SETU Code Lab
Design Document

Diarmuid O'Neill

South East Technological University

12/02/2026

Table of Contents

Introduction

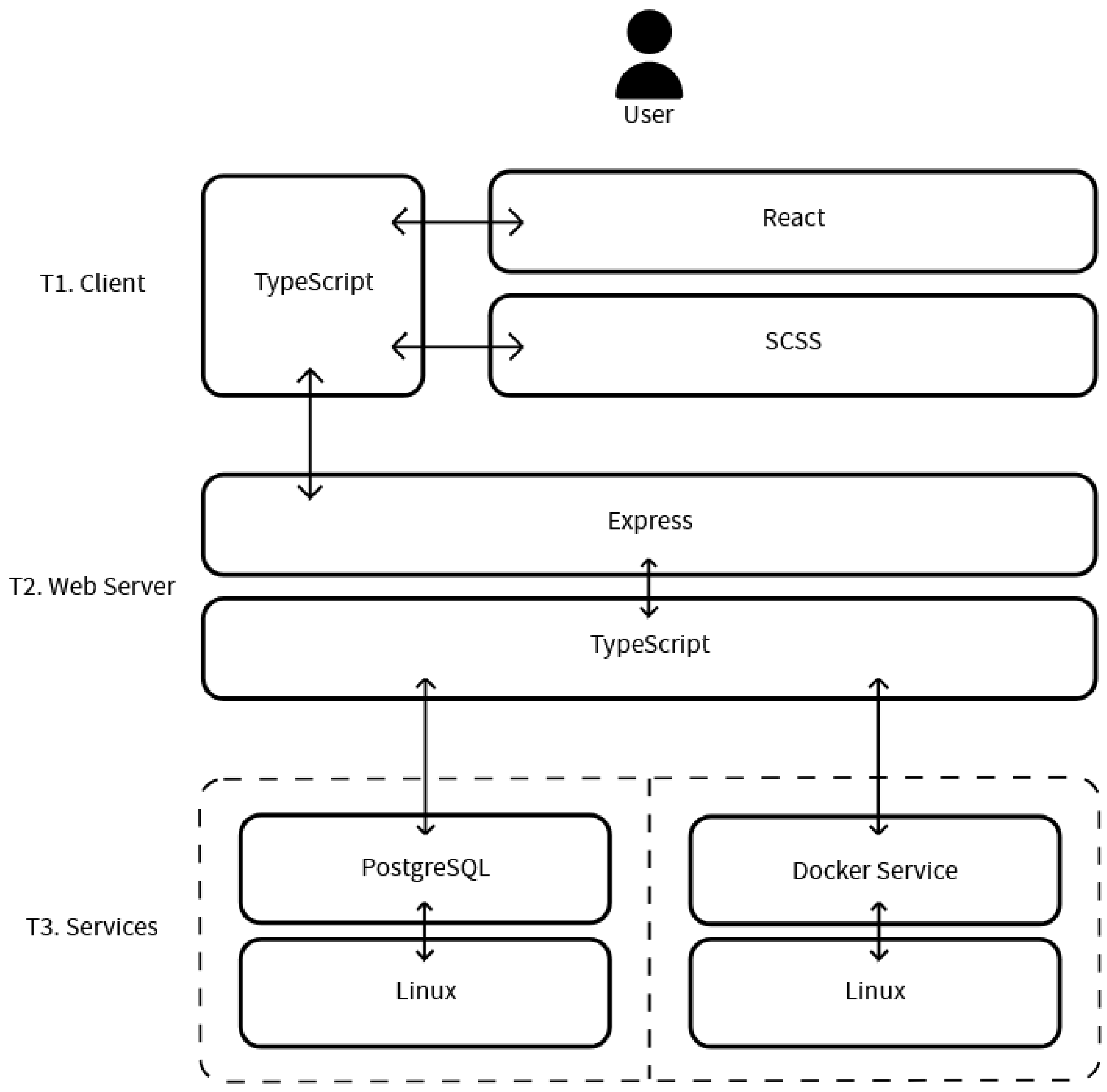
The purpose of this document is to outline the proposed design for SETU Code Lab. It will explore how each part of the project is intended to be implemented. It includes sections on hosting, sequence diagrams, important algorithms, database design and the user interface.

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.

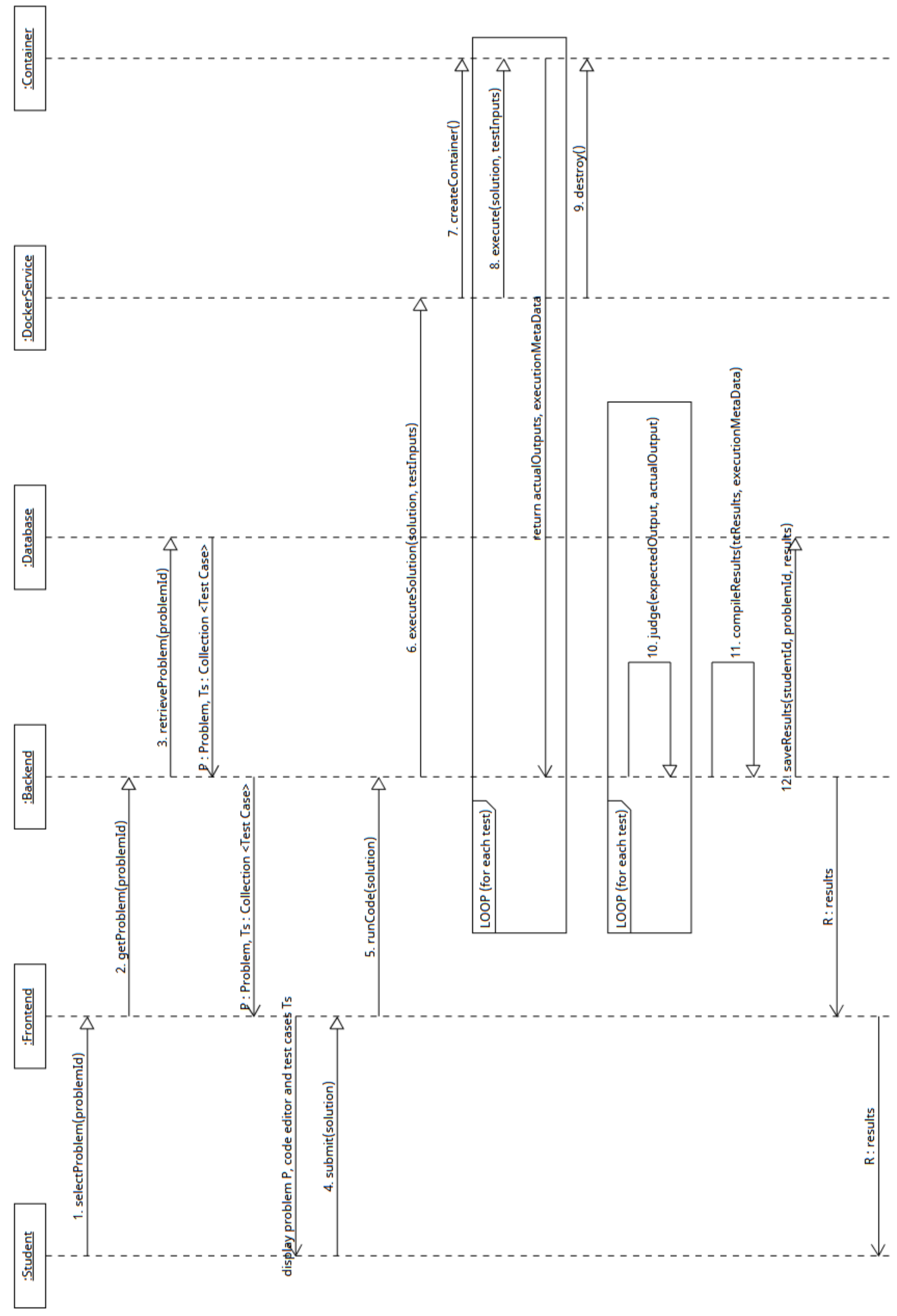
Architecture Diagram



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.

