**Software Requirements Specification**

**for**

**PEST**

**Pest Expelling System Technology**

**Version 1 approved**

**Prepared by Adryel Arizaga**

**Victor Orozco**

**James**

**Daniel Gonzalez**

**Squirrel-Net**

**2 October 2017**

# **Table of Contents**

Table of Contents................................................................................................................... <pg #>

Revision History.......................................................................................................................<pg #>

1. Introduction............................................................................................................<pg #>

1.1. Purpose................................................................................................<pg #>

1.2. Intended Audience and Reading Suggestions......................................<pg #>

1.3. Product Scope............................................................................................<pg #>

1.4. Definitions, Acronyms, and Abbreviations .................................................<pg #>

1.5. References..................................................................................................<pg #>

2. Overall Description....................................................................................................... <pg #>

2.1. Product Perspective........................................................................................ <pg #>

2.2. Product Functions...........................................................................................<pg #>

2.3. User Classes and Characteristics....................................................................<pg #>

2.4. Operating Environment....................................................................................<pg #>

2.5. Design and Implementation Constraints..........................................................<pg #>

2.6. User Documentation......................................................................................<pg #>

2.7. Assumptions and Dependencies....................................................................<pg #>

2.8. Apportioning of Requirements...................................................................... <pg #>

3. External Interface Requirements................................................................................….<pg #>

3.1. User Interfaces............................................................................................... <pg #>

3.2. Hardware Interfaces.......................................................................................<pg #>

3.3. Software Interfaces........................................................................................ <pg #>

3.4. Communications Interfaces...........................................................................<pg #>

4. Requirements Specification.............................................................................................<pg #>

4.1. Functional Requirements...............................................................................<pg #>

4.2. External Interface Requirements...................................................................<pg #>

4.3. Logical Database Requirements....................................................................<pg #>

4.4. Design Constraints.........................................................................................<pg #>

5. Other Nonfunctional Requirements........................................................................... <pg #>

5.1. Performance Requirements........................................................................... <pg #>

5.2. Safety Requirements..................................................................................... <pg #>

5.3. Security Requirements....................................................................................<pg #>

5.4. Software Quality Attributes..............................................................................<pg #>

5.5. Business Rules...............................................................................................<pg #>

6. Other Requirements.........................................................................................................<pg #>

Appendix A: Glossary.........................................................................................................................<pg #>

Appendix B: Analysis Models............................................................................................... <pg #>

Appendix C: To Be Determined List......................................................................................<pg #>

# 

# 

# 

# **Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**<Add rows as necessary when the document is revised. This document should be consistently updated and maintained throughout your project. If ANY requirements are changed, added, removed, etc., immediately revise your document.>**

# **1. Introduction**

The Introduction section of the Software Requirements Specifications (SRS) document should give an overview of the entire document. As you write this document, remember that the purpose of this document is to tell software engineers WHAT the software is supposed to do. The details on HOW the software will be implemented should be specified in the Software Design Document (SDD).

## 

## **1.1 Purpose**

Squirrels and shit

The purpose of this document is to define the specifications and requirements of the PEST project. This document will outline what PEST is designed to do alongside the limitations of the aforementioned project.

The purpose of the PEST project itself is to create an automated system that will repel pests (including, but not limited to: squirrels, wombats, and polar bears) from a predetermined area, such as a garden or patio area.

*Identify the purpose of this document and the product whose software requirements are specified in this document. Be sure to mention the version / revision / release number of the software and also mention if the document covers all aspects of the software or only a small part of the software. Do not iterate over the table of contents here. Give a general overview of what this document contains.*

## **1.2 Intended Audience and Reading Suggestions**

Squirrel haters

This documentation is intended as a technical resource for but not limited to: future maintenance developers, project managers, marketing staff, testers, and documentation writers. This SRS is organized into five (6) sections: Introduction, Overall Description of PEST and its abilities, descriptions of the various Interfaces (User, Software, Hardware, Communicationes), Functionality Requirements, Non-Functional Requirements, and Requirements that do not classify into the other sections.

