REVA UNIVERSITY
Bengaluru, India

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

AN ASSIGNMENT-PROJECT REPORT

ON

# Android application to monitor student using QR code

Submitted in partial fulfilment of the requirements for the

Degree of

**BACHELOR OF TECHNOLOGY IN**

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

Divya L-R21EF216

Geethanjali kumar -R21EF219

Gunashree S-R21EF221

Harsha M -R21EF224

Mamatha P-R21EF234

Under the guidance of

**Narendra Babu**

School of CSE

## Introduction :

A state-of-the-art Android software called QR Attend was created to make staff and student attendance tracking easier. Attendance tracking is made simple and effective with QR Attend's powerful features and user-friendly UI. Students and faculty members can use their own login forms to access the application and get started.

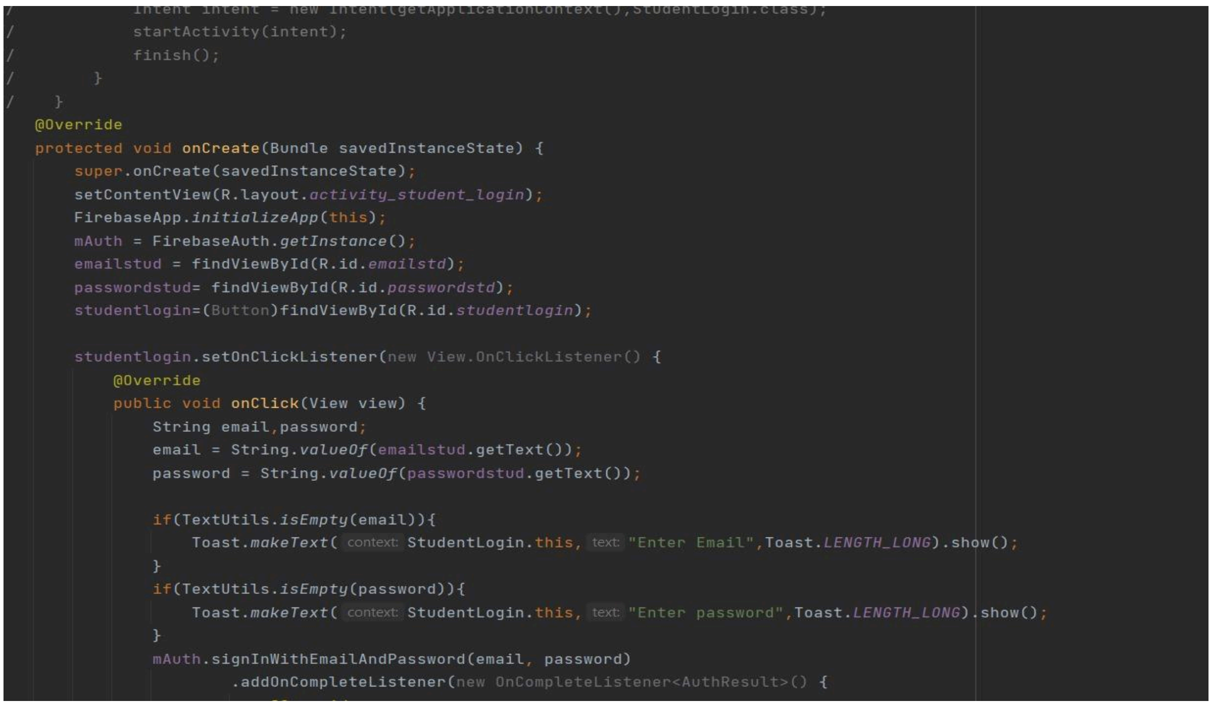
The capacity of QR Attend to produce QR codes based on a location is one of its most notable characteristics. This creative feature greatly streamlines the attendance procedure. Each class session can have a unique QR code created by the faculty that contains important information like the location. Students can quickly and properly record their attendance by using their smartphones to scan these QR codes.

## Library Used:

### Firebase Authentication:

Firebase is a powerful tool for managing user authentication in web and mobile applications, providing secure access to your app's resources. Specifically tailored for students and faculty, Firebase Authentication offers seamless integration of email authentication, simplifying the process of verifying user identities.

For students and faculty members, Firebase Authentication offers a straightforward approach to account creation and login using their email addresses. Through Firebase's robust backend services, users can securely register their accounts with their respective email addresses and passwords, ensuring confidentiality and data integrity.

