VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI-590018, KARNATAKA

A Mini Project Report
on

FACE REALITY

Submitted in partial fulfillment of the requirements for the VI Semester of degree of ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi*

by

Nidish G **1RN19IS094**
Nischita CN **1RN19IS096**

Under the Guidance of

Ms. Priyanka M R
Assistant Professor
Department of ISE

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvaradhana Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru-560098

2021-2022

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhana Road, Rajarajeshwari Nagar post,
Channasandra,Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled "**FACE REALITY**" has been successfully completed by **Nidish G (1RN19IS094) and Nischita CN (1RN19IS096),** bonafide students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belgavi** during academic year **2021-2022**. The mini project report has been approved as it satisfies the academic requirements with respect to the Mobile Applications Development laboratory.

Ms. Priyanka M R Mini Project Guide Assistant Professor Department of ISE	**Dr. Suresh L** Professor and HOD Department of ISE RNSIT	**Dr. M K Venkatesha** Principal RNSIT

External Viva

Name of the Examiners	**Signature with Date**
1. _____	**1.** _____
2. _____	**2.** _____

DECLARATION

We, **NIDISH G [USN: 1RN19IS094]**, **NISCHITA CN [USN: 1RN19IS096],** students of VI Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Project entitled "**FACE REALITY**" has been carried out by us and submitted in partial fulfillment of the requirements for the *VI Semester of degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belgavi* during academic year 2021-2022.

Place: Bengaluru

Date:

Nidish G **(1RN19IS094)**

Nischita CN **(1RN19IS096)**

ABSTRACT

The project is based on an augmented reality app which is a software application that integrates digital visual content into the users real-world environment.AR software is used for training, work and consumer applications in many industries including healthcare, public safety, gas and oil.AR is a fast-growing technology and this application implements AR filters on 3D model in real time. Implements a camera module for detecting a face on which the AR filters are implemented. The picture can be clicked through a button which then saves the captured image in gallery which can also be directly accessed through the notification which pops after the camera button is clicked. In general, Augmented Reality apps are written in special 3D programs that allow the developer to tie animation or contextual digital information in the computer program to an augmented reality marker in the real world. Some of the APIs are available across Android and IOS to enable shared AR experiences.

ACKNOWLEDGMENT

The fulfillment and rapture that go with the fruitful finishing of any assignment would be inadequate without the specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

We would like to profoundly thank **Management** of **RNS Institute of Technology** for providing such a healthy environment to carry out this Mobile Application Development Laboratory with Mini Project Work.

We would like to express our thanks to our Principal **Dr. M K Venkatesha** for his support and inspired us towards the attainment of knowledge.

We wish to place on record our words of gratitude to **Dr. Suresh L,** Professor and Head of the Department, Information Science and Engineering, for being the enzyme and master mind behind our Mobile Application Development Laboratory with Mini Project Work.

We would like to express our profound and cordial gratitude to our Mini Project guide**, Ms. Priyanka M R,** Assistant Professor, Department of Information Science and Engineering for their valuable guidance, constructive comments, continuous encouragement throughout the Mini Project Work and guidance in preparing report.

We would like to thank all other teaching and non-teaching staff of Information Science & Engineering who have directly or indirectly helped us to carry out the Mini Project Work.

Also, we would like to acknowledge and thank our parents who are source of inspiration and instrumental in carrying out this Mini Project Work.

NIDISH G

NISCHITA CN

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

iOS - iPhone Operating System

Inc. - Incorporated

OS - Operating System

SDK - Software Development Kit

HTML - Hyper Text Markup Language

SQL - Structured Query Language

API - Application Programming Interface

DVM - Dalvik Virtual Machine

JVM - Java Virtual Machine

IDE - Integrated Development Environment

BMP - Bitmap (Image Format)

RAM - Random Access Memory

XML - Extensible Markup Language

UI - User Interface

LSB - Least Significant Bit

DDMS - Dalvik Debug Monitor Server

Chapter 1

INTRODUCTION TO ANDRIOD

1.1 History

In past mobile phones were used only to make calls but with the introduction of smartphone the mobile phone has evolved to a low powered hand held processing system. This evolution was caused by the operating system for the mobile phones making them smart that have processing and storage of their own. Now the mobile provides numerous functionalities from calling to texting, multimedia sharing, emails, socializing applications, word processor, excel sheets to various multiplayer games and much more.

The operating system for these hand held devices are iOS by Apple Inc., Windows by Windows Inc. and Android by Google. Among the competitors in smartphone operating system industry Android holds the largest market share in terms of units shipped worldwide and number of users.

Android is an open source operating system based on Linux kernel on which applications run on an application framework that controls the activities supported by the libraries and Dalvik virtual machine which compiles and converts all java class files into a single file. There can be number of virtual machines running simultaneously on a single device handling different applications or instances of an application.

Android operating system provides memory management, process management to the applications and services running. Each release of android improved user experience and brought enhanced features. In 2012 Android became the most popular operating system for mobile devices, surpassing Apple's iOS, and, as of 2020, about 75 percent of mobile devices run Android.

1.2 Android Versions

The development of the Android operating system was started in 2003 by Android, Inc. Later on, it was purchased by Google in 2005. The beta version of Android OS was released on November 5, 2007, while the software development kit (SDK) was released on November 12, 2007. The first Android mobile was publicly released with Android 1.0 of the T-Mobile G1

(aka HTC Dream) in October 2008. The first Android version which was released under the numerical order format was Android 10.

