VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JnanaSangama, Belgaum-590018

A Mobile Application Development Laboratory Mini Project Report

ON

"Event Ticket Booking System"

Submitted in Partial fulfillment of the Requirements for VI of the Degree of

Bachelor of Engineering in Computer Science & Engineering

By

SUNIL KUMAR P (1CR21CS414)

Under the Guidance of,

Prof. Poonam Tijare, Assistant Professor, Dept. of CSE

Prof. Kartheek G C R., Assistant Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI, BANGALORE-56003

CMR INSTITUTE OF TECHNOLOGY

#132, AECS LAYOUT, IT PARK ROAD, KUNDALAHALLI,BANGALORE-560037

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mobile Application Development project work entitled **"Event Ticket Booking System"** has been carried out by **SUNIL KUMAR P (1CR21CS414)** bonafide students of CMR Institute of Technology in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2022- 2023**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. This Mobile Application Development project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Prof. Poonam Tijare **Assistant Professor** **Dept. of CSE, CMRIT**	**Dr. Shreekanth M Prabhu** **Professor & Head** **Dept. of CSE, CMRIT**

External Viva

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

DECLARATION

We, the students of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the mini-project work entitled "**Event Ticket Booking System**" has been successfully completed under the guidance of **Prof. Poonam Tijare,** Computer Science and Engineering Department, CMR Institute of technology, Bangalore. This work is submitted in partial fulfillment of the requirements toward the VI semester Computer Graphics Laboratory with Mini-Project (18CSL67) for the degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2022 - 2023. Further the matter embodied in the project report has not been submitted previously by anybody toward the requirements of Computer Graphics Laboratory with Mini-Project (18CSL67) in the VI degree or diploma to any university.

Place: Bengaluru

Date: 01-07-2023

Team members: **Signature**

SUNIL KUMAR P (1CR21CS414) _____

ABSTRACT

The Event Ticket Booking System is a mobile application developed using Android Studio and Kotlin programming language, with the backend powered by Firebase Realtime Database. The application aims to provide users with a convenient platform to browse and book tickets for various events. The Firebase Realtime Database ensures real-time updates and data synchronization, allowing users to make informed decisions while booking tickets. The system incorporates features such as user registration and login, event listing and details, ticket selection, booking history, and real-time updates. The project report discusses the objective, features, technology stack, architecture, database design, challenges faced, and future enhancements of the Event Ticket Booking System. The system successfully achieves its goal of providing a user-friendly interface for event ticket bookings, enhancing the overall user experience.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out my final year project

I have a great pleasure in expressing my deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bangalore, for his constant encouragement.

I would like to thank **Dr. Shreekanth M Prabhu,**Professor and Head, Department of Computer Science and Engineering, CMRIT, Bangalore, who has been a constant support and encouragement throughout the course of this project.

I consider it a privilege and honor to express my sincere gratitude to my guide **Prof. Poonam Tijare , Assistant Professor,** Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of this review.

I also extend my thanks to all the faculty of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I would like to thank my parents and friends for all their moral support they have given me during the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Event Ticket Booking System is a comprehensive Android application developed using Android Studio and Kotlin. It offers a range of features including user registration, event browsing, ticket booking, event hosting, profile management, and logout.

The app leverages the power of Firebase Realtime Database for efficient and reliable data storage and retrieval. With its intuitive user interface and seamless functionality, the system provides a user-friendly experience for booking event tickets.

Whether users are looking to attend an event or organizers are seeking a platform to manage their events, this app serves as a reliable and efficient solution for all their ticket booking needs.

1.1 Objectives:

Objectives of a Event Ticket Booking Syatem is:

1. Implementing Core Functionality: The focus should be on implementing the core features required for an event ticket booking app, including event browsing, ticket selection, and booking confirmation. This will help beginner developers gain hands-on experience in developing functional components.

2. User Interface Design: The project should involve designing a simple and intuitive user interface. Beginner developers can learn about layout managers, views, and basic styling to create a visually appealing and user-friendly app.

3. Data Management: Implementing basic data management, such as storing event information and user bookings locally using Firebase.

Chapter 2

System Requirements

2.1 Recommended Operating Systems

- Windows: 64-bit version of Windows (8,10 or11).
- Mac : version 10.14 or later.
- Linux : Ubuntu 16.04 or higher.

2.2 Minimum System Requirements

- Operating systems: Windows 10,11 MacOS X.
- Processors : Intel i5, ARM based processors eg: M1,M2.
- Size :>120 GB Hard disk space.
- RAM: minimum 8 GB.
- GPU : any integrated or dedicated GPU with minimum 1024MB.

2.3 Minimum Mobile Requirements

- Android version > 5.0
- Ram : minimum 512 MB
- Size : minimum 10MB free space
- Screen Size : 5 inches

CHAPTER 3

DESIGN

3.1 ER Diagram

Event Ticket Booking System has 2 main entities which are user and event. And they have the relation so that 1 user can host many events and 1 event can be hosted by a single user. And 1 user can buy only one ticket of the event and 1 event can many users.

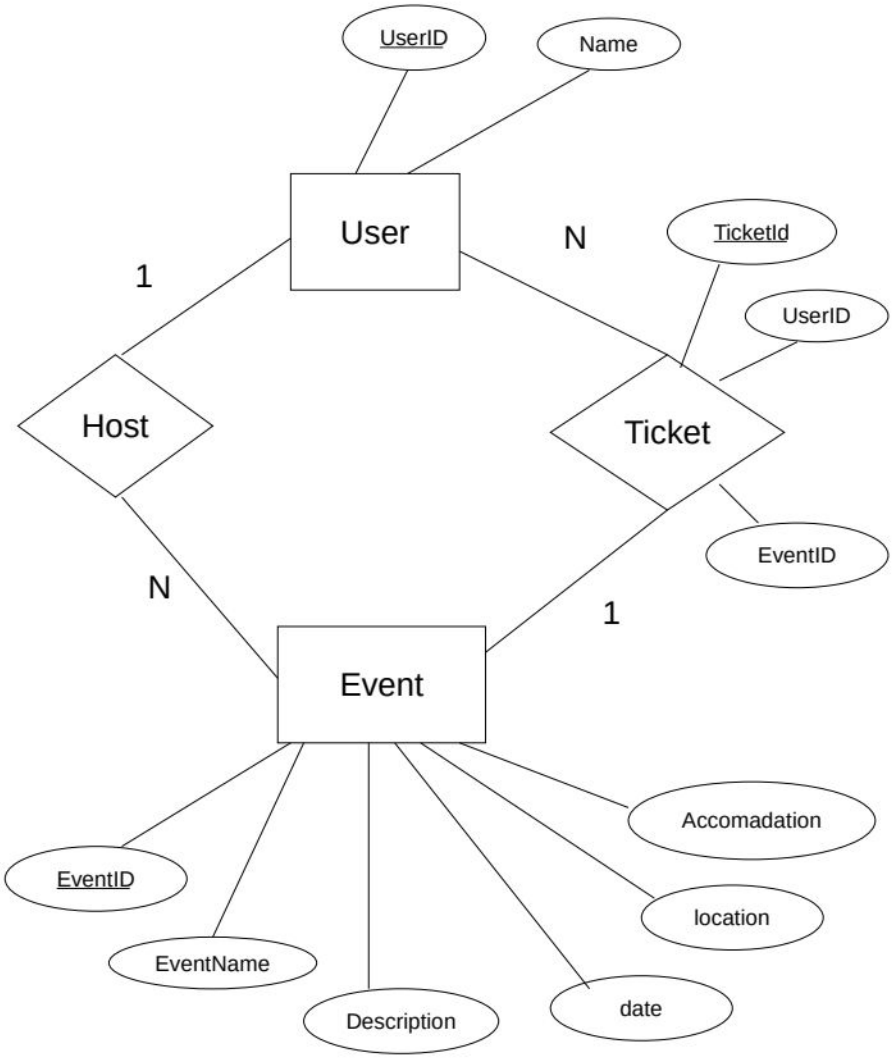

Fig 3.1 : ER Diagram.

