# Project Design Document:

**Section 1 - Project Description**

1.1 Project

A-Maze-ing Runner

1.2 Description

**The Project is a 2D real-time streaming maze runner game. Multiple players run in a maze through different complexity levels to reach the finish point before opponents. Players need to overcome obstacles, traps, puzzling shortcut or collect widgets for the advantages to against other players.**

1.3 Revision History

|  |  |  |
| --- | --- | --- |
| Date | Comment | Author |
| 28 July 2024 | 1st Revision | Chow, Tsz Chun Samuel |
|  |  |  |
|  |  |  |
|  |  |  |

**Section 2 - Overview**

2.1 Purpose

The project focus on providing a real-time streaming game play with low latency and targeting for game or maze lovers to compete and solve the maze.

2.2 Scope

Maze Generation:

Maze is generated through a map file when games started. The map file is a text file storing numbers for object identification to construct the maze. The maze can be a perfect maze or braid maze.

User Interface:

Real-time rendering of the maze. Player uses keyboard to control the character. The game currently supports up to maximum 4 players at a time.

Real-Time Streaming:

Continuous updates and smooth transitions when user navigate the maze. A Low-latency communication is ensured to feedback different user actions during gaming.

Synchronization:

Dynamic objects in the game is synchronized as well as the players positions and actions.

2.3 Requirements

Break down the requirements to provide a ballpark estimate.

2.3.1 Functional Requirements

- Requirement 1 (R1): The system shall auto detect if the server is running or not when player

game started.

- Requirement 2 (R2): The system shall auto assign player id (1 to 4) to the user who joins and

request game start accordingly.

- Requirement 3 (R3): Any player quit the game should not affect the gaming availability and

experience.

- Requirement 4 (R4): Once the number of the game board players is less than the maximum

player design. It allows players to join any time.

- Requirement 5 (R5): Player should be able to control the game character by keyboard.

2.3.2 Non-Functional Requirements  
Non-functional requirements describe how the system performs a function, focusing on aspects such as:

1) Performance:

Design should provide 60 FPS refresh rate. The system is designed with scalable maximum number of players point of arts.

2) Reliability (availability, error handling):

Each player gaming loop should have at least 1ms interval in purpose of CPU resources saving.

2.3.3 Technical Requirements  
  
  - Hardware: The system shall run on servers with at least 4GB of RAM and 1 CPU for a 20

players gaming sizes.

  - Software: The system shall be developed using Java and run with a WebSocket environment.

2.3.5 Estimates  
Provide a breakdown of tasks and estimated hours required to complete them.

|  |  |  |
| --- | --- | --- |
| # | Description | Hrs. Est. |
| 1 | Create an empty UI board with a simple colored box with keyboard control | 3 hrs |
| 2 | Create Maven project with Jakarta WebSocketServer and Client, text simple communication | 3 hrs |
| 3 | Create each player client WebSocket handler | 1 hrs |
| 4 | Test communication to server with object information display | 1 hrs |
| 5 | Create data format center, design the player object, and design the player object information in terms of byte format style | 3 hrs |
| 6 | Introduce multiplayer instance with player factory | 2 hrs |
| 7 | Broadcasting players’ information and rendering all players’ character in all player’s user interface | 3 hrs |
| 8 | Characters, maze objects Sprite design | 3 hrs |
| 9 | Maze map design and rendering | 3 hrs |
| 10 | Maze object collider algorithm | 3 hrs |
|  | TOTAL: | 25 hrs |

