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

Draft: April 1, 2019

Spring 2019

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

Abstract

The use of mobile augmented reality by consumers, and research in the field has become more prominent in the last decade. This has allowed for completely new approaches in solving current problems using this technology as there is a year-on-year increase on smartphone users across the world.

This proposal presents the use of augmented reality in museum navigation on mobile devices. After conducting stakeholder research, there were clear issues presented by current solutions on the market through the form of paper maps. Augmented reality library research was conducted on various platforms to find the appropriate toolkit for the proposed system, and UI/UX prototyping prioritised key design aspects of the system. Following this, the technical architecture and user stories are defined through the model-view controller architectural pattern, along with technologies to be used during implementation. Methods and approaches to implementation are outlined, namely through the agile methodology along with consulting various testing methods.

Contents

List of Figures

List of Tables

Acknowledgements

Chapter 1

Introduction

1.1 Motivation

1.2 Purpose & Scope

1.3 Assumptions

1.4 Coverage

Chapter 2

Background and Literature Review

2.1 Background

2.2 AR Libraries

2.3 Software Architecture

2.4 Hardware - Arduino and Raspberry Pis

Chapter 3

Project Management Processes

3.1 Agile vs Waterfall vs Lean

3.2 Software Development Lifecycle

3.3 Test-Driven Development

3.4 Repository Management

Chapter 4

Requirements

4.1 Gathering

4.2 Stakeholders

4.3 System

4.4 Functional

4.5 Non-Functional

Chapter 5

Design

5.1 Models

5.1.1 Use Case

5.1.2 Activity

5.1.3 Sequence

5.2 User Interface

5.3 Accessibility

5.4 User consultations

Chapter 6

Implementation

6.1 Backlog

6.2 Sprint Outlines

6.3 Front-end

6.4 Back-end

6.5 Hardware

6.6 Challenges

Chapter 7

Testing & Quality Assurance

7.1 Testing conducted

7.1.1 Unit Testing

7.1.2 Integration Testing

7.1.3 Performance and stress testing

7.1.4 Regression testing

7.1.5 User Acceptance Testing (UAT)

7.1.6 Beta Testing

7.2 Deployment

7.3 Formative evaluation

7.4 Functional requirements review

7.5 Non-Functional requirements review

Chapter 8

Project evaluation

8.1 Summative evaluation

8.2 Future developments

Appendix A

User & Stakeholder Research

Appendix B

User Stories

B.1 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.

B.1.1 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.

B.2 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.

