Project Title: Developing IDE for Competitive Programming

Supervised By
Musabbir Hasan Sammak
Senior Lecturer
Department of Software Engineering
Daffodil International University

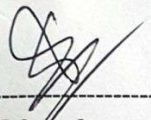
Submitted By
Md Shamim Ahamed
ID: 213-35-775
Department of Software Engineering
Daffodil International University

This Project report has been submitted in fulfillment of the requirements for the Degree of Bachelor of Science in Software Engineering.

APPROVAL

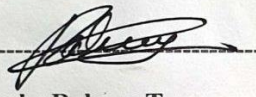
This Project titled on "**Developing IDE for Competitive Programming**", submitted by **Md Shamim Ahamed (ID: 213-35-775)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS



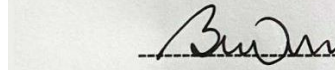
Chairman

Dr. S M Hasan Mahmud
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



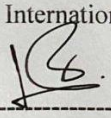
Internal Examiner 1

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 2

Khalid Been Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

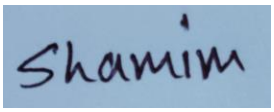


External Examiner

Dr. Md. Sazzadur Rahman
Professor
Institute of Information Technology
Jahangirnagar University

DECLARATION

I hereby declare that this project has been completed under the supervision of Musabbir Hasan Sammak, Senior Lecturer, Department of Software Engineering, Daffodil International University. I also affirm that this project is my original work, submitted for the degree of B.Sc. in Software Engineering, and neither the entire work nor any portion has been previously submitted for another degree at this or any other university.



.....

Md Shamim Ahamed
ID : 213-35-775
Department of Software Engineering
Daffodil International University

Certified By:

.....

Musabbir Hasan Sammak
Senior Lecturer
Department of Software Engineering
Daffodil International University

ACKNOWLEDGEMENT

I am profoundly grateful to Musabbir Hasan Sammak, my project supervisor, for his steady guidance, prompt feedback, and constructive criticism. His encouragement turned every setback into a learning opportunity. I also thank the support staff, instructors of *Ostad* and Esraq Humayun, former Lecturer of the Software Engineering Department at *Daffodil International University* for sharing real-world insights that shaped the system's requirements. My classmates and friends patiently tested early prototypes and suggested practical improvements. Finally, my deepest appreciation goes to my parents for their unwavering moral support and for tolerating late-night coding sessions throughout the semester.

ABSTRACT

The BrainBox IDE is a simple, browser-based platform created to improve the way competitive programming is taught in universities. In many cases, programming classes in Bangladesh still depend on separate local tools and manual email submissions, which are slow and often inconsistent. BrainBox IDE solves this problem by giving students and teachers one shared, web-based system where everything can be done in one place.

Students get a clean interface where they can view problems, write code, test it, and submit with a single click. They also receive instant results against test cases, which keeps them motivated and helps them learn from mistakes. Teachers get their own dashboard where they can easily create and update problems, track submissions, and review student work without going through piles of emails. The system is built with Next.js for the frontend and Firebase for the backend, and hosted on Vercel. This makes it fast, secure, and very affordable to run.

All major features - registration, login, problem-solving, and submission—were tested to make sure they work smoothly. The project is cost-friendly, reliable, and designed to meet real classroom needs. It supports the larger goal of modernizing education and fits with Bangladesh's Smart Education vision. In short, BrainBox IDE makes learning programming more organized for students and less stressful for teachers.

Table of Contents

APPROVAL ii
DECLARATION iii
ACKNOWLEDGEMENT iv
ABSTRACT v
Introduction 1
1.1 Project Overview and Purpose 1
1.2 Project Planning and Initiation 1
1.2.1 Feasibility Study 1
1.3 Target User Profile and Tentative Elicitation Process 6
Target User 1: Student 6
Target User 2: Faculty 6
1.4 System Requirements 7
1.4.1 Development Environment Requirements 7
1.4.2 User Environment Requirements 7
1.5 Project Scheduling 8
a. Time Frame / Gantt Chart (30 Weeks) 8
Design and Implementation 10
2.1 Functional Requirements 10
2.2 Non-Functional Requirements 12
2.3 Object-oriented System Design using UML 13
a. Use Case Diagram 13
b. Case Description 14
c. Activity Diagram 21
d. Sequence Diagram 32
e. ER Diagram 43
Software Testing 44
3.1 Testing Features 44
3.2 Testing Strategies 44
3.3 System Testing 45

Deployment and Maintenance 46
4.1 Software Release Life Cycle 46
User Manual 47
5.1 For Students 47
5.1.1 Registration 47
5.1.2 Login 48
5.1.3 Reset Password 49
5.1.4 Problem List 50
5.1.5 Problem Details and Code Editor 51
5.2 For Faculty 52
5.2.1 Login 52
5.2.2 Registration 53
5.2.3 Reset Password 54
5.2.4 Students Submissions 55
5.2.5 Create or Update Problems 56
5.2.6 Faculty Profile View 57
5.2.7 Student Submission Details View 58
Project Summary 60
6.1 Keywords 60
6.2 Summary 60
APPENDIX A: Source Code & Live Website Link 60
Accounts Clearance 61
Originality Report 62

List of Abbreviations

DIU = Daffodil International University
SWE = Software Engineering Department
CSE = Computer Science and Engineering
IT = Information Technology
IDE = Integrated Development Environment
$ = American Dollar
FR = Functional Requirements
NFR = Non-Functional Requirements
N/A = Not Applicable
CRUD = Create, Read, Upload, Delete
UML = Unified Modeling Language

Introduction

1.1 Project Overview and Purpose

The **BrainBox IDE for Competitive Programming** project was developed to move a step further with the workflow of programming education within university environments. The main purpose was to create a combined, web-accessible platform that replaces the fragmented use of local development environments and manual submission methods. The system gives role-based access for students and faculty and gives a browser-based code editor with secure execution and a simple easy to use administrative dashboard for problem management and submission tracking.

1.2 Project Planning and Initiation

1.2.1 Feasibility Study

Feasibility Area	Key Findings
Market	Addresses a clear and unmet demand in the Bangladeshi educational market.
Technical	Uses a modern, low-risk technology stack.
Financial	Exceptionally cost-effective compared to commercial alternatives.
Operational	Aligns seamlessly with existing academic workflows, ensuring easy adoption.

Phase 1: Preliminary Analysis & Project Scope

1. Problem Statement

In the current academic landscape in Bangladesh, programming education relies on a fragmented and inefficient way. While students write code on various local or online sites, leading to inconsistencies in their development environments. Code Submissions to faculty are typically handled manually through email, which is not designed for code evaluation.

Actor	Use Case	Description / Relationships
Student & Faculty	**Register**	Creates a new user account.
	Login	Signs into the system to access dashboards.
	Logout	Securely ends the user session.
	View Profile	Views their own user profile information.
	Reset Password	An optional action that can be performed from the Login screen.
Student	**View Problem List**	Sees a list of all available programming problems.
	Submit Solution	Writes and submits code for a specific problem.
Faculty	**Manage Problems**	A main task that includes creating, updating, and viewing problems.
	View Submissions	Views a list of all submissions from students.
	Create Problem	Included in Manage Problems. Adds a new problem to the system.
	Update Problem	Included in Manage Problems. Modifies an existing problem.
	View Problem Details	Included in Manage Problems. Loads the details of a specific problem.

2. Proposed Solution

- **For Students:** BrainBox will provide a clean, browser-based IDE where students can access specific problems set by faculty or admins, write code, test it against sample test cases, and submit their final solutions with a single submission button click.
- **For Faculty:** BrainBox will provide a simple and easy to use administrative dashboard. They can create and manage a set of programming problems, assign them to courses, and view all student submissions in an organized way .

3. Project Scope

In-Scope Features:

- **User Authentication:** A secure sign up and sign in system with two user roles: **Student** and **Faculty**.
- **Student Dashboard:** with a view of a list of available programming problems.
- **Integrated IDE:** A problem solving window for code writing, problem description, and run and submit code.
- **Faculty Dashboard:** For faculty to Create, Read, Update, Delete Problem statement for students.
- **Submission Tracking:** A system for faculty to view a list of all student submissions, including the submitted code, status (Correct/Incorrect), and output.

Out-of-Scope Features:

- Advanced, automated plagiarism detection.
- Live collaboration features (real-time code sharing or chat).
- Integration with external Learning Management Systems (LMS) like Moodle or Canvas.
- Public leaderboards or advanced gamification elements.
- Support for team-based projects or submissions.

Phase 2: Market Feasibility

The target market is universities with CSE/IT/SWE departments. A significant need exists to replace inefficient manual submission systems and expensive foreign software. The platform's faculty-centric dashboard provides a key competitive advantage over generic coding websites.

Factor	Analysis (Bangladesh Perspective)	Viability
Target Market	Public and private universities with CSE/IT departments.	**High**

Market Need	High demand for a centralized, affordable platform to replace inefficient manual submission systems and expensive foreign software.	**High**
Competitive Advantage	Purpose-built for academic workflows with a faculty-centric dashboard, a feature lacking in generic competitive programming sites.	**High**
Alignment	Supports national "Smart Bangladesh" goals by promoting local EdTech solutions.	**High**

Phase 3: Technical Feasibility

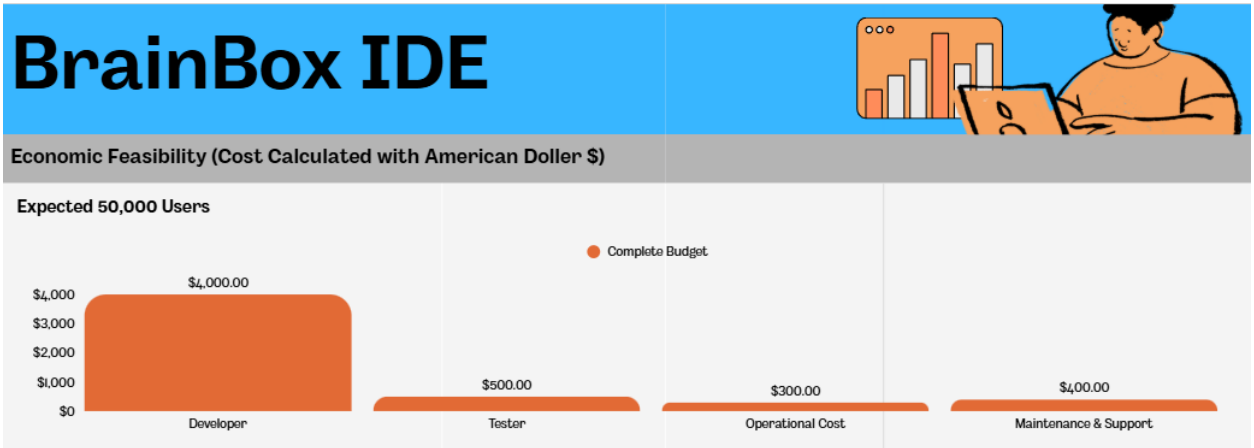
This study validates that the chosen technology stack (Next.js, Firebase, Secure API) is modern, scalable, and poses low technical risk, ensuring the project can be successfully built and maintained.

Component	Chosen Technology	Feasibility & Rationale
Frontend	Next.js / React	**High:** Excellent performance, strong local developer community, and rapid UI de development with TailwindCSS.
Backend	Firebase (Auth, Firestore)	**High:** Reduces complexity, ensures security, and provides a scalable, real-time database with minimal infrastructure management.
State Management	Recoil	**High:** Modern, efficient, and simple state management designed for React, reducing boilerplate code.
Code Execution	Secure API Sandbox	**High:** The most secure and viable method. It offloads

		significant security risks and infrastructure complexity, making the project manageable.

Phase 4: Financial Feasibility

The project is financially viable with minimal cost (around $15). Development relies on open-source software, while deployment is handled by the generous free tiers of Vercel for hosting and Firebase for backend services, which are sufficient for an initial setting.



Financial Item	Complete Budget	Initial Budget	Duration
Developer	$4,000.00	$0.00	8 Months
Tester	$500.00	$0.00	1 Week
Operational Cost	$300.00	$15.00	Monthly
Maintenance & Support	$400.00	$0.00	Monthly
Total Budget:	***$5,200.00***	***$15.00***	

Figure 1.1 : Estimated Budget

1.3 Target User Profile and Tentative Elicitation Process

Target User 1: Student

Aspect	Details
Profile	University students in introductory to advanced programming courses.
Goals	To access assigned problems from a single dashboard, write and test code in a browser-based IDE, and receive instant feedback on submissions.
Elicitation	Feedback was gathered through informal interviews with student peers, focusing on the pain points of current submission systems. Early prototypes of the IDE were tested for usability.

Target User 2: Faculty

Aspect	Details
Profile	University instructors and teaching assistants.
Goals	To create and manage a centralized bank of programming problems, track submissions per student and per problem, and view detailed code and output for evaluation.
Elicitation	Requirements were gathered by consulting with faculty members to model the dashboard and problem management features based on their existing grading workflows.

1.4 System Requirements

1.4.1 Development Environment Requirements

Category	Requirements
Operating System	Windows, macOS or Linux
Software	1. Node js (Version 18.x or later) 2. Code Editor (VS code recommended) 3. Git and a command-line interface (CLI)
Accounts	1. GitHub 2. Vercel 3. Google Firebase 4. HackerRank Code Execution API 5. Judge0 Code Execution API

1.4.2 User Environment Requirements

Category	Requirements
Hardware	Any Modern Computer
Software	Web Browser
Network	Stable broadband or mobile data connection

1.5 Project Scheduling

a. Time Frame / Gantt Chart (30 Weeks)

This schedule outlines the 30-week duration, breaking the project into five distinct phases with specific milestones.

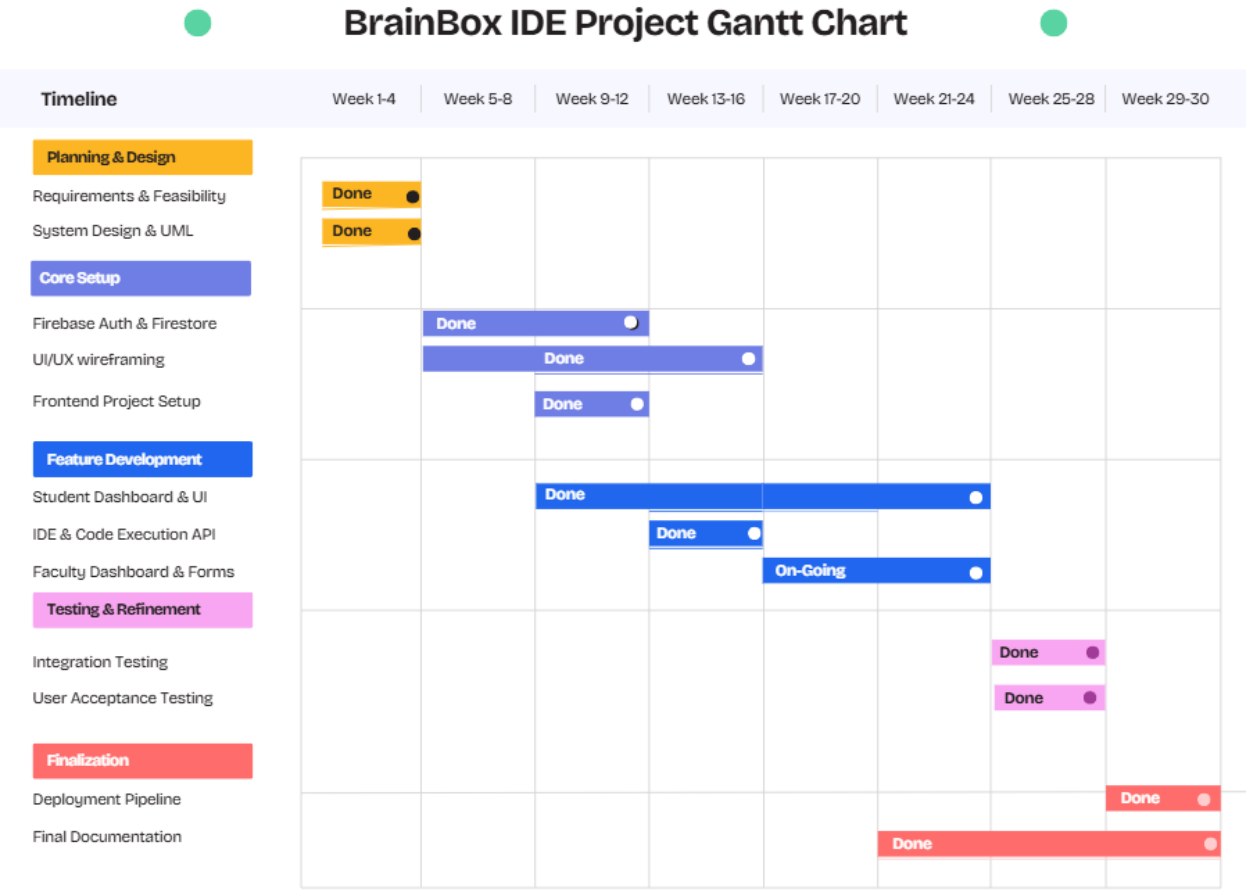


Figure 1.2 : Gantt Chart

b. Risk Management

ID	Risk Description	Mitigation Strategy
R1	Third-party code execution API becomes unavailable.	Abstract API calls into a dedicated service module to allow for easy replacement with an alternative like Judge0.
R2	Security vulnerability in the code submission process.	Rely exclusively on the secure sandbox of the external API. Enforce strict Firestore Security Rules to prevent unauthorized database access.
R3	Scope creep introduces unmanageable features.	Adhere strictly to the defined functional requirements. Defer non-essential features like live leaderboards to a future version.

Design and Implementation

2.1 Functional Requirements

FR01	**User Registration**
Description	Users shall register with a name, ID, email, password, and role (Student/Faculty).
Stakeholders	Student, Faculty

FR02	**User Login**
Description	Users shall log in with their email and password to access their respective dashboards.
Stakeholders	Student, Faculty

FR03	**View Problem List**
Description	Students shall be able to view a clear, organized list of problems assigned to them.
Stakeholders	Student

FR04	**Solve and Submit Problem**
Description	Students shall be able to write, run, and submit code against a selected problem within the integrated IDE.
Stakeholders	Student

FR05	**Create Problems**
Description	Faculty shall access a dashboard to create and update programming problems.

Stakeholders	Faculty

FR06	**View Problem Statements**
Description	Students shall access a dashboard to view all the programming problems.
Stakeholders	Student

FR07	**View Submissions**
Description	Faculty shall be able to view a filterable list of all student submissions to track progress and evaluate work.
Stakeholders	Faculty

FR08	**Automated Code Evaluation**
Description	The system shall execute submitted code and automatically verify its output against the problem's expected output.
Stakeholders	N/A

FR09	**Reset Password**
Description	Users can ask for a password reset link to be sent to their email.
Stakeholders	Students, Faculty

FR10	**User Logout**
Description	The system shall allow a logged-in user to securely terminate their session and return to the Sign In screen.
Stakeholders	Student, Faculty

2.2 Non-Functional Requirements

ID	Category	Description
NFR1	**Performance**	The application must be highly responsive. Page loads, especially the problem list and IDE, must render in under 3 seconds on a standard broadband connection. Code execution should return a result within 10 seconds.
NFR2	**Security**	All user data, especially passwords, must be encrypted. The system must be protected against common web vulnerabilities (XSS). Database access must be strictly controlled by role-based Firestore Security Rules.
NFR3	**Usability**	The user interface for both students and faculty must be intuitive and require minimal to no training. The workflow for solving a problem or creating an assignment should be clear and straightforward.
NFR4	**Scalability**	The serverless architecture (Vercel and Firebase) must be able to handle load increases, supporting at least 100 concurrent users during peak times (e.g., during an exam) without performance degradation.
NFR5	**Reliability**	The application should maintain an uptime of 99.9%. It must handle errors gracefully, providing clear feedback to the user without crashing.
NFR6	**Maintainability**	The code must be well-structured, commented, and follow modern React/Next.js best practices to allow for easy updates and bug fixes by future developers.