*This section should mention what types of readers the document is intended for such as developers, project managers, marketing staff, users, testers, and documentation writers.*

*Describe what this SRS contains and how it is organized. If each type of reader should interpret this document differently, give a suggested reading sequence for the document that is relevant to that particular reader type. Example: A tester might only be interested in the detailed requirements list, or the marketing staff may only be interested in the very high level descriptions of the project.*

## 

## **1.3 Product Scope**

The software to be produced will be named “PEST Control”. PEST Control will serve as a preferences system to initialize the Squirrel-Box and to alter the default settings of the Squirrel-Box. The software will open when plugged into a computer’s USB port and will display a GUI for the user to easily alter the various settings associated with the Squirrel-Box. This includes: establishing “night hours”, specifying the specific sound effects/vocalizations that will be used when the Squirrel-Box is triggered, the behaviour of the “snooze” button, and a user-manual for operations of the Squirrel-Box.

Upon release of PEST Control, users will be able to individually alter and personalize -- to a degree -- the functions of their personal Squirrel-Box. PEST Control is specifically limited to altering the abilities of only their own, individual Squirrel-Box by setting a password, should they so desire.

*In this section:*

*● Identify the software product(s) to be produced by name.*

*● Explain what the software will do. If necessary mention what the software will not do.*

*● Describe how the software will be used once released, include benefits of the software, objectives, goals, etc.*

*This section should give a high-level summary of the software. Do not list all of the requirements here.*

## **1.4 Definitions, Acronyms, and Abbreviations**

SH - squirrel hater

PIR - Passive InfraRed

GUI - Graphic User Interface

LED - Light Emitting Diode

PEST - Pest Expelling System Technology

Squirrel-Box - The Combination of the Raspberry Pi and the container that houses it

PEST Control - The Software used to alter preferences associated with the Squirrel-Box

Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SRS document. This information may be listed here, in an appendix at the end of the document, or in a completely separate document. If the information is not directly listed in this section provide a note that specifies where the information can be found.

## **1.5 References**

*This section should provide the following:*

*● A complete list of all documents referenced in the SRS.*

*● A complete list of any Web addresses referenced in the SRS.*

*● For each reference mention the title, author, version number, date, and the source or location of the reference.*

*● Generally you should provide enough information for each reference so the reader of the SRS can easily obtain copies of these references.*

*References may include items such as: user interface style guides, contracts, standards, other SRS documents, use case documents, supplemental documents that were researched while writing this SRS or while designing the software.*

# 

# **2. Overall Description**

This section explains the general facts that affect the software and its requirements. You should not state any specific requirements in this section. Here you will provide a background for the actual list of requirements that will be defined in section 4. The point in this section, is to give a high level description of the requirements in plain English for the benefit of the customer / client.

Th

## **2.1 Product Perspective**

This section should show how the software may relate to other products. If the software is completely independent of any other products / systems, you should also mention that here.

This software can be implemented into a security system.

If the SRS describes software that is a component of a larger system, this section should describe how the requirements of this software relate to the larger system. Be sure to mention and identify interfaces between the larger system and the software that this SRS defines.

This section should also mention how the software in this SRS compares to similar systems currently on the market (if any exist). Mention similarities / differences / motivations for creating this software when compared to an existing product. If the software is a completely new innovation, be sure to mention that as well.

You may provide a block diagram here showing the major components of the larger system, external interfaces, and how your software fits in with the larger system. This is NOT a design or architecture picture. It is simply here to provide a context for the reader to understand where your software fits with the larger system. Your software should be shown as a black box. Remember, it is the purpose of the Software Design Document to present the internal workings of your product.

## **2.2 Product Functions**

This section provides a summary of the major functions that the software will perform. The functions should be organized in a way that makes them understandable to the customer / client or anyone who is reading this document.