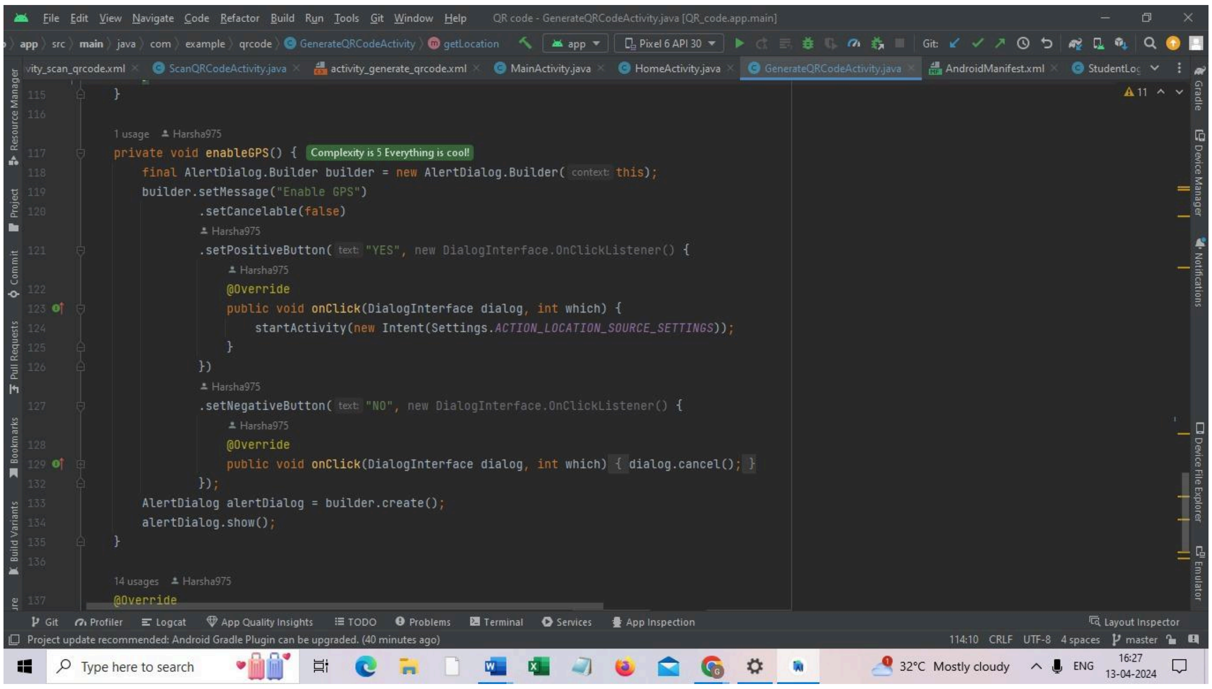


```
/            Intent intent = new Intent(getApplicationContext(),StudentLogin.class);
/            startActivity(intent);
/            finish();
/        }
/    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_student_login);
        FirebaseApp.initializeApp(this);
        mAuth = FirebaseAuth.getInstance();
        emailstud = findViewById(R.id.emailstd);
        passwordstud= findViewById(R.id.passwordstd);
        studentlogin=(Button)findViewById(R.id.studentlogin);

        studentlogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String email,password;
                email = String.valueOf(emailstud.getText());
                password = String.valueOf(passwordstud.getText());

                if(TextUtils.isEmpty(email)){
                    Toast.makeText( context: StudentLogin.this, text: "Enter Email",Toast.LENGTH_LONG).show();
                }
                if(TextUtils.isEmpty(password)){
                    Toast.makeText( context: StudentLogin.this, text: "Enter password",Toast.LENGTH_LONG).show();
                }
                mAuth.signInWithEmailAndPassword(email, password)
                        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
```

## Enabling GPS

Developers can prompt users to enable GPS location services within their Android applications, ensuring that location-based features can function properly. By integrating this functionality, developers can enhance the user experience by providing guidance on enabling essential device features while maintaining control over the app's requirements for location data.

It utilizes classes and methods from the Android SDK to create a dialog box (AlertDialog.Builder) and handle user interactions (DialogInterface.OnClickListener). Specifically, it leverages the AlertDialog.Builder class to construct a dialog with message and button options, and utilizes the Settings.ACTION_LOCATION_SOURCE_SETTINGS intent to direct the user to the device's location settings screen when the "YES" button is clicked.



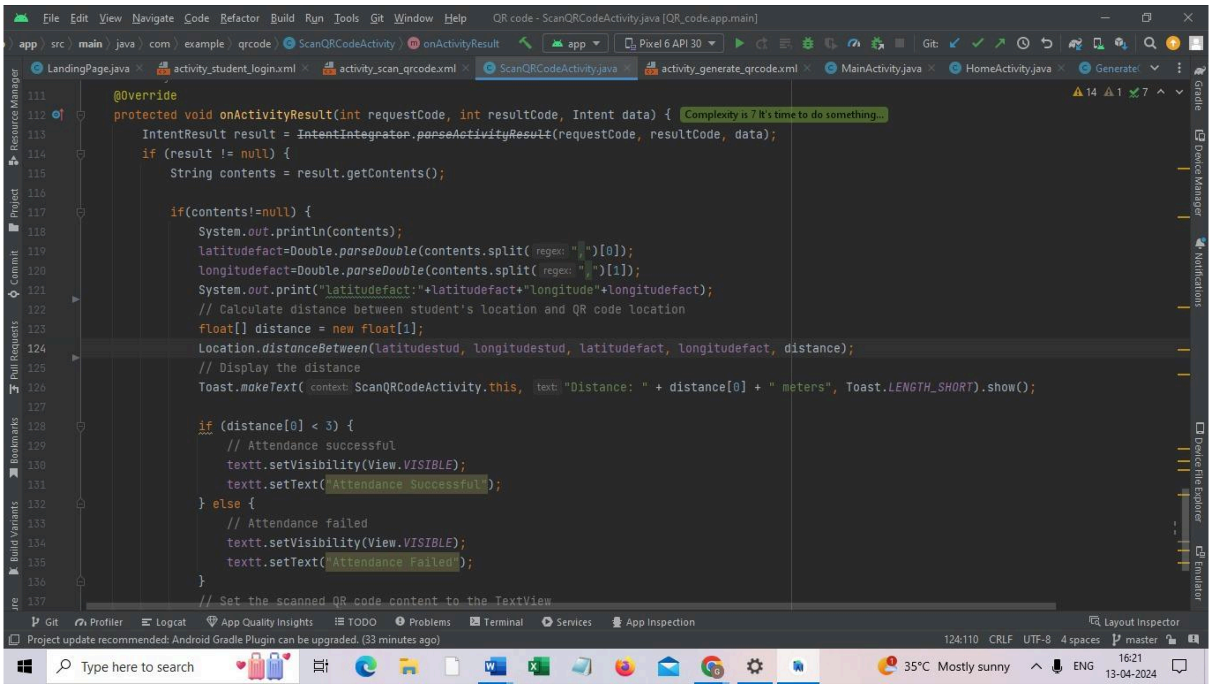
```java
    }

    private void enableGPS() {
        final AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
        builder.setMessage("Enable GPS")
                .setCancelable(false)
                .setPositiveButton( text: "YES", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
                    }
                })
                .setNegativeButton( text: "NO", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) { dialog.cancel(); }
                });
        AlertDialog alertDialog = builder.create();
        alertDialog.show();
    }

    @Override
```

