

**SE  
TU**

Ollscoil  
Teicneolaíochta  
an Oirdheiscirt  
  
South East  
Technological  
University

---

# **SETU Code Lab**

## **Design Document**

Diarmuid O'Neill

South East Technological University

12/02/2026

## Table of Contents

Introduction .....	4
Hosting .....	4
Architecture Diagram .....	5
Sequence Diagrams .....	6
Solve a Problem .....	6
Login .....	7
Sign Up .....	7
View Problems .....	8
Create Problem / Test Cases.....	8
Update Problem / Test Cases.....	9
Create / Update Class.....	10
Algorithms.....	11
Test Harness for Running Submitted Code.....	11
Approach to the use of Artificial Intelligence .....	12
Database.....	13
Entity Relationship Diagram .....	13
SQL Statements .....	14
Table Creation.....	14
Login/SignUp .....	16
View Problems.....	16
Solve Problem.....	16
CRUD Problem .....	17
CRUD Test Case.....	17
CRUD Course .....	18
User Interface.....	20
Logo .....	20
High-Level UI Flow .....	20
Login .....	21
Sign Up .....	21
View Problems .....	22
View Problem > Solve Problem .....	22

View Problem > Solve Problem >Submission Alert.....	23
View Problem > Manage Problems .....	23
View Problem > Manage Problems > Create/Update Problem .....	24
View Problem > Manage Classes .....	24
View Problem > Manage Classes > Create/Update Class .....	25
Automated Deployment Script.....	26

# Introduction

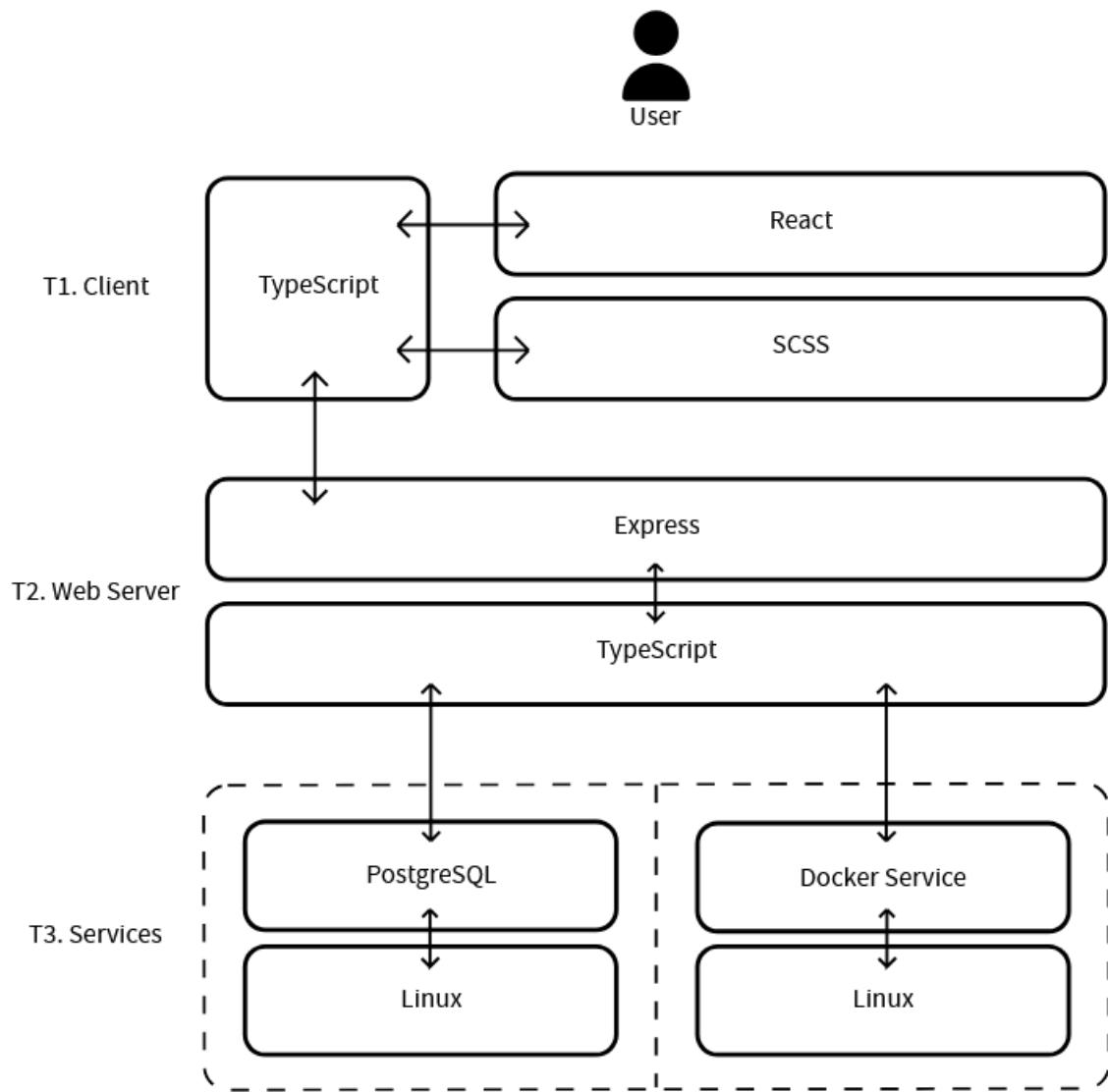
The purpose of this document is to outline the proposed design for SETU Code Lab. It will explore how each part of the project is intended to be implemented. It includes sections on hosting, sequence diagrams, important algorithms, database design and the user interface.

## Hosting

The system will be hosted using DigitalOcean's Droplet service. This is a virtual private server (VPS). The \$12 per month regular option provides 2GBs of RAM, 1 CPU, 50GB of SSD storage and 1TB of bandwidth which should be enough for SETU Code Lab with some optimization. The chosen operating system for this server is Ubuntu 24.04 as it is very stable, provides excellent Docker support and is familiar to the developer.

For security purposes an SSH key will be generated and used to access the server console. This is faster and more secure than the password option that is offered by DigitalOcean. The Github repository containing the project will be cloned onto the server and the needed dependencies will be installed such as Node.js, Node Package Manager (NPM) and Nginx for serving the frontend.

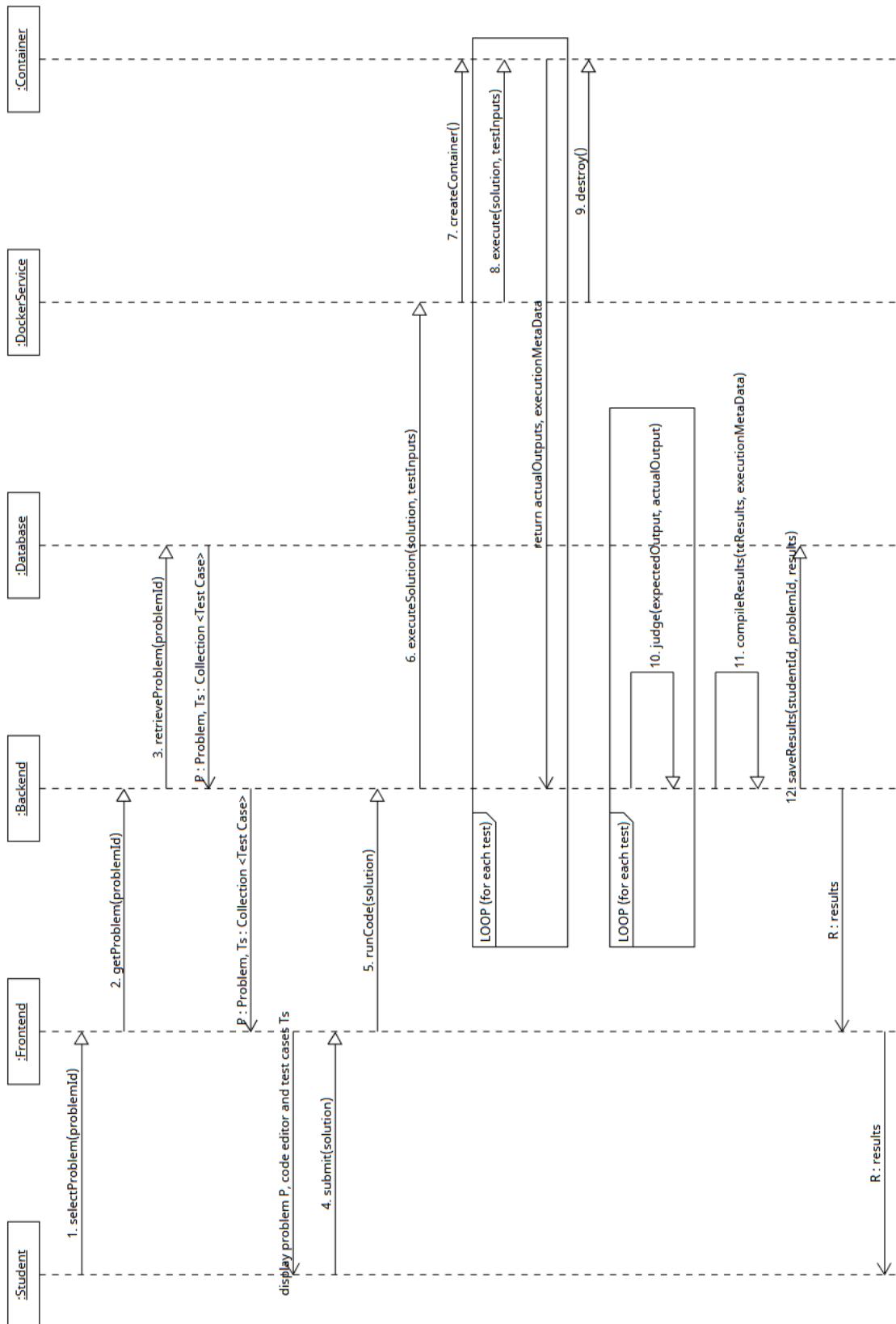
# Architecture Diagram



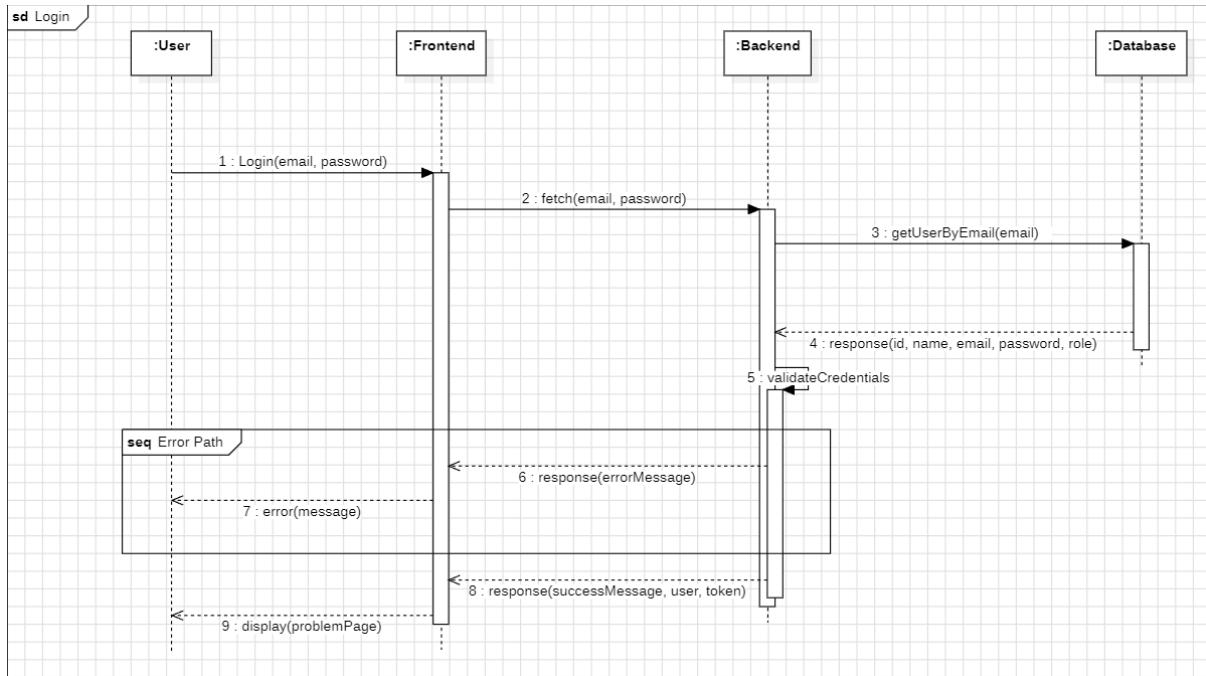
SETU Code Lab uses a three-tier architecture. Tier one (T1) is the client and is built using React, Typescript and SCSS. This is where the user interface resides. Tier two (T2) is the web server and is implemented using Node.js (this is a JavaScript runtime which allows TypeScript to run on a web server) and Express.js which is a Node.js framework used to simplify the routing process. This is where the application logic resides. Tier three (T3) is the services layer and consists of a PostgreSQL database which stores all the application data and the Docker service which is used to create temporary containers in which user submitted code runs in isolation.

