Coding Contest Calendar (KONTEST)

A Project Report

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE

(INFORMATON TECHNOLOGY)

By

SHAIKH ARBAZ ALI SIKANDAR ALI
Roll Number: 560

Under the esteemed guidance of

Dr. Pallavi Tawde

Coordinator (DCIS)

NAGINDAS KHANDWALA COLLEGE

(Empowered Autonomous College)

(Affiliated to University of Mumbai)

MUMBAI - 400 064

MAHARASHTRA

Coding Contest Calendar (KONTEST)

A Project Report

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE

(INFORMATON TECHNOLOGY)

By

SHAIKH ARBAZ ALI SIKANDAR ALI
Roll Number: 560

Under the esteemed guidance of

Dr. Pallavi Tawde

Coordinator (DCIS)



NAGINDAS KHANDWALA COLLEGE

(Empowered Autonomous College)

(Affiliated to University of Mumbai)

MUMBAI - 400 064

MAHARASHTRA

2023 – 24

NAGINDAS KHANDWALA COLLEGE
(Empowered Autonomous College)
(Affiliated to University of Mumbai)
MUMBAI - 400 064
MAHARASHTRA

DEPARTMENT OF COMPUTER AND INFORMATION SCIENCE

CERTIFICATE

This is to certify that the project titled, **Coding Contest Calendar (KONTEST)**, is bonafied work of **Shaikh Arbaz Ali Sikandar Ali** bearing Roll No: 560 is submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

04/03/24
Internal Guide

04/03/24
Coordinator

15/3/24
External Examiner

15 MAR 2024
Date

College Seal

DECLARATION

I hereby declare that the project entitled, **Coding Contest Calendar** done at **Nagindas Khandwala College**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university. The project is done in partial fulfillment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.



SHAIKH ARBAZ ALI

ACKNOWLEDGEMNT

I wish to express my heartfelt gratitude and appreciation to all those who played a pivotal role in making my project, "Coding Contest Calendar," a reality. Their unwavering support and encouragement were instrumental in the successful completion of this project.

A special thanks goes to my parents for their continuous support and for providing me with the resources I needed.

I am very Thankful to **Dr. (Mrs.) Ancy Jose (Director) and Prof. Dr. Moushumi Datta (Principal) and Mona Mehta (Vice Principal)** of Nagindas Khandwala College for their kind co-operation in the completion of my project and **Dr. Pallavi Tawde**, Coordinator of the Department of Computer and Information Science, for her support and facilitation.

I take this opportunity to express my deep sense of gratitude towards my project guide **Dr. Pallavi Tawde** for the help and letting me work as I wished without putting pressure.

I would like to thank the entire staff of our department for their support, cooperation, and provision of necessary resources. My project experience was rewarding and fulfilling, and it wouldn't have been possible without the conducive learning environment provided by the college. Finally, I thank Nagindas Khandwala College of Commerce, Arts, and Science (Empowered Autonomous) for providing the right ambiance and amenities that contributed significantly to the timely and satisfactory completion of this project.

ABSTRACT

The "Coding Contest Calendar" project is a mobile application designed to simplify and enhance the experience of coding enthusiasts and programmers by providing a centralized platform for discovering and tracking upcoming coding contests and hackathons from various popular coding platforms. In the growing world of programming competitions, staying informed about upcoming contests across multiple platforms can be a tough task. This project addresses this challenge by offering a user-friendly and intuitive mobile app that aggregates contest details such as start and end times, registration links, and platform-specific information.

This project aims to simplify the contest participation experience for users and providing a platform for staying up-to-date with contests and ensuring they make the most of their coding journey.

Table of Contents

Topic	Page No.
Chapter 1: INTRODUCTION	1
1.1 Background of the Project	2
1.2 Scope of the Project	3
1.3 Objectives of the Project	4
1.4 Applicability	5
CHAPTER 2: GAP ANALYSIS/ LITERATURE REVIEW	6
CHAPTER 3: REQUIREMENTS AND ANALYSIS	17
3.1 Problem Definition	18
3.2 Requirements Specification	20
3.3 Planning and Scheduling	22
CHAPTER 4: SYSTEM DESIGN	24
4.1 Schema Design	25
4.2 UML Diagrams / Block Diagram/ Circuit Diagram/ Algorithms Design	27
4.3 User interface design	31
CHAPTER 5: IMPLEMENTATION AND TESTING	38
5.1 Code	39
5.2 Testing Approach and Test Cases	61
CHAPTER 6: DISCUSSIONAND CONCLUSION	66
6.1 Discussion	67
6.2 Conclusion	68
CHAPTER 7: LIMITATIONS	69
CHAPTER 8: FUTURE WORK	71
Abbreviations	74
References	75

LIST OF FIGURES

Figure No.	Figure Name	Page No.
1	Tracking Project Progress	23
2	Class Diagram	27
3	Activity Diagram	28
4	Use Case Diagram	29
5	Sequence Diagram	30
6	Splash Screen	31
7	Landing Window	31
8	Sign Up Screen	32
9	Sign In Screen	32
10	Contest Details	33
11	Contest Page	34
12	User Profile	35
13	Profile Update	36
14	API Call	37
15	JUNIT4 Testing	64
16	Local Unit Testing	65

LIST OF TABLES

Table No.	Table Name	Page No.
1	Summary Table	16
2	Project Planning Task	22
3	Date of Completion	22
4	Data Preparation	23
5	User (Schema)	25
6	Contest (Schema)	25
7	UserContest (Schema)	26
8	UserNotification (Schema)	26
9	Sign Up (Testing)	61
10	Sign In (Testing)	62

CHAPTER 1

INTRODUCTION

1.1 Background

The KONTEST App (Coding Contest Calendar) project aims to create a user-friendly Android application that provides coding enthusiasts with a centralized platform to access details about upcoming coding contests. The current landscape requires users to navigate multiple websites and platforms to find contest information, resulting in inefficiencies and missed opportunities.

The app's primary goal is to streamline the contest discovery process and enhance user experience by curating a comprehensive list of coding contests from renowned platforms like LeetCode, Codeforces, and HackerRank. Users can set reminders for contests they wish to participate in, ensuring timely notifications and increased engagement.

The inspiration behind the app arises from the growing interest in competitive programming and coding challenges among students, professionals, and coding hobbyists. The app aims to cater to users of all skill levels and backgrounds, fostering a sense of inclusivity and cross-platform engagement within the coding community.

By centralizing contest information, the Coding Contest Calendar app seeks to empower users to enhance their coding skills, achieve recognition in competitive programming, and build a vibrant community of coding enthusiasts united by their passion for challenges.

API Integration from kontest.net one of the pivotal elements of this project is the seamless integration with the API provided by kontest.net, accessible via the link **https://kontests.net/api/v1/all**. Alternative API that is from CodeForces will be used if kontest.net goes down unexpectedly **https://codeforces.com/api/contest.list**

In conclusion, the Coding Contest Calendar app addresses the need for a convenient and comprehensive platform for discovering and participating in coding contests. Through a user-friendly interface and a diverse range of contests, the app aims to facilitate continuous learning and growth within the coding community.

1.2 Scope

The scope of the KONTEST (Coding Contest Calendar) app is to create a user-friendly android application that serves as a one stop solution for coding enthusiasts seeking information about upcoming coding contests. The app will offer a curated and up to date list of contests from popular coding platforms like LeetCode, Codeforces, and HackerRank etc.

Users will have the convenience of exploring all upcoming and ongoing contests in a single place, eliminating the need to visit multiple websites. They can easily view essential contest details, such as contest names, start and end times, durations, and links to official contest pages.

App will allow users to set reminders for contests they wish to participate. They will receive timely notifications before the contests begin ensuring they never miss an opportunity to compete.

The primary focus of this application will be on providing a seamless user experience making the app accessible to coding enthusiasts of all skill levels. The app's intuitive design will enable effortless navigation and quick access to relevant contest information.

1.3 Objectives

- To Offer Centralized Contest Discovery: The project's goal is to create a centralized platform for coding enthusiasts to discover upcoming coding contests across various popular coding platforms.

- To Provide Contest Details: The application will furnish users with vital information such as contest start and end times, registration links, and more.

- To Enable Timely Contest Reminders: The project aims to empower users to set reminders for contests they intend to participate in, ensuring they don't miss important events.

- To Create a User-Friendly Interface: The primary aim is to design an app with a user-friendly interface that offers intuitive navigation and a seamless user experience.

- To Facilitate Specific Platform Selection: Users will have the ability to filter and view contests from specific platforms like LeetCode and others.

1.4 Applicability

- Educational Institutions: Professors, instructors, and students can use it as a supplementary tool for organizing in-house coding contests or preparing for external competitions.

- Individual Coders: Helpful for enthusiast coder to remain up to dated about different contests on different platforms.

- Skill Development: As a practical learning tool the app's applicability lies in helping users improve their coding skills, problem-solving abilities, and algorithmic thinking.

- Coding Communities: The app centralized platform fosters community engagement and collaboration among coding enthusiasts.

CHAPTER 2

GAP ANALYSIS/ LITERATURE REVEIW

Existing System Challenges:

- Scattered Contest Information: Prior to this application, users were required to navigate various coding platforms individually to find information about upcoming contests and hackathons, resulting in a time-consuming and cumbersome process.

- Difficulty in Staying Informed: Coding enthusiasts often struggled to stay informed about the multitude of contests across different platforms, leading to missed opportunities and a lack of awareness about coding events.

- Lack of Contest Notifications: The existing systems lacked a mechanism for users to receive timely notifications about upcoming contests, leaving them unaware of contest schedules and registration deadlines.

- Absence of Google Calendar Integration: Earlier systems lacked integration with popular calendar tools like Google Calendar, preventing users from easily adding contest reminders to their schedules and managing their time effectively.

- Prior to this application, users encountered challenges with the user interface in existing systems, which often lacked user-friendliness and intuitive design.

Proposed System Coding Contest Calendar (KONTEST) Solutions:

- Centralized Contest Discovery: The Coding Contest Calendar eliminates the need for users to visit multiple platforms by offering a central hub where they can access comprehensive information about upcoming and ongoing coding contests from various sources.

- Streamlined Contest Updates: This application simplifies the process of staying up-to-date with coding contests by aggregating event details from different platforms, ensuring that coding enthusiasts have easy access to a complete list of opportunities.

- Effortless Contest Notifications: With this proposed application, users can receive contest notifications, addressing the previous lack of a notification mechanism. They can set reminders for contests of interest, enhancing their participation in coding events.

- Seamless Google Calendar Integration: Unlike the earlier systems, this application seamlessly integrates with Google Calendar, enabling users to add contest reminders to their schedules with ease, improving time management and organization.

- The Coding Contest Calendar project prioritizes an enhanced and user-friendly interface, ensuring that users can effortlessly navigate and access contest information.

Literature Review:

1. Chougale et al. (2021), In the paper titled “Firebase-Overview and Usage” delves into Firebase, a Google-backed application development platform, examining its versatile features and impact on Android app development. It highlights Firebase diverse services, including analytics, authentication, cloud messaging, and real-time database, the review emphasizes its role in streamlining development processes. Practical use cases, such as optimizing onboarding flows and customizing user experiences, underscore Firebase's adaptability. While acknowledging Firebase's advantages, the review notes potential challenges like limited query functions and the need for additional back-end tools.

2. Khawas & Shah (2018), In the paper, titled "Exploring Firebase for Android App Development" shows the application of Firebase, a NoSQL technology, in Android app development, highlighting its advantages over traditional Relational Database Management Systems (RDBMS). It introduces various Firebase services, compares them with SQL and MS SQL Server, and provides a practical guide for developers. The literature review emphasizes differences in data storage, schema flexibility, and data models. The study concludes by emphasizing Firebase's benefits in Android app development, reducing reliance on third-party languages and ensuring secure communication with the database using Java.

3. Keny (2019), In the paper titled "A Review on Kotlin and Android Studio Java" compiles studies on Android app development using Java and Kotlin. Prof. Patel, Panchal, and R. K. conducted a comparative study on transitioning from Java to Kotlin. Holla and Katti focused on security issues in Android apps, proposing enhancements for increased safety. Android Authority's video highlighted reasons to adopt Kotlin for Android development. Bose, Mukherjee, Kundu, and Banerjee's comparative study emphasized Kotlin's ease of use, predicting widespread adoption. These studies collectively contribute to understanding Kotlin's features, security concerns in Android apps, and the ongoing Kotlin vs. Java debate in Android development.

4. Ofoeda et al. (2019), In the paper "Application Programming Interface (API) Research: A Review of the Past to Inform the Future" by Joshua Ofoeda, Richard Boateng, and John Effah conducts a systematic literature review analyzing 104 articles from 2010 to 2018. The focus includes themes, frameworks, methodologies, publication outlets, and levels of analysis in API research. Findings reveal a predominance of atheoretical research, emphasizing technological aspects over social issues. The review identifies a neglect of business and managerial applications of APIs. The authors suggest future research directions, advocating for a more holistic sociotechnical perspective in API studies. This synthesis contributes to understanding the current state of API research and guides future investigations in the field.

5. Sansakar & Kosarkar (2016), In the paper "Research on Development of Android Applications" by Mrs. Prachi Sasankar and Mrs. Usha Kosarkar provides a thorough exploration of the Android platform and its application development. It introduces the Android operating system, emphasizing its open-source nature and Linux kernel foundation. The Android application framework is discussed, outlining components such as views, content providers, a resource manager, a notification manager, and an activity manager. The paper delves into the layers of the Android system, categorizing components like the Dalvik virtual machine, the applications layer, application framework layer, libraries layer, Android Runtime, and the Linux kernel.