Project update recommended: Android Gradle Plugin can be upgraded. (40 minutes ago)

## Location Validation:

Location validation is a process of Verifying whether the device's location services are activated and that the required authorizations are in place to access the device's location is known as location validation. This is essential for applications that depend on location-based features like navigation, mapping, or location-aware services.

location validation is performed by first checking if GPS is enabled on the device using the LocationManager class. If GPS is not enabled, the user is prompted to enable it through the enableGPS() method. Once GPS is enabled, the app checks for location permissions using the checkSelfPermission() method. If permissions are granted, the app retrieves the device's last known location using various location providers (GPS, network, passive). The retrieved location is then used to update the UI with latitude and longitude coordinates and potentially generate a QR code based on this location.

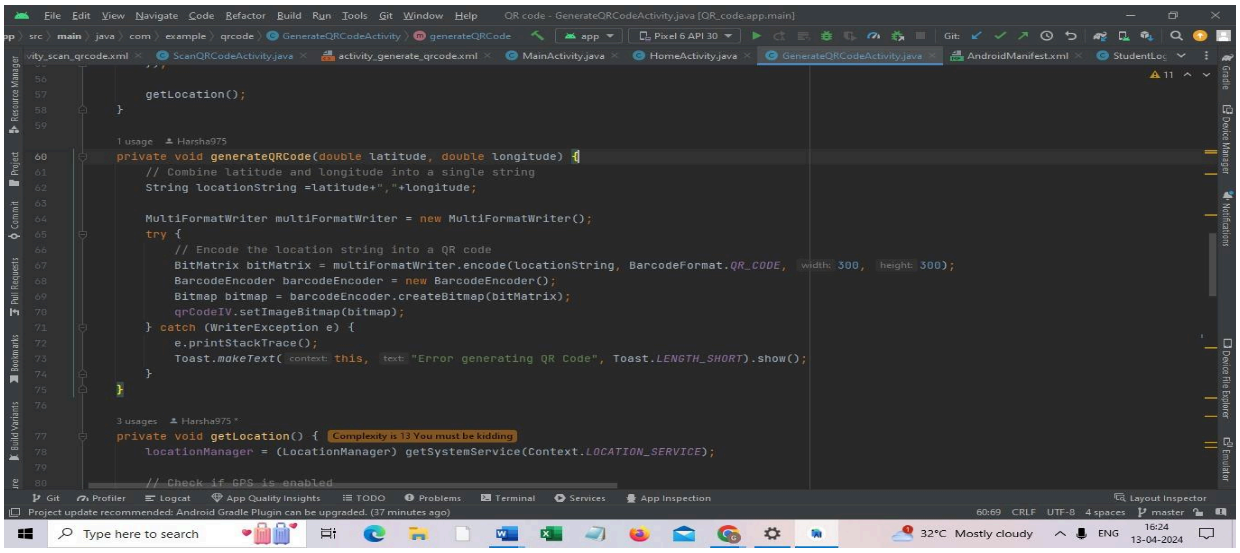


```java
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {  Complexity is 7 It's time to do something...
        IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
        if (result != null) {
            String contents = result.getContents();

            if(contents!=null) {
                System.out.println(contents);
                latitudefact=Double.parseDouble(contents.split( regex: ",")[0]);
                longitudefact=Double.parseDouble(contents.split( regex: ",")[1]);
                System.out.print("latitudefact:"+latitudefact+"longitude"+longitudefact);
                // Calculate distance between student's location and QR code location
                float[] distance = new float[1];
                Location.distanceBetween(latitudestud, longitudestud, latitudefact, longitudefact, distance);
                // Display the distance
                Toast.makeText( context: ScanQRCodeActivity.this,  text: "Distance: " + distance[0] + " meters", Toast.LENGTH_SHORT).show();

                if (distance[0] < 3) {
                    // Attendance successful
                    textt.setVisibility(View.VISIBLE);
                    textt.setText("Attendance Successful");
                } else {
                    // Attendance failed
                    textt.setVisibility(View.VISIBLE);
                    textt.setText("Attendance Failed");
                }
                // Set the scanned QR code content to the TextView
```

## Generate QR Code:

The PackageManager, Location, and LocationManager libraries handle package management and location services. They provide essential functionalities for accessing package information and retrieving device location data.

