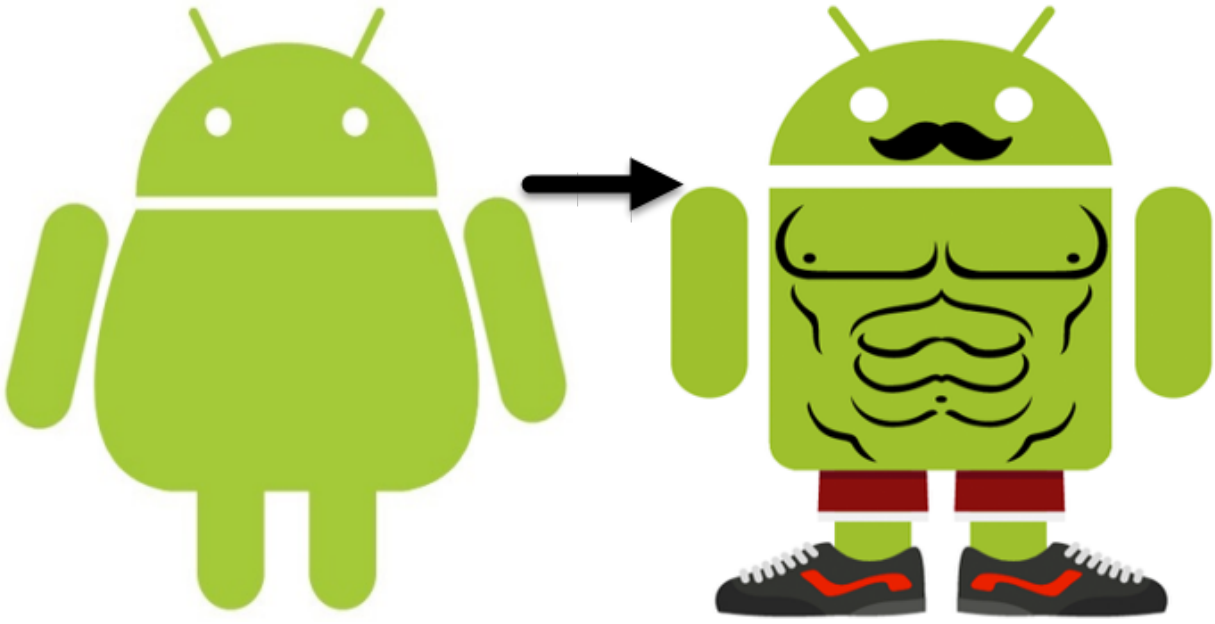

Figure 3

Technical Architecture Document
Android Fitness Application

Adam Flood
x00110480

1. Use Cases

Use Case One

The following use cases listed are subject to change and may or may not be included in the prototypes.

1.1 Title: User Registration

1.2 Primary Actor/Actors: User / Azure

1.3 Scope: Azure Database

1.4 Level: User goal

1.5 (Story)
The user will click on the register button in the application and will fill in the relevant fields to register to the application.

Use Case Two

1.1 Title(goal): User Login

1.2 Primary Actor/Actors: User / Azure

1.3 Scope: Azure Database

1.4 Level: User Goal

1.5 (Story)
The user will click on the login button when they wish to login. However, the login action can only be completed once the user has registered to the system

Use Case Three

1.1 Title(goal): Adding to a to-do list

1.2 Primary Actor: User / Azure

1.3 Scope: Azure Database

1.4 Level: User Goal

1.5 (Story)
The user can add items to a to-do list to keep track of tasks that need to be completed.
The tasks are sent to to azure sql database and are added to a to-do items table.
The value is then returned to the application and displayed underneath the text box with an optional check box beside it so when the user checks the box the item will be removed from the list.

Use Case Four

1.1 Title(goal):Deleting from a list

1.2 Primary Actor: User / Database

1.3 Scope: Azure Database

1.4 Level: User Goal

1.5 (Story)
The user will be able to delete items from a list or a table once they are added.

Use Case Five

1.1 Title(goal): Updating a list

1.2 Primary Actor: User / Database / Azure

1.3 Scope: Azure Database

1.4 Level: User Goal

1.5 (Story)
The user will be able to update an item once it has been added to a list or table.