2.3 Object-oriented System Design using UML

a. Use Case Diagram

Primary Actors : Students, Faculty
Secondary Actors : Admin

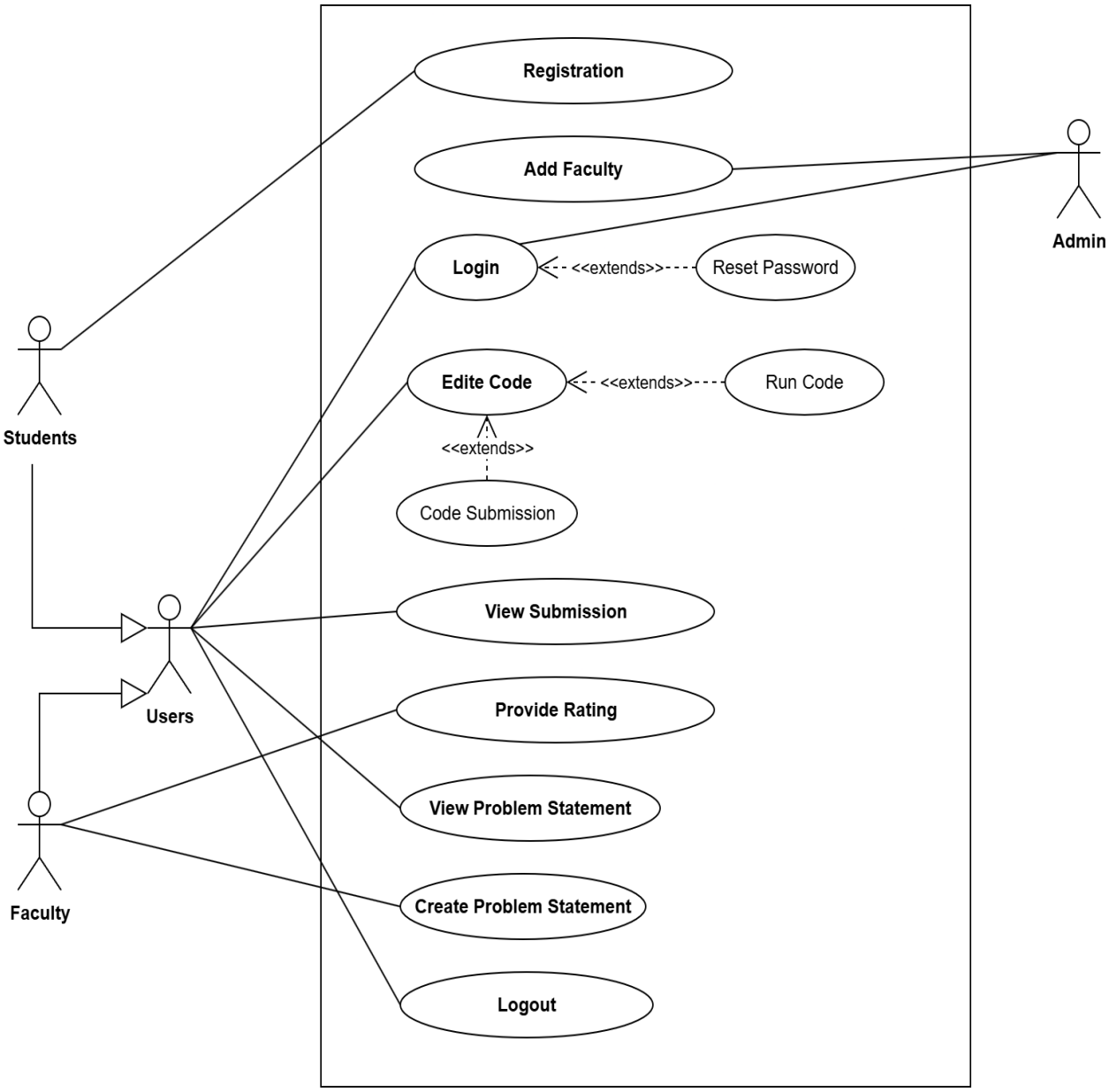


Figure 2.1 : Use Case Diagram

b. Case Description

Case Description-01: User Registration

Use Case	Register for a BrainBox IDE Account
Goal	To create a new user account with detailed profile information.
Precondition	The user does not have an existing account.
Success End Condition	A new user account is created in the database, and the user is directed to the "Sign In" screen.
Failed End Condition	Registration fails, an error message is displayed, and the user remains on the "Sign Up" screen.
Primary Actors:	Student, Faculty
Secondary Actors:	N/A
Description / Main Success Scenario	1. The user clicks "Create account" from the Sign In screen. 2. User fills in: "Full Name," "Student/Faculty ID," "Email," "Display Name," "Department," "Password," and "Confirm Password." 3. Users select their role ("Student" or "Faculty") from the "I am a..." dropdown. 4. The user clicks the "Register" button. 5. The system validates the data, creates the account, and navigates to the "Sign In" screen.
Alternative Flows	**5a. Password Mismatch:** If "Password" and "Confirm Password" do not match, an error is shown. **5b. Email Exists:** If the email is already in use, the system displays an error message.
Quality Requirements	All user data must be securely handled (NFR2). The form must be clear and easy to complete (NFR3).

Case Description-02: User Login

Use Case	Sign In to BrainBox IDE

Goal	To securely access the user's role-specific dashboard.
Precondition	User must have a registered account.
Success End Condition	User is authenticated and redirected to their corresponding dashboard (Main Dashboard for Students, Faculty Admin Dashboard for Faculty).
Failed End Condition	Authentication fails, an error message is displayed, and the user remains on the "Sign In" screen.
Primary Actors:	Student, Faculty
Secondary Actors:	
Description / Main Success Scenario	1. The user navigates to the "Sign In" screen. 2. The user enters their "Email" and "Password." 3. The user selects their role from the "Sign in as..." dropdown. 4. The user clicks the "Log In" button. 5. The system validates credentials and redirects the user to their dashboard.
Alternative Flows	**5a. Invalid Credentials:** If the email, password, or role is incorrect, an error is displayed. **Forgot Password:** User can click "Forgot Password?" to initiate the password reset process.
Quality Requirements	Login must be fast and responsive (**NFR1**). The connection must be secure (**NFR2**).

Case Description-03: View Problem List

Use Case	View Problem List
Goal	To view a clear, sortable list of available programming problems.
Precondition	Users must be logged in as a Student.
Success End Condition	The "Main Dashboard" displays a table of problems with their title, difficulty, and category.
Failed End Condition	The dashboard fails to load problems and displays an error or empty state.
Primary Actors:	Student
Secondary Actors:	N/A

Description / Main Success Scenario	1. After a successful login, the student is directed to the "Main Dashboard." 2. The system fetches and displays a list of problems in the "BrainBox Problem Table." 3. The table shows columns for "Status," "Title," "Difficulty," and "Category." 4. The student clicks on a problem row to open the "Problem Solving Screen."
Alternative Flows	**No Problems:** If no problems are available, the table is empty with a message indicating so.
Quality Requirements	The dashboard must load in under 3 seconds (**NFR1**). The interface must be clean and intuitive (**NFR3**).

Case Description-04: Solve and Submit Problem

Use Case	Solve and Submit Problem
Goal	To write, test, and submit a code solution for a selected problem.
Precondition	A student is logged in and has selected a problem from the dashboard.
Success End Condition	The code is successfully submitted, evaluated, and the result is stored.
Failed End Condition	The submission or test run fails, and the user is notified.
Primary Actors:	Student
Secondary Actors:	N/A
Description / Main Success Scenario	1. The student views the problem details on the "Problem Solving Screen." 2. The student selects a "Language" and writes their code. 3. (Optional) The student clicks "Run Code" to test against custom input. 4. The student clicks the "Submit" button to finalize. 5. The system evaluates the code and updates the submission status.
Alternative Flows	**Compilation/Runtime Error:** If the code fails during a test run, the error is displayed in the "Output" field.

Quality Requirements	Code execution should return a result within 10 seconds (**NFR1**). The IDE must be easy to use (**NFR3**).

Case Description-05: Create Problems

Use Case	Create or Update a Problem
Goal	To add a new programming problem or modify an existing one.
Precondition	User is logged in as a **Faculty** member.
Success End Condition	The problem is successfully saved to the database. A confirmation message appears.
Failed End Condition	The save operation fails. An error message is shown.
Primary Actors:	Faculty
Secondary Actors:	
Description / Main Success Scenario	1. Faculty navigates to the "Manage Problems" section. 2. **To Create:** Faculty fills in all problem detail fields and clicks "Save to DB." 3. **To Update:** Faculty enters a "Problem ID to Load & Edit," clicks "Load," modifies the form, and clicks "Save to DB."
Alternative Flows	**Invalid ID:** If the ID entered to load a problem does not exist, an error is shown. **Clear Form:** Faculty can click "Clear / New" to reset the form fields.
Quality Requirements	The form must be straightforward for managing complex problem data (**NFR3**). Access must be strictly limited to faculty (**NFR2**).

Case Description-06: View Problem Statements

Use Case	View Problem Details
Goal	To load and see the details of a specific problem.
Precondition	User must be logged in as a Student.
Success End Condition	The problem's details are fetched and shown in the pop up display.
Failed End Condition	The problem ID is not found. An error message is shown.

Primary Actors:	Student
Secondary Actors:	N/A
Description / Main Success Scenario	4. Students navigate to the "Dashboard" section. 5. Click a specific Problem from the list. 6. If problem details are successfully fetched from the server it will show the problem details.
Alternative Flows	**6a. Invalid ID:** If the ID entered to load a problem does not exist, an error is shown.
Quality Requirements	The form must be straightforward for managing complex problem data (**NFR3**).

Case Description-07: View Student Submission

Use Case	View Student Submissions
Goal	To track and review all student submissions.
Precondition	Users must be logged in as a Faculty member.
Success End Condition	A list of submissions is displayed, and the faculty can view details for any submission.
Failed End Condition	An error occurs while fetching submissions.
Primary Actors:	Faculty
Secondary Actors:	N/A
Description / Main Success Scenario	1. Faculty logs in and views the "Faculty Admin Dashboard." 2. The "Student Submissions" panel displays a table with all submissions. 3. Faculty clicks on a specific submission row. 4. A "Submission Details" modal appears, showing the student's code and execution output.
Alternative Flows	**No Submissions:** If there are no submissions, the table is empty.
Quality Requirements	The submission list must load quickly (**NFR1**). The details modal must be clear and informative (**NFR3**).

Case Description-08: Automated Code Evaluation

Use Case	Automated Code Evaluation
Goal	To automatically execute submitted code and determine a result.
Precondition	A student has clicked the "Submit" button for a problem.
Success End Condition	The system correctly determines the code's status and performance metrics and saves the result.
Failed End Condition	The evaluation cannot be completed due to an error.
Primary Actors:	N/A
Secondary Actors:	N/A
Description / Main Success Scenario	1. The system receives the student's code and the problem's test cases. 2. It sends the code and input to the Code Execution API. 3. The API executes the code and returns the output, status, memory, and time. 4. The system compares the actual output with the expected output. 5. The final result is saved to the submission record in the database.
Alternative Flows	**Code Error:** If the code has a compilation error, runtime error, or exceeds limits, the API returns the appropriate error status, which the system logs as the result.
Quality Requirements	The evaluation must be secure (**NFR2**), reliable (**NFR5**), and fast (**NFR1**).

Case Description-09: Reset Password

Use Case	Reset Password
Goal	To get a password reset link via email.
Precondition	The user has forgotten their password but knows their registered email.
Success End Condition	A password reset email is sent. A confirmation message is shown.
Failed End Condition	The email is not found. An error message is shown.

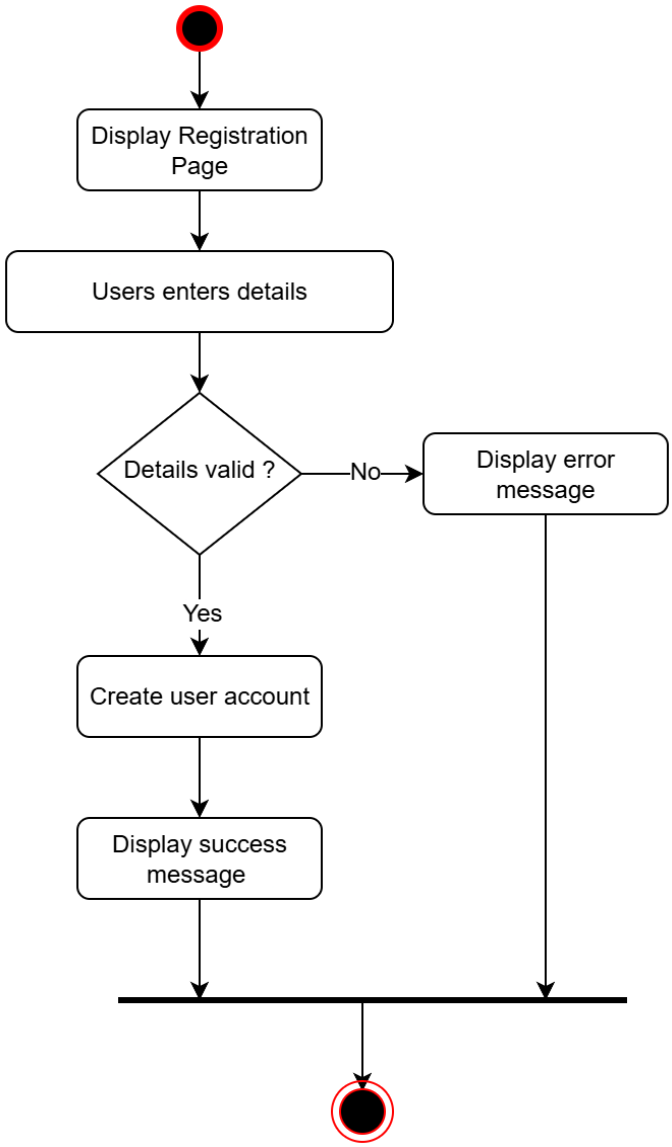
Primary Actors:	Student, Faculty
Secondary Actors:	N/A
Description / Main Success Scenario	1. The user clicks "Forgot Password?" on the "Sign In" screen. 2. The user enters their registered email address. 3. User clicks the "Reset Password" button. 4. The system sends a reset link to that email.
Alternative Flows	**4a. Email Not Found:** If the email doesn't exist in the system, an error is shown.
Quality Requirements	The process must be secure (**NFR2**). The user gets clear instructions (**NFR3**).

Case Description-10: User Logout

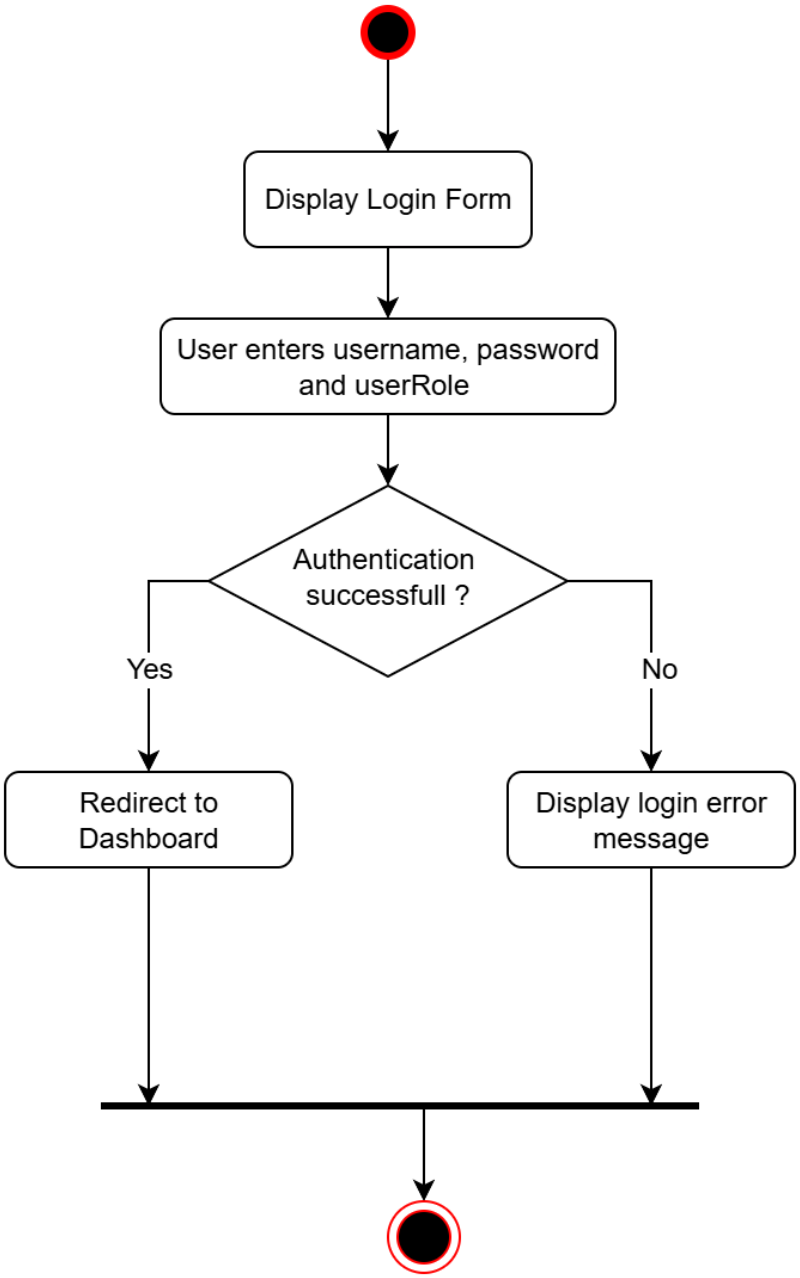
Use Case	User Logout
Goal	To securely log out of the system.
Precondition	The user is currently logged in.
Success End Condition	The user's session is terminated, a logout notification is shown, and they are redirected to the "Sign In" screen.
Failed End Condition	The logout process fails, and the user remains logged in.
Primary Actors:	Student, Faculty
Secondary Actors:	N/A
Description / Main Success Scenario	1. Users click the "Logout" or "Sign Out" button/icon in their dashboard. 2. The system calls the authentication service to terminate the session. 3. The system displays a brief notification ("You have been logged out successfully."). 4. The user is redirected to the "Sign In" screen.
Alternative Flows	**Logout Fails:** If the authentication service fails to terminate the session, an error message is displayed, and the user remains logged in.
Quality Requirements	The logout process must be immediate and secure (**NFR2**). The user should receive clear confirmation (**NFR3**).