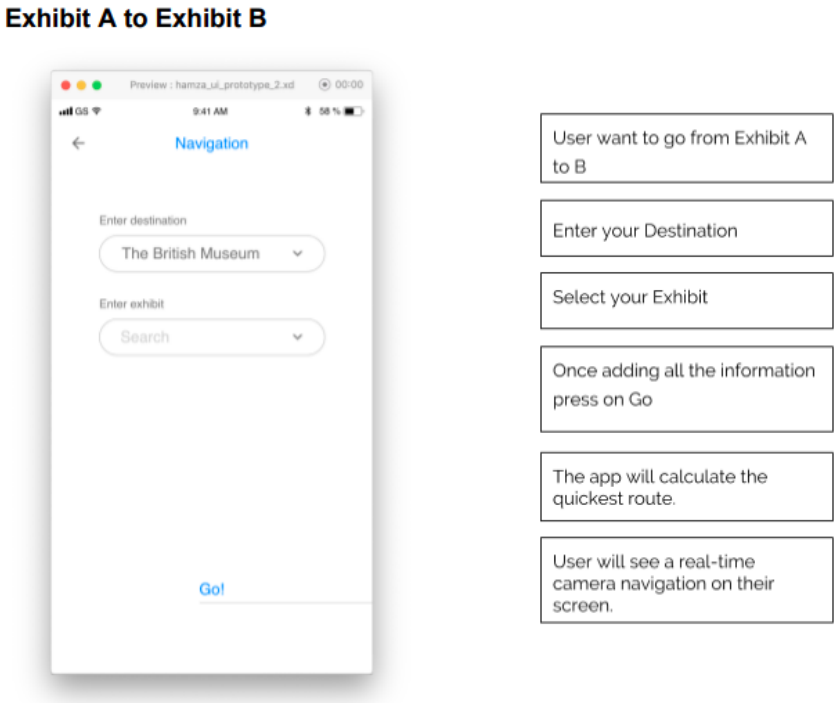

Figure B.1: Going from point A to point B

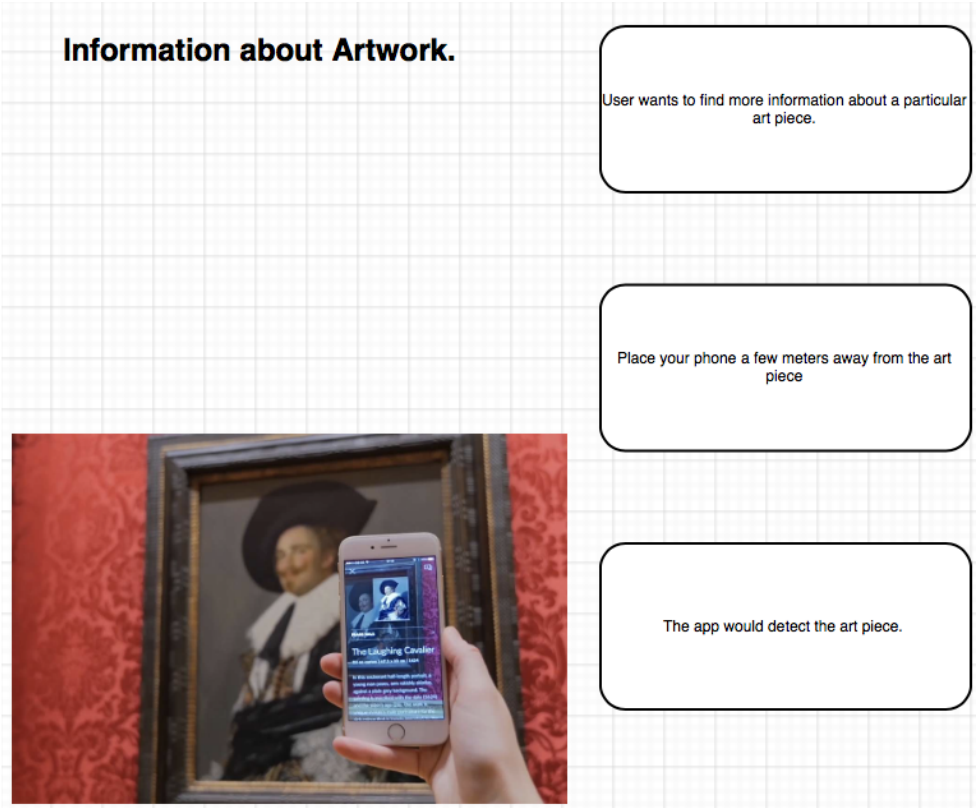

Figure B.2: Getting information from exhibition

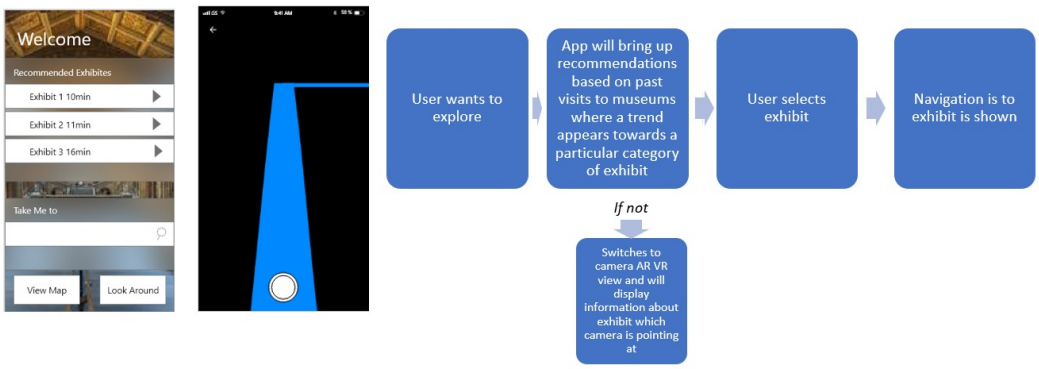

Figure B.3: Exploring the museum

Appendix C

Documentation Plan

C.1 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.

C.2 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.

C.3 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.

C.4 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.

C.5 Existing Documentation

This document should be read in conjunction with:

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

C.6 Documentation Specifications

C.6.1 Platforms

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

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

C.6.2 Distribution & Delivery

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

C.6.3 Terminology

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

C.7 Process & Schedule

C.7.1 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.

C.7.2 Milestones

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

C.7.3 Change Control

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

- 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.

C.8 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.

C.9 Issues

None found at the time of publication.

Appendix D

Testing Plan

D.1 Introduction

D.1.1 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.

D.1.2 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.

D.1.3 References

This document should be read in conjunction with:

- Proposal
- GitLab repository testing plan
- Gantt chart

D.2 Objectives and Tasks

D.2.1 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.

D.2.2 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.

D.3 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.

D.3.1 Alpha Testing (Unit Testing)

The alpha 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 Alpha Testing
- Alpha 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 Alpha 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)
- Alpha Test formal Sign-Off

D.3.2 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.

D.3.3 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'.

D.3.4 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.

D.3.5 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 Alpha 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.

D.3.6 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]

D.4 Hardware Requirements

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

D.5 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.

D.6 Control Procedures

D.6.1 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.

D.6.2 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.

D.7 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

D.8 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

D.9 Risks/Assumptions

D.10 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 E

Deployment Plan

E.1 Introduction

E.1.1 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.

E.1.2 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.

E.1.3 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

E.1.4 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

E.2 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

E.2.1 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.

E.2.2 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

E.3 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

E.4 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.

E.4.1 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.

E.5 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.

E.6 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.

E.6.1 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 F

Testing

Appendix G

User & Stakeholder Feedback