Food Delivery App

1. **Changing theme using provider**
2. **Create a file for dark theme:**

**Dark\_theme.dart**

import 'package:flutter/material.dart';

ThemeData darkTheme = ThemeData(

  colorScheme: ColorScheme.dark(

      surface: const Color.fromARGB(255, 20, 20, 20), //background

      primary: const Color.fromARGB(255, 122, 122, 122),

      secondary: const Color.fromARGB(255, 30, 30, 30),

      tertiary: const Color.fromARGB(255, 47, 47, 47),

      inversePrimary: Colors.grey.shade300),

);

1. **Create a file for light theme:**

**Light\_theme.dart**

import 'package:flutter/material.dart';

ThemeData lightTheme = ThemeData(

  colorScheme: ColorScheme.light(

      surface: Colors.grey.shade300, //background

      primary: Colors.grey.shade500,

      secondary: Colors.grey.shade100,

      tertiary: Colors.white,

      inversePrimary: Colors.grey.shade700),

);

1. **Create a file for theme provider: do not forgot to add provider in pubspec.yaml**

import 'package:flutter/material.dart';

import 'package:food\_delivery/themes/dark\_theme.dart';

import 'package:food\_delivery/themes/light\_theme.dart';

class ThemeProvider extends ChangeNotifier {

  ThemeData \_themeData = lightTheme;

  //get theme

  ThemeData get themeData => \_themeData;

  //check dark theme

  bool get isDarkTheme => \_themeData == darkTheme;

  set themeData(ThemeData themeData) {

    \_themeData = themeData;

    notifyListeners();

  }

//change theme

  void toggleTheme() {

    if (\_themeData == lightTheme) {

      themeData = darkTheme;

    } else {

      themeData = lightTheme;

    }

  }

}

1. **Now register provider in main.dart also use theme:providers theme**

**Main.dart**

import 'package:flutter/material.dart';

import 'package:food\_delivery/screens/login\_screen.dart';

import 'package:food\_delivery/themes/theme\_provider.dart';

import 'package:provider/provider.dart';

void main() {

  runApp(

    ChangeNotifierProvider(

      create: (context) => ThemeProvider(),

      child: const MyApp(),

    ),

  );

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      title: 'Food Delivery',

      home: const LoginScreen(),

      //accessing theme provider

      theme: Provider.of<ThemeProvider>(context).themeData,

    );

  }

}

1. **Changing using toggle button**

**Setting\_screen.dart**

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:food\_delivery/themes/theme\_provider.dart';

import 'package:provider/provider.dart';

class SettingScreen extends StatefulWidget {

  const SettingScreen({super.key});

  @override

  State<SettingScreen> createState() => \_SettingScreenState();

}

class \_SettingScreenState extends State<SettingScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: const Text("setting"),

      ),

      body: Column(

        children: [

          Container(

            margin: const EdgeInsets.symmetric(horizontal: 20, vertical: 5),

            padding: const EdgeInsets.all(20),

            decoration: BoxDecoration(

              color: Theme.of(context).colorScheme.secondary,

              borderRadius: BorderRadius.circular(10),

            ),

            child: Row(

              mainAxisAlignment: MainAxisAlignment.spaceBetween,

              children: [

                Text(

                  "Dark Theme",

                  style: TextStyle(

                    color: Theme.of(context).colorScheme.inversePrimary,

                  ),

                ),

                CupertinoSwitch(

                  value: Provider.of<ThemeProvider>(context, listen: false)

                      .isDarkTheme,

                  onChanged: (value) {

                    Provider.of<ThemeProvider>(context, listen: false)

                        .toggleTheme();

                  },

                ),

              ],

            ),

          ),

        ],

      ),

    );

  }

}

1. **Credit card adding**

class PaymentScreen extends StatefulWidget {

  const PaymentScreen({super.key});

  @override

  State<PaymentScreen> createState() => \_PaymentScreenState();

}

class \_PaymentScreenState extends State<PaymentScreen> {

  String cardNumber = '';

  String expiryDate = '';

  String cardHolderName = '';

  String cvvCode = '';

  GlobalKey<FormState> key = GlobalKey<FormState>();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: AppColors.surface(context),

      appBar: AppBar(

        backgroundColor: Colors.transparent,

        title: const Text("Payment"),

      ),

      body: SingleChildScrollView(

        child: Column(

          children: [

            CreditCardWidget(

              cardNumber: cardNumber,

              expiryDate: expiryDate,

              cardHolderName: cardHolderName,

              cvvCode: cvvCode,

              showBackView: false,

              isChipVisible: true,

              isHolderNameVisible: true,

              onCreditCardWidgetChange: (value) {},

              animationDuration: Durations.long1,

              isSwipeGestureEnabled: true,

            ),

            CreditCardForm(

              cardNumber: cardNumber,

              expiryDate: expiryDate,

              cardHolderName: cardHolderName,

              cvvCode: cvvCode,

              onCreditCardModelChange: (value) {

                setState(() {

                  cardNumber = value.cardNumber;

                  expiryDate = value.expiryDate;

                  cardHolderName = value.cardHolderName;

                  cvvCode = value.cvvCode;

                });

              },

              formKey: key,

            ),

            const AddVerticleSpace(height: 210),

            CustomButton(

              label: "Pay",

              onTap: () {

                if (key.currentState!.validate()) {

                  showDialog(

                    context: context,

                    builder: (context) {

                      return AlertDialog(

                        title: const Text("Confirm Payment"),

                        content: SingleChildScrollView(

                          child: Padding(

                            padding: const EdgeInsets.all(8.0),

                            child: Column(

                              crossAxisAlignment: CrossAxisAlignment.start,

                              children: [

                                Text("Card Number : $cardNumber"),

                                Text("Expiry Date : $expiryDate"),

                                Text("Card Holder Name : $cardHolderName"),

                                Text("CV Code : $cvvCode"),

                              ],

                            ),

                          ),

                        ),

                        actions: [

                          TextButton(

                            onPressed: () {

                              Navigator.pop(context);

                            },

                            child: const Text("Cancel"),

                          ),

                          TextButton(

                            onPressed: () {

                              Navigator.pop(context);

                              Navigator.push(

                                context,

                                MaterialPageRoute(

                                  builder: (context) {

                                    return const PaymentProcessing();

                                  },

                                ),

                              );

                            },

                            child: const Text("Pay"),

                          ),

                        ],

                      );

                    },

                  );

                }

              },

            ),

          ],

        ),

      ),

    );

  }

}

1. **Operations to cart**

//user cart

  final List<CartItem> \_cart = [];

//1. add to cart

  void addToCart(Food food, List<FoodAddon> selectedAddons) {

    //cart alredy with same food

    CartItem? cartItem = \_cart.firstWhereOrNull((items) {

      //checking if samee food

      bool isSameFood = items.food == food;

      //checking if same adddons

      bool isSameAddon =

          const ListEquality().equals(items.selectedAddon, selectedAddons);

      return isSameAddon && isSameFood;

    });

    //if item alredy exist=>just increase quantity

    if (cartItem != null) {

      cartItem.quantity++;

    }

    //else add new item to the cart

    else {

      \_cart.add(CartItem(food: food, selectedAddon: selectedAddons));

    }

    notifyListeners();

  }

//2. remove from

  void removeFromCart(CartItem cartItem) {

    int cartIndex = \_cart.indexOf(cartItem);

    //checking never negative

    if (cartIndex != -1) {

      if (\_cart[cartIndex].quantity > 1) {

        \_cart[cartIndex].quantity--;

      } else

      //if 1 remove from cart

      {

        \_cart.removeAt(cartIndex);

      }

    }

    notifyListeners();

  }

//3. get total price

  double totalPrice() {

    double total = 0.0;

    for (CartItem cartItem in \_cart) {

      //item total

      double itemTotal = cartItem.food.price;

      //checking addons

      for (FoodAddon foodAddon in cartItem.selectedAddon) {

        itemTotal += foodAddon.price;

      }

      total += itemTotal \* cartItem.quantity;

    }

    return total;

  }

//4. get total numbers of items in cart

  int getTotalItemCount() {

    int itemCount = 0;

    for (CartItem cartItem in cart) {

      itemCount += cartItem.quantity;

    }

    return itemCount;

  }

//5. clear cart

  void clearCart() {

    \_cart.clear();

    notifyListeners();

  }

  //helpers

//geneerate receipt

  String receipt() {

    final receipt = StringBuffer();

    receipt.writeln("Here is Your Receipt");

    receipt.writeln();

    //formating data

    String formatdate =

        DateFormat('dd/MM/yyyy HH:mm:ss').format(DateTime.now());

    receipt.writeln(formatdate);

    receipt.writeln();

    receipt.writeln("\_\_\_\_\_\_\_\_\_\_\_\_");

    receipt.writeln();

    for (int i = 0; i < cart.length; i++) {

      //adding detals

      receipt.writeln(

          "${i + 1}) ${cart[i].quantity} X ${cart[i].food.name} - ${formatMoney(cart[i].food.price)}");

      //adding addons

      if (cart[i].selectedAddon.isNotEmpty) {

        receipt.writeln("Add-ons: ${formatingAddons(cart[i].selectedAddon)}");

        receipt.writeln();

      }

    }

    receipt.writeln();

    receipt.writeln("\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

    receipt.writeln("");

    //items price

    receipt.writeln("Total Items : ${getTotalItemCount()}");

    receipt.writeln("Total Price : ${formatMoney(totalPrice())}");

    return receipt.toString();

  }

//format doubke value into money

  String formatMoney(double price) {

    return "${price.toStringAsFixed(2)} Rs";

