Department of Computing
Goldsmiths, University of London

Augmented Reality Navigation System for Commercial Spaces

Report

by

Arif Kharoti, Nicholas Orford-Williams, Hardik Ramesh,
Gabriel Sampaio Da Silva Diogo, Hamza Sheikh, Jonathan Tang
Software Projects – Group 14

Spring 2019

Submitted in partial fulfillment for the degree of
Bachelor of Science in *Computer Science*

Abstract

Contents

List of Figures

List of Tables

Nomenclature

AR	Augmented Reality
CI/CD	Continuous Integration/Continuous Deployment
GDPR	General Data Protection Regulation 2016/679
IP	Intellectual Property
MVP	Minimal Viable Product
SDK	Software Development Kit
SDLC	Software Development Life Cycle
TDD	Test Driven Development

Acknowledgements

Chapter 1

Introduction

1.1 Motivation

This project revolves around the use of augmented reality (AR) navigation on smartphones. AR is the superimposing of a computer-generated image onto a user's view of the real world [1]. This technology first came about in the 1960s [2] but recently gained wide-spread consumer attention after the use of it on Snapchat filters [3], and the 2016 game *Pokémon Go*.

The need for such an is justifiable. People find themselves lost in unfamiliar spaces such as museums, immersed by the culture around them. This project aims to tackle this issue by allowing users to restore their orientation by having an AR platform, routing users to their destination. This approach is considerably more extensive, and dynamic in comparison to the existing 2-D solutions available in the market. The platform will use the device's camera to work out its surrounding, and produce a highlighted line on the screen to their destination in real-time.

This concept has various applications to other scenarios such as finding products in a supermarket, or books in a library. Furthermore, the concept could also use machine learning in identifying user traits in places visited in a museum in order to give personalised recommendations at other similar exhibi-

tions.

1.2 Purpose & Scope

The purpose of the project is to provide a solution to the problem at hand. A real-time navigation system to assist users to their desired location within a museum. The significant part of this project is achieved by calculating the quickest route to the user's destination based from the user's current location, and displaying this route on the user's device through the implementation of AR-assisted superimposition of navigational lines that the user would follow.

1.3 Assumptions

It is assumed that the reader is familiar with the supporting documents of this report, including the proposal, system requirements specification, the documentation, testing and deployment plans. These documents outline the concept of the project and detail the procedures to achieve the purpose of the project.

1.4 Coverage

This report fully describes the project undertaken, containing eight main sections:

1. Introduction - Gives an introduction to the project, outlining the motivation, purpose and scope of the project. Includes any assumptions based on the reader regarding any prior reading and the coverage of this report.

2. Background - Analysis of current projects and literature available in this

area. It also contains analysis of AR libraries which held potential to be implemented, and a discussion of the technologies behind each of the proposed libraries as well as their respective Operating System (OS). Further, an analysis on the use of Arduino and Rasberry Pis in current solutions.

3. Project Management - Providing a breakdown of the development methodology and justification of adopted approaches. This section of the report will also analyse the Software Development Life Cycle (SDLC) and outline the roles of all members involved.

4. Requirements - Outlining the various requirements of the project; the gathering of user requirements to be able to develop a minimal viable product for end-users. Analysing the stakeholder, system, functional and non-functional requirements.

5. Design - This section will define how the system will achieve its purpose. Analysing the different use cases and activities of the application. Discussing the technical architecture, user interface and accessibility features of the application whilst ackowledging at which parts of process user consultation was necessary.

6. Implementation - Discussion of the implementation and justification of the decisions made. Detailed reports with reference to the backlog of the development, outlining sprints, splitting up front-end & back-end development and system hardware. Included in this section is also reports of how the team enforced ethical audit, evaluation of the challenges that were faced during implementation and what steps were taken to overcome these challenges.

7. Testing & Quality Assurance - Analysis of the test-driven development approach adopted by the team and discussion of the various types of testing conducted:

    - Unit Testing

- Integration Testing
- Performance and Stress Testing
- Regression Testing
- User Acceptance Testing
- Beta Testing

In addition, a formative evaluation will be given in subject of testing and detailed reviews of the function and non-functional requirements in regards to the original purpose of the project.

8. Project Evaluation - Evaluation of the success of the implementation in meeting the needs discovered in the background research is given in quantitative and qualitative terms. Analysis of the successes and failures of the project, and discussion of the advances made. Possible extensions and further work that could be undertaken are then discussed.

In addition to these main sections there is a glossary and bibliography of references at the end of the report, along with a number of appendices. These appendices contain:

- Appendix A - Research conducted around end-users and stakeholders which were key in requirements gathering.
- Appendix B - User stories and scenarios in which a user would use the application. Screenshots of the user interface and camera activity and the procedure in which the user would use the application are given.
- Appendix C - The System Requirements Specification: includes the purpose, scope, system overview, references, definitions, use cases, functional and non-functional requirements.
- Appendix D - The Documentation Plan: outlines the strategy for creating all documentation associated with the sofware release.
- Appendix E - The Testing Plan: in accordance to the IEEE Standard for Software Test Documentation (ANSI/IEEE Standard 829-1983), out-

lines and describes the testing approach and overall framework that was followed during the implementation phase.

- Appendix F - The Deployment Plan: designed to ensure that the system successfully reaches its users and maintenance of the system allows for new features to be delivered seamlessly. Details further information about the development of the system.
- Appendix G - The testing conducted during the implementation and development phase of the project.
- Appendix H - User & Stakeholder feedback.

Chapter 2

Background and Literature Review

2.1 Background

Recently, a new exciting segment of the technology sector has sprung up – indoor navigation with a catalyst of growth in indoor navigational research. Most current solutions on the market cater very well for a basic navigation of large open public spaces, though failing to display a competency to more complex navigational requirements coupled with an even proportion of navigational and interactive content with well-presented data. Through the use of augmented reality, the concept can provide an interactive solution for museums and exhibitions – with a wide future scope.

Since most museums and galleries use a portable audio guide, user experiences can be vastly improved by the use of a phone. Currently only a few solutions can be found; the Orpheo group [4] provide a unique app for each place meaning that their solution is somewhat cumbersome to regular museum users who would wish to have a hassle-free setup process. As we hope to appeal to museums and by virtue of this, museum-goers, having one app whereby the user can simply walk into a museum or exhibition and be greeted with relevant information to be a vital differentiating factor [5]. Even though the intention

is to create one app for every museum and exhibition for the project, each client would have a large input on the content of their institution within the app.

If a museum, wanted a solution for navigation, due to the low number of museum-specific competitors, would choose to use a standard indoor map- ping software [6]. However, while there are many options out there from Google and Mapspeople [7] who set out to provide this, they lack important exigences that are very imperative for museums like heavily integrated augmented reality, intelligent tour guiding from your location, and virtual reality to take a scene from the museum, for instance, and place the user to the artefact's original time and place.

2.2 AR Libraries

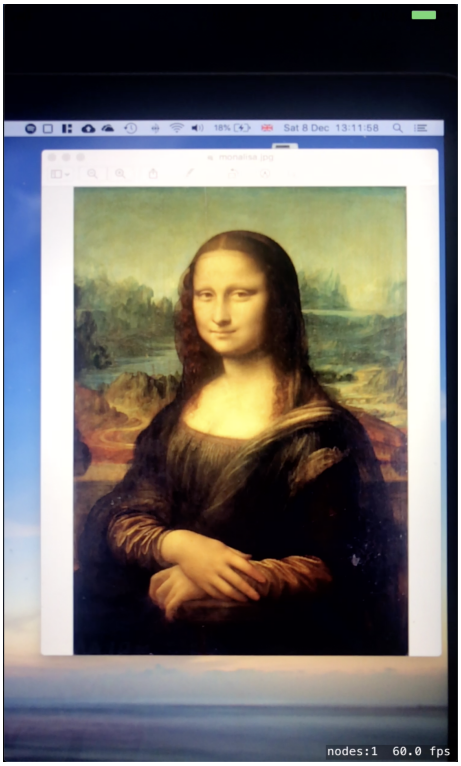
The AR component of an AR based application is perhaps the most crucial component. Therefore the group evaluated three libraries before developing any software. Those were, ARKit developed by Apple for its native iOS platform, the AR Core library by Google for Android systems, and Vuforia - a cross-platform SDK built by PTC for Unity/Android, specialising in image recognition.

2.2.1 ARKit

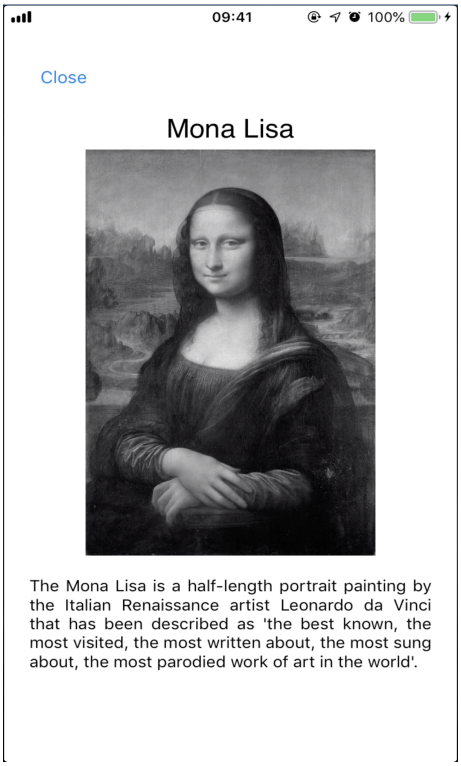
ARKit was a strong candidate in becoming the library of choice because of its nativity to iOS devices.

Although the application is only supported by Objective-C/Swift, it has more thorough and straight-forward documentation than Android's AR Core and PTC's Vulforia. This would possibly make the idea easier to manifest because the debugging process would likely run more smoothly. But due to the lack of the total group's familiarity with the platform. Only one person was able to

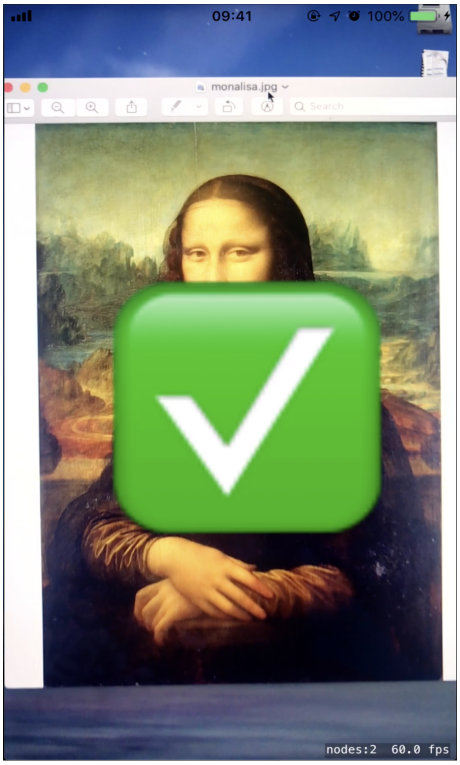
partake in creating the image recognition prototype (Figure 2.1).



(a) Camera over image



(b) Image recognised and displaying information



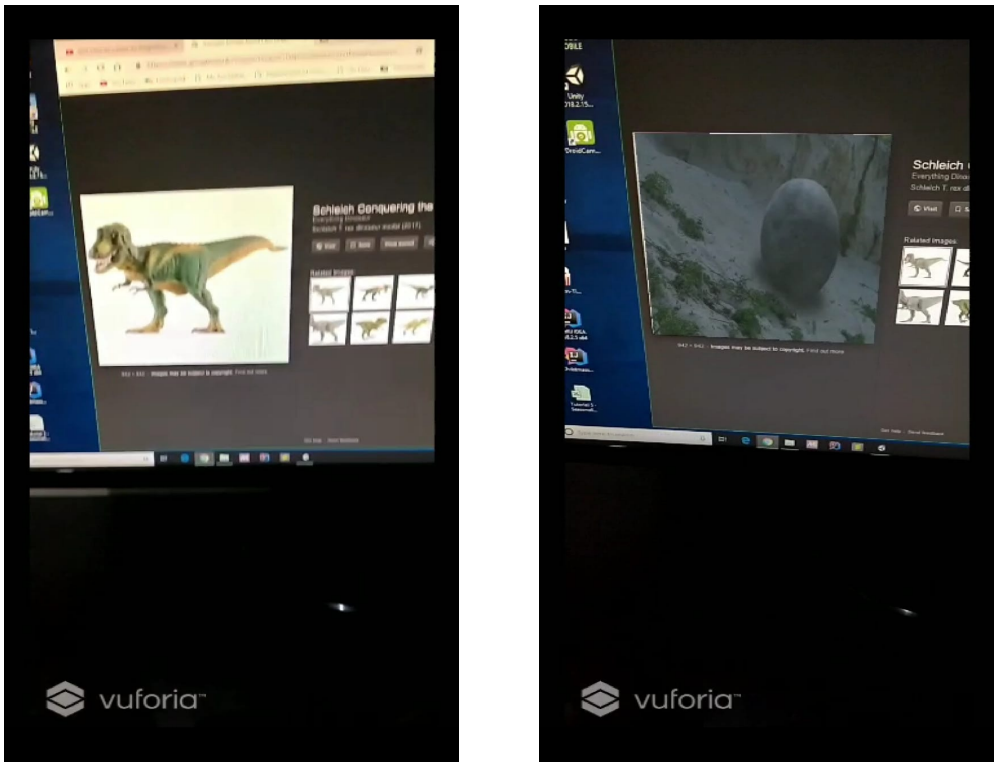
(c) Image scanned before; showing the green tick

Figure 2.1: ARKit prototyping on iOS device

2.2.2 Vuforia

Unity is a cross-platform game engine, used to test a simple AR camera prototype where the device's camera hovers an image, and displays information about that image on the device. The application was built using Vuforia, an SDK that enables recognition, and tracking of image targets. This library can be used for the exploration case in the use case model. Although, there are a lack of tools for locating user current location compared to Android.