3.2 Schema Diagram

Event Ticket Booking System has 3 tables based and the ER diagram.

Event tables userid inherits its values from User table UserID.

Ticket tables userid inherits its values from User table UserID.

Ticket tables Eventid inherits its values from Event table EventID.

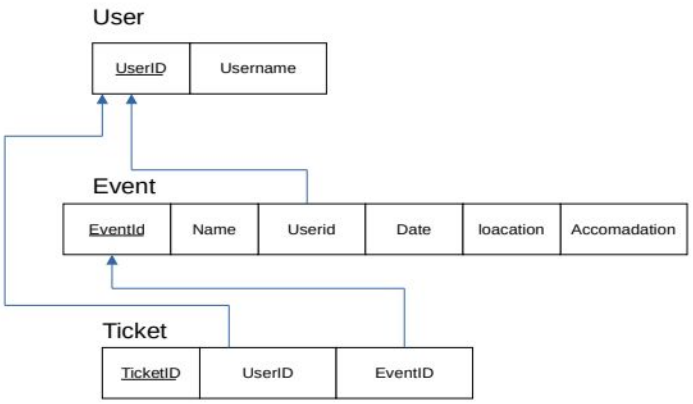

Fig 3.2 : Schema Diagram.

3.3 Different Layout Designs

activity_main.xml is the launcher activity for user login. It consists of a LinearLayout with vertical orientation, containing an ImageView for the logo, two EditText fields for email and password input, a login Button, and a TextView for signup.

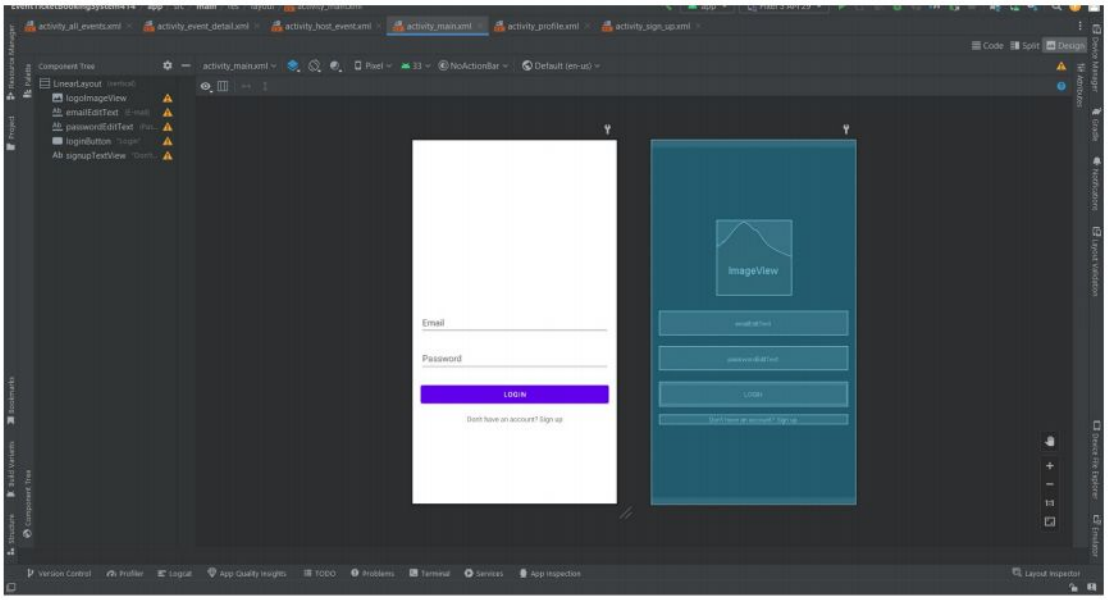

Fig 3.3 : activity_main.xml

activity_sign_up.xml consists of a LinearLayout with vertical orientation, three EditText fields for Full name, email and password, a login Button, and a TextView for login.

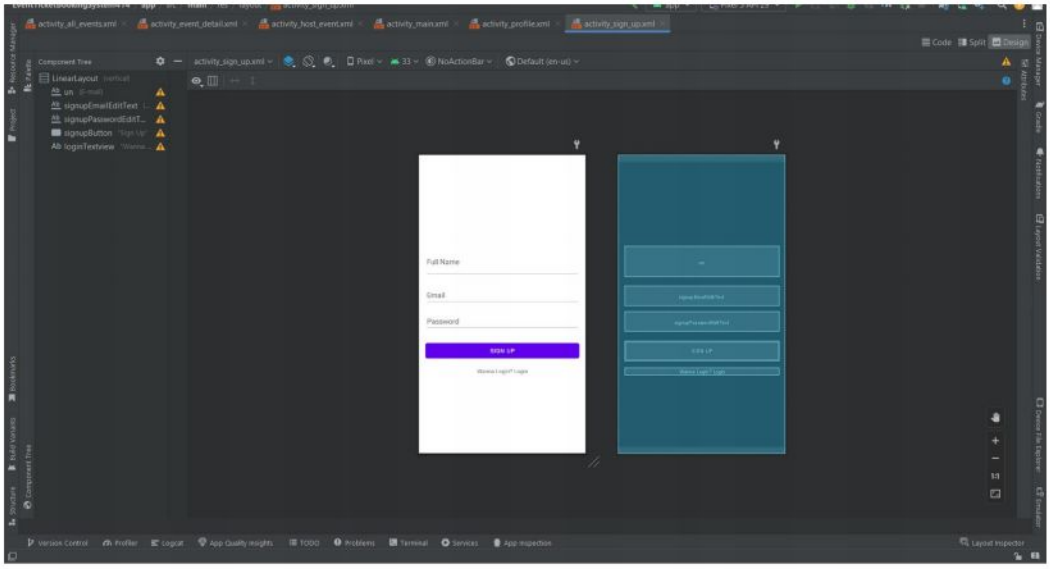

Fig 3.4 : activity_sign_up.xml

activity_all_events.xml is the layout for the main screen of the Event Ticket Booking System. It consists of a RelativeLayout as the root layout and includes the following components:
Toolbar : Contains a title and a hamburger icon for navigation.
NavigationView : A side navigation drawer with options for profile, hosting an event, and logout.
ScrollView : Contains a LinearLayout that serves as a container for event cards.

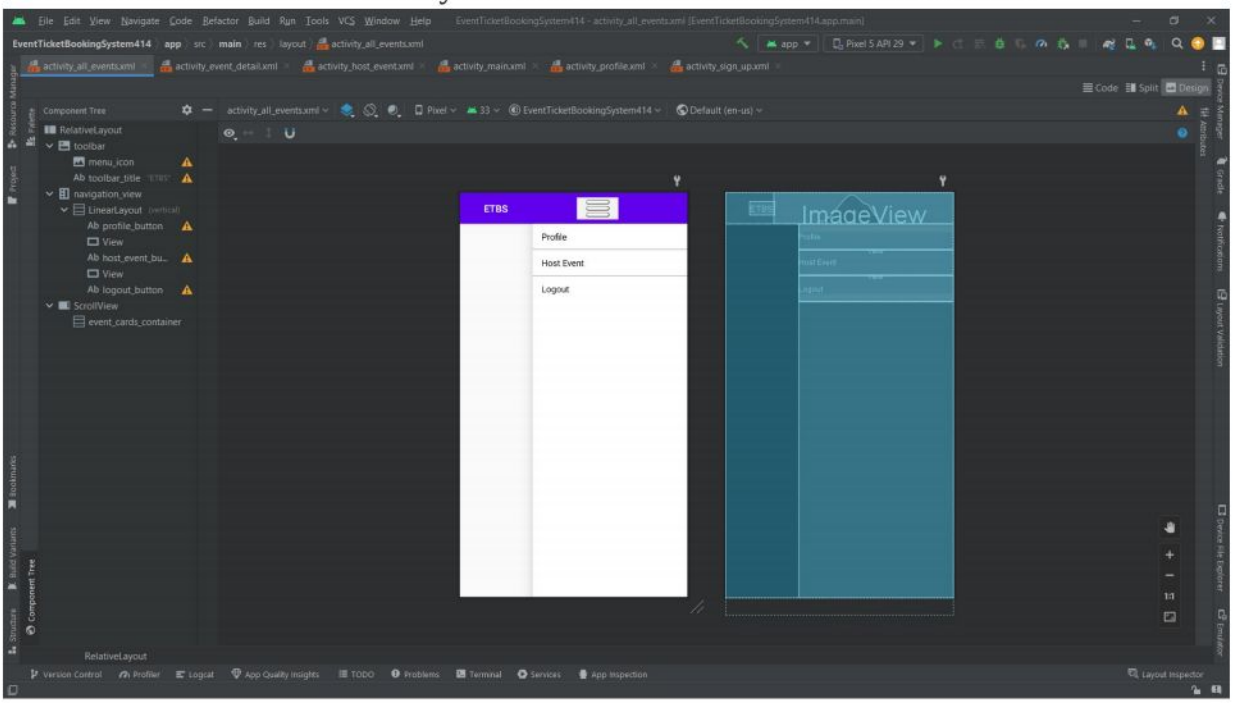

Fig 3.5 : activity_all_events.xml

activity_profile.xml consists of two LinearLayout with vertical orientation, two TextView fields for Full name, email, Hosted events and Registerd events.

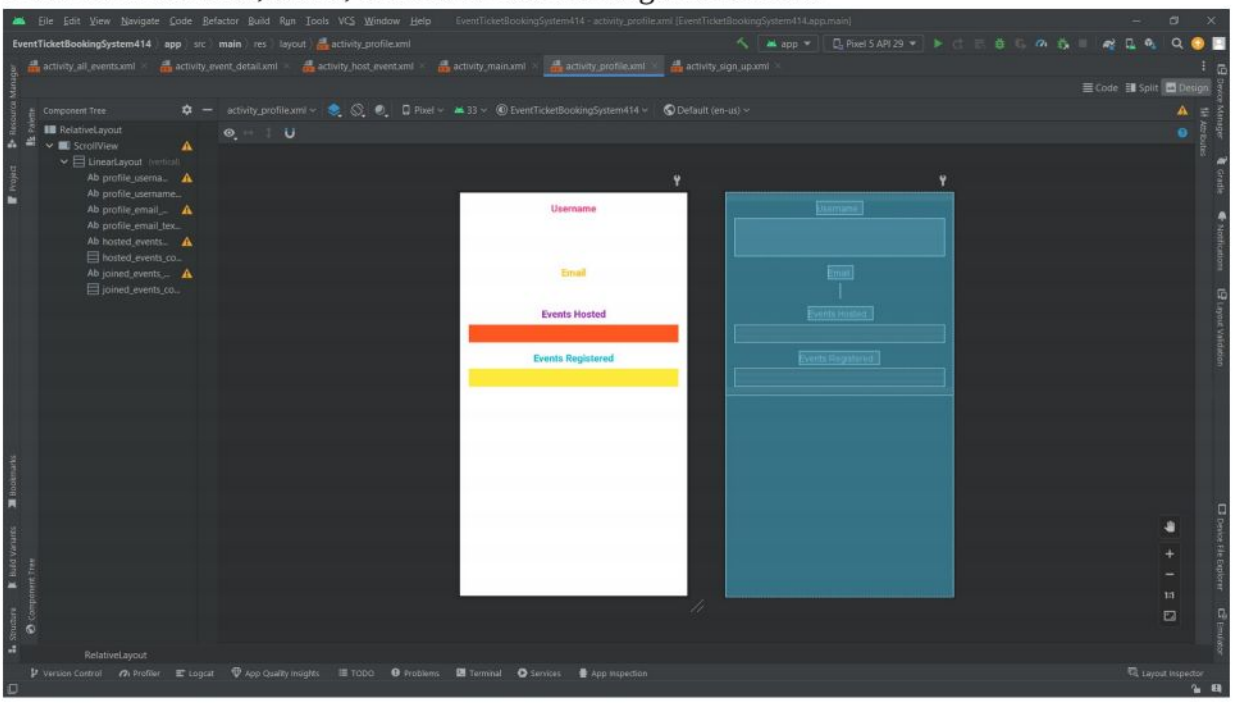

Fig 3.6 : activity_profile.xml

activity_host_event.xml consists of a LinearLayout with vertical orientation, five EditText fields for Event name, description, date, location and available accomadation, a login Button, and a TextView for login.

