Context

**Here you give details of the development or investigation of the new material proposed in 'New Ideas'. This must be done in a business-like manner. The development of any software must follow a suitable analysis and design methodology. There are CASE tools available to you for some methodologies, others will have to be a 'paper' design. An investigation must also follow a suitable methodology and use appropriate techniques and tools.**

**Software-based projects, requiring the production of a software solution for a set of requirements, should demonstrate that the software development has undergone appropriate analysis, design, project management, structured programming and testing. An investigation must produce a technical outcome from some development (software or hardware (e.g. networks, displays)) or testing (e.g. of system/network performance, system security, HCI/usability analysis).**

**Some projects aim to provide software for general use as their final product and these must include relevant aspects of HCI (Human Computer Interaction) and address such features of usability such as 'user friendliness' and most likely employ GUI (graphical user interface) standards such as Windows.**

**In any case, students often ask what should go in this chapter, how to describe what they have done, what is relevant, how much of existing work to include, what to include from what they have done, etc. The simplest and surest way is to refer to your diary of the work you have done and report on it in chronological order.**

**The complete requirements analysis, problem analysis & design of software must be done rigorously and included in full in an appendix. Avoid cross-referencing it too often, thus causing the reader to keep flicking pages back and forth, rather reproduce sections that you wish to draw the reader's attention to. That is, highlight the parts that you found particularly difficult to implement and feel rather proud of having solved. Do not include lengthy descriptions of standard techniques or methodologies, simply state that 'such-and-such was designed using such-and-such technique (give a reference, not just 'SSADM' but 'SSADM [James 1996]' where the reference is a standard text on the technique!)' and highlight where you found shortcomings in the technique that didn't quite cope with your particular problem. Highlight exceptions to the standard.**

* Talk about the tools used for development (android, node, unity)
* As per leanUX produce a paper prototype of the application, showcase the alterations that have occurred from user testing
* What are the tools I will use
* Development diary?
* Highlight key areas of development (the game, fitness integration), hosting on Android market place)
* Check what is meant by this-

“The complete requirements analysis, problem analysis & design of software must be done rigorously”

# Introduction

The following chapter is focused around the development of the idea proposed in the Chapter Three: New Ideas. Within; a summary of the steps taken whilst following the Lean UX project methodology, the tools used for development are discussed with a focus on the reasoning behind their choice, a summary of key elements of the system that were developed with the aim of meeting the projects goals and finally a discussion around some of the challenges faced during the development of this project.

# Lean UX project methodology

This section is a summary of the steps taken by the development team to accurately follow the process of managing a project using the Lean UX project methodology. For a full writeup of the process and results of the individual steps please refer to **Appendix A: Lean UX**.

The first stage of Lean UX is to identify a problem with the current status quo, this is to say some situation where the goals of a target audience are not being fully met. In this case, the problem identified in Chapter 3: New Ideas, was that there are no solutions that offer a more passive approach to user fitness and entertainment, and that the uses of such a solution have yet to be fully explored.

Once a problem had been identified the project team conducted research into the user demographic of the proposed application, this step is conducted as it aids in empathizing with the target demographic, and helps to identify any bias or false assumptions the project team may have. These steps are all in aid of creating a better targeted solution for the users. The result of this stage is a list of assumptions the project team had about their target audience.

Once the previous stage was complete any assumptions the development team have decided upon were ranked in order of priority, the justification for this is to generate a testing schedule that focuses upon the highest priority items first. Priority is determined by a combination of two factors; knowledge and risk. Those assumptions where the team had little knowledge of the validity of the assumption but presented a high risk to the projects failing, if the assumption was wrong were ranked as the highest priority and so would need to be tested first.

With a prioritised list of assumptions, the next task was to develop a series of hypotheses, these are summaries of the assumptions discussed in the previous sections with the key difference that these hypotheses also contained a marker for their success. This allows the team to be able to quantifiably tell if the hypothesis in question is in fact true and valid. Below in figure 1 is an example of one of the hypotheses created as part of this exercise\*

“I believe that NHS staff need a way to keep track of the location of wheelchairs around a trust and that creating a system to do this will both save time and improve NHS staff satisfaction, I will be able to see if this is true by creating a prototype of the system and using on a small test group.”