c. Activity Diagram

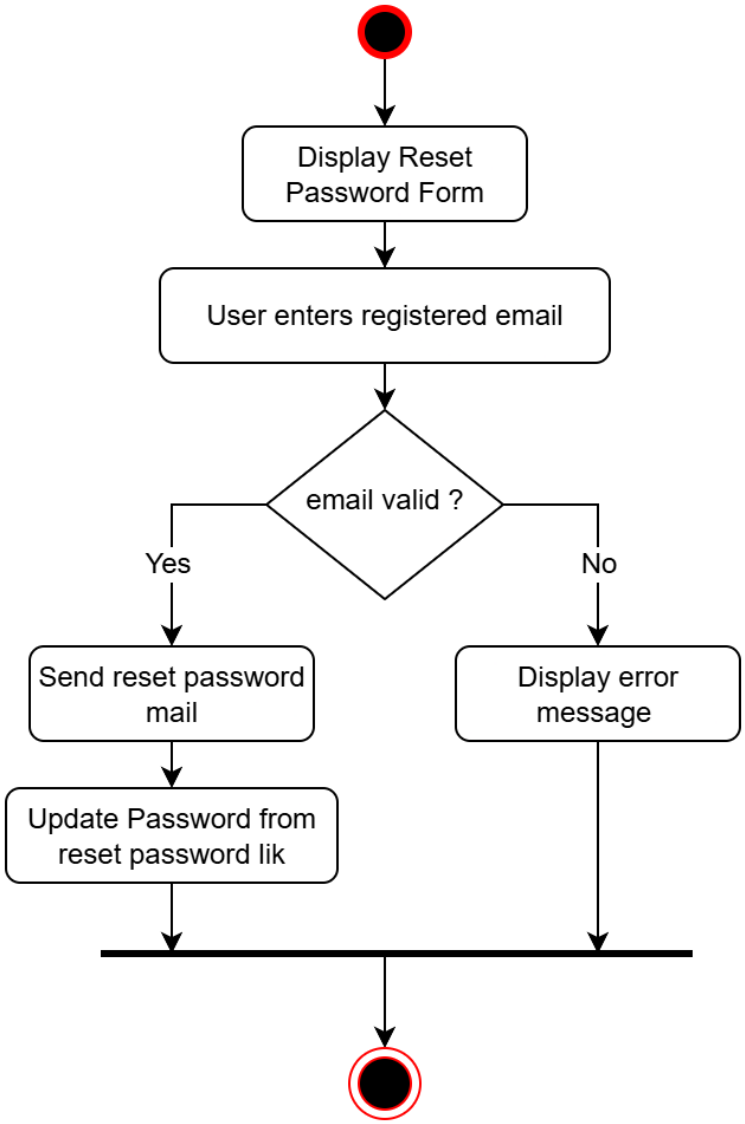
c.1 Registration Activity Diagram



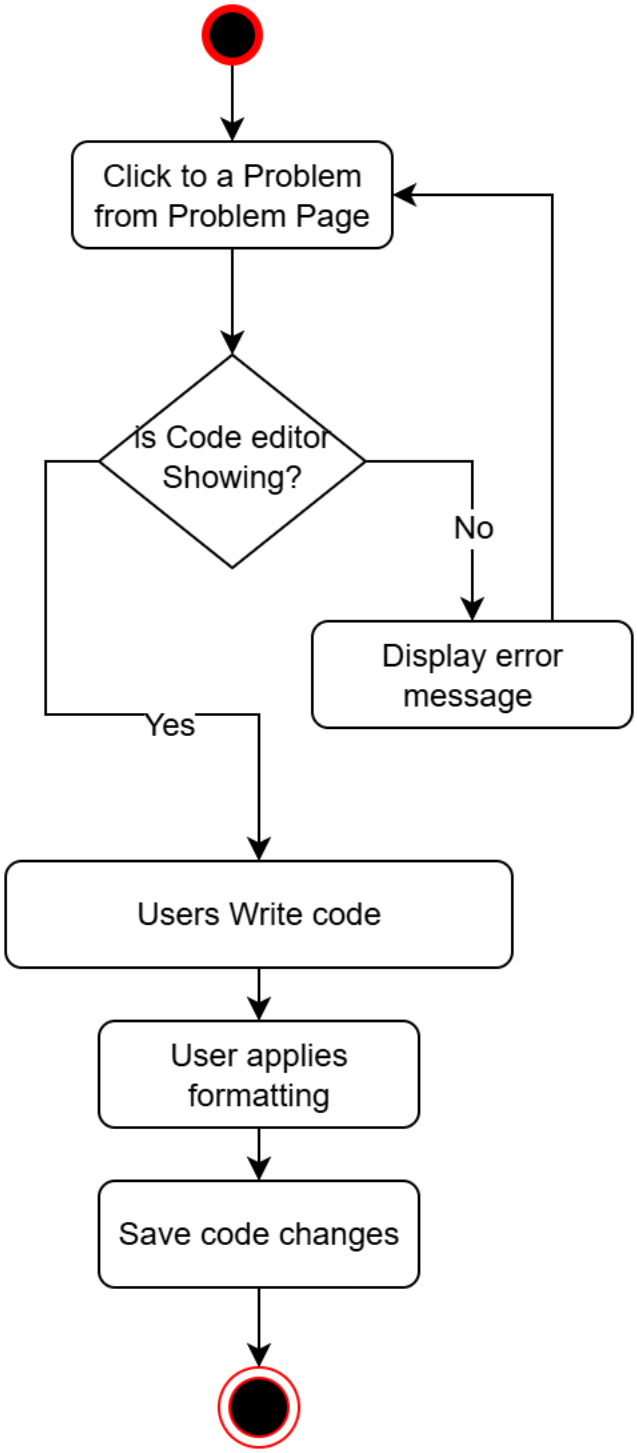
c.2 Login Activity Diagram



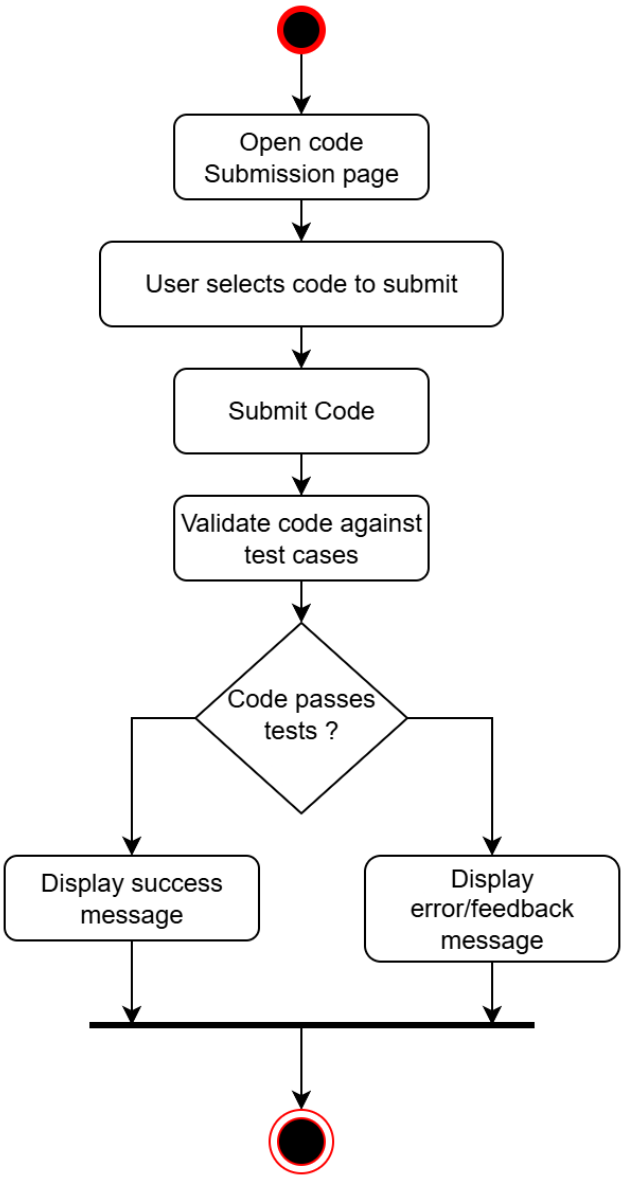
c.3 Reset Password Activity Diagram



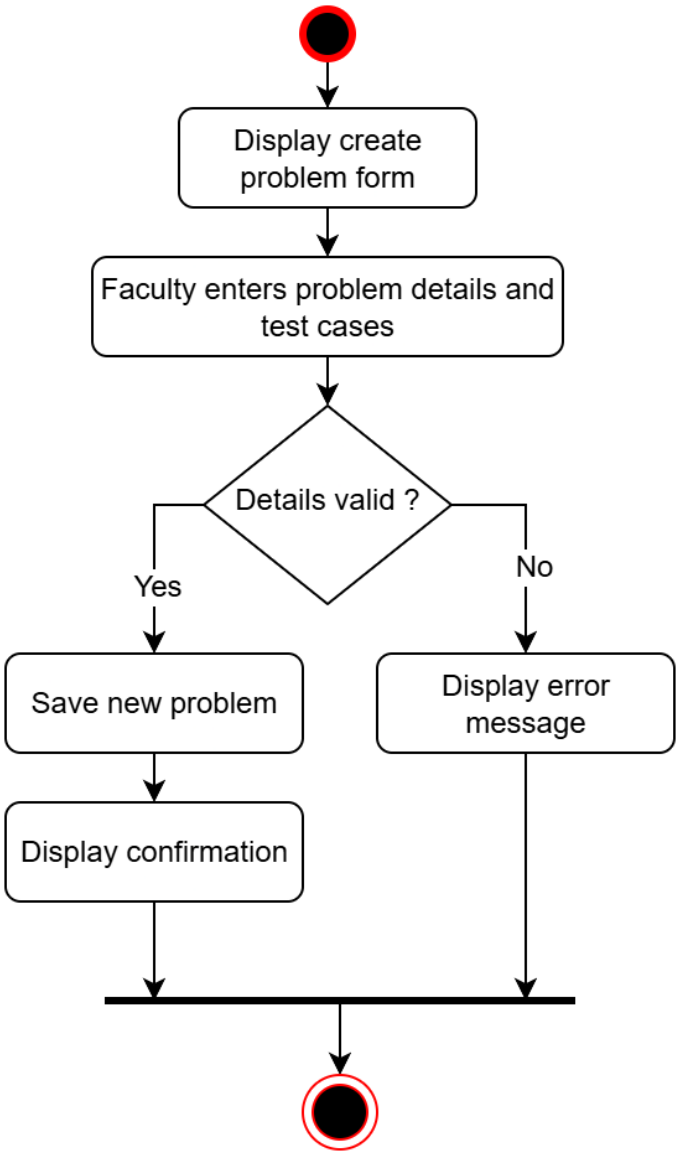
c.4 Code Editor Activity Diagram



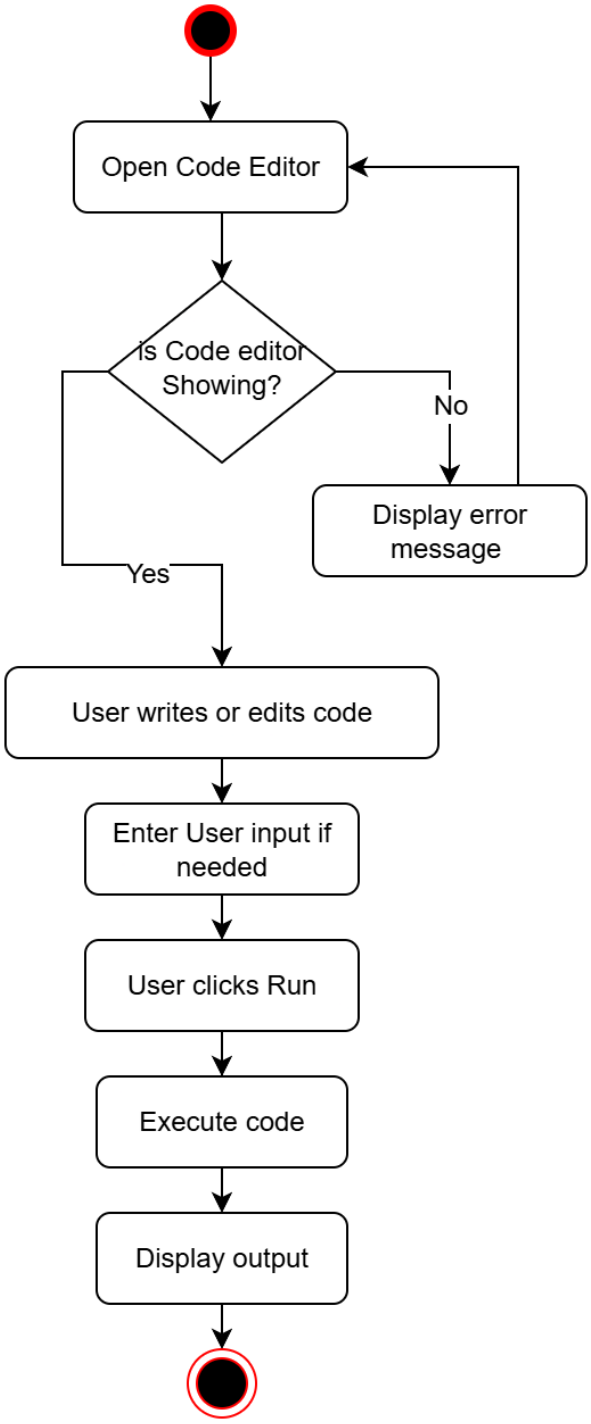
c.5 Code Submission Activity Diagram



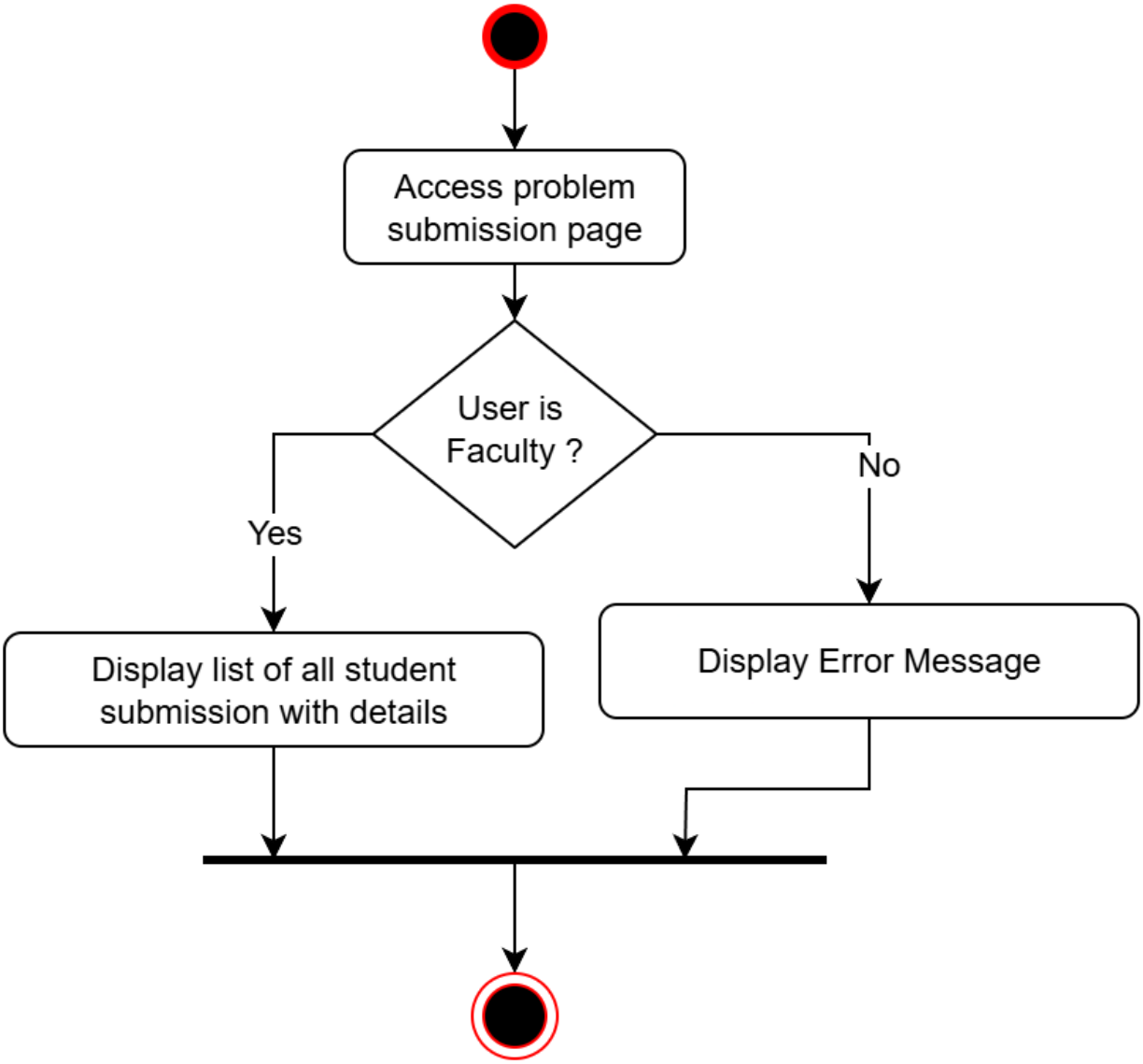
c.6 Create Problem Statements Activity Diagram



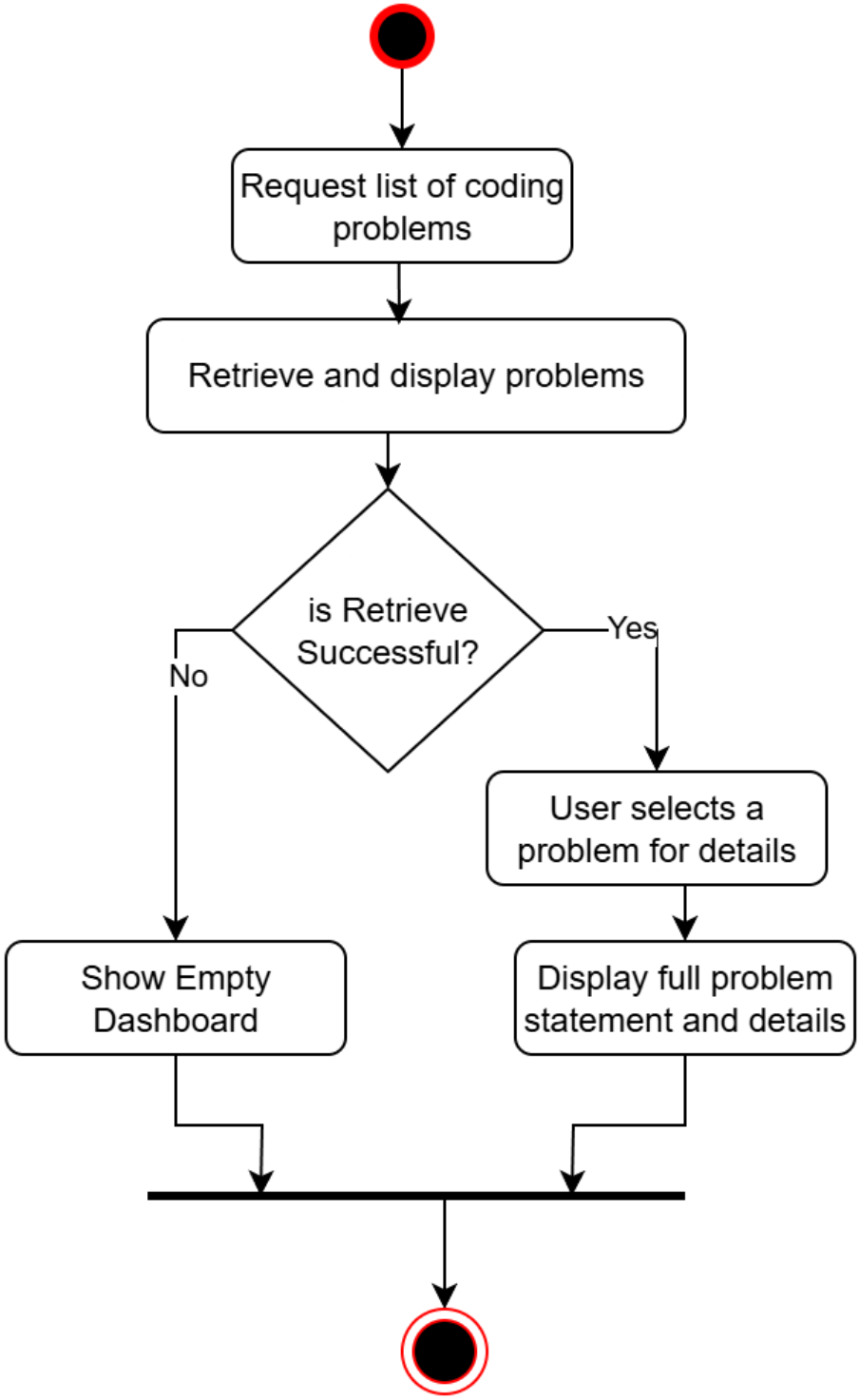
c.7 Run Code Activity Diagram



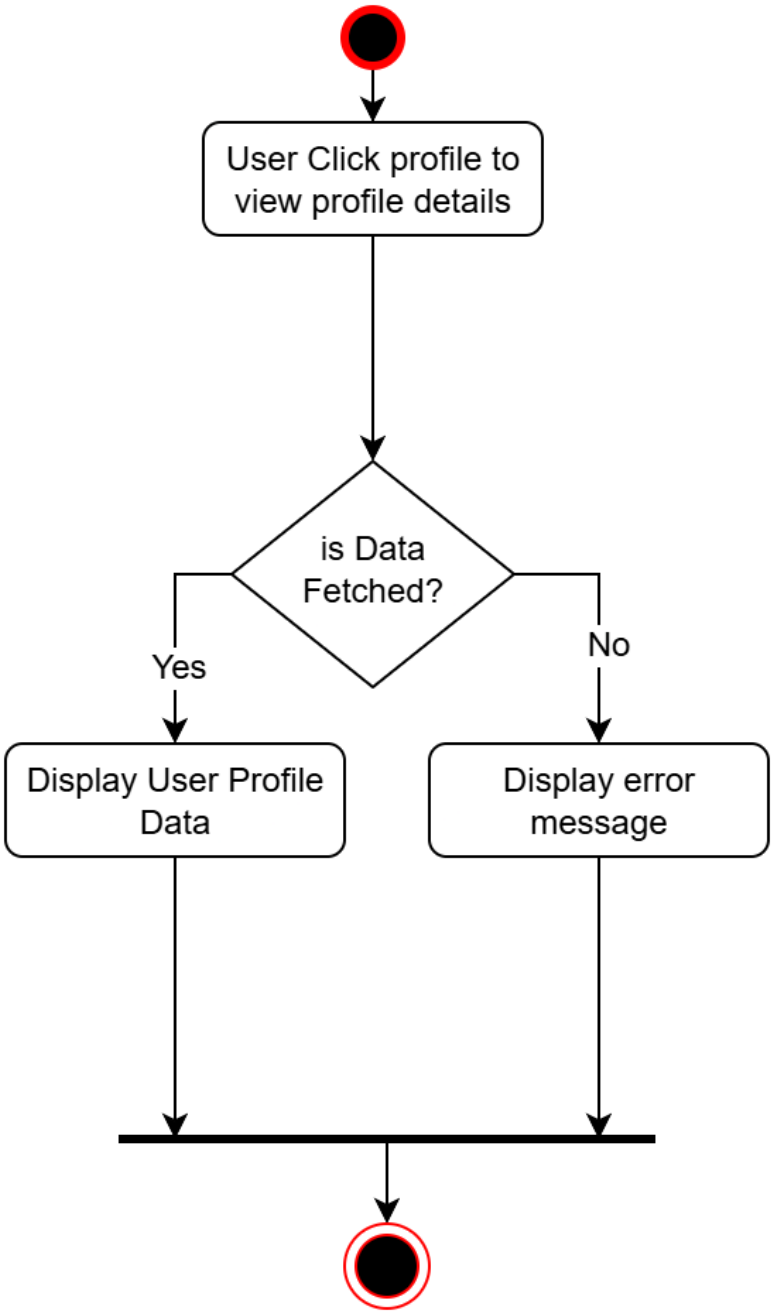
c.8 View Student Submission Activity Diagram