Code name	Version numbers	API level	Release date
No codename	1.0	1	September 23, 2008
No codename	1.1	2	February 9, 2009
Cupcake	1.5	3	April 27, 2009
Donut	1.6	4	September 15, 2009
Eclair	2.0 - 2.1	5 - 7	October 26, 2009
Froyo	2.2 - 2.2.3	8	May 20, 2010
Gingerbread	2.3 - 2.3.7	9 - 10	December 6, 2010
Honeycomb	3.0 - 3.2.6	11 - 13	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	October 18, 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	July 9, 2012
KitKat	4.4 - 4.4.4	19 - 20	October 31, 2013
Lollipop	5.0 - 5.1.1	21- 22	November 12, 2014
Marshmallow	6.0 - 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 - 7.1.2	25	October 4, 2016
Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018
Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020

Table 1.2.1 Android Versions

1.3 Android Architecture

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

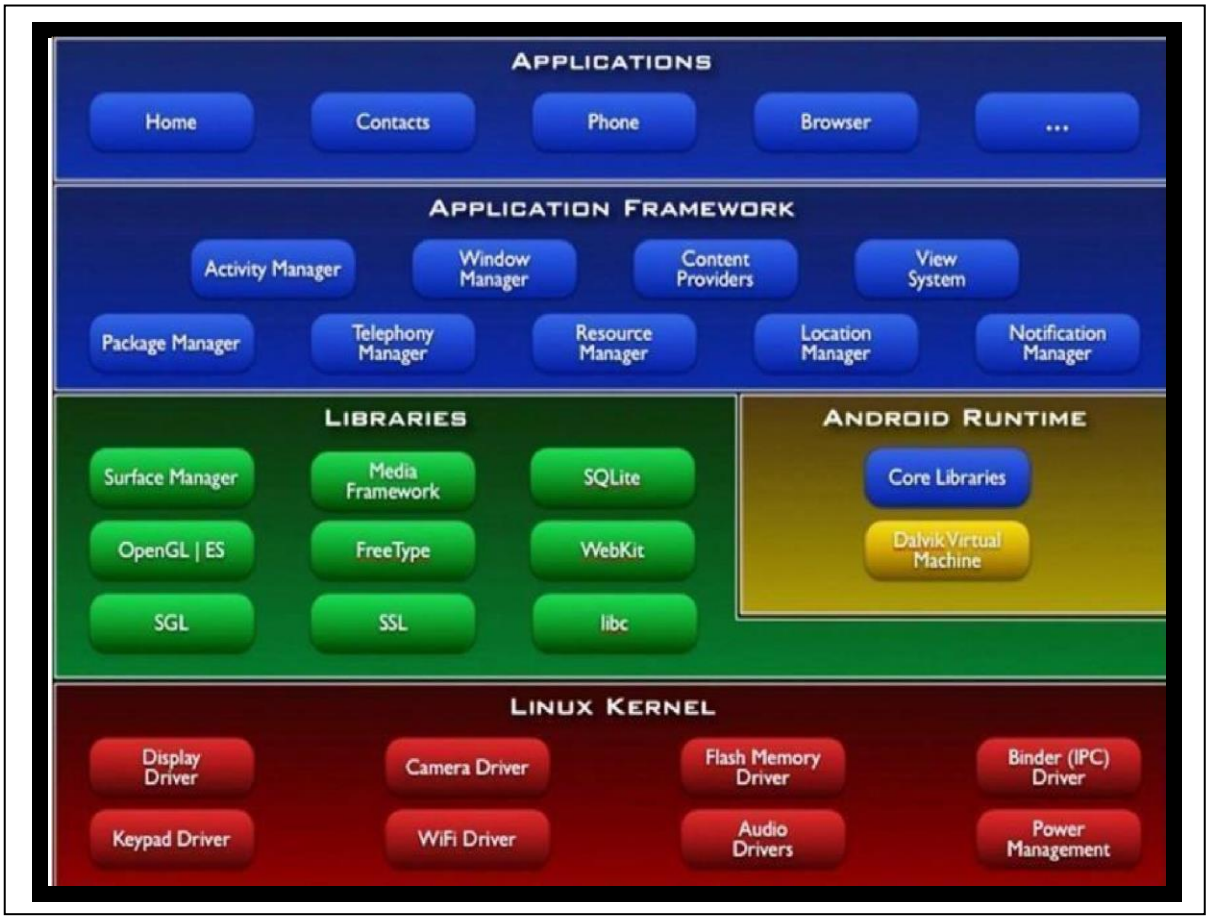

Figure 1.3.1 Android Architecture diagram

- **Linux Kernel:** This is the layer at the very bottom of the Android architecture. All other layers run on top of the Linux kernel and rely on this kernel to interact with the hardware. This layer contains all the essential hardware drivers which help to control and communicate with the hardware. It provides the basic functionality like Process Management, Memory Management and Device Management like Camera, Display, Flash etc.

- **Libraries:** This is a set of common functions of the application framework that enables the device to handle different types of data. Some of the most important set of libraries that are included are – Web kit which is the browser engine to display HTML, OpenGL used to

render 2- D or 3-D graphics on to the screen, SQLite which is a useful repository for storing and sharing of application data.

