Software Requirements Specification

for

Codecatalyst

Prepared by

Group 5

Hasan Mahmud (ID: MUH2225007M)

Md. Sabbir Ahamed(ID: ASH2225005M)

Sadia Nur Safa (ID: BFH2225032F)

Nelema Jahan (ID: BFH2125032F)

Institute of Information Technology

Noakhali Science and Technology University

08 July, 2025

Table of Contents

List of Figure

1. Introduction

The Software Requirements Specification (SRS) introduction contains the SRS's policy, scope, references, and summary. This document's goal is to gather information about proposed system as we name it as "Codecatalyst" and is to give readers a greater understanding of it by outlining the issue statement in great detail. While defining the qualities of a high-quality product, it also emphasizes the advantages and requirements of the participants. Details on the "Codecatalyst" can be found on this document.

1.1 Purpose of the Document

This Software Requirements Specification (SRS) document is intended to provide a comprehensive description of the requirements and functionality of the **Codecatalyst** system. Codecatalyst is designed to be an **online competitive programming and contest hosting platform** that allows users to register, participate in coding contests, submit solutions, and view real-time leaderboards.

The primary purpose of this document is to define the **functional and non-functional requirements** of the system in a clear and unambiguous manner. It serves as a **reference for developers**, **designers**, **testers**, and **project stakeholders** throughout the software development lifecycle. This document ensures all parties involved have a shared understanding of the system to be developed.

The intended audience for this document includes:

- **Software Developers** – to implement the features as specified.
- **Test Engineers** – to derive test cases and validate the system behavior.
- **Project Managers** – to monitor progress and ensure delivery within scope.
- **System Architects** – to design a scalable and secure infrastructure.
- **Stakeholders & Clients** – to review and approve the documented system capabilities.
- **End Users (Admin and Contestants)** – indirectly, for understanding how the system works.

1.2 Scope of the System

- User registration, login, and multi-factor authentication
- Contest creation, management, and scheduling
- Problem set management with support for multiple languages (C, C++, Java, Python)
- Code submission via file upload or text editor
- Automated judging and verdict generation
- Real-time leaderboard updates and ranking system

- Profile management for users and admins
- Admin dashboard with user moderation and problem management
- Cheating detection and system security logging
- Support for thousands of concurrent users
- Light blue, minimalistic, user-friendly interface
- Role-based access control (Admin, User)
- Secure, encrypted data handling and authentication
- Real-time feedback and system notifications

1.3 Definitions, Acronyms, and Abbreviations

This section provides definitions for all document names, acronyms, and abbreviations. The application domain's terms and concepts are defined.

Term / Acronym	Description
SRS	Software Requirements Specification
UI / UX	User Interface / User Experience
IDE	Integrated Development Environment (e.g., online text/code editor)
DB	Database
MFA	Multi-Factor Authentication
HTML	HyperText Markup Language – used to structure web pages
CSS	Cascading Style Sheets – used for styling the UI
JavaScript	A scripting language used to implement dynamic behaviors on web pages
Java	A programming language used for backend services and application logic
MySQL	An open-source relational database management system
GitHub	A code hosting platform for version control and collaboration
Figma	A collaborative interface design tool for creating UI prototypes

2. Overall Description

2.1 Product Perspective

Codecatalyst is a standalone web application but may integrate with other services such as email gateways (for verification), external code execution environments (judging systems), and analytics tools (for performance tracking). It follows a **modular client-server architecture**, where the front end communicates with the backend via **RESTful APIs**.

While the system is self-contained, it may optionally integrate with:

- **Version control systems** (e.g., GitHub APIs for importing code)
- **Cloud platforms** for scalability
- **CI/CD pipelines** for future automated deployments

The product will be developed using a modern web stack:

- **Frontend**: HTML, CSS, JavaScript, Figma designs
- **Backend**: Java with Spring Boot
- **Database**: MySQL
- **Authentication**: JWT (JSON Web Tokens)

2.2 Stake Holders and Characteristics

Internal Stakeholders

These are individuals or groups directly involved in the development, maintenance, or operation of the Codecatalyst system.

- **Project Owner / Client**
  - Initiates and funds the project
  - Approves deliverables and defines core objectives
- **Development Team**
  - Responsible for designing, coding, integrating, and deploying the system
  - Ensures that functional and non-functional requirements are implemented correctly
- **UI/UX Designers**
  - Design the frontend interface using tools like Figma
  - Ensure usability, consistency, and accessibility
- **QA Testers**
  - Validate all features through manual and automated testing
  - Ensure bug-free releases and performance reliability
- **System Administrator / DevOps Team**
  - Manages server environments, databases, and deployment pipelines
  - Ensures uptime, performance, and security of the platform

External Stakeholders

These stakeholders interact with the system but are not part of the development or operational team.

- **Contestants / Registered Users**
  - Primary users of the platform who participate in contests, submit code, and view leaderboards
  - Their feedback is crucial for platform improvement
- **Platform Administrators** *(End-user admins)*
  - Manage the problem sets, moderate submissions, and monitor user activity
  - Ensure fair usage and content accuracy
- **Visitors / Guests**
  - Unregistered users who may browse public problems or contest information
  - Potential future users or promoters of the platform
- **Investors / Sponsors** *(optional future role)*
  - May be interested in funding or expanding the platform
  - Expect performance reports, user metrics, and reliability

Stakeholder Matrix:

Interest / Power	High Interest	Low Interest
High Power	Project Owner, User	Investors/Sponsor
Low Power	Developer Team, I/UX Designers , QA Testers	Guest, Marketing

S2.3 Operating Environment

The Codecatalyst system will be deployed in a standard web-based environment and accessed via modern browsers. The system will also be optimized for both desktop and mobile viewports.

Client-side (Frontend):

- Supported Browsers: Chrome, Firefox, Edge, Safari (latest versions)
- Device Compatibility: Desktop, Laptop, Tablet (Responsive Design)

Server-side (Backend):

- OS: Linux-based cloud server (Ubuntu 20.04 or later)
- Web Server: Apache Tomcat or Nginx
- Programming Language: Java 17+
- Framework: Spring Boot
- Database: MySQL
- JavaScript runtime (for frontend tooling): Node.js

2.4 Design and Implementation Constraints

In order to ensure the success of the Codecatalyst project, several design and implementation constraints have been defined. These constraints shape how the system is structured, the technologies used, and the environment in which the system will function. They also ensure consistency across development, testing, deployment, and usage.

2.4.1 Language Constraints

The Codecatalyst platform is divided into frontend and backend systems, each developed using industry-standard technologies:

HTML (Hypertext Markup Language)

HTML is used to structure the visual elements of all web pages, including forms, buttons, tables, and content sections. It defines how information is displayed and navigated.

- Chosen for its universal browser support
- Lightweight and easy to maintain
- Free and open standard, ideal for educational and competitive platforms

CSS (Cascading Style Sheets)

CSS provides styling and layout to the HTML structure. Codecatalyst uses a light blue theme with a minimalistic design approach.

- Enables responsive design for mobile and desktop
- Controls visual consistency across all pages
- Separates presentation from structure

JavaScript

JavaScript enables interactivity and dynamic updates in real-time, especially for pages like leaderboards and submission results.

- Handles client-side validations and animations
- Allows asynchronous data communication via APIs (AJAX, Fetch)
- Widely supported and integrates smoothly with HTML/CSS

Java with Spring Boot

The server-side logic is built using Java and the Spring Boot framework. It handles all business operations, authentication, and data processing.

- Offers built-in security, REST APIs, and modular architecture
- Enables rapid backend development and scalability
- Ideal for complex business logic such as automated judging

SQL (MySQL)

MySQL is used as the relational database to manage structured data such as users, contests, problems, and submission logs.

- Reliable, open-source, and ACID-compliant
- Easily integrated with Spring Boot
- Supports efficient queries and indexing for performance

2.4.2 Tool Constraints

The following tools are mandated for design, development, and documentation:

- **Figma** – For prototyping UI with a consistent, minimalistic look
- **Draw.io** – For creating and exporting DFDs, use case diagrams, etc.
- **VS Code** – Standard IDE for both frontend and backend development
- **Git & GitHub** – For version control, issue tracking, and team collaboration

2.4.3 Platform & Deployment Constraints

- The system will be **web-based only** (no mobile or desktop apps in the MVP).
- Hosted on **Linux-based cloud environments** such as DigitalOcean or Heroku.

- Backend must support **RESTful APIs**, and all communication must be secured using **HTTPS**.
- The application must be compatible with modern browsers: **Chrome, Firefox, Safari, Edge**.
- User sessions must use **JWT** for secure, stateless authentication.

2.4.4 Security & Compliance Constraints

- Passwords must be securely hashed (e.g., using bcrypt or SHA-based algorithms).
- The platform must offer **account deletion** and **data export** options to comply with basic privacy laws (similar to GDPR principles).
- Sandboxing and rate-limiting must be implemented for submitted code to prevent malicious execution.
- Admin access should be protected by **role-based access control (RBAC)** and MFA.

2.4.5 Design Constraints

- The interface must follow **minimalist design principles**, ensuring a clean and distraction-free experience for coders.
- Consistent use of the **light blue color theme**, as defined in Figma designs, must be applied across all pages.
- Pages like homepage, contest lobby, and submission panel must follow **component-based design** for reuse and maintainability.

3. Functional Requirements

3.1 User Registration and Login

FR-1	**Users (participants, problem setters, admins) must be able to register and log into their Codecatalyst accounts.**
Description	A new user should be able to create an account using email, password, or third-party services like Google. Existing users can log in using their credentials. The system must support multi-factor authentication for added security.
Stakeholders	User (Participant, Admin, Problem Setter), System
Priority	High

3.2 Contest Participation

FR-2	**Registered users should be able to view available contests and participate in them.**
Description	The user can browse upcoming or ongoing contests, register for participation, and be redirected to the contest interface during the scheduled time. Participants must agree to the contest rules before starting.
Stakeholders	User (Participant), System
Priority	High

3.3 Code Submission and Judging

FR-3	**Participants can submit solutions for problems, which will be automatically judged.**
Description	Users submit their code through a web editor or file upload. The system compiles and runs the code against test cases. Based on the output, results (e.g., Accepted, Wrong Answer, Time Limit Exceeded) are shown in real time.
Stakeholders	User (Participant), System
Priority	High

3.4 Leaderboard and Results

FR-4	**Users should be able to view live and final leaderboards of contests.**
Description	The system displays participant rankings based on performance. Points, penalties, and submission time are calculated dynamically. Users can view both ongoing and archived results.
Stakeholders	User (Participant, Admin), System
Priority	Medium

3.5 Problem Management (Admin)

FR-5	**Admin and authorized users can create, update, or delete problems.**
Description	The admin dashboard provides an interface for managing problem statements, input/output formats, constraints, and test cases. Problems can be assigned to contests or stored in the general problemset.
Stakeholders	Admin, Problem Setter, System
Priority	High

3.6 User and Role Management (Admin)

FR-6	**Admin can manage users, assign roles (e.g., moderator, setter), and deactivate accounts.**
Description	From the admin dashboard, roles such as Contest Moderator or Problem Setter can be assigned. Admins can ban users, reset passwords, or escalate permissions based on system roles.
Stakeholders	Admin, System
Priority	High

3.7 Profile Management

FR-7	**Users should be able to update personal info and change their password.**
Description	Users access a personal dashboard where they can view or update their name, email, and profile picture. Password reset is supported through verification.
Stakeholders	User, System
Priority	Medium

3.8 Code Editor and Language Support

FR-8	**The platform should support multiple programming languages and a real-time editor.**
Description	Users may choose from C, C++, Java, or Python. The editor supports syntax highlighting and basic formatting. Alternatively, files may be uploaded directly.
Stakeholders	User, System
Priority	Medium

4. Non-Functional Requirements

4.1 Performance

NFR-1	**The system should support fast loading times and low latency.**
Description	Pages, especially the contest interface and leaderboard, must load in under 2 seconds. Judging should be completed within 1–2 seconds per test case.

Stakeholders	User, System
Priority	High

4.2 Scalability

NFR-2	**The platform must be able to handle thousands of concurrent users.**
Description	Codecatalyst should function reliably even during peak usage (e.g., during large contests), ensuring stability and performance.
Stakeholders	System Admin, User
Priority	High

4.3 Security

NFR-3	**The system must ensure secure access and data protection.**
Description	Multi-factor authentication, encrypted passwords, secure file submissions, and protection from malicious code must be implemented.
Stakeholders	All Users, Admin
Priority	High

4.4 Usability and UX

NFR-4	**The user interface should be easy to use and responsive.**
Description	The platform must be intuitive, mobile-friendly, and include real-time updates (e.g., submission status, score updates). A dark/light theme toggle may be provided.
Stakeholders	User, System
Priority	Medium

4.5 Availability

NFR-5	**The system should be available all the time.**

Description	Downtime should be minimized. Scheduled maintenance should be announced in advance.
Stakeholders	User, Admin
Priority	High

4.6 Maintainability and Extensibility

NFR-6	**The system should allow for easy updates and integration of new features.**
Description	The architecture must support modular updates, such as adding new languages or AI-based judging in future versions.
Stakeholders	Developer, System
Priority	Medium

5. Requirement Prioritization

We used the **MoSCoW prioritization technique** to categorize each requirement into:

- **M** – Must have
- **S** – Should have
- **C** – Could have
- **W** – Won't have (at this time)

5.1 Prioritization Table

ID	Requirement Summary	Category	MoSCoW Priority
FR-1	User Registration and Login	Functional	**Must**
FR-2	Contest Participation	Functional	**Must**
FR-3	Code Submission and Judging	Functional	**Must**
FR-4	Leaderboard and Results	Functional	**Should**
FR-5	Problem Management (Admin)	Functional	**Must**
FR-6	User and Role Management (Admin)	Functional	**Must**
FR-7	Profile Management	Functional	**Should**
FR-8	Code Editor and Language Support	Functional	**Should**
NFR-1	Performance	Non-Functional	**Must**
NFR-2	Scalability	Non-Functional	**Must**

NFR-3	Security	Non-Functional	**Must**
NFR-4	Usability and UX	Non-Functional	**Should**
NFR-5	Availability	Non-Functional	**Must**
NFR-6	Maintainability and Extensibility	Non-Functional	**Should**

6. Validation and Verification Plan

6.1 Validation Techniques:

Validation ensures that the product built is the right product — it meets the needs of the end users and stakeholders.

Technique	**Description**
Walkthroughs	Step-by-step review sessions involving developers, QA, and stakeholders to validate requirement understanding and design logic.
User Feedback	Collect real-time feedback to verify user satisfaction and usability.
Stakeholder Reviews	Regular validation with stakeholders to ensure requirements align with business goals.