On the other hand, the ZXing library assists in barcode operations, offering capabilities for generating and decoding various barcode types. Complementing this, the BarcodeEncoder and QRGenerator libraries specialize in generating barcode images and QR codes, respectively, streamlining the process within Android applications.



```java
        getLocation();
    }

    1 usage  Harsha975
    private void generateQRCode(double latitude, double longitude) {
        // Combine latitude and longitude into a single string
        String locationString =latitude+","+longitude;

        MultiFormatWriter multiFormatWriter = new MultiFormatWriter();
        try {
            // Encode the location string into a QR code
            BitMatrix bitMatrix = multiFormatWriter.encode(locationString, BarcodeFormat.QR_CODE,  width: 300,  height: 300);
            BarcodeEncoder barcodeEncoder = new BarcodeEncoder();
            Bitmap bitmap = barcodeEncoder.createBitmap(bitMatrix);
            qrCodeIV.setImageBitmap(bitmap);
        } catch (WriterException e) {
            e.printStackTrace();
            Toast.makeText( context: this,  text: "Error generating QR Code", Toast.LENGTH_SHORT).show();
        }
    }

    3 usages  Harsha975 *
    private void getLocation() {  Complexity is 13 You must be kidding
        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

        // Check if GPS is enabled
```

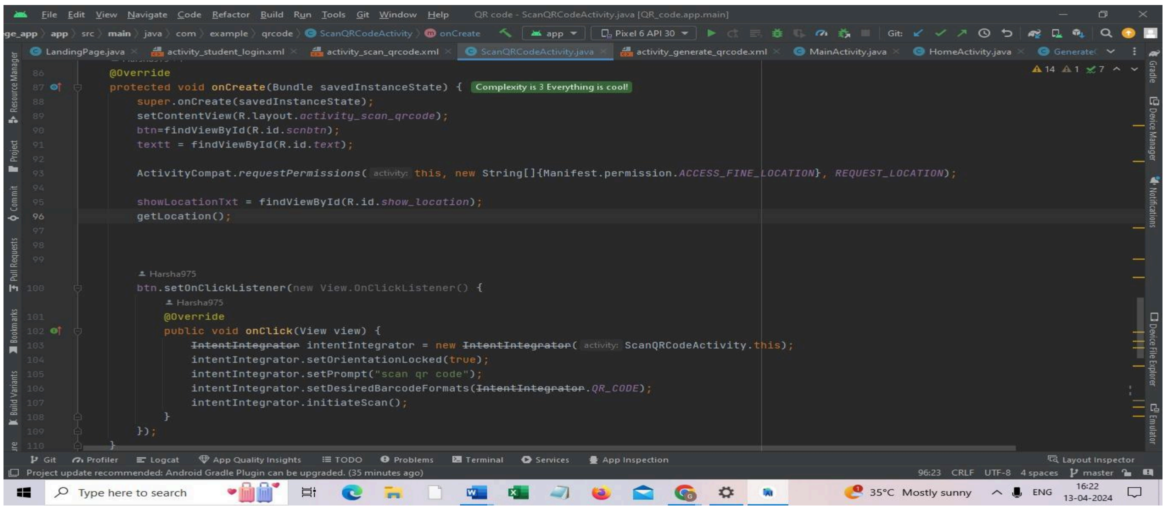
This method, named **`generateQRCode`**, is responsible for creating a QR code image based on latitude and longitude coordinates. It first combines the latitude and longitude values into a single string, representing the location information. Using the `MultiFormatWriter` class from the `com.google.zxing.*` library, the method attempts to encode the location string into a QR code. The `encode` method takes parameters such as the content to encode (locationString), the barcode format (BarcodeFormat.QR_CODE), and the width and height of the resulting barcode image (300x300 pixels in this case). If the encoding process is

successful, the method uses the `BarcodeEncoder` class to create a bitmap image of the barcode. Finally, it sets this bitmap image as the content of an `ImageView` named `qrCodeIV`, effectively displaying the QR code within the application's user interface. In case of any errors during the encoding process, such as `WriterException`, the method prints the stack trace and displays a short error message using a `Toast`.

## Scan QR Code:

The application integrates ZXing for barcode scanning and utilizes IntentIntegrator and IntentResult classes to manage scan initiation and retrieval of results. Additionally, it interacts with Android's location services through Location and LocationManager classes, enabling access to device coordinates.

Meanwhile, the inclusion of DialogInterface and AlertDialog enhances user interaction by facilitating prompts for enabling location services when necessary. Moreover, the code snippet exhibits responsible permission handling using Manifest.permission and ActivityCompat, ensuring compliance with Android's permission model for bolstered privacy and security. These components collectively fortify the application's functionality, accentuating seamless barcode scanning, location-based services, user engagement, and meticulous permission management.