A summary of some key core Android libraries available to the Android developer is as follows

- android.app − Provides access to the application model and is the cornerstone of all Android applications.
- android.content − Facilitates content access, publishing and messaging between applications and application components.
- android.database − Used to access data published by content providers and includes SQLite database management classes.
- android.opengl − A Java interface to the OpenGL ES 3D graphics rendering API
- android.os − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- android.text − Used to render and manipulate text on a device display.
- android.view − The fundamental building blocks of application user interfaces.
- android.widget − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- android.webkit − A set of classes intended to allow web-browsing capabilities to be built into applications.

- **Android Runtime:** The Android runtime mainly consist of the Dalvik Virtual Machine (DVM). DVM is very much like the standard Java Virtual Machine (JVM) except that it is optimized for mobile devices that have low processing power and low memory. DVM generates a.dex file from the .class file at compile time and provides higher efficiency in low resources devices. Each application has its own process and an instance of DVM. Android runtime also provides core libraries that enable the Android developers to create applications using the Java language.
- **Application Framework:** The Android runtime mainly consist of the Dalvik Virtual Machine (DVM). DVM is very much like the standard Java Virtual Machine (JVM) except that it is optimized for mobile devices that have low processing power and low memory. DVM generates a.dex file from the .class file at compile time and provides higher

efficiency in low resources devices. Each application has its own process and an instance of DVM. Android runtime also provides core libraries that enable the Android developers to create applications using the Java language.

- **Applications:** This is the topmost layer in the architecture and the layer where the application that we develop fits in. This layer provides several pre-installed applications that are default for certain things like Contacts Books, Browser etc.

1.4 Android Studio Installation

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools.

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and GitHub integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

PROCEDURE TO BE FOLLOWED TO DOWNLOAD AND INSTALL ANDROID STUDIO:

STEP 1 : Android Studio and the Software Development Kit can be downloaded directly from any web browser using the below link.

https://developer.android.com/studio

STEP 2 : Android Studio is available for Mac, Windows, and Linux desktop platforms.

Windows

To install Android Studio on Windows, proceed as follows:

i. If you downloaded an .exe file (recommended), double-click to launch it.If you downloaded a .zip file, unpack the ZIP, copy the android-studio folder into your Program Files folder, and then open the android-studio > bin folder and launch studio64.exe (for 64-bit machines) or studio.exe (for 32-bit machines).

ii. Follow the setup wizard in Android Studio and install any SDK packages that it recommends.

Mac

To install Android Studio on your Mac, proceed as follows:

i. Launch the Android Studio DMG file.
ii Drag and drop Android Studio into the Applications folder, then launch it.
iii Select if you want to import previous Android Studio settings, then press OK
iv. The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

Linux

To install Android Studio on Linux, proceed as follows:

i. Unpack the .zip file you downloaded to an appropriate location for your applications, such as within /usr/local/ for your user profile, or /opt/ for shared users. If you're using a 64-bit version of Linux, make sure you first install the required libraries for 64-bit machines.
ii. To launch Android Studio, open a terminal, navigate to the android-studio/bin/ directory and execute studio.sh.
iii. Select whether you want to import previous Android Studio settings or not, then click OK.
iv. The Android Studio Setup Wizard guides you through the rest of the setup, which includes downloading Android SDK components that are required for development.

Chapter 2

INTRODUCTION TO PROJECT

2.1 Overview of the Project

Augmented Faces permit the application to naturally distinguish various regions of an individual's face, and utilize those areas to overlay resources, for example, surfaces and models in a way that appropriately matches the contours and regions of an individual face. ARCore is a stage for building Augmented reality applications on Android. Augmented Face is a subsystem of ARCore that permits your application to:

- Naturally, recognize various areas of any individual's identified face, and utilize those regions to overlay resources, for example, surfaces and models in a way that appropriately matches the contours and regions of an individual face.
- Utilize the 468-point face mesh that is given by ARCore to apply a custom texture over a distinguished face.

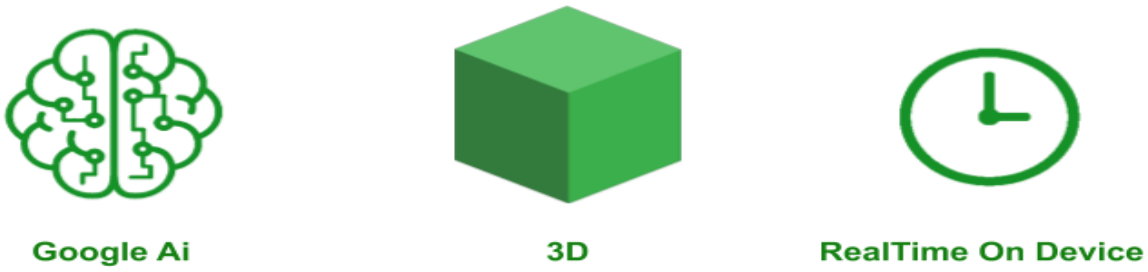

Figure 2.1.1 Google AI and Real time 3D

2.2 Aim of the Project

To appropriately overlay textures and 3D models on an identified face, ARCore provides detected regions and an augmented face mesh. This mesh is a virtual depiction of the face and comprises the vertices, facial regions, and the focal point of the user's head. At the point when a user's face is identified by the camera, ARCore performs the following steps to generate the augmented face mesh, as well as center and region poses:

- It distinguishes the center pose and a face mesh.

- The center pose, situated behind the nose, is the actual center point of the user's head (in other words, inside the skull).
- The face mesh comprises of many vertices that make up the face and is characterized relative to the center pose.