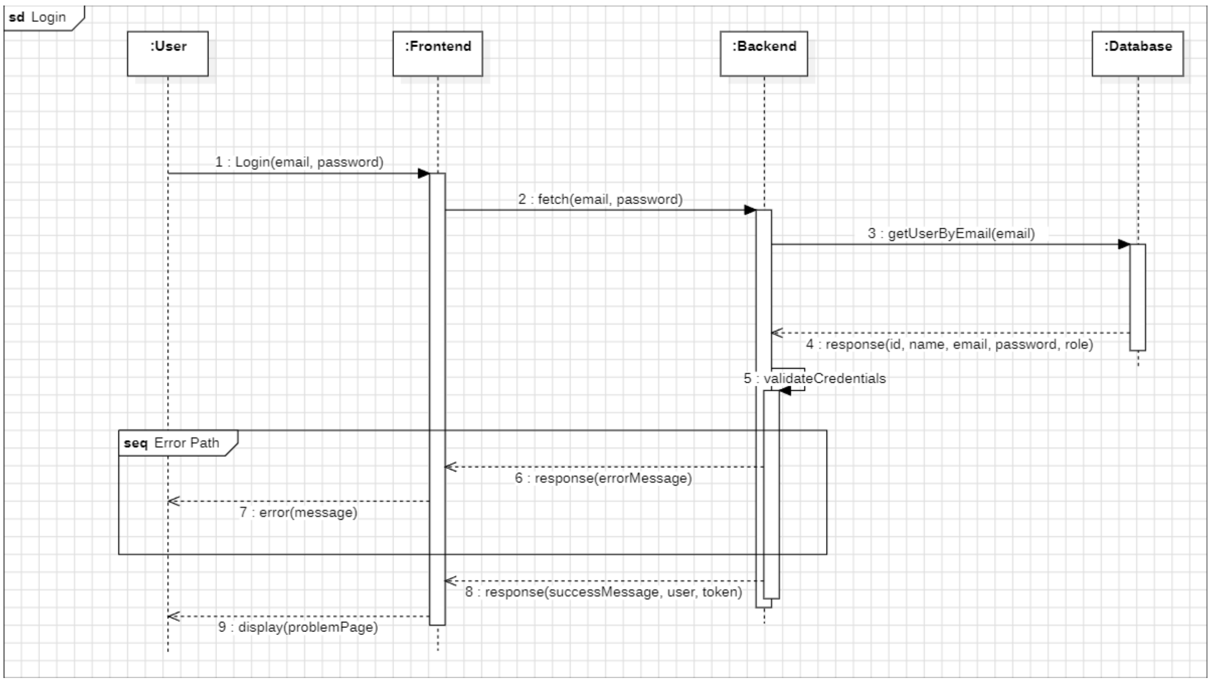
Sequence Diagrams

Solve a Problem



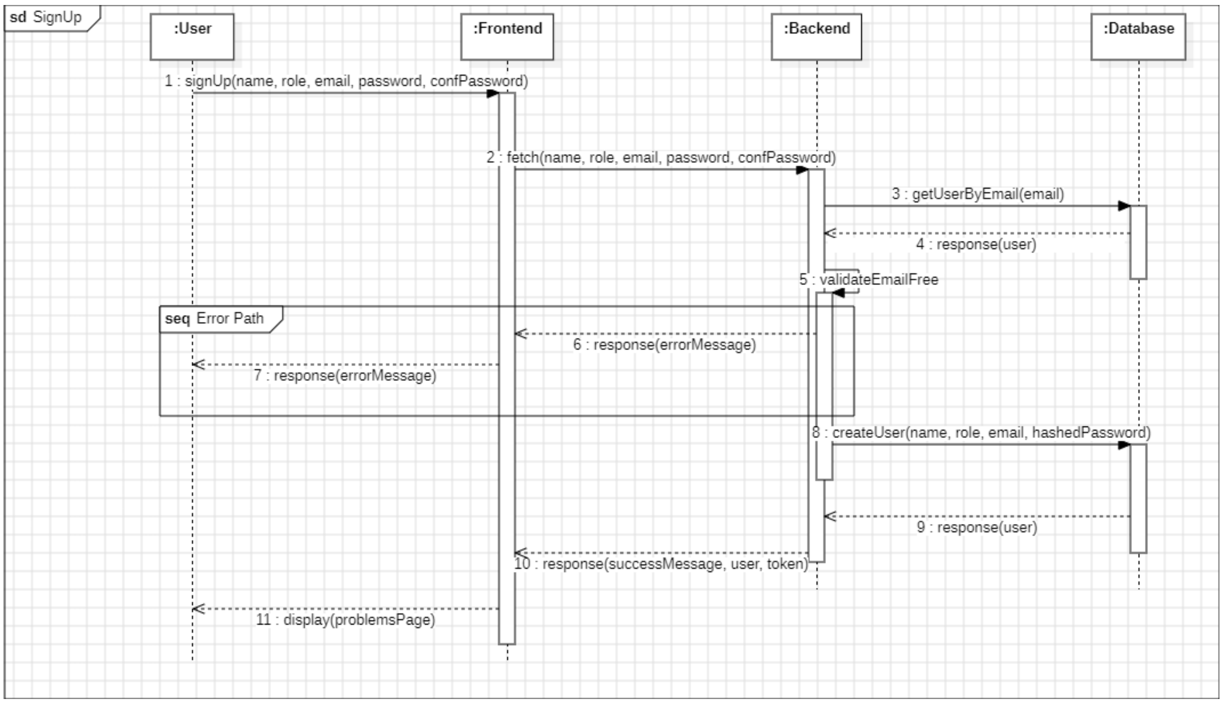
:Student
:Frontend
:Backend
:Database
:DockerService
:Container
1. selectProblem(problemId)
2. getProblem(problemId)
3. retrieveProblem(problemId)
P : Problem, Ts : Collection <Test Case>
P : Problem, Ts : Collection <Test Case>
display problem P, code editor and test cases Ts
4. submit(solution)
5. runCode(solution)
6. executeSolution(solution, testInputs)
7. createContainer()
8. execute(solution, testInputs)
return actualOutputs, executionMetaData
9. destroy()
LOOP (for each test)
10. judge(expectedOutput, actualOutput)
LOOP (for each test)
11. compileResults(tcResults, executionMetaData)
12. saveResults(studentId, problemId, results)
R : results
R : results

Login



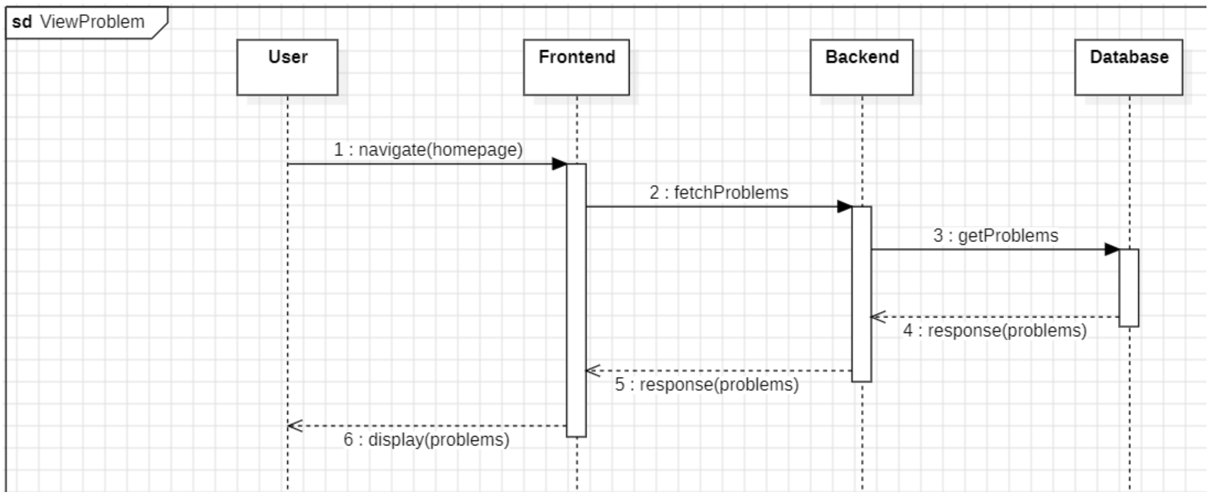
sd Login
:User
:Frontend
:Backend
:Database
1 : Login(email, password)
2 : fetch(email, password)
3 : getUserByEmail(email)
4 : response(id, name, email, password, role)
5 : validateCredentials
seq Error Path
6 : response(errorMessage)
7 : error(message)
8 : response(successMessage, user, token)
9 : display(problemPage)

Sign Up

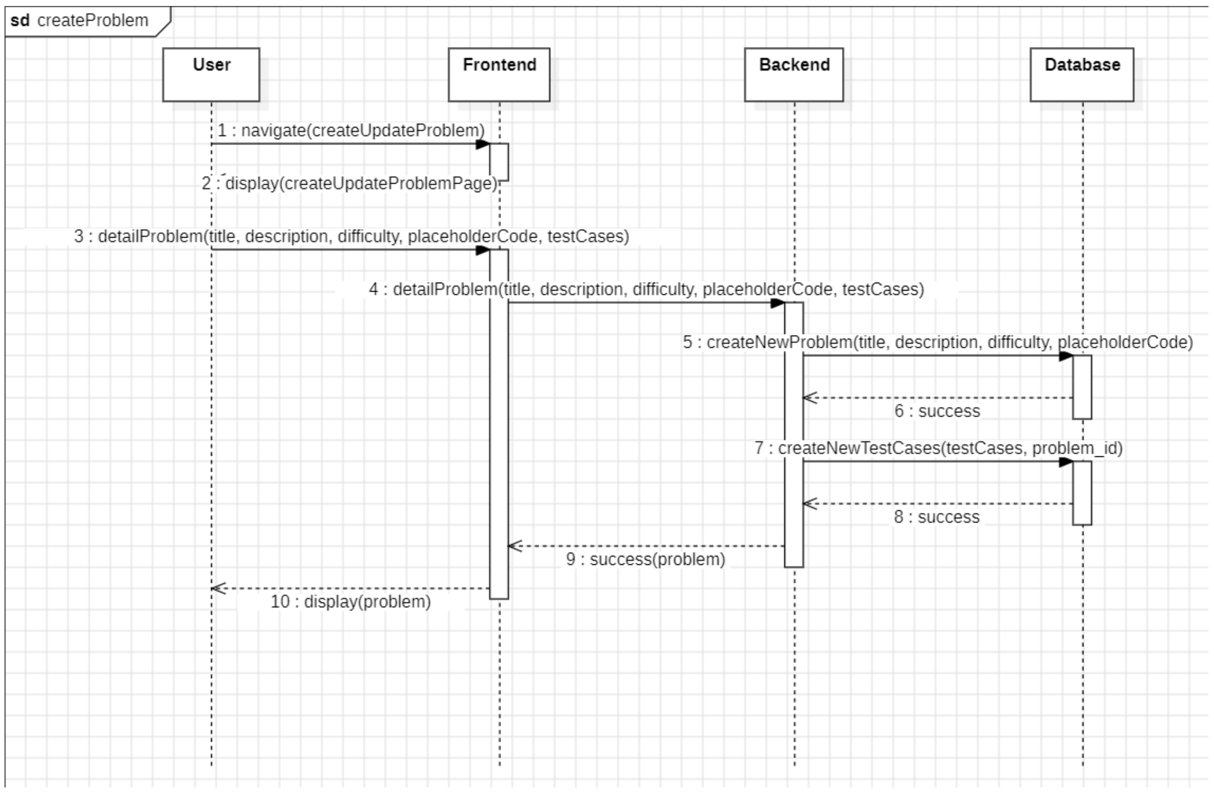


sd SignUp
:User
:Frontend
:Backend
:Database
1 : signUp(name, role, email, password, confPassword)
2 : fetch(name, role, email, password, confPassword)
3 : getUserByEmail(email)
4 : response(user)
5 : validateEmailFree
seq Error Path
6 : response(errorMessage)
7 : response(errorMessage)
8 : createUser(name, role, email, hashedPassword)
9 : response(user)
10 : response(successMessage, user, token)
11 : display(problemsPage)

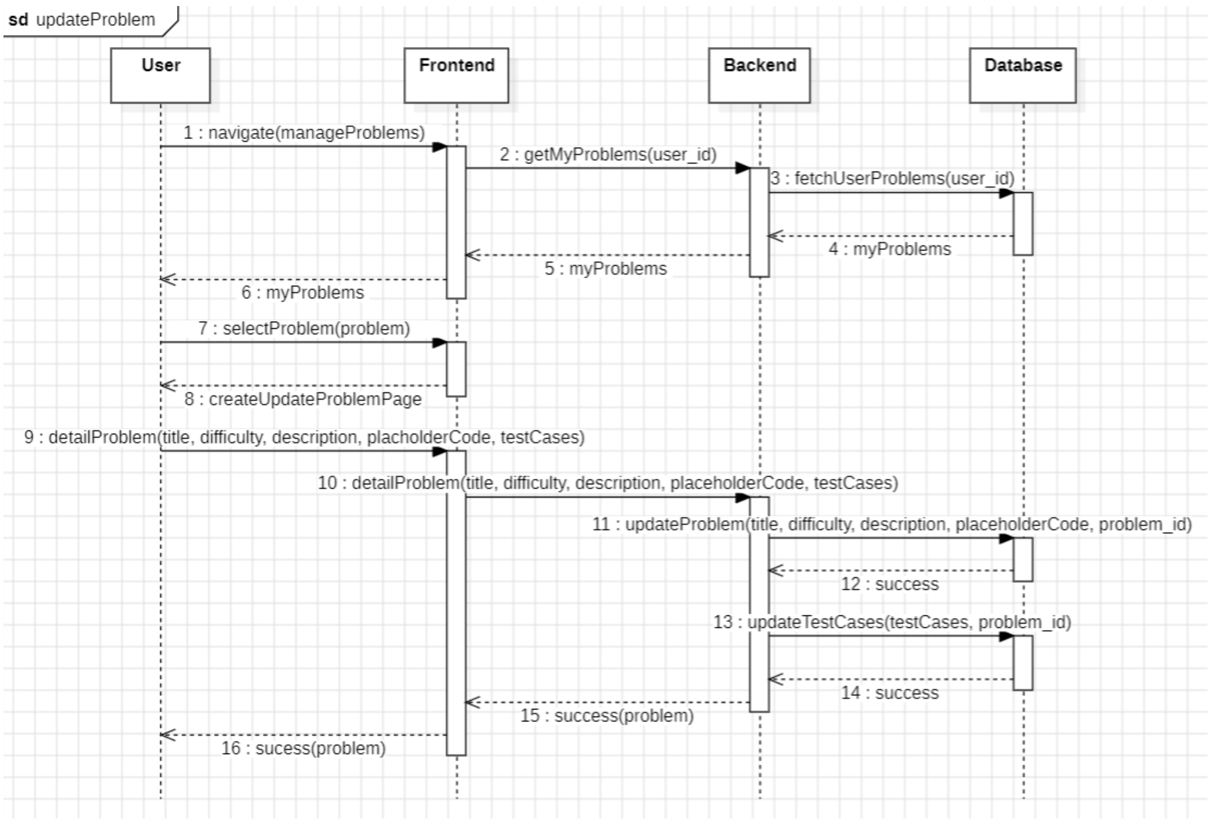
View Problems



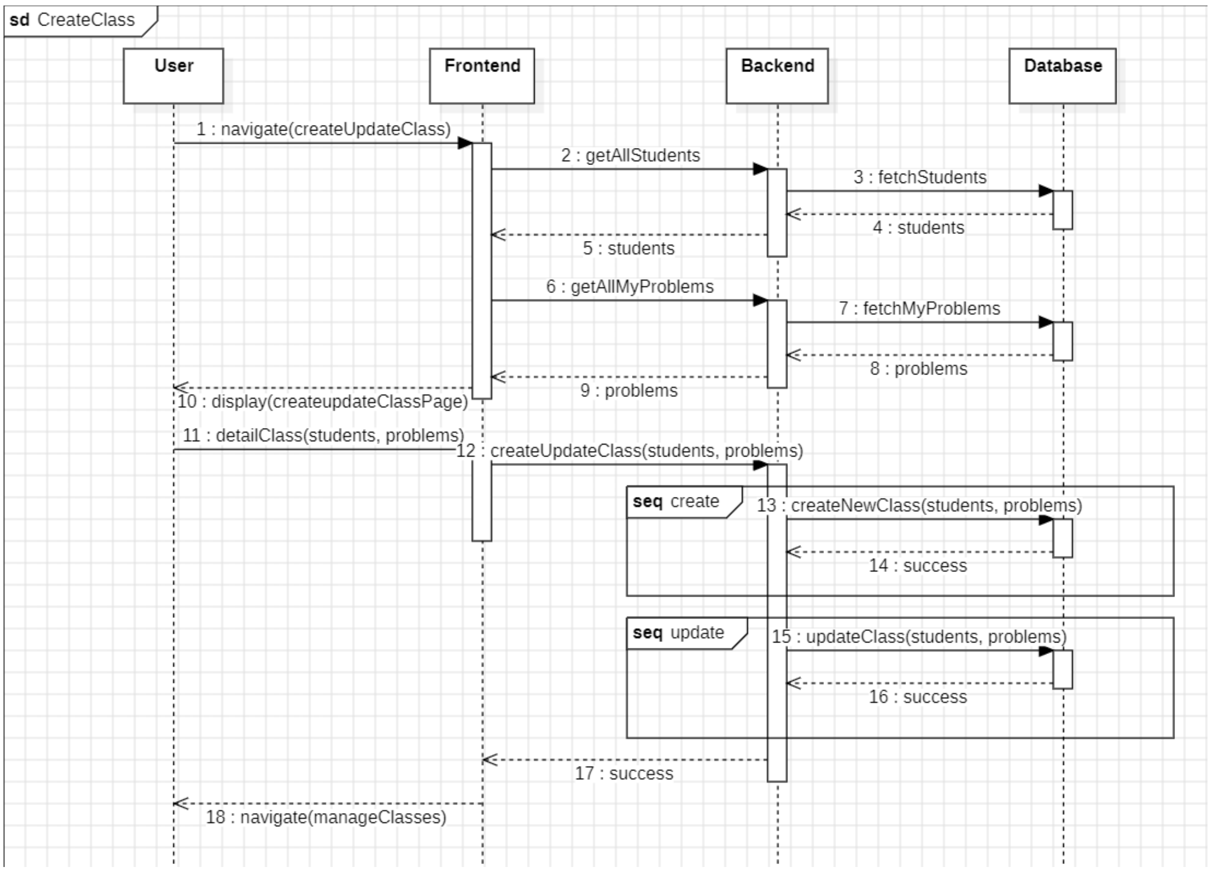
Create Problem / Test Cases


sd createProblem
User
Frontend
Backend
Database
1 : navigate(createUpdateProblem)
2 : display(createUpdateProblemPage)
3 : detailProblem(title, description, difficulty, placeholderCode, testCases)
4 : detailProblem(title, description, difficulty, placeholderCode, testCases)
5 : createNewProblem(title, description, difficulty, placeholderCode)
6 : success
7 : createNewTestCases(testCases, problem_id)
8 : success
9 : success(problem)
10 : display(problem)

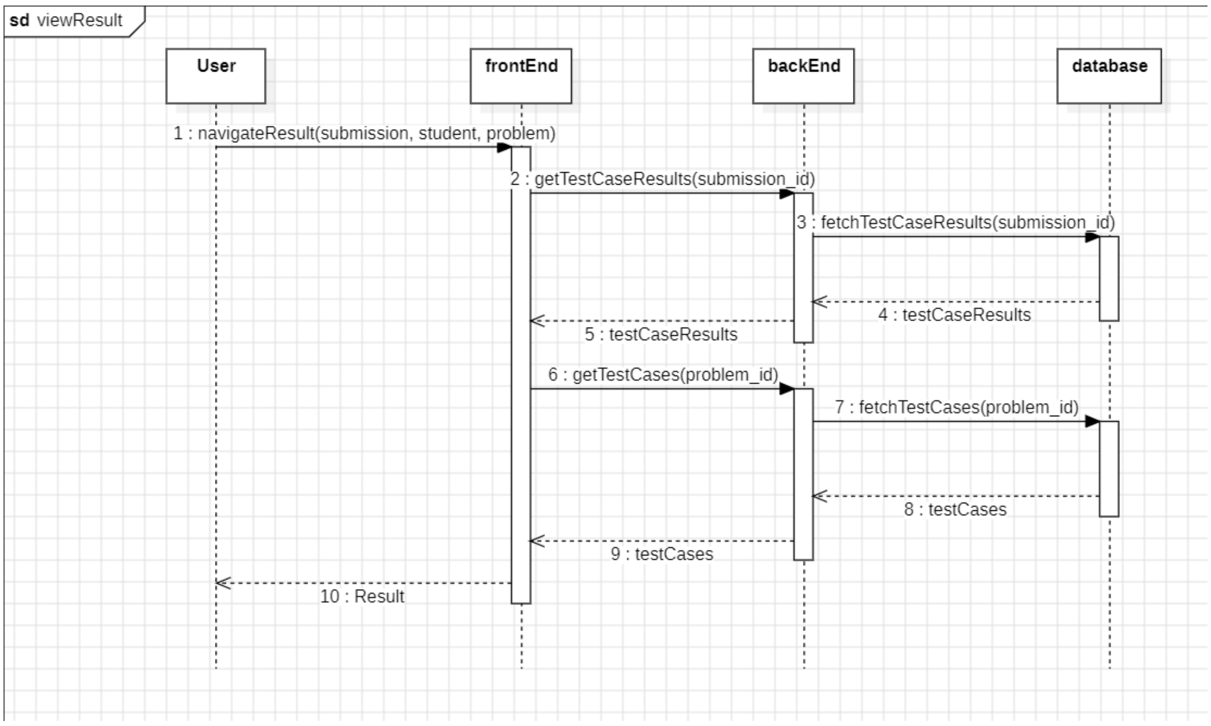
Update Problem / Test Cases


sd updateProblem
User
Frontend
Backend
Database
1 : navigate(manageProblems)
2 : getMyProblems(user_id)
3 : fetchUserProblems(user_id)
4 : myProblems
5 : myProblems
6 : myProblems
7 : selectProblem(problem)
8 : createUpdateProblemPage
9 : detailProblem(title, difficulty, description, placholderCode, testCases)
10 : detailProblem(title, difficulty, description, placeholderCode, testCases)
11 : updateProblem(title, difficulty, description, placeholderCode, problem_id)
12 : success
13 : updateTestCases(testCases, problem_id)
14 : success
15 : success(problem)
16 : sucess(problem)