6.2 Verification Techniques

Verification ensures that the product is built correctly — it meets the documented specifications.

Technique	**Description**
Inspections	Manual review of code, documents, and interfaces to verify compliance with standards.

Unit Testing	Test individual modules (e.g., login, code submission) for expected behavior.
Traceability Matrix	Ensure that all requirements are covered by test cases and validated thoroughly.

6.3 Requirement Traceability Matrix (RTM)

Requirement ID	Requirement Description	Test Case ID	Verification Method	Progress
FR-1	User Registration and Login	TC-01	Login form testing, MFA	Completed
FR-2	Contest Participation	TC-02	Simulated contest entry	Completed
FR-3	Code Submission and Judging	TC-03	Multiple language testing	Completed
FR-4	Leaderboard and Results	TC-04	Real-time update check	Completed
FR-5	Problem Management (Admin)	TC-05	Admin panel inputs	In Progress
FR-6	Role Management	TC-06	Role permission testing	Not Started
FR-7	Profile Management	TC-07	Edit profile verification	Completed
FR-8	Code Editor + Multi-language Support	TC-08	Editor and language tests	In Progress
NFR-1	Performance	TC-09	Load & latency benchmarks	In Progress
NFR-2	Scalability	TC-10	Simulate 500+ users	Not Started
NFR-3	Security	TC-11	Penetration test, SSL	Completed
NFR-4	Usability	TC-12	UI consistency review	Completed
NFR-5	Availability	TC-13	Uptime monitoring	Completed
NFR-6	Maintainability and Extensibility	TC-14	Code modularity analysis	Not Started

7. UML Diagrams

7.1 Class Diagram for Codecatalyst

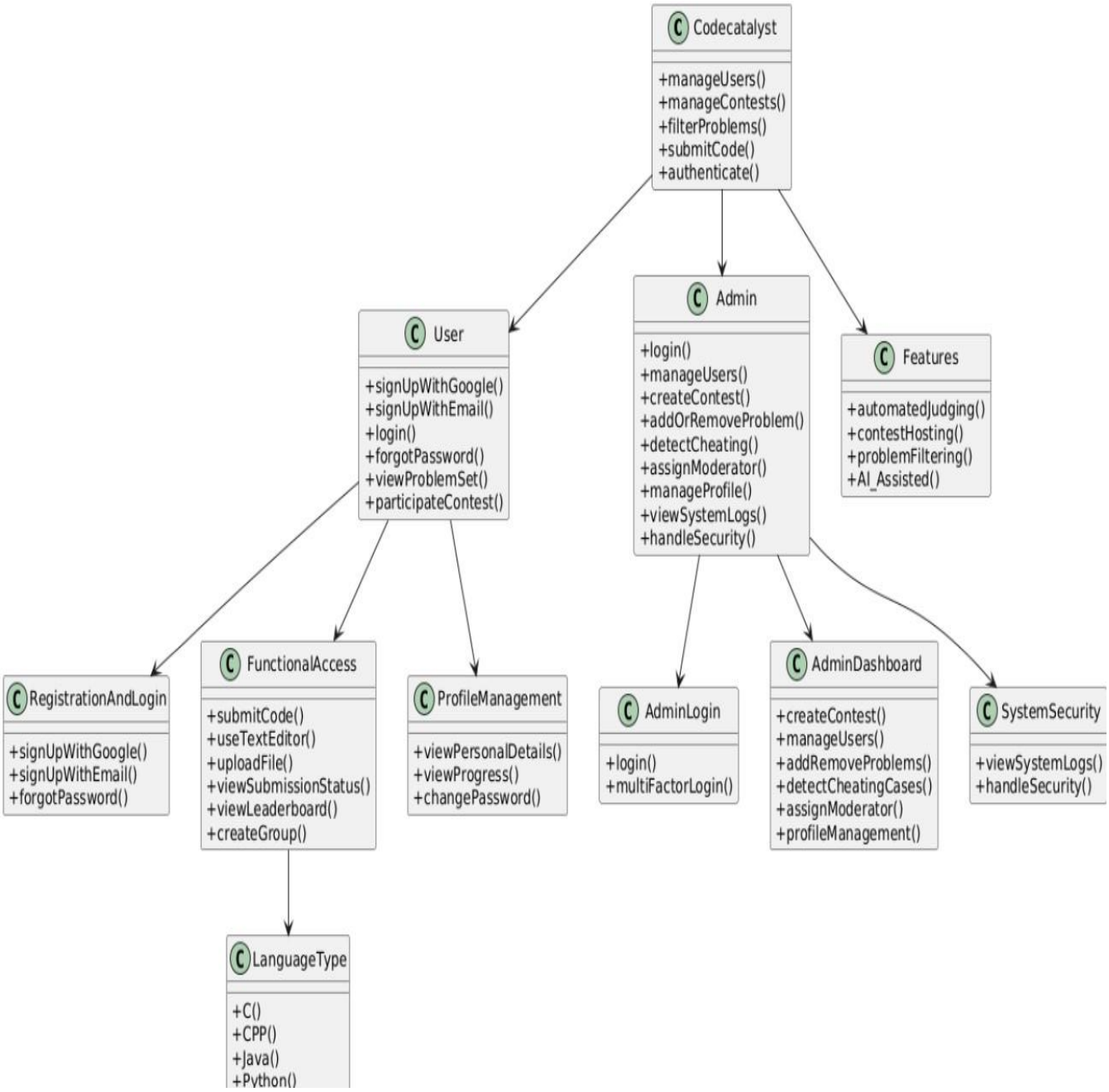


Figure 1: Class Diagram

7.2 Data Flow Diagram for Codecatalyst:

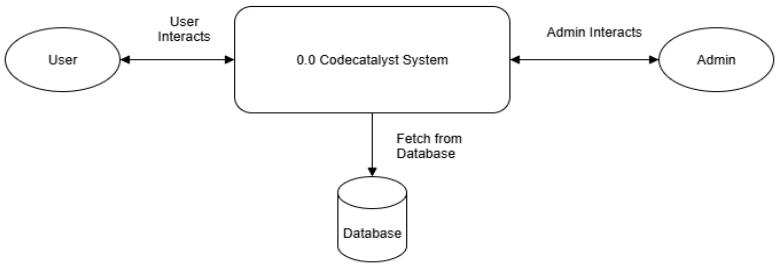


Fig: Level 0 DFD

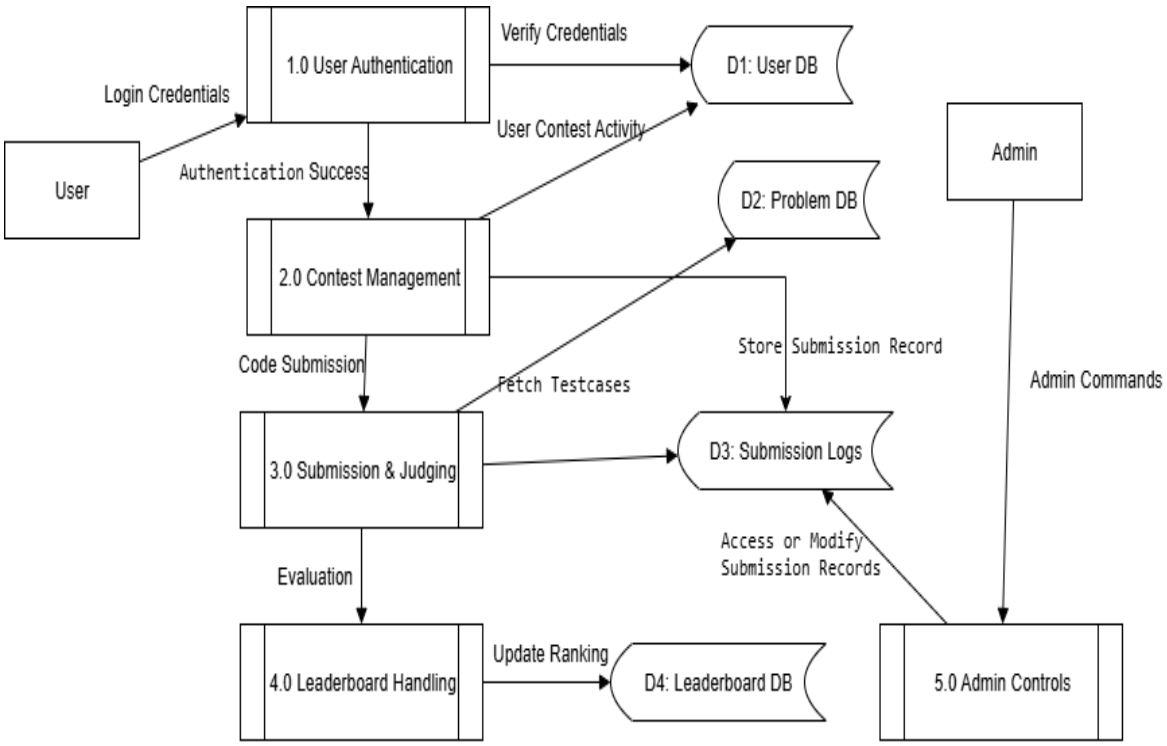


Fig: Level 1 DFD

Figure 2: Data Flow Diagram

7.3 Use Case Diagram for CodeCatalyst:

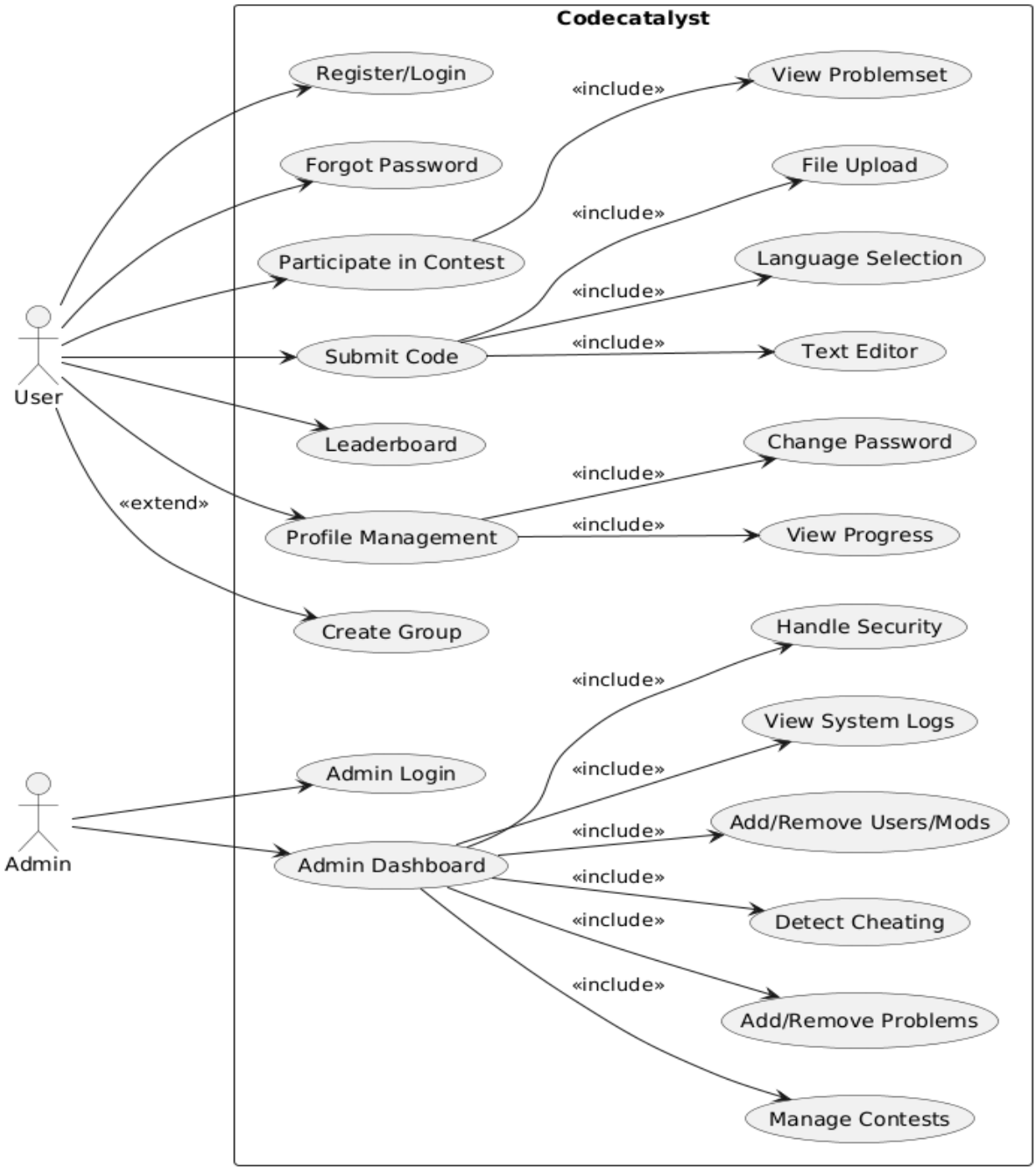


Figure 3:Use Case Diagram

Use case description:

Use Case-01: Register/Login

Field	Description
Use Case Name	Register/Login
Actor	User
Description	User can create an account or log into the system.
Includes	View Problemset, File Upload
Precondition	User is not logged in
Postcondition	User is authenticated and logged into the system
Trigger	User opens the login or registration page
Main Flow	1. User accesses login/registration form 2. Enters email and password 3. Submits form 4. System verifies credentials 5. Redirects to dashboard
Alternative Flow	1. User enters wrong credentials → error message displayed 2. Email already exists → prompt to log in 3. Server error → try again later

Use Case-02: Forgot Password

Field	Description
Use Case Name	Forgot Password
Actor	User
Description	Allows user to reset their password via email/OTP.
Includes	–
Precondition	User provides valid recovery credentials
Postcondition	Password is reset
Trigger	User clicks on "Forgot Password" link
Main Flow	1. User opens forgot password page 2. Enters registered email 3. Receives OTP or reset link 4. Resets password

Alternative Flow	1. User enters unregistered email → error message 2. OTP expired → resend option 3. Server down → try again later

Use Case-03: Participate in Contest

Field	Description
Use Case Name	Participate in Contest
Actor	User
Description	Users can register and take part in live or scheduled contests.
Includes	–
Precondition	User must be logged in
Postcondition	Contest interface is shown
Trigger	User navigates to contest list and clicks "Join"
Main Flow	1. User browses contests 2. Clicks "Register" 3. Accepts rules 4. Joins contest interface at scheduled time
Alternative Flow	1. Contest full → user notified 2. Contest expired → show message 3. Already joined → redirect to waiting screen

