

**THE DUNGEON OF DOOOM**

**SOFTWARE ARCHITECTURE DOCUMENT**



REVISION HISTORY



|  |  |  |
| --- | --- | --- |
| **DOCUMENT NUMBER:** | **RELEASE/REVISION:** | **RELEASE/REVISION DATE:** |
| 1 | V1.0 | Monday, Nov 28 |

TABLE OF CONTENTS



[1. Introduction](#page3) [3](#page3)

[1.1 Purpose](#page3) [3](#page3)

[1. 2 Scope](#page4) [3](#page4)

[1. 3 Glossary and acronym list](#page5) [3](#page5)

[1. 4 Functional Requirements](#page5) [4](#page5)

[1.5 Non-functional requirements](#page8) [4](#page8)

[2. Architecture overview](#page9) [5](#page9)

[2.1 Three-Tier Architecture](#page10)  5

[2.2 Design Pattern](#page11)  5

1. INTRODUCTION



The purpose of this project is to analyse, specify, design and implement a multiplayer on-line game called “The dungeon of dooom”. The main focus of our project is in implementing object-orientated analysis and design with SCRUM and Agile development methodologies for project development and management.

Apart from that, the system is designed and going to implement in 3-tier architecture for better implementation. Finally the quality assurance of our system is achieved through appropriate testing.

* 1. PURPOSE



This document provides an architectural overview of the online multiplayer game. The primary purpose of The Dungeon of Doom is to set up an online-based multiplayer game, where player can play game with a software player.

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions, which have been made on the system.

1. 2 SCOPE



The project team and developers are working intently on making this project a fully operational and as robust as possible. This document applies to the overall design of the system. It contains information relating to the architectural design of the software, the Structure of the Database, and of the physical servers hosting the site.

1. 3 GLOSSARY AND ACRONYM LIST



Term list:

* Javascript: (originally) web-browser interpreted programming language for enhancing web sites in a dynamic way.
* Scrum: is an iterative and incremental agile software development framework for managing product development.
* Java: dynamic, object-oriented programming language

Acronym list:

* SAD: Software Architecture Document.
* API: Application Programming Interface, a protocol used as an interface to allow communication between different components.
* CSS: Cascading-Style Sheets, document that describes the appearance of web pages.
* JSON: JavaScript Object Notation, a text-based standard for human-readable data exchange.
* TDD: Test-driven development
* MVC: Model-View-Controller, a software architecture pattern that separates the physical way to store data, the business logic and the appearance to the user.

1. 4 FUNCTIONAL REQUIREMENTS



1. The framework requirements are
2. a user interface suitable for a human player
3. a program interface suitable for a software player (a “bot”)
4. a means to define the structure and behaviour of a dungeon-like game
5. a means to track the scores of players
6. The name of the game is “The Dungeon of Dooom” and its requirements are
7. A dungeon comprising a collection of roooms
8. A rooom has (at least):
9. Gold: sometimes, which the player can collect; the amount required to win/exit is announced at the start of the game
10. Passages: leading to other roooms or to the exit
11. Other items: whose presence and function is game-defined
12. A dungeon can be of arbitrary size
13. A dungeon should contain atleast as much gold as is required to win, and atleast on exit passage.
14. A visualization of the (whole) dungeon, which must be updated after each action that changes it
15. A player’s location must be shown on the dungeon visualization
16. A bot’s location must be shown on the dungeon visualization
17. Player interactions with the dungeon include:
18. Indicating which way to move
19. Picking up gold (or other items)
20. Looking to find out about the current room
21. Leaving the game (giving up)
    1. NON-FUNCTIONAL REQUIREMENTS



1. The proposed system must be as robust as possible so that the system must be able to recognize and handle any kind of errors in features.
2. The system is implementing with a mix of appropriate languages.
3. 2. ARCHITECTURE OVERVIEW



This document is the first approach to present the information of this project in a structured fashion and discuss its architecture.

2.1 THREE-TIER ARCHITECTURE



The online multiplayer will be implement by three-tier client–server architecture model. Three-tier architecture allows any one of the three tiers to be upgraded or replaced independently. The user interface is implemented on a desktop PC, which is client side of multiplayer game and uses a standard graphical user interface with different modules running on the application server. The relational database management system on the database server contains the computer data storage logic, which is implemented using json and msql.

The three tiers in a three-tier architecture are:

* Presentation Tier: It includes the top level and presents the information related to services available on client side. This tier sends data to other tiers to communicate. For eg: sending result of game to database. All communication with the Presentation layer is done through Web Services.
* Application Tier: Also called the logic tier or business logic, this tier is responsible for controlling application functionality by performing detailed processing.
* Data Tier: Houses database servers where information is stored and retrieved. The Data Layer provides persistence for the system and all communication is done through SQL queries and views. Data in this tier is kept independent of application servers or business logic. For eg: storing score, storing records of players

2.2 DESIGN PATTERN



The design pattern associate with our multiplayer game is MVC (Model View Controller) design pattern. The MVC design pattern clearly separated the web application’s behaviour, presentation and control. The modularity of this design pattern allows for easier code reuse, more centralized control, bugs easier to track down and code easier to modify. The client will be running on presentation layer. The presentation, or view, of the multiplayer game will be implemented in HTML, CSS and Javascript. The database layer includes json files and mysql.

The business tier includes the model classes of multiplayer game.

Apart from that, the system needs to satisfy any functional, non-functional or aesthetic needs in a software system. The Dungeon of Doom follows the three-tier architectural style, which includes presentation tier, business tier, and data tier. The following is a simple description of what will be included in each of the tiers:

Presentation Tier

Data Tier

Business Tier

* Presentation Tier: used to present the information to the player.
* Business Tier: used to implement the logic of the system
* Data Tier: To storing the data and other external services that the system may use.