Design Document

Code vs Code

Connor Collins
Emilio Lopez
John Park
Rohan Ramakrishnan
Yifan Wang
Legeng Yi

Table of Contents:

I. Purpose:

People are looking for a friendly and competitive way to improve their coding ability, but don't have the proper medium to compete with their friends. Code vs Code seeks to provide an environment for friends to send challenges to each other to progress their programming skills.

Code vs Code provides an intuitive IDE for users to create unique and fun programming challenges for their friends. Our site allows the user to write, compile, and run code, and see the results of their compilation. The results of the compilation include runtime, completion time, and percentage of test cases passed. We also allow users, via a unique URL, to share their results and challenge their friends to do better. Our web application provides a template for users to create their own coding challenges and test cases from scratch, fostering a community of collaboration and competition.

II. Design Outline

A. Client-Server Outline and Interaction

This project will be a web based application that allows users to practice coding problems online with friends. The application will allow code to be compiled and executed online. This application will use a Client-Server model, with the server implementing the Model-View-Controller design pattern. The server will handle multiple client requests simultaneously. A database will be attached to the server and allow storage of coding problems, solutions, and runtime stats.

1. Client
    a. Here the user will connect with our software via a browser and interact with it.
    b. The client will display a list of coding problems for the user to choose from.
    c. The client will allow the user to search, complete, and create coding problems.
2. Server
    a. The server will manage traffic of the application and the database.
    b. The server will transfer the user profile, attempted solutions from the databases when client requests
    c. The server will retrieve coding solution, problem from the database after

3. Database
    a. The database will contain all the information of the users and their coding problems.
    b. It will also store images for the user's profile picture

B. High Level Overview of the System

We are going to use a client-server architecture for our project. There will be only one server and many clients that will connect to this server. The user will interact with the client, which then will send the interaction's information to the server. The server will retrieve the data needed from the database. Then the server will send back the information to the client and the user will be able to see that data displayed in the client. When a user wants to work on a new problem the client will send a request of the problem to the server, which will then look for it in the database, retrieve the problem and send it to the client, who will display it to the user.

III. Design Issues

A. Functional Issues

Issue 1: How should users login to the application?

Option 1: Facebook interface

Option 2: Own database implementation

Decision: Database

Discussion: We will use own database for users' login because then we won't need to handle learning a new API for facebook. If we have time, we will do the facebook login as well, but the priority is for users to have their own logins.

Issue 2: What languages will we use for the application?

Option 1: Java

Option 2: Node.js

Option 3: Python

Decision: Java

Discussion:We will use Java and Javascript as languages for developing the application. Since we are all familiar with Java, we decided to use the Spring framework for our implementation.

Issue 3: What database solution should we choose for the project?

Option 1: MySQL

Option 2: Oracle

Decision: We are going to use MySQL for the web development

Discussion: MySQL is free, lightweight, and easy to use, yet powerful, secure, and scalable. It is small and fast, so it's ideal for web development.

Issue 4: Do we let the user choose what programming language to use?

Option 1: Yes

Option 2: No

Decision: Yes

Discussion: There are APIs available that can support multiple language compilations and testing we can directly use so we'll provide a list of available programming languages that are supported by the API we choose, such as Java and C, for the user to choose from.

Issue 5: Which web hosting service are we going to use?

Option 1: Our own server

Option 2: AWS

Option 3: CS department servers

Decision: AWS

Discussion:We have experience with deploying on AWS, and using the CS departments servers would be insufficient for the number of users we anticipate having in the future .

Issue 6: Should we build an API wrapper for the Sphere engine API?

Option 1: Yes

Option 2: No

Decision: Yes.

Discussion: We will build an API wrapper for the Sphere engine API because it will be more manageable to interact with the API through a wrapper than directly through the Problems Controller class. In addition, the API wrapper provides a level of reusability we may not have in a more direct interaction.

B. Non-Functional Issues

Issue 1: Do we save the code that user wrote?

Option 1: Yes

Option 2: No

Decision : Yes

Discussion: We will save the code that user wrote if he or she logs out because even though it requires more database work, it will be critical to save the user's work in case of any internet or connection issue.

Issue 2: Should we show how many test cases the user has passed?

Option 1: Yes

Option 2: No

Decision: Yes

Discussion: We will show how many testcases that users passed because that will help the user understand what they are doing right and what they can yet improve.

Issue 3: Should we have a global leader board?

