**Low-Level Architecture and Data Models**

**P01: JAM**

**<team member names & ids>**

| **Student ID** | **Name** |
| --- | --- |
| **24100175** | **Abdul muiz** |
| **24100127** | **Abdur Rafae Haroon** |
| **24100282** | **Waleed Nadeem** |
| **24100308** | **Hafsa Ahmed** |
|  |  |

**Table of Contents**

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) System Architecture 4

[2.1 Architecture Diagram—](#_1fob9te)As it is in the prototype code 4

[2.2 Architecture Diagram—](#_3znysh7)As it should-be 4

[3.](#_2et92p0) Data Models 5

[4.](#_tyjcwt) Tools and Technologies 6

[5.](#_3dy6vkm) Who Did What? 7

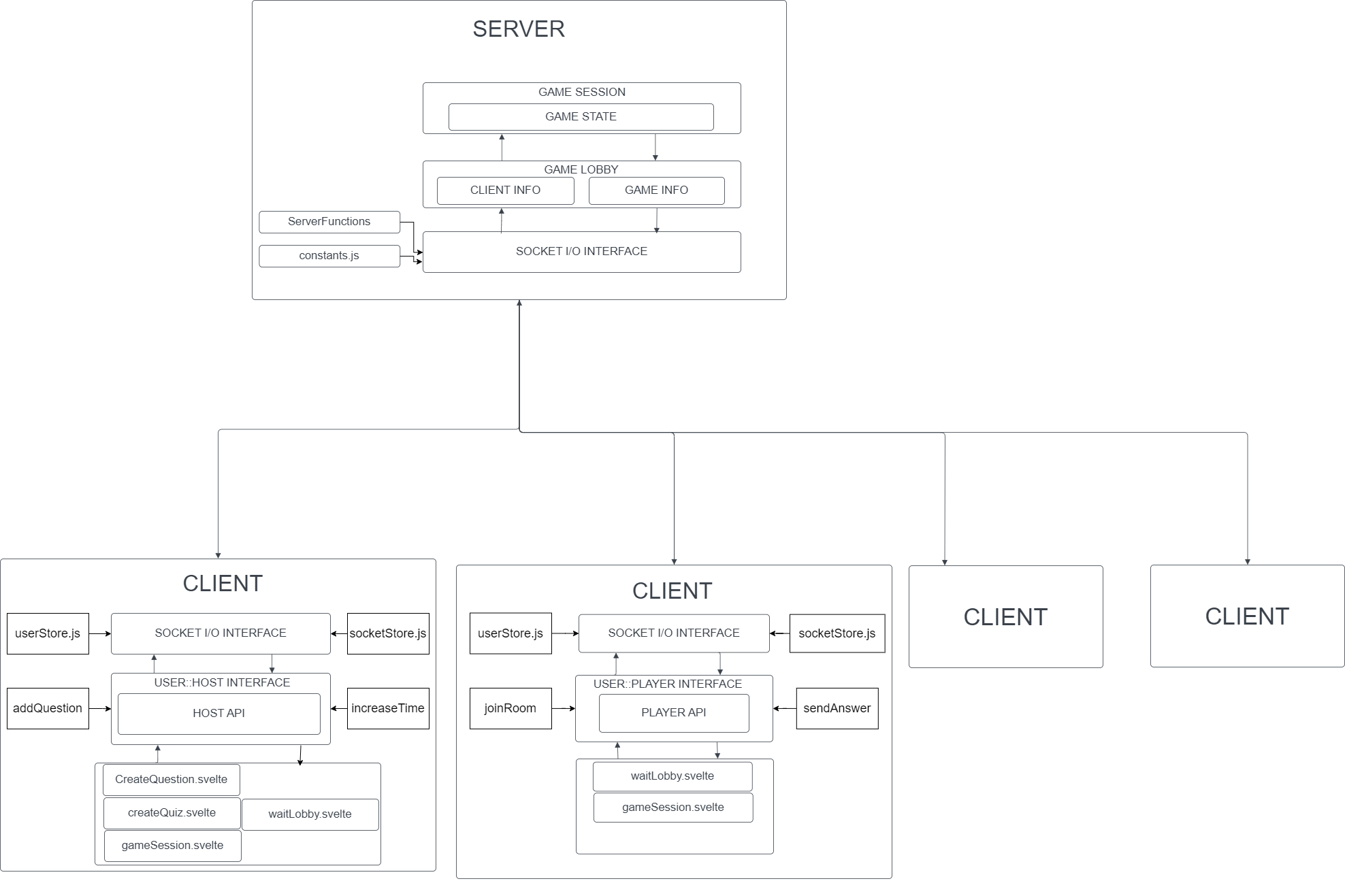
[6.](#_1t3h5sf) Review checklist 7

# Introduction

The proposed project is a user-hosted trivia web application. This application aims to create an interactive quiz-based learning experience, taking its inspiration from the popular game-based learning platform, ‘Kahoot!’. Designed for students, teachers, trainers or simply anyone who wants to challenge their friends to a quiz, the extent of the web application’s features allow anyone to quickly create and host a quiz-based session and let others join in and participate. Thus, within a session, there will exist a host user who creates the trivia while the participants joining voluntarily act as players. The players will use the quiz ID given by the host to join a particular session.

