**VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI.**

**MAD Mini Project Report on**

# "ASSOCIATION OF COMPUTER ENGINEER ACE APP"

**Submitted by**

1. **SHILPA DATTAJIRAO PATIL 2KL20CS087**
2. **CHAITRA MANJUNATH GUNAGA 2KL20CS022**
3. **AYESHA MULLA 2KL20CS018**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**KLE Dr. M. S. SHESHGIRI**

**COLLEGE OF ENGINEERING & TECHNOLOGY,**

**UDYAMBAG, BELAGAVI – 590 008**

**Academic Year 2022-23**

**KLE DR. M. S. SHESHGIRI COLLEGE OF ENGINEERING & TECHNOLOGY, Udyambag, Belagavi-590008.**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

## Certificate

This is to certify that the CGV Mini Project entitled **"ASSOCIATION OF COMPUTER ENGINEERS ACE APP"** carried out by **Ms. CHAITRA MANHUNATH GUNAGA** bearing **USN** 2KL20CS022 ,**Ms. SHILPA DATTAJIR AO PATIL** bearing **USN** 2KL20CS087 **and Ms. AYESHA MULLA** bearing **USN** 2KL20CS018 have satisfactorily completed the academic requirements for the partial fulfilment of ***MAD Laboratory with Mini Project (18CSMP68)*** of VI Semester Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi for the Academic year 2022-2023.

**Project Coordinator** **HOD**

| | |
|---|---|
| **Internal Examiner** | **External Examiner** |
| Name: | Name: |
| Signature: | Signature: |
| Date: | Date: |

## ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to Principal and Management, KLE Dr. M.S.Sheshgiri College of Engineering and Technology, Belagavi, for continuous support and provision of all necessary requirements needed for the completion of this project.

We would also like the thank our HOD Dr. Rajashri Khanai , and our mentor Prof.Anirudha Potdar and Prof. Anita Kenchannavar for guiding us throughout the project and for providing such great source of guidance through out the course of the completion of the project.

Our hearty gratitude also goes out to all the faculty members of Computer Science and Engineering department, members of KLE Dr. MSSCET, and our parents for their constant encouragement, valuable guidance and for providing necessary help and support in all ways possible.

# ABSTRACT

The Association of Computer Engineers (ACE) is an innovative mobile application designed to connect and empower computer engineers worldwide. With the rapid advancements in technology, the need for a platform that fosters collaboration, knowledge sharing, and professional growth within the computer engineering community has become crucial. ACE fills this gap by providing a comprehensive and user-friendly app that caters specifically to the needs of computer engineers.

The ACE app offers a range of features and functionalities that enhance networking and knowledge exchange among computer engineers. Users can create personalized profiles, showcasing their skills, expertise, and professional achievements. The app's robust search and recommendation system enables users to connect with like-minded professionals, forming valuable networks and communities. ACE also hosts discussion forums, where members can engage in lively conversations, seek advice, and share industry insights.

In addition to networking opportunities, ACE offers a diverse selection of educational resources. The app provides access to a vast library of technical articles, tutorials, and video lectures, covering various aspects of computer engineering. Users can stay updated with the latest trends, emerging technologies, and industry news through curated content tailored to their interests.

ACE is not just a networking and learning platform but also a career development tool. The app features a job board where users can explore job opportunities specifically targeted towards computer engineers. It also offers career guidance resources, including interview tips, resume building assistance, and mentorship programs, to help users advance in their professional journeys.

With its intuitive interface, ACE aims to bridge the gap between computer engineers across different geographical locations and facilitate collaboration on global projects. By connecting professionals from diverse backgrounds, ACE fosters a culture of knowledge sharing, creativity, and innovation.

# TABLE OF CONTENTS

# CHAPTER 1: INTRODUCTION

## 1.1BACKGROUND

The Association of Computer Engineers (ACE) app was developed in response to the growing demand for a centralized platform that caters specifically to the needs of computer engineers. Computer engineering is a rapidly evolving field, encompassing various disciplines such as software development, hardware design, networking, and systems analysis. As technology continues to advance, computer engineers face the challenge of staying updated with the latest trends, expanding their professional networks, and accessing relevant resources.

Recognizing this need, a team of computer engineering professionals and experts came together to create the ACE app. The primary goal was to provide a comprehensive solution that would empower computer engineers by offering a range of features tailored to their unique requirements. The app aimed to bridge the geographical gaps, foster collaboration, and enhance professional growth within the computer engineering community.

Extensive research and market analysis were conducted to understand the specific pain points and requirements of computer engineers. This information was used to design the ACE app, ensuring that it addressed key areas such as networking, knowledge sharing, career development, and access to educational resources.

The development process involved a multidisciplinary team comprising software engineers, UX/UI designers, and domain experts in computer engineering. Their combined expertise and insights guided the creation of a user-friendly interface, intuitive navigation, and seamless functionalities.

Throughout the development phase, user feedback played a crucial role. Focus groups and beta testing were conducted to gather insights, identify potential improvements, and ensure that the app met the needs and expectations of its target audience—computer engineers at all stages of their careers.

With a robust and feature-rich design, the ACE app was launched, aiming to empower computer engineers globally. The app continues to evolve and adapt to the changing needs of its users, incorporating new features, expanding its resources, and enhancing the overall user experience.

Overall, the ACE app stands as a testament to the commitment of its creators in supporting the professional development and growth of computer engineers. By providing a centralized platform, ACE strives to create a thriving community that fosters collaboration, knowledge sharing, and career advancement for computer engineers worldwide.

### 1.1.1 IMPORTANCE OF ANDROID APPS

Android apps hold significant importance in today's digital landscape. Here are some key reasons why Android apps are important:

**Wide User Base:** Android is the most popular mobile operating system globally, with a substantial user base. Developing an Android app allows you to reach a wide audience and tap into the vast market of Android users.

**Market Penetration:** Android's market share is consistently increasing, surpassing other platforms. By having an Android app, you can penetrate deeper into the mobile market and gain a competitive edge.

**Accessibility:** Android apps are accessible to users across various devices, including smartphones, tablets, smart TVs, and wearables. This versatility ensures that your app can reach users on different platforms and devices, increasing its usability and potential user engagement.

**Customization and Flexibility:** Android provides a highly customizable platform, allowing developers to create apps that meet specific business requirements and user preferences. Android's open-source nature enables developers to modify the operating system and leverage a wide range of APIs and tools to build feature-rich and innovative apps.

**Google Play Store:** Android apps are primarily distributed through the Google Play Store, which serves as a centralized marketplace for millions of apps. This platform offers extensive visibility and discoverability for your app, making it easier for users to find, download, and install your application.

**Revenue Generation:** Android apps provide numerous monetization opportunities, including in-app purchases, advertisements, subscriptions, and premium versions. By capitalizing on these revenue streams, Android apps can generate income and contribute to your business's financial success.

**Integration with Google Services:** Android seamlessly integrates with various Google services, such as Google Maps, Google Analytics, Google Drive, and Google Pay. These integrations allow developers to leverage powerful functionalities and enhance the user experience by integrating their apps with popular Google services.

