GOVERNMENT POLYTECHNIC, AURANGABAD

(An Autonomous Institute of Government of Maharashtra)

DEPARTMENT OF COMPUTER ENGINEERING

"PURSUIT FOR EXCELLENCE"

PROJECT REPORT
ON

"Visiting Faculty Billing Application"

SUBMITTED
BY

Deshmukh Harshad Laxmikant (206008)
Holkar Adesh Shivaji(206012)
Patil Bhavesh Yogesh(206021)

GUIDED
BY

Mr.V.B.Patil
(Lecturer in Computer Engineering)

ACADEMIC YEAR 2022-23
DEPARTMENT OF COMPUTER ENGINEERING

GOVERNMENT POLYTECHNIC, AURANGABAD

(An Autonomous Institute of Government of Maharashtra)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that

1. **Deshmukh Harshad Laxmikant (206007)**
2. **Holkar Adesh Shivaji (206012)**
3. **Patil Bhavesh Yogesh (206021)**

have successfully completed project work titled " **Android visitor billing application"** during the academic year 2022-2023, in partial fulfillment of the **Diploma in Computer Engineering** of Government Polytechnic, Aurangabad. To the best of our knowledge and belief, this project work has not been submitted elsewhere.

Date:

Mr.V.B.Patil
(Lecturer in Computer Engg.)
Project Guide

Mrs. S. S. Jaiswal
(H.O.D CO)

Mr. A. M. Jinturkar
(Principal)

ACKNOWLEDGEMENT

We take an immense pleasure in thinking **Mr. A. M. Jinturkar**, the principal, Government Polytechnic, Aurangabad, our source of inspiration. We wish to express our deep sense of gratitude to **Mrs. S. S. Jaiswal,** our respected Head of Department, Computer Engineering, and **Mr.V.B.Patil,** our guide for having permitted us to carry out this project under his valuable guidance and useful suggestions, which have helped us in completing the project in time.

I would also like to thank to all our faculty members of our department for their valuable suggestion in the process of this project work. Finally, yet importantly, I would like to express our thanks to our beloved parents for their blessings.

Last but never the least, our friends and classmates, they have helped and co-operated a lot for the successful completion of this project.

1. Deshmukh Harshad Laxmikant (206008)
2. Holkar Adesh Shivaji (206012)
3. Patil Bhavesh Yogesh (206021)

CONTENTS

List of Figures

ABSTRACT

The Android Visitor Billing Application is developed as a final year project to provide visitors in a college with an efficient and user-friendly way of managing their payments and expenses. The application is developed using Android Studio and Java programming language.

The application allows visitors to register their account and make payments for various service. The application also provides a search functionality for visitors to filter and view their transaction history report. The application uses a secure payment gateway for online transactions and provides real-time notifications for pending payments.The Android Visitor Billing Application provides visitors with a seamless and hassle-free payment experience. The application is designed to be scalable and can be easily customized to meet the needs of different colleges and universities.

The project demonstrates the ability to design and develop a complex Android application that meets the requirements of the users. The project also showcases the importance of using modern technologies and development practices to create high-quality applications.

Overall, the Android Visitor Billing Application provides a valuable solution for visitors in a college to manage their payments and expenses in a convenient and secure way.

1. INTRODUCTION

The Android Visitor Billing Application is a personal project developed to manage payments for visitors attending theory and practical lectures at a college. The application is designed to provide visitors with a convenient and secure way to make payments for lectures attended and interact with the Head of Department (HOD) for payment-related queries.

The application allows visitors to create an account and view their payment history for theory and practical lectures separately. The application also allows visitors to pay for multiple lectures at once, reducing the number of transactions required.

The Android Visitor Billing Application also includes a feature for visitors to interact with the HOD directly. Visitors can send payment-related queries to the HoD and receive a response in real-time. This feature eliminates the need for visitors to physically visit the HoD's office to clarify payment-related queries. The application uses a secure payment gateway for online transactions, ensuring the safety of visitors' financial information. The application also provides real-time notifications for pending payments and reminders for upcoming lectures.

The main objective of this project is to provide a convenient and efficient payment system for visitors attending lectures at a college. This project aims to showcase the importance of using modern technologies and development practices to create high-quality Android applications that meet the needs of users.

This report covers various aspects of the application, such as its features, technologies used, user interface, application architecture, implementation details, testing and validation, and results and conclusions. The report is intended for visitors attending lectures at a college and the HOD, who will benefit from the convenience and efficiency of the Android Visitor Billing Application.

Overall, the Android Visitor Billing Application provides visitors with a hassle-free payment experience for attending lectures, eliminating the need for manual payment methods and physical visits to the HOD's office. This project demonstrates the ability to design and develop a complex Android application that meets the requirements of users and highlights the importance of technology in simplifying and streamlining payment systems.

1.1 Android Programming Language

Android is the most popular mobile operating system in the world, making it a great platform for reaching a large audience. Java is the primary programming language used for Android app development, and it has a large community of developers, making it easier to find resources and support when developing an Android app.

Android provides a range of features and tools specifically designed for mobile app development, such as the ability to access device hardware and sensors, which can be useful for developing apps that interact with the user's environment. Java is a highly versatile programming language that can be used for developing a wide range of applications, making it a good choice for developing a payment application like the Visitor Billing Application.

Android provides built-in security features such as sandboxing and permission-based access to user data, which are important for protecting sensitive user data like payment information. Java and Android provide a high degree of customization, allowing you to tailor your development environment to your specific needs and preferences.

In summary, using Android and Java for your Android Visitor Billing Application ensures that your app is accessible, secure, and highly customizable, making it the ideal choice for your personal project.

1.2Firebase Database

Firebase is a product of Google that helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and more securely. No programming is required on the Firebase side, making it easy to use its features more efficiently. It provides services to Android, iOS, web, and Unity. It provides cloud storage. It uses NoSQL for the database for the storage of data.

In simple words, Firebase is a software development platform that helps in building web and mobile applications with its 18 services. These 18 services of this BaaS solution also include purposeful APIs and four beta products. In addition, it is compatible to integrate with Android, web, iOS, and Unity setups.

Tech and commerce giants which are using the cloud and BaaS services of Firebase are Alibaba Travels, Stack, Twitch, and Instacart.

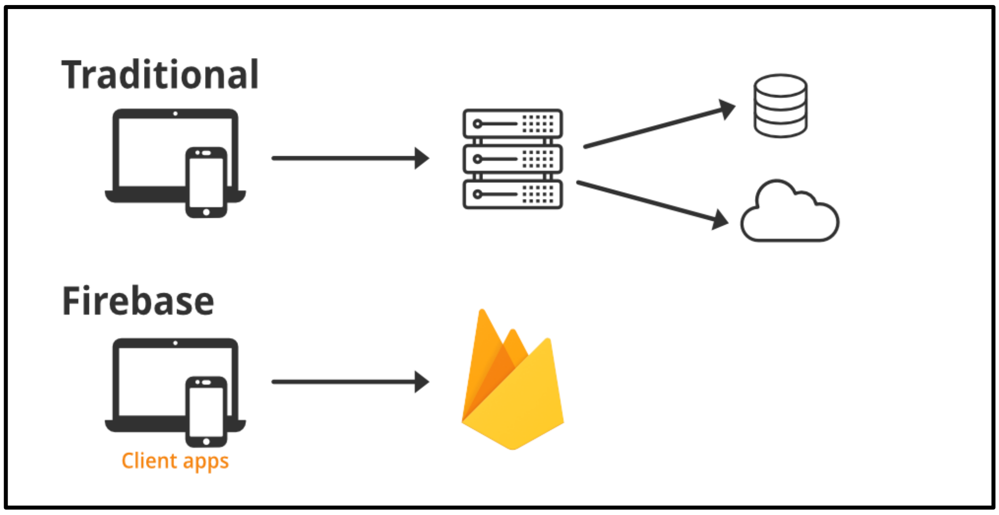


Fig. 1.1

Brief History of Firebase:

Firebase was initially an online chat service provider to various websites through API and ran with Envolve. It got popular as developers used it to exchange application data like a game state in real-time across their users more than the chats. This resulted in the separation of the Envolve architecture and its chat system. The Envolve architecture was further evolved by its founders James Tamplin and Andrew Lee, to what modern-day Firebase is in the year 2012.

Features of Firebase:

Mainly there are 3 categories in which Firebase provides its services.

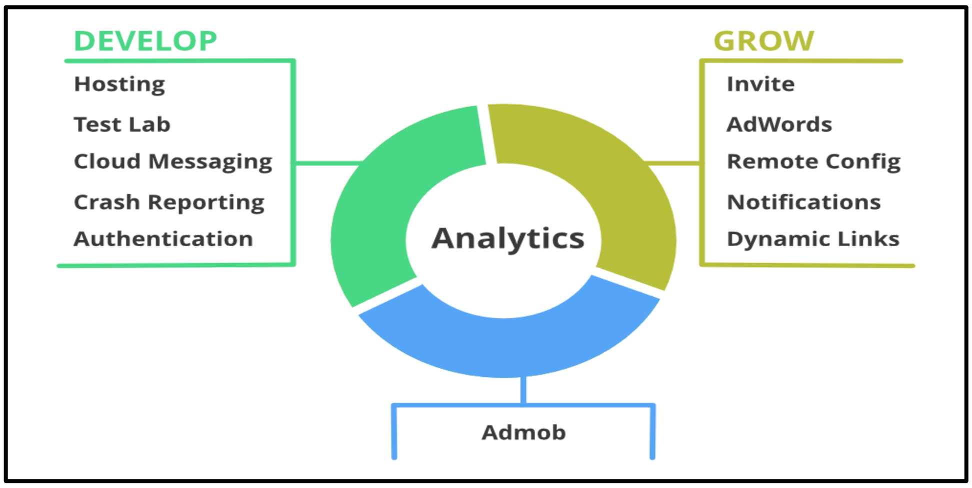


Fig. 1.2

Build better applications

This feature mainly includes backend services that help developers build and manage their applications better. Services included under this feature are :

Realtime Database: The Firebase Realtime Database is a cloud-based NoSQL database that manages your data at the blazing speed of milliseconds. In simplest terms, it can be considered as a big JSON file.

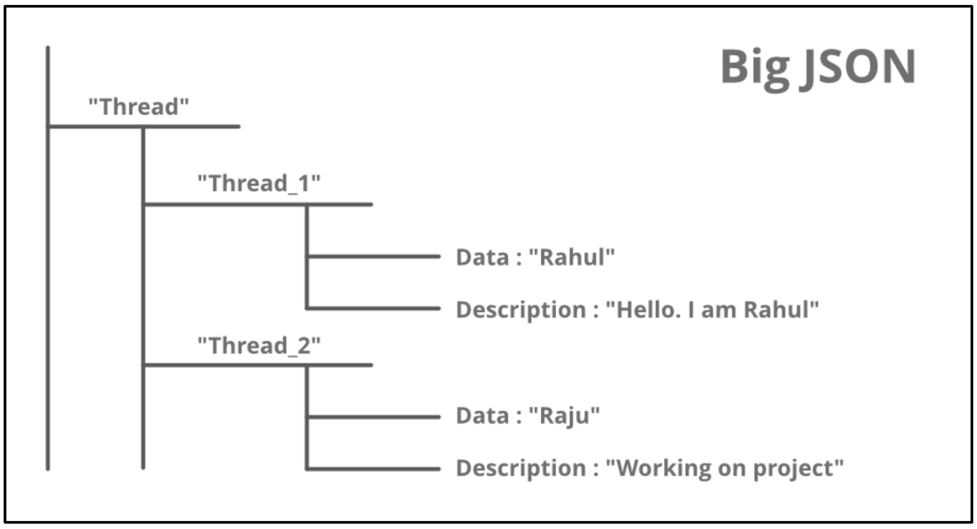

Big JSON
"Thread"
"Thread_1"
Data : "Rahul"
Description : "Hello. I am Rahul"
"Thread_2"
Data : "Raju"
Description : "Working on project"

Fig. 1.3

The Firebase Realtime Database is cloud-hosted. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our Apple platforms, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

Key capabilities

Realtime	Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.
Offline	Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is re-established, the client device receives any changes it missed, synchronizing it with the current server state.
Accessible from Client Devices	The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.
Scale across multiple databases	With Firebase Realtime Database on the Blaze pricing plan, you can support your app's data needs at scale by splitting your data across multiple database instances in the same Firebase project. Streamline authentication with Firebase Authentication on your project and authenticate users across your database instances. Control access to the data in each database with custom Firebase Realtime Database Security Rules for each database instance.

How does it work?

The Firebase Realtime Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, realtime events continue to fire, giving the end user a responsive experience. When the device regains connection, the Realtime Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Realtime Database provides a flexible, expression-based rules language, called Firebase Realtime Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Realtime Database is a NoSQL database and as such has different optimizations and functionality compared to a relational database. The Realtime Database API is designed to only allow operations that can be executed quickly. This enables you to build a great realtime experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

Connect your App to Firebase

Create a Database

1. Navigate to the Realtime Database section of the Firebase console. You'll be prompted to select an existing Firebase project. Follow the database creation workflow.

2. Select a starting mode for your Firebase Security Rules:

Test mode

Good for getting started with the mobile and web client libraries, but allows anyone to read and overwrite your data. After testing, make sure to review the Understand Firebase Realtime Database Rules section.

Note: If you create a database in Test mode and make no changes to the default world-readable and world-writeable Rules within a trial period, you will be alerted by email, then your database rules will deny all requests. Note the expiration date during the Firebase console setup flow.

To get started with the web, Apple, or Android SDK, select test mode.

Locked mode

Denies all reads and writes from mobile and web clients. Your authenticated application servers can still access your database.

3. Choose a location for the database.

Depending on the location of the database, the URL for the new database will be in one of the following forms:

- *DATABASE_NAME*.firebaseio.com (for databases in us-central1)
- *DATABASE_NAME.REGION*.firebasedatabase.app (for databases in all other locations)

4. Click Done.

When you enable Realtime Database, it also enables the API in the Cloud API Manager.

Cloud Firestore:

The cloud Firestore is a NoSQL document database that provides services like storing, syncing, and querying through the application on a global scale. It stores data in the form of objects also known as Documents. It has a key-value pair and can store all kinds of data like strings, binary data, and even JSON trees.