The AugmentedFace class utilizes the face mesh and center pose to distinguish face regions present on the client's face. These regions are:

- Right brow (RIGHT_FOREHEAD)
- Left temple (LEFT_FOREHEAD)
- Tip of the nose (NOSE_TIP)

Augmented faces don't require uncommon or special hardware, such as a depth sensor. Rather, it uses the phone's camera and machine learning to provide three snippets of data:

- Generates a Face Mesh: a 468 points dense 3D face mesh, which allows you to pan detailed textures that accurately follow facial moments.
- Recognizes the Pose: points on a person's face, anchored based on the generated Face Mesh, which is useful for placing effects on or close to the temple and nose.
- Overlays and position textures and 3D models based on the face mesh generated and recognized regions.

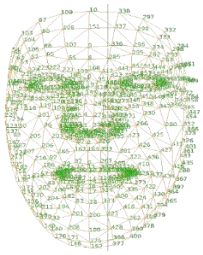

Figure 2.2.1 3-D Mesh

Chapter 3

SYSTEM DESIGN

3.1 System Requirements

3.1.1 Hardware Requirements:

- Processor: Core i5 or above
- RAM: 8GB or more
- Hard Disk: 100GB or more

3.1.2 Software Requirements:

- Operating System: Windows 7 or above
- IDE: Android Studio
- API Level: 24 or above

3.2 User Interface

XML Code for the initial UI screen:

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/constraintLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.miniproject.facereality.augmentedfaces.AugmentedFacesActivity">


  <android.opengl.GLSurfaceView
      android:id="@+id/surfaceview"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="top"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
```

Figure 3.2.1 User Interface

```
    <Button
        android:id="@+id/button_screenshot"
        android:layout_width="75dp"
        android:layout_height="75dp"
        android:background="@drawable/round_button"
        android:textAlignment="center"
        android:textColor="#000"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.957" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="410dp"
        android:layout_height="138dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0"
        app:srcCompat="@drawable/rectangle" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figure 3.2.2 User Interface Code

Chapter 4

IMPLEMENTATION

4.1 Reference Terminologies

- **Trackable**: A Trackable is an interface which can be followed by ARCore and something from which Anchors can be connected to.
- **Anchor**: It describes a fixed location and orientation in the real world. To stay at a fixed location in physical space, the numerical description of this position will update as ARCore's understanding of the space improves. Anchors are hashable and may for example be used as keys in HashMaps.
- **Pose**: At the point when you need to state wherein the scene you need to put the object and you need to specify the location in terms of the scene's coordinates. The Pose is the means by which you state this.
- **Session:** Deals with the AR framework state and handles the session lifecycle. This class offers the primary passage to the ARCore API. This class permits the user to make a session, configure it, start or stop it and, above all, receive frames that allow access to the camera image and device pose.
- **Textures:** Textures are especially helpful for Augmented Faces. This permits you to make a light overlay that lines up with the locales of the identified face(s) to add to your experience.
- **ArFragment:** ARCore utilizes an ArFragment that provides a lot of features, for example, plane finding, permission handling, and camera set up. You can utilize the fragment legitimately in your activity, however at whatever point you need custom features, for example, Augmented Faces, you should extend the ArFragment and set the proper settings. This fragment is the layer that conceals all the compound stuff (like OpenGL, rendering models, etc) and gives high-level APIs to load and render 3D models.
- **ModelRenderable:** ModelRenderable renders a 3D Model by attaching it to a Node.
- **Sceneform SDK:** Sceneform SDK is another library for Android that empowers the quick creation and mix of AR experiences in your application. It joins ARCore and an amazing physically-based 3D renderer.

4.2 Manifest File

Every app project must have an AndroidManifest.xml file (with precisely that name) at the root of the project source set. The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.

The permissions that the app needs in order to access protected parts of the system or other apps are within this file. It also declares any permissions that other apps must have if they want to access content from this app.

```
<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.miniproject.facereality">

  <uses-permission android:name="android.permission.CAMERA"/>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <!-- Limits app visibility in the Google Play Store to ARCore supported devices
       (https://developers.google.com/ar/devices). -->
  <uses-feature android:name="android.hardware.camera.ar" android:required="true"/>
  <uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

Figure 4.2.1 Manifest File

```
  <uses-feature android:name="android.hardware.camera.ar" android:required="true"/>
  <uses-feature android:glEsVersion="0x00020000" android:required="true" />

  <application
      android:allowBackup="false"
      android:icon="@drawable/face_reality_logo"
      android:label="@string/app_name"
      android:theme="@style/AppTheme"
      android:usesCleartextTraffic="false"
      android:requestLegacyExternalStorage="true"
      tools:ignore="GoogleAppIndexingWarning">

    <activity
        android:name="com.miniproject.facereality.augmentedfaces.AugmentedFacesActivity"
        android:configChanges="orientation|screenSize"
        android:exported="true"
        android:theme="@style/Theme.AppCompat.NoActionBar"
        android:screenOrientation="locked">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
    <!-- Indicates whether "Google Play Services for AR" (ARCore) is "required" or "optional". -->
    <meta-data android:name="com.google.ar.core" android:value="required" />
    <provider
        android:name="androidx.core.content.FileProvider"
        android:authorities="${applicationId}.provider"
        android:exported="false"
        android:grantUriPermissions="true">
      <meta-data
          android:name="android.support.FILE_PROVIDER_PATHS"
          android:resource="@xml/provider_paths" />
    </provider>
  </application>
```

Figure 4.2.2 Manifest File Code

4.3 Permission Check

At the time of first installation the app requests for access for media and photo of the device.

