() +
                              "\nStart Time:" + data.getStartTime() +
"\nLocation:" + data.getLocation() + "\nResponse: " + data.getStatus() +
                              "\nGuests:";
                  iter = 0;
                  for (String guest:data.getGuests()) {
                        if (iter!=0)
                              text += ", ";
                        text += guest;
                        iter++;
                  }
                  text += "\nEnd Event\n\n";
            }
            try {
                  FileOutputStream out =
context.openFileOutput("eventify.txt",0);
                  out.write(text.getBytes());
                  out.close();
            } catch (Exception e) { }
      }
      /*
       * Method: public static ArrayList<EventData> getAllEvents(Context
context);
       *
       * This method reads the file, parses the text, creates a list of all
of the events,
       * and returns the list.
       */
      public static ArrayList<EventData> getAllEvents(Context context) {
            ArrayList<EventData> data = new ArrayList<EventData>();
            String newText;
            ArrayList<String> fileParts, temp;
            byte[] bytes = new byte[1000];

            try {
                  FileInputStream in =
context.openFileInput("eventify.txt");
                  in.read(bytes);
                  in.close();
            } catch (Exception e) { }
            newText = new String(bytes);
            fileParts = new
ArrayList<String>(Arrays.asList(newText.split("\n\n")));
            for (int i=0;i<fileParts.size()-1;i++) {
                  String name, from, startDate, startTime, location, status;
                  ArrayList<String> guests;

                  name = from = startDate = startTime = location = status =
"";
                  guests = new ArrayList<String>();
                  temp = new
ArrayList<String>(Arrays.asList(fileParts.get(i).split("\n")));
                  for (int iter = 1; iter<temp.size()-1; iter++) {
                        ArrayList<String> parts = new
ArrayList<String>(Arrays.asList(temp.get(iter).split(":",2)));
                        if (parts.get(0).contains("Event Name"))
                              name = parts.get(1);
                        else if (parts.get(0).contains("Owner"))
                              from = parts.get(1);
                        else if (parts.get(0).contains("Start Date"))
                              startDate = parts.get(1);
                        else if (parts.get(0).contains("Start Time"))
                              startTime = parts.get(1);
                        else if (parts.get(0).contains("Location"))
                              location = parts.get(1);
                        else if (parts.get(0).contains("Response"))
                              status = parts.get(1);
                        else if (parts.get(0).contains("Guests"))
                              guests = new
ArrayList<String>(Arrays.asList(parts.get(1).split(",")));
                  }
                  data.add(new
EventData(name,from,startDate,startTime,location,status,guests));
            }
            return data;
      }
      /*
       * Method: public static void addEvent(EventData event, Context
context);
       *
       * This method adds an event to the file. For simplicity, we query all
of the
       * events currently in the file using getAllEvents(Context context),
add the event
       * to the event list, and rewrite the entire file.
       */
      public static void addEvent(EventData event, Context context) {
            ArrayList<EventData> events;

            events = getAllEvents(context);
            events.add(event);
            writeEvents(events, context);
      }
      /*
       * Method: public static void deleteEvent(String eventName, Context
context);
       *
       * This method deleted the specified event. The event to be deleted is
identified
       * by the combination of the event name and the event creator. For
simplicity, we
       * query all of the events currently in the file using
getAllEvents(Context context),
       * iterate over the list to find the specified event, remove that
event from the list
       * and rewrite the entire file.
       */
      public static void deleteEvent(String eventName, String owner, Context
context) {
            ArrayList<EventData> events;
            int count = 0;

            events = getAllEvents(context);
            for (EventData event: events) {
                  if (eventName.equals(event.getName()) &&
owner.equals(event.getFrom())) {
                        events.remove(count);
                        break;
                  }
                  count++;
            }
            writeEvents(events, context);
      }
      /*
       * This method is a stub to create dummy data.
       * DELETE LATER!!!!
       */
      public static void loadDummyData(Context context) {
            ArrayList<EventData> data = new ArrayList<EventData>();
            ArrayList<String> guests = new ArrayList<String>();
            EventData data2;

            guests.add("Bryson Lam");
            guests.add("Jeff Matsunaga");
            guests.add("Matt Tu");
            data2 = new EventData("Jeff's Birthday Party","Me","1/25/14",
"10:00pm","Jeff's House","Attending",guests);
            data.add(data2);
            data2 = new EventData("Matt's Birthday Party","Matt
Tu","11/18/13","11:00pm","Matt's House","Pending",guests);
            data.add(data2);
            data2 = new EventData("Bryson's Birthday Party","Bryson
Lam","6/12/13","11:00pm","Bryson's House","Pending",guests);
            data.add(data2);
            EventData.writeEvents(data,context);
      }
      public static void clearData(Context context) {
            String text = "";
            try {
                  FileOutputStream out =
context.openFileOutput("eventify.txt",0);
                  out.write(text.getBytes());
                  out.close();
            } catch (Exception e) { }
      }
      public static void sendSMS(Context context,ArrayList<String> numbers,
String msg) {
            if (numbers.size()<=0)
                  return;
          SmsManager smsManager = SmsManager.getDefault();
          for (String number:numbers)
            smsManager.sendTextMessage(number, null, msg, null, null);
          Toast.makeText(context, "SMS Sent", Toast.LENGTH_SHORT).show();
      }
}
```

8.2d: InviteGuests.java

```
package com.eventify;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.provider.ContactsContract;
import android.view.View;
import android.widget.ListView;

public class InviteGuests extends Activity {
      private String _eventName, _location, _startDate, _startTime;
      private ArrayList<String> contactList;
      ArrayList<String> al = null;
      private ListView tableRows;
      public int position = 0;
      SelectedAdapter selectedAdapter;
      String pStr;

      /*
       * Handler gets the table cell clicks from the list view
       * and checks or unchecks the checkbox.
       */
      Handler boxHandler = new Handler() {
            public void handleMessage(Message msg) {
                  position = msg.arg1;
                  if (msg.arg2 == 1) {
                        String s = contactList.get(position);
                        String s1 = s.substring(0, 1);
                        if (msg.arg2 == 1) {
                              s = s.substring(2);
                              if (s1.equals("0"))
                                    s = "1,"+s;
                              else
                                    s = "0,"+s;
                              contactList.set(position,s);
                              al.set(position, s);
                        }
                  }
                  selectedAdapter.setSelectedPosition(position);
            }
      };
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.contactlist);
        _eventName = getIntent().getExtras().getString("eventName");
        _location = getIntent().getExtras().getString("location");
        _startDate = getIntent().getExtras().getString("startDate");
        _startTime = getIntent().getExtras().getString("startTime");
        contactList = getContactList();
        configure();
    }
    public void toConfirm(View view) {
      ArrayList<String> guests = new ArrayList<String>();
      for (String c:contactList)
            if (c.startsWith("1,"))
                  guests.add(parse(c));
      Intent intent = new Intent(this, CreateConfirm.class);
            intent.putExtra("eventName", _eventName);
            intent.putExtra("location", _location);
            intent.putExtra("startDate", _startDate);
            intent.putExtra("startTime", _startTime);
            intent.putStringArrayListExtra("guests", guests);
            startActivity(intent);
    }
    private ArrayList<String> getContactList() {
      ArrayList<String> contacts = new ArrayList<String>();
      Cursor people = 
getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_UR
I,
                  new String[] 
{ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,ContactsContract.Common
DataKinds.Phone.NUMBER},null,null,null);
      int indexName = 
people.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME);
      int indexNumber = 
people.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);