Use Case Six

1.1 Title(goal): User Authentication

1.2 Primary Actor: User / Facebook / Azure

1.3 Scope: The internet / Azure

1.4 Level: Summary / User Goal

1.5 (Story)
The user will be authenticated with their choice of either facebook or google and be given a unique ID.

2. Technical Architecture

2.1 Software Components

Databases, app engines, mobile platforms

-Android Studio
-Volley
-Java
-Android Mobile Platform
-Azure Cloud Sql database
-Azure Mobile web services

2.2 Platform Libraries

APIs and languages (per component)
Azure Easy API's - used for implementing API's into the application easier

Google Fitness API - consists of multiple API's (xml + java)

-Sensors API - provides access to raw sensor data streams from sensors on the android device.

-Recording API - provides automated storage of fitness data using subscriptions.

-History API - provides access to the fitness history and lets apps perform bulk operations.

-Sessions API - provides functionality to store fitness data with session metadata.

-Bluetooth Low Energy API - provides access to low energy bluetooth sensors.

-Config API provides custom data types and additional settings for Google fit.

Google Maps Api - To build custom maps using multiple map types.(xml + java)
Google Maps Directions API - To enable direction functionality to plan routes(xml + java)
Barcode API - To scan barcodes to gain information(xml + java)
Azure Web API.

Some API's will be added in time as I may need more functionality for certain purposes.
I will also be trying to code my own API in C# using the entity framework (or implement one) that will deploy the application in visual studio and connect to azure. The entity framework supports the development of the API.

2.3 Distribution and Deployment

1. Cloud, RESTful, JDBC, sessionless etc

-Azure Cloud
-Azure SQL cloud database
-Azure Mobile services
-Visual studio Deployment using an API
-Restful service
-Web API

The below diagram (fig.1) depicts the structure of client applications and windows azure services.
The client application in my project will be an android device and it will communicate with windows azure mobile services to use SQL databases and to also provide user authentication such as using the application with facebook or google.