```
private static final int REQUEST_EXTERNAL_STORAGE = 1;
private static final String[] PERMISSION_STORAGE = {
        Manifest.permission.READ_EXTERNAL_STORAGE,
        Manifest.permission.WRITE_EXTERNAL_STORAGE,
        Manifest.permission.CAMERA,
};



public static void verifyStoragePermission(Activity activity) {
  int permission = ActivityCompat.checkSelfPermission(activity, Manifest.permission.WRITE_EXTERNAL_STORAGE);

  if (permission != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(
            activity,
            PERMISSION_STORAGE,
            REQUEST_EXTERNAL_STORAGE);
  }
}
```

Figure 4.3.1 Permission Check Code

App permissions help support user privacy by protecting access to the following: Restricted data, such as system state and a user's contact information. Restricted actions, such as connecting to a paired device and recording audio and accessing media files.Upon granting the required permissions the app runs successfully.

4.4 Notifications

A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open an app or take an action directly from the notification.

The project includes the implementation of notification when the camera button is clicked, tapping on the notification will take the user directly to the gallery application where the captured image will be stored.

```
private void createNotificationChannel() {
  // Create the NotificationChannel, but only on API 26+ because
  // the NotificationChannel class is new and not in the support library
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    CharSequence name = getString(R.string.channel_name);
    String description = getString(R.string.channel_description);
    int importance = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
    channel.setDescription(description);
    // Register the channel with the system; you can't change the importance
    // or other notification behaviors after this
    NotificationManager notificationManager = getSystemService(NotificationManager.class);
    notificationManager.createNotificationChannel(channel);
  }
}

//random int gen
public int gen() {
  Random r = new Random( System.currentTimeMillis() );
  return ((1 + r.nextInt(2)) * 10000 + r.nextInt(10000));
}

@Override
```

Figure 4.4.1 Notification

```
Intent intent = new Intent();
intent.setAction(android.content.Intent.ACTION_VIEW);
intent.setType("image/*");
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
//intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
//intent.setDataAndType(Uri.fromFile(file), "image/*");

PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, PendingIntent.FLAG_IMMUTABLE);

NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);

NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_launcher)
        .setContentTitle("Image Saved Successfully")
        .setContentText("Image saved in Pictures folder. File name starts with FaceReality_timestamp")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(pendingIntent)
        .setAutoCancel(true);
int notificationId = gen();
notificationManager.notify(notificationId, builder.build());
```

Figure 4.4.2 Notification Code

4.5 Toast Message

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

The application implements a toast message which pops up everytime the camera button is clicked to display the message saying that the picture has been captured.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  verifyStoragePermission(this);
  createNotificationChannel();
  surfaceView = findViewById(R.id.surfaceview);
  btnScreenshot = findViewById(R.id.button_screenshot);
  final MediaPlayer mp =MediaPlayer.create(this,R.raw.shutter);
  btnScreenshot.setOnClickListener(view -> {
    Log.v("nid", "pan button clicked");
    //used to make toast
    Toast.makeText(getBaseContext(),"Image taken",Toast.LENGTH_SHORT).show();
    //used to take screenshot
    printOptionEnable = true;
    //used to play the shutter audio
    mp.start();



  });
  displayRotationHelper = new DisplayRotationHelper(/*context=*/ this);

  // Set up renderer.
  surfaceView.setPreserveEGLContextOnPause(true);
  surfaceView.setEGLContextClientVersion(2);
  surfaceView.setEGLConfigChooser(8, 8, 8, 8, 16, 0); // Alpha used for plane blending.
  surfaceView.setRenderer(this);
  surfaceView.setRenderMode(GLSurfaceView.RENDERMODE_CONTINUOUSLY);
  surfaceView.setWillNotDraw(false);

  installRequested = false;
}
```