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
QR code - ScanQRCodeActivity.java [QR_code.app.main]
LandingPage.java
activity_student_login.xml
activity_scan_qrcode.xml
ScanQRCodeActivity.java
activity_generate_qrcode.xml
MainActivity.java
HomeActivity.java
Complexity is 3 Everything is cool!

```java
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scan_qrcode);
        btn=findViewById(R.id.scnbtn);
        textt = findViewById(R.id.text);

        ActivityCompat.requestPermissions( this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);

        showLocationTxt = findViewById(R.id.show_location);
        getLocation();



        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                IntentIntegrator intentIntegrator = new IntentIntegrator( ScanQRCodeActivity.this);
                intentIntegrator.setOrientationLocked(true);
                intentIntegrator.setPrompt("scan qr code");
                intentIntegrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE);
                intentIntegrator.initiateScan();
            }
        });
    }
```

Project update recommended: Android Gradle Plugin can be upgraded. (35 minutes ago)

1. **Activity Initialization:** The `onCreate` method initializes the activity by calling the superclass's `onCreate` method and setting the content view to the layout defined in `activity_scan_qrcode.xml`.

2. **Permission Request:** It requests the `ACCESS_FINE_LOCATION` permission using `ActivityCompat.requestPermissions`. This permission is needed for accessing the device's precise location.

3. **UI Initialization:** It initializes UI elements such as a button (`btn`) and a text view (`textt`) by finding their respective views in the layout XML.

4. **Location Retrieval:** It calls the `getLocation` method to retrieve the device's current location. This method is presumably responsible for obtaining the location and updating the UI accordingly.

5. **Button Click Listener:** It sets an `OnClickListener` for the button (`btn`). When clicked, it initiates the QR code scanning process using the `IntentIntegrator` class from a barcode scanning library (likely a third-party library). The scanning parameters, such as orientation lock, prompt message, and desired barcode formats, are configured before initiating the scan.

Overall, this code sets up the `ScanQRCodeActivity`, requests location permission, initializes UI components, retrieves the device's location, and initiates the QR code scanning process when the button is clicked.

## Backend :

### MainActivity:

This `MainActivity` serves as the entry point for the application. It sets the layout to display a fullscreen splash screen (`activity_main.xml`) for a specified duration using a `Handler`. After the delay, it navigates to the `LandingPage` activity, initiating the main functionality or user interface of the application.

### LandingPage:

The LandingPage activity acts as the entry point of the application, presenting faculty and student login options. It features buttons for faculty (loginfact) and student (loginstd) logins, facilitating navigation to corresponding functionalities. Through its intuitive interface, users can seamlessly access login options based on their roles, enhancing user experience and application usability.

### FacultyLogin:

The FacultyLogin activity handles the authentication process for faculty members. Upon user input, it verifies the email and password fields. If they are empty, it displays corresponding toast messages. Then, it uses Firebase Authentication to sign in the faculty user. If the authentication is successful, it redirects the user to the GenerateQRCodeActivity, indicating a successful login. If authentication fails, it displays an authentication failure message.

### StudentLogin:

The StudentLogin activity facilitates the authentication process for student users. Upon user input, it validates the email and password fields. If either field is empty, it displays corresponding toast messages prompting the user to fill in the required information. Using Firebase Authentication, it attempts to sign in the student user with the provided credentials. If the authentication is successful, it redirects the user to the ScanQRCodeActivity, indicating a successful login. In case of authentication failure, it displays a toast message indicating the failure.

**HomeActivity:**

The `HomeActivity` class serves as the central hub of the application, offering options for generating and scanning QR codes. It initializes buttons for each functionality and assigns click listeners to facilitate navigation. Through intuitive button interactions, users can seamlessly access the desired QR code functionalities, enhancing the overall user experience of the application.

**ScanQRCodeActivity:**

The ScanQRCodeActivity facilitates real-time QR code scanning, location retrieval, and attendance tracking. It dynamically handles QR code scanning events, calculates distances between student and QR code locations, and updates attendance status accordingly. Through seamless integration of the ZXing library for scanning and location services for coordinate retrieval, it provides a comprehensive solution for efficient attendance management.

**GenerateQRCode:**

The GenerateQRCodeActivity dynamically generates QR codes reflecting the device's real-time location coordinates. It seamlessly integrates location retrieval and QR code generation functionalities, ensuring accurate representation of the current location. Through intuitive UI elements and permission handling, it offers a user-friendly experience for sharing location information via QR codes.

## Front end :

activity_home:

The home activity serves as the application's main hub, making it simple for teachers and students to access various features. The ability to create and scan QR codes, see attendance records, and log in or out are among the capabilities available to users.

activity_faculty:

This activity provides the portal through which faculty members can access features linked to attendance. With this user-friendly interface, faculty members can quickly manage student attendance, see attendance data, and create QR codes for class sessions.

activity_student_login:

This activity gives students access to their login page where they can mark attendance, create QR codes if necessary, and view their attendance records. It guarantees individualised access to student-specific features and safe authentication.

activity_landing_page:

The landing page activity serves as the initial screen that users encounter upon launching the application. It may include a brief introduction to the app, key features, and options to proceed with student or faculty login.