# Sequence Diagrams

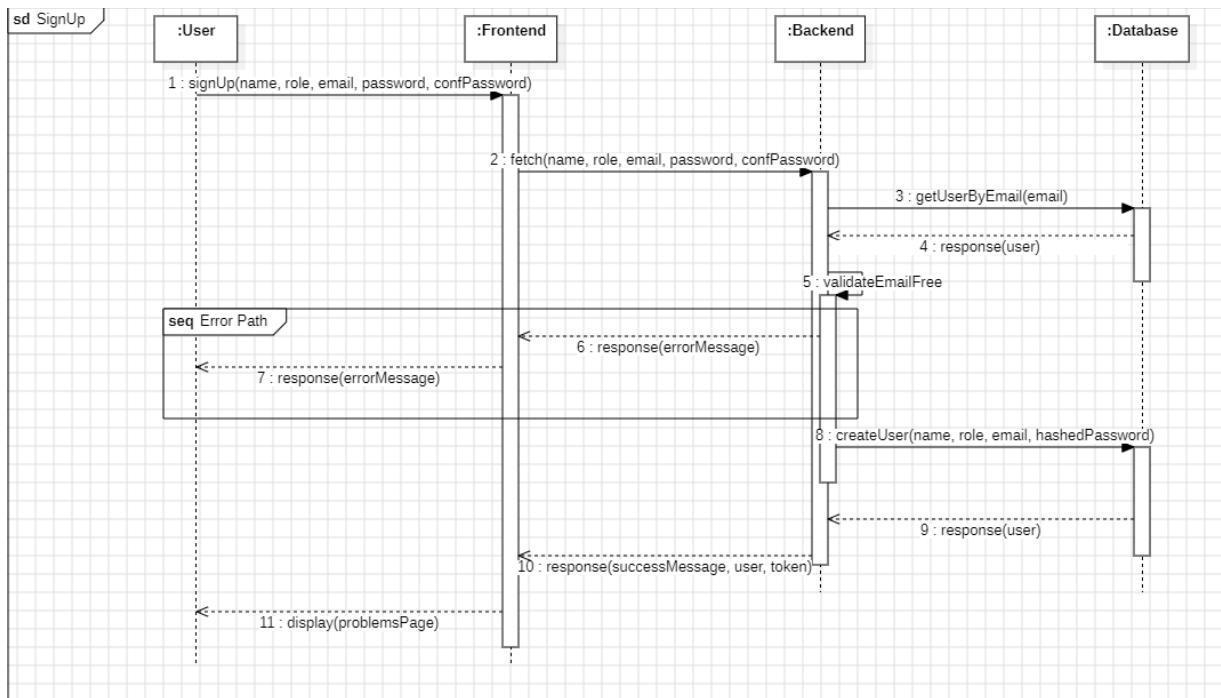
## Solve a Problem



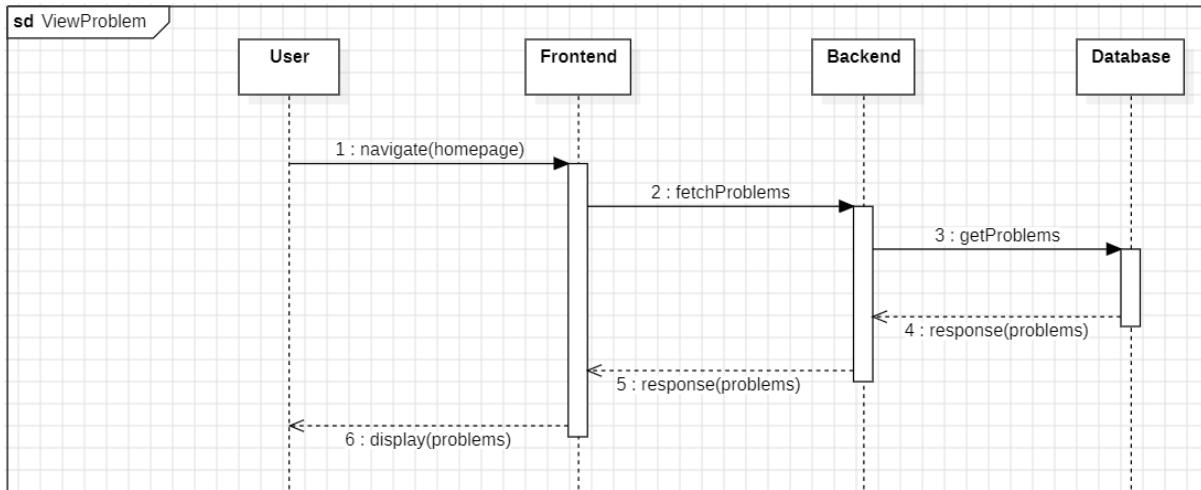
## Login



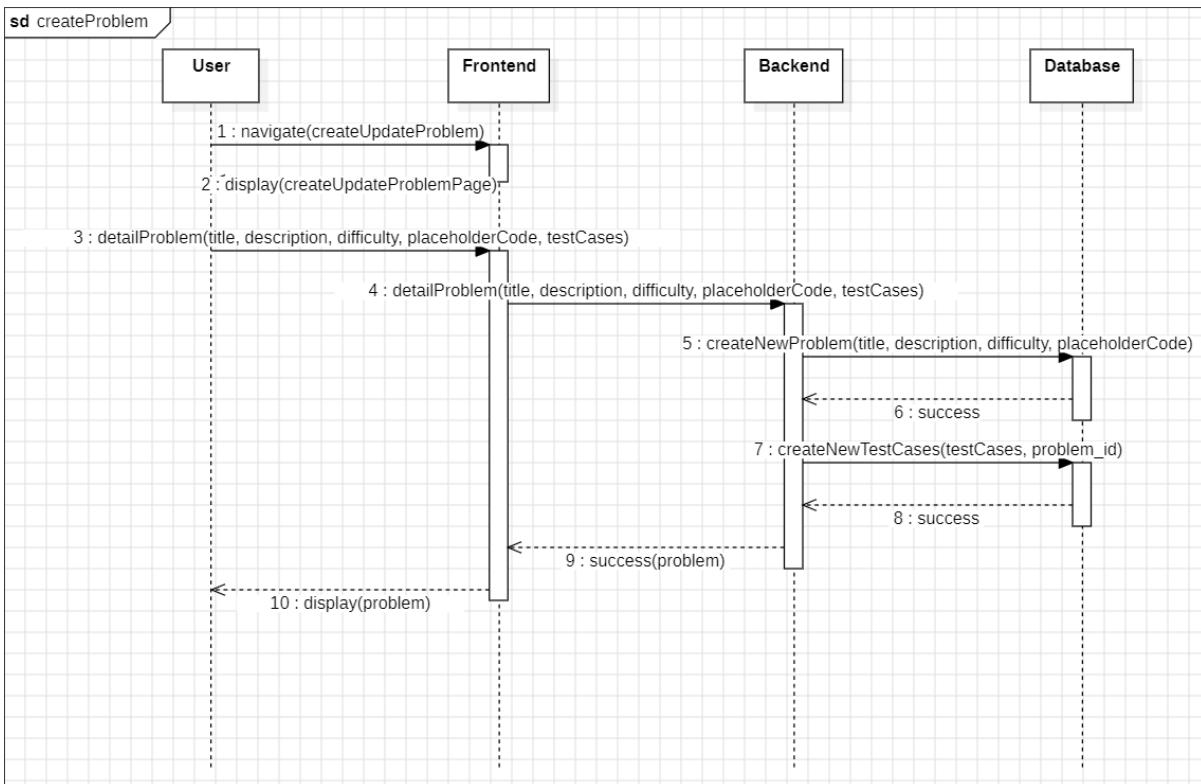
## Sign Up



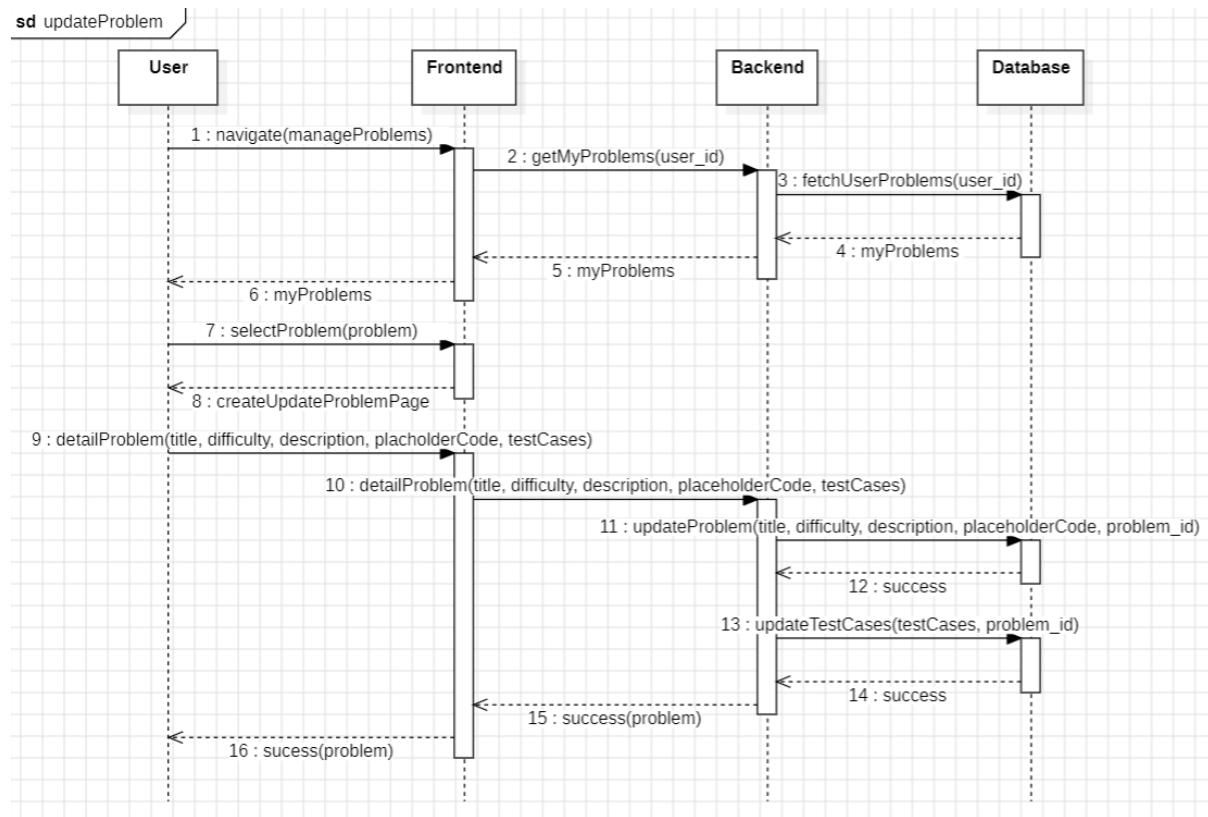
## View Problems



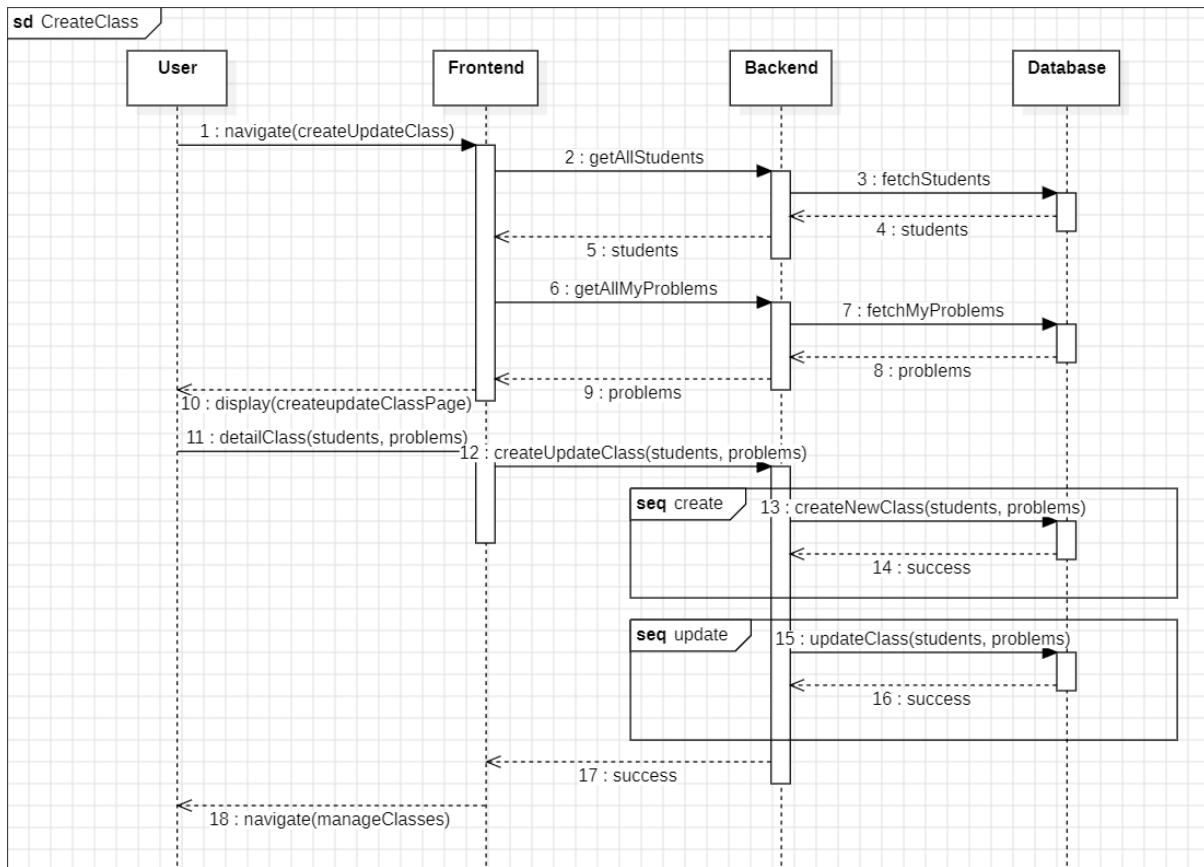
## Create Problem / Test Cases



## Update Problem / Test Cases



## Create / Update Class



# Algorithms

## Test Harness for Running Submitted Code

To run student submitted functions they must be injected into a static string harness which contains a main class and the needed dependency imports to map sample inputs to function parameters. This allows any inputted java function to run in isolation in a docker container with sample inputs from associated test cases. This harness