6. Sharma et. al (2022), In the research paper titled "Research on Development of Android Applications" explores the growing field of mobile application development, focusing on the Android platform. It discovers the challenges in the competitive mobile marketing landscape and highlights the need for a specific approach to ensure successful app development. The paper covers various factors influencing successful app development, providing specific examples and explanations. Additionally it shows Android features, architecture, activity lifecycle, and new application development features.

7. Putranto et. al (2020), In the paper titled "A Comparative Study of Java and Kotlin for Android Mobile Application Development," presented at the 2020 3rd International Seminar on Research of Information Technology and Intelligent Systems, conducts a comparative analysis of two popular programming languages, Java and Kotlin, for Android application development. The study aims to evaluate the strengths and weaknesses of each language in the context of building Android applications. The comparison involves various aspects, including source code evaluation, performance testing on different devices, and assessment of data usage.

8. Ardito et. al (2020), In the research paper titled "Effectiveness of Kotlin vs. Java in Android App Development Tasks" empirically assesses Kotlin and Java in the context of Android app development, comparing their development effectiveness, maintainability, and ease of development. The study finds no significant difference in maintainability between Kotlin and Java. However, Kotlin demonstrates superior code conciseness, leading to reduced lines of code, and exhibits a lower frequency of NullPointerExceptions.

9. Mohammad & Martin (2023), In the preprint titled “Leveraging ChatGPT in Android App Development” the study discusses the challenges of programming for beginners, introducing ChatGPT as a potential solution due to its language generation capabilities. The literature review covers ChatGPT's development, capabilities, and limitations, summarizing existing research on its applications, including code generation and problem solving. The study contributes to the evolving field of AI-assisted development and offers insights for novice users seeking to create applications using Java scripting and Android Studio.

10. Nishanthan (2020), In the paper explores the comparative analysis of Dalvik Virtual Machine and Android Runtime in the Android operating system, focusing on architectural differences and efficiency. The study compares the attributes of DVM and ART, including Just-In-Time and Ahead-Of-Time strategies. It evaluates their optimization and memory aspects using a prime number calculation algorithm, indicating potential dominance of ART in future Android versions. The literature review incorporates diverse sources, including Android documentation, security discussions, and benchmark research, providing a comprehensive overview of the subject.

11. Gupta et al. (2018): "Enhancing Mobile App Security through Behavior Analysis and Machine Learning" investigates the integration of behavior analysis and machine learning techniques to enhance the security of mobile applications on the Android platform. By analyzing user behavior patterns, the study aims to identify potential security threats and vulnerabilities, providing a proactive approach to securing sensitive user data. The research contributes valuable insights into the application of advanced technologies to fortify the security posture of Android mobile apps.

12. Kim et al. (2017): In the paper "Firebase: Impact on Android App Development," the authors delve into the impact of Firebase, a Google-backed platform, on Android app development. The study evaluates Firebase's versatile features, such as real-time database and cloud messaging, emphasizing its role in streamlining development processes. By highlighting practical use cases and discussing potential security concerns, the research guides developers in harnessing the full potential of Firebase while maintaining a vigilant approach to implementation.

13. Rahman et al. (2016): "Efficient Data Fetching in Android Apps" focuses on optimizing data fetching processes in Android applications. The study proposes efficient strategies and algorithms to enhance the speed and responsiveness of data retrieval, ultimately improving the overall performance of Android apps. By addressing the challenges associated with data fetching, the research contributes practical solutions to developers aiming to create more efficient and user-friendly mobile applications.

14. Arun et al. (2019): In "User Interface Design Trends in Android Development," the authors explore contemporary trends in designing rich user interfaces for Android applications. The study identifies and analyzes the evolving design patterns and user interface elements that contribute to a more engaging and user-friendly experience. By providing insights into the latest design trends, the research assists developers in creating visually appealing and intuitive interfaces, enhancing the overall user experience of Android apps.

15. Jain et al. (2015): The paper titled "Kotlin for Android Development: A Comprehensive Study" provides an in-depth analysis of Kotlin's adoption and impact on Android app development. The study evaluates the language's features, performance, and integration capabilities, offering insights into its suitability as an alternative to Java. By comparing the strengths and weaknesses of Kotlin in the context of Android app development, the research guides developers and decision-makers in making informed choices for their projects.

16. Li et al. (2018): "API Integration Challenges in Android Development" investigates the challenges associated with integrating external APIs into Android applications. The study explores common issues such as compatibility, security concerns, and performance bottlenecks, providing recommendations for overcoming these challenges. The research sheds light on the intricate aspects of API integration, emphasizing the importance of addressing compatibility issues and ensuring robust security measures to enhance the overall performance of Android applications.

17. Chen et al. (2017): In the paper "Firebase vs. AWS: A Comparative Analysis for Mobile App Backend Services," the authors conduct a comparative analysis of Firebase and Amazon Web Services (AWS) as backend services for mobile applications. The study assesses factors such as scalability, ease of use, and cost-effectiveness to guide developers in choosing the most suitable platform. The research aids developers in making informed decisions regarding backend services, considering key aspects like scalability and cost-effectiveness.

18. Wang et al. (2019): "Optimizing Network Communication in Kotlin-Based Android Apps" focuses on optimizing network communication processes in Android applications developed using Kotlin. The study proposes strategies to minimize latency, enhance data transfer efficiency, and improve the overall network performance of Kotlin-based apps. The research contributes valuable insights into enhancing Kotlin-based app performance by addressing network communication challenges.

19. Xu et al. (2016): The study "Augmented Reality in Android App Development" explores the integration and impact of augmented reality (AR) features in Android applications. The authors assess the usability and user experience enhancements brought about by AR, providing insights into its potential applications and challenges. The research highlights the evolving landscape of Android app development with the integration of augmented reality, offering guidance on leveraging AR for improved user experiences.

20. Singh et al. (2017): In "Cross-Platform Mobile Development: A Comparative Review," the authors compare different cross-platform development frameworks for Android applications. The study evaluates frameworks such as React Native and Xamarin, considering factors like performance, code reusability, and development efficiency. The research aids developers in selecting the most suitable cross-platform framework for Android app development, emphasizing the importance of performance and code reusability.

Summary Table:

Year	Authors	Title	Description
2015	Jain et al.	"Kotlin for Android Development: A Comprehensive Study"	Comprehensive study evaluating Kotlin's adoption and impact on Android app development, providing insights into its features, performance, and integration capabilities.
2016	Rahman et al.	"Efficient Data Fetching in Android Apps"	Focus on optimizing data fetching processes in Android applications, proposing efficient strategies and algorithms to enhance speed and responsiveness of data retrieval, ultimately improving overall performance of Android apps.
2016	Sansakar & Kosarkar	"Research on Development of Android Applications"	Thorough exploration of Android platform and application development, covering operating system, application framework, and layers of the Android system.
2017	Chen et al.	"FB vs. AWS: A Comparative Analysis for Mobile ABS	Comparative analysis of Firebase and AWS as backend services, assessing factors like scalability and cost-effectiveness to guide developers' decisions.
2017	Kim et al.	"Firebase: Impact on Android App Development"	Delve into the impact of Firebase on Android app development, evaluating its versatile features like real-time database and cloud messaging.
2018	Gupta et al.	"Enhancing Mobile App Security through Behavior Analysis and ML"	Investigation into integrating behavior analysis and machine learning techniques to enhance security of mobile applications on the Android platform.
2018	Khawas & Shah	"Exploring Firebase for Android App Development"	Application of Firebase in Android app development, highlighting its advantages over traditional RDBMS and providing practical guidance for developers.
2018	Li et al.	"API Integration Challenges in Android Development"	Investigation of challenges associated with integrating external APIs into Android apps, providing recommendations for overcoming compatibility and security issues.
2019	Arun et al.	"UI Design Trends in Android Development"	Exploration of contemporary trends in designing rich user interfaces for Android applications, identifying evolving design patterns and elements.
2019	Ofoeda et al.	" API Research: A Review of the Past to Inform the Future"	Synthesis of API research, analyzing 104 articles, revealing predominance of atheoretical research and suggesting future research directions.

2019	Wang et al.	"Optimizing Network Communication in Kotlin-Based APPs"	Focus on optimizing network communication processes in Android applications developed using Kotlin.
2020	Ardito et al.	"Effectiveness of Kotlin vs. Java in Android App Development Tasks"	Empirical assessment comparing Kotlin and Java in Android app development, finding no significant difference in maintainability, with Kotlin demonstrating superior code conciseness.
2020	Singh et al.	"Cross-Platform Mobile Development: A Comparative Review"	Comparison of different cross-platform development frameworks for Android applications, evaluating frameworks like React Native and Xamarin.
2021	Chougale et al.	"Firebase-Overview and Usage"	Exploration of Firebase, examining its features and impact on Android app development.
2022	Sharma et al.	"Research on Development of Android Applications"	Exploration of mobile application development on the Android platform, focusing on challenges.
2022	Putranto et al.	"A Comparative Study of Java and Kotlin "	Comparative analysis of Java and Kotlin for Android app development testing, and data usage assessment.
2023	Mohammad & Martin	"Leveraging ChatGPT in Android App Development"	Discussion on using ChatGPT to address programming challenges for beginners in Android app development, emphasizing its language generation capabilities and potential applications.
2023	Nishanthan	"Comparative Analysis of DVM and Android Runtime in the AOS "	Comparative analysis of DVM and ART, focusing on architectural differences and efficiency, evaluating attributes like JIT and AOT strategies, and indicating potential dominance of ART in future.
2023	Sansakar & Kosarkar	"Leveraging ChatGPT in Android App Development"	Discussion on leveraging ChatGPT for Android app development, addressing challenges faced by beginners.

Table 1: Summary Table of Literature Reviews

CHAPTER 3

REQUIREMENT AND ANALYSIS

3.1 Problem Definition

In the world of competitive programming and coding challenges, coding enthusiasts, students, and professionals alike frequently participate in various coding contests and hackathons. These events are hosted on a multitude of online platforms, such as Codeforces, LeetCode, HackerRank, and many others. The diverse nature of these platforms, while offering a wide range of opportunities, also poses several challenges for participants.

Before the introduction of the Coding Contest Calendar project, individuals interested in coding contests and hackathons faced a significant challenge. Contest information was fragmented across various platforms, each with its unique schedule and registration process. This fragmentation made it cumbersome for participants to find and track contests efficiently. The sheer volume of contests hosted across different platforms made it easy for users to miss out on valuable opportunities to test their skills and compete.

Existing systems provided no effective mechanism for users to receive timely notifications about upcoming contests. This lack of a notification system meant that potential participants were often unaware of contest schedules, registration deadlines, and other critical information.

Another significant issue was the absence of integration with widely used calendar tools like Google Calendar. This prevented users from easily adding contest reminders to their schedules and managing their time effectively, leading to missed contests and scheduling conflicts.

Significance of the Problem:

The challenges outlined above were not only frustrating for coding enthusiasts but also hindered their ability to engage actively in coding contests and hackathons. These contests serve as valuable opportunities for skill development, networking, and career advancement. Therefore, addressing these challenges is of utmost importance as it:

- Enhances access to coding contests, ensuring that enthusiasts can explore a comprehensive list of opportunities.
- Empowers users to stay informed about the latest contests, promoting continuous learning and skill development.
- Provides a mechanism for users to receive contest notifications, reducing the likelihood of missed events.
- Integrates with popular calendar tools, enabling effective time management and organization for participants.

Solution Approach:

The Coding Contest Calendar project aims to provide a holistic solution to these challenges. By centralizing contest information, offering personalized features, implementing a notification system, and integrating with Google Calendar, the project seeks to create a user-friendly and efficient platform that empowers coding enthusiasts to fully engage in coding contests and hackathons. This solution not only addresses the existing problems but also enhances the overall experience of coding enthusiasts in the competitive programming community.

3.2 Requirements Specification

- User Authentication and Profiles:

  User Registration: Users must be able to register for an account using their email or social media accounts.

  User Profiles: Each user should have a profile where they can personalize their experience and manage contest preferences.

  User Authentication: Robust user authentication mechanisms are required to ensure data security and user privacy.

- Contest Data Integration:

  Multiple Platforms: The system must integrate with various coding platforms' APIs to collect and update contest information. Contest data must be updated in real time using the API to provide users with the most current information.

  Comprehensive Listings: The system should compile a comprehensive list of coding contests, including details like contest name, platform, start and end times, registration links, and descriptions.

- User-Friendly Interface:

  Intuitive Design: The user interface must be intuitive, user-friendly, and visually appealing to cater to users of varying technical backgrounds.

  Responsive Design: The application should be responsive and compatible with various devices and screen sizes for a consistent user experience.

- Personalization and Notifications:

  Contest Tracking: Users should be able to track contests they are interested in and receive personalized contest recommendations.

  Notification System: The system should provide users with the option to receive notifications about upcoming contests, contest start times, and registration deadlines.

  Google Calendar Integration: Users must be able to seamlessly integrate contest reminders into their Google Calendar for efficient scheduling.

- Performance and Scalability:

  Optimized Performance: The application should perform efficiently, with fast load times and responsive features.

  Scalability: The system must be designed to handle a growing user base and increasing contest data without performance degradation.

- **API** Integration from kontest.net:

  One of the pivotal elements of this project is the seamless integration with the API provided by kontest.net, accessible via the link **https://kontests.net/api/v1/all**. This API provides with comprehensive and up-to-date information about coding contests and hackathons. By using this API the project ensures that users have access to a centralized hub of contest details, including contest names, platforms, start and end times, registration links, and descriptions.

3.3 Planning and Scheduling

In the context of the Coding Contest Calendar project planning is essential to ensure that all project phases are well-defined, resources are allocated efficiently and tasks are executed in a coordinated manner. One invaluable tool for visualizing this plan is the Gantt chart. In our project the Gantt chart serves as a dynamic roadmap providing a graphical representation of tasks, durations, and deadlines.