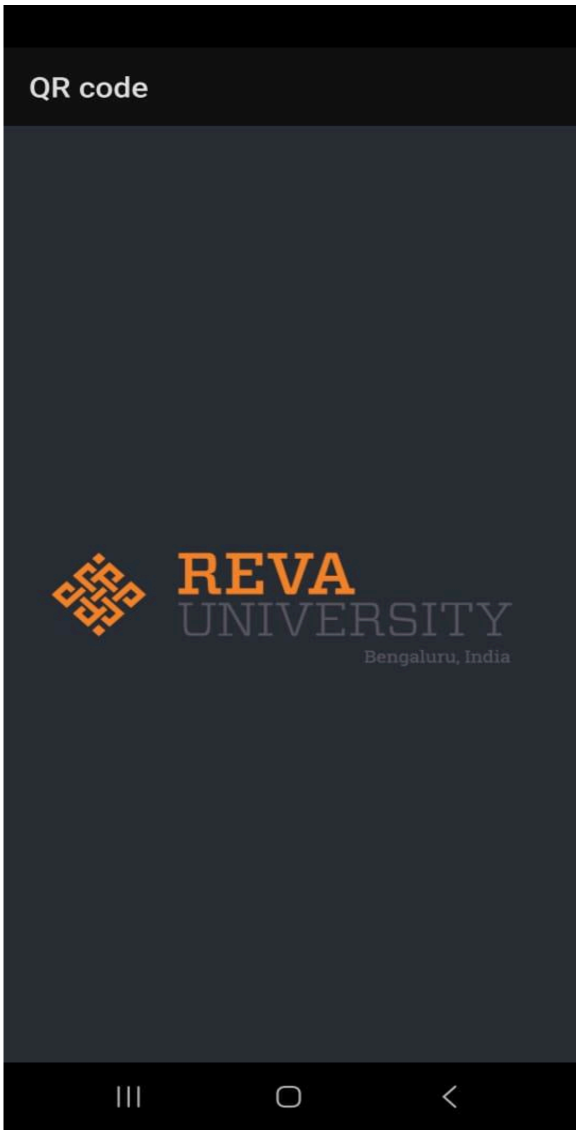
activity_generate_qrcode:

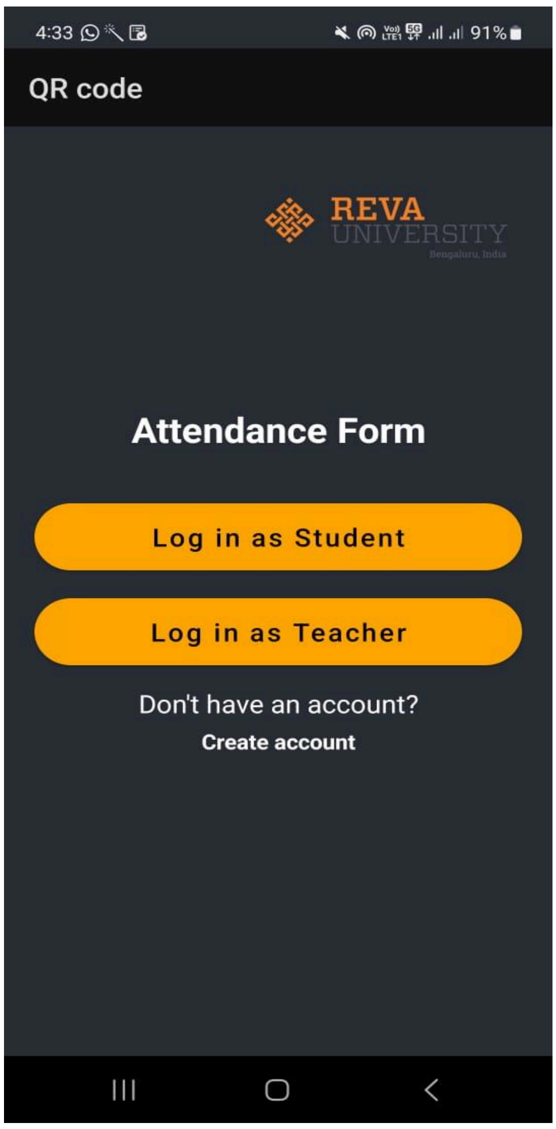
This activity allows users to create QR codes for usage in class, especially for faculty members. In order to guarantee that each class has a unique QR code, they can alter the code according to the day and time of the session. This feature improves accuracy and streamlines the attendance tracking procedure.

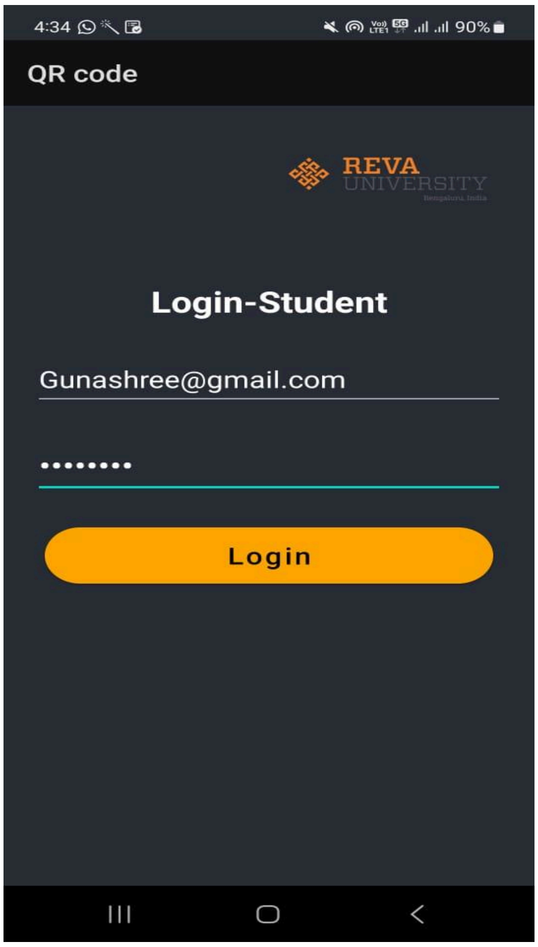
activity_scan_qrcode:

Students, in particular, can utilise this activity to register their attendance for class sessions by scanning QR codes that faculty members present. Students can use their mobile devices to effortlessly and effectively record their attendance thanks to the interface.