Use Case-04: Submit Code

Field	Description
Use Case Name	Submit Code
Actor	User
Description	Allows users to write and submit code for evaluation.
Includes	Language Selection, Text Editor
Precondition	A contest or problem must be active
Postcondition	Code is compiled and output/error is returned
Trigger	User clicks "Submit" button after writing/uploading code
Main Flow	1. User selects problem 2. Writes/submits code 3. Chooses language 4. Clicks submit 5. System compiles & shows result
Alternative Flow	1. Compilation error → show error message 2. Language unsupported → prompt user 3. Timeout → suggest optimization

Use Case-05: Leaderboard

Field	Description
Use Case Name	Leaderboard
Actor	User
Description	Displays ranking of users based on scores/time.
Includes	–
Precondition	Contest must exist
Postcondition	Leaderboard shown
Trigger	User clicks "Leaderboard"
Main Flow	1. System fetches scores 2. Ranks participants 3. Displays time/penalty/scores 4. Shows final/live status
Alternative Flow	1. Leaderboard not available → show message 2. No participants yet → show empty state

Use Case-06: Profile Management

Field	Description
Use Case Name	Profile Management
Actor	User
Description	Users can update profile information.
Includes	Change Password, View Progress
Precondition	User is logged in
Postcondition	Profile updated
Trigger	User clicks "Profile" or "Settings"
Main Flow	1. User accesses profile 2. Updates name/email/password 3. Submits changes 4. System saves updates
Alternative Flow	1. Invalid input → show error 2. No changes detected → notify user 3. Session expired → prompt re-login

Use Case-07: Create Group

Field	Description
Use Case Name	Create Group
Actor	User
Description	Allows users to create discussion/study/practice groups.
Includes	–
Precondition	User is authenticated
Postcondition	Group is created
Trigger	User clicks on “Create Group” button

Main Flow	1. User opens the group section 2. Clicks “Create Group” 3. Enters group name/details 4. Submits
Alternative Flow	1. Group name exists → show error 2. Empty name → ask for input 3. Submission fails → retry

Use Case-08: Admin Login

Field	Description
Use Case Name	Admin Login
Actor	Admin
Description	Allows admin to log in with credentials.
Includes	–
Precondition	Valid admin credentials
Postcondition	Admin logged in
Trigger	Admin visits login page and submits credentials
Main Flow	1. Admin enters username/password 2. Clicks login 3. System verifies
Alternative Flow	1. Invalid credentials → error 2. Too many attempts → lockout timer

Use Case-09: Admin Dashboard

Field	Description
Use Case Name	Admin Dashboard
Actor	Admin
Description	Admin control panel to manage users, problems, and contests.
Includes	All admin functionalities below
Precondition	Admin logged in
Postcondition	Admin functionalities available
Trigger	Admin selects “Dashboard” after login
Main Flow	1. Admin opens dashboard 2. Accesses different modules (users, problems, contests)
Alternative Flow	1. Server down → show error 2. Permissions mismatch → restrict access

Use Case-10: Add/Remove Users/Mods

Field	Description
Use Case Name	Add/Remove Users/Mods
Actor	Admin
Description	Admin can create or delete user/mod accounts.
Includes	–
Precondition	Admin privileges
Postcondition	User/moderator list updated
Trigger	Admin clicks “Manage Users”
Main Flow	1. Admin views user list 2. Selects user 3. Adds/modifies/removes user
Alternative Flow	1. Action fails → retry 2. Unauthorized action → show warning

Use Case-11: Add/Remove Problems

Field	Description
Use Case Name	Add/Remove Problems
Actor	Admin
Description	Admin can upload or remove problems.
Includes	–
Precondition	Admin privileges
Postcondition	Problemset updated
Trigger	Admin accesses problem management
Main Flow	1. Admin clicks “Add Problem” 2. Enters details 3. Submits 4. Problem added or removed
Alternative Flow	1. Incomplete info → show error 2. Duplicate problem ID → prompt to change

Use Case-12: Manage Contests

Field	Description
Use Case Name	Manage Contests
Actor	Admin
Description	Admin creates, updates, or deletes coding contests.
Includes	–
Precondition	Admin privileges

Postcondition	Contests managed
Trigger	Admin clicks "Contest Management"
Main Flow	1. Admin creates new contest 2. Adds problems 3. Sets time 4. Publishes
Alternative Flow	1. Time overlaps → show conflict 2. Missing problem list → prevent publication

Use Case-13: Detect Cheating

Field	Description
Use Case Name	Detect Cheating
Actor	Admin
Description	System detects and flags cheating using algorithms.
Includes	–
Precondition	Submissions must exist
Postcondition	Suspicious cases flagged
Trigger	Admin opens submission analysis or automated system runs
Main Flow	1. System compares code 2. Finds similarity 3. Flags suspicious entries
Alternative Flow	1. False positives → allow manual override 2. No match → no action

Use Case-14: Handle Security

Field	Description
Use Case Name	Handle Security
Actor	Admin
Description	Monitor and take actions against threats.
Includes	–
Precondition	Admin access
Postcondition	Issues resolved, system secured
Trigger	Security alert or admin opens security module
Main Flow	1. Admin views logs 2. Checks activity 3. Blocks users/IP 4. Takes corrective action
Alternative Flow	1. No access to logs → contact higher admin 2. Issue false alarm → dismiss manually

Use Case-15: View System Logs

Field	Description
Use Case Name	View System Logs
Actor	Admin
Description	Admin can view logs of system/user activities.
Includes	–
Precondition	Admin privileges
Postcondition	Logs retrieved
Trigger	Admin selects “View Logs” from dashboard
Main Flow	1. Admin filters logs 2. Views activities chronologically 3. Exports/downloads if needed
Alternative Flow	1. Logs corrupted → show error 2. No activity → show “empty” state