  }

//format list of add on into string summery

  String formatingAddons(List<FoodAddon> addon) {

    return addon

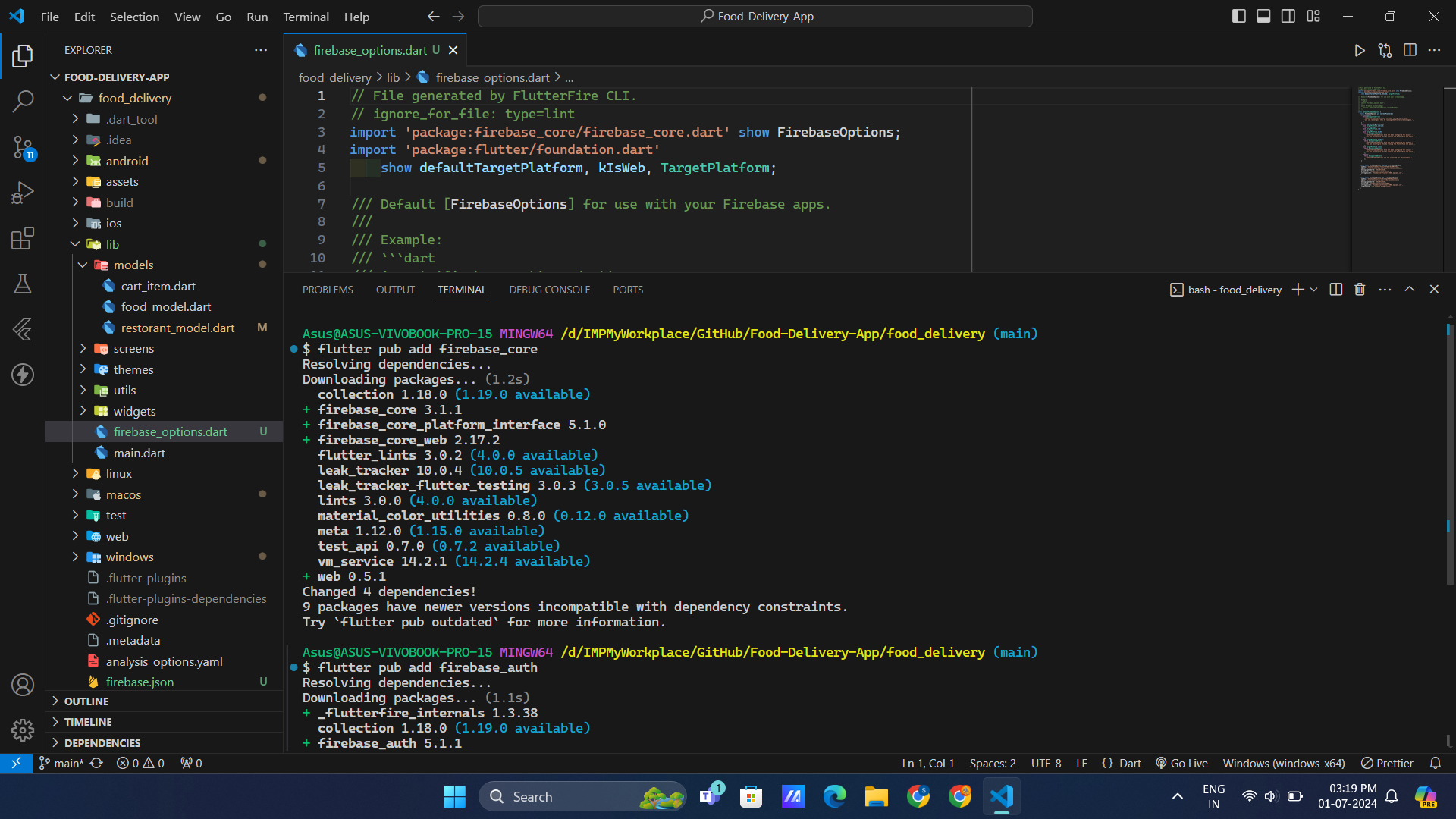
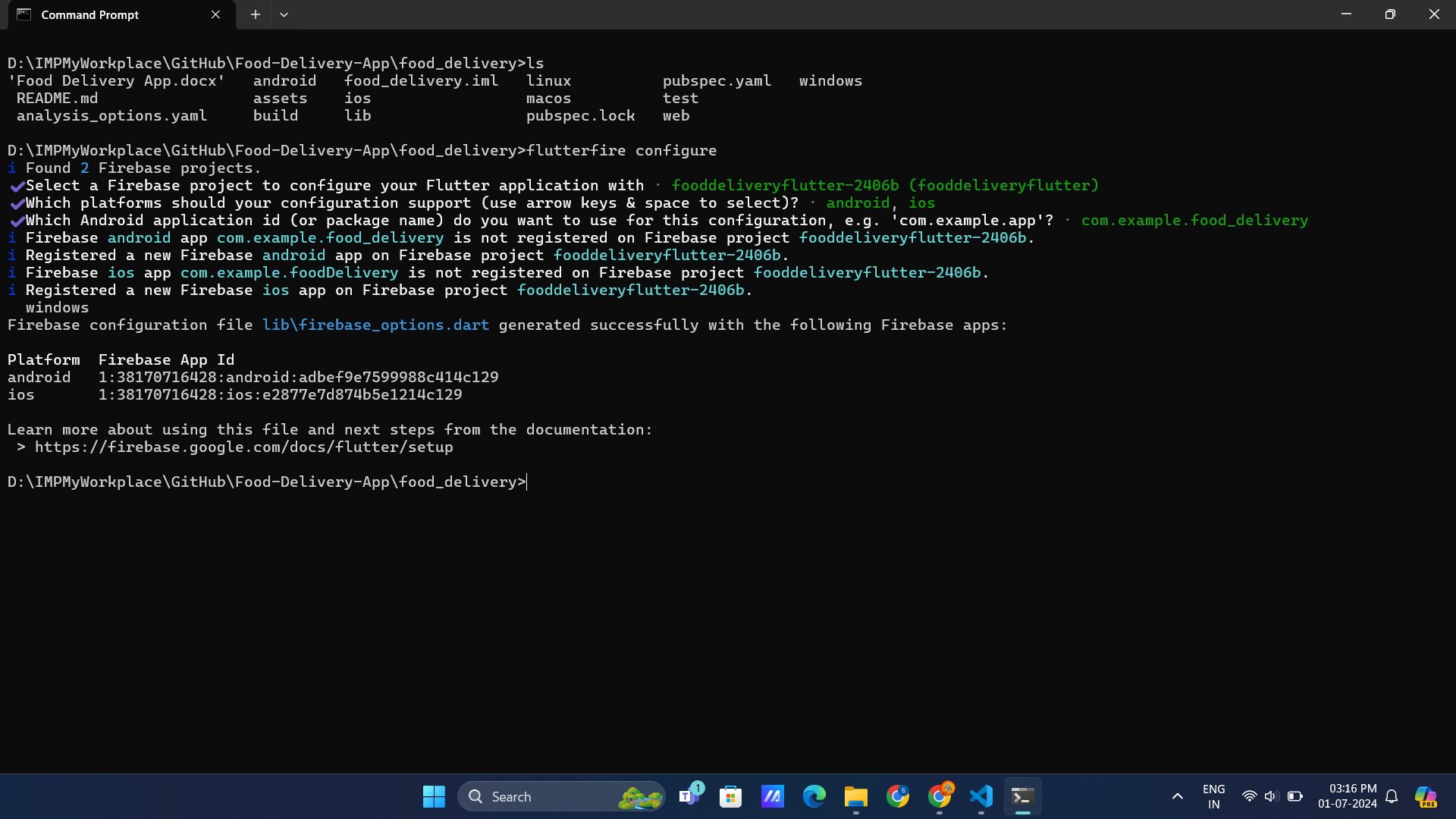