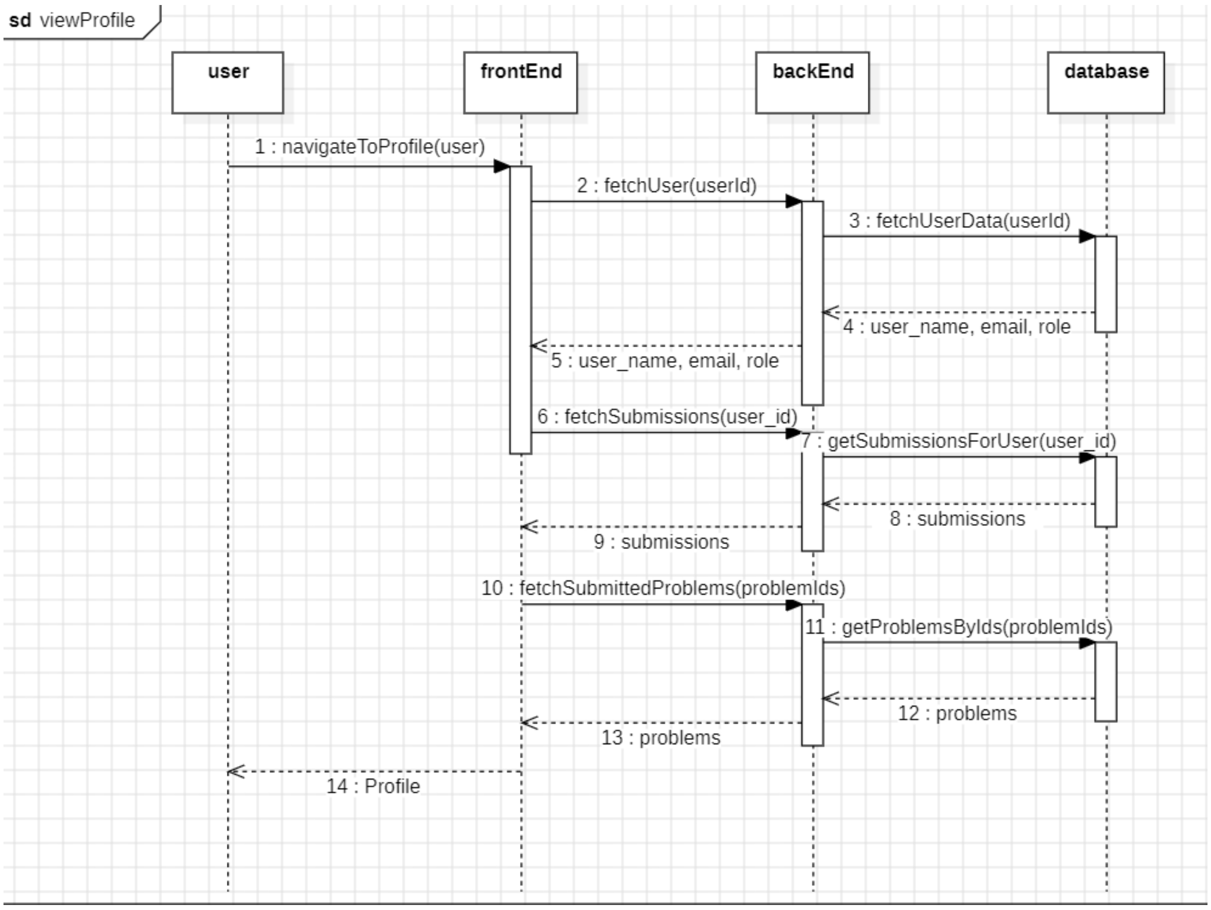
Create / Update Course


sd CreateClass
User
Frontend
Backend
Database
1 : navigate(createUpdateClass)
2 : getAllStudents
3 : fetchStudents
4 : students
5 : students
6 : getAllMyProblems
7 : fetchMyProblems
8 : problems
9 : problems
10 : display(createupdateClassPage)
11 : detailClass(students, problems)
12 : createUpdateClass(students, problems)
seq create
13 : createNewClass(students, problems)
14 : success
seq update
15 : updateClass(students, problems)
16 : success
17 : success
18 : navigate(manageClasses)

View Results


sd viewResult
User
frontEnd
backEnd
database
1 : navigateResult(submission, student, problem)
2 : getTestCaseResults(submission_id)
3 : fetchTestCaseResults(submission_id)
4 : testCaseResults
5 : testCaseResults
6 : getTestCases(problem_id)
7 : fetchTestCases(problem_id)
8 : testCases
9 : testCases
10 : Result

View Profile


sd viewProfile
user
frontEnd
backEnd
database
1 : navigateToProfile(user)
2 : fetchUser(userId)
3 : fetchUserData(userId)
4 : user_name, email, role
5 : user_name, email, role
6 : fetchSubmissions(user_id)
7 : getSubmissionsForUser(user_id)
8 : submissions
9 : submissions
10 : fetchSubmittedProblems(problemIds)
11 : getProblemsByIds(problemIds)
12 : problems
13 : problems
14 : Profile

View Leaderboard

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}'
`
```

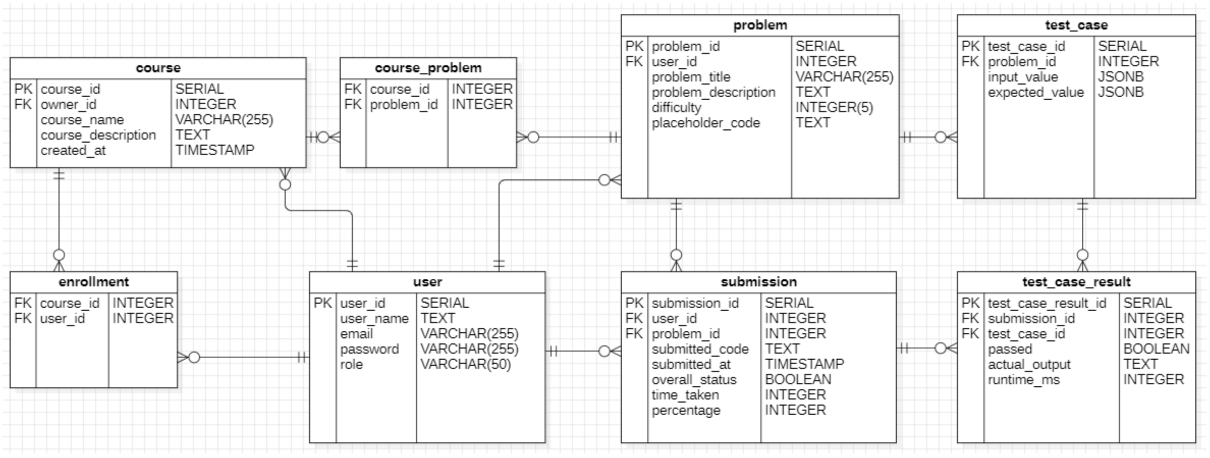
Approach to the use of Artificial Intelligence

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



course

PK	course_id	SERIAL
FK	owner_id	INTEGER
	course_name	VARCHAR(255)
	course_description	TEXT
	created_at	TIMESTAMP

course_problem

FK	course_id	INTEGER
FK	problem_id	INTEGER

problem

PK	problem_id	SERIAL
FK	user_id	INTEGER
	problem_title	VARCHAR(255)
	problem_description	TEXT
	difficulty	INTEGER(5)
	placeholder_code	TEXT

test_case

PK	test_case_id	SERIAL
FK	problem_id	INTEGER
	input_value	JSONB
	expected_value	JSONB

enrollment

FK	course_id	INTEGER
FK	user_id	INTEGER

user

PK	user_id	SERIAL
	user_name	TEXT
	email	VARCHAR(255)
	password	VARCHAR(255)
	role	VARCHAR(50)

submission

PK	submission_id	SERIAL
FK	user_id	INTEGER
FK	problem_id	INTEGER
	submitted_code	TEXT
	submitted_at	TIMESTAMP
	overall_status	BOOLEAN
	time_taken	INTEGER
	percentage	INTEGER

test_case_result

PK	test_case_result_id	SERIAL
FK	submission_id	INTEGER
FK	test_case_id	INTEGER
	passed	BOOLEAN
	actual_output	TEXT
	runtime_ms	INTEGER

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,
    percentage      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,
    created_at           TIMESTAMP DEFAULT now()
);
```

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 *
```

fetchProblemsByUserId SQL Statement

```
SELECT problem.*, users.user_name
    FROM problem AS problem
    JOIN users AS users
    ON problem.user_id = users.user_id
    WHERE problem.user_id = $1
```

getAllAvailableProblems SQL Statement

```
SELECT * FROM problem
    WHERE user_id = $1 OR user_id = 1
```

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
```

View Results

fetchStudentsOnCourse SQL Statement

```
SELECT u.user_id, u.user_name FROM users u
    JOIN enrollment e ON u.user_id = e.user_id
    WHERE e.course_id = $1
```

getSubmissionsForCourse SQL Statement

```
SELECT DISTINCT ON (s.user_id, s.problem_id)
            s.submission_id,
            s.user_id,
            s.problem_id,
            s.submitted_code,
            s.submitted_at,
            s.overall_status,
            s.time_taken,
            s.percentage
        FROM submission s
        WHERE s.user_id = ANY($1)
          AND s.problem_id = ANY($2)
          AND s.submitted_at > $3::timestamp
        ORDER BY
            s.user_id,
            s.problem_id,
            s.submitted_at DESC
```

fetchTestCaseResults SQL Statement

```
SELECT * FROM test_case_result WHERE submission_id=$1
```

View Profile

fetchUserData SQL Statement

```
SELECT user_id, user_name, email, role FROM users
    WHERE user_id = $1
```

deleteAccount SQL Statement

```
DELETE FROM users WHERE user_id = $1
```

getSubmissionsForUser SQL Statement

```
SELECT * FROM submission
    WHERE user_id = $1
    ORDER BY submitted_at DESC
```

getProblemsByIds SQL Statement

```
SELECT * FROM problem WHERE problem_id=ANY($1::int[])
```