7.4 Sequence Diagram for Codecatalyst

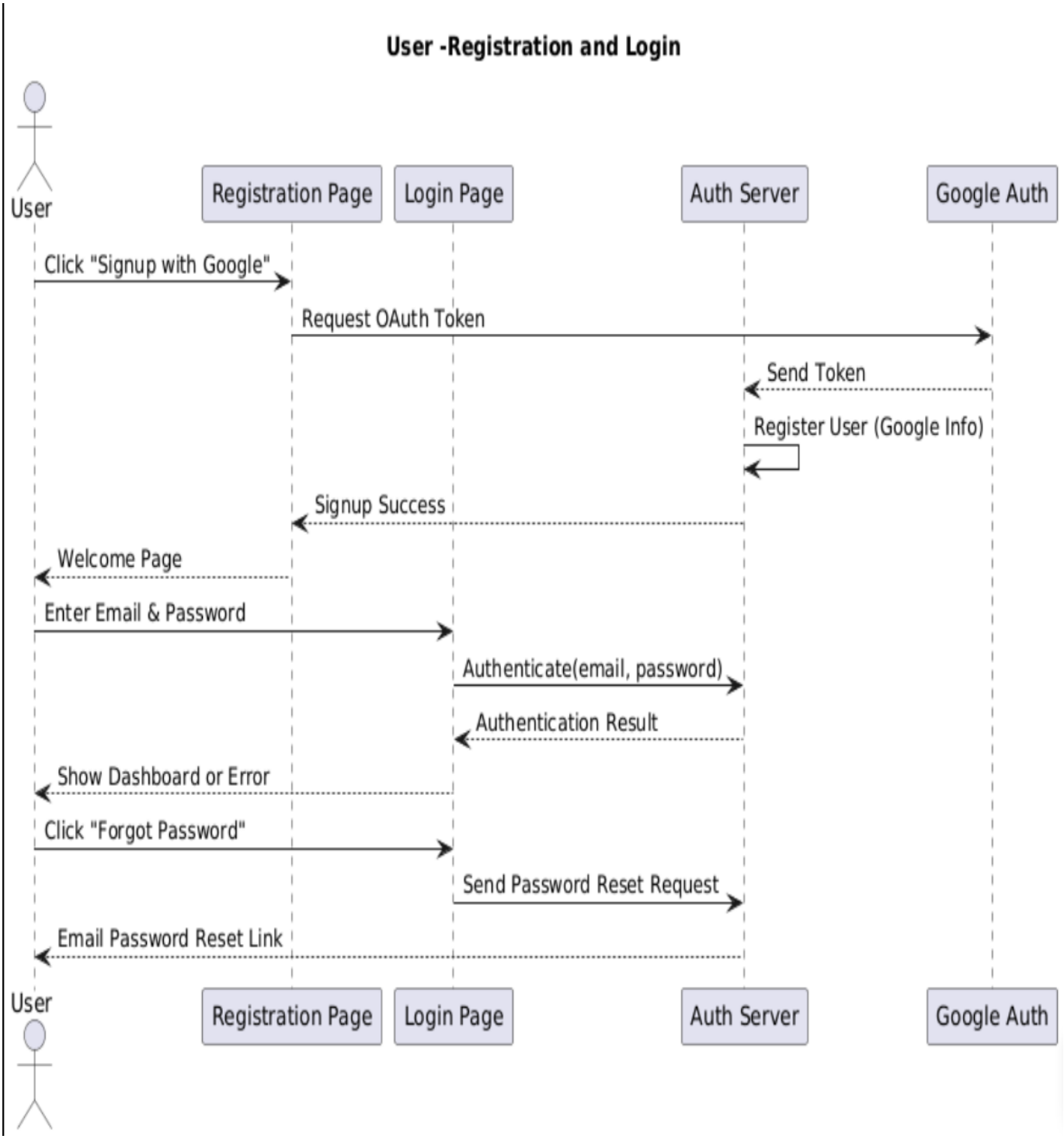


Figure 4: Registration and Login

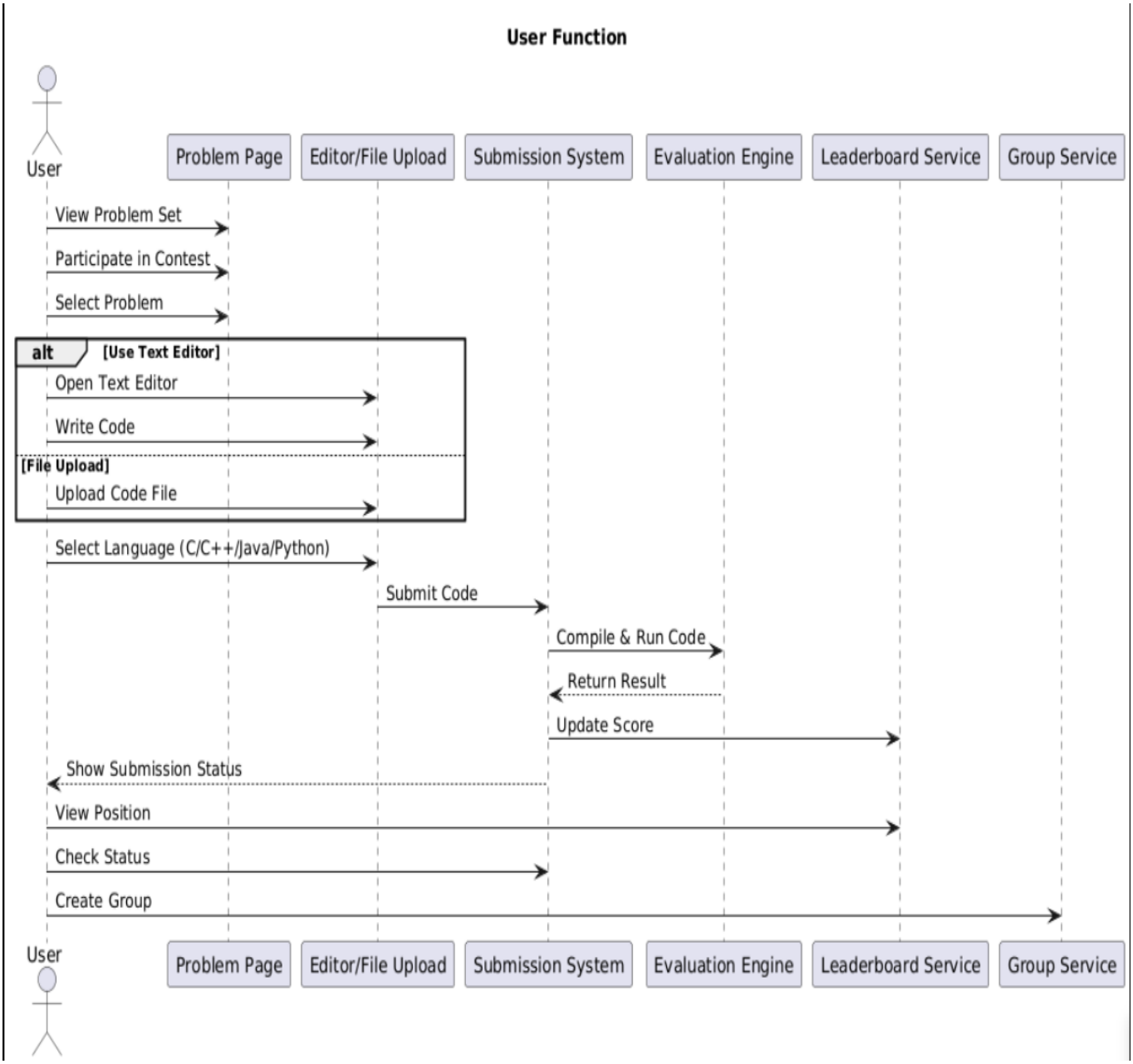


Figure 5: User Function

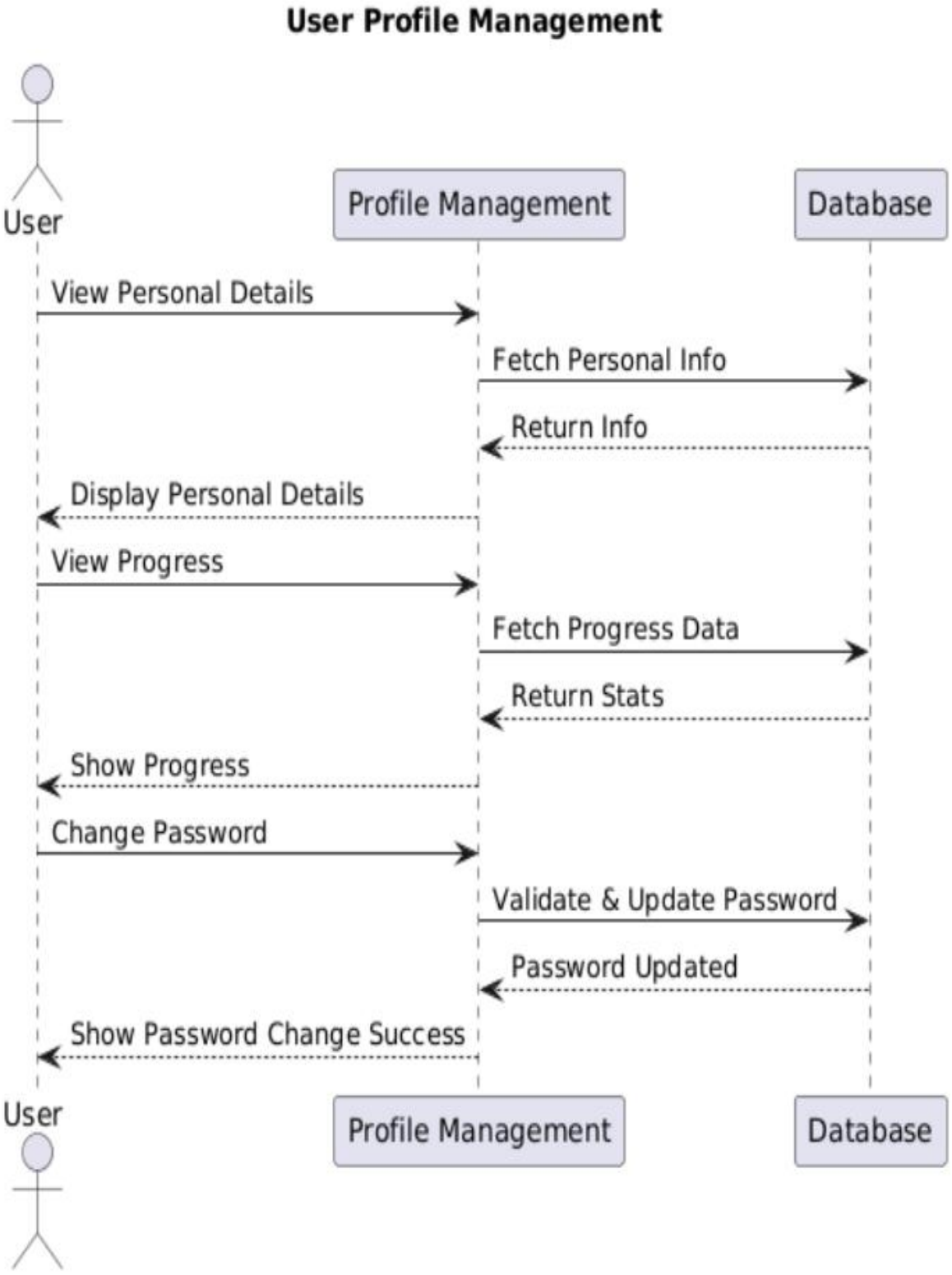


Figure 6: User Profile Management

Poblem Filtering & Sorting

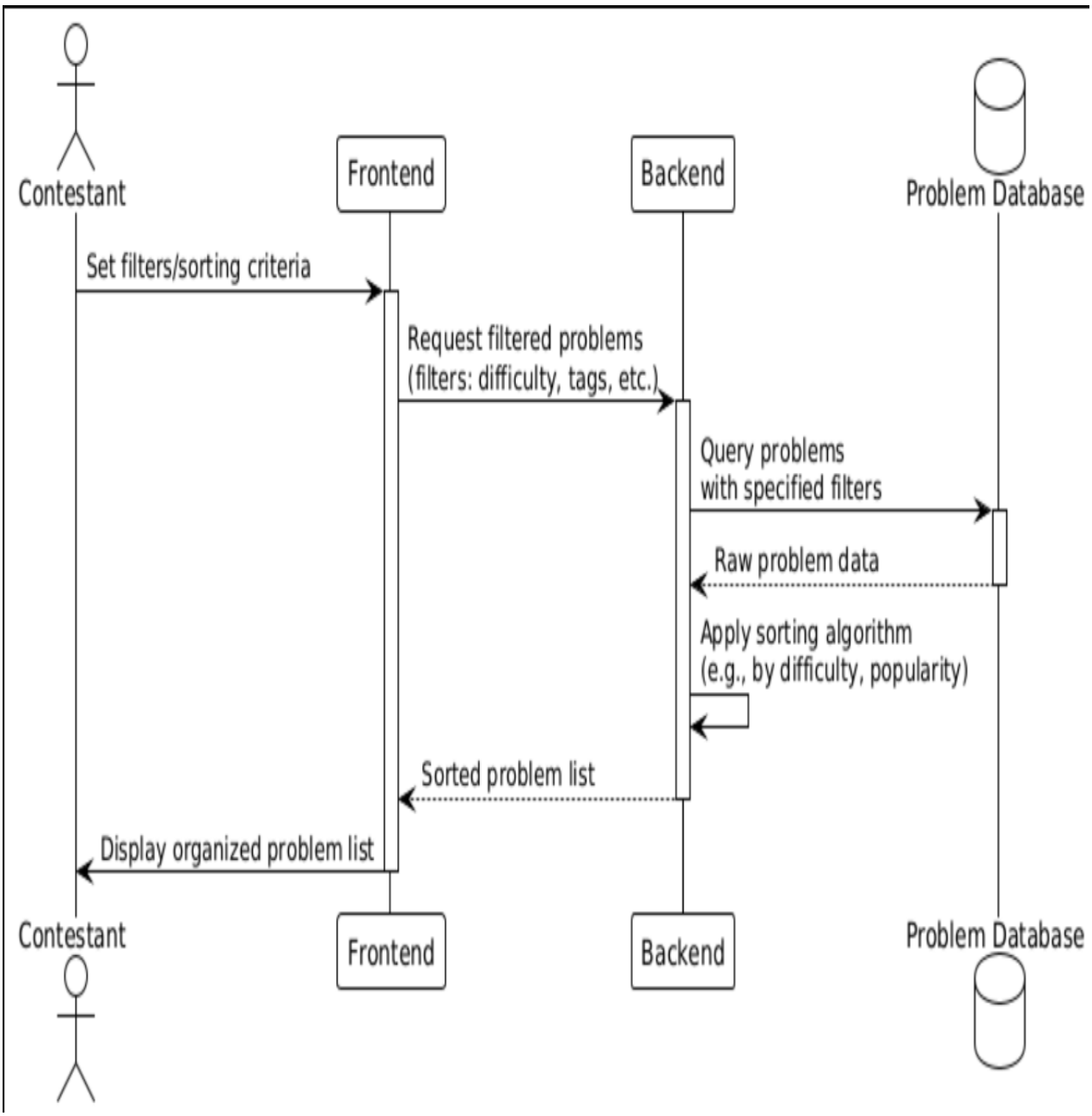


Figure 7: Filtering

Admin Login

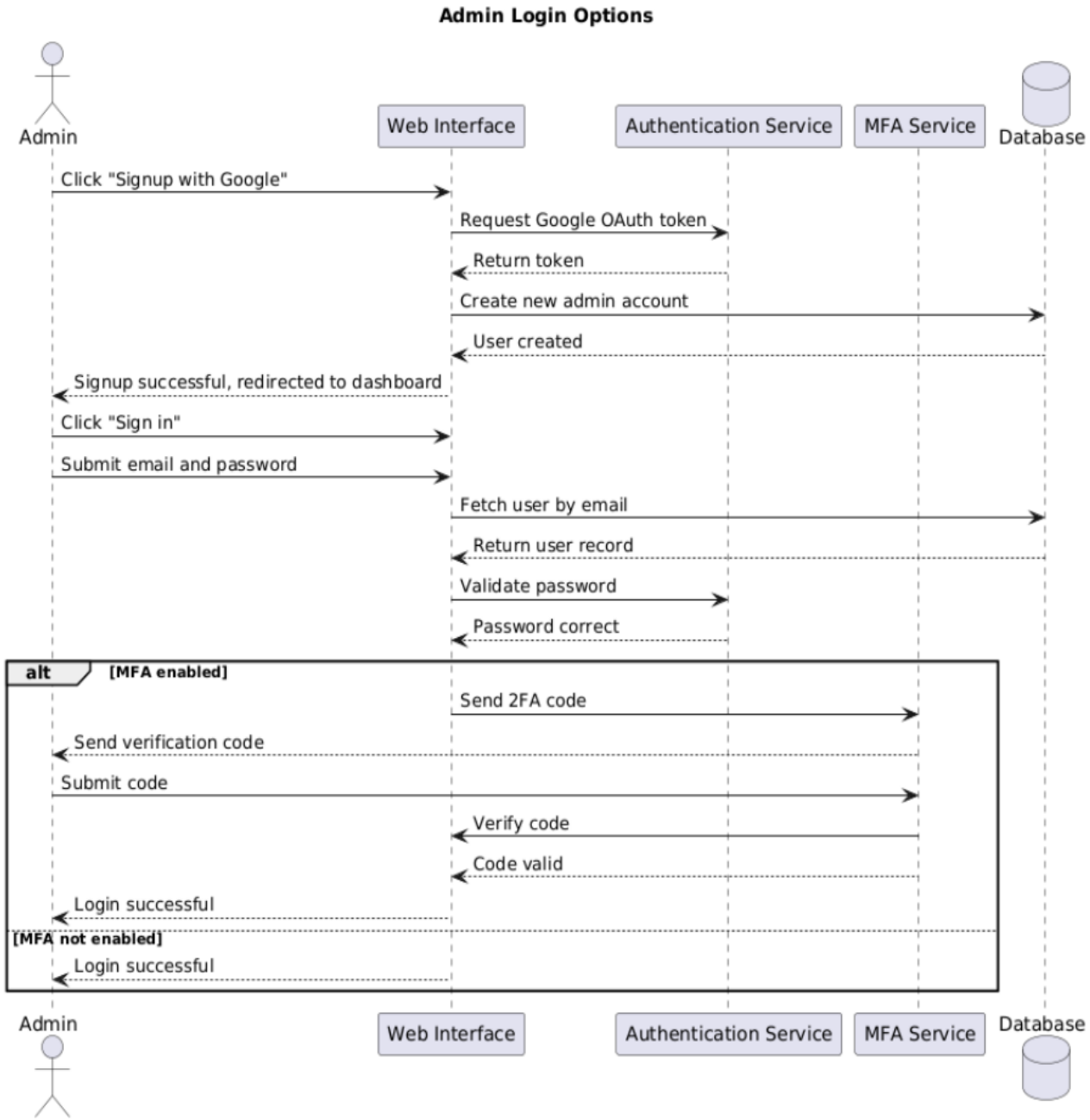


Figure 8: Admin Login

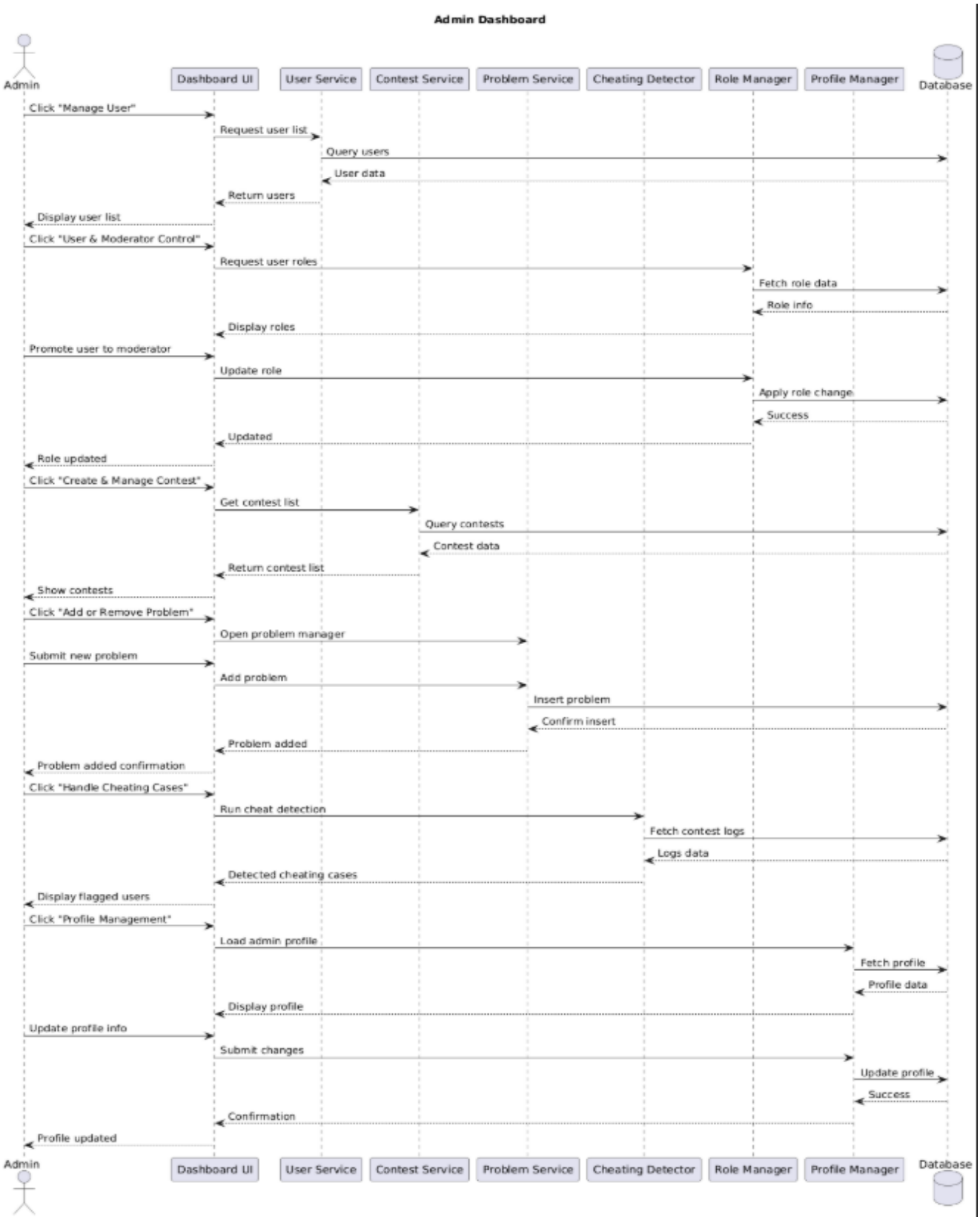