PLANNED DATES			
TASK	PLAN START DATE	WORKING DAYS	PLAN END DATE
REQUIREMENT GATHERING	15-06-2023	30	28-07-2023
FEASBILITY ANALYSIS	28-07-2023	25	04-09-2023
RISK ANALYSIS	04-09-2023	13	22-09-2023
DESIGNING	22-09-2023	10	10-10-2023
CODING	10-10-2023	26	16-11-2023
TESTING	16-11-2023	70	28-02-2024

Table 2: Project Planning Task

ACTUAL DATE				
ACTUAL START DATE	ACTUAL DAYS	ACTUAL END DATE	% Completion	Holidays
15-06-2023	35	04-08-2023	100%	29-06-2023
04-08-2023	25	12-09-2023	100%	29-07-2023
12-09-2023	49	23-11-2023	100%	15-08-2023
23-11-2023	30	08-01-2024	100%	19-08-2023
08-01-2024	25	13-02-2024	100%	07-09-2023
13-02-2024	10	28-02-2024	100%	28-09-2023
				02-10-2023
				24-10-2023
				12-11-2023
				27-11-2023
				25-12-2023
				26-01-2024
				19-02-2024

Table 3: Date of Completion

DATA PREPARATION			
TASK	ACTUAL START DATE	NO. OF DAYS	No. of Days completed
REQUIREMENT GATHERING	15-06-2023	19	19
FEASBILITY ANALYSIS	04-07-2023	31	31
RISK ANALYSIS	04-08-2023	22	22
DESIGNING	26-08-2023	30	30
CODING	25-09-2023	119	119
TESTING	22-01-2024	34	34

Table 4: Data Preparation

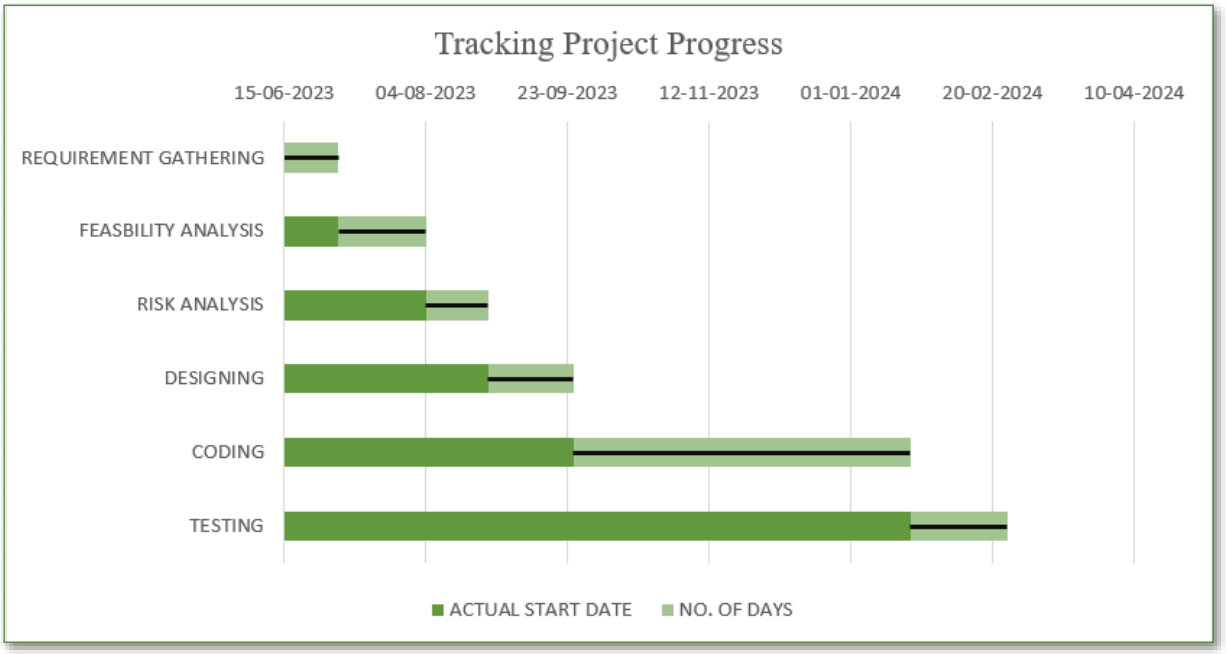


Figure 1: Tracking Project Progress

CHAPTER 4
SYSTEM DESIGN

4.1 Schema Design

User Table:

Field Name	Data type	Constraint	Description
UserID	VARCHAR(255)	Primary Key	Unique identifier for each user
Name	VARCHAR(255)	Not null	User's username
Email	VARCHAR(255)	Not Null	User's email address
Password	VARCHAR(255)	Not Null	Hashed user password
Registration Date	DATETIME	Not Null	Date and time when the user registered

Table 5: User Table

Contest Table:

Field Name	Data type	Constraint	Description
ContestID	VARCHAR(255)	Primary Key	Unique identifier for each contest
Contest Name	VARCHAR(255)	Not null	Name of the coding contest
Contest Platform	VARCHAR(255)	Not null	Platform where the contest is hosted
Start Time	DATETIME	Not null	Start date and time of the contest
End Time	DATETIME	Not null	End date and time of the contest
Registration Link	VARCHAR(255)	---	Link for contest registration
Description	TEXT	---	Description of the contest
Created By	INT	Foreign Key	User ID of the contest creator (linked to User.UserID)

Table 6: Contest Table

UserContest Table (Many-to-Many Relationship):

Field Name	Data Type	Constraint	Description
UserContestID	VARCHAR	Primary Key	Unique identifier for each user-contest entry
UserID	INT	Foreign Key	User ID (linked to User.UserID)
ContestID	INT	Foreign Key	Contest ID (linked to Contest.ContestID)
Status	VARCHAR(50)	---	User's status for the contest (e.g., Participating, Completed)

Table 7: UserContest Table

UserNotification Table:

Field Name	Data Type	Constraint	Description
NotificationID	INT	Primary Key	Unique identifier for each notification
UserID	INT	Foreign Key	User ID (linked to User.UserID)
ContestID	INT	Foreign Key	Contest ID (linked to Contest.ContestID)
Notification Type	VARCHAR(50)	---	Type of notification (e.g., Contest Start, Registration Deadline)
Notification Date/Time	DATETIME	---	Date and time when the notification is sent

Table 8: UserNotification Table

4.2 UML Diagrams / Block Diagram/ Circuit Diagram/ Algorithms Design

1. Class Diagram

The class diagram provides a structural overview of Coding Contest Calendar application (KONTEST) key components their attributes, and methods. It visualizes how classes interact and form the foundation of project's design. In Coding Contest Calendar project the class diagram reflects the core entities and their interactions. Users can login and manage their profiles. This class diagram serves as a blueprint for the application's architecture and guiding in implementing the necessary classes, attributes, and methods. It helps ensure that the application is well-structured and facilitating user interactions with coding contests, notifications, and user profiles.

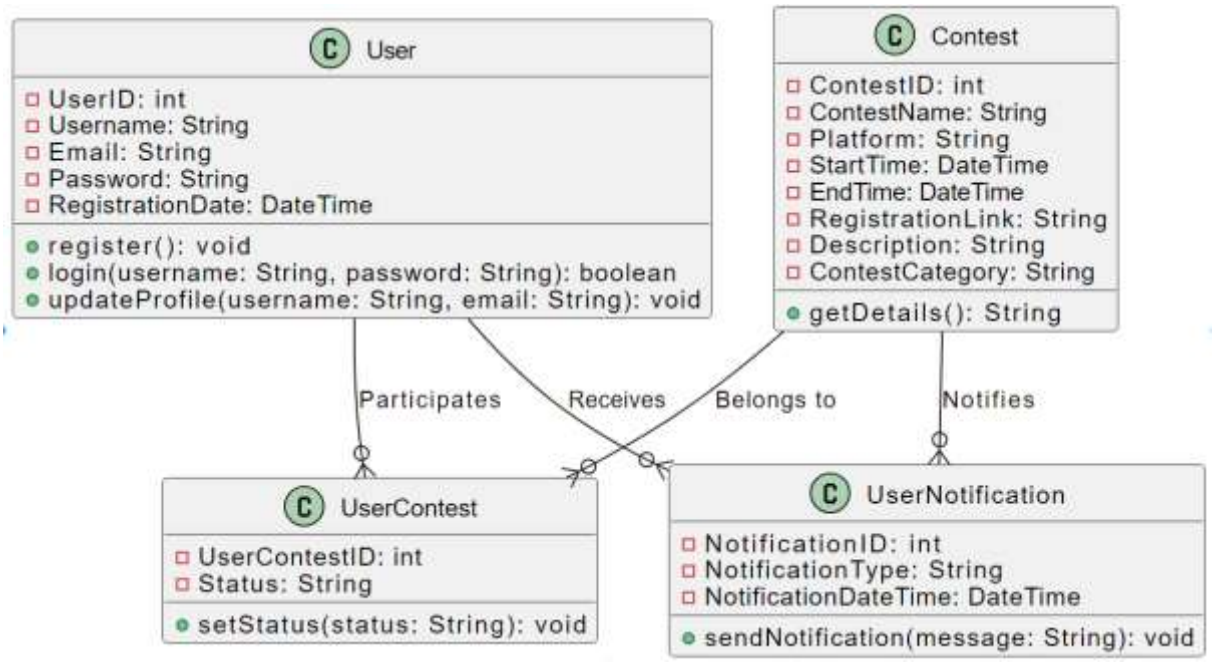


Figure 2: Class Diagram

2. Activity Diagram

An activity diagram is a visual representation of a system workflow. This diagram illustrates the sequence of activities, actions and decision points. In this project the activity diagram provides a clear depiction of how users interact with the application. It outlines the steps involved in user registration, data validation, and various functionalities available to registered and unregistered users.

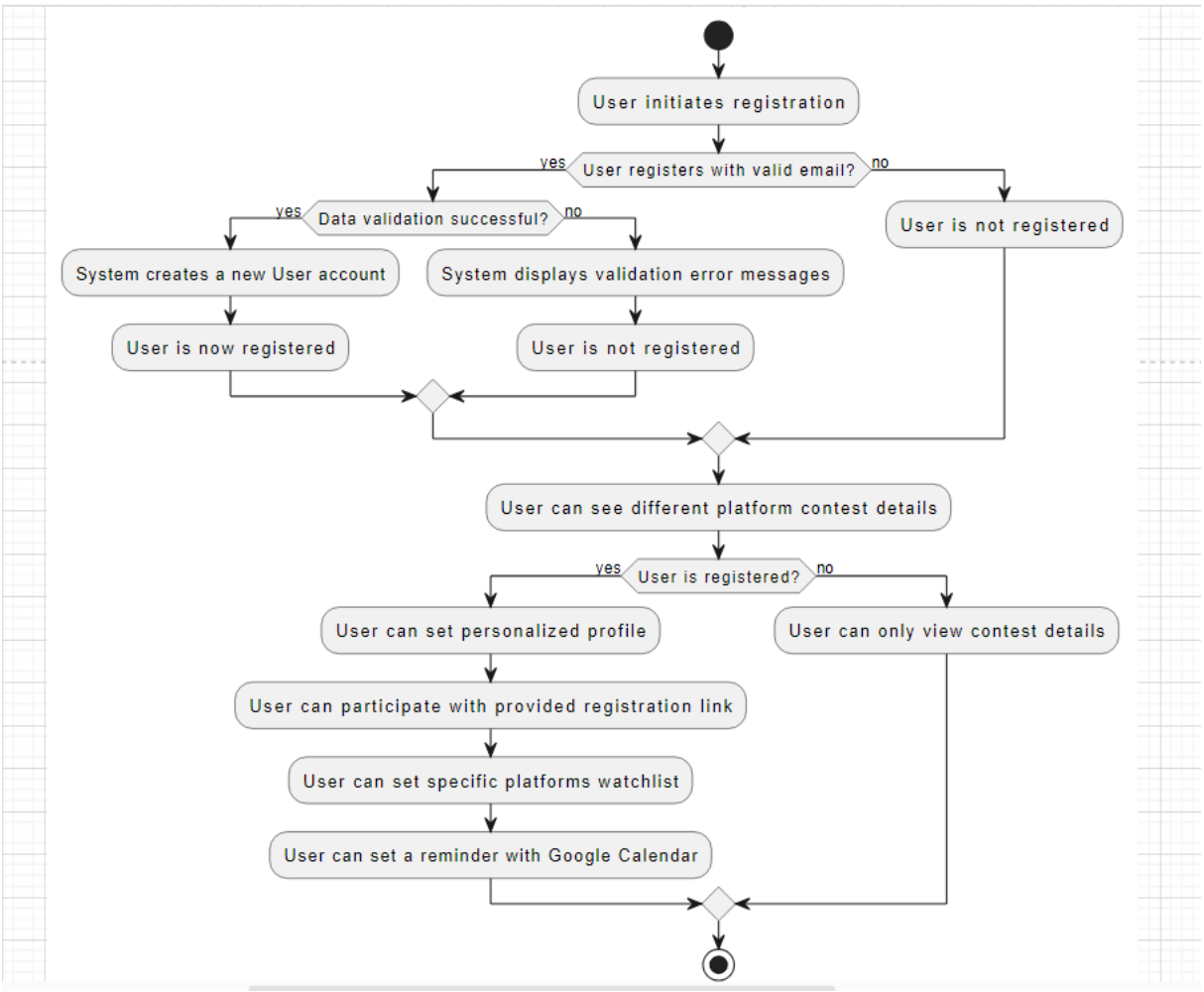


Figure 3: Activity Diagram

3. Use Case Diagram.

A use case diagram is a graphical representation that outlines the interactions between various actors and a system, illustrating how users or external entities interact with the system to achieve specific goals/perform certain tasks. In the Coding Contest Calendar, the use case diagram visually presents the main actions that users can perform within the application, such as registration, viewing contest details, setting personalized profiles, participating in contests, managing a watchlist, and setting reminders. It provides a high-level understanding of the systems functionality and user interactions, serving as a valuable tool for project planning.