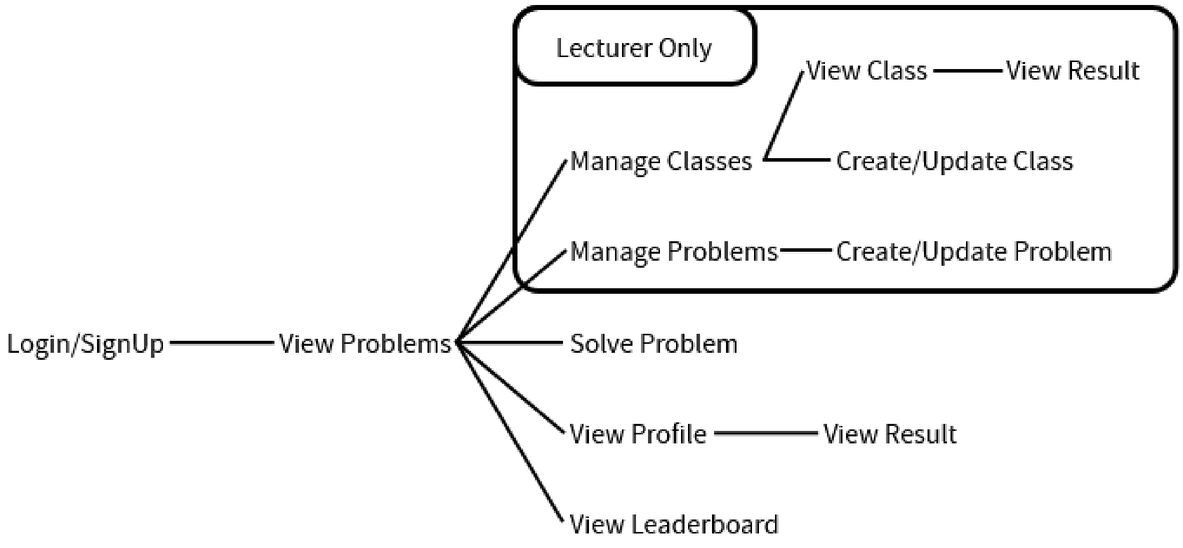
User Interface

Logo

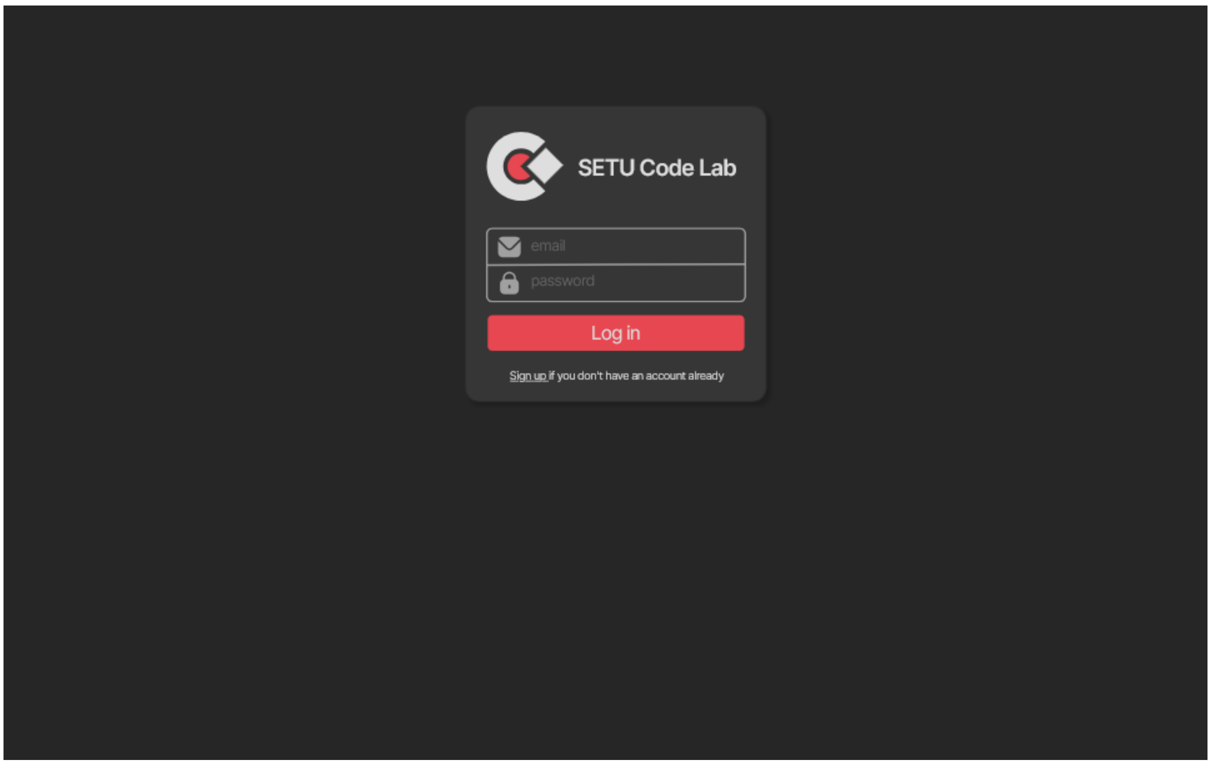
This is the logo and favicon for SETU Code Lab. It has been designed in the shape of the letter C and is intended to be simple and recognizable. The light grey and bright red colours have been chosen as they contrast well with the dark background chosen for the rest of the platform.

High-Level UI Flow

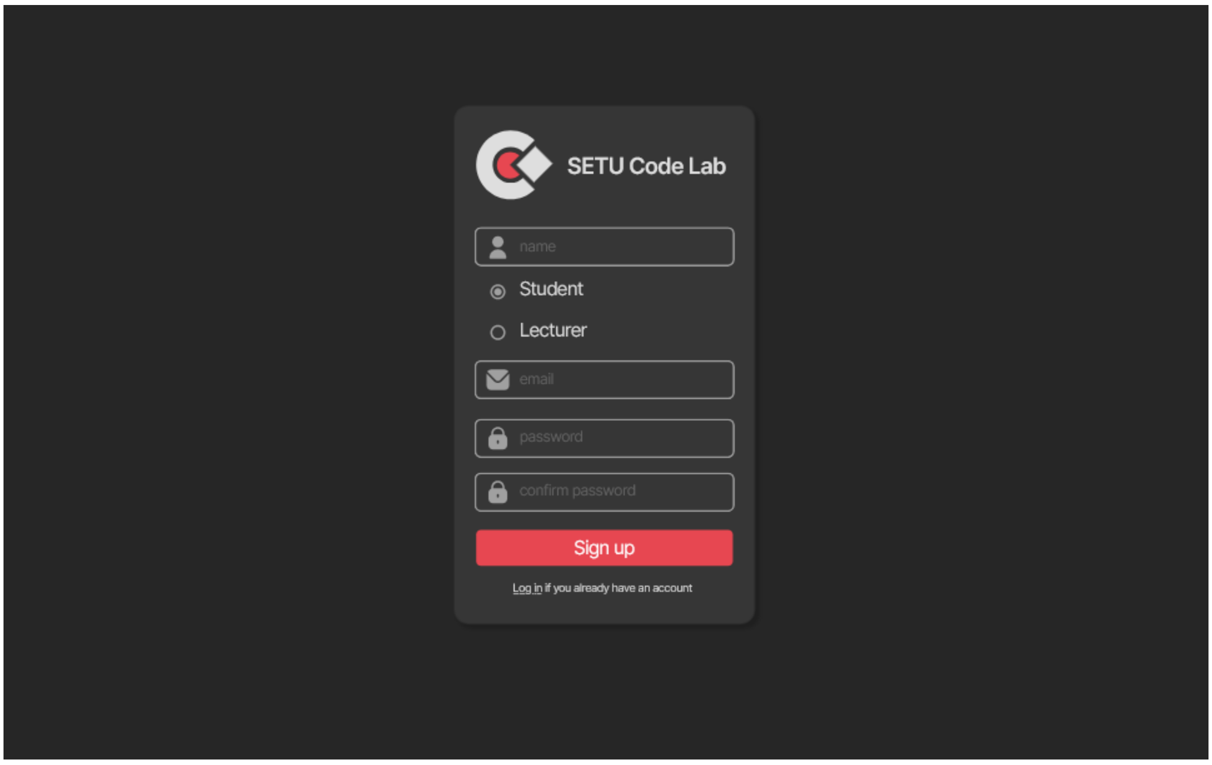
The diagram below shows how to navigate to each screen on the platform. Additional UI elements such as pop-ups and drop-down menus are not shown here, just the main screens. Items in the lecturer only box are only accessible to users with the lecturer role.



