**Sprint 1 - Endurance Design Document**

**October XX, 2019**

Page Break

**Table of Contents**

**1.EXECUTIVE SUMMARY3**

1.1Project Overview3

1.2Purpose and Scope of this Specification3

**2.PRODUCT/SERVICE DESCRIPTION3**

2.1Product Context3

2.2User Characteristics3

2.3Assumptions3

2.4Constraints3

2.5Dependencies4

**3.REQUIREMENTS4**

3.1Functional Requirements5

3.2Security5

*3.2.1Protection5*

*3.2.2Authorization and Authentication6*

3.3Portability6

**4.REQUIREMENTS CONFIRMATION/STAKEHOLDER SIGN-OFF6**

**5.SYSTEM DESIGN6**

5.1Algorithm6

5.2System Flow6

5.3Software6

5.4Hardware6

5.5Test Plan7

5.6Task List/Gantt Chart7

5.7Staffing Plan7

1. Page Break
2. Executive Summary
3. ***Project Overview***

In this project, the robot must navigate its way through a rectangular course, stopping at each corner, continuing to the next corner, ending up back at the starting point.

1. ***Purpose and Scope of this Specification***

This sprint is to test our control over the robot we were given. It will display to the professor, our ability to code a successful and applicable program.

**In scope**

* Robot must travel in a 22’ X 11’8” rectangle
* Robot will switch colors along each side of the rectangle and speak the required phrase upon reaching a corner

**Out of Scope**

* Robot will follow along the tape with no over rolling and turn on tape each way exactly the entire shape.

Product/Service Description

In this section, describe the general factors that affect the product and its requirements. This section should contain background information, not state specific requirements (provide the reasons why certain specific requirements are later specified).

1. ***Product Context***

How does this product relate to other products? Is it independent and self-contained? Does it interface with a variety of related systems? Describe these relationships or use a diagram to show the major components of the larger system, interconnections, and external interfaces.

1. ***User Characteristics***

Create general customer profiles for each type of user who will be using the product. Profiles should include:

* Student/faculty/staff/other
* experience
* technical expertise
* other general characteristics that may influence the product

1. ***Assumptions***

List any assumptions that affect the requirements, for example, equipment availability, user expertise, etc. For example, a specific operating system is assumed to be available; if the operating system is not available, the Requirements Specification would then have to change accordingly.

1. ***Constraints***

Describe any items that will constrain the design options, including

* parallel operation with an old system
* audit functions (audit trail, log files, etc.)
* access, management and security
* criticality of the application
* system resource constraints (e.g., limits on disk space or other hardware limitations)
* other design constraints (e.g., design or other standards, such as programming language or framework)

1. ***Dependencies***

List dependencies that affect the requirements. Examples:

* This new product will require a daily download of data from X,
* Module X needs to be completed before this module can be built.

3.Requirements

* Describe all system requirements in enough detail for designers to design a system satisfying the requirements and testers to verify that the system satisfies requirements.
* Organize these requirements in a way that works best for your project. See **Error! Reference source not found.Error! Reference source not found.**, **Error! Reference source not found.** for different ways to organize these requirements.
* Describe every input into the system, every output from the system, and every function performed by the system in response to an input or in support of an output. (Specify what functions are to be performed on what data to produce what results at what location for whom.)
* Each requirement should be numbered (or uniquely identifiable) and prioritized.

See the sample requirements in Functional Requirements, and **Error! Reference source not found.**, as well as these example priority definitions:

**Priority Definitions**

The following definitions are intended as a guideline to prioritize requirements.

* Priority 1 – The requirement is a “must have” as outlined by policy/law
* Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits
* Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS **must be** either A or I" or "It **would be nice** if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

* A good requirement is:
* Correct
* Unambiguous (all statements have exactly one interpretation)
* Complete (where TBDs are absolutely necessary, document why the information is unknown, who is responsible for resolution, and the deadline)
* Consistent
* Ranked for importance and/or stability
* Verifiable (avoid soft descriptions like “works well”, “is user friendly”; use concrete terms and specify measurable quantities)
* Modifiable (evolve the Requirements Specification only via a formal change process, preserving a complete audit trail of changes)
* Does not specify any particular design
* Traceable (cross-reference with source documents and spawned documents).