      people.moveToFirst();
      do {
            String name = people.getString(indexName);
            String number = people.getString(indexNumber);
            contacts.add("0,\t" + name + "\n\t\t" + number);
      } while (people.moveToNext());
      Collections.sort(contacts);
      return contacts;
    }
      private void configure() {
            al = new ArrayList<String>();
            for(String s : contactList)
                  al.add(s);
            selectedAdapter = new SelectedAdapter(this,0,al);
            selectedAdapter.setNotifyOnChange(true);
            tableRows = (ListView)findViewById(R.id.table);
            tableRows.setAdapter(selectedAdapter);
            selectedAdapter.setHandler(boxHandler);
            selectedAdapter.setSelectedPosition(position);
            tableRows.setSelection(position);
      }
    /*
     * Parsers returns the name and number of the contact. The name is
     * seperated from the number by a '\n'.
     */
      private String parse(String text) {
            ArrayList<String> parts;

            parts = new ArrayList<String>(Arrays.asList(text.split(",",2)));
            parts.set(1,parts.get(1).replaceAll("\t", ""));
            parts.set(1,parts.get(1).replaceAll("\n", "/"));
            return parts.get(1);
      }
}
```

8.2e: MainActivity.java

```
package com.eventify;
```

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;

public class MainActivity extends Activity {
      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
      }
      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            getMenuInflater().inflate(R.menu.main, menu);
            return true;
      }
      public void toCreateActivity(View view) {
            //Takes user to create event page
            Intent intent = new Intent(this, CreateActivity.class);
            startActivity(intent);
      }
      public void toYourEvents(View view) {
            //Takes user to "Your Events" page
            Intent intent = new Intent(this, YourEventsActivity.class);
            startActivity(intent);
      }
      public void toUpcomingEvents(View view) {
            //Takes user to "Upcoming Events" page
            Intent intent = new Intent(this, UpcomingEventsActivity.class);
            startActivity(intent);
      }
      public void click(View view) {
            EventData.clearData(getApplicationContext());
      }
}
```

8.2f: SelectedAdapter.java

```
package com.eventify;
import java.util.List;

import android.content.Context;
import android.graphics.Color;
import android.os.Handler;
import android.os.Message;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.TextView;
```

```
public class SelectedAdapter extends ArrayAdapter<String>{
      private Handler handler = null;
      private int selectedPos = -1;

      public SelectedAdapter(Context context, int textViewResourceId,
List<String> objects) {
            super(context, textViewResourceId, objects);
      }

      public void setSelectedPosition(int pos) {
            selectedPos = pos;
            notifyDataSetChanged();
      }

      public int getSelectedPosition(){
            return selectedPos;
      }

      public void setHandler(Handler h) {
            handler = h;
      }

      @Override
      public View getView(int position, View convertView, ViewGroup parent)
{
            View v = convertView;
            if (v == null) {
                  LayoutInflater vi =
(LayoutInflater)this.getContext().getSystemService(Context.LAYOUT_INFLATER_S
ERVICE);
                  v = vi.inflate(R.layout.selected_row, null);
            }
            TextView label = (TextView)v.findViewById(R.id.txtExample);
            String s = this.getItem(position).toString();
            String s1 = s.substring(0, 1);
            s = s.substring(2);
            if (selectedPos == position)
                  label.setBackgroundColor(Color.WHITE);
            else
                  label.setBackgroundColor(Color.WHITE);
            label.setText(s);
            final int tmpPos = position;
            label.setOnClickListener( new OnClickListener() {
                  public void onClick(View m) {
                        if (handler != null) {
                              Message message = handler.obtainMessage();
                              message.arg1 = tmpPos;    //use this to find
in array
                              message.arg2 = 0;         //zero clicked label
                              handler.sendMessage(message);
                        }
                  }
            });
            CheckBox checkbox = (CheckBox)v.findViewById(R.id.checkbox);
            if (s1.equals("0"))
                  checkbox.setChecked(false);
            else
                  checkbox.setChecked(true);
            checkbox.setOnClickListener( new OnClickListener() {
                  public void onClick(View m) {
                        if (handler != null) {
                              Message message = handler.obtainMessage();
                              message.arg1 = tmpPos; //use this to find in
array
                              message.arg2 = 1;          //one clicked
checkbox
                              handler.sendMessage(message);
                        }
                  }
            });
            return(v);
      }
}
```

8.2g:SMSReceiver.java

```
package com.eventify;

import java.util.ArrayList;
import java.util.Arrays;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.widget.Toast;

 public class SMSReceiver extends BroadcastReceiver {

       private static final String SMS_RECEIVED =
"android.provider.Telephony.SMS_RECEIVED";

      @Override
      public void onReceive(Context context, Intent intent) {
           String smsData, sender;
           if (intent.getAction().equals(SMS_RECEIVED)) {
               Bundle pudsBundle = intent.getExtras();
               Object[] pdus = (Object[]) pudsBundle.get("pdus");
               SmsMessage message = SmsMessage.createFromPdu((byte[])
pdus[0]);
               smsData = message.getMessageBody();
               sender = message.getOriginatingAddress();
               if (smsData.contains("EventifyMessage:EventInvite::")) {
                     ArrayList<String> dataFromText, guests;
                     EventData data;

                     dataFromText = new
ArrayList<String>(Arrays.asList(smsData.split("::")));
                     guests = new
```

```
ArrayList<String>(Arrays.asList(dataFromText.get(5).split(",")));
                     data = new
EventData(dataFromText.get(1),sender,dataFromText.get(2),dataFromText.get(3)
,dataFromText.get(4),"Pending",guests);
                     EventData.addEvent(data, context);
                     Toast.makeText(context, "SMS Received",
Toast.LENGTH_SHORT).show();
                     abortBroadcast();
               }
               else if (smsData.contains("EventifyMessage:EventDelete::")) {
                     ArrayList<String> dataFromText;

                     dataFromText = new
ArrayList<String>(Arrays.asList(smsData.split("::")));
                     EventData.deleteEvent(dataFromText.get(1), sender,
context);
                     Toast.makeText(context, "SMS Received",
Toast.LENGTH_SHORT).show();
                     abortBroadcast();
               }
               else if
(smsData.contains("EventifyMessage:EventDeleteResponse::")) {
                     ArrayList<String> dataFromText;
                     ArrayList<EventData> data =
EventData.getAllEvents(context);

                     dataFromText = new
ArrayList<String>(Arrays.asList(smsData.split("::")));
                     dataFromText.set(1,dataFromText.get(1).trim());
                     for (EventData e:data)
                           if (e.getName().equals(dataFromText.get(1)) &&
e.getFrom().equals("Me")) {
                                 ArrayList<String> guests = e.getGuests();
                                 for (int i=0;i<guests.size();i++) {
                                       if
(sender.equals(guests.get(i).split("/")[1])) {
                                             guests.remove(i);
                                             break;
                                       }
                                 }
                                 EventData.deleteEvent(e.getName(), "Me",
context);
                                 e.setGuests(guests);
                                 EventData.addEvent(e, context);
                                 String msg =
"EventifyMessage:EventUpdate::" + e.getName() + "::" + e.getStartDate() +
"::" + e.getStartTime() +
                                             "::" + e.getLocation() + "::";
                                  for (int i=0;i<guests.size();i++)
                                    if (i==0)
                                          msg += guests.get(i);
                                    else
                                          msg += "," + guests.get(i);
                                 ArrayList<String> numbers = new
ArrayList<String>();
                                 for (String guest:guests)
```

```
                                          numbers.add(guest.split("/")[1]);
                                 EventData.sendSMS(context, numbers, msg);
                                 break;
                           }
                     Toast.makeText(context, "SMS Received", 
Toast.LENGTH_SHORT).show();
                     abortBroadcast();
               }
               else if (smsData.contains("EventifyMessage:EventUpdate")) {
                     ArrayList<String> dataFromText, guests;
                     EventData data;
                     ArrayList<EventData> events = 
EventData.getAllEvents(context);

                     dataFromText = new 
ArrayList<String>(Arrays.asList(smsData.split("::")));
                     guests = new 
ArrayList<String>(Arrays.asList(dataFromText.get(5).split(",")));
                     for (EventData e:events)
                           if (e.getName().equals(dataFromText.get(1)) && 
e.getFrom().equals(sender)) {
                                 data = new 
EventData(dataFromText.get(1),sender,dataFromText.get(2),dataFromText.get(3)
,dataFromText.get(4),e.getStatus(),guests);
                                 EventData.deleteEvent(e.getName(), 
e.getFrom(), context);
                                 EventData.addEvent(data, context);
                                 break;
                           }
                     Toast.makeText(context, "SMS Received", 
Toast.LENGTH_SHORT).show();
                     abortBroadcast();
               }
           }
      }
 }
```

8.2h:UpcomingEventsActivity.java

```
package com.eventify;

import java.util.ArrayList;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
```

```
public class UpcomingEventsActivity extends Activity {
      ArrayList<String> tableRows;

      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.upcomingevents);
            ListView lv = (ListView)findViewById(R.id.upcominglist);
            ArrayList<EventData> data;
            tableRows = new ArrayList<String>();

            data = EventData.getAllEvents(getApplicationContext());
            for (EventData e:data)
                  if (!e.getFrom().equals("Me")) {
                        String text = "\nEvent Name: " + e.getName() +
"\nOwner: " + e.getFrom() + "\nLocation: " + e.getLocation() +
                                    "\nStart Date: " + e.getStartDate() +
"\nStart Time: " + e.getStartTime() + "\nStatus: " + e.getStatus() +
                                    "\nGuests:\n";
                        for (String guest:e.getGuests())
                        text += "\t" + guest + "\n";
                        tableRows.add(text);
                  }
            ArrayAdapter<String> arrayAdapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, tableRows);
            lv.setAdapter(arrayAdapter);
            registerForContextMenu(lv);
      }
      @Override
      public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenuInfo menuInfo) {
            if (v.getId()==R.id.upcominglist) {
                  menu.setHeaderTitle("Response?");
                  String[] menuItems = {"Attending","Maybe","Decline"};
                  for (int i = 0; i<menuItems.length; i++)
                        menu.add(Menu.NONE, i, i, menuItems[i]);
            }
      }
      @Override
      public boolean onContextItemSelected(MenuItem item) {
            AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
            int index = item.getItemId();
            String eventString = tableRows.get(info.position);
            if (index == 0) {
                  ArrayList<EventData> data =
EventData.getAllEvents(getApplicationContext());
                  String[] parts = eventString.split("\n");
                  parts = parts[1].split(":");
                  parts[1] = parts[1].trim();
                  for (EventData e:data)
                        if (e.getName().equals(parts[1])) {
                              e.setAttending();
                              EventData.deleteEvent(e.getName(),
e.getFrom(), getApplicationContext());
                              EventData.addEvent(e,
getApplicationContext());
```

```
                              break;
                        }
            }
            else if (index == 1) {
                  ArrayList<EventData> data =
EventData.getAllEvents(getApplicationContext());
                  String[] parts = eventString.split("\n");
                  parts = parts[1].split(":");
                  parts[1] = parts[1].trim();
                  for (EventData e:data)
                        if (e.getName().equals(parts[1])) {
                              e.setMaybe();
                              EventData.deleteEvent(e.getName(),
e.getFrom(), getApplicationContext());
                              EventData.addEvent(e,
getApplicationContext());
                              break;
                        }
            }
            else if (index == 2) {
                  ArrayList<EventData> data =
EventData.getAllEvents(getApplicationContext());
                  String[] parts = eventString.split("\n");
                  parts = parts[1].split(":");
                  parts[1] = parts[1].trim();
                  for (EventData e:data)
                        if (e.getName().equals(parts[1]) &&
!e.getFrom().equals("Me")) {
                              String msg =
"EventifyMessage:EventDeleteResponse::" + e.getName();
                              ArrayList<String> numbers = new
ArrayList<String>();
                              numbers.add(e.getFrom());
                              EventData.sendSMS(getApplicationContext(),
numbers, msg);
                              EventData.deleteEvent(e.getName(),
e.getFrom(), getApplicationContext());
                              break;
                        }
            }
            Intent intent = new Intent(this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            return true;
      }
}
```

8.2i:YourEventsActivity.java

```
package com.eventify;

import java.util.ArrayList;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class YourEventsActivity extends Activity {
      ArrayList<String> tableRows;

      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.yourevents);
            ListView lv = (ListView)findViewById(R.id.listView1);
            ArrayList<EventData> data;
            tableRows = new ArrayList<String>();

            data = EventData.getAllEvents(getApplicationContext());
            for (EventData e:data)
                  if (e.getFrom().equals("Me")) {
                        String text = "\nEvent Name: " + e.getName() +
"\nLocation: " + e.getLocation() + "\nStart Date: " + e.getStartDate() +
                              "\nStart Time: " + e.getStartTime()  +
"\nGuests:\n";
                        for (String guest:e.getGuests())
                        text += "\t" + guest + "\n";
                        tableRows.add(text);
                  }
            ArrayAdapter<String> arrayAdapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, tableRows);
            lv.setAdapter(arrayAdapter);
          registerForContextMenu(lv);
      }
      @Override
      public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenuInfo menuInfo) {
            if (v.getId()==R.id.listView1) {
                  menu.setHeaderTitle("Delete Event?");
                  String[] menuItems = {"Yes","No"};
                  for (int i = 0; i<menuItems.length; i++)
                        menu.add(Menu.NONE, i, i, menuItems[i]);
            }
      }
      @Override
      public boolean onContextItemSelected(MenuItem item) {
            AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo)item.getMenuInfo();
            int index = item.getItemId();
            String eventString = tableRows.get(info.position);
            if (index == 0) {
                  ArrayList<EventData> data =
EventData.getAllEvents(getApplicationContext());
                  String[] parts = eventString.split("\n");
                  parts = parts[1].split(":");
                  parts[1] = parts[1].trim();
                  for (EventData e:data)
                        if (e.getFrom().equals("Me") &&
e.getName().equals(parts[1])) {
                              ArrayList<String> guests = e.getGuests();
                              for (int i=0;i<guests.size();i++)

      guests.set(i,guests.get(i).split("/")[1].trim());
                              String message =
"EventifyMessage:EventDelete::" + e.getName();
                              EventData.sendSMS(getApplicationContext(),
guests, message);
                              EventData.deleteEvent(e.getName(), "Me",
getApplicationContext());
                              break;
                        }
            }
            Intent intent = new Intent(this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            return true;
      }
}
```

8.3: Eventify Layout Source Code (XML)