**Developer-Friendly Ecosystem:** Android offers a robust and developer-friendly ecosystem with comprehensive documentation, abundant resources, and a thriving community. This ecosystem fosters innovation, encourages collaboration, and provides developers with the tools and support they need to create high-quality apps.

**Fragmentation and Device Diversity:** Although device fragmentation can present challenges, it also offers opportunities. Android supports a wide array of devices with varying screen sizes, hardware capabilities, and specifications. By catering to this diversity, Android apps can target a broad range of devices and reach a larger user base.

In summary, Android apps are vital for businesses and developers due to their extensive user base, market penetration, accessibility, customization options, monetization potential.

### 1.1.2 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Android app development. It provides a comprehensive set of tools and features that empower developers to create high-quality, robust, and feature-rich Android applications. Here are a few key aspects of Android Studio:

**Powerful Development Environment:** Android Studio offers a powerful and user-friendly development environment, equipped with a code editor, project management tools, and an intuitive user interface. It streamlines the development process, enabling developers to write, debug, and test their code efficiently.

**Android Emulator:** Android Studio includes a built-in Android Emulator, which allows developers to test their apps on virtual devices with different configurations and Android versions. This feature helps ensure app compatibility across a wide range of devices.

**Intelligent Code Editor:** The IDE features a smart code editor that provides suggestions, auto-completion, and real-time error checking. It supports various programming languages, including Java and Kotlin, and offers advanced features like code refactoring and navigation, making coding faster and more efficient.

**Rich Layout Editor:** Android Studio provides a visual Layout Editor that allows developers to design app layouts intuitively. It offers drag-and-drop functionality, precise control over UI elements, and instant previews, enabling developers to create visually appealing and responsive user interfaces.

**Advanced Debugging and Profiling Tools:** Android Studio offers a suite of powerful debugging and profiling tools to identify and fix issues in the app's code and performance. These tools include breakpoints, logcat viewer, memory and CPU profilers, and network analysis tools, ensuring high-quality and optimized app development.

**Gradle Build System:** Android Studio integrates the Gradle build system, which automates the build process and manages dependencies efficiently. It allows developers to customize build configurations, manage libraries, and handle resource optimization, simplifying the app building process.

**App Publishing and Distribution:** Android Studio provides seamless integration with the Google Play Store, enabling developers to easily publish their apps to the global Android user base. It offers tools for generating signed APKs, managing app versions, and monitoring user feedback, simplifying the app distribution process.

**Extensive Plugin Ecosystem:** Android Studio supports a wide range of plugins and extensions, allowing developers to customize and enhance the IDE's functionality. These plugins provide additional tools, libraries, and integrations, enabling developers to streamline their workflows and leverage third-party resources.

Overall, Android Studio is a powerful and feature-rich IDE that empowers developers to build cutting-edge Android applications. Its comprehensive set of tools, intuitive interface, and seamless integration with the Android ecosystem make it an indispensable tool for Android app development.

### 1.1.3 FIREBASE

Firebase is a powerful and comprehensive mobile and web development platform offered by Google. It provides developers with a suite of tools and services that simplify and accelerate the development process. Here are a few key aspects of Firebase:

**Real-time Database:** Firebase offers a real-time NoSQL database that enables developers to store and sync data in real time across multiple clients. This feature is particularly useful for building collaborative applications, chat apps, and other scenarios where real-time data synchronization is critical.

**Authentication and Authorization:** Firebase provides built-in authentication and authorization services, allowing developers to easily add user authentication to their apps. It supports various authentication methods like email/password, social media logins, and anonymous authentication, helping developers implement secure user authentication quickly.

**Cloud Firestore:** Firebase's Cloud Firestore is a flexible and scalable cloud database that provides powerful querying capabilities, offline data synchronization, and automatic scaling. It allows developers to build complex data models and perform efficient queries, ensuring efficient data management for their apps.

**Cloud Functions:** Firebase's Cloud Functions allows developers to write serverless functions that can be triggered by events in the Firebase ecosystem. This feature enables developers to implement server-side logic without managing servers, helping them handle background tasks, process data, and integrate with external services seamlessly.

**Cloud Storage:** Firebase's Cloud Storage provides developers with secure and scalable cloud-based file storage. It allows easy upload and retrieval of user-generated content such as images, videos, and documents. Developers can also leverage Firebase Security Rules to control access to stored files.

**Analytics and Crash Reporting:** Firebase offers comprehensive analytics tools to track user engagement, app usage, and conversions. It provides valuable insights to optimize app performance and user experience. Additionally, Firebase Crash Reporting helps developers identify and resolve app crashes, enabling them to deliver stable and reliable apps.

**Cloud Messaging:** Firebase Cloud Messaging (FCM) enables developers to send notifications and messages to users across various platforms. It supports both targeted and mass messaging, helping developers engage with users and keep them informed about important updates and events.

**Hosting and Dynamic Links:** Firebase allows developers to host their web content and static files with ease. It provides a fast and reliable content delivery network (CDN) for hosting web apps, along with support for custom domains. Firebase Dynamic Links simplify the sharing of deep links, enabling developers to direct users to specific content within their apps.

Overall, Firebase offers a comprehensive set of tools and services that facilitate mobile and web app development.

## 1.2 MOTIVATION & OBJECTIVE

### MOTIVATION

There are several motivations and benefits for creating an Association of Computer Engineers (ACE) app. Here are a few:

**Knowledge Sharing and Innovation:** The ACE app encourages knowledge sharing among computer engineers, promoting innovation and creativity within the field. Through discussion forums, members can seek advice, share experiences, and collaborate on projects. This collaborative environment fosters the exchange of ideas and drives technological advancements within the computer engineering community.

**Community Building:** The ACE app helps create a sense of community and belonging for computer engineers. It brings professionals together, irrespective of geographical boundaries, fostering a supportive environment where individuals can connect, collaborate, and find mentorship. This community-building aspect strengthens professional relationships and provides a platform for individuals to seek guidance and support.

Overall, the ACE app provides a comprehensive platform that addresses the unique needs of computer engineers. It fosters networking, professional development, career opportunities, knowledge sharing, community building, and visibility within the industry. By creating and utilizing such an app, computer engineers can enhance their professional growth, stay connected with their peers, and contribute to the advancement of the field as a whole.

### OBJECTIVES

The objectives of the Association of Computer Engineers (ACE) app can be outlined as follows:

**Continuous Learning:** Offer a comprehensive selection of technical articles, tutorials, video lectures, and industry insights to keep computer engineers updated with the latest advancements and best practices in the field.

**User Experience:** Design a user-friendly and intuitive app interface that ensures a seamless and enjoyable user experience, making it easy for computer engineers to navigate and utilize the app's features.

**Continuous Improvement:** Gather user feedback, conduct surveys, and analyze app usage data to continuously improve and enhance the features, functionalities, and overall user experience of the ACE app.

By achieving these objectives, the ACE app aims to empower computer engineers, foster collaboration and innovation, and contribute to the professional growth and success of individuals within the computer engineering community.

## 1.3PROBLEM STATEMENT

Design and implement a ACE app that aims to address these challenges by providing a comprehensive solution that offers networking opportunities, access to relevant resources, career development support, a supportive community, efficient knowledge sharing mechanisms, and increased visibility within the computer engineering industry.

By tackling these problems, the ACE app strives to empower computer engineers and contribute to their professional growth, collaboration, and success. Its these challenges that we need to tackle by making an app which will improve the connectivity between people especially engineers in this case. This app can be used by students , employees to reach out and connect spread information and knowledge which is the motto of the association of computer engineers (ACE).

## 1.4 PROBLEM DISCUSSION

When building the Association of Computer Engineers (ACE) app using Android Studio, you may encounter several challenges. Some common problems that developers face during the app development process using Android Studio include:

**Learning Curve**: Android Studio has a learning curve, especially for developers who are new to the platform or have limited experience with Java or Kotlin programming languages. Understanding the IDE, its features, and the Android development ecosystem may take time and effort.

**Device Fragmentation:** Android devices come in various screen sizes, resolutions, and hardware configurations. Ensuring that the ACE app is compatible and functions optimally across different devices can be challenging. Testing on various devices and handling device-specific issues, such as performance variations, can be time-consuming.

**User Interface Design:** Designing an intuitive and visually appealing user interface requires a good understanding of Android's UI components, layout techniques, and Material Design guidelines. Striking the right balance between aesthetics and functionality while ensuring a consistent user experience across devices can be challenging.

**API Integration:** Integrating various APIs, such as networking tools, authentication services, and cloud storage, into the ACE app can be complex. Dealing with different API documentation, authentication mechanisms, error handling, and data parsing can present challenges during the integration process.

**Performance Optimization:** Ensuring smooth performance and responsiveness of the ACE app is crucial for a good user experience. Optimizing memory usage, minimizing network requests, implementing efficient data caching, and handling background tasks can be challenging but necessary to deliver a performant app.

**Data Management:** Efficiently managing data within the app, including user profiles, networking connections, and resource repositories, requires careful database design and implementation. Handling data synchronization, ensuring data consistency, and implementing effective data storage strategies can pose challenges during development.

**Testing and Debugging:** Thorough testing and debugging are essential to identify and fix issues in the ACE app. Testing across different device configurations, screen sizes, and operating system versions, as well as handling edge cases and ensuring app stability, can be time-consuming and complex.

**App Deployment and Distribution:** Preparing the ACE app for deployment to the Google Play Store involves meeting various requirements, such as app signing, creating store listings, and adhering to publishing guidelines. Ensuring a smooth rollout and handling user feedback and app updates can be challenging during the distribution phase.

To overcome these challenges, it is essential to leverage resources such as official Android documentation, online tutorials, and community forums. Continuous learning, thorough testing, and collaborating with experienced Android developers can help navigate and overcome the challenges of building the ACE app using Android Studio.

# CHAPTER 2: SOFTWARE REQUIREMENTS & SPECIFICATIONS

## 2.1 FUNCTIONAL REQUIREMENTS

**User Registration and Authentication:** Allow users to create an account and provide necessary information. Implement a secure authentication mechanism to ensure user privacy and data protection.

**User Profiles:** Enable users to create and manage their profiles, including personal information, skills, experience, and educational background. Allow users to update their profiles and add a profile picture.

**Knowledge Sharing:** Offer a platform for computer engineers to share articles, tutorials, and technical insights. Allow users to post and comment on discussions related to various topics in computer engineering. Enable users to upvote, bookmark, or save articles and discussions for future reference.

**Resource Repository:** Provide a repository of resources such as technical articles, whitepapers, research papers, and case studies. Organize resources into categories or tags for easy navigation and searchability. Allow users to contribute to the resource repository by submitting relevant content.

**Events and Meetups:** Display information about relevant industry events, conferences, and meetups. Allow users to RSVP, view event details, and participate in discussions related to events.

**User Settings and Preferences:** Allow users to customize app settings, notification preferences, and privacy options. Provide options to control visibility of certain profile information or restrict access to specific features.

## 2.2 NON- FUNCTIONAL REQUIREMENTS:

**Performance:** The app should have fast response times and load content quickly to provide a smooth user experience. It should be able to handle concurrent user interactions and scale effectively as the user base grows.

**Usability:** The app should have an intuitive and user-friendly interface to ensure ease of use for computer engineers of varying technical backgrounds. It should follow established design principles and guidelines to provide a consistent and familiar user experience.

**Compatibility:** The app should be compatible with a wide range of Android devices, operating system versions, and screen sizes to ensure broad accessibility for users. It should also consider compatibility with future updates and changes to the Android platform.

**Security:** The app should employ robust security measures to protect user data, including encryption of sensitive information and secure authentication mechanisms. It should adhere to industry standards and best practices for data security and privacy.

## 2.3 HARDWARE REQUIREMENTS

Hardware requirement involves defining prerequisites that need to be there on a computer for the application to function optimally.
The hardware requirements for our project are:

• **Processor:** INTEL / AMD
• **Main memory**: 16 GB RAM (Min.)
• **Hard Disk:** Built-in
• **Keyboard:** QWERTY
• **Mouse:** 2 or 3 Button mouse
• **Monitor:** 1024 x 768

## 2.4 SOFTWARE REQUIREMENTS

Software requirement involves defining prerequisites that need to be installed on a computer for the application to function optimally.
The software requirements for our project are:

• **Programming language** – Java, XML
• **Operating system** – Windows 10 / 11
• **IDE** – Android Studio
• **Database** – Firebase

## 2.5 LANGUAGES USED FOR IMPLEMENTATION

Java and XML are two crucial components in Android app development. Here's a brief overview of their roles in building the ACE (Association of Computer Engineers) app:

**Java:**
Java is the primary programming language used for Android app development. It provides a robust and object-oriented foundation for building Android applications. In the ACE app, Java is used for implementing the app's logic, handling data processing, managing user interactions, and integrating with various Android APIs and frameworks.

Key uses of Java in the ACE app development process include:

**Activity and Fragment Classes:** Java is used to create and define activities and fragments, which are the building blocks of the app's user interface and interaction logic.

**Event Handling:** Java allows you to define event listeners and handle user interactions, such as button clicks, navigation actions, or form submissions.

**Networking:** Java provides classes and libraries for making network requests, fetching data from APIs, and handling responses in the app.

**Data Processing:** Java allows you to manipulate and process data retrieved from external sources or stored locally in the app, such as filtering, sorting, or performing calculations.

**Business Logic:** Java is responsible for implementing the app's business logic, including handling user authentication, managing user profiles, connecting with external services, and performing data validation.

**XML (Extensible Markup Language):**

XML is a markup language used for structuring and describing data. In Android app development, XML is primarily used for defining the user interface layout and configuring various app resources.

