**FOLDER “bookinghotel” có:**

* lib:

- Folder “model”:

+ File “app\_constants.dart”:

import 'package:guest\_hotel/model/contact\_model.dart';

import 'package:guest\_hotel/model/user\_model.dart';

class AppConstants {

  static UserModel currentUser = UserModel();

  ContactModel createContactFromUserModel() {

    return ContactModel(

        id: currentUser.id,

        firstName: currentUser.firstName,

        lastName: currentUser.lastName,

        displayImage: currentUser.displayImage);

  }

  static final Map<int, int> daysInMonths = {

    1: 31,

    2: DateTime.now().year % 4 == 0 ? 29 : 28,

    3: 31,

    4: 30,

    5: 31,

    6: 30,

    7: 31,

    8: 31,

    9: 30,

    10: 31,

    11: 30,

    12: 31,

  };

  static final Map<int, String> monthDict = {

    1: "January",

    2: "February",

    3: "March",

    4: "April",

    5: "May",

    6: "June",

    7: "July",

    8: "August",

    9: "September",

    10: "October",

    11: "November",

    12: "December",

  };

}

+ File “booking\_model.dart”:

import 'package:guest\_hotel/model/contact\_model.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

class BookingModel {

  String id = "";

  PostingModel? posting;

  ContactModel? contact;

  List<DateTime>? dates;

  BookingModel();

  getBookingInfoFromFirestoreFromPosting(

      PostingModel posting, DocumentSnapshot snapshot) async {

    posting = posting;

    List<Timestamp> timestamps = List<Timestamp>.from(snapshot['dates']) ?? [];

    dates = [];

    timestamps.forEach((timestamps) {

      dates!.add((timestamps.toDate()));

    });

    String contactID = snapshot['userID'] ?? "";

    String fullName = snapshot['name'] ?? "";

    \_loadContactInfo(id, fullName);

    contact = ContactModel(id: contactID);

  }

  \_loadContactInfo(String id, String fullName) {

    String firstName = "";

    String lastName = "";

    firstName = fullName.split(" ")[0];

    lastName = fullName.split(" ")[1];

    contact = ContactModel(id: id, firstName: firstName, lastName: lastName);

  }

  createBooking(

      PostingModel postingM, ContactModel contactM, List<DateTime> datesM) {

    posting = postingM;

    contact = contactM;

    dates = datesM;

    dates!.sort();

  }

}

+ File “contact\_model.dart”:

import 'package:guest\_hotel/model/user\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

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

import 'package:flutter/material.dart';

class ContactModel {

  String? id;

  String? firstName;

  String? lastName;

  String? fullName;

  MemoryImage? displayImage;

  ContactModel({

    this.id = "",

    this.firstName = "",

    this.lastName = "",

    this.displayImage,

  });

  String getFullNameOfUser() {

    return fullName = "${firstName!} ${lastName!}";

  }

  UserModel createUserFormContact() {

    return UserModel(

      id: id!,

      firstName: firstName!,

      lastName: lastName!,

      displayImage: displayImage!,

    );

  }

  getContactInfoFromFirestore() async {

    DocumentSnapshot snapshot =

        await FirebaseFirestore.instance.collection('users').doc(id).get();

    firstName = snapshot['firstName'] ?? "";

    lastName = snapshot['lastName'] ?? "";

  }

  getImageFromStorage() async {

    if (displayImage != null) {

      return displayImage!;

    }

    final imageData = await FirebaseStorage.instance

        .ref()

        .child("userImages")

        .child(id!)

        .child("$id.png")

        .getData(1024 \* 1024);

    displayImage = MemoryImage(imageData!);

    return displayImage;

  }

}

+ File “conversation\_model.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/contact\_model.dart';

import 'package:guest\_hotel/model/message\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

class ConversationModel {

  String? id;

  ContactModel? otherContact;

  List<MessageModel>? messages;

  MessageModel? lastMessage;

  ConversationModel() {

    messages = [];

  }

  addConversationToFirestore(ContactModel otherContact) async {

    List<String> userNames = [

      AppConstants.currentUser.getFullNameOfUser(),

      otherContact.getFullNameOfUser(),

    ];

    List<String> userIDs = [

      AppConstants.currentUser.id!,

      otherContact.id!,

    ];

    Map<String, dynamic> conversationDataMap = {

      'lastMessageDateTime': DateTime.now(),

      'lastMessageText': "",

      'userNames': userNames,

      'userIDs': userIDs,

    };

    DocumentReference reference = await FirebaseFirestore.instance

        .collection('conversations')

        .add(conversationDataMap);

    id = reference.id;

  }

  addMessageToFirestore(String messageText) async {

    Map<String, dynamic> messageData = {

      'dataTime': DateTime.now(),

      'senderID': AppConstants.currentUser.id,

      'text': messageText

    };

    await FirebaseFirestore.instance

        .collection('conversations/${id}/messages')

        .add(messageData);

    Map<String, dynamic> conversationData = {

      'lastMessageDateTime': DateTime.now(),

      'lastMessageText': messageText

    };

    await FirebaseFirestore.instance

        .doc('converstations/${id}')

        .update(conversationData);

  }

  getConversationInfoFromFirestore(DocumentSnapshot snapshot) {

    id = snapshot.id;

    String lastMessageText = snapshot['lastMessageText'] ?? "";