The Vuforia library was used to test a simple AR camera prototype where the device's camera hovers an image, and displays information about that image on the device.



(a) Camera over image (b) Video superimposed on top of image

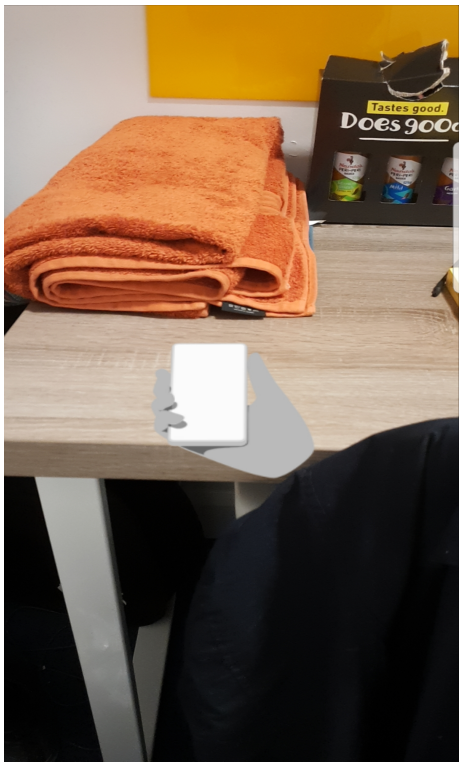
Figure 2.2: Vuforia prototyping on Android device

2.2.3 AR Core (Android)

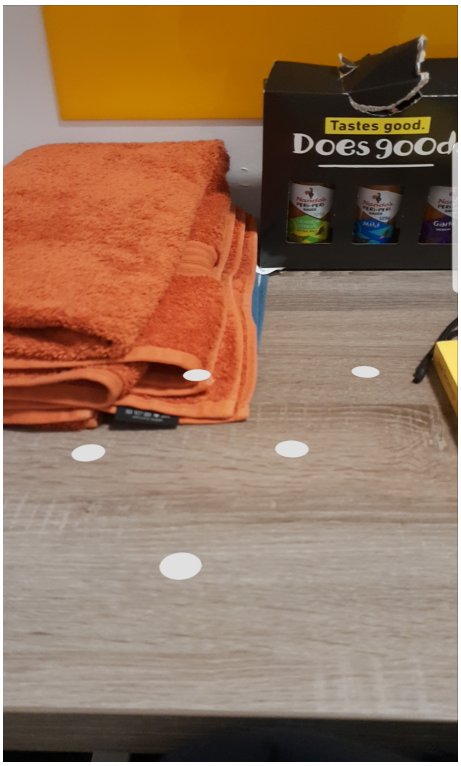
AR Core was used to create an AR experience (Figure 2.3) giving the user the ability to superimpose a 3D object when the camera is focused on a flat

surface.

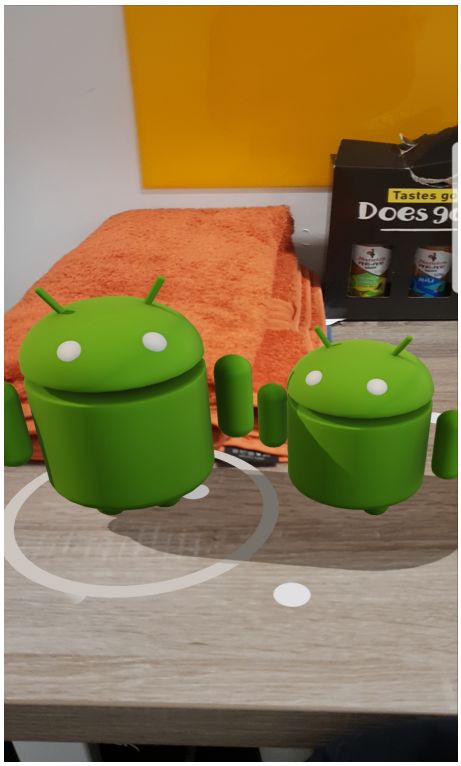
During the process it was found that unlike ARKit, the library is suitable with Java/Kotlin, hence the group would have an easier time acclimating the nature of AR Core. As AR Core is a native Android SDK, it can be integrated with other Android features such as Android's GPS and Android's WiFi libraries. Despite this, the documentation was scattered and vague, making it a difficult program to debug with. After exploring this prototyping process and discussing the accessibility of AR Core(Android) and android to the stakeholders, the group distinguished it as the most suitable library for the project.



(a) Initial view



(b) Detection of surface



(c) Objects superimposed on surface

Figure 2.3: ARCore prototyping on Android device

2.3 Software Architecture

By analyzing software architecture to identify the potential risks, and to verify the quality of software have been addressed in the design. It will helps to determine which software will be useful to use in terms of developing our application. The team identify two system to choose from Android and Apple. Within those two system, the team need to find which one of the system is best for our application.

2.3.1 Android

An open-source OS developed by Google, it provides an SDK for development of an application. It uses *.apk* extension that can be downloaded and installed on mobile phones. In order to make AR application, we use ARCore to create our app. There are many advantages and disadvantages when choosing the ARCore for android.

2.3.2 Advantages

- Android Emulator - It is nearly three faster in CPU, I/O and RAM compared to ARKit(Apple). This will help our application to load/get faster in user screen.
- Easy to log GPS location compared to iOS.
- Easy to build UI with Layout Editor.
- Easy to run on multiple android devices such as Samsung, Google Pixel, OnePlus,etc.
- Easy to create a simple 3D model that will show on a mobile device when its camera targeted flat surface.
- ARCore can detect horizontal surfaces that is similar to motion tracking, and can accurately anchor virtual objects.

- Cost effective

2.3.3 Disadvantages

- It's more difficult to code complex layouts and animations.

2.3.4 Apple(iOS)

It is a mobile operating system developed by apple. It provides ARKit SDK for development of application. In order to create an Ar application, you have to use ARKit. There are many advantages and disadvantages when choosing the ARKit for apple.

2.3.5 Advantages

- ARKit using the Swift, which is easy to learn.
- Stable and fast motion tracking.
- It helps to load our application in user screen faster.

2.3.6 Disadvantages

- limitation of ARKit map compare to Android
- Not easy to log GPS location.
- It is not flexible only support iOS devices.
- Not easy to create an 3D model for naviagation.

2.4 Hardware - Arduino and Raspberry Pis

As part of our exploration it was recommended, based off lengthy research and from an interlocution with the project's domain expert the need for a to a physical beacon.

This role of this beacon, no matter the configuration, was to broadcast a signal. Three main current technologies exist exist: optical infra-red, Bluetooth and, WiFi. Speaking to industry specialists, the former – which underpins their navigation technology and would yield pin-point accuracy – important if you are dealing with small indoor spaces, but this is expensive; Bluetooth, while less accurate, is far cheaper to manufacture and develop; the latter-most (thricely), WiFi offers the least in accuracy and not too expensive. From this probing, Bluetooth became the team's solution.

Turning in to more a physical project which was not anticipated, a task was set to design a beacon. Looking at relatable past projects, the favourable direction was using a Arduino (UNO R3 Mega 2560) or the Raspberry Pi 3 Model B+. Both solutions gave a clear, small and light-weight canvas on which to design our beacon. However, with a smaller physical footprint – also in keeping with the project's eco-friendly policy, the Arduino was cheaper. This was a priority to get the technology to have a wide-spread appeal.

Taking forward the Arduino platform, it was kitted out with a Bluetooth HC-05 RF transceiver. This setup was able to broadcast a simple Bluetooth signal where software can decipher a signal strength from which a distance could be determined. Despite this, it became apparent that there was a need to have further invest in more beacons to give a more accurate and reliable result via triangulation. This was not a suitable solution as one underpinning goal of this project is to make our technology assessable and available to a wide range of applications – an increase in cost would hinder this.

Chapter 3

Project Management Processes

3.1 Development Methodology

In order to produce high-quality software, a development methodology must be selected to monitor, and control all aspects of the project that is deliverable according to the stakeholder requirements within the expected milestones.

Six variables were identified that should be managed throughout the project [8]. The **cost** of the project has to be affordable, and need to factor actions that lead to overspending, or opportunities to cut cost. Allied to this, **time** is the only cost in the project. Ensuring the product delivers on its **quality**, and is fit for purpose along with the **scope** where there is an agreement between stakeholders on the deliverables. It is important not to deliver beyond the scope as this is a common source of delays, overspends and uncontrolled change, known as 'scope creep'. Managing the impact of **risks** is essential, but to assess how much risk there is, and what can be done is of importance. Finally, maintaining the **benefits** of the stakeholders is of particular interest so that there is a clear understanding of the purpose.

With these factors in mind, the waterfall, agile, and lean methodologies were

considered for the project. Waterfall is a fixed sequential framework where it is linear in nature. Each of the eight processes in waterfall needs to be completed before moving onto the following stage. Despite the easy understanding of the methodology due to strong documentation approaches, this did not suit the project as changes cannot be easily accommodated [9]. If there were errors during the implementation stage discovered during testing, then it would be expensive and difficult to go back. Further, there is greater reliance over the initial requirements; if these are incorrect they can be detrimental to the project.

Lean focuses on removing waste from a process in order to deliver products quicker, lowering delivery costs without sacrificing quality, and providing a flexible response to users. This model narrows down to completing the minimal viable product (MVP) before considering extra features or content, though sometimes the full potential of the project is not reached. Similar to waterfall, the documentation needs to be explicitly precise in the requirements. Further, there were many skills to be acquired by the team throughout the project, and learning on to go is not possible in lean.

Agile is an iterative, incremental approach that is open to changing requirements with frequent feedback from end users. By breaking down the project into small chunks, it makes it easy to prioritise and add or drop features mid product, which is impossible in traditional waterfall projects. There is a requirement to adopt a more responsive, test and learn approach to the development activities. Breaking down tasks into iterations can allow the team to concentrate on high quality implementation. Such an approach delivers measurable results quickly, engages stakeholders earlier, and facilitates a more frequent and reliable product release cycle. However, due to the quick space of the process, the quality of the delivered software could decrease.

With these benefits in Agile, the team decided to adopt the agile methodology with the scrum framework. Scrum allows for a highly adaptive change in requirements during the entire cycle, and high transparency. The product owner is constantly in touch with the team [10], and daily stand-up meetings

eliminates any misconceptions and confusion. Risk is controlled since issues are identified in advance before getting unmanageable. Kanban was not used since the project required iteration, and prioritising tasks on hand is harder with a Kanban board as all tasks are placed at equal importance with this framework.

3.2 Software Development Life Cycle

In the scrum team, the project supervisor served as the product owner where they were responsible for maximizing the value of the product resulting from work of the development team. This ensures that the end product delivers value to the end user. The project lead took on the role of scrum master to ensure that stakeholder requirements were met, and to be accountable for the delivery of the sprint goals. The scrum master also took on the role to set and adjust the priorities of the product backlog since they had a greater understanding of the project than the product owner. Having a flat line structure allowed the development team to be self-organising and cross-functional, with all the skills as a team necessary to create the product increment. According to the framework, scrum recognises no sub-teams in the development team [11], and this was avoided during implementation. This was so the whole team is accountable for all actions rather than those who have specialised areas of focus. Further, this allowed us to be dynamic in allocating resources to certain tasks when problems arose.

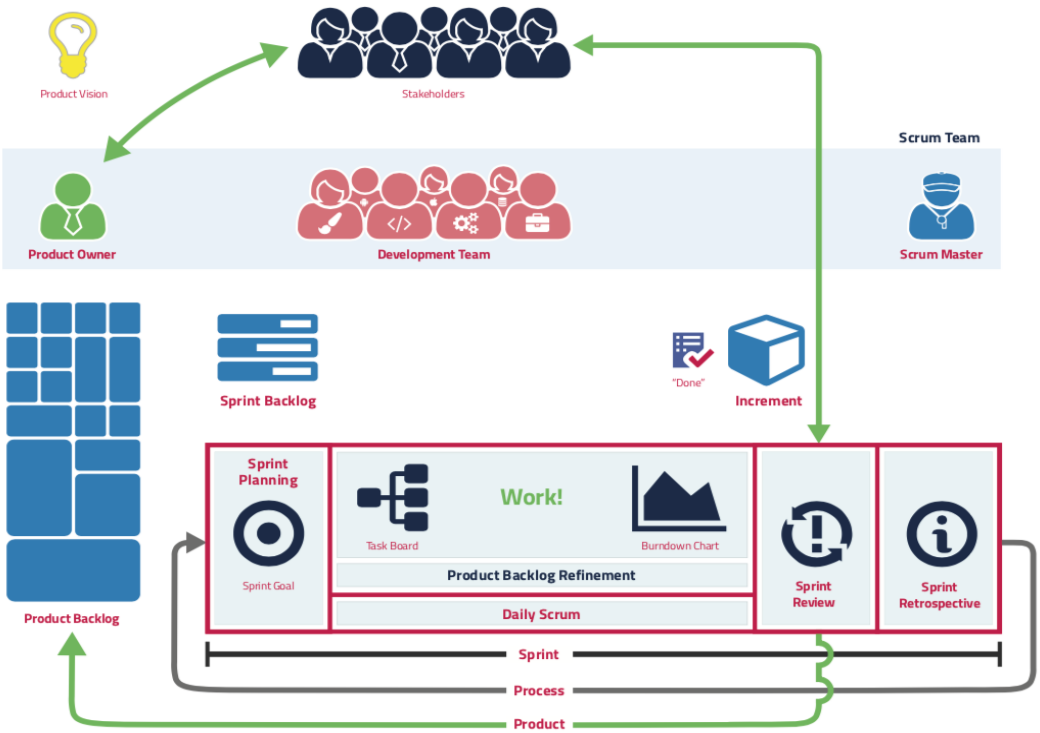


Figure 3.1: The Scrum Workflow [12]

Scrum events took place such as sprint planning, daily scrum, review, retrospective, and backlog refinement. Sprint planning always took place on the first day of the sprint where three key indicators are outlined, what can get done in the sprint, how the work is done, and the sprint goal [11], providing the development team an objective. This event usually lasts about one to two hours for a two week sprint. Due to careful planning at the start of every sprint, this led to successful sprints during implementation and being able to deliver on requirements.