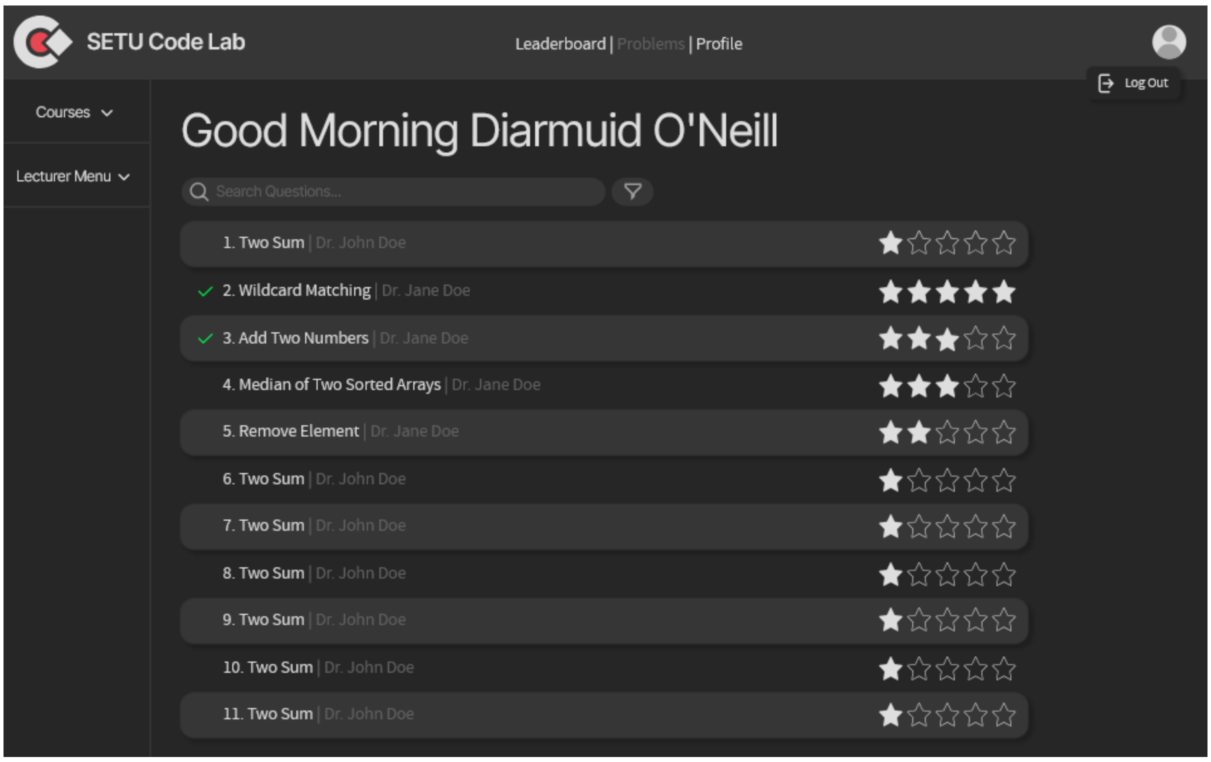
Login



Sign Up



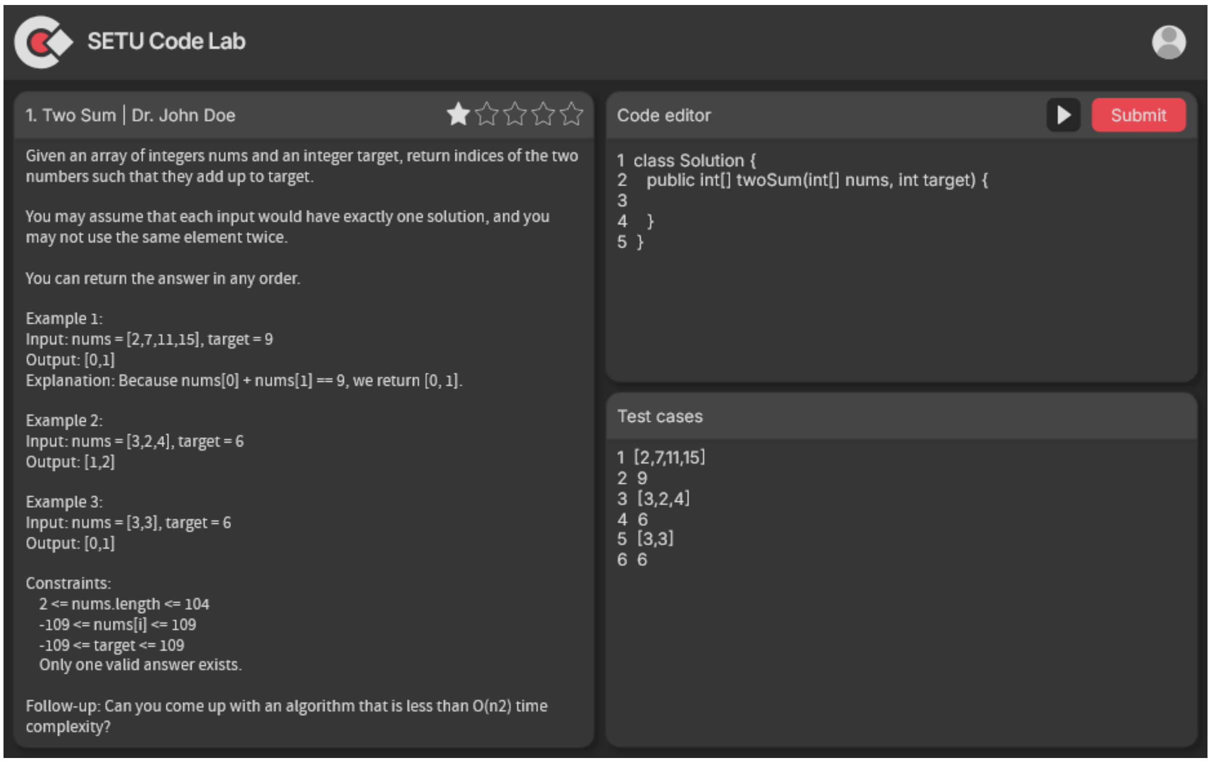
View Problems



Good Morning Diarmuid O'Neill

	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



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 <= nums.length <= 104
-109 <= nums[i] <= 109
-109 <= target <= 109
Only one valid answer exists.

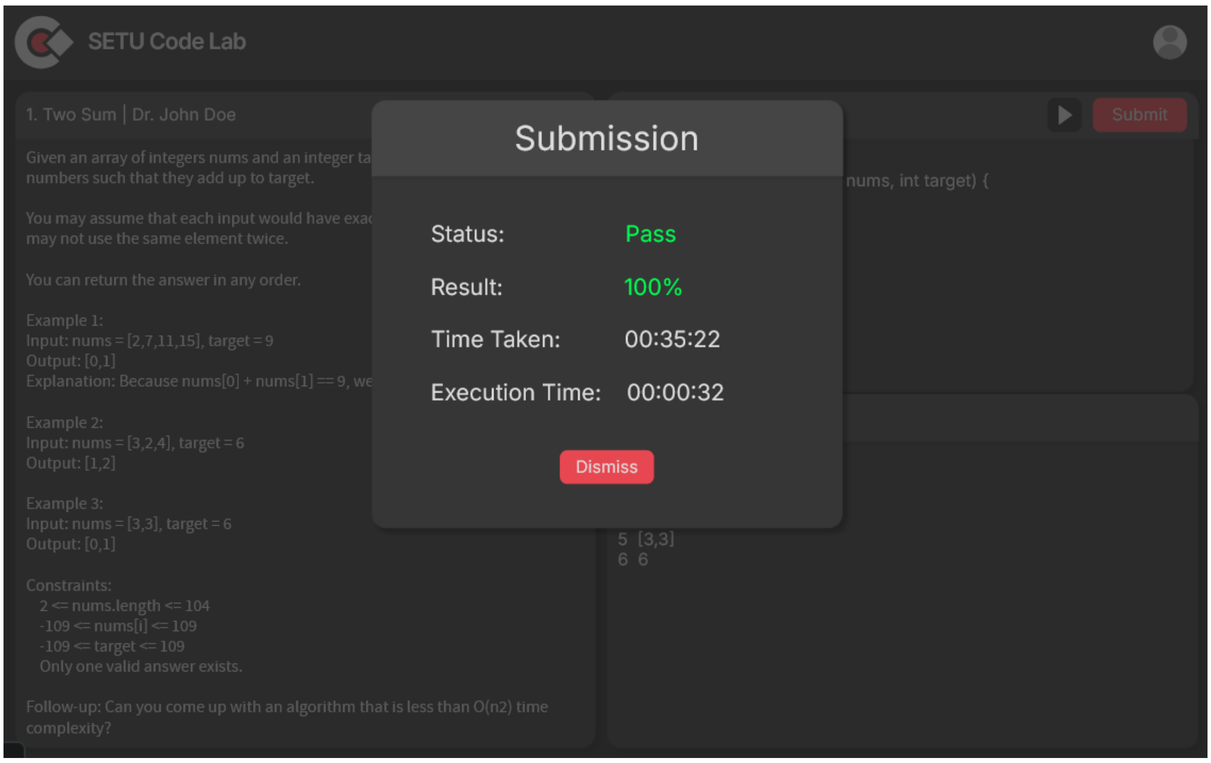
Follow-up: Can you come up with an algorithm that is less than O(n2) time complexity?

```
class Solution {
  public int[] twoSum(int[] nums, int target) {

  }
}
```

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

View Problem > Solve Problem >Submission Alert



SETU Code Lab

1. Two Sum | Dr. John Doe

Given an array of integers nums and an integer ta
numbers such that they add up to target.

You may assume that each input would have exac
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

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 <= nums.length <= 104
-109 <= nums[i] <= 109
-109 <= target <= 109
Only one valid answer exists.

Follow-up: Can you come up with an algorithm that is less than O(n2) time complexity?

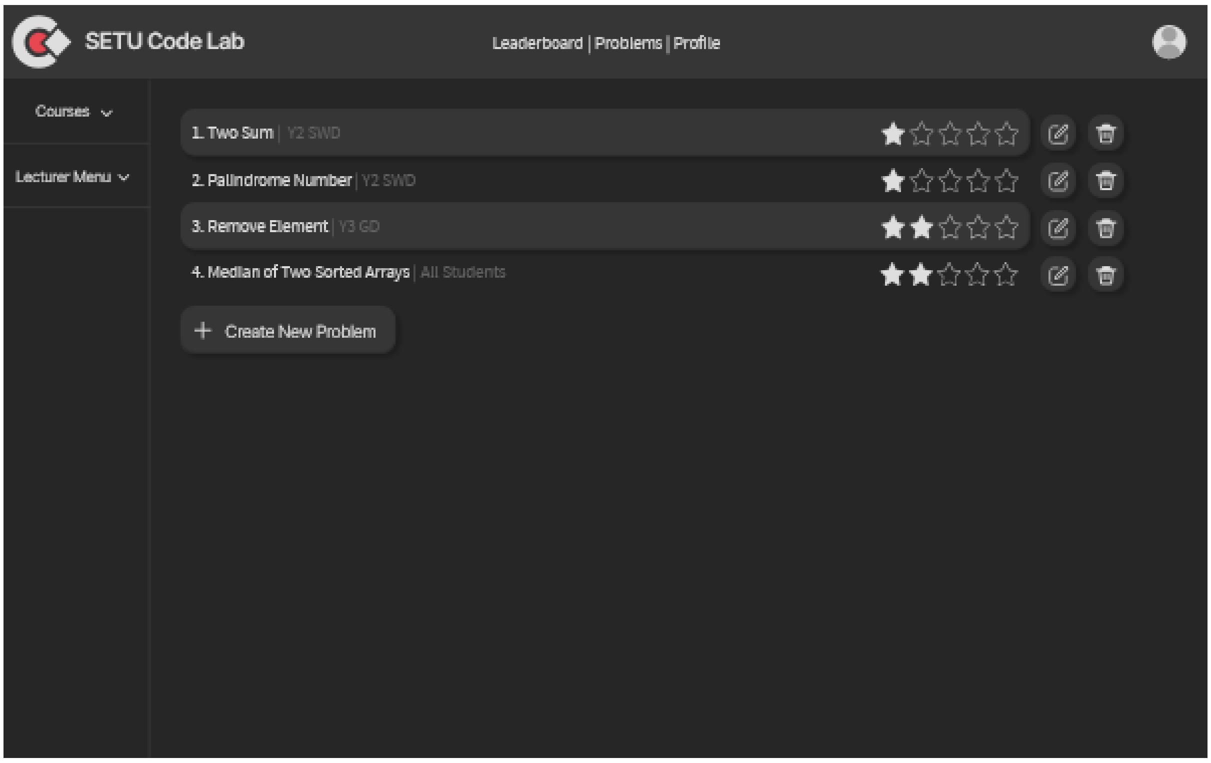
```
nums, int target) {
```

```
5  [3,3]
6  6
```

