Modern Event App – Research Document

October 11th 2015

Mark Lordan – X00100551

Contents

[Detailed Discussion 1](#_Toc432340830)

[Existing Applications in this domain 2](#_Toc432340831)

[Platform, Technologies and Libraries 3](#_Toc432340832)

[Potential Platforms 3](#_Toc432340833)

[Android 3](#_Toc432340834)

[Microsoft Azure 3](#_Toc432340835)

[Development Environments 6](#_Toc432340836)

[Android Studio 6](#_Toc432340837)

[Visual Studio 6](#_Toc432340838)

[Libraries and APIs 6](#_Toc432340839)

[Google APIs – Maps, Calendar, Places, Cloud Messaging 6](#_Toc432340840)

[Git and Node.js 6](#_Toc432340841)

[Android UI elements 6](#_Toc432340842)

[The risks 7](#_Toc432340843)

[Bibliography 8](#_Toc432340844)

# Detailed Discussion

This project is a native Android application, designed to be used by anyone to connect with people with similar interests via events. Rather than using Facebook events which are mostly ‘click to attend’ functionality, the proposed application will add a competitive element to attending events, allow advanced customization of individual event pages, and personalize user experiences based on preferences.

This application will require users to create an account to enable the full application experience, but users can skip this authentication to preview events; to interact an account must be made. An alternative to creating an account would be application authentication from Google, Twitter or Facebook. Once registered, users can alter what information is available to event attendees, such as public display name (full name or use custom username) and contact information. User preferences are also chosen at this stage to provide targeted events to each user, a method to reduce the amount of events which may not be desired by the user. All of these personal settings can be altered afterwards from the application if a change of mind is made.

The primary interaction for users is the event overview list, which gives a snippet of information about events based on location, preferences, or both. The location setting can be tuned from close-by to worldwide to allow global access. Events themselves can also have a proximity setting, where only users within a certain distance can view or join the event. Other settings such as public, private, invite-only or request-to-join can also be applied to events. The search function within the application can search for events using a variety of different criteria, and can also be used to search for other users. For each event, the attendees can be viewed, any information the host has created can be viewed, and the option to attend is also available here. All attending events are saved to a ‘My events’ section of the app, which shows created and attending events for the user.

Once a user selects an event to attend, a chat system will be available for event attendees to communicate between one another. Upon attending an event, the host of the event can confirm attendees using a close-proximity attending feature. NFC where available can be used to confirm attendees, which then increases the user’s score / rank within the app. This ranking system will add a competitive aspect to events, giving an extra incentive to attend.

# Existing Applications in this domain

The event organizer domain is extensive, with well-established options already in place. Facebook have an Events feature which allows for basic events to be made and joined. This is probably one of the most used event features in the world, however it is very basic in what it offers users. Many users of Facebook are pestered constantly with event invitations, and the turnout for events is usually less than what is indicated on the event page. Eventbrite is an alternative to Facebook events, but it is heavily integrated with Facebook by connections to contacts and friends via the social network. Using this application without Facebook becomes frustrating as there are messages on almost every page regarding ‘Connect to Facebook’. The general usage of Eventbrite is quite good, and it also includes payment transactions for paid events. This is a feature which could be implemented in the project to give users a better experience.

Meetup is another example of an event-based community finder, which allows users to customize their experience based on very specific interests. The suggestions based on interests are very accurate, however the features and the usability of the application are questionable. Each community has very basic information, with little to no differences between each other by showing a simple list of upcoming and past events. Events themselves are also identical, with an RSVP button and other small snippets of information. Reviews on the Google Play store reveal that many users of the application enjoy it, but it lacks custom details for events and communities. The final existing application is Nvite, a currently invite-only event system for the iOS platform. Nvite is marketed as an advanced event management application, with a focus on analytics and sales where Nvite take a percentage of each sale. The application is shown to be easy to use and somewhat dynamic, a direction which I want to expand on further.

These are the major existing applications similar to the proposed project, however I feel that they lack the customization abilities that I aim to include for each event. Giving event organizers the ability to add or remove modules to the event page will provide a variation to users that will allow better interaction and integration with other attendees.