Conducting daily scrum proved challenging from the beginning as members were not always on campus at the same time; to resolve this, stand-up would take place almost every two days, and at least four times a week. In both terms at the start of every lab, and every supervisor meeting a ten minute stand-up would happen in person. The other two meetings would happen virtually on `appear.in` for ten minutes, the day before a milestone deadline (Thursday), and either Saturday or Sunday - depending on the group's availability. Using this model allowed for quick decision-making, and removing any obstacles the

team faced.

A sprint review took place to audit the work completed, the positives, negatives, and problems of the sprint. At this point, the group would decide if the tasks assigned were at a "done" status, subject to review of the product owner. All requirements that were outlined within a reasonable time constraint must be reached to achieve this status. If there are any outstanding tasks to be completed, then this would filter through to the following sprint planning, and the product backlog will be edited accordingly. This formed part of the sprint retrospective, but mainly focused on how well the team administered the methodology. Other key stakeholders are consulted here, giving any important feedback to the completed work.

The team used a scrum board in order to track tasks and make any changes during sprints. This promoted transparency amongst the group, and ensured accountability was retained. The project used other scrum artifacts discussed in chapter five. Overall, the execution of the Agile methodology was good, since two members of the team had worked together in projects using Agile in industry - leading to a greater understanding by everyone of how it works. However, during the first two sprints the scrum master was involved in the sprints themselves, and that lead to work not meeting requirements. This was rectified in the remaining sprints, where timely interventions at impediments took place by them so that momentum was conserved, and the outlined requirements were achieved.

3.3 Test-Driven Development (TDD)

Another development methodology used was TDD. Before completing implementation for the sprint, the unit tests were written so that the focus when writing source code is to make all the tests pass. This allows for the software architecture to be thought out beforehand, and ensure that the core functionality is acting on the requirements. Also, this methodology is proved to reduce

defect density, and reduces cost since the testing time is dramatically reduced [13]. Further, TDD fits into Agile's iterative approach, providing ease of allocating resources in sprints, and all developers have a great understanding in this.

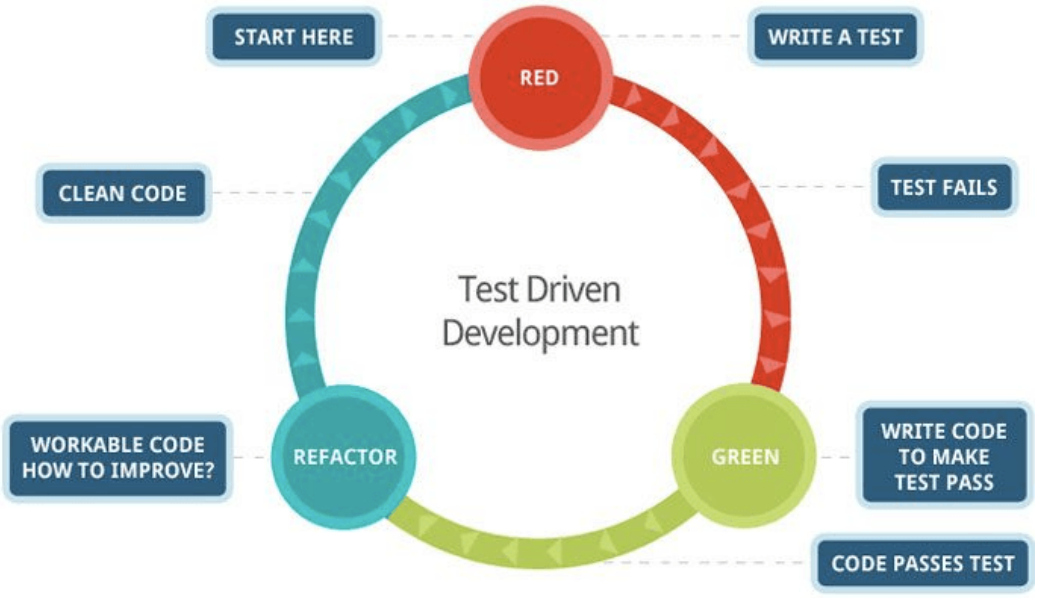


Figure 3.2: The TDD Workflow [14]

Business-driven development, and acceptance test-driven development were considered, but a more thorough and rigorous testing system that focused mainly on the implementation was favored to deliver a working MVP.

Chapter 4

Requirements

4.1 Stakeholders

Stakeholders hold a very important position during a product's development; often influencing the outcome of a project by way of its scope, means of function and overall effectiveness and success. In foresight of this impact, we were keen to detect and appoint all appropriate audiences to this rank.

Fundamentally, the group at the centre is the End User; in near-to-every circumstance, if those at the bottom on any sort of hierarchy reject an idea or practice, those above will not benefit through its implementation. Furthermore, this is exacerbated as this project will largely change the End User's experience of a museum which, if rejected, may lead to a decline in visits and, thereafter, in funding. To understand our potential impact, expectations from visitors and members of museum staff of the Science Museum and the Natural History Museum were noted.

Though this project was the product of an academic environment, it is not too unlike the real world in that we had we had direction set by this project's supervisor, Dr. Basil Elmasri and standards set by Dr. Nick Hine – the module leader. Given this project had to adhere to College's marking criteria, our supervisors responsibility was to ensure that we were at least doing so. Fur-

thermore, the quality of work produced would be representative of them.

As this project is framed around being a main point of interaction between museums and visitors, it could affect the experience and potentially having an adverse effect. Because of this, those in industry had a stake in this project. 'Those in industry' is two-fold: the technology sector and creators of exhibits or those who know about the representation thereof. It was a priority to get an experienced understanding so a line of communication with an company called fuse who have an incubator company, Pathfindr (navigate), who provide indoor navigation solutions, via the Managing Director, Matt Isherwood.

Seeking opinions from those in the business of being responsible for exhibits or creative works, a priority was to gauge this stakeholder via showing prototype designs and ideas. Getting the full scope of their opinion in relation to this project as our project will directly change how the end users interacts with the exhibit, leading to either hindering or bettering their experience.

Finally, the team behind this project, have a large stake in its success. Not only that if there was not a quality report, but this new part of the technology sector is in its infancy so needs innovative ideas. Further to this, a ill digested report would act favorably towards the project sponsor, Dr Elmasri.

4.2 Gathering

In presenting our System Requirements Specification (SRS) to our stakeholders, we were able to determine via our surveying their approval. Along with (cite stakeholder survey response) Feedback was positive and we were further encouraged by staff describing the current system of charging the current system of charging for maps and their current app solution as inadequate and cited often having to help visitors having simple issues navigating. We made it our business to visualise as much as possible the user experience to ease its otherwise, seemingly complicated description. Constructing the diagram showing our use case made it easy to demonstrate the project and, furthermore, in-

spired, jointly with our user requirements, the first graphical prototypes took shape. Using Adobe XD, we quickly made a mockup of the final design which allowed for a clear commination twix us and the stakeholders.

4.3 System

The system requirements that it is an Android Mobile application, written in Android studios. using Java. The reason we decided to use Android was because Android is accessible on more devices and is able to greater number of audience. The SRS was designed to show the structured collection of the requirements of the system along with its operational environments, and external interfaces. The specification was written according to the ISO standard for systems and software engineering. The purpose, scope, and system overview serve to provide an outline of the overall platform. Use cases diagrams from chapter four are fully described in the specification. The functional requirements comprise of the system behaviour, the functions and features (what the system should do). It considers the key features such as the user navigation and its relative implications. Whereas the non-functional requirements place constraints on how the system should do it. This considers the usability of the application, describing aspects such as performance, user actions, and safety.

A software requirement specification otherwise known as SRS is a description of a software system to be developed. SRS is used to layout the functional and non-functional requirements of a system. As well as that, a software requirement specification should include definitions of both system and user requirements, it should describe fully what the software will be.

Our system specification requirements clearly defines what type of software application we are going to be building. We used the SRS to know all the requirements for our application, which helped us in designing the software. System specification requirements creates a bond between the developers and

clients, it allows them come to a conclusion about what should be implemented and how it should be implemented. Without a SRS it would be hard to know what needs/ doesn't need to be implement, also it creates a lot of mess and confusion between the clients and developers, and creates a mismatch of expectations. Which in the end is disadvantageous to both customer and developer.

Constraints, Assumptions and Dependencies

1. **Internet Connection**: The application would not be able to query mapping services or have access to exhibit information otherwise.
2. **Android**: Users of this application are Android device users that requires assistance in museum navigation. Devices that support basic dependencies of the application is expected for proper user experience.
3. **ARCore enabled device**: users device must be compatible with ARCore for them to have access to the application's AR navigation.

4.4 Functional

The functional requirements comprise of the system behaviour, the functions and features (what the system should do). It considers the key features such as the user navigation and its relative implications.

The user should be able to use the application to navigate between the given current location and the required destination. The system should be able to display navigational routes in real-time from the users given location calculating the quickest possible route. A 3D line should be superimposed through augmented reality making use of the users camera to display the navigation to the user's destination. Camera recognition, detecting artwork/exhibits should display further information about the exhibit. When the user arrives at destination, the system should give a recommendation based on their route.

4.5 Non-Functional

The non-functional requirements place constraints on how the system should do it. This considers the usability of the application, describing aspects such as performance, user actions, and safety.

1. **Performance**: The system should respond quickly to user input, e.g user wants to find more information about an art piece or whenever they search up a location. The system should not require extensive CPU usage, it should not slow the device down inconveniencing the user.

2. **Usability**: The system should have a simple layout, with appropriate colour used in appropriate contexts. The language used on the app should be easy to understand for the users. Having done research based on the user preference, it was discovered that users dislike too much text on their menu screen.

3. **Data Usage**: Data usage should be kept to a minimum, only querying the relevant information (user location and exhibit information). Also, the app would require internet connection in order to calculate the real-time distance of the final destination from the user's current location.

4. **Safety**: The system needs to have the ability to detect immutable objects obstructing the user's path. This can reduce common user accidents when using a mobile phone whilst walking.

5. **Security**: Ensuring that the device's current location cannot be obtained by unauthorised third-party users is crucial in ensuing the security of using the platform.

Chapter 5

Design

5.1 Technical Architecture

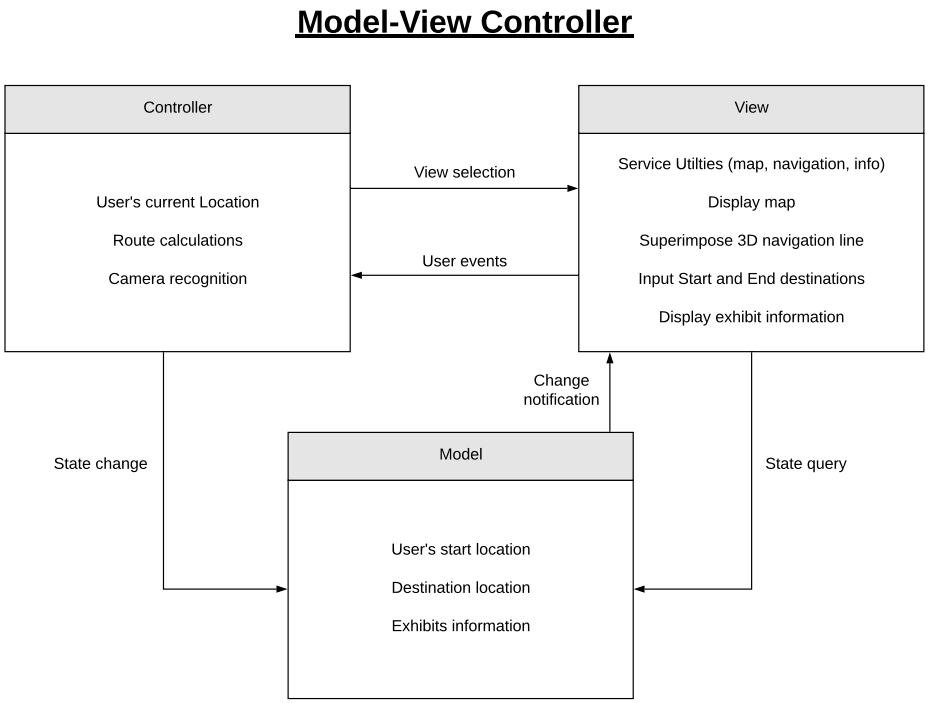


Figure 5.1: MVC

To develop the Muse application. A reliable technical foundation was necessary, and with the explained user-based nature of the application, the speculation was quickly drawn that the Model-View Controller (MVC) architecture

(Figure 5.1) would be the most relevant to creating the optimal version of the application. The group had already been familiar with the architecture, and is proven to be massively successful in the mobile application market. In order to make sure this pattern would suit it , technologies needed to be picked around the model.