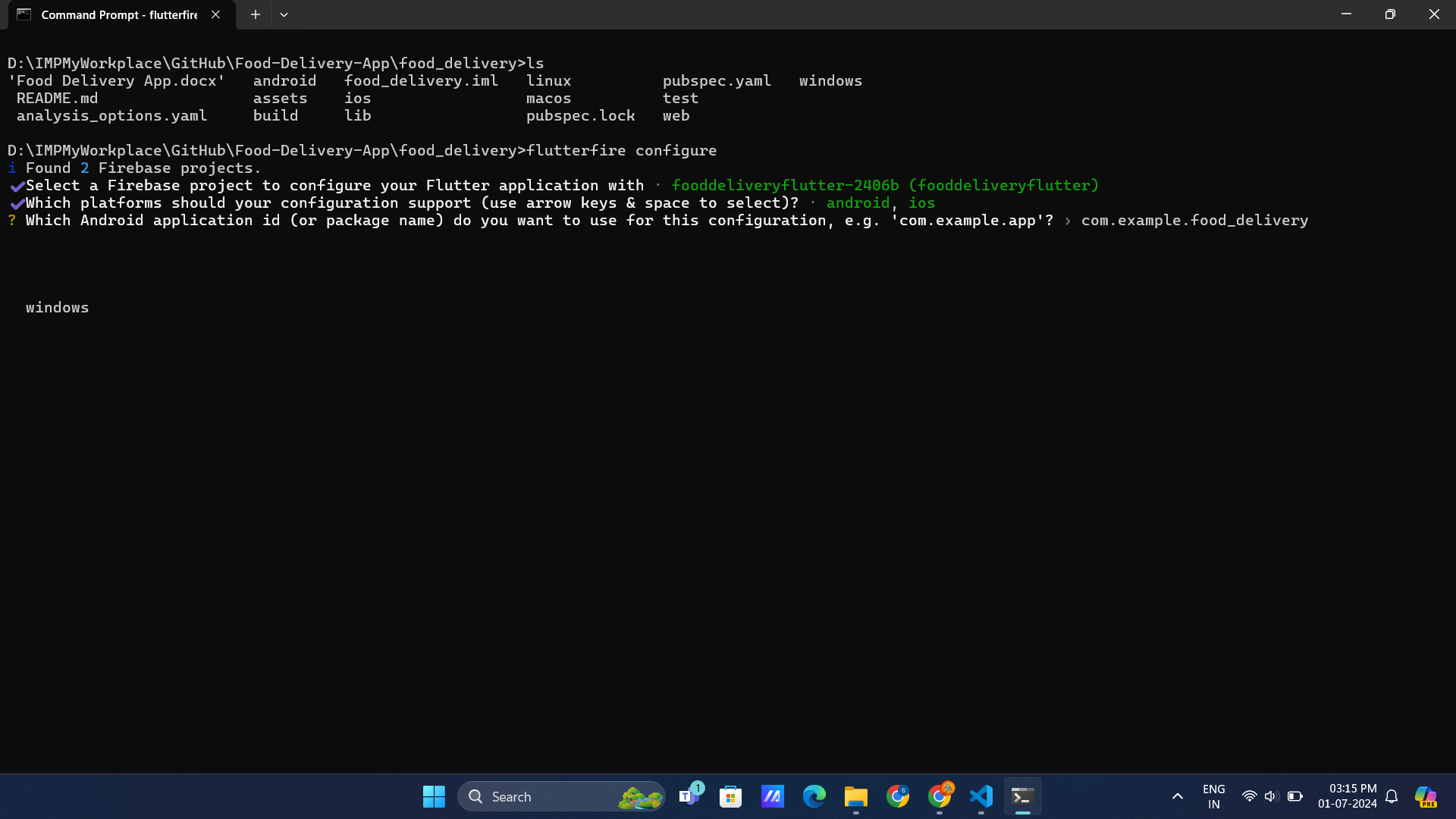
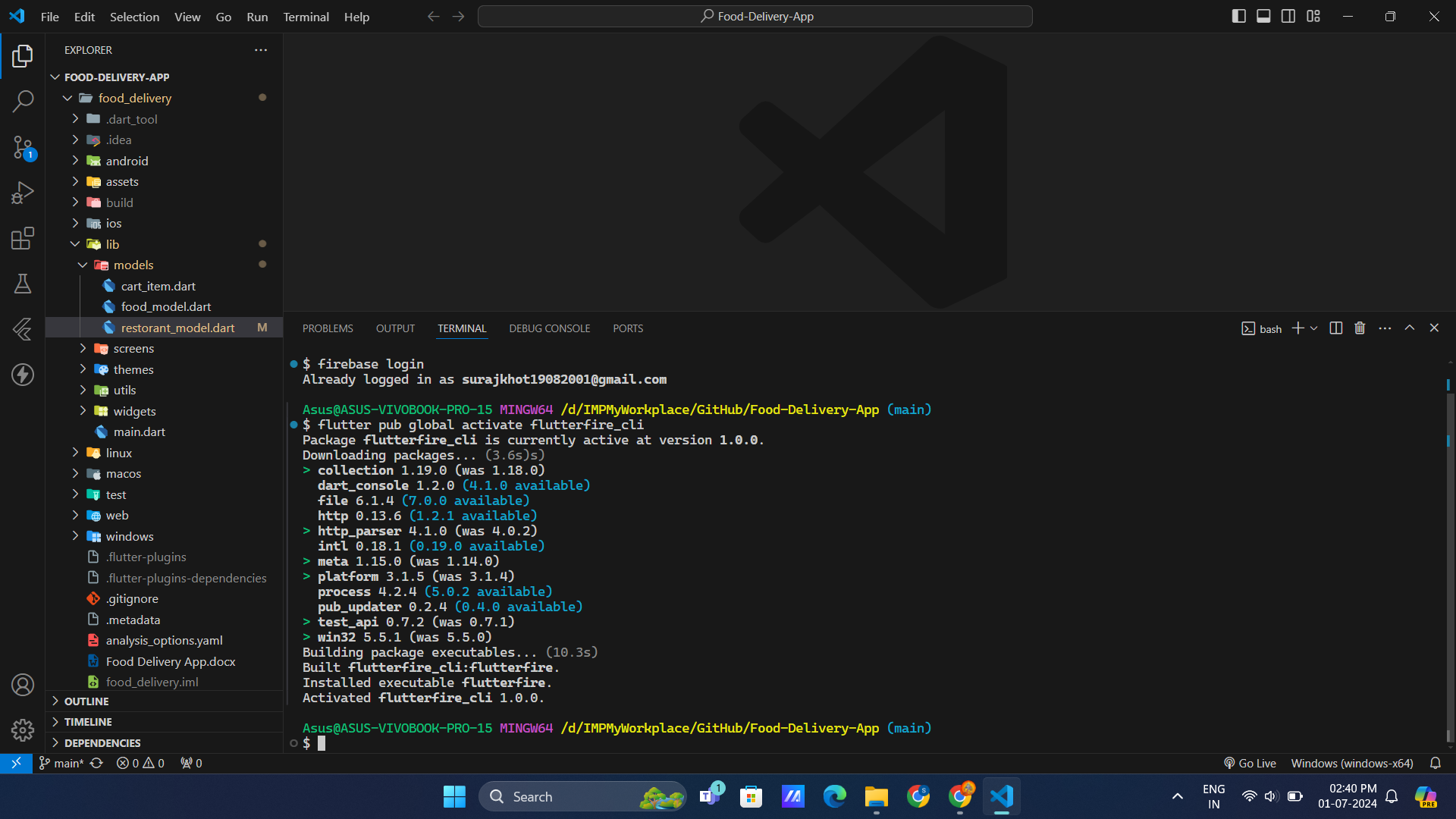
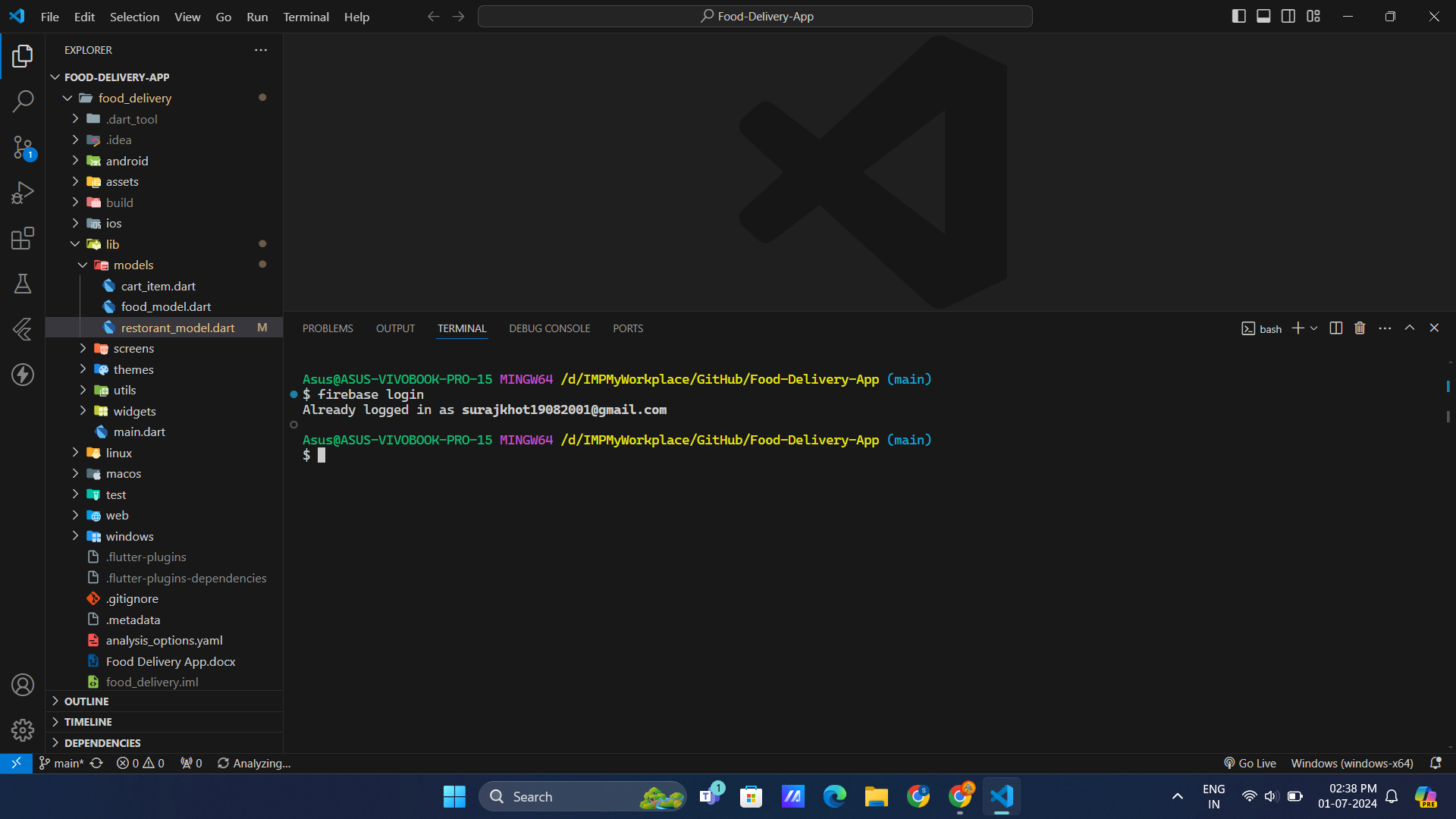
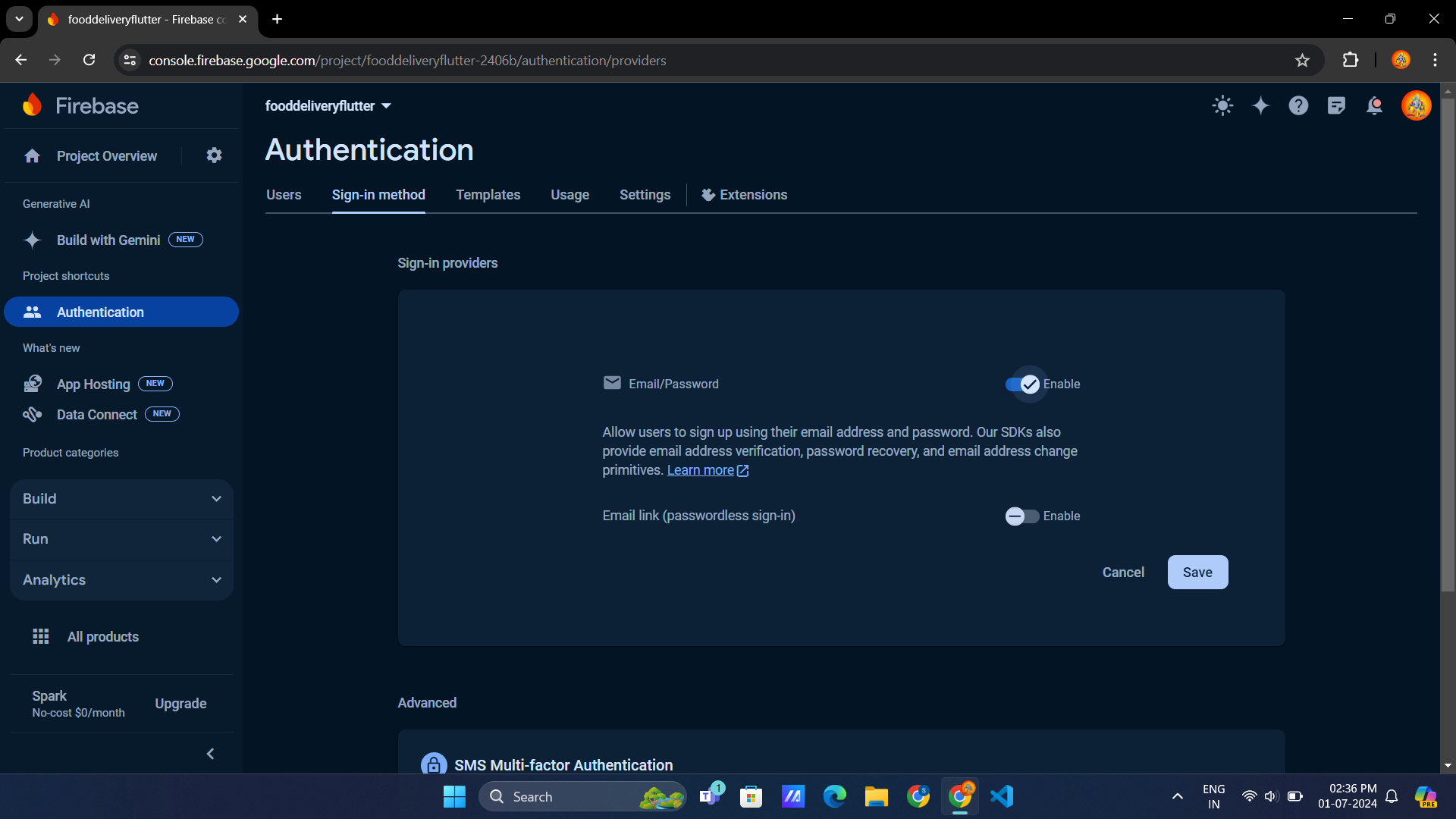
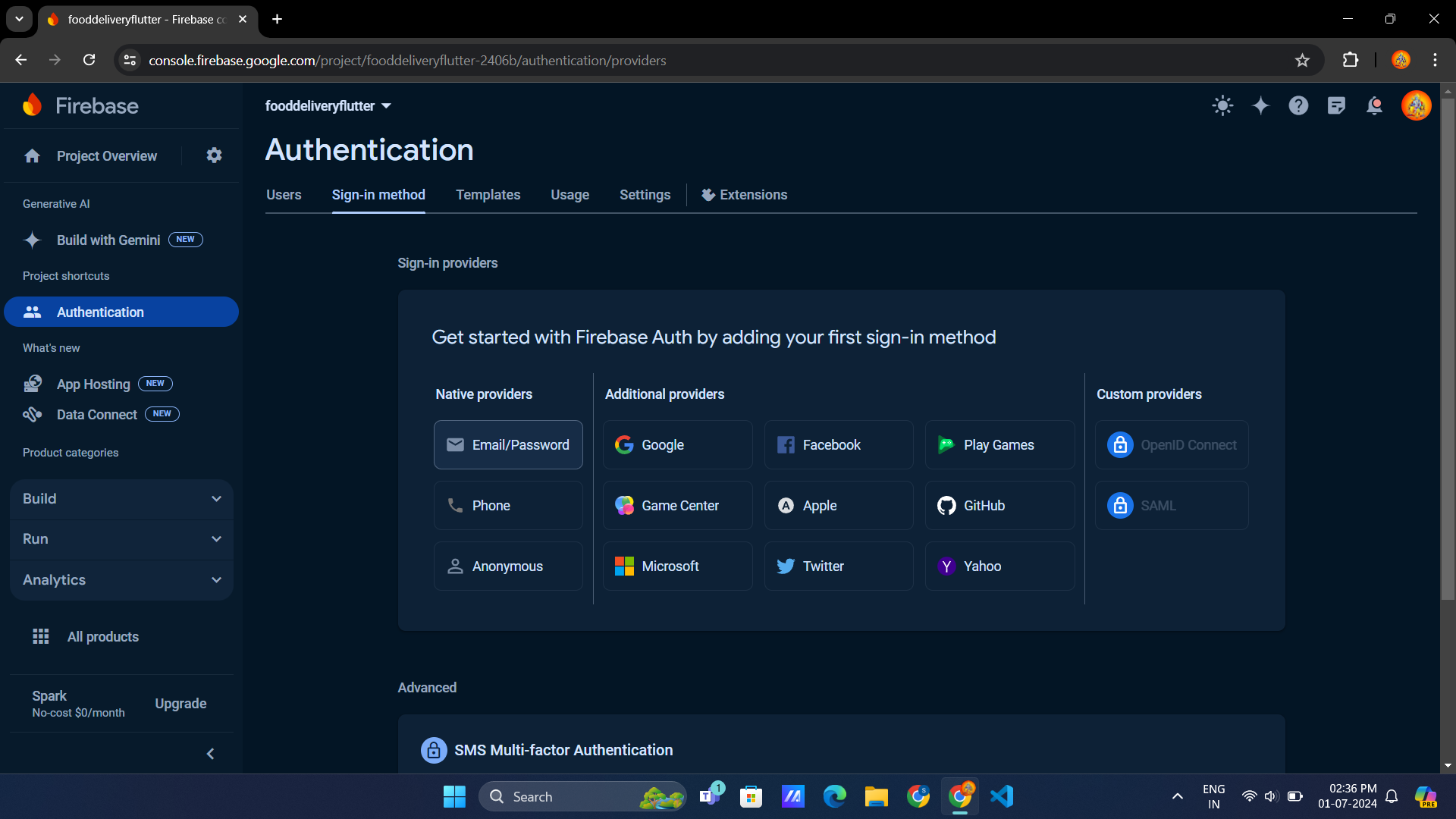
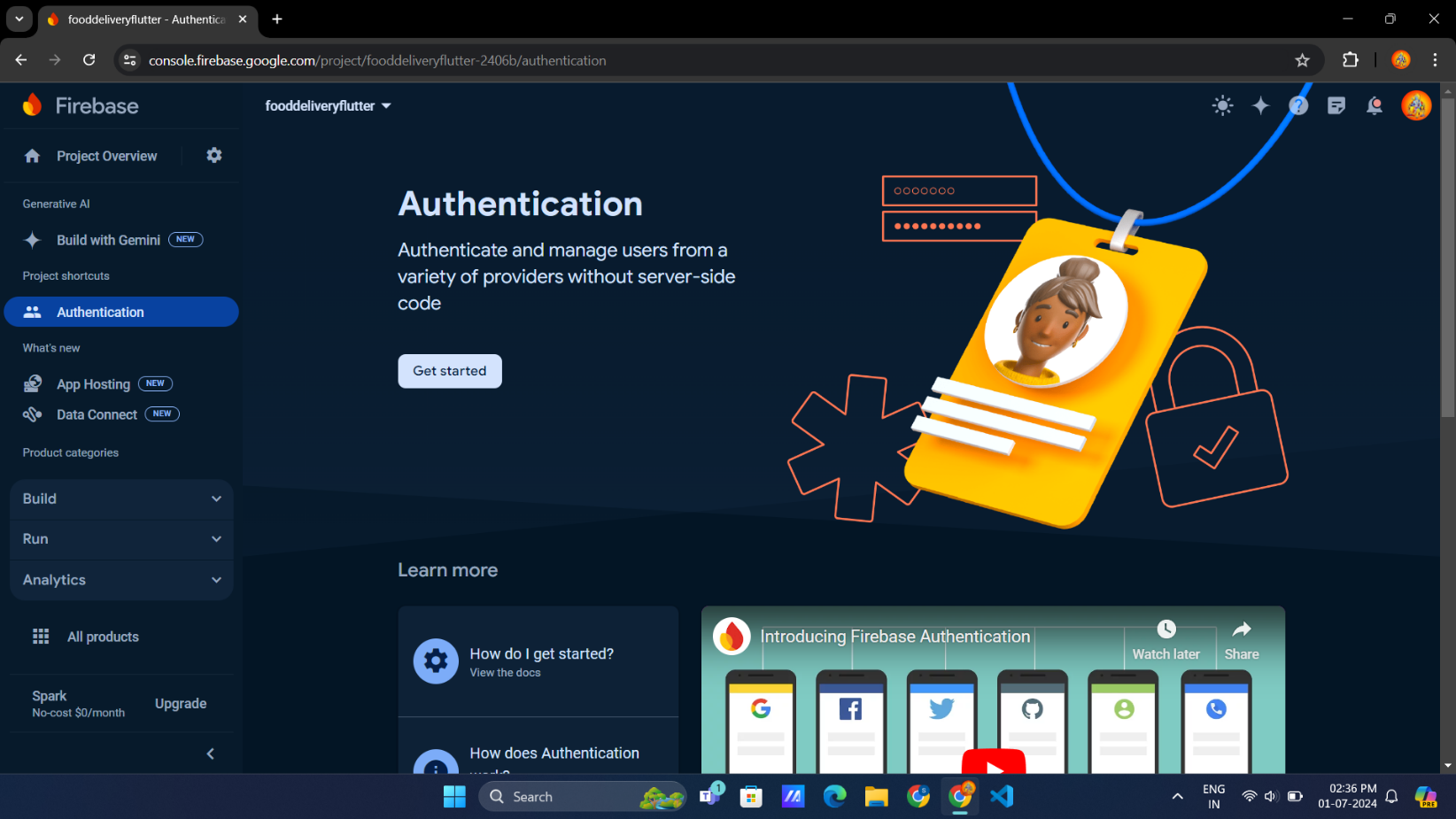