is mostly the same for all submissions; however, some parts need to be dynamic to allow functions with parameters of different primitive and complex types and functions with different numbers of parameters to run. The harness looks like this:

```
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.*;  
  
public class Main {  
    static final ObjectMapper mapper = new ObjectMapper();  
    ${code}  
    static class Input {  
        ${inputFields}  
    }  
    public static void main(String[] args) {  
        try {  
            Input input = mapper.readValue(args[0], Input.class);  
            ${functionCallLine}  
            System.out.println(result);  
        } catch (Exception e) {  
            System.out.println("ERROR:" + e.getMessage());  
        }  
    }  
}  
${code} is the student's submitted code, ${inputFields} are all the input parameter types and names, and ${functionCallLine} contains the final return type, the function itself and the input parameters. "Input input = mapper.readValue(args[0], Input.class);" maps sample inputs from problem test cases, these are passed as arguments when the main function is called. Below are the commands that run inside the docker container which allow the java code to execute with test case inputs (${processedInput} is the test case inputs).
```

```
"sh", "-c", `  
cat << 'EOF' > Main.java  
${preprocessedCode}  
EOF  
javac Main.java  
java -cp "./app/*" Main '${processedInput}'`
```

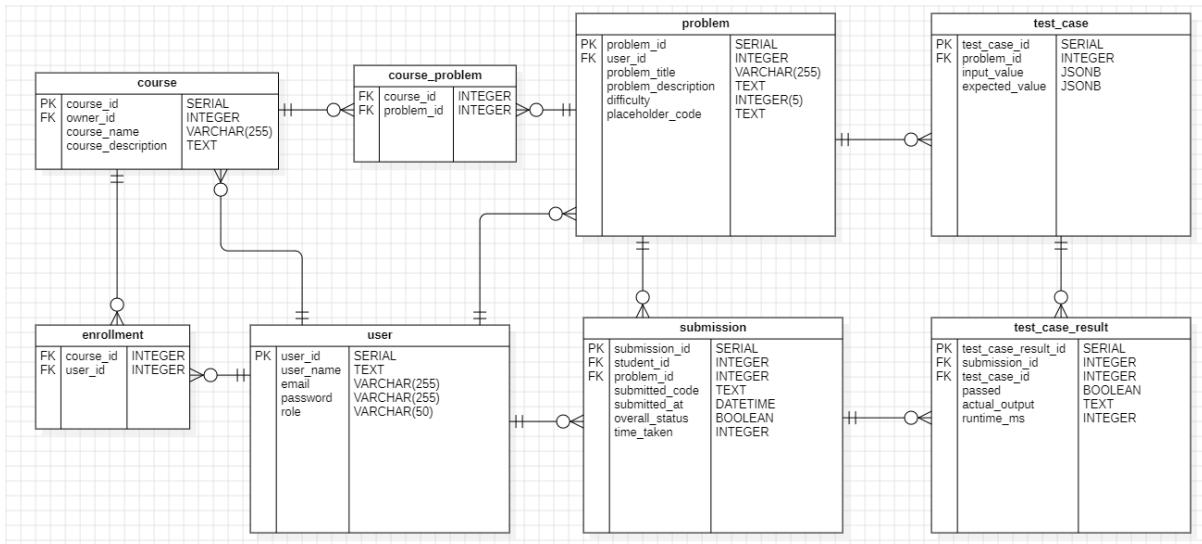
## Approach to the use of Artificial Intelligence

