Dominic Ferraro, Cameron Germano, and Dennis Lupin

DeSales University | [Company address]

Project Bubba’s

**Software Design Document Specification Template**

The Software Design Specification (SDS) sections provide you with guidelines related to the structure and the contents of SDS document. The Software Design Specification document includes at least these sections.

For the project, your team may have good reasons for wanting to deviate from this proposed outline. If a section is not applicable in your case, do not delete it; instead, give the topic heading and write "Not applicable".

You will note that there is some overlap in the content between different documents (i.e. the User Requirements Specification Document and the Software Design Specification Document.) This redundancy allows each document to stand on its own.

***ONLY THE SECTION TITLES COLORED IN ORANGE ARE REQUIRED TO BE COMPLETED.***

***DO NOT DELETE THE SECTIONS YOU ARE NOT COMPLETING AS THEY ARE A PART OF THE DOCUMENT***

Contents

[1 Introduction 3](#_Toc505762450)

[1.1 Purpose of this document 3](#_Toc505762451)

[1.2 Scope of the development project 3](#_Toc505762452)

[1.3 Definitions, acronyms, and abbreviations 3](#_Toc505762453)

[1.4 References 3](#_Toc505762454)

[1.5 Overview of document 3](#_Toc505762455)

[2 System architecture description 3](#_Toc505762456)

[2.1 Overview of modules / components 3](#_Toc505762457)

[2.2 Structure and relationships 3](#_Toc505762458)

[2.3 User interface 3](#_Toc505762459)

[2.4 User interface issues 3](#_Toc505762460)

[3 Detailed description of components (ONLY 2 ARE REQUIRED) 3](#_Toc505762461)

[3.1 X Component (or Class or Function ...) 3](#_Toc505762462)

[3.2 Y Component (or Class or Function ...) 4](#_Toc505762463)

[4 Reuse and relationships to other products 4](#_Toc505762464)

[5 Design decisions and tradeoffs 4](#_Toc505762465)

[6 Pseudocode for components 4](#_Toc505762466)

[7 Appendices (if any) 4](#_Toc505762467)

[Software component template for section 3 4](#_Toc505762468)

# Introduction

## Purpose of this document

Full description of the main objectives of the SDS document.

## Scope of the development project

This will be similar to what was written in the SRS.

## Definitions, acronyms, and abbreviations

Be sure to alphabetize!

## References

This section will include technical books and documents related to design issues. Be certain that the references you give are complete and in the appropriate format.

## Overview of document

A short description of how the rest of the SDS is organized and what can be found in the rest of the document. This is not simply a table of contents. Motivate and briefly describe the various parts!

