**Software Test Plan**

**for**

**Lightweight Educational Assistant (LEA) the Lego Robot**

**Prepared by FLAIR**

**Lauren Massey**

**Jessica Updegrove**

**Embry-Riddle Aeronautical University - SE555**

**Version: 1 3/29/2016**

**Table of Contents**

**1. Test Plan Identifier (Jess)**

**2. Introduction (Jess)**

2.1 Objectives

2.2 Testing Strategy

2.3 Scope

2.4 Reference Material

2.5 Definitions and Acronyms

**3. Test Items (Jess)**

3.1 Program Modules

3.2 Job Control Procedures

3.3 User Procedures

3.4 Operator Procedures

**4. Features to be Tested (Jess)**

**5. Features not to be Tested (Jess)**

**6. Approach (Lauren)**

6.1 Component Testing

6.2 Integration Testing

6.3 Conversion Testing

6.4 Job Stream Testing

6.5 Interface Testing

6.6 Security Testing

6.7 Recovery Testing

6.8 Performance Testing

6.9 Regression Testing

6.10 Acceptance Testing

6.11 Beta Testing

**7. Pass/Fail Criteria (Lauren)**

7.1 Suspension Criteria

7.2 Resumption Criteria

7.3 Approval Criteria

7.4 Individual Test Cases

**8. Testing Process (Lauren)**

8.1 Test Deliverables

8.2 Testing Tasks

8.3 Responsibilities

8.4 Resources

8.5 Schedule

**9. Environmental requirements (Jess)**

9.1 Hardware

9.2 Software

9.3 Security

9.4 Tools

**Table of Contents (Con’t)**

95 Publications

9.6 Risks and Assumptions

**10. Approvals (Jess)**

**1. Test Plan Identifier (Jess)**

lea.test.1

**2. Introduction (Jess)**

The Software Test Plan (STP) is designed to prescribe the scope, approach, resources, and schedule of all testing activities. The plan must identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

**2.1 Objectives**

The testing activities for LEA will be focused on exposing any major operating flaws. The scope of this testing will be for individual components and component interaction. It will not include testing by consumers. Testing will be approached on an individual basis, with initial testing being conducted by the developers during the development stage. Upon completion of the project, additional testing will be conducted for integration of components and elusive bugs. Resources used for this testing will be the JUnit testing suite that allows for unit testing. Testing will be conducted the week before the project is due, approximately April 18 to April 22, in preparation for delivery of the final product.

The objective of this testing plan is to create a close to error-free application that runs on the lego robot EV3 brick. The application shall allow LEA to move about her environment freely and interact with users using speech recognition. Major work activities included development of the listening component and the interaction logic and processing. A major milestone in this project was completion of successful speech recognition and incorporation of the raspberry pi with the lego EV3 brick. This application was developed over a three-month period throughout two sprint cycles, with no budget.

**2.2 Testing Strategy**

Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item. The testing strategy for this application will be unit testing applied through the JUnit test suite. Each test will feature a unique application component and ensure proper behavior of LEA.

**2.3 Scope**

This plan will not be updated again upon completion of the project. Unscheduled updates may be performed in the case of unsuccessful testing techniques that must be modified prior to application deployment. Updates will be marked in subsequent version numbers, with whole steps indicating major updates and fractional steps indicating minor changes or changes in formatting.

Testing will be performed at several points in the life cycle as the product is constructed. Testing is a very 'dependent' activity. As a result, test planning is a continuing activity performed throughout the system development life cycle. Test plans must be developed for each level of product testing.

**2.4 Reference Material**

This project was authorized by Shafagh Jafer in January 2016 with a project deadline of April 26, 2016. This project features the following documentation: Software Requirements Specification, Use Cases, Javadoc Documentation, Test Plan Document, and Software Design Specification.

**2.5 Definitions and Acronyms**

SRS - Software Requirements Specification.

LEA - Lightweight Educational Assistant.

**3. Test Items (Jess)**

The program will be tested in parts and then integrated and tested as a whole before release.

**3.1 Program Modules**

The application was developed in two main modules: the interaction logic and the backend robot functionality such as movement, listening, and speaking. The interaction logic followed the strategy design pattern and was implemented in Java using the third-party library Lejos. The robot functionality was developed in Java and C using Eclipse and a Raspberry Pi. The Raspberry Pi allowed for more control over the speech recognition software and the ability for the robot to “listen” to users via a microphone.

The interaction logic and robot functionality must be tested simultaneously as the robot depends on all functionality to be working in order for it to function. Only the movement module can be tested separately from the rest of the system as it is the only feature to work independently of an interaction sequence. The interaction logic can be tested via forced keywords or by proper inputs from the listening module. After testing via the software, LEA must be tested in a “real world” application by having her move about and interact with a user. This will help ensure that LEA operates well enough to interact with users after being deployed.

**3.2 Job Control Procedures**

LEA operates using a pre-loaded jar file on the EV3 brick.

**3.3 User Procedures**

Users will need to start the jar file on the EV3 brick to start the program. Users then need to interact with LEA using the approved list of keywords in order to continue moving throughout the software logic.

**3.4 Operator Procedures**

The application was tested on a Lego Mindstorms EV3 brick that has the Lejos library installed.

**4. Features to be Tested (Jess)**

This project will include several testing methods, including JUnit tests and live application tests that verify the full functionality of the application. The features of the product that will be tested include the movement algorithm, speaking, listening, and the interaction tree. LEA must be able to speak clearly and listen to users well enough to continue a conversation. The ability to listen and recognize speech patterns and keywords is imperative to the success of this project.

**5. Features not to be Tested (Jess)**

For the purpose of the initial release of the product, the features remaining untested are higher-level safety checks in the movement logic (such as avoiding stairs or holes/drops) and the validity of the collected metadata. These features are excluded in the interest of time and to meet project deadlines.