You can provide a bullet list summary of each function and / or provide simple diagrams to show the different functions and their relationships. Again, these diagrams do not show the design of the product, just the relationships between the functions.

PEST is intended to function in two (2) states: the first when an entity is present, and the second when on standby. When on standby, the software is designed in such a way that the finished product will run “silently” and be in constant search for an entity to trigger the infrared sensor. The second state is when the infrared sensor is triggered, in which case the various “alarm” systems will be triggered, all of which function using a randomizing algorithm to add various and protect against desensitization to the alarm system.

The first of the randomized systems is a series of LED lights that will flash in various patterns and intensity. This function is primarily set to be used during the user-defined “night hours” where the light will be able to be used to full effect to ward off various nocturnal pests.

The second is a built-in speaker system that will play a random sound as another line of defense against timid pests. The speaker system will be disabled during the user-defined “night hours”, as to not cause a public or private disturbance during inconvenient times.

Again, describe what the functions of the system need to do, not how you are going to implement them.

## **2.3 User Classes and Characteristics**

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important.

## **2.4 Operating Environment**

Provide details on the environment in which the software will live. This should include the hardware platform, operating system, and any other software components or applications that will affect your software and which have to coexist with your software.

The platform that is intended for which the software will be is a Raspberry Pi 3, in which the operating system that will be running on it is Raspbian.

## **2.5 Design and Implementation Constraints**

There are no regulatory policies at the moment of writing that would potentially impede the ability to complete development of the software.

The Raspberry Pi in question is limited to 16gb (gigabytes) of information, which could potentially slow or threaten development of any form of memory-related issue is encountered.

Development on will be limited to the capabilities of the Raspian Operating system.

The entirety of the project will be created using the Python Language

*Give a general description of any items that will influence the ability of the software developers to implement the product. These can include things such as:*

*● Regulatory policies within the organization that may affect development the software.*

*● Hardware limitations*

*● Interfaces with other applications*

*● Parallel operation*

*● Higher-order language requirements*

*● Reliability requirements*

*● Safety and security considerations.*

*● Memory Constraints*

Be sure to list any constraints that will affect the software. Generally this is to give an overview of any non-functional requirements that will be detailed in a later section.

## 

## **2.6 User Documentation**

List any user documentation (user manuals, online guides, tutorials, etc) that will be delivered along with the software. Specify in what format each document will given.

User manual

## **2.7 Assumptions and Dependencies**

List any other factors that may affect the requirements stated in this document. These are not design constraints (as mentioned above) but are factors that if changed will affect the requirements.

For example, you might assume that a specific operating system is available on the hardware allocated for this product. If this operating system were not available, then the SRS would have to be changed accordingly.

List any third-party or commercial components that you plan to use. List any software component dependencies that you might want to reuse from another project.

This section is a catch-all for anything that will affect your software design that did not fit into any of the previous sections. If anything will affect your requirements, mention it here.

## **2.8 Apportioning of Requirements**

List any requirements that might be delayed until future versions of the system.

# 

# 

# **3. External Interface Requirements**

Again, this section is a high level description for the non-technical people who may be reading this document. Everything in this section is to detail how your software interacts with any external interfaces, whether these are other software or even hardware interfaces.

## **3.1 User Interfaces**

GUI will include option to alter the light and sound. A button that will act as a “Snooze” feature, in the event where the user needs to do some gardening work.

Error Messages?

Hints?

Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

Be sure to mention any requirements that must conform to the Americans with Disabilities Act.

## **3.2 Hardware Interfaces**

The way the software and hardware will interact with this product is mainly through the sensor. If heat is detected through the sensor then the program will activate the light/sound. The only supported device for this product is the Raspberry Pi.

Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

This is not a description of hardware requirements such as "This program will run on a PC with 16 gigabytes of RAM." This section details the actual hardware devices that your application will interact with. Items such as robotics components, lighting, audio / video equipment, etc. If your software does not have an hardware interface requirements, then state this instead.