8.3a: activity_create_confirm.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/EventTitle"
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <Button
        android:id="@+id/createactualevent"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginRight="@dimen/activity_horizontal_margin"
        android:text="@string/CreateEvent"
        android:onClick="finishCreate" />
</LinearLayout>
```

8.3b: activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/toCreate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/CreateEvent"
        android:onClick="toCreateActivity" />
    <Button
        android:id="@+id/toYourEvents"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/YourEvents"
        android:onClick="toYourEvents" />
    <Button
        android:id="@+id/toUpcomingPage"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/UpcomingEvents"
        android:onClick="toUpcomingEvents" />
    <Button
        android:id="@+id/Clear"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/Clear"
        android:onClick="click" />
</LinearLayout>
```

8.3c: contactlist.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button_send"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="14sp"
        android:onClick="toConfirm"
        android:text="@string/Done" />
    <ListView
        android:id="@+id/table"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:choiceMode="singleChoice"
        android:background="#FFFFFF" />
</LinearLayout>
```

8.3d: createactivity.xml

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    tools:context=".CreateActivity">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
            <TextView
                  android:id="@+id/EventTitle"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginTop="@dimen/activity_vertical_margin"

      android:layout_marginRight="@dimen/activity_horizontal_margin"
                  android:text="@string/EventTitle"
                  android:textAppearance="?android:attr/textAppearanceSmall"
/>
            <EditText
                  android:id="@+id/EventTitleInput"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin"
                  android:ems="10"
                  android:inputType="text" />
            <TextView
                  android:id="@+id/LocationTitle"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin"
                  android:layout_marginTop="10dp"
                  android:text="@string/Location"
                  android:textAppearance="?android:attr/textAppearanceSmall"
/>
            <EditText
                  android:id="@+id/LocationInput"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin"
                  android:ems="10"
                  android:inputType="text" />
            <TextView
                  android:id="@+id/DateTitle"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin"
```

```
                  android:layout_marginTop="10dp"
                  android:text="@string/PickDate"
                  android:textAppearance="?android:attr/textAppearanceSmall"
/>
            <DatePicker
                  android:id="@+id/datePicker1"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin"
                  android:calendarViewShown="false" />
            <TextView
                  android:id="@+id/TimeTitle"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin"
                  android:layout_marginTop="10dp"
                  android:text="@string/PickTime"
                  android:textAppearance="?android:attr/textAppearanceSmall"
/>
            <TimePicker
                  android:id="@+id/timePicker1"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin" />
            <Button
                  android:id="@+id/toInviteGuests"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"

      android:layout_marginRight="@dimen/activity_horizontal_margin"

      android:layout_marginBottom="@dimen/activity_vertical_margin"
                  android:onClick="toInviteGuests"
                  android:text="@string/InviteGuests" />
      </LinearLayout>
</ScrollView>
```

8.3e: selected_row.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:weightSum="8.0">
    <TextView
        android:id="@+id/txtExample"
        android:layout_width="266dp"
        android:layout_height="57dp"
        android:textColor="#000000"
        android:textSize="18sp" />
    <CheckBox
        android:id="@+id/checkbox"
        android:layout_width="0dip"
        android:layout_height="wrap_content"
        android:text=""
        android:layout_weight="6.0" />
</LinearLayout>
```

8.3f: upcomingevents.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:weightSum="8.0">
      <ListView
        android:id="@+id/upcominglist"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
    </ListView>
</LinearLayout>
```

8.3g: yourevents.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:weightSum="8.0">
    <ListView
        android:id="@+id/listView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </ListView>
</LinearLayout>
```

Santa Clara
University

Senior (Undergraduate) Student Thesis Publication Agreement

(SCU is Licensee)

Theses completed in partial fulfillment of an undergraduate degree must be deposited in the University Archives and made publicly available in the SCU Library. Furthermore, SCU has the right to copy, digitize and publish the title, author, and abstract of each thesis on the World Wide Web for SCU's Library use. Authors may choose whether or not to make their complete thesis publicly available on the World Wide Web, as set forth below.

1) Student Name/s ("Author/s"): Bryson Lam, Jeff Matsunaga, Matt Tu

2) Effective Date: 6/6/2013

3) Thesis Title (the "Thesis"): Eventify: An Android-based Event Planning Application

4) Open Access Publishing: Open access publishing means that the Author's complete Thesis will be available via the World Wide Web (as defined below) and not restricted to viewing only by library patrons, users of on-campus computers, and other SCU-authenticated readers.

a) ✓ Yes, I want to provide open access publishing of the Thesis via the World Wide Web and am therefore electing **electronic submission** of the Thesis. I understand that I will not be eligible to receive royalties.

b) ____ No, I do not want to provide open access publishing of the Thesis via the World Wide Web and am therefore electing **paper submission** of the Thesis. I understand that my name, and the title and abstract of the Thesis will still be available via the World Wide Web.

If Author has selected "Yes" above, please answer the following question:

c) ✓ No delay of publication, I want the Thesis to be available on the World Wide Web as soon as it is published.

d) ____ I want to delay publication of the Thesis on the World Wide Web. Delay publication of the Thesis on the World Wide Web for:

____ 6 month embargo; ____ 1 year embargo; ____ 2 year embargo.

Embargos start from the date on which the Thesis enters the library catalog. While under embargo, the Thesis will be available only to users of on-campus computers and other SCU-authenticated readers, but not to readers outside the SCU network. Once the embargo is lifted, all users will have access to the electronic version of the Thesis via the World Wide Web.

5) Copyright: Author acknowledges that he or she owns the copyright for the Thesis, except if the work has been developed in the course of or pursuant to a sponsored project or other agreement between the University and a third party. In such cases, the terms of the applicable third-party agreement shall govern the disposition of rights in copyright, and the Author may need to have the third party co-sign this publication agreement.

6) World Wide Web: For purposes of this Agreement, the "World Wide Web" means the World Wide Web, the Internet, and any successor or other computer or communication network or technology capable of electronically distributing content.

AUTHOR ACKNOWLEDGES THAT HE OR SHE HAS READ AND AGREES TO THIS PUBLICATION AGREEMENT AND THE ATTACHED TERMS AND CONDITIONS, AND THAT BY SIGNING BELOW, AUTHOR IS BOUND BY THE TERMS OF THIS PUBLICATION AGREEMENT AND THE ATTACHED TERMS AND CONDITIONS.

Dean of Degree-Granting School

Signed: ____________

Printed Name: Godfrey Mungal, Dean