It is likely that some students using SETU Code Lab will attempt to use generative AI to assist them with completing problems. This is not how the platform is intended to be used as it hinders students' ability to learn coding concepts effectively. While it is impossible to prevent the use of generative AI entirely, the platform aims to increase the work factor to discourage most users from using it. A file called antiCheat.ts has been created which disables copy/paste functionality and can detect when a user switches tabs. This is only active on the “solve a problem” screen and can be bypassed in certain situations. This only acts as a deterrent to the use of AI assistance.

```
export const useAntiCheat = () => {
    const [shouldAutoSubmit, setShouldAutoSubmit] = useState(false);
    useEffect(() => {
        const prevent = (e: Event) => e.preventDefault();
        const preventContext = (e: MouseEvent) =>
e.preventDefault();
        const handleKeyDown = (e: KeyboardEvent) => {
            if ((e.ctrlKey || e.metaKey) && ["c", "v", "x",
"a"].includes(e.key.toLowerCase())) {
                e.preventDefault()
                alert("Warning: Copy/Paste is disabled");
            }
        };
        const handleVisibility = () => {
            if (document.hidden) {
                setShouldAutoSubmit(true);
                alert("You left the tab. Your work has been
automatically submitted.");
            }
        }
        document.addEventListener("visibilitychange",
handleVisibility);
        document.addEventListener("copy", prevent);
        document.addEventListener("paste", prevent);
        document.addEventListener("cut", prevent);
        document.addEventListener("contextmenu", preventContext);
        document.addEventListener("keydown", handleKeyDown);
        return () => {
            document.removeEventListener("visibilitychange",
handleVisibility);
            document.removeEventListener("copy", prevent);
            document.removeEventListener("paste", prevent);
            document.removeEventListener("cut", prevent);
            ...
        }
    });
}
```

# Database

## Entity Relationship Diagram



# SQL Statements

## Table Creation

### Problem Table

```
CREATE TABLE problem (
    problem_id      SERIAL PRIMARY KEY,
    user_id         INTEGER REFERENCES users(user_id),
    problem_title   VARCHAR(255),
    problem_description TEXT,
    difficulty      INTEGER CHECK (difficulty BETWEEN 1 AND 5),
    placeholder_code TEXT
);
```

### Test\_case Table

```
CREATE TABLE test_case (
    test_case_id    SERIAL PRIMARY KEY,
    problem_id      INT REFERENCES problem(problem_id) ON DELETE CASCADE,
    input_value     JSONB NOT NULL,
    expected_value  JSONB NOT NULL
);
```

### User Table

```
CREATE TABLE users (
    user_id         SERIAL PRIMARY KEY,
    user_name       TEXT NOT NULL,
    email           VARCHAR(255) UNIQUE NOT NULL,
    password        VARCHAR(255) NOT NULL,
    role            VARCHAR(50) NOT NULL CHECK (role IN ('student','lecturer'))
);
```

### Submission Table

```
CREATE TABLE submission (
    submission_id   SERIAL PRIMARY KEY,
    user_id         INT REFERENCES users(user_id),
    problem_id      INT REFERENCES problem(problem_id) ON DELETE CASCADE,
    submitted_code  TEXT,
    submitted_at    TIMESTAMP DEFAULT now(),
    overall_status  BOOLEAN,
    time_taken      INT
);
```

### **Test\_case\_result Table**

```
CREATE TABLE test_case_result (
    test_case_result_id SERIAL PRIMARY KEY,
    submission_id INT REFERENCES submission(submission_id) ON
DELETE CASCADE,
    test_case_id INT REFERENCES test_case(test_case_id) ON
DELETE CASCADE,
    passed BOOLEAN,
    actual_output TEXT,
    runtime_ms INTEGER
);
```

### **Course Table**

```
CREATE TABLE course (
    course_id SERIAL PRIMARY KEY,
    owner_id INT REFERENCES users(user_id) ON DELETE
CASCADE,
    course_title VARCHAR(255),
    course_description TEXT
);
```

### **Enrollment Table**

```
CREATE TABLE enrollment (
    course_id INT REFERENCES course(course_id) ON DELETE
CASCADE,
    user_id INT REFERENCES users(user_id) ON DELETE
CASCADE,
    PRIMARY KEY (course_id, user_id)
);
```

### **Course\_problem Table**

```
CREATE TABLE course_problem (
    course_id INT REFERENCES course(course_id) ON DELETE
CASCADE,
    problem_id INT REFERENCES problem(problem_id) ON
DELETE CASCADE,
    PRIMARY KEY (course_id, problem_id)
);
```

## Login/SignUp

### **createUser SQL Statement**

```
INSERT INTO users (user_name, role, email, password)
VALUES ($1, $2, $3, $4)
RETURNING *
```

### **getUserByEmail SQL Statement**

```
SELECT user_id
AS id, user_name
AS name, email, password, role
FROM users
WHERE email = $1
```

## View Problems

### **fetchProblems SQL Statement**

```
SELECT problem.*, users.user_name
FROM problem
AS problem
JOIN users AS users
ON problem.user_id = users.user_id
```

## Solve Problem

### **fetchTestCases SQL Statement**

```
SELECT *
FROM test_case
WHERE problem_id=$1
```

### **createSubmission SQL Statement**

```
INSERT INTO submission (user_id, problem_id, submitted_code,
overall_status, time_taken)
VALUES ($1, $2, $3, $4, $5)
RETURNING *
```

