SYSTEM ANALYSIS AND DESIGN LAB

CSE 4408

CodeNex

Next Generation Coding platform

Systems Analysis Team Members:

MD Akib Haider
ID:210041222
Faiyaz Abrar
ID:210041214
Sakeef Hossain
ID:210041220
Sameen Yeaser
ID:210041234

Date of Submission:
June 10, 2024

Contents

1 Executive Summary

In today's technology-driven world, where machine learning and AI dominate system architecture, the value of human critical thinking is more important than ever. Problem-solving is key to honing this skill, yet many online platforms lack effective problem-solving trackers and up-solving(more like re-solving with new approach) methods for users to track their progress. Without external cues and triggers, it's easy to lose sight of one's problem-solving journey.

Key pain Points and Drawbacks of current systems:

- Popular online judges lack up-solving benefits of a particular problem. Learning at the initial phase comes with memorizing and certain ad hoc problems require solving the same problem multiple times with spaced repetition.
- Existing platforms have plenty of problems that often lead users to fall into the "Beginners Learning Trap" where people focus on the same genre without pushing themselves to explore a new genre of problems.
- Current systems focus more on ratings and difficulties of problems rather than specific problem type based progress graphs. We will give feedback to the user based on his desired goals to achieve and current solved problem numbers to illustrate current progress.

Our aim is to develop a tailored online judge that caters to all types of coders, including beginners. This platform will provide guidance, encourage consistency, and allow users to set specific goals based on difficulty levels. By implementing a smart up-solve and revision tracker, we ensure that users not only solve problems but also maintain quality and revisit key concepts.

Moreover, in our most advanced version, the platform will foster a sense of community, allowing users to track each other's progress and engage in healthy competition. Additional features such as event scheduling for coding events and personalized user calendars will enhance the overall learning experience.

2 Background

Current system and business process overview:

- The current online judge platform contains plethora of problems that frequently demotivate user and they feel lost while not seeing their progress.
- The user experience is not optimized, with a lack of real-time judging and upsolving tracking features, hindering user engagement and skill improvement.
- Missing of revision techniques causes a lot of effort to regain ones progress after some breaks from problem solving. Waste of time and energy is crystal clearly seen as the success curve is pretty low. Around 90-92% user fail to stick into the process in just 1 years of interval

Motivation of new system addressing the challenge:
Although our system can trigger user with effective and popular study methods like active recall and spaced repetition in a flashcard, where same problems will come repeatedly with certain time gap. People learn gradually and when a problem is encountered at first and the solution/hunch is found, it won't be lost rather will be saved and given as clue at second attempt. This process will continue and build strong concept of any Data Structure, Algorithm topic or random problem solving genre. Event scheduling and community engagement will act as a trigger and reward system will eventually help building a strong habit of regular problem solving.

3 Proposed System Functionality

Functional Requirements and specifications:

- User Registration and Authentication: Users should be able to register for an account with their handle and password. Google APIs can be used for signing up with a user email address.
- Problem Repository: Provide a vast repository of problems categorized by difficulty level, topic. Each problem should include a detailed description, input/output format, constraints, and sample test cases.

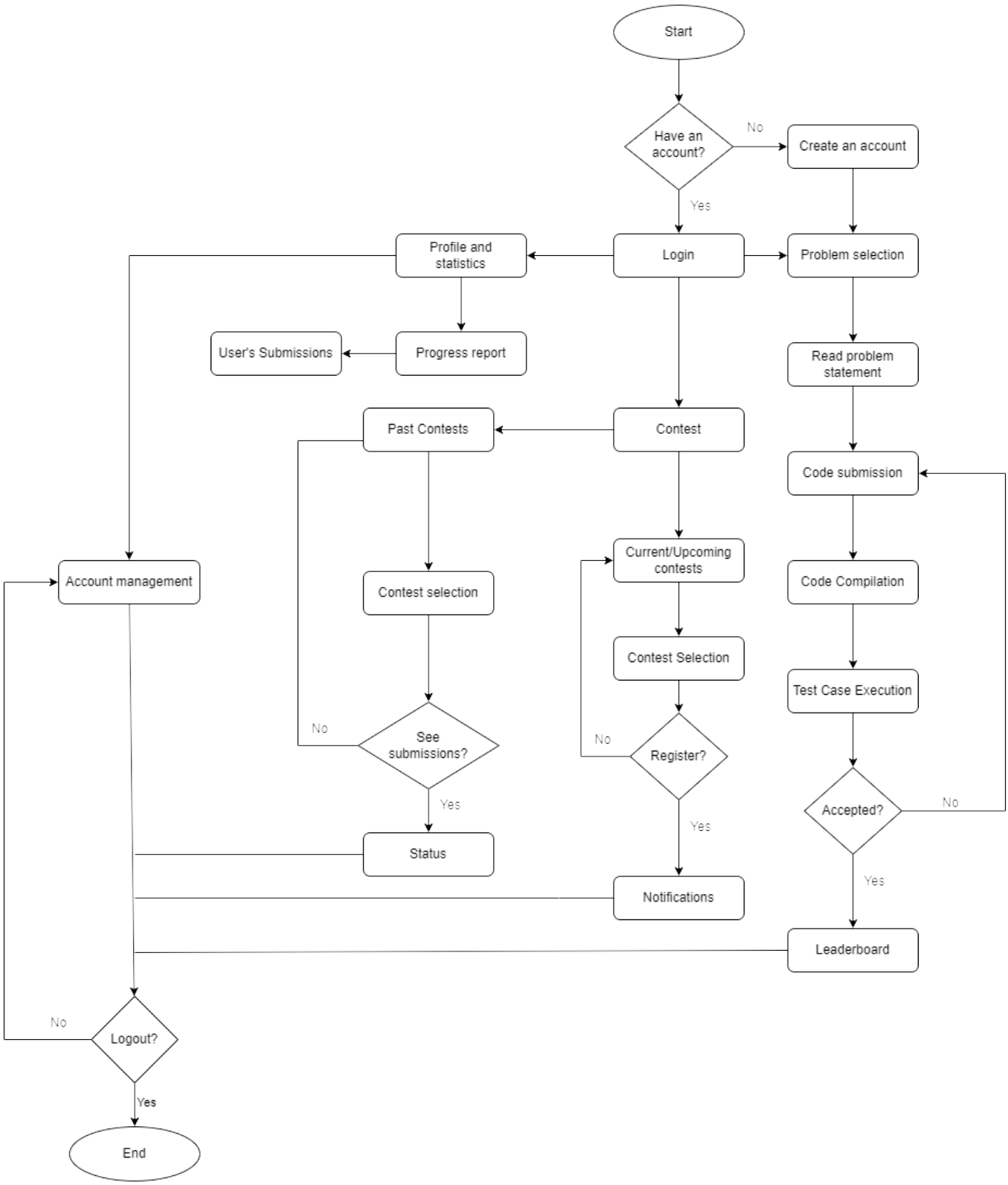

Figure 1: Current system flow chart

- Contests and Practice Mode:

  Host regular contests with predefined start and end times.

- Solution Submission: Users should be able to submit solutions in various programming languages (e.g., C++, Java, Python)
- Validate and compile submitted code before running against test cases.
- Submission History: Solved Problems version tracking with given verdict of every encountered problems
- Up-solving panel with Flashcard Integration

Personas or User Roles:

1. **Contestant:**
   - Login and logout to the system.
   - Submit code solutions.
   - Register for contests and access contest details and problems and View contest results.
2. **Problem Setter (Among Contestant):**
   - Create and submit problems for contests.
   - Define test cases and constraints for problems.
   - Review and edit problem statements.

Interactions with other external systems or interfaces:

- Popular Online Judge APIs like CodeForces, AtCoder, TopCoder to fetch problem details and sample solution.
- Version Control Systems like Github linking.
- Collaboration with other platforms for certain reward and contest arrangements.

4 Technical Specifications

Key technologies and frameworks:

Backend: Express.js (Node.js) for building the backend infrastructure.

Frontend: Utilize modern JavaScript frameworks/libraries such as ReactJS, Typescript along with advanced HTML and CSS for user interface

Database: A robust NoSQL database management system MongoDB to store user data, problem details, submissions.

API Integration: APIs provided by online judges. Also, merging with compiler and interpreter APIs, separate route handling in various programming languages, ensuring flexibility of language

Compatibility with Existing Infrastructure:

Utilizing restAPIs and architectures following industry standards

Version control systems like GIT for codebase changes and submission history

ensuring compatibility with existing infrastructure by following standard web protocols.

Security, Access Control, and Permissions Needs:

- Duplicate login credentials checking
- Specific and separate roles for Admin activities and

  problem setting and stress testing.
- Contest participation restrictions based on difficulty

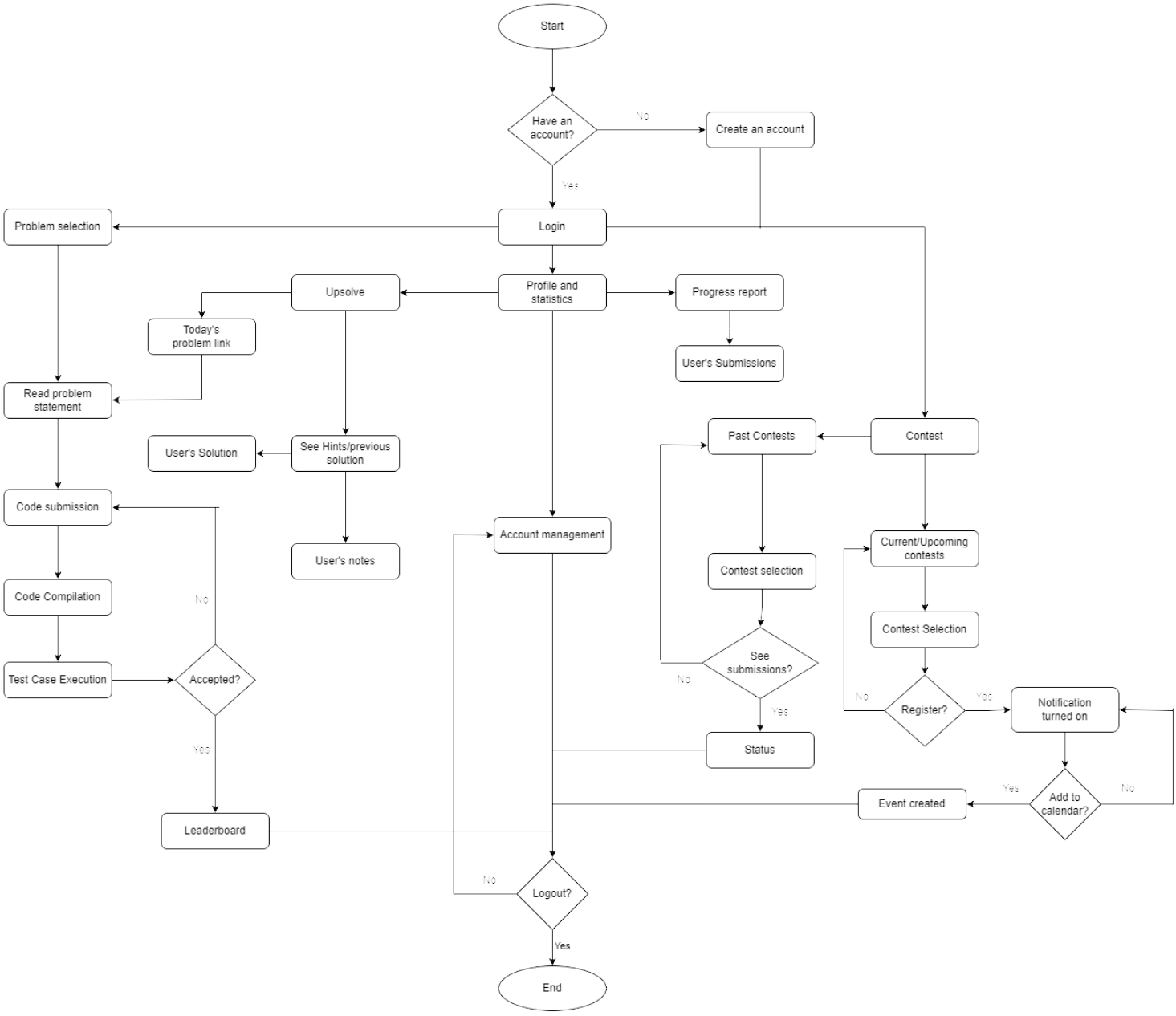

Figure 2: System flowchart of proposed system

5 System Alternatives

Alternative systems(eg: codechef, leetcode) pros vs cons evaluation		
Points	Our System Pros	Alternative system characteristics
Smart Upsolve	Smarter upsolve using flash card technique	Inconsistency issues of user experiences and challenges
Revision Tracker	Suitable revision history enabling to track problems solved before	Limited revision history in codechef and codeforces
Performance Tracker	Performance tracker based on different parameters, not only solve count but also different categories and difficulties set	Lack of progress graphs based on a variety of parameters and limited performance analysis
Coder's Calendar	An interactive calendar to store contest events in order to provide an organized view	Contest Tracking is present but lack of a proper view in a personal scheduler
Reward System Samoa	Vouchers and merchandise to motivate users to solve more	Many of the system alternatives lack a reward or recognition system

Alternative System Evaluation in a Decision Matrix

Feature	CodeChef	LeetCode	AtCoder	Codeforces
Smart Upsolve	- No triggered upsolving Feature	- No triggered Upsolving Feature	- Submission Rerun available	- Problem Solving Assistant present
Revision Tracker	- Limited Revision History	checkmark Detailed Revision Tracker	- Revision History available	- Limited Revision Tracker
CP Analysis	- Only codeChef Rating System	✓Explore Mode and Problem Sets	✓User Ratings and Contests Analysis	✓Comprehensive User and Contest Analysis
Problem Recommendation	- Limited problem recommendation	✓Explore Mode suggests problems	✓Problem Tags and Recommendations	✓Suggests Problems Based on Submissions
Interview Preparation	✓CodeChef for Schools	✓LeetCode Premium for Mock Interviews	- Limited Interview Preparation	- Limited Interview Preparation
Platform UX/UI	✓User-friendly Interface	✓Clean and Intuitive Interface	✓Minimalistic Interface	✓User-friendly and Intuitive Interface
Language Support	✓Wide Range of Programming Languages	✓Support for Multiple Languages	✓Support for C++, Python, and more	✓Extensive Language Support
Lacking Features/Notes	- Advanced problem recommendation system	- Detailed contest tracking and history	- Extensive educational content	- Limited revision tracking

Table 1: Comparison of Features across Online Judges

6 Feasibility Analysis

Technical Feasibility :
i. Software :

- Backend Design: Express.js (Node.js) for building the backend infrastructure
- Frontend Design: Utilize modern JavaScript frameworks/libraries such as Angular, or Bootstrap for user interface
- Database Design: Robust database management system like PostgreSQL
- API Integration: APIs provided by online judges. Also, merging with compiler and interpreter APIs in various programming languages, ensuring flexibility of language

ii. Hardware :

- Problem hosting and solution database Servers with high performance CPU.
- Adequate Storage and backup system node balancing Network-Attached Storage devices(NAS).

iii. Complexity :

- Real-Time Code Execution: Implementing a feature that allows users to execute and test their code in real-time presents technical challenges.
- Content Management : Developing this system with version control and content categorization for coding problems, solutions introduces complexity to the project.

iv. Risks :

- Security Vulnerabilities : Failure to implement robust security measures could lead to compromised user data and reputational damage.
- Load Balance : Maintaining huge number of response from multiple user at contest time can be challenging.

Operational Feasibility : Adequate throughput and response time :

- Optimized Code Execution: We will implement optimized code execution processes to minimize latency and maximize throughput.
- Scalable Infrastructure: The platform will be built on a scalable infrastructure that can dynamically allocate resources based on demand.
- Monitoring and Performance Tuning: Continuous monitoring of system performance ensures that the system maintains adequate throughput and response time levels.

Our system of operation is engineered to provide unparalleled levels of throughput and response time, setting a new standard for online coding platforms.
Economic Feasibility :
i. Break-Even Point :

- Projected costs and revenues outlined over 10 years, aiming for break-even with adequate membership and revenue growth.

ii. ROI analysis :

- First year: -45%
- Fifth year: -1.7%
- Tenth year: 2.6%
- Fifteenth year: 34%

ECONOMIC FEASIBILITY
TANGIBLE COST AND BENEFITS

07

CRITERIA	ESTIMATED BENEFITS(YEARLY)
PROJECT DEVELOPMENT (DESIGNING, CODING, TESTING, DEBUGGING)	FREE OF COST (ONLY MANPOWER REQUIRED)
SOFTWARE LICENSE, CLOUD SERVER AND HOSTING	BDT -8500
PLATFORM RESEARCH, ANALYSIS AND EXPERIMENTS	BDT -6000
REWARDS AND MERCH MAINTENANCE	BDT -7000
PROMOTIONAL ACTIVITIES AND MARKETING	BDT -5000
IT INFRASTRUCTURE	BDT -8000
OPERATIONS OVERHEAD	BDT -5000
ADVERTISEMENT REVENUE	BDT 15000

Figure 3: TANGIBLE COST AND BENEFITS

ECONOMIC FEASIBILITY
INTANGIBLE COST AND BENEFITS

08

CRITERIA	ESTIMATED COSTS(YEARLY)
DEVELOPMENT TIME AND TOTAL WORKHOURS	10 HOURS PER MEMBER PER WEEK. IN TOTAL 300-320 HOURS
USER ADOPTION CHALLENGES	SIMPLE USING, EASY AVAILABILITY; SKEPTICISM OF USER CHOICE
MAINTENANCE OVERHEAD	TECHNICAL ISSUES ADRESSING AND FREQUENT RESPONSE TIME UPDATES
LEARNING CURVE	LESS KNOWLEDGE PRE-REQUISITE
CRITERIA	**ESTIMATED BENEFITS(YEARLY)**
COMMUNITY BUILDING AND ENGAGEMENT	STUDENTS, EDUCATORS, NEW LEARNERS BLENDING
BRAND REPUTATION	ENHANCED REPUTATION WITH POSITIVE FEEDBACK
EDUCATIONAL IMPACT	PROFOUND IMPACT ON COMPUTER SCIENCE EDUCATION
SOCIO-ECONOMIC IMPACT	INNOVATION SHOWCASING AND IDEA GENERATION

Figure 4: INTANGIBLE COST AND BENEFITS

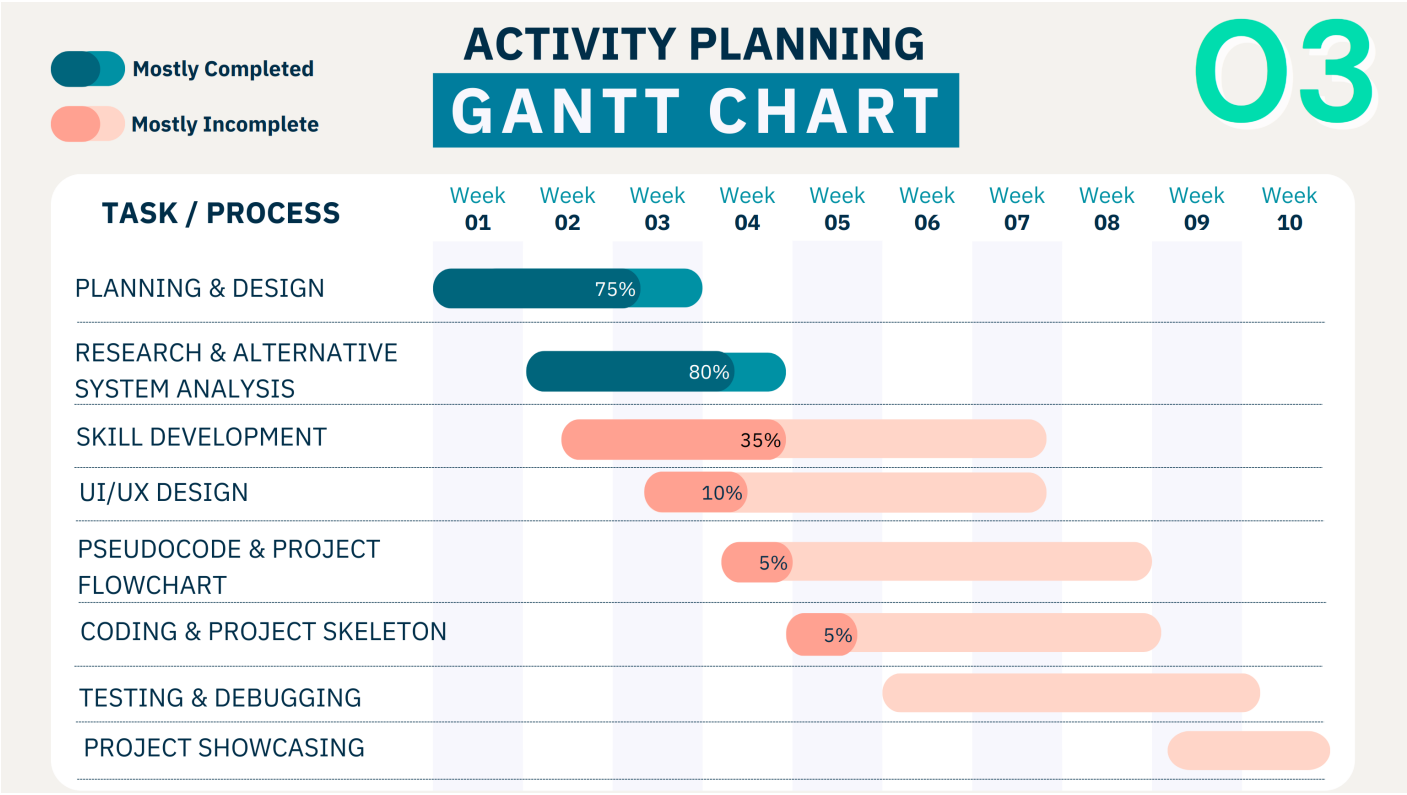

Figure 5: Gantt Chart

7 Activity Planning and Control

Activity Timeline

- Codenex proposal and planning (Weeks 1-3)
- Feasibility analysis (Weeks 2-4)
- Analysing requirements (Weeks 2-5)
- Frontend learning and documentation (Weeks 3-6)
- Backend learning and documentation (Weeks 4-8)
- UI/Ux design and analyze (Weeks 5-8)
- Backend implementation (Weeks 6-9)
- Testing and debugging (Weeks 7-9)
- System launching (Weeks 8-10)

8 Requirements Elicitation and Analysis

Survey Process

- Conducted via Google Forms
- Contestant: 83.3% Problem Setter: 16.7%

Key Findings

- User Demographic: Two types of target audience were kept in mind while distributing the surveys which are Contestants and Problem Setters.

- User Expectation: The majority of respondents were contestants, highlighting their significant presence in the target user base.

i. Functional Requirements (Contestant)

- Must provide user access to features like smart upsolving and performance tracker
- Should provide a platform for contestants to discuss problem-solving approaches and collaborate with peers.
- Could provide detailed analytics on code performance, such as time complexity and memory usage.

ii. Nonfunctional Requirements(Contestant)

- Must provide a user-friendly interface a clear dashboard of available coding problem.
- Should optimize code execution to minimize waiting times for results.
- Could allow code editor environment customization features.

i. Functional Requirements(Problem Setter)

- Must have a secured interface with zero buffering issues.
- Should have flexibility in problem setting.
- Could have Interactive input-output structure.

ii. Nonfunctional Requirements(Problem Setter)

- Must have provide stress test-constraint flexibility.
- Should have provide long-term enhancements in problem diversity.
- Could have provide community feedback.

9 Project Documentation

Github URL : https://github.com/aaakloo-waiting/CodeNex
Guidelines : CodeNex is an interactive online platform for coding enthusiasts. It allows users to solve programming challenges and run code snippets in C, C++, Java, and Python.

User Capabilities

Authentication Users can authenticate themselves using Google authentication through PassportJS.
Problem Management Authenticated users have the ability to add, modify, and remove programming problems, complete with customized test cases.
Code Execution All users are able to execute their code directly on the problem page. Signed-in users can also submit their code for verification.
Submission Feedback Upon submitting code, users receive immediate feedback on whether their solution is accepted, incorrect, or has timed out (TLE).
History Users can access their submission history for each problem and review their code.
Status Users have access to recent submissions from all users.

Running Code

SimpleOJ utilizes Node.js child processes for executing code written in C, C++, Java, and Python. Backend management ensures secure and isolated runtime environments for each code snippet.

Technology Stack

Frontend Developed using ReactJS to create a dynamic user interface.
Backend Built on NodeJS with Express, providing a robust server-side framework.
Database MongoDB is employed for data storage, offering flexibility and scalability.
Authentication PassportJS handles user authentication for secure logins.
Concurrency Management Bull Queue efficiently manages concurrent code execution requests.
Containerization Docker is used to containerize the application, ensuring consistent environments across different machines.