Figure 4.5.1 Toast Code

4.6 Attachment of models and assests

Augmented Reality is adding additional information or objects to real world settings. Assets are added to the 3D models in real time .

```
@Override
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
  GLES20.glClearColor(0.1f, 0.1f, 0.1f, 1.0f);

  // Prepare the rendering objects. This involves reading shaders, so may throw an IOException.
  try {
    // Create the texture and pass it to ARCore session to be filled during update().
    backgroundRenderer.createOnGlThread(/*context=*/ this);
    augmentedFaceRenderer.createOnGlThread(this, "models/freckles.png");
    augmentedFaceRenderer.setMaterialProperties(0.0f, 1.0f, 0.1f, 6.0f);
    noseObject.createOnGlThread(/*context=*/ this, "models/nose.obj", "models/nose_fur.png");
    noseObject.setMaterialProperties(0.0f, 1.0f, 0.1f, 6.0f);
    noseObject.setBlendMode(ObjectRenderer.BlendMode.AlphaBlending);
    rightEarObject.createOnGlThread(this, "models/forehead_right.obj", "models/ear_fur.png");
    rightEarObject.setMaterialProperties(0.0f, 1.0f, 0.1f, 6.0f);
    rightEarObject.setBlendMode(ObjectRenderer.BlendMode.AlphaBlending);
    leftEarObject.createOnGlThread(this, "models/forehead_left.obj", "models/ear_fur.png");
    leftEarObject.setMaterialProperties(0.0f, 1.0f, 0.1f, 6.0f);
    leftEarObject.setBlendMode(ObjectRenderer.BlendMode.AlphaBlending);

  } catch (IOException e) {
    Log.e(TAG, "Failed to read an asset file", e);
  }
}
```

```
    float scaleFactor = 1.0f;

    // Face objects use transparency so they must be rendered back to front without depth write.
    GLES20.glDepthMask(false);

    // Each face's region poses, mesh vertices, and mesh normals are updated every frame.

    // 1. Render the face mesh first, behind any 3D objects attached to the face regions.
    float[] modelMatrix = new float[16];
    face.getCenterPose().toMatrix(modelMatrix, 0);
    augmentedFaceRenderer.draw(
        projectionMatrix, viewMatrix, modelMatrix, colorCorrectionRgba, face);

    // 2. Next, render the 3D objects attached to the forehead.
    face.getRegionPose(RegionType.FOREHEAD_RIGHT).toMatrix(rightEarMatrix, 0);
    rightEarObject.updateModelMatrix(rightEarMatrix, scaleFactor);
    rightEarObject.draw(viewMatrix, projectionMatrix, colorCorrectionRgba, DEFAULT_COLOR);

    face.getRegionPose(RegionType.FOREHEAD_LEFT).toMatrix(leftEarMatrix, 0);
    leftEarObject.updateModelMatrix(leftEarMatrix, scaleFactor);
    leftEarObject.draw(viewMatrix, projectionMatrix, colorCorrectionRgba, DEFAULT_COLOR);

    // 3. Render the nose last so that it is not occluded by face mesh or by 3D objects attached
    // to the forehead regions.
    face.getRegionPose(RegionType.NOSE_TIP).toMatrix(noseMatrix, 0);
    noseObject.updateModelMatrix(noseMatrix, scaleFactor);
    noseObject.draw(viewMatrix, projectionMatrix, colorCorrectionRgba, DEFAULT_COLOR);
  }
```

Figure 4.6.1 Attachment of Models and assets Code

4.7 Capturing Image

On clicking the camera button screenshot is taken which captures the view on the screen . Provides a mechanisms to issue pixel copy requests to allow for copy operations from surface to Bitmap. A Surface is generally created by or from a consumer of image buffers (such as a SurfaceTexture), and is handed to some kind of producer (such as OpenGL, or CameraDevice) to draw into.

```
try {
  if (printOptionEnable) {
    printOptionEnable = false ;
    Log.i("nid", "printOptionEnable if condition:" + printOptionEnable);
    int w = width_surface ;
    int h = height_surface  ;

    Log.i("nid", "w:"+w+"-----h:"+h);

    int[] b =new int[(int) (w*h)];
    int[] bt =new int[(int) (w*h)];
    IntBuffer buffer=IntBuffer.wrap(b);
    buffer.position(0);
    GLES20.glReadPixels(0, 0, w, h,GLES20.GL_RGBA,GLES20.GL_UNSIGNED_BYTE, buffer);
    for(int i=0; i<h; i++)
    {
      //remember, that OpenGL bitmap is incompatible with Android bitmap
      //and so, some correction need.
```

Figure 4.7.1 Screenshot Copying code

```
  for(int j=0; j<w; j++)
  {
    int pix=b[i*w+j];
    int pb=(pix>>16)&0xff;
    int pr=(pix<<16)&0x00ff0000;
    int pix1=(pix&0xff00ff00) | pr | pb;
    bt[(h-i-1)*w+j]=pix1;
  }
}
Bitmap inBitmap = null ;
if (inBitmap == null || !inBitmap.isMutable()
        || inBitmap.getWidth() != w || inBitmap.getHeight() != h) {
  inBitmap = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
}
//Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
inBitmap.copyPixelsFromBuffer(buffer);
inBitmap = Bitmap.createBitmap(bt, w, h, Bitmap.Config.ARGB_8888);
```

Figure 4.7.2 Screenshot Copying code

4.8 Sending Image to the pictures folder

To provide a more enriched user experience, many apps allow users to contribute and access media that's available on an external storage volume. The framework provides an optimized index into media collections, called the *media store*, that allows for retrieving and updating these media files more easily. Even after the app is uninstalled, these files remain on the user's device.

```
String timeStamp=String.valueOf(mytimestamp);
//String myfile="nid"+timeStamp+".jpeg";

ContextWrapper cw = new ContextWrapper(getApplicationContext());
File directory = cw.getDir("imageDir", Context.MODE_PRIVATE);
String fileName = "FaceReality_"+timeStamp + ".jpg";
File file = new File(directory, fileName);
if (!file.exists()) {
  Log.d("nid", file.toString());
  FileOutputStream fos;
  try {
    fos = new FileOutputStream(file);
    inBitmap.compress(Bitmap.CompressFormat.JPEG, 90, fos);
    fos.flush();
    fos.close();
    MediaStore.Images.Media.insertImage(getContentResolver(), file.getAbsolutePath(), file.getName(), file.getName());
```

Figure 4.8.1 Sending Images to pictures folder

Chapter 5

TESTING

Software testing in an essential phase in the development life cycle of an application. Testing ensures that the developed system meets its functional and non-functional requirements. Two important terms in software testing are Verification and Validation. Verification is the process of evaluating work-products like requirement specs, design specs and test cases etc. of different development phases to make sure that they meet the requirements for that phase. It ensures that the system is built in the right way. Whereas Validation is the process of evaluating the software at the end of the development phase to make sure that it meets the business requirements. It is used to make sure that the product fulfils its intended use and that the end product is built right.