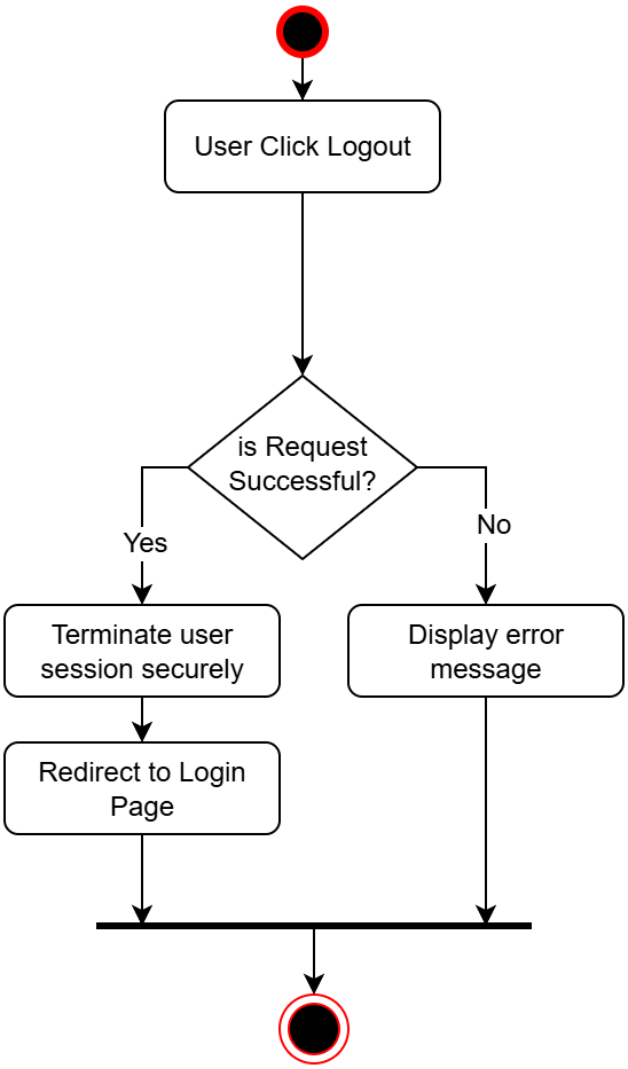
c.9 View Problem Details Activity Diagram