5.1.1 Model

The model of the MVC architecture is known to be the core component of an application. This is where data is stored and manipulated in this case, the users location, destination, and information held by the exhibit. These were to be manifested as a hard-coded location within a virtual geo-space to simulate the venue, and string data regarding the destination.

5.1.2 View & Controller

Android devices were the obvious conclusion as the platforms to develop for because of group/stakeholder familiarity with the system. So the group embarked on builiding the front-end using Android Studio along with Adobe, and the back-end using AR Core and Java. The front-end involved a login screen, a map screen, an info screen and navigation screen; all were which controlled by the Java and Kotlin code in the back-end.

5.2 Models

The following models are based on user interaction with the application.

5.2.1 Use Case

The use case diagram is crucial because it clarifies the lifecycle of a user on a user-based application. With this model, developing a user-based application

like this becomes more manageable.

The use-case diagram (Figure 5.2) deals with the two states that the user could be in when using the Muse application; the lost state and the exploration state.

The lost state is initialised by the user when they request a destination. Which is then; followed by the application picking up their current location, calculating the quickest route between their requested destination and current location and displaying it to them via the superimposition of a line in their camera view. The user needs to follow the route until they arrive at their destination so they can consider another destination given by the application.

If the user considers another destination, or starts the application with no destination in mind. They have declared themselves to be in the exploration state. Which then initialises itself with by looking at the users past visits (if they have any) and current location to calculate recommendations. The user then chooses one of the recommendations so that the application can again, extract their location and use it to calculate the best route between them and the location of the recommendation. From this point, the exploration state is identical to the lost state. The user follows the superimposed line and reaches their destination for the lifecycle to either begin again, or end.

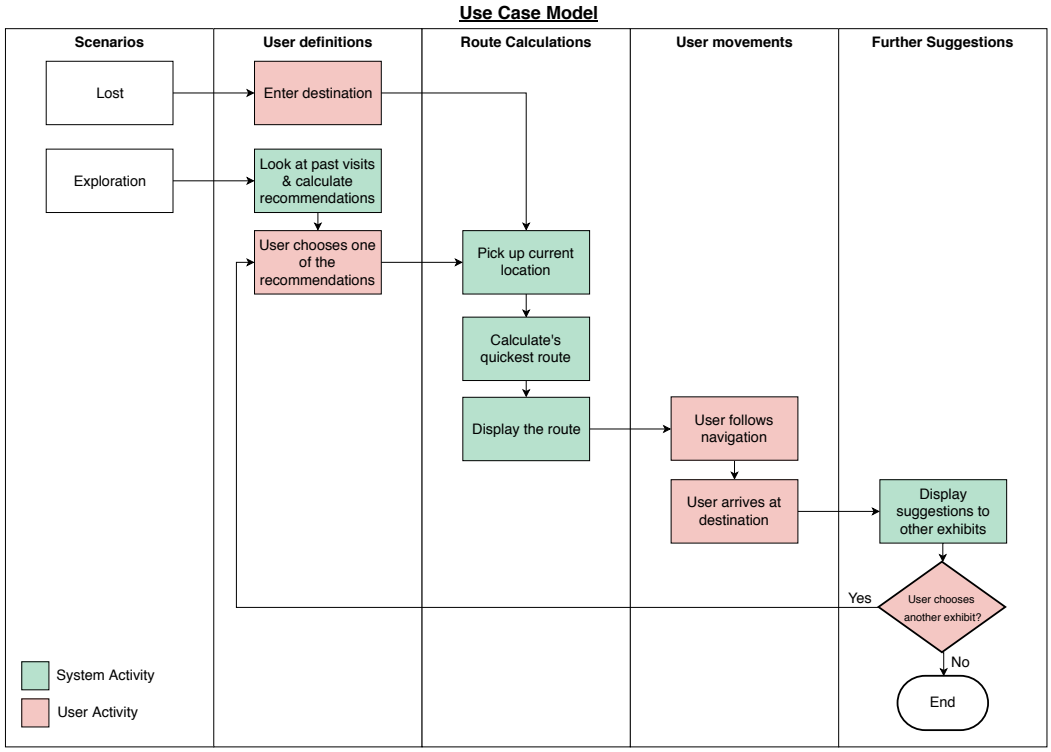


Figure 5.2: Use case diagram

5.2.2 Activity

The activity diagram (Figure 5.3) distinguishes the details in two sections of the use case diagram, user definitions, and route calculations.

The route calculation process in the application, is linear (Figure 5.3). The user location is received from the GPS - this distinguishes what venue the user is in. The building makeup is then requested from the server in order to make an analysis on the geospacial coordinates that the user will have to traverse. After the analysis is complete, the route between the user and whatever destination they requested in the user definition section can be calculated and the route can be superimposed.

The user definitions that the activity diagram (Figure 5.3) deals with are case specific to the exploration case. If the user would like to explore their venue, and they do not have an account on the application; they are prompted to make an account so that the application can provide optimised recommendations.

In the case that they have an account, their previously visited locations can be used in presenting recommendations.

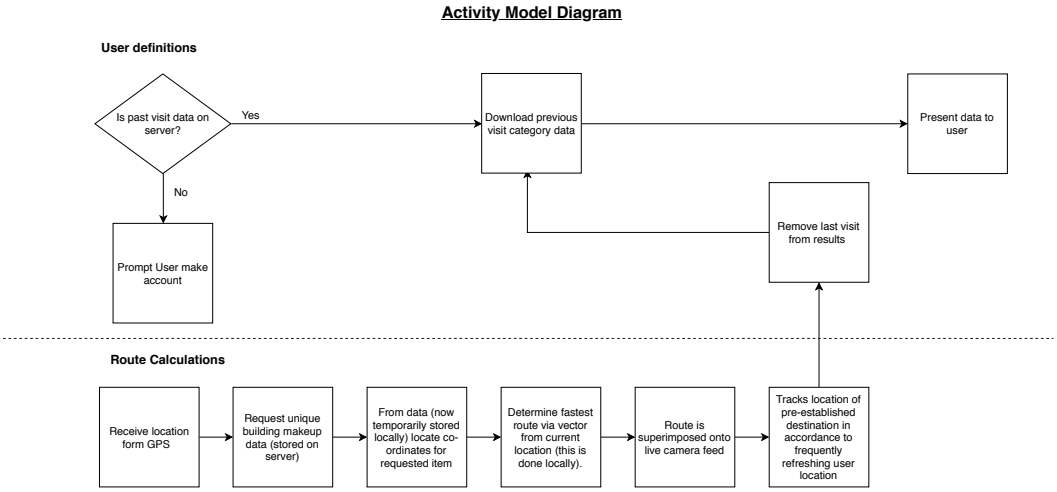


Figure 5.3: Activity diagram

5.3 User Interface

The project lends substantial importance to its user interface and experience. As it will be used by people with a wide range of technical ability, the aim will be to make the app as simple as possible without having an impinging effect on any service the end product will feature. This prerequisite was clearly outlined in the surveying of museum guests and staff alike. The first mission was determining what interfaces, and experiences currently exists within the museum sector. Many museums employed simple interfaces but due to their mass-manufacturing, their design felt unoptimised with simple bare-bones media not beyond text and images. Furthermore, this design would fail to deliver anything more complex than texts and images.

The approach to the UI/UX prototyping was to create different interface mock

ups and exhibit them alongside existing solutions. An initial storyboard was drawn up and three potential interfaces (Figure 5.4 5.5 5.6) were designed and shown to stakeholders. The feedback gained from the stakeholders was invaluable in the process as it allowed for the group to consider all aspects of the prototypes. It was decided that a final version of the application's user interface would be designed, implementing all the positive attributes, and combining it into one (Figure 5.7), whilst also considering the negative attributes.

