Software Version Description

for

<Project>

**Version <version\_number>**

**Prepared by Team 5**

**<date created>**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1.** **Introduction 1**

1.1 System Overview 1

1.2 Version Overview 1

1.3 Team Contributions 1

1.4 Document Conventions 1

1.5 References 1

**2.** **Inventory of Materials 1**

2.1 Documents Released 1

2.2 Executable Media Released 1

2.3 Software Projects\Assemblies Released 2

2.4 Test Projects\Scripts Released 2

**3.** **Completed Work 2**

3.1 Feature 1 2

**4.** **Current Design 2**

4.1 Current Software Components 2

4.2 Description and Evaluation of Current Architecture 3

4.3 Major Software Interfaces 3

4.4 Evaluation of Classes 3

4.5 Incorporating Design Patterns 3

4.6 Data Schema 3

4.7 Violations to Design Principles 3

**5.** **Remaining Work 3**

5.1 Remaining Software Features 3

5.2 Known Issues and Bugs 3

**Appendix A: Glossary 4**

**Appendix B: GitHub Information 4**

**Revision History**

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

# Introduction

## System Overview

*< This paragraph shall briefly state the purpose of the system and the software to which this document applies. >*

## Version Overview

*< This paragraph shall briefly state the state the changes made for this version of the software to which this document applies.>*

## Team Contributions

*<Breakdown of the contributions for each team member. This should give a report on what portions of the software system each team member contributed to. Note, writing documentation or putting together powerpoint slides are part of everyone’s responsibilities, just as coding and testing are everyone’s responsibilities. A student who does not contribute to one area or another will receive a lower contribution score. Everyone in the group should be able to identify clearly how they contributed to the development. This is not meant to prevent your group from working as a team. Instead it is meant to make sure that every member of the team is included in the project.>*

## Document Conventions

*<Any conventions used within this document>*

## References

*<List any other documents or Web addresses to which this document refers. >*

# Inventory of Materials

## Documents Released

*<Describe any relevant documents pertaining to this version of the software system. This can include relevant README files and user manuals.>*

## Executable Media Released

*<Describe any executable files that are included with this software system>*

## Software Projects\Assemblies Released

*<Identify and describe the current projects and assemblies included in your system. You do not need to include any ubiquitous assemblies used by the .Net framework or Java.>*

## Test Projects\Scripts Released

*<Identify and describe the test projects and scripts available for this version. Include information about any mocked items that may have been needed for testing.>*

# Completed Work

## Feature 1

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

3.1.1 Description

*<Provide a short description of the completed feature.>*

3.1.2 Completed Functionality

*<Enumerate the completed functionality associated with this feature. These must be clear and well-written statements of the functionality>*

3.1.3 Related Unit Test Cases

*<This should be a table of all unit test cases. It should have the name of the test case, the number of the functionality it tests from the above enumeration, and a color coded highlighting to indicate pass or failure. For the color coded highlighting, use green to indicate that a test passed, red to indicate that a test failed, and yellow to indicate that a test was not ran. Include the percentage of tests that were ran, the percentage of tests that succeeded, and the percentage of tests that failed.>*

3.1.4 Related Acceptance Tests

*<User acceptance tests should be written in the Gherkin format and should indicate specific test cases that were tested against the system.>*

# Current Design

## Description and Evaluation of Current Architecture

*<Explain the decisions made for the current architecture of the system. Evaluate the current architecture of the system against the principles that have been discussed up to the current point in the course. Defend the decisions that were made for your architecture. It is recommended to use specific examples from your software system to support your claims.>*

## Major Software Interfaces

*<Describe the major interfaces included in the software system that facilitate communication between the layers and explain each interface’s adherence to SOLID principles..>*

## Evaluation of Classes

*<For each class in your system, give a UML diagram that shows the class name, members, and methods of the class. You should give a description of the responsibilities of the class and explain its adherence to SOLID principles.>*

## Current Software Components

*<Describe and diagram the logical components of the current version of the software system. Your diagram should include the logical relationships between components. A UML package diagram may be appropriate for this purpose. You should also include a table that describes the responsibilities of each component.>*

## Current Software Layers

*<Describe and discuss the logical layers of the current version of the software system along with what components and classes comprise it. Also provide indicators for what layers can communicate with each other. A UML package diagram may be appropriate for this purpose. You should also include a table that describes the responsibilities of each layer.>*

## Incorporating Design Patterns

*<Describe the current design patterns that are being used within your software system and why they are relevant to the system.>*

## Data Schema

*<Describe the schema used for persistent storage. This may be a database schema or file structure used for permanent storage. >*

## Violations to Design Principles

*<Describe any known violations to the design principles discussed in this class and why they were necessary.>*

# Remaining Work

## Remaining Software Features

*<Any software features that are remaining to be implemented>*

## Known Issues and Bugs

*<Any known issues with the current software system. If your team decides to use an issue tracking system, please include information for viewing the system.>*

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SVD, including acronyms and abbreviations.>*

**Appendix B: GitHub Information**

*<Name of your group’s GitHub Repository and a list of features that are currently being used. For example, issue tracking system, wiki, etc.*>