System Design for SFWRENG 4G06

Team #12, Team 12.0
Kanugalawattage, Anton
Subedi, Dipendra
Rizkalla, Youssef
Leung, Tamas
Zhao, Zhiming

January 18, 2023

1 Revision History

Date	**Version**	**Notes**
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	**description**
SFWRENG 4G06	The Software Engineering Capstone Project Course at McMaster University
Figma	A collaborative web application for interface design

Contents

List of Tables

List of Figures

3 Introduction

This document describes the design for the CodeChamp system. The design document is split up into three documents, the Module Interface Specification, Module Guide and the system design document. Other relevant documentation is listed below:

1. Development Plan
2. System Requirements Specification
3. Hazard Analysis

4 Purpose

This document is written to describe the architecture and the design decisions in the CodeChamp system. Primarily, it introduces the scope of the system to demonstrate the possible interactions with the outside world. Moreover, it gives an overview of the project, recounting the important components from a high level and describing the normal behavior of the system. Additionally, it gives a high-level overview of the event handling mechanisms in place for undesired behavior. Since the CodeChamp system is user-centric, the document also describes mock-ups for the user interface design of the system. Finally, a timeline is given for the implementation of the system.

5 Scope

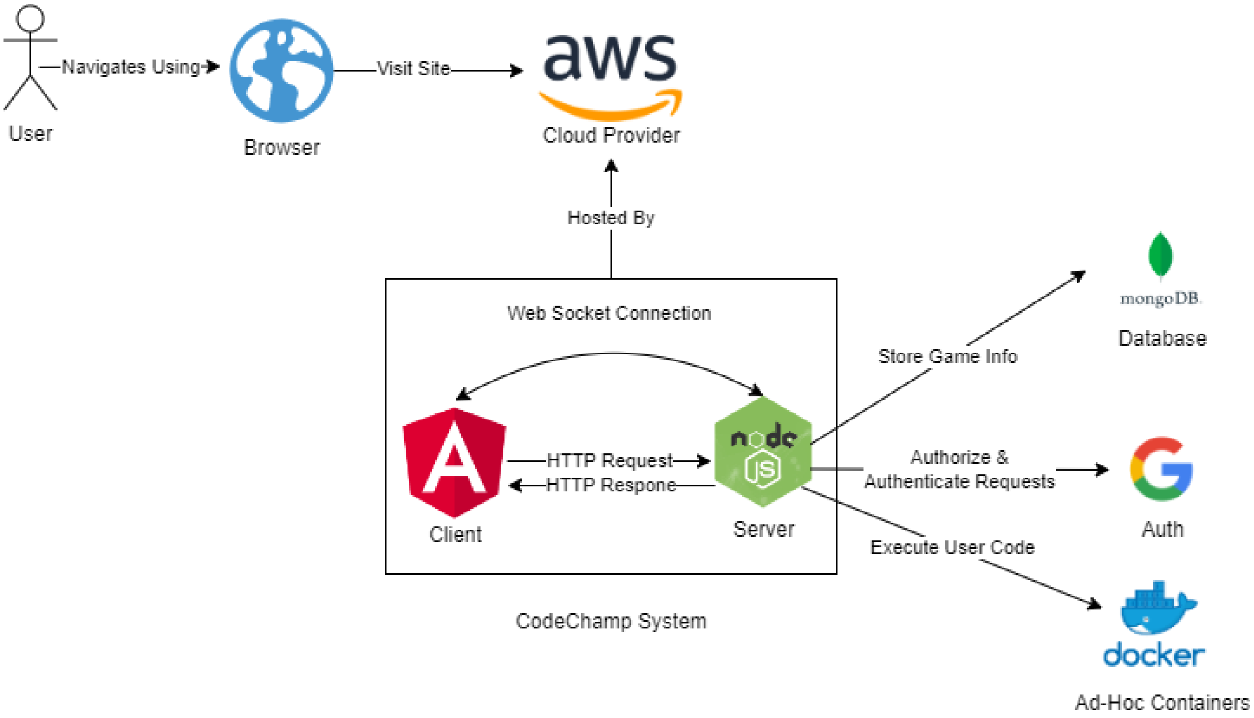


Figure 1: Context Diagram for the CodeChamp System

6 Project Overview

6.1 Normal Behaviour

1. User navigates to site using web browser
2. User logs into the application
3. User creates/joins a game
4. The game starts once the lobby is ready
5. User plays the game by attempting to solve the problem
6. User proceeds to the next round upon successfully completing a round
7. User loses after being unable to complete a round/User wins by winning all the rounds
8. User checks his personal statistics and match history by going to the Home Page and click on the personal profile button

9. User checks the leaderboard by going to the Home Page and click on the leaderboard button

6.2 Undesired Event Handling

Undesired events will be passed onto the front-end to be handled. The front-end will display a dialog mentioning the event in natural language and log the event in details.