Submission

Status:	Pass
Result:	100%
Time Taken:	00:35:22
Execution Time:	00:00:32

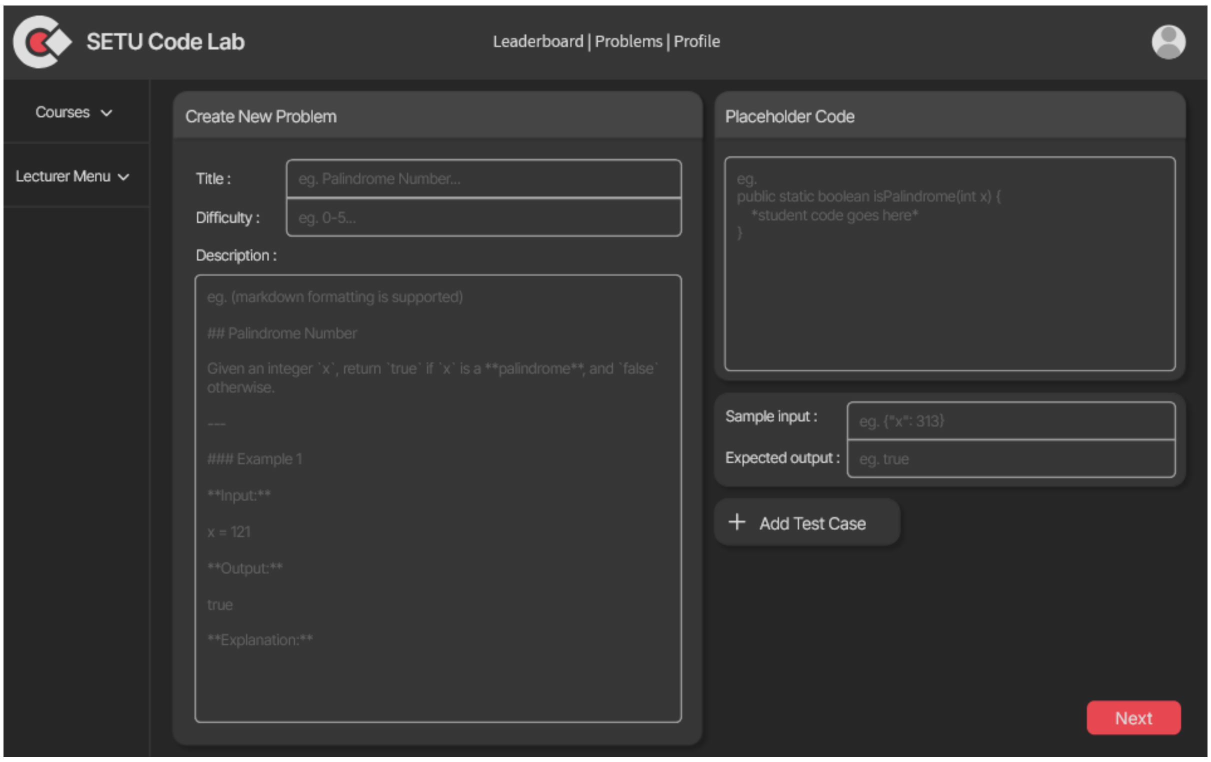
View Problem > Manage Problems



SETU Code Lab

1. Two Sum | Y2 SWD ★☆☆☆☆
2. Palindrome Number | Y2 SWD ★☆☆☆☆
3. Remove Element | Y3 GD ★★☆☆☆
4. Median of Two Sorted Arrays | All Students ★★☆☆☆

View Problem > Manage Problems > Create/Update Problem



Create New Problem

Title : eg. Palindrome Number...

Difficulty : eg. 0-5...

Description :

```
eg. (markdown formatting is supported)

## Palindrome Number

Given an integer `x`, return `true` if `x` is a **palindrome**, and `false`
otherwise.

---

### Example 1

**Input:**

x = 121

**Output:**

true

**Explanation:**
```

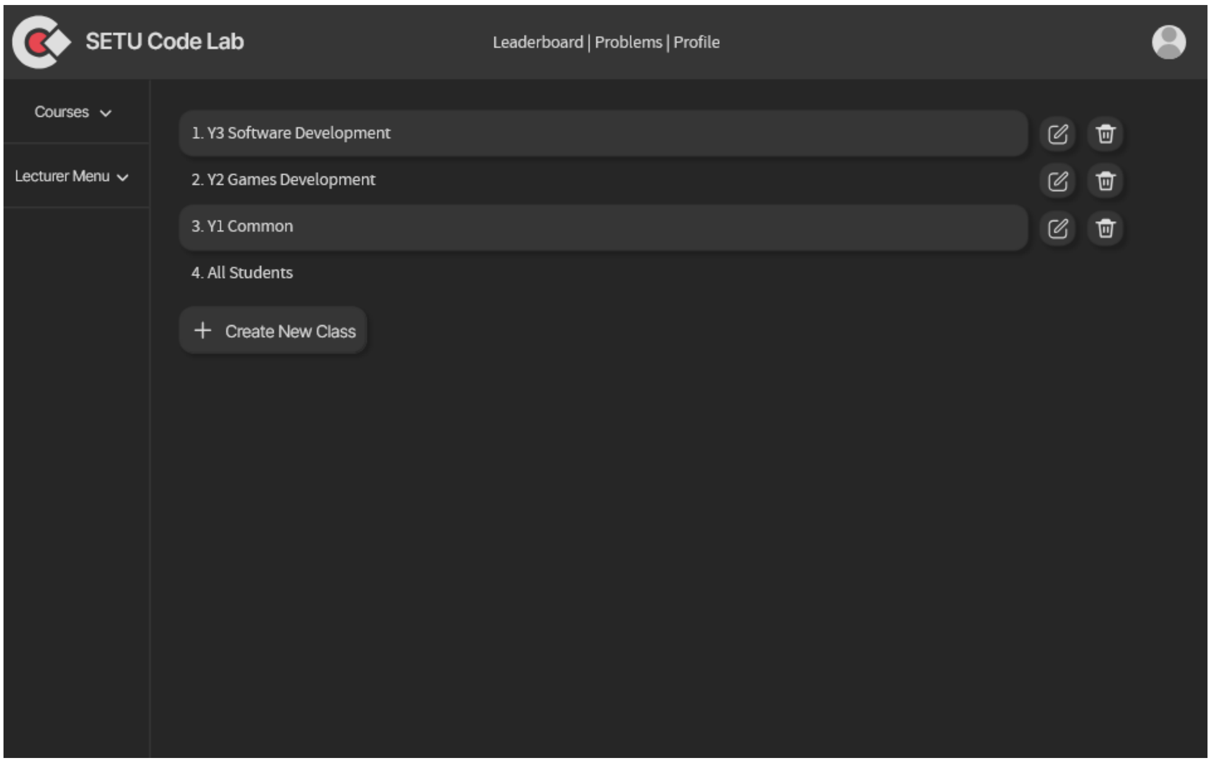
Placeholder Code

```
eg.
public static boolean isPalindrome(int x) {
   *student code goes here*
}
```