Prototype 1

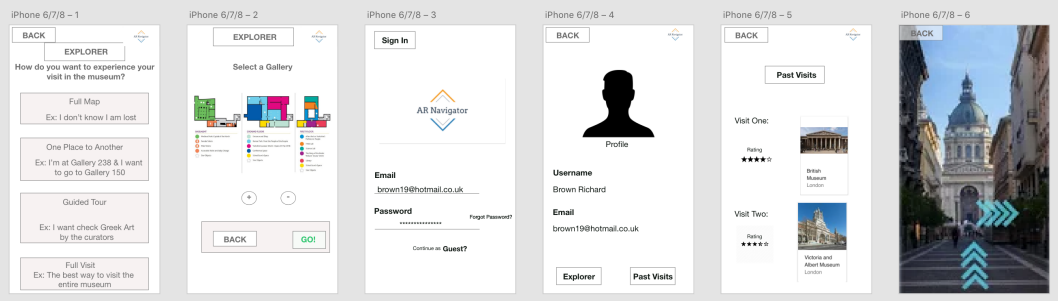


Figure 5.4: Overview of UI Prototype 1

Prototype 2

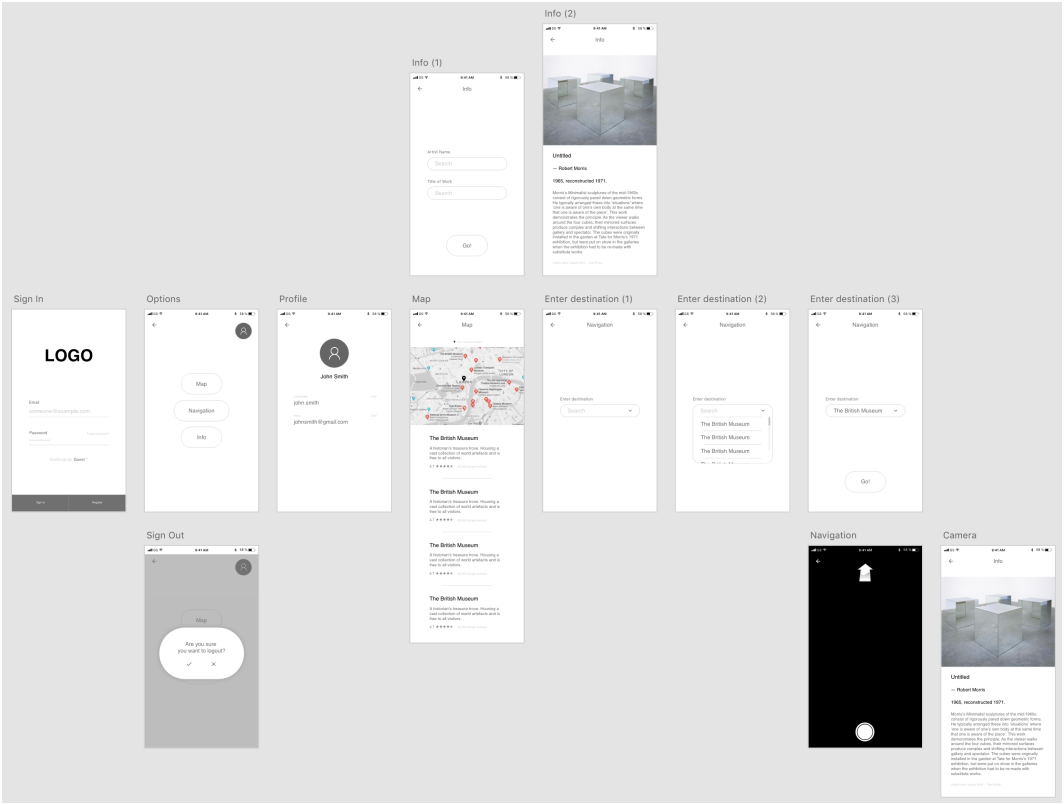


Figure 5.5: Overview of UI Prototype 2

Prototype 3

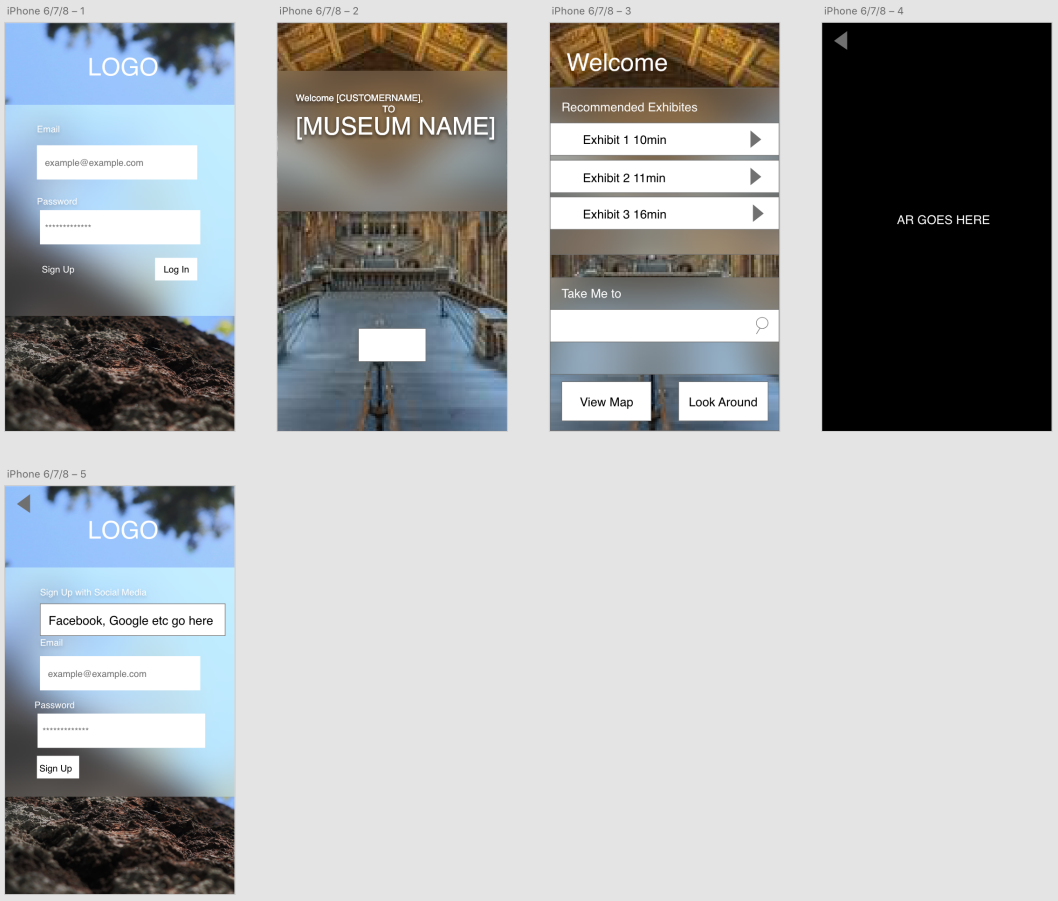


Figure 5.6: Overview of UI Prototype 3

Final Prototype

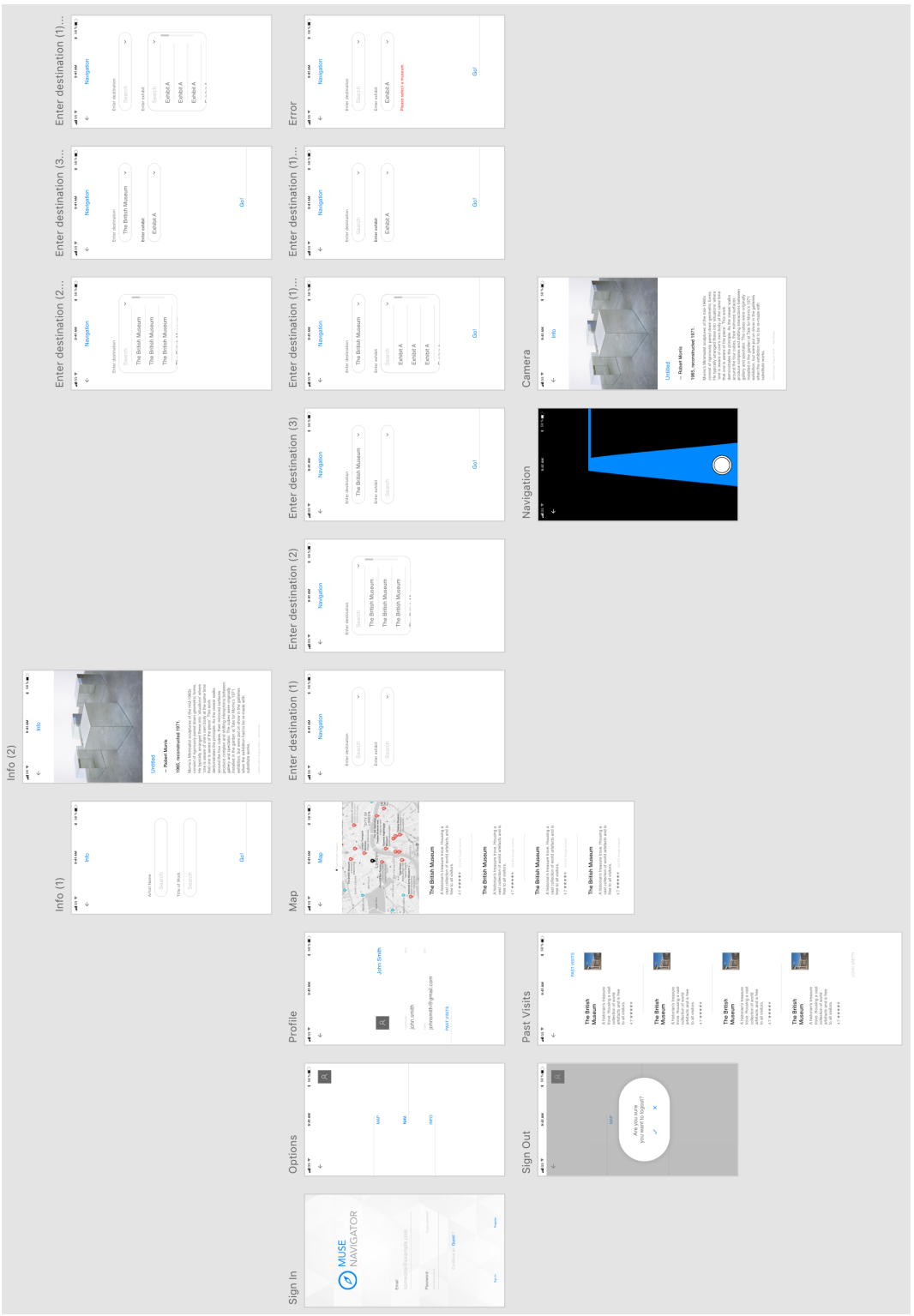


Figure 5.7: Overview of final UI prototype

5.4 Accessibility

Accessibility is about making the product to majority of end users. The application provides accessibility as part of the service we are offering to our audience and a way to make our app more generally appealing. Some accessibility options would include features such as relating to mobility, colour perception, and literacy.

1. **Screen Reading**: Users tend to rely on a screen reader to help them interact with the app, including UI elements, such as name, role, description, state and value. This feature would allow the user to use the app without any difficulties since the layout is simple and straightforward.

2. **Keyboard Accessibility**: This accessibility feature would allow users to interact with all UI elements within the application by keyboard only. It enables users to navigate through the app by using arrow keys. Allows the users to activate UI elements in the app by using Enter keys. Allows the users to enter their current and final location using the keyword.

3. **Scale**: We used scale to allow users to zoom and resize some elements, our main thought behind this was to help people with visual impairment, especially for images that include words. We also made sure to not start with a font size that is too small in general for many users, the reason behind this was because everyone's vision deteriorates as they age; and we wanted to make sure our app is usable for users of any age. In future iterations, we are thinking of allowing differences in vision, the way we are doing this will be by providing different scaling options for our users in the app settings. This means they can change the font size for easier reading or even shrink or enlarge the UI as well for better vision.

4. **Vision**: The text and UI in the app is designed to support high-contrast theme. Whilst colour is important, it must not be the only channel of communicating information. For instance, users who are colour blind would not be able to distinguish some colour status indicators from their

surroundings. Therefore, other visual cues such as text are included in order to ensure the on the app information is accessible to everyone.

5.5 User consultations

End-users and other key stakeholders such as an interested domain expert, Matt Isherwood, Managing Director at Fuse, were consulted during the design stage to clarify user needs and requirements. In addition to online interactions with stakeholders, the group had also conducted field research, and had face-to-face talks with potential users at the Natural History Museum, Science Museum, and the V&A museum to obtain a more nuanced input from key stakeholders. When prototyping the application, drawing and designing the user interface was not as simple as bringing ideas to life - it was imperative that users were consulted when making these decisions. More specifically, the group also consulted Fine Art students studying at the University of Oxford who provided a more in-depth perspective on user experience. These students were consulted to get a better understanding of what it is they would like for the application to provide. For example, some students had said that they would like for the application to provide further information about a specific exhibit that the museum does not have on display.

Upon consulting the users, it became apparently easier to actualize the designs put forward - acknowledging the fact that the users had all required a system that was relatively simple to navigate around. However, users were also consulted when it came to the technical aspect of the project. For example, deciding which medium would be best to track user location was one of the questions that arised in the early stages of development. In that regard, Isherwood was consulted about this, and brought to light the plethora of solutions that were readily available to begin development, primarily highlighting that the use of Bluetooth beacons would be most efficient to implement given the scope of the project.

Chapter 6

Implementation

6.1 Backlog

Prototype 1

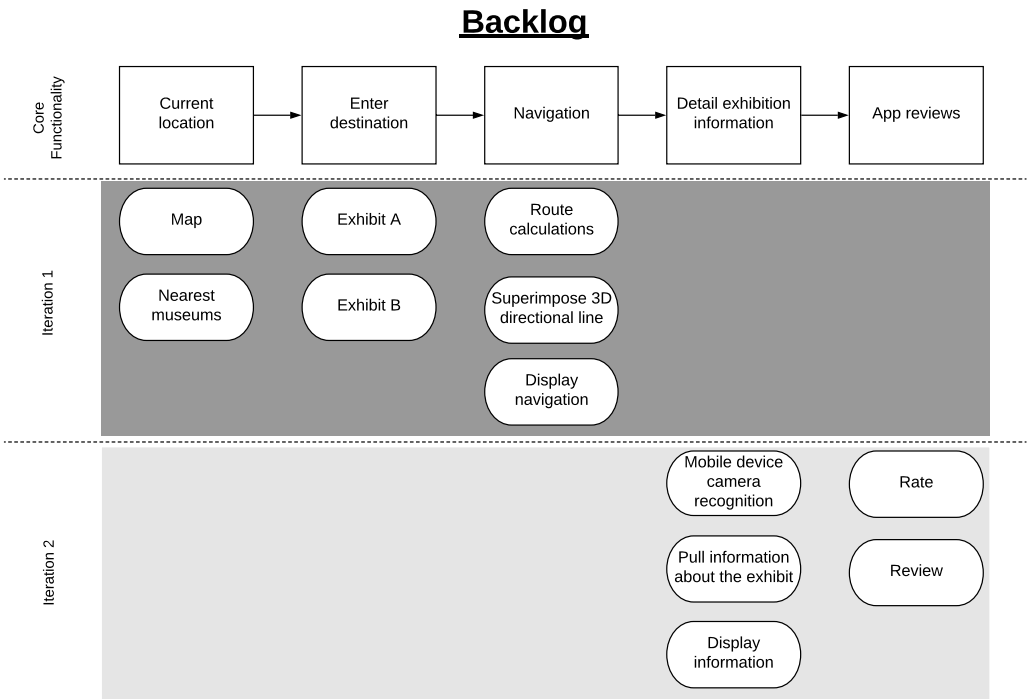


Figure 6.1: Inital Product Backlog

The product backlog was used as the single source of requirements in order to prioritise releases and allow for transparent iteration planning. From the requirements, and project scope each of the five core functionalities were broken down into its major components. Considering the time estimates, the scrum team decided that to demonstrate originality in the project, and to build the base functionalities first, both the route calculations and the AR implementation were to form the MVP. Features which required a database such as exhibition item/image recognition, and user profiles/logins were placed in the app's second release. These components would take longer than the amount of development time available along with the base functionalities. The team recognised the added time for the construction and integration of a database to the app would increase the risk in not delivering a functional MVP. As a living scrum artifact, the backlog changed accordingly when there were requirements, enhancements and fixes during the implementation and testing phases of each sprint.

Function	**Estimate Time (Days)**
Displaying map	0.5
Finding nearest museums	3
Getting start and end locations	5
Route calculations	10
Superimpose 3D directional line	12
Displaying navigation	10
Mobile device camera recognition	2
Retrieve exhibit information	10
Display exhibit information	1
Rating the app	1
Reviewing the museum visit	3

Table 6.1: Table of time estimates for each function

6.2 Sprint Outlines

During implementation, four sprints were conducted each lasting one to two weeks. The first sprint lasting one week with the goal of building a beacon for calculating distances between users and objects. The follow sprint focused on route calculations; being able to calculate the route between two points in a given space. For the AR part, this was divided into two separate sprint, with the initial one focusing on rendering and anchoring objects in real-time, and afterwards the integration with all base functionalities.

6.3 Front-end

Front-end development was carried out in Java on Android Studio.

It was carried out in the early stages of implementation due to the fact that the team had already recieved user feedback for the UI/UX prototypes. After combining the positive features of the initial design prototypes into one, it became easier to visualise the end-result. With aims to create an user interface that is user friendly and intuitive, various features of the application are self-explanatory.

As soon as the application is opened, the user is faced with the login page asking for credentials. The idea behind having a user account system is so that the user can keep track of their past visits to museums and review various sites; this was an idea that was put in place for future developments but was not included in the MVP. However, if the user does not have an account they are also able to 'Continue as a Guest' and use the services freely without saving any data. The forms are easy to fill in that does involve any technical complexity.