6.3 Component Diagram

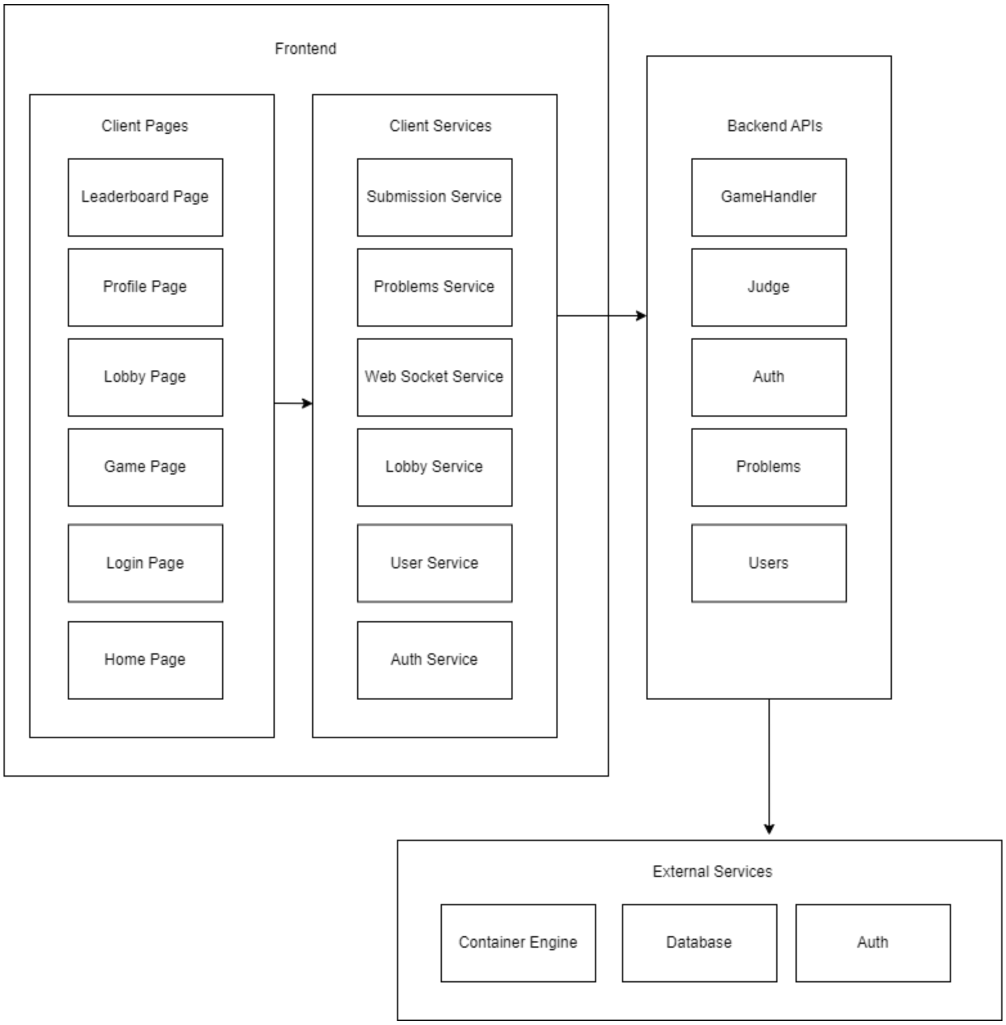


Figure 2: Component Diagram for the CodeChamp System

6.4 Connection Between Requirements and Design

Req.	Decisions
NFR11	Users use username, password, email to create accounts, Google Oauth is used as implementation
FR6, FR14, FR15	Using containerized engine to run code compilation, avoiding effects of compiling malicious code
FR3	Game cuts player count in half in each round to have at most 4 rounds
FR.23	Using unique yet human readable IDs/lobby codes for ease of joining.
FR.13	The system allows developers to create/modify tests through network calls
FR.1	System uses random matchmaking instead of the other option, skill based match making.

Table 1: Requirements and Design Decisions made

7 User Interfaces

The following mock-ups were produced using Figma for the CodeChamp user interface:

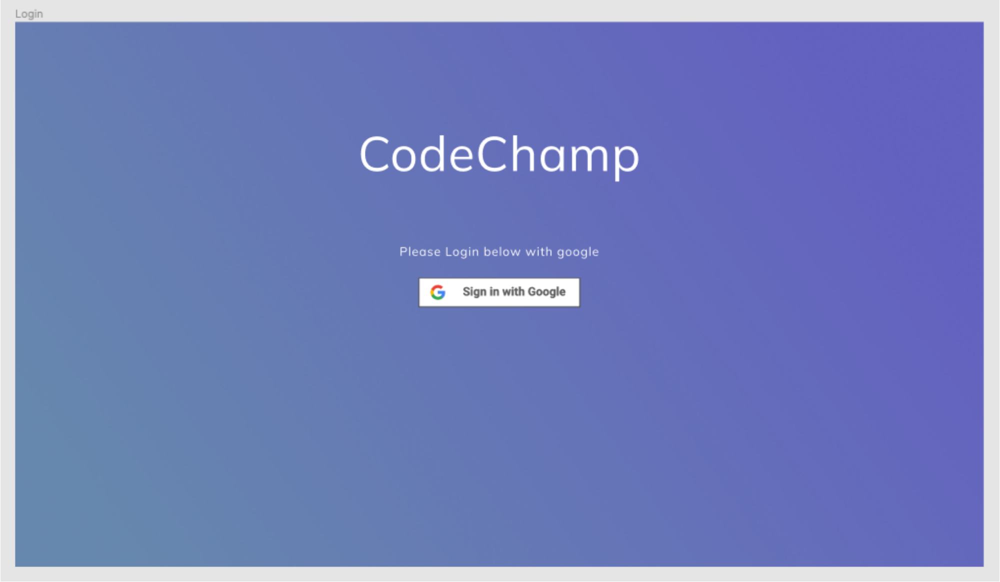


Figure 3: Mock-up for CodeChamp's Login Page

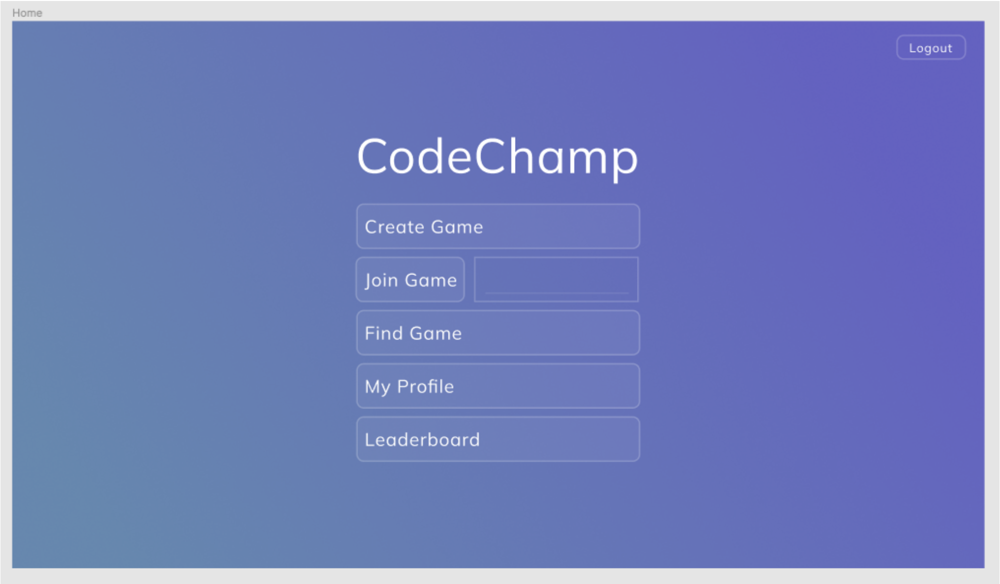