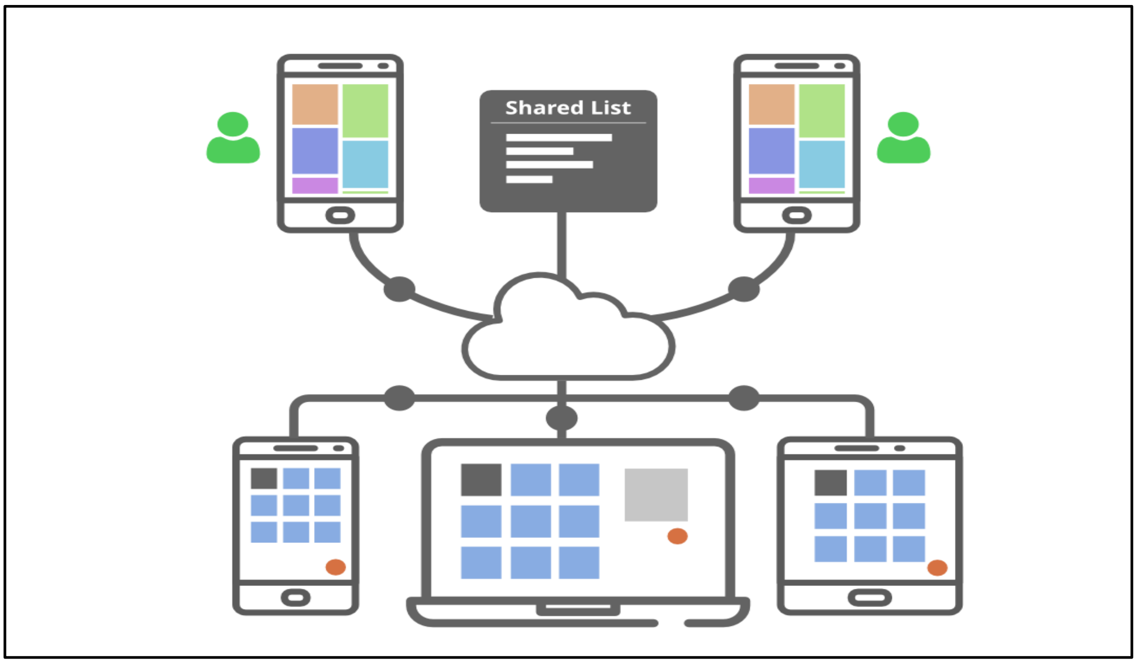


Fig.1.4

Authentication: Firebase Authentication service provides easy-to-use UI libraries and SDKs to authenticate users to your app. It reduces the manpower and effort required to develop and maintain the user authentication service. It even handles tasks like merging accounts, which if done manually can be hectic.

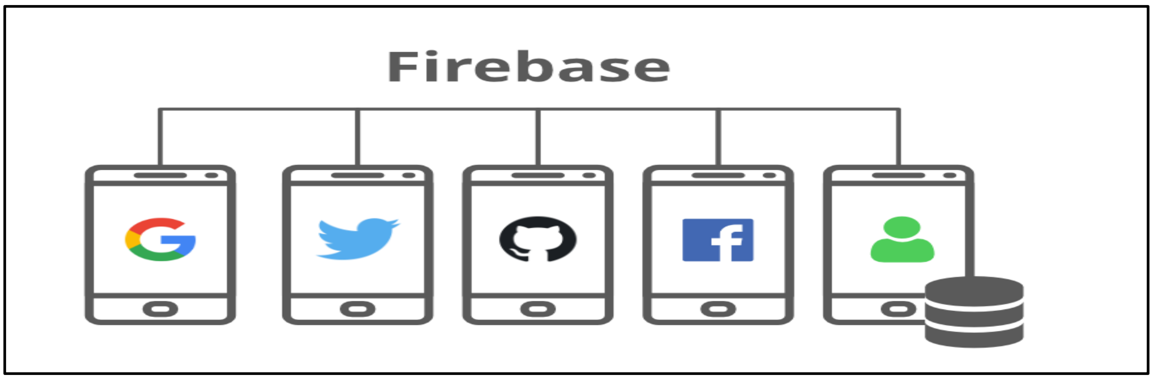


Fig. 1.5

1.3 Android studio application :

Android Studio is the official integrated development environment (IDE) for Android app development, and there are several reasons why you would use it for developing an Android Visitor Billing Application for your personal project:

1) Built-in Android emulator: Android Studio comes with a built-in emulator that allows you to test your app on a range of virtual devices, making it easier to develop and test your app without needing access to physical Android devices.
2) Easy to use user interface: Android Studio has a user-friendly interface that makes it easy to navigate and use, even for beginner developers.
3) Comprehensive tools: Android Studio provides a range of tools and features that simplify the development process, including code completion, debugging, and layout editors.
4) Integration with other Google services: Android Studio is integrated with other Google services, such as Firebase and Google Cloud Platform, making it easier to incorporate these services into your app.
5) Support for multiple programming languages: Android Studio supports multiple programming languages, including Java, Kotlin, and C++, giving you the flexibility to choose the language that best suits your needs.
6) Active community support: Android Studio has an active and supportive community of developers, making it easier to find answers to your questions and get help when you need it.
7) to authenticate users to your app. It reduces the manpower and effort required to develop and maintain the user authentication service. It even handles tasks like merging accounts, which if done manually can be hectic.

1.4 Lottiefiles animation

Lottie is a mobile library for Android and iOS that parses Adobe After Effects animations exported as json with Bodymovin and renders them natively on mobile!

LottieFiles is a library and platform that allows developers to easily integrate and use Lottie animations in their Android applications. Lottie is an open-source animation file format developed by Airbnb, and it enables the creation of complex animations that can be rendered in real-time.

It uses JSON files to describe the animation data, including keyframes, shapes, and timings.

In Android Studio, you can incorporate Lottie animations using the LottieFiles library, which provides a convenient way to add, play, and control Lottie animations within your app.

Here's a step-by-step explanation of how to use LottieFiles in Android Studio:

1. Add the LottieFiles dependency: Open your app-level **build.gradle** file and add the following line in the dependencies block:

```
implementation 'com.airbnb.android:lottie:4.1.0'
```

2. Download Lottie animation files: Lottie animations can be created using various tools such as Adobe After Effects, Haiku Animator, or the official Lottie web editor (https://lottiefiles.com/). Download the Lottie animation file (in **JSON** format) that you want to use in your Android app.

3. Place the Lottie animation file in your Android project: Copy the downloaded **JSON file** into the **assets** or **res/raw** directory of your Android project. If the directory doesn't exist, create it.

4. Add the Lottie animation view to your layout: In your XML layout file, add the following code to include the Lottie animation view:

```
LottieAnimationView animationView = findViewById(R.id.animation_view);

animationView.setAnimation("your_animation.json");

animationView.playAnimation();
```

5. Load and control the animation in your Java code: In your Java code, you can load the animation and control it using the LottieAnimationView.
For example:

```
<com.airbnb.lottie.LottieAnimationView

    android:id="@+id/animation_view"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    app:lottie_autoPlay="true"

    app:lottie_fileName="your_animation.json"   />
```

2. REQUIREMENTS

2.1 Software Requirement Specifications

1) Operating System Front End Back End Server Documentation :

Windows 10

2) Frontend Software:

• XML: The main benefit of xml is that you can use it to take data from a program like Microsoft SQL, convert it into XML then share that XML with other programs and platforms. You can communicate between two platforms which are generally very difficult.The main thing which makes XML truly powerful is its international acceptance. Many corporation use XML interfaces for databases, programming, office application mobile phones and more. It is due to its platform independent feature.

• Material UI: Material UI is an open-source, front-end framework for React components that has 60,500 plus stars on github. It is built using Less. Less (stands for Leaner Style Sheets), is a backwardcompatible language extension for CSS. Material UI is based on Google's Material Design to provide a high-quality, digital experience while developing front-end graphics. Material Design focuses on providing bold and crisp designs – it builds textures by focusing on how the components cast shadows and reflect light.

• Lottie Animation: A Lottie is a JSON-based animation file format that you can use on any platform as easily as static assets. They are small files that work on any device and can scale up or down without pixelation. And the best part — Lottie animations don't require any coding knowledge!

• Glide: It is an Image Loader Library for Android developed by bumptech and is a library that is recommended by Google. It has been used in many Google open source projects including Google I/O 2014 official application. It provides animated GIF support and handles image loading/caching.

2.2 Hardware Requirement Specifications

1. Operating System: The minimum requirement would be Android 5.0 (Lollipop) or higher. However, it's advisable to target a more recent version to ensure compatibility with a wider range of devices.

2. Processor: Most modern Android devices are equipped with processors that can handle typical app requirements. However, it's recommended to target devices with at least a quad-core processor for smoother performance.

3. RAM: The amount of RAM required depends on the complexity of your application. As a general guideline, aim for a minimum of 2GB of RAM to ensure acceptable performance. However, if your app involves resource-intensive tasks or complex calculations, consider targeting devices with higher RAM, such as 3GB or 4GB.

4. Storage: The storage requirement for your visitor billing application largely depends on the size of the app itself, as well as any additional data it may store locally. Ensure that your app is optimized in terms of storage usage, as users might have devices with limited storage capacity. Aim for an app size below 100MB, and provide options for users to clear cached data or move the app to an SD card if necessary.

5. Display: Design your application to be compatible with a variety of screen sizes and resolutions. Ensure that the user interface (UI) elements are responsive and adaptable to different screen sizes, ranging from small smartphones to larger tablets.

6. Connectivity: Your visitor billing application might require internet connectivity to sync data, retrieve visitor information, or communicate with a server. Consider the availability of cellular data or Wi-Fi connections and design your app to handle different connectivity scenarios gracefully.

2.3 Functional Requirement

2.3.1 Performance Requirement

1.Scalability:

The application should be able to handle a growing number of users, lecture records, and requests without significant degradation in performance. It should be designed to accommodate potential future increases in data volume and user traffic.

2.Efficient Data Processing:

The salary calculation and request approval processes should be optimized to minimize processing time and resource utilization.Algorithms and data structures should be efficient to handle large datasets without causing performance bottlenecks.

3.Caching and Optimization:

Utilize caching mechanisms to store frequently accessed data, such as user profiles and commonly used lecture details, to reduce database queries and improve response time.Apply optimization techniques, such as query optimization and code optimization, to enhance overall system performance.

4.Network Efficiency:

Minimize network bandwidth consumption by optimizing data transfer protocols and compressing data where applicable.Use asynchronous communication and background tasks to avoid blocking the user interface and provide a smooth user experience.

2.3.2 Interface Requirement

1. Request Approval/Rejection:

Design a user interface for department heads or authorized personnel to review and manage the submitted requests. Provide clear options for approving or rejecting requests, along with the ability to add comments or feedback if necessary.

2. Reporting and Analytics:

Create an interface to generate reports and analytics based on different parameters, such as date range, visitor, or department. Present the reports in a visually appealing and easily understandable format, such as graphs, tables, or charts.

3. Admin Dashboard:

Design an intuitive dashboard for administrators to manage user accounts, roles, and permissions. Include features to view and manage all requests, adjust salaries if needed, and perform administrative tasks efficiently.Ensure that the application's user interface is responsive and adapts well to different screen sizes and orientations.

2.3.3 Operational Requirement

1.Platform Compatibility:

The application should be compatible with a range of Android devices and versions, ensuring broad accessibility for users.Specify the minimum supported Android version and device requirements.

2.Installation and Deployment:

Provide clear instructions for users to install and set up the application on their Android devices. Ensure that the installation process is straightforward and does not require complex configurations.

3.User Support and Documentation:

Develop comprehensive documentation that includes user guides and FAQs to assist users in understanding and using the application.Provide contact information or a support system for users to seek assistance or report issues.

4.System Performance Monitoring:

Implement mechanisms to monitor the performance of the application, such as server response times, database performance, and resource utilization.Consider integrating monitoring tools or logging frameworks to track system health and identify potential bottlenecks

2.3.4 Mobile Specification

PLATFORM:

OS Android 10, upgradable to Android 11, MIUI 12.5 E

CPU:Octa-core (2x2.0 GHz Cortex-A75 & 6x1.8 GHz Cortex-A55)

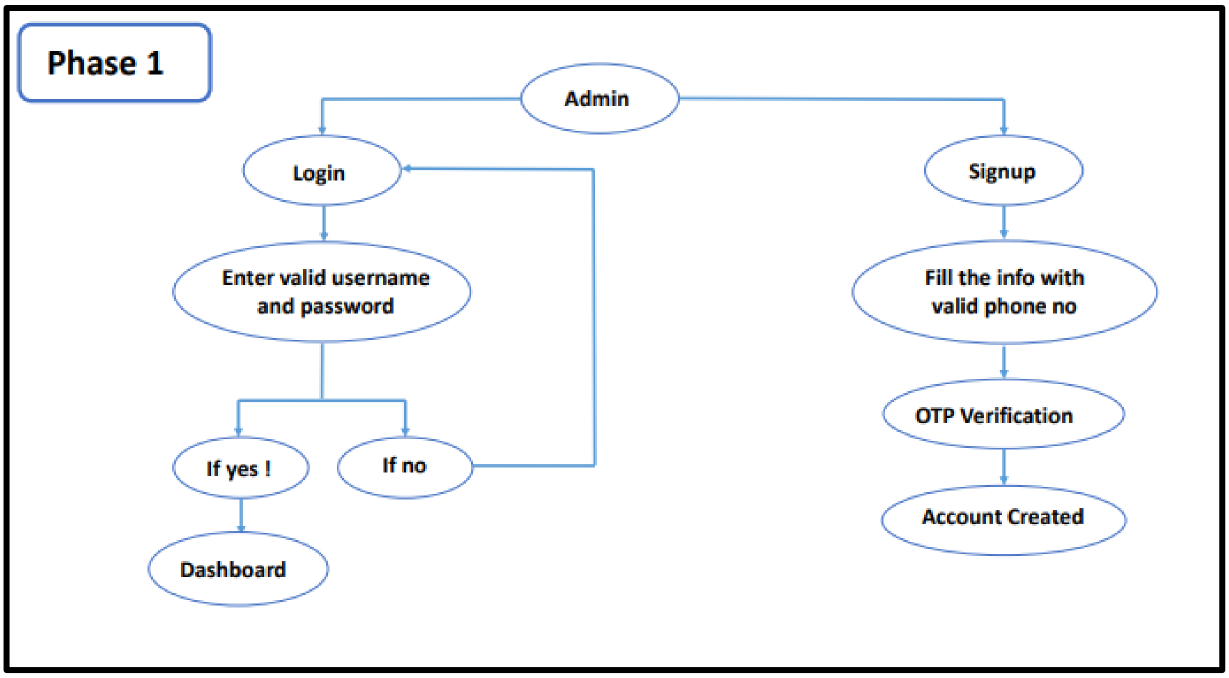
MEMORY:

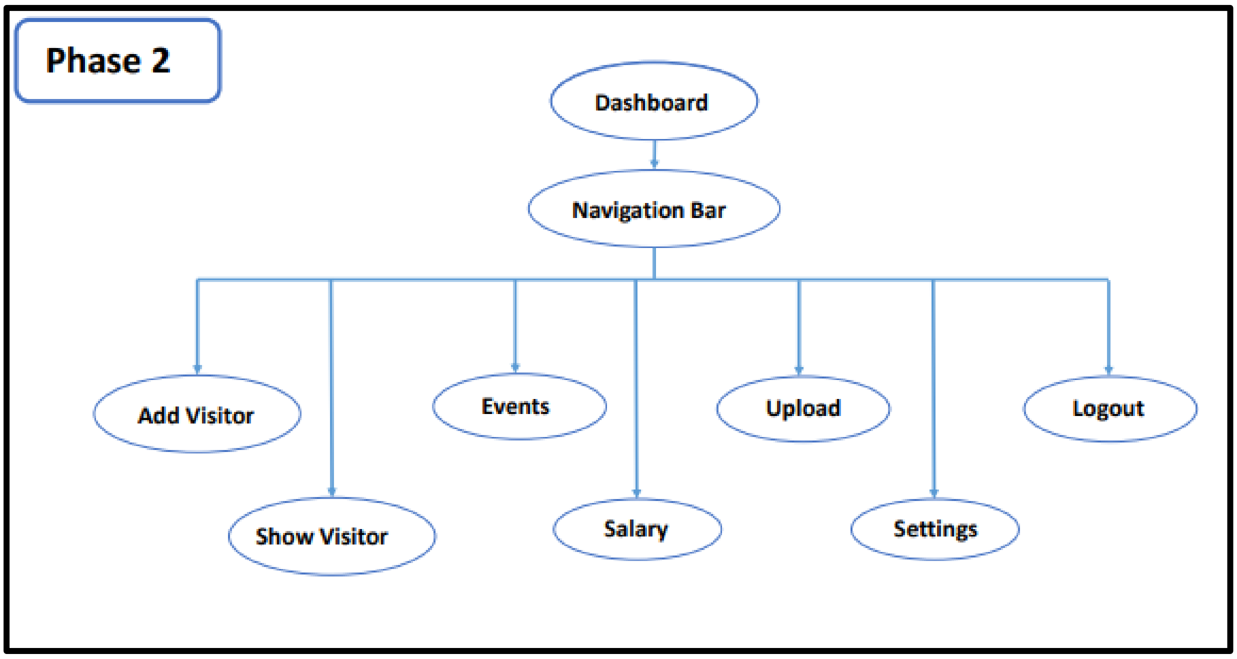
Card slot: microSDXC (dedicated slot)

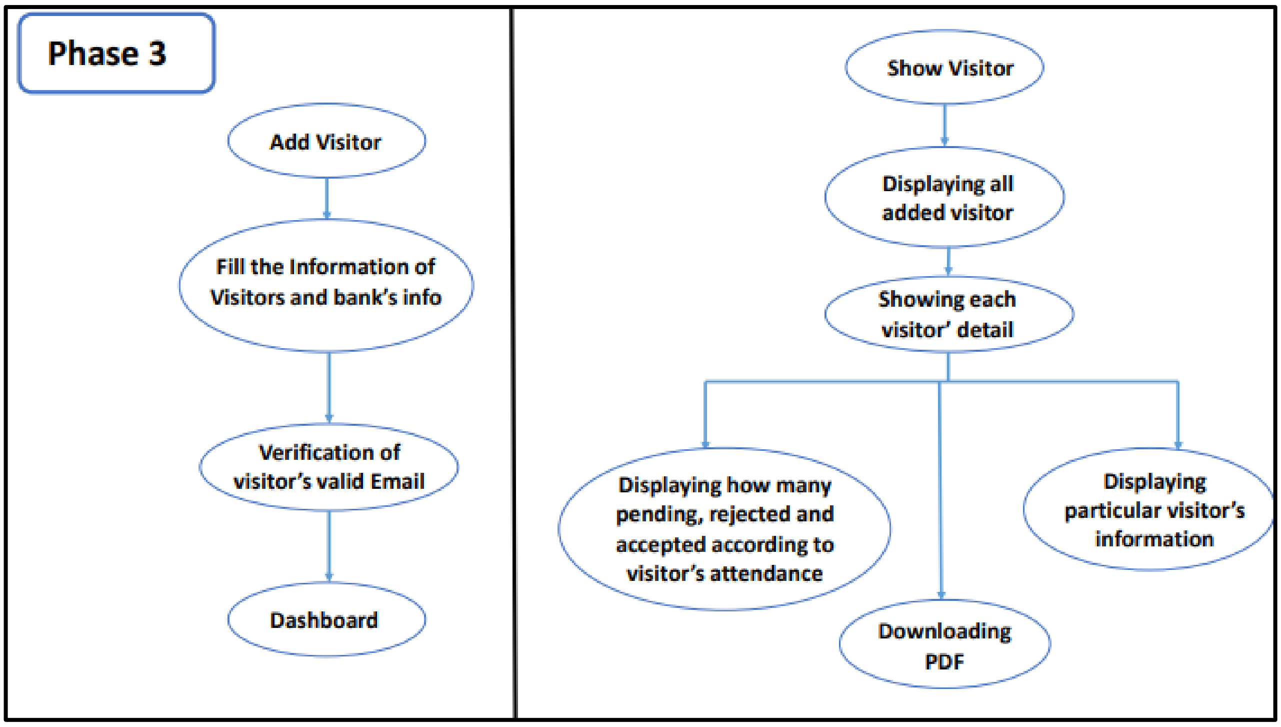
Internal: 4GB RAM, 128 GB

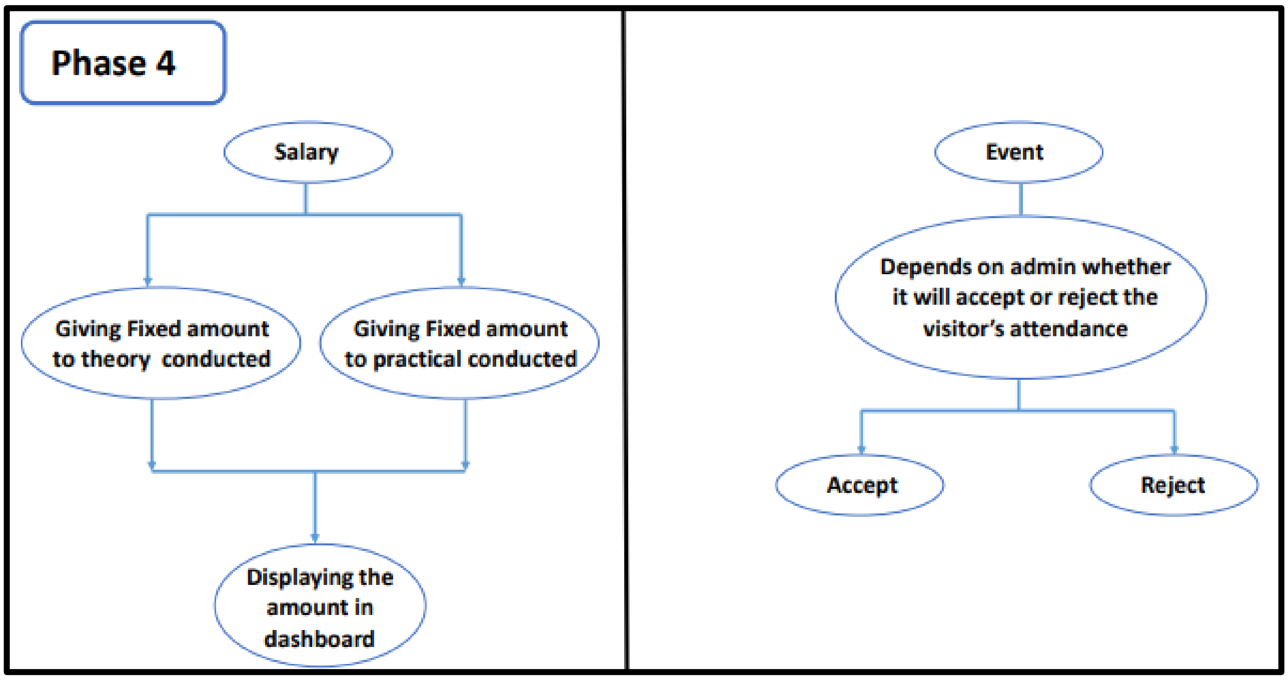
3. METHODOLOGY

3.1 Admin Application(HOD)

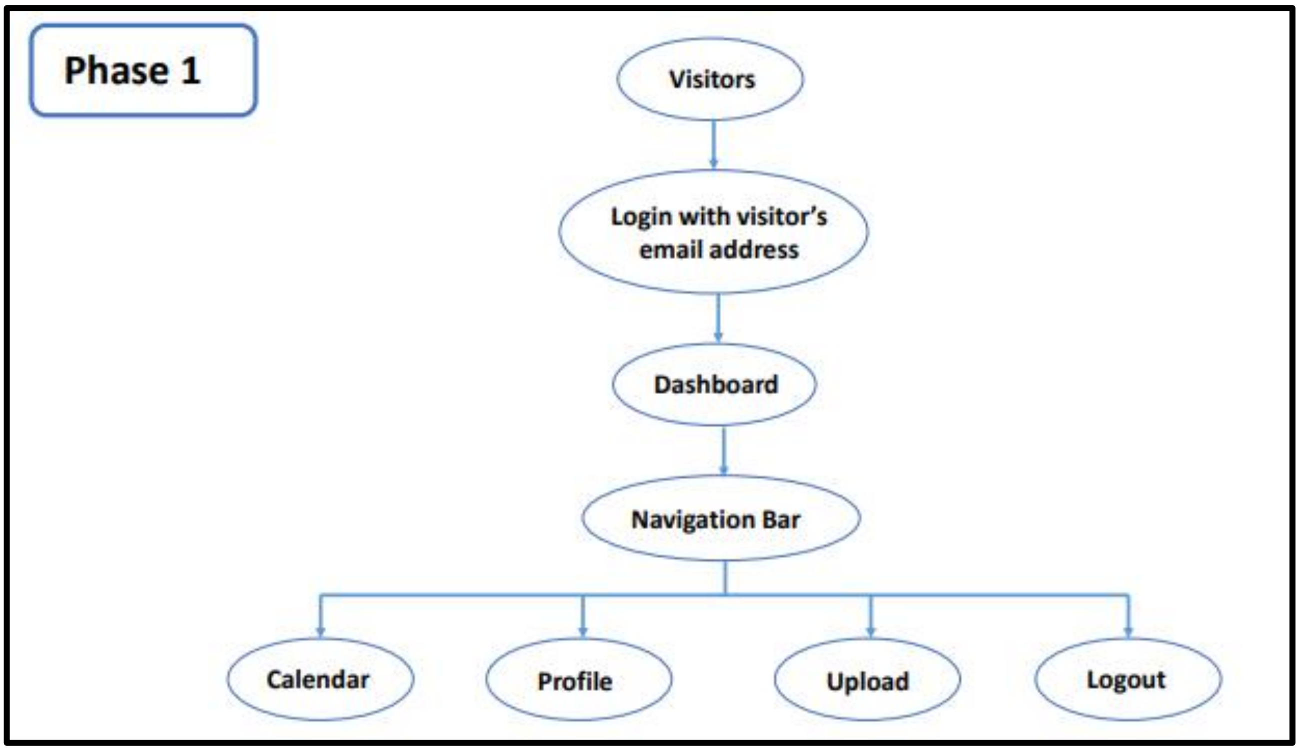


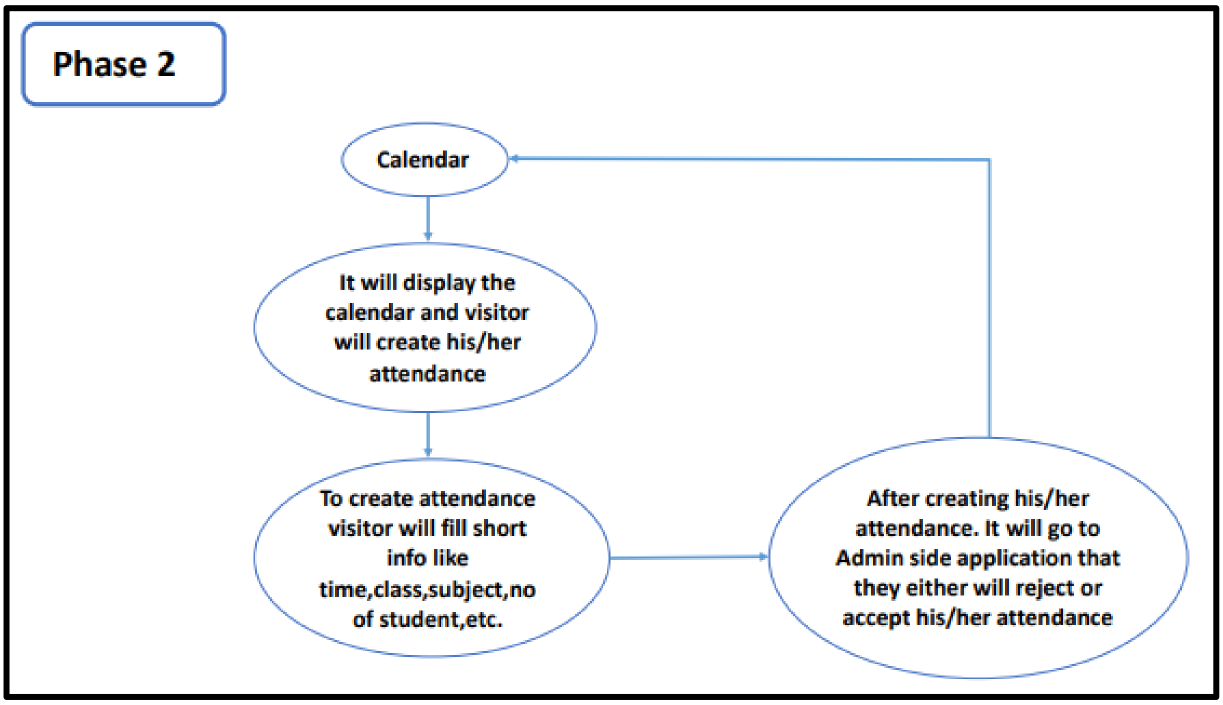



Phase 3
Add Visitor
Fill the Information of Visitors and bank's info
Verification of visitor's valid Email
Dashboard
Show Visitor
Displaying all added visitor
Showing each visitor' detail
Displaying how many pending, rejected and accepted according to visitor's attendance
Downloading PDF
Displaying particular visitor's information