### **createTestCaseResult SQL Statement**

```
INSERT INTO test_case_result (submission_id, test_case_id, passed,
actual_output, runtime_ms)
VALUES ($1, $2, $3, $4, $5)
```

## CRUD Problem

### **insertProblem SQL Statement**

```
INSERT INTO problem (user_Id, problem_title, problem_description,
difficulty, placeholder_code)
    VALUES ($1, $2, $3, $4, $5)
    RETURNING *
```

### **updateProblem SQL Statement**

```
UPDATE problem
    SET problem_title=$1, problem_description=$2, difficulty=$3,
placeholder_code=$4
    WHERE problem_id=$5 RETURNING *
```

### **deleteProblem SQL Statement**

```
DELETE FROM problem WHERE problem_id=$1
    RETURNING *
```

## CRUD Test Case

### **createTestCase SQL Statement**

```
INSERT INTO test_case (problem_id, input_value, expected_value)
    VALUES ($1, $2::json, $3::json)
    RETURNING *
```

### **updateTestCase SQL Statement**

```
UPDATE test_case
    SET input_value=$1, expected_value=$2
    WHERE test_case_id=$3
    RETURNING *
```

### **deleteTestCase SQL Statement**

```
DELETE FROM test_case
    WHERE test_case_id=$1
    RETURNING *
```

## CRUD Course

### **fetchAllStudents SQL Statement**

```
SELECT user_id AS student_id, user_name AS student_name
      FROM users
     WHERE role='student'
```

### **AddUserToCourse SQL Statement**

```
INSERT INTO enrollment (course_id, user_id)
VALUES (1, $1)
RETURNING *
```

### **fetchCourseByUserId SQL Statement**

```
SELECT c.*
      FROM course c
     JOIN enrollment e ON c.course_id = e.course_id
    WHERE e.user_id = $1
```

### **fetchCreatedCourseByUserId**

```
SELECT * FROM course WHERE owner_id=$1
```

### **insertCourse SQL Statement**

```
INSERT INTO course (owner_id, course_title, course_description)
VALUES ($1, $2, $3)
RETURNING *
```

### **insertCourseProblem SQL Statement**

```
INSERT INTO course_problem (course_id, problem_id)
VALUES ($1, $2) RETURNING *
```

### **insertEnrollment SQL Statement**

```
INSERT INTO enrollment (course_id, user_id) VALUES ($1, $2)
RETURNING *
```

### **updateCourseDetails SQL Statement**

```
UPDATE course SET course_title=$1, course_description=$2 WHERE
course_id=$3 RETURNING *
```

**deleteCourseProblems SQL Statement**

```
DELETE FROM course_problem WHERE course_id=$1
```

**deleteEnrollmentsByCourseId SQL Statement**

```
DELETE FROM enrollment WHERE course_id=$1
```

**fetchProblemIdsFromCourse SQL Statement**

```
SELECT problem_id FROM course_problem WHERE course_id = $1
```

**fetchStudentIdsFromCourse SQL Statement**

```
SELECT user_id FROM enrollment WHERE course_id = $1
```

# User Interface

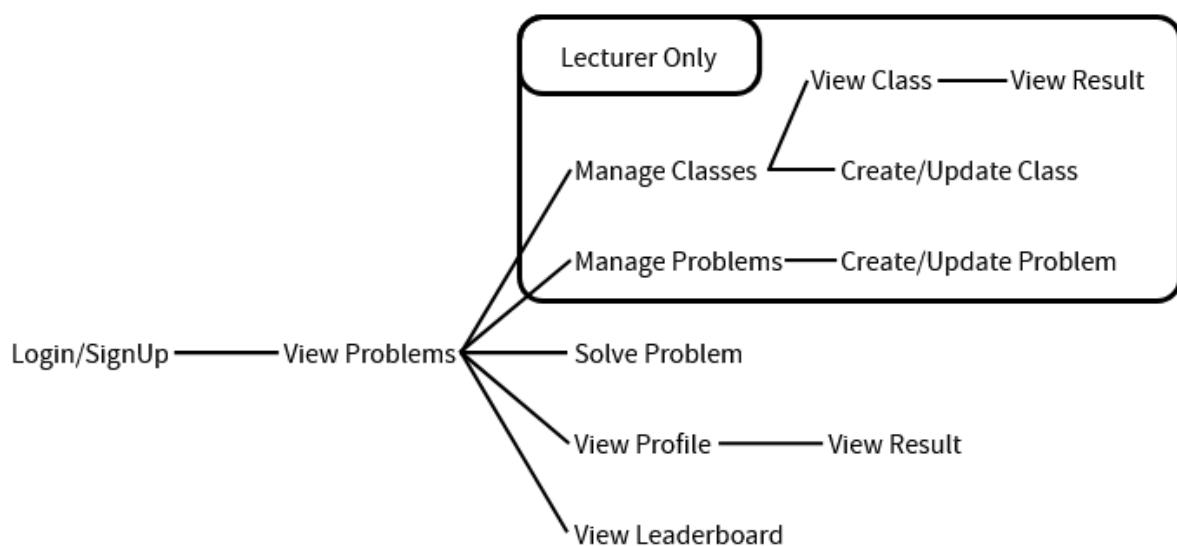
## Logo

This is the logo and favicon for SETU Code Lab. It has been designed in the shape of the letter C and is intended to be simple and recognizable. The light grey and bright red colours have been chosen as they contrast well with the dark background chosen for the rest of the platform.



## High-Level UI Flow

The diagram below shows how to navigate to each screen on the platform. Additional UI elements such as pop-ups and drop-down menus are not shown here, just the main screens. Items in the lecturer only box are only accessible to users with the lecturer role.