Option 1: Yes

Option 2: No

Decision: No.

Discussion: We will not have a global leader board because it is an additional feature that seems unreasonable to complete in our minimal time frame. We want to focus on friend interactions for our project.

Issue 4: Should the user should be able to create their own coding problems?

Option 1: Yes

Option 2: No

Decision: Yes

Discussion: By creating their own problems the users can be creative and challenge their friends in a fun way.

Issue 5: Can the user see the solution of the challenger after he or she finished a problem?

Option 1: Yes

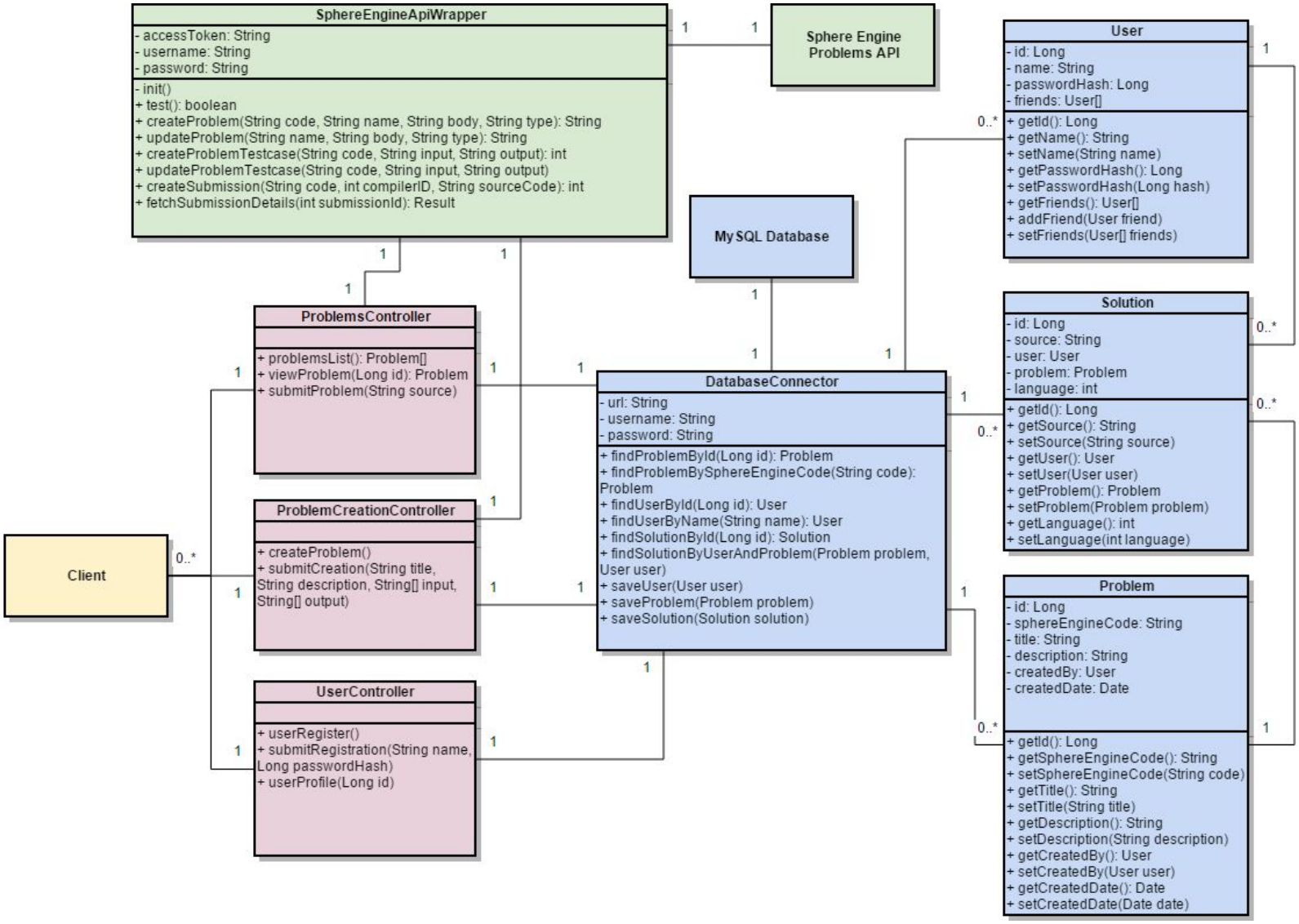
Option 2: No

Decision: Yes

Discussion: SInce the purpose of our application is to educate users, as well as to foster a competitive nature, it is important to show solutions to the user.

IV. Design Details

A. Class Design



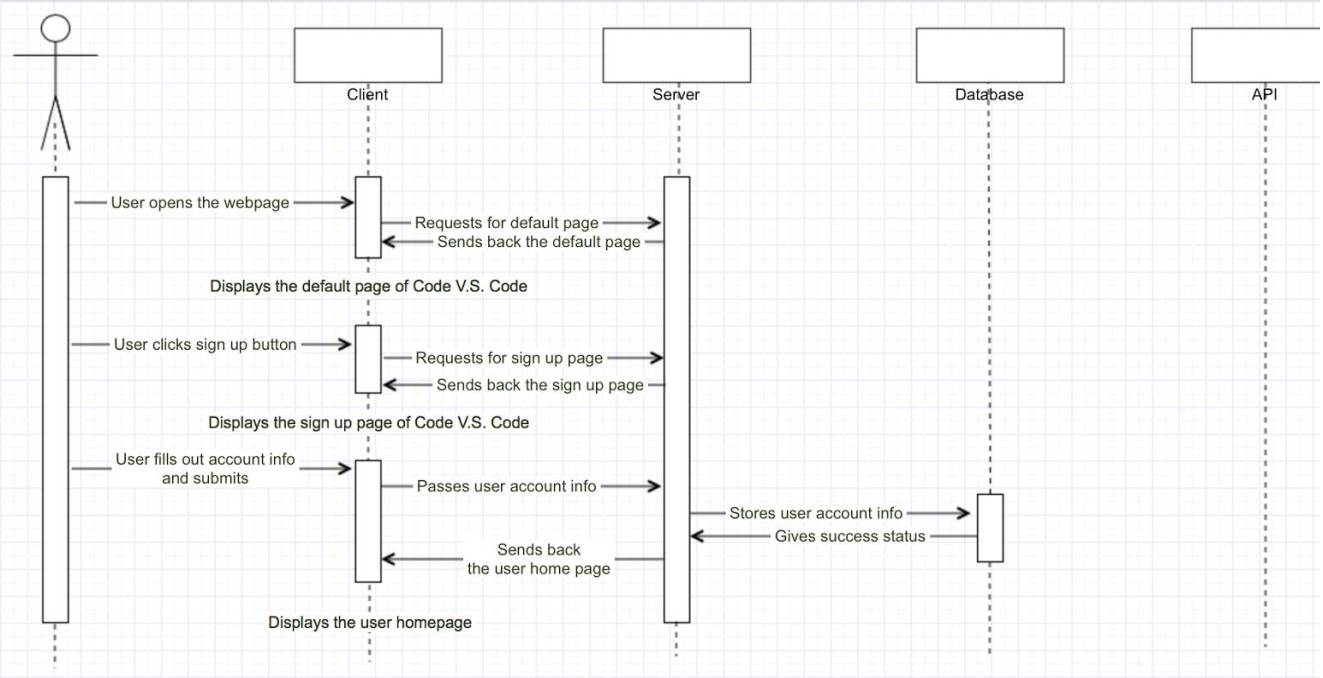
B. Description of classes and methods

- ProblemController
    - Spring MVC controller class for handling requests related to coding problems.
    - It can return a list of all problems, a problem with the IDE, and the results page after a user submits a problem to the client.
- ProblemCreationController
    - Spring MVC controller class for creation of problems and uploading testcases to the Sphere Engine API
    - It can return a view for the user to create a problem, and will upload and store the problem in the database after the user submits the creation.
- UserController
    - Spring MVC controller class for handling requests related to user profiles and user account creation.
    - Contains components to verify user registration information and store the information after validation.
    - Will be able to return a user profile page of the specified user ID.
- SphereEngineApiWrapper
    - A class designed to wrap the Sphere Engine Problems RESTful API in order to simplify and unify accesses to the API.
    - Requests an access token with the username and password upon initialization.
    - Uses that access token and information passed through its methods to access the API and return results.
- DatabaseConnector
    - A class using JDBC designed to simplify and unify the use of the MySQL database.
    - Provides various methods to query and save to the MySQL database.
- User
    - Entity which contains the information of a user, including login information and friends.
    - Contains a list of in-progress solutions to various problems.
- Solution
    - Entity which contains a user's solution to a particular problem.
    - Each solution is mapped to one user and one problem.
    - Designed to be able to keep a user's progress in a problem even after they logout.
- Problem
    - Entity designed to keep a coding problem's data.

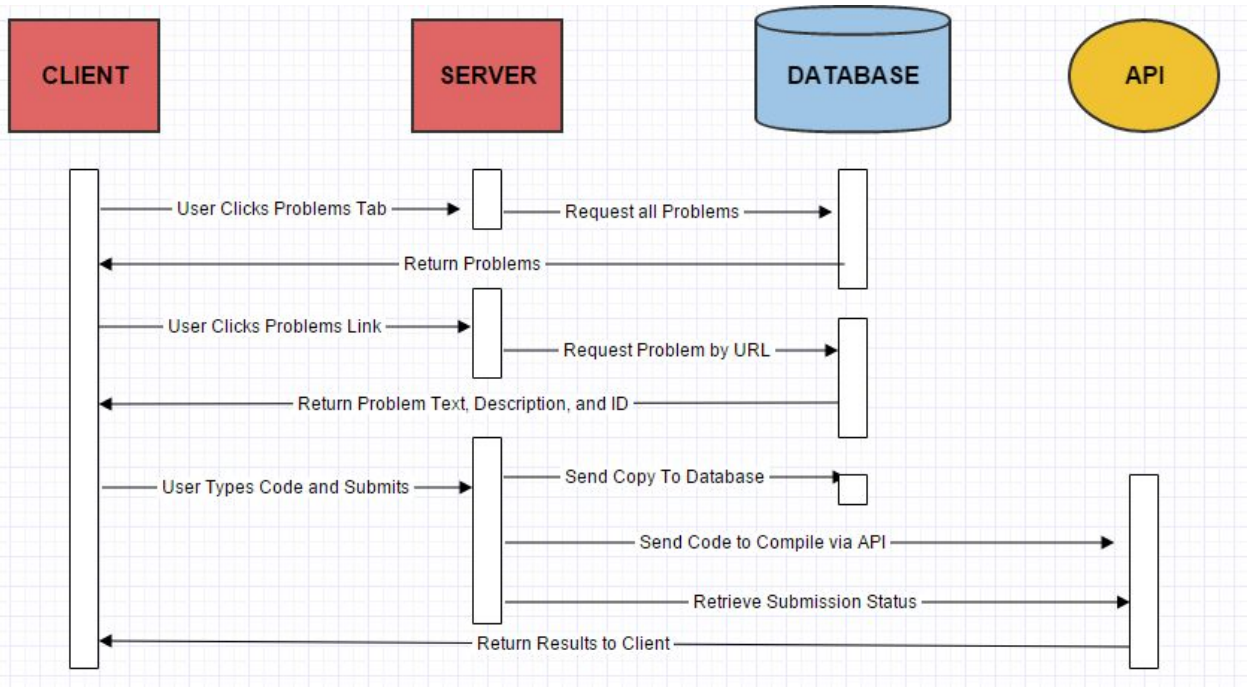
- Contains a unique identifier needed to access the problem on the Sphere Engine API.
- Contains a title and description that will be returned when a user requests a problem.