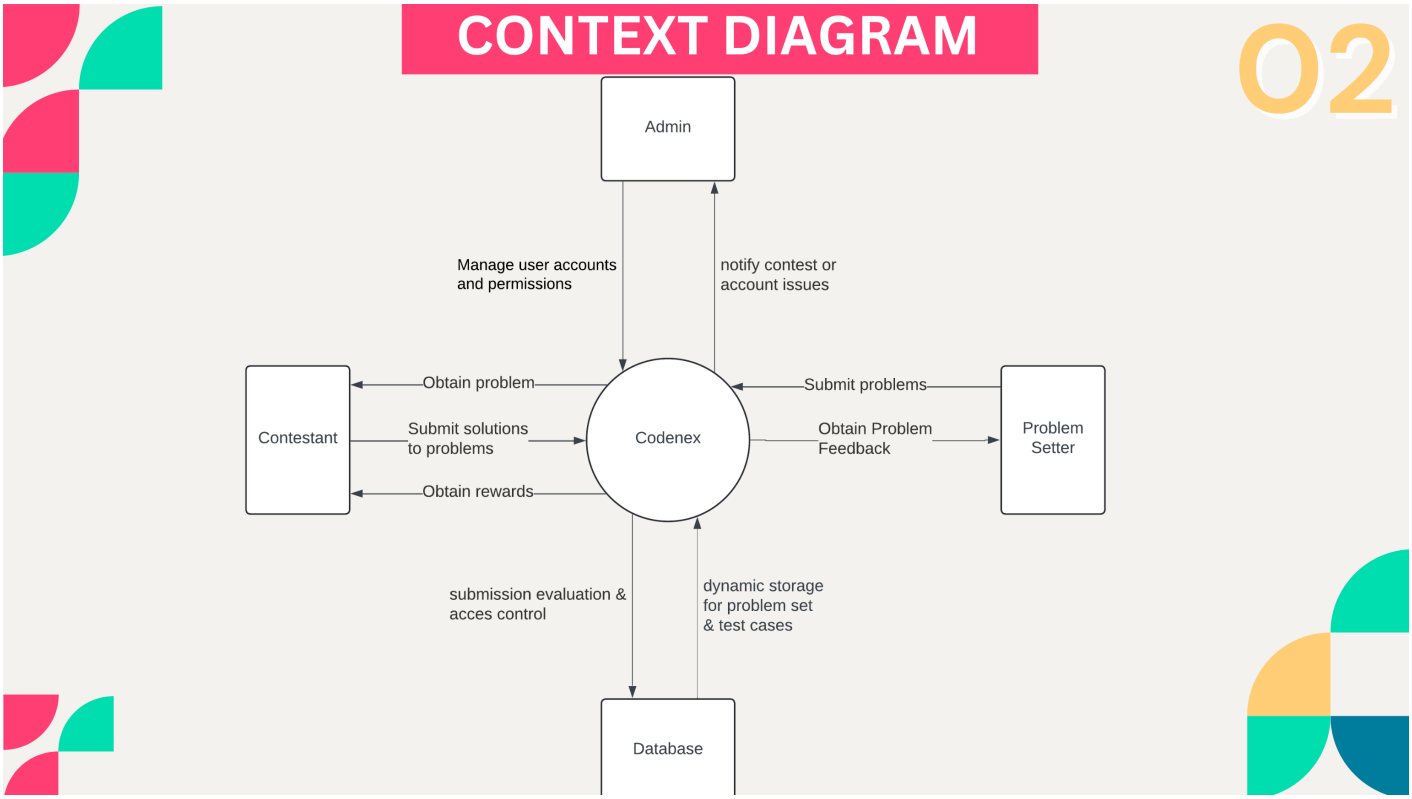

Figure 6: Context Diagram

Future Goals

Complexity Analysis Planning to implement memory complexity analysis for running code. Currently acquiring resources and knowledge for this implementation.

Project summary

CodeNex aims to provide coding enthusiasts with a secure and user-friendly platform for skill enhancement through problem-solving and contest participation. With features like real-time judging, upsolving method, progress checker, event updates, the project promises to deliver a seamless experience. By emphasizing user-centric design, hope that this initiative will bring value to both users fulfilling commercial purposes with economic, operational feasibility.