Figure 1

*\*The format perspective of the hypotheses is in first person, this is in line with the guidelines of Lean UX (Add lean UX reference here)*

The last step in this process is the creation of minimum viable products that are created to test each individual hypothesis for validity. **Add information about the testing of hypotheses with the MVP’s**

**ALSO talk about paper prototype in this section**

# Development tools

The following section outlines the main tools used during the development of this project as well as justifications for these choices. Additional tools such as Git, Photoshop and Atom were used throughout the project, however the contributions of these tools to the overall project are too minor to receive full discussion in this section.

## Android and Android studio

One of the objectives of this project is that the application should be designed to operate in on a mobile operating system, this is to allow for ease of access to the applications, as so long as the user has their mobile device with them they will be able to make use of the proposed application. Several operating systems are available that could be used to host the proposed application. Android has been chosen as the most suitable for the following reasons (Martin, 2016);

* Android holds the largest market share of any mobile operating system, in Q2 2016 it saw a market share of 87.6% with Apple’s iOS holding 11.7%. Table 1 below shows the dominance of the android platform over other platforms between 2015 and 2016 (IDC: Smartphone OS market share, 2016)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Period** | **Android** | **iOS** | **Windows Phone** | **Others** |
| **2015Q3** | 84.3% | 13.4% | 1.8% | 0.5% |
| **2015Q4** | 79.6% | 18.6% | 1.2% | 0.5% |
| **2016Q1** | 83.4% | 15.4% | 0.8% | 0.4% |
| **2016Q2** | 87.6% | 11.7% | 0.4% | 0.3 |

* Google hold a large userbase in 2015 they announced Android had 1.4 billion active Android users (Callaham, 2015), such a high user base would enable the application to gain access to a greater number of potential users improving upon the validity of any potential scientific testing conducted within the realms of this project

### Justification for a single platform approach

With android set as the primary choice of operating system, the question did present itself as to what method of development would be used to produce the Android code, indeed there were several options available each offering different feature sets, some of the options present such as Cordova and HTML5 presented the opportunity to create an application that would work cross platform with Android, IOS and within a Web browser (Vensi, Inc., 2017) (Rajput, 2017). However, these options were passed up in favour of developing through Android Studio with Java for the following reasons;

* Google Fit is an open API that can be accessed using an android application, as the application proposed in the report intended to use fitness data to improve the users experience with the app the use of the Google Fit API was a crucial piece of functionality. The choice of Android Studio was in a large part influenced by this key functionality as access to the Google Fit API was not available through the presented other options as the trade-off of having multiplatform availability meant the lack of access to functionality native specifically to the Android platform
* Android Studio also provides the use of the Android Virtual Device manager (AVD), the AVD allows for the testing of an application on devices with multiple different configurations such as operating system version and screen size, as the proposed application is designed to appeal and work for a large audience, testing with these variations is another key reason Android Studio was chosen as the platform of development for this project.

## Node.js, Express and CouchDb

The tools in this section all make up a web server set up to host and manage the application.

Node.js is a community driven JavaScript based programming language that allows for non-blocking I/O in an event-driven architecture (dzone.com, 2017). Node.js has been found to be 10 times faster in I/O operations than competitors such as Java (Tilkov and Vinoski, 2010). This is achieved using an asynchronous event loop that does not block up other requests, this allows Node.js to offer a performance that does not decrease with a higher number of web calls. The choice of Node.js for this project is due to its ability to scale well with simultaneous web calls, this will allow the application to scale easily with an increasing user base without the need for any substantial changes to the server architecture.