Key uses of XML in the ACE app development process include:

**Layouts:** XML is used to define the structure and arrangement of user interface elements in activities and fragments. XML layout files specify the position, size, and behavior of UI components such as buttons, text views, lists, and images.

**Resources:** XML is used to define app resources, such as strings, colors, dimensions, styles, and themes. These resources can be accessed and utilized in Java code to ensure consistency and modularity throughout the app.

**Menus:** XML is used to create menus for the app, defining the options and actions available to users in various contexts.

**Data Configuration:** XML can be used for configuring data files, such as defining database schemas, specifying network endpoints, or storing configuration settings.

By utilizing Java and XML effectively, you can develop the ACE app with a well-structured user interface, robust functionality, and efficient data processing capabilities.

# CHAPTER 3: DESIGN AND ANALYSIS

## 3.1 DESIGN

Designing the ACE (Association of Computer Engineers) app involves creating an intuitive and visually appealing user interface that aligns with the app's objectives and enhances the user experience. Here are some key considerations for designing the app:

**User-Centric Approach:** Keep the target audience of computer engineers in mind throughout the design process. Understand their needs, preferences, and goals to create a user-centric experience.

**Home Screen:** The home screen should provide a snapshot of the app's key features and allow users to access important sections quickly. Consider incorporating widgets or cards to display personalized recommendations, upcoming events, or recent activities.

**User Profiles:** Design user profiles that allow computer engineers to showcase their skills, experience, and achievements. Include sections for profile pictures, professional summaries, education details, work history, and areas of expertise.

**Networking and Connections:** Create a user-friendly interface for searching and connecting with other computer engineers. Include options to filter search results based on location, skills, or industry experience. Implement a seamless connection request and acceptance mechanism.

**Content Organization:** Group related content into logical categories to make it easy for users to find resources, articles, discussions, and job opportunities. Implement sorting and search functionalities to enhance content discoverability.

**Resource Repository:** Design a visually appealing and easily navigable resource repository. Use appropriate layouts and filters to categorize resources by type, topic, or relevance. Include options for users to rate, bookmark, and share resources.

**Job Opportunities:** Design a dedicated section for job listings, including company names, job titles, descriptions, and application details. Allow users to filter and search for job opportunities based on location, experience level, or specific skills.

**Responsiveness and Accessibility:** Design the app to be responsive and adaptable to different screen sizes and orientations. Ensure that the app meets accessibility guidelines, including support for screen readers, adjustable font sizes, and appropriate color contrasts.

**Iterative Design and User Feedback:** Continuously gather user feedback and iterate on the design based on user needs and preferences. Conduct usability testing to identify any usability issues or pain points and make necessary improvements.

Remember, the design should align with the objectives of the ACE app and provide a seamless and enjoyable experience for computer engineers. Regularly evaluate and refine the design based on user feedback to ensure continuous improvement.

## 3.2ANALYSIS

To provide an analysis of the ACE (Association of Computer Engineers) app, it's important to assess various aspects of the app's functionality, user experience, and potential impact. Here's an analysis framework that covers key areas:

**Functionality:**
Evaluate the app's adherence to functional requirements, such as user registration, networking, knowledge sharing, job opportunities, and event management. Assess if all the required features have been implemented effectively.
Identify any gaps or limitations in functionality that may impact user engagement or hinder the app's ability to meet its objectives.

**User Experience (UX):**
Assess the app's usability, intuitiveness, and overall user experience. Evaluate if the user interface is well-designed, visually appealing, and easy to navigate.
Consider user feedback and conduct usability testing to identify areas of improvement and address any pain points or issues faced by users.
Analyze how effectively the app promotes user engagement, facilitates connections, and encourages knowledge sharing among computer engineers.

**Performance:**
Evaluate the app's performance in terms of responsiveness, loading times, and overall speed. Consider factors such as network requests, data processing, and resource optimization.
Identify any performance bottlenecks or areas where the app may experience slowdowns, freezes, or crashes. Determine if the app can handle increased user traffic and data load.

**Security and Privacy:**
Assess the app's security measures and adherence to best practices for data protection and user privacy. Evaluate if sensitive user information is properly encrypted, and if secure authentication mechanisms are in place.
Analyze if the app follows industry standards for securing user data, and if there are measures to prevent unauthorized access or data breaches.

**Impact and User Engagement:**
Analyze the app's potential impact on the target audience of computer engineers and the broader computer engineering community.
Assess user engagement metrics such as active user base, frequency of app usage, and level of participation in knowledge sharing and networking activities.
Consider if the app effectively meets the needs of computer engineers and contributes to their professional growth, networking opportunities, and career development.

**Technical Considerations:**
Evaluate the app's compatibility with various Android devices, operating system versions, and screen sizes. Determine if it provides a consistent experience across different platforms.
Analyze the integration of external services or APIs, such as Firebase, and assess if they are utilized effectively to enhance app functionality and user experience.

**Future Scope and Enhancements:**
Identify potential areas for future improvement and enhancement based on user feedback, market trends, and evolving needs of computer engineers.
Consider additional features or functionalities that can be added to the app to enhance its value proposition and keep it relevant in the rapidly evolving field of computer engineering.

By conducting a thorough analysis of the ACE app in these areas, you can identify strengths, weaknesses, and opportunities for improvement. This analysis can inform decisions for further development, updates, and enhancements to ensure the app continues to meet the needs of its users effectively.

.

# CHAPTER 4: IMPLEMENTATION AND SCREENSHOTS

## 4.1 ADMIN APP

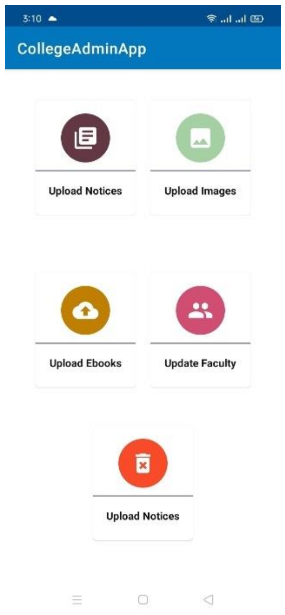


**Fig 4.1.1 Admin app Home Page**

This app contains all these functionalities as shown in the icon. These features are provided to the admin to control all the viewing of the users. These features and their functionalities are described as below:

**Upload Notices:**

The admin can upload any notice needed to be displayed onto the user app. This is done by using the database.This activity allows users to select an image from the gallery and upload it to Firebase Storage, storing its download URL in the Firebase Realtime Database. Here's an overview of the code:

**Initialization:** Various view elements and variables are initialized, including card view, spinner, button, image view, and Firebase references. ProgressDialog is used to show the upload progress.
**onCreate() Method:** Sets the layout for the activity and retrieves references to views. Sets up the spinner for selecting the image category.Sets click listeners for the "Add Image" card view and "Upload Notice" button.

**uploadImage() Method:** Compresses the selected image into a byte array. Creates a StorageReference for the image file with a unique name. Uploads the image to Firebase Storage using putBytes() method. Retrieves the download URL of the uploaded image using getDownloadUrl() method. Calls the uploadData() method to store the download URL in the database.