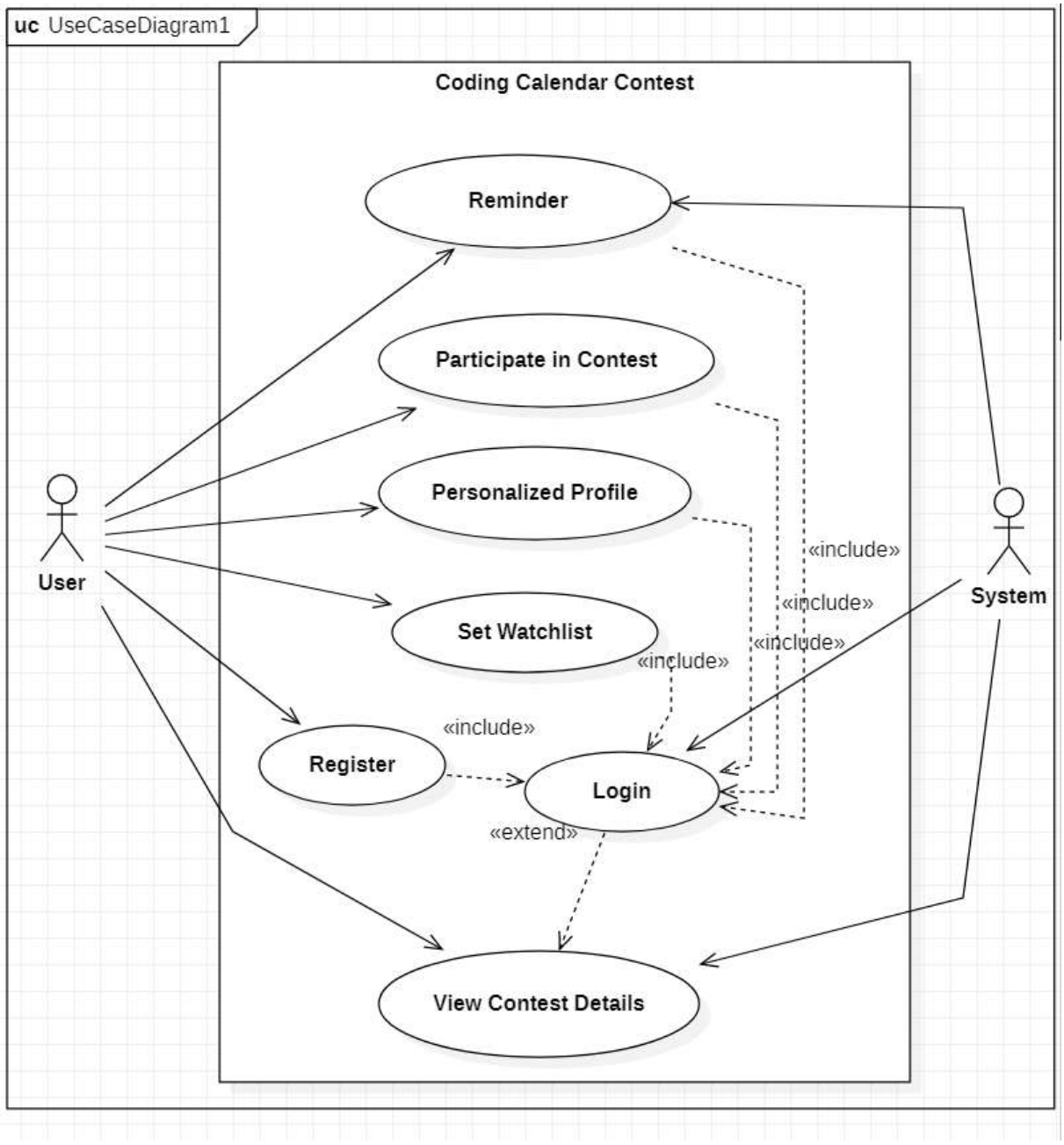


Figure 4: Use Case Diagram

4. Sequence Diagram.

The sequence diagram provides a visual representation of the interactions between users and the Coding Contest Calendar System. It showcases the seamless flow of actions, starting with user login, followed by the ability to view contest details, select contests for participation, generate registration links, and set reminders using Google Calendar. The diagram highlights the orchestration of events, depicting how users navigate through the system features to make the most of the Coding Contest Calendar application. It serves as a valuable tool for understanding the dynamic user-system interactions within the project, ensuring a user-friendly and efficient experience for coding enthusiasts.

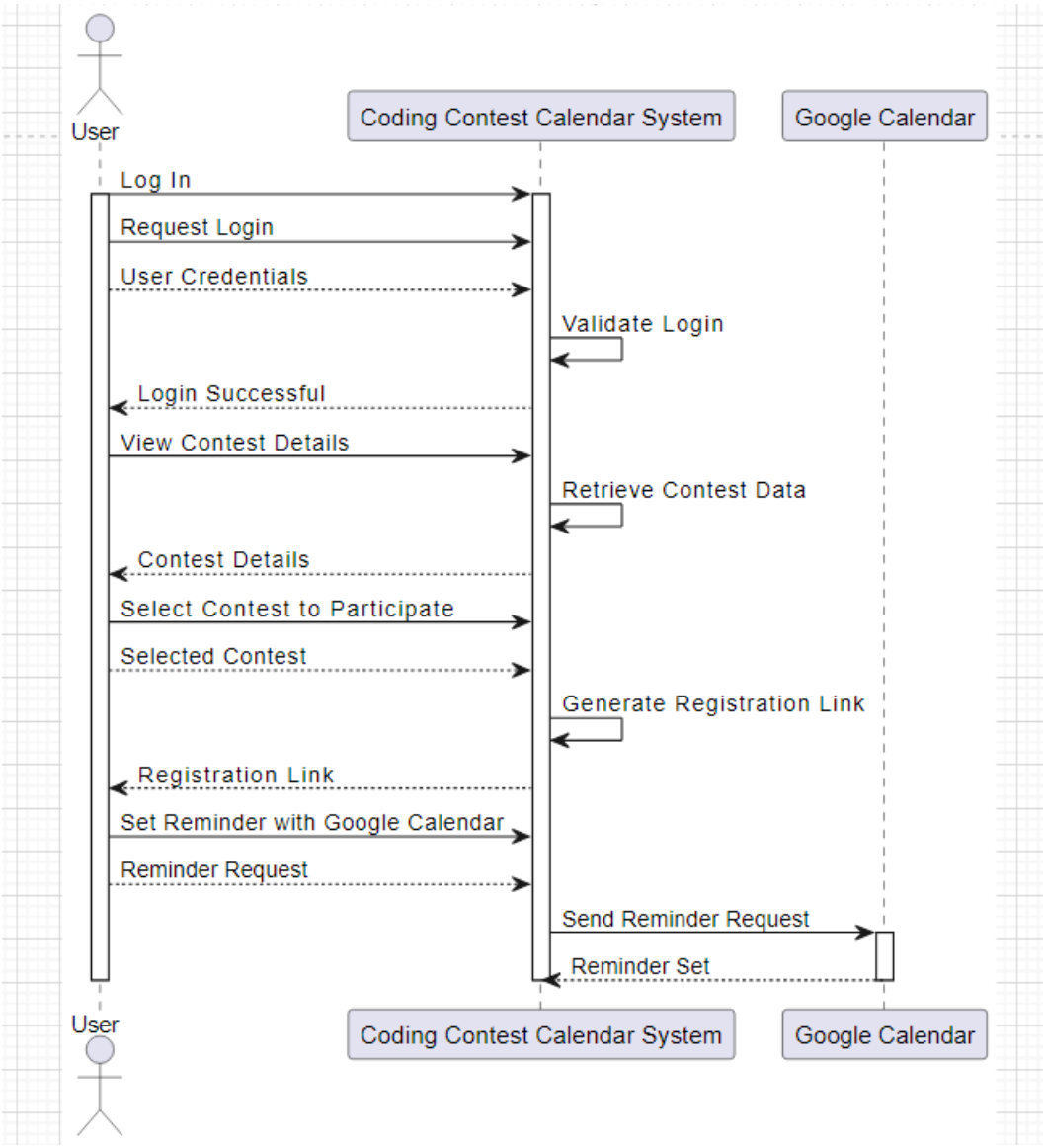


Figure 5: Sequence Diagram

4.3 User Interface Design

1. Splash Screen than rediredted to Landing Page



Figure 6: Splash Screen

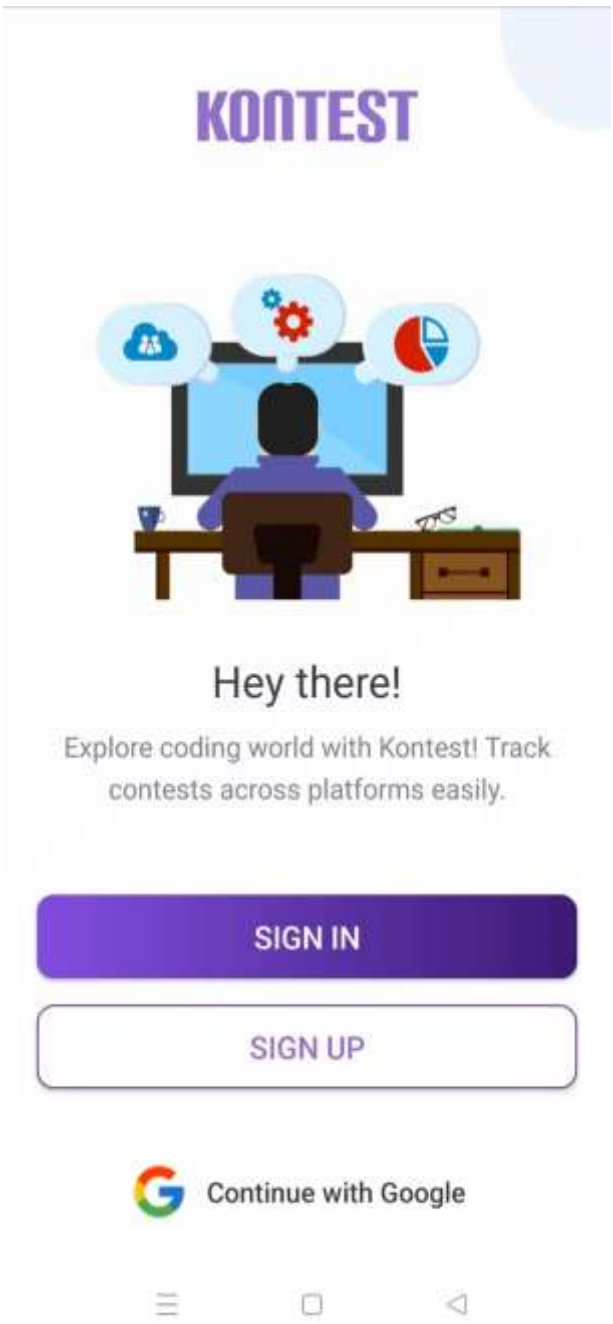


Figure 7: Landing Window

2. Sign up & Sign

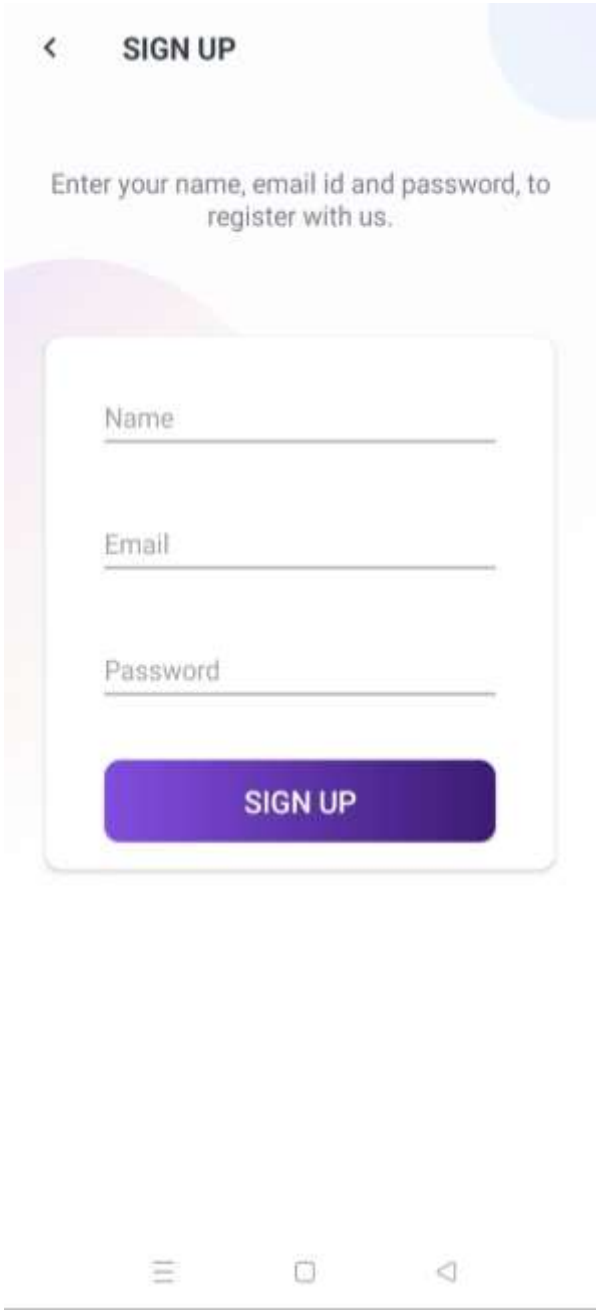


Figure 8: Sign Up Screen

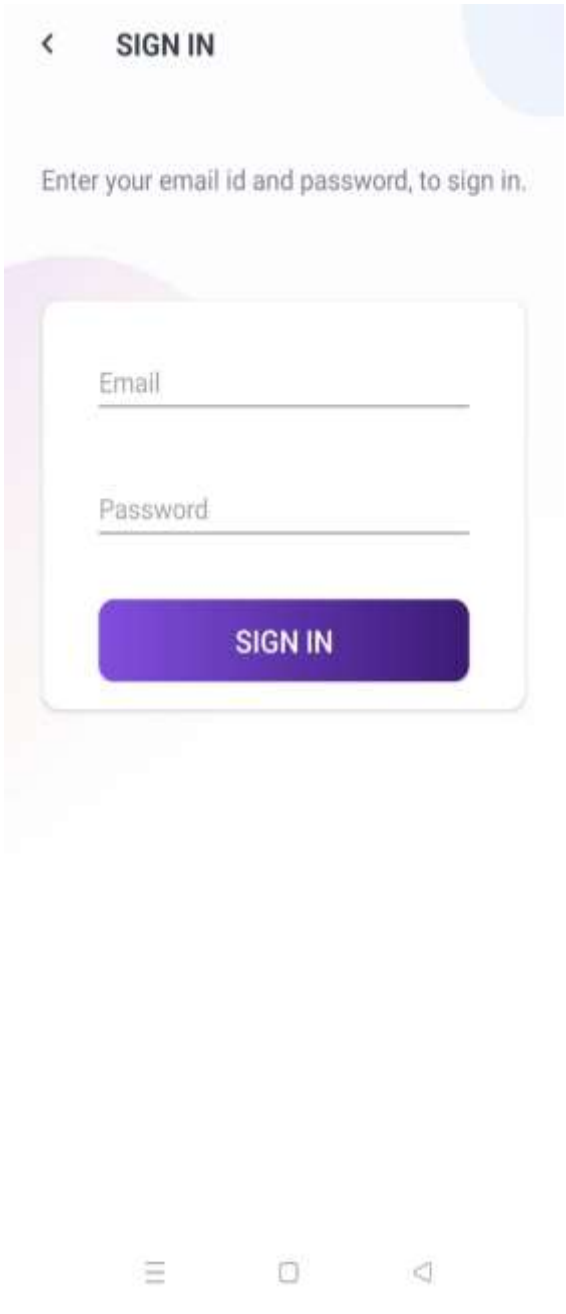


Figure 9: Sign In Screen

3. Home page, accessible after a successful login with valid credentials, dynamically fetches contest details from the API and presents them to the user.



Figure 10: Contest Details

4. The user is redirected to the contest page using the application's built-in browser.

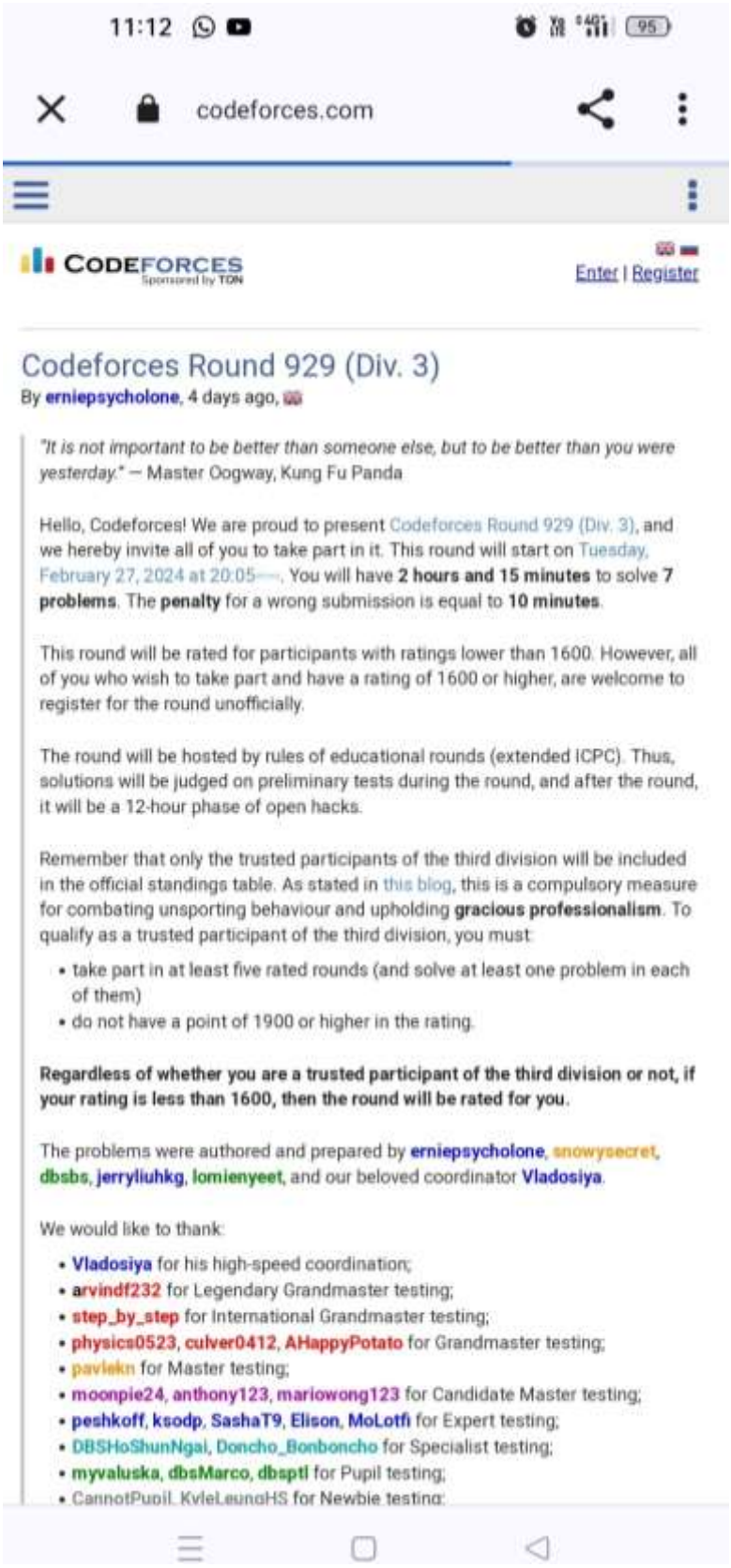


Figure 11: Contest Page (CodeForces)

5. User profile and data are stored in the Firebase database, retrieved in this section, and can also be updated with new information.

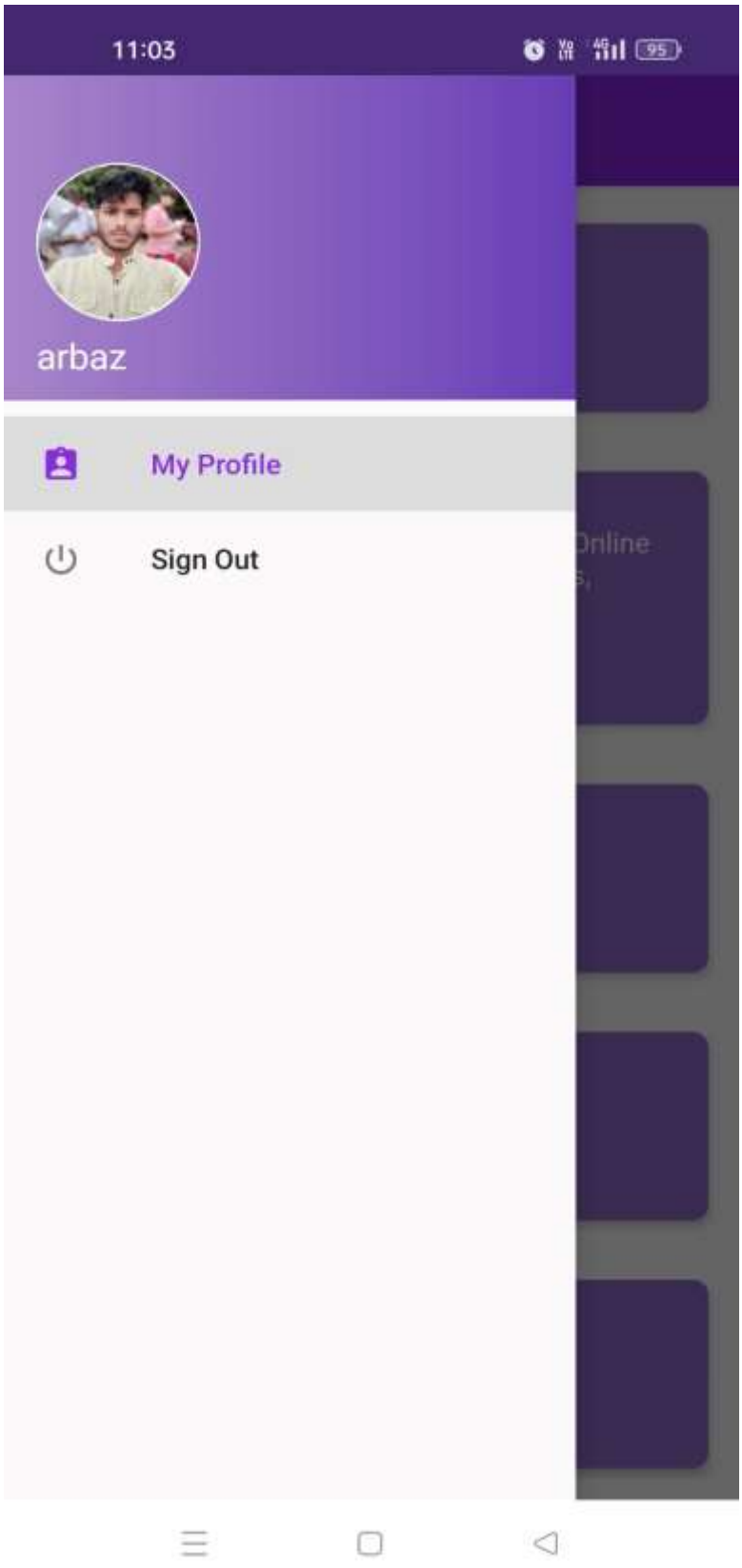


Figure 12: User Profile

6. The user profile here can be updated, and it gets synchronized with the real-time Firebase database

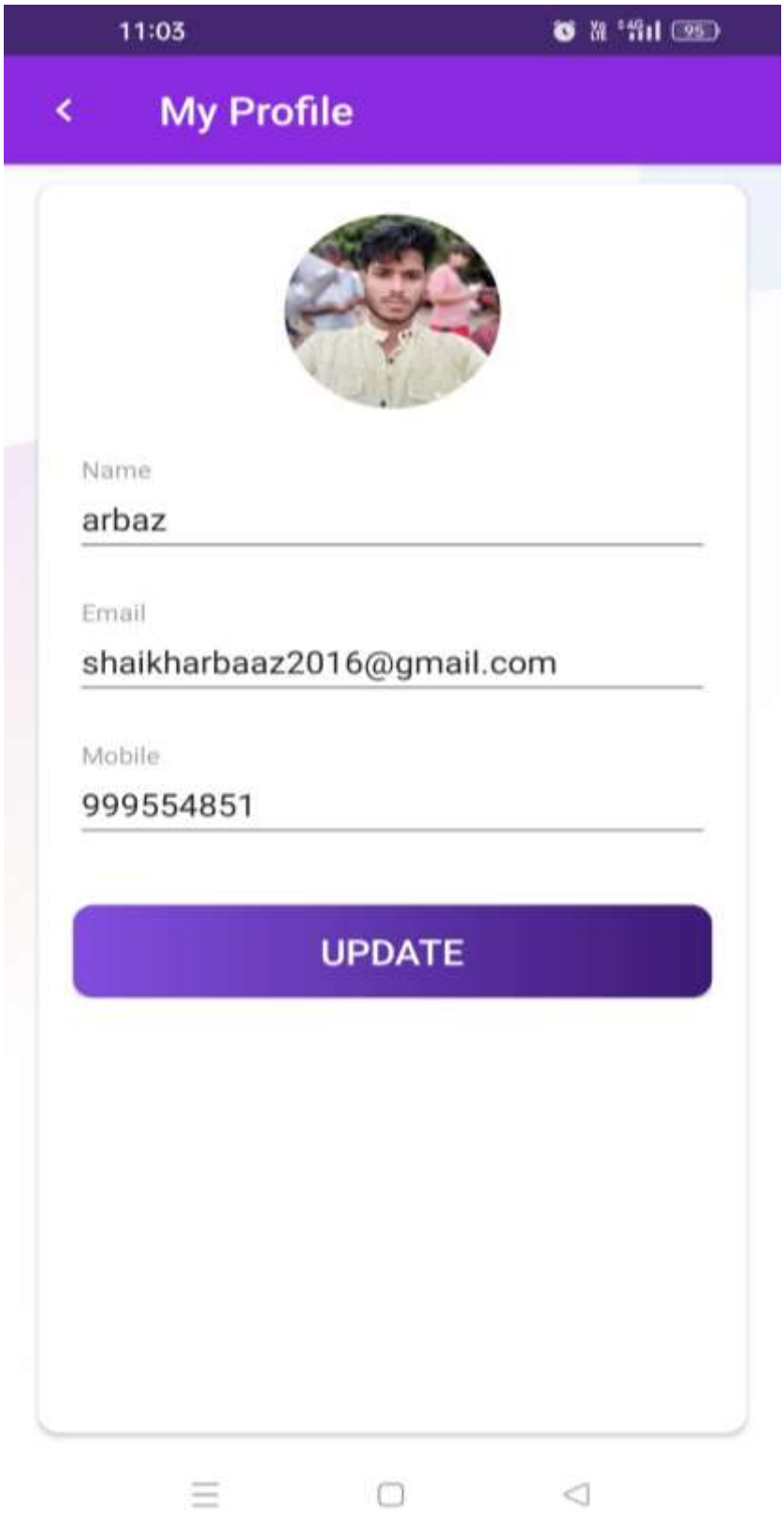


Figure 13: Profile Update

7. API call to fetch contest details.

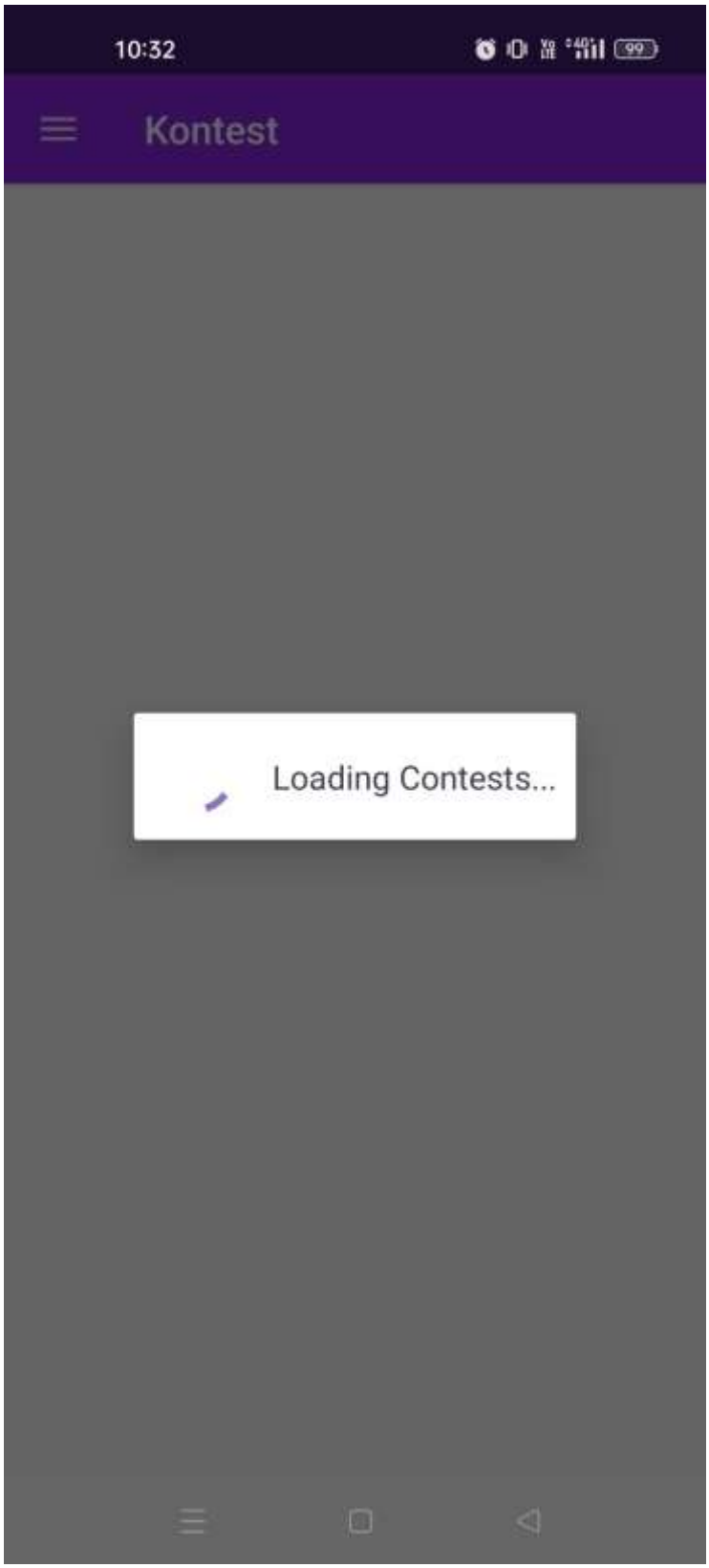


Figure 14: API Call

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Code

Splash Screen:

```
package com.kontest2.activities
// import required libraries
class SplashActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        //This call the parent constructor
        super.onCreate(savedInstanceState)
        // This is used to align the xml view to this class
        setContentView(R.layout.activity_splash)

        // This is used to hide the status bar and make the splash screen as a full screen activity.
        window.setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        )
        // This is used to get the file from the assets folder and set it to the title textView.
        val typeface: Typeface =
            Typeface.createFromAsset(assets, "carbon bl.ttf")
        tv_app_name.typeface = typeface

        // Adding the handler to after the a task after some delay.
        Handler().postDelayed({

            // Here if the user is signed in once and not signed out again from the app. So next time while coming into the app
            // we will redirect him to MainScreen or else to the Intro Screen as it was before.

            // Get the current user id
            val currentUserID = FirestoreClass().getCurrentUserID()
            // Start the Intro Activity
```

```
            if (currentUserID.isNotEmpty()) {
                // Start the Main Activity
                startActivity(Intent(this@SplashActivity, MainActivity::class.java))
            } else {
                // Start the Intro Activity
                startActivity(Intent(this@SplashActivity, IntroActivity::class.java))
            }
            finish() // Call this when your activity is done and should be closed.
        }, 2500) // Here we pass the delay time in milliSeconds after which the splash activity will disappear.
    }
}
```

Sign up:

```
package com.kontest2.activities

import android.os.Bundle
import android.text.TextUtils
import android.view.WindowManager
import android.widget.Toast
import com.google.android.gms.tasks.OnCompleteListener
import com.google.firebase.auth.AuthResult
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
import com.kontest2.R
import com.kontest2.firebase.FirestoreClass
import com.kontest2.model.User
import kotlinx.android.synthetic.main.activity_sign_up.*


class SignUpActivity : BaseActivity() {


    /**
     * This function is auto created by Android when the Activity Class is created.
```

```
 */
override fun onCreate(savedInstanceState: Bundle?) {
    //This call the parent constructor
    super.onCreate(savedInstanceState)
    // This is used to align the xml view to this class
    setContentView(R.layout.activity_sign_up)

    // This is used to hide the status bar and make the splash screen as a full screen activity.
    window.setFlags(
        WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN
    )
    setupActionBar()
    // Click event for sign-up button.
    btn_sign_up.setOnClickListener {
        registerUser()
    }
}
/**
 * A function for actionBar Setup.  */
private fun setupActionBar() {
    setSupportActionBar(toolbar_sign_up_activity)
    val actionBar = supportActionBar
    if (actionBar != null) {
        actionBar.setDisplayHomeAsUpEnabled(true)
        actionBar.setHomeAsUpIndicator(R.drawable.ic_black_color_back_24dp)
    }
    toolbar_sign_up_activity.setNavigationOnClickListener { onBackPressed() }
}

/**
 * A function to register a user to our app using the Firebase.
```

```
     * For more details visit: https://firebase.google.com/docs/auth/android/custom-auth
     */
    private fun registerUser() {
        // Here we get the text from editText and trim the space
        val name: String = et_name.text.toString().trim { it <= ' ' }
        val email: String = et_email.text.toString().trim { it <= ' ' }
        val password: String = et_password.text.toString().trim { it <= ' ' }

        if (validateForm(name, email, password)) {
            // Show the progress dialog.
            showProgressDialog(resources.getString(R.string.please_wait))
            FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(
                    OnCompleteListener<AuthResult> { task ->

                        // If the registration is successfully done
                        if (task.isSuccessful) {

                            // Firebase registered user
                            val firebaseUser: FirebaseUser = task.result!!.user!!
                            // Registered Email
                            val registeredEmail = firebaseUser.email!!

                            val user = User(
                                firebaseUser.uid, name, registeredEmail
                            )

                            // call the registerUser function of FirestoreClass to make an entry in the
database.
                            FirestoreClass().registerUser(this@SignUpActivity, user)
                        } else {
                            Toast.makeText(
```

```
                    this@SignUpActivity,
                    task.exception!!.message,
                    Toast.LENGTH_SHORT
                ).show()
            }
        })
    }
}

/**
 * A function to validate the entries of a new user.
 */
private fun validateForm(name: String, email: String, password: String): Boolean {
    return when {
        TextUtils.isEmpty(name) -> {
            showErrorSnackBar("Please enter name.")
            false
        }
        TextUtils.isEmpty(email) -> {
            showErrorSnackBar("Please enter email.")
            false
        }
        TextUtils.isEmpty(password) -> {
            showErrorSnackBar("Please enter password.")
            false
        }
        else -> {
            true
        }
    }
}
```

```
    /**
     * A function to be called the user is registered successfully and entry is made in the
firestore database.
     */
    fun userRegisteredSuccess() {


        Toast.makeText(
            this@SignUpActivity,
            "You have successfully registered.",
            Toast.LENGTH_SHORT
        ).show()


        // Hide the progress dialog
        hideProgressDialog()


        /**
         * Here the new user registered is automatically signed-in so we just sign-out the user
from firebase
         * and send him to Intro Screen for Sign-In
         */
        FirebaseAuth.getInstance().signOut()
        // Finish the Sign-Up Screen
        finish()
    }
}
```

Sign in:

```
package com.kontest2.activities
// import required libraries
class SignInActivity : BaseActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        //This call the parent constructor
        super.onCreate(savedInstanceState)
        // This is used to align the xml view to this class
```

```
    setContentView(R.layout.activity_sign_in)
    // This is used to hide the status bar and make the splash screen as a full screen activity.
    window.setFlags(
        WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN
    )
    setupActionBar()
    btn_sign_in.setOnClickListener {
        signInRegisteredUser()
    }
}
private fun setupActionBar() {
    setSupportActionBar(toolbar_sign_in_activity)
    val actionBar = supportActionBar
    if (actionBar != null) {
        actionBar.setDisplayHomeAsUpEnabled(true)
        actionBar.setHomeAsUpIndicator(R.drawable.ic_black_color_back_24dp)
    }
    toolbar_sign_in_activity.setNavigationOnClickListener { onBackPressed() }
}
/**
 * A function for Sign-In using the registered user using the email and password.
 */
private fun signInRegisteredUser() {
    // Here we get the text from editText and trim the space
    val email: String = et_email.text.toString().trim { it <= ' ' }
    val password: String = et_password.text.toString().trim { it <= ' ' }
    if (validateForm(email, password)) {
        // Show the progress dialog.
        showProgressDialog(resources.getString(R.string.please_wait))
        // Sign-In using FirebaseAuth
        FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password)
```

```
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    // Calling the FirestoreClass signInUser function to get the data of user from database.
                    FirestoreClass().loadUserData(this@SignInActivity)
                } else {
                    Toast.makeText(
                        this@SignInActivity,
                        task.exception!!.message,
                        Toast.LENGTH_LONG
                    ).show()
                }
            }
        }
    }
    /**
     * A function to validate the entries of a user.
     */
    private fun validateForm(email: String, password: String): Boolean {
        return if (TextUtils.isEmpty(email)) {
            showErrorSnackBar("Please enter email.")
            false
        } else if (TextUtils.isEmpty(password)) {
            showErrorSnackBar("Please enter password.")
            false
        } else {
            true
        }
    }
    /**
     * A function to get the user details from the firestore database after authentication.
     */
```

```
    fun signInSuccess(user: User) {

        hideProgressDialog()

        startActivity(Intent(this@SignInActivity, MainActivity::class.java))
        this.finish()
    }
}
```

Google Authentication:

```
// Google
    private lateinit var auth: FirebaseAuth
//    private val RC_SIGN_IN = 123

    override fun onCreate(savedInstanceState: Bundle?) {
        //This call the parent constructor
        super.onCreate(savedInstanceState)
        // This is used to align the xml view to this class
        setContentView(R.layout.activity_intro)

        // This is used to hide the status bar and make the splash screen as a full screen activity.
        window.setFlags(
            WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN
        )

        // This is used to get the file from the assets folder and set it to the title textView.
        val typeface: Typeface =
            Typeface.createFromAsset(assets, "carbon bl.ttf")
        tv_app_name_intro.typeface = typeface

        btn_sign_in_intro.setOnClickListener {
```

```
        // Launch the sign in screen.
        startActivity(Intent(this@IntroActivity, SignInActivity::class.java))
    }

    btn_sign_up_intro.setOnClickListener {

        // Launch the sign up screen.
        startActivity(Intent(this@IntroActivity, SignUpActivity::class.java))
    }

    // Configure Google Sign In
    val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(getString(R.string.web_client_id))
        .requestEmail()
        .build()

    val mGoogleSignInClient = GoogleSignIn.getClient(this, gso)
    auth = FirebaseAuth.getInstance()

        googleSignInBtn.setOnClickListener {
            showProgressDialog("Signing with Google...")
            val signInIntent = mGoogleSignInClient.signInIntent
            startActivityForResult(signInIntent, RC_SIGN_IN)
    }

}
```

Volley Instance for fetching API data:

```
package com.kontest2.activities

import android.content.Context
import com.android.volley.Request
import com.android.volley.RequestQueue
import com.android.volley.toolbox.Volley

class SingletonVolleyInst constructor(context: Context) {

    companion object {
        @Volatile
        private var INSTANCE: SingletonVolleyInst? = null
        fun getInstance(context: Context) =
            INSTANCE ?: synchronized(this) {
                INSTANCE ?: SingletonVolleyInst(context).also {
                    INSTANCE = it
                }
            }
    }

    private val requestQueue: RequestQueue by lazy {
        // applicationContext is key, it keeps you from leaking the
        // Activity or BroadcastReceiver if someone passes one in.
        Volley.newRequestQueue(context.applicationContext)
    }
    fun <T> addToRequestQueue(req: Request<T>) {
        requestQueue.add(req)
    }
}
```

Contest List Adapter Code:

```
package com.kontest2.activities


import android.view.LayoutInflater

import com.kontest2.R

import android.view.View

import android.view.ViewGroup

import android.widget.TextView

import androidx.recyclerview.widget.RecyclerView


class ContestListAdapter(private val listener:ContestItemClicked):
RecyclerView.Adapter<ContestViewHolder>() {


    private val items: ArrayList<Contest> = ArrayList()


    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
ContestViewHolder {

        val view = LayoutInflater.from(parent.context).inflate(R.layout.contest_view_layout,
parent, false)

        val viewHolder = ContestViewHolder(view)

        view.setOnClickListener{

            listener.onContClicked(items[viewHolder.adapterPosition])

        }

        return viewHolder

    }


    override fun getItemCount(): Int {

        return items.size

    }


    override fun onBindViewHolder(holder: ContestViewHolder, position: Int) {

        val currentItem = items[position]

//        holder.site_name.text = currentItem.site_name
```

```
        holder.cont_name.text = currentItem.cont_name
        holder.start_time.text = currentItem.start_time
//        holder.end_time.text = currentItem.end_time
    }

    fun updateContest(updatedContest: ArrayList<Contest>) {
        items.clear()
        items.addAll(updatedContest)

        notifyDataSetChanged()
    }
}

class ContestViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
//    val site_name: TextView = itemView.findViewById(R.id.site_name)
    val cont_name: TextView = itemView.findViewById(R.id.cont_name)
    val start_time: TextView = itemView.findViewById(R.id.start_time)
//    val end_time: TextView = itemView.findViewById(R.id.end_time)

}

interface ContestItemClicked {
    fun onContClicked(item: Contest)
}
```

Class of data type for storing contest details:

```
package com.kontest2.activities

data class Contest(
    val site_name: String,
    val cont_name: String,
```

```
    val start_time: String
    val end_time: String,
    val url: String
)
```

Firebase firestore Configuration:

```
package com.kontest2.firebase

import android.app.Activity
import android.util.Log
import android.widget.Toast
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.firestore.SetOptions
import com.kontest2.activities.MainActivity
import com.kontest2.activities.MyProfileActivity
import com.kontest2.activities.SignInActivity
import com.kontest2.activities.SignUpActivity
import com.kontest2.model.User
import com.kontest2.utils.Constants

/**
 * A custom class where we will add the operation performed for the firestore database.
 */
class FirestoreClass {

    // Create a instance of Firebase Firestore
    private val mFireStore = FirebaseFirestore.getInstance()

    /**
     * A function to make an entry of the registered user in the firestore database.
     */
```

```
    fun registerUser(activity: SignUpActivity, userInfo: User) {

        mFireStore.collection(Constants.USERS)
            // Document ID for users fields. Here the document it is the User ID.
            .document(getCurrentUserID())
            // Here the userInfo are Field and the SetOption is set to merge. It is for if we wants to merge
            .set(userInfo, SetOptions.merge())
            .addOnSuccessListener {

                // Here call a function of base activity for transferring the result to it.
                activity.userRegisteredSuccess()
            }
            .addOnFailureListener { e ->
                activity.hideProgressDialog()
                Log.e(
                    activity.javaClass.simpleName,
                    "Error writing document",
                    e
                )
            }
    }

    /**
     * A function to SignIn using firebase and get the user details from Firestore Database.
     */
    fun loadUserData(activity: Activity) {

        // Here we pass the collection name from which we wants the data.
        mFireStore.collection(Constants.USERS)
            // The document id to get the Fields of user.
            .document(getCurrentUserID())
```

```
            .get()
            .addOnSuccessListener { document ->
                Log.e(activity.javaClass.simpleName, document.toString())


                // Here we have received the document snapshot which is converted into the User Data model object.
                val loggedInUser = document.toObject(User::class.java)!!


                // Here call a function of base activity for transferring the result to it.
                when (activity) {
                    is SignInActivity -> {
                        activity.signInSuccess(loggedInUser)
                    }
                    is MainActivity -> {
                        activity.updateNavigationUserDetails(loggedInUser)
                    }
                    is MyProfileActivity -> {
                        activity.setUserDataInUI(loggedInUser)
                    }
                }
            }
            .addOnFailureListener { e ->
                // Here call a function of base activity for transferring the result to it.
                when (activity) {
                    is SignInActivity -> {
                        activity.hideProgressDialog()
                    }
                    is MainActivity -> {
                        activity.hideProgressDialog()
                    }
                    is MyProfileActivity -> {
                        activity.hideProgressDialog()
```

```
            }
        }
        Log.e(
            activity.javaClass.simpleName,
            "Error while getting loggedIn user details",
            e
        )
    }
}


/**
 * A function to update the user profile data into the database.
 */
fun updateUserProfileData(activity: MyProfileActivity, userHashMap: HashMap<String,
Any>) {
    mFireStore.collection(Constants.USERS) // Collection Name
        .document(getCurrentUserID()) // Document ID
        .update(userHashMap) // A hashmap of fields which are to be updated.
        .addOnSuccessListener {
            // Profile data is updated successfully.
            Log.e(activity.javaClass.simpleName, "Profile Data updated successfully!")


            Toast.makeText(activity, "Profile updated successfully!",
Toast.LENGTH_SHORT).show()


            // Notify the success result.
            activity.profileUpdateSuccess()
        }
        .addOnFailureListener { e ->
            activity.hideProgressDialog()
            Log.e(
                activity.javaClass.simpleName,
                "Error while creating a board.",
```

```
                e
            )
        }
    }

    /**
     * A function for getting the user id of current logged user.
     */
    fun getCurrentUserID(): String {
        // An Instance of currentUser using FirebaseAuth
        val currentUser = FirebaseAuth.getInstance().currentUser

        // A variable to assign the currentUserId if it is not null or else it will be blank.
        var currentUserID = ""
        if (currentUser != null) {
            currentUserID = currentUser.uid
        }

        return currentUserID
    }
}
```

Making API call, Fetching Data, Formatting data to better user experience then using recycler view to present it.

```
  //  API
    private fun fetchData() {
        showProgressDialog("Loading Contests...")
        val url = "https://codeforces.com/api/contest.list"
        val jsonObjectRequest = JsonArrayRequest(
            Request.Method.GET,
            url,
            null,
            { response ->
```

```
            val contestArray = ArrayList<Contest>()
            for (i in 0 until response.length()) {
                val contestJsonObject = response.getJSONObject(i)
                val site_name = contestJsonObject.getString("site")
                val cont_name = contestJsonObject.getString("name")
                val start_time = contestJsonObject.getString("startTimeSeconds")
                val end_time = contestJsonObject.getString("end_time")
                val url = contestJsonObject.getString("url")
                val contest = Contest(
                    site_name, cont_name, start_time, end_time, url
                      cont_name, start_time
                )
                contestArray.add(contest)
            }

            mAdapter.updateContest(contestArray)
            hideProgressDialog()
        },
        { error ->
            // Handle error
        }
    )
    SingletonVolleyInst.getInstance(this).addToRequestQueue(jsonObjectRequest)
}


// Function to convert seconds to a human-readable date and time
fun convertSecondsToDateTime(seconds: Long): String {
    val sdf = SimpleDateFormat("dd-MM-yyyy HH:mm:ss", Locale.getDefault())
    val date = Date(seconds * 1000) // Convert seconds to milliseconds
    return sdf.format(date)
}
```

```
//    Coder forces
private fun fetchData() {
    showProgressDialog("Loading Contests...")
    val url = "https://codeforces.com/api/contest.list"
    val jsonObjectRequest = JsonObjectRequest(
        Request.Method.GET,
        url,
        null,
        { response ->
            val contestArray = ArrayList<Contest>()
            if (response.has("result")) {
                val result = response.getJSONArray("result")
                for (i in 0 until result.length()) {
                    val contestJsonObject = result.getJSONObject(i)
                    val cont_name = contestJsonObject.getString("name")
                    val start_time = contestJsonObject.getLong("startTimeSeconds")
                    val formattedDateTime = convertSecondsToDateTime(start_time.toLong())
                    val contest = Contest(cont_name, formattedDateTime.toString())
                    contestArray.add(contest)
                }
                mAdapter.updateContest(contestArray)
            }
            hideProgressDialog()
        },
        { error ->
            // Handle error
            hideProgressDialog()
        }
    )
    SingletonVolleyInst.getInstance(this).addToRequestQueue(jsonObjectRequest)
}
```

```
    override fun onContClicked(item: Contest) {
        val u = "https://codeforces.com/"
        val builder =  CustomTabsIntent.Builder()
        val customTabsIntent = builder.build()
//        customTabsIntent.launchUrl(this, Uri.parse(item.url))
        customTabsIntent.launchUrl(this, Uri.parse(u))
    }
}
```

Firebase JSON file:

```
{
  "project_info": {
    "project_number": "1078987968633",
    "project_id": "kontest-785ee",
    "storage_bucket": "kontest-785ee.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:1078987968633:android:40fd0576be4a009d5f44a8",
        "android_client_info": {
          "package_name": "com.kontest"
        }
      },
      "oauth_client": [
        {
          "client_id": "1078987968633-
bsgh6qqocpc3qjimm2p7a93ssc85fsc2.apps.googleusercontent.com",
          "client_type": 3
        }
```

```
    ],
    "api_key": [
      {
        "current_key": "AIzaSyBlnNyW2UIcj_-0dKNFNg3vr7qc3v4XhTI"
      }
    ],
    "services": {
      "appinvite_service": {
        "other_platform_oauth_client": [
          {
            "client_id": "1078987968633-
bsgh6qqocpc3qjimm2p7a93ssc85fsc2.apps.googleusercontent.com",
            "client_type": 3
          }
        ]
      }
    }
  }
 ],
 "configuration_version": "1"
}
```

5.2 Testing Approach and Test Cases

Black Box Testing

1. Sign up

Test Case ID	Inputs	Expected Result	Actual Result	Remark
TC01	Name: "arbaz", Email: "arbaaz@gmail.com", Password: "password123"	Successful registration	Successful	Pass
TC02	Name: "sk", Email: "sk@gmail.com", Password: "123"	Registration fails due to short password (min. 6 characters)	Registration failed: Password too short	Pass
TC03	Name: "raj", Email: "raj$k.coml", Password: "pword123"	Registration fails due to invalid email format	Registration failed: Invalid email format	Pass
TC04	Name: "", Email: "ali@gmail.com", Password: "12467"	Registration fails due to missing name	Registration failed: Name is required	Pass
TC05	Name: "ALi", Email: "", Password: "089dda"	Registration fails due to missing email	Registration failed: Email is required	Pass
TC06	Name: "Yusuf", Email: "yusuf@gmail.com", Password: "p1"	Registration fails due to short password (min. 6 characters)	Registration failed: Password too short	Pass
TC07	Name: "Mohd", Email: "md@gmail.com", Password: "password"	Registration fails due to weak password (lacking complexity)	Registration failed: Password is weak	Fail

Table 9: Sign Up Testing (BB)

2. Sign in.

Test Case ID	Inputs	Expected Result	Actual Result	Remark
TC1	Email: “arbaaz@gmail.com”, Password: “password123”	Successful sign-in	Successful	Pass
TC2	Email: “arbaaz@gmail.com”, Password: “invalidpass”	Sign-in fails due to incorrect password	Sign-in failed: Incorrect password	Pass
TC3	Email: “ali@yahoo.com”, Password: “password123”	Sign-in fails due to invalid email format	Sign-in failed: Invalid email format	Pass
TC4	Email: “mohd@gmail.com”, Password: “”	Sign-in fails due to missing password	Sign-in failed: Password is required	Pass
TC5	Email: “”, Password: “123989133”	Sign-in fails due to missing email	Sign-in failed: Email is required	Pass
TC6	Email: “arbaaz08ali@gmail.com”, Password: “972789414”	Sign-in fails due to non-existent email	Sign-in failed: Email not found	Fail
TC7	Email: “shaikharbaaz@gmail.com”, Password: “password123”	Sign-in fails due to an already logged-in user trying to sign in again	Sign-in failed: User already logged in	Fail

Table 10: Sign In Testing (BB)

Instrumented Testing (Android Studio)

Code:

```
package com.kontest2

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.kontest", appContext.packageName)
    }
}
```

Result:

Pass. This test is an instrumented test, meaning it runs on a real device or emulator. Its primary purpose is to test the behavior of your app in a realistic environment.

@RunWith(AndroidJUnit4::class): This annotation indicates that this is an Android Instrumented Test and should be run with the AndroidJUnit4 test runner.

@Test Annotation on useAppContext(): This annotation denotes that the useAppContext() function is a test method.

InstrumentationRegistry.getInstrumentation().targetContext: This obtains the Context of the app under test.

assertEquals("com.kontest", appContext.packageName): This assertion checks whether the package name of the app under test is equal to "com.kontest". If it is, the test passes; otherwise, it fails.