When the user has passed the login page, they are able to see the menu. The menu has three different services that the user can utilise. The first being the 'Map' which shows the user their current location and their surroundings on

a 2D map. The 'Nav' menu takes the user to the screen which the user has to then enter their desired destination or exhibit. After all the information has been entered by the user, the user's device displays the AR-enabled camera with the highlighted navigational line directing the user to their destination. However, if the user's device is not compatible with Google's ARCore SDK then the user will not be able to see this screen and are instead faced with an error message. Lastly, the 'info' page will allow the user to see additional information about a specific exhibit of their choice - this also requires the user to fill out a simple form.

As a result of all the planning and prototyping, when it came to actualising the design in Android Studio it was as simple as writing code for implementation - no further designing was needed, with exception of minor changes such as font sizes.

6.4 Back-end

6.5 Hardware

It was devised that the project needed to pursue a hardware solution which would be able to transmit the user's location to compute a route solution. Utilising the Arduino (UNO R3 Mega 2560)'s small foot-print, a simple beacon plan was sought consisting of the Arduino coupled with the Bluetooth HC- 05 RF transceiver.

Figure 6.2 displays the basic layout of the beacon aside from power input via the board's USB-B socket. This configuration was able to send the strength of the Bluetooth frequency thereby allowing the computation of a distance.

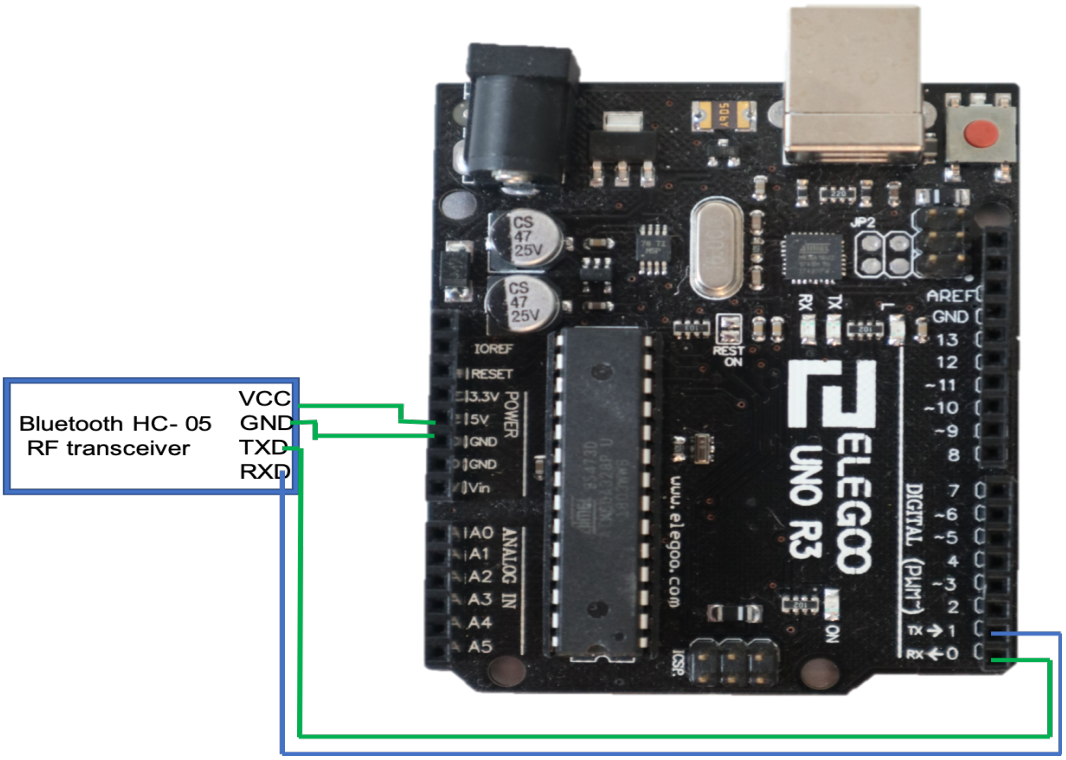


Figure 6.2: Arduino layout

However, this implementation had several failures; to get more accurate readings, the project would need at least three beacons to demonstrate triangulation wherein three signals are emitted from three angles to get an accurate location. During the testing there was only sufficient means for the procurement of one beacon – this led to an inability to fairly showcase the ideal usage, hindering the projects in this method.

While it was advised by our domain expert that this Bluetooth solution would be the best policy, it was not practicable given our resources. In view of these drawbacks, the hardware proposal was ceased. The research provided the team with enough of the methods and models to pursue a purely theoretical solution.

6.6 Ethical Audit

In order to be compliant with GDPR, user locations are recorded during the runtime of the application, and upon termination this data is deleted from memory in order to prevent unauthoried access to the user's location. Users are asked initially to give the Bluetooth, camera, and location permissions to use particular functions.

Since the app uses museum map, and exhibition details, the intellectual property for those will not be under the jurisdiction of the team. Rather, the project members owns the intellectual properties of all technologies that are being created and used in conjunction with third-party information. Since assigning data and storing details to specific users was not implemented in this iteration, in future versions the app will need to prevent attacks such as SQL injections.

6.7 Challenges

Chapter 7

Testing & Quality Assurance

7.1 Testing conducted

This section present the testing and quality assurance of our project. During development our team created various testing rule to improve maintainability and limit bugs in our application. We explore various testing techniques used in Software Development Life Cycle(SDLC) and Test-Driven Development(TDD). All testing list of the system has be detailed in the Appendix G (Testing).

7.1.1 Unit Testing

The unit testing was carried out in house by the team. The aim for this testing to evaluate the quality of the application and finding bugs to ensure the product works as expected. We have tested 14 different unit testing for our application and 5 of them test did not pass.

7.1.2 Integration Testing

It has been tested using incremental testing, taking on the 'top down' approach. As the nature of the application is an Augmented Reality navigation app, we

can not tested AR & Bluetooth functionality because it is still in progress to make the feature live.

7.1.3 Performance and stress testing

Performance testing examines stability, usability, reliability, responsiveness and speed of our application therefore testing was carried out to assess the performance of the application. Stress testing was carried out to check he upper limits of the application. As application will used in commercial space so our application need to make sure it does not crash during high volume of users. We have run and test our application in various devices to see the performance of each features in different mobile devices and also to check if the application crash during the installation process.

7.1.4 Regression testing

It is the process of testing changes to our application to make sure that the older iteration still works with the new features. Before releasing a new version of application, the old version are tested against the new version to make sure that all the old features still work with new program. During development, a two various branch was created in Git-Lab for route calculation. By testing and merging two branches will shows if previous iterations are still compatible with v.0.0.1 and v.0.0.2. During the testing, we pass this features in our application.

7.1.5 User Acceptance Testing (UAT)

The purpose behind UAT is to conform to the system being developed and ready for operational use according to the specified user requirements. We have tested seven various user acceptance where four of them can not be executed because the development team has not written the code. The only test we pass during testing was when the login my our application as guest.

During the testing we found that user can not use camera functionality and select destination. This feature will be conducted in future with our next version.

7.1.6 Beta Testing

This is the final stage of the testing, where the application released to an external test group consisting of real users. It helps us to find if their is any bugs in our application before releasing to public. We carried out one UI function testing where user can input credentials, destination and exhibit information. At the end, the program pass all the users input and shows the correct outcome.

7.2 Deployment

Hamza Week 2

After the application is ready and successfully released, a notification will be sent to stakeholders and clients. All iterations of the system will be detailed in the changelog.

Notification of deployment procedure (Taken from Appendix F):

1. Check all procedures and ensure everything is done.
2. Email client for meeting.
3. Present the client with the application.
4. Sign off documents.
5. Email client with application information.
6. Release of project and approval of supervisor.

To deliver a solution of quality, the project employs a set of practices of continuous automated integration and deployment throughout development. The

set of practices (properly pipelined, automated and frequently executed) allows for the developers to recieve feedback quickly about the quality of the software. To facilitate continuous integration effort the project uses GitLab's in house features.

Source Code Control (Management): The project has its own repository in GitLab so that the source code and supporting documents can be tracked, maintained, versioned, and audited.

Build Automation: Build Automation is handled using Android Studio. All separate software components of the project are Android projects. For example, the route calculations was considered as a separate project in the repository, and the AR was built in a separate Android project. This allows for a common build interface and specification of concise instructions for the components testing, building and upload to the project's distribution repository. This is the initial initial step for continuous integration.

Unit Test Automation: Android Studio has JUnit built-in which is an open source unit test framework. During development, unit testing becomes much easier to perform as Android Studio allows for this as a feature.

Deployment Automation: Using an online continuous integration and delivery platform such as CircleCI, it is possible to automate the deployment process through the use scripts. Google also provides an API to edit a Play-Store listing which will be used to upload the APK and publish it. This can be done directly through HTTP as it will have to be used in combination with CircleCI.

7.3 Formative evaluation

Everyone

7.4 Functional requirements review

To verify and validate that the requirements have been met, as mentioned previously, the team had adopted a test-driven development (TDD) approach which ensured that the team was working towards a fully functional system. The system successfully navigates a user from one point to another. This was achievable by displaying navigational routes in real-time and superimposing a 3-D line with AR technology. The quickest route was calculated by implementing the A* algorithm and tested in regression testing during development. However, other requirements have not been yet due to discussions made by the team regarding further iterations of the application. For example, although developed in protoyping - the requirement of implementing camera recognition on artworks/exhibits and displaying further information about them was not met but was decided to be inlcuded in further iterations of the application. However considering the complexity of the application, this was a reasonable decision to make as the Minimal Viable Product agreed upon by the team was to deliver the foundational service of the system: AR-assisted navigation. Lastly, the requirement of having the system suggest recommendations based on the user's route was also not met. However, this was also decided by the team to not be included in the first release of the application; suggesting recommendations is a feature that can only be implemented with a database. Due to the lack of resources available at hand, having an account database would be too costly as it would require the use of a server. Stakeholder feedback encouraged the introduction of an account system, therefore implementation of an account system is possible in further development of the application.

7.5 Non-Functional requirements review

As with the functional requirements, implementing a TDD approach greatly helped the team adhere to the requirements as much as possible, ensuring the foundational requirements are met. For example, the performance of the

system should not be slow - after a new feature was added to the system, the developers would then test-run the app to make sure that the new feature did not affect the performance. Similarly the usability was also tested to ensure ease of use - going back to the stakeholders to gain feedback on the simplicity of the navigation of the system. Once the stakeholders were happy, it meant that the requirement was met. In terms of the data usage of the application on the user's device; the only real data being sent out by the device is the user location and internet connectivity. It was imperative the system requested the user for location permissions and the application is also able to function on Wi-Fi so this requirement was always kept in mind during development. In terms of safety, development has not reached a level of complexity where the user would need to concerned about hitting an immutable object in their path - however, further development will always house this concern and take it into consideration. Having avoided the use of Bluetooth for route calculations, the concern of security in the user's location being accessed by third-party users was invalidated.

Chapter 8

Project evaluation

8.1 Summative evaluation

8.2 Future developments

By design, this project is geared towards to meeting the requirement of the museum sector, including public galleries. However, the main objective navigation, being so widely applicable, with adequate research there are many prudent areas of development. However, with the saturation of navigation and mapping software, this project should strive to stick to the indoor avenue.

In a commercial context the most important future development would be retail navigation. As the product is based on finding the quickest route to a point in a museum, taking the project to this stage would not require any fundamental change. As in the museum concept, a retail version would again involve finding the best solution in terms of pure navigation and also proposing certain shops in-keeping with the user's preferences. However, it should be made clear that based on our current model which involves studying the layout of a candidate area, a need to find a quicker method of digitising a space is desirable.

Appendix A

User & Stakeholder Research

Appendix B

User Stories

Use Case Model

Two scenarios have been taken into account, where the user gets lost in the museum, and the user wants to explore the museum. When a user is lost, they need to enter their destination where the app will receive their current location, and find the quickest route from the user's current position. The user follows that navigation until they arrive at their destination. For the exploration, the app will show the details where user know what they going to see in the museum.

Activity Model

This is based on the back-end of the application for example when the user searches about the museum, this history saved in the server where if the user wants to go to the same place then they can use our function called past visit.

User & Acceptance Stories

This will describe what will be achieved once the application is ready to be used by the user. A diagram has been created based on different scenarios where it can be found if the application has achieved the user needs.