Phase 4
Salary
Giving Fixed amount to theory conducted
Giving Fixed amount to practical conducted
Displaying the amount in dashboard
Event
Depends on admin whether it will accept or reject the visitor's attendance
Accept
Reject

3.2 Visiting Faculty Application




Phase 2
Calendar
It will display the calendar and visitor will create his/her attendance
To create attendance visitor will fill short info like time,class,subject,no of student,etc.
After creating his/her attendance. It will go to Admin side application that they either will reject or accept his/her attendance

4. IMPLEMENTATION

4.1 Admin Application – VisitorFunds

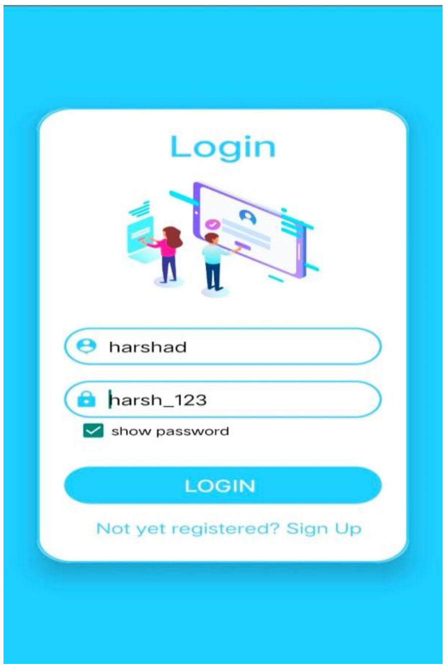


Fig 4.1-(1) Login Page

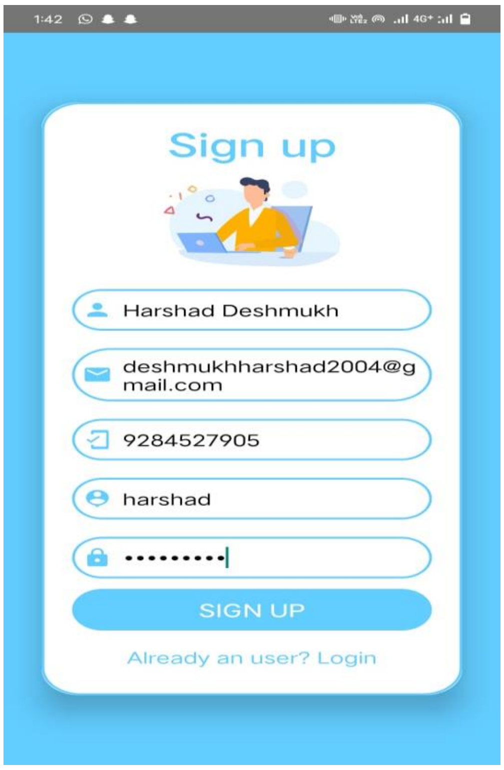


Fig 4.1-(2) Sign Up Page

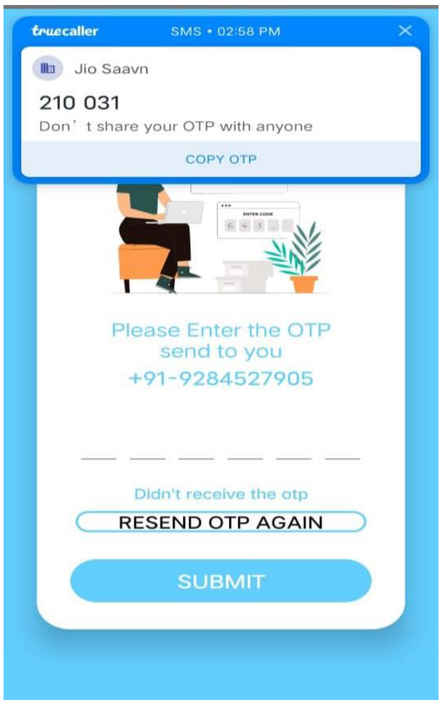


Fig 4.1-(3) OTP Verification Page

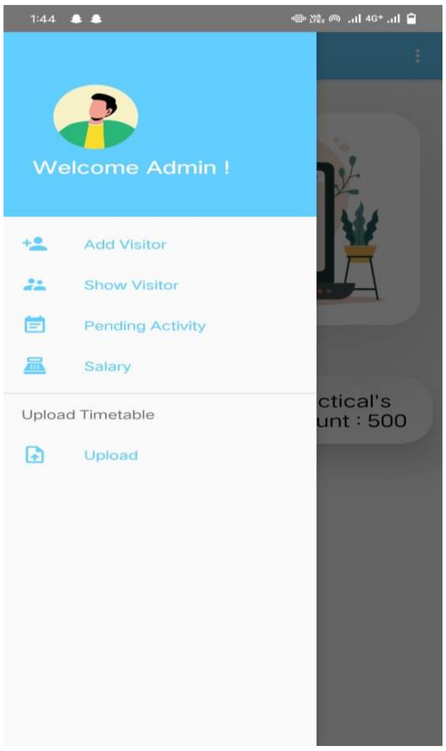
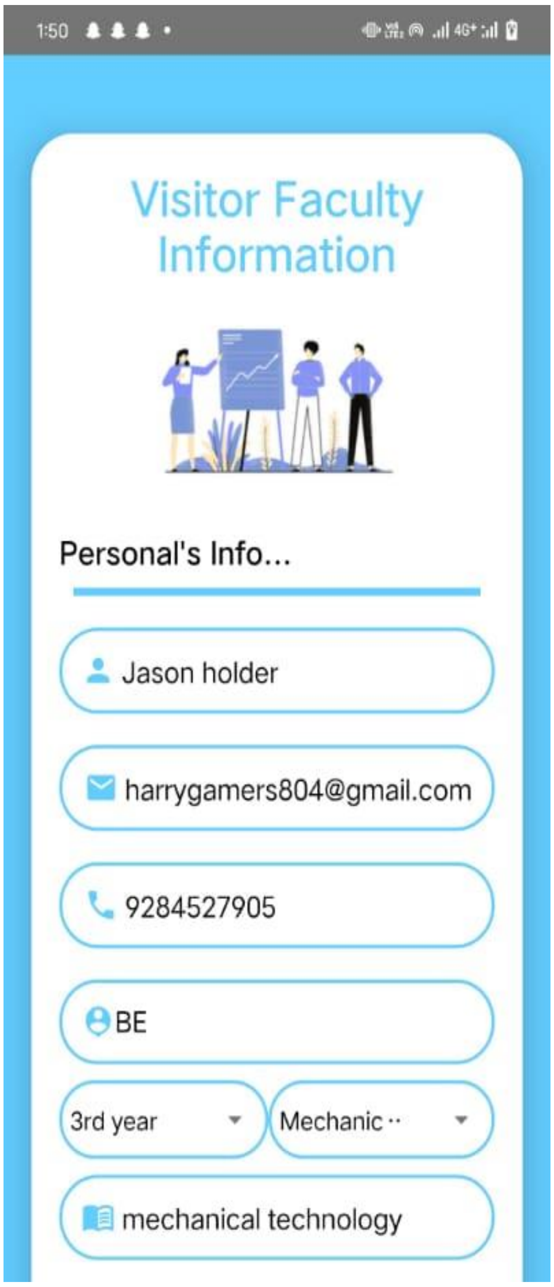


Fig 4.1-(4) Dashboard Page



Jason holder

harrygamers804@gmail.com

9284527905

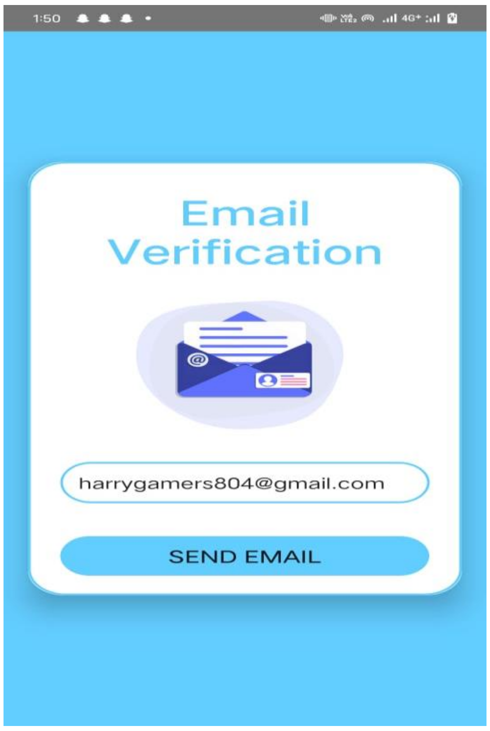
BE

3rd year

Mechanic ··

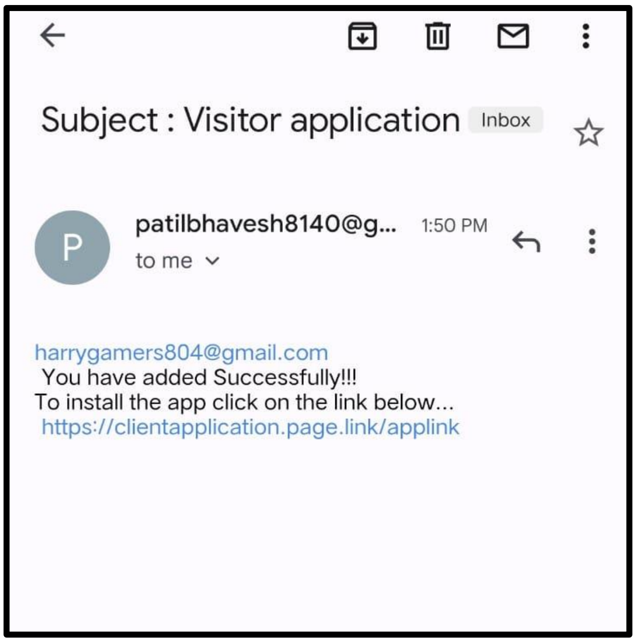
mechanical technology

Fig 4.1-(5) Visitor faculty info Page



harrygamers804@gmail.com

Fig 4.1-(6) Email verification Page

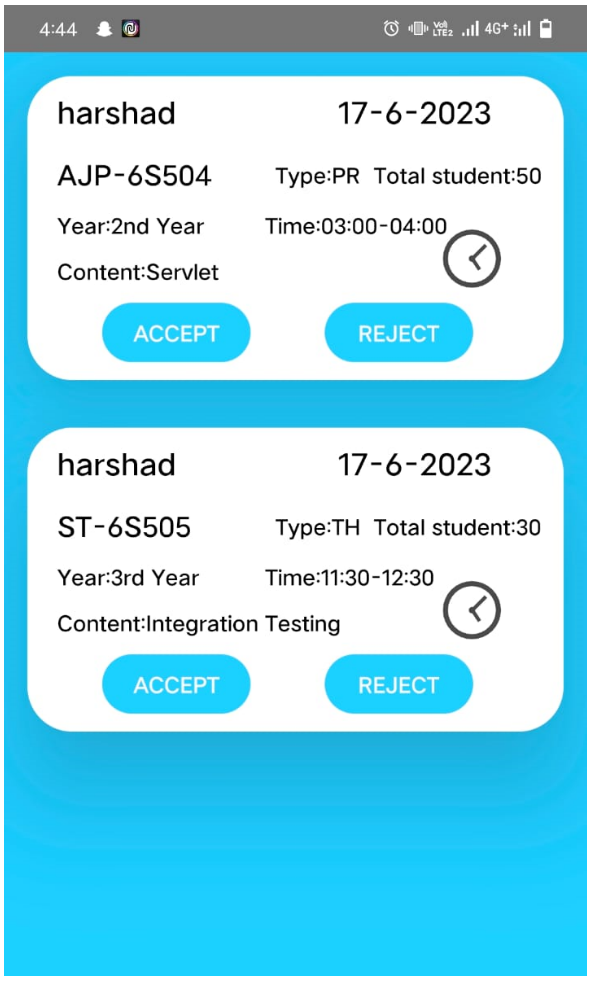


Subject : Visitor application Inbox

patilbhavesh8140@g... 1:50 PM
to me

harrygamers804@gmail.com
You have added Successfully!!!
To install the app click on the link below...
https://clientapplication.page.link/applink

Fig 4.1-(7) Email Receiving Page

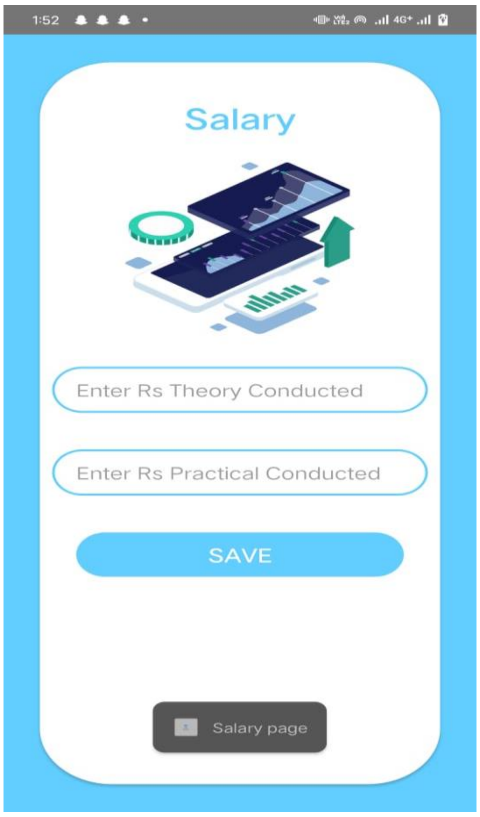


4:44

harshad 17-6-2023
AJP-6S504 Type:PR Total student:50
Year:2nd Year Time:03:00-04:00
Content:Servlet
ACCEPT REJECT

harshad 17-6-2023
ST-6S505 Type:TH Total student:30
Year:3rd Year Time:11:30-12:30
Content:Integration Testing
ACCEPT REJECT