Figure 15: JUNIT 4 Testing

Local Unit Testing (Android Studio)

Code

```
package com.kontest2

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

Result:
This test is a local unit test, meaning it runs on the development machine (host). Its primary purpose is to test the logic of your code in isolation from the Android framework.

@Test Annotation on addition_isCorrect(): This annotation denotes that the addition_isCorrect() function is a test method.

assertEquals(4, 2 + 2): This assertion checks whether the result of 2 + 2 is equal to 4. If it is, the test passes; otherwise, it fails.

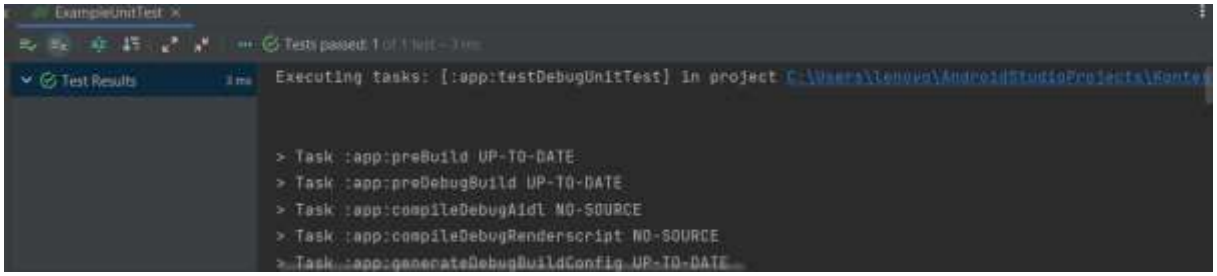


Figure 16: Local Unit Testing

CHAPTER 6
DISCUSSIONAND CONCLUSION

6.1 Discussion

- Platform Integration Challenges:
  Integrating data from diverse coding platforms such as LeetCode and Codeforces presented substantial challenges. Overcame discrepancies in APIs, data structures, and authentication methods through a flexible integration strategy. The experience highlighted the need for adaptability when working with disparate data sources.

- Security Considerations:
  Recognizing the significance of securing user data, especially in an app dealing with personal information. Implemented robust measures to ensure secure data transmission and storage, emphasizing user authentication security. Emphasizing security from the project's inception proved crucial in gaining user trust.

- User Experience Enhancement:
  Prioritizing a seamless user experience for easy navigation and interaction. Incorporated user feedback and usability studies to iteratively enhance the app's usability. Improved user satisfaction and engagement, making the app more user-friendly.

- Real-time Data Challenges:
  Ensuring real-time updates on upcoming and ongoing contests posed challenges. Implemented strategies to minimize latency, ensuring timely delivery of contest information. Realizing the importance of timely data updates for user satisfaction.

- Third-Party API Management:
  The reliance on third-party APIs introduced complexities and potential points of failure. Developed contingency plans to address unexpected API changes or disruptions. Highlighting the importance of robust API management strategies in maintaining app functionality.

- User Privacy and Compliance:
  Strict adherence to data privacy regulations and the terms of service of integrated coding platforms. Addressing legal and compliance challenges to ensure the app's ethics.

6.2 Conclusion

In the conclusion of the CCC project we find ourselves at the intersection of innovation and practicality. This venture embarked upon the creation of a dynamic platform catering to the vibrant community of coding enthusiasts. The journey was both challenging and enlightening, providing a profound learning experience for all those involved.

The project successfully addressed the burgeoning demand for a centralized hub that delivers comprehensive information on coding contests from diverse platforms such as LeetCode, Codeforces, and more. Through meticulous integration of APIs, the application seamlessly connects users with real-time data, keeping them abreast of upcoming and ongoing coding challenges.

Key achievements include overcoming the intricate challenges associated with disparate APIs, ensuring secure data transmission and storage, and enhancing the overall user experience. The iterative development process, influenced by user feedback, resulted in an intuitive interface that aligns with the expectations of our discerning audience.

As we navigate through the intricacies of third-party API management, the project stands resilient in its commitment to providing a reliable and efficient service. The realization of real-time updates, coupled with strategies to mitigate latency, attests to the technical prowess applied in the development phase.

In essence, the KONTEST project exemplifies the fusion of technological acumen, user-centric design, and a relentless pursuit of excellence. It is not just an application; it is a testament to the collaborative efforts that drive innovation in the ever-expanding realm of technology.

CHAPTER 7
LIMITATIONS

7.1 Limitations

- Dependency on External APIs:
  The project heavily relies on third-party APIs for real-time contest data. Any disruptions or modifications to these APIs may impact the application's functionality.

- Latency in Contest Updates:
  Due to the nature of external APIs and their update cycles, there might be instances where real-time contest updates face latency, leading to a brief delay in reflecting the latest information.

- Platform-Specific Constraints:
  Each coding platform follows its own set of rules and API structures. Adapting to changes in these structures requires constant monitoring and adjustments, posing a challenge to maintain uniformity.

- Limited Historical Data:
  The application primarily focuses on upcoming and ongoing contests. Access to historical data, such as past contest details, may be limited, constraining a comprehensive retrospective analysis.

- User Authentication Challenges:
  While efforts are made to ensure a seamless user experience, challenges related to user authentication, especially concerning custom and Google authentication, may be encountered.

- Platform Accessibility:
  The application is currently designed for Android platforms, limiting accessibility for users on other operating systems.

- Security and Privacy Concerns:
  Although security measures are implemented, the app interacts with sensitive user data. Continued vigilance and updates are required to address emerging security threats and privacy concerns.

CHAPTER 8
FUTURE WORK

8.1 Future Work

- Integration of Contest Reminders:
  Adding a notification system for upcoming contests would improve user engagement, ensuring they never miss an event.

- Historical Data Repository:
  Building a repository for historical contest data would enable users to explore past events, analyze trends, and make informed decisions.

- Cross-Platform Compatibility:
  Expanding the application's compatibility to iOS and other platforms would broaden its user base, fostering inclusivity.

- Community Features:
  Introducing features that encourage community engagement, such as discussion forums, user reviews, or collaborative problem-solving, could enhance the social aspect of the application.

- Offline Functionality Enhancement:
  Improving the app's functionality in offline mode would cater to users with limited internet connectivity, ensuring a seamless experience regardless of the network conditions.

- Machine Learning Integration:
  Leveraging machine learning algorithms could provide personalized contest recommendations based on a user's historical participation and preferences.

- Code Snippet Repository:
  Including a repository of code snippets or solutions for past problems from various contests could serve as a valuable resource for users looking to enhance their coding skills.

- Gamification Elements:
  Introducing gamification elements, such as badges, leaderboards, or challenges, could add a competitive and motivational aspect to the user experience.

- Continuous API Monitoring:
  Establishing a robust system for continuous monitoring of external APIs ensures prompt adaptation to changes and minimizes disruptions in real-time contest updates.

- User Feedback Mechanism:
  Implementing a feedback system allows users to report issues, suggest improvements, and actively contribute to the iterative development of the application.

- Integration with Educational Platforms:
  Collaborating with educational platforms to integrate educational resources, tutorials, or coding courses could position the app as a comprehensive learning tool.

Abbreviations:

Abbreviation	Meaning
CCC	Coding Contest Calendar (Project Name)
API	Application Programming Interface
KONTEST	Alternate Name to Coding Contest Calendar
UML	Unified Modelling Language
DVM	Dalvik Virtual Machine
JVM	Java Virtual Machine
ART	Android Run Time
GCP	Google Cloud Platform
AWS	Amazon Web Services
ABS	Android Backend Services
JIT	Just In Time
AOT	Ahead Of Time
AOS	Android Operating System
AI & ML	Artificial Intelligence & Machine Learning
BBT	Black Box Testing
FIRE	Firebase Integrated Resources and Environment
APK	Android Application Package
JSON	JavaScript Object Notation.

References:

1. Supriya S. & Pawar, S. B. (2015). Comparative Study of SQL & NoSQL Databases. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). Volume 4 Issue 5, May 2015
2. Sasankar P., & Kosarkar U. (2016). Research on Development of Android Applications, in IOSR Journal of Computer Engineering e-ISSN: 2278-0661, 2278-8727 PP 28-32
3. Panchal R. K. & Patel A. K. (2017). A comparative study: Java Vs Kotlin Programming in Android, in International Journal of Innovative Trends in Engineering & Research, September 2017, vol 2 Issue 9, pp 4-10
4. Khawas C. & Shah P. (2018). Application of Firebase in Android App Development-A Study, International Journal of Computer Applications (0975 – 8887) Volume 179 – No.46
5. Mateus B. & Martinez M. (2019). An empirical study on quality of Android applications written in Kotlin language, Empir. Softw. Eng., DOI: https://doi.org/10.1007/s10664-019-09727-4.
6. Jawad H. M. (2019). Android Mobile App Development as a Motivation towards Computer Programming. https://sci-hub.se/10.1109/EIT.2019. 8833858.
7. Joshua O., Richard B., John E. (2019). Application Programming Interface (API) Research: A Review of the Past to Inform the Future, in International Journal of Enterprise Information Systems Volume 15, Issue 3.
8. Keny N. (2019). A Review on Kotlin and Android Studio Java, in International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, 2019
9. Ardito L. et. al (2020). Effectiveness of Kotlin vs. Java in Android App Development Tasks, in Information and Software Technology 127:106374 DOI:10.1016/j.infsof.2020.106374
10. Putrant B. P., et. al (2020). A Comparative Study of Java and Kotlin for Android Mobile Application Development, in 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) DOI: 10.1109/ISRITI51436.2020.9315483
11. Chougale P., Yadav V., & Gaikwad A. (2021). Firebase Overview and Usage, in International Research Journal of Modernization in Engineering Technology and Science, Volume 03 Issue:12, December 2021 e-ISSN: 2582-5208 www.irjmets.com
12. Nishanthan M. R., (2022). An Empirical Study on Architecture and Efficiency of Dalvik VM and Android Runtime (ART), in International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Available at www.ijraset.com

13. Mohammad I. B. A., Martin J. (2023). Leveraging ChatGPT in Android App Development, in preprints.org doi: 10.20944/preprints202307.0660.v1
14. I. P. Kalle, “Weather Information Integrated Into A Travel Application by.”
agiledata, “Introduction to Test Driven Development (TDD).” http://agiledata.org /essays/tdd.html.
15. F. Muntenescu,android-architecture-patterns-part-1-model-view-controller, medium.com. https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6.
16. Gupta, R., et al. (2018). "Enhancing Mobile App Security through Behavior Analysis and Machine Learning." Journal of Mobile Technology in Medicine, 7(2), 59-65.
17. Kim, Y., et al. (2017). "Firebase: Impact on Android App Development." International Journal of Computer Applications, 160(5), 18-22.
18. Rahman, A., et al. (2016). "Efficient Data Fetching in Android Apps." International Journal of Computer Applications, 142(13), 1-5.
19. Arun, S., et al. (2019). "User Interface Design Trends in Android Development." International Journal of Computer Applications, 975, 8887.
20. Jain, A., et al. (2015). "Kotlin for Android Development: A Comprehensive Study." International Journal of Computer Applications, 123(8), 16-22.
21. Li, S., et al. (2018). "API Integration Challenges in Android Development." Journal of Software Engineering and Applications, 11(2), 56-65.
22. Chen, Q., et al. (2017). "Firebase vs. AWS: A Comparative Analysis for Mobile App Backend Services." International Journal of Computer Applications, 170(1), 15-20.
23. Wang, H., et al. (2019). "Optimizing Network Communication in Kotlin-Based Android Apps." Journal of Computer Science and Technology, 19(2), 56-62.
24. Xu, L., et al. (2016). "Augmented Reality in Android App Development." Journal of Mobile Technology in Medicine, 5(3), 32-38.
25. Singh, R., et al. (2017). "Cross-Platform Mobile Development: A Comparative Review." International Journal of Computer Applications, 169(8), 1-5.
26. developer.android.com, “Android Studio release notes.” https://developer.android.com /studio/releases.
27. developers.google. Online “https://developers.google.com/maps/documentation/android-sdk/overview.
28. “Android Programming: The Big Nerd Ranch Guide" by Bill Phillips, Chris Stewart, Big Nerd Ranch.

29. "Kotlin Programming: The Big Nerd Ranch Guide" by Josh Skeen, David Greenhalgh, Big Nerd Ranch.
30. "Firebase Essentials - Android Edition" by Neil Smyth.
31. "Effective Java" by Joshua Bloch.
32. "RESTful API Design: Best Practices in API Design with REST" by Matthias Biehl.
33. https://developer.android.com/studio/documentation
34. https://stackoverflow.com/
35. https://developer.android.com/guide
36. https://www.androidauthority.com/
37. Android App Development For Dummies" by Michael Burton, Wiley, 3rd Edition, 2018.
38. "Kotlin in Action" by Dmitry Jemerov, Svetlana Isakova, Manning Publications, 1st Edition, 2017.
39. "Firebase: Up and Running" by Mike McDonald, Sean Shadmand, O'Reilly Media, 1st Edition, 2019.
40. "Head First Android Development" by Dawn Griffiths, David Griffiths, O'Reilly Media, 2nd Edition, 2017.
41. "Kotlin for Android Developers" by Antonio Leiva, Leanpub, 1st Edition, 2015.
42. "Learning Kotlin by building Android Applications" by Eunice Adutwumwaa Obugyei, Packt Publishing, 1st Edition, 2018.
43. "Mastering Firebase for Android Development" by Ashok Kumar S, Packt Publishing, 1st Edition, 2018.
44. "Effective Kotlin: 52 Specific Ways to Improve Your Use of Kotlin" by Marcin Moskala, Igor Wojda, Addison-Wesley, 1st Edition, 2018.
45. "RESTful Java Web Services" by Jose Sandoval, Packt Publishing, 3rd Edition, 2015.
46. "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin, Prentice Hall, 1st Edition, 2008.