## **3.3 Software Interfaces**

List any other software products and interfaces that your requirements must utilize. For each required product, list the name, version number, and source.

Document any APIs that your software will have to access in order to interact with other software products. Also document how your software will communicate / pass information to the external software.

Example, your customer uses MySQL version 1 and you are required to use that in your design, then you must specify this here.

This section is NOT for specifying software that you think would be good to use. This is for customer-specific requirements that you HAVE to interact with.

## **3.4 Communications Interfaces**

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

## 

# 

# **4. Requirements Specification**

This section contains all the necessary software requirements with enough detail to allow designers to accurately design the software to satisfy those requirements, and to allow testers of the software to verify that all requirements have been satisfied. The requirements should include a description of every input to the system, every output, and all functions performed by the system in response to an input or output.

The biggest thing to remember is that this section is for the software developers (technical people) while the previous sections were for the customers / non-technical people.

Also remember that this is not HOW things will be implemented, but WHAT will be implemented.

Requirements should be written according to the following:

1. Specific requirements should be correct, unambiguous, complete, consistent, ranked for importance and / or stability, verifiable, modifiable, and traceable.

2. Specific requirements should be cross-referenced to earlier documents that are relevant.

3. All requirements should be uniquely identifiable using a consistent numbering system, i.e. 1.1, 1.2, 1.1.2, and so on.

4. Requirements should be organized in a logical manner to provide the most readability.

Use the following format for each requirement:

The system shall... (this means this requirement is mandatory).

The system should... (this means a desired feature, but may be delayed until later).

This system may... (A optional, nice-to-have feature that might not be implemented).

Remember to number each requirement for traceability. Use a system such as 1.1, 1.1.1, 1.1.2.1, and so on. Each requirement need to be testable. Avoid statements that are general and vague such as "The system shall be easy to use." or "The system shall be developed using good software engineering practices."

Do not include examples. Remember that this is a specification and the designer should be able to read this and build the system without having to bother the customer again. Every minute detail must be documented here.

EVERYTHING in section 4 must be written following the above guidelines.

1.1 The system shall monitor state changes in the PIR sensor.

1.2 The system shall output a red LED in the absence of any PIR sensor state changes.

1.3 The system shall output a green LED in the event of a PIR sensor state change.

1.4 The system shall output audio in the event of a PIR sensor state change.

1.5 The system shall provide a GUI to set user settings, show console output, etc.

2.1 The system should activate the camera in the event of a PIR sensor state change.

2.2 The system should use openCV image recognition to confirm target is a squirrel in the

event of a PIR sensor state change.

2.3 The system should take snapshots and store them on local storage in the event of target

confirmation.

2.3 The system should execute squirrel deterrence protocol upon target confirmation.

3.1 The system may upload snapshots from the camera to a remote server, in the event of

target confirmation.

3.2 The system may provide an interactive web interface, including but not limited to unique

user accounts, live camera view, and viewing of archived camera snapshots.

3.3 The system may provide a mobile app with the same features as the web interface.

## **4.1 Functional Requirements**

Functional requirements define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as “shall” statements starting with "The system shall…”

These include:

● Validity checks on the inputs

● Exact sequence of operations

● Responses to abnormal situation, including

○ Overflow

○ Communication facilities

○ Error handling and recovery

● Effect of parameters

● Relationship of outputs to inputs, including

○ Input/Output sequences

○ Formulas for input to output conversion

It may be appropriate to partition the functional requirements into sub-functions or sub-processes. This does not imply that the software design will also be partitioned that way.

This section should be as detailed as possible, again, listing WHAT your software is going to do, not HOW you are going to accomplish it.

## **4.2 External Interface Requirements**

This contains a detailed description of all inputs into and outputs from the software system. It complements the interface descriptions in section 3 but does not repeat information there. Remember section 3 presents information oriented to the customer/user while section 4 is oriented to the developer.