c.10 View Profile Activity Diagram

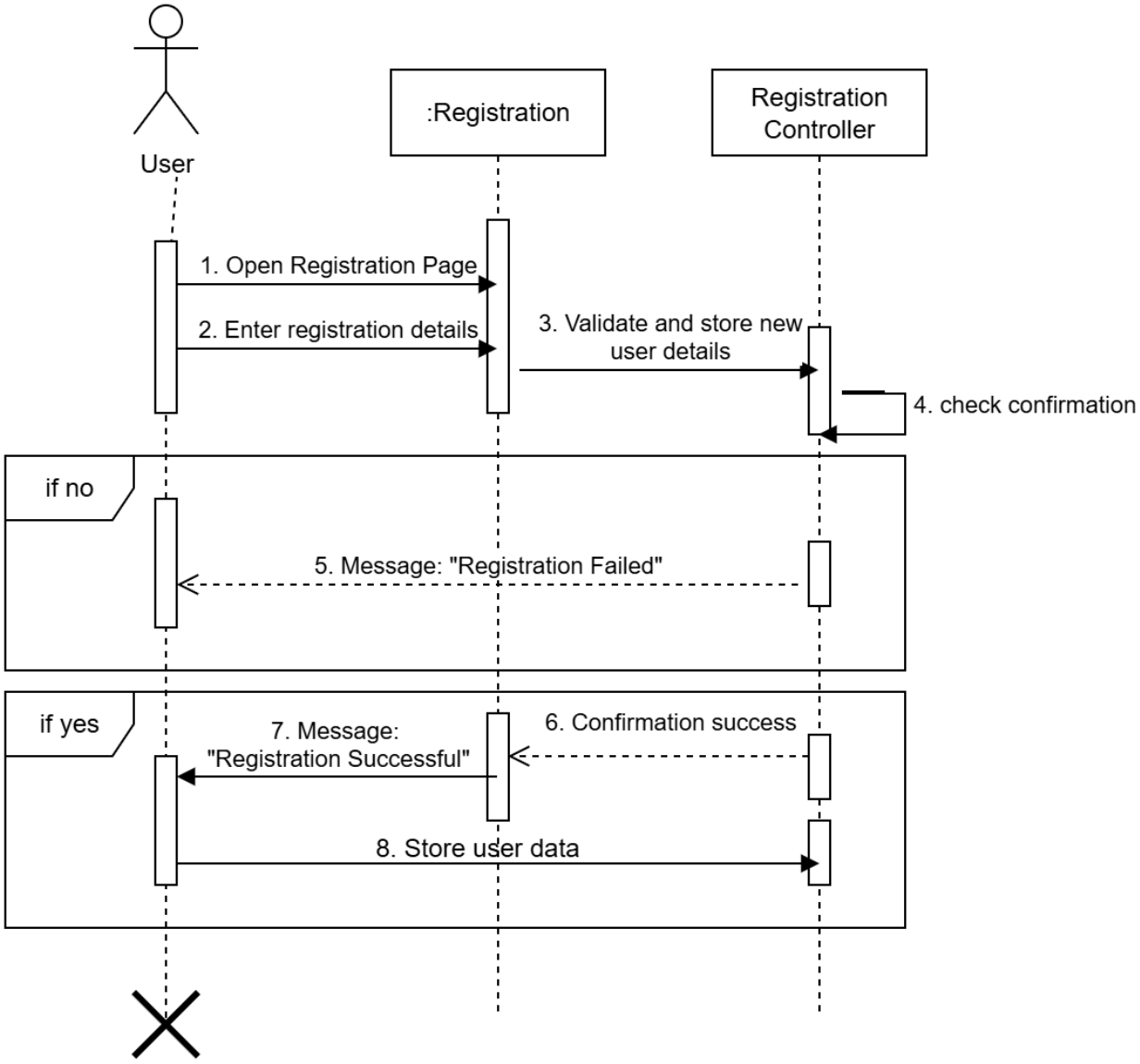


c.11 Logout Activity Diagram

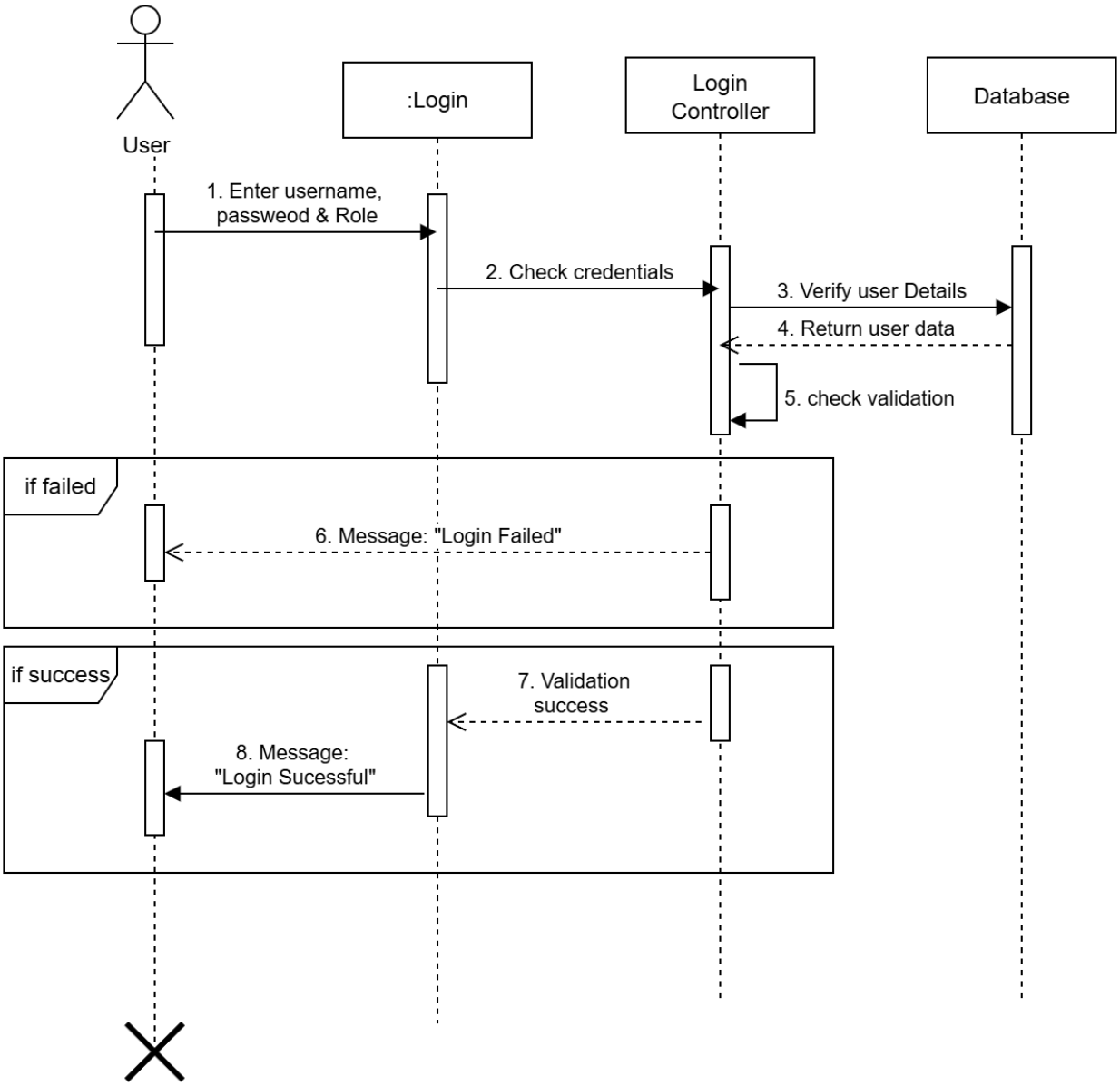


d. Sequence Diagram

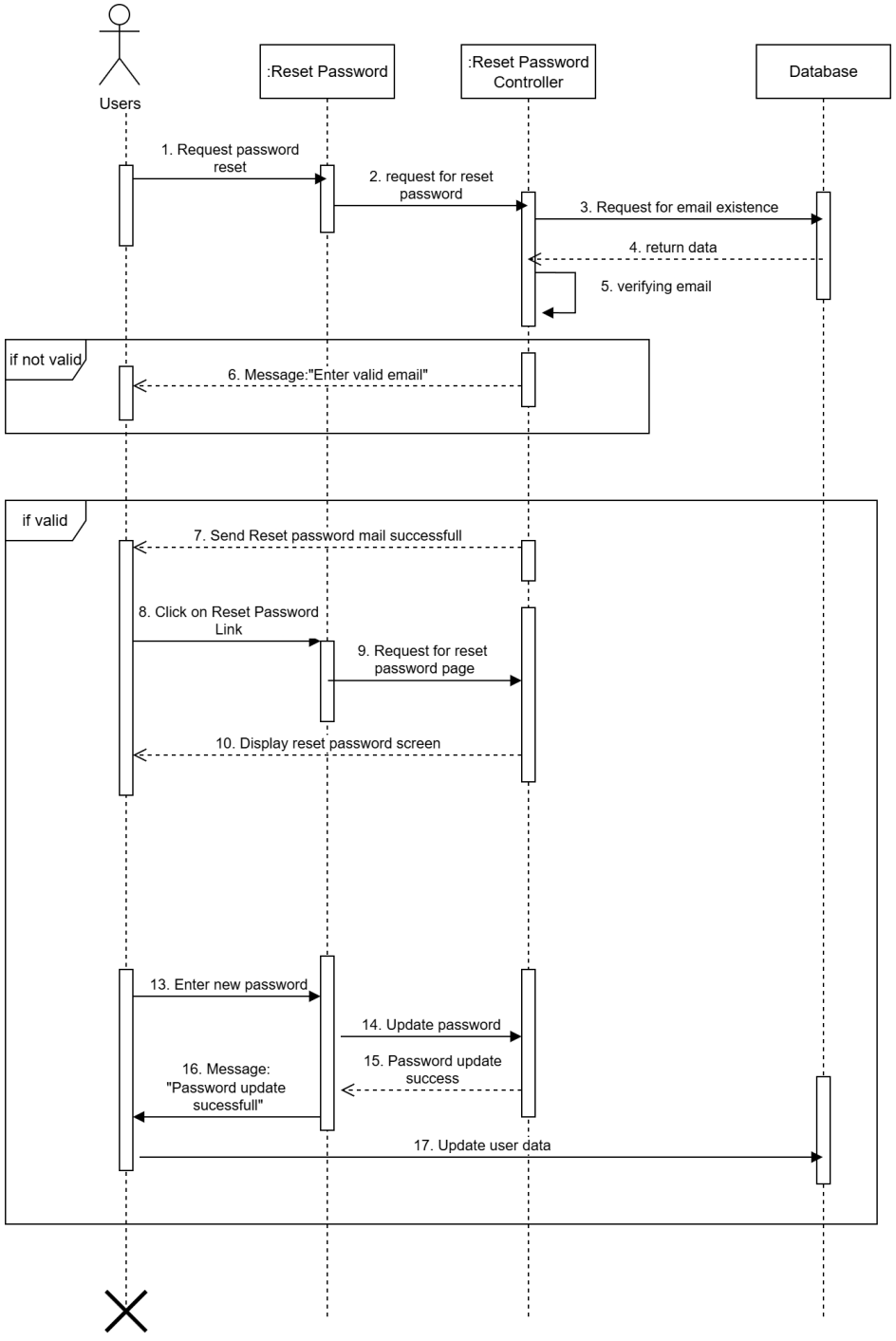
d.1 Registration Sequence Diagram



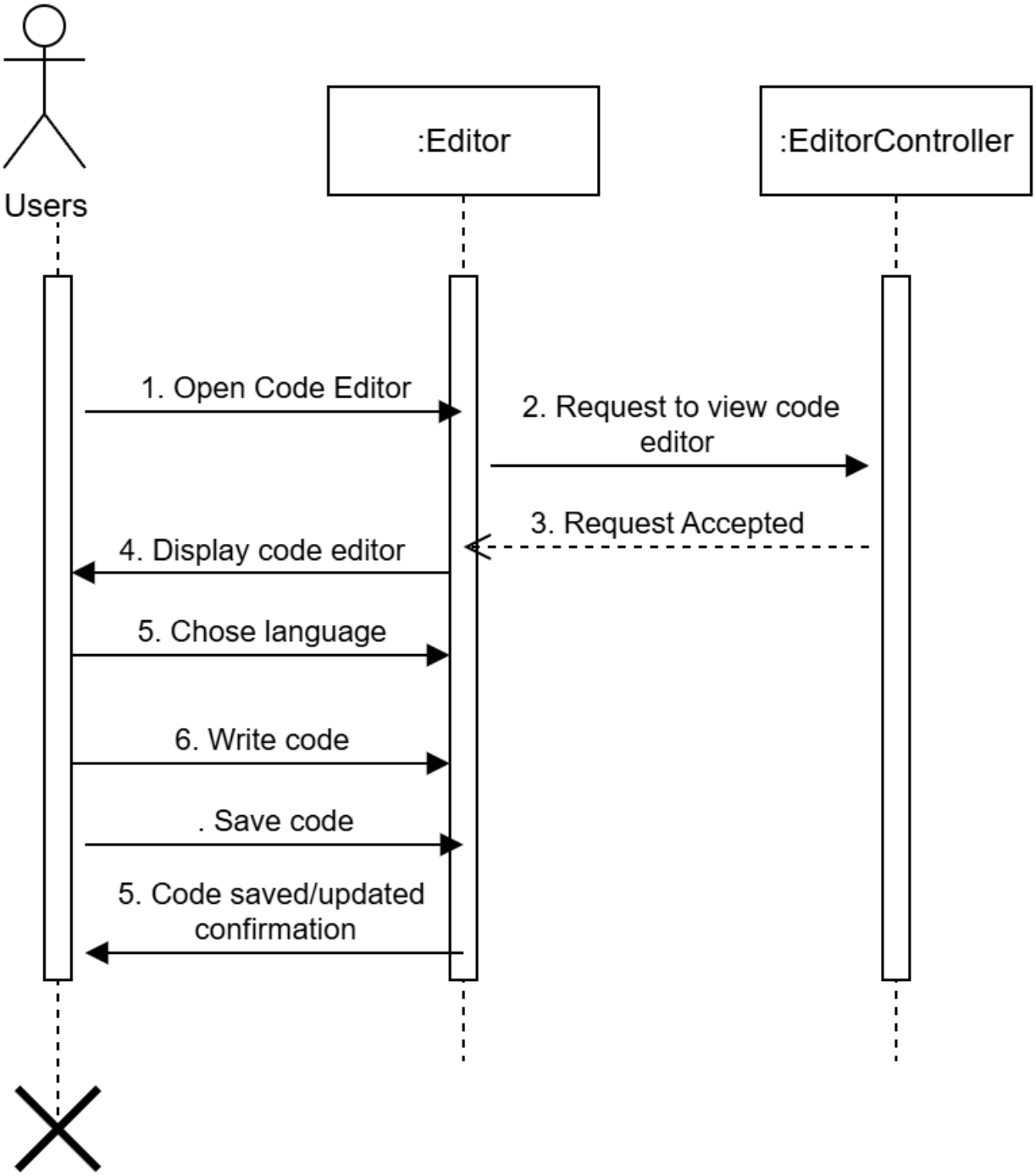
d.2 Login Sequence Diagram



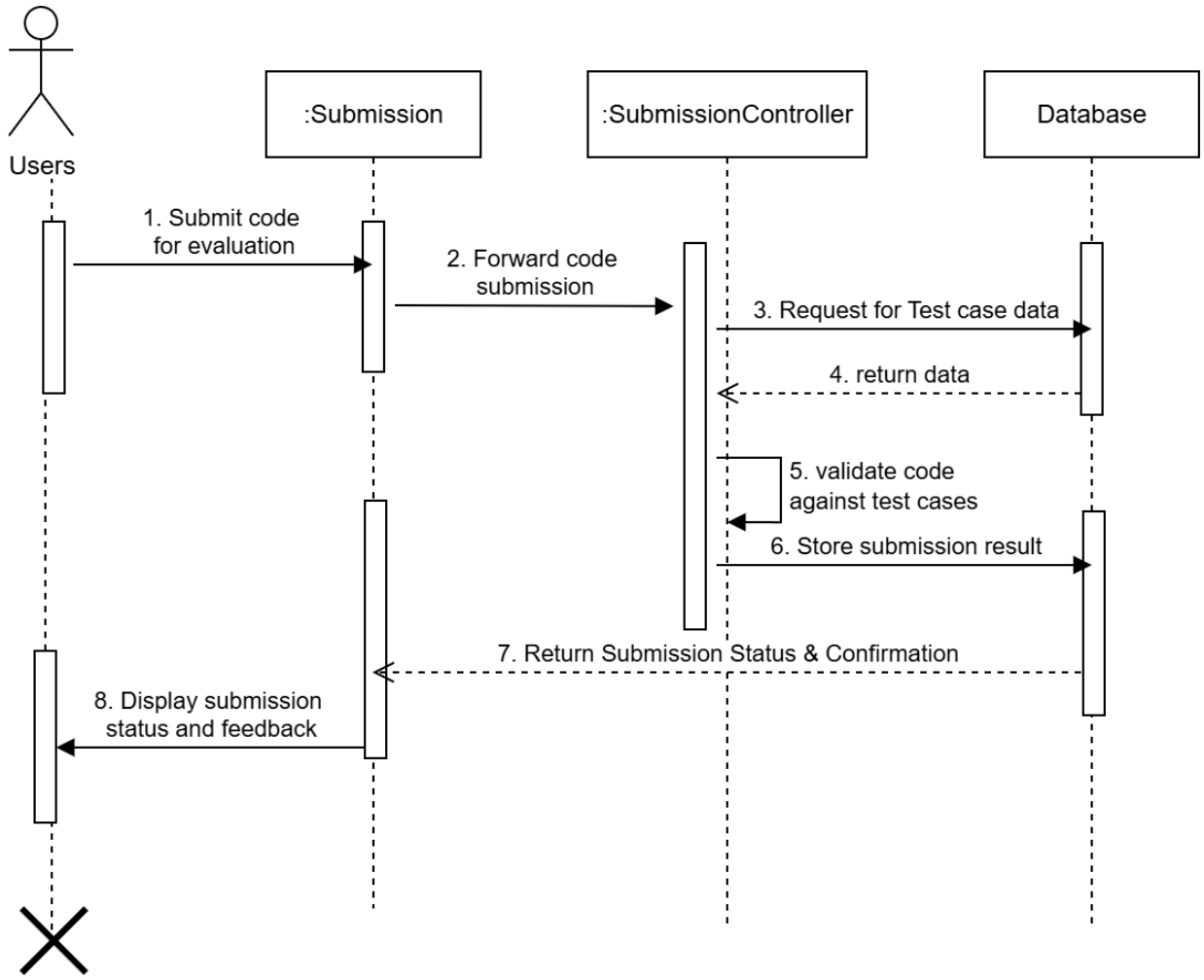
d.3 Reset Password Sequence Diagram



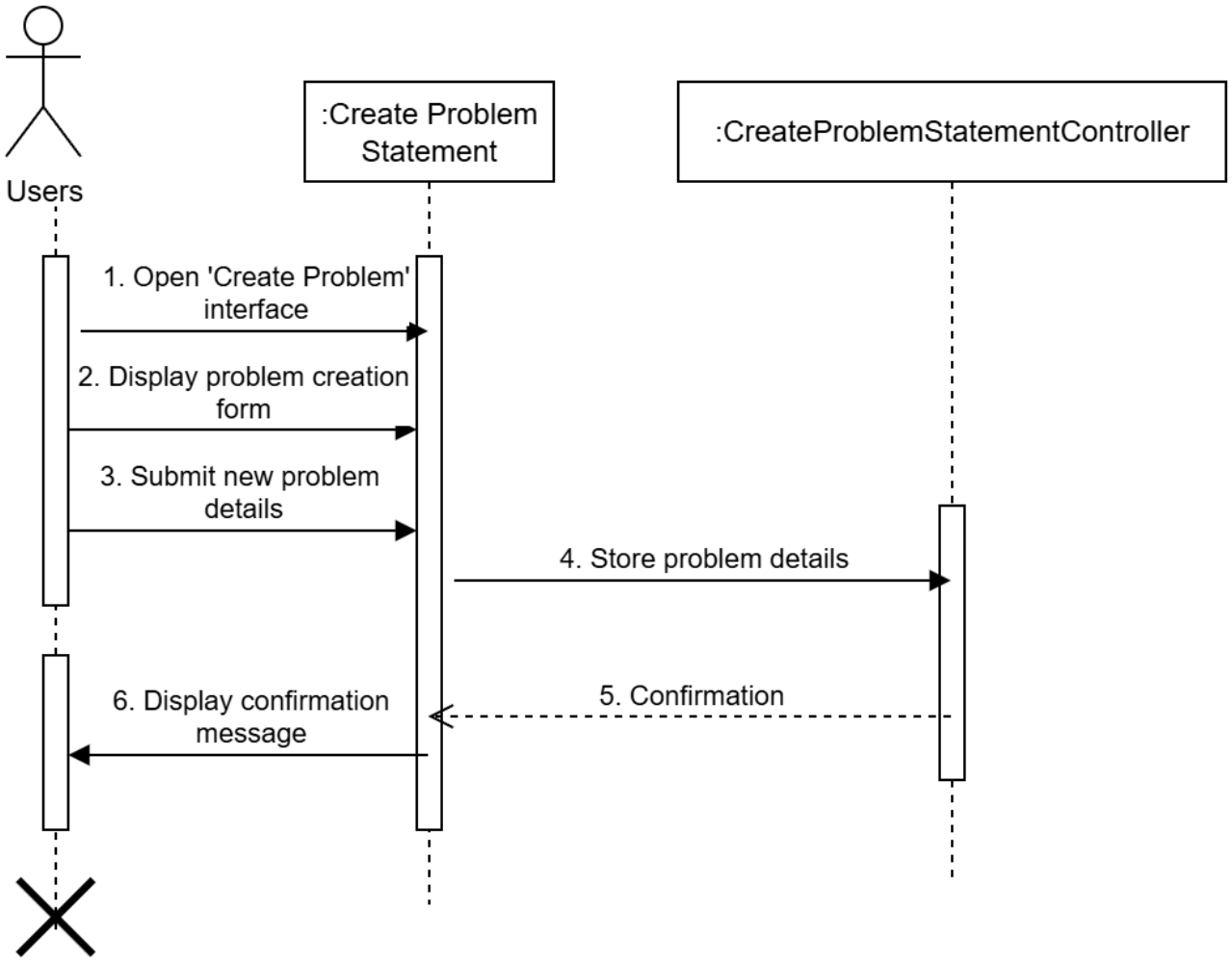
d.4 Code Editor Sequence Diagram



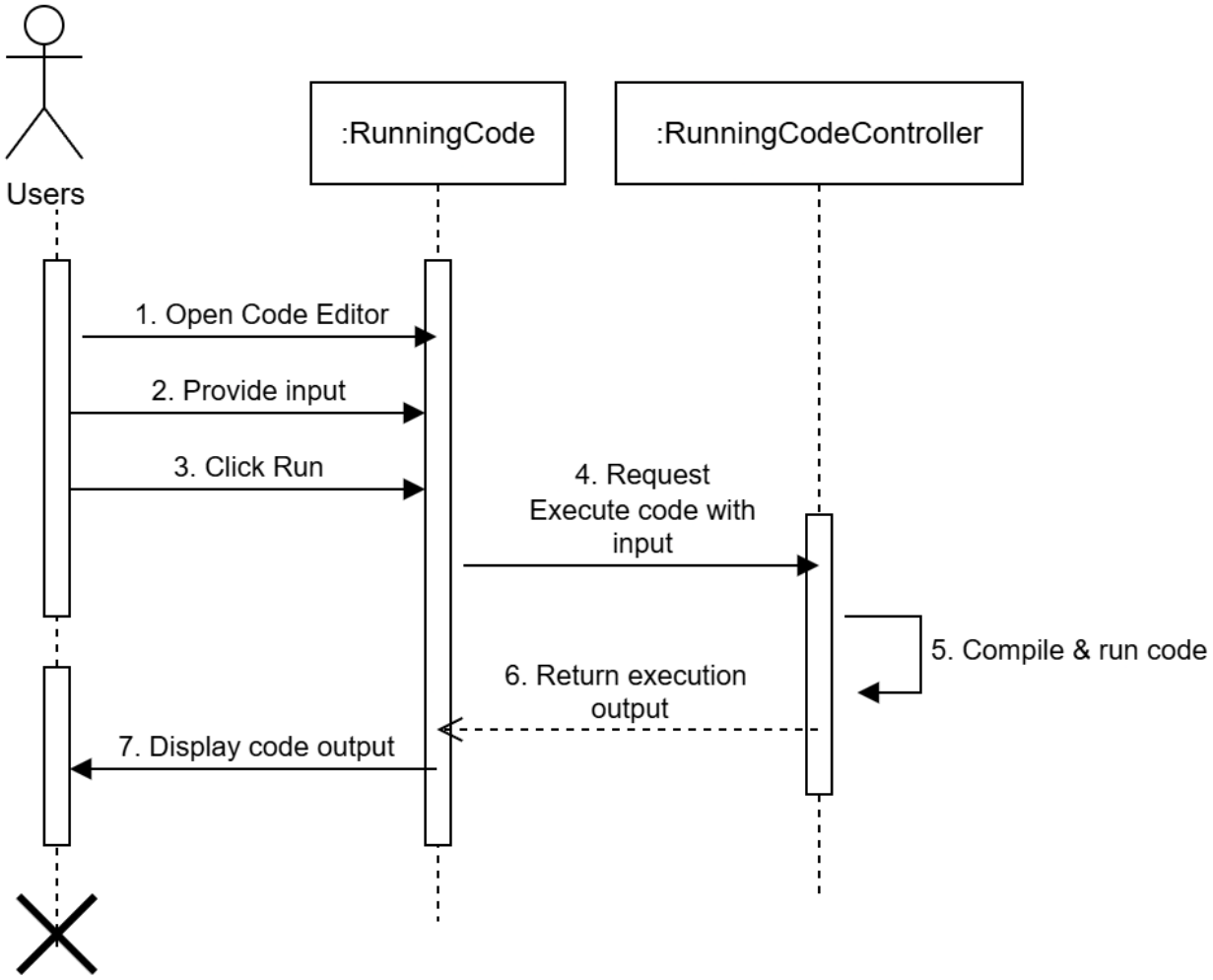
d.5 Code Submission Sequence Diagram



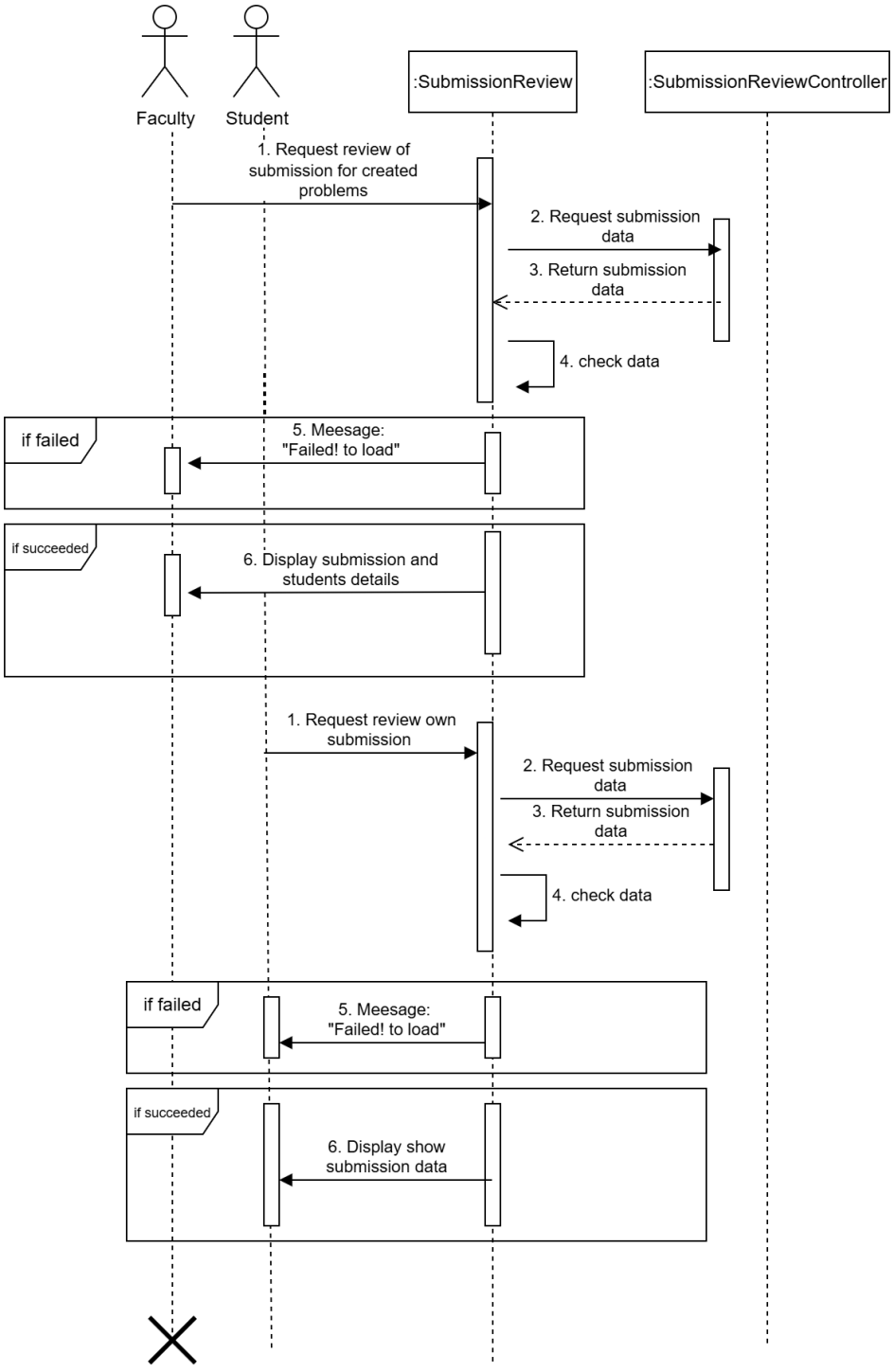
d.6 Create Problem Statements Sequence Diagram



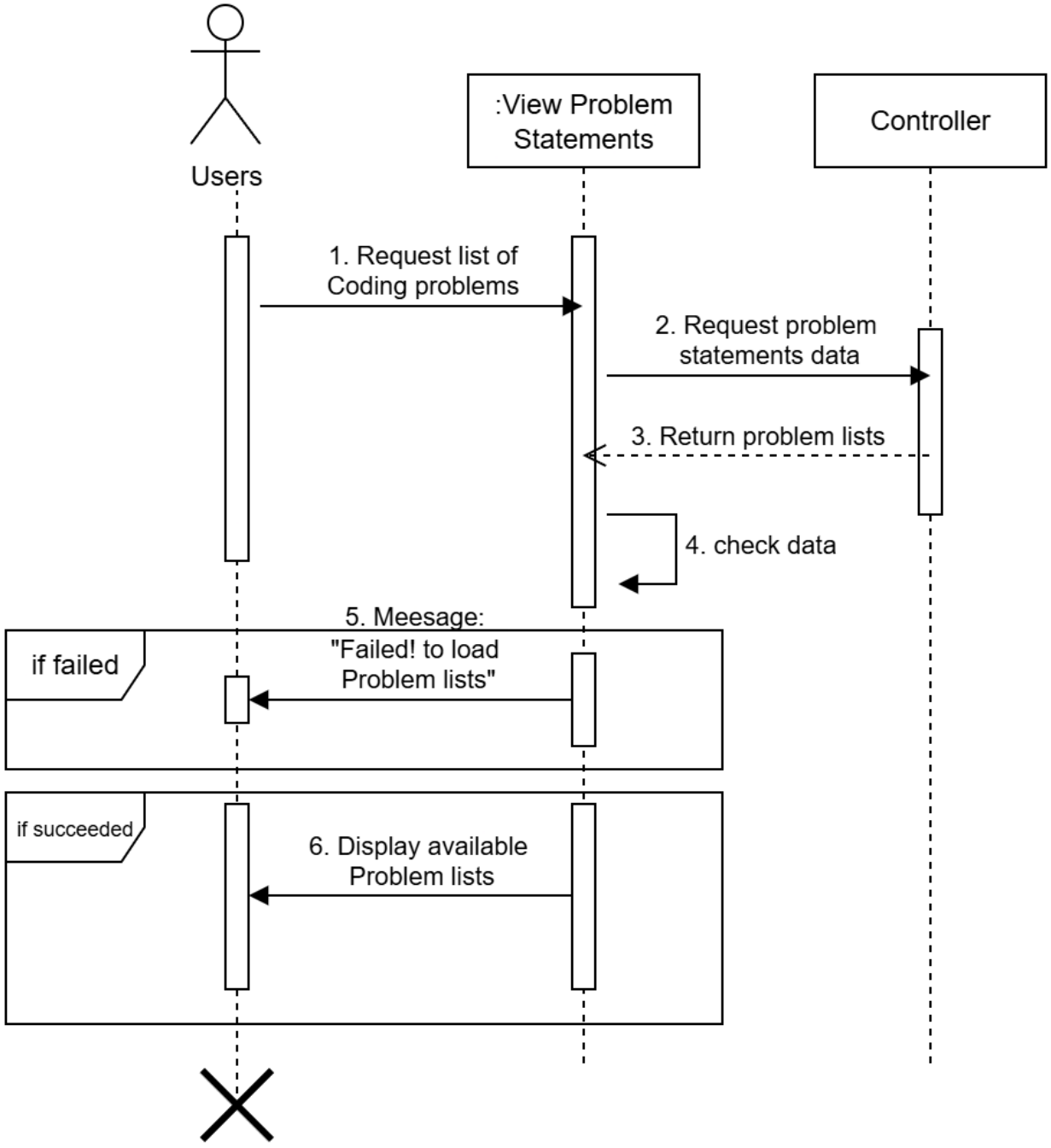
d.7 Run Code Sequence Diagram



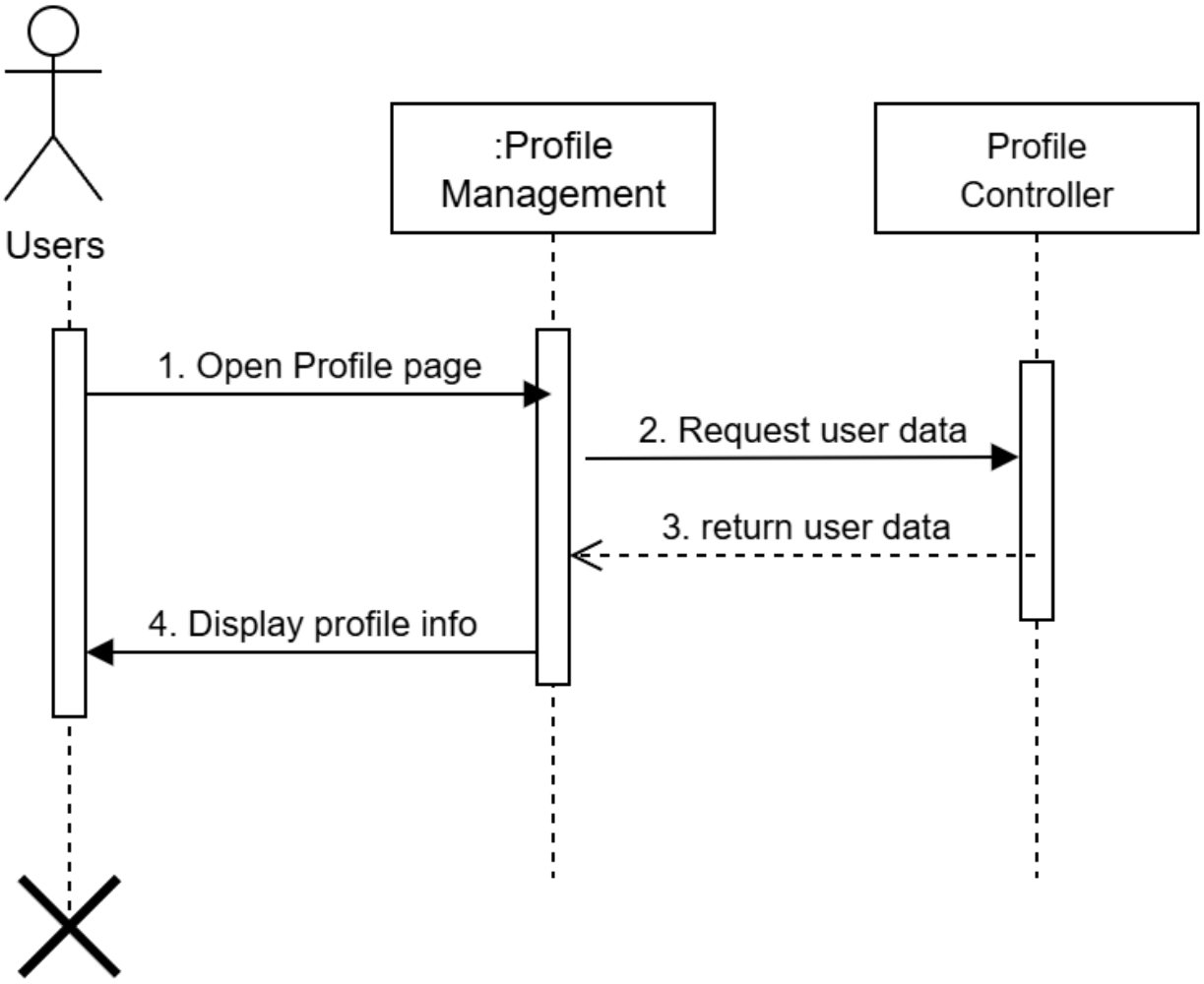
d.8 View Student Submission Sequence Diagram