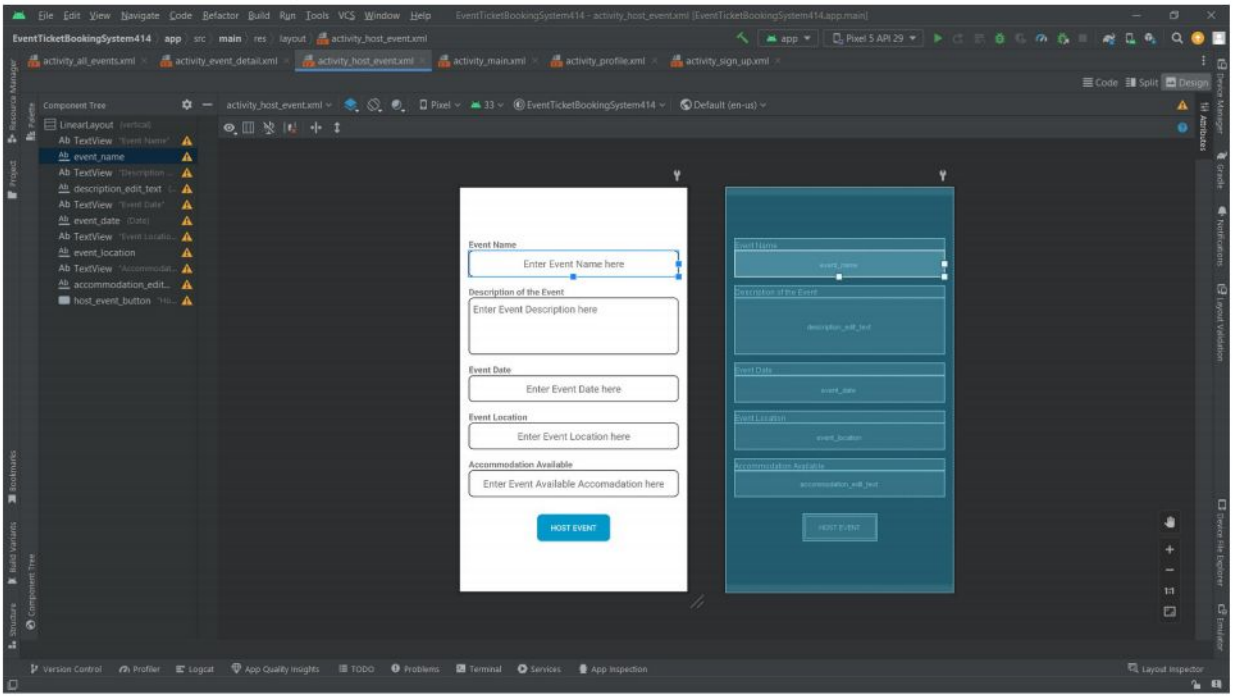

Fig 3.7 : activity_host_event.xml

CHAPTER 4

IMPLEMENTATION

4.1 AndroidManifest.kt

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.EventTicketBookingSystem414"
        tools:targetApi="31">
        <activity
            android:name=".ProfileActivity"
            android:exported="false"
            android:theme="@style/Theme.MaterialComponents.Light.NoActionBar" />
        <activity
            android:name=".Event_DetailActivity"
            android:exported="false"
            android:theme="@style/Theme.MaterialComponents.Light.NoActionBar"/>
        <activity
            android:name=".HostEventActivity"
            android:exported="false"
            android:theme="@style/Theme.MaterialComponents.Light.NoActionBar" />
        <activity
            android:name=".AllEventsActivity"
            android:exported="false"
            android:theme="@style/Theme.MaterialComponents.Light.NoActionBar" />
        <activity
            android:name=".SignUpActivity"
            android:exported="false"
            android:theme="@style/Theme.MaterialComponents.Light.NoActionBar" />
```

```
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.MaterialComponents.Light.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

4.2 MainActivity.kt

```
package com.example.eventticketbookingsystem414
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import com.google.android.gms.tasks.Task
import com.google.firebase.auth.AuthCredential
import com.google.firebase.auth.GoogleAuthProvider
class MainActivity : AppCompatActivity() {
    lateinit var emailEditText: EditText
    lateinit var passwordEditText: EditText
    lateinit var loginButton: Button
    lateinit var signupTextView: TextView
    lateinit var auth: FirebaseAuth
    private val RC_SIGN_IN = 9001
    public override fun onStart() {
        super.onStart()
```

```
        val currentUser = auth.currentUser
        if (currentUser != null) {
            val intent = Intent(this, AllEventsActivity::class.java)
            startActivity(intent)
            finish() // Close the current activity to prevent going back to the login screen
        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken(getString(R.string.default_web_client_id))
            .requestEmail()
            .build()
        val googleSignInClient = GoogleSignIn.getClient(this, gso)
        // Initialize auth
        auth = Firebase.auth
        emailEditText = findViewById(R.id.emailEditText)
        passwordEditText = findViewById(R.id.passwordEditText)
        loginButton = findViewById(R.id.loginButton)
        signupTextView = findViewById(R.id.signupTextView)
        loginButton.setOnClickListener {
            val email = emailEditText.text.toString().trim()
            val password = passwordEditText.text.toString().trim()
            if (email.isNotEmpty() && password.isNotEmpty()) {
                auth.signInWithEmailAndPassword(email, password)
                    .addOnCompleteListener(this) { task ->
                        if (task.isSuccessful) {
                            val intent = Intent(this, AllEventsActivity::class.java)
                            startActivity(intent)
                            finish() // Close the current activity to prevent going back to the login screen
                        } else {
                            Toast.makeText(
                                this,
                                "Authentication failed.",
                                Toast.LENGTH_SHORT
                            ).show()
```

```
                }
            }
        } else {
            Toast.makeText(this, "Please enter valid credentials", Toast.LENGTH_SHORT).show()
        }
    }
    signupTextView.setOnClickListener {
        val intent = Intent(this, SignUpActivity::class.java)
        startActivity(intent)
    }
}
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == RC_SIGN_IN) {
        val task: Task<GoogleSignInAccount> =
        }
    }
}
private fun firebaseAuthWithGoogle(idToken: String) {
    val credential: AuthCredential = GoogleAuthProvider.getCredential(idToken, null)
    auth.signInWithCredential(credential)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                val intent = Intent(this, AllEventsActivity::class.java)
                startActivity(intent)
                finish() // Close the current activity to prevent going back to the login screen
            } else {
                Toast.makeText(this, "Authentication failed.", Toast.LENGTH_SHORT).show()
            }
        }
}
}
```

4.3 SignUpActivity.kt

```
package com.example.eventticketbookingsystem414
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
class SignUpActivity : AppCompatActivity() {
    private lateinit var un: EditText
    private lateinit var signupEmailEditText: EditText
    private lateinit var signupPasswordEditText: EditText
    private lateinit var signupButton: Button
    private lateinit var loginTextView: TextView
    private lateinit var auth: FirebaseAuth
    private lateinit var database: DatabaseReference
    public override fun onStart() {
        super.onStart()
        val currentUser = auth.currentUser
        if (currentUser != null) {
            val intent = Intent(this, AllEventsActivity::class.java)
            startActivity(intent)
            finish()
        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sign_up)
        auth = Firebase.auth
        database = FirebaseDatabase.getInstance().reference
        un = findViewById(R.id.un)
        signupEmailEditText = findViewById(R.id.signupEmailEditText)
```

```
        signupPasswordEditText = findViewById(R.id.signupPasswordEditText)
        signupButton = findViewById(R.id.signupButton)
        loginTextView = findViewById(R.id.loginTextview)
        signupButton.setOnClickListener {
            val username = un.text.toString().trim()
            val email = signupEmailEditText.text.toString().trim()
            val password = signupPasswordEditText.text.toString().trim()
            if (email.isNotEmpty() && password.isNotEmpty()) {
                auth.createUserWithEmailAndPassword(email, password)
                    .addOnCompleteListener(this) { task ->
                        if (task.isSuccessful) {
                            val userId = auth.currentUser?.uid
                            if (userId != null) {
                                val userRef = database.child("users").child(userId)
                                userRef.child("username").setValue(username)
                                userRef.child("email").setValue(email)
                                val intent = Intent(this, MainActivity::class.java)
                                startActivity(intent)
                                finish()
                                Toast.makeText(this, "Signup Successful", Toast.LENGTH_SHORT).show()
                            }
                        } else {
                            Log.d("SignUpActivity", "Authentication failed => $email -- $password")
                            Toast.makeText(this, "Authentication failed.", Toast.LENGTH_SHORT).show()
                        }
                    }
            } else {
                Toast.makeText(this, "Please enter valid credentials", Toast.LENGTH_SHORT).show()
            }
        }
        loginTextView.setOnClickListener {
            val intent = Intent(this, MainActivity::class.java)
            startActivity(intent)
        }
    }
}
```

4.4 AllEventsActivity.kt

```
package com.example.eventticketbookingsystem414
import android.content.Intent
import android.graphics.Typeface
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.LinearLayout
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.cardview.widget.CardView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.*
import com.google.firebase.ktx.Firebase
class AllEventsActivity : AppCompatActivity() {
    private lateinit var logoutBtn: TextView
    private lateinit var hostEventBtn: TextView
    private lateinit var profileBtn: TextView
    private lateinit var eventCardsContainer: LinearLayout
    private lateinit var errorMessageTextView: TextView
    private lateinit var database: DatabaseReference
    private lateinit var un: String
    private lateinit var eventID: String
    val firebaseAuth = FirebaseAuth.getInstance()
    val currentUser: FirebaseUser? = firebaseAuth.currentUser
    public override fun onStart() {
        super.onStart()
        currentUser?.let {
            un = it.uid // Get the user's unique ID
        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
setContentView(R.layout.activity_all_events)
val menuIcon: View = findViewById(R.id.menu_icon)
val navigationView: View = findViewById(R.id.navigation_view)
logoutBtn = findViewById(R.id.logout_button)
hostEventBtn = findViewById(R.id.host_event_button)
profileBtn = findViewById(R.id.profile_button)
eventCardsContainer = findViewById(R.id.event_cards_container)
logoutBtn.visibility = View.GONE // Hide the logout button initially
hostEventBtn.visibility = View.GONE // Hide the host event button initially
navigationView.visibility = View.GONE
menuIcon.setOnClickListener {
    toggleVisibility(navigationView)
    toggleVisibility(logoutBtn) // Toggle the visibility of the logout button
    toggleVisibility(hostEventBtn) // Toggle the visibility of the host event button
}
profileBtn.setOnClickListener {
    val intent = Intent(this, ProfileActivity::class.java)
    startActivity(intent)
}
logoutBtn.setOnClickListener {
    // Handle logout button click
    Firebase.auth.signOut()
    val intent = Intent(this, MainActivity::class.java)
    startActivity(intent)
    finish() // Close the current activity to prevent going back to the signup screen
}
hostEventBtn.setOnClickListener {
    // Handle host event button click
    val intent = Intent(this, HostEventActivity::class.java)
    startActivity(intent)
}
// Initialize Firebase Database
database = FirebaseDatabase.getInstance().reference.child("events")
// Fetch events from Firebase Database and create event cards dynamically
database.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(snapshot: DataSnapshot) {
        eventCardsContainer.removeAllViews() // Clear existing event cards
```

```
            for (eventSnapshot in snapshot.children) {
                val event = eventSnapshot.getValue(EventModel::class.java)
                if (event != null) {
                    val eventCard = createEventCard(
                        eventSnapshot.key ?: "",
                        event.name,
                        event.description,
                        event.date,
                        event.location,
                        event.accommodationAvailable?.toInt(),
                        event.email
                    )
                    eventCardsContainer.addView(eventCard)
                }
            }
        }
        override fun onCancelled(error: DatabaseError) {
            // Handle database read error
            val errorMessage = "Database read error: ${error.message}"
            Toast.makeText(this@AllEventsActivity, errorMessage, Toast.LENGTH_SHORT).show()
        }
    })
}
private fun createEventCard(
    eventKey: String,
    eventName: String?,
    description: String?,
    eventDate: String?,
    eventLocation: String?,
    accommodationAvailable: Int?,
    hostedByEmail: String?
): CardView {
    val cardView = CardView(this)
    val layoutParams = LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.WRAP_CONTENT
    )
```

```
        layoutParams.setMargins(0, 0, 0, 16) // Add margin between event cards
        cardView.layoutParams = layoutParams
        cardView.cardElevation = 4f
        val cardContentLayout = LinearLayout(this)
        cardContentLayout.orientation = LinearLayout.VERTICAL
        cardContentLayout.setPadding(16, 16, 16, 16)
        val eventNameTextView = TextView(this)
        eventNameTextView.text = eventName ?: ""
        eventNameTextView.textSize = 18f
        eventNameTextView.setTypeface(null, Typeface.BOLD)
        val descriptionTextView = TextView(this)
        descriptionTextView.text = description ?: ""
        val eventDateTextView = TextView(this)
        eventDateTextView.text = eventDate ?: ""
        val eventLocationTextView = TextView(this)
        eventLocationTextView.text = eventLocation ?: ""
        val accommodationTextView = TextView(this)
        accommodationTextView.text = "Accommodation available: ${accommodationAvailable ?: 0}"
        val hostedByEmailTextView = TextView(this)
        hostedByEmailTextView.text = "Hosted by: ${hostedByEmail ?: ""}"
        val bookButton = Button(this)
        bookButton.text = "Book"
        val ticketsRef = FirebaseDatabase.getInstance().reference.child("tickets")
            ticketsRef.orderByChild("username").equalTo(un).addListenerForSingleValueEvent(object :
ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                val bookedEvents = mutableListOf<String>()
                for (ticketSnapshot in snapshot.children) {
                    val ticket = ticketSnapshot.getValue(TicketModel::class.java)
                    ticket?.eventID?.let { bookedEvents.add(it) }
                }
                // Disable the button if the user has already booked the ticket for this event
                if (bookedEvents.contains(eventKey)) {
                    bookButton.isEnabled = false
                    bookButton.text = "Booked"
                } else {
                    bookButton.isEnabled = true
```

```
            bookButton.setOnClickListener {
                bookButton.isEnabled = false // Disable the book button
                decrementAccommodation(eventKey, accommodationAvailable)
            }
        }
        cardContentLayout.addView(eventNameTextView)
        cardContentLayout.addView(descriptionTextView)
        cardContentLayout.addView(eventDateTextView)
        cardContentLayout.addView(eventLocationTextView)
        cardContentLayout.addView(accommodationTextView)
        cardContentLayout.addView(hostedByEmailTextView)
        cardContentLayout.addView(bookButton)
    }
    override fun onCancelled(error: DatabaseError) {
        // Handle database read error
        val errorMessage = "Database read error: ${error.message}"
        Toast.makeText(this@AllEventsActivity, errorMessage, Toast.LENGTH_SHORT).show()
    }
  })
  cardView.addView(cardContentLayout)
  return cardView
}
private fun decrementAccommodation(eventKey: String, accommodationAvailable: Int?) {
  if (accommodationAvailable != null && accommodationAvailable > 0) {
    database = FirebaseDatabase.getInstance().reference.child("events")
    val updatedAccommodation = (accommodationAvailable - 1).toString()
    database.child(eventKey).child("accommodationAvailable").setValue(updatedAccommodation)
      .addOnCompleteListener { task ->
        if (task.isSuccessful) {
          Toast.makeText(
            this@AllEventsActivity,
            "Accommodation booked successfully!",
            Toast.LENGTH_SHORT
          ).show()
        } else {
          Toast.makeText(
            this@AllEventsActivity,
```

```
                    "Failed to book accommodation. Please try again.",
                    Toast.LENGTH_SHORT
                ).show()
            }
        }

        database = FirebaseDatabase.getInstance().reference.child("tickets")
        val ticketID = database.child("tickets").push().key ?: ""
        val userRef = database.child(ticketID)
        userRef.child("username").setValue(un)
        userRef.child("eventID").setValue(eventKey)
        // Update the accommodation value in Firebase Database
    }
}
private fun toggleVisibility(view: View) {
    if (view.visibility == View.GONE) {
        view.visibility = View.VISIBLE // Show the view if it's hidden
    } else {
        view.visibility = View.GONE // Hide the view if it's visible
    }
}
}
```