One of the most important tools to test and debug an Android app is the Dalvik debug monitor server (DDMS) that is part of the Android framework. DDMS helps you to debug your code as it prints errors, warning and other information from your code. It also provides stack traces for exceptions on the Logcat output.

Various other testing strategies have been adopted to make sure the correctness of the Face Reality app. They are discussed in this chapter.

5.1 Unit Testing

Unit testing is a strategy in software testing where individual components in a software are tested for correctness.

Following are the unit testing test cases for the Face Reality app:

Table 5.1 Main Screen Unit Testing

Sl. No	Test Case	Action	Result
1	On clicking the Shutter button	Picture must be captured	Pass
2	On clicking the Shutter button	Shutter sound must be heard.	Pass

3	On clicking the Shutter button	Toast must be displayed	Pass
4	On clicking the Shutter button	Notification must appear	Pass

5.2 Integration Testing

Integration testing is a strategy in software testing where different modules are combined and test to make sure they work together correctly. It is done when the components are unit test and the main objective is to test the interfaces between different components.

Following are the integration testing test cases for the Face Reality app:

Table 5.2 Integration Testing

Sl.No	Test Case	Action	Result
1	On clicking the shutter button in the main screen	Notification must appear.	Pass
2	On clicking the notification	User must be redirected to the gallery app where the image is stored.	Pass

5.3 System Testing

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

Following are the system testing test cases for the Face Reality app:

Table 5.3 System Testing

Sl. No	Test Case	Action	Result
1	On clicking the app icon	The main activity is launched.	Pass
2	If permission is denied or not yet accepted	Permission dialog box is prompted.	Pass

Chapter 6

RESULTS

All the Activities provided in the application and its operations have been presented in as snapshots. A detailed view of all the snapshots of the application is given in this section.

6.1 Main Screen Interface

The Main screen consist of the camera UI with a button to click the picture which then saves the photo .When a picture is clicked, a toast message is shown with a shutter sound. Notification is sent to the users smartphone on capturing the picture.

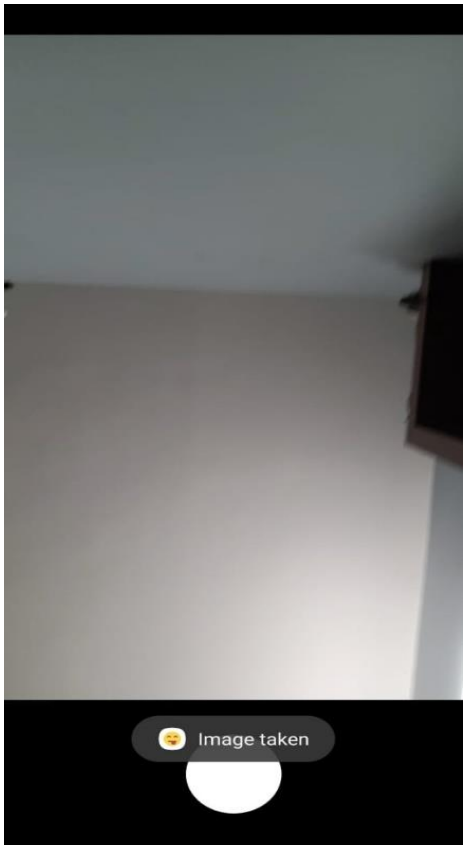

Figure 6.1.1 Main Activity

6.2 Notifications

Message notification is displayed on capturing the image which displays a popup to the user to choose an application to view the image captured.