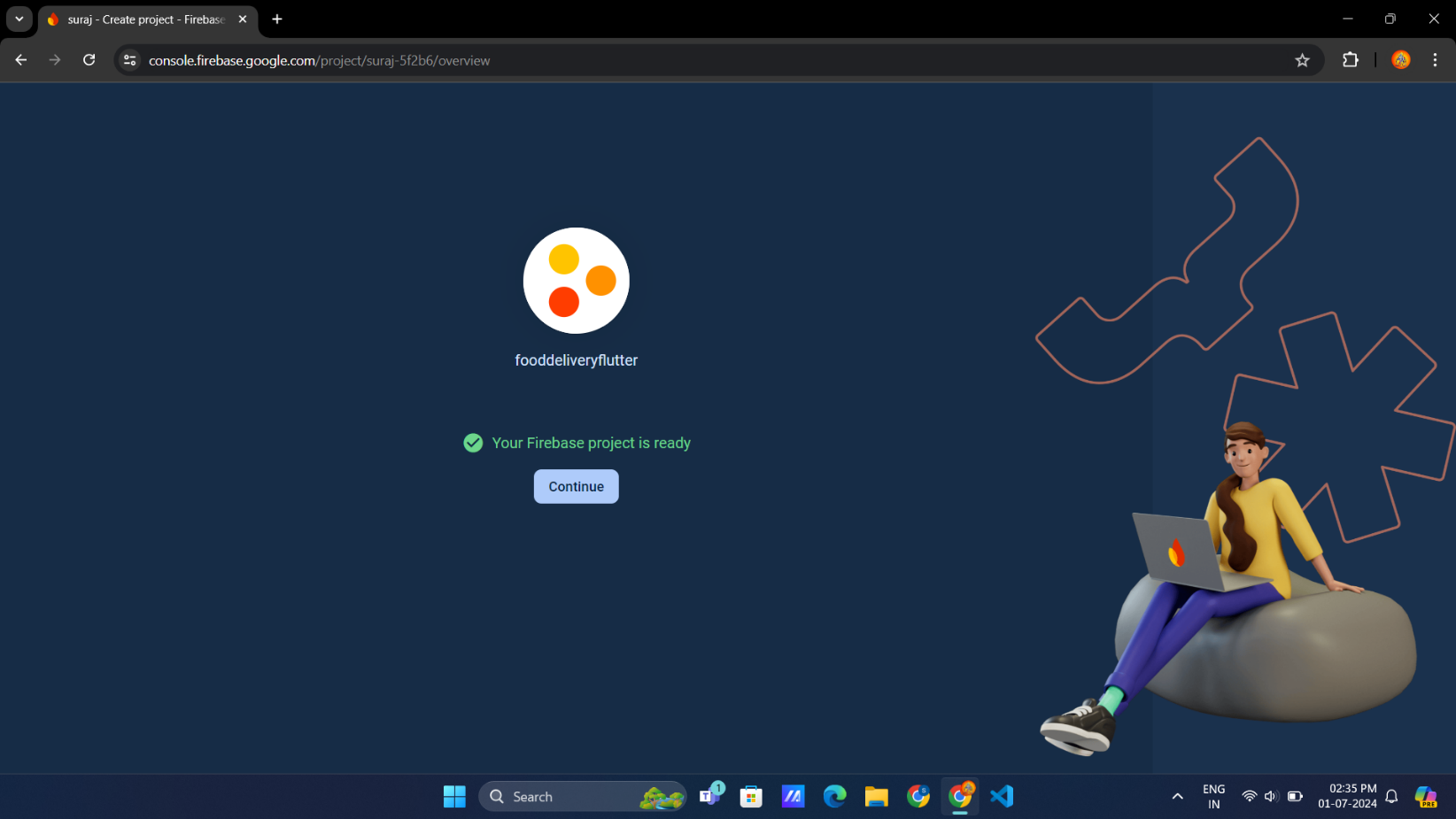
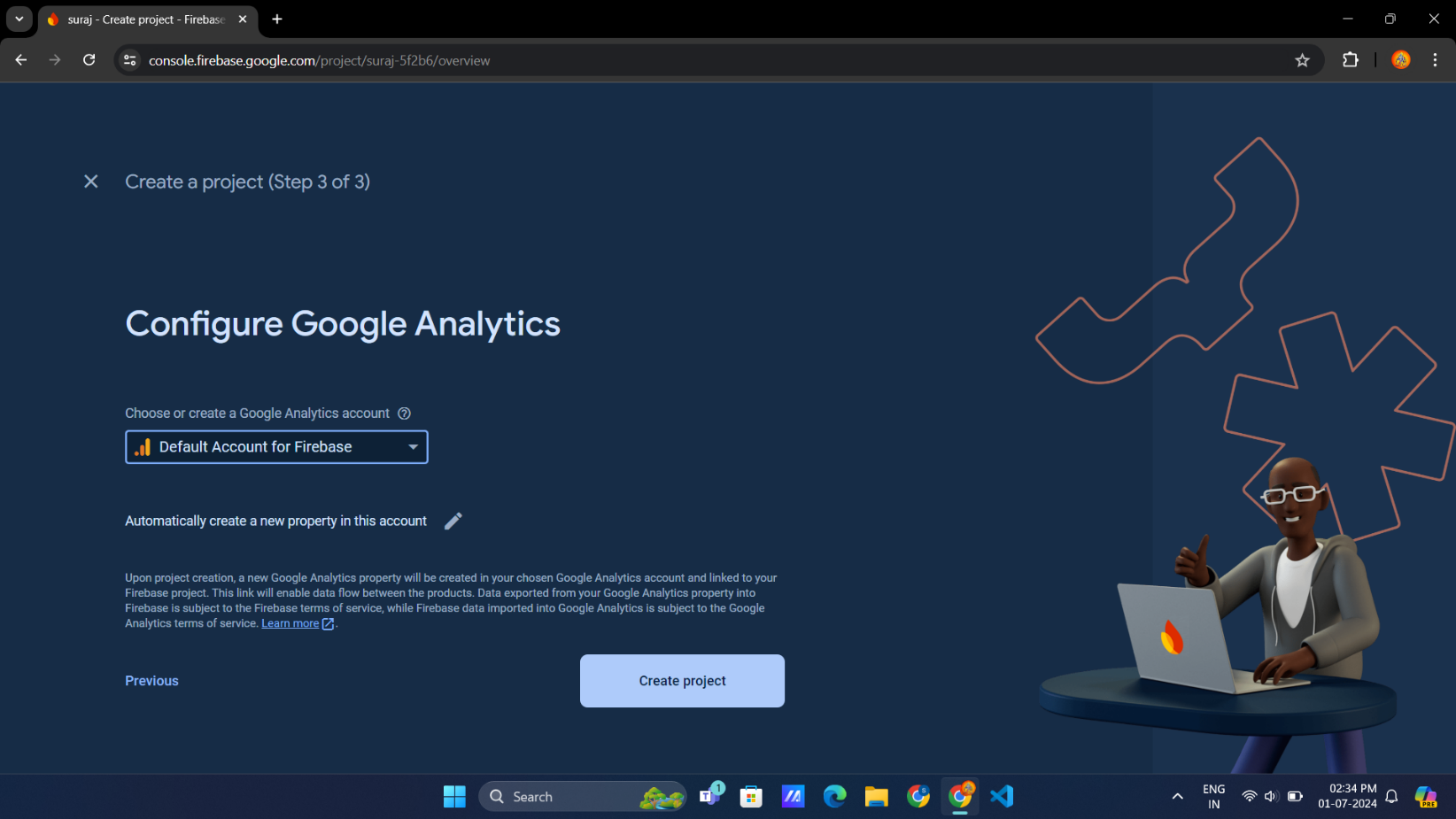
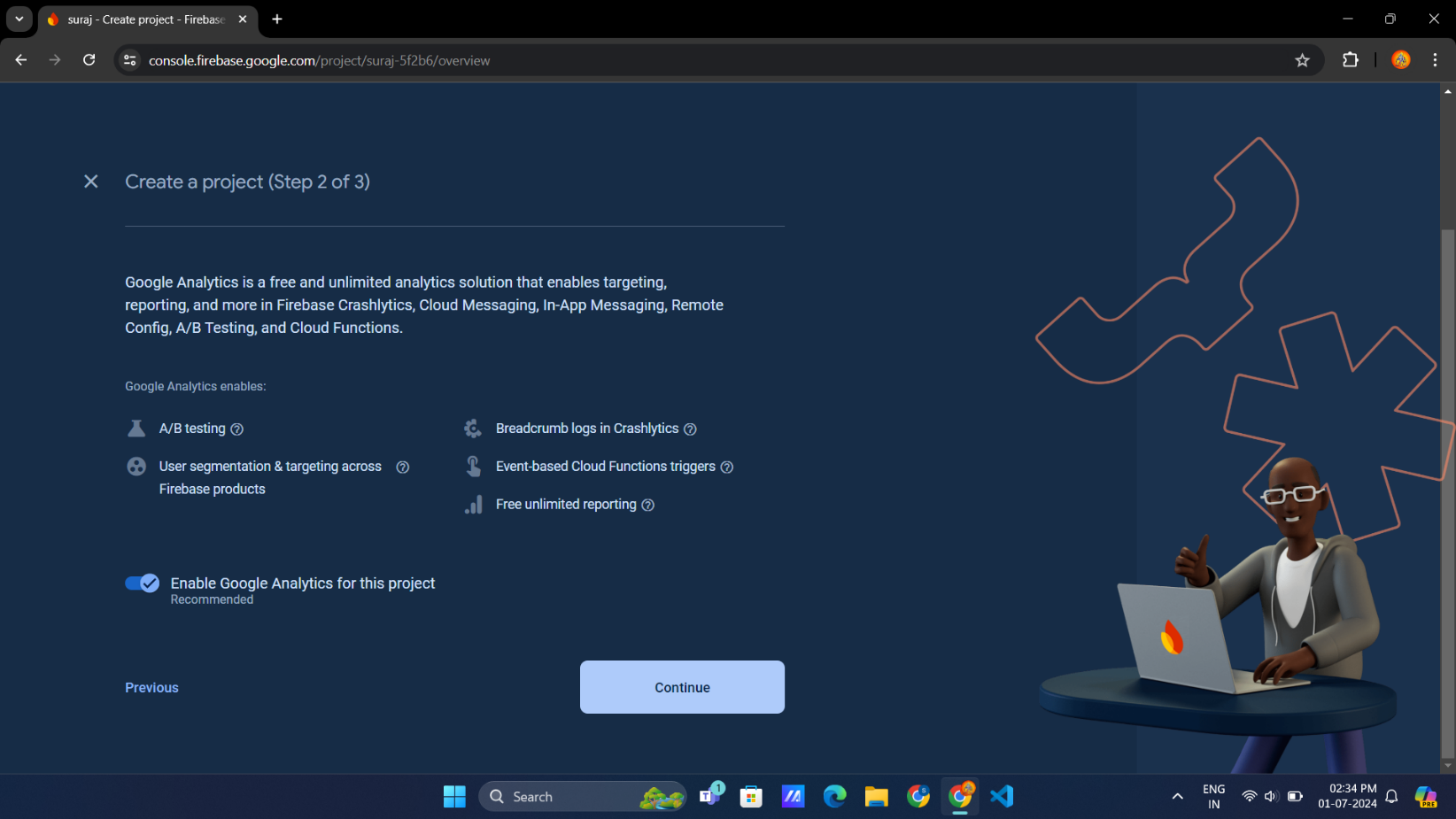
        .map((addon) => "${addon.name} (${formatMoney(addon.price)})")