CHAPTER 5

RESULTS AND DISCUSSION

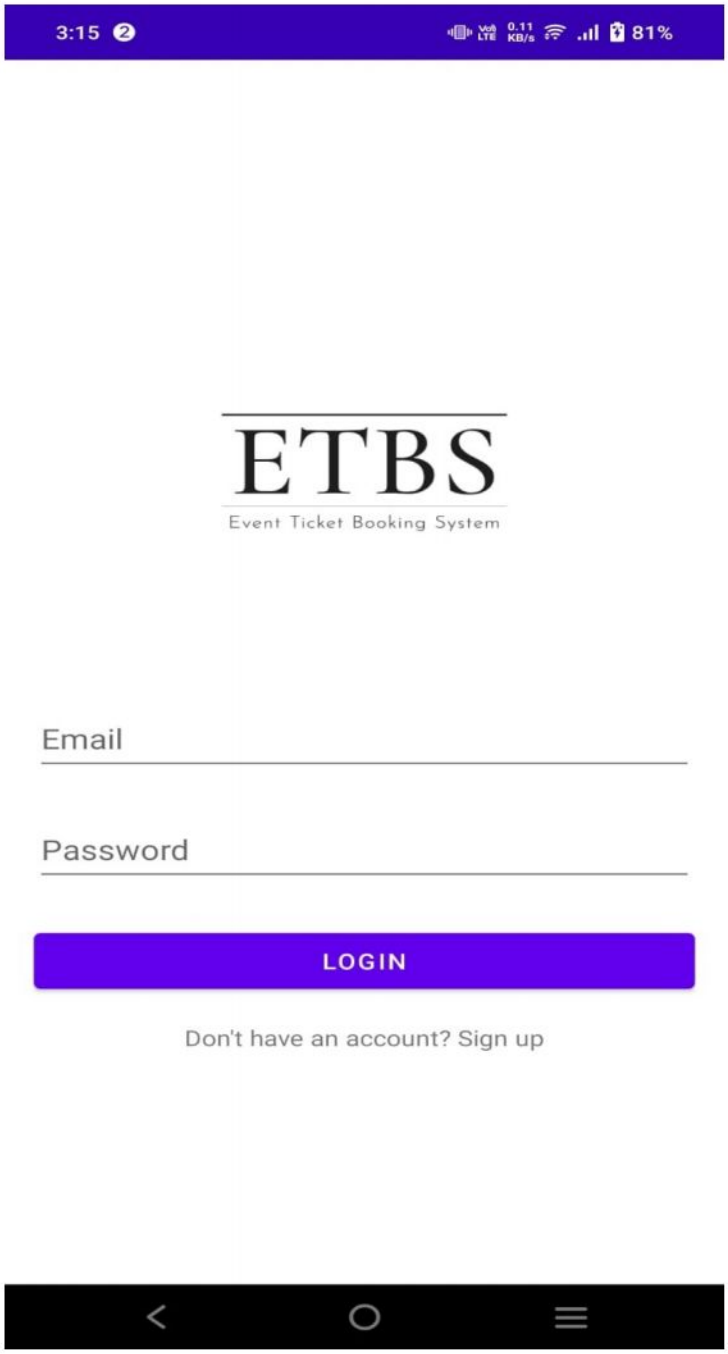

Fig 5.1 loginpage

This the launcher page where the user logins to his account if he doesn't have an accounts they can click on signup to move on to the signup procedure.

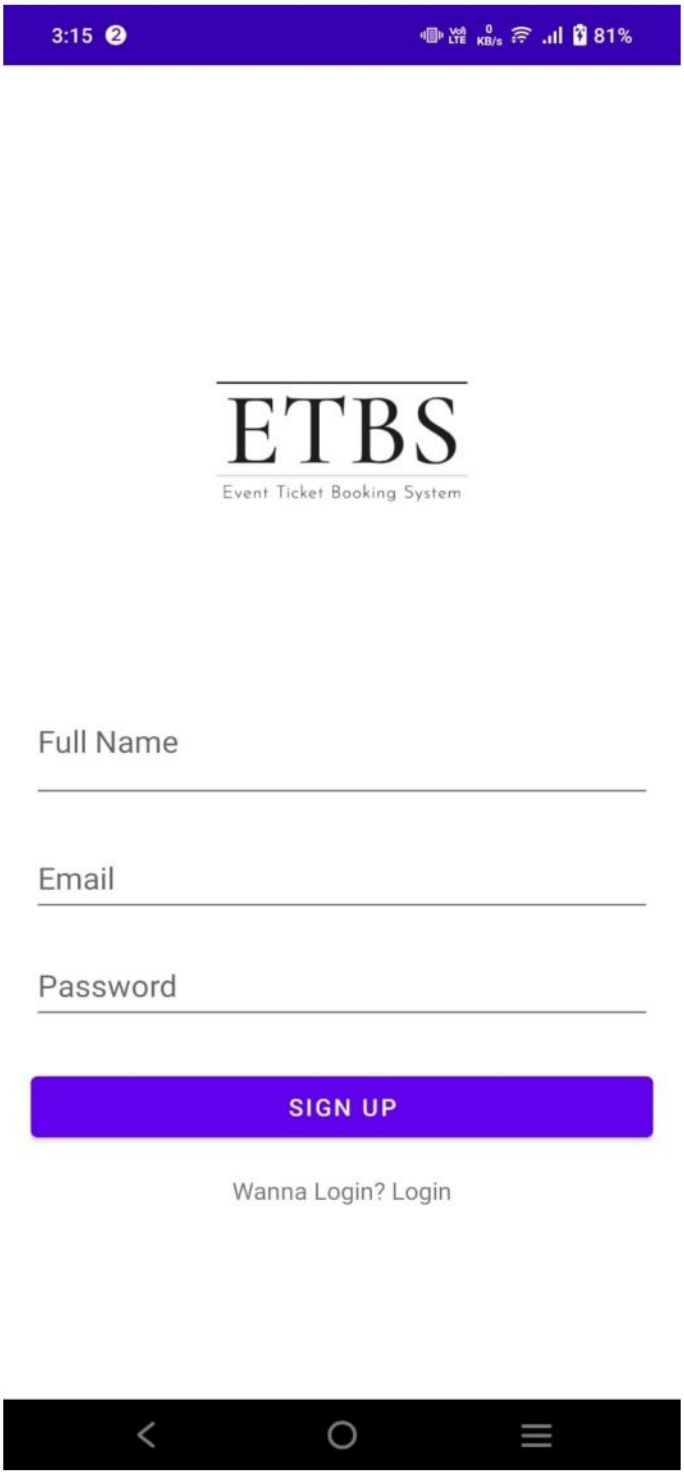

Fig 5.2 signup page

This the Signup page where the user Registers his account if he has already registered and has an accounts they can click on Login to move on to the Login procedure.