Figure 9: Admin Dashboard

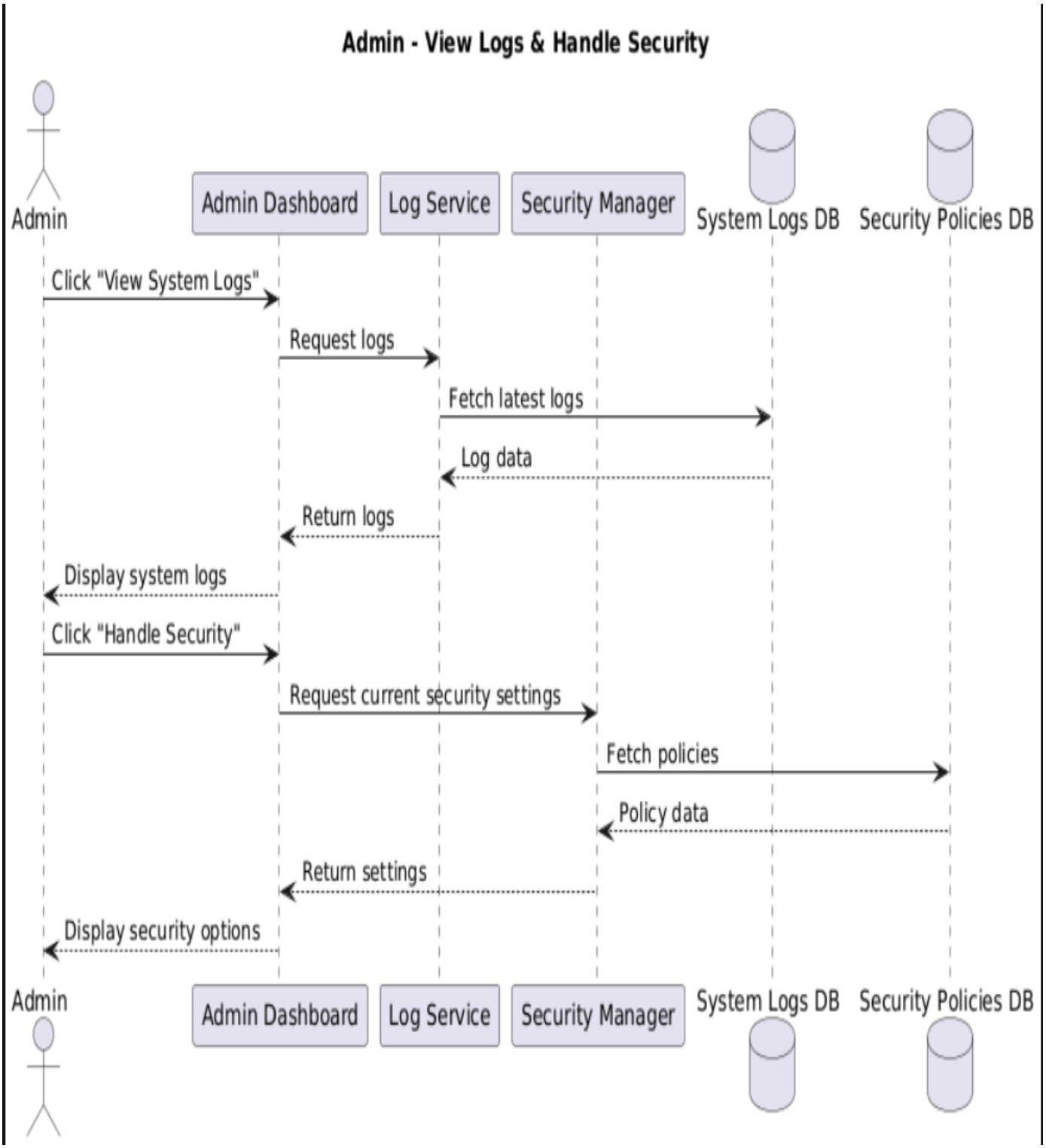


Figure 70: View Logs

Judging Function

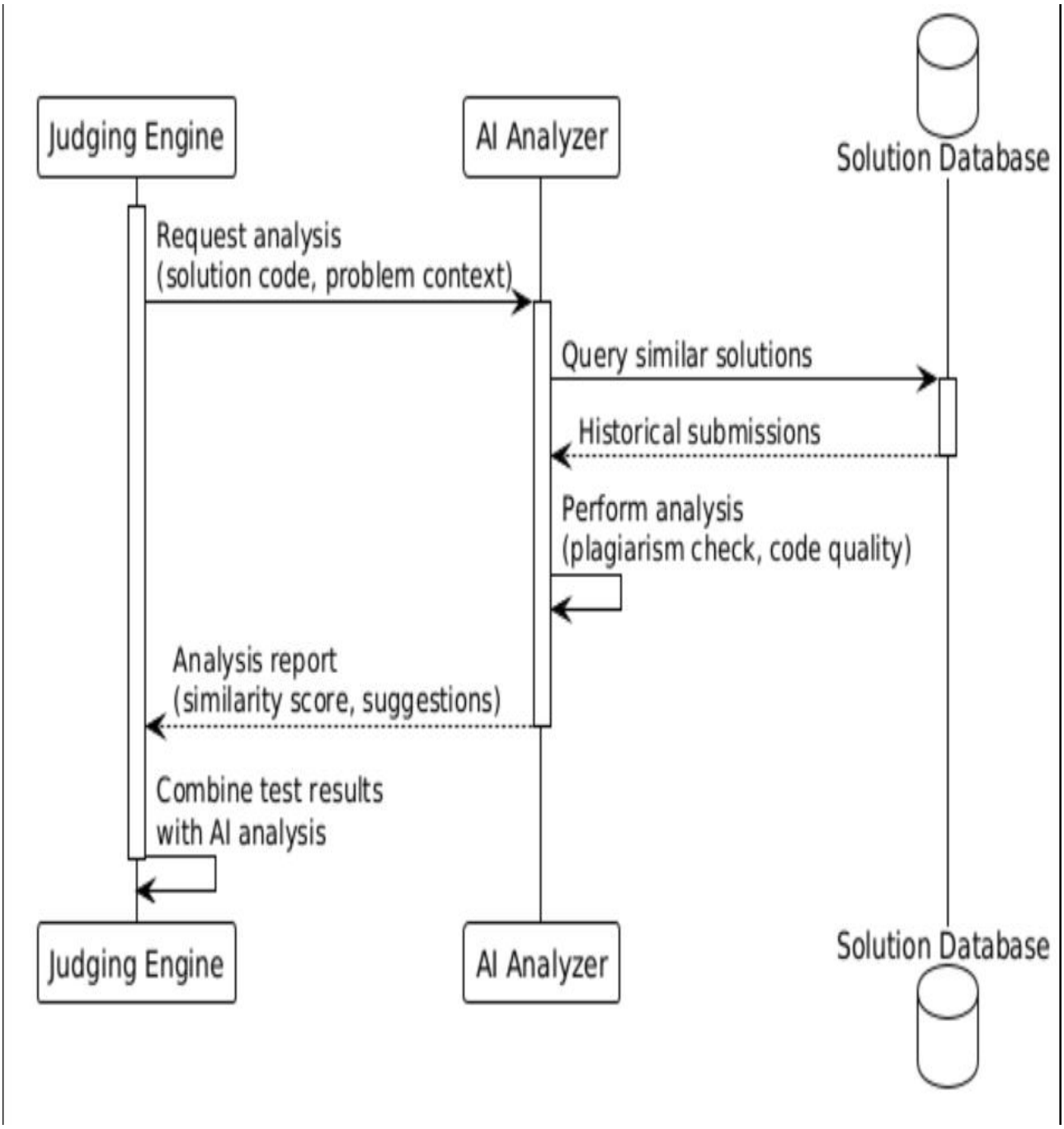


Figure 1: Judging Function

7.5 Activity Diagram for Codecatalyst:

User Registration & Login Flow - Codecatalyst

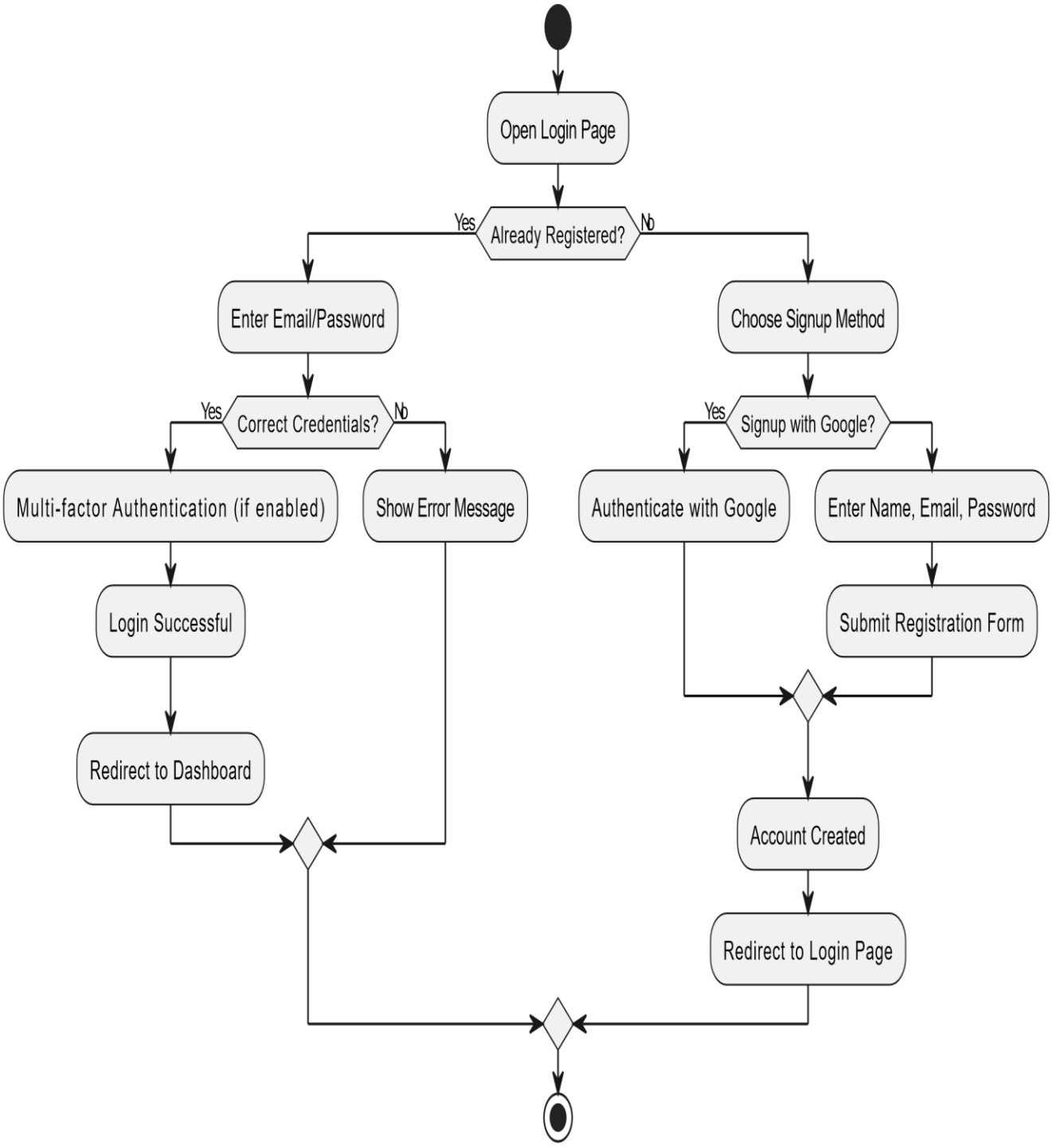


Figure 12: Registration & Login(Sequence)`

User Contest Participation Flow – Codecatalyst

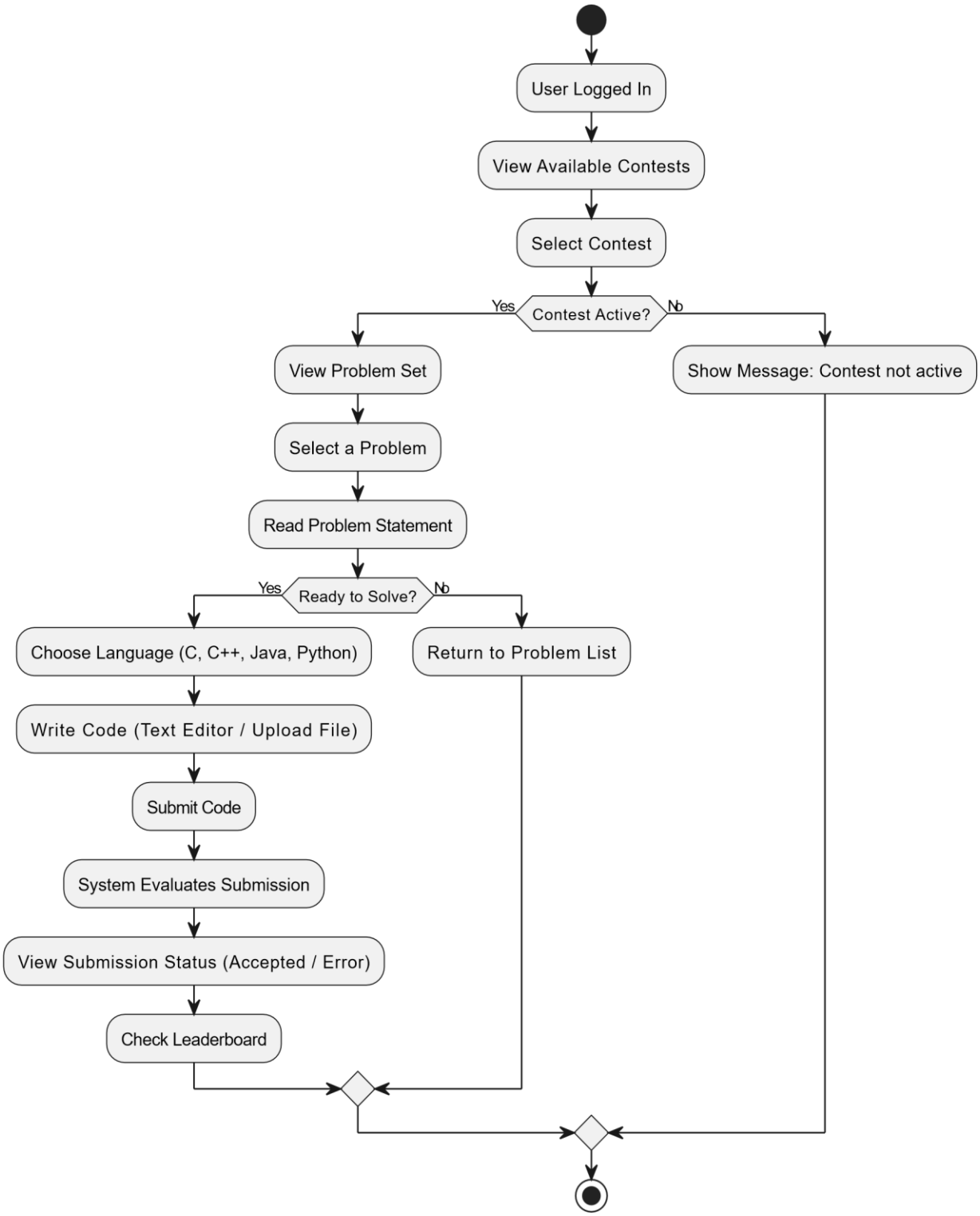


Figure 13: Contest Participation(Sequence)