Fig 4.1-(8) Pending Activity Page



1:52

Salary

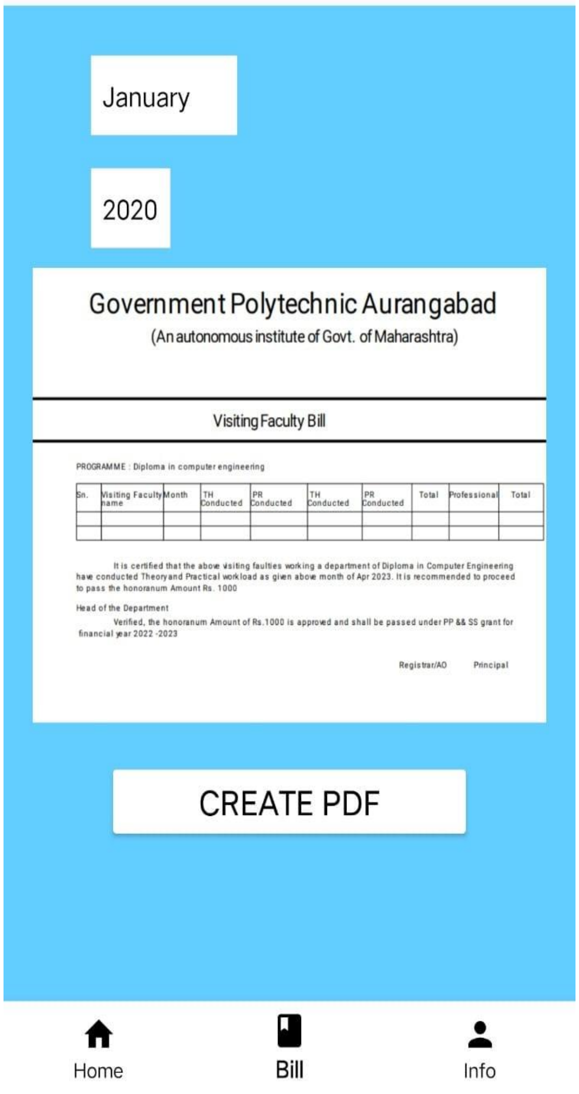
Enter Rs Theory Conducted

Enter Rs Practical Conducted

SAVE

Salary page

Fig 4.1-(9) Salary fixing Page


January

2020

Government Polytechnic Aurangabad

(An autonomous institute of Govt. of Maharashtra)

Visiting Faculty Bill

PROGRAMME : Diploma in computer engineering

Sn.	Visiting Faculty name	Month	TH Conducted	PR Conducted	TH Conducted	PR Conducted	Total	Professional	Total

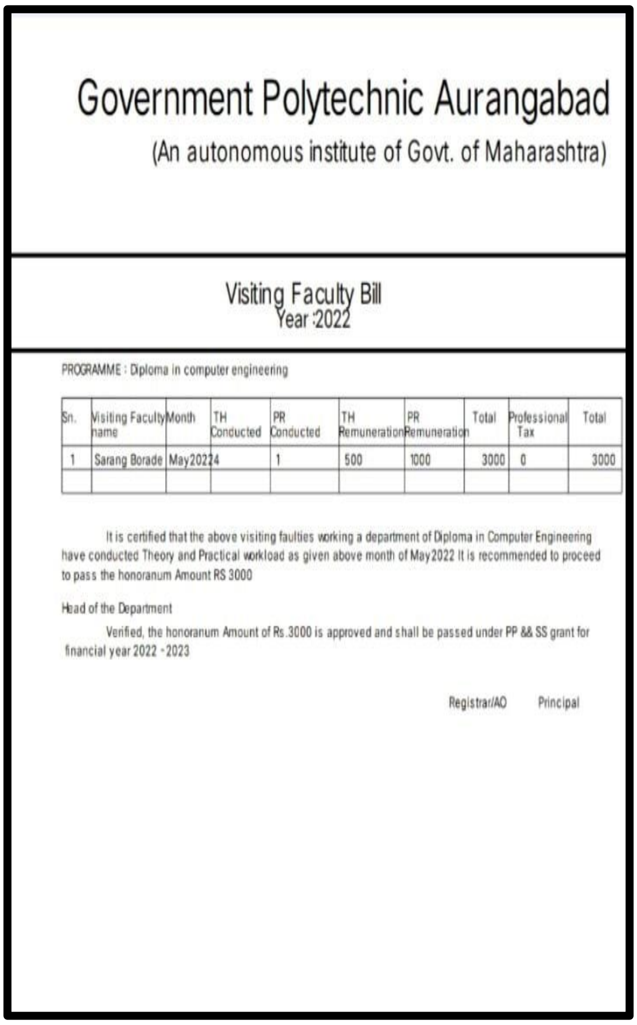
It is certified that the above visiting faulties working a department of Diploma in Computer Engineering have conducted Theory and Practical workload as given above month of Apr 2023. It is recommended to proceed to pass the honoranum Amount Rs. 1000

Head of the Department

Verified, the honoranum Amount of Rs.1000 is approved and shall be passed under PP && SS grant for financial year 2022 -2023

Registrar/AO Principal

Fig 4.1-(10) Bill Generation Page


Government Polytechnic Aurangabad

(An autonomous institute of Govt. of Maharashtra)

Visiting Faculty Bill
Year :2022

PROGRAMME : Diploma in computer engineering

Sn.	Visiting Faculty name	Month	TH Conducted	PR Conducted	TH Remuneration	PR Remuneration	Total	Professional Tax	Total
1	Sarang Borade	May2022	4	1	500	1000	3000	0	3000

It is certified that the above visiting faulties working a department of Diploma in Computer Engineering have conducted Theory and Practical workload as given above month of May2022 It is recommended to proceed to pass the honoranum Amount RS 3000

Head of the Department

Verified, the honoranum Amount of Rs.3000 is approved and shall be passed under PP && SS grant for financial year 2022 - 2023

Registrar/AO Principal

Fig 4.1-(11) Billing PDF format

4.2 Faculty Application – VisitorFunds

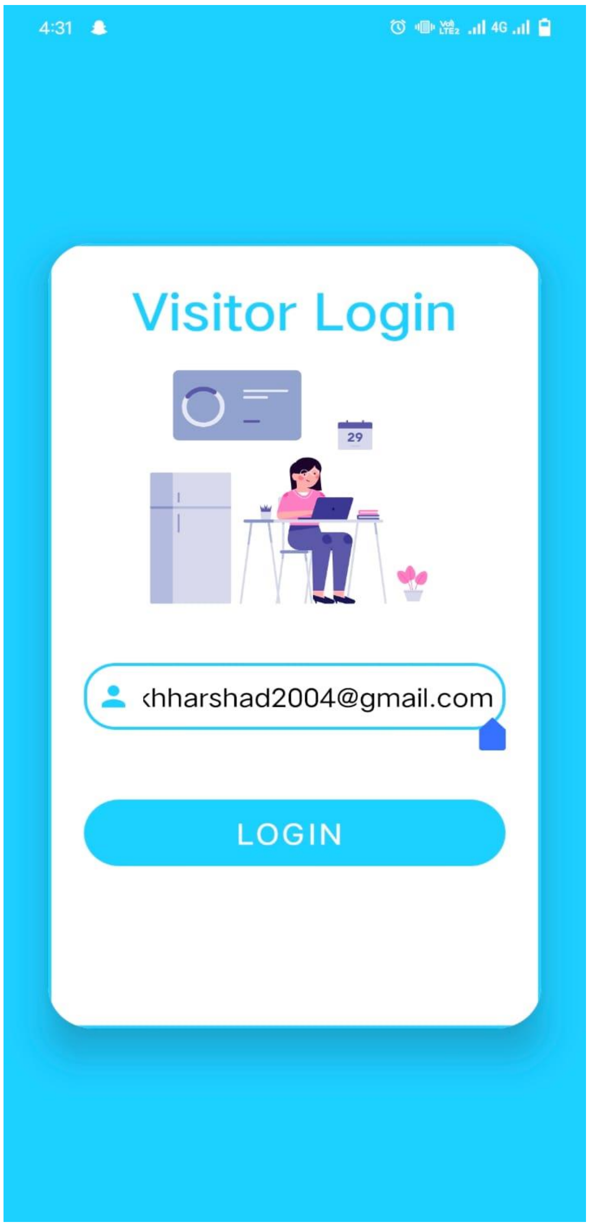


Fig 4.2-(1) Visitor Login Page

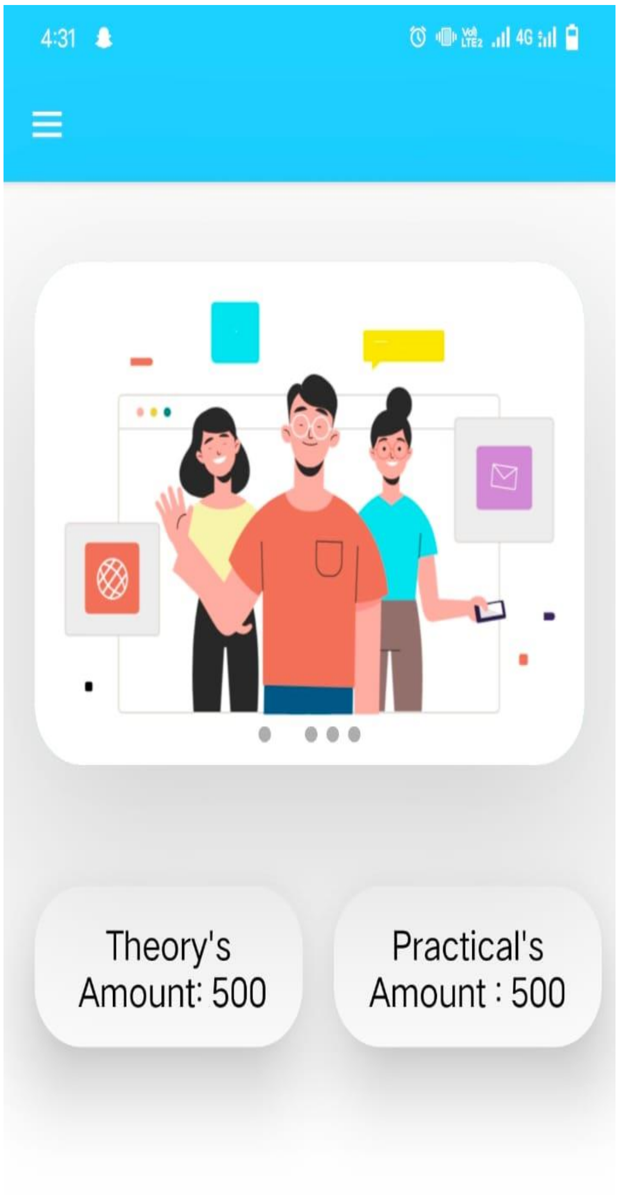


Fig 4.2-(2) Dashboard Page

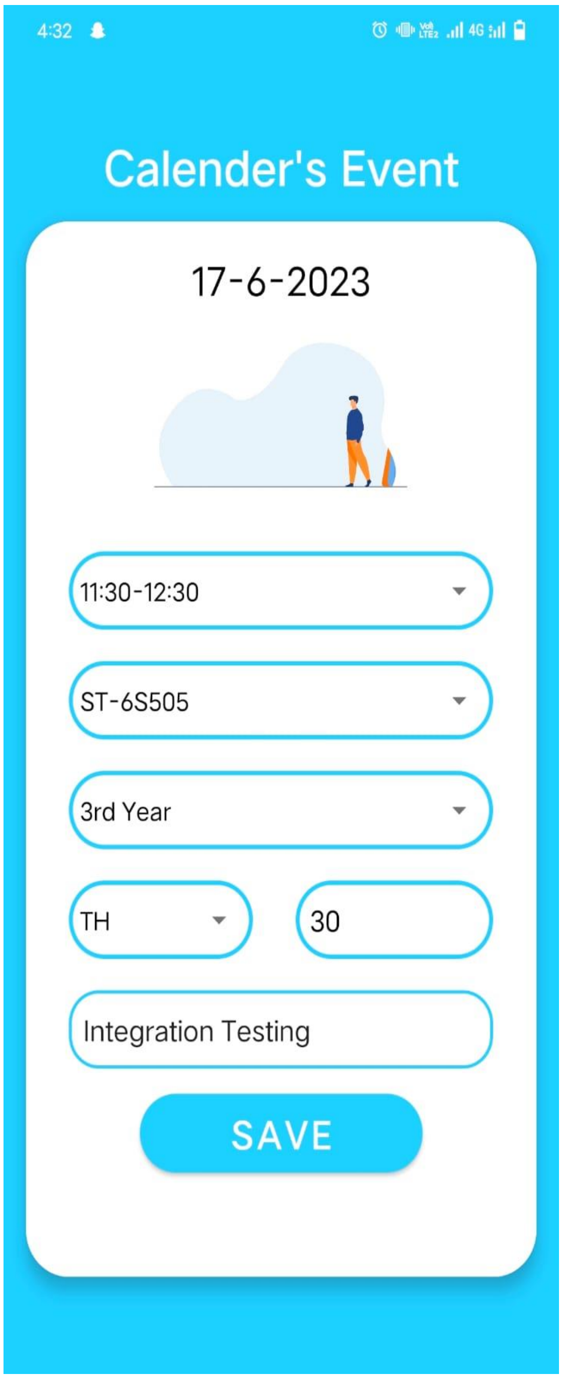
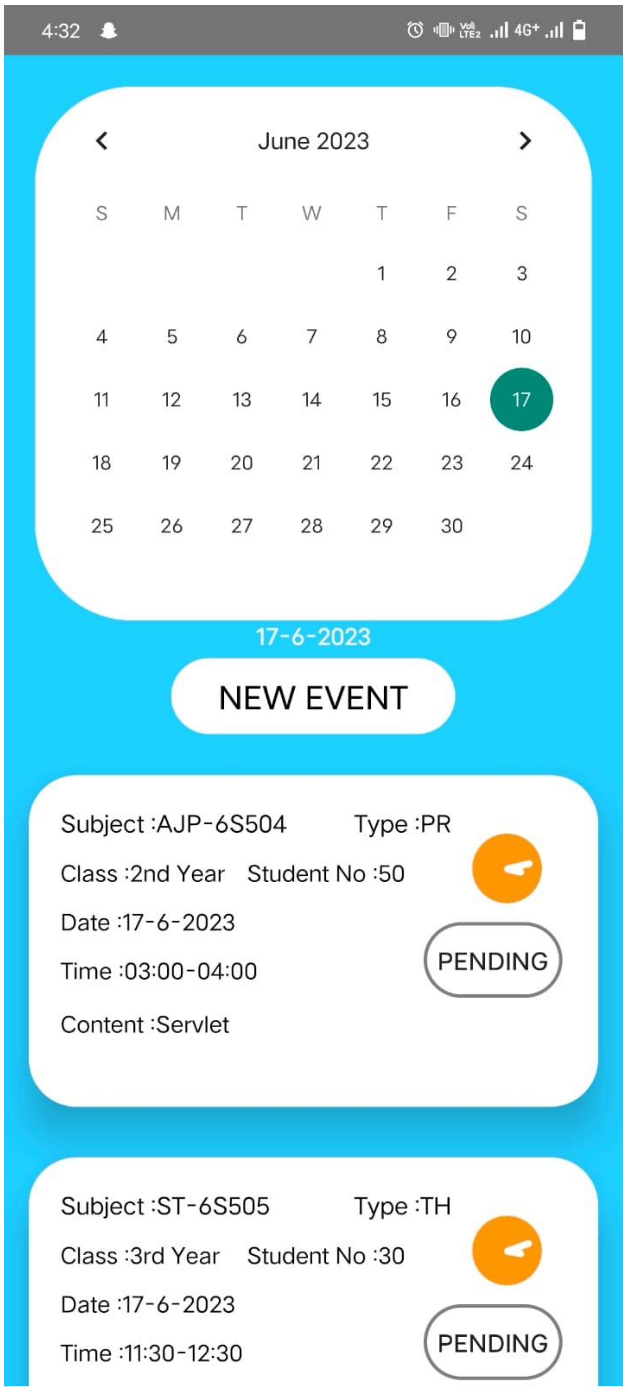


Fig 4.2-(3) Visitor Event Page



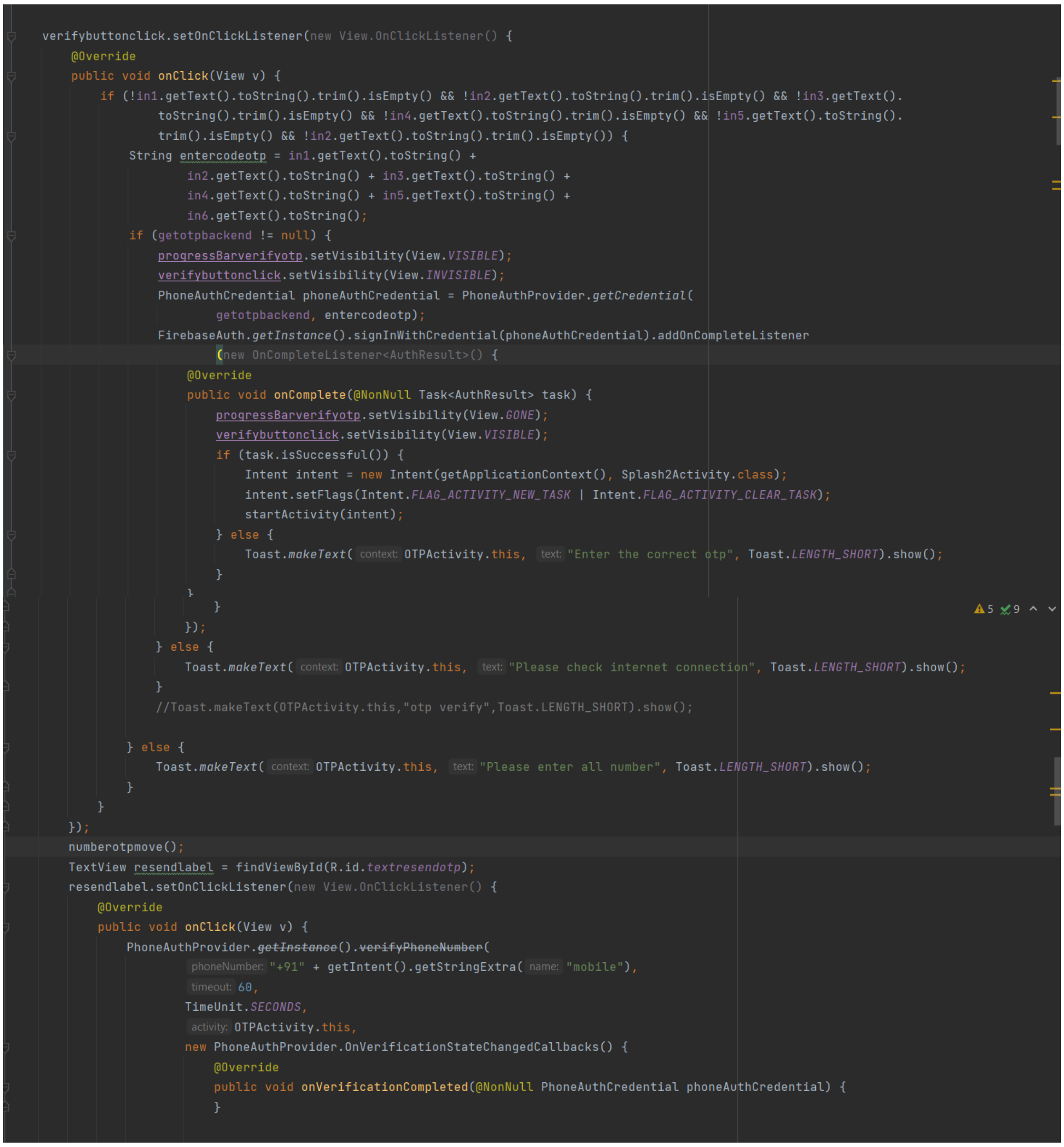
Subject :AJP-6S504 Type :PR
Class :2nd Year Student No :50
Date :17-6-2023
Time :03:00-04:00
Content :Servlet

Subject :ST-6S505 Type :TH
Class :3rd Year Student No :30
Date :17-6-2023
Time :11:30-12:30

Fig 4.2-(4) Calendar Activity Page

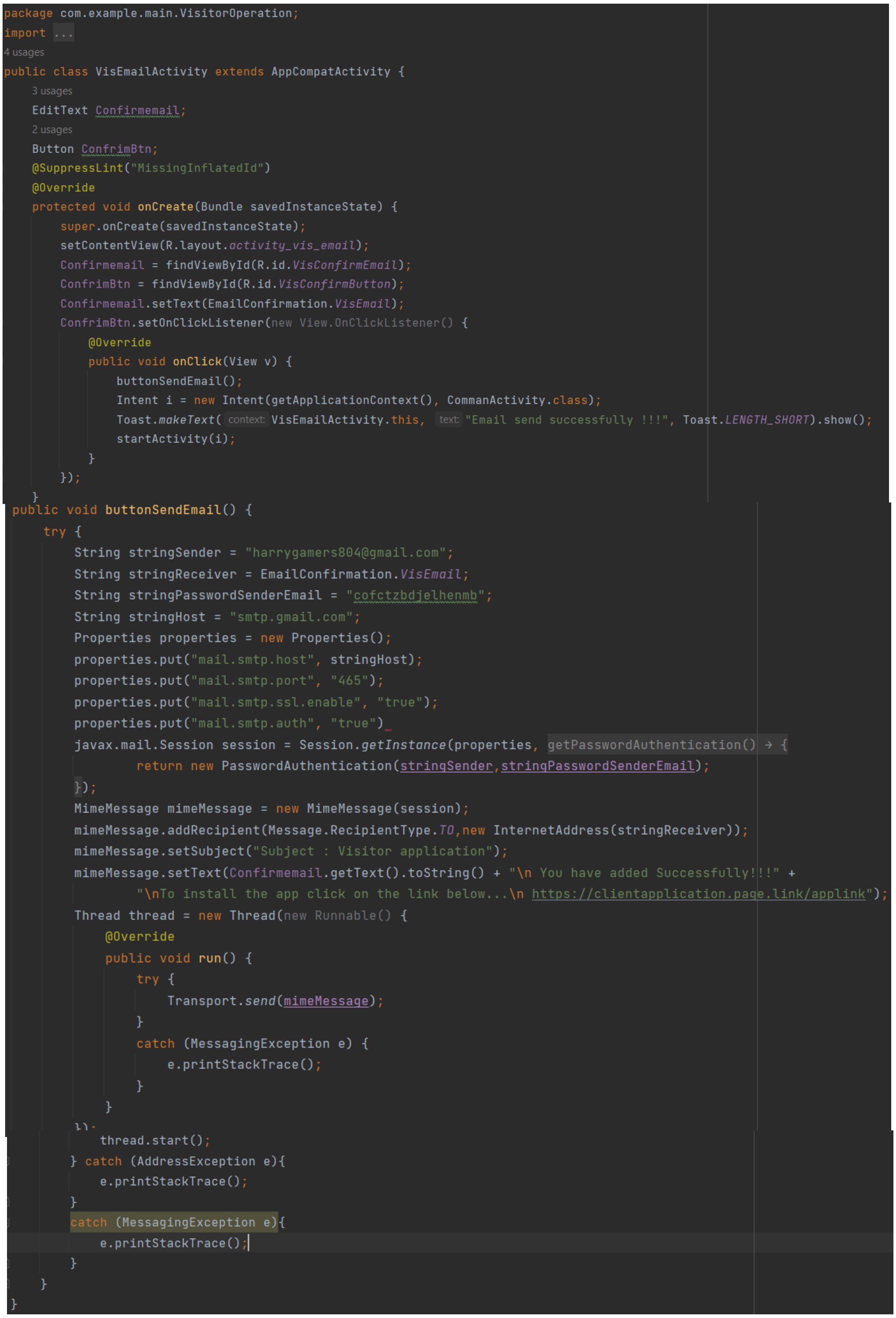
5. SAMPLE CODE

5.1 OTP Verification



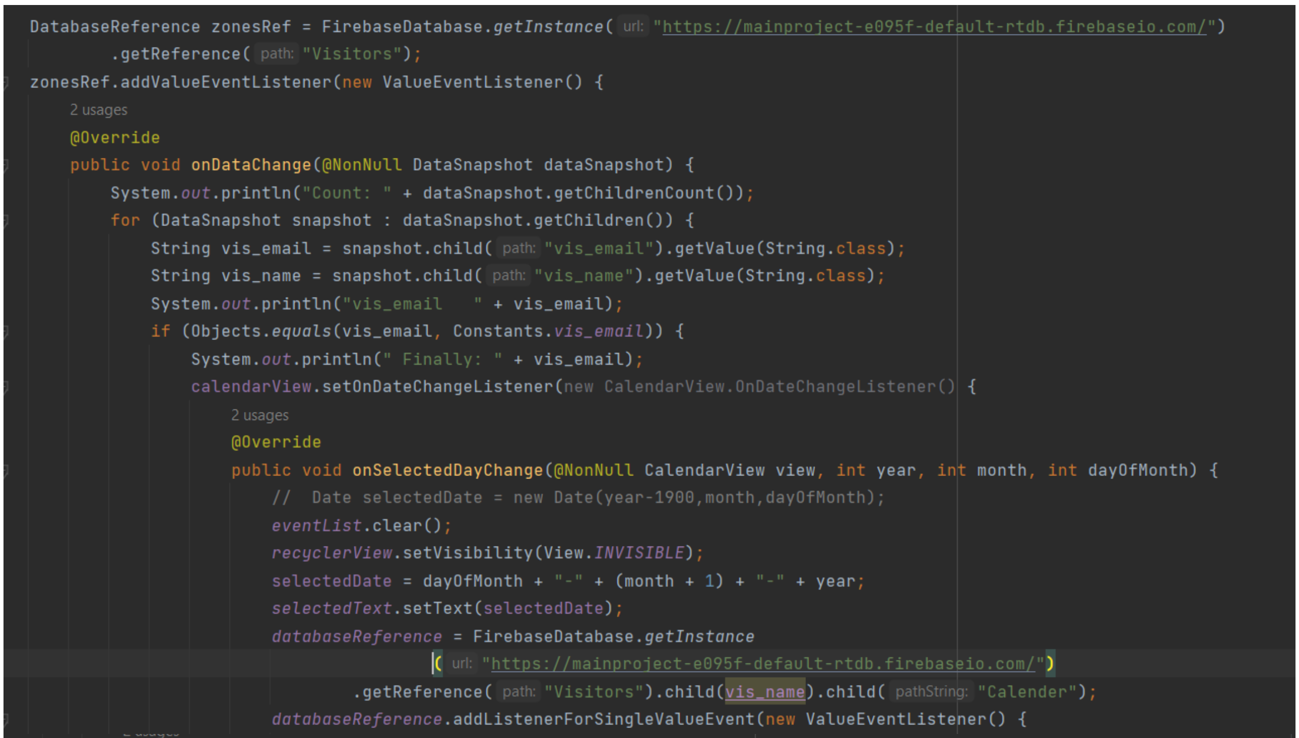
```java
verifybuttonclick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (!in1.getText().toString().trim().isEmpty() && !in2.getText().toString().trim().isEmpty() && !in3.getText().
                toString().trim().isEmpty() && !in4.getText().toString().trim().isEmpty() && !in5.getText().toString().
                trim().isEmpty() && !in2.getText().toString().trim().isEmpty()) {
            String entercodeotp = in1.getText().toString() +
                    in2.getText().toString() + in3.getText().toString() +
                    in4.getText().toString() + in5.getText().toString() +
                    in6.getText().toString();
            if (getotpbackend != null) {
                progressBarverifyotp.setVisibility(View.VISIBLE);
                verifybuttonclick.setVisibility(View.INVISIBLE);
                PhoneAuthCredential phoneAuthCredential = PhoneAuthProvider.getCredential(
                        getotpbackend, entercodeotp);
                FirebaseAuth.getInstance().signInWithCredential(phoneAuthCredential).addOnCompleteListener
                        (new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        progressBarverifyotp.setVisibility(View.GONE);
                        verifybuttonclick.setVisibility(View.VISIBLE);
                        if (task.isSuccessful()) {
                            Intent intent = new Intent(getApplicationContext(), Splash2Activity.class);
                            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                            startActivity(intent);
                        } else {
                            Toast.makeText( context: OTPActivity.this,  text: "Enter the correct otp", Toast.LENGTH_SHORT).show();
                        }
                    }
                        }
                    });
            } else {
                Toast.makeText( context: OTPActivity.this,  text: "Please check internet connection", Toast.LENGTH_SHORT).show();
            }
            //Toast.makeText(OTPActivity.this,"otp verify",Toast.LENGTH_SHORT).show();

        } else {
            Toast.makeText( context: OTPActivity.this,  text: "Please enter all number", Toast.LENGTH_SHORT).show();
        }
    }
});
numberotpmove();
TextView resendlabel = findViewById(R.id.textresendotp);
resendlabel.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        PhoneAuthProvider.getInstance().verifyPhoneNumber(
                phoneNumber: "+91" + getIntent().getStringExtra( name: "mobile"),
                timeout: 60,
                TimeUnit.SECONDS,
                activity: OTPActivity.this,
                new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
                    @Override
                    public void onVerificationCompleted(@NonNull PhoneAuthCredential phoneAuthCredential) {
                    }
```

5.2 EMAILVERIFICATION

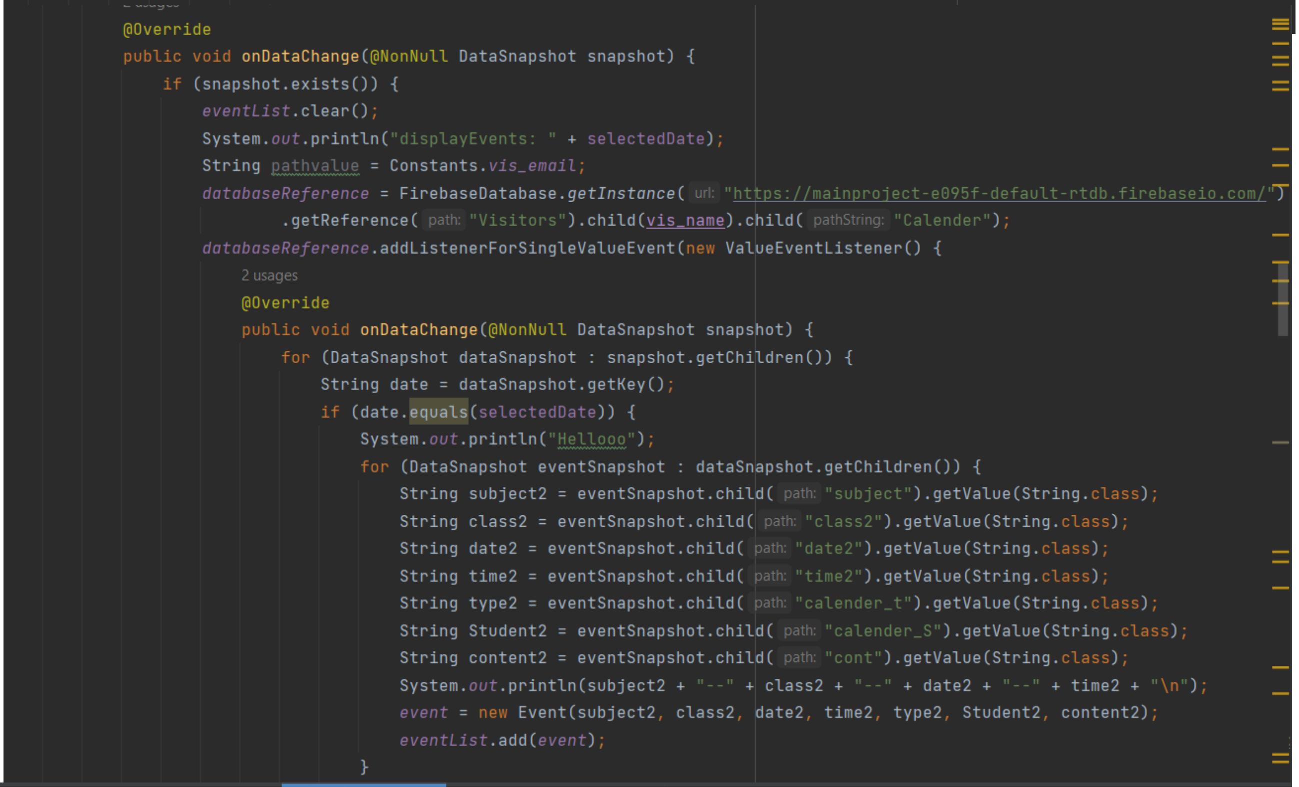


```
package com.example.main.VisitorOperation;
import ...
4 usages
public class VisEmailActivity extends AppCompatActivity {
    3 usages
    EditText Confirmemail;
    2 usages
    Button ConfrimBtn;
    @SuppressLint("MissingInflatedId")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_vis_email);
        Confirmemail = findViewById(R.id.VisConfirmEmail);
        ConfrimBtn = findViewById(R.id.VisConfirmButton);
        Confirmemail.setText(EmailConfirmation.VisEmail);
        ConfrimBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                buttonSendEmail();
                Intent i = new Intent(getApplicationContext(), CommanActivity.class);
                Toast.makeText( context: VisEmailActivity.this,  text: "Email send successfully !!!", Toast.LENGTH_SHORT).show();
                startActivity(i);
            }
        });
    }
 public void buttonSendEmail() {
    try {
        String stringSender = "harrygamers804@gmail.com";
        String stringReceiver = EmailConfirmation.VisEmail;
        String stringPasswordSenderEmail = "cofctzbdjelhenmb";
        String stringHost = "smtp.gmail.com";
        Properties properties = new Properties();
        properties.put("mail.smtp.host", stringHost);
        properties.put("mail.smtp.port", "465");
        properties.put("mail.smtp.ssl.enable", "true");
        properties.put("mail.smtp.auth", "true");
        javax.mail.Session session = Session.getInstance(properties, getPasswordAuthentication() → {
                return new PasswordAuthentication(stringSender,stringPasswordSenderEmail);
        });
        MimeMessage mimeMessage = new MimeMessage(session);
        mimeMessage.addRecipient(Message.RecipientType.TO,new InternetAddress(stringReceiver));
        mimeMessage.setSubject("Subject : Visitor application");
        mimeMessage.setText(Confirmemail.getText().toString() + "\n You have added Successfully!!!" +
                "\nTo install the app click on the link below...\n https://clientapplication.page.link/applink");
        Thread thread = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Transport.send(mimeMessage);
                }
                catch (MessagingException e) {
                    e.printStackTrace();
                }
            }
        });
            thread.start();
        } catch (AddressException e){
            e.printStackTrace();
        }
        catch (MessagingException e){
            e.printStackTrace();
        }
    }
}
```

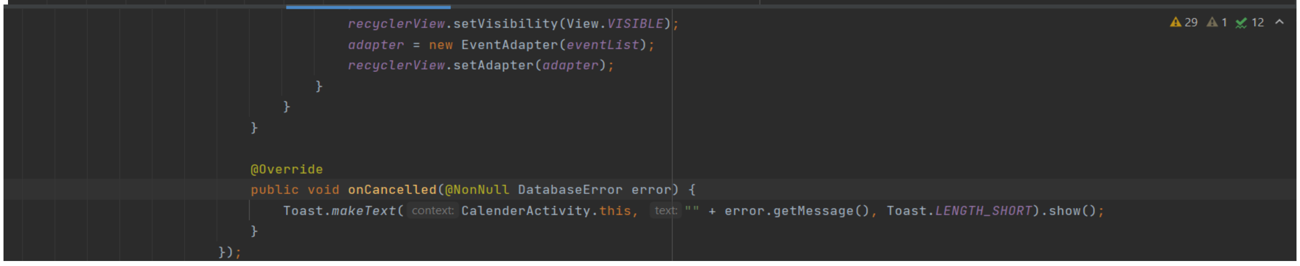
5.3 CALENDER ACTIVITY



```
DatabaseReference zonesRef = FirebaseDatabase.getInstance( url: "https://mainproject-e095f-default-rtdb.firebaseio.com/")
        .getReference( path: "Visitors");
zonesRef.addValueEventListener(new ValueEventListener() {
    2 usages
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        System.out.println("Count: " + dataSnapshot.getChildrenCount());
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            String vis_email = snapshot.child( path: "vis_email").getValue(String.class);
            String vis_name = snapshot.child( path: "vis_name").getValue(String.class);
            System.out.println("vis_email   " + vis_email);
            if (Objects.equals(vis_email, Constants.vis_email)) {
                System.out.println(" Finally: " + vis_email);
                calendarView.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
                    2 usages
                    @Override
                    public void onSelectedDayChange(@NonNull CalendarView view, int year, int month, int dayOfMonth) {
                        //  Date selectedDate = new Date(year-1900,month,dayOfMonth);
                        eventList.clear();
                        recyclerView.setVisibility(View.INVISIBLE);
                        selectedDate = dayOfMonth + "-" + (month + 1) + "-" + year;
                        selectedText.setText(selectedDate);
                        databaseReference = FirebaseDatabase.getInstance
                                (  url: "https://mainproject-e095f-default-rtdb.firebaseio.com/")
                                .getReference( path: "Visitors").child(vis_name).child( pathString: "Calender");
                        databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {
```

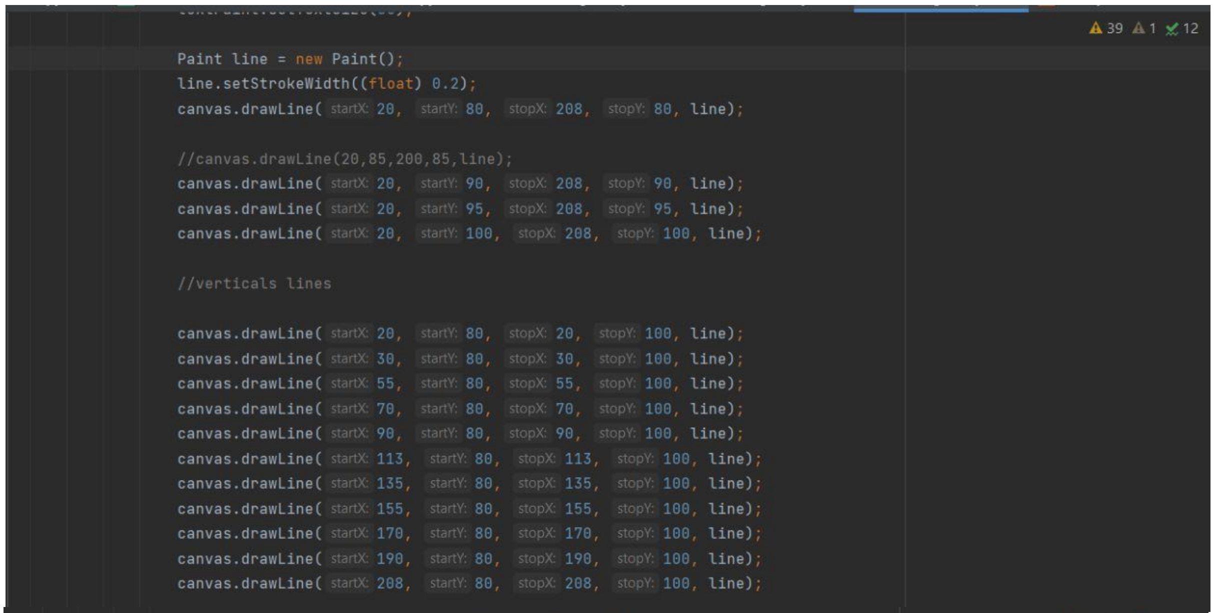


```
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    eventList.clear();
                    System.out.println("displayEvents: " + selectedDate);
                    String pathvalue = Constants.vis_email;
                    databaseReference = FirebaseDatabase.getInstance( url: "https://mainproject-e095f-default-rtdb.firebaseio.com/")
                            .getReference( path: "Visitors").child(vis_name).child( pathString: "Calender");
                    databaseReference.addListenerForSingleValueEvent(new ValueEventListener() {
                        2 usages
                        @Override
                        public void onDataChange(@NonNull DataSnapshot snapshot) {
                            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                                String date = dataSnapshot.getKey();
                                if (date.equals(selectedDate)) {
                                    System.out.println("Hellooo");
                                    for (DataSnapshot eventSnapshot : dataSnapshot.getChildren()) {
                                        String subject2 = eventSnapshot.child( path: "subject").getValue(String.class);
                                        String class2 = eventSnapshot.child( path: "class2").getValue(String.class);
                                        String date2 = eventSnapshot.child( path: "date2").getValue(String.class);
                                        String time2 = eventSnapshot.child( path: "time2").getValue(String.class);
                                        String type2 = eventSnapshot.child( path: "calender_t").getValue(String.class);
                                        String Student2 = eventSnapshot.child( path: "calender_S").getValue(String.class);
                                        String content2 = eventSnapshot.child( path: "cont").getValue(String.class);
                                        System.out.println(subject2 + "--" + class2 + "--" + date2 + "--" + time2 + "\n");
                                        event = new Event(subject2, class2, date2, time2, type2, Student2, content2);
                                        eventList.add(event);
                                    }
```

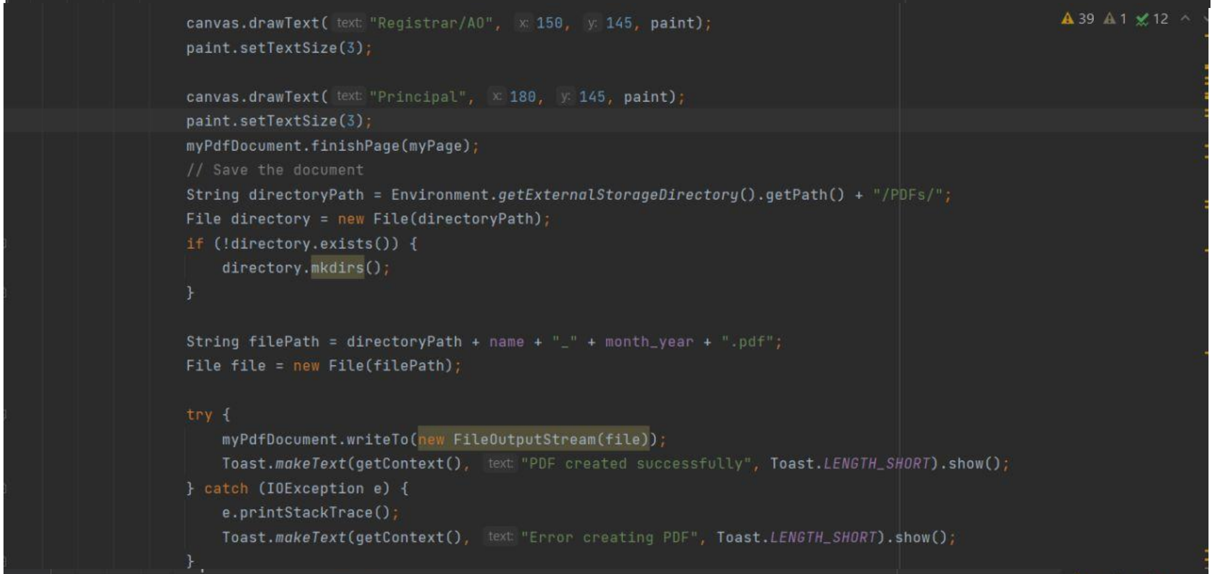


```
                                    recyclerView.setVisibility(View.VISIBLE);
                                    adapter = new EventAdapter(eventList);
                                    recyclerView.setAdapter(adapter);
                                }
                            }
                        }

                        @Override
                        public void onCancelled(@NonNull DatabaseError error) {
                            Toast.makeText( context: CalenderActivity.this,  text: "" + error.getMessage(), Toast.LENGTH_SHORT).show();
                        }
                    });
```