It contains both content and format as follows:

● Name of item

● Description of purpose

● Source of input or destination of output

● Valid range, accuracy and/or tolerance

● Units of measure

● Timing

● Relationships to other inputs/outputs

● Screen formats/organization

● Window formats/organization

● Data formats

● Command formats

● End messages

Name: PIR Sensor

Purpose: monitor and report any motion detected within range

Output Destination: GPIO / Raspberry Pi

Range: 7 meters

Relationships: Input from this device will determine the output of the audio and 2 LED lights

Name: LED light

Purpose: to output light

Input Source: GPIO / Raspberry Pi

Relationships: Output of LED is dependent on input from PIR sensor

Name: Audio speakers

Purpose: to output audible sound

## **4.3 Logical Database Requirements**

This section specifies the logical requirements for any information that is to be placed into a database.

This may include:

● Types of information used by various functions

● Frequency of use

● Accessing capabilities

● Data entities and their relationships

● Integrity constraints

● Data retention requirements

If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show complex data relationships. Remember a diagram is worth a thousand words of confusing text.

## **4.4 Design Constraints**

Specify design constraints that can be imposed by other standards, hardware limitations, etc. This should be a more technical description of the overview given in section 2.5.

# **5. Other Nonfunctional Requirements**

## **5.1 Performance Requirements**

This section specifies any numerical / statistical requirements imposed on the software such as:

● The number of terminals to be supported

● The number of simultaneous users to be supported

● Amount and type of information to be handled

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms. For example, "95% of the transactions shall be processed in less than 1 second" rather than, "An operator shall not have to wait for the transaction to complete."

(Note: Numerical limits applied to one specific functional requirement are normally specified as part of that requirement and should be listed in section 4. This part is more for performance / statistical requirements imposed on the entire system as a whole.)

## **5.2 Safety Requirements**

Possible light bulb popping. Possible high frequency noises. Seizures.

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.

## **5.3 Security Requirements**

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

## **5.4 Software Quality Attributes**

Flexibility/Adaptability/Portability: Easily movable, only limited by the length of the power cable and the sensor’s area of detection.

Availability: Raspberry Pi’s, resistors, sensor and led’s are fairly easy to come across online.

Correctness:

:

Interoperability:

:

Reliability:

Maintainability/Reusability: As long as the raspberry Pi works then the device can be reused and maintained.

Robustness: With the raspberry Pi encased, the device will be heavily protected.

Testability: Easily debuggable with the simplicity of the GUI.

Usability:

Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

## **5.5 Business Rules**

If any errors occur, the squirrel box can be easily connected to a laptop forfixing

List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.

# 

# 

# **6. Other Requirements**

Define any other requirements not covered elsewhere in the SRS. This might include internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project in this section.

This is pretty much a catch-all for things that do not fit in a previous section.

# **Appendix A: Glossary**

Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.

If this section is very short you may include it in section 1.4. If your list is very long you may include it here and put a reference to this Appendix in section 1.4.

# **Appendix B: Analysis Models**

Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.

# **Appendix C: To Be Determined List**

Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.

**General Template Directions:**

● This page should not appear in the final document.

● Replace any **<Insert x here...>** placeholders with the actual text. Yes this also means replacing the angle <> brackets as well. Your document should not have ANY **blue** text in it. If the text had any kind of formatting (bold, italics, font size) keep the same formatting. DO change the font color to black.

● Remove any **<green text>** placeholders. These are descriptions about what to write in each section and may also provide examples of what to write. Do not simply copy / adapt the wording to fit your project. All green text should be replaced with your own text or removed completely in some cases.

● **DO NOT** change the formatting of the document this includes:

• font sizes

• font weights

• italics NOT in angle brackets

• margins

• spacing

• adding or removing sections of the document.

● Each major numbered section (1, 2, 3 etc.) should start on a new page.

● When the document is finished, update the table of contents with the correct page numbers, and make sure the numbers are properly aligned to the right.