Requirement Analysis and use cases on

**CodeCrakers: Competitive Programming Tracker Desktop Application**

Course Teacher: Prof. Dr. Kazi Masudul Alam  
Professor,   
Computer Science and Engineering Discipline,   
Khulna University, Khulna  
  
  
  
  
Developed By:

Pranto Mondol (230211)  
Toma Rani (230221)  
Sobuj Chandra Paul (230231)

Table of Contents

[**1. INTRODUCTION** 3](#_Toc206191785)

[**2. INTERVIEW QUESTION** 3](#_Toc206191786)

[**3. INTERVIEW SCRIPT** 4](#_Toc206191787)

[**4. OBSERVATION** 6](#_Toc206191788)

[**5. REQUIREMENTS** 6](#_Toc206191789)

[**5.1 Functional Requirements** 6](#_Toc206191790)

[**5.1.1 Menu Requirements** 6](#_Toc206191791)

[**5.1.2 Core Application Functions** 7](#_Toc206191792)

[**5.2 Structural Requirements** 7](#_Toc206191793)

[**5.2.1 Frontend** 7](#_Toc206191794)

[**5.2.2 Backend** 7](#_Toc206191795)

[**5.2.3 Database** 7](#_Toc206191796)

[**5.3 Non-Functional Requirements** 7](#_Toc206191797)

# **1. INTRODUCTION**

The purpose of CodeCrakers is to provide a desktop application for competitive programmers to track, monitor, and improve their problem-solving performance across multiple online platforms. Users can link their platform usernames, view statistics, track weekly progress, manage offline problems, and edit their profiles. The system will also allow visualization of problem-solving trends and provide goal-setting and reminders to enhance productivity.

# **2. INTERVIEW QUESTION**

1. Who will primarily use this application?

2. What is the main purpose of this application for you?

3. Should the application require registration and login?

4. After login, what initial setup should the user perform?

5. Should the application automatically create tracking sections for each platform?

6. What should be shown inside a platform box when clicked?

7. Should the app track progress for a specific date range?

8. What statistics are important for each platform?

9. Should the list of solved problems include problem names and URLs?

10. Should the app allow filtering solved problems by topic or difficulty?

11. Should the application fetch all data automatically from online platforms?

12. How should offline problems (custom problems) be added?

13. How should ratings for offline problems be assigned?

14. Should the app store and show weekly progress history?

15. Should the user be able to export stats?

16. Should the app send reminders to practice or join contests?

17. Should the app allow changing platform usernames later?

18. Should it work offline for stored data?

19. Should the UI be dark/light mode switchable?  
20. Should security be applied for user data?

# **3. INTERVIEW SCRIPT**

1. Who will primarily use this application?  
 => Competitive programmers, university students, and coding enthusiasts.

2. What is the main purpose of this application for you?  
=> To track competitive programming progress across multiple platforms automatically.

3.Should the application require registration and login?  
=> Yes, to secure user data and provide personalized tracking.

4. After login, what initial setup should the user perform?  
=> Add their usernames/handles for all platforms they use (Codeforces, LeetCode, AtCoder, CodeChef, etc.).

5. Should the application automatically create tracking sections for each platform?  
=> Yes, one dedicated “box” per platform showing that user’s stats.

6. What should be shown inside a platform box when clicked?  
=> Detailed statistics like problems solved, ratings, submissions, contests, etc.

7. Should the app track progress for a specific date range?  
=> Yes, show statistics for “Current Week” or “Current Phase” (e.g., 9 August 2025 – 15 August 2025).

8. What statistics are important for each platform?  
=>

* + Max Rating
  + Problems Solved Count
  + Submissions Count
  + Max Rated Problem
  + Average Problem Rating
  + Contests Participation Count
  + Virtual Contests Participation Count
  + List of Solved Problems

9. Should the list of solved problems include problem names and URLs?

=> Yes, so users can revisit them easily.

10. Should the app allow filtering solved problems by topic or difficulty?  
=> Yes, for better analysis.

11. Should the application fetch all data automatically from online platforms?  
=> Yes, via APIs.

12. How should offline problems (custom problems) be added?  
=> User should enter manually with fields for problem name, source, rating, topic, and date.

13. How should ratings for offline problems be assigned?  
=> Manually by the user with a suggested rating scale.

14. Should the app store and show weekly progress history?  
=> Yes, so users can compare their past performance.

15. Should the user be able to export stats?  
=> Yes, as PDF/CSV for sharing.

16. Should the app send reminders to practice or join contests?  
=> Yes, via desktop notifications.

17. Should the app allow changing platform usernames later?  
=> Yes, in case the user updates their handles.

18. Should it work offline for stored data?  
=> Yes, but online is required for fetching latest platform data.

19. Should the UI be dark/light mode switchable?  
=> Yes, for comfort.

20. Should security be applied for user data?  
=> Yes, encrypted local storage and secure authentication.

# **4. OBSERVATION**

1. The app must **require login and registration** for personalization.
2. After login, user must **enter usernames for all platforms**.
3. The system should **auto-generate a box for each platform** added.
4. Clicking a box should show **comprehensive stats** for that platform.
5. Statistics must be **date-range aware** (Current Week/Phase).
6. Must include **problem lists, ratings, and participation counts**.
7. **Offline problem handling** requires manual entry and rating assignment.
8. **Filtering, exporting, and UI customization** are important features.
9. The system must work **offline for stored data** but fetch updates online.
10. Security and privacy of data are essential.

# **5. REQUIREMENTS**

## **5.1 Functional Requirements**

### **5.1.1 Menu Requirements**

* Triple-line menu for profile editing and settings.
* Options to log out or change platform usernames.

### **5.1.2 Core Application Functions**

* Register/Login
* Add platform usernames
* Auto-generate platform in boxes
* View detailed statistics per platform
* Add offline problems
* Visualize weekly or phase statistics
* Export statistics (CSV/PDF)

## **5.2 Structural Requirements**

### **5.2.1 Frontend**

* Desktop UI framework with responsive design.
* Graphical charts for statistics.

### **5.2.2 Backend**

* Handles API integration and data fetching.
* Manages problem storage and user authentication.

### **5.2.3 Database**

* SQLite for offline storage.
* Stores problems, statistics, user data securely.

## **5.3 Non-Functional Requirements**

* Security: Encrypted data storage and secure authentication.
* Maintainability: Easy to add new platforms or features.
* Portability: Cross-platform desktop support.