10 Data Flow and UML Diagrams

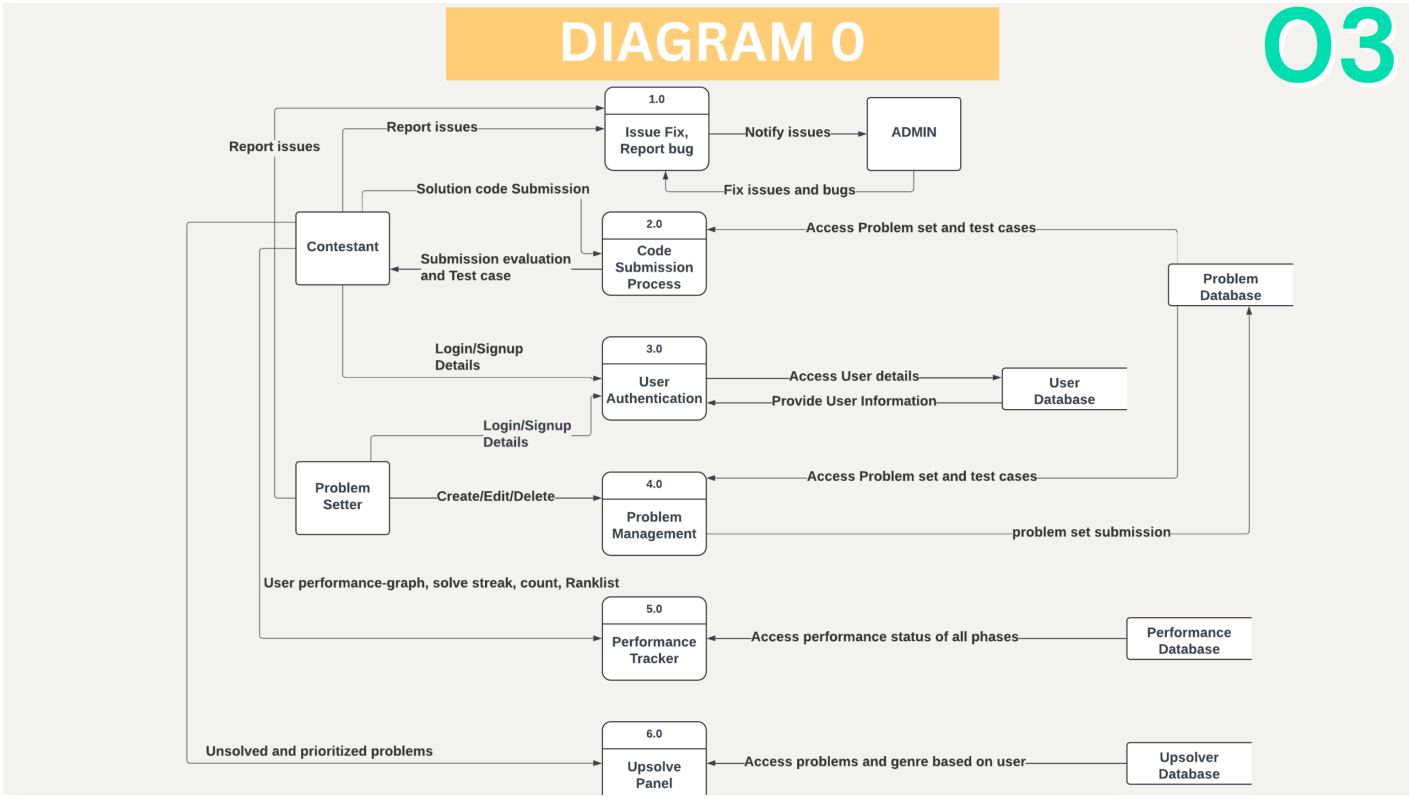

Figure 7: Level 0 Diagram

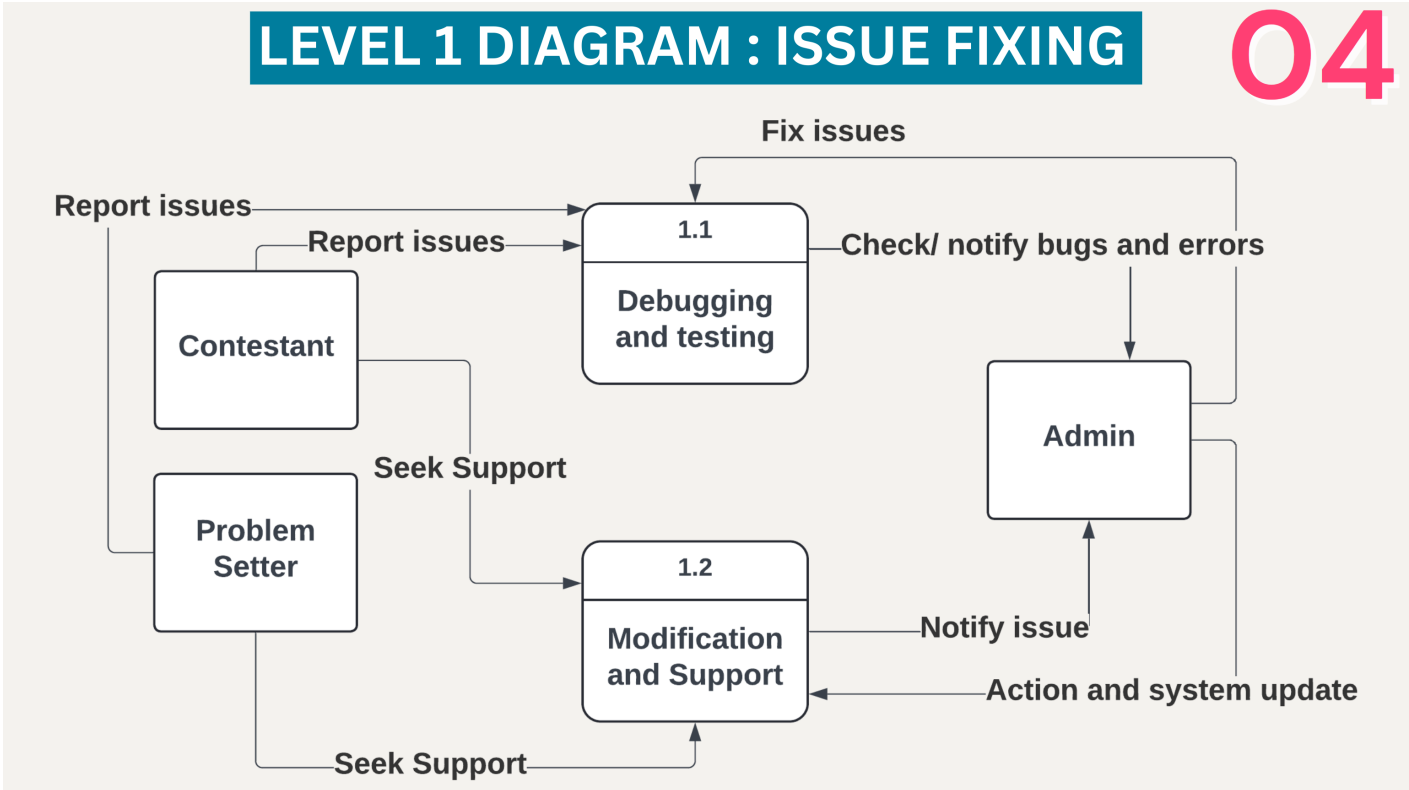

Figure 8: Level 1 Diagram: Bug Fxing

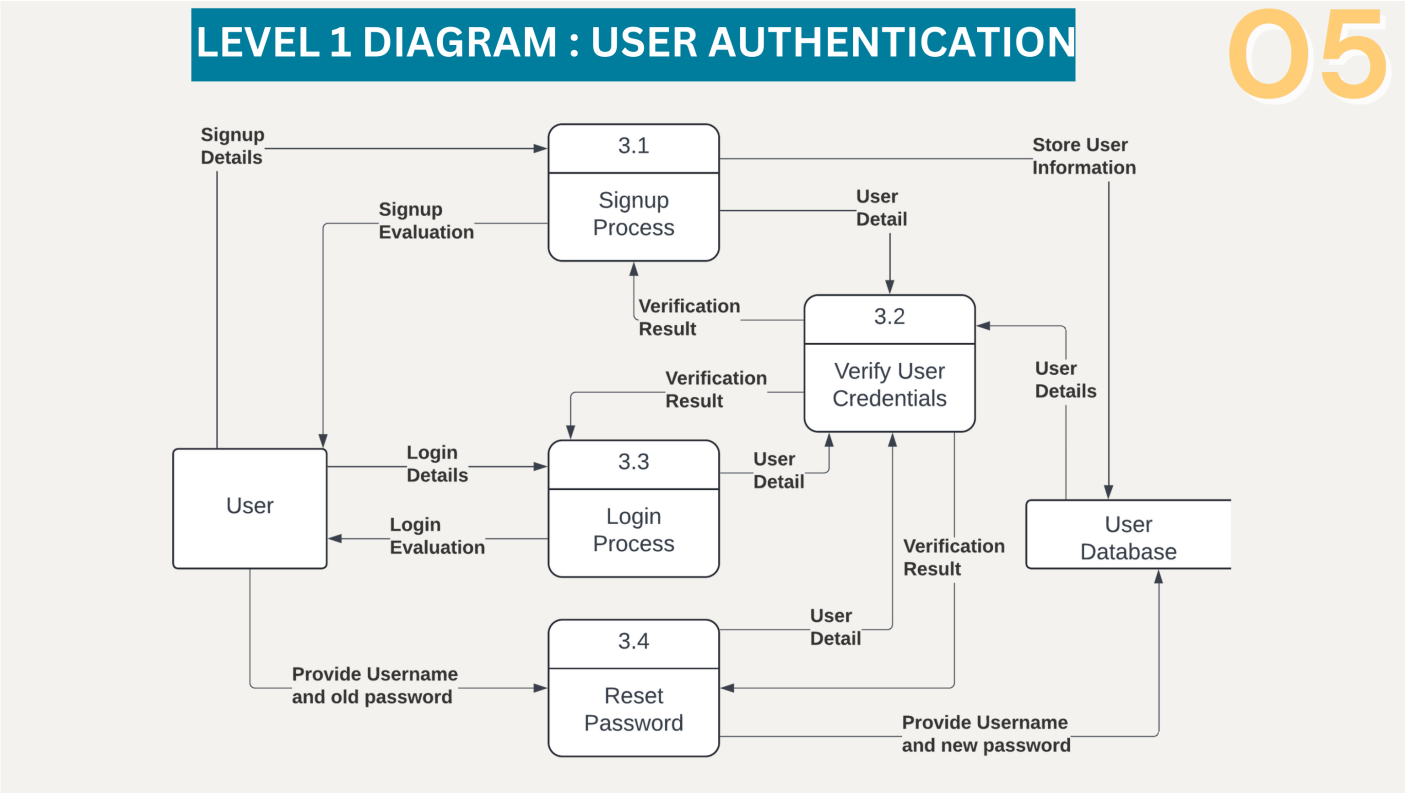