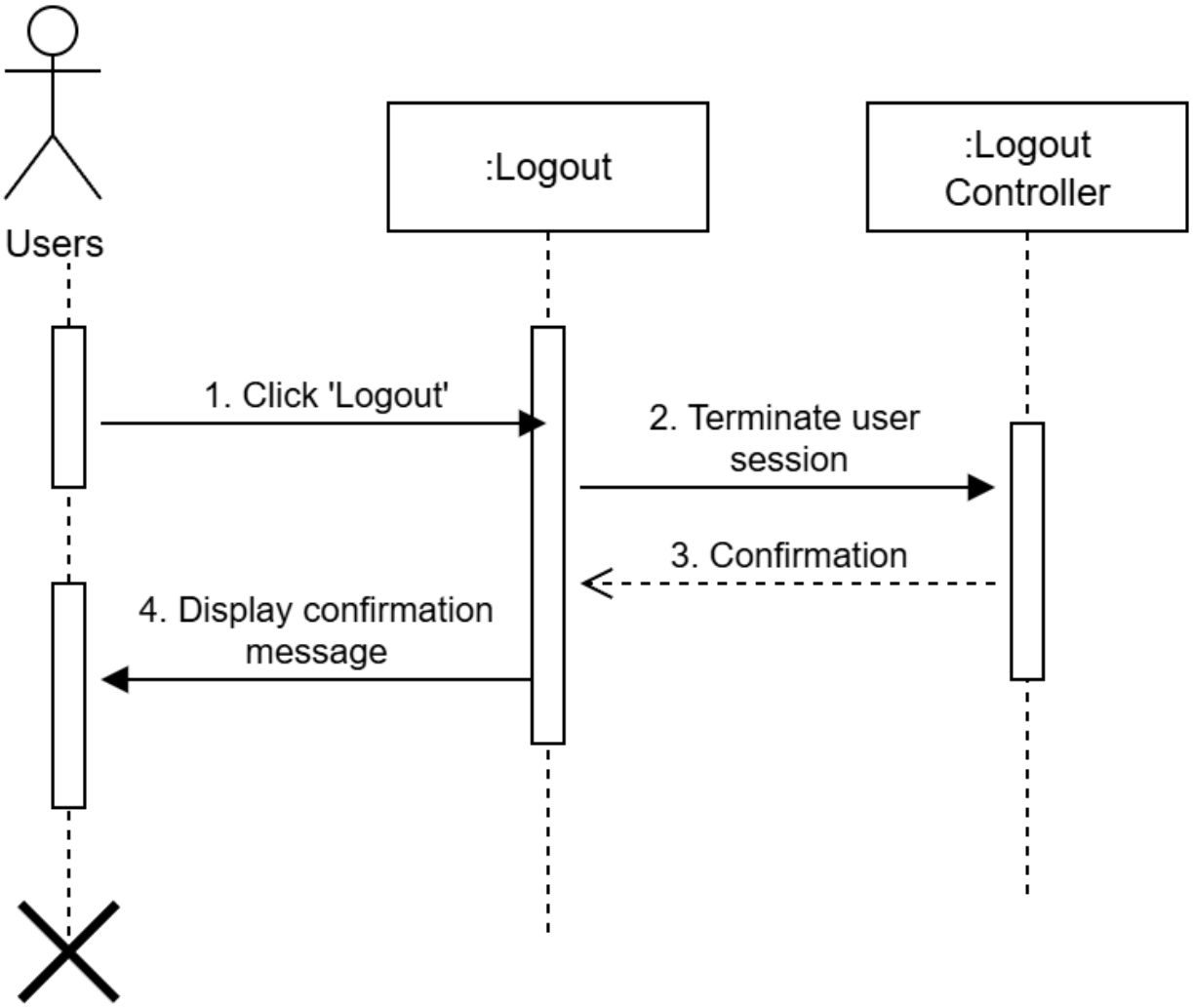
d.9 View Problem Details Sequence Diagram



d.10 View Profile Sequence Diagram



d.11 Logout Sequence Diagram



e. ER Diagram

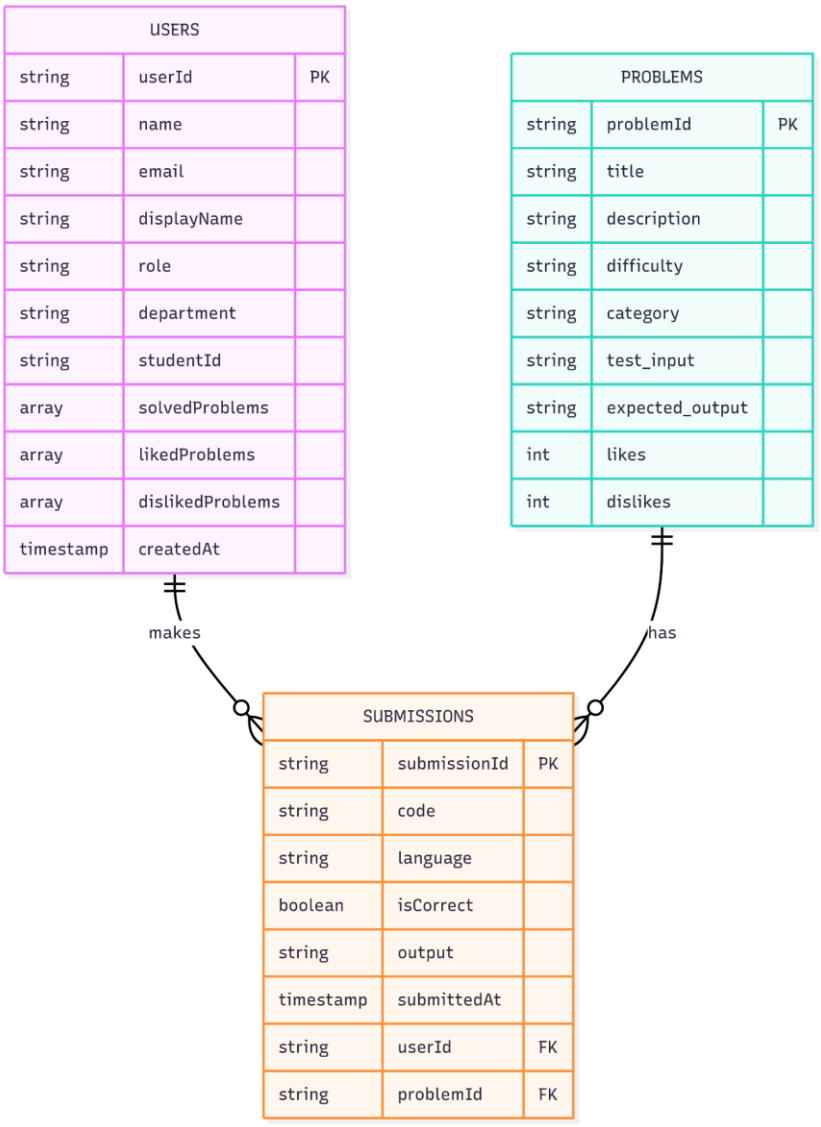


Figure 2.2 : ER Diagram

Software Testing

3.1 Testing Features

This section outlines the core features of the BrainBox IDE that were tested to ensure the application meets its functional requirements. Testing was focused on verifying the complete workflow for both Student and Faculty roles.

Feature Area	Specific Functionality Tested
User Authentication	User Registration, User Login & Logout, Role-Based Access, Unauthorized Access
Problem Management (Faculty)	Create New Problem, Read/Load Existing Problem, Update Existing Problem
Problem Solving (Student)	View Problem List, View Problem Details, Code Submission & Evaluation
Submission Tracking (Faculty)	View Submission List, View Submission Details

3.2 Testing Strategies

- **Unit Testing:** Ensured individual React components rendered correctly.
- **Integration Testing:** Verified that components like the problem list and the IDE modal interacted correctly.
- **Manual E2E Testing:** Performed end-to-end user workflow tests for both Student and Faculty roles.

3.3 System Testing

This section provides a more comprehensive set of test cases for key system functionalities to verify that they behave as expected under various conditions.

Test Case ID	Feature	Test Steps	Expected Result	Status
TC-01	**(Positive)** User Registration	1. Fill out the signup form with valid data. 2. Click "Register".	The user is created in the database and redirected to the login page.	**Pass**
TC-02	**(Negative)** User Registration - Email Exists	1. Fill out the signup form with an email that is already registered. 2. Click "Register".	An error message "Email already in use" is displayed.	**Pass**
TC-03	**(Positive)** User Login	1. Enter a valid email and password. 2. Select the correct role. 3. Click "Log In".	The user is redirected to their corresponding dashboard.	**Pass**
TC-04	**(Negative)** User Login - Invalid Password	1. Enter a valid email and an incorrect password. 2. Click "Log In".	An error message "Invalid credentials" is displayed.	**Pass**
TC-05	**(Positive)** Create Problem	1. Log in as Faculty. 2. Fill out the "Manage Problems" form. 3. Click "Save".	A success notification appears, and the new problem is saved in the database.	**Pass**
TC-06	**(Positive)** Update Problem	1. Log in as Faculty. 2. Load an existing problem. 3. Change the problem's title. 4. Click "Save".	A success notification appears, and the problem is updated in the database.	**Pass**
TC-07	**(Positive)** Submit Correct Code	1. Log in as a Student. 2. Select a problem. 3. Submit a correct solution.	A success notification appears, and the submission is marked as "Correct".	**Pass**
TC-08	**(Negative)** Submit	1. Log in as a Student. 2. Select a problem.	A notification appears, and the submission is	**Pass**

	Incorrect Code	3. Submit a solution that produces the wrong output.	marked as "Incorrect".	
TC-09	**(Negative)** Submit Code with Error	1. Log in as Student. 2. Submit code that will not compile	The output shows a "Compilation Error" message.	**Pass**
TC-10	**(Positive)** View Submission Details	1. Log in as Faculty. 2. Go to the "Student Submissions" list. 3. Click on a submission.	A modal appears showing the student's code and the detailed output.	**Pass**
TC-11	**(Positive)** Reset Password	1. Click "Forgot Password?". 2. Enter a registered email address. 3. Click "Reset Password".	A confirmation message appears, and a password reset email is sent.	**Pass**
TC-12	**(Negative)** Unauthorized Access	1. Log out of the system. 2. Attempt to navigate directly to the admin dashboard URL.	The user is redirected back to the login page.	**Pass**

Deployment and Maintenance

4.1 Software Release Life Cycle

The project followed an Agile, iterative approach.

- **Alpha Stage:** Focused on core backend setup (Firebase Auth, Firestore rules) and basic UI components.
- **Beta Stage:** Implemented the full student problem-solving workflow and the faculty dashboard. This version was shared with a small group of peers for initial feedback.
- **Release Stage:** After fixing bugs identified in beta, the final application was deployed.
- **Deployment:** The project is deployed on **Vercel**. The GitHub repository's main branch is linked to Vercel, enabling a Continuous Deployment pipeline where every git push to main automatically triggers a new live build.

- **Maintenance:** Post-deployment maintenance involves monitoring the Vercel and Firebase dashboards for errors and usage spikes. Future updates will be managed through Git branches and pull requests to ensure stability.

User Manual

5.1 For Students

5.1.1 Registration

From the dashboard users need to click Sign in and then choose Create account option to get the registration form. After filling the registration form properly they need to click the Register button to complete the process.

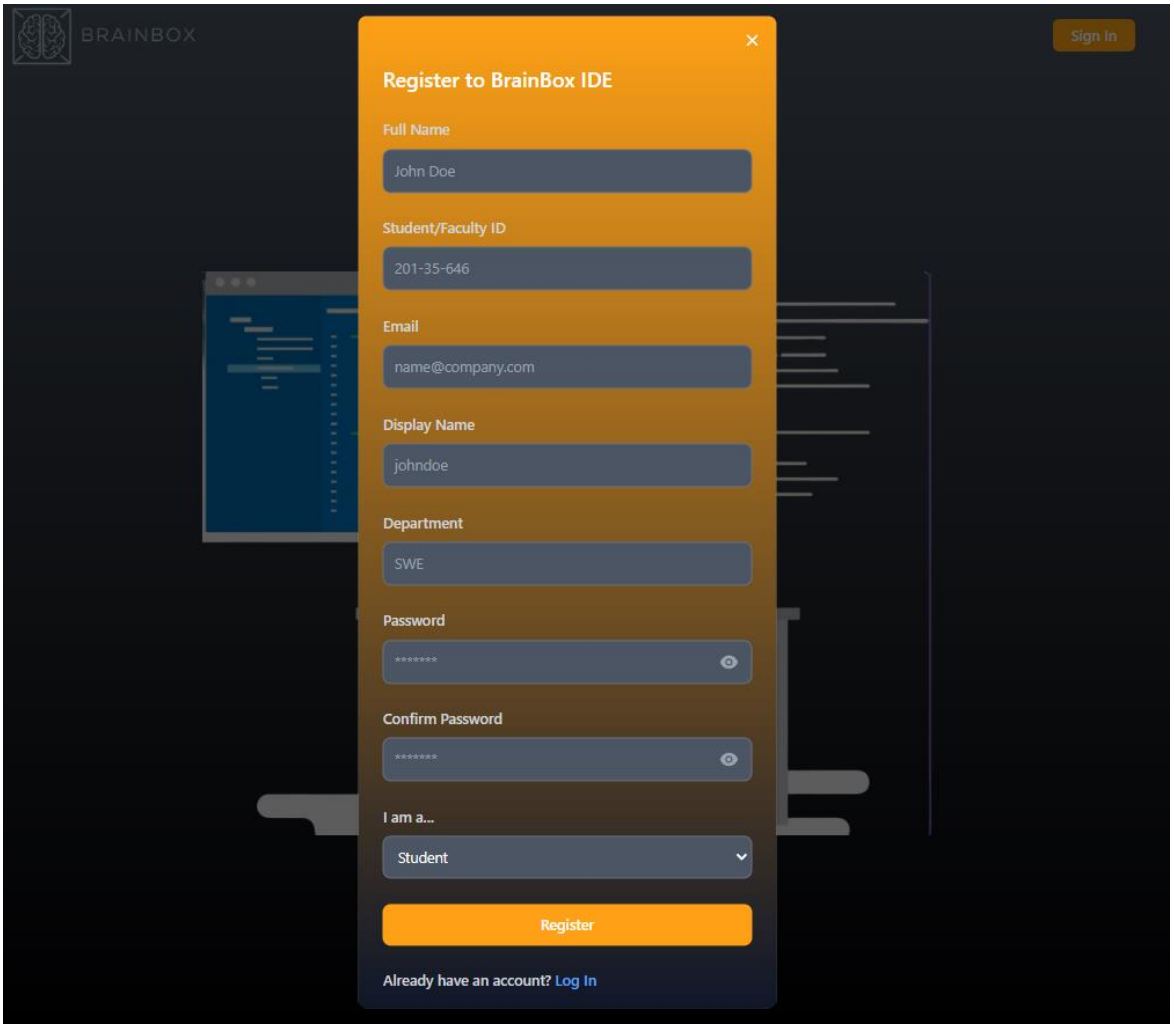


Figure 5.1 : Registration process (Student)

5.1.2 Login

From the dashboard users need to click Sign in and then they will get the sign in form. After filling the sign in form properly they need to click the Login button to complete the process.

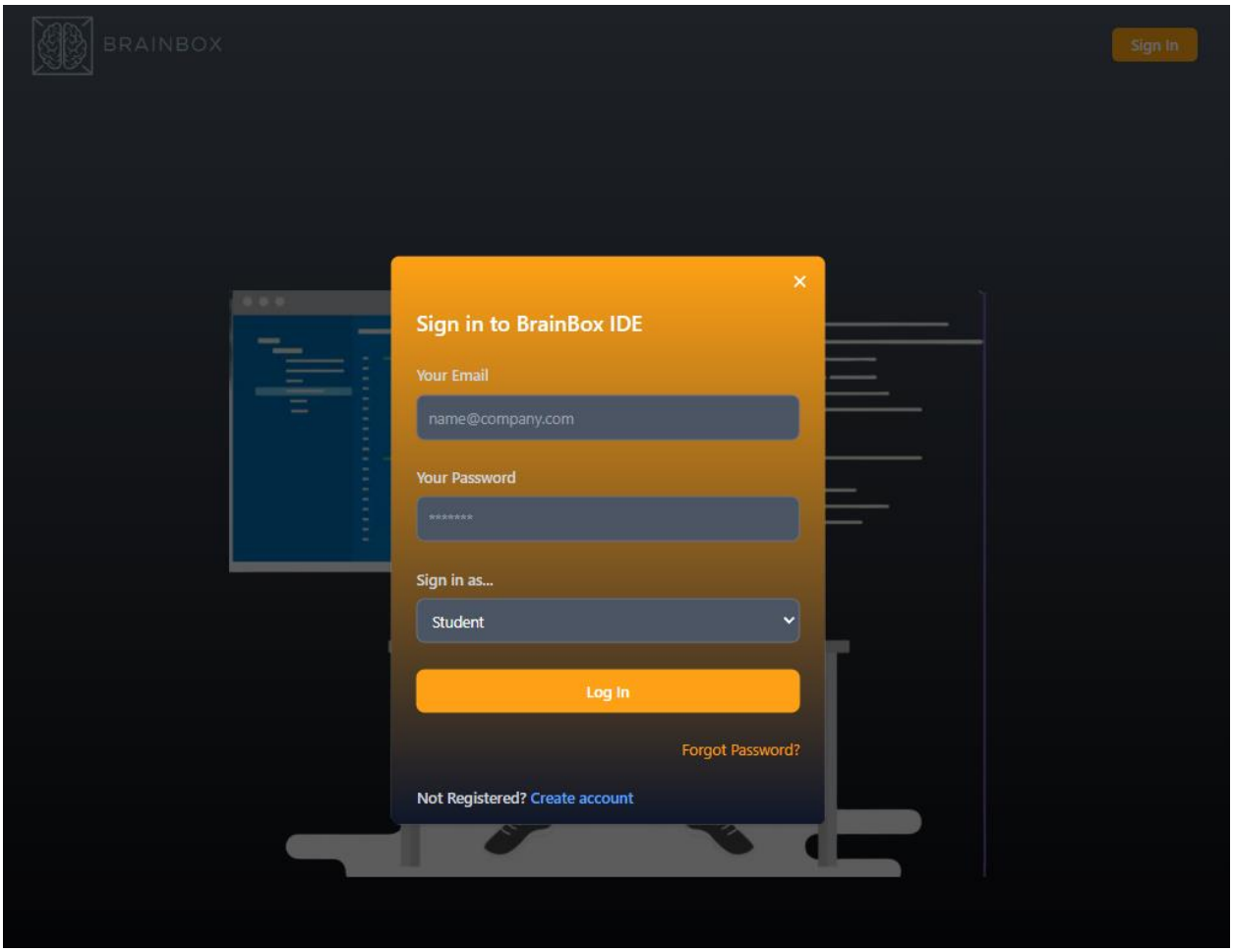


Figure 5.2 : Login process (Student)

5.1.3 Reset Password

If users forget their password they need to click forget password from the login page and they will get the reset password screen. After filling the field and clicking the reset password button with registered mail to the system they will receive the reset password link on their email.

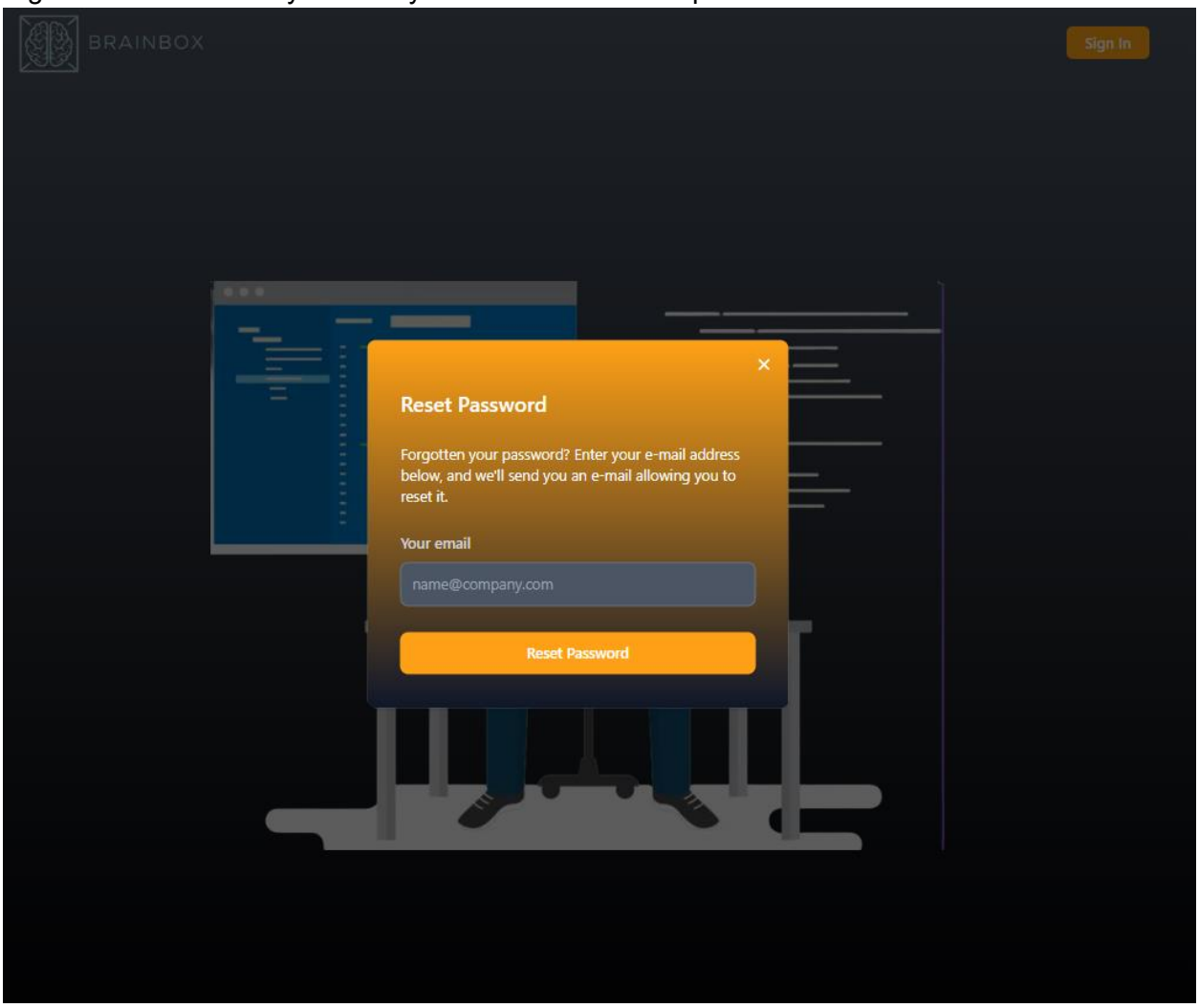


Figure 5.3 : Reset password process (Student)

5.1.4 Problem List

After Login or without login users can view the problem list screen.

