Software Requirements Specification

for

Input Multiplexer (iMux)

Version 1.0 approved

Prepared by Christopher Hartz

National University: SEN601

25 July 2017

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

4. System Features 4

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

5. Other Nonfunctional Requirements 4

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

6. Other Requirements 5

Appendix A: Glossary 5

Appendix B: Analysis Models 5

Appendix C: To Be Determined List 6

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Hartz | 25 July 2017 | Initial Document | 1.0 |
|  |  |  |  |

# Introduction

## Purpose

This Document defines the software requirement for Input Multiplexer (iMux), A Google AI Cloud Services Interface Application

The Input Multiplexer application will be developed as per the requirements specified in this document

This document will list all the features and functions that will be implemented in the Input Multiplexer Application.

## Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

This

## Intended Audience and Reading Suggestions

This Document is intended to be a living document that serves a communication method for all stakeholders. In between each spiral of development identified stakeholders included project designers, programmers, deployment clients, and users will use this document as a reference.

The SRS contains a detailed specification for the first prototype of the Input Multiplexer

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

## Product Scope

Input Multiplexer (iMux) aims to integrate a multitude of different human users who are using a different input methods (i.e. voice, graphical, or text) and exercising different roles (i.e. supervisor or technician) to work on a single application seamlessly. The software will use Google’s AI services to integrate AI agents as additional actors in the workflow to facilitate complex interactions among humans and to automate tasks. iMux will enable a highly responsive and easy to use interface for applications which have a hard time requirements and rely heavily on human interaction, such as fraud detection, disaster emergency response, or real time cyber security breach detection and mitigation.

This Document will address the first demonstration prototype which includes a subset of features. Version 1.0 will:

1. Enable three people to work with one application simultaneously. One user will give voice commands, one will issue text commands and one will issue GUI based commands.
2. The project will implement the software in python and use Google’s Speech Recognition Service Google's Voice to Text AI service, and enters the received text on the command line.
3. A game will be used to simulate an environment with hard time requirements and demonstrate the use of multiple inputs.

Later versions of this document will address additional features such as classification of input data using Google’s TensorFlow and inclusion of independent AI actors along with human actors.

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

## References

1. Google Python Style Guide: <https://google.github.io/styleguide/pyguide.html>
2. Pygame; Python GUI: <https://www.pygame.org/docs/>
3. Tensor Flow: <https://www.tensorflow.org/>
4. Google Cloud Tutorials: <https://www.youtube.com/playlist?list=PLQVvvaa0QuDfGVb3yucqvKtUgwOJgZWCm>
5. Python Speech Recognition: <https://pypi.python.org/pypi/SpeechRecognition/>
6. Game code contributions: [http://pygame.org/project-python+pygame+tower+defence-1296-4410.html#](http://pygame.org/project-python+pygame+tower+defence-1296-4410.html)

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

# Overall Description

## Product Perspective

iMux is a new class of application. It leverage the recent trend in large cloud service providers. Microsoft, Amazon and Google have begun to move off of the mobile divices as the future of revenue growth on to AI services, believing they can revolutionize a broad range of business across the entire economy. For example Machine Vision is being usedi n agricutual drones to identify pests allowing the use of pesticides on only those areas of the crop that are infected (MIT Technology Review: <https://www.technologyreview.com/s/526491/agricultural-drones/> ) or in a recycling center where a camera and a robotic arm sort trash (Forbes: <https://www.forbes.com/sites/jenniferhicks/2017/04/04/this-recycling-robot-uses-artificial-intelligence-to-sort-your-recyclables/#4a8221a42d35> )

Although Machine Learning techniques are deployed widly in industry, the use of AI as a services promises to radically reduce costs which will enbable the use of the technology in many more sectors. This trend is similar to Amazon Web Services. Amazon opned up the use of its cloud to any user after it had mastered the functionality it had developed for its own shopping service. This move revolutionized computing and the offering of AI services is the next step in this trend.

iMux will automate and facilite the interaction of AI agents and humans to increase response times to time critical events. Currently

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

## Product Functions

iMux will be build using a MacBook running python 2.7 and Google’s Cloud API.

There will be three front end interfaces. One interface per input type, Text, Voice, and GUI, corresponding to three separate actors (Engineer, Supervisor, and Technician)

Data for version 1.0 will be stored on the MacBook

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

## User Classes and Characteristics

There are three user classes based on technical expertise and job role:

1. Supervisor
2. Engineer
3. Technician

Users will interact with the program via Voice, Text, and GUI per their user class.