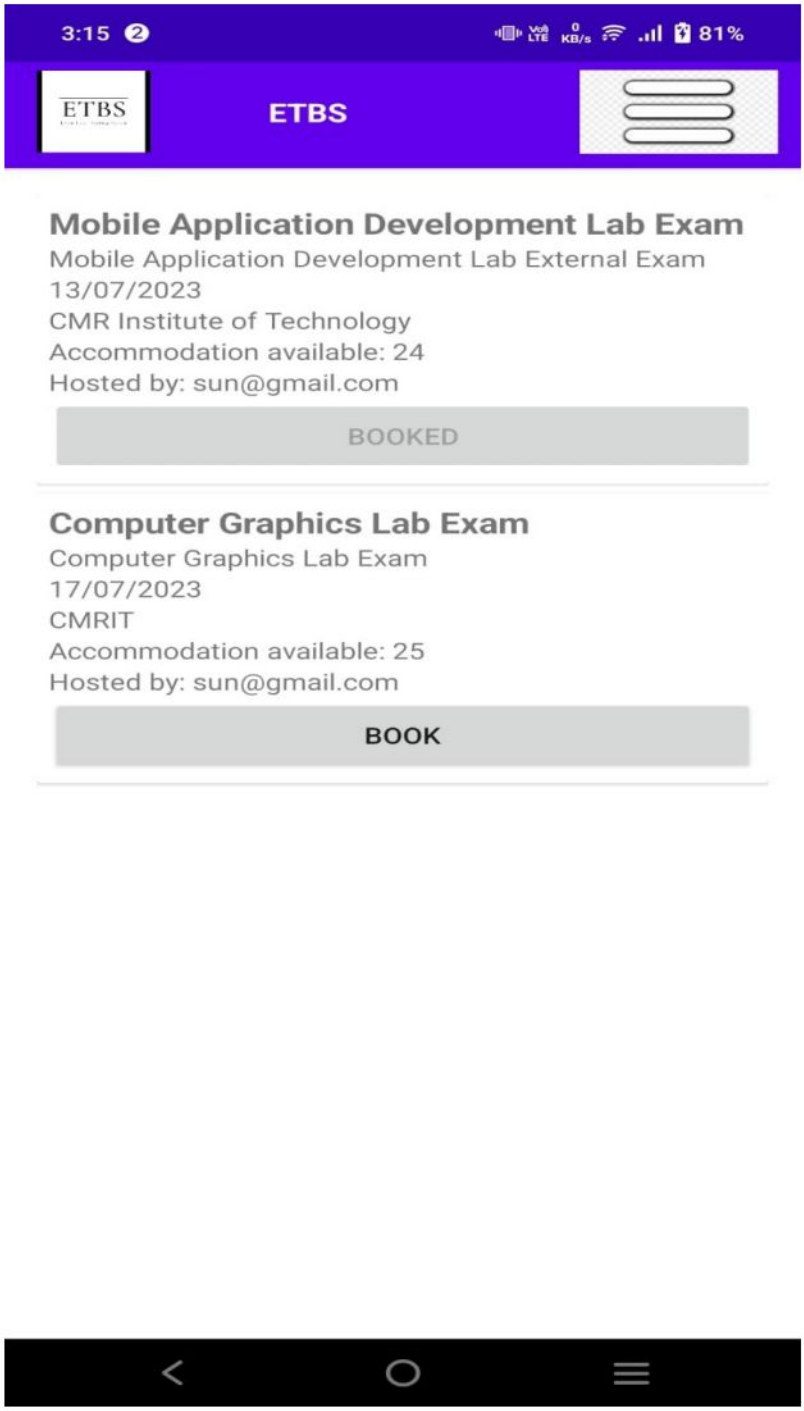

Fig 5.3 All Events page

This the All Events page where the user can view all the events hosted by the users and other users events also and book a ticket. And for to do anything else he can click on the hamburger menu.

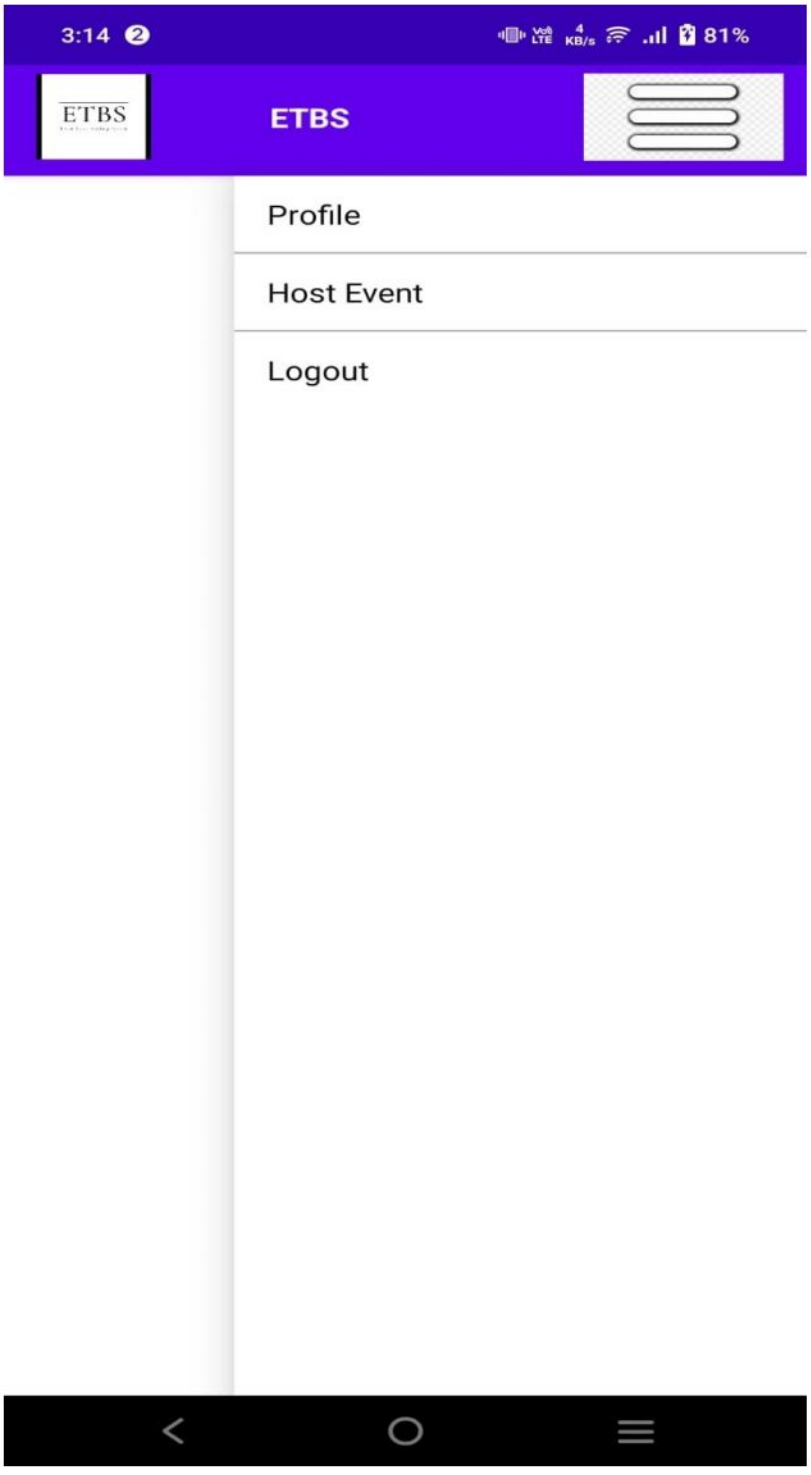

Fig 5.4 navigationpage

This the navigation menu which appears when a user clicks on the hamburger menu in all events page from here the user can move to his profile, or Can host an Event or Logout.