An additional justification behind the choice of Node.js is the community behind it, with the use of the Node Package Manager (NPM) users have easy access to packages created by the community to fulfil many different requirements, one such package available through the NPM is Express. Express is a web application framework that allows for the relatively simple creation of a RESTful API that in the scope of this project will allow the mobile application to send data to and receive data from the application server. Express was chosen as the most suitable method of managing the http server as it is designed to be light weight and easy to use as well as the documentation available for Express is widely available and well maintained.

The final component within the http server architecture is a database with which to store data to allow the application to be persistent even if a user were to swap their primary mobile device. CouchDB a NoSQL JSON document based database was selected for use in this project due to its ease of use, especially within a JavaScript based environment as the JSON based format used by CouchDB allows for simple translation across multiple languages. The API used to access CouchDB is written in REST, this matches up with the use of Express meaning there will be no additional overhead required to pull documents and send them from the server.

The point could be argued that an SQL database would have performed as efficiently in this project as CouchDB, the justification for using a NoSQL database over one based on SQL is also related to the choices of all the tools that make up the web server, developer preference. Whilst in industry **employed by IBM** the author gathered experience using the tools discussed in this section as these were a part of the tools used daily by several teams within the organisation.

## Unity and C#

The final major tool used throughout the development of the project, Unity and by extension C# (as this is the primary development language within Unity) is a freely available, cross platform game development tool kit that can be used to develop 2d or 3d games. As Unity is able to produce applications that work natively on Android it satisfies the major project requirement of creating an application that targets mobile devices. Unity was selected over other available tools such as Unreal Engine and LibGDX for a number of reasons;

Unity is extremely popular amongst mobile game developers with 34% of the top 1000 free mobile games being created using unity (Unity, 2017), as such the community around the tool is also vast meaning access to documentation for Unity is readily available and simple to find.

The drag and drop nature of the Unity interface is clean and easy to understand, this also enables the user to focus more on the functionality of the game as opposed to the rendering of items within a scene, as this is done by Unity automatically (Craighead, Burke and Murphy, 2007).

### Closing remarks

It should be noted however that during the course of this project, issues did arise with the use of Unity that could have been overcome through the use of a different library such as LibGDX, these issues will be discussed in greater detail in the sections below.

# Key features

Within this section, the key features of the application will be discussed with a focus around the functionality they provide, and the justification for the implementation of these features in the chosen manner, this section will be concluded with a discussion of some of the issues that were overcome during the implementation phase of this project.

## Paper prototype

As part of the Lean UX project methodology testing hypotheses as quickly as possibly is of high importance, to this end a minimum viable product was created using a paper prototype to test hypothesis 1;

***Insert hypothesis 1 here***

--not sure I like this

## Match 3 Game

Game itself

Fitness goal

Store for rewards

High scores

(Dgkanatsios, 2015)

The Match3 game represents the visual element of the project, this is the only element on of development that is visible to the user

At the close of Chapter 3: New Ideas the project had a style of game and theme with which to replicate in order to appeal to the target audience of the application, to this end research was conducted to find materials that would be suitable for creating a game that matched the style and aesthetic the project team were aiming for. During this research phase the author came across a

## Node.js Webserver

A Node.js server was created in order to implement the following piece of functionality;

### Social user interaction

With the project objective in mind of creating a game that allows for some form of social interaction between users, the Node.js server would allow for data concerning individual player’s performance in game to be stored in a database separate from the user’s physical device, this allows the game controller present on a user’s device to request information such as other players scores resulting in the functionality for a globally available high scoring system. In its current state the webserver has functionality enabled to record and retrieve a global high score system as well as the ability to retrieve the high scores of users within a certain range, this allows a user to see the scores of other users who are close to their score for a particular level with the goal of fostering some competition and interaction between these users.