Figure 9: Level 1 Diagram: Authentication

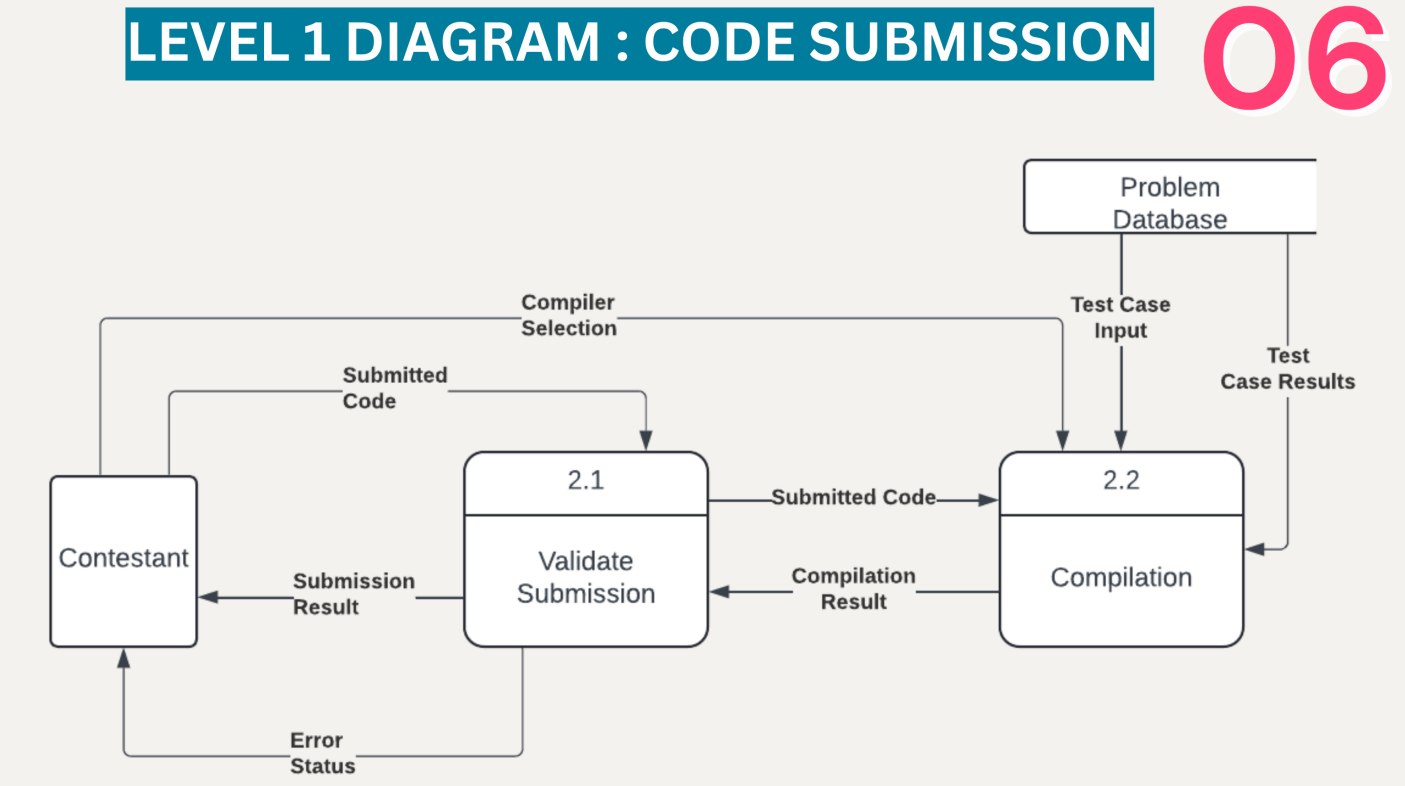

Figure 10: Level 1 Diagram: Problem Creation

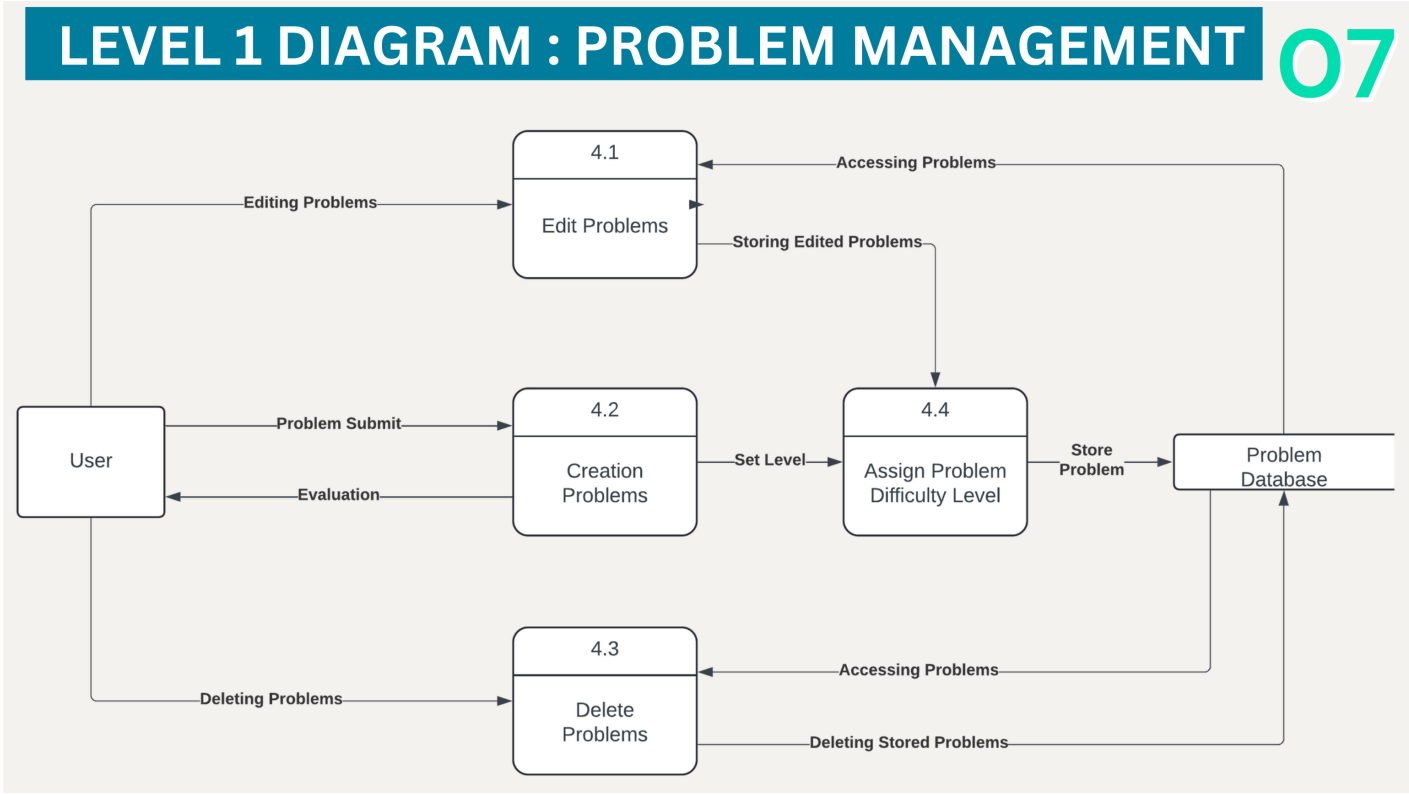

Figure 11: Level 1 Diagram: Problem List Categorizing