Date: 6/7/2013

Author

Signed: ____________

Printed Name: Bryson Lam

Date: 6/6/2013

Additional Authors

Signed: ____________

Printed Name: Matt Tu

Date: 6/6/2013

Additional Authors

Signed: ____________

Printed Name: ____________

Date: ____________

Additional Authors

Signed: Jeff Matsunaga

Printed Name: Jeff Matsunaga

Date: 6/6/2013

Additional Authors

Signed: ____________

Printed Name: ____________

Date: ____________

Additional Authors

Signed: ____________

Printed Name: ____________

Date: ____________

Terms and Conditions

1) **License Grant**. Author hereby grants to SCU, the non-exclusive, royalty free, fully paid-up, irrevocable, unrestricted, transferable, sublicensable, worldwide and perpetual right to use, adapt, reproduce (including in digital format), distribute, prepare derivative works of, publicly perform, broadcast, and/or display the Thesis, in any and all media now known or hereafter devised, in connection with SCU's library use. Author acknowledges that the above license grant includes the right for SCU to include the title, abstract, bibliography and any other information necessary regarding the Thesis for inclusion in SCU's library's online catalog, which is available via the World Wide Web. SCU will limit publication in connection with the Author's restrictions set forth on the cover page of this Agreement. In the event the Thesis is not published by SCU in accordance with the restrictions set forth on the cover page of this Agreement, Author's sole remedy and SCU's sole liability, shall be for SCU to use commercially reasonable efforts to correct such error.

2) **Representations and Warranties of Author**. Author hereby represents and warrants that: (i) Author owns all right, title and interest in the Thesis; (ii) Author has the power and authority to enter into this Agreement and perform its obligations hereunder; (iii) the licensed Thesis will not violate any laws, regulations or ordinances, or the rights of any third party and will not give rise to any claims of such violation, including, without limitation, claims of libel, slander, defamation, copyright infringement, infringement of moral rights, trademark infringement, false designation of origin, disparagement, violation of privacy, publicity, identity or other proprietary rights, piracy or plagiarism; and (iv) to the extent that Author is required to obtain rights, licenses, permissions, clearances, approvals and/or attribution information necessary for SCU to utilize the Thesis, Author will do so accurately and completely, and the Thesis shall incorporate the necessary credit and/or attribution information. Author represents and warrants that it understands that SCU will rely on the contents of this Agreement and Author shall not have the right to enjoin the exploitation of the Thesis or to rescind any rights granted to SCU hereunder.

3) **No Approval**. Author hereby relinquishes any right to examine or approve any use, publication, modification, exhibition or other exploitation of the Thesis in connection with SCU's library use.

4) **Indemnification**. Licensee shall indemnify SCU, and its affiliated corporations, and all officers, directors, agents, employees, representatives and associates thereof, and save and hold each and all of them harmless of and from any and all loss, cost, damage, liability and expense, including attorneys' fees, with respect to any claim that the Thesis infringes any third party intellectual property rights or otherwise arising from any breach of the representations and warranties set forth in Section 2 (Representations and Warranties of Author).