**uploadData() Method:** Creates a reference to the selected image category in the Firebase Realtime Database.Generates a unique key under the category to store the download URL.Sets the value of the unique key to the download URL.Displays success or failure messages using Toast.

**openGallery() Method**: Creates an intent to open the gallery and select an image.Starts the gallery activity using startActivityForResult().

**onActivityResult() Method:** Retrieves the selected image data from the intent result.Converts the image data into a Bitmap.Sets the Bitmap as the image view's source.

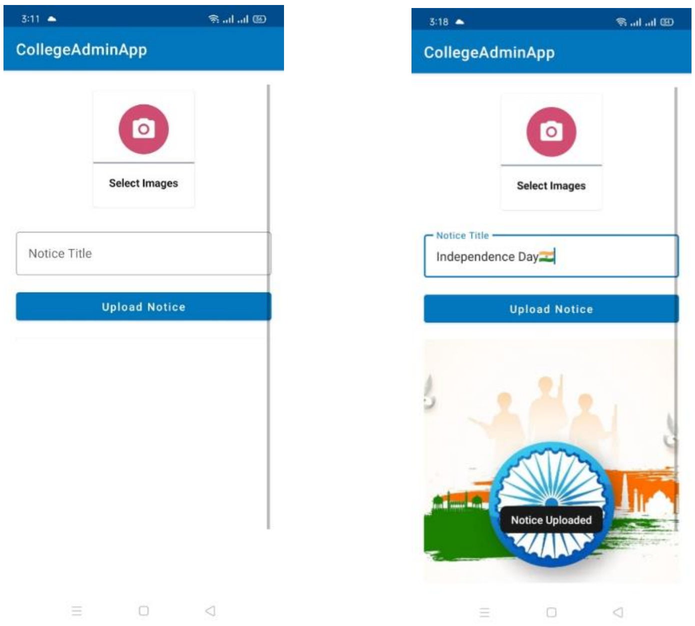


**Fig 4.1.2 Upload notice Page**

**Upload Images:**
The admin can upload any images of the events that have occurred that are needed to be displayed onto the user app. This is done by using the database. And a similar to the above functionality is performed here and the admin can upload the images to the firebase database.

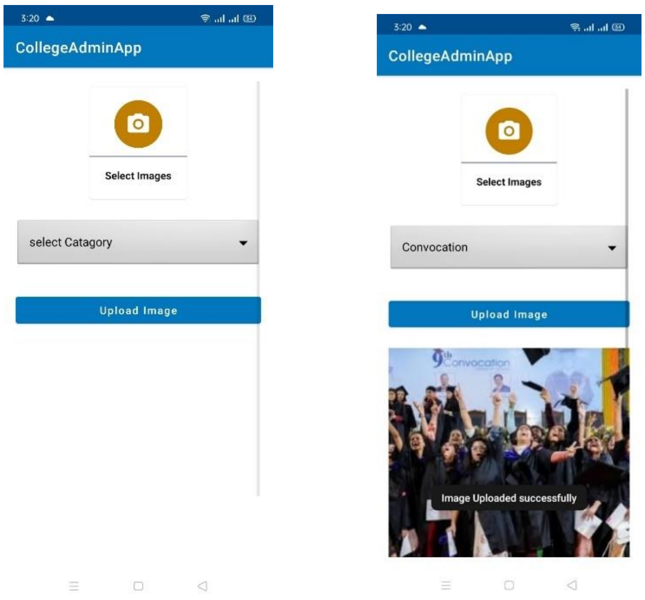
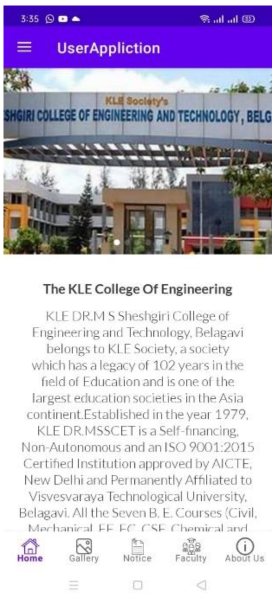


**Fig 4.1.3 Upload Images Page**

## 4.2USER APP

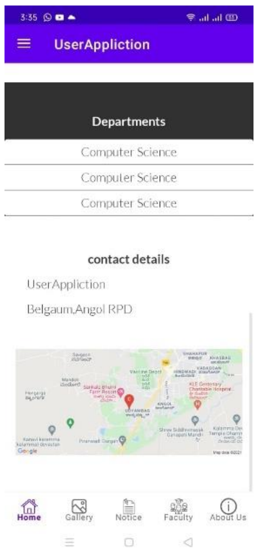


The KLE College Of Engineering

KLE DR.M S Sheshgiri College of Engineering and Technology, Belagavi belongs to KLE Society, a society which has a legacy of 102 years in the field of Education and is one of the largest education societies in the Asia continent.Established in the year 1979, KLE DR.MSSCET is a Self-financing, Non-Autonomous and an ISO 9001:2015 Certified Institution approved by AICTE, New Delhi and Permanently Affiliated to Visvesvaraya Technological University, Belagavi. All the Seven B. E. Courses (Civil, Mechanical, EE, EC, CSE, Chemical and

**Fig 4.2.1 User App Home Page**

The interface of the user app is given as below. Since the app is catored to our college KLE Dr MS Sheshgiri College of engineering and Technology, we have displayed the information of the college here. This is contained in the home page of the app.



Departments

Computer Science

Computer Science

Computer Science

contact details

UserAppliction

Belgaum,Angol RPD

**Fig 4.2.2 Google Map**

Here is some additional information about our college, when we click on the image we get the google map location of our college. This has been implemented by creating an intent to a url link as shown in the code below.

```
private void openMap() {
      Uri uri = Uri.parse("https://goo.gl/maps/Jf6up8ktkFFXT75w5 ");
      Intent intent = new Intent(Intent.ACTION_VIEW, uri);
      intent.setPackage("com.google.android.apps.maps");
      startActivity(intent);
   }
```

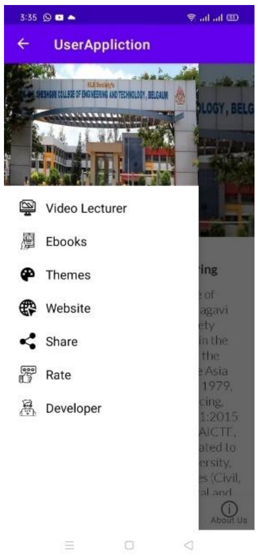


**Fig 4.2.3 Menu Bar**

We also have a side nav bar which will give you the page accordingly, where we have the intents to move to new pages. This side nav bar has pages like you can view Video lectures uploaded by the admin of the organization. And you can see eBook that are present in the database for the users to refer to for study purposes. The website for this app is given here also, so you can view a website if that's your preference. At last we added share, rate and developers information so help the developer.