### Anti-cheat safeguard

Despite the low probability of this occurring, an additional reason behind the choice of a Node.js webserver was to remove the opportunity for cheating within the user base. Game development guidelines suggest that using a webserver to handle updates to player information where possible is a sensible move as it guards against any possibility for foul play (Bramer, 2014). This functionality is achieved by making the player controller within the game send requests to the server when a major game event occurs, such as when a player fails a level and loses a life. The player controller sends a message to the server updating it of the event, the server will then go through the steps of updating the player’s information and initiates a function to refresh the players missing lives. A snapshot of this can be seen below in ***FIGURE sasdasdasd***;

***Perhaps in another appendix?***

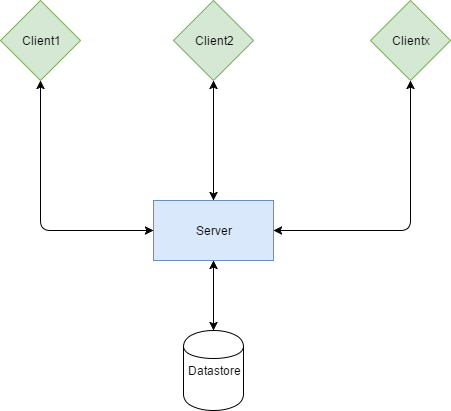
### Historical data

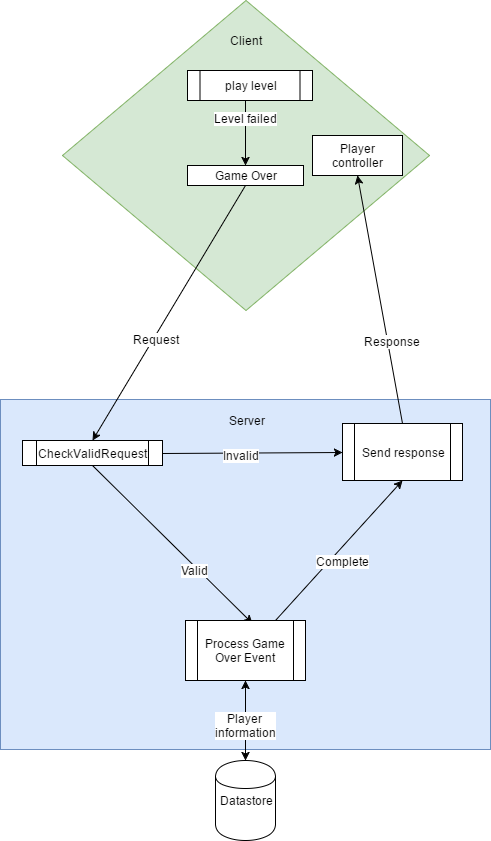
The final major element of the Node.js webserver is functionality designed around the ability to store historical data about the applications usage over time, recording historical data opens up the ability to monitor the success of the application by checking how often users are logging in and making use of the various features of the application. Historical recording also opens up the possibility for further research such as the potential research questions discussed briefly towards the end of Chapter 3: New Ideas. The historical data recorded during the development and testing of this application also became useful when it came to writing Chapter 5: Results.

**Code snippet of historical data capture.**

Asynchronous RESTful API

Anti cheat monitoring





## Google Fit API

### Introduction

Despite Unity hosting much of the mobile application code in this project, the functionality around collecting fitness data, essential to the aim of this project was inaccessible to the platform due to its nature as a cross platform entity having to balance many plates at the same time. In order to gain access to the user’s fitness a wrapper application was created using Android Studio that would gather the required information and send it to the Node.js webserver, wherein it could be assessed by the game portion of the project.

### Selecting a fitness metric

The wrapper application had functionality to determine the current user of a mobile device, establish a connection to the Google Fit api, then pull a variety of different historical fitness data types from the users fitness store. The Google Fit API has access to several types of data that were considered for use in this project such as;

* “com.google.calories.expended“ - This data type would return the total calories expended by the user within a given timeframe. An implementation of this data type could have required the user burn a certain number of calories in order to receive an in-game reward. However, this method of reward was passed over due to its lack of interactivity and in relative terms quantifiability, as calorific burning would be difficult to for a user to convert in to what actual exercise is required to meet said goal, this is especially so when such a goal could simply be set by the application as opposed to one based on calorie burning
* “com.google.activity.summary” – The activity summary datatype is able to return full fitness activities the user partook in within a given timeframe, this opened up the possibility of setting up more challenging goals for users, for example to partake in an activity that involved using a bike. However, this complexity was ultimately the reason this data type was not used, as it did not really fit in with the applications goal of creating an experience that doesn’t require a high amount of engagement in its fitness aspect
* “com.google.step\_count.delta” – The delta step count data type can be used to return the number of steps a user has taken within a given time, this can therefore be used to determine the number of steps a user took during a single day. This datatype was chosen as the basis value with which to build the fitness portion of the application around.