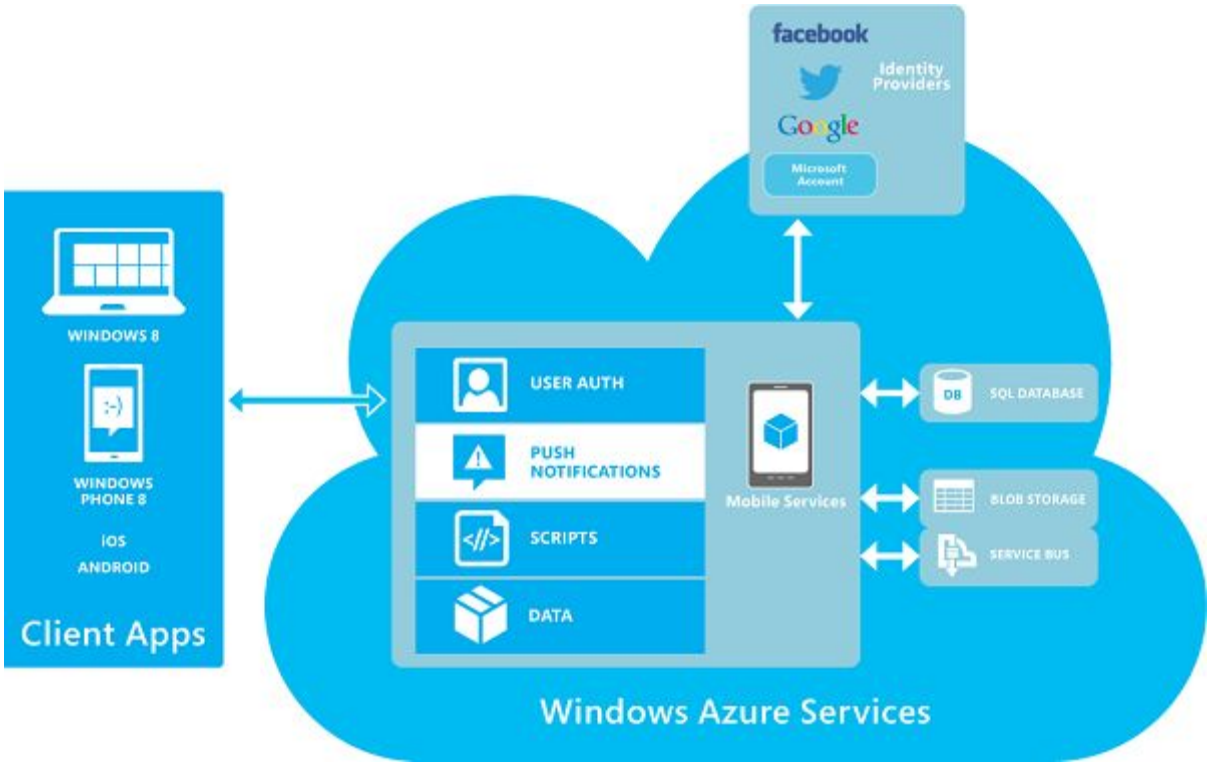

Figure 1

2. Security eg. HTTPS, certificates, authentication, etc.

-Encrypt passwords
-Facebook authentication
-Google authentication
-More security methods to be added in time

Basically, once the service is deployed it will listen through HTTP for the usual requests such as GET, PUT, POST and DELETE from the client(android application). The client will get authenticated with google or facebook. This happens by the client receiving a unique ID that gets sent to the database and gets stored as an ID for that particular user. When a new user logs in a token is generated and the user is sent to be saved into the database using PUT over HTTP, which can be done using the volley android library.

The RESTful service will be listening, awaiting the PUT request so it can store the request into the database, this RESTful service may also retrieve the request using GET.

The below diagram(fig.2) demonstrates this authorization usage further by showing the clients access to the app service authentication functionality and their user code.