# System architecture description

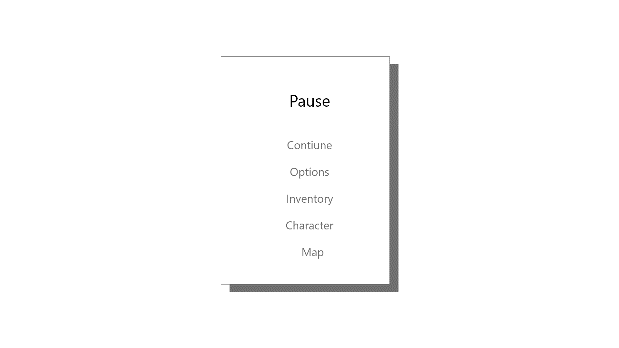
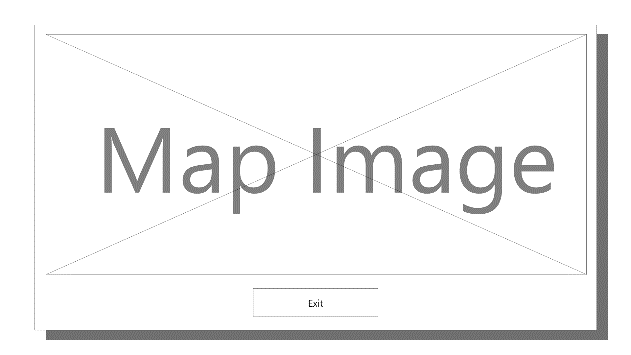
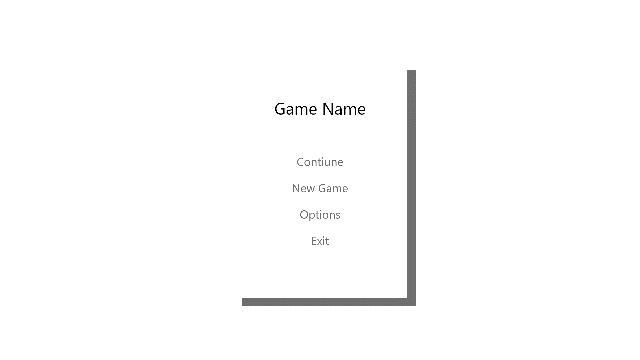
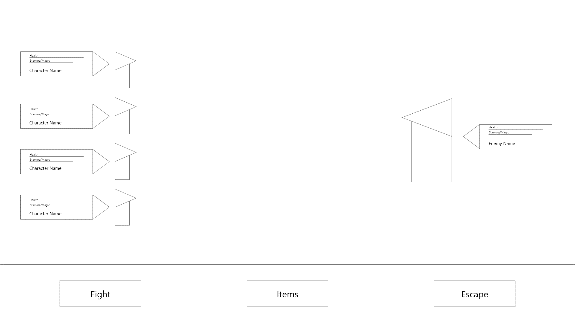
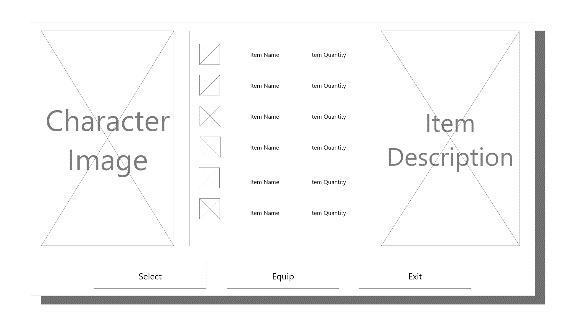
## Overview of modules / components

This subsection will introduce the various components and subsystems.

## Structure and relationships

Make clear the interrelationships and dependencies among the various components. Structure charts can be useful here. A simple finite state machine can be useful in demonstrating the operation of the product. Include explanatory text to help the reader understand any charts.

## User interface



## User interface issues

No error messages will be displayed in the game. If they do something that is not valid nothing will occur. For example, if they try to walk into a wall the character won’t do anything.

# Detailed description of components (ONLY 2 ARE REQUIRED)

## Save File Component (or Class or Function ...)

Use exactly the template shown at the end of the document.

|  |  |
| --- | --- |
| Identification | Save File |
| Type | Text File |
| Purpose | To resume a game save |
| Function | To read and right values obtained from playing the game |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | Accessed by selecting save game from the main menu. |
| Interfaces | If game cannot be saved or loaded due to a corrupted save file an error message will be displayed. |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| Data | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary. |

## Leveling progression Component (or Class or Function ...)

|  |  |
| --- | --- |
| Identification | Leveling progression |
| Type | A class |
| Purpose | To progress the character throughout the game |
| Function | To balance stats out over the progression of the game. Stats will multiply more quickly in the beginning of the game versus at the end |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | This component is dependent on any feature of the game that rewards XP, such as combat. |
| Interfaces | When the player levels up they will be prompted on which stats they would like to increase |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| Data | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary. |

# Reuse and relationships to other products

For teams doing enhancement work, reuse is an important issue. Most enhancement work should focus on extending, rather than replacing, the design and product development from earlier semesters. For teams doing new development, reuse can also be an important strategy. In some cases, there is freeware that could be incorporated. In other cases, there are existing modules or classes that could be adapted. Another possibility is the use of special tools that produce open source results and thus permissible under the terms of this course.

This section should include the following subsections as appropriate:

* How reuse is playing a role in your product design
* How reuse is playing a role in your product implementation (and the motivation for changes)
* If you are not reusing material that is available, then give motivation for why it is being thrown out.

# Design decisions and tradeoffs

Keeping the game simplistic is a core aspect of our design approach. We would like to avoid confusing the user of where their attention should be concentrated. We are giving the user only what is needed and thus brings us closer to the goal of being an old school RPG.

# Pseudocode for components

Utilize the use cases to create pseudocode for components.

# Appendices (if any)