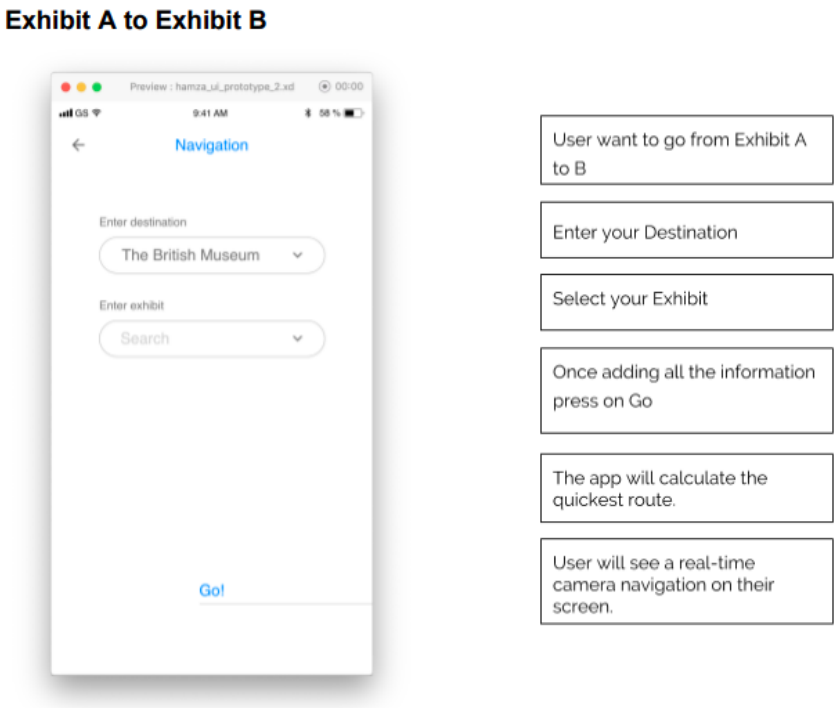


Figure B.1: Going from point A to point B

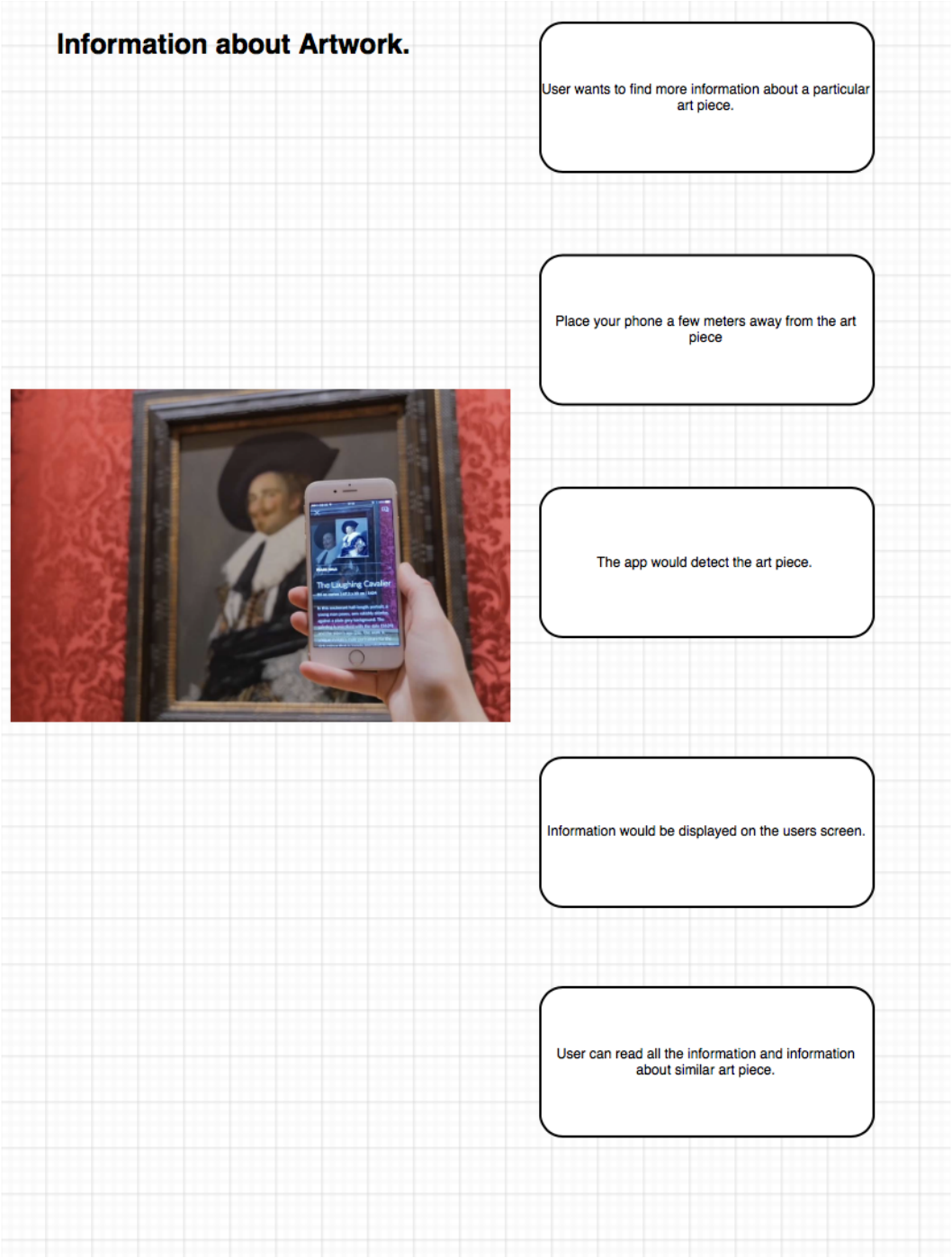


Figure B.2: Getting information from exhibition

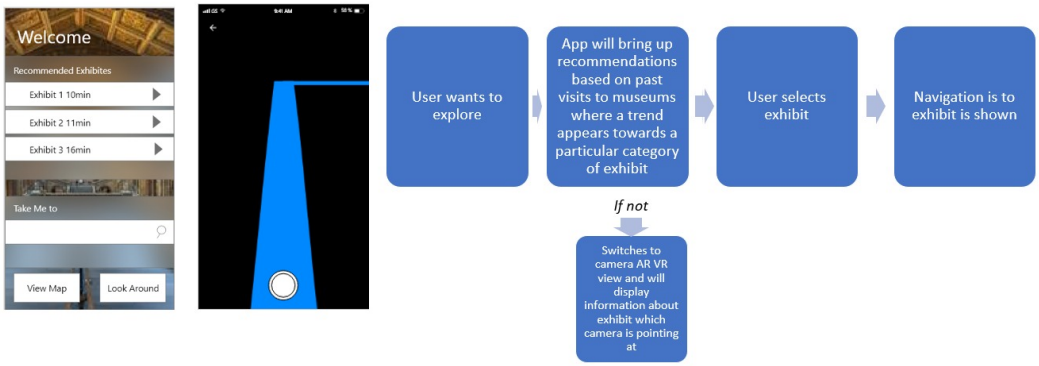


Figure B.3: Exploring the museum

Appendix C

Systems Requirements Specification

Purpose

The main goal of this concept is to provide users a solution to indoor museum navigation through an exciting, and enjoyable experience using augmented reality. It includes users being lost, or searching for a specific location within the museum. It was discovered through field research that the concept would make life easier for users and the museums since it would allow easy access to the information based on exhibitions.

Scope

This project will include creating an AR application for people to get an enjoyable journey in the museum. The project will be completed by 29 April 2019. The AR application will include simple navigation system to direct various part of the museum. Getting information on the user screen using the user's camera, and explore various museum using the system. The platform will be developed on Android due to the high usage of Google's AR library,

ARCore.

System Overview

The application will perform all the basic tasks to help users with their journey in the museum. Such as navigating from point A to B, getting the user back on track in case they are lost, allowing the user to view information based on camera recognition of an exhibit.

References

This specification should be read in conjunction with the following publications:

IEEE 24765-2017 - ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary [15]

IEEE Std 29148-2011, ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering [16]

IEEE Std 730-2014, IEEE Standard for Software Quality Assurance Processes [17]

IEEE Std 24748-4-2016 - ISO/IEC/IEEE International Standard for Systems and Software Engineering – Life Cycle Management – Part 4: Systems Engineering Planning [18]

Definitions

Activity: A set of cohesive tasks of a process, which transforms inputs into outputs. [ISO/IEC/IEEE 12207:2008]

Augmented reality: A technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view.

Functional requirement: A requirement that specifies a function that a system or system component must perform. [ISO/IEC/IEEE 24765:2010]

Non-functional requirement: The measurable criterion that identifies a quality attribute of a function or how well a functional requirement must be accomplished. A non-functional requirement is always an attribute of a functional requirement. [ISO/IEC/IEEE 730:2014]

Performance: Degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage. [ISO/IEC/IEEE 24765:2017]

Stakeholder: Individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their need and expectations. [ISO/IEC/IEEE 15288:2015]

Usability: Extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. [ISO/IEC/IEEE 25064:2013]

User: Individual or group that interacts with a system or benefits from a system during its utilisation. [ISO/IEC 25010:2011]

Use Cases

The use cases have been defined as follows:

1. Use Case Model
2. Activity Model
3. User & Acceptance Stories
    (a) In Exhibit going from A to B

(b) Getting information from an exhibition

(c) Exploring the museum

(d) User get lost in the museum

Use Case Model

Two scenarios have been taken into account, where the user gets lost in the museum, and the user wants to explore the museum.

When a user is lost, they need to enter their destination where the app will receive their current location, and find the quickest route from the user's current position. The user follows that navigation until they arrive at their destination. For the exploration, the app will show the details where user know what they going to see in the museum.

Reference to use case model

Activity Model

This is based on the back-end of the application for example when the user searches about the museum, this history saved in the server where if the user wants to go to the same place then they can use our function called past visit.

Reference to activity model

User & Acceptance Stories

This will describe what will be achieved once the application is ready to be used by the user. A diagram has been created based on different scenarios where it can be found if the application has achieved the user needs.

Reference to the 3 user stories

Functional Requirements

1. Needs to be able to navigate the user from point A to B.
2. The system should be able to display navigational routes in real-time.
3. It should be able to calculate the quickest route.
4. A 3D line should be superimposed through augmented reality to display the navigation to the user's destination.
5. Camera recognition on artwork/exhibits, displaying further information about the exhibit.
6. When user arrives at destination, the system should give a recommendation based on their route.

Non-functional Requirements

1. **Performance**: The system should respond quickly to user input, e.g user wants to find more information about an art piece or whenever they search up a location. The system should not require extensive CPU usage, it should not slow the device down inconveniencing the user.
2. **Usability**: The system should have a simple layout, with appropriate colour used in appropriate contexts. The language used on the app should be easy to understand for the users. Having done research based on the user preference, it was discovered that users dislike too much text on their menu screen.
3. **Data Usage**: Data usage should be kept to a minimum, only querying the relevant information (user location and exhibit information). Also, the app would require internet connection in order to calculate the real-time distance of the final destination from the user's current location.
4. **Safety**: The system needs to have the ability to detect immutable objects obstructing the user's path. This can reduce common user accidents

when using a mobile phone whilst walking.

5. **Security**: Ensuring that the device's current location cannot be obtained by unauthorised third-party users is crucial in ensuing the security of using the platform.

Appendix D

Documentation Plan

Introduction

This documentation plan outlines the strategy for creating all documentation associated with the software release. This document is addressed to project team, and supervisors to inform them about the documentation efforts that is undertaken for the release.

Scope

The documentation plan includes the development of updates to all users and developers that are required for the software release. Specifically, it covers the creation and updating:

- User guides
- Product website
- Release notes

Scope of the development activity providing updates to the above documents. These activities requires the involvement of user testing.

Assumptions

It is assumed that readers of the document are familiar with the previous stages to the project and the associated strategies in place. It is also assumed that the required resources will be available to achieve the objectives of the plan, and that there are no risks other than those identified in the section on Risks.

Constraints

Constraints on this documentation project are the available time from the resources as outlined at the start of the project along with the product delivery schedule. Changes or delays in product delivery will affect the documentation plan.

Existing Documentation

This document should be read in conjunction with:

- Proposal
- Testing plan
- Gantt chart
- Application release notes

Documentation Specifications

Platforms

All documentation is accessible on all platforms via a PDF, and on all browser-compliant platforms.

Distribution & Delivery

PDFs of all documentation will be available on the product's website.

Terminology

Terminology will be maintained throughout the documentation as of the proposal document.

Process & Schedule

Activities

The following activities will be undertaken to produce the documentation:

- Creating indexes for user guides.
- Merger of all application notes from previous releases into guides.
- Documenting source code and approaches.
- Creating testing documentation.
- Creating release notes.
- Creating read me files for each component.

Milestones

Given the diversity of activities, and information streams, estimated milestones are based on the current availability of required resources:

Change Control

Change control for documentation is similar to changes in the source code:

Milestone	**Delivery Date**
Implementation Ends	4th March 2019
Updated files to reviewers	6th March 2019
Initial review complete	29th March 2019
Revisions complete	22nd April 2019
Review Complete	25th April 2019
Program and Report Release	29th April 2019

- During documentation development, changes and error corrections are communicated directly with the appropriate author.
- After the end of the implementation phase, changes or corrections are communicated in the same way as above, but the author is responsible for prioritizing the requested fixes to determine which ones should be made in the remaining time before release.
- Major documentation changes shall be treated the same as bug releases, and will be handled in conjunction with the next applicable major release.

Risks

The risks identified have a potential to affect the delivery schedule:

- Due to the volume of changes and enhancement to the product throughout the development process, so long as the scope has been correctly identified, this document can be time appropriate to all of the development activities.
- If there are changes to the scope, the depth of coverage of the documentation may be amended, or the target date extended.
- Delays in turnaround of reviews prevent on-time delivery. To reduce this risk, authors of the document will have as much advance notices as possible of the requirement for a review.

Issues

None found at the time of publication.

Appendix E

Testing Plan

Introduction

Purpose

This testing plan in accordance to the IEEE Standard for Software Test Documentation (ANSI/IEEE Standard 829-1983), outlines and describes the testing approach and overall framework that will drive the testing of the implementation phase of the project.

The document outlines:

- Testing Strategy: structure and descriptions of testing.
- Execution Strategy: describes how the test will be performed and processed to analyse defects to the platform, and resolutions to the defects.
- Test Management: processing how to deal with testing platforms and events that take place during execution.

Audience

- Project members will conduct tasks specified in this document, and provide working updates to it. All members will be accountable for the

results.