        .join(", ");

  }

1. **Now its time to work on firebase**

****

****

1. Using it on frontend
2. Now initializing Firebase app in main.dart 🡺

import 'package:firebase\_core/firebase\_core.dart';

import 'package:food\_delivery/firebase\_options.dart';

import 'package:food\_delivery/services/auth/check\_login\_logout.dart';

import 'package:food\_delivery/utils/exports.dart';

//making async app

void main() async {

  //initaliziing app

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(options: DefaultFirebaseOptions.currentPlatform);

  //runapp

  runApp(MultiProvider(

    providers: [

      //theme

      ChangeNotifierProvider(

        create: (context) => ThemeProvider(),

      ),

      //restorent

      ChangeNotifierProvider(

        create: (context) => Restorant(),

      ),

    ],

    child: const MyApp(),

  ));

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      title: 'Food Delivery',

      home: const CheckLoginLogout(),

      //accessing theme provider

      theme: Provider.of<ThemeProvider>(context).themeData,

    );

  }

}

1. Creating methods 🡺

import 'package:firebase\_auth/firebase\_auth.dart';

class AuthService {

  //get instance

  final FirebaseAuth \_auth = FirebaseAuth.instance;

  //get user

  User? currentUser() {

    return \_auth.currentUser;

  }

  //sign in

  Future<UserCredential> signIn(String email, password) async {

    try {

      //sign in

      UserCredential userCredential = await \_auth.signInWithEmailAndPassword(

          email: email, password: password);

      return userCredential;

    }

    //error handling

    on FirebaseAuthException catch (e) {

      throw Exception(e.code);

    }

  }

  //sign up

  Future<UserCredential> signUp(String email, password) async {

    try {

      //sign up

      UserCredential userCredential = await \_auth

          .createUserWithEmailAndPassword(email: email, password: password);

      return userCredential;

    }

    //error handling

    on FirebaseAuthException catch (e) {

      throw Exception(e.code);

    }

  }

  //log out

  Future<void> logOut() async {

    return \_auth.signOut();

  }

}

1. Using Stream builder to check user login-out process 🡺

import 'package:firebase\_auth/firebase\_auth.dart';

import 'package:flutter/material.dart';

import 'package:food\_delivery/screens/home\_screen.dart';

import 'package:food\_delivery/screens/login\_screen.dart';

class CheckLoginLogout extends StatefulWidget {

  const CheckLoginLogout({super.key});

  @override

  State<CheckLoginLogout> createState() => \_CheckLoginLogoutState();

}

class \_CheckLoginLogoutState extends State<CheckLoginLogout> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: StreamBuilder(

        stream: FirebaseAuth.instance.authStateChanges(),

        builder: (context, snapshot) {

          //user is logged in

          if (snapshot.hasData) {

            return const HomeScreen();

          }

          //user is not logged in

          else {

            return const LoginScreen();

          }

        },

      ),

    );

  }

}

1. Now signing up

//logic

  void register() async {

//instance of auth service

    AuthService authService = AuthService();