C. Sequence diagrams

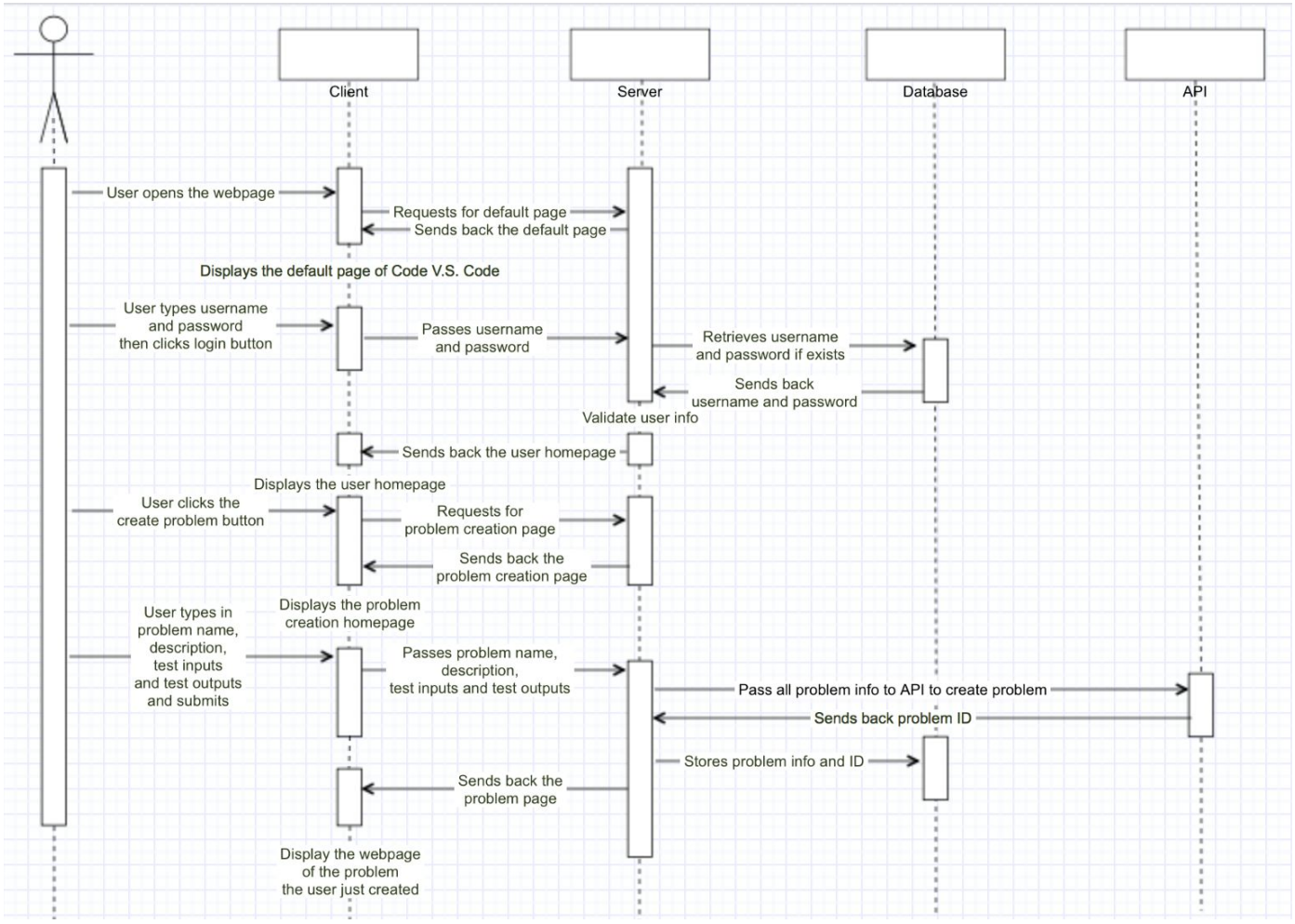
1. Profile Creation



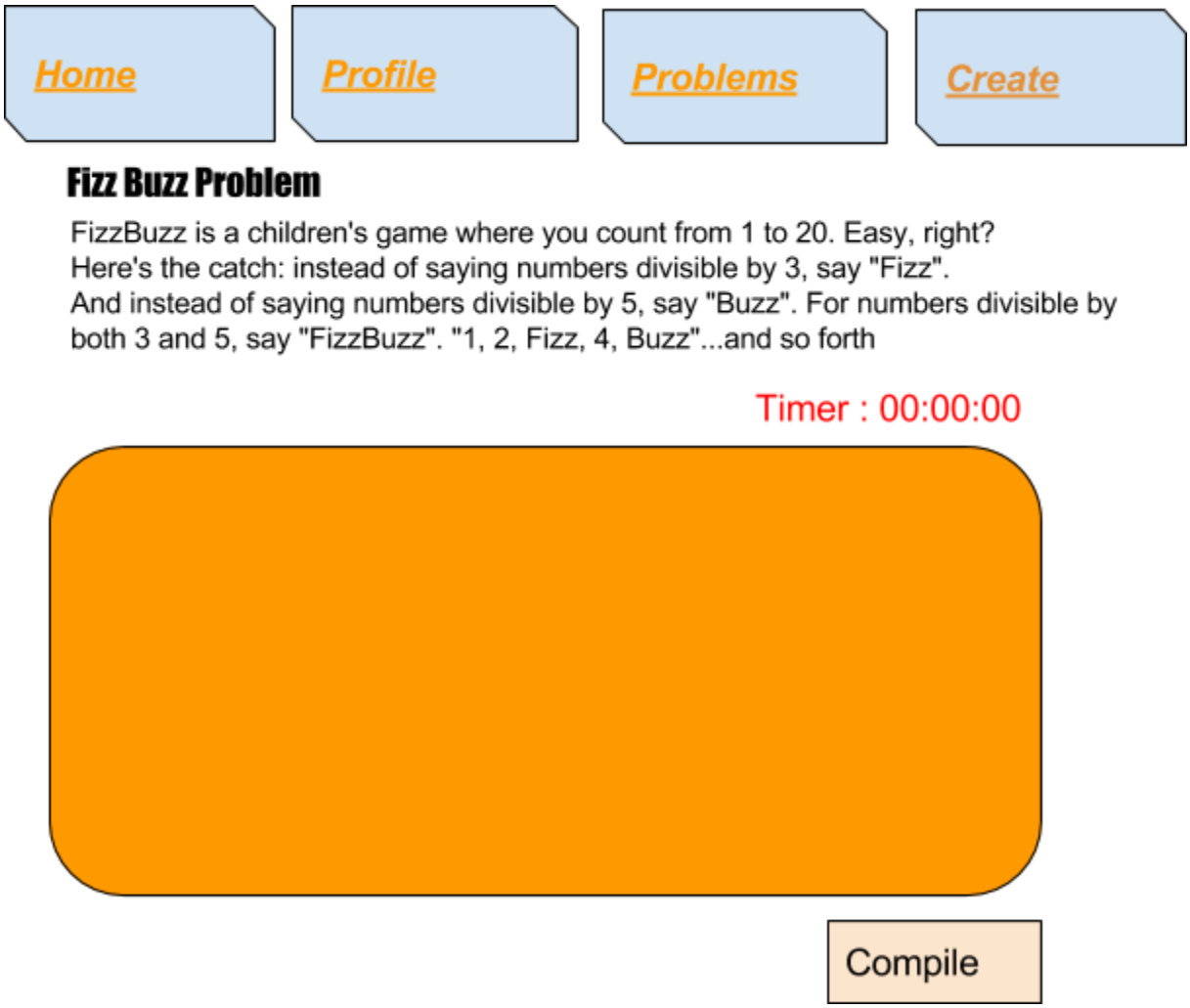
2. Solve problems

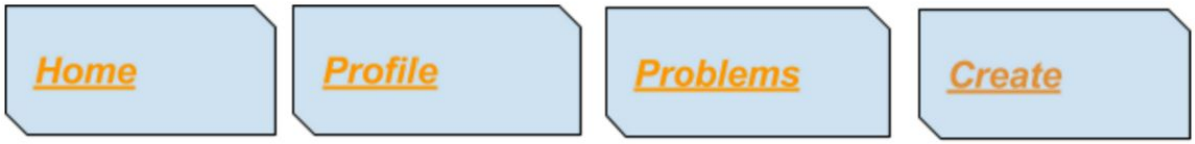


3. Create a problem



D. Mock UI


Home
Profile
Problems
Create
Fizz Buzz Problem
FizzBuzz is a children's game where you count from 1 to 20. Easy, right?
Here's the catch: instead of saying numbers divisible by 3, say "Fizz".
And instead of saying numbers divisible by 5, say "Buzz". For numbers divisible by both 3 and 5, say "FizzBuzz". "1, 2, Fizz, 4, Buzz"...and so forth
Timer : 00:00:00
Compile