5) **General**. This Agreement constitutes the final, complete and exclusive agreement between the parties with respect to the subject matter hereof, and supersedes any prior or contemporaneous agreement, either written or oral. This Agreement shall be governed by the laws of the State of California, without regard to its conflicts of law provisions. Both parties hereby consent to the exclusive jurisdiction and venue of the state and federal courts located in Santa Clara County, California. SCU may freely assign this Agreement. If any provision of this Agreement is held by a court of competent jurisdiction to be contrary to law, such provision will be changed and interpreted so as to best accomplish the objectives of the original provision to the fullest extent allowed by law, and if no feasible interpretation will save such provision, it shall be severed from this Agreement, and the remaining provisions remain in full force and effect. The failure of either party to enforce any provision of this Agreement shall in no way be construed to be a present or future waiver of such provision, nor in any way affect the right of either party to enforce such provision thereafter. The express waiver by either party of any provision of this Agreement shall not constitute a waiver of the other party's future obligation to comply with such provision.

Eventify: An Android Event Handling Application

By Jeff Matsunaga, Matt Tu, and Bryson Lam

Background

- Lack of event planning solutions
- Easy way to create events on the go
- Current Products
  - Facebook Events
  - Group Text Messaging

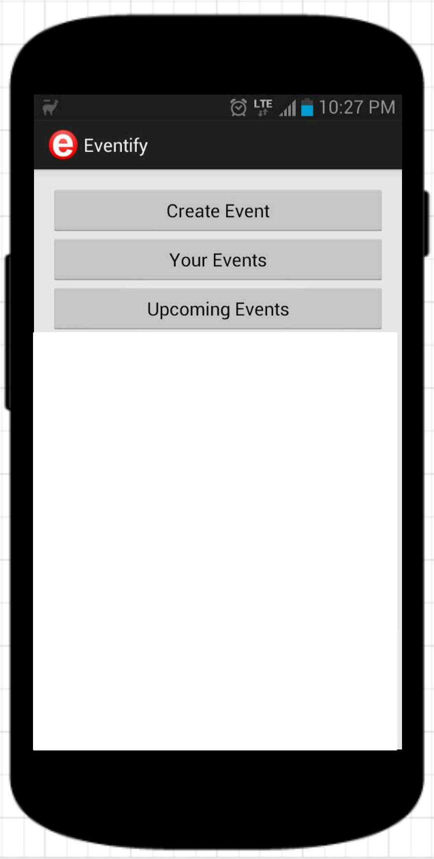
Existing Products

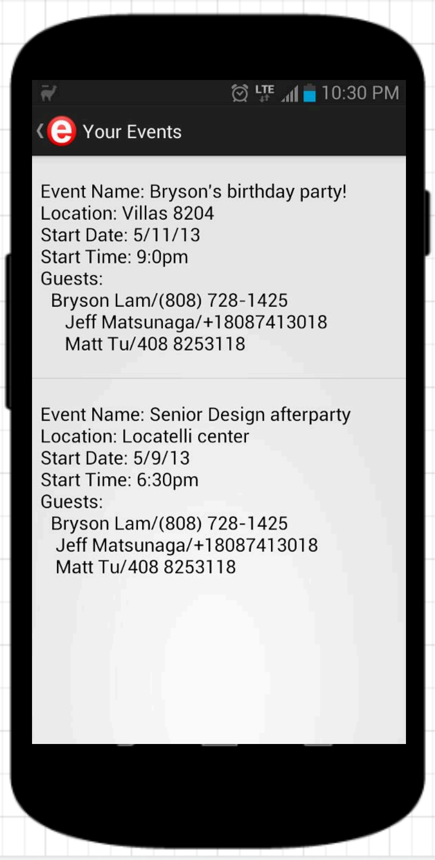
	Pros	Cons
Facebook Events	- Large user base - Suitable for large events	- Invitation Spam - Time-consuming process - Not useful for small events
Group Text Messaging	- Simple to use - Typical for smaller events	- Implemented differently on different operating systems - Does not scale for large group sizes

Our Solution

- Android-based application
- Lightweight
  - Should not consume excessive storage space
- Specifically designed for event planning
- Avoid use of a centralized server
  - Does not rely on internet access

Conceptual Model

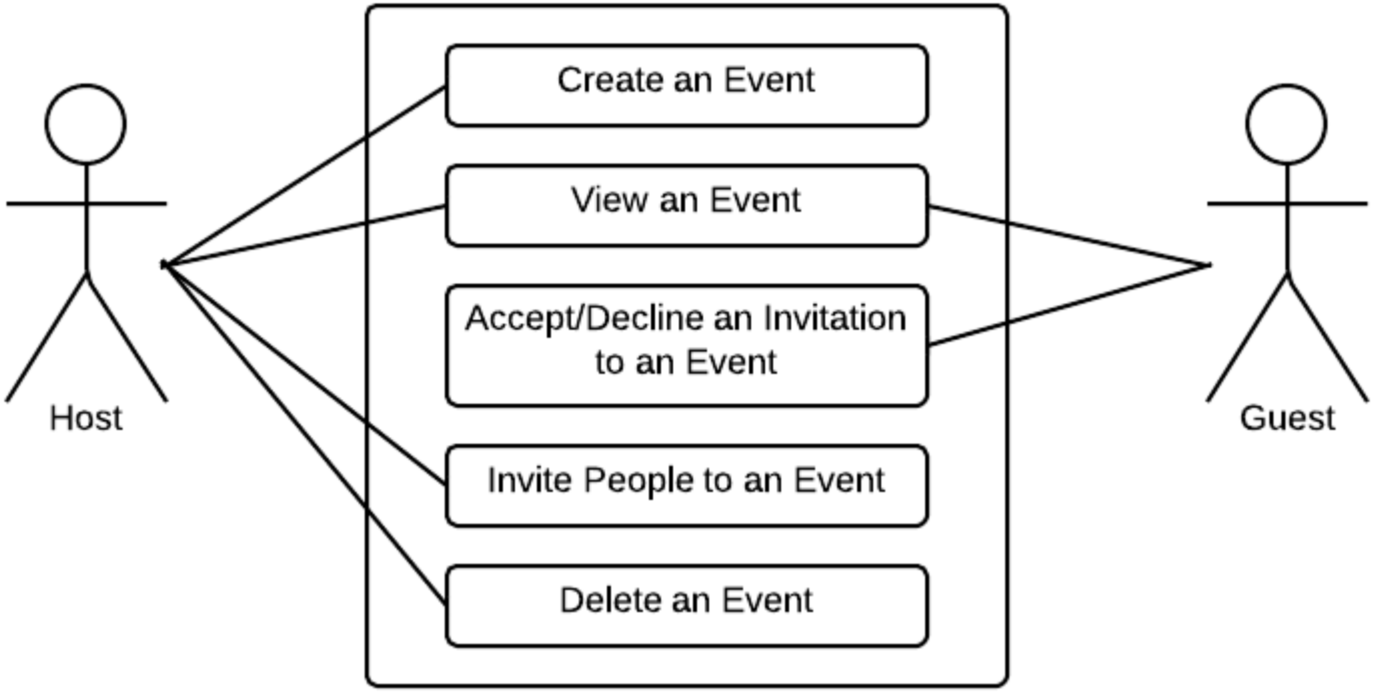




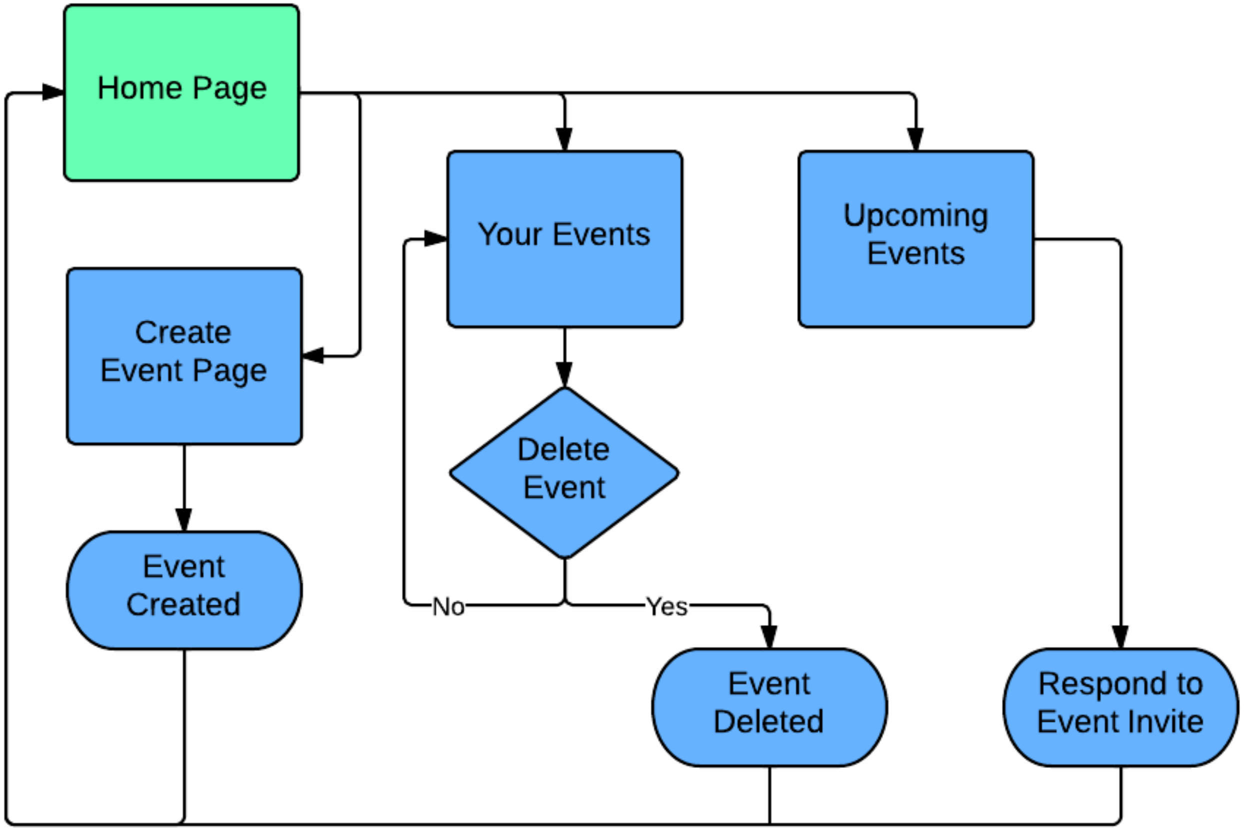
Event Name: Bryson's birthday party!
Location: Villas 8204
Start Date: 5/11/13
Start Time: 9:0pm
Guests:
Bryson Lam/(808) 728-1425
Jeff Matsunaga/+18087413018
Matt Tu/408 8253118

Event Name: Senior Design afterparty
Location: Locatelli center
Start Date: 5/9/13
Start Time: 6:30pm
Guests:
Bryson Lam/(808) 728-1425
Jeff Matsunaga/+18087413018
Matt Tu/408 8253118

Use Cases


Create an Event
View an Event
Accept/Decline an Invitation to an Event
Invite People to an Event
Delete an Event
Host
Guest

Activity Diagram



Requirements

Functional	Nonfunctional
• Event functionality • Basic create, invite, and deletion of events • Guest lists	• Friendly user interface • Able to accommodate different event sizes • More customizable event options

Design Constraints

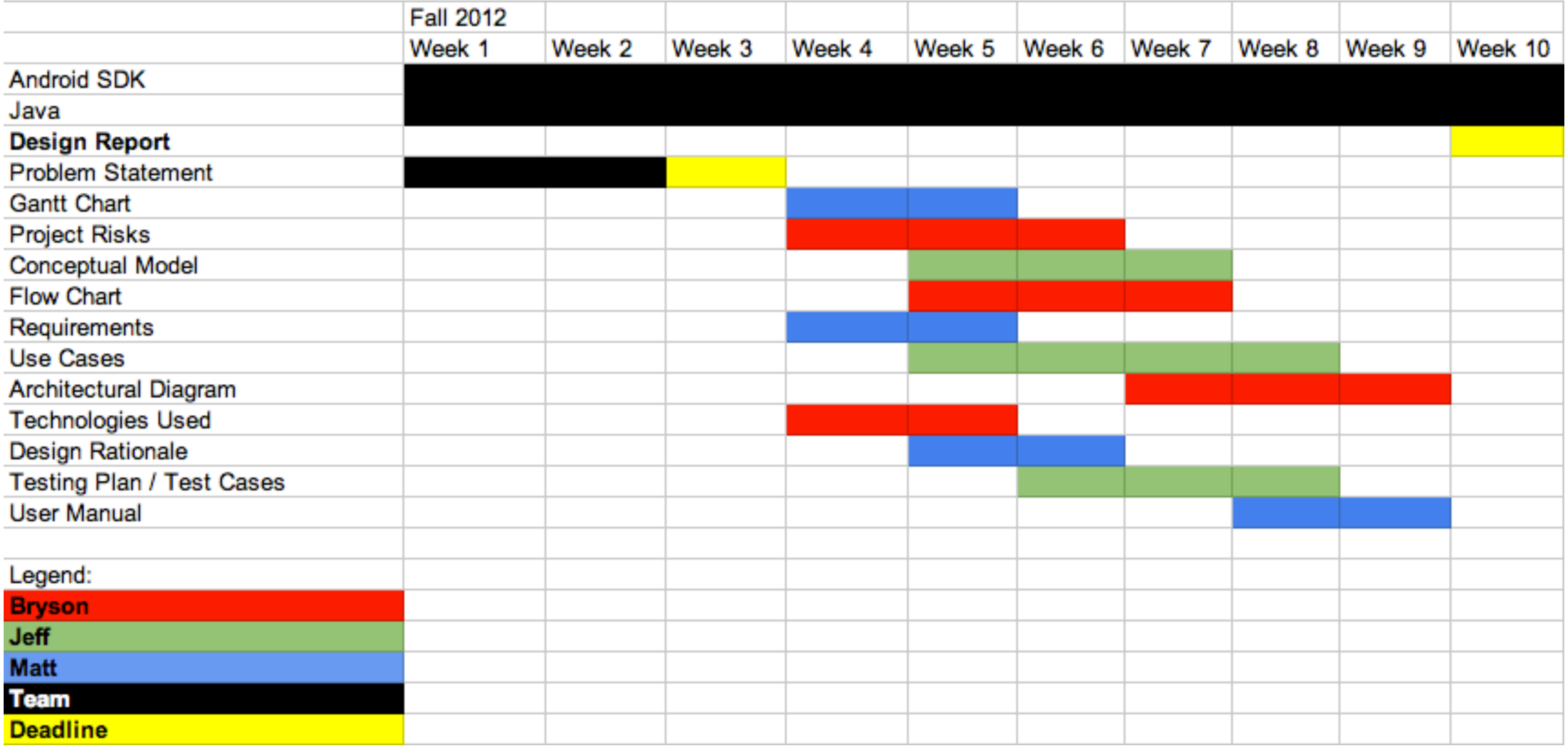
- Limitations of Android SDK
- Does not require a central server/database
  - Relies on internal file storage
- Does not require extensive user registration
