Department of Computing
Goldsmiths, University of London

Augmented Reality Navigation System for Commercial Spaces

Testing Plan

by

Arif Kharoti, Nicholas Orford-Williams, Hardik Ramesh, Gabriel Sampaio Da Silva Diogo, Hamza Sheikh, Jonathan Tang

Software Projects – Group 14

Version 1.0
Spring 2019

1 Introduction

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.

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.

1.3 References

This document should be read in conjunction with:

- Proposal
- GitLab repository testing plan
- Gantt chart

2 Objectives and Tasks

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.

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.

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.

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

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.

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'.

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.

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.

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]

4 Hardware Requirements

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

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.

6 Control Procedures

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.

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.

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

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

9 Risks/Assumptions

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.