1. ***Functional Requirements***

In the example below, the requirement numbering has a scheme - BR\_LR\_0## (BR for Business Requirement, LR for Labor Relations). For small projects simply BR-## would suffice. Keep in mind that if no prefix is used, the traceability matrix may be difficult to create (e.g., no differentiation between '02' as a business requirement vs. a test case)

The following table is an example format for requirements. Choose whatever format works best for your project.

For Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req#** | **Requirement** | **Comments** | **Priority** | **Date Rvwd** | **SME Reviewed / Approved** |
| ENDUR\_01 |  |  |  |  |  |
| ENDUR\_02 |  |  |  |  |  |
| ENDUR\_03 |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| ENDUR\_XX |  |  |  |  |  |

1. ***Security***
2. **Protection** : To protect from prying eyes and malicious intent these are the measures recommended:

* encryption
* 2 Step Verification
* Backups made of program
* Private collaboration repositories

1. **Authorization and Authentication**

* 2 Step authentication
* Malware detecting
* Repository is only open to verified collaborators.
* Pubcookie software is attached to send unauthorized users back to login

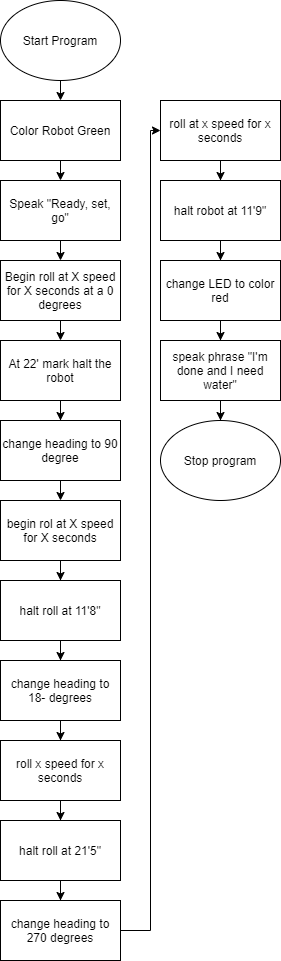
1. Requirements Confirmation/Stakeholder sign-off

Include documentation of the approval or confirmation of the requirements here. For example:

|  |  |  |
| --- | --- | --- |
| **Meeting Date** | **Attendees (name and role)** | **Comments** |
| MM/DD/YY | My group member names | confirmed all except ENDUR\_XX |
| MM/DD/YY | My group member names | confirmed…………. |

1. System Design
2. ***Algorithm*** 
   1. ***Robot will start up and turn LED green***
   2. ***It will speak the phrase ‘Ready, Set, Go’***
   3. ***Robot will roll at heading 0 degrees***
   4. ***Will travel 22’ and stop***
   5. ***Will turn to heading 90 degrees***
   6. ***Will roll 11’8” and stop***
   7. ***Will turn to heading 180 degrees***
   8. ***Will roll 21’ 5” and stop***
   9. ***Will turn to heading 270 degrees***
   10. ***Will roll 11’9” and stop***
   11. ***LED will turn red***
   12. ***Will speak the phrase ‘I’m done and I need water’***
   13. ***Program will terminate***

2. ***System Flow***



1. ***Software***

Program will be created using a mix of block code and javascript. Block code will be sourced from Sphero API and javascript will be used to convert collaboration back into block code.

1. ***Hardware***

This program will be developed on the Sphero Block Code generator and use Githhub repositories as the collaboration space. The robot itself is a sphero BOLT model.

1. ***Test Plan***

Include a test plan showing all unit tests performed for this application, Include test rational, test date, staff member, pass/fail status

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Reason for Test Case** | **Test Date** | **Expected Output** | **Observed Output** | **Staff Name** | **Pass/Fail** |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. ***Task List/Gantt Chart***

Embed your gantt chart here

1. ***Staffing Plan***

Insert a chart/table that depicts the roles and responsibilities of each team member that worked on this project

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Role | Responsibility | Reports To |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |