The Long Distance Deployer

System Specification

Written By

Duncan Hook

Table of Contents

Executive Summary

I am seeking to create a system to allow game companies to access and test their builds remotely. This system known as the Long Distance Deployer will allow developers to upload builds to the Long Distance Deployer server, and later access them from home using a cloud-based virtual machine that will stream directly to their home. This document contains the details relating to the development of this system, a conceptualization of the necessary classes, hardware, software, and UI requirements. This project's risk, feasibility, and functionality assessment are discussed more in the system proposal. This document is meant to serve as a guide to the more technical aspects of development.

1.0 Introduction

1.1 Problem Statement / Project Vision

The current quarantine caused by the outbreak of Covid-19 has left many companies, specifically game companies, looking for a way to efficiently continue working on their projects. The development of the Long Distance Deployer (LDD) seeks to fill this need. The LDD will allow game developers to access their builds when they are away from the office; it will help alleviate (if not outright remove) download times and allow many people to use their home equipment even if it normally wouldn't be powerful enough to test with.

The primary stakeholders of the LDD are the developers themselves. The LDD will greatly increase developers' quality of life (both during and after the Covid-19 quarantine) and improve development times due to being able to work on their projects even when the developers cannot come into the office. Other stakeholders include the customers, marketing and advertising teams, public relations teams, and digital and physical storefronts. These all are not directly influenced by the development of the LDD, but are indirectly influenced by the LDD's effects. All these stakeholders are affected by the fact that the LDD will help developers release their products on time with higher quality than they otherwise could without the LDD.

Refer to section 1 of the system proposal for more information on the project's vision, scope, and stakeholders.

1.2 System Services

The use-case scenarios referenced below can be found in section 5.3 of the system proposal, which begins on page 24.

- **Allow developers to upload builds onto the LDD server:** Employees in the office can access the LDD and upload build along with naming it so that employees away from the office can access it remotely. (Use-case 4).
- **Take in and validate employee's credentials to be able to access the LDD:** Before allowing full access to the LDD and its contents, the LDD will require the employee to input their credentials, which consist of a username and password. (Use-case 2).
- **Take in and validate the security key to access the LDD:** After the employee', the LDD will then require the employee to input the current security key before providing access. (Use-case 3).
- **Create a cloud-based virtual machine on the server for the employee:** After allowing access, the LDD will create a cloud-based virtual machine on its server for that employee specifically. (Use-case 7).
- **Allow employees to access the LDD from home or other remote locations with access to computers and the internet:** The LDD must allow Employees who are away from the office to access the system. (Use-case 6).

- **Distinguish if the employee's internet can support the full breadth of the build:** The LDD needs to monitor its connection to the remote employee and determine if the ping is suitable and if any packets are being dropped to determine if the employee's internet quality is suitable for best performance at the current quality. (Use-case 8)
- **Dynamically adjust the quality to improve performance:** The LDD needs to either upgrade or downgrade the quality of its streamed content to improve the performance of the project/build the employee is testing. (Use-case 9)

1.3 Nonfunctional Requirements and Design Constraints

- The servers running the LDD will require constant internet connection.
- The LDD must be easy to work with, it must both accept new builds easily and access builds for remote employees easily.
- The LDD needs to be accessible from all types of computers.
- The LDD needs to be able to be accessed from multiple computers simultaneously without disrupting service.
- The LDD should take no longer than a minute or two to begin projecting the build requested by a remote employee.
- The LDD must store the projects, builds, and any other information stored on its servers securely.
- Employee credentials and source code should be encrypted.
- Must have an additional security key that is changeable on top of the required employee credentials.
- The LDD needs to be able to run for hours on end without shutting down or overheating.
- The LDD needs to able to interpret the connectivity to and from the employee accessing the system.

Refer to section 4.4 (page 19) and section 1.6 (page 6) of the system proposal for more information about the nonfunctional requirements, and constraints respectively.

1.4 System Evolution

The LDD will initially launch with all the requirements listed above. After the LDD has been launched and integrated with the company's work pipeline, several improvements could be implemented down the line. First, the ability of developers who are out of the office to upload builds from their remote location would help further improve the convenience the LDD provides. Second, allowing multiple remote developers to engage in a single session together (if the project requires such an improvement) would also serve to improve the quality of testing remote developers could perform. Finally, upgrading the servers the LDD is stored on when needed will ensure that the LDD can maintain high-quality functionality.

Refer to section 6.0 (page 37) of the system proposal to read more in-depth analysis on possible, post-launch, improvements that could be made to the LDD.

1.5 Document Outline

The rest of this document will follow this outline:

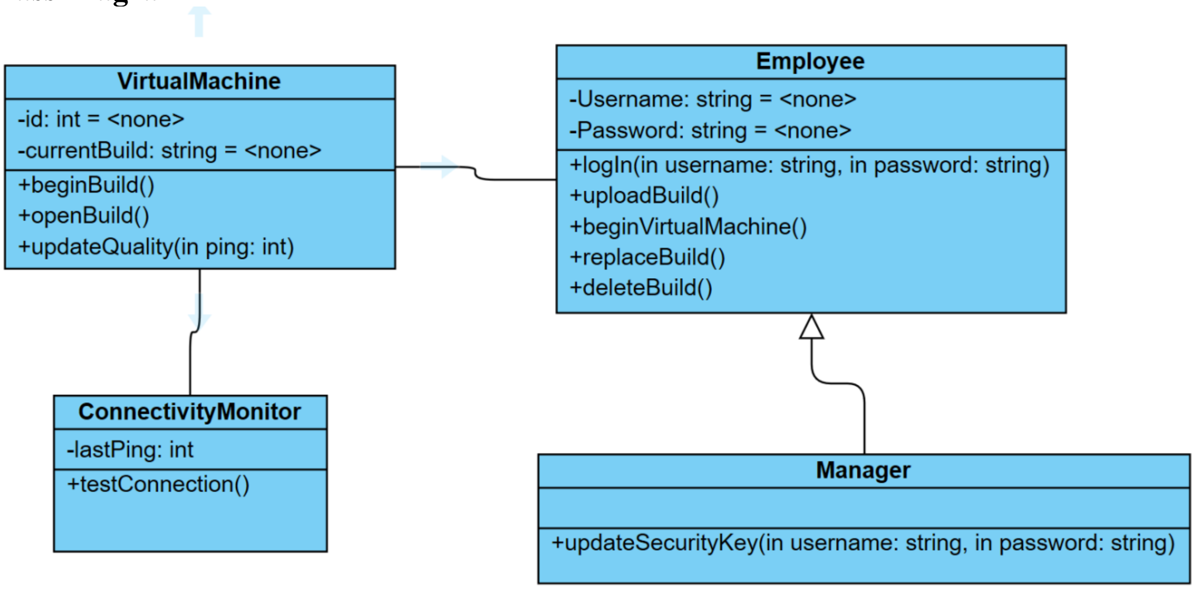
- **Structural Model:** Shows a class model and more in-depth individual class descriptions and details.
- **Architecture Design:** Shows the infrastructure model for the required hardware and software of the LDD.
- **User-Interface:** Demonstrates the requirements for the user-interfaces the LDD requires.
- **Appendices:** Contains the bibliography of documents used as references in the creation of this document.

2.0 Structural Model

2.1 Introduction

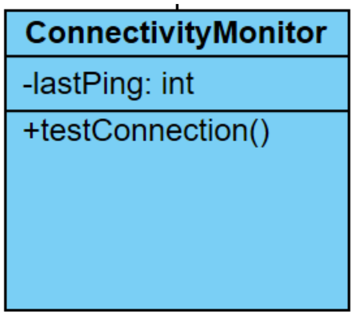
This section will define and describe the major classes used in the LDD. First, the class diagram will show all the major classes and how they relate to one another, followed by a metadata section. The metadata section will look more in-depth about each class and the contents within it.

2.2 Class Diagram


VirtualMachine
-id: int = <none>
-currentBuild: string = <none>
+beginBuild()
+openBuild()
+updateQuality(in ping: int)
Employee
-Username: string = <none>
-Password: string = <none>
+logIn(in username: string, in password: string)
+uploadBuild()
+beginVirtualMachine()
+replaceBuild()
+deleteBuild()
ConnectivityMonitor
-lastPing: int
+testConnection()
Manager
+updateSecurityKey(in username: string, in password: string)

2.3 Metadata

2.3.1 ConnectivityMonitor



Description: Monitors connectivity between the server and the employee.
Visibility: Public
Is Abstract: No

Additional Information:

Attributes

Name	Description	Read Only?	Multiplicity
lastPing	Stores the previous ping value	No	1

Operations

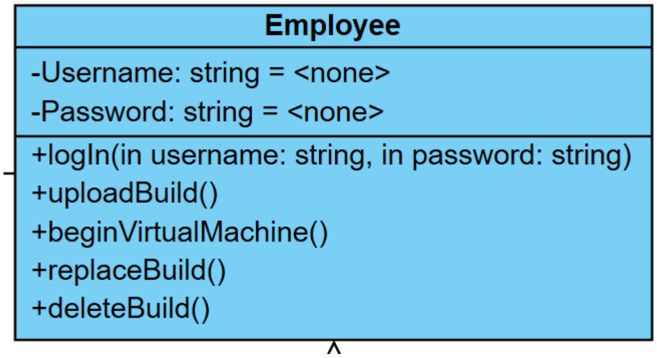
Name	Description	Is Query	Is Polymorphic
testConnection	Compares current ping to lastPing to see if performance has improved.	No	No

Processing Outlines

testConnection

```
    Retrieve current Ping
    Compare to lastPing
    IF currentPing > lastPing call updateQuality()
    ELSE IF currentPing = lastPing return
    ELSE call updateQuality()
```

2.3.2 Employee


Employee
-Username: string = <none>
-Password: string = <none>
+logIn(in username: string, in password: string)
+uploadBuild()
+beginVirtualMachine()
+replaceBuild()
+deleteBuild()

Attributes

Name	Description	Read Only?	Multiplicity
Username	Employee’s username for LDD	Yes	1
Password	Employee’s password for LDD	No	1

Operations

Name	Description	Is Query	Is Polymorphic
logIn	Starts the log in process to access the LDD	Yes	No
uploadBuild	Adds a new build to the LDD	No	No
beginVirtualMachine	Prompts the LDD to start the virtual machine initiation process.	Yes	No
replaceBuild	Replace a build on the LDD with a new one	Yes	No
deleteBuild	Delete a build on the LDD.	Yes	No

Processing Outlines

logIn

```
    Prompts the user to input username and password
    Compare input with username and password stored in the database
        IF the combination exists move onto the next step
        ELSE re-prompt user to input username and password with error message “username or password incorrect please try again”
    Prompt user to input security key
    Compare input with the present security key
        IF the input and security key match allow access to LDD
        ELSE re-prompt user to input security key with error message “incorrect key, please re-input security key”
```

uploadBuild

```
    prompts the user to select build from a local device to upload onto the LDD
    Confirm with the user that they want to upload the selected build
     IF yes, continue
     ELSE prompt the user to select another build to upload from a local device
 prompts the user to give a name to associate with the build
    upload the build and give it the name that the user input
```

beginVirtualMachine

```
    Create a virtual machine associated with the virtual machine class
```

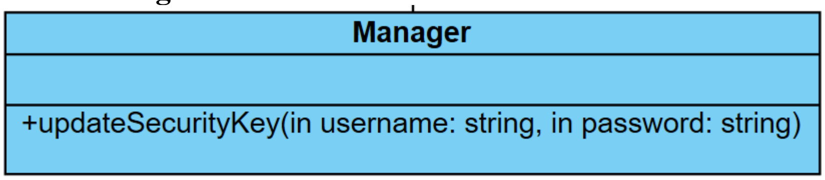
replaceBuild

```
    Checks to see if there are any builds on the LDD
        IF there are 1 or more builds on the LDD continue
     Else Break with the error message "no builds to replace"
    Prompts the user to select one of the builds stored on the LDD
    Confirm with the user that they want to delete the selected build
     IF yes, continue
        ELSE prompt the user to select another build to delete
 Prompts the user to select new build from a local device onto the LDD
    Confirm with the user that they want to upload the selected build
     IF yes, continue
     ELSE prompt the user to select another build to upload from a local device
 Prompts the user to give a name to associate with build to upload
    Delete build stored on the LDD, upload a new build to the LDD
```

deleteBuild

```
    Checks to see if there are any builds on the LDD
        IF there are 1 or more builds on the LDD continue
     Else Break with the error message "No builds to delete."
 Prompts the user to select a build stored on the LDD
    Confirm with the user that they want to delete the selected build
     IF yes, continue
     ELSE prompt the user to select another build to delete
    Delete selected build from the LDD
```

2.3.3 Manager


Manager
+updateSecurityKey(in username: string, in password: string)

Operations

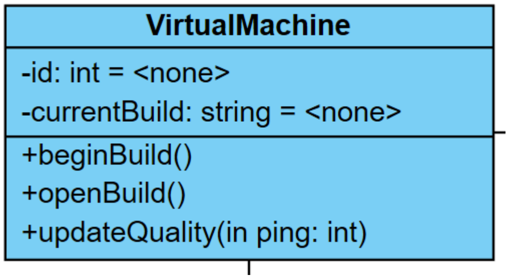
Name	Description	Is Query	Is Polymorphic
updateSecurityKey	Updates the security key stored in the database	Yes	No

Processing Outlines

updateSecurityKey

```
    Prompts the user for their username and password
    Compare input with username and password stored in the database
        IF the combination exists move onto the next step
        ELSE re-prompt user to input username and password with error message "username or password incorrect please try again"
    Prompt user to input security key
        Compare input with the present security key
        IF the input and security key match allow access to LDD
        ELSE re-prompt user to input security key with error message "incorrect key, please re-input security key"
    Prompt user to input new security key twice
    Validate that the two inputs are identical
        IF identical continue
        ELSE re-prompt user to input two security keys with message "security keys must match"
    Replace the security key in the database with the new security key
```

2.3.4 VirtualMachine



Attributes

Name	Description	Read Only?	Multiplicity
ID	The virtual machines unique ID generated on initialization	Yes	1
currentBuild	Shows which build from the LDD is loaded onto the virtual machine	Yes	1

Operations

Name	Description	Is Query	Is Polymorphic
beginBuild	Starts the loaded build and begins streaming to the user	No	No
openBuild	Loads build from the LDD	Yes	No
updateQuality	Changes quality of the stream to improve performance	No	No

Processing Outlines

```
beginBuild
    Boot build selected in openBuild
    Open connection to the user's computer to begin streaming
    Begin streaming build onto the user's computer
openBuild
    Checks to see if there are any builds on the LDD
        IF there are 1 or more builds on the LDD continue
     Else Break with the error message "No builds to load please upload a build first."
    Prompts the user to select a build that is loaded onto the LDD
    Loads selected build onto the VirtualMachine to be streamed to the user
updateQuality
    Checks to see whether the ping passed from the connectivity monitor warrants an increase or decrease of quality.
        IF ping warrants an increase of quality
            Decrease streaming quality to decrease strain on the user's computer
        ELSE
            Increase streaming quality as the increase will not affect performance according to connectivity
```

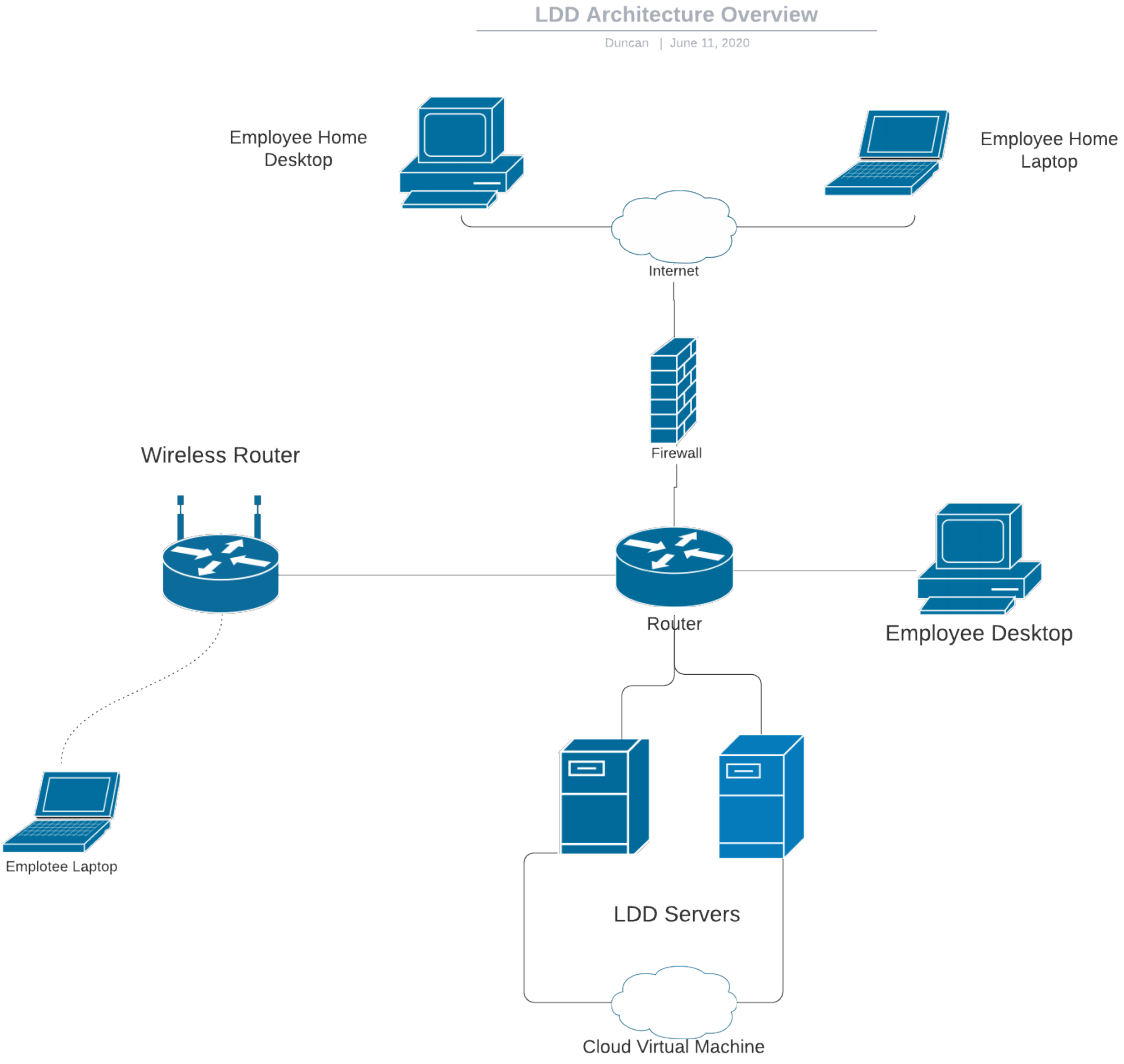
3.0 Architecture Design

3.1 Introduction

This section will discuss the proposed architecture of the LDD, the required hardware and software requirements, and the security plan that will address a variety of possible threats to the LDD. The main components are the LDD servers, and employee computers both at home and in the office. Since the system will connect over the internet, a firewall will be especially important to protect the LDD and the valuable information held within.

3.2 Infrastructure Model

3.2.1 Deployment Diagram – Architecture Overview



3.3 Hardware and Software Requirements

3.3.1 Required Hardware Components

- **Server(s)**: The LDD will be stored on a server, thus at least a single server will be required to run the LDD. Depending on company size, more servers may be preferred or required as many employees may need to access the system at the same time.
- **High Speed Router(s)**: As there will most likely be several servers storing the LDD, using a router to get an internet connection to all the servers simultaneously and easily is highly recommended. Since the servers will be streaming a cloud-based server, it is highly preferred that the router can handle high-speed connections to provide the highest quality and performance. Depending on how many servers the company has, more than one router may be required**.**
- **Computers to send home with employees:** As some employees' home computers may not be able to handle the strain that the LDD may produce, having some stronger computers to lend temporarily will lessen the chance that an employee will not be able to utilize the LDD from home.

3.3.2 Required Software Components

- **Anti-Virus Software:** Due to the nature of the information stored on the LDD, anti-virus software is a must to help mitigate the possibility of viruses and keep their impact as low as possible in case a virus still finds its way onto the LDD.
- **Data Encryption Software:** As with the virus, encryption will help keep the information and IP stored on the LDD safe and secure.
- **Firewall software:** As one last additional safety measure, having a high-end firewall will help prevent hackers and worms from gaining access to the LDD and the sensitive content held within.

3.4 Security Plan

	Threats	Fire	Flood	Power Loss	Circuit Failure	Virus	Internal Intruder	External Intruder	Eavesdrop
Components									
Server		1, 2, 3, 4	1, 3, 4	1, 5, 6	1, 4, 5, 6	1, 4, 7, 8, 9, 10,11	3, 9, 10,11	3, 9, 10, 11	7, 8, 9, 10, 11
Office Computers		1, 2, 3, 4	1, 3, 4	1, 5, 6	1, 4, 5, 6	1, 4, 7, 8, 9, 10,11	3, 9, 10,11	3, 9, 10, 11	7, 8, 9, 10, 11
Home Computers		--	--	--	--	1, 4, 7, 8, 9, 10,11	9, 10, 11	9, 10, 11	7, 8, 9, 10, 11

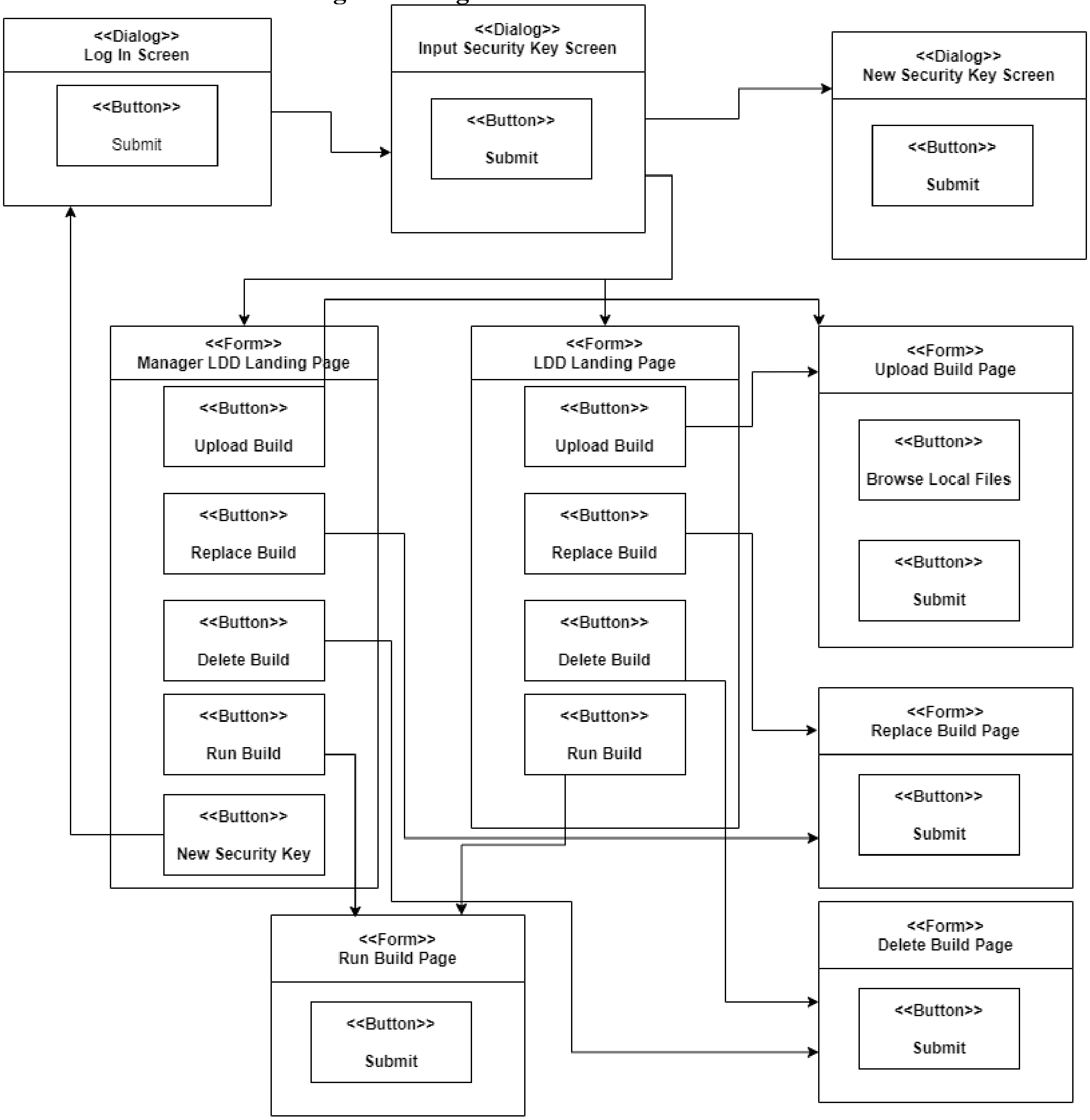
1. Disaster Recovery Plan
2. Halon Fire Suppression System
3. Staff Emergency Training
4. Backed up data offsite
5. Surge Protectors
6. Backup Generator
7. Anti-Virus Software
8. Firewall Software
9. Secure Passwords.
10. Semi-Frequently Updated Security Key
11. Encrypted Data

4.0 User-Interface

4.1 User-Interface Requirements and Constraints

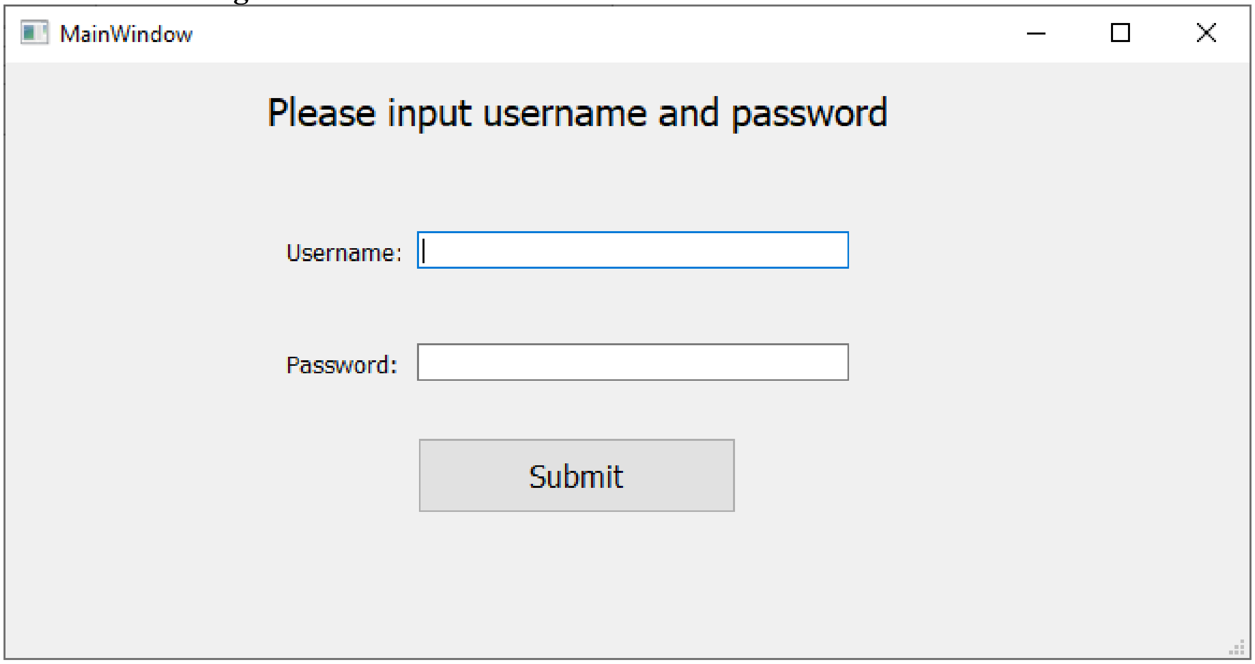
This final section is focused on the UI that the user will experience, and will be broken into two parts. The first part will be a window navigation diagram which will demonstrate how the user will navigate through different screens, and how those screens interact. The second section will show a more detailed representation of how those screens should look. The LDD is primarily a tool to allow developers to upload builds and begin testing from home as quickly and conveniently as possible. Keeping this in mind the UI for the LDD should be simple to work with and easy to understand.

4.2 Window Navigation Diagram



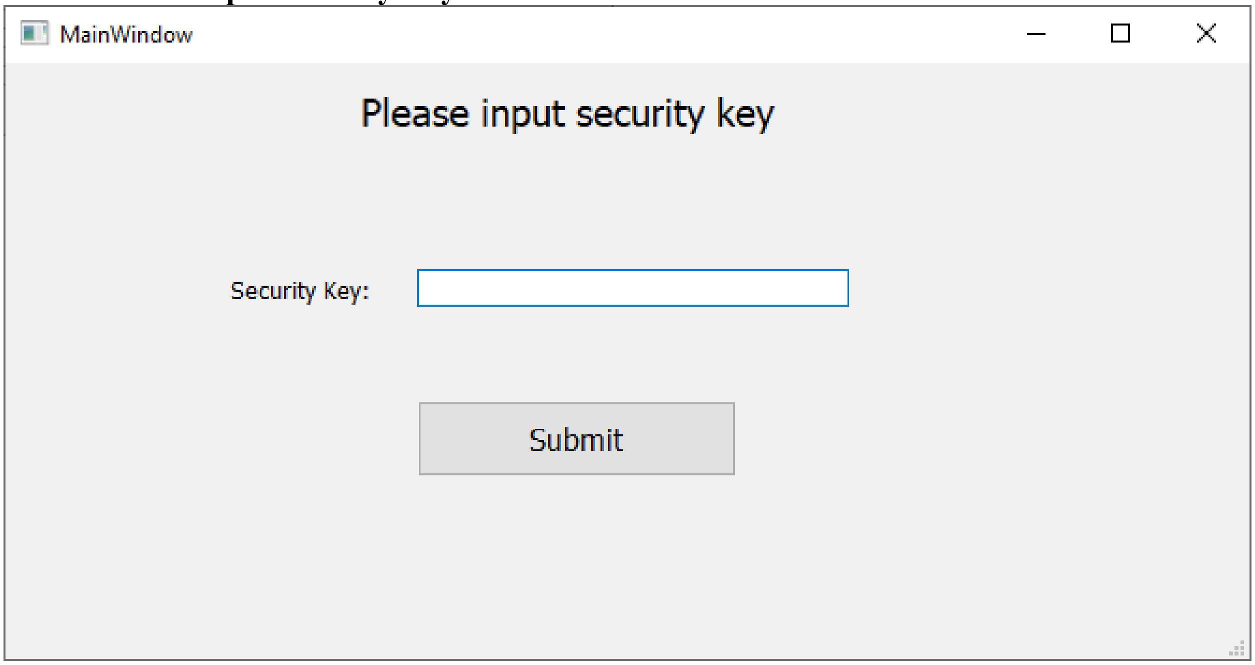
4.3 Forms: Screen / User-Interaction Design

Log in screen:



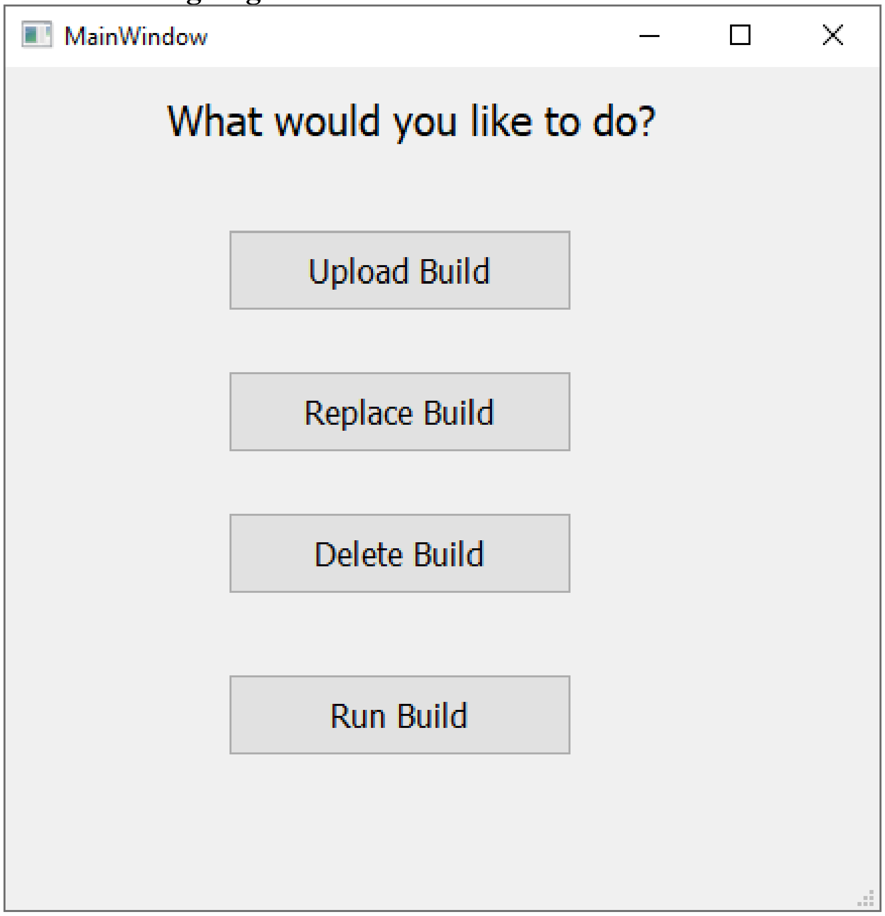
Please input username and password

Input Security Key Screen:



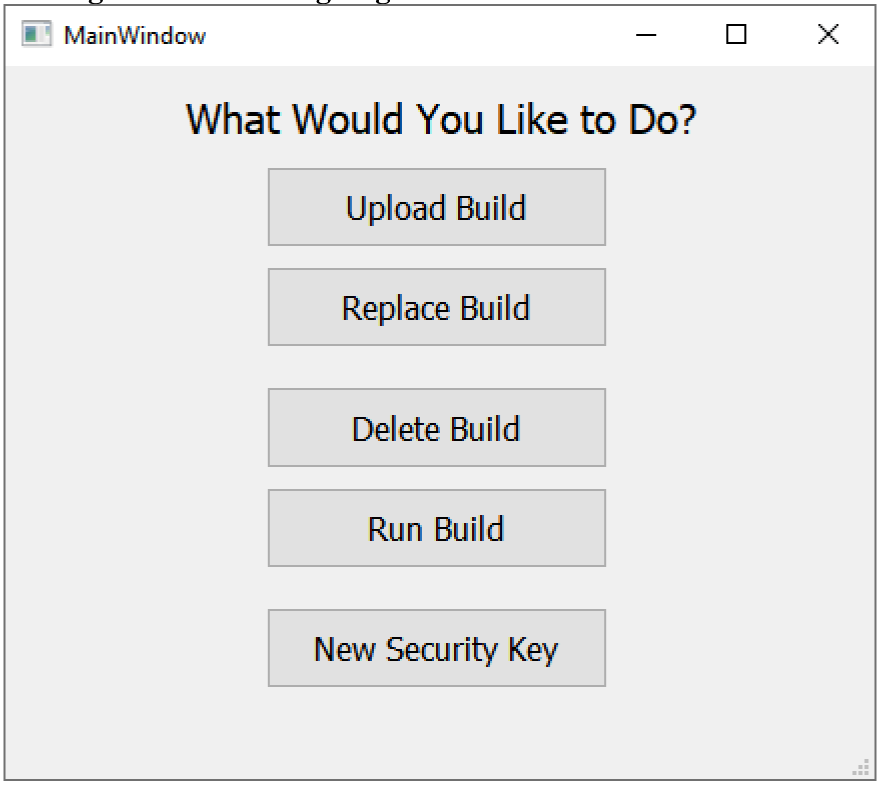
Please input security key

LDD Landing Page:



What would you like to do?

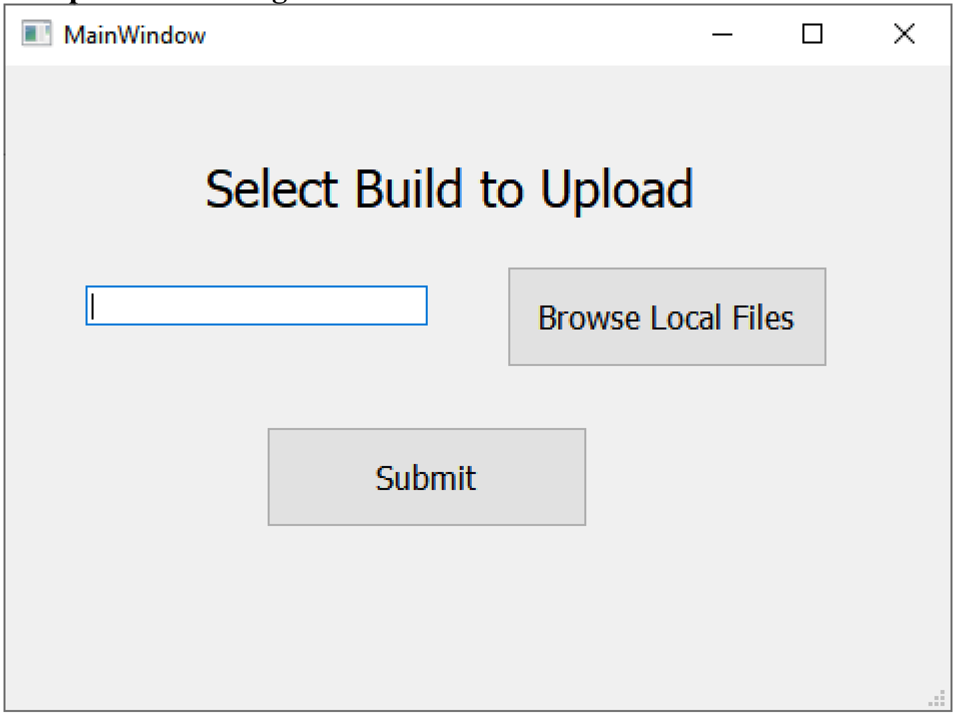
Manager LDD Landing Page



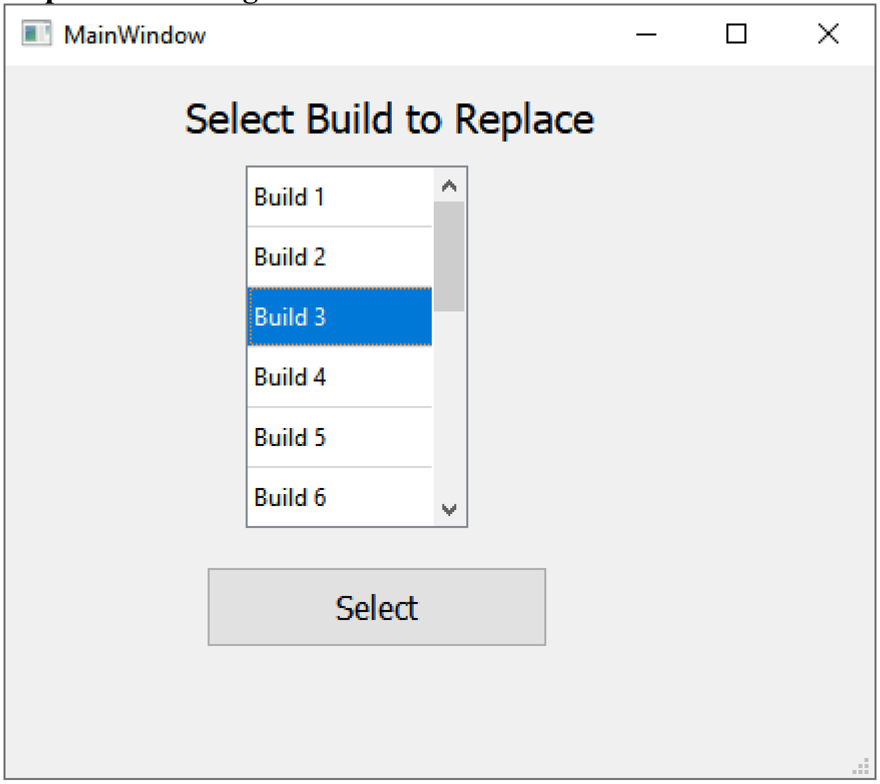
MainWindow

What Would You Like to Do?

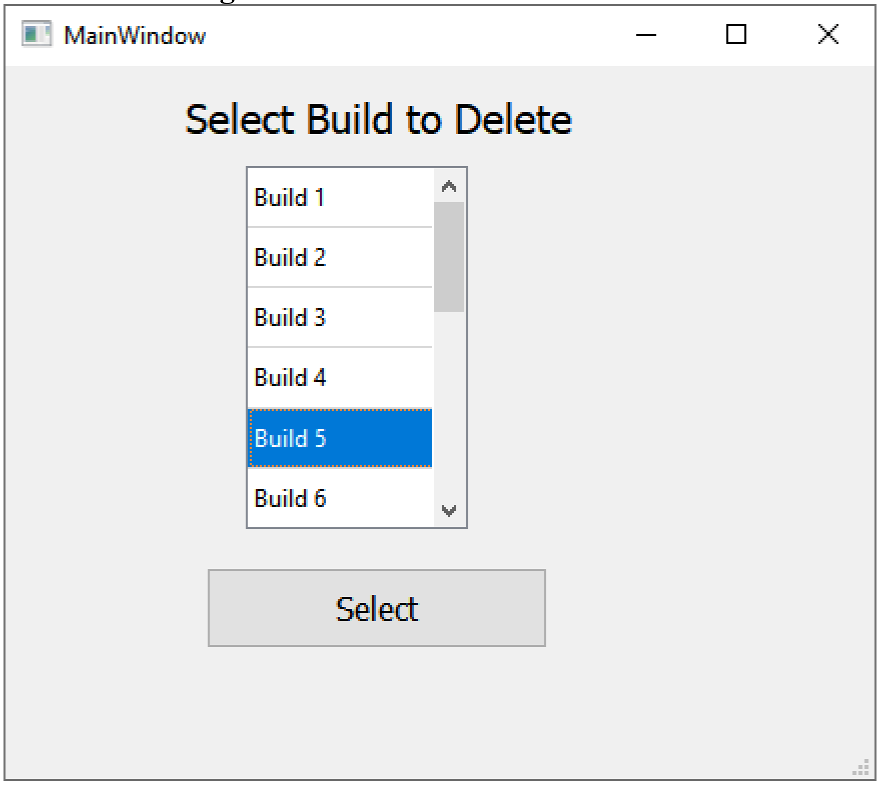
Upload Build Page:



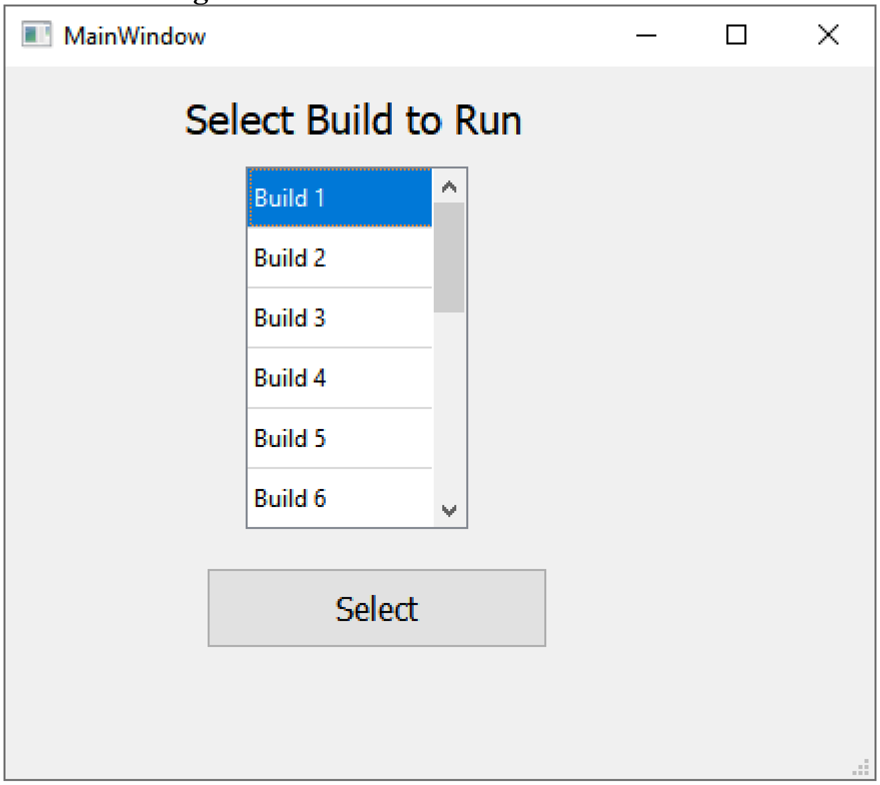
Replace Build Page:



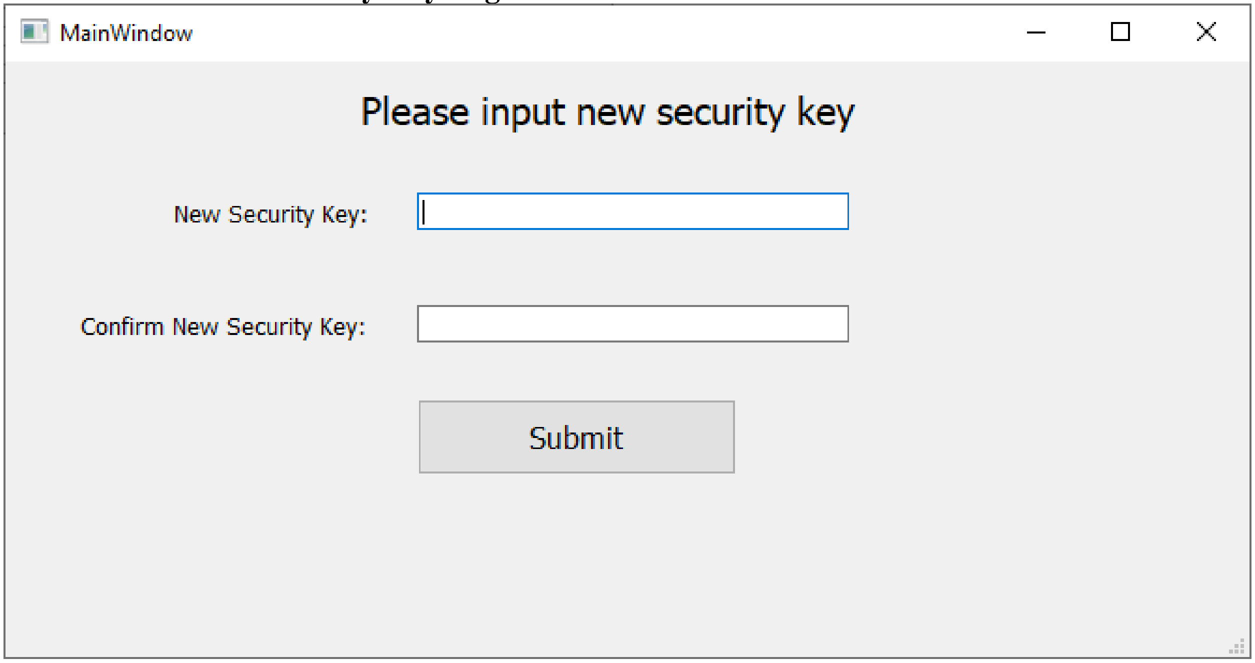
Delete Build Page:



Run Build Page:



New Security Key Page:



MainWindow

Please input new security key

New Security Key:

Confirm New Security Key:

5.0 Appendices

5.1 Bibliography / References

Cameron, Andrew (2020). *Systems Design*, various lectures [PowerPoint slides/Word Documents]. Retrieved from http://canvas.spu.edu.

Larman, Craig. Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design and Iterative Development. Third ed., Prentice-Hall, 2005.

Various Authors, “System Specification FIMS”, “Food for Everyone (FFE) System Specification”, “Food, Resources, Expenses, and Distribution System (FREDS) System Specification”, CSC 3150. Seattle Pacific University, Seattle. Report.