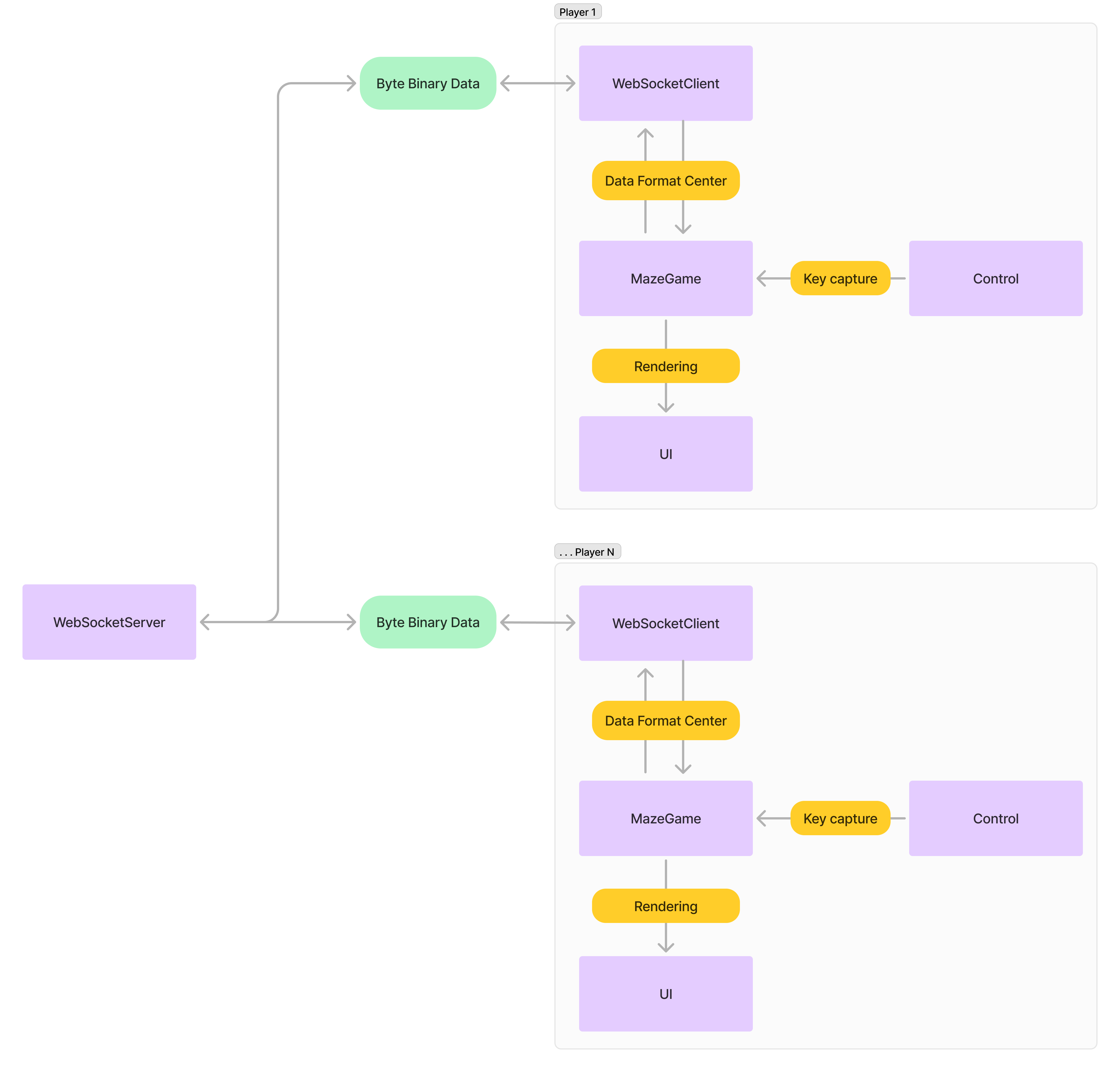
2.3.6 Traceability Matrix

Cross reference this document with your requirements document and link where you satisfy each requirement (In Progress)

|  |  |
| --- | --- |
| SRS Requirement | SDD Module |
| Req 1 |  |
| Req 2 |  |
| Req 3 |  |
| Req 4 |  |

**Section 3 - System Architecture**

3.1 Overview



A diagram of a network

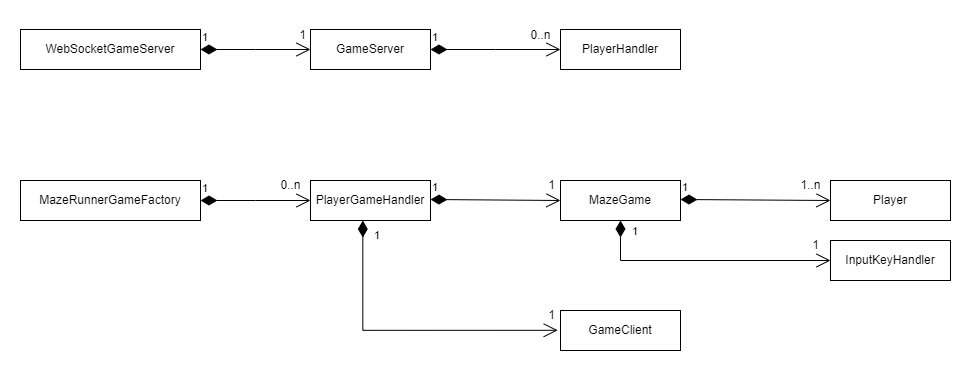
Description automatically generated

The system introduces communication between one game WebSocketServer and multiple WebSocketClient players. The data would be transmitted through the byte binary format in a low latency design. Meanwhile, DataFormatCenter is aided to convert the binary data back to useful object information for the MazeGame object to proceed the game logic and render the UI. On the other hand, MazeGame would capture user input for the character control at the same time transferring the updated character information to the WebSocketServer in the binary format. Any player client’s updated information received at the WebSocketServer would be broadcasted to the rest of the players.