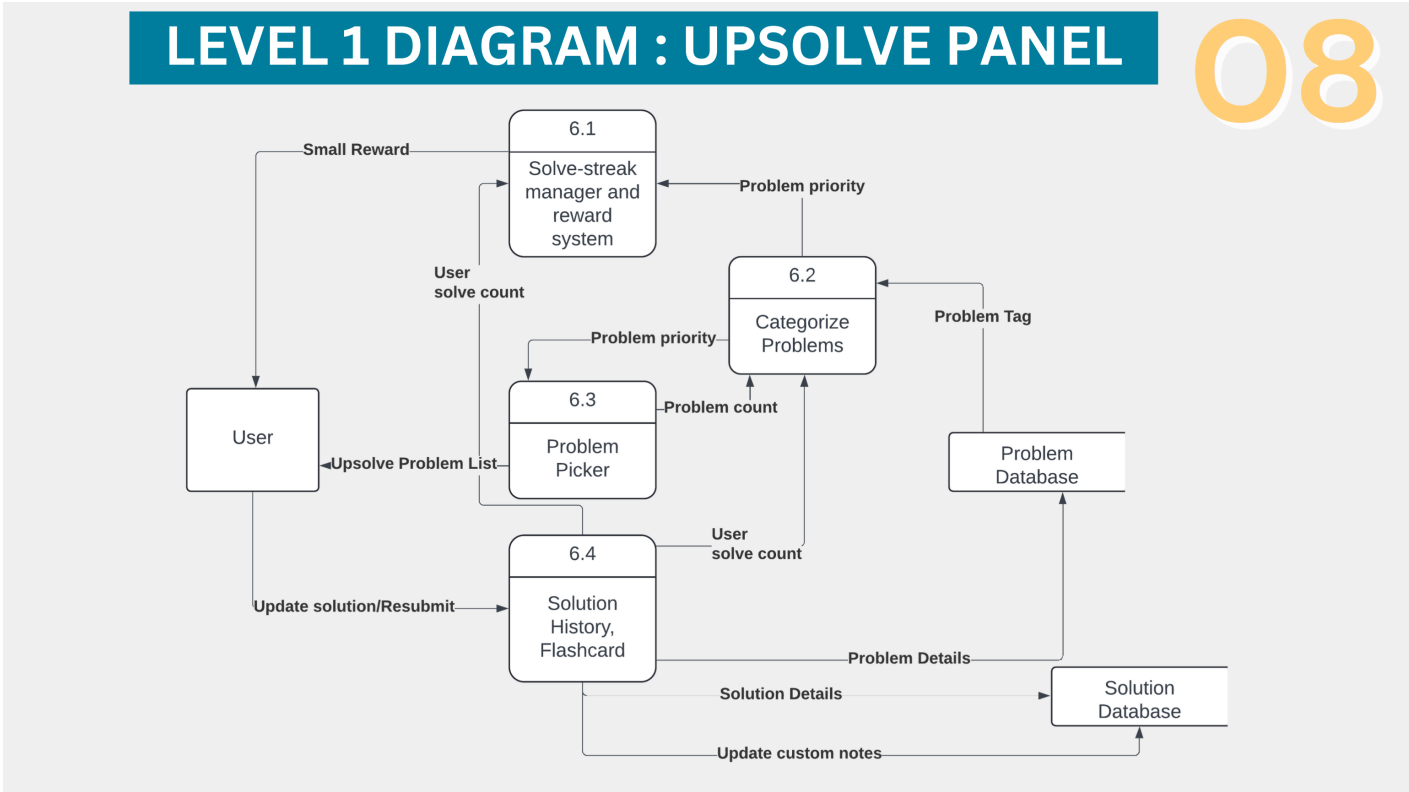

Figure 12: Level 1 Diagram: User Note and Upsolve

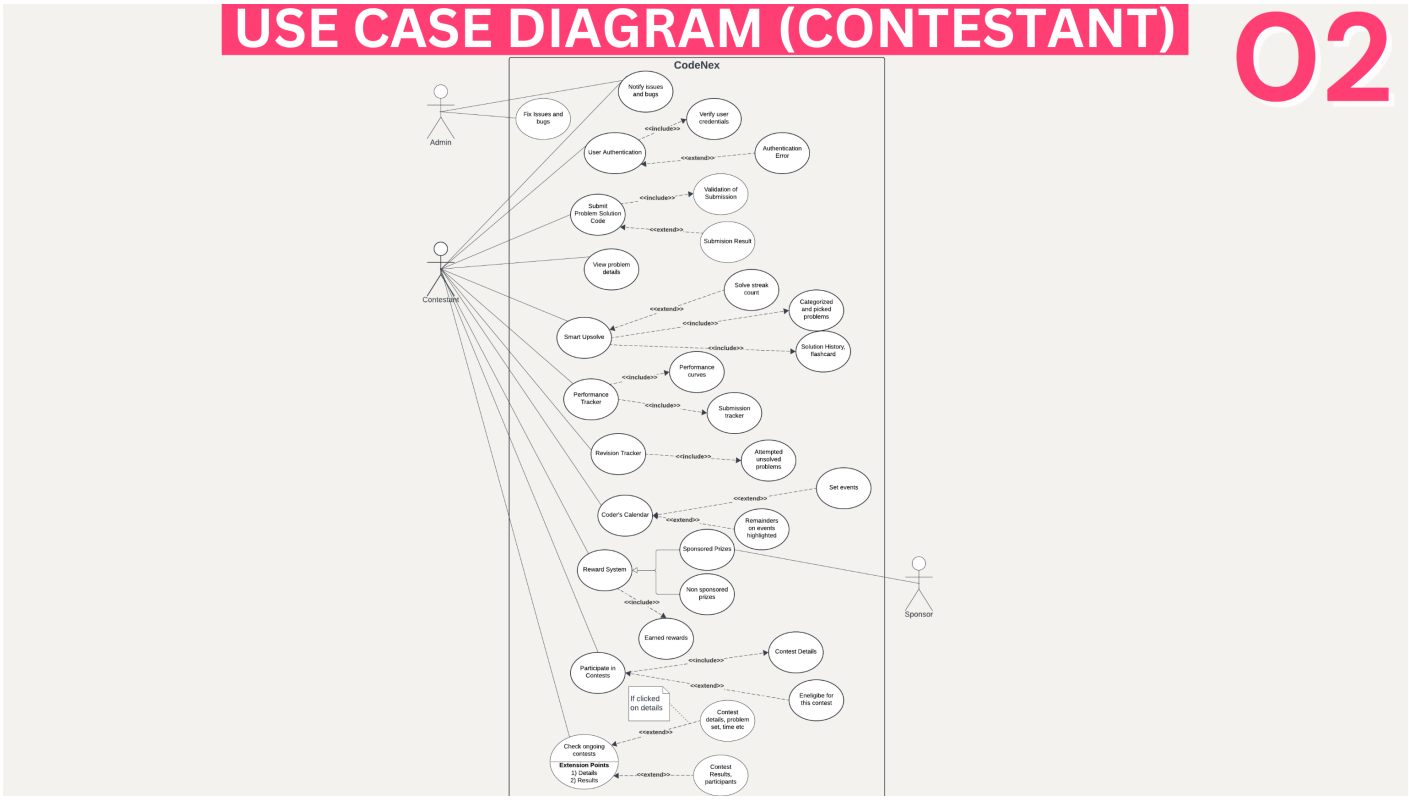

Figure 13: Use Case Diagram: Contestant

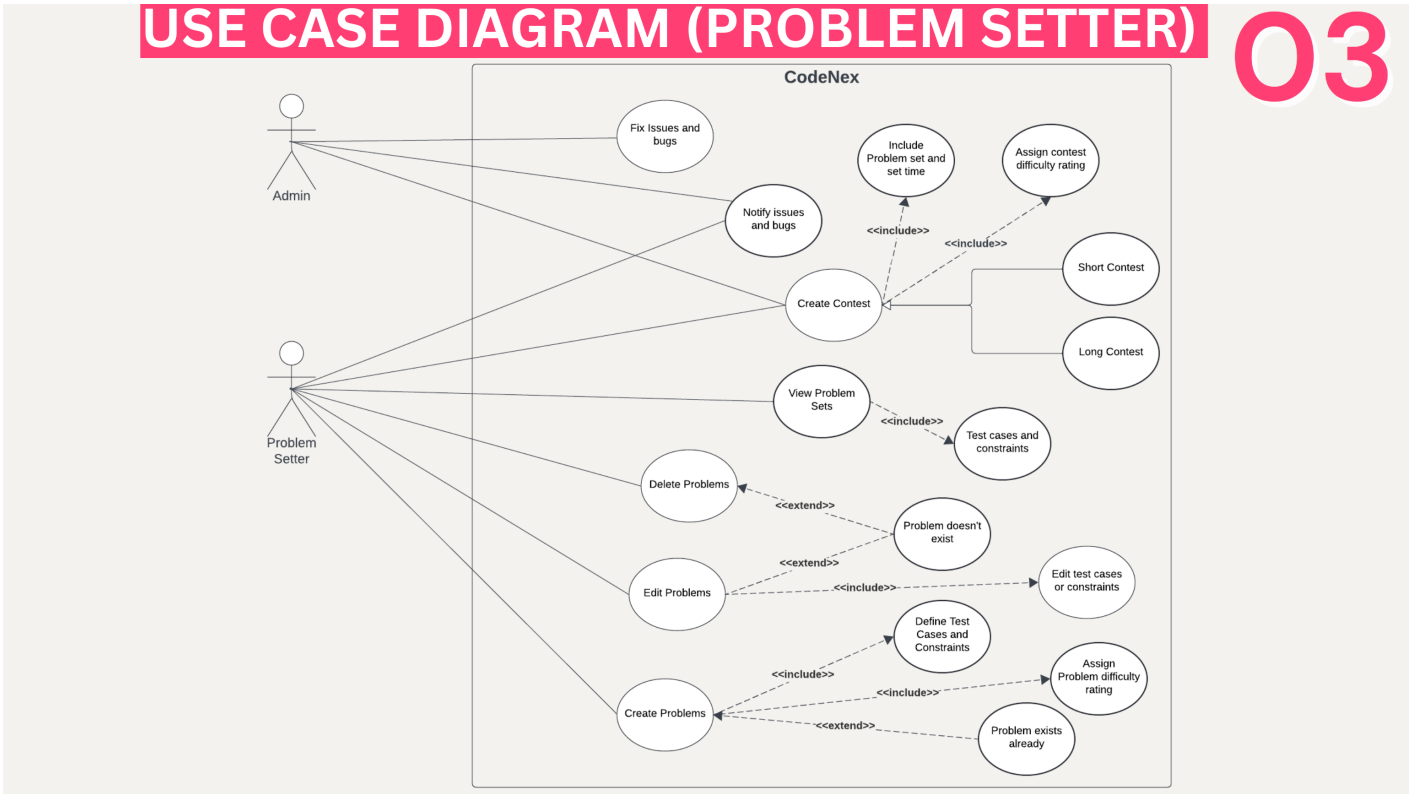

Figure 14: Use Case Diagram: Problem Setter