Code vs Code

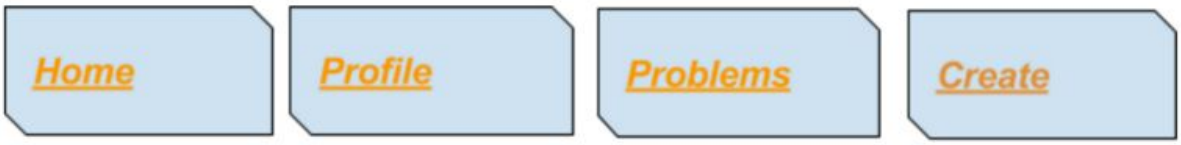
ID _____

Password _____

Login with Facebook

Sign In

Sign Up



Problems

Problem #1

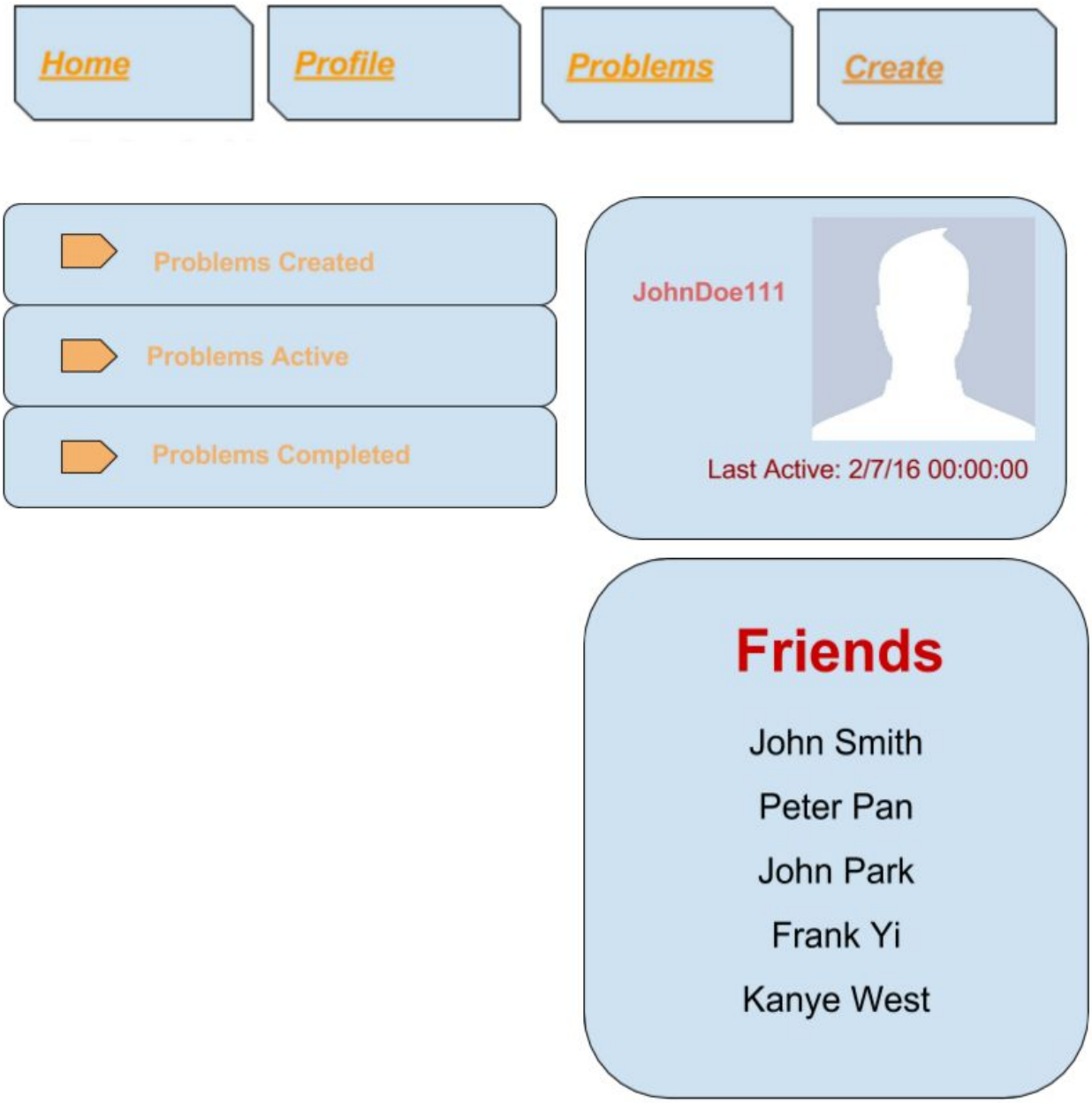
Date created : 2/7/16 Created by: John Doe

Problem #2

Date created : 2/7/16 Created by: John Doe

Problem #3

Date created : 2/7/16 Created by: John Doe


Home
Profile
Problems
Create
Problems Created
Problems Active
Problems Completed
JohnDoe111
Last Active: 2/7/16 00:00:00
Friends
John Smith
Peter Pan
John Park
Frank Yi
Kanye West


Home
Profile
Problems
Create
Stats
Runtime : 00:00:00
Completion Time : 00:00:00
Test Cases
3/3 passed
Test Case 1
Test Case 2
Test Case 3
Challenge Your Friends
URL : codevscode.com/ajsl9301
Share this with your friends