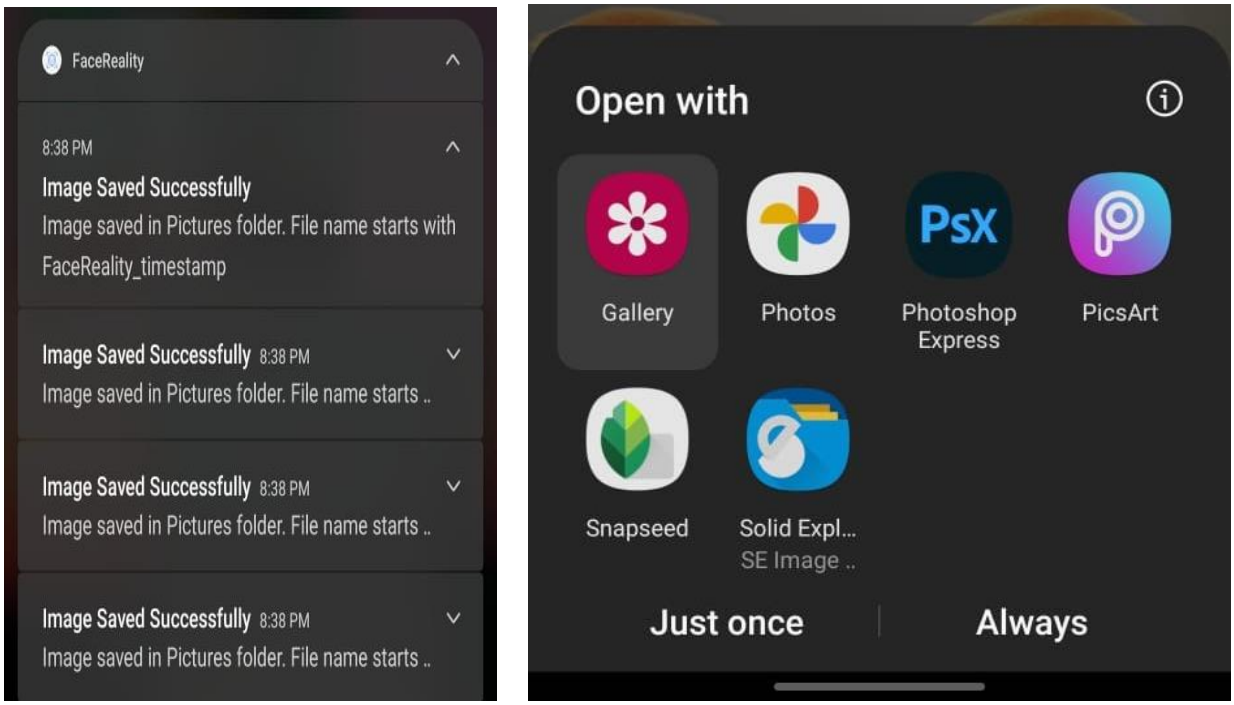

Figure 6.2.1 Notification screenshots

6.3 The 3D Models in Real Time

AR filters implemented on 3D models in real time has been shown in the following images.

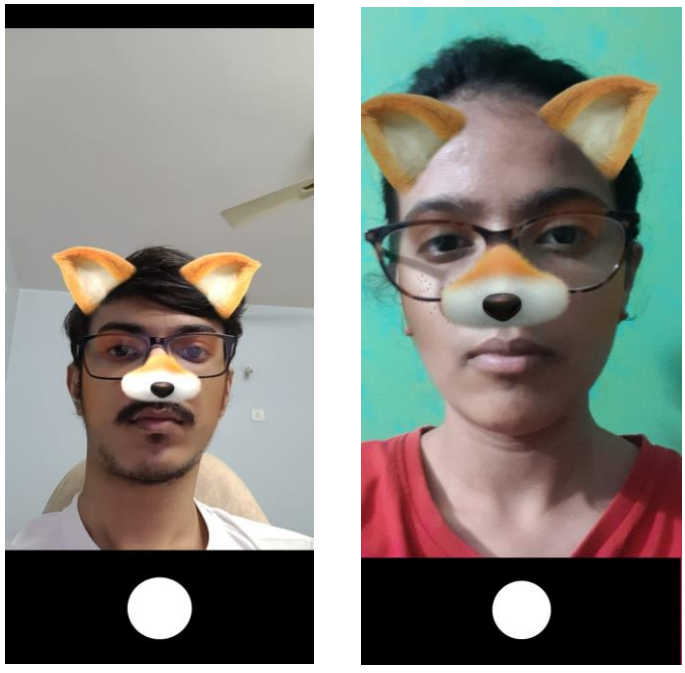

Figure 6.3.1 3-D Models in Real time

6.4 Pictures saved in gallery

Below are the pictures viewed from the gallery once they are captured through the application**.**

Figure 6.4.1 Pictures saved in Gallery

6.5 Permissions

On installing the application prompt displays which asks for user permission to take pictures and to access media files.

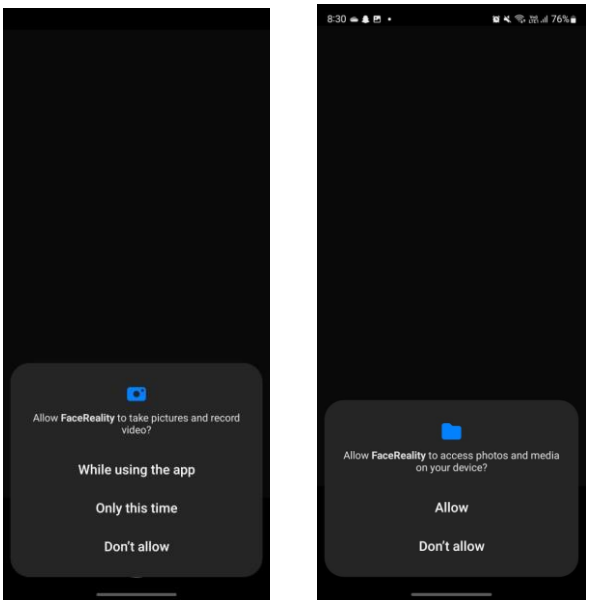

Figure 6.5.1 Permissions

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

Face Reality is a really an interesting application which implements Augmented Reality technology and 3D models which in real time renders the whole model on a human face with help of 2D image and using AI to generate a 400-point 3D mesh which then intern is used to attach he model without the use of depth sensing technology required. The application also provides the feature of saving the image captured into the gallery. Users can also be redirected to the gallery on tapping the notification which pops up once the image is captured.

7.2 Future Enhancements

- Application can be improvised by implementing the same for capturing videos.
- Implementing a gallery app within the application for storing all the media files.
- Creating and applying more face filters.
- Making the application more dynamic in nature.

REFERENCES

- https://www.tutorialspoint.com/android/android_architecture.htm
- https://www.stackoverflow.com
- https://www.geeksforgeeks.com
- https://developers.google.com/ar/develop/java/quickstart