# System Architecture

## Architecture Diagram



## Architecture Diagram

***(DIAGRAM WILL BE SAME)***

**Description**

The system architecture diagram derived from the prototype code is an apt representation of the architecture of the code as it should be therefore, no updation to the diagram was required.

The classes/ modules in the diagram are as follows:

* **Server:**

The server class contains three modules, all of which collectively cover the server side’s functionality and logic:

1. Game Session: It manages the state of the game
2. Game Lobby: It manages information related to clients as well as the state of the game.
3. Socket I/O Interface: It receives input from external components, ServerFunctions and constants.js, and manages communication with the client side through socket IO.

* **Client (Host):**

The client class for a host contains two modules:

1. Socket I/O interface: It handles communication with the server using socket IO, and receives input from userStore.js and socketStore.js.
2. User: Host Interface: It contains the Host API and receives input from various functions inside the CreateQuestion.svelte, createQuiz.svelte, gameSession.svelte, and waitLobby.svelte components for example, addQuestion and increaseTime from createQuiz.svelte.

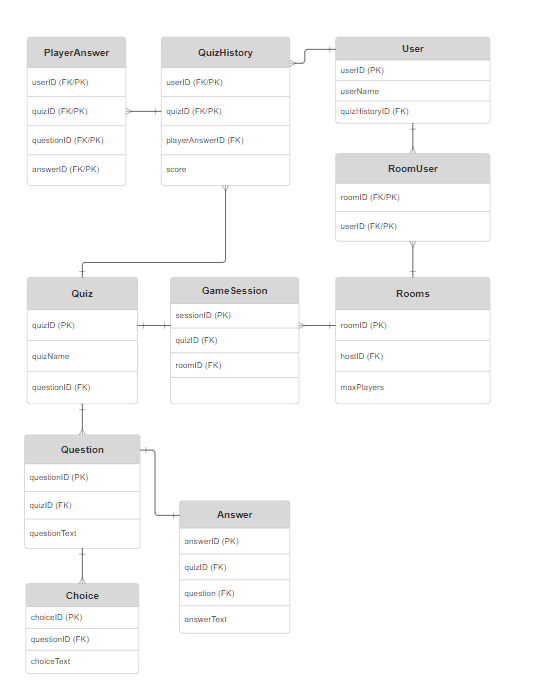
* **Client (Player):**

The client class for a player contains two modules:

1. Socket I/O interface: It handles communication with the server using socket IO, and receives input from userStore.js and socketStore.js.
2. User: Player Interface: It contains the Player API and receives input from various functions inside the gameSession.svelte, and waitLobby.svelte components for example, sendAnswer from waitLobby.svelte.

Thus, the host and the players interact with the server through the common Socket IO interface with their own APIs. The various Svelte components serve as the user interfaces triggering actions in the respective User Interfaces.

# Data Models



# Tools and Technologies

<List down tools and technologies that you are using for development and deployment. Make sure that you mention name and version of the tools.>

**Development tools:**

* Front-end framework:
  + Svelte (latest version: 4.2.1): A lightweight and efficient JS framework for building user interfaces.
* Backend framework:
  + Node.js (version 16+): A javascript run-time for server-side development.
  + Express.js (latest version): A Node.js library for building web applications, simplifying routing and server-side development.
* Database:
  + Supabase (latest version): An open-source alternative to Firebase that offers real-time database and authentication services.
* Real-time communication:
  + Socket.io (latest: 4.7.2): A JS library for real-time bidirectional event-based communication between the client and the server.
* Version control:
  + Git (latest version): A distributed version control system for tracking changes in the project.
* Code editor:
  + VS Code (latest version): A popular source code editor with support for various programming languages and frameworks.
* Package manager:
  + npm (Node package manager, combined with Node.js): Used for managing and installing project dependencies.
* Project management:
  + Jira: A management tool to facilitate agile development, thereby allowing efficient tracking of tasks and project progress.

**Deployment tools:**

* Deployment platform (Backend):
  + Heroku: A platform-as-a-service (PaaS) to streamline and simplify the process of deploying and managing web applications.
* Deployment platform (Frontend):
  + Vercel: A platform-as-a-service (Paas) that makes the deployment of web applications seamless and efficient.
* Scalability:
  + Amazon EC2 auto-scaling: For automatically adjusting the number of EC2 instances in response to changing traffic loads. It will be a great choice for scaling our web application based on the network traffic.

**Database tools:**

* Supabase (latest version): Will be used as a primary database for storage and retrieval.

# Who Did What?

| **Name of the Team Member** | **Tasks done** |
| --- | --- |
| Abdul Muiz | 3 |
| Hafsa Ahmed | 2 |
| Abdur Rafae Haroon | 4 |
| Waleed Nadeem | 2 |
| Bisma Nawaz | 1 |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

| **Section** **Title** | **Reviewer Name(s)** |
| --- | --- |
| 1,2,4 | Abdul Muiz |
| 4 | Hafsa Ahmed |
|  |  |
|  |  |