## OUTPUT:






4:34 90%

QR code

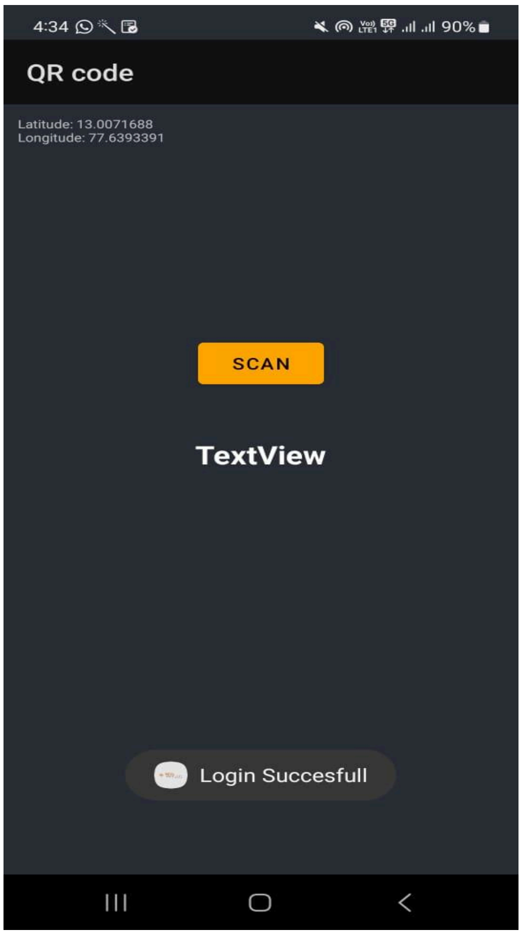
REVA UNIVERSITY

Login-Student

Gunashree@gmail.com

••••••••

Login


4:34 90%

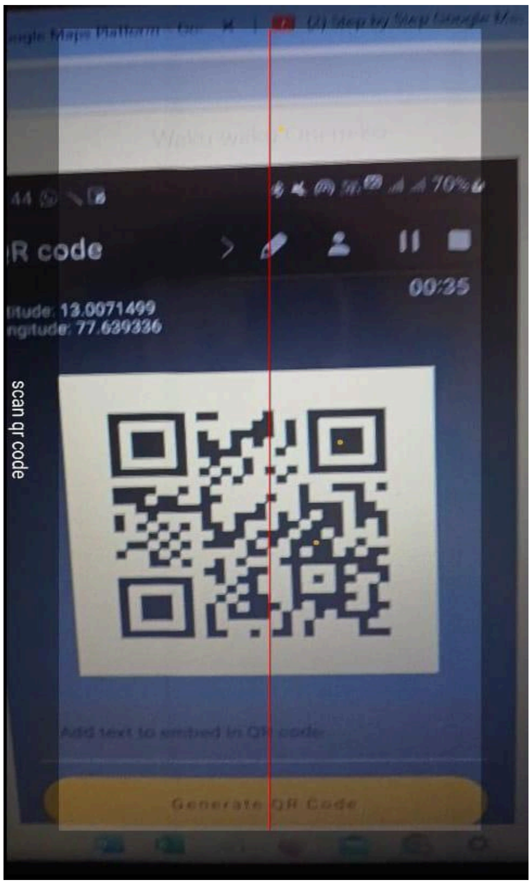
QR code

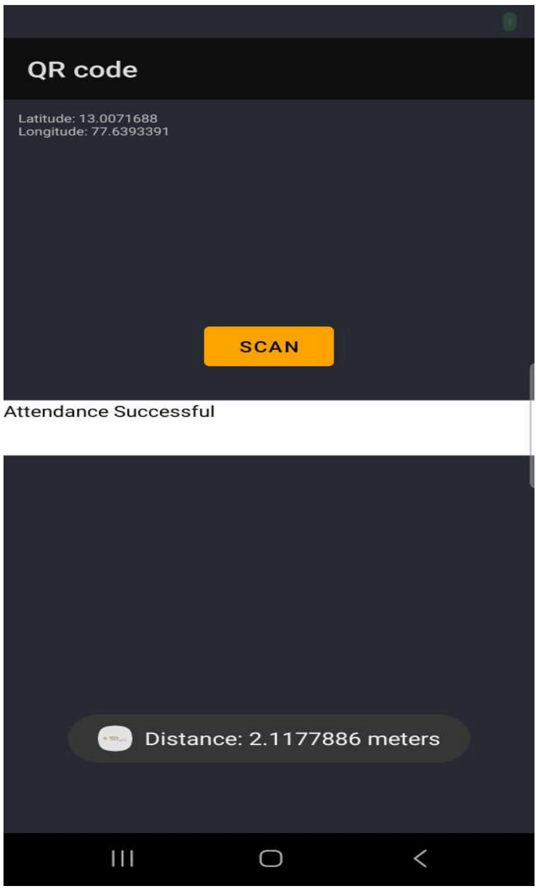
Latitude: 13.0071688
Longitude: 77.6393391