    //matching passwords then register

    if (passwordController.text == confirmPasswordController.text) {

      //creating user

      try {

        await authService.signUp(emailController.text, passwordController.text);

      } catch (e) {

        ShowingSnackBar().showingSnackBar(

          context,

          e.toString(),

        );

      }

    } else {

      ShowingSnackBar().showingSnackBar(

        context,

        "Password Dont Match!",

      );

    }

  }

1. Sign in

//login method

  void login() async {

//instance of authservice

    AuthService authService = AuthService();

    try {

      setState(() {

        isLoading = true;

      });

      await authService.signIn(emailController.text, passwordController.text);

      //nav to home

      Navigator.pushReplacement(

        context,

        MaterialPageRoute(

          builder: (context) {

            return const HomeScreen();

          },

        ),

      );

    } catch (e) {

      ShowingSnackBar().showingSnackBar(

        context,

        e.toString(),

      );

    } finally {

      setState(() {

        isLoading = false;

      });

    }

  }

1. Log out

   HomeDrawerTile(

            title: "L O G O U T",

            icon: Icons.logout,

            onTap: () {

              //creating instance

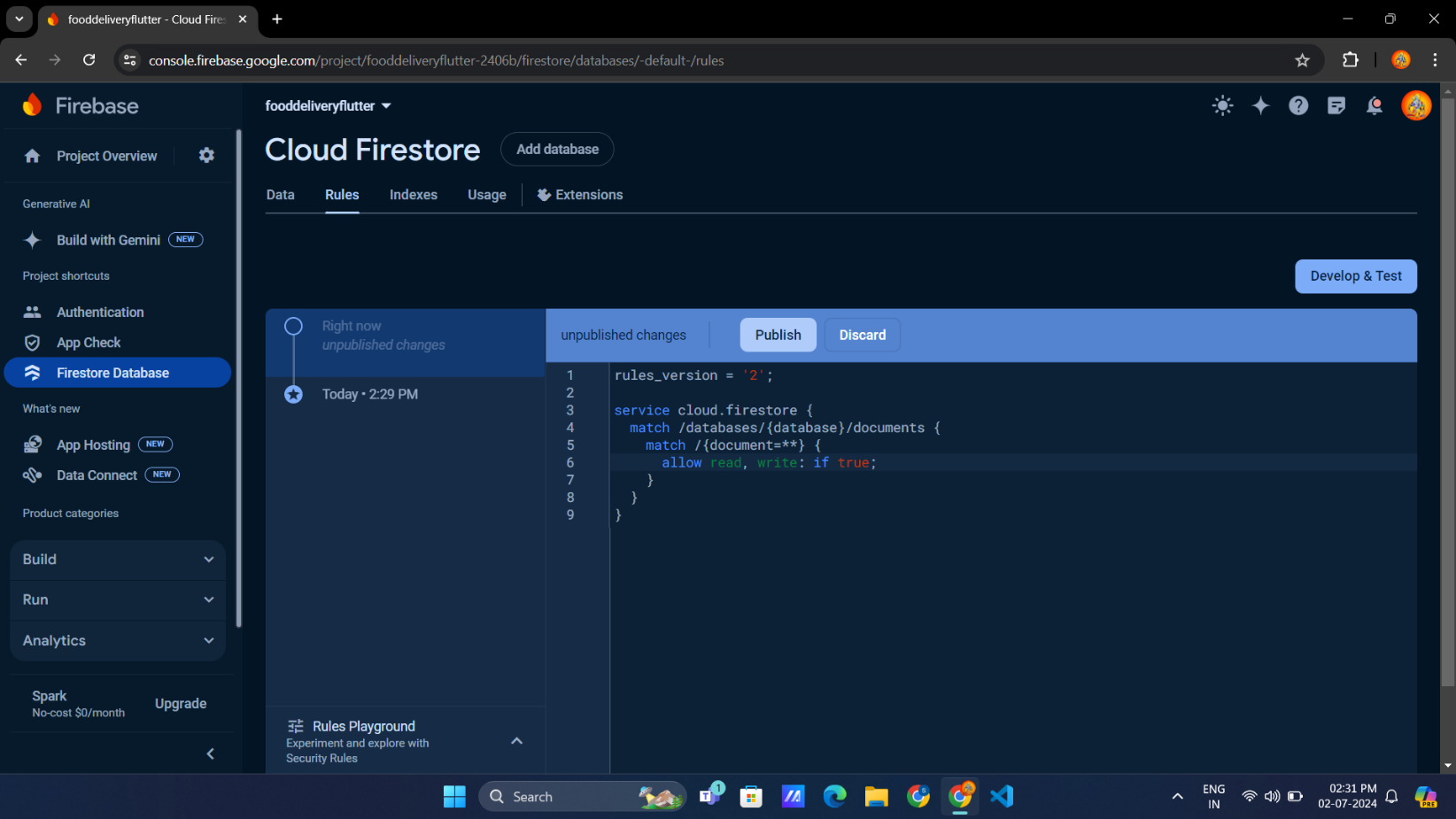
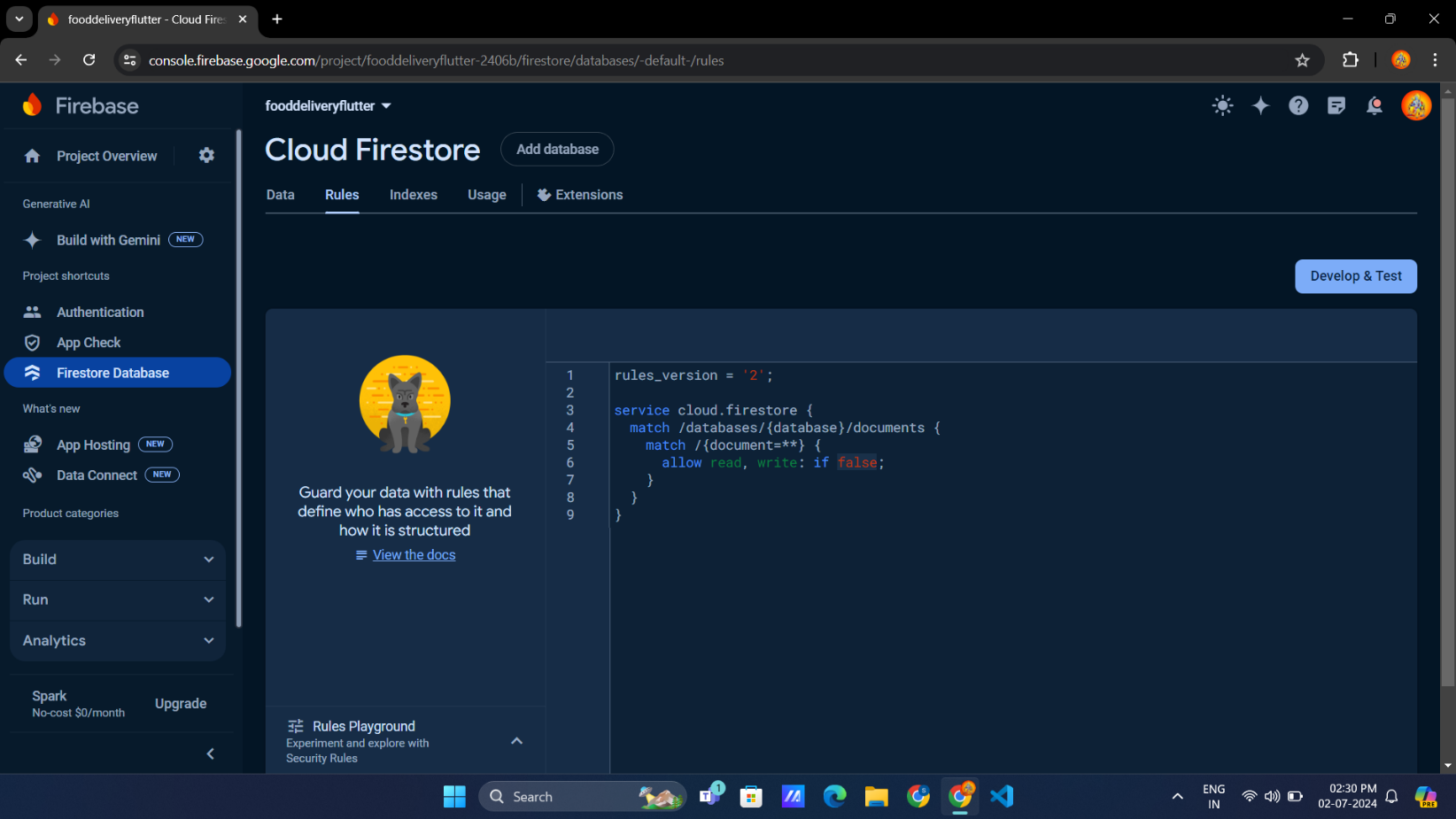
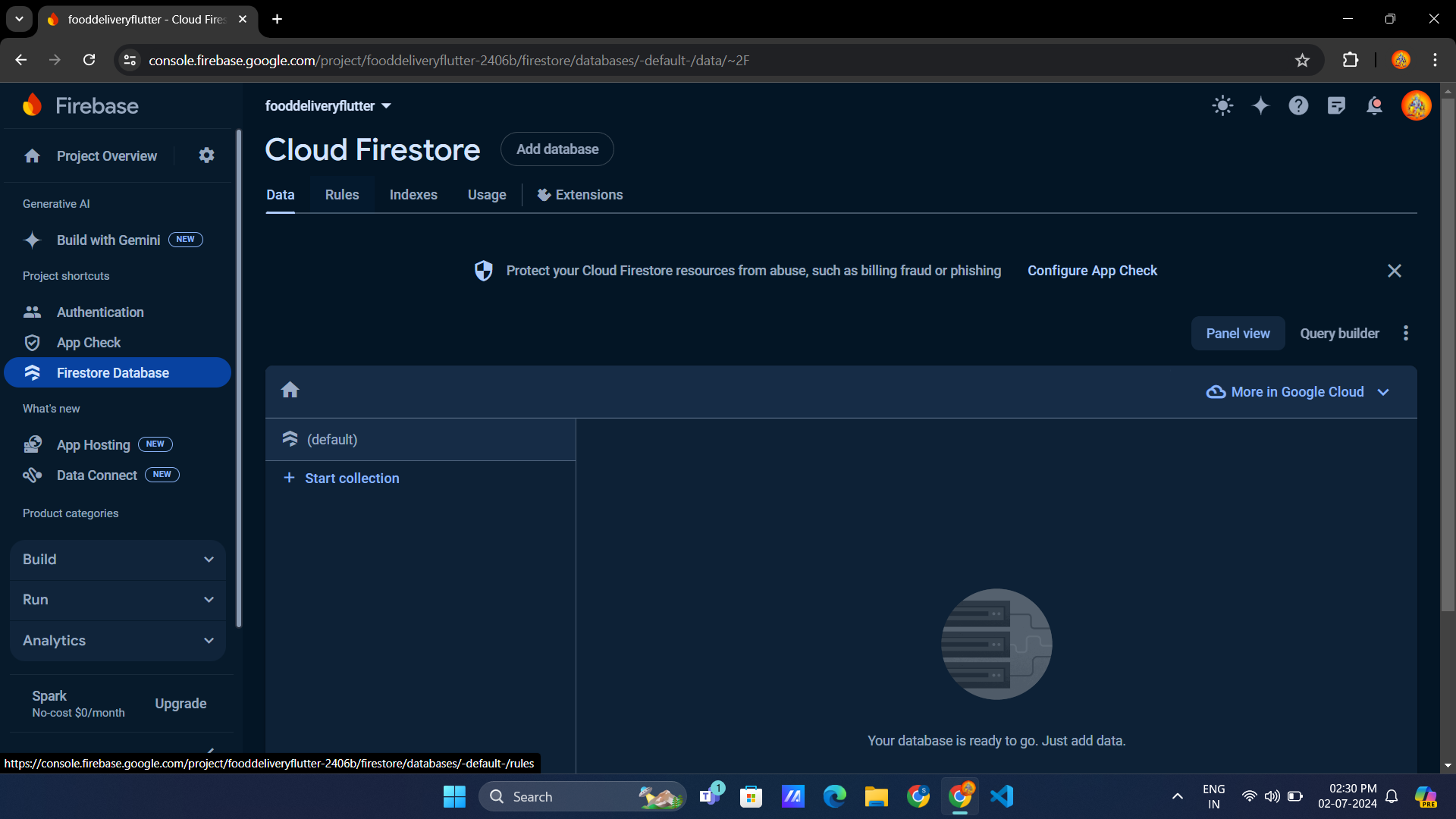
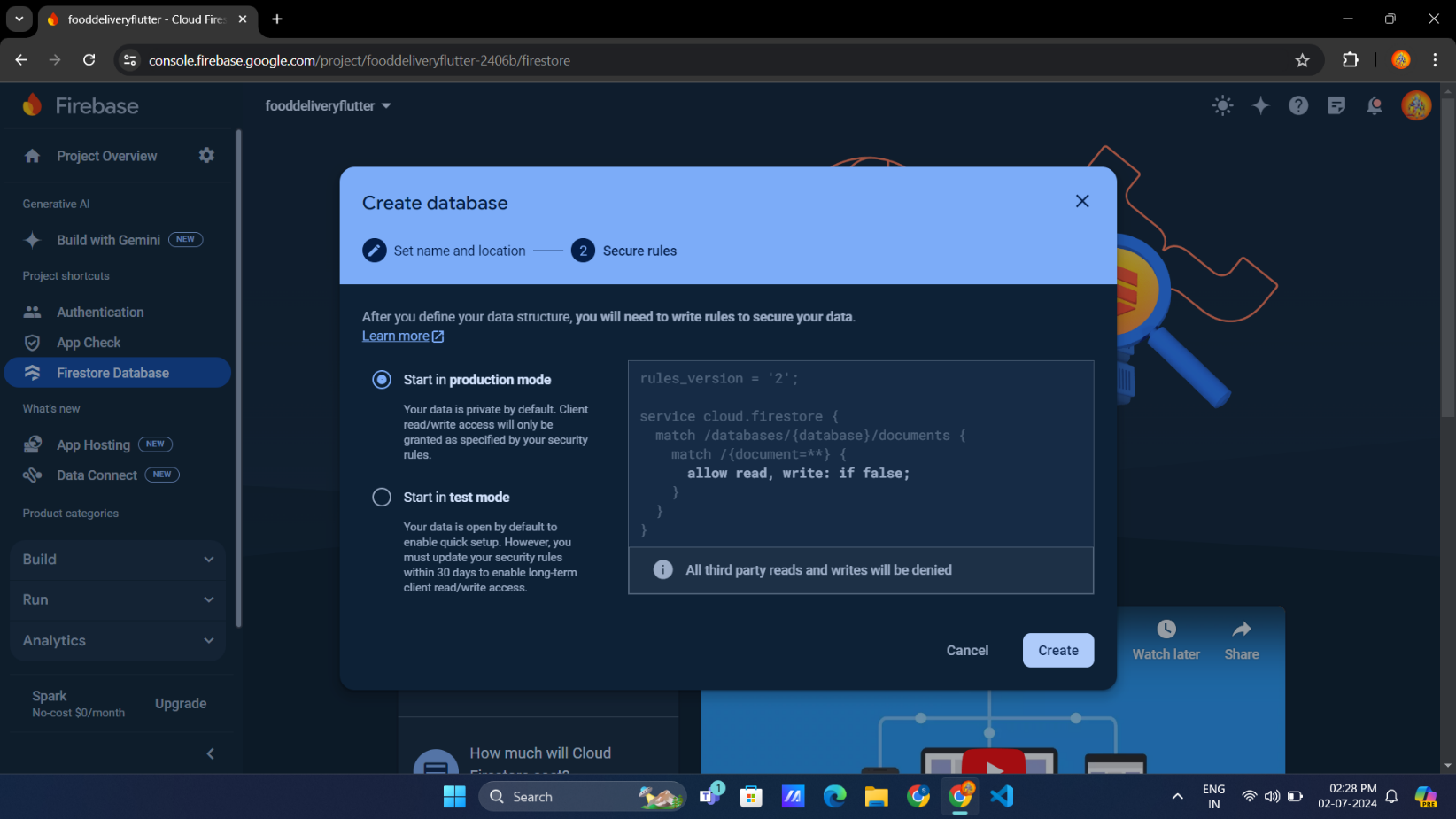
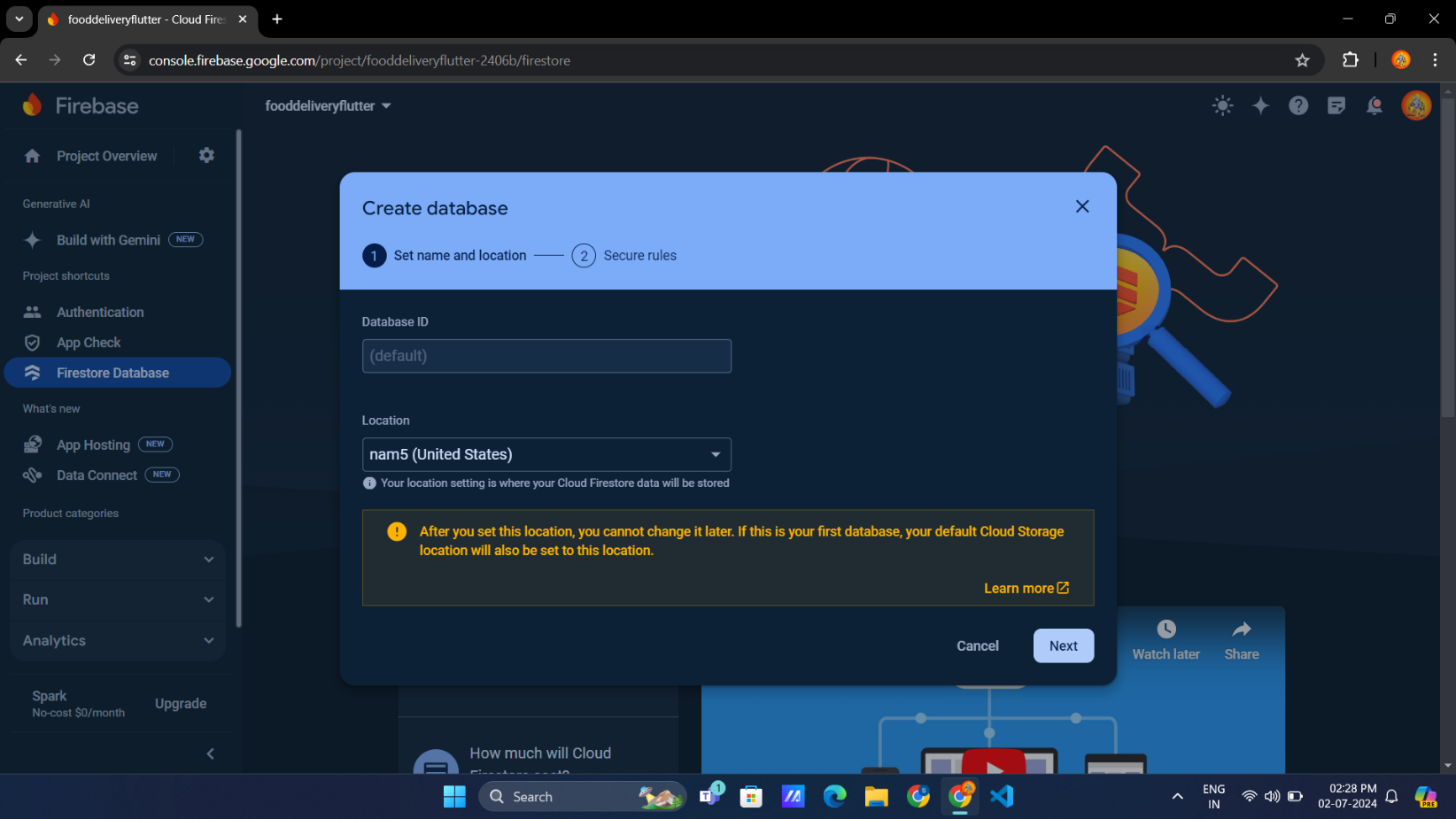
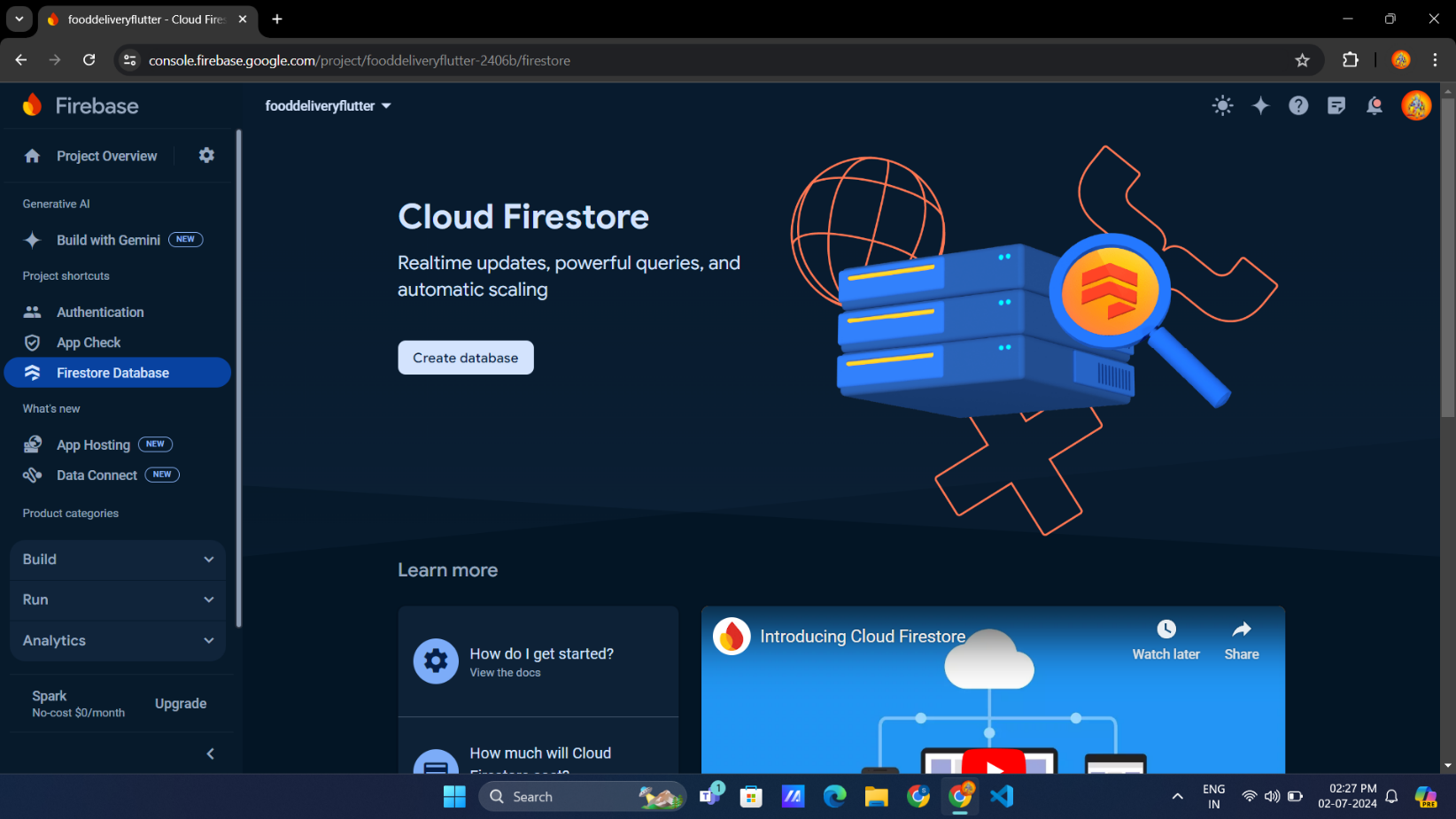
              AuthService authService = AuthService();