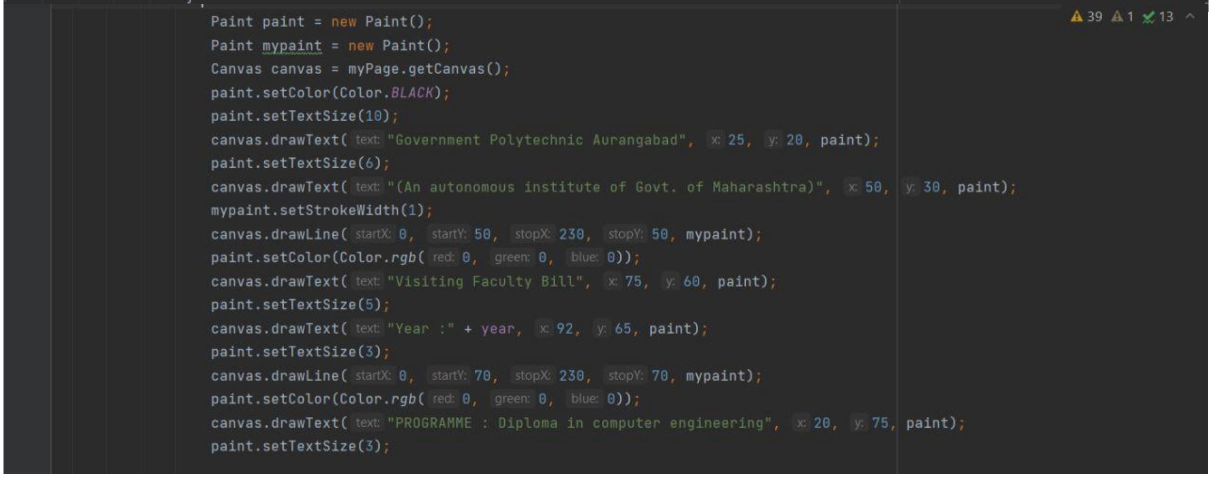
5.4 BILL GENERATION



```java
Paint line = new Paint();
line.setStrokeWidth((float) 0.2);
canvas.drawLine( startX: 20,  startY: 80,  stopX: 208,  stopY: 80, line);

//canvas.drawLine(20,85,200,85,line);
canvas.drawLine( startX: 20,  startY: 90,  stopX: 208,  stopY: 90, line);
canvas.drawLine( startX: 20,  startY: 95,  stopX: 208,  stopY: 95, line);
canvas.drawLine( startX: 20,  startY: 100,  stopX: 208,  stopY: 100, line);

//verticals lines

canvas.drawLine( startX: 20,  startY: 80,  stopX: 20,  stopY: 100, line);
canvas.drawLine( startX: 30,  startY: 80,  stopX: 30,  stopY: 100, line);
canvas.drawLine( startX: 55,  startY: 80,  stopX: 55,  stopY: 100, line);
canvas.drawLine( startX: 70,  startY: 80,  stopX: 70,  stopY: 100, line);
canvas.drawLine( startX: 90,  startY: 80,  stopX: 90,  stopY: 100, line);
canvas.drawLine( startX: 113,  startY: 80,  stopX: 113,  stopY: 100, line);
canvas.drawLine( startX: 135,  startY: 80,  stopX: 135,  stopY: 100, line);
canvas.drawLine( startX: 155,  startY: 80,  stopX: 155,  stopY: 100, line);
canvas.drawLine( startX: 170,  startY: 80,  stopX: 170,  stopY: 100, line);
canvas.drawLine( startX: 190,  startY: 80,  stopX: 190,  stopY: 100, line);
canvas.drawLine( startX: 208,  startY: 80,  stopX: 208,  stopY: 100, line);
```



```java
canvas.drawText( text: "Registrar/AO",  x: 150,  y: 145, paint);
paint.setTextSize(3);

canvas.drawText( text: "Principal",  x: 180,  y: 145, paint);
paint.setTextSize(3);
myPdfDocument.finishPage(myPage);
// Save the document
String directoryPath = Environment.getExternalStorageDirectory().getPath() + "/PDFs/";
File directory = new File(directoryPath);
if (!directory.exists()) {
    directory.mkdirs();
}

String filePath = directoryPath + name + "_" + month_year + ".pdf";
File file = new File(filePath);

try {
    myPdfDocument.writeTo(new FileOutputStream(file));
    Toast.makeText(getContext(),  text: "PDF created successfully", Toast.LENGTH_SHORT).show();
} catch (IOException e) {
    e.printStackTrace();
    Toast.makeText(getContext(),  text: "Error creating PDF", Toast.LENGTH_SHORT).show();
}
```



```java
Paint paint = new Paint();
Paint mypaint = new Paint();
Canvas canvas = myPage.getCanvas();
paint.setColor(Color.BLACK);
paint.setTextSize(10);
canvas.drawText( text: "Government Polytechnic Aurangabad",  x: 25,  y: 20, paint);
paint.setTextSize(6);
canvas.drawText( text: "(An autonomous institute of Govt. of Maharashtra)",  x: 50,  y: 30, paint);
mypaint.setStrokeWidth(1);
canvas.drawLine( startX: 0,  startY: 50,  stopX: 230,  stopY: 50, mypaint);
paint.setColor(Color.rgb( red: 0,  green: 0,  blue: 0));
canvas.drawText( text: "Visiting Faculty Bill",  x: 75,  y: 60, paint);
paint.setTextSize(5);
canvas.drawText( text: "Year :" + year,  x: 92,  y: 65, paint);
paint.setTextSize(3);
canvas.drawLine( startX: 0,  startY: 70,  stopX: 230,  stopY: 70, mypaint);
paint.setColor(Color.rgb( red: 0,  green: 0,  blue: 0));
canvas.drawText( text: "PROGRAMME : Diploma in computer engineering",  x: 20,  y: 75, paint);
paint.setTextSize(3);
```

6.PERFORMANCE ANALYSIS

6.1 Testing

1. Compatibility Testing:

Test the application on different Android devices, screen sizes, and OS versions to ensure compatibility and consistent behavior. Verify that the application functions as intended across a range of devices, including smartphones and tablets.

2. Data Integrity Testing:

Test data validation mechanisms to ensure that the application rejects invalid or inconsistent data entries. Verify that salary calculations are accurate and match the predefined rates and formulas. Validate that data is stored and retrieved correctly from the database.

3. Error Handling and Recovery Testing:

Test the application's ability to handle errors gracefully and recover from failures. Simulate different error scenarios, such as network interruptions, invalid inputs, or database failures, and verify that the application handles them appropriately.

4. User Acceptance Testing:

Involve end-users, such as visitors, faculty members, and department heads, to participate in user acceptance testing. Gather feedback on the application's usability, functionality, and overall user experience. Incorporate user feedback to make improvements and refinements to the application.

5. Regression Testing:

Perform regression testing after bug fixes, enhancements, or updates to ensure that existing functionality has not been adversely affected.

Re-test critical functionalities and verify that previously fixed issues have not resurfaced.

6.2 Test Cases

6.2.1 Test cases of Admin Site

TEST ID	TEST OBJECTIVE	PRE-REQUISITES	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS
TC_1	Users get authenticated with valid Usernames and Password	All fields filled on the login page	Valid username and password	The user should get access to an app; show dashboard page	Users get access to an app; show the dashboard page	Pass
TC_2	Add visitor faculty field	Add visitor faculty navigation bar gets activated	All fields get filled with new Visitor faculty information	Visitor faculty information should be added to the particular Admin account	New Visitor faculty information gets added to the created admin account	Pass
TC_3	Verification of Email	New Visitor added to a created admin account	Click on send email button	It should send a CLIENT APP link to the verified visitor faculty email	It sends a CLIENT APP link to the verified visitor faculty email	Pass
TC_4	Showing all visitor faculty	Show Visitor navigation bar gets activated	Click on Show Visitor Faculty	It should display all added Visitor Faculty in cardview	It displayed all added Visitor Faculty in cardview	Pass

TEST ID	TEST OBJECTIVE	PRE-REQUISITES	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS
TC_5	Pending Activity	Pending Activity Navigation gets activated	Click on	It should display all verification pending lectures on the event page	It displayed all verification pending lectures on the event page	Pass
TC_6	Lecture field	All pending lecture	Click on the "Accept" or "Reject" button	If the lecture is verified by admin, it should send the mail to the client about it and change the status in the database	The lecture gets verified by the admin and it sends the mail to the client about it and the status gets changed in the database	Pass
TC_7	Track of Visitor Faculty Activities	It enabled the Activity page of Visitor Faculty	None	It should show all verified, not verified, and rejected lectures counting of visitor faulty	It displayed all verified, not verified, and rejected lectures counting of Visitor Faulty	Pass
TC_8	Bill Generation	It activated month and year spinner	Select the month and year for bill generation	It should generate bills according to the month and year selected by admin and convert them into pdf format	It generates bills according to month and year selected by admin and converts them into pdf format	Pass

6.2.2 Test cases of Visiting Faculty Side

TEST ID	TEST OBJECTIVE	PRE-REQUISITES	INPUT DATA	EXPECTED RESULT	ACTUAL RESULT	STATUS
TC_1	Email Verification	Get an Email from the admin	None	A client should get an email of verification, CLIENT APP LINK and their username	Client get email Of verification from admin with CLIENT APP LINK with username	Pass
TC_2	Login with a valid username	The login page gets activated	Valid Username	A client should get access Client app	A client gets access to the Client app with a valid username	Pass
TC_3	Calendar Field	Calendar get Activated	Click on the "New Event" button	It should accept the lecture's detail according to the selected date and time	It accepted the lecture detail according to the selected date and time by the client	Pass
TC_4	Daily Lecture Detail	All fields are filled with lecture detail	Click on the "Save" button	It should store all lecture detail in realtime Firebase	It stored all lecture detail in realtime Firebase	Pass
TC_5	All pending Activity	Lecture activity get stored	The open Calendar navigation bar	It should display all pending lecture activities according to the selected date	It displays all pending lecture activities according to the selected date	Pass

TEST ID	**TEST OBJECTIVE**	**PRE-REQUISITES**	**INPUT DATA**	**EXPECTED RESULT**	**ACTUAL RESULT**	**STATUS**
TC_6	Track of Visiting Faculty Activities	It enabled the Activity page of Visitor Faculty	None	It should show all verified, not verified, and rejected lectures counting of visitor faulty	It displayed all verified, not verified, and rejected lectures counting of Visitor Faulty	Pass
TC_7	Salary Amount Field	Opened dashboard page	None	It should display the salary amount which decides by the admin	It displays the salary amount in cardview which decide by the admin	Pass
TC_8	Creating lecture activities that already exist	All fields are filled with lecture activity	Select the date and time which are already selected	It should display an alert message as this lecture already exists and sent to admin for validation	It displays an alert message as this lecture already exists and is sent to admin for validation	Pass
TC_9	Accepted lecture cardview get disappear without APK	Admin Accepted the pending lecture	None	It should disappear cardview which has accepted status without APK of the app	It doesn’t disappear cardview which has accepted status without the APK of the app	Fail

7. CONCLUSION

In conclusion, Android Visitor Billing Application developed for this personal project provides a valuable solution for manging payments for visitors attending lectures at a College. It is a testament to the potential of modern technologies and development in creating high-quality Android applications that meet the needs of users

The Android Visitor Billing Application is a personal project developed to manage payments for visitors attending lectures at a college. The application has been designed to provide a convenient and secure way for visitors to manage their payments for theory and practical lectures, and interact with the Head of Department (HOD) for payment-related queries.

Overall, the Android Visitor Billing Application developed for this personal project has successfully achieved its objectives of providing a convenient and secure payment system for visitors attending lectures at a college. The application has demonstrated the ability to design and develop a complex Android application that meets the requirements of users and highlights the importance of technology in simplifying and streamlining payment systems.

8. FUTURE SCOPE

The Android Visitor Billing Application developed for this personal project has been successful in meeting the requirements of visitors attending lectures at a college. However, there are several areas where the application could be improved to provide an even better user experience and meet the evolving needs of users.

Here are some of the potential future enhancements to the application:

1. Integration with College-wide Payment System: Currently, the Android Visitor Billing Application only supports payments for theory and practical lectures. However, integrating the application with a college-wide payment system could allow visitors to make payments for other services such as library fines, hostel fees, and other miscellaneous expenses.
2. Integration with Online Learning Management System: With the ongoing trend of online learning, integrating the application with an online learning management system could allow visitors to make payments for online courses and access course material through the application.
3. Integration with NFC Payments: The application could be enhanced to support Near Field Communication (NFC) payments, which would allow visitors to make payments by simply tapping their phone on an NFC-enabled payment terminal.
4. Integration with Chatbot for HOD Interaction: Integrating the application with a chatbot for HOD interaction could provide visitors with a more automated and streamlined way to interact with the HOD for payment-related queries.
5. Integration with Social Media Platforms: Integrating the application with social media platforms such as Facebook and Twitter could allow visitors to share payment-related information with their peers and receive updates about upcoming lectures and payment deadlines.
6. Optimization for Accessibility: The application could be optimized for accessibility to ensure that it can be used by visitors with disabilities or special needs.

9. REFERENCES

.https://www.bing.com/search?pglt=41&q=android+studio&cvid=c1b03a832f1945e1a0b10e88918f89a9&aqs=edge.0.0j46j0l7.6738j0j1&FORM=ANNTA1&PC=LCTS#

https://www.bing.com/search?q=lottie+files&qs=n&form=QBRE&sp=-1&ghc=1&lq=0&pq=lottie+files&sc=10-12&sk=&cvid=9B6BA14D095C411E8B57E71F1EAABEC2&ghsh=0&ghacc=0&ghpl=#

https://www.bing.com/aclk?ld=e8mFHb9YCB62G0gjawo4g0vjVUCUyqvVR1XsDDmXM3xULSh5NwwW5IKmnclvXozA4wXT25lxmgv4G5gx9X073hLna21k2EI8GiZ3pURg3nTtDXVMPJjAtzCZQdaiJDU8_8Ff_9Flcest9B0LcFNABLz1OKoyCA-epiRYDwQ1XSNODXjsun1YBFZyp2CwuBg05u36ahqw&u=aHR0cHMlM2ElMmYlMmZ

https://www.bing.com/search?q=firebase+con&qs=n&form=QBRE&sp=-1&ghc=1&lq=0&pq=firebase+co&sc=10-11&sk=&cvid=BBAAF5F49E1D424C8FF4BDF4E60A752E&ghsh=0&ghacc=0&ghpl=#

https://www.bing.com/search?q=chatgpt&qs=n&form=QBRE&sp=-1&ghc=1&lq=0&pq=chatgpt&sc=10-7&sk=&cvid=51C5D5C93C884DA1B2D2954F7C8600D5&ghsh=0&ghacc=0&ghpl=#