# Platform, Technologies and Libraries

## Potential Platforms

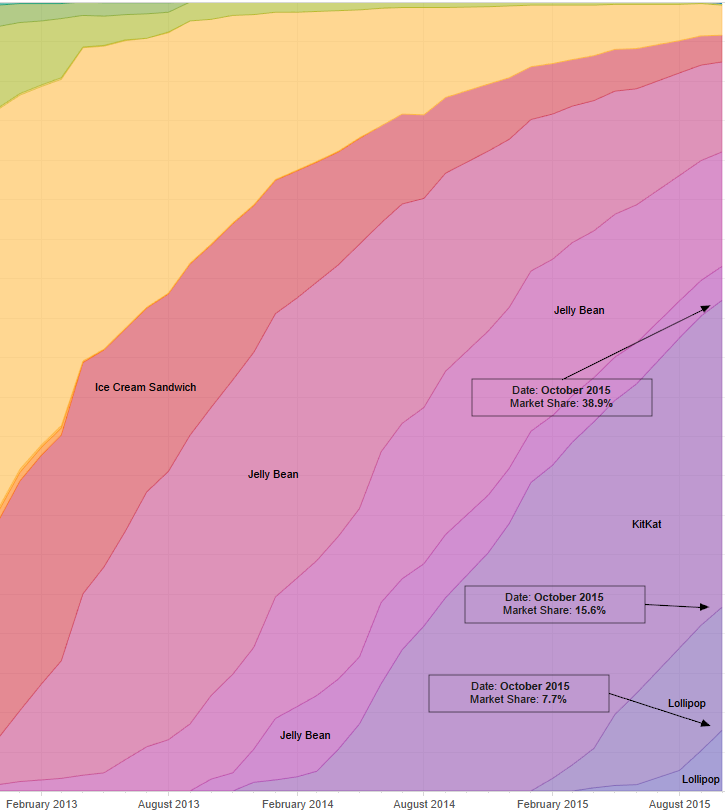
### Android

The smartphone market is widely dominated by the Android operating system, estimated at 82.8% for the second quarter of 2015. Due to this huge market share, developing the application for Android would be an obvious choice, along with my previous knowledge of the Android platform from last year. The Android version for development is another factor when developing for Android, as the Android market itself is currently split between nine different versions (those that are above 0.1% market share) From my Interactive Media Design project, the chart on the following page illustrates the historical distribution of all Android versions. The combined Android market share from KitKat and above total 62.2%, which indicates that KitKat would be the optimal version to target. A higher target results in a drop of compatible devices by almost 40%, reducing the potential user base significantly.

### Microsoft Azure

Although I will be developing for Android, a large portion of the project will be located on Microsoft Azure, where a large part of the complexity will reside. In my 3rd year project, we used Azure in a similar way, but we were unaware of the potential it had. The majority of data operations were then performed on the Android device, where the application was waiting on certain processes to finish. Multiple database calls were being performed directly from the client, and the result of which was a huge number of problems with retrieving data within a certain timeframe. It was only towards the end of the project did we utilize the API features available, which acted as an intermediary between the database and the client. From here, we could write much more complex functions and perform the majority of computational processes, with a single call and a single JSON return element to and from the client.

As the application backend is available via HTTP GET and POST requests, the client platform is generally unimportant. A client application could also be written in C# (or WinJS) for Windows, or Objective-C / Swift for iOS and these clients could implement the Azure backend easily. Due to the scale of work for each platform, I think a cross-platform solution would not be completed successfully in the time-frame, but it is something which I would definitely attempt if I were to continue development of the project.



Android historical version distribution. Made with Tableau software. Data from Google Developer Dashboard and Bidouille.org

## Development Environments

### Android Studio

Android Studio is now out of its beta stage, and is currently at version 1.4 in the stable release channel. It is the official Android IDE, and is widely recommended over alternatives such as Eclipse. Android Studio provides a large number of features, such as a WYSIWYG XML editor, source control integration, advanced debugging and code analysis, along with many other features.

### Visual Studio

I will be using Visual Studio for certain elements of the Azure cloud backend, such as the SQL Server Tools that allow me to connect directly to the Azure SQL database.

## Libraries and APIs

### Google APIs – Maps, Calendar, Places, Cloud Messaging

A number of popular Google APIs will be used to integrate a number of functions into each event. Users will be able to get directions or view the venue using the Maps / Places API, and can add event times to their calendar using the Calendar API. An event-based chat system is a feature to allow for quick and easy communication between attendees and the host(s), and Google Cloud Messaging (GCM) could be used to achieve this. I have been researching alternatives to using GCM in the event that this cannot be used.