The navbar is been made using fragments which have functionality of showing gallery, notices, the faculty and about us page. These fragments of the nav bar are meant to take the supplied data from the firebase database and store here accordingly. Here we'll see only one of these in detail that is the notice data here.
Here you can see the images that were stored in the database from the admin side of the app are seen here. Here Ill explain with the notice part of user app is an implementation of a RecyclerView adapter for displaying notice items. The adapter is responsible for inflating the layout for each item and binding the data to the corresponding views.

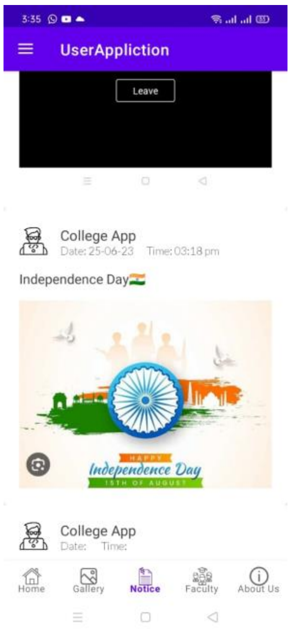


**Fig 4.2.4 Notice Fragment**

Here's a breakdown of the code for the notice fragment :

1. The `NoticeAdapter` class extends `RecyclerView.Adapter` and takes a context and a list of `NoticeData` objects as parameters in the constructor.

2. In the `onCreateViewHolder` method, the adapter inflates the layout for each item using the `LayoutInflater` and returns an instance of the `NoticeViewAdapter` class, which holds references to the views in the item layout.

3. The `onBindViewHolder` method is called for each item in the list. It retrieves the current `NoticeData` object based on the position and binds the data to the views in the `NoticeViewAdapter` holder.

4. Inside the `onBindViewHolder` method, the title, date, and time are set using the `setText` method on the corresponding TextViews in the holder.

5. If the `NoticeData` object has an image URL, Glide library is used to load the image into the `addNoticeImage` ImageView in the holder.

6. The `getItemCount` method returns the size of the list.

7. The `NoticeViewAdapter` class is a ViewHolder that holds references to the views in the item layout. In its constructor, the views are initialized using `findViewById`.

By using this adapter, you can populate a RecyclerView with a list of notice items, where each item displays the title, date, time, and an optional image.

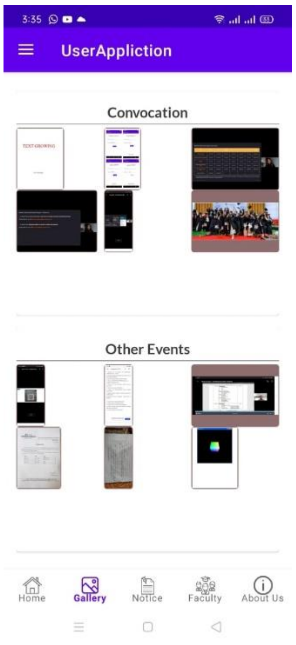


**Fig 4.2.4 Gallery Fragment**

A fragment called `NoticeFragment` that retrieves data from a Firebase Realtime Database and displays it in a RecyclerView.

Here's a breakdown of the code:

1. The `NoticeFragment` class extends `Fragment` and contains references to the RecyclerView (`addNoticeRecycler`) and ProgressBar (`progressBar`).

2. In the `onCreateView` method, the fragment's layout is inflated, and the RecyclerView and ProgressBar are initialized.

3. The Firebase Realtime Database reference is obtained using `FirebaseDatabase.getInstance().getReference().child("Notice")`.

4. The RecyclerView is configured with a LinearLayoutManager and set to have a fixed size.

5. The `getNotice` method is called to retrieve the data from the Firebase Realtime Database.

6. Inside the `getNotice` method, a `ValueEventListener` is added to the Firebase Database reference. This listener retrieves the data from the database and populates the `list` ArrayList with `NoticeData` objects.

7. The `NoticeAdapter` is instantiated with the `list` ArrayList and set as the adapter for the RecyclerView.

8. The adapter is notified of the data change using `adapter.notifyDataSetChanged()`.

9. The ProgressBar is hidden (`progressBar.setVisibility(View.GONE)`), and the RecyclerView is set with the adapter.

10. If there is an error while retrieving data from the database, the `onCancelled` method is called, and an error toast is displayed.

By using this `NoticeFragment`, you can display the notice data from the Firebase Realtime Database in a RecyclerView with the help of the `NoticeAdapter`.

## 4.3 DATABASE

To create a database in Firebase for your Android Studio app, follow these steps:

1**. Set up Firebase Project:**
   - Go to the Firebase console website (https://console.firebase.google.com/) and log in with your Google account.
   - Click on "Add project" or select an existing project to use.
   - Provide a name for your project and select your country/region.
   - Click "Create project" to create the project in Firebase.

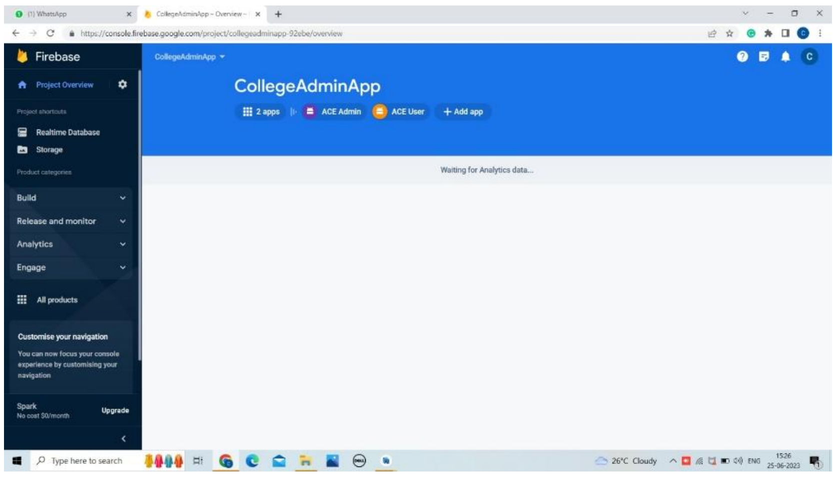


**Fig 4.3.1 Setup of project database**

2. **Connect Android Studio to Firebase:**
   - In the Firebase console, click on the Android icon to add your Android app to the project.
   - Enter your Android package name (e.g., com.example.myapp) and click "Register app".
      - Download the google-services.json file provided by Firebase.