## Login



SETU Code Lab

email

password

**Log in**

[Sign up](#) if you don't have an account already

## Sign Up



SETU Code Lab

name

Student

Lecturer

email

password

confirm password

**Sign up**

[Log in](#) if you already have an account

## View Problems

The screenshot shows the 'SETU Code Lab' interface. At the top, there's a navigation bar with links for 'Leaderboard', 'Problems', and 'Profile'. On the right, there's a 'Log Out' button and a user profile icon. Below the navigation, there are two buttons: 'Manage Problems' and 'Manage Classes'. A search bar with placeholder text 'Search Questions...' and a dropdown arrow is positioned next to it. To the right of the search bar is a 'Difficulty' filter with a star rating scale from one to five stars. The main content area displays a list of 11 problems, each with a title, author, and difficulty rating:

Problem Title	Author	Difficulty
1. Two Sum	Dr. John Doe	★☆☆☆☆
✓ 2. Wildcard Matching	Dr. Jane Doe	★★★★★
✓ 3. Add Two Numbers	Dr. Jane Doe	★★★☆☆
4. Median of Two Sorted Arrays	Dr. Jane Doe	★★★☆☆
5. Remove Element	Dr. Jane Doe	★★★☆☆
6. Two Sum	Dr. John Doe	★☆☆☆☆
7. Two Sum	Dr. John Doe	★☆☆☆☆
8. Two Sum	Dr. John Doe	★☆☆☆☆
9. Two Sum	Dr. John Doe	★☆☆☆☆
10. Two Sum	Dr. John Doe	★☆☆☆☆
11. Two Sum	Dr. John Doe	★☆☆☆☆

## View Problem > Solve Problem

This screenshot shows the details for the '1. Two Sum' problem. The problem title is '1. Two Sum | Dr. John Doe' with a difficulty rating of ★☆☆☆☆. The problem description states: 'Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.' It notes that each input would have exactly one solution and no element can be used twice. The example provided is: Input: nums = [2,7,11,15], target = 9. The output is [0,1] because nums[0] + nums[1] == 9. Constraints are listed as: 2 <= nums.length <= 104, -109 <= nums[i] <= 109, and -109 <= target <= 109. It also mentions that only one valid answer exists. A follow-up question asks if an algorithm with O(n^2) time complexity can be improved.

**Code editor**

```
1 class Solution {  
2     public int[] twoSum(int[] nums, int target) {  
3         // Implementation  
4     }  
5 }
```

**Test cases**

```
1 [2,7,11,15]  
2 9  
3 [3,2,4]  
4 6  
5 [3,3]  
6 6
```

## View Problem > Solve Problem >Submission Alert

The screenshot shows a dark-themed modal window titled "Submission". Inside, the status is listed as "Pass" and the result is "100%". The time taken was "00:35:22" and the execution time was "00:00:32". A "Dismiss" button is at the bottom left. The background shows a problem statement for "Two Sum" by Dr. John Doe, detailing the problem requirements, examples, and constraints.

1. Two Sum | Dr. John Doe

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.

Example 1:

Input: nums = [2,7,11,15], target = 9  
Output: [0,1]  
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:

Input: nums = [3,2,4], target = 6  
Output: [1,2]

Example 3:

Input: nums = [3,3], target = 6  
Output: [0,1]

Constraints:

$2 \leq \text{nums.length} \leq 104$   
 $-109 \leq \text{nums}[i] \leq 109$   
 $-109 \leq \text{target} \leq 109$   
Only one valid answer exists.

Follow-up: Can you come up with an algorithm that is less than  $O(n^2)$  time complexity?

Status: Pass

Result: 100%

Time Taken: 00:35:22

Execution Time: 00:00:32

Dismiss

## View Problem > Manage Problems

The screenshot shows a dark-themed list of problems under the "Manage Problems" section. Each problem has a title, a difficulty level (e.g., Y2 SWD), and a star rating. To the right of each row are edit and delete icons.

Problem Title	Difficulty	Star Rating	Action
1. Two Sum	Y2 SWD	5 stars	Edit Delete
2. Palindrome Number	Y2 SWD	5 stars	Edit Delete
3. Remove Element	Y3 GD	5 stars	Edit Delete
4. Median of Two Sorted Arrays	All Students	5 stars	Edit Delete

+ Create New Problem

## View Problem > Manage Problems > Create/Update Problem

The screenshot shows the 'Create New Problem' section of the SETU Code Lab interface. On the left, there are navigation links for 'Manage Problems' and 'Manage Classes'. The main area has a title 'Create New Problem' and fields for 'Title' (eg. Palindrome Number...) and 'Difficulty' (eg. 0-5...). Below these are sections for 'Description' (with a note about markdown support) and 'Example 1' (showing input 'x = 121' and output 'true'). To the right, there's a 'Placeholder Code' box containing a template code snippet. At the bottom right is a 'Next' button.

## View Problem > Manage Classes

The screenshot shows the 'Manage Classes' section of the SETU Code Lab interface. On the left, there are navigation links for 'Manage Problems' and 'Manage Classes'. The main area lists four classes: '1. Y3 Software Development', '2. Y2 Games Development', '3. Y1 Common', and '4. All Students', each with edit and delete icons. At the bottom left is a 'Create New Class' button.

## View Problem > Manage Classes > Create/Update Class

The screenshot shows a dark-themed web application for managing classes. At the top left is the logo 'SETU Code Lab'. To its right are links for 'Leaderboard', 'Problems', and 'Profile'. On the far right is a user icon. The main area has a header 'Create New Class'. On the left, under 'Manage Problems', there's a 'Title:' input field with placeholder text 'eg. Y3 Software Development...'. Below it are two dropdown menus: 'Add Problem' and 'Add Student', each with four items listed. A large red 'Next' button is at the bottom right.

Problem	Student
1. Two Sum	1. Stuart Rossiter
2. Palindrome Number	2. Diarmuid O'Neill
3. Remove Element	3. Conor Hendley
4. Median of Two Sorted Arrays	4. Isalah Andres

[Next](#)

# Automated Deployment Script

The following script automates the deployment process. It works by pulling any code changes from the remote GitHub repository, running the frontend build step, running the backend build step and finally restarting the backend application.

```
#!/usr/bin/env bash

# Exit immediately if a command exits with a non-zero status
set -e

cd /var/www/SETU_Code_Lab || exit 1

git pull

cd /var/www/SETU_Code_Lab/SETU_Code_Lab_Code/client || exit 1

npm run build

cd /var/www/SETU_Code_Lab/SETU_Code_Lab_Code/server || exit 1

npm run build

pm2 restart setucl-backend

echo "Deployment Successful"
```