Course: Computer Science

Module: Mobile Web Application Development

Course leader: Dr. Sivasankari Sivakumar

Student: Mohamad Sharif

Student ID: 21587035

Video demonstration:

https://www.youtube.com/watch?v=IUmLbLBl7ws&ab_channel=MohamadSharif

1. Introduction

The app idea is very straight forward, it’s a bicycle shop where you can buy and order bicycles from. The app provides an intuitive and user-friendly interface, ensuring a seamless and enjoyable browsing experience. The database that I used to build the app is the firebase real-time database by google and the reason that I chose it is because google firebase provides real-time synchronization, enabling instant updates across all connected devices. This is crucial for an e-commerce app where users and administrators need real-time information on product availability, order status, and other dynamic data. And for the wireframe I used Figma as the UI of Figma is simple to use and intuitive. It's a great option for creating a wireframe because programmers can easily become familiar with the tools and capabilities. When measured against certain other design tools, the learning curve is rather low.and I also used git to be able to work on multiple devices and the also control the version in case I write a wrong code I can always roll back, and at the end of this report I will provide a link to my GitHub so you can be able the access the full application. Lastly All the back-end code is all written using **kotlin.**

Report structure

1. **Introduction:** in this section I will explain about the application and the technologies used in the application and why I chose them.

2. **Wireframing, UI Design and development:** this section will be focusing on pre-development sketches, wireframes; and the final UI of the application.

3. **Back End development:** in this section I will talk about challenges that I faced while developing the back-up of my application.

4. **Reflection &Discussion:** in this section I will discuss what I learned throughout the development of the app.

5. **Conclusion and Future work:** in this section I will say my final words, critics, and future changes.

6. **References:** in this section I will mention the resources used that helped me developing the app.

7. **Appendix/Code:** in this section I will be sharing my back-end code.

2. Wireframing, User Interface design and development

I created the app's basic and elegant wireframes and user interface. Off-white, black, and white make up the color palette.

Wireframe and UI

Figure 1.1

This is the Main activity that the user will see when entering the application, in this activity the user can look at products, add items to the cart and search for an item

Note: User does not need to login to add items to the cart

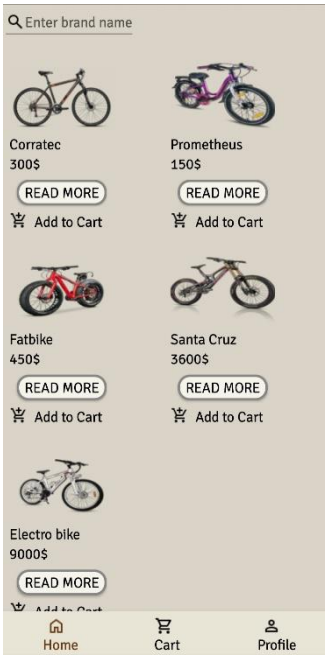


Figure 1.2

Figure 2.1

This is the registration activity where users can create their account, the activity includes a radio buttons and an input filed with hint text for each input field.

Figure 2.2

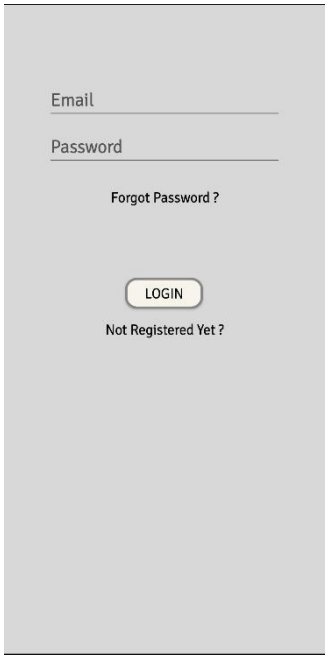


Figure 3.1

This is the login activity and the design of it is very Simple, in the activity the user can login using their email address and password also the activity offers options like forgot password and not registered yet.

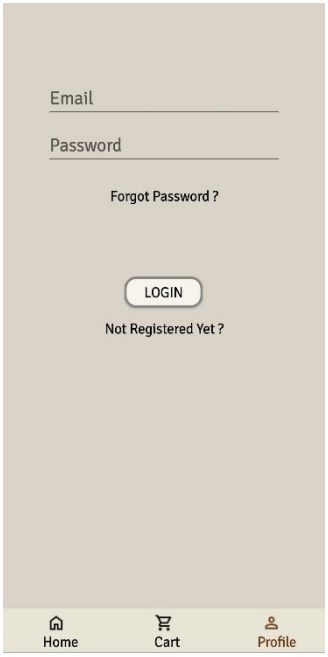


Figure 3.2

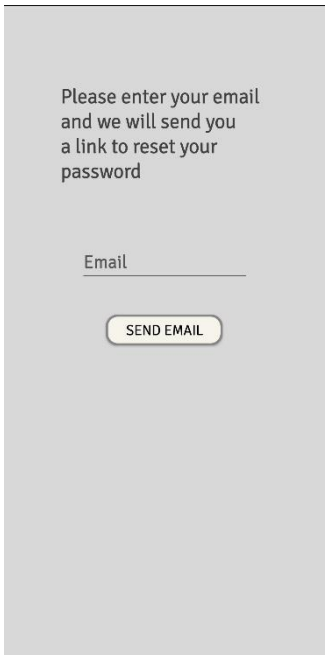


Figure 4.1

This is the forgot password activity and its very simple And straight forward. The user enters their email and They will receive a link to reset their password from

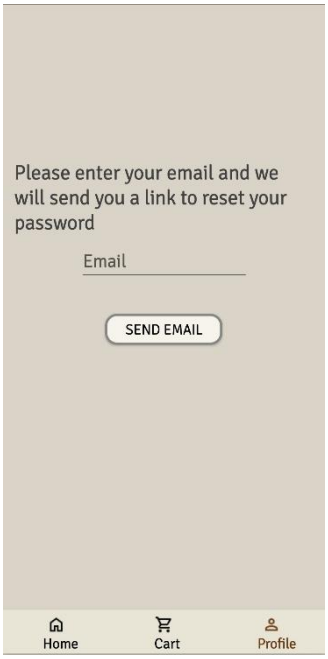


Figure 4.2

Figure 5.1

This is the edit profile activity and, in this activity, the user will be able to edit their details such like username, birthday date and gender. Users will not be able to edit their email as it will be considered their unique identified.

Figure 5.2

Figure 6.1

This is the cart activity and it has a simple design with Options to remove items or increase it's quality, and it also display the total price of the cart.

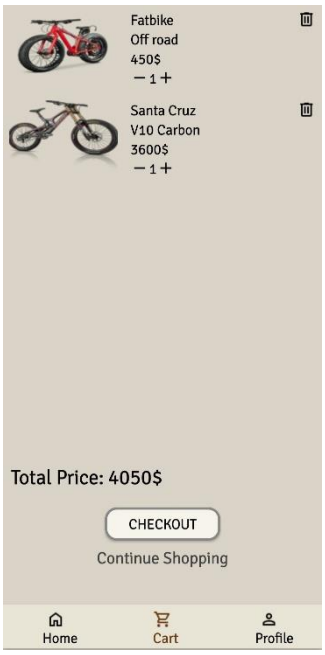


Figure 6.2

Considering that the required wireframes for this assignment is 6 I will be sharing the rest of my application as screen shots only without the a wireframe.

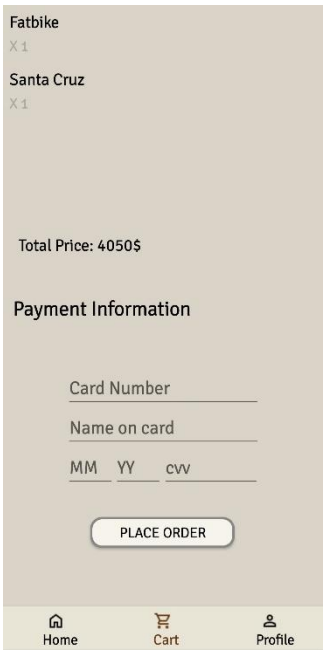


Figure 7.1

This is the check out activity, the user will see this page after clicking On the Check Out button, on the cart activity. If the user is not logged in another activity will be displayed to ask the user to either log in or sign up.

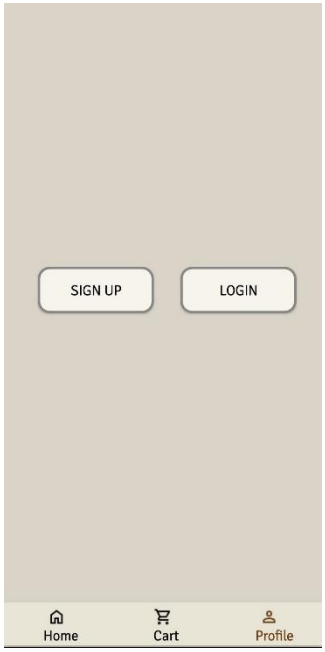
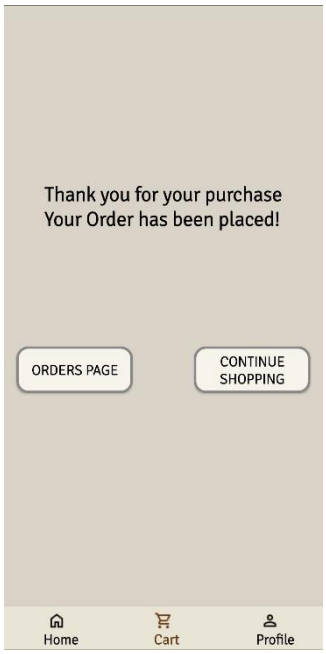


Figure 7.2

Figure 7.3



After entering a valid card Number and a valid card expiration date and a cvv the user will be redirected to this activity which includes a thank you message and two buttons “continue shopping” and “orders page” button.

Figure 8.1

This is the orders page activity in this page the user can see their orders and by clicking on any order they will be redirected to another activity which will include the order details like items that have been orders and the total amount paid, order Status and order date.

Figure 8.2

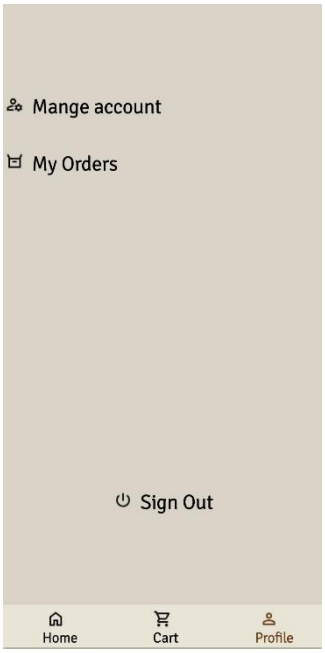


Figure 9

This is the profile activity its very simple it has three buttons
“Mange Account”, “My orders” and a Sign out button
also if the user is not logged in the user will be redirected to
the activity where the user will be asked to either log in or sign up

Figure 10

This activity will be displayed when the user clicks on the read more
Button on the Main activity, the purpose of this activity is to show
The bicycle details such as bicycle brand, model, description and price.

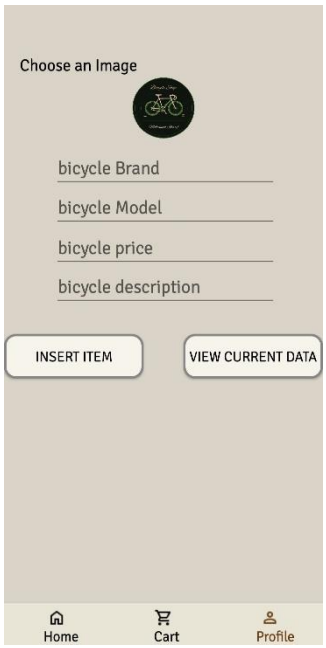


Figure 11.1

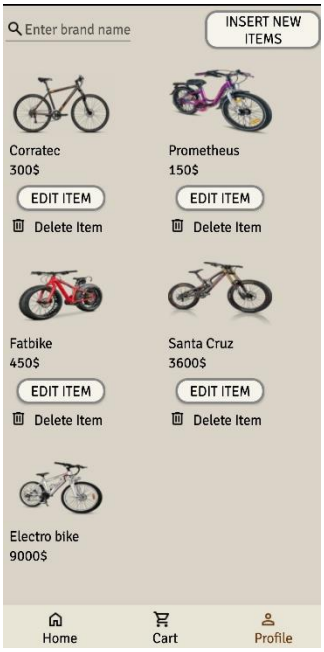


Figure 11.2

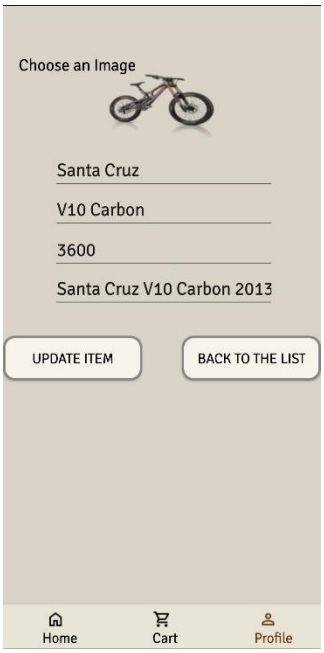


Figure 11.3

Lastly the admin panel, the panel allows you to insert, edit and delete items from the database

3. Back End development: critical parts

The hardest part for me was creating a local storage cart and the reason I chose to do it is because I wanted all users either logged in or not to be able to add items to the cart. I created the local database using **SharedPreferences** API. Which then it will generate for me a local JSON database this is an example function of how the prosses is done:

```
private fun saveCartItems(cartItems: Set<Bicycle>) {
    val sharedPreferences = getSharedPreferences("Cart",
Context.MODE_PRIVATE)
    val editor = sharedPreferences.edit()
    editor.putStringSet("cart_items", cartItems.map { Gson().toJson(it)
}.toSet())
    editor.apply()
}
and after doing a successful order only the order will be pushed to the
database here is how I'm doing  this :

private fun placeOrder() {
    // Get user information
    val userId = FirebaseAuth.getInstance().currentUser?.uid

    // Check if the user is signed in
    if (userId != null) {
        val cartItems = getCartItems()
```

```
        // Check if the cart is not empty
        if (cartItems.isNotEmpty()) {
            val orderItems = cartItems.map {
                OrderItem(
                    bicycleId = it.id,
                    brand = it.brand,
                    model = it.model,
                    quantity = it.quantity,
                    price = it.price.toDouble()
                )
            }

            val totalAmount = cartItems.sumByDouble { it.price.toDouble() *
it.quantity }

            // Create an Order object
            val order = Order(
                orderId = generateOrderId(),
                items = orderItems,
                totalAmount = totalAmount,
                orderDate = getCurrentDateTime(),
                )

            // Save the order to Firebase Realtime Database
            saveOrderToDatabase(userId, order)

            // Clear the cart after placing the order
            clearCart()

            Toast.makeText(this, "Order placed successfully!",
Toast.LENGTH_SHORT).show()
        } else {
            Toast.makeText(this, "Your cart is empty. Add items before
placing an order.", Toast.LENGTH_SHORT).show()
        }
    } else {
        // Handle the case where the user is not signed in
        Toast.makeText(this, "User not signed in. Please sign in to place
an order.", Toast.LENGTH_SHORT).show()
    }
}
```

as you can see in this function it first double checks whether the user is logged in or not before placing the order then it fetch the user id then it also double checks if the cart is empty and after those checks the function pushes the order to the database.

The kotlin code includes validations for activities such as preventing the user to check out with an empty carts and decreasing the quantity of an item when its 1.

```
This is the check out cart validation code:

cartItems = getCartItems().toMutableList() // Initialize cartItems
checkOutButton.setOnClickListener {
    if (cartItems.isEmpty()) {
```

```
        Toast.makeText(this, "Pleases add items before checking out.",
Toast.LENGTH_SHORT).show()
    } else {
        auth = FirebaseAuth.getInstance()
        authListener = FirebaseAuth.AuthStateListener { firebaseAuth ->
            val user = firebaseAuth.currentUser
            if (user == null) {
                // User is not logged in, navigate to the Login or Sign Up
activity
                val intent = Intent(this@Cart, LoginOrSignUp::class.java)
                startActivity(intent)
            } else {
                // User is logged in, navigate to the Checkout activity
                val intent = Intent(this@Cart, CheckOut::class.java)
                intent.putExtra("TotalPrice", calculateTotalPrice())
                intent.putExtra("CartItems", ArrayList(getCartItems()))
                startActivity(intent)
            }
        }
        // Add the AuthStateListener to check the user's authentication
status
        auth.addAuthStateListener(authListener)
    }
}
```

And I also included input validation for both login and sign up activity.

```
Login activity validation :


LoginBtn.setOnClickListener{
    if ((loginFiled.text.toString() == "admin") &&
(passwordFiled.text.toString() == "admin")) {
        val user = auth.currentUser
        val Intent = Intent(this, adminLogin::class.java)
        startActivity(Intent)
    } else {
        if (isEmailValid(loginFiled.text.toString().trim()))
            loginUser(loginFiled.text.toString().trim(),
passwordFiled.text.toString())
        else
            ValadationText.text = "Please enter a valid email"
    }
}
```

As you can see the app first checks if the input is admin, and the pass is admin which is the admin's login email and password and if not, it will validate the input email and if it passed the validation, it would process to the login.

```
Sign Up activity validation:

signupButton.setOnClickListener {
    // Fetch EditText values when the button is clicked
    val birthDate =
findViewById<EditText>(R.id.AgeSignUpAct).text.toString() // Retrieve date
of birth
    val email =
findViewById<EditText>(R.id.EmailSignUpAct).text.toString().trim()
```

```
    val userName =
findViewById<EditText>(R.id.usernameSignUpAct).text.toString().trim()
    val password =
findViewById<EditText>(R.id.passwordSignUpAct).text.toString()
    val password2 =
findViewById<EditText>(R.id.ConfoirmPasswordSignUpAct).text.toString()
    val genderGroup = findViewById<RadioGroup>(R.id.radioGroupSignUpAct)
    val ValidationError =
findViewById<TextView>(R.id.ValidationTextSignUpAct)
    val genderId = genderGroup.checkedRadioButtonId
    val selectedGender = when (genderId) {
        R.id.MaleRadioSignUpAct -> "Male"
        R.id.FemaleRadioSignUpAct -> "Female"
        else -> "Null" // Handle if no gender is selected
    }
    // Validating user's input
    if (isEmailValid(email))
        if (isPasswordMatched(password, password2))
            if (isPasswordValid(password))
                if (selectedGender == "Null")
                    ValidationError.text = "Please select your gender"
                else
                    if (isDateOfBirthValid(birthDate))
                        registerNewUser(email, password, userName,
birthDate, selectedGender)
            else
                ValidationError.text = "Please enter a valid date of birth"
            else
                ValidationError.text = "Password should contain at least 6
characters"
        else
            ValidationError.text = "Password does not match"
    else
        ValidationError.text = "Please enter a valid email"
}
```

The sign up validation validate all the inputs, and also after all this validations there is one last validation that will be done in the registerNewUser function and its to check if the email is already registered in the database or not, here is how I'm checking this:

```
override fun onDataChange(dataSnapshot: DataSnapshot) {
    if (dataSnapshot.exists()) {
        Log.d("SignUpActivity", "email already exists")
        errorText.text = "email already exists"
    } else {
                // do the sign up process
        }
}
```

And I also included validation for the other activity like forgot password, Manage account, etc...

Overall other then creating the local cart storage I did not face any other challenge although I could say that the deadline was kind of short to get all this work done but I manged to get it done.

4. Reflection &Discussion:

Creating this project taught me a lot of things such as working with APIs and creating databases and many other things.

I did use some AI tools such as DALL-E to generate some photos for me and I also used chatgpt to help me build the local storage as I had hard time understanding how it worked at the begging.

5. Reflection

I'm happy with the outcome of my first Android Studio experience and have learned a lot that will help me in the future. But I'll keep working to make the app better.

6. Conclusion and Future work

I want to build an app that can be fully handled and monitored without the need of using the physical database and I think I achieved that goal by adding the admin panel to the app.

Lastly this is my GitHub repo for this code:
https://github.com/Mohamad-ctrl/MWAD-Final-Project

7. References

- **https://stackoverflow.com/**
- **https://in-kotlin.com/**
- **https://developer.android.com/**
- **https://chat.openai.com/**
- **https://www.figma.com/**
- **https://www.dalle3.org/**
- **(freeCodeCamp.org) https://www.youtube.com/watch?v=EExSSotojVI&t=46527s&ab_channel=freeCodeCamp.org**
- **https://fonts.google.com/icons**

8. Appendix/Code

<----------> MainActivity.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.text.Editable
import android.text.TextWatcher
import android.util.Log
import android.view.LayoutInflater
import android.view.View
```

```
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.google.gson.Gson


class MainActivity : AppCompatActivity() {
    private val database = FirebaseDatabase.getInstance()
    private val myRef = database.getReference("bicycles")
    private lateinit var auth: FirebaseAuth
    private lateinit var authListener: FirebaseAuth.AuthStateListener
    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();
        auth = FirebaseAuth.getInstance()
        val welcomeText = findViewById<TextView>(R.id.welcomeTextMainAct)
        val brandEditText = findViewById<EditText>(R.id.brandEditText)
        brandEditText.setOnClickListener {
            val brandName = brandEditText.text.toString()
            readDataFromDatabase(brandName)
        }
        readDataFromDatabase("")

        authListener = FirebaseAuth.AuthStateListener { firebaseAuth ->
            val user = firebaseAuth.currentUser
            if (user == null) {
                // User is signed out, update UI accordingly
                welcomeText.visibility = View.GONE
            } else {
                // User is signed in, update UI with the signed-in user's information
                user.reload().addOnCompleteListener {
```

```
                val updatedUser = FirebaseAuth.getInstance().currentUser
                val userName = user.displayName
                welcomeText.text = "Welcome, ${userName ?: "User"}"
            }
        }
    }


}

override fun onStart() {
    super.onStart()
    auth.addAuthStateListener(authListener)
}

override fun onStop() {
    super.onStop()
    auth.removeAuthStateListener(authListener)
}

//      fun homeIcon(view: View) {
//          val intent = Intent(this@MainActivity, MainActivity::class.java)
//          startActivity(intent)
//   }
fun profileIcon(view: View) {
    val intent = Intent(this@MainActivity, Profile::class.java)
    startActivity(intent)
}
fun cartIcon(view: View){
    val intent = Intent(this@MainActivity, Cart::class.java)
    startActivity(intent)
}
private fun readDataFromDatabase(brandName: String) {
    myRef.addListenerForSingleValueEvent(object : ValueEventListener {
        override fun onDataChange(dataSnapshot: DataSnapshot) {
            val bicycles = mutableListOf<Bicycle>()

            for (snapshot in dataSnapshot.children) {
                val brand = snapshot.child("brand").getValue(String::class.java)

                if (brandName.isEmpty() || brand.equals(brandName, ignoreCase = true)) {
                    val model = snapshot.child("model").getValue(String::class.java)
                    val price = snapshot.child("price").getValue(Int::class.java)
                    val description = snapshot.child("description").getValue(String::class.java)
                    val imageURL = snapshot.child("imageURL").getValue(String::class.java)
                    val bicycleId = snapshot.key
```

```
                if (model != null && price != null && description != null && imageURL != null &&
bicycleId != null) {
                    bicycles.add(Bicycle(bicycleId, brand.toString(), model, description, price, imageURL,
false, 1))
                }
            }
        }

        val recyclerView = findViewById<RecyclerView>(R.id.recyclerViewMainAct)
        recyclerView.layoutManager = GridLayoutManager(this@MainActivity, 2)
        val adapter = BicycleAdapter(bicycles)
        recyclerView.adapter = adapter
      }

      override fun onCancelled(databaseError: DatabaseError) {
        Log.e("MainActivity", "Error reading data from database", databaseError.toException())
      }
    })
  }


  private fun addToCart(item: Bicycle) {
    val cartItems = getCartItems().toMutableSet()
    val existingItem = cartItems.find { it.id == item.id }

    if (existingItem != null) {
      existingItem.quantity++
    } else {
      cartItems.add(item.copy(addedToCart = true, quantity = 1))
    }

    saveCartItems(cartItems)
    Toast.makeText(this@MainActivity, "Item has been added to cart",
Toast.LENGTH_SHORT).show()
  }

  private fun saveCartItems(cartItems: Set<Bicycle>) {
    val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
    val editor = sharedPreferences.edit()
    editor.putStringSet("cart_items", cartItems.map { Gson().toJson(it) }.toSet())
    editor.apply()
  }

  private fun getCartItems(): Set<Bicycle> {
    val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
    val cartItemsSet = sharedPreferences.getStringSet("cart_items", emptySet()) ?: emptySet()
    return cartItemsSet.map { Gson().fromJson(it, Bicycle::class.java) }.toSet()
  }
```

```
    inner class BicycleAdapter(private val bicycleList: List<Bicycle>) :
        RecyclerView.Adapter<BicycleAdapter.BicycleViewHolder>() {

        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): BicycleViewHolder {
            val itemView = LayoutInflater.from(parent.context)
                .inflate(R.layout.item_holder, parent, false)
            return BicycleViewHolder(itemView)
        }

        @SuppressLint("SetTextI18n")
        override fun onBindViewHolder(holder: BicycleViewHolder, position: Int) {
            val currentItem = bicycleList[position]
            val imageUri = Uri.parse(currentItem.ImageURL)

            holder.brandTextView.text = currentItem.brand
            holder.priceTextView.text = "${currentItem.price}$"

            Log.e("MainActivity", "Image URL after parsing ($imageUri) || Image Url before parsing
${currentItem.ImageURL}")
            // Load the image using Glide
            Glide.with(holder.itemView.context)
                .load(imageUri)
                .timeout(60000) // Set a longer timeout (in milliseconds)
                .into(holder.bikeImage)

            holder.readMoreButton.setOnClickListener {
                val description = currentItem.description
                val image = currentItem.ImageURL
                val model = currentItem.model
                val brand = currentItem.brand
                val price = currentItem.price

                val intent = Intent(holder.itemView.context, ShowBicycle::class.java)
                intent.putExtra("description", description)
                intent.putExtra("image", image)
                intent.putExtra("model", model)
                intent.putExtra("brand", brand)
                intent.putExtra("price", price)
                intent.putExtra("Bicycle", currentItem)

                holder.itemView.context.startActivity(intent)
            }

            holder.addToCartLayout.setOnClickListener {
                addToCart(currentItem)
            }
        }
```

```
        override fun getItemCount() = bicycleList.size

        inner class BicycleViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
            val brandTextView: TextView = itemView.findViewById(R.id.textBrand)
            val priceTextView: TextView = itemView.findViewById(R.id.textPrice)
            val bikeImage: ImageView = itemView.findViewById(R.id.imageBicycle)
            val readMoreButton: Button = itemView.findViewById(R.id.buttonReadMore)
            val addToCartLayout = itemView.findViewById<LinearLayout>(R.id.AddToCart)
        }
    }
}
```

<----------> SignUp.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.app.DatePickerDialog
import android.content.Intent
import android.graphics.PorterDuff
import android.os.Bundle
import android.text.InputType
import android.text.method.HideReturnsTransformationMethod
import android.text.method.PasswordTransformationMethod
import android.util.Log
import android.util.Patterns
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.ProgressBar
import android.widget.RadioGroup
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.UserProfileChangeRequest
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import java.text.ParseException
import java.text.SimpleDateFormat
import java.util.Calendar
```

```
import java.util.Locale

class SignUp : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    private lateinit var database: FirebaseDatabase
    private lateinit var usersRef: DatabaseReference
    private lateinit var progressBar: ProgressBar

    @SuppressLint("MissingInflatedId", "SetTextI18n", "CutPasteId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_sign_up)

        // Initialize Firebase instances
        auth = FirebaseAuth.getInstance()
        database = FirebaseDatabase.getInstance()
        usersRef = database.getReference("users")

        // Call the function to register a new user
        progressBar = findViewById(R.id.loadingSignUpAct)
        val signupButton = findViewById<Button>(R.id.SignUpBtnSignUpAct)
        val birthDateButton = findViewById<ImageView>(R.id.ShowCalBtnSignUpAct)
        val calendar = Calendar.getInstance()
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();

        val dateSetListener = DatePickerDialog.OnDateSetListener { view, year, monthOfYear,
dayOfMonth ->
            // Set the selected date to the EditText field or use it as needed
            val selectedDate = "$dayOfMonth/${monthOfYear + 1}/$year"
            findViewById<EditText>(R.id.AgeSignUpAct).setText(selectedDate)
        }

        birthDateButton.setOnClickListener {
            val datePickerDialog = DatePickerDialog(
                this@SignUp,
                dateSetListener,
                calendar.get(Calendar.YEAR),
                calendar.get(Calendar.MONTH),
                calendar.get(Calendar.DAY_OF_MONTH)
            )
            datePickerDialog.datePicker.maxDate = System.currentTimeMillis() // Set max date as current
date
            datePickerDialog.show()
        }

        signupButton.setOnClickListener {
            // Fetch EditText values when the button is clicked
```

```
        val birthDate = findViewById<EditText>(R.id.AgeSignUpAct).text.toString() // Retrieve date of
birth
        val email = findViewById<EditText>(R.id.EmailSignUpAct).text.toString().trim()
        val userName = findViewById<EditText>(R.id.usernameSignUpAct).text.toString().trim()
        val password = findViewById<EditText>(R.id.passwordSignUpAct).text.toString()
        val password2 = findViewById<EditText>(R.id.ConfoirmPasswordSignUpAct).text.toString()
        val genderGroup = findViewById<RadioGroup>(R.id.radioGroupSignUpAct)
        val ValidationError = findViewById<TextView>(R.id.ValidationTextSignUpAct)
        val genderId = genderGroup.checkedRadioButtonId
        val selectedGender = when (genderId) {
            R.id.MaleRadioSignUpAct -> "Male"
            R.id.FemaleRadioSignUpAct -> "Female"
            else -> "Null" // Handle if no gender is selected
        }
        // Validating user's input
        if (isEmailValid(email))
            if (isPasswordMatched(password, password2))
                if (isPasswordValid(password))
                    if (selectedGender == "Null")
                        ValidationError.text = "Please select your gender"
                    else
                        if (isDateOfBirthValid(birthDate))
                            registerNewUser(email, password, userName, birthDate, selectedGender)
                else
                    ValidationError.text = "Please enter a valid date of birth"
                else
                    ValidationError.text = "Password should contain at least 6 characters"
            else
                ValidationError.text = "Password does not match"
        else
            ValidationError.text = "Please enter a valid email"
    }
  }

   private fun registerNewUser(email: String, password: String, userName: String, birthDate: String,
selectedGender: String) {
      progressBar.visibility = View.VISIBLE
      val errorText = findViewById<TextView>(R.id.ValidationTextSignUpAct)
      // Check if the username already exists
      usersRef.orderByChild("email").equalTo(email).addListenerForSingleValueEvent(object :
ValueEventListener {
          @SuppressLint("SetTextI18n")
          override fun onDataChange(dataSnapshot: DataSnapshot) {
              if (dataSnapshot.exists()) {
                  // Username already exists, handle accordingly (show error, prompt for a new username,
etc.)
                  Log.d("SignUpActivity", "email already exists")
                  errorText.text = "email already exists"
```

```
            } else {
                // Username doesn't exist, proceed with user registration
                auth.createUserWithEmailAndPassword(email, password)
                    .addOnCompleteListener(this@SignUp) { task ->
                        progressBar.visibility = View.GONE
                        if (task.isSuccessful) {
                            // User registration successful
                            val user = auth.currentUser
                            val profileUpdates = UserProfileChangeRequest.Builder()
                                .setDisplayName(userName) // Set the display name
                                .build()
                            user?.updateProfile(profileUpdates)
                                ?.addOnCompleteListener{ updateTask ->
                                    if (updateTask.isSuccessful) {
                                        Log.d("SignUpActivity", "User display name updated")
                                    } else {
                                        Log.d("SignUpActivity", "Error in updating the user's display name")
                                    }
                                }
                            user?.sendEmailVerification()
                                ?.addOnCompleteListener { verificationTask ->
                                    if (verificationTask.isSuccessful) {
                                        errorText.text = "Please check your email to verify your email"
                                    } else {
                                        errorText.text = "Error in verifying the email"
                                    }
                                }
                            Toast.makeText(this@SignUp, "Sign is completed !", Toast.LENGTH_SHORT).show()
                            // Save additional user data to the database
                            user?.uid?.let { userId ->
                                saveUserDataToDatabase(userId, userName, birthDate, selectedGender, email)
                            }
                        } else {
                            // Handle user registration failure
                            Log.e("SignUpActivity", "User registration failed: ${task.exception}")
                        }
                    }
            }
        }

        override fun onCancelled(databaseError: DatabaseError) {
            // Handle error while checking for username existence
            Log.e("SignUpActivity", "Error checking email existence: ${databaseError.message}")
        }
    })
}
```

```
    private fun saveUserDataToDatabase(userId: String, userName: String, birthDate: String, gender:
String, email: String) {
        val userData = HashMap<String, Any>()
        userData["username"] = userName
        userData["birthDate"] = birthDate
        userData["email"] = email
        userData["gender"] = gender // Save gender

        usersRef.child(userId).setValue(userData)
            .addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    // User data saved successfully
                    Log.d("SignUpActivity", "User data saved to database")
                } else {
                    // Handle failure to save user data
                    Log.e("SignUpActivity", "Failed to save user data: ${task.exception}")
                }
            }
    }
    private fun isPasswordMatched(password: String, password2: String): Boolean {
        return password == password2
    }
    fun isEmailValid(Email: String): Boolean {
        return if (Email.contains('@')) {
            Patterns.EMAIL_ADDRESS.matcher(Email).matches()
        } else {
            Email.isNotBlank()
        }
    }
        private fun isPasswordValid(password: String): Boolean {
        return password.length > 6
    }

    fun isDateOfBirthValid(dateOfBirth: String): Boolean {
        val dateFormat = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())
        dateFormat.isLenient = false // Set leniency to false for strict date parsing

        try {
            // Attempt to parse the date string
            dateFormat.parse(dateOfBirth)
            return true // Date format is valid
        } catch (e: ParseException) {
            return false // Date format is invalid
        }
    }
    fun AlreadyHaveAnAccount(view: View){
        val Intent = Intent(this, Login::class.java)
        startActivity(Intent)
```

```
    }

        fun homeIcon(view: View) {
            val intent = Intent(this@SignUp, MainActivity::class.java)
            startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@SignUp, Profile::class.java)
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@SignUp, Cart::class.java)
        startActivity(intent)
    }


}
```

<----------> Login.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Intent
import android.graphics.PorterDuff
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.ContactsContract.Profile
import android.util.Log
import android.util.Patterns
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.ProgressBar
import android.widget.TextView
import androidx.appcompat.app.ActionBar
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.UserProfileChangeRequest
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class Login : AppCompatActivity() {
    @SuppressLint("SetTextI18n")
```

```
private lateinit var auth: FirebaseAuth
private lateinit var database: FirebaseDatabase


@SuppressLint("MissingInflatedId", "SetTextI18n")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_login)

    // Initialize Firebase instances
    auth = FirebaseAuth.getInstance()
    database = FirebaseDatabase.getInstance() // Initialize FirebaseDatabase instance
    val LoginBtn = findViewById<Button>(R.id.LoginBtnLoginAct)
    val loginFiled = findViewById<EditText>(R.id.EmailLoginAct)
    val passwordFiled = findViewById<EditText>(R.id.passwordLoginAct)
    val ValadationText = findViewById<TextView>(R.id.ValidationTextLoginAct)
    val actionBar: ActionBar? = supportActionBar
    actionBar?.hide();
    LoginBtn.setOnClickListener{
        if ((loginFiled.text.toString() == "admin") && (passwordFiled.text.toString() == "admin")) {
            val user = auth.currentUser
            val Intent = Intent(this, adminLogin::class.java)
            startActivity(Intent)
        } else {
            if (isEmailValid(loginFiled.text.toString().trim()))
                loginUser(loginFiled.text.toString().trim(), passwordFiled.text.toString())
            else
                ValadationText.text = "Please enter a valid email"
        }
    }
}

@SuppressLint("SetTextI18n")
private fun loginUser(email: String, password: String) {
    val errorText = findViewById<TextView>(R.id.ErrorTextLogin)

    auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                val user = auth.currentUser
                if (user != null && !user.isEmailVerified) {
                    user.sendEmailVerification()
                        .addOnCompleteListener { verificationTask ->
                            if (verificationTask.isSuccessful) {
                                errorText.text = "Please check your email to verify your email"
                                Log.d("LoginActivity", "Verification email sent")
                            } else {
                                errorText.text = "Error in verification"
```

```
                        Log.e(
                            "LoginActivity",
                            "Error in verification",
                            verificationTask.exception
                        )
                    }
                }
            } else {
                startActivity(Intent(this@Login, MainActivity::class.java))
            }
        } else {
            errorText.text = "Login failed: ${task.exception?.message}"
            Log.e("LoginActivity", "Login failed: ${task.exception}")
        }
    }
}

fun ForgotPassBtn(view: View){
    val Intent = Intent(this@Login, ResetPasswordActivity::class.java)
    startActivity(Intent)
}

fun NotRegisteredBtn(view: View){
    val Intent = Intent(this@Login, SignUp::class.java)
    startActivity(Intent)
}

fun isEmailValid(Email: String): Boolean {
    return if (Email.contains('@')) {
        Patterns.EMAIL_ADDRESS.matcher(Email).matches()
    } else {
        Email.isNotBlank()
    }
}
    fun homeIcon(view: View) {
        val intent = Intent(this@Login, MainActivity::class.java)
        startActivity(intent)
}
fun profileIcon(view: View) {
    val intent = Intent(this@Login, Profile::class.java)
    startActivity(intent)
}
fun cartIcon(view: View){
    val intent = Intent(this@Login, Cart::class.java)
    startActivity(intent)
}
}
```

<--> ResetPasswordActivity.kt: <--
--->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Intent
import android.graphics.PorterDuff
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Patterns
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import com.google.firebase.auth.FirebaseAuth

class   ResetPasswordActivity : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    @SuppressLint("MissingInflatedId", "SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_reset_password)
        auth = FirebaseAuth.getInstance()
        val validationText = findViewById<TextView>(R.id.ValidationTextResetPassAct)
        val resetButton = findViewById<Button>(R.id.SendEmailBtnResestPassAct)
        val emailEditText = findViewById<EditText>(R.id.EmailFieldResetPassAct)
        val profileIcon = findViewById<ImageView>(R.id.profileIconResetPassAct).drawable
        profileIcon.setColorFilter(resources.getColor(R.color.GoldenColor), PorterDuff.Mode.SRC_IN)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();

        resetButton.setOnClickListener {
            val email = emailEditText.text.toString().trim()

            if (email.isEmpty()) {
                // Handle empty email
                validationText.text = "Please enter an email"
            } else {
                if (isEmailValid(email)) {
                    auth.sendPasswordResetEmail(email)
                        .addOnCompleteListener { task ->
```

```
                    if (task.isSuccessful) {
                        // Password reset email sent successfully
                        Toast.makeText(
                            this,
                            "Password reset email sent to $email",
                            Toast.LENGTH_SHORT
                        ).show()
                        finish() // Finish the activity after sending the email
                    } else {
                        // Handle password reset failure
                        Toast.makeText(
                            this,
                            "Failed to send password reset email",
                            Toast.LENGTH_SHORT
                        ).show()
                    }
                }
            } else
                validationText.text = "Please enter a valid email"
        }
    }
}
    fun homeIcon(view: View) {
        val intent = Intent(this@ResetPasswordActivity, MainActivity::class.java)
        startActivity(intent)
}
fun profileIcon(view: View) {
    val intent = Intent(this@ResetPasswordActivity, Profile::class.java)
    startActivity(intent)
}
fun cartIcon(view: View){
    val intent = Intent(this@ResetPasswordActivity, Cart::class.java)
    startActivity(intent)
}
fun isEmailValid(Email: String): Boolean {
    return if (Email.contains('@')) {
        Patterns.EMAIL_ADDRESS.matcher(Email).matches()
    } else {
        Email.isNotBlank()
    }
}
}
```

<----------> Cart.kt: <---------->

```
package com.example.mwadfinalproject
```

```
import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.media.Image
import android.net.Uri
import com.example.mwadfinalproject.Bicycle
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.google.firebase.auth.FirebaseAuth
import com.google.gson.Gson

class Cart : AppCompatActivity() {
    private lateinit var cartRecyclerView: RecyclerView
    private lateinit var cartAdapter: CartAdapter
    private lateinit var cartItems: MutableList<Bicycle>
    private lateinit var auth: FirebaseAuth
    private lateinit var authListener: FirebaseAuth.AuthStateListener

    inner class CartAdapter(private val cartItemList: MutableList<Bicycle>) :
        RecyclerView.Adapter<CartAdapter.CartViewHolder>() {

        override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CartViewHolder {
            val itemView = LayoutInflater.from(parent.context)
                .inflate(R.layout.item_holder_cart, parent, false)
            return CartViewHolder(itemView)
        }

      private fun getCartItemsFromSharedPreferences(): MutableList<Bicycle> {
          val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
          val cartItemsJsonSet = sharedPreferences.getStringSet("cart_items", emptySet())

          val gson = Gson()
          val cartItems = mutableListOf<Bicycle>()

          cartItemsJsonSet?.forEach { jsonItem ->
```

```
            val item = gson.fromJson(jsonItem, Bicycle::class.java)
            cartItems.add(item)
        }

        return cartItems
    }
    @SuppressLint("NotifyDataSetChanged")
    fun removeItem(position: Int, context: Context) {
        cartItemList.removeAt(position)
        notifyItemRemoved(position)
        notifyItemRangeChanged(position, cartItemList.size)
        saveCartItemsToSharedPreferences(cartItemList)
        (context as Cart).updateCartItems(cartItemList)
    }

    private fun saveCartItemsToSharedPreferences(cartItems: List<Bicycle>) {
        val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
        val editor = sharedPreferences.edit()
        val gson = Gson()
        val cartItemsJsonSet = cartItems.map { gson.toJson(it) }.toSet()
        editor.putStringSet("cart_items", cartItemsJsonSet)
        editor.apply()
    }

     @SuppressLint("SetTextI18n", "NotifyDataSetChanged")
     override fun onBindViewHolder(holder: CartViewHolder, position: Int) {

         val currentItem = cartItemList[position]
         val imageUri = Uri.parse(currentItem.ImageURL)
         val PriceToString = currentItem.price.toString()

         // Set the data to your views in the ViewHolder
         holder.textBrand.text = currentItem.brand
         holder.textModel.text = currentItem.model
         holder.textPrice.text = "$PriceToString$"
         holder.quantityTextView.text = currentItem.quantity.toString()
         Glide.with(holder.itemView.context)
             .load(imageUri)
             .into(holder.BikeImage)



         holder.increaseButton.setOnClickListener {
             val updatedItem = cartItemList[position].copy(quantity = cartItemList[position].quantity + 1)
             cartItemList[position] = updatedItem
             saveCartItemsToSharedPreferences(cartItemList) // Save the updated list
             updateCartItems(cartItemList) // Notify the adapter about the dataset change
             updateTotalPrice()
```

```
            Toast.makeText(this@Cart, "Item's quantity has been increased",
Toast.LENGTH_SHORT).show()
        }

        holder.decreaseButton.setOnClickListener {
            if(cartItemList[position].quantity == 1) {
                Toast.makeText(this@Cart, "minimum quantity is 1", Toast.LENGTH_SHORT).show()
            } else {
                if (cartItemList[position].quantity > 0) {
                    val updatedItem =
                        cartItemList[position].copy(quantity = cartItemList[position].quantity - 1)
                    cartItemList[position] = updatedItem
                    saveCartItemsToSharedPreferences(cartItemList) // Save the updated list
                    updateCartItems(cartItemList) // Notify the adapter about the dataset change
                    updateTotalPrice()
                    Toast.makeText(
                        this@Cart,
                        "Item's quantity has been decreased",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }
        }




        holder.deleteButton.setOnClickListener {
            removeItem(position, holder.itemView.context)
            updateTotalPrice()
        }
    }

    override fun getItemCount() = cartItemList.size

    inner class CartViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val textBrand: TextView = itemView.findViewById(R.id.textBrandCartItem)
        val textModel: TextView = itemView.findViewById(R.id.textModelCartItem)
        val textPrice: TextView = itemView.findViewById(R.id.textPriceCartItem)
        val BikeImage: ImageView = itemView.findViewById(R.id.imageCartItem)
        val decreaseButton: ImageView = itemView.findViewById(R.id.imageDecreaseQuantity)
        val increaseButton: ImageView = itemView.findViewById(R.id.imageIncreaseQuantity)
        val deleteButton: ImageView = itemView.findViewById(R.id.imageDeleteCartItem)
        val quantityTextView: TextView = itemView.findViewById(R.id.textQuantityCartItem)
        // Other views in the cart item layout
    }
}
fun updateCartItems(updatedCartItems: MutableList<Bicycle>) {
```

```
        cartItems.clear()
        cartItems.addAll(updatedCartItems)
        cartAdapter.notifyDataSetChanged() // Notify the adapter about the dataset change
        updateTotalPrice()
        Log.d("CartActivity", "updateCartItems() called")
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_cart)
        // Initialize RecyclerView and its adapter
        cartRecyclerView = findViewById(R.id.recyclerViewCartAct)
        cartAdapter = CartAdapter(getCartItems().toMutableList())
        cartRecyclerView.layoutManager = LinearLayoutManager(this)
        cartRecyclerView.adapter = cartAdapter
        cartItems = getCartItems().toMutableList() // Initialize cartItems
        updateTotalPrice()
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();
        val checkOutButton = findViewById<Button>(R.id.checkOutButton)
        checkOutButton.setOnClickListener {
            if (cartItems.isEmpty()) {
                Toast.makeText(this, "Pleases add items before checking out.",
Toast.LENGTH_SHORT).show()
            } else {
                auth = FirebaseAuth.getInstance()
                authListener = FirebaseAuth.AuthStateListener { firebaseAuth ->
                    val user = firebaseAuth.currentUser
                    if (user == null) {
                        // User is not logged in, navigate to the Login or Sign Up activity
                        val intent = Intent(this@Cart, LoginOrSignUp::class.java)
                        startActivity(intent)
                    } else {
                        // User is logged in, navigate to the Checkout activity
                        val intent = Intent(this@Cart, CheckOut::class.java)
                        intent.putExtra("TotalPrice", calculateTotalPrice())
                        intent.putExtra("CartItems", ArrayList(getCartItems()))
                        startActivity(intent)
                    }
                }
                // Add the AuthStateListener to check the user's authentication status
                auth.addAuthStateListener(authListener)
            }
        }

    }
    private fun calculateTotalPrice(): Int {
        var totalPrice = 0
```

```
        // Iterate through the cart items and calculate the total price
        for (item in cartItems) {
            totalPrice += item.price * item.quantity
        }
        return totalPrice
    }

    // Function to update the total price in the UI
    @SuppressLint("SetTextI18n")
    private fun updateTotalPrice() {
        val total = calculateTotalPrice()
        val totalPriceTextView = findViewById<TextView>(R.id.totalPriceTextView)
        totalPriceTextView.text = "Total Price: $total$" // Update the TextView with the total price
        Log.e("Cart", "update Total Is Called")
    }

        fun homeIcon(view: View) {
            val intent = Intent(this@Cart, MainActivity::class.java)
            startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@Cart, Profile::class.java)
        startActivity(intent)
    }
//    fun cartIcon(view: View){
//        val intent = Intent(this@Cart, Cart::class.java)
//        startActivity(intent)
//    }


    // Retrieve cart items from SharedPreferences
    private fun getCartItems(): MutableList<Bicycle> {
        val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
        val cartItemsJsonSet = sharedPreferences.getStringSet("cart_items", emptySet())

        val gson = Gson()
        val cartItems = mutableListOf<Bicycle>()

        cartItemsJsonSet?.forEach { jsonItem ->
            val item = gson.fromJson(jsonItem, Bicycle::class.java)
            cartItems.add(item)
        }

        return cartItems
    }
```

```
}
```

<----------> Bicycle.kt: <---------->

```
package com.example.mwadfinalproject

import android.graphics.ColorSpace.Model
import android.os.Parcel
import android.os.Parcelable

data class Bicycle(
    val id: String,
    val brand: String,
    val model: String,
    val description: String,
    val price: Int,
    val ImageURL: String,
    var addedToCart: Boolean,
    var quantity: Int
) : Parcelable {
    constructor() : this("", "", "", "", 0, "", false, 0)
    // Implement Parcelable methods here

    // Example constructor for Parcelable
    constructor(parcel: Parcel) : this(
        parcel.readString() ?: "",
        parcel.readString() ?: "",
        parcel.readString() ?: "",
        parcel.readString() ?: "",
        parcel.readInt(),
        parcel.readString() ?: "",
        parcel.readByte() != 0.toByte(),
        parcel.readInt()
    )

    override fun writeToParcel(parcel: Parcel, flags: Int) {
        parcel.writeString(id)
        parcel.writeString(brand)
        parcel.writeString(model)
        parcel.writeString(description)
        parcel.writeInt(price)
        parcel.writeString(ImageURL)
        parcel.writeByte(if (addedToCart) 1 else 0)
        parcel.writeInt(quantity)
    }
```

```
    override fun describeContents(): Int {
        return 0
    }

    companion object CREATOR : Parcelable.Creator<Bicycle> {
        override fun createFromParcel(parcel: Parcel): Bicycle {
            return Bicycle(parcel)
        }

        override fun newArray(size: Int): Array<Bicycle?> {
            return arrayOfNulls(size)
        }
    }
}
```

<----------> Order.kt: <---------->

```
package com.example.mwadfinalproject

data class Order(
    val orderId: String? = null,
    val items: List<OrderItem> = emptyList(),
    val totalAmount: Double = 0.0,
    val orderDate: String? = null,
    val status: String? = "Confirmed"
    // Add other properties as needed
) {
    // Add a no-argument constructor
    constructor() : this("", emptyList(), 0.0, "", "Confirmed")
}

data class OrderItem(
    val bicycleId: String? = null,
    val brand: String? = null,
    val model: String? = null,
    val quantity: Int = 0,
    val price: Double = 0.0
) {
    // Add a no-argument constructor
    constructor() : this("", "", "", 0, 0.0)
}
```

<----------> Orders.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.widget.LinearLayout
import android.widget.TextView
import androidx.appcompat.app.ActionBar
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class Orders : AppCompatActivity() {
    private lateinit var ordersRef: DatabaseReference
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_orders)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();
        val userId = FirebaseAuth.getInstance().currentUser?.uid
        ordersRef = FirebaseDatabase.getInstance().getReference("orders").child(userId.orEmpty())
        fetchOrders()
    }
    private fun fetchOrders() {
        val ordersList = mutableListOf<Order>()

        ordersRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                for (orderSnapshot in snapshot.children) {
                    val order = orderSnapshot.getValue(Order::class.java)
                    order?.let { ordersList.add(it) }
                }

                // Display the fetched orders
                displayOrders(ordersList)
            }

            override fun onCancelled(error: DatabaseError) {
                Log.e("Orders Activity", "Error fetching the orders")
            }
        })
```

```
    }
    @SuppressLint("SetTextI18n")
    private fun displayOrders(orders: List<Order>) {
        val itemsContainer = findViewById<LinearLayout>(R.id.itemsContainer)
        val inflater = LayoutInflater.from(this@Orders)

        for (order in orders) {
            val orderView = inflater.inflate(R.layout.item_holder_orders, itemsContainer, false)

            val orderDateTextView = orderView.findViewById<TextView>(R.id.textBrandCartItem)
            val orderStatusTextView = orderView.findViewById<TextView>(R.id.textModelCartItem)
            val ShowOrderBtn = orderView.findViewById<LinearLayout>(R.id.ShowOrderBtn)

            orderDateTextView.text = "Ordered on ${order.orderDate}"
            orderStatusTextView.text = "Order Status: ${order.status}"
            ShowOrderBtn.setOnClickListener{
                val intent = Intent(this@Orders, ShowOrder::class.java)
                intent.putExtra("OrderId", order.orderId)
                startActivity(intent)
            }

            itemsContainer.addView(orderView)
        }
    }
    fun homeIcon(view: View) {
        val intent = Intent(this@Orders, MainActivity::class.java)
        startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@Orders, Profile::class.java)
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@Orders, Cart::class.java)
        startActivity(intent)
    }
}
```

<----------> Profile.kt: <---------->

```
package com.example.mwadfinalproject

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
```

```
import android.widget.Button
import android.widget.LinearLayout
import androidx.appcompat.app.ActionBar
import com.google.firebase.auth.FirebaseAuth

class Profile : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    private lateinit var authListener: FirebaseAuth.AuthStateListener

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_profile)
        val SignOutBtn = findViewById<LinearLayout>(R.id.SignOutProfileAct)
        val ManageAccBtn = findViewById<LinearLayout>(R.id.ManageAccountProfileAct)
        val OrdersBtn = findViewById<LinearLayout>(R.id.OrdersProfileAct)

        OrdersBtn.setOnClickListener{
            startActivity(Intent(this, Orders::class.java))
        }
        ManageAccBtn.setOnClickListener{
            startActivity(Intent(this, ManageAccount::class.java))
        }

        auth = FirebaseAuth.getInstance()
        authListener = FirebaseAuth.AuthStateListener { firebaseAuth ->
            val user = firebaseAuth.currentUser
            val actionBar: ActionBar? = supportActionBar
            actionBar?.hide();
            if (user == null) {
                intent = Intent(this@Profile, LoginOrSignUp::class.java)
                startActivity(intent)
            }
        }
        SignOutBtn.setOnClickListener {
//          // Sign the user out
            auth.signOut()
        }

    }
    override fun onStart() {
        super.onStart()
        // Add the AuthStateListener when the activity starts
        auth.addAuthStateListener(authListener)
    }

    override fun onStop() {
        super.onStop()
        // Remove the AuthStateListener when the activity stops
```

```
        auth.removeAuthStateListener(authListener)
    }

        fun homeIcon(view: View) {
            val intent = Intent(this@Profile, MainActivity::class.java)
            startActivity(intent)
    }
//    fun profileIcon(view: View) {
//        val intent = Intent(this@Profile, Profile::class.java)
//        startActivity(intent)
//    }
    fun cartIcon(view: View){
        val intent = Intent(this@Profile, Cart::class.java)
        startActivity(intent)
    }
}
```

<--> ManageAccount.kt: <-->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.app.DatePickerDialog
import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.util.Patterns
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.RadioButton
import android.widget.RadioGroup
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import androidx.appcompat.app.AppCompatActivity
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.UserProfileChangeRequest
import com.google.firebase.database.*
import java.text.ParseException
import java.text.SimpleDateFormat
import java.util.Calendar
import java.util.Locale

class ManageAccount : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
```

```
    private lateinit var usersRef: DatabaseReference
    private lateinit var database: FirebaseDatabase
    private lateinit var email: String


    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_manage_account)
        auth = FirebaseAuth.getInstance()
        usersRef = FirebaseDatabase.getInstance().getReference("users").child(auth.currentUser?.uid ?:
"")

        val editTextUsername = findViewById<TextView>(R.id.EmailTextManageAccAct)
        val editTextEmail = findViewById<EditText>(R.id.editTextUserName)
        val editTextBirthDate = findViewById<EditText>(R.id.editTextBirthDate)
        val radioGroupGender = findViewById<RadioGroup>(R.id.radioGroupGender)
        val radioButtonMale = findViewById<RadioButton>(R.id.radioButtonMale)
        val radioButtonFemale = findViewById<RadioButton>(R.id.radioButtonFemale)
        val buttonUpdateProfile = findViewById<Button>(R.id.buttonUpdateProfile)
        val calendar = Calendar.getInstance()
        val birthDateButton = findViewById<ImageView>(R.id.ShowCalBtnEditProfileAct)
        val ValidationError = findViewById<TextView>(R.id.ValidationTextEditProfileAct)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();
        val dateSetListener = DatePickerDialog.OnDateSetListener { view, year, monthOfYear,
dayOfMonth ->
            // Set the selected date to the EditText field or use it as needed
            val selectedDate = "$dayOfMonth/${monthOfYear + 1}/$year"
            findViewById<EditText>(R.id.editTextBirthDate).setText(selectedDate)
        }

        birthDateButton.setOnClickListener {
            val datePickerDialog = DatePickerDialog(
                this@ManageAccount,
                dateSetListener,
                calendar.get(Calendar.YEAR),
                calendar.get(Calendar.MONTH),
                calendar.get(Calendar.DAY_OF_MONTH)
            )
            datePickerDialog.datePicker.maxDate = System.currentTimeMillis() // Set max date as current
date
            datePickerDialog.show()
        }

        // Fetch current user data and populate the fields
        usersRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
```

```
            val user = snapshot.getValue(User::class.java)
            editTextUsername.text = "Email: ${user?.email}"
            editTextEmail.setText(user?.username)
            editTextBirthDate.setText(user?.birthDate)
            email = user?.email.toString()
            // Set the gender based on fetched data
            if (user?.gender == "Male") {
                radioButtonMale.isChecked = true
            } else if (user?.gender == "Female") {
                radioButtonFemale.isChecked = true
            }
        }

        override fun onCancelled(error: DatabaseError) {
            // Handle error
        }
    })

    buttonUpdateProfile.setOnClickListener {
        val userName = editTextEmail.text.toString()
        val birthDate = editTextBirthDate.text.toString()
        val selectedGender = when (radioGroupGender.checkedRadioButtonId) {
            R.id.radioButtonMale -> "Male"
            R.id.radioButtonFemale -> "Female"
            else -> "Null"
        }

        // Update user data in Firebase Database
        // Validating user's input

usersRef.orderByChild("username").equalTo(userName).addListenerForSingleValueEvent(object :
ValueEventListener {
            @SuppressLint("SetTextI18n")
            override fun onDataChange(dataSnapshot: DataSnapshot) {
                if (dataSnapshot.exists()) {
                    ValidationError.text = "Username already exists please choose a another one"
                } else {
                    if (isDateOfBirthValid(birthDate)) {
                        if (selectedGender == "Null") {
                            ValidationError.text = "Please Select a gender"
                        } else {
                            val user = auth.currentUser
                            val userData = User(userName, email, birthDate, selectedGender)
                            usersRef.setValue(userData)
                                .addOnCompleteListener { task ->
                                    if (task.isSuccessful) {
                                        val profileUpdates = UserProfileChangeRequest.Builder()
                                            .setDisplayName(userName) // Set the display name
```

```
                        .build()
                    user?.updateProfile(profileUpdates)
                        ?.addOnCompleteListener{ updateTask ->
                            if (updateTask.isSuccessful) {
                                Log.d("ManageAccount", "User display name updated")
                            } else {
                                Log.d("ManageAccount", "Error in updating the user's display name")
                            }
                        }
                    Toast.makeText(this@ManageAccount, "Profile updated",
Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(this@ManageAccount, "Failed to update profile",
Toast.LENGTH_SHORT).show()
                }
            }
        }
    } else
        ValidationError.text = "Please enter a valid date of birth"
}
}

override fun onCancelled(databaseError: DatabaseError) {
    // Handle error while checking for username existence
    Log.e("SignUpActivity", "Error checking username existence: ${databaseError.message}")
}
})
}
}
data class User(
    val username: String = "",
    val email: String = "",
    val birthDate: String = "",
    val gender: String = ""
)
private fun isPasswordValid(password: String): Boolean {
    return password.length > 6
}

fun isDateOfBirthValid(dateOfBirth: String): Boolean {
    val dateFormat = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault())
    dateFormat.isLenient = false // Set leniency to false for strict date parsing

    try {
        // Attempt to parse the date string
        dateFormat.parse(dateOfBirth)
        return true // Date format is valid
    } catch (e: ParseException) {
```

```
            return false // Date format is invalid
        }
    }
    fun homeIcon(view: View) {
        val intent = Intent(this@ManageAccount, MainActivity::class.java)
        startActivity(intent)
    }
        fun profileIcon(view: View) {
        val intent = Intent(this@ManageAccount, Profile::class.java)
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@ManageAccount, Cart::class.java)
        startActivity(intent)
    }
}
```

<----------> AfterCheckOut.kt: <---------->

```
package com.example.mwadfinalproject

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button
import androidx.appcompat.app.ActionBar

class AfterCheckOut : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_after_check_out)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();
        val OrdersPagebtn = findViewById<Button>(R.id.OrdersPage)
        val continueShoppingBtn = findViewById<Button>(R.id.continueShoppingBtn)

        OrdersPagebtn.setOnClickListener{
            startActivity(Intent(this, Order::class.java))
        }
        continueShoppingBtn.setOnClickListener{
            startActivity(Intent(this, MainActivity::class.java))
        }
    }

    fun homeIcon(view: View) {
```

```
        val intent = Intent(this@AfterCheckOut, MainActivity::class.java)
        startActivity(intent)
    }

    fun profileIcon(view: View) {
        val intent = Intent(this@AfterCheckOut, Profile::class.java)
        startActivity(intent)
    }

    fun cartIcon(view: View) {
        val intent = Intent(this@AfterCheckOut, Cart::class.java)

    }
}
```

<----------> CheckOut.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Context
import java.text.ParseException
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.LinearLayout
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.FirebaseDatabase
import com.google.gson.Gson
import java.text.SimpleDateFormat
import java.util.Date
import java.util.Locale
import java.util.UUID

class CheckOut : AppCompatActivity() {
    private val database = FirebaseDatabase.getInstance()
    private val auth = FirebaseAuth.getInstance()
    private val currentUser = auth.currentUser
```

```
    private val ordersRef = currentUser?.uid?.let {
database.getReference("users").child(it).child("subOrders") }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_check_out)
        val ValidtionText = findViewById<TextView>(R.id.ValidationTextCheckOutAct)
        val cardNumber = findViewById<EditText>(R.id.CardNumberFieldCheckOutAct)
        val CardYear = findViewById<EditText>(R.id.YearFiledCheckOutAct)
        val CardMonth = findViewById<EditText>(R.id.MonthFiledCheckOutAct)
        val CardCCV = findViewById<EditText>(R.id.ccvFiledCheckOutAct).text
        val NameOnCard = findViewById<EditText>(R.id.NameOnCardFieldCheckOutAct).text.toString()
        val TotalPriceText = findViewById<TextView>(R.id.TotalPriceCheckOutAct)
        val PlaceOrderBtn = findViewById<Button>(R.id.PlaceOrderBtn)
        TotalPriceText.text = "Total Price: ${calculateTotalPrice()}$"
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide()
        displayCartItems()
        PlaceOrderBtn.setOnClickListener {
            var CardDate: String = "${CardMonth.text}/${CardYear.text}"
            if (isValidCreditCardNumber(cardNumber.text.toString())) {
                if (isValidCreditCardExpiration(CardDate)) {
                    if (isValidCCV(CardCCV.toString())) {
                        placeOrder()
                        startActivity(Intent(this, AfterCheckOut::class.java))
                    } else
                        ValidtionText.text = "please enter a valid CCV number"
                } else
                    ValidtionText.text = "Please enter a valid card expiration date"
            } else
                ValidtionText.text = "Please enter a valid card number"
            Log.e("CheckOut Activity", "Date: $CardDate")
            Log.e("CheckOut Activity", "CVV: $CardCCV")
        }
    }
    private fun placeOrder() {
        // Get user information
        val userId = FirebaseAuth.getInstance().currentUser?.uid

        // Check if the user is signed in
        if (userId != null) {
            val cartItems = getCartItems()

            // Check if the cart is not empty
            if (cartItems.isNotEmpty()) {
                val orderItems = cartItems.map {
                    OrderItem(
                        bicycleId = it.id,
```

```
                brand = it.brand,
                model = it.model,
                quantity = it.quantity,
                price = it.price.toDouble()
            )
        }

        val totalAmount = cartItems.sumByDouble { it.price.toDouble() * it.quantity }

        // Create an Order object
        val order = Order(
            orderId = generateOrderId(),
            items = orderItems,
            totalAmount = totalAmount,
            orderDate = getCurrentDateTime(),
            )

        // Save the order to Firebase Realtime Database
        saveOrderToDatabase(userId, order)

        // Clear the cart after placing the order
        clearCart()

        Toast.makeText(this, "Order placed successfully!", Toast.LENGTH_SHORT).show()
      } else {
        Toast.makeText(this, "Your cart is empty. Add items before placing an order.",
Toast.LENGTH_SHORT).show()
      }
    } else {
      // Handle the case where the user is not signed in
      Toast.makeText(this, "User not signed in. Please sign in to place an order.",
Toast.LENGTH_SHORT).show()
    }
  }

  // Function to generate a unique order ID
  private fun generateOrderId(): String {
    return UUID.randomUUID().toString()
  }

  // Function to get the current date and time
  private fun getCurrentDateTime(): String {
    val sdf = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault())
    return sdf.format(Date())
  }

  // Function to save the order to Firebase Realtime Database
  private fun saveOrderToDatabase(userId: String, order: Order) {
```

```
    val database = FirebaseDatabase.getInstance()
    val ordersRef = database.getReference("orders")

    // Save the order under the user's ID
    ordersRef.child(userId).child(order.orderId ?: "").setValue(order)
}

// Function to clear the cart after placing an order
private fun clearCart() {
    val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
    val editor = sharedPreferences.edit()
    editor.clear()
    editor.apply()
}
private fun getCartItems(): Set<Bicycle> {
    val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
    val cartItemsSet = sharedPreferences.getStringSet("cart_items", emptySet()) ?: emptySet()
    return cartItemsSet.map { Gson().fromJson(it, Bicycle::class.java) }.toSet()
}
fun homeIcon(view: View) {
    val intent = Intent(this@CheckOut, MainActivity::class.java)
    startActivity(intent)
}
fun profileIcon(view: View) {
    val intent = Intent(this@CheckOut, Profile::class.java)
    startActivity(intent)
}
fun cartIcon(view: View){
    val intent = Intent(this@CheckOut, Cart::class.java)
    startActivity(intent)
}
@SuppressLint("SetTextI18n")
private fun displayCartItems() {
    val cartItems = getCartItems()

    val itemsContainer = findViewById<LinearLayout>(R.id.itemsContainer)
    val inflater = LayoutInflater.from(this@CheckOut)

    for (item in cartItems) {
        val itemView = inflater.inflate(R.layout.check_out_list, itemsContainer, false)
        val itemName = itemView.findViewById<TextView>(R.id.itemName)
        val itemQuantity = itemView.findViewById<TextView>(R.id.itemQuantity)

        itemName.text = item.brand // Set the name of the item
        itemQuantity.text = "X ${item.quantity}" // Set the quantity of the item

        itemsContainer.addView(itemView)
    }
```

```
    }

    fun isValidCreditCardExpiration(expirationDate: String): Boolean {
        try {
            // Define the expected date format
            val dateFormat = SimpleDateFormat("MM/yy", Locale.US)
            dateFormat.isLenient = false  // Disable lenient parsing

            // Parse the expiration date string into a Date object
            val expiration = dateFormat.parse(expirationDate)

            // Get the current date
            val currentDate = Date()

            // Check if the expiration date is in the future and not before the current date
            return expiration != null && expiration.after(currentDate)
        } catch (e: ParseException) {
            // Handle parsing exceptions, such as invalid date format
            return false
        }
    }

    fun isValidCreditCardNumber(cardNumber: String): Boolean {
        return cardNumber.length == 16
    }
    fun isValidCCV(ccv: String): Boolean {
        // Check if the CCV contains only digits and has a valid length
        return ccv.matches(Regex("\\d+")) && ccv.length in 3..4
    }
    private fun calculateTotalPrice(): Int {
        val cartItems = getCartItems()
        var totalPrice = 0

        // Iterate through the cart items and calculate the total price
        for (item in cartItems) {
            totalPrice += item.price * item.quantity
        }
        return totalPrice
    }


}
```

<----------> ShowOrder.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.widget.LinearLayout
import android.widget.TextView
import androidx.appcompat.app.ActionBar
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class ShowOrder : AppCompatActivity() {
    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_show_order)
        val orderId = intent.getStringExtra("OrderId")
        val totalPaidText = findViewById<TextView>(R.id.TotalPaid)
        val OrderDateText = findViewById<TextView>(R.id.OrderDateShowOrderAct)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();
        displayOrderItems(orderId.toString())
        val userId = FirebaseAuth.getInstance().currentUser?.uid

        getOrderDetails(orderId.toString(), userId.toString(),
            onSuccess = { orderDetails ->
                // Use the order details as needed
                totalPaidText.text = "Total paid: ${orderDetails.totalAmount}$"
                OrderDateText.text = "Ordered On: ${orderDetails.orderDate}"
            },
            onError = { error ->
                // Handle the error
                Log.e("Show Order Activity", "Error: ${error.message}")
            }
        )
    }
    @SuppressLint("SetTextI18n")
    private fun displayOrderItems(orderId: String) {
        val userId = FirebaseAuth.getInstance().currentUser?.uid
        val itemsContainer = findViewById<LinearLayout>(R.id.itemsContainerShowOrdersAct)
        val inflater = LayoutInflater.from(this@ShowOrder)
```

```
    // Reference to the items for the specific order
    val orderItemsRef =
FirebaseDatabase.getInstance().getReference("orders").child(userId.orEmpty()).child(orderId).child("i
tems")

    orderItemsRef.addListenerForSingleValueEvent(object : ValueEventListener {
      override fun onDataChange(snapshot: DataSnapshot) {
        for (itemSnapshot in snapshot.children) {
          val item = itemSnapshot.getValue(Bicycle::class.java)

          // Assuming Bicycle class has 'brand' and 'quantity' properties
          item?.let {
            val itemView = inflater.inflate(R.layout.check_out_list, itemsContainer, false)
            val itemName = itemView.findViewById<TextView>(R.id.itemName)
            val itemQuantity = itemView.findViewById<TextView>(R.id.itemQuantity)

            itemName.text = it.brand // Set the name of the item
            itemQuantity.text = "X ${it.quantity}" // Set the quantity of the item

            itemsContainer.addView(itemView)
          }
        }
      }

      override fun onCancelled(error: DatabaseError) {
        // Handle the error
      }
    })
  }
  data class OrderDetails(
    val orderDate: String,
    val totalAmount: Double
  )

  fun getOrderDetails(orderId: String, userId: String, onSuccess: (OrderDetails) -> Unit, onError:
(DatabaseError) -> Unit) {
    // Reference to the specific order
    val orderRef =
FirebaseDatabase.getInstance().getReference("orders").child(userId).child(orderId)

    // Add a listener to retrieve the order details
    orderRef.addListenerForSingleValueEvent(object : ValueEventListener {
      override fun onDataChange(snapshot: DataSnapshot) {
        // Check if the order exists
        if (snapshot.exists()) {
          // Get the order date and total amount from the order
          val orderDate = snapshot.child("orderDate").getValue(String::class.java)
```

```
                val totalAmount = snapshot.child("totalAmount").getValue(Double::class.java)
                val OrderStatus = snapshot.child("")

                // Call the onSuccess callback with the order details
                if (orderDate != null && totalAmount != null) {
                    val orderDetails = OrderDetails(orderDate, totalAmount)
                    onSuccess(orderDetails)
                } else {
                    // Handle the case where orderDate or totalAmount is null
                    onError(DatabaseError.fromException(Exception("Order details are null")))
                }
            } else {
                // Handle the case where the order does not exist
                onError(DatabaseError.fromException(Exception("Order does not exist")))
            }
        }

        override fun onCancelled(error: DatabaseError) {
            // Handle the error
            onError(error)
        }
    })
  }


}
```

<----------> ShowBicycle.kt <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import com.bumptech.glide.Glide
import com.google.gson.Gson
```

```
class ShowBicycle : AppCompatActivity() {
    private lateinit var currentItem: Bicycle
    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_show_bicycle)
        val brand = intent.getStringExtra("brand")
        val image = intent.getStringExtra("image")
        val model = intent.getStringExtra("model")
        val description = intent.getStringExtra("description")
        val price = intent.getIntExtra("price", 0)
        val StringPrice = price.toString()
        currentItem = intent.getParcelableExtra("Bicycle")!!
        val imageUri: Uri? = if (!image.isNullOrEmpty()) Uri.parse(image) else null
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();

        val bikeBrand = findViewById<TextView>(R.id.BrandTextShowBikeAct).apply {
            text = brand
        }
        val bikeModel = findViewById<TextView>(R.id.ModelTextShowBikeAct).apply {
            text = model
        }
        val bikeDescription = findViewById<TextView>(R.id.descriptionTextShowBikeAct).apply {
            text = description
        }
        val bikePrice = findViewById<TextView>(R.id.pricTextShowBikeAct).apply {
            text = " $StringPrice$"

        }

        val bikeImage = findViewById<ImageView>(R.id.bikeImageShowBikeAct)

        imageUri?.let {
            Glide.with(this)
                .load(it)
                .into(bikeImage)
        }

        }

    fun homeIcon(view: View) {
            val intent = Intent(this@ShowBicycle, MainActivity::class.java)
            startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@ShowBicycle, Profile::class.java)
```

```
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@ShowBicycle, Cart::class.java)
        startActivity(intent)
    }
    private fun getCartItems(): Set<Bicycle> {
        val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
        val cartItemsSet = sharedPreferences.getStringSet("cart_items", emptySet()) ?: emptySet()
        return cartItemsSet.map { Gson().fromJson(it, Bicycle::class.java) }.toSet()
    }
    private fun saveCartItems(cartItems: Set<Bicycle>) {
        val sharedPreferences = getSharedPreferences("Cart", Context.MODE_PRIVATE)
        val editor = sharedPreferences.edit()
        editor.putStringSet("cart_items", cartItems.map { Gson().toJson(it) }.toSet())
        editor.apply()
    }

    private fun AddToCart(item: Bicycle) {
        val cartItems = getCartItems().toMutableSet()
        val existingItem = cartItems.find { it.id == item.id }

        if (existingItem != null) {
            existingItem.quantity++
        } else {
            cartItems.add(item.copy(addedToCart = true, quantity = 1))
        }
        saveCartItems(cartItems)
        Toast.makeText(this@ShowBicycle, "Item has been added to cart",
Toast.LENGTH_SHORT).show()
    }

    fun addToCart(view: View) {
        AddToCart(currentItem)
    }
}
```

<----------> LoginOrSignUp.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Intent
```

```
import android.graphics.PorterDuff
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.ImageView
import androidx.appcompat.app.ActionBar

class LoginOrSignUp : AppCompatActivity() {
    @SuppressLint("MissingInflatedId")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login_or_sign_up)
        val LoginBtn = findViewById<Button>(R.id.LoginBtnLoginOrSignUpAct)
        val SignUpBtn = findViewById<Button>(R.id.SignUpBtnLoginOrSignUpAct)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();

        LoginBtn.setOnClickListener {
            val intent = Intent(this, Login::class.java)
            startActivity(intent)
        }
        SignUpBtn.setOnClickListener {
            val intent = Intent(this, SignUp::class.java)
            startActivity(intent)
        }
    }


    fun homeIcon(view: View) {
        val intent = Intent(this@LoginOrSignUp, MainActivity::class.java)
        startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@LoginOrSignUp, Profile::class.java)
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@LoginOrSignUp, Cart::class.java)
        startActivity(intent)
    }
}
```

<----------> adminLogin.kt <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.app.Activity
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.ProgressBar
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.google.firebase.storage.FirebaseStorage

class adminLogin : AppCompatActivity() {
    private val database = FirebaseDatabase.getInstance()
    private val myRef = database.getReference("bicycles")
    private val IMAGE_PICK_CODE = 1000 // Arbitrary request code
    private lateinit var progressBar: ProgressBar
    @SuppressLint("MissingInflatedId", "SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_admin_login)
        progressBar = findViewById(R.id.loadingAdminLoginAct)
        val InsertBtn = findViewById<Button>(R.id.insertBtn)
        val Price = findViewById<EditText>(R.id.Price)
        val Model = findViewById<EditText>(R.id.Model)
        val Brand = findViewById<EditText>(R.id.Brand)
        val descriptionField = findViewById<EditText>(R.id.description)
        val ItemImage = findViewById<ImageView>(R.id.ItemImage)
        val curDataBtn = findViewById<Button>(R.id.ViewCurrentDataAdminLoginAct)
        val validationError = findViewById<TextView>(R.id.ValidationTextAdminLoginAct)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();

        curDataBtn.setOnClickListener {
            val intent = Intent(this, DisplayData::class.java)
            startActivity(intent)
        }
```

```
    ItemImage.setOnClickListener {
        // Create an Intent to pick an image from the gallery
        val intent = Intent(Intent.ACTION_PICK)
        intent.type = "image/*"
        startActivityForResult(intent, IMAGE_PICK_CODE)
    }

    InsertBtn.setOnClickListener {
        val pricestr = Price.text.toString()
        val priceInt: Int = pricestr.toIntOrNull() ?: 0
        val description = descriptionField.text.toString()
        val brand = Brand.text.toString()
        val model = Model.text.toString()

        // Validation: Check if the image is selected
        val imageUri = ItemImage.tag as? Uri
        if (imageUri != null) {
            if (isValidPrice(pricestr)) {
                uploadImageToFirebaseStorage(imageUri, brand, model, priceInt, description)
            } else
                validationError.text = "Please enter a valid price"
        } else
            validationError.text = "Please select a photo for the bicycle"
    }
}

private fun uploadImageToFirebaseStorage(
    imageUri: Uri,
    brand: String,
    model: String,
    price: Int,
    description: String
) {
    val storage = FirebaseStorage.getInstance()
    val storageRef = storage.reference
    val imagesRef = storageRef.child("images")

    // Generate a unique filename for the image
    val imageFileName = "image_${System.currentTimeMillis()}.jpg"
    val imageRef = imagesRef.child(imageFileName)
    progressBar.visibility = View.VISIBLE

    // Upload the image file to Firebase Storage
    imageRef.putFile(imageUri)
        .addOnSuccessListener { taskSnapshot ->
            // Image uploaded successfully, get the download URL
            imageRef.downloadUrl.addOnSuccessListener { uri ->
```

```
                progressBar.visibility = View.GONE
                val imageURL = uri.toString()

                // Call the function to write data to the database, including the image URL
                writeDataToDatabase(brand, model, price, description, imageURL)
            }
        }
        .addOnFailureListener { e ->
            progressBar.visibility = View.GONE
            // Handle failed image upload
            Log.e("adminLogin", "Image upload failed: $e")
        }
    }

    private fun writeDataToDatabase(
        brand: String,
        model: String,
        price: Int,
        description: String,
        imageURL: String
    ) {
        val bicycleId = myRef.push().key

        val bicycle = HashMap<String, Any>()
        bicycle["brand"] = brand
        bicycle["model"] = model
        bicycle["price"] = price
        bicycle["description"] = description
        bicycle["imageURL"] = imageURL

        bicycleId?.let {
            myRef.child(it).setValue(bicycle)
                .addOnSuccessListener {
                    Toast.makeText(this@adminLogin, "Item has been inserted to the database",
Toast.LENGTH_SHORT).show()
                    Log.d("adminLogin", "Bicycle data saved to database")
                }
                .addOnFailureListener {
                    Log.e("adminLogin", "Error writing bicycle data to database", it)
                }
        }
    }


    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        val ItemImage = findViewById<ImageView>(R.id.ItemImage)
```

```
        if (requestCode == IMAGE_PICK_CODE && resultCode == Activity.RESULT_OK && data != null) {
            val imageUri: Uri? = data.data
            findViewById<ImageView>(R.id.ItemImage).setImageURI(imageUri)
            ItemImage.tag = imageUri // Store the image URI in the tag for future use
        }
    }
    fun isValidPrice(priceInput: String): Boolean {
        // Check if the input is not empty
        if (priceInput.isEmpty()) {
            return false
        }

        // Try parsing the input as a double
        try {
            val price = priceInput.toInt()

            // Check if the price is a positive number
            if (price >= 0) {
                return true
            }
        } catch (e: NumberFormatException) {
            // Handle the case where the input is not a valid number
            return false
        }

        return false
    }

    fun homeIcon(view: View) {
        val intent = Intent(this@adminLogin, MainActivity::class.java)
        startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@adminLogin, Profile::class.java)
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@adminLogin, Cart::class.java)
        startActivity(intent)
    }
}
```

<----------> EditBicycle.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.app.Activity
```

```
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.View
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.ProgressBar
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import androidx.appcompat.app.AppCompatActivity
import com.bumptech.glide.Glide
import com.google.firebase.database.*
import com.google.firebase.storage.FirebaseStorage
import com.google.firebase.storage.StorageReference
import com.google.firebase.storage.UploadTask
import java.util.*

class EditBicycle : AppCompatActivity() {
    private val database = FirebaseDatabase.getInstance()
    private val myRef = database.getReference("bicycles")
    private lateinit var progressBar: ProgressBar
    private val IMAGE_PICK_CODE = 1000
    private lateinit var usersRef: DatabaseReference
    private lateinit var storageRef: StorageReference
    private lateinit var bicycleId: String
    private lateinit var oldImageURL: String


    @SuppressLint("SetTextI18n")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_edit_bicycle)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide()
        bicycleId = intent.getStringExtra("bicycleId").toString()
        usersRef = myRef.child(bicycleId)
        storageRef = FirebaseStorage.getInstance().reference.child("images")

        progressBar = findViewById(R.id.loadingEditBikeAct)
        val UpdateBtn = findViewById<Button>(R.id.UpdateBtnEditBikeAct)
        val priceEdit = findViewById<EditText>(R.id.PriceEditBikeAct)
        val modelEdit = findViewById<EditText>(R.id.ModelEditBikeAct)
        val brandEdit = findViewById<EditText>(R.id.BrandEditBikeAct)
        val descriptionFieldEdit = findViewById<EditText>(R.id.descriptionEditBikeAct)
        val itemImageEdit = findViewById<ImageView>(R.id.ItemImageEditBikeAct)
        val curDataBtn = findViewById<Button>(R.id.ViewCurrentDataEditBikeAct)
```

```
curDataBtn.setOnClickListener {
   val intent = Intent(this, DisplayData::class.java)
   startActivity(intent)
}

itemImageEdit.setOnClickListener {
   // Create an Intent to pick an image from the gallery
   val intent = Intent(Intent.ACTION_PICK)
   intent.type = "image/*"
   startActivityForResult(intent, IMAGE_PICK_CODE)
}

usersRef.addListenerForSingleValueEvent(object : ValueEventListener {
   @SuppressLint("SetTextI18n")
   override fun onDataChange(snapshot: DataSnapshot) {
      oldImageURL = snapshot.child("imageURL").getValue(String::class.java).toString()
      val bike = snapshot.getValue(Bicycle::class.java)
      val image = intent.getStringExtra("bicycleImage")
      val imageUri = Uri.parse(image)
      priceEdit.setText(bike?.price.toString())
      modelEdit.setText(bike?.model)
      brandEdit.setText(bike?.brand)
      descriptionFieldEdit.setText(bike?.description)
      Glide.with(this@EditBicycle)
         .load(imageUri)
         .into(itemImageEdit)
   }

   override fun onCancelled(error: DatabaseError) {
      // Handle error
   }
})

UpdateBtn.setOnClickListener {
   val validationError = findViewById<TextView>(R.id.ValidationTextEditBikeAct)
   val price = priceEdit.text.toString()

   // Validate the price input
   if (isValidPrice(price)) {
      val priceInt: Int = price.toInt()

      val model = modelEdit.text.toString()
      val brand = brandEdit.text.toString()
      val descriptionField = descriptionFieldEdit.text.toString()

      // Update data in the database
      val bikeData = mapOf(
```

```
                "brand" to brand,
                "model" to model,
                "price" to priceInt,
                "description" to descriptionField
            )

            usersRef.updateChildren(bikeData)
                .addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        // Check if a new image is selected, and update it if necessary
                        val imageUri: Uri? = itemImageEdit.tag as? Uri
                        if (imageUri != null) {
                            uploadImage(imageUri)
                            deleteImageFromStorage(oldImageURL)
                        } else {
                            // No new image selected, show a toast or take appropriate action
                            Toast.makeText(this@EditBicycle, "Data has been updated",
Toast.LENGTH_SHORT).show()
                        }
                    } else {
                        Toast.makeText(this@EditBicycle, "Failed to update data",
Toast.LENGTH_SHORT).show()
                    }
                }
        } else {
            validationError.text = "Please enter a valid price"
        }
    }

  }

  override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (resultCode == Activity.RESULT_OK && requestCode == IMAGE_PICK_CODE) {
        // Handle the result of image selection
        val imageUri: Uri? = data?.data
        findViewById<ImageView>(R.id.ItemImageEditBikeAct).setImageURI(imageUri)
        findViewById<ImageView>(R.id.ItemImageEditBikeAct).tag = imageUri // Store the image URI
in the tag for future use
    }
  }

  private fun uploadImage(imageUri: Uri?) {
    if (imageUri != null) {
        val imageFileName = "image_${System.currentTimeMillis()}.jpg"
        val imageRef = storageRef.child(imageFileName)
        progressBar.visibility = ProgressBar.VISIBLE
```

```
        // Upload the new image file to Firebase Storage
        imageRef.putFile(imageUri)
            .addOnSuccessListener { taskSnapshot: UploadTask.TaskSnapshot ->
                // Image uploaded successfully, get the download URL
                imageRef.downloadUrl.addOnSuccessListener { uri ->
                    progressBar.visibility = ProgressBar.GONE
                    val newImageURL = uri.toString()

                    // Update the image URL in the database
                    usersRef.child("imageURL").setValue(newImageURL)
                        .addOnCompleteListener { task ->
                            if (task.isSuccessful) {
                                Toast.makeText(
                                    this@EditBicycle,
                                    "Data has been updated",
                                    Toast.LENGTH_SHORT
                                ).show()
                            } else {
                                Toast.makeText(
                                    this@EditBicycle,
                                    "Failed to update image",
                                    Toast.LENGTH_SHORT
                                ).show()
                            }
                        }
                }
            }
            .addOnFailureListener { e ->
                progressBar.visibility = ProgressBar.GONE
                // Handle failed image upload
                Toast.makeText(this@EditBicycle, "Image upload failed: $e",
Toast.LENGTH_SHORT).show()
            }
    }
  }
  private fun deleteImageFromStorage(imageURL: String) {
    // Get a reference to the old image file in Firebase Storage
    val oldImageRef = FirebaseStorage.getInstance().getReferenceFromUrl(imageURL)

    // Delete the old image file
    oldImageRef.delete()
        .addOnSuccessListener {
            // File deleted successfully
            Toast.makeText(this@EditBicycle, "Old image has been deleted",
Toast.LENGTH_SHORT).show()
        }
        .addOnFailureListener { e ->
            // Handle the failure to delete the old image
```

```
            Toast.makeText(this@EditBicycle, "Failed to delete old image: $e",
Toast.LENGTH_SHORT).show()
        }
    }
    fun isValidPrice(priceInput: String): Boolean {
        // Check if the input is not empty
        if (priceInput.isEmpty()) {
            return false
        }

        // Try parsing the input as a double
        try {
            val price = priceInput.toInt()

            // Check if the price is a positive number
            if (price >= 0) {
                return true
            }
        } catch (e: NumberFormatException) {
            // Handle the case where the input is not a valid number
            return false
        }

        return false
    }
    fun homeIcon(view: View) {
        val intent = Intent(this@EditBicycle, MainActivity::class.java)
        startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@EditBicycle, Profile::class.java)
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@EditBicycle, Cart::class.java)
        startActivity(intent)
    }
}
```

<----------> DisplayData.kt: <---------->

```
package com.example.mwadfinalproject

import android.annotation.SuppressLint
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
```

```
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.ActionBar
import androidx.recyclerview.widget.GridLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import com.google.firebase.storage.FirebaseStorage

class DisplayData : AppCompatActivity() {
    private val database = FirebaseDatabase.getInstance()
    private val myRef = database.getReference("bicycles")
    private lateinit var auth: FirebaseAuth
    private lateinit var authListener: FirebaseAuth.AuthStateListener

    fun homeIcon(view: View) {
        val intent = Intent(this@DisplayData, MainActivity::class.java)
        startActivity(intent)
    }
    fun profileIcon(view: View) {
        val intent = Intent(this@DisplayData, Profile::class.java)
        startActivity(intent)
    }
    fun cartIcon(view: View){
        val intent = Intent(this@DisplayData, Cart::class.java)
        startActivity(intent)
    }
    private fun readDataFromDatabase(brandName: String) {
        myRef.addListenerForSingleValueEvent(object : ValueEventListener {
            override fun onDataChange(dataSnapshot: DataSnapshot) {
                val bicycles = mutableListOf<Bicycle>()

                for (snapshot in dataSnapshot.children) {
                    val brand = snapshot.child("brand").getValue(String::class.java)
```

```
            if (brandName.isEmpty() || brand.equals(brandName, ignoreCase = true)) {
                val model = snapshot.child("model").getValue(String::class.java)
                val price = snapshot.child("price").getValue(Int::class.java)
                val description = snapshot.child("description").getValue(String::class.java)
                val imageURL = snapshot.child("imageURL").getValue(String::class.java)
                val bicycleId = snapshot.key

                if (model != null && price != null && description != null && imageURL != null &&
bicycleId != null) {
                    bicycles.add(Bicycle(bicycleId, brand.toString(), model, description, price, imageURL,
false, 1))
                }
            }
        }

        val recyclerView = findViewById<RecyclerView>(R.id.recyclerViewDisplayDataAct)
        recyclerView.layoutManager = GridLayoutManager(this@DisplayData, 2)
        val adapter = BicycleAdapter(bicycles)
        recyclerView.adapter = adapter
      }

      override fun onCancelled(databaseError: DatabaseError) {
        Log.e("adminLogin", "Error reading data from database", databaseError.toException())
      }
    })
  }
  private fun deleteImageFromStorage(imageURL: String) {
    // Get a reference to the old image file in Firebase Storage
    val oldImageRef = FirebaseStorage.getInstance().getReferenceFromUrl(imageURL)

    // Delete the old image file
    oldImageRef.delete()
      .addOnSuccessListener {
        // File deleted successfully
        Toast.makeText(this@DisplayData, "Bicycle data has been deleted from database",
Toast.LENGTH_SHORT).show()
      }
      .addOnFailureListener { e ->
        // Handle the failure to delete the old image
        Toast.makeText(this@DisplayData, "Failed to delete old image: $e",
Toast.LENGTH_SHORT).show()
      }
  }

  inner class BicycleAdapter(private val bicycleList: List<Bicycle>) :
    RecyclerView.Adapter<BicycleAdapter.BicycleViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): BicycleViewHolder {
```

```
    val itemView = LayoutInflater.from(parent.context)
        .inflate(R.layout.item_holder_admin, parent, false)
    return BicycleViewHolder(itemView)
}
private fun deleteBicycle(bicycleId: String) {
    // Delete the bicycle from the database
    myRef.child(bicycleId).removeValue()
        .addOnSuccessListener {
            Log.d("adminLogin", "Bicycle data has been deleted from database")
        }
        .addOnFailureListener {
            Log.e("adminLogin", "Error deleting bicycle data from database", it)
        }
}

@SuppressLint("SetTextI18n")
override fun onBindViewHolder(holder: BicycleViewHolder, position: Int) {
    val currentItem = bicycleList[position]
    val imageUri = Uri.parse(currentItem.ImageURL)
    val bicycleId = currentItem.id


    holder.brandTextView.text = currentItem.brand
    holder.priceTextView.text = "${currentItem.price}$"

    Glide.with(holder.itemView.context)
        .load(imageUri)
        .into(holder.bikeImage)

    holder.EditButton.setOnClickListener {

        val intent = Intent(holder.itemView.context, EditBicycle::class.java)
        intent.putExtra("bicycleId", bicycleId)
        intent.putExtra("bicycleImage", currentItem.ImageURL)

        holder.itemView.context.startActivity(intent)
    }

    holder.DeleteItem.setOnClickListener {
        val imageURL = currentItem.ImageURL
        deleteBicycle(bicycleId)
        deleteImageFromStorage(imageURL)
        holder.container.visibility = View.GONE
    }
}

override fun getItemCount() = bicycleList.size
```

```
        inner class BicycleViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
            val brandTextView: TextView = itemView.findViewById(R.id.textBrand)
            val priceTextView: TextView = itemView.findViewById(R.id.textPrice)
            val bikeImage: ImageView = itemView.findViewById(R.id.imageBicycle)
            val EditButton: Button = itemView.findViewById(R.id.EditButton)
            val DeleteItem = itemView.findViewById<LinearLayout>(R.id.DeleteItem)
            val container = itemView.findViewById<LinearLayout>(R.id.holderContin)
        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_display_data)
        val actionBar: ActionBar? = supportActionBar
        actionBar?.hide();
        val insertbtn = findViewById<Button>(R.id.InsertNewItemsBtnDisplayDataAct)
        val brandEditText = findViewById<EditText>(R.id.brandEditTextDisplayDataAct)

        insertbtn.setOnClickListener {
            startActivity(Intent(this, adminLogin::class.java))
        }

        brandEditText.setOnClickListener {
            val brandName = brandEditText.text.toString()
            readDataFromDatabase(brandName)
        }
        readDataFromDatabase("")
    }
}
```