Sample input : eg. {"x": 313}

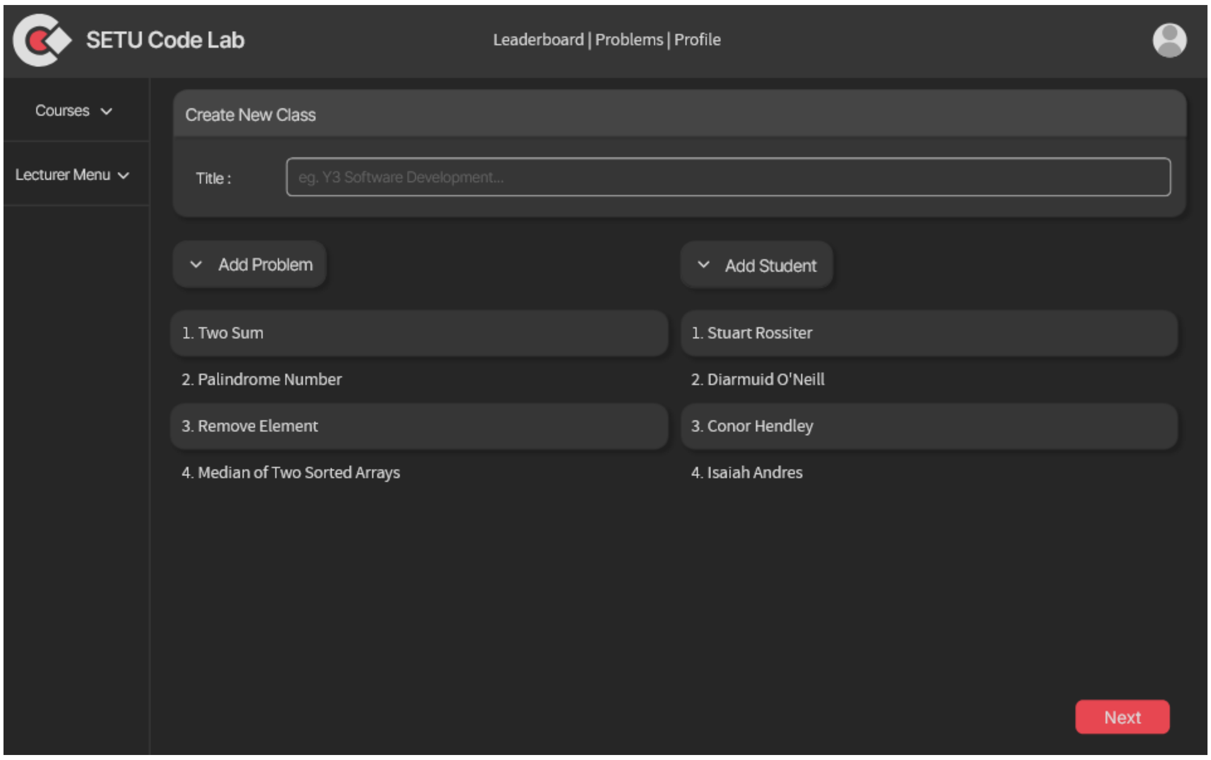
Expected output : eg. true

View Problem > Manage Courses



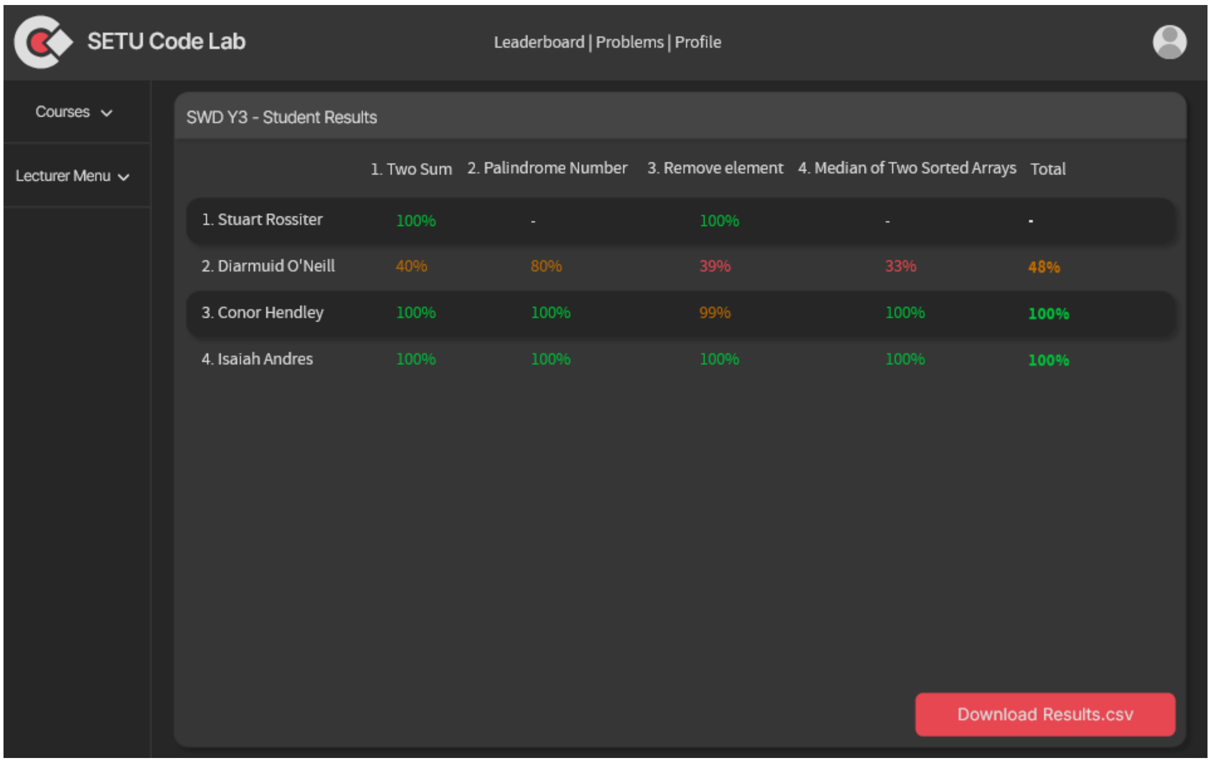
1. Y3 Software Development
2. Y2 Games Development
3. Y1 Common
4. All Students

View Problem > Manage Courses > Create/Update Course



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. Isaiah Andres

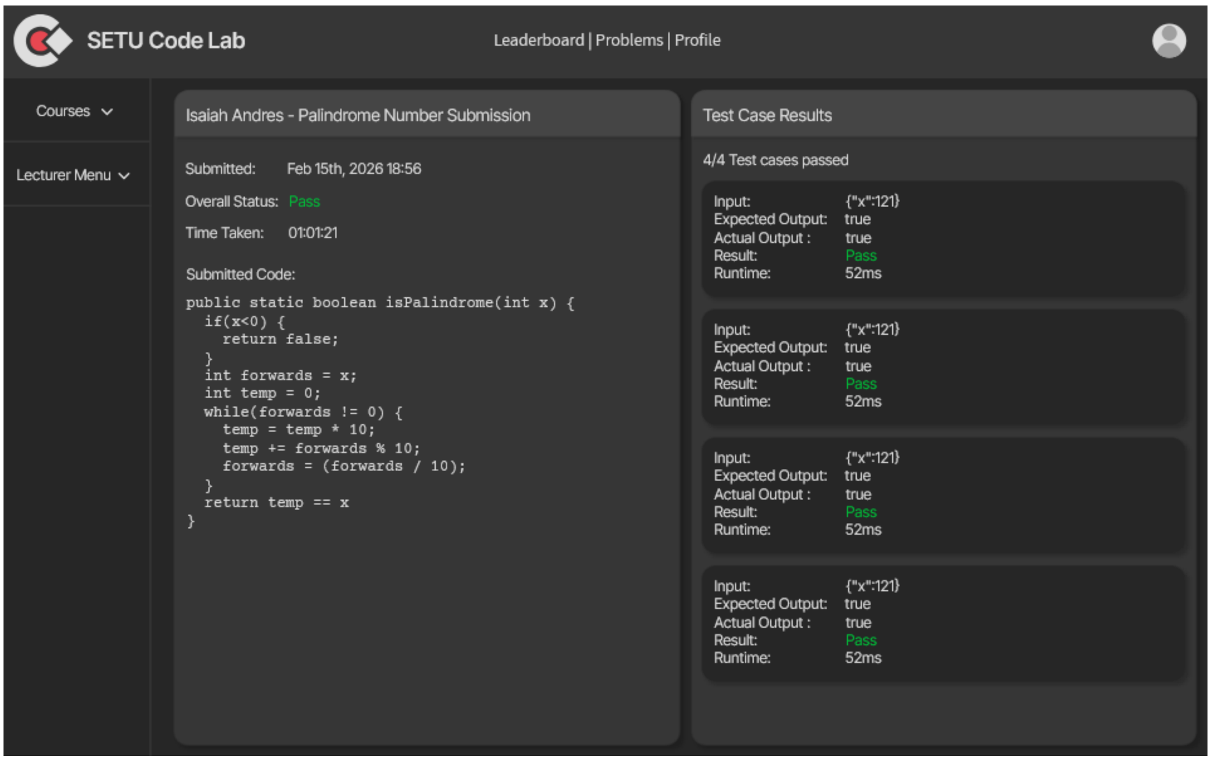
View Problem > Manage Courses > View Course



SWD Y3 - Student Results

	1. Two Sum	2. Palindrome Number	3. Remove element	4. Median of Two Sorted Arrays	Total
1. Stuart Rossiter	100%	-	100%	-	-
2. Diarmuid O'Neill	40%	80%	39%	33%	48%
3. Conor Hendley	100%	100%	99%	100%	100%
4. Isaiah Andres	100%	100%	100%	100%	100%

View Problem > Manage Course > View Course > View Result



Isaiah Andres - Palindrome Number Submission

Submitted: Feb 15th, 2026 18:56

Overall Status: Pass

Time Taken: 01:01:21

Submitted Code:

```
public static boolean isPalindrome(int x) {
  if(x<0) {
    return false;
  }
  int forwards = x;
  int temp = 0;
  while(forwards != 0) {
    temp = temp * 10;
    temp += forwards % 10;
    forwards = (forwards / 10);
  }
  return temp == x
}
```

Test Case Results

4/4 Test cases passed

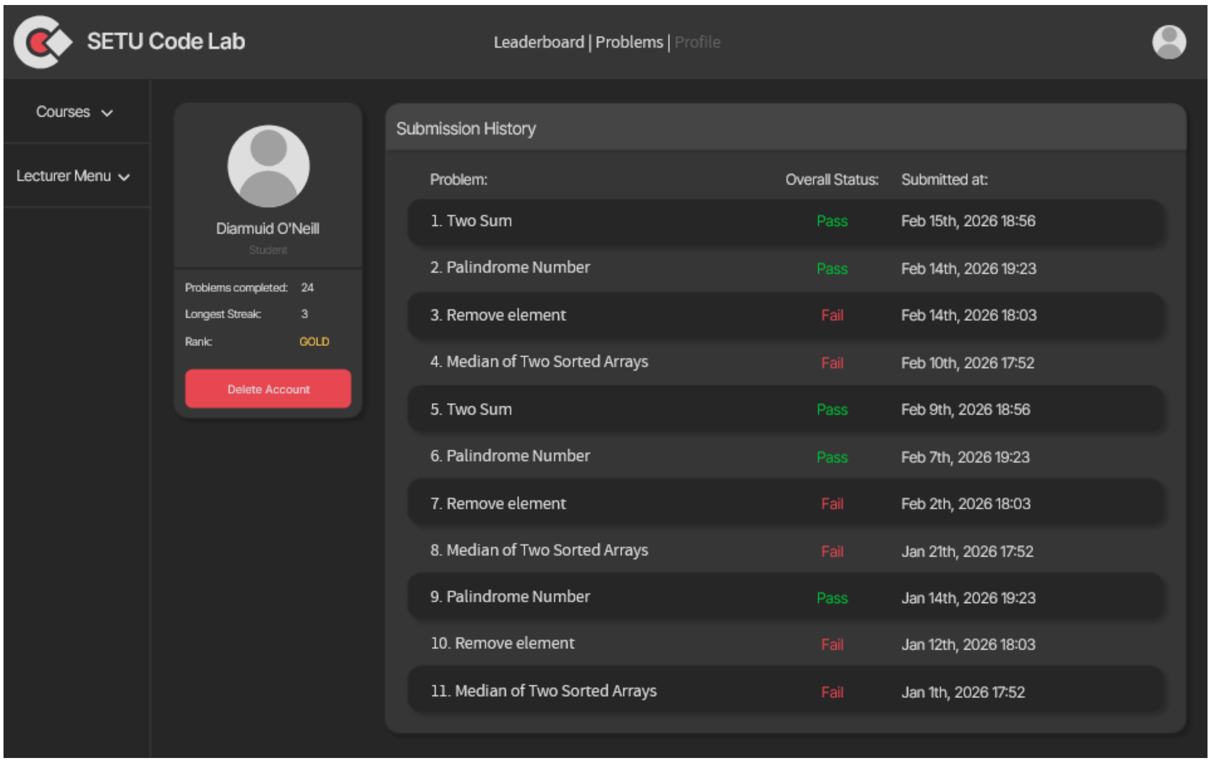
Input:	{"x":121}
Expected Output:	true
Actual Output :	true
Result:	Pass
Runtime:	52ms

Input:	{"x":121}
Expected Output:	true
Actual Output :	true
Result:	Pass
Runtime:	52ms

Input:	{"x":121}
Expected Output:	true
Actual Output :	true
Result:	Pass
Runtime:	52ms

Input:	{"x":121}
Expected Output:	true
Actual Output :	true
Result:	Pass
Runtime:	52ms

View Problem > Profile



Diarmuid O'Neill

Student

Problems completed:	24
Longest Streak:	3
Rank:	GOLD

Submission History

Problem:	Overall Status:	Submitted at:
1. Two Sum	Pass	Feb 15th, 2026 18:56
2. Palindrome Number	Pass	Feb 14th, 2026 19:23
3. Remove element	Fail	Feb 14th, 2026 18:03
4. Median of Two Sorted Arrays	Fail	Feb 10th, 2026 17:52
5. Two Sum	Pass	Feb 9th, 2026 18:56
6. Palindrome Number	Pass	Feb 7th, 2026 19:23
7. Remove element	Fail	Feb 2th, 2026 18:03
8. Median of Two Sorted Arrays	Fail	Jan 21th, 2026 17:52
9. Palindrome Number	Pass	Jan 14th, 2026 19:23
10. Remove element	Fail	Jan 12th, 2026 18:03
11. Median of Two Sorted Arrays	Fail	Jan 1th, 2026 17:52

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"
```