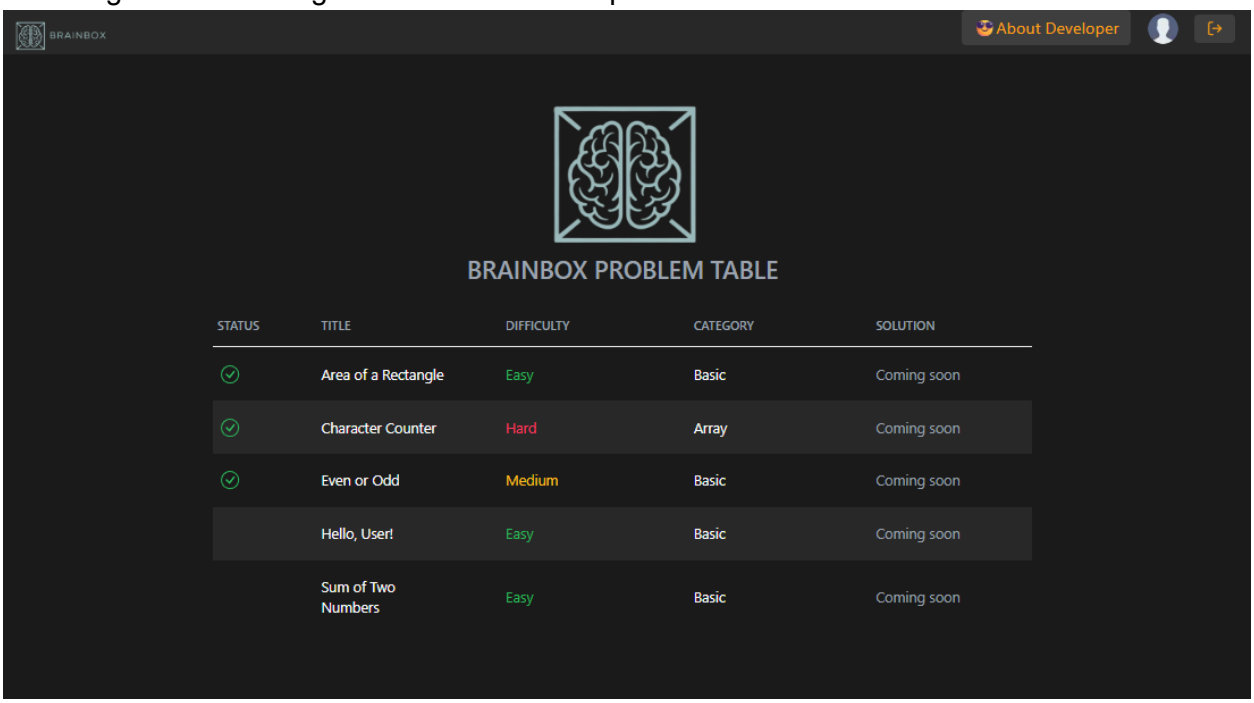


Figure 5.4 : Problem List

5.1.5 Problem Details and Code Editor

After login, users need to click which problem they want to solve. Then they will get the problem details and code editor screen. Here they can choose language from the drop down option in the top-right corner. They can also write code from the code editing option. There is run code option for code execution and a submit button for code submission.

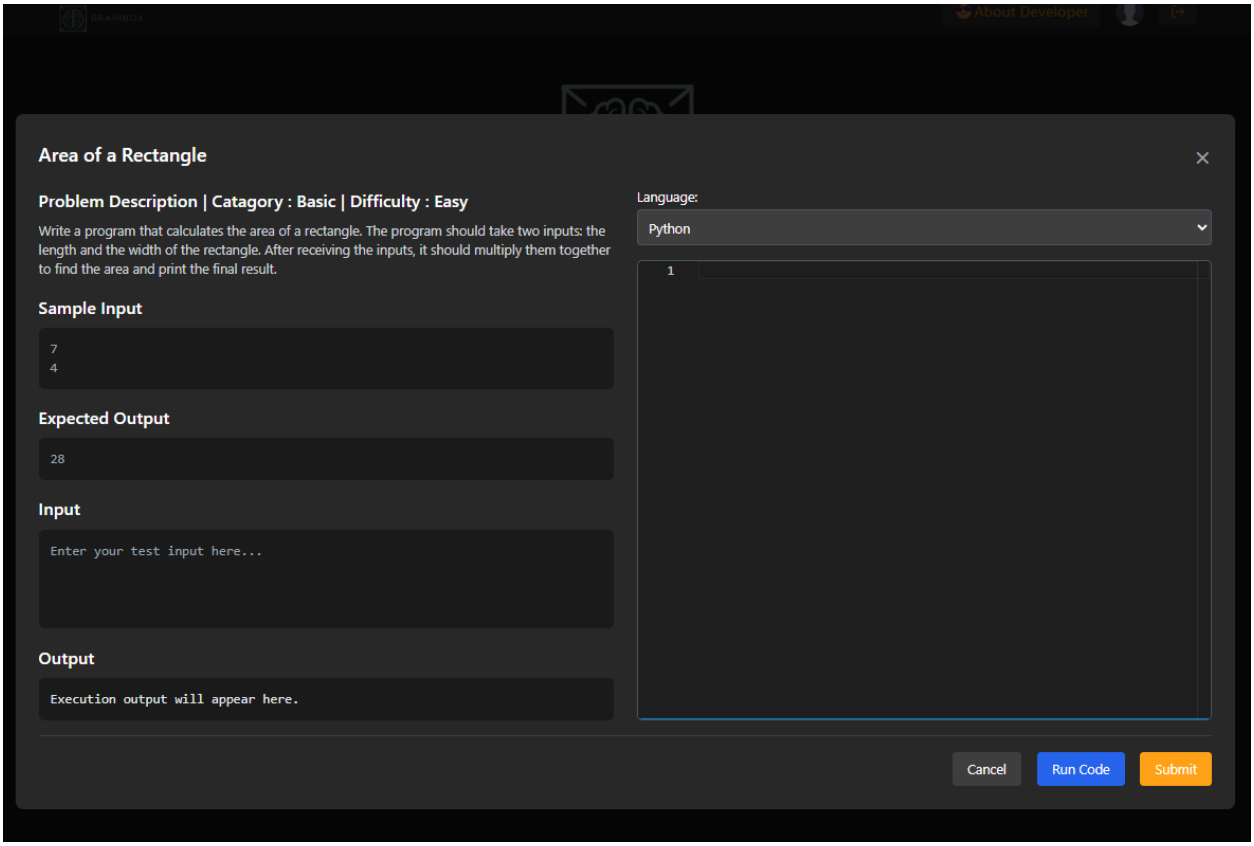


Figure 5.5 : Problem Details and Code Editor

5.2 For Faculty

5.2.1 Login

From the dashboard users need to click Sign in and then they will get the sign in form. After filling the sign in form properly they need to click the Login button to complete the process.

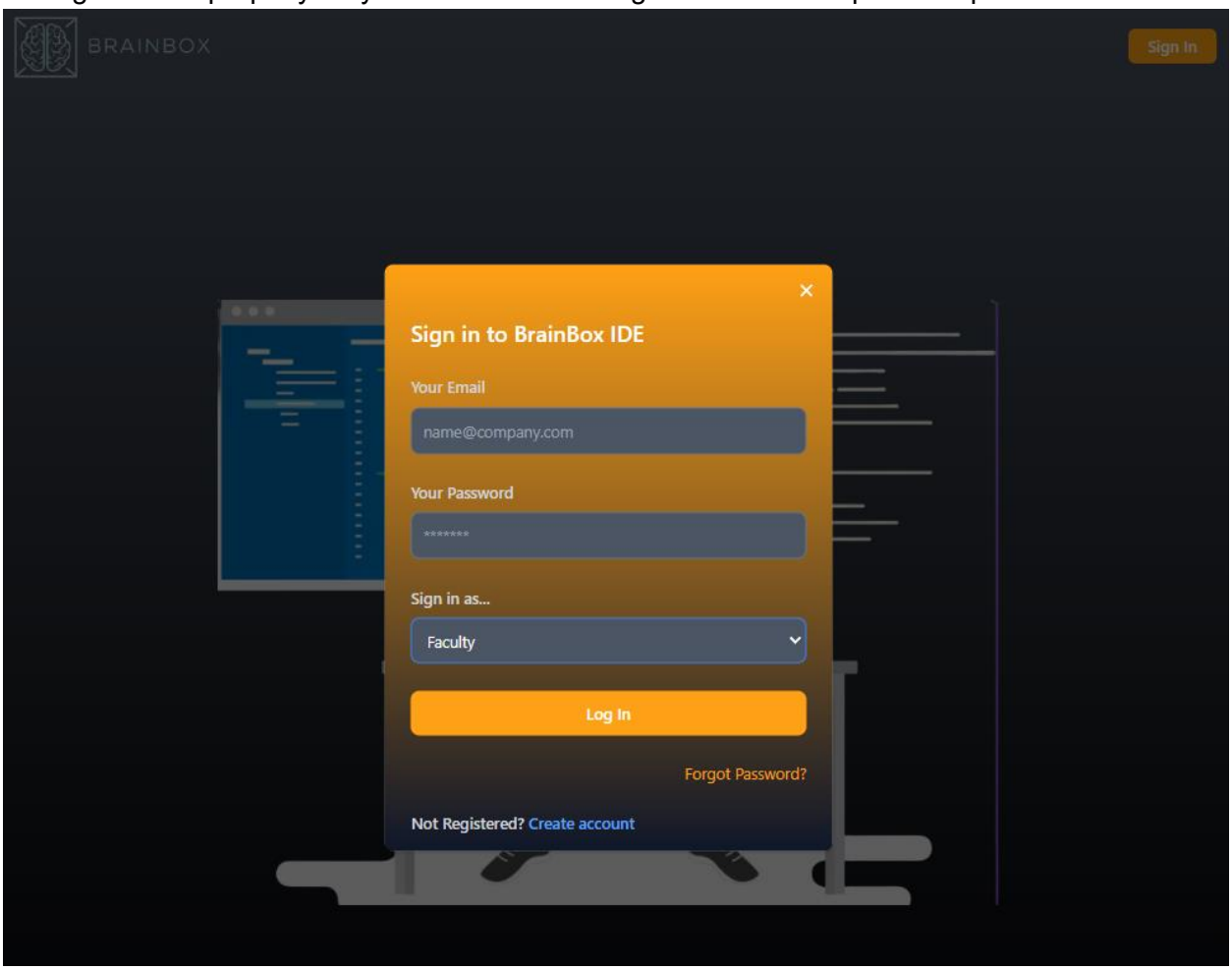


Figure 5.6 : Login Process (Faculty)

5.2.2 Registration

From the dashboard users need to click Sign in and then choose Create account option to get the registration form. After filling the registration form properly they need to click the Register button to complete the process.

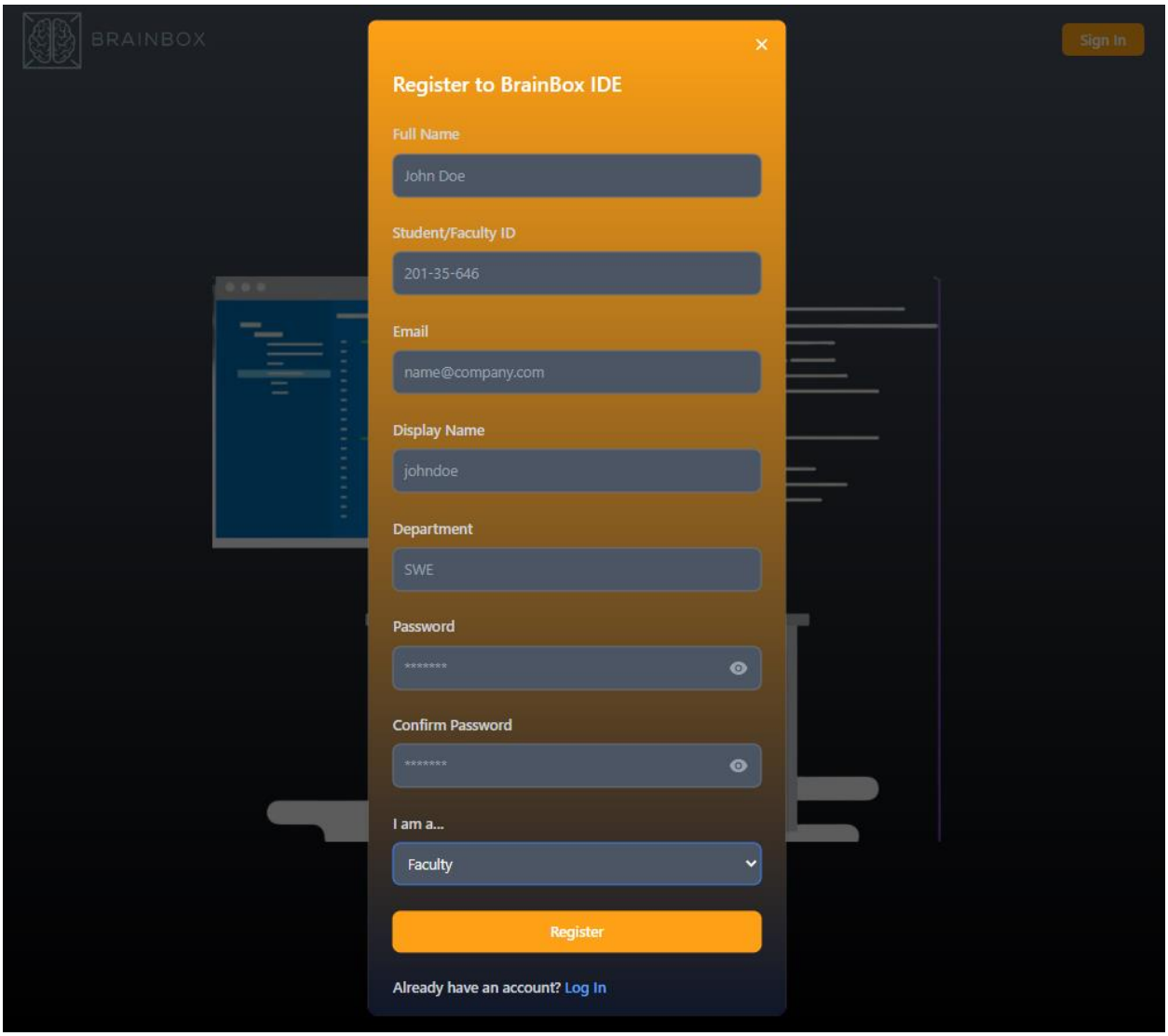


Figure 5.7 : Registration process (Faculty)

5.2.3 Reset Password

If users forget their password they need to click forget password from the login page and they will get the reset password screen. After filling the field and clicking the reset password button with registered mail to the system they will receive the reset password link on their email.

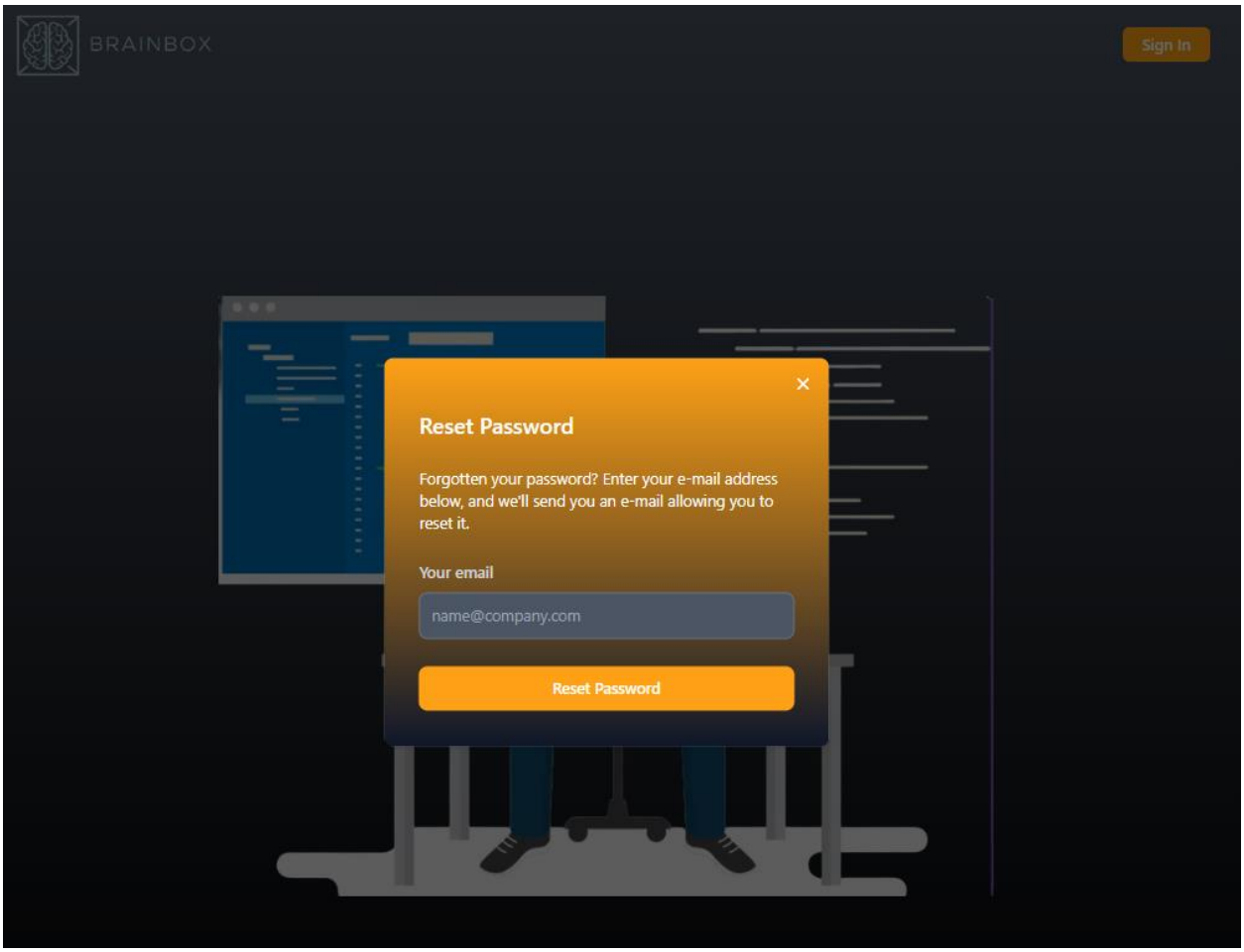


Figure 5.8 : Reset Password process (Faculty)

5.2.4 Students Submissions

Faculty members can view the Student submission option from the dashboard after login to the system. Here they can view each student's code submission.

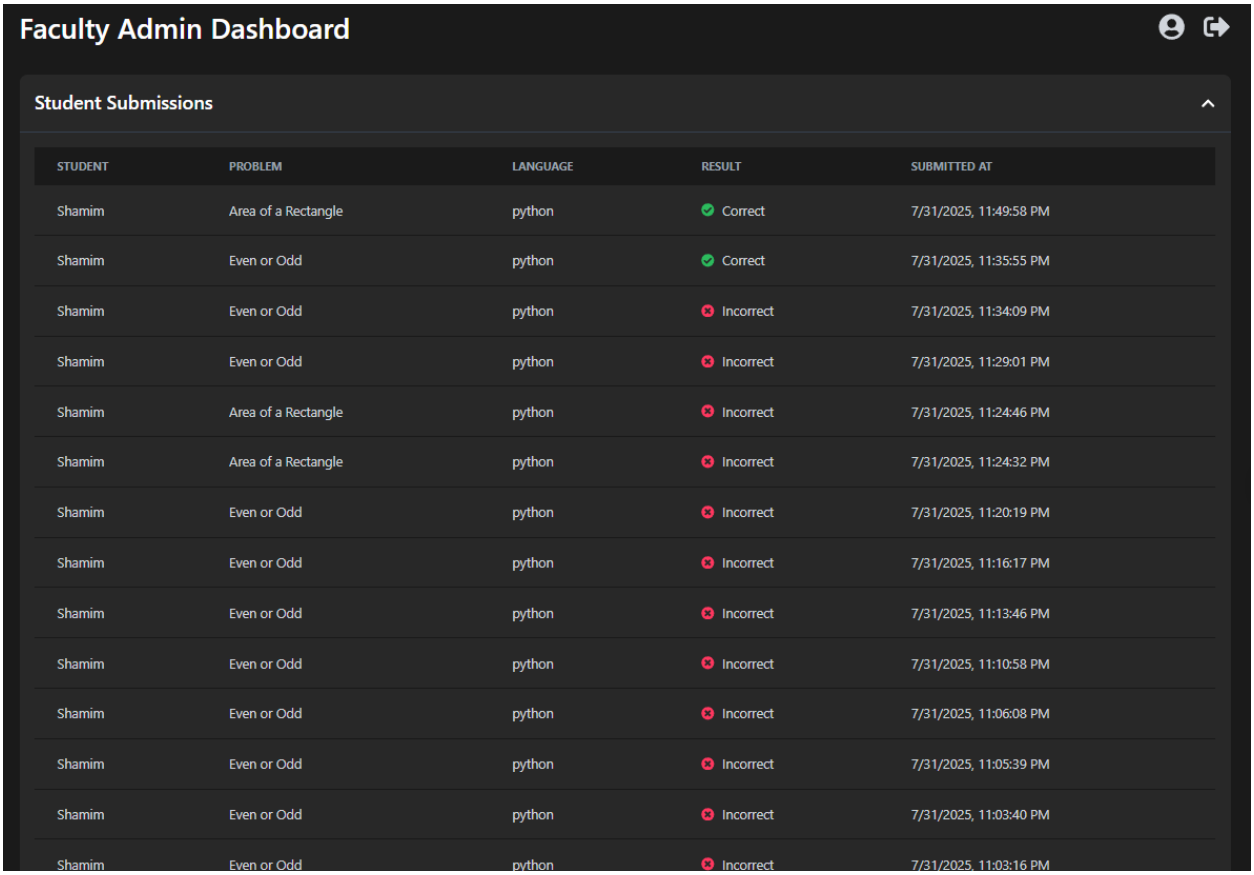


Figure 5.9 : Student Submissions

5.2.5 Create or Update Problems

Faculty/Admins can Update or create any problem statements by filling the form from the dashboard.