3**. Add Firebase SDK to your Android Studio project:**
   - Open your Android Studio project.
   - In the Project view, navigate to the app-level build.gradle file (usually located in the "app" module).
   - Add the following dependencies to the dependencies section:

```
implementation 'com.google.firebase:firebase-database:20.0.0'
```

   - At the end of the file, add the following line to apply the plugin:

```
apply plugin: 'com.google.gms.google-services'
```

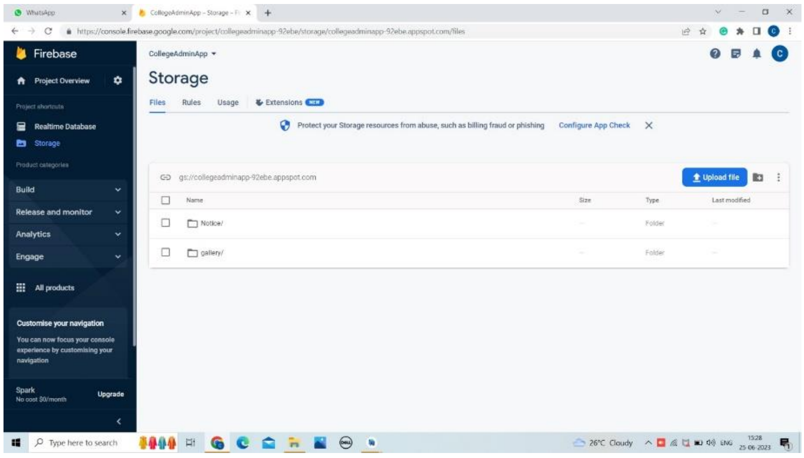
4**. Add the google-services.json file to your project:**
   - Copy the downloaded google-services.json file to the "app" directory of your Android Studio project.

5. **Sync Gradle and verify the connection:**
- Sync your project with Gradle files by clicking the "Sync Now" button or selecting "Sync Project with Gradle Files" from the "File" menu.
- Once the syncing is complete, the Firebase SDK will be added to your project.

6. **Access Firebase Realtime Database:**
- In your code, you can access the Firebase Realtime Database using the FirebaseDatabase class.
- Retrieve a reference to the database using the getInstance() method:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
```

- You can then create references to specific locations in the database and perform operations like reading, writing, and querying data.



Customise your navigation

You can now focus your console experience by customising your navigation

Protect your Storage resources from abuse, such as billing fraud or phishing

gs://collegeadminapp-92ebe.appspot.com

| Name | Size | Type | Last modified |
|---|---|---|---|
| Notice/ | — | Folder | — |
| gallery/ | — | Folder | — |

**Fig 4.3.2 Project database tables**

To connect two apps using Firebase to transfer images, you can follow these steps:

1. **Set up Firebase Projects:**
- Create Firebase projects for both apps by following the steps mentioned earlier in "Set up Firebase Project".
- Obtain the google-services.json files for both apps from the Firebase console.

2. **Configure Firebase Dependencies:**
- In each app's build.gradle file, add the following dependencies to enable Firebase services:

```
implementation 'com.google.firebase:firebase-storage:20.0.0'
implementation 'com.google.firebase:firebase-database:20.0.0'
```

3. **Add google-services.json files:**
- Copy the google-services.json file for each app to their respective "app" directories.

4. **Initialize Firebase in Apps:**
- In each app's main activity or application class, initialize Firebase by adding the following code:

```
FirebaseApp.initializeApp(this);
```

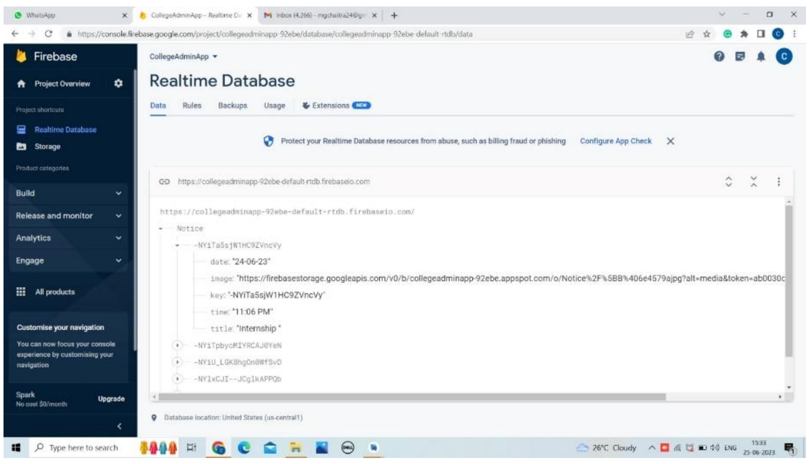
5. **Upload Image:**
- In the app that wants to send the image, use Firebase Storage to upload the image to the cloud storage. Here's an example:

```
StorageReference storageRef = FirebaseStorage.getInstance().getReference();
StorageReference imageRef = storageRef.child("images/image.jpg");

// Get the local file URI
Uri fileUri = Uri.fromFile(new File("path/to/local/image.jpg"));

// Upload the image file
UploadTask uploadTask = imageRef.putFile(fileUri);

// Monitor the upload progress and handle success/failure
uploadTask.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
        // Image upload successful
        // Retrieve the image URL and send it to the other app
        imageRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {
            @Override
            public void onSuccess(Uri downloadUri) {
                String imageUrl = downloadUri.toString();
                // Send the imageUrl to the other app
            }
        });
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        // Image upload failed
    }
});
```



Realtime Database

Protect your Realtime Database resources from abuse, such as billing fraud or phishing  Configure App Check

https://collegeadminapp-92ebe-default-rtdb.firebaseio.com

```
https://collegeadminapp-92ebe-default-rtdb.firebaseio.com/
- Notice
    - -NYiTa5sjW1HC9ZVncVy
        date: "24-06-23"
        image: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/Notice%2F%5BB%406e4579ajpg?alt=media&token=ab0030c
        key: "-NYiTa5sjW1HC9ZVncVy"
        time: "11:06 PM"
        title: "Internship "
    + -NYiTpbycMIYRCAJ0YeN
    + -NYiU_LGK8hgOn0WfSvD
    + -NYlxCJI--JCglkAPPQb
```

Database location: United States (us-central1)

Fig 4.3.3 Notice Table

6. **Receive Image:**

- In the app that wants to receive the image, you need to listen for the image URL sent from the other app. You can use Firebase Realtime Database for this purpose.
- When the sender app uploads the image, it can store the image URL in the Realtime Database under a specific node.
- In the receiver app, create a DatabaseReference to listen for changes in the image URL node and retrieve the URL when it changes.

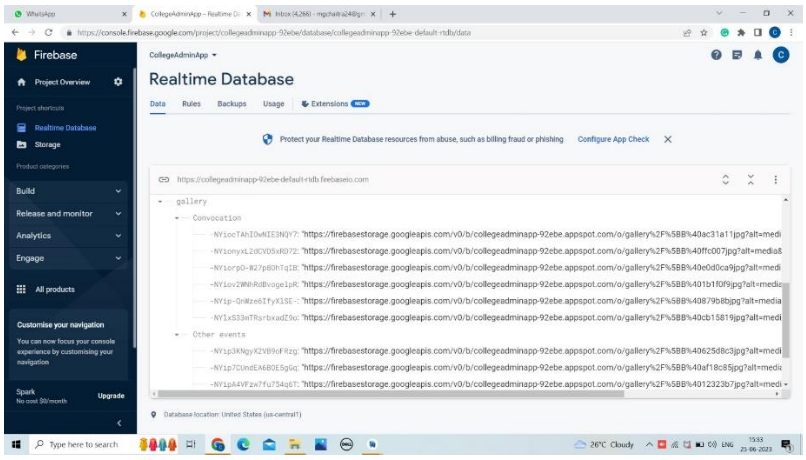
```
DatabaseReferenceimageRef= FirebaseDatabase.getInstance().getReference("imageURLNode");
imageRef.addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
      String imageUrl = dataSnapshot.getValue(String.class);
      // Use the imageUrl to load/display the image in your app
   }

   @Override
   public void onCancelled(@NonNull DatabaseError databaseError) {
      // Error occurred while retrieving the image URL
   }
});
```

7. **Sending Image URL:**

- In the sender app, after successfully uploading the image and getting the download URL, you can store the URL in the Realtime Database under the specific node that the receiver app is listening to.