### Integration into the wider application

With a metric selected to base the fitness element of this application from the next step necessitated enabling this functionality in a way the end user is able to interact with it. With this goal in mind a system was set up within the Unity project that allowed users to claim a daily reward of in game currency for attaining a certain number of steps each day. At present the current daily target for all users has been set to 8000 steps, this value has been chosen based on the findings presented in Tudor-Locke and Bassett, 2004, in which up to 7500 steps is considered within the norm for general activity throughout a day, with the goal being set only slightly above the realms of normal activity a user may become motivated to achieve their daily goal if they are able to see how close they are to achieving it.

### Reward benefit

Rewards generated from taking part in the fitness aspect of the application are required by the project objectives to provide some in game benefit, to this end when a user completes their daily goal they are granted in game currency, which can be used to purchase a number of player bonuses, such as the ability to fully refresh the characters life count, and a purchase that increases the players maximum available lives. Several other bonuses such as new gameplay elements were considered for the project however due to constraints due to time they were unable to be added to the final version of this application.

## Integration of all components

# Discussion

# Testing

# HCI considerations

# References

Dgkanatsios (2015). Building a match-3 game (like Candy Crush) in Unity. [online] Dimitris-Ilias Gkanatsios. Available at: https://dgkanatsios.com/2015/02/25/building-a-match-3-game-in-unity-3/ [Accessed 6 Jan. 2017].

Martin, C. (2016) .NHS System Proposal. Unpublished manuscript, Nottingham Trent University.

IDC: Smartphone OS market share (2016) Available at: http://www.idc.com/prodserv/smartphone-os-market-share.jsp (Accessed: 15 November 2016).

Callaham, J. (2015) Google says there are now 1.4 billion active Android devices worldwide. Available at: http://www.androidcentral.com/google-says-there-are-now-14-billion-active-android-devices-worldwide (Accessed: 14 November 2016).

Vensi, Inc. (2017). Benefits of Apache Cordova Cross Platform Mobile App Development. [online] Available at: http://www.vensi.com/benefits-of-phonegap-cross-platform-mobile-app-development/ [Accessed 23 Feb. 2017].

Rajput, M. (2017). Top 10 Reasons to Use HTML5 for Mobile App Development - Developer's Feed. [online] Developer's Feed. Available at: http://www.developersfeed.com/top-10-reasons-use-html5-mobile-app-development/ [Accessed 23 Feb. 2017].

dzone.com. (2017). What are the Benefits of Node.js? - DZone Performance. [online] Available at: https://dzone.com/articles/what-are-benefits-nodejs [Accessed 25 Feb. 2017].

Tilkov, S. and Vinoski, S. (2010). Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing, 14(6), pp.80-83.

Craighead, J., Burke, J. and Murphy, R. (2007). Using the Unity Game Engine to Develop SARGE: A Case Study. 1st ed.

Unity. (2017). Unity - Fast Facts. [online] Available at: https://unity3d.com/public-relations [Accessed 26 Feb. 2017].

Your Bibliography: Bramer, B. (2014). Blueprint Networking Tutorials. [online] Unrealengine.com. Available at: https://www.unrealengine.com/blog/blueprint-networking-tutorials [Accessed 15 Mar. 2017].

Your Bibliography: Tudor-Locke, C. and Bassett, D. (2004). How Many Steps/Day Are Enough?. Sports Medicine, 34(1), pp.1-8.

# Bibliography