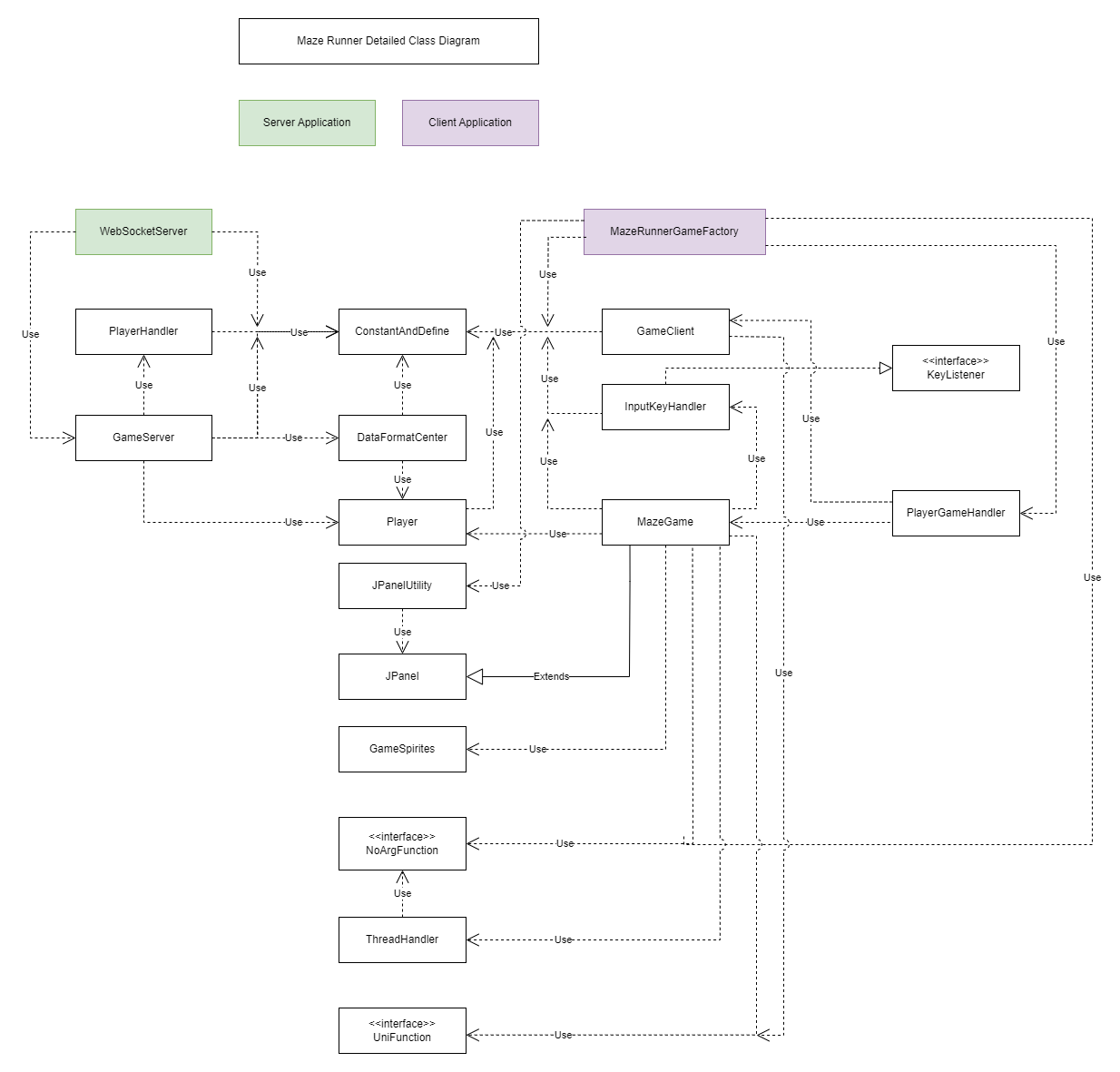
3.2 Architectural Diagrams

a) Class Diagram

Overview:



Detailed:



**Section 4 - Data Dictionary**

Brief description of each element in this module or a link to an actual data dictionary

(In Progress)

|  |
| --- |
| Table |

|  |  |  |
| --- | --- | --- |
| Field | Notes | Type |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Section 5 – Data Design**

Describe the data contained in databases and other shared structures between domains or within the scope of the overall project architecture

5.1 Persistent/Static Data

Describe/illustrate the logical data model or entity relationship diagrams for the data

(In Progress)

5.1.1 Dataset

Describe persisted object/dataset and its relationships to other entities/datasets using Logical Data Model (e.g. Entity Relationship Diagram).

(In Progress)

Example:  
Entities:

a) User  
  - Attributes: UserID (PK), Username, Password, Email, Role  
  - Relationships: One-to-Many with Orders

**Section 6 - User Interface Design**

6.1 User Interface Design Overview

Colored dot are players

A screenshot of a game

Description automatically generated

6.2 User Interface Navigation Flow

One Single UI

6.3 Use Cases / User Function Description

One Single UI, have prompt to show which player wins.