Figure 4: Mock-up for CodeChamp's Home Page

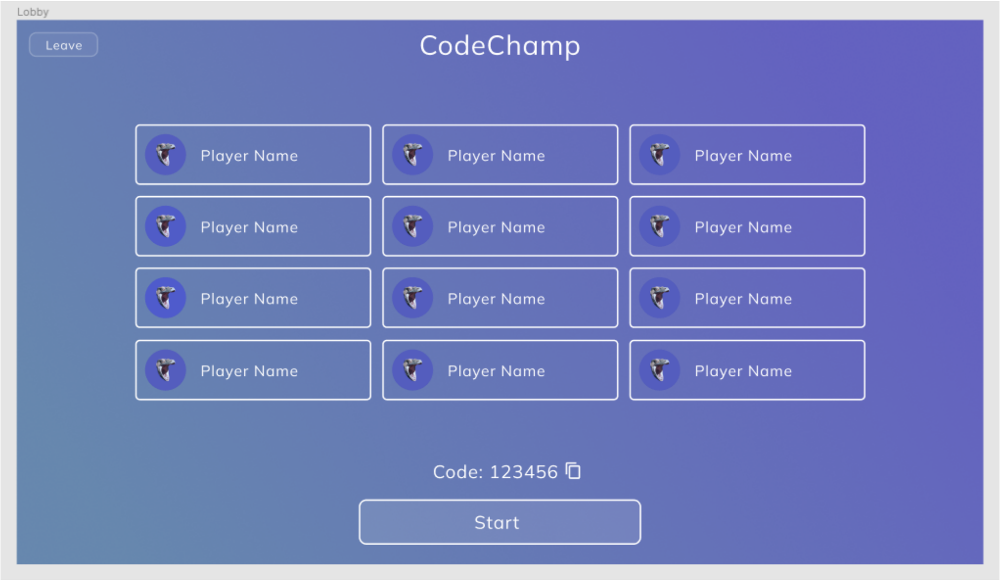
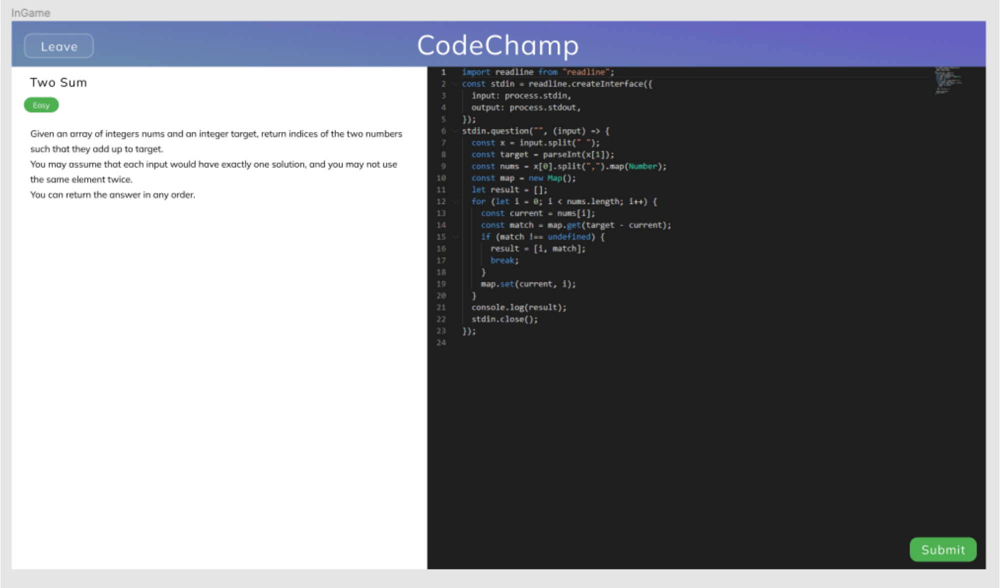


