1 Introduction

1.1 Purpose

This document describes the `PresDroid` product version 0.1. `PresDroid` aims to facilitate the creation of presentations that can be easily controlled through a simple Android UI. This SRS covers the entirety of the `PresDroid` project.

1.2 Document Conventions

Todo or takeout

1.3 Intended Audience

1. Potential User (People wanting to give a presentation)
2. Dr. Pruski
3. Software Development Team
4. Software Testing and Validation Team
5. System Architect

This SRS contains the Product Scope, Overall Description, External Interface Requirements, System Features, Non-Functional Requirements, and Other Requirements of the system. The end of the SRS will cover the terms used (Glossary), the models of various aspects of the system (Analysis Models), and aspects of the system to still be determined (To-Be Determined List).

1.4 Product Scope

The system will have 2 main components that interact with the user. The 1st component is a desktop application that will enable the user to create and display presentations. The 2nd component is an Android application that will allow the presenter of the presentation (created via the 1st Component) to control the presentation. The presenter will have the following capabilities through the android application:

- Scroll/flip through presentation slides
- Zoom in/out of the presentation
- Highlight text on the presentation
- Draw figures on the slides
- Point to specific content (like a laser pointer) through a simple Android UI and gestures

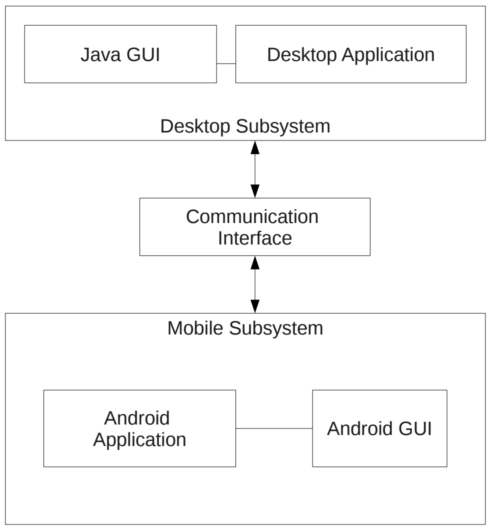
1.5 References

TODO but I don't think we have anything to refers to !!!

2 Overall Description

2.1 Product Perspective

The `PresDroid` is a new stand-alone system that does not depend on any other user systems. The desktop application portion will target the Windows and Linux operating system environments. The mobile application will be targeted for Android mobile phones, specifically Luis' and Angel's phone.



2.2 Product Functions

The desktop subsystem will perform the following functions:

- Build a new presentation
- Present previously built presentation

- Sync with mobile subsystem
- Interract with mobile subsystem

The mobile subsystem will provide the following functions:

- Sync with desktop subsystem
- Accept gesture input
- Interract with desktop subsystem

2.3 User Classes and Characteristics

The user groups will include students or professors giving simple, quick, and effective presentations.

2.4 Operating Environment

The hardware requirements for `PresDroid` are a Windows or Linux computer where a projector is compatible and Android mobile phone. The computer and phone used must have networking (HTTP or Bluetooth) capabilities.

2.5 Design and Implementation Constraints

TODO: remove or add

2.6 User Documentation

User manual.
Help section in desktop and mobile application.

2.7 Assumptions and Dependencies

We assume this will work with Windows and Linux computers.
We assume Bluetooth messaging will not be data-constrained.
We assume the computers will have bluetooth messaging capabilities.
We assume there will be FOUR members in our group, not 3 or 3.001, jusk kidding, ... not really, Chris.

3 External Interface Requirements

3.1 User Interfaces

UI Characteristics for both Desktop and Mobile Applications:

- The applications will use the GUI Frameworks provided by the system they are built upon. For example, the Mobile application will make use of the native Android GUI Framework (Widgets, Spinners, etc.) while the Desktop Application will make use of the Java Swing Framework (JFrames, JButtons, etc.).
- The applications must provide a button within the UI which will present the user with a Help dialog box or screen.
- The applications will present errors in two ways:
  - 1) If the error is critical (such as the connection has been lost between the desktop application and mobile application), then a Message/Dialog Box will appear that the user must acknowledge before proceeding.
  - 2) If the error is NOT critical (such as invalid input), then text should appear towards the bottom of the UI in red text to notify the user of a non-critical error occurring.
- The layout of UI elements should adapt to suit to the system that application is installed on. For example, the Android Application UI should be able to adapt to different types of mobile devices (Phone and Tablet) so that elements in the UI dont appear in unexpected places.

3.2 Hardware Interfaces

The interfaces between both applications and the underlying hardware will be provided through the frameworks used in development such as the Android SDK/Framework and the Java Framework. As mentioned earlier, the supported devices will include Android devices for the Mobile Application and Windows and Linux machines for the desktop application.

3.3 Software Interfaces

Both the Mobile and Desktop subsystems will be built upon Java and Android frameworks, respectively. These frameworks will provide abstract interfaces to operating system features such as file management and multi-threading. Both subsystems (mobile and desktop) will need to create and process a variety of external messages, including, but not limited to:

- Pair up with mobile or desktop application request
- Zoom presentation in/out message
- Move pointer on presentation screen to a specific point message
- Kill connection between desktop and mobile applications message

3.4 Communications Interfaces

The two major subsystems (as depicted in the Product Perspective section above) will communicate using a Communication Interface. This Interface will store all messages or requests to be processed and will handle communication between the two subsystems. It will provide a simple and consistent interface where a component can easily check if it has any requests or messages to process. An example of a request is an incoming request for the desktop application (from the Mobile application) to pair up with the desktop application so that the two applications can be in sync and can communicate. This Communication Interface will need to rely on some form of networking (HTTP, Bluetooth, etc.) to be provided by the host machine. The Communication Interface will need to run on a separate thread so that the program will not stall while waiting for incoming messages. The Communication Interface must make sure to protect against corruption of its message pool through the use of Semaphores and/or Mutexes, etc.

4 System Features

4.1 System Feature 1

4.1.1 Description and Priority

4.1.2 Stimulus/Response Sequences

4.1.3 Functional Requirements

- REQ-1:
- REQ-2:

4.2 System Feature 2 (and so on)

5 Other Nonfunctional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

5.5 Business Rules

6 Other Requirements

7 Appendix A: Glossary

8 Appendix B: Analysis Models

9 Appendix C: To Be Determined List