Admin Login and Dashboard Access – Codecatalyst

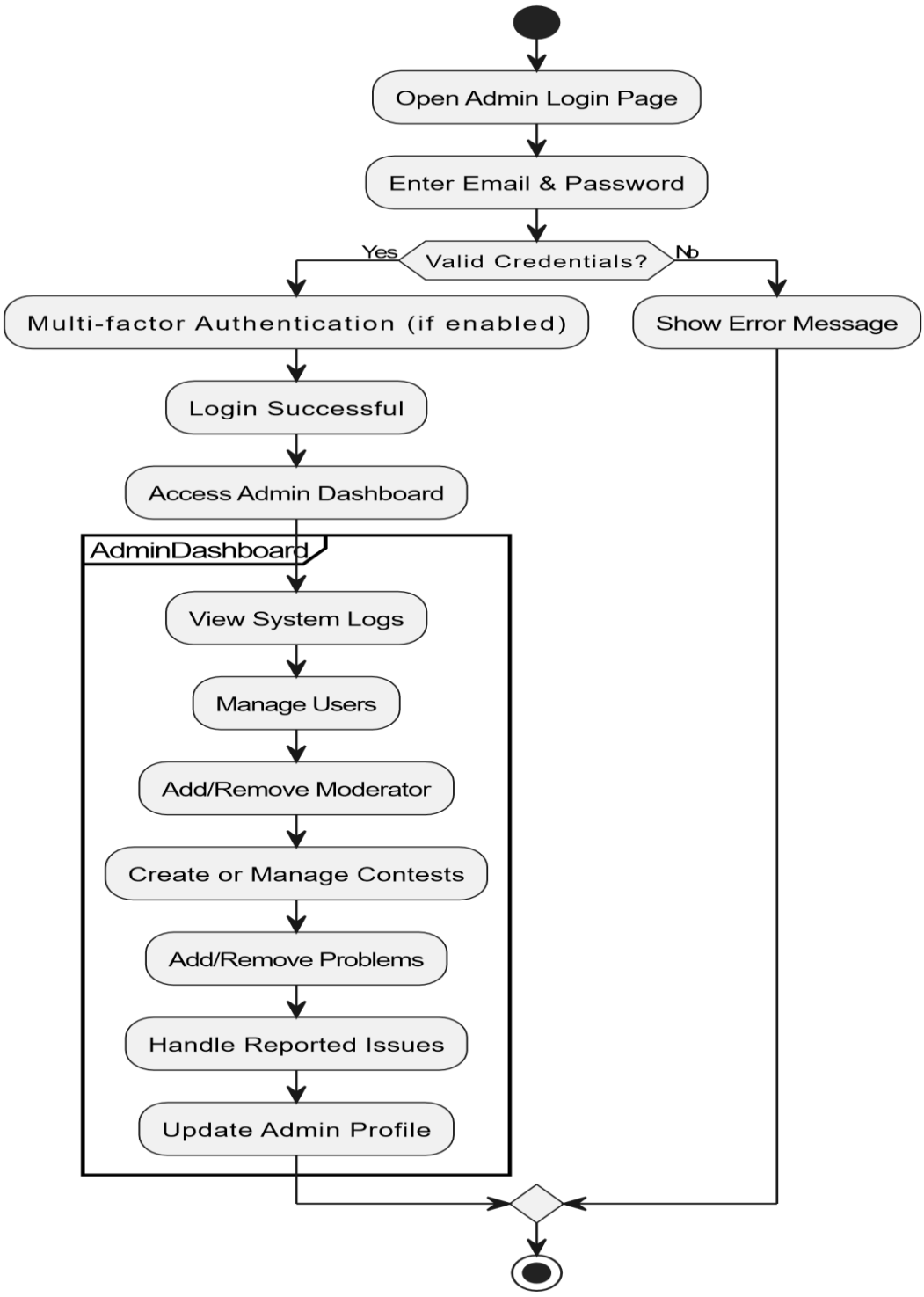


Figure 14: Admin Login & Dashboard(Sequence)

Problem Management Flow - Admin Side (Codecatalyst)

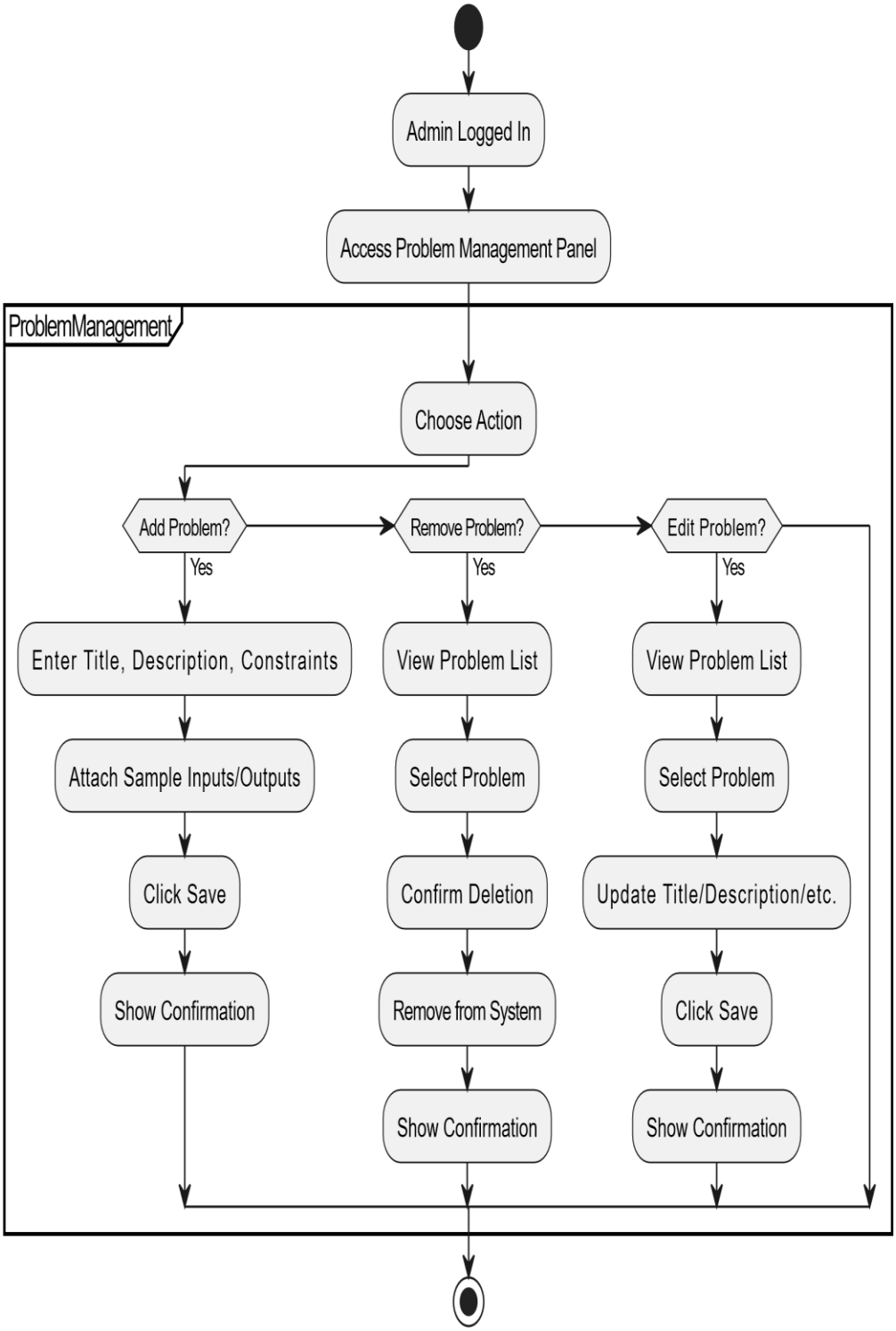


Figure 15: Problem Management(Sequence)

Code Submission Flow - User Side (Codecatalyst)

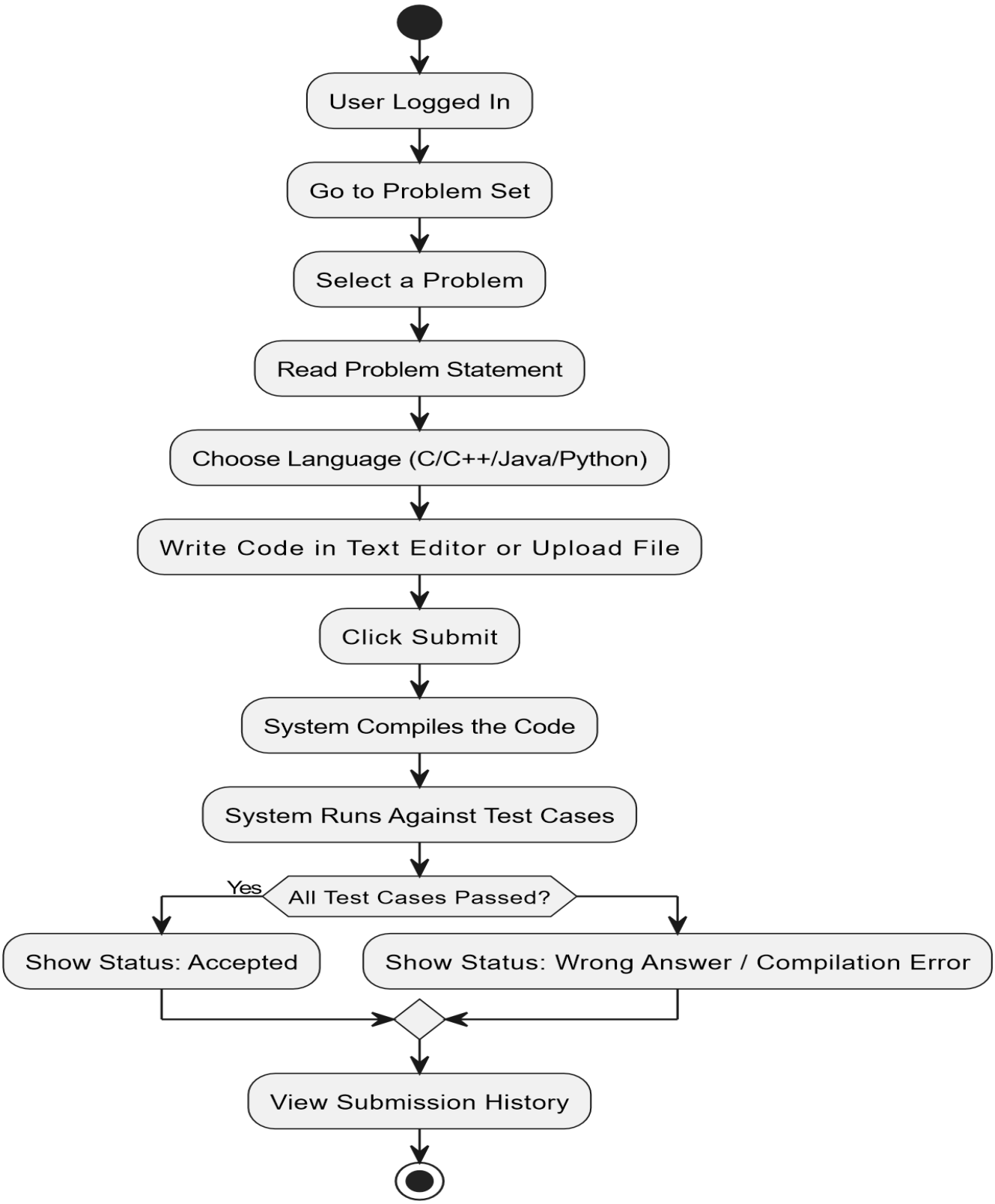


Figure 16: Code Submission(Sequence)

Profile Management Flow - User Side (Codecatalyst)

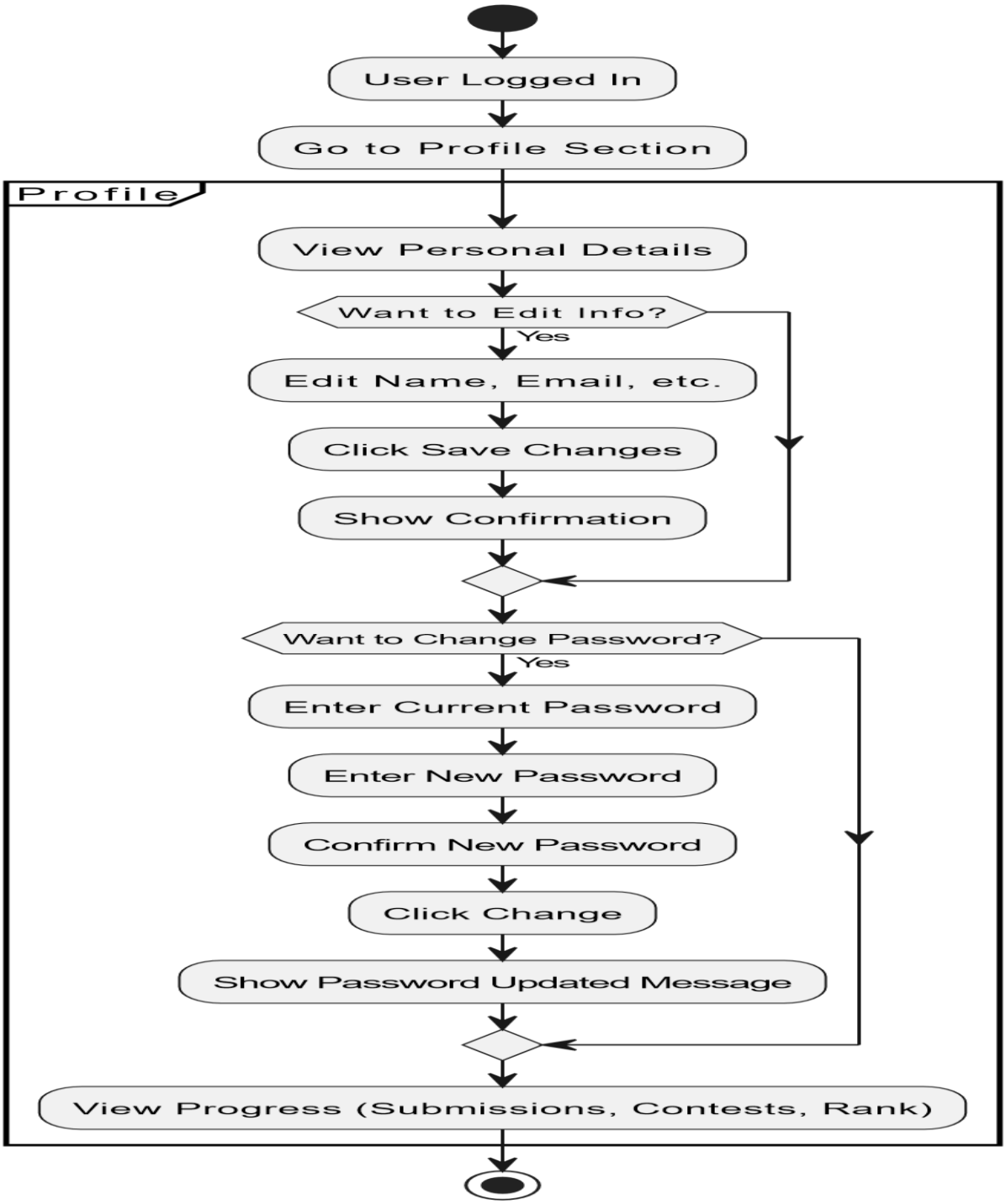


Figure 17: Profile Management(Sequence)

Leaderboard & Submission Status Flow - User Side (Codecatalyst)

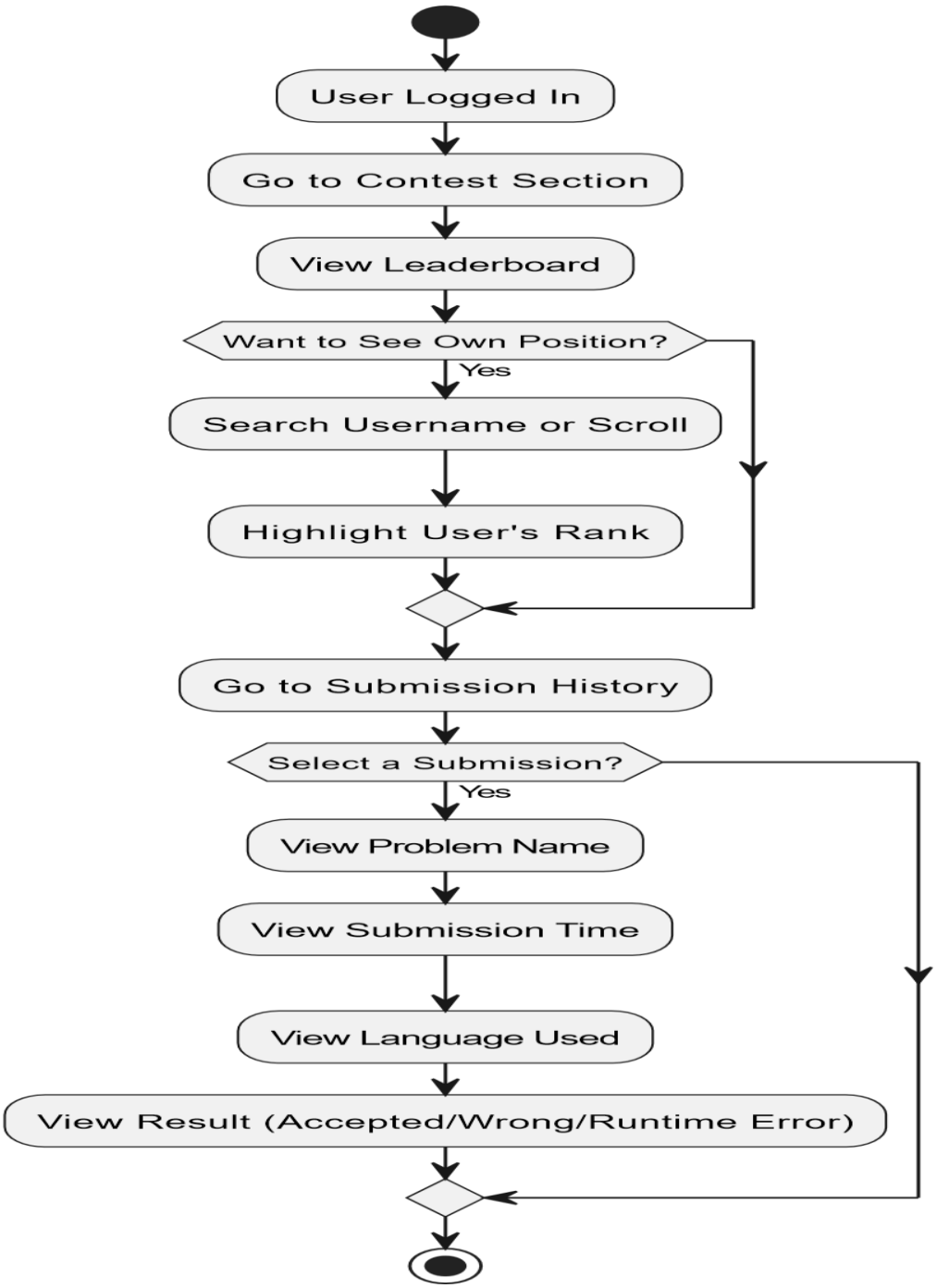
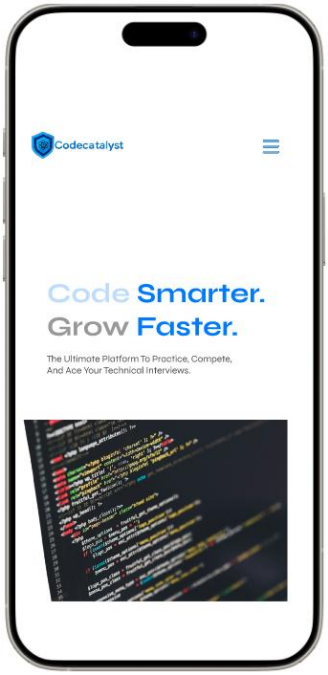


Figure 18: Leaderboard(Sequence)

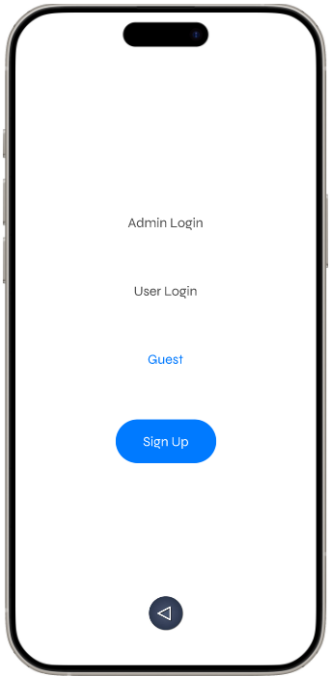
8. Prototyping

8.1 Low-Fidelity Prototype

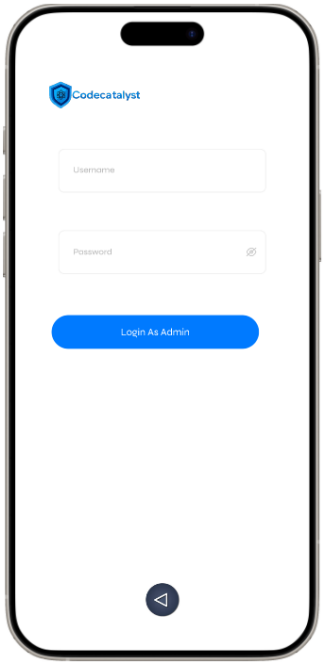
Main Page



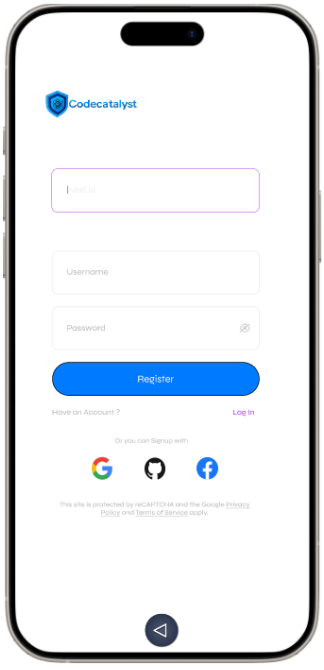
Main Page Navigation



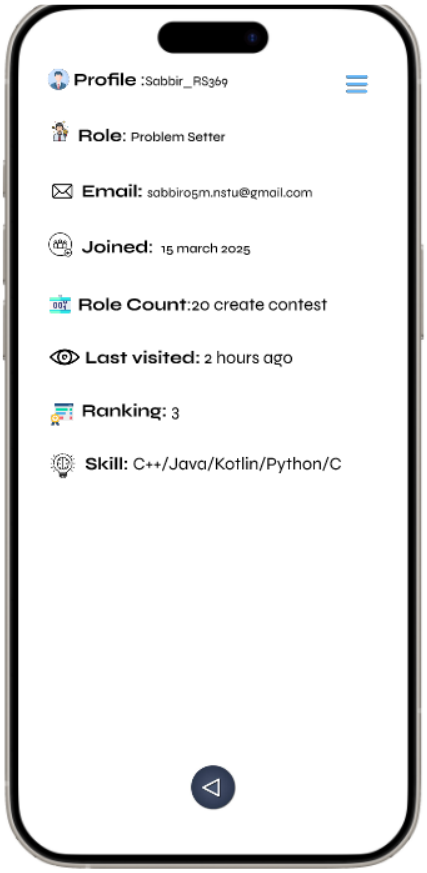
Admin Login Page



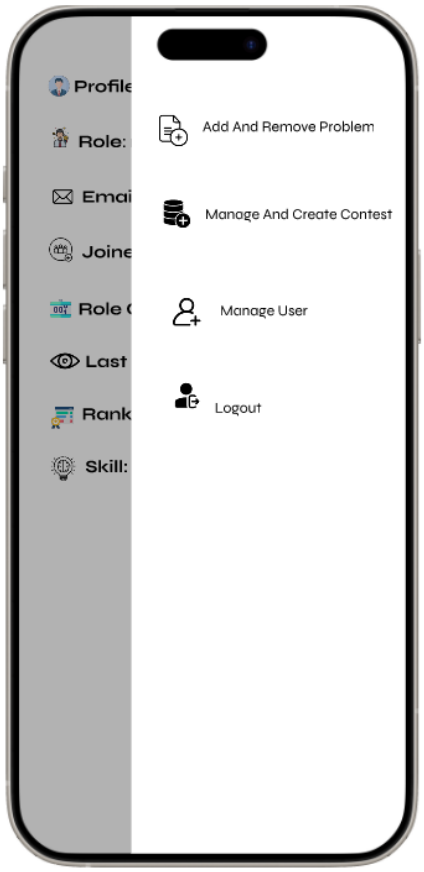
User Registration Page



Admin Profile

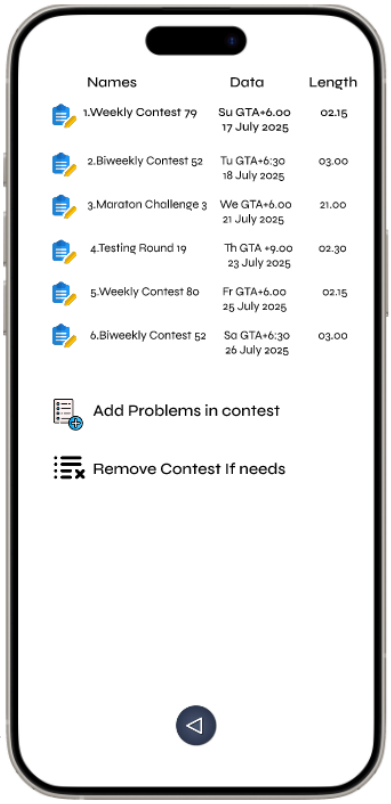


Admin Navigation

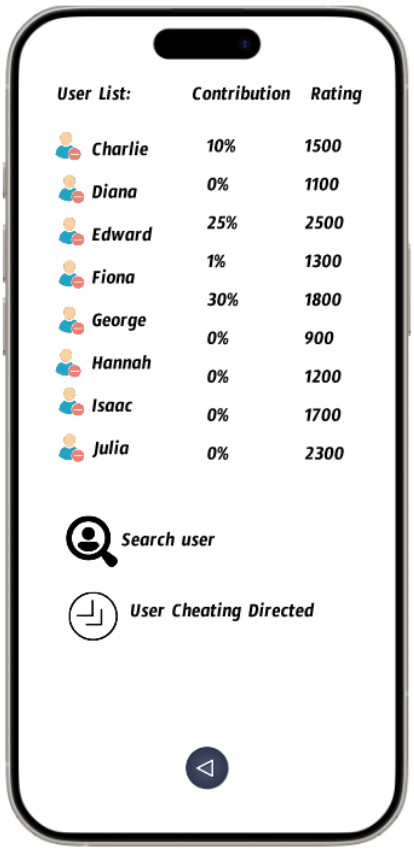


Add and Remove Problem

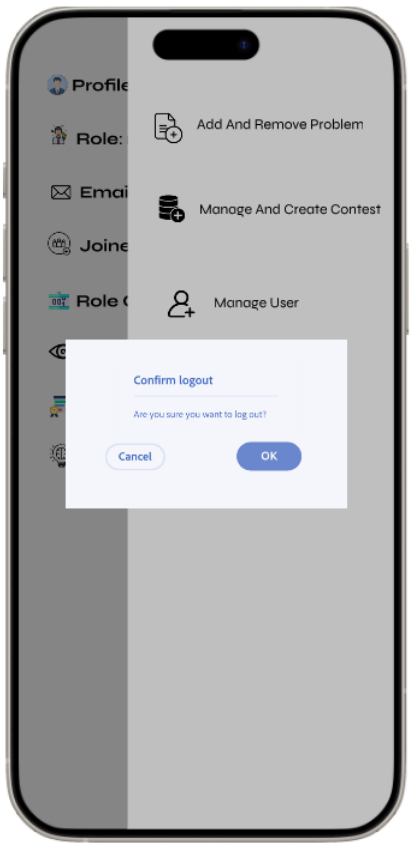
Manage and Create Contest



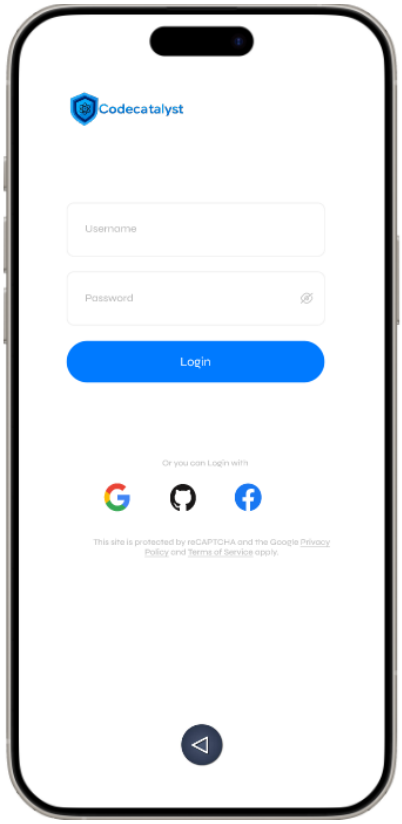
Manage User



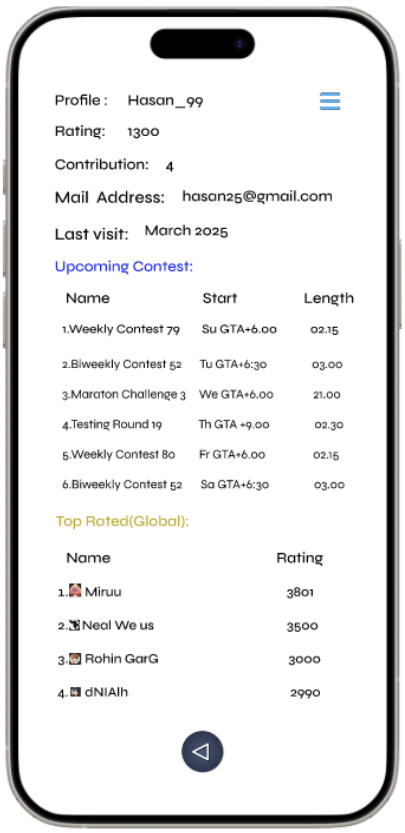
Admin Logout



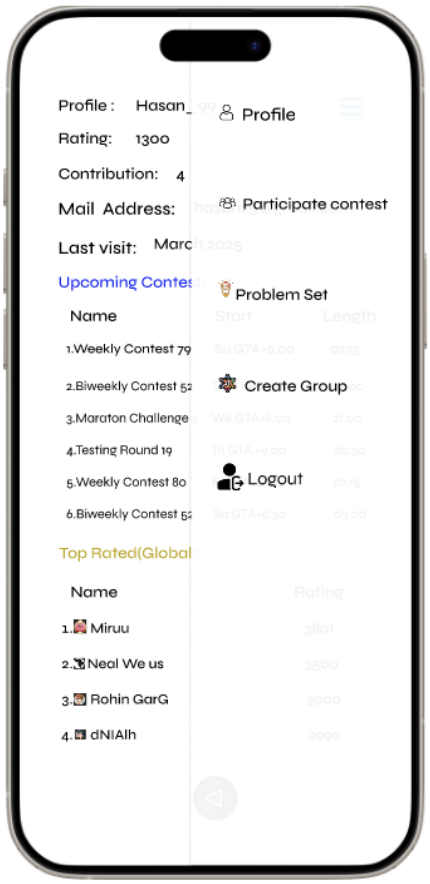
User Login



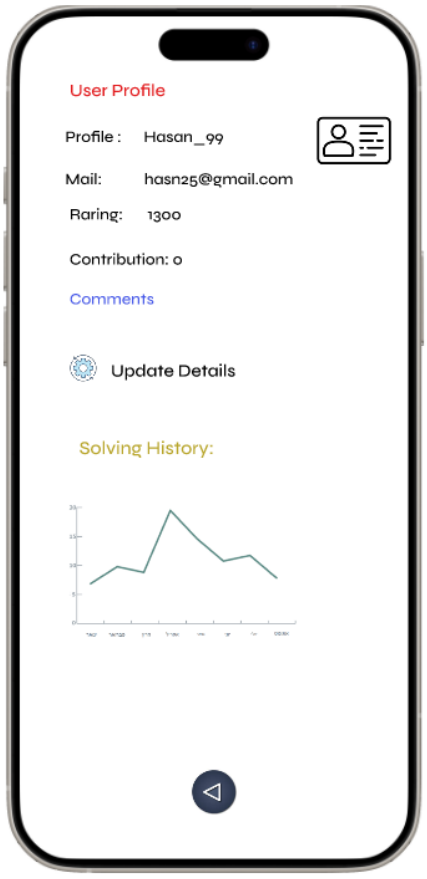
User Login Profile



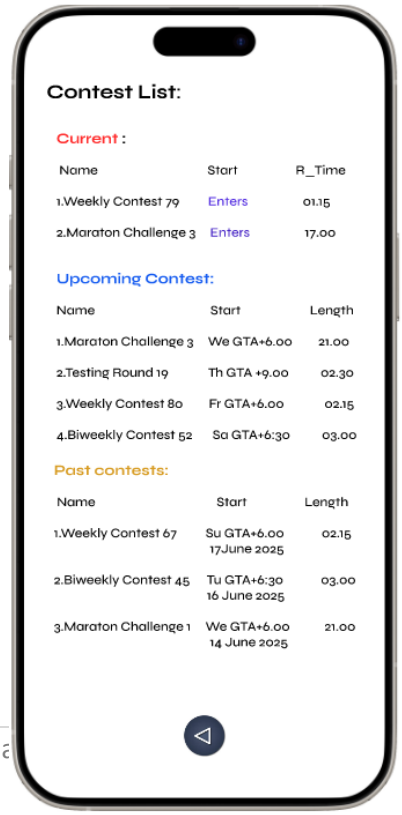
User Navigation



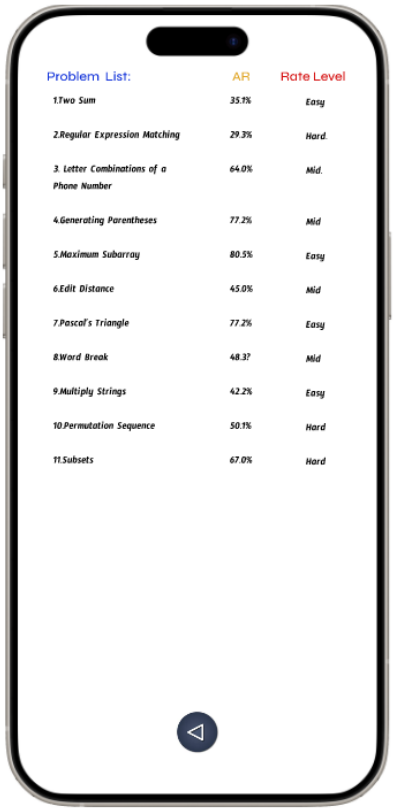
User Profile



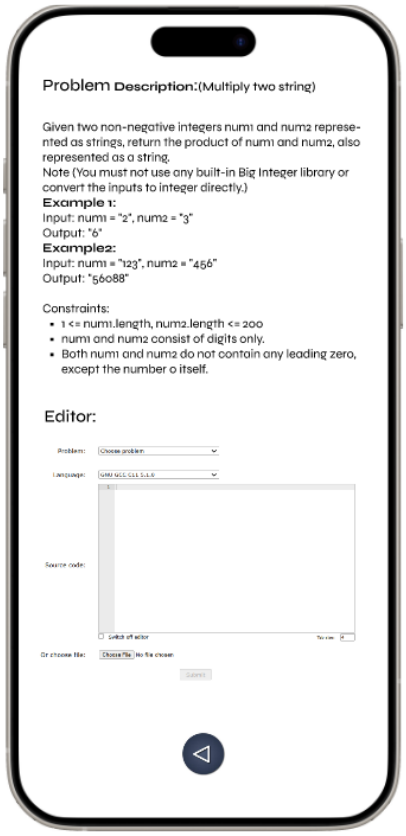
Participate Contest



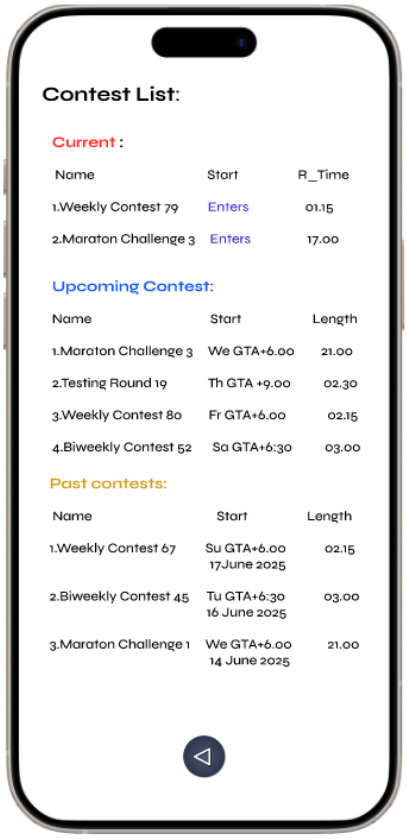
Problem Set



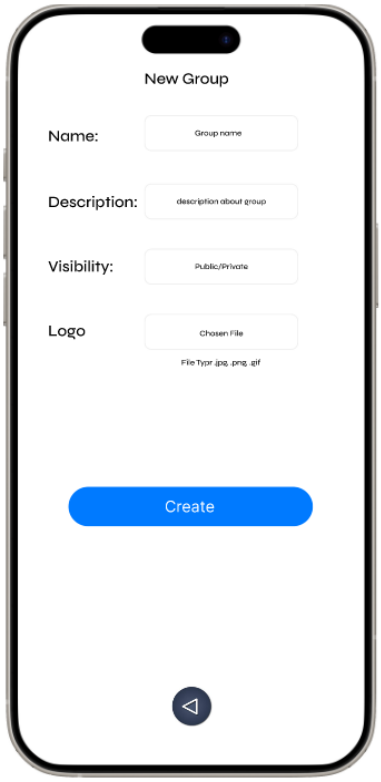
Problem Description



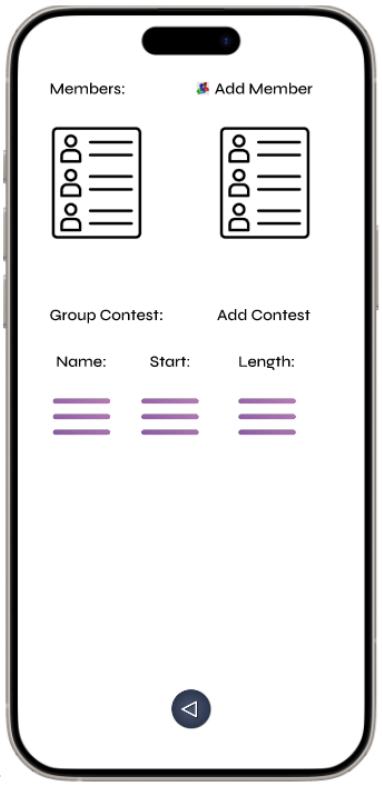
Contest List



Create Group



Group Page



Tools Used

For the creation and refinement of these prototypes, **Figma** was used due to its collaborative capabilities, component reuse features, and pixel-perfect frame design options. Figma allowed seamless iteration and real-time feedback incorporation from stakeholders, ensuring the design aligns closely with project goals.

9. Advanced Requirement Management

CodeCatalyst employs a structured Requirements Change Management process to ensure that all proposed changes to software requirements are properly documented, evaluated, approved, and communicated with minimal disruption to the project.

9.1 Change Handling Process

Identifying Change Requests

1. Changes may originate from clients, internal teams (QA, developers, PMs), or market demands.

2. Each request is formally logged using a Change Request Form via project tools like Jira .

Impact Analysis

Conducted by product managers and developers.

1.Existing functionalities

2.UI/UX

3.Codebase stability

4.Delivery timeline

5.Resource allocation

Change Evaluation

1. The feasibility, cost, and priority of the change are evaluated.

2. Evaluations consider business value, alignment with project goals, and current development stage.

Change Approval

1.A **Change Control Board** —comprising Product Managers, Tech Leads, QA Leads, and Stakeholder Reps—reviews each request. They vote to:

=>Approve

=>Reject

=>Postpone

Implementation & Tracking

Once a change request is approved, it is integrated into the sprint planning process to ensure structured and timely implementation. Progress is tracked using updated Jira tasks, and team velocity is visualized through sprint burndown charts, allowing the team to monitor completion status and identify potential bottlenecks early.

Documentation & Communication

All approved changes are thoroughly documented to maintain transparency and traceability. This includes updates to the Software Requirements Specification (SRS), sprint notes, and release changelogs. Stakeholders are kept informed through concise email summaries, interactive dashboards, and regular stakeholder meetings, ensuring alignment and shared understanding throughout the development lifecycle.

9.2 Risk Management

Risk management in CodeCatalyst is a proactive and ongoing process, with identification, evaluation, and mitigation strategies integrated throughout the SDLC (Software Development Life Cycle).

Identified Risks & Mitigation Strategies

Risk Description	Probability	Impact	Mitigation Strategy
Uncontrolled requirement changes during development	Medium	High	Strict , baseline requirement locking
Unauthorized access or data leak	Low	Critical	Data encryption, regular penetration testing
Slow response times under user load	Medium	High	Load testing, scalable architecture
Loss of key team members during development	Low	Medium	Knowledge base, thorough documentation, team backups

Failure to comply with data protection laws	Low	High	Legal reviews, automated compliance audits

10. Appendix

This section includes supporting materials that contributed to the requirement gathering and design process of the Codecatalyst project. These resources help validate stakeholder needs and inform the functional and non-functional requirements presented in this SRS.

10.1 References

- IEEE Std 830-1998 – Recommended Practice for Software Requirements Specifications.

10.2 Interview Notes

Several stakeholder interviews were conducted to understand expectations, features, and challenges with existing online judges.

Interview Summary – Admin :

- Wants the ability to schedule contests, manage problems, and access analytics.
- Concerned about cheating and auto-plagiarism detection.
- Interested in real-time monitoring and submission review.

Interview Summary – Competitive Programmer (Student):

- Emphasized a need for real-time feedback and low-latency judging.
- Asked for dark mode, keyboard shortcuts, and clean leaderboard UI.
- Highlighted lack of beginner guidance on many existing platforms.

Interview Summary – Problem Setter:

- Requested markdown/editor support for problem description formatting.
- Wants to test solutions against hidden datasets.
- Asked for visibility control (public/private test cases).

10.3 Survey Results

A digital survey was distributed among 60 students and 4 faculty members who participated in online contests previously.

Key Highlights:

Question	Most Common Response
Which online judge do you use?	Codeforces (68%), AtCoder (21%), others (11%)
Major frustration with current platforms?	Slow judging & UI complexity
Desired feature?	Real-time feedback, responsive design, customizable themes
Would you like contests from faculty?	Yes – 84%
Feature you'd love?	Group contests, code discussions, live editorial

Conclusion:

- Users highly value usability, speed, and contest diversity.
- Admins are concerned with plagiarism and platform reliability.
- Developers seek a well-documented and testable backend for judge logic.