Figure 5: Mock-up for CodeChamp's Lobby Page



Two Sum

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.

```
import readline from "readline";
const stdin = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});
stdin.question("", (input) => {
  const x = input.split(" ");
  const target = parseInt(x[1]);
  const nums = x[0].split(",").map(Number);
  const map = new Map();
  let result = [];
  for (let i = 0; i < nums.length; i++) {
    const current = nums[i];
    const match = map.get(target - current);
    if (match !== undefined) {
      result = [i, match];
      break;
    }
    map.set(current, i);
  }
  console.log(result);
  stdin.close();
});
```

Figure 6: Mock-up for CodeChamp's Game Page

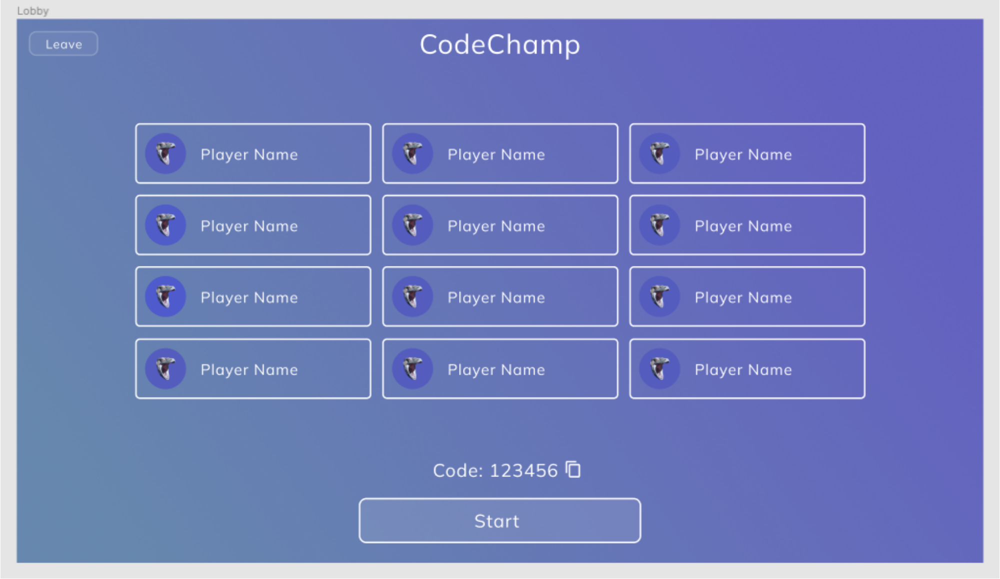
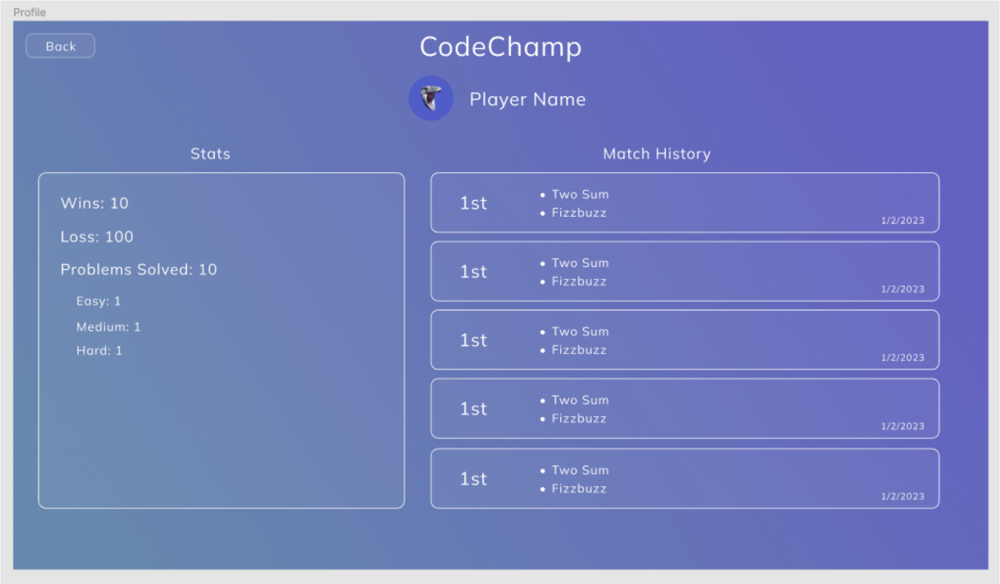