```
DatabaseReference imageRef =
FirebaseDatabase.getInstance().getReference("imageURLNode");
imageRef.setValue(imageUrl);
```



```
https://collegeadminapp-92ebe-default-rtdb.firebaseio.com
gallery
  Convocation
    -NYiocTAhIOwNIE3NQY7: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%40ac31a11jpg?alt=medi
    -NYionyxL2dCVD5xRD72: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%40ffc007jpg?alt=media&
    -NYiorpO-W27p8OhTqIB: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%40e0d0ca9jpg?alt=medi
    -NYiov2WNhRdBvogelpR: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%401b1f0f9jpg?alt=media
    -NYip-QnWze6IfyX1SE-: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%40879b8bjpg?alt=media
    -NYixS33mTRsrbxadZ9o: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%40cb15819jpg?alt=medi
  Other events
    -NYip3KNgyX2VB9oFRzg: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%40625d8c3jpg?alt=medi
    -NYip7CUndEA6BOE5gGq: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%40af18c85jpg?alt=media
    -NYipA4VFzw7fu754q6T: "https://firebasestorage.googleapis.com/v0/b/collegeadminapp-92ebe.appspot.com/o/gallery%2F%5BB%4012323b7jpg?alt=medi
```

**Fig 4.3.4 Notice Table**

# CHAPTER 5: CONCLUSION AND FURTHER SCOPE

## 5.1 GENERAL CONSTRAINTS

While developing the project code mentioned above, here are some general constraints that may have been faced:
1. **Resource Constraints:** Developing a project code often involves managing limited resources such as memory, processing power, and storage. The code needs to be optimized and efficient to ensure it runs smoothly within the available resources.

2. **Platform Constraints:** The code may need to adhere to specific platform constraints or requirements. For example, if the code is intended for a particular operating system or hardware platform, it should be compatible and meet the platform's specifications and limitations.

3. **Performance Constraints:** Depending on the nature of the project, there may be performance constraints to consider. For instance, real-time applications like games often require the code to execute within strict time limits to ensure smooth gameplay and responsiveness.

4**. Compatibility Constraints:** The code may need to interact with existing systems, libraries, or APIs, requiring compatibility with specific versions, protocols, or formats. It may involve adhering to industry standards or conforming to specific integration requirements.

5. **Maintainability Constraints:** The code should be structured and documented in a way that ensures easy maintenance and future updates. It may involve following coding conventions, organizing code into modular components, and providing sufficient documentation for understanding and modifying the code in the future.

6**. Usability Constraints**: If the code includes a user interface or user interaction, usability constraints come into play. The code should be designed with user-friendly interfaces, intuitive controls, and appropriate feedback mechanisms, considering factors such as accessibility, responsiveness, and user experience.

7. **Security Constraints:** Depending on the project's nature and requirements, security constraints may be critical. The code may need to implement secure practices, such as input validation, data encryption, and access control, to protect against potential vulnerabilities or unauthorized access.

8. **Scalability Constraints:** If the code is expected to handle increasing amounts of data or user traffic, scalability constraints become important. The code should be designed in a way that allows for future growth and can handle higher loads without significant performance degradation.

9. **Legal and Compliance Constraints:** Depending on the project's context and industry, there may be legal and compliance constraints to consider. For example, the code may need to comply with data protection regulations, copyright laws, or industry-specific standards.

It's important to note that the specific constraints faced during the development of the project code may vary based on the project's requirements, scope, and context.

## 5.2 CONCLUSION

The ACE app is a comprehensive college administration app that allows users to perform various tasks related to college management. The app provides features such as student attendance tracking, fee management, noticeboard, and image gallery. It utilizes Firebase for backend services like authentication, database management, and storage.

The app consists of several activities and fragments that are designed to provide a user-friendly interface for different functionalities. For example, the Attendance activity allows teachers to mark attendance for students, while the Fee activity helps in managing student fee payments. The Noticeboard fragment displays important notices and updates for students and faculty members. The Image Gallery activity enables users to upload and view images related to different events and categories.

Firebase plays a crucial role in the app's functionality. It handles user authentication, ensuring secure access to the app's features. Firebase Realtime Database is used to store and retrieve data related to attendance, fees, notices, and images. Firebase Storage is utilized to store images uploaded by users in the image gallery. The app interacts with these Firebase services using the Firebase SDK and APIs.

Overall, the ACE app streamlines various administrative tasks for college management, enhancing efficiency and communication within the college community. It provides a centralized platform for teachers, students, and administrators to perform their respective roles effectively. With its intuitive interface and integration with Firebase, the ACE app offers a seamless and user-friendly experience for all users involved in college administration.

## REFERENCES:

Here are some references that you can use for further learning and development of the ACE app:

1. **Android Developer Documentation:** The official documentation provided by Google for Android development. It covers various topics, including activities, fragments, RecyclerView, Firebase integration, and more. You can find it at: https://developer.android.com/docs

2. **Firebase Documentation:** The official documentation for Firebase, which provides detailed information on integrating Firebase services into your Android app. It covers topics like Firebase Authentication, Realtime Database, Storage, and more. You can find it at: https://firebase.google.com/docs

3. **Android RecyclerView:** This guide by Vogella provides an in-depth explanation of how to use RecyclerView to display lists or grids of data in your Android app. It covers topics like creating RecyclerView adapters, handling item clicks, and optimizing performance. You can find it at: https://www.vogella.com/tutorials/AndroidRecyclerView/article.html

4. **Android CardView:** The official documentation for the CardView widget, which allows you to display information in a card-like format. It provides details on how to use CardView in your app and customize its appearance. You can find it at: https://developer.android.com/guide/topics/ui/layout/cardview

5. **Glide Image Loading Library**: Glide is a popular image loading and caching library for Android. It provides a simple and efficient way to load and display images from various sources, including Firebase Storage. You can learn more about Glide and its usage in the official documentation: https://bumptech.github.io/glide/

6. **Android UI Design Guidelines**: This resource provides guidelines and best practices for designing user interfaces in Android apps. It covers topics like layouts, typography, colors, icons, and more. Following these guidelines can help you create a visually appealing and user-friendly app. You can find the official design guidelines at: https://developer.android.com/design