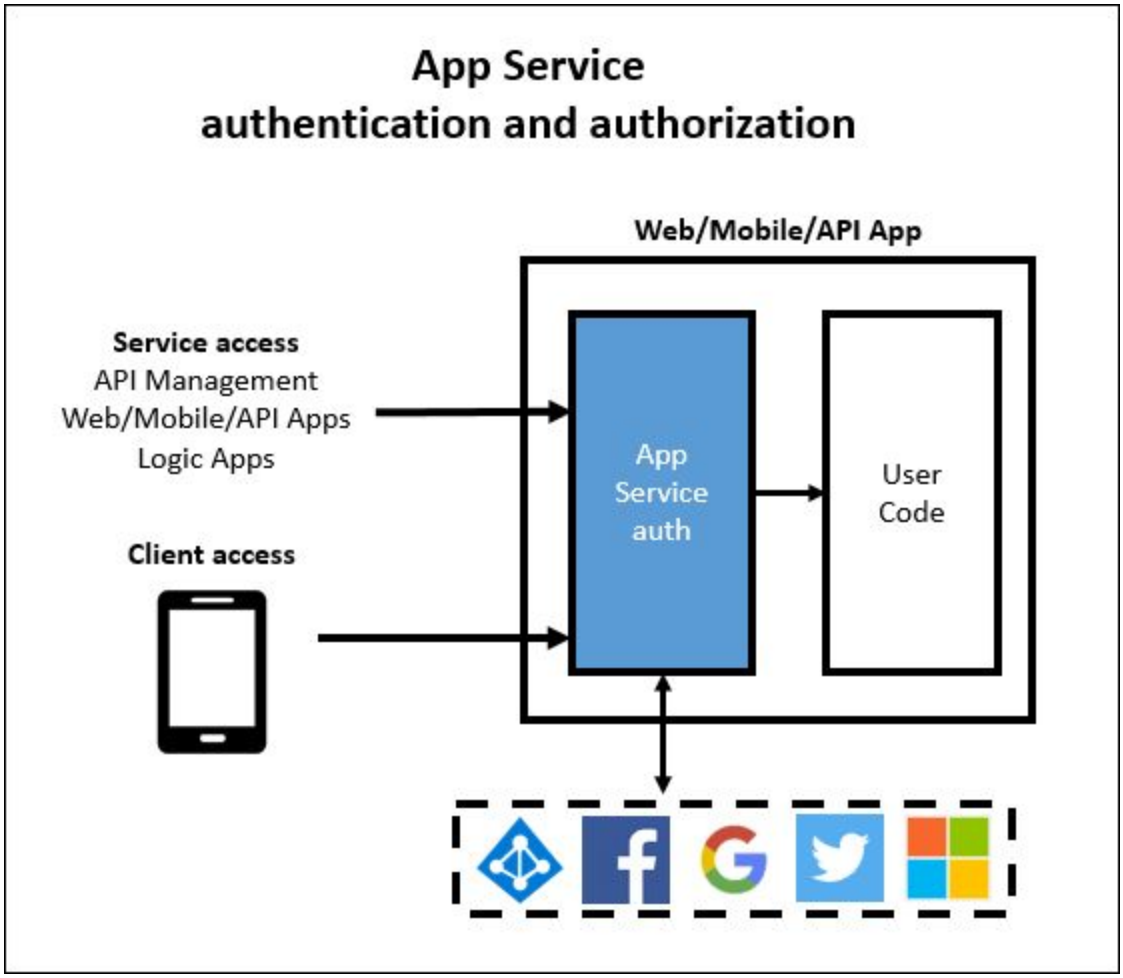

Figure 2

2.4 Risks

Discuss the risks which may affect the delivery of the project

There are many risks that affect the delivery of the project, one of which is the fact that I am using applications and services that I have not yet used during my time at the college. Implementing new technologies such as API's, cloud services, online databases and android studio can be excellent for project development but can ultimately contribute to the downfall. At the moment, implementing different services is very challenging as I am unfamiliar with them especially the backend architecture.

Another risk is that I am using multiple libraries. From gathering data and reading forums on libraries, sometimes libraries are not compatible with others and will throw errors when combined into one project. The issue with this is that an error may not get displayed and the application will continue to run like nothing happened when really some libraries are not even being used.

One of the main risks to this project in my opinion is myself. Each time I implement functionality I may be thinking “how do I improve this?” when really the functionality I originally outlined is fully working and this can over-complicate things and cause issues with the project. Time management will be impacted in particular as I need to progress efficiently throughout the development phase to meet deadlines.

3. Prototype

3.1 Prototype deliverable for week 8

List the use cases to be delivered in the first prototype and discuss the testing strategy.

The use cases that are to be developed for the first simple prototype are the following:

Adding to a to-do list: Once access has been obtained to the application , the user can now access the to-do list function available in the application to add tasks.

Although this is the main “use case” that will be developed in this prototype I will also be making some sort of skeleton version of my application that has multiple UI screens that the user can navigate between but do not provide functionality as of yet.

The testing strategy for the first prototype will be some unit testing along with collecting feedback from friends that I will ask to test my application.

3.2 Prototype deliverable for week 11

List the use cases to be delivered in the second prototype and discuss the testing strategy.

The use cases that are to be developed for the second prototype are the following:

User Registration: The user will click on the register button in the application and will fill in the relevant fields to register to the application.

User Login:The user will click on the login button when they wish to login. However, the login action can only be completed once the user has registered to the system

Deleting from a list: The user will be able to delete items from a list or a table once they are added.

Updating a list: The user will be able to update an item once it has been added to a list or table.

User Authentication: The user will be authenticated with their choice of either facebook or google and be given a unique ID.

Some of these use cases may not be in the week 11 prototype depending on the difficulty of implementing them but if they are not implemented they will be added over the christmas break. I would also like to add more screens to the skeleton version of the app to try and provide a full playthrough of the app without functionality.

4. References

Using the harvard referencing system.

Figure 3
DailyTUT | Technology Blog. 2016. *10 Best Fitness Apps for Android Smartphone Users*. [ONLINE] Available at:https://www.dailytut.com/apps-tech/fitness-apps-android.html. [Accessed 26 October2016].

Figure 2
Microsoft Azure. 2016. *Authentication and authorization for API Apps in Azure App Service | Microsoft Azure*. [ONLINE] Available at: https://azure.microsoft.com/en-us/documentation/articles/app-service-api-authentication/. [Accessed 25 October 2016].

Figure 1
En tu casa o en la mía. 2016. *Entrando en casa*. [ONLINE] Available at: http://blogs.encamina.com/en-tu-casa-o-en-la-mia/2014/09/03/entrando-en-casa/. [Accessed 25 October 2016].