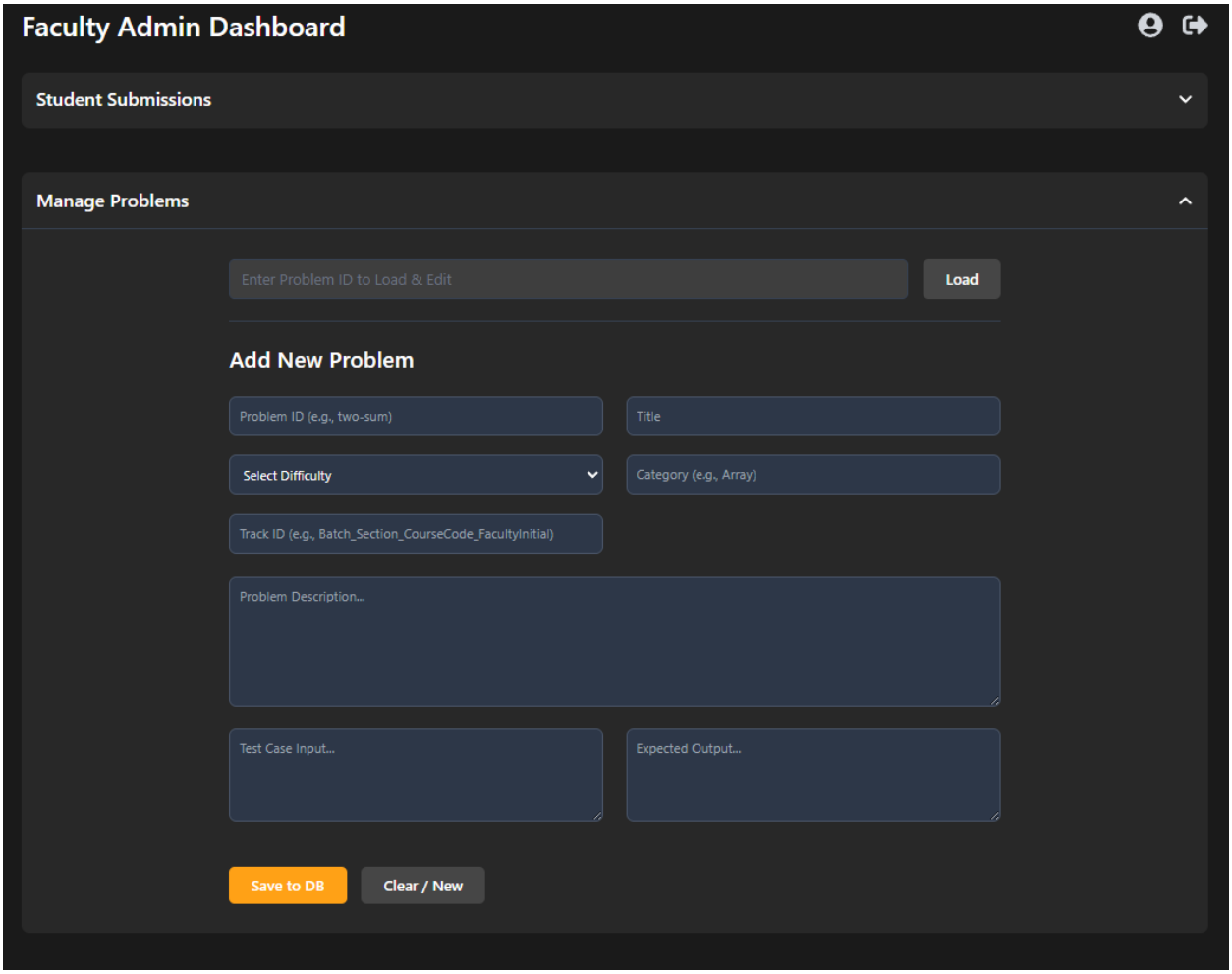


Figure 5.10 : Create or Update problems

5.2.6 Faculty Profile View

Faculty/Admins can view their own profile by clicking their profile icon in the top-right corner on the screen.

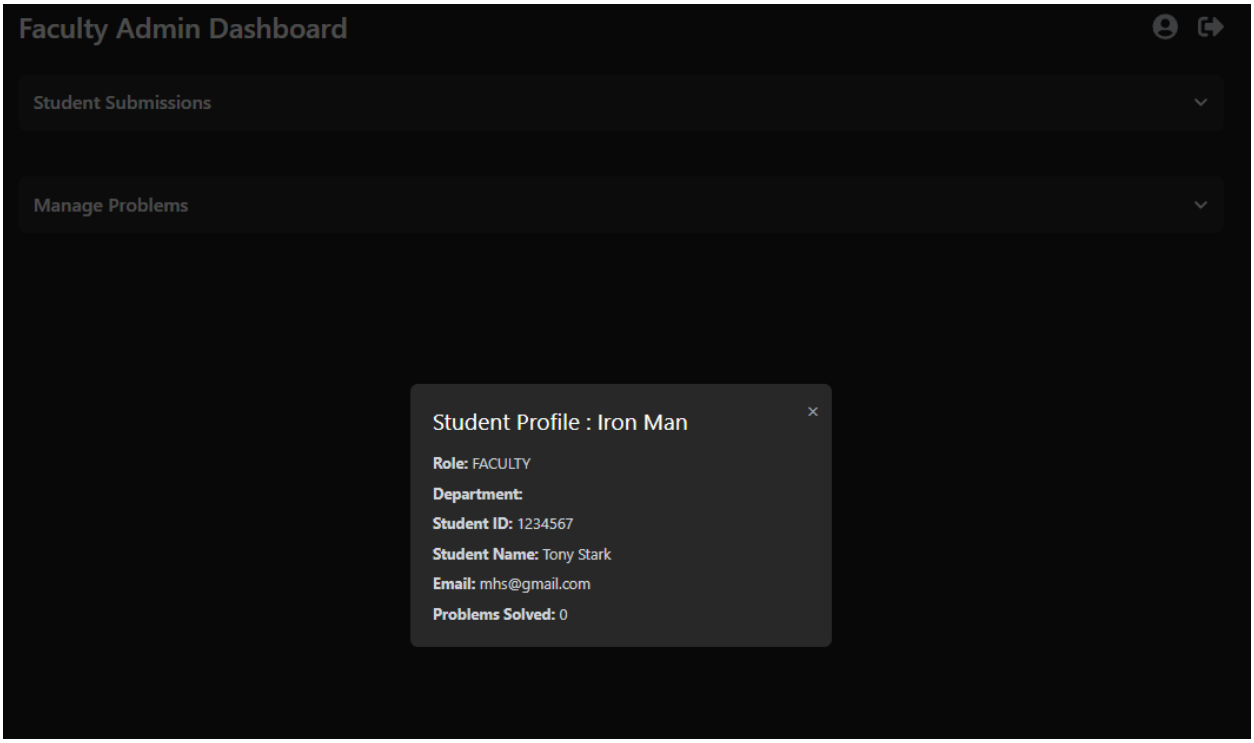


Figure 5.11 : Faculty Profile view

5.2.7 Student Submission Details View

Faculty/Admins can view submission details by clicking any specific student submission from their dashboard.

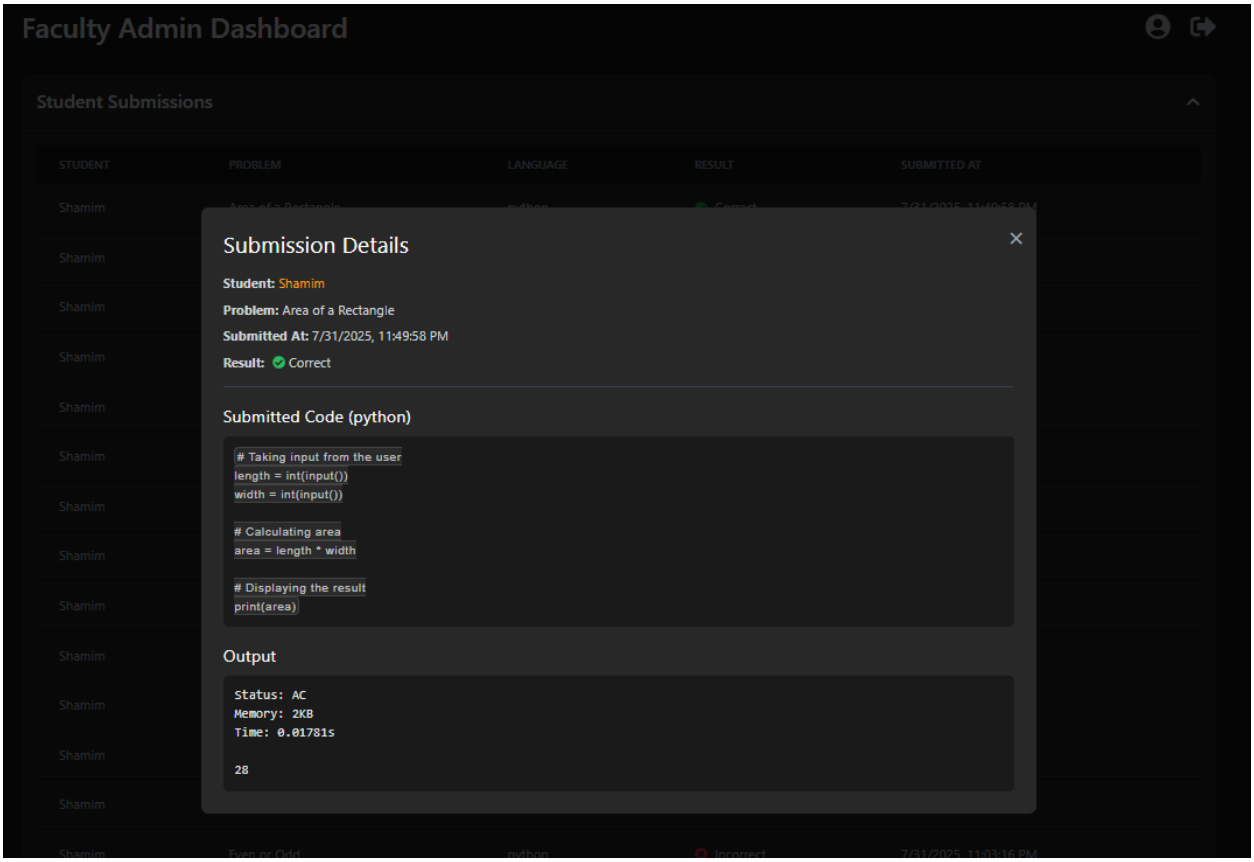


Figure 5.12 : Student Submission Details View

5.2.8 Student Profile View

Faculty/Admins can view Students profile details just by clicking students name on the student submission screen.

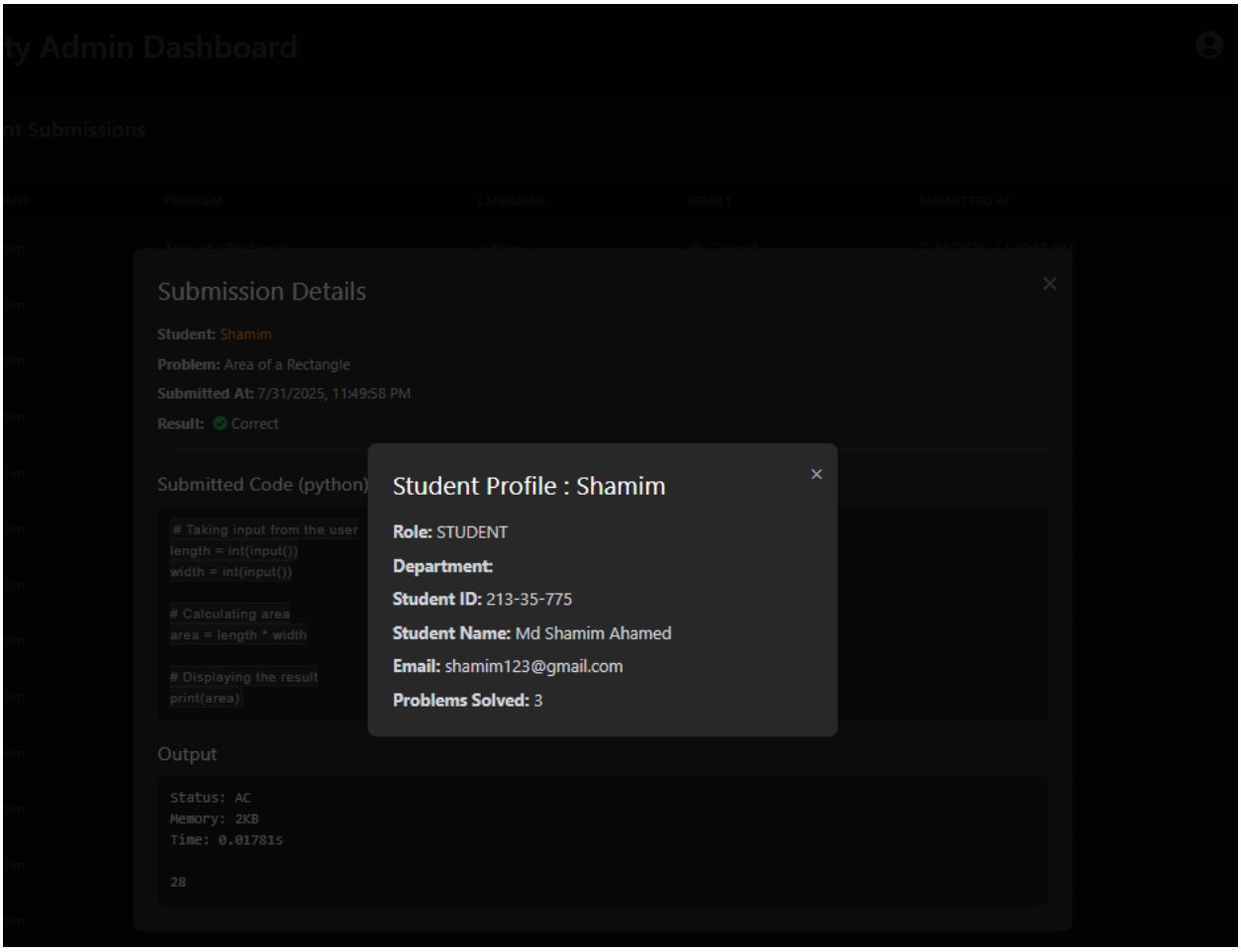


Figure 5.13 : Students Profile View

Project Summary

6.1 Keywords

Online Code Editor, Education, Automation

6.2 Summary

This project is a web-based **Integrated Development Environment (IDE)** built to make programming education easier in universities.

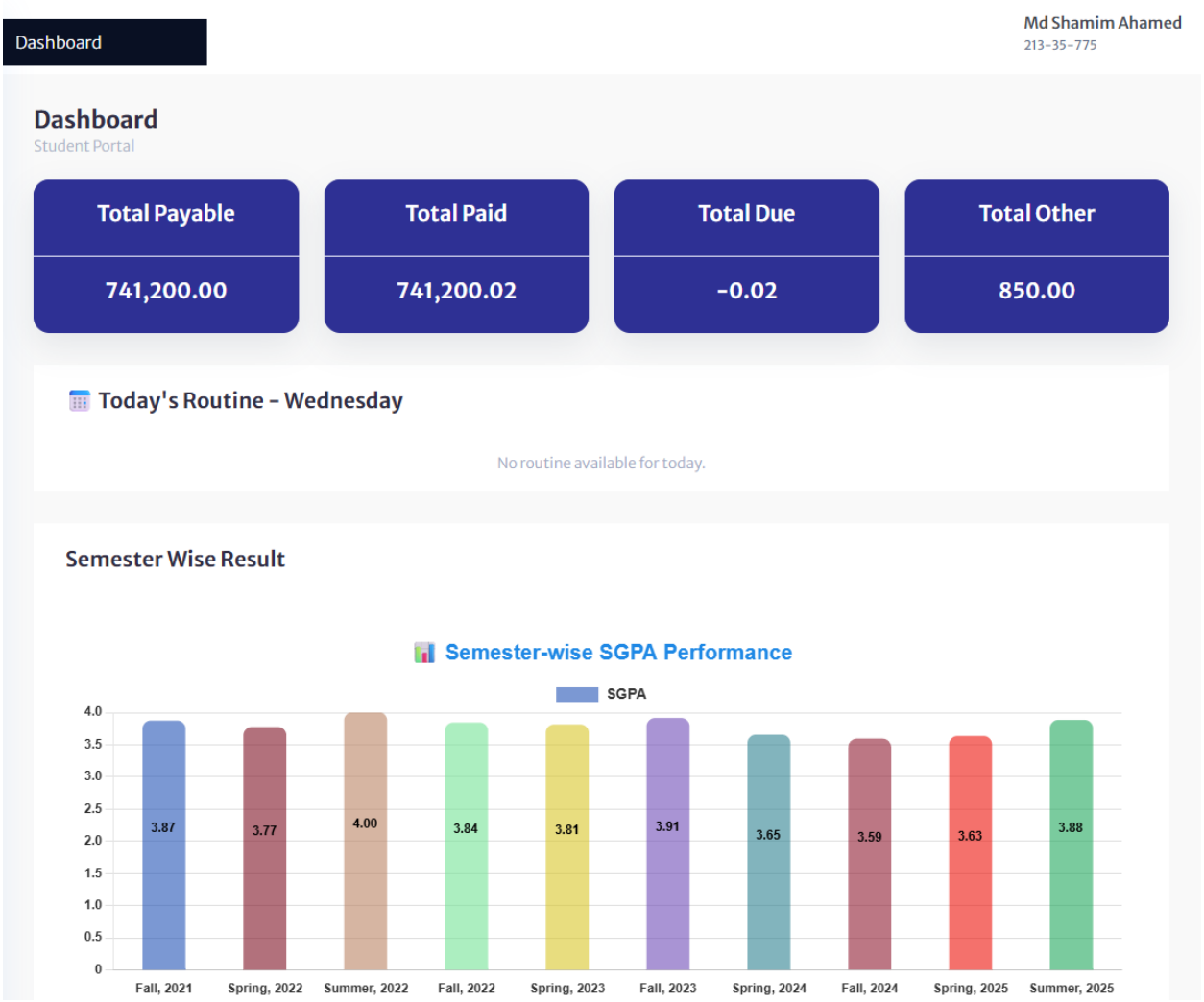
- **For Students:** It provides a single place to view programming problems, write code in a browser, and submit it for instant checking.
- **For Faculty:** It gives teachers a dashboard to create and manage problems, and automatically track all student submissions without manual work.

The goal is to replace the old, messy system of using different local editors and submitting assignments through email. The system was fully tested to make sure user registration, login, problem management, and code submission all work correctly and securely.

APPENDIX A: Source Code & Live Website Link

1. Source Code : https://github.com/shamim36/BrainBox-IDE-Original
2. Live Website : https://brain-box-ide-original.vercel.app

Accounts Clearance



Originality Report

213-35-775

ORIGINALITY REPORT

15%	10%	1%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	5%
2	Submitted to Daffodil International University Student Paper	4%
3	Submitted to NCC Education Student Paper	1%
4	Submitted to University of Central England in Birmingham Student Paper	1%
5	Submitted to Swinburne University of Technology Student Paper	1%
6	Submitted to University of Suffolk Student Paper	1%
7	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
8	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
9	Submitted to KMD Institute (KMD002) Student Paper	<1%
10	Submitted to Higher Education Commission Pakistan Student Paper	<1%
11	123dok.com Internet Source	<1%