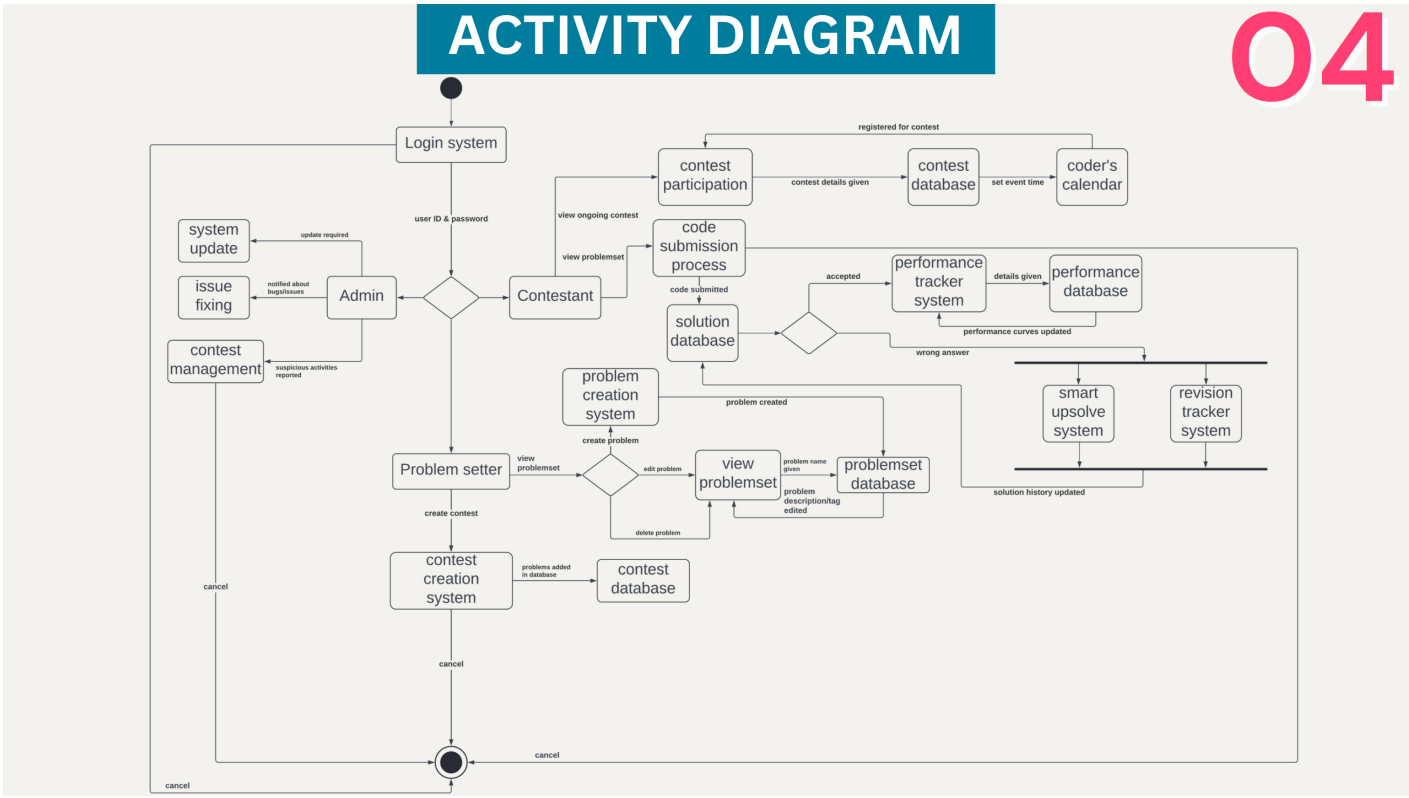

Figure 15: Activity Diagram

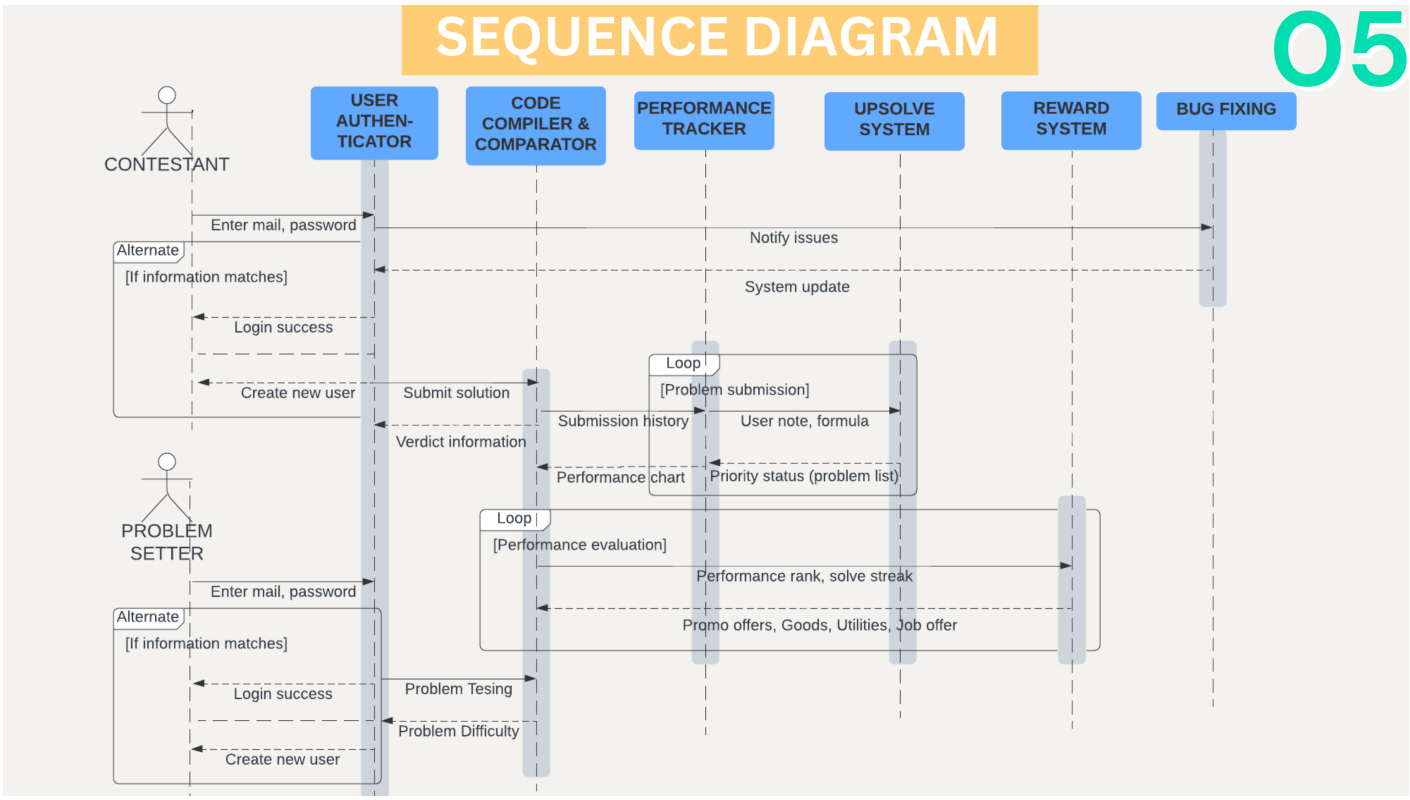

Figure 16: Sequence Diagram

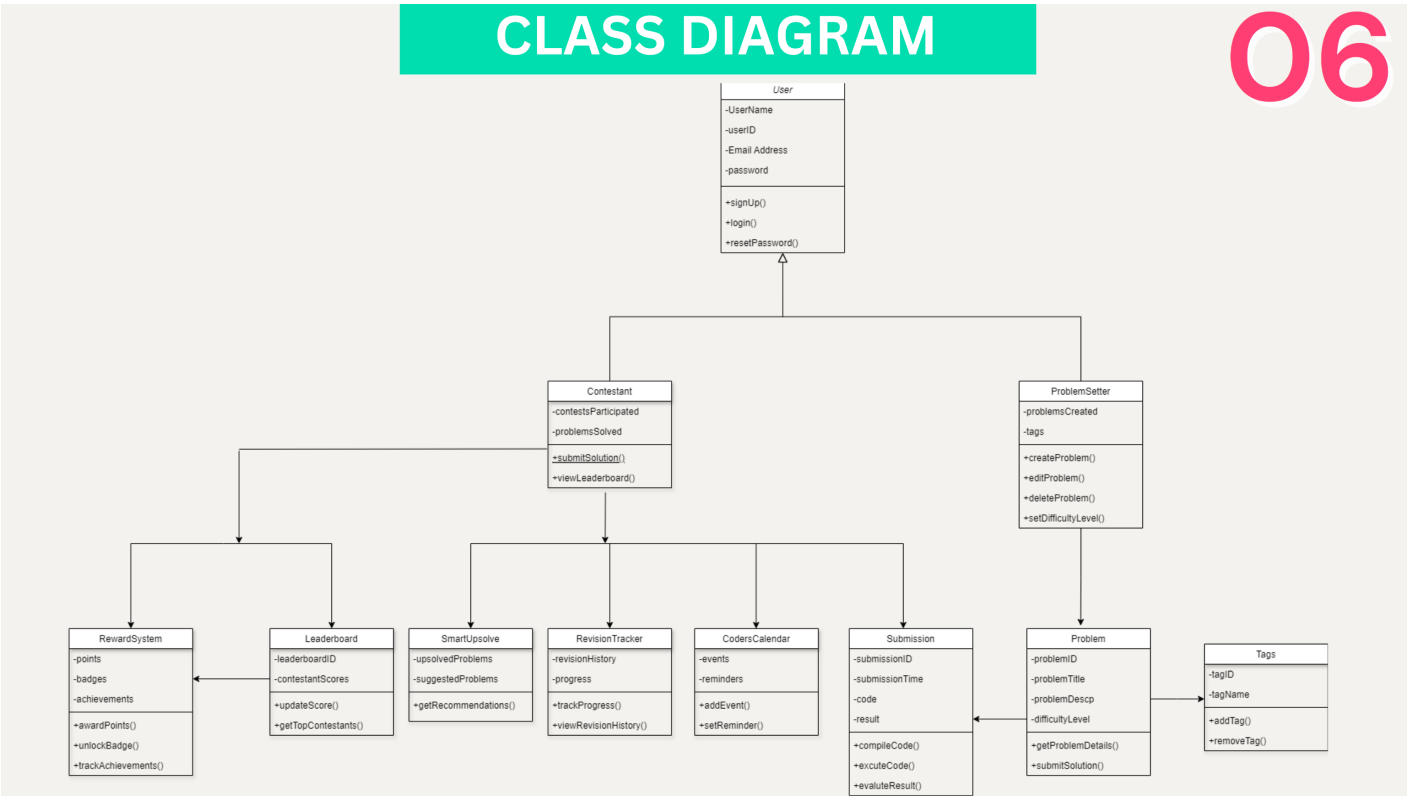

Figure 17: Class Diagram