Figure 7: Mock-up for CodeChamp's Leaderboard Page



CodeChamp

Player Name

Stats

Wins: 10

Loss: 100

Problems Solved: 10

Easy: 1

Medium: 1

Hard: 1

Match History

1st	• Two Sum • Fizzbuzz	1/2/2023
1st	• Two Sum • Fizzbuzz	1/2/2023
1st	• Two Sum • Fizzbuzz	1/2/2023
1st	• Two Sum • Fizzbuzz	1/2/2023
1st	• Two Sum • Fizzbuzz	1/2/2023

Figure 8: Mock-up for CodeChamp's Profile Page

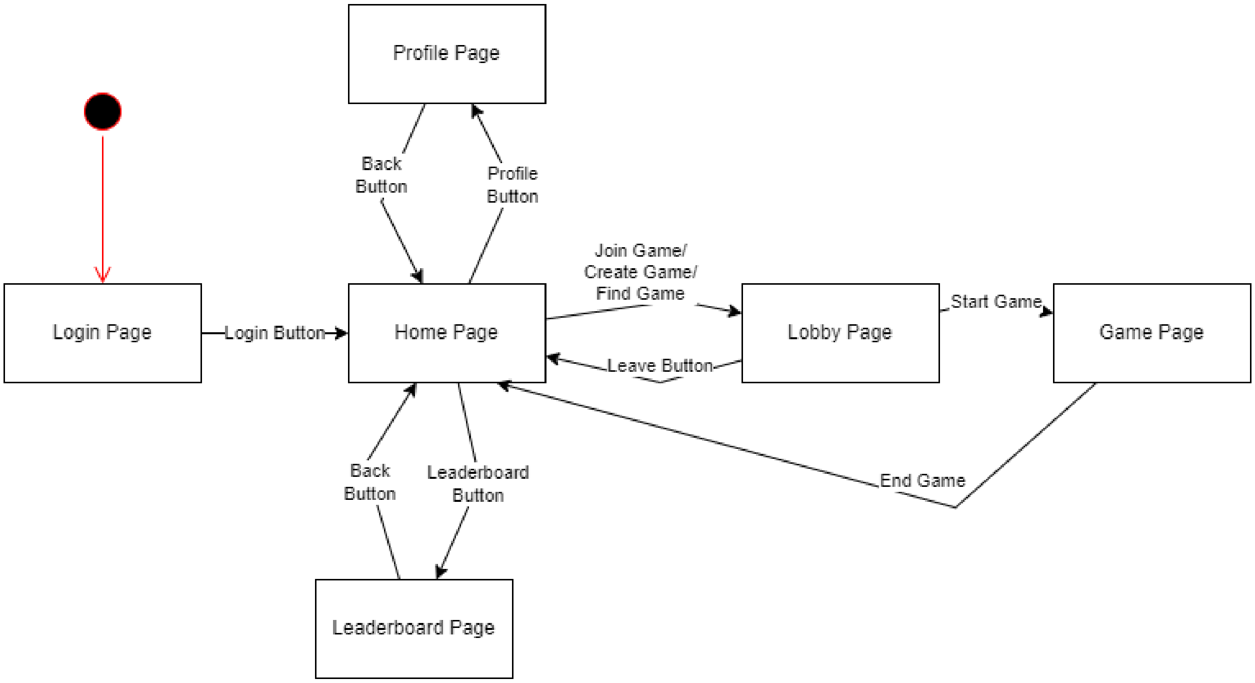


Figure 9: State Machine for front end pages

8 Design of Hardware

N/A

9 Design of Electrical Components

N/A

10 Design of Communication Protocols

Using HTTPS to send requests and retrieve data from the servers and database. For bidirectional messaging between clients and servers, WebSocket technology is used. Technologies are not implemented by the CodeChamp team.

11 Timeline

Module	Completion By Date	Responsible Engineer(s)
ClientT Module	2022-12-30	Anton
GameT Module	2022-12-30	Anton
MatchT Module	2023-02-11	Zhiming
UserT Module	2023-02-11	Zhiming
UserStatsT Module	2023-02-11	Zhiming
ProblemT Module	2022-11-14	Tamas
Difficulty Module	2022-11-14	Tamas
TestCaseT Module	2022-11-14	Youssef
SubmissionT Module	2022-11-14	Youssef
JudgeResultT Module	2022-11-14	Youssef
JudgeVerdict Module	2022-11-14	Youssef
TestCaseVerdictT Module	2022-11-14	Youssef
HomePage Module	2023-1-21	Zhiming, Dipendra
ProfilePage Module	2023-1-21	Zhiming, Tamas
LeaderboardPage Module	2023-1-21	Zhiming
LobbyPage Module	2023-1-10	Dipendra, Anton
GamePage Module	2023-1-10	Dipendra, Anton, Tamas
LoginPage Moudle	2023-1-3	Dipendra, Tamas
SubmissionService Module	2022-11-14	Dipendra
ProblemsService Module	2022-11-14	Dipendra
UserService Module	2023-02-11	Tamas, Zhiming
AuthService Module	2023-1-3	Tamas
LobbyService Module	2023-1-14	Anton, Dipendra
WebSocketService Module	2022-12-30	Anton, Tamas
GameHandler Module	2023-1-10	Anton, Tamas