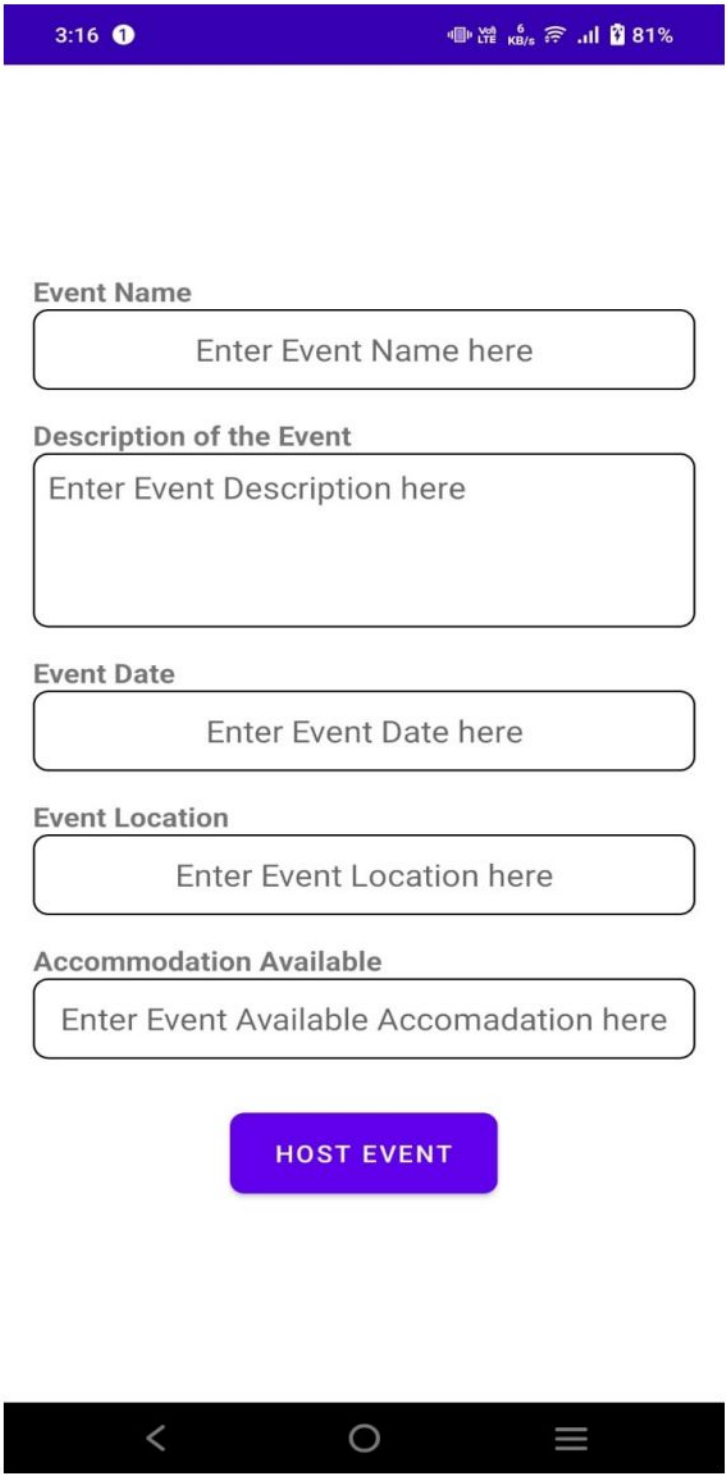

Fig 5.5 hostevent

When the user clicks on the host button in the navigation menu the user will be moved to this page Host Event, here user can fill the details and host an Event and user can only host events which starts from tommorow

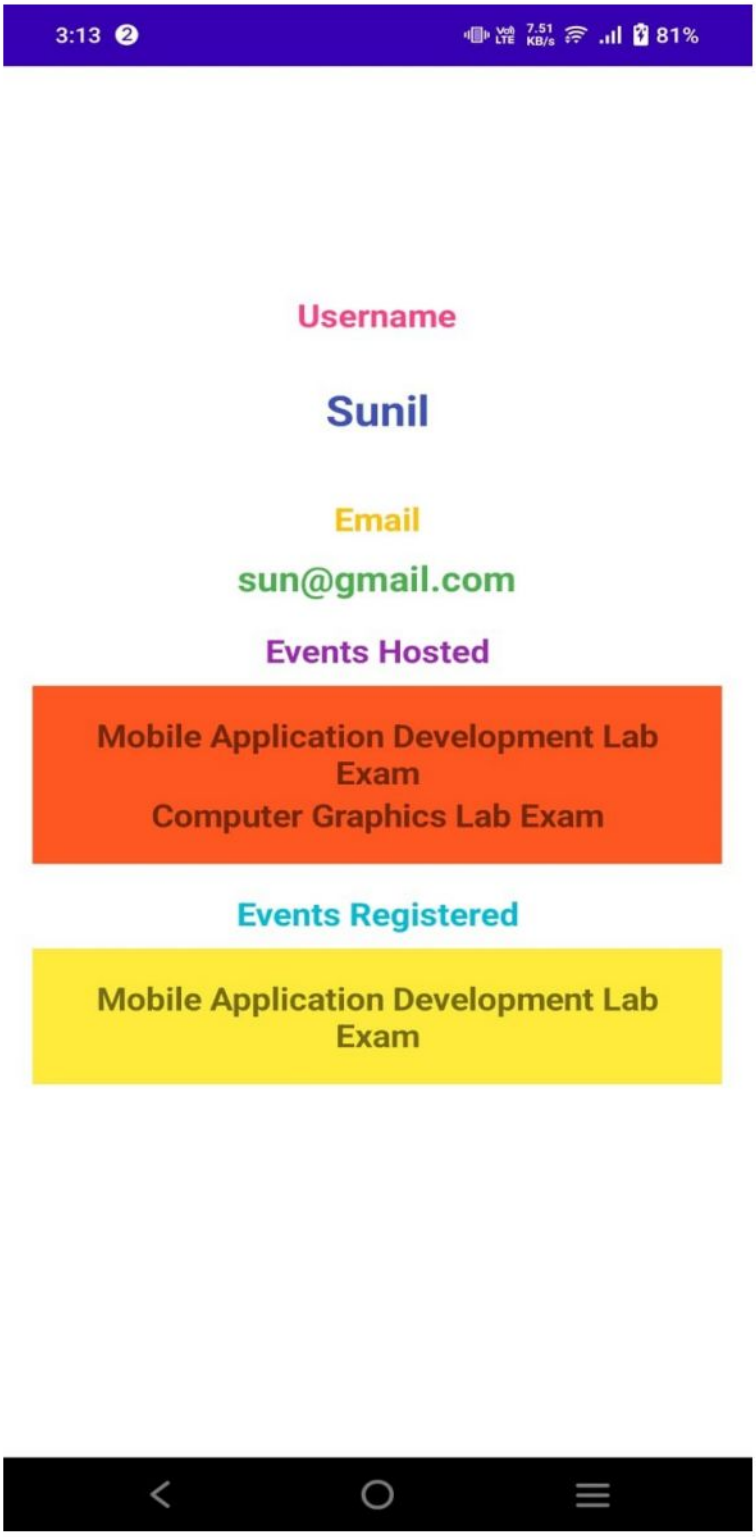

Fig 5.6 profile page

When the user clicks on the profile button in the navigation menu the user will be moved to this page Profile, here user can see his details and all the events he has hosted and all the events he has registered.

Fig 5.7 Event details page

When the user clicks any events he will be able see the event details and the users which are particpating in the events.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The Event Ticket Booking System developed using Android Studio and Firebase Realtime Database offers a user-friendly platform for browsing and booking event tickets. The implemented features, such as user login, event listing, event hosting, and profile management, provide a solid foundation for an efficient ticket booking system.

6.2 FUTURE SCOPE

- Online Payment Gateway Integration: Implementing a secure and reliable online payment gateway integration will enable users to make ticket purchases directly through the app. This can be achieved by integrating popular payment gateways such as PayPal, Stripe, or Braintree. It will provide users with a seamless and convenient payment experience while ensuring the security of their financial transactions.

- Multiple Payment Options: Expand the payment options available to users by integrating different payment methods such as credit cards, debit cards, net banking, digital wallets, and mobile payment solutions. Offering a variety of payment options will cater to the preferences and convenience of a wider range of users.

REFERENCES

[1] Google Developer Training, “Andriod developer Fundamentals Course – Concept Reference”, Google Developer Training Team, 2017.

[2] Erik Hellman, “Andriod Programming – Pushing the Limits”, 1st Edition, Wiley India Pvt Ltd, 2014.

[3] Bill Philips, Chris Stewart and Kristin Marsicano “Andriod Programming : The Big Nerd Ranch Guide”, 3rd Edition, 2017.

[4] https://developer.andriod.com.

[5] Firebase Documentation, https://firebase.google.com/docs/

[6] Firebase Authentication, https://firebase.google.com/docs/auth/

[7] Firebase Realtime Database, https://firebase.google.com/docs/database/

[8] Android Studio Docs, https://developer.android.com/docs/

REFERENCES

[1] Angel, Edward. *Interactive Computer Graphics: A top-down approach with OpenGL*. Addison-Wesley Longman Publishing Co., Inc., 1996.

[2] Hearn, D., Baker, M. P., & Baker, M. P. (2004). *Computer graphics with OpenGL* (Vol. 3). Upper Saddle River, NJ:: Pearson Prentice Hall.

[3] How To Setup OpenGL freeGLUT in CodeBlocks ? Computer Graphics | Windows 10/8/7,**"https://www.youtube.com/watch?v=VMsTI_CC-jc**