Plant view – an augmented reality android application

Matthew Frost – L1426439

Contents

[Introduction 1](#_Toc477162717)

[Abstract 1](#_Toc477162718)

[Rationale 1](#_Toc477162719)

[Ethical, Legal and Social Issues 2](#_Toc477162720)

[Design 2](#_Toc477162721)

[Requirements 2](#_Toc477162722)

[Must 2](#_Toc477162723)

[Should 2](#_Toc477162724)

[Could 3](#_Toc477162725)

[Won’t have this time 3](#_Toc477162726)

# Introduction

## Abstract

An augmented reality Android application that displays information relevant to the user depending on their location. The application was originally developed for a client that operated on an industrial site, so the data collected would be relevant to their use case, such as the temperature of a pipe. However, the app is generalised enough that it can work with any numerical data set, for example the energy usage of buildings at the university. The data is displayed on a graph to show how it changes over time and statistical analysis is applied to highlight any anomalies in the data.

To find the location of the user, the Android device’s GPS is utilised to allow the acquisition of the device’s latitude and longitude to find the position and the bearing to find which direction it is facing.

A separate Google Maps web application has been developed to allow the mapping of locations against data in an SQL database. Both the Android app and web app communicate with the data sources using Node JS web services. The web services are used to store and retrieve location points as well as pull the data for each location wherever it is stored.

## Rationale

The project proposal was initially provided by a local company, Sabisu, who develop reporting tools for customers within the oil and gas industry. They were looking for an augmented reality Android application that would allow a user on an industrial site to walk around with a tablet and view information on what was around them. Sabisu also asked for a web application that would allow points of interest to be plotted on a map, this would be used by admin users.

There were not many technical limitations put in place within the project proposal from Sabisu, all they asked for was an augmented reality Android application, and an admin web application to go along with it. This meant that the project could be approached with any methods or technologies that would best suit the given task.

Furthermore, this project allows for a range of technologies to be used including Android, Node JS and SQL. This would help with employability as both web applications and mobile applications are what the current market is moving towards.

## Ethical, Legal and Social Issues

Since this project involves a real client it is important that the British Computer Society code of conduct (British Computing Society, 2017)is adhered to. The section that would apply specifically would be “professional competence and integrity” meaning that work should only be undertaken that I think I am competent of and I should be willing to accept criticisms and alternative viewpoints.

As my application is intended to be used within the chemical industry it could be reporting some critical information to the user. However, as the application is only reading the information from a data source that has been inputted by another system the application does not have any ethical issues in this regard.

# Research & Analysis

## Justification of choices

### Android Development Language

For Android development there are a number of language and libraries that can be used. After some research there were a number of languages that seemed to be popular and potentially suitable for the project, these included:

* C/C++ (Android Native Development Kit (NDK))
* Java (Android Software Development Kit (SDK))
* Kotlin (Android SDK)
* C#/Vuforia (Unity)

Since the Android operating system is built in C and C++ it is possible to create applications for Android in these languages. Applications built using the NDK are often faster than those written in a Java based language as they do not need to run on the Java Virtual Machine (JVM). However, for this project the speed limitations of the JVM are not going to be an issue and the lack of support for the NDK compared to the Java SDK mean that C and C++ were ruled out for the choice of Android development language.

During the initial research period the head of Sabisu suggested that the Unity engine could provide a solid method for creating augmented reality applications. With some research, it was found that there is a library for Unity called Vuforia which provides an easy way to add augmented reality and image recognition to an application. However, for the image recognition to work it had to have knowledge of the shape beforehand and shapes had to be complex to enable more accurate recognition. As this application would be looking at 3D shapes such as tanks and pipes it was clear that using Vuforia would not work.

## Requirements

The initial proposal provided contained a number of requirements, some were necessary and some were just “nice to haves”. After analysing the requirements provided, it was clear that it would not be possible to complete them all in time and some of them would not be possible to work on outside of the Sabisu offices. Therefore, it was necessary to remove any requirements that would not be feasible as well as anything that would not be implemented in time and would not affect the end product too much. To categorise and prioritise the requirements the MoSCoW (Must have, Should have, Could have, Will not have) system was used, this ensured that if all of the minimum requirements were met then a shippable product would be produced. Requirements that come under “must have” are requirements that would make the project useless if they were not included. Requirements that come under “should have” are requirements that are important to the project but are not critical to success. Requirements that come under “could have” are “nice to haves”, they may add more polish or functionality to the project but are less important than “should have” requirements. Anything that falls under “will not have” will not be done in this release, they are items that are feasible but will not be able to be included in the given time period.

For example, one of the requirements was to have it connect with the Sabisu platform to integrate with some of their APIs. However, this would require having a VPN for their network during development and it is not a feature that is necessary for the application to work.

### Must

* The web application must allow the user to create, edit and delete locations
* The web application must display all created locations on the Google map interface
* The web service must return all created locations that have not been deleted
* The web service must be able to return all locations within one-hundred meters of a given latitude and longitude
* The web service must allow locations to be created, edited and deleted in the database
* The web service must be able to return all data associated with a given location
* The Android application must use the device’s GPS to get the current location
* The Android application must access the device’s camera to create an augmented reality application.
* The Android application must interact with the web service to get the locations near the device
* The Android application must retrieve the data associated with a given location
* The Android application must display the retrieved information on a graph

### Should

* The Android application should perform analytics on the data retrieved to identify any anomalies in the data
* The Android application should use the device’s compass to get the direction the device is facing

### Could

* The Android application could implement QR code scanning to retrieve information about the location associated with that QR code
* The Android application could have offline capabilities to store the data if internet connectivity is lost

### Won’t have this time

* The Android application won’t have connectivity to enable Sabisu log ins
* The Android application won’t use the Sabisu API’s

# Design

# Implementation

# Testing & Evaluation

# Reflection

# References

# Appendices