Judge Module	2023-01-28	Youssef
Auth Module	2023-1-3	Tamas
Problems Module	2022-11-14	Dipendra, Youssef
User Module	2023-02-11	Dipendra

Table 2: Timeline for CodeChamp Module Implementation

A Interface

UI design of CodeChamp

B Mechanical Hardware

None

C Electrical Components

None

D Communication Protocols

HTTPS & WebSockets

E Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Problem Analysis and Design. Please answer the following questions:

1. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

   - Increased game size, this will increase the competition and allow for comparison of a larger pool of competitors through one game.
   - Skill based matchmaking, helps keep the games competitive and ultimately for the competitors to improve as they compete at their skill level.
   - Evaluations on every code submission, currently evaluations are rate limited but evaluating every submission will allow the user to get more and constant feedback on their solutions. Reducing this wait time will improve their focus and attention to the game.

2. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select documented design? (LO_Explores)

   - Explored skill based matchmaking, opted for the randomly matchmaking due its simplicity as calculating their skill level will require complex calculations and matchmaking algorithms. Skill based matchmaking also requires a large player base in order to have meaningful results

- Explored following the MVC model for CodeChamp, this model was not the best as bi-directional messaging is needed for a real-time interactive game. Chose WebSockets to achieve the bi-directional messaging model.
- Explored designing our own authentication module, opted for using Google Auth as it is widely accepted and most clients will already have an account. Also reduces the sign-up process to a single click.