

COMP 4770 – Architecture Document Specification

The second milestone for COMP 4770 is to write an Architecture Document for the system you will be creating. This document is intended to give a high-level overview, as well as technical description of your system, and its underlying architecture. You will use this document as a guide when implementing your system, so make it as detailed as possible.

For UML diagram specifications and examples use: <https://www.smartdraw.com/uml-diagram/>

It is expected that your final game will deviate from this document, but try to make it as accurate as possible.

Due Date: February 6th, 2019 @ 11:59pm (on GitHub)

Architecture Document

1. Introduction

- 1.1. Purpose – What is the purpose of this document?
- 1.2. Scope – What exactly will this document cover?
- 1.3. Definitions / Abbreviations – List any technical terminology used in the document
- 1.4. References – List any outside references used in the creation of the document

2. Architectural Goals / Constraints

- 2.1. Goals – What are the goals the system attempts to achieve
- 2.2. Constraints – What are the constraints on the system?

3. Logical View

- 3.1. Major Logical Components – Describe the major logical components (including project modules) of the system and include a high-level *UML Package Diagram* showing their organization and interaction.
- 3.2. Classes – Describe the most important classes in the system, their relation to the major logical components, and their organization / interactions. Give *UML Class Diagrams* for each class discussed.

4. Use Case View

- 4.1. UML State Diagram – Describe the major user flow within the system and give a high-level *UML State Diagram*. Major states for your game may include “Main Menu”, “Playing Game”, “Level Editor”, “Player Profile”, etc. The state diagram should show all possible transitions between these major system states.
- 4.2. Architecturally Significant Use Cases – Take the significant uses cases from your Software Requirements document, and for each of them create a software UML Sequence Diagram.

5. Process View

- 5.1. Processes / Threads – List all processes and threads which run during normal system operation. This should address issues such as startup and shutdown, concurrently running system components, fault tolerance.
- 5.2. Classes – Give a standard UML diagram showing the assignment of system classes to system processes.

6. Deployment View

- 6.1. Deployment View – Give a description of the deployment of the system, and the physical nodes required to be running / set up for the system to function properly. Give the related *UML Deployment Diagram*.

7. Size and Performance

- 7.1. Size – Describe how many users the system should support
- 7.2. Memory – Describe how large the database / asset storage system will be. For our case, this will include max number of registered users, how much data each user requires, how many levels can be made, etc.
- 7.3. Performance – For each major task in the game, describe how long each should take (upper/lower bound). Also list game performance such as ideal frames per second, loading times, etc.

8. Database Organization

- 8.1. Overview – Give the high-level view of the organization of the system's database structure.
- 8.2. Specifics – For each major database item, give specific storage details, and database organization.

9. Component Organization

- 9.1. Overview – Give a high-level overview of the component architecture of the system. Describe where the files and software are stored during development and maintenance.
- 9.2. Component Diagram – Give a detailed *UML Component Diagram* of the organization of the physical software components of the system. These include source code, assets, binaries, websites, configuration files, etc.