SCAN

TextView

Login Succesfull



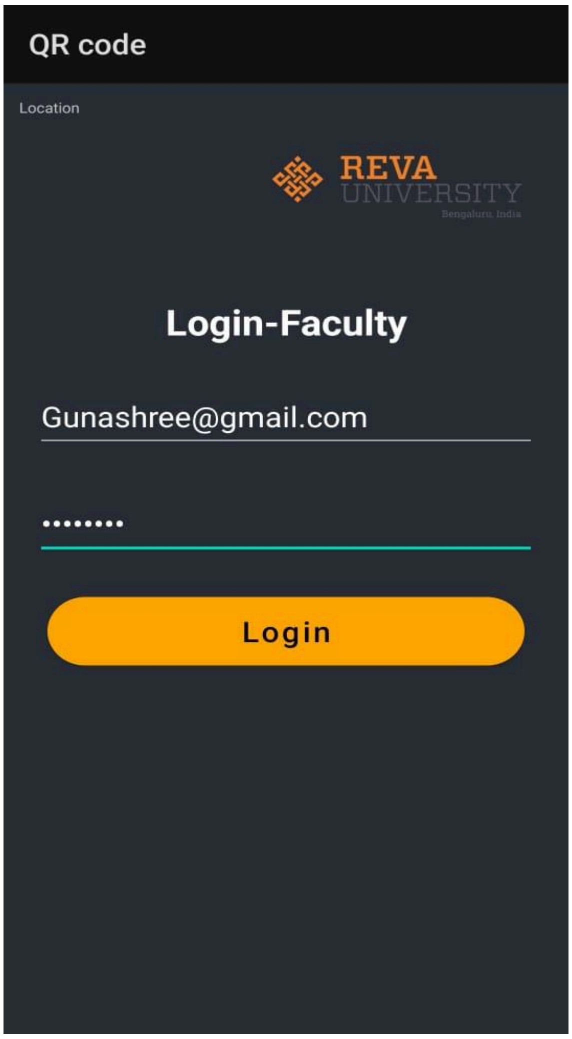

QR code

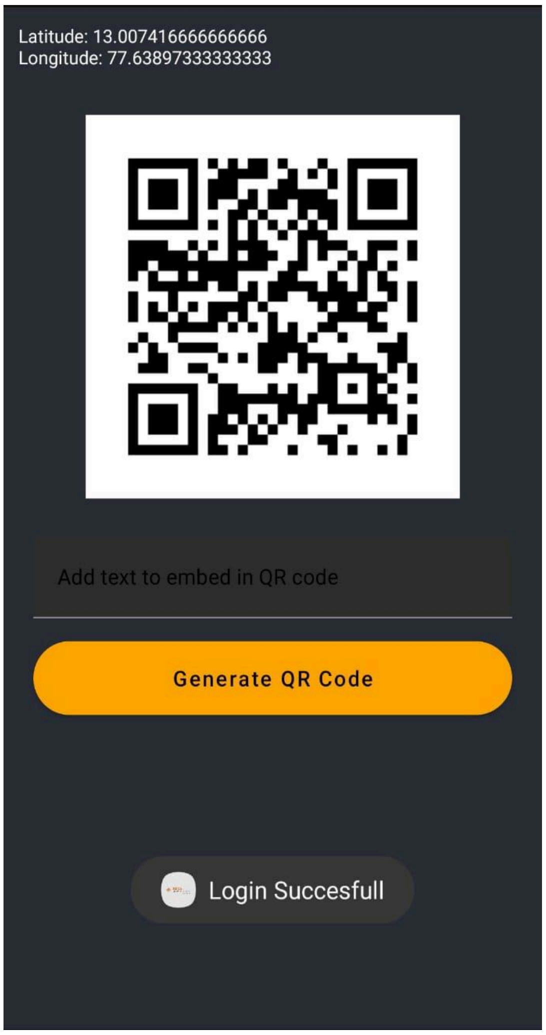
Latitude: 13.0071688
Longitude: 77.6393391

SCAN

Attendance Successful

Distance: 2.1177886 meters




Latitude: 13.007416666666666
Longitude: 77.63897333333333

## Conclusion :

The student login form provides a simple and secure way for students to authenticate into the application using their credentials. Firebase Authentication has been utilized to streamline the authentication process, ensuring confidentiality and data integrity. By integrating email authentication, students can easily register and log in to access the application's resources, fostering a seamless user experience.

The QR code generation feature caters to faculty members, offering a convenient way to generate QR codes based on specific parameters such as locationOverall, these functionalities enhance the application's usability and functionality, catering to the needs of both students and faculty members. By providing a robust authentication mechanism for students and a practical QR code generation feature for faculty, the application facilitates effective communication and resource management within the educational environment. This project demonstrates the power of technology in improving educational workflows and fostering collaboration between students and faculty members.