<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 to satisfy.>

## Operating Environment

iMux can run on any Platform (Mac, Windows, or Linux) running Python 2.7 with the Google Cloud API and Pygame libraries installed.

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

## Design and Implementation Constraints

iMux is designed to facilitate communication between a variety of roles in a control room like setting. It must coexist in an evronment with a number of other communication technologies such as chat and tex and provided added benefit. To do this it must be responsive and the application will be designed with a low latency between commands and execution.

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>

## User Documentation

User instructions will be generated for each role.

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

The following assumptions will be made while developing iMux

1. User will have a low latency connection to Google’s Cloud Services
2. User will have the requisite python libraries installed
3. User’s environment will be low enough noise in order to enable speech recognition
4. The client computer will not have other applications that interfere with iMux’s use of the microphone

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# External Interface Requirements

## User Interfaces

\*\*\*

<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.>

## Hardware Interfaces

<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.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## 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.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<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.>

## 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.>

## Software Quality Attributes

<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.>

## Business Rules

None.

<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.>

# Other Requirements

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

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.>

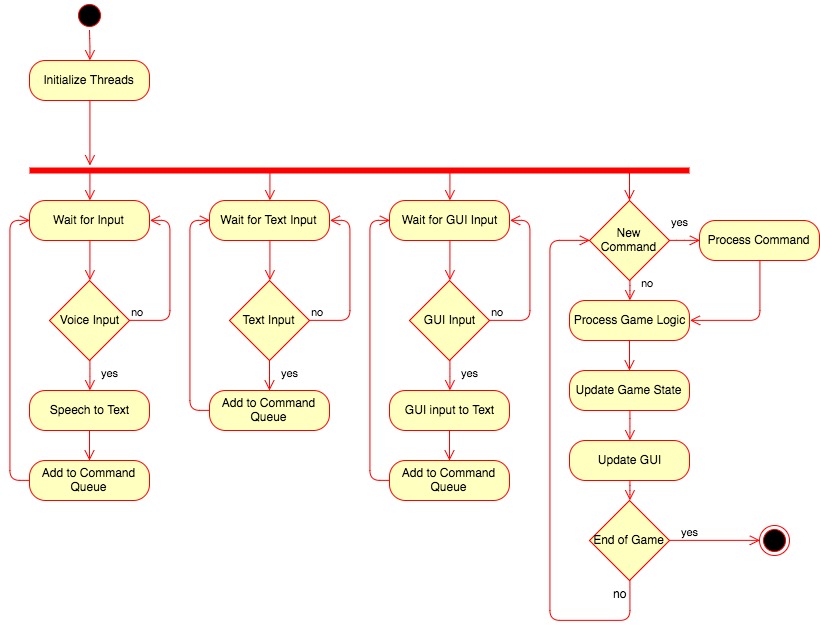
Appendix B: Analysis Models

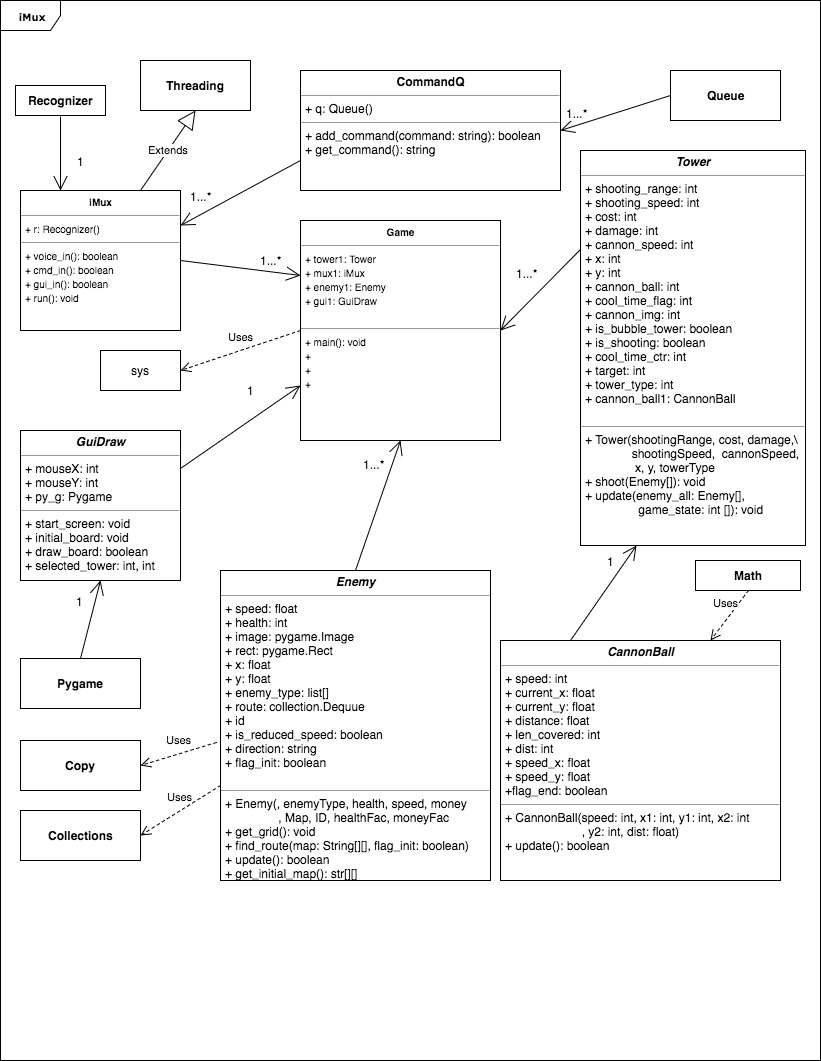
<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