- The project lead plans the testing activities in the overall project schedule, and tracks the performances of the test.
- The project supervisor will ensure that the plan is met by the team and provide further test cases if necessary to important functionalities.

References

This document should be read in conjunction with:

- Proposal
- GitLab repository testing plan
- Gantt chart

Objectives and Tasks

Test Objectives

The objective of testing is to verify the functionality of the platform is in accordance to the outlines of the proposal. Test execute and verify test scripts, identifies and fixes various levels of defects.

Assumptions

General:

- During the execution of testing, the current project plan acts as a precondition.
- Software delivered by from the development side must be in accordance with the development plans so it is functionally usable and in testable units.

- The quality of the development tests are to be performed in the agreed manner and thoroughness.
- Testers for each sprints should be available in accordance with the test schedule.
- Defects will be tracked through GitLab. Any defect fixes planned will be shared with Test Team prior to applying the fixes on the Test environment.
- There is no environment downtime during test due to outages or defect fixes.
- The system will be treated as a black box; if the information shows correctly on device, and in the reports, it will be assumed the program is functioning.

UAT:

- UAT test execution will be performed by end users and the team will create the UAT script.

Testing Strategy

There are three key aspects to this version of the platform. For both route calculations and augmented reality implementation, unit and integration testing will be pivotal in ensuring these functions are implemented rigorously. UI testing will mainly focus on unit testing but user acceptance testing will be heavily featured to match the requirements of the stakeholders.

Unit Testing

The unit testing will be carried out in-house by the team. The application will be tested when development has reached 70% or above. The goals of this

testing will be to evaluate the quality of the application, finding any bugs, ensuring the product works and is ready to be tested by the users.

Entry Criteria

- 70% – 90% complete and assurance to go ahead with unit testing
- Unit test cases should be designed and reviewed
- Testing environment set up and and stability confirmed

Exit Criteria

- All test cycles should be complete
- All the unit tests should be executed and Passed
- Alpha version of the application frozen (i.e., no additional features, no modifications to existing features, no dropping of the existing features)
- Unit test formal Sign-Off

System and Integration Testing

This type of testing will verify the behaviour of the integrated hardware and software environment of the complete system. Helping to evaluate the system's compliance with its specified requirements.

Integration will be tested using incremental testing, taking on the 'top down' approach. First testing each module of the application individually and then continue testing by considering other modules in addition. As the nature of the application is an Augmented Reality navigation app, testing the Bluetooth functionality prior to testing the AR functionality is one example of how integration testing would occur. This would be followed by testing of both these modules combined.

Performance and Stress Testing

Performance testing examines responsiveness, stability, scalability, reliability, speed and resource usage of the software and infrastructure. As development of the application will be done under the agile methodology, continuous testing will be carried out to assess the performance of the application.

Stress testing will be carried out to check the upper limits of the application, testing it under extreme loads. As the software developed will be an application used for navigational purposes in a commercial space, an example definition of a test case would be with a very high number of users, which is known as a 'Spike Test'.

User Acceptance Testing (UAT)

UAT will be performed at the very end of development, prior to the product going live. This testing will be carried out by real users who will decide whether or not the acceptance requirements provided by the team are to be accepted or rejected. One example of an acceptance requirement would be "the route calculation must be accurate". This is an optimal phase for also identifying any bugs.

Acceptance criteria will be gathered by the team with real users comparing the system to the initial requirements. During testing, the team will **Assist In UAT**, who will be on stand-by to help users in the case of any difficulty. However, the main reason for the team to be on stand-by is to record results and log any bugs etc.

Beta Testing

Beta testing is the final stage of the testing phase, where the application will be released to an external test group consisting of real users. The main entry criteria being that the development should be 90% - 95% completed, all

components either fully or almost complete for testing. At this point in testing, the unit testing should be signed off. Any bugs identified will be handled promptly and feedback analysed to ensure application satisfies the user. Test cases written in this phase will be clearly outlined, defining which feature is being examined, such as UI, Bluetooth recognition, location handling, route calculations etc.

Validation and Defect Management

- It is the responsibility of testers to open defects and link them to the corresponding code, and assign an initial severity and status, before retesting and closing the defect. It is the responsibility of the project lead to ensure the defects are fixed in a timely manner and according to the project and testing plan.

Severity categories (from softwaretestinghelp.com):

1. Critical - The bug is critical enough to crash the system, or cause potential data loss. It causes an abnormal return to the operating system. Or it causes the application the hang and requires a re-boot of the system.
2. High - It causes a lack of vital program functionality with workaround
3. Medium - This Bug will degrade the quality of the System. However there is an intelligent workaround for achieving the desired functionality. Or this bug prevents other areas of the product from being tested. However other areas can be independently tested.
4. Low - There is an insufficient or unclear error message, which has minimum impact on product use.
5. Cosmetic - There is an insufficient or unclear error message that has no impact on the product use.

Testing metrics allows the measurements and level of success of test that will be developed with the project lead.

- Test preparation and execution status - To report on % complete [daily/weekly]
- Daily execution - To report on pass, fail, total defects, critical defects [daily]
- Project weekly status - Project driven reporting (as requested by supervisor) [weekly]

Hardware Requirements

- Raspberry Pi with Bluetooth beacon
- Mobile device with Bluetooth beacon
- Android device with Android 5.0 (Lollipop) or higher

Test Schedule

Tests will be executed during each sprints as outlined in the Gantt chart, and a final testing stage will be completed for the testing milestone.

Control Procedures

Problem Reporting

If there are defective software found during testing, the project lead will assign the defect to the team, and fix it before it is sent back to testing. Project lead will require approval to ensure the updated software matches the requirements of the test.

Change Requests

If modifications to the software are made, the project lead is required to sign off the changes and review the changes to the current platform. If there are changes that will affect the existing platform, then these particular modules will have to be identified.

Features to Be Tested

To be read in conjunction to the backlog:

- Receiving user inputs for user destination
- Route calculations
- Superimposing 3D directional line
- Display navigation

Features Not to Be Tested

These features will appear in later iterations. This is due to the short implementation time available for the project.

- Finding museums nearby
- Mobile device camera recognition
- Receiving and displaying information about the exhibit
- Rating and dealing with user reviews of platform

Risks/Assumptions

Tools

All tests will be mainly conducted on the Android Studio testing suite. JUnit will specifically conduct unit testing, and GitLab will have continuous integration and continuous delivery in order to ensure integration to the current platform is successful.

All testing artifacts such as the test cases themselves are stored on the GitLab repository.

All tests should be tested on devices higher than Android 5.0 (Lollipop) that have allowed Bluetooth to be used.

Appendix F

Deployment Plan

Introduction

Purpose

The purpose of the deployment plan is to ensure that the system successfully reaches its users and new features to the system are delivered successfully. The aim of the deployment plan is to provide a detailed schedule of events, persons responsible, and dependencies required to integrate the new version of the app with the previous version. It should minimize the impact of the integration of the new system on the users and stakeholders.

Assumptions

The application would have a place for users to:

- Have a login activity, where the user can enter credentials to login, or continue as a guest.
- To login to the application, enter their location, and destination.
- Route calculation takes place (finding the shortest route to the destination).

- The user can retrieve their current location, and map.
- The user can take a quick snap of a exhibit and the application provides more information about it.

Dependencies

Dependencies that can hinder or slow the process of deployment are:

- Dependent on using an Android device
- Dependent on Google's ARCore Software Development Kit (SDK)
- Dependent on Google's Map services
- Unavoidable change of plans
  - Change of user requirements
  - Research fails
  - Implementation fails
- Time management
  - Group meeting
  - Supervisor meeting
  - Weekly delegated tasks
  - Milestones

Constraints

- Reliance with Google's map services (server can crash)
- Repository could be down
- Group member availability:
  - Not all members are mutually available

  - Conflicting time schedules

- Internet Connection/Wi-Fi issues
- Database error: Existing accounts may not be able to login if the DB server is down

Assessment of Deployment Readiness

1. Verify that the application does not have any broken links and that all content is accessible.
2. Verify that all dependent files have been uploaded to the relevant directories so that they can be accessed from other calls.
3. Supervisor approval

Product Content

Configuration would include the following:

- Accuracy and reliability of separation of programming plans by members.
- How easy to download all the documentation, report or code from gitlab.

Deviations and Waivers

Deviations from the original plan included:

- Implemented Route Calculations using Wi-Fi instead of Bluetooth
- Changed our 3D Model from a directional arrow to a navigational line
- Implemented outside commercial spaces instead of within a museum setting - to deliver applicable concept

Phase Rollout

Phase I

- Map showing user's current location
- Route calculations
- Superimposed 3D directional line
- Display navigation

Phase II

- Show nearest museums to the user's current location
- Camera recognition of exhibits
- Request and pull information about exhibit
- Display the information

Phase III

- Account database
- Registered users can store their visited museums
- User can rate and review the visited museums

Notification of Deployment

After the application is successfully released, a notification will be sent to stakeholders and clients. All iterations of the system will be detailed in the changelog.

Steps

1. Check all procedures and ensure everything is done.
2. Email client for meeting.

3. Present the client with the application.
4. Sign off development plan documents.
5. Email client with application information.
6. Release of project and approval of supervisor.

Deployment Systems

Continuous Integration and Continuous Delivery (CI/CD) will be used to deliver any changes to the system. Automated tests are written to for each new feature to ensure that less bugs are passed to the production stage and captured early by regression, reducing the risks at every release. Gitlab have built-in tools to support CI/CD, which provides a simplified setup and execution of software development using continuous methodology.

DevOps

Much like Agile development, adopting a DevOps culture allows for smoother processes within development.

- **Building the right product**: After code has been written, the team will be able to receive faster feedback as a result of live testing.
- **Improved productivity**: As delivery would be continuous, the developers and testers will be additionally efficient as testing environments are easier to set up (see A/B Testing below).
- **Reliable releases**: Smaller and more frequent releases would allow for less changes made to the code as well as the bugs.

A/B Testing

The main idea behind A/B Testing is that testing can be performed on variants based on live environments. Since development followed the agile methodology as well conforming to a test-driven development approach - testing was performed in conjunction with the coding. After new code has been written and when testing is required, the environment would already be set up for current and future tests.

Appendix G

Testing

Appendix H

User & Stakeholder Feedback

Bibliography

[1] O. D. Online. (2018). Augmented reality, [Online]. Available: https://en.oxforddictionaries.com/definition/augmented_reality (visited on Nov. 9, 2018).

[2] I. D. Foundation. (Sep. 2018). Augmented reality – the past, the present and the future, [Online]. Available: https://www.interaction-design.org/literature/article/augmented-reality-the-past-the-present-and-the-future (visited on Nov. 9, 2018).

[3] A. Jamieson, "Snapchat users paying up to thousands for custom filters to celebrate life events", The Guardian, May 2016. [Online]. Available: https://www.theguardian.com/technology/2016/may/03/snapchat-geofilter-custom-filter-prom-wedding (visited on Dec. 1, 2018).

[4] O. Group. (2018). Mobile applications, [Online]. Available: https://orpheogroup.com/uk/mobile-applications/.

[5] Microsoft. (2017). Path guide: A new approach to indoor navigation, [Online]. Available: https://www.microsoft.com/en-us/research/blog/path-guide-new-approach-indoor-navigation/.

[6] A. Murphy. (2017). Retail in museums: Continuing the visitor experience through the museum shop, [Online]. Available: https://advisor.museumsandheritage.com/features/retail-museums-continuing-visitor-experience-museum-shop/.

[7] Mapspeople, 2018. [Online]. Available: http://www.mapspeople.com/.

[8] AXELOS, Manging Successful Projects with PRINCE2. TSO, 2014.

[9] Smartsheet. (2018). What's the difference? agile vs scrum vs waterfall vs kanban, [Online]. Available: https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban.

[10] Globalluxsoft. (Oct. 2017). 5 popular software development models with their pros and cons, [Online]. Available: https://medium.com/globalluxsoft/5-popular-software-development-models-with-their-pros-and-cons-12a486b569dc.

[11] K. Schwaber and J. Sutherland, "The scrum guide - the definitive guide to scrum: The rules of the game", p. 7, Nov. 2017. [Online]. Available: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf.

[12] APD, Scrum summary sheet, 2017.

[13] A. Bulajic, S. Sambasivam, and R. Stojic, "Overview of the test driven development", Proceedings of Informing Science and IT Education Conference (InSITE) 20 pp. 165–187, Jun. 2012. [Online]. Available: http://proceedings.informingscience.org/InSITE2012/InSITE12p165-187Bulajic0052.pdf.

[14] K. D. Science and Engineering. (2018). Software engineering fundamentals - best practices, [Online]. Available: https://blog.k2datascience.com/software-engineering-fundamentals-best-practices-b5105d155c6d.

[15] IEEE, "International standard - systems and software engineering -vocabulary", ISO/IEC/IEEE Std 24765-2017, 2017. DOI: 10.1109/IEEESTD.2017.8016712.

[16] ——, "International standard - systems and software engineering – life cycle processes –requirements engineering", ISO/IEC/IEEE Std 29148-2018, 2018. DOI: 10.1109/IEEESTD.2018.8559686.

[17] ——, "Standard for software quality assurance processes", IEEE Std 730-2014, 1998. DOI: 10.1109/IEEESTD.2014.6835311.

[18] ——, "International standard for systems and software engineering – life cycle management – part 4: Systems engineering planning", IEEE Std 24748-4-2016, 2016. DOI: 10.1109/IEEESTD.2016.7470727.