  - Finds users based on the phone contacts

Project Risks

Risks	Consequences	Probability	Severity	Impact	Mitigation Strategy
Unfamiliarity with programming languages/SDK	Less development time	0.7	0.9	6.3	• Spend time in the beginning to become familiarized with tools
Failure to note changes	Unknowingly overwriting changes	0.6	0.5	3.0	• Comment code • Group meetings
Time Constraints	Features may not be implemented	0.3	7.0	2.1	• Adhere to project timeline • Delegate tasks
Sickness	Less productivity	0.4	5.0	2.0	• Redistribute tasks

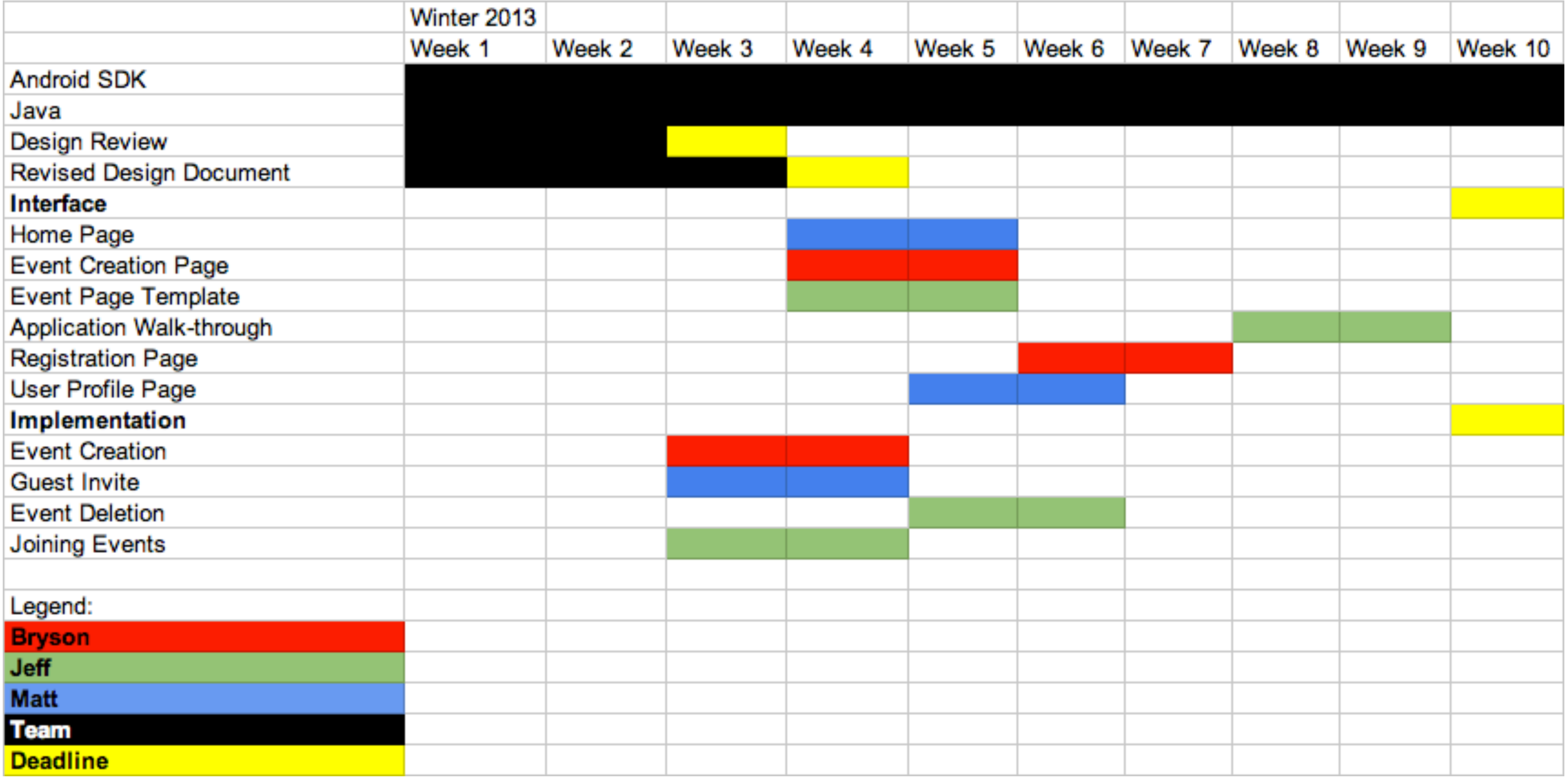
Probability x Severity = Impact

Developmental Timeline (Fall Qtr.)



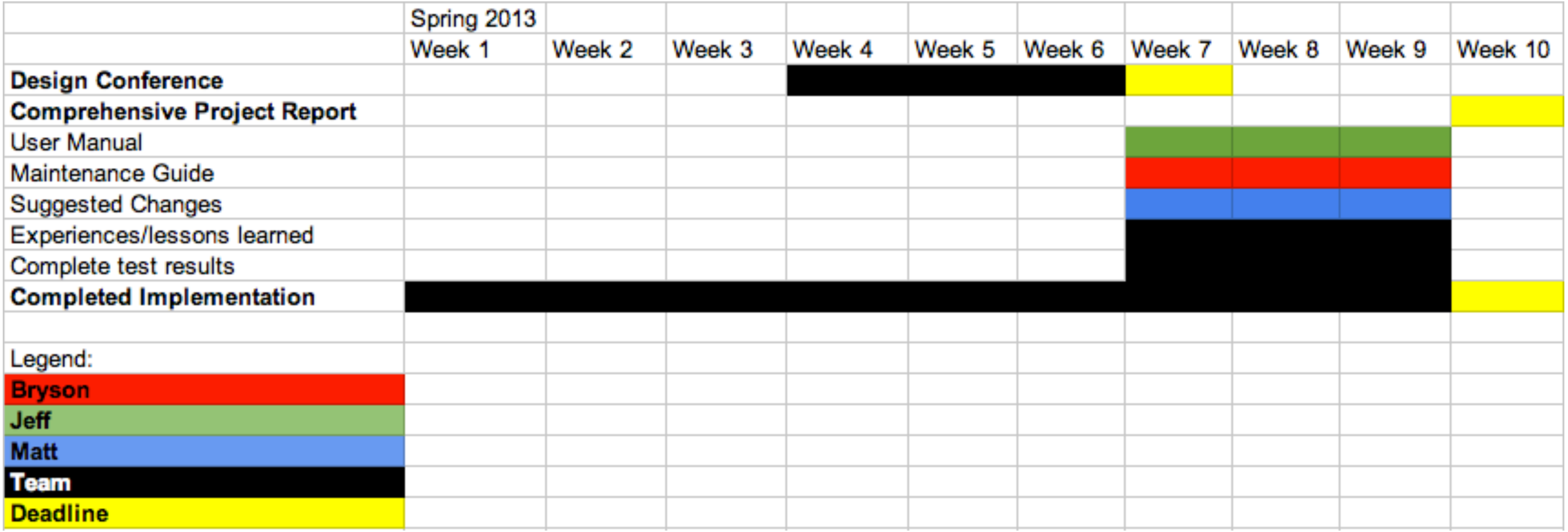
	Fall 2012									
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Android SDK	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Java	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Design Report										🟨
Problem Statement	⬛	⬛	🟨							
Gantt Chart				🟦	🟦					
Project Risks				🟥	🟥	🟥				
Conceptual Model					🟩	🟩	🟩			
Flow Chart					🟥	🟥	🟥			
Requirements				🟦	🟦					
Use Cases					🟩	🟩	🟩	🟩		
Architectural Diagram							🟥	🟥	🟥	
Technologies Used				🟥	🟥					
Design Rationale					🟦	🟦				
Testing Plan / Test Cases						🟩	🟩	🟩		
User Manual								🟦	🟦	
Legend:										
🟥 **Bryson**										
🟩 **Jeff**										
🟦 **Matt**										
⬛ **Team**										
🟨 **Deadline**										

Developmental Timeline (Winter Qtr.)



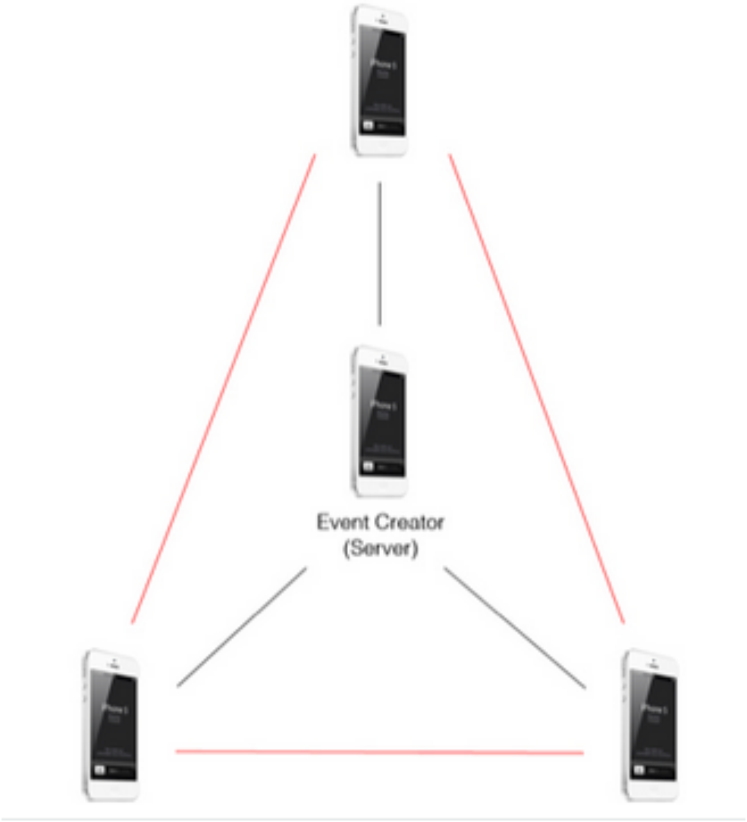
	Winter 2013									
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Android SDK	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Java	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
Design Review	⬛	⬛	🟨							
Revised Design Document	⬛	⬛	⬛	🟨						
Interface										🟨
Home Page				🟦	🟦					
Event Creation Page				🟥	🟥					
Event Page Template				🟩	🟩					
Application Walk-through								🟩	🟩	
Registration Page						🟥	🟥			
User Profile Page					🟦	🟦				
Implementation										🟨
Event Creation			🟥	🟥						
Guest Invite			🟦	🟦						
Event Deletion					🟩	🟩				
Joining Events			🟩	🟩						
Legend:										
🟥 **Bryson**										
🟩 **Jeff**										
🟦 **Matt**										
⬛ **Team**										
🟨 **Deadline**										

Developmental Timeline (Spring Qtr.)



	Spring 2013									
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
Design Conference				⬛	⬛	⬛	🟨			
Comprehensive Project Report										🟨
User Manual							🟩	🟩	🟩	
Maintenance Guide							🟥	🟥	🟥	
Suggested Changes							🟦	🟦	🟦	
Experiences/lessons learned							⬛	⬛	⬛	
Complete test results							⬛	⬛	⬛	
Completed Implementation	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	🟨
Legend:										
🟥 **Bryson**										
🟩 **Jeff**										
🟦 **Matt**										
⬛ **Team**										
🟨 **Deadline**										

Architectural Diagram



Technologies Used

- Android SDK
  - Java
  - XML
  - Internal file storage
- Cellular Network

Design Rationale

- **Network Topology**
  - Peer-to-peer network
  - Phone numbers as addresses
  - Event creator = server; event attendees = clients
- **Message Queuing**
  - Ensuring messages are sent and received
  - Handled by the cellular network

Live Demo

Future Features

- Location and navigation support
- Updates via notifications
- Event modification capabilities
- Event data polls

Lessons Learned

- Having many features can be difficult to implement
- Expand upon features in different releases
- Proof of concept: Cellular network-based application is possible for Android phones

Questions?