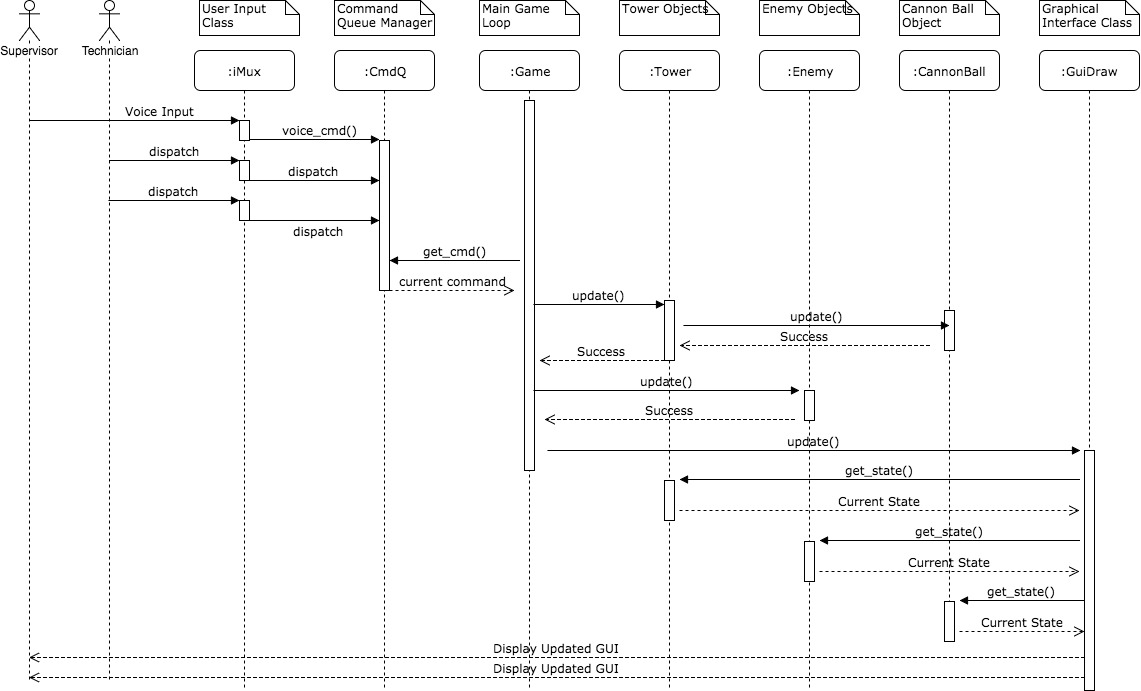
Appendix C: To Be Determined List

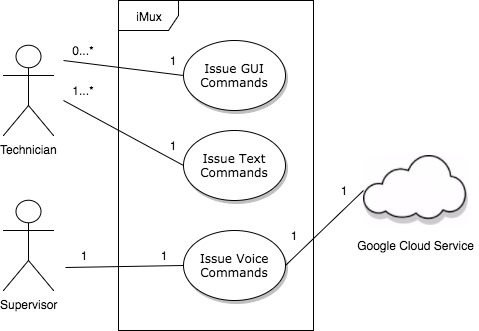
<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>

Activity Diagram



Class Diagram

Sequence Diagram

Use Case Diagram