### Git and Node.js

Microsoft Azure implements source control for Mobile Services, which I can use to checkout, create additional scripts, and push the changes back to the cloud. The Mobile Service I’m using will use a JavaScript (Node.js) backend which I have previous experience with and an understanding of.

### Android UI elements

Certain UI elements may require additional libraries to achieve the desired layout, in which case they will be added as a dependency in the build.gradle file. These dependencies will allow the usage of particular elements, such as a navigation drawer or a specific style. All of these libraries will be from GitHub for transparency. The AppCompat library package will be used to implement Fragments, ViewPagers, TabStrips and other elements, which are core components to the application layout.

# The risks

There are a number of risks to this project which must be considered, particularly the scope and size of the system.

I believe Azure could be a risk to the usability of the application, due to the nature in which applications are loaded and unloaded to and from the running state. While in the free tier of any service, Website, Mobile Service etc. Azure unloads services based on recent activity, i.e if there has been a period of time where no requests have been made to the application, it is unloaded to save resources for other running and active applications. Although this process saves resources, it causes the first request to the application to take a long period of time to complete as the application must be reloaded back into a running state. After some basic testing, I encountered this problem on Azure websites and Mobile Services after an idle time of between 5 – 10 minutes. The first request to the service after this period of time took approximately 25 seconds to complete, which would be an incredibly long wait time for users of the mobile application. A workaround to this would be to create a scheduler with a job to access a HTTP method of the service every 5 minutes to refresh the application and keep it loaded in memory. Azure limits the free-tier scheduled jobs to every hour, to schedule a job for less than that requires the standard paid tier. A mobile service running in the standard tier also resolves this issue, however costs play into both so a comparison must be made on the price differences.

Proximity-based confirmation may be a difficult task to complete, as I initially planned to use NFC to pass a confirmation message between hosts and attendees. Any clients without NFC would be missing out on this feature, so an alternative would be needed. I could try and implement a Wi-Fi or Bluetooth connection feature, but this may not solve the issue. The most relevant option for this problem is Nearby, a proximity-based API from Google that appears to send messages based on proximity to other devices.

Google Cloud Messaging may have certain issues when it comes to enabling a chat system, as the guides in the Google Dev console does not have a straightforward guide to client-to-client messaging. If GCM cannot support this feature, an alternative will be needed. Nick Landry, a Microsoft Technical Evangelist created a cross-platform chat application using Azure as the backend, which could provide an insightful resource for creating my own chat system within Azure and Android.

# Bibliography

1. Eventbrite App (2015) [online] Available at:

<https://play.google.com/store/apps/details?id=com.eventbrite.attendee&hl=en>

[Accessed 11th October 2015]

1. Meetup (2015) [online] Available at:

<https://play.google.com/store/apps/details?id=com.meetup&hl=en>

[Accessed 11th October 2015]

1. Nvite (2015) [online] Available at:

<https://nvite.com/> [Accessed 11th October 2015]

1. IDC, (2015). *Smartphone OS Market Share, 2015 Q2* [online] Available at:

<http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

[Accessed 11th October 2015]

1. Google Android Developer Dashboard (2015) [online] Available at:

<https://developer.android.com/about/dashboards/index.html>

[Accessed 11th October 2015]

1. Bidouille (2015) [online] Available at:

<http://www.bidouille.org/misc/androidcharts>

[Accessed 11th October 2015]

1. Glenn Gailey, Azure (2015) *Store your mobile service project code in source control* [online] Available at:

<https://azure.microsoft.com/en-us/documentation/articles/mobile-services-store-scripts-source-control/> [Accessed 11th October 2015]

1. AppCompat Library (2015) [online] Available at:

<https://developer.android.com/tools/support-library/features.html>

[Accessed 11th October 2015]

1. Simon J.K. Pendersen (2014) *Windows Azure WebSites and Cloud Services Slow on First Request* [online] Available at:

<http://wp.sjkp.dk/windows-azure-websites-and-cloud-services-slow-on-first-request/>

[Accessed 11th October 2015]

1. Nick Landry (2014) *AzureChatR: Building a Cross-Platform Chat App* [online]Available at:

<https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/Azure-ChatR-Building-a-Cross-Platform-Chat-App> [Accessed 11th October 2015]