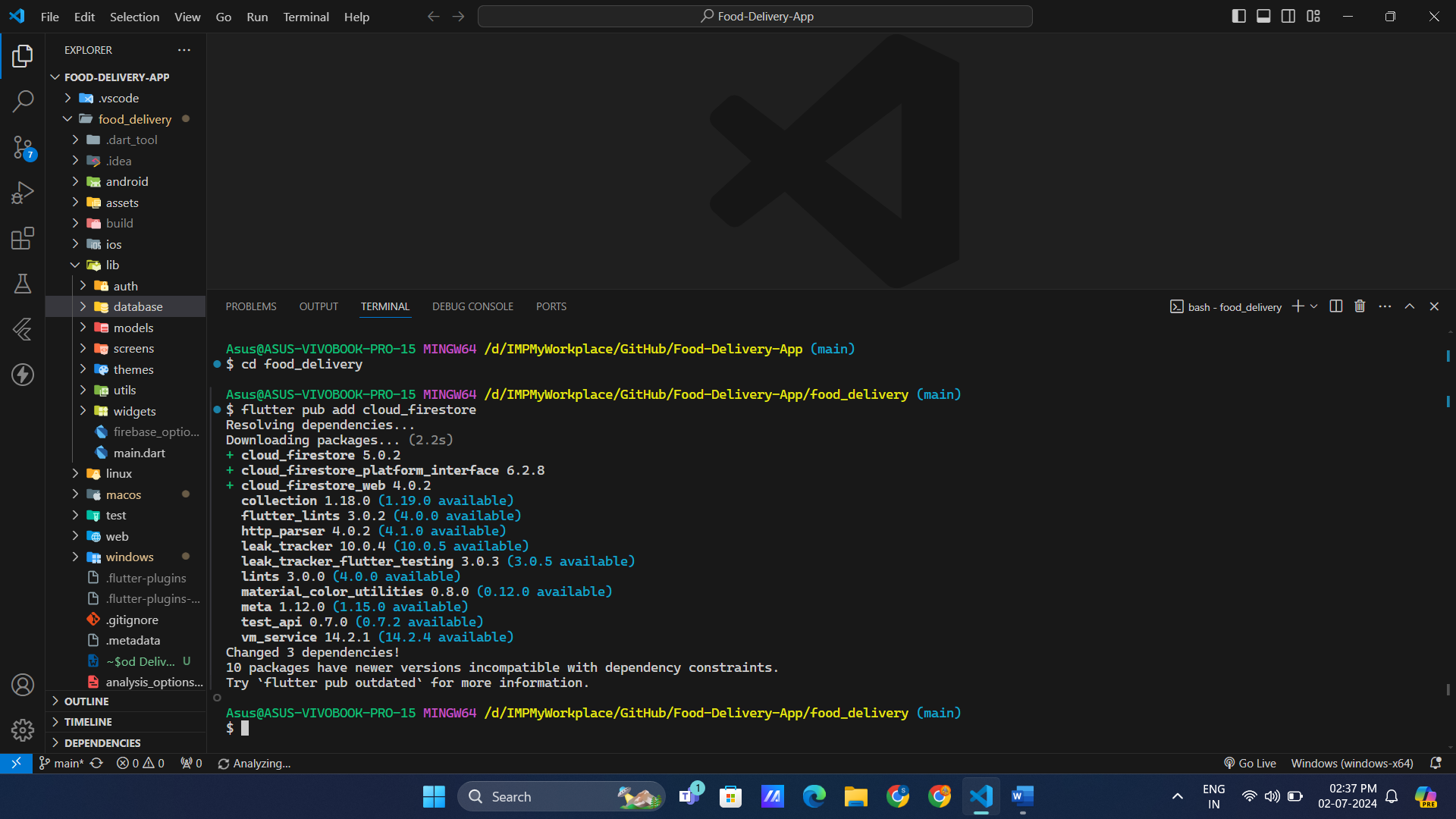
              authService.logOut();

            },

          ),

1. Now let’s create database to store data





1. Updating delivery address
2. Add to provider

class Restorant extends ChangeNotifier {

  //delivery address initaily

  String \_deliveryAddress = "kolhapur, Maharastra,416203";

//codes

  String get deliveryAddress => \_deliveryAddress;

//codes

//6. update address

  void updateAddress(String address) {

    \_deliveryAddress = address;

    notifyListeners();

  }

//codes

1. Now use it in frontend

import 'package:food\_delivery/utils/exports.dart';

class MyCurrentAddress extends StatefulWidget {

  const MyCurrentAddress({super.key});

  @override

  State<MyCurrentAddress> createState() => \_MyCurrentAddressState();

}

class \_MyCurrentAddressState extends State<MyCurrentAddress> {

//controller

  TextEditingController addressController = TextEditingController();

  void locationSearchBox(BuildContext context) {

    showDialog(

      context: context,

      builder: (context) {

        return AlertDialog(

          title: const Text("Your Location"),

          content: TextFormField(

            controller: addressController,

            decoration: const InputDecoration(hintText: "current address"),

          ),

          actions: [

            MaterialButton(

              onPressed: () => Navigator.pop(context),

              child: const Text("Cancel"),

            ),

            MaterialButton(

              onPressed: () {

                //updating address

                String newAddress = addressController.text;

                context.read<Restorant>().updateAddress(newAddress);

                Navigator.pop(context);

              },

              child: const Text("Save"),

            ),

          ],

        );

      },

    );

  }

  @override

  Widget build(BuildContext context) {

    return Padding(

      padding: const EdgeInsets.all(25.0),

      child: Column(

        crossAxisAlignment: CrossAxisAlignment.start,

        children: [

          Text(

            "Diliver now",

            style: TextStyle(color: AppColors.primary(context)),

          ),

          GestureDetector(

            onTap: () => locationSearchBox(context),

            child: Row(

              children: [

                //address

                Consumer<Restorant>(

                  builder: (context, value, child) {

                    return Text(

                      value.deliveryAddress,

                      style: TextStyle(

                        color: AppColors.primary(context),

                        fontWeight: FontWeight.bold,

                      ),

                    );

                  },

                ),

                const Icon(Icons.keyboard\_arrow\_down\_rounded)

              ],

            ),

          )

        ],

      ),

    );

  }

}

1. Now add to receipt

//geneerate receipt

  String receipt() {

    final receipt = StringBuffer();

    receipt.writeln("Here is Your Receipt");

    receipt.writeln();

    //formating data

    String formatdate =

        DateFormat('dd/MM/yyyy HH:mm:ss').format(DateTime.now());

    receipt.writeln(formatdate);

    receipt.writeln();

    receipt.writeln("\_\_\_\_\_\_\_\_\_\_\_\_");

    receipt.writeln();

    for (int i = 0; i < cart.length; i++) {

      //adding detals

      receipt.writeln(

          "${i + 1}) ${cart[i].quantity} X ${cart[i].food.name} - ${formatMoney(cart[i].food.price)}");

      //adding addons

      if (cart[i].selectedAddon.isNotEmpty) {

        receipt.writeln("Add-ons: ${formatingAddons(cart[i].selectedAddon)}");

        receipt.writeln();

      }

    }

    receipt.writeln();

    receipt.writeln("\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

    receipt.writeln("");

    //items price

    receipt.writeln("Total Items : ${getTotalItemCount()}");

    receipt.writeln("Total Price : ${formatMoney(totalPrice())}");

    //adding address

    receipt.writeln("\_\_\_\_\_\_\_\_\_\_\_\_");

    receipt.writeln(deliveryAddress);

    return receipt.toString();

  }

1. Now store it in db

import 'package:food\_delivery/database/firestore\_db.dart';

import 'package:food\_delivery/utils/exports.dart';

import 'package:food\_delivery/widgets/my\_receipt.dart';

class PaymentProcessing extends StatefulWidget {

  const PaymentProcessing({super.key});

  @override

  State<PaymentProcessing> createState() => \_PaymentProcessingState();

}

class \_PaymentProcessingState extends State<PaymentProcessing> {

  //instance of db

  FirestoreDb firestoreDb = FirestoreDb();

  //add receipt to db when user come to pay

  @override

  void initState() {

    super.initState();

    String address = context.read<Restorant>().deliveryAddress;

    String receipt = context.read<Restorant>().receipt();

    firestoreDb.saveOrdersToDb(receipt, address);

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      backgroundColor: AppColors.surface(context),

      appBar: AppBar(

        backgroundColor: Colors.transparent,

      ),

      body: const SingleChildScrollView(

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          children: [

            AddVerticleSpace(height: 20),

            Padding(

              padding: EdgeInsets.all(8.0),

              child: MyReceipt(),

            ),

          ],

        ),

      ),

      bottomNavigationBar: bottom(context),

    );

  }

}