**6. Approach (Lauren)**

**6.1 Component Testing**

Component testing will include unit testing and manual testing where the developer(s) of the component verify each aspect works as planned, according to the requirements in the SRS. The acceptance criteria should be predefined in the planning documents. Any unit tests should be created as soon as possible; therefore as aspect of LEA are completed they will be tested such as hearing, response to voice, and response to physical obstacles .

**6.2 Integration Testing**

Integration testing will include manual testing as well as adapting current unit tests to use the new project structure (to verify the integration did not break any previously working components).

**6.3 Conversion Testing**

Omitted - we have no old data elements that will need to be converted to new formats.

**6.4 Job Stream Testing**

Job Stream Testing will occur on LEA. This testing will ensure the sequence of response tables are reached correctly through the correct word recognition. This will also ensure speed and accuracy of recognition and response.

**6.5 Interface Testing**

User Interface testing will be covered by the Job Stream Testing.

**6.6 Security Testing**

Omitted - no security is present in this application.

**6.7 Recovery Testing**

Omitted - there is no persistent data that needs to be ensured between run times of the application.

**6.8 Performance Testing**

Performance testing will be done completely manually. It will verify things like responsiveness with the design requirement specifications (if present), or tester/developer discretion.

**6.9 Regression Testing**

Omitted - there is no regression testing needed.

**6.10 Acceptance Testing**

Acceptance testing will simply test and verify that all tests in the design documents have passed.

**6.11 Beta Testing**

Omitted - there is not a group of non-developers (i.e. non-biased testers) that we can give the product to.

**7. Pass/Fail Criteria (Lauren)**

In order to pass testing, each item must behave as defined in the SRS document. For items not specifically addressed in the SRS document, items must behave without error and produce the intended result as designed. Individual tests are included in section 6.4.

**7.1 Suspension Criteria**

Testing will be completely suspended when an individual component results in a fatal error. The component will then be debugged and re-implemented. Testing will be partially suspended when an individual component results in a minor error or deviation from the behavior defined in the SRS document. Errors with LEA will be tested immediately to resolve issues are they arise.

**7.2 Resumption Criteria**

Testing will be resumed upon re-implementation of the failed component. Test items that previously failed will be re-tested. If the component fails again, testing will be suspended and the process continues until all components meet the approval criteria defined in section 6.4.

**7.3 Approval Criteria**

Testing will be approved when 90% of the tests behave as defined in the SRS document. This allows for movement into further development and consideration for redesign. Testing of an individual component will be approved when the component executes without error and as expected. This will ensure that individual components are behaving correctly.

**7.4 Individual Test Cases/Use Cases**

*Lauren* – Testing the response of LEA to voice

Requirement: Section 4.1.2 – Interact with Users

LEA will cease movement upon hearing her name; therefore, I spoke LEA clearly and concisely and LEA stopped moving.

*Lauren* – Testing the response of LEA to voice

Requirement: Section 4.1.2 – Interact with Users

LEA will begin introduction upon hearing her name after stopping. I said “LEA” and the robot stopped moving and began her pre-programmed introduction.

*Lauren* – LEA will hand out Flyers

Requirement: Section 4.3 – Hand out Flyers

Once prompted to give out a flyer, LEA will move her arms to give a flyer to user based of requested information. After going through LEA’s tree of decisions, I asked LEA for a flyer on scholarships and she lifts her arm to give a scholarship flyer.

**8. Testing Process (Lauren)**

**8.1 Test Deliverables**

Each test conducted will be documented in this document and will include a short summary including the input, expected output, and actual output.

**8.2 Testing Tasks**

In order to perform testing the following tasks shall be completed first: Each team member shall set up at least one JUnit test case, the application shall be installed with proper xmls on an android device, and the application including any additional documents shall be on GitHub so all have access.

**8.3 Responsibilities**

The android development team shall focus on testing and refining the android part of the application either on an emulator or actual device. The image processing team will focus on the tests of the back end of the app.

**8.4 Resources**

The only tool to used for the testing of this application is JUnit and ECLEmma. JUnit will be used to test specific methods in the application. ECLEmma will be used to determine the coverage of the tests.

**8.5 Schedule**

All testing shall be complete by the Sunday 29 November, 2015 by 8:00pm.

**9. Environmental requirements (Jess)**

**9.1 Hardware**

The tests on the hardware of which the application would run on requires the following:

1. A Lego Mindstorms EV3 brick with Lejos installed.
2. A robot built following the EV3 Meg design (can be found on the Lego Mindstorms official website).
3. A Raspberry Pi with a microphone and wired connection to the EV3 brick.

The tests do not require a network connection.

**9.2 Software**

The software testing is broken into two sections, robot functionality (moving, listening, speaking) and interaction logic (conversation tree).

The robot functionality requires the following to test properly:

1. The movement must be tested with the robot.
2. The listening/speaking can be tested with the robot or with the EV3 brick only.

The interaction logic requires the following to test properly:

1. The robot functionality must be completed before the interaction logic can be fully tested.
2. The listening module must be able to recognize keywords and signal to the robot that a keyword was heard.

**9.3 Security**

There are no major security concerns when testing this software.

**9.4 Tools**

No special tools were used in the testing of this application.

**9.5 Publications**

No documents or publications were used in testing this application.

**9.6 Risks and Assumptions**

Times is only one significant constraint on the testing of this application. The completion of the listening and speaking modules is taking more time than originally anticipated. Testing may need to be reduced or minimized in order to meet the submission deadline.

**10. Approvals (Jess)**

|  |  |  |
| --- | --- | --- |
| Name | Signature | Date |
| Lauren Massey |  | March 29, 2016 |
| Jessica Updegrove |  | March 29, 2016 |