    Timestamp lastMessageDateTimestamp =

        snapshot['lastMessageDateTime'] ?? Timestamp.now();

    DateTime lastMessageDateTime = lastMessageDateTimestamp.toDate();

    lastMessage = MessageModel();

    lastMessage!.dateTime = lastMessageDateTime;

    lastMessage!.text = lastMessageText;

    List<String> userIDs = List<String>.from(snapshot['userIDs']) ?? [];

    List<String> userNames = List<String>.from(snapshot['userNames']) ?? [];

    otherContact = ContactModel();

    for (String userID in userIDs) {

      if (userID != AppConstants.currentUser!.id) {

        this.otherContact!.id = userID;

        break;

      }

    }

    for (String name in userNames) {

      if (name != AppConstants.currentUser.getFullNameOfUser()) {

        otherContact!.firstName = name.split(" ")[0];

        otherContact!.lastName = name.split(" ")[1];

      }

    }

  }

}

+ File “message\_model.dart”:

import 'package:guest\_hotel/model/contact\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:timeago/timeago.dart' as timeago;

class MessageModel {

  ContactModel? sender;

  String? text;

  DateTime? dateTime;

  MessageModel();

  String getMessageDateTime() {

    return timeago.format(dateTime!);

  }

  getMessageInfoFromFirestore(DocumentSnapshot snapshot) {

    Timestamp lastMessageTimeStamp = snapshot['dateTime'] ?? Timestamp.now();

    dateTime = lastMessageTimeStamp.toDate();

    String senderID = snapshot['senderID'] ?? "";

    sender = ContactModel(id: senderID);

    text = snapshot['text'] ?? "";

  }

}

+ File “posting\_model.dart”:

import 'package:guest\_hotel/global.dart';

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/booking\_model.dart';

import 'package:guest\_hotel/model/contact\_model.dart';

import 'package:guest\_hotel/model/review\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

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

import 'package:flutter/cupertino.dart';

import 'package:get/get.dart';

class PostingModel {

  String? id;

  String? name;

  String? type;

  double? price;

  String? description;

  String? address;

  String? city;

  String? country;

  double? rating;

  ContactModel? host;

  List<String>? imageNames;

  List<MemoryImage>? displayImages;

  List<String>? amenities;

  Map<String, int>? beds;

  Map<String, int>? bathrooms;

  List<BookingModel>? bookings;

  List<ReviewModel>? reviews;

  PostingModel(

      {this.id = "",

      this.name = "",

      this.type = "",

      this.price = 0,

      this.description = "",

      this.address = "",

      this.city = "",

      this.country = "",

      this.host}) {

    displayImages = [];

    amenities = [];

    beds = {};

    bathrooms = {};

    rating = 0;

    bookings = [];

    reviews = [];

  }

  setImagesName() {

    imageNames = [];

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

      imageNames!.add("image$i.png");

    }

  }

  getPostingInfoFromFirestore() async {

    DocumentSnapshot snapshot =

        await FirebaseFirestore.instance.collection('postings').doc(id).get();

    getPostingInfoFromSnapshot(snapshot);

  }

  getPostingInfoFromSnapshot(DocumentSnapshot snapshot) {

    address = snapshot['address'] ?? "";

    amenities = List<String>.from(snapshot['amenities']) ?? [];

    bathrooms = Map<String, int>.from(snapshot['bathrooms']) ?? {};

    beds = Map<String, int>.from(snapshot['beds']) ?? {};

    city = snapshot['city'] ?? "";

    country = snapshot['country'] ?? "";

    description = snapshot['description'] ?? "";

    String hostID = snapshot['hostID'] ?? "";

    host = ContactModel(id: hostID);

    imageNames = List<String>.from(snapshot['imageNames']) ?? [];

    name = snapshot['name'] ?? "";

    price = snapshot['price'].toDouble() ?? 0.0;

    rating = snapshot['rating'].toDouble() ?? 2.5;

    type = snapshot['type'] ?? "";

  }

  getAllImagesFromStorage() async {

    displayImages = [];

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

      final imageData = await FirebaseStorage.instance

          .ref()

          .child("postingImages")

          .child(id!)

          .child(imageNames![i])

          .getData(1024 \* 1024);

      displayImages!.add(MemoryImage(imageData!));

    }

    return displayImages;

  }

  getFirstImageFromStorage() async {

    if (displayImages!.isEmpty) {

      return displayImages!.first;

    }

    final imageData = await FirebaseStorage.instance

        .ref()

        .child("postingImages")

        .child(id!)

        .child(imageNames!.first)

        .getData(1024 \* 1024);

    displayImages!.add(MemoryImage(imageData!));

    return displayImages!.first;

  }

  getAmenititesString() {

    if (amenities!.isEmpty) {

      return "";

    }

    String amenitiesString = amenities.toString();

    return amenitiesString.substring(1, amenitiesString.length - 1);

  }

  double getCurrentRating() {

    if (reviews!.length == 0) {

      return 4;

    }

    double rating = 0;

    reviews!.forEach((review) {

      rating += review.rating!;

    });

    rating /= reviews!.length;

    return rating;

  }

  getHostFromFirestore() async {

    await host!.getContactInfoFromFirestore();

    await host!.getImageFromStorage();

  }

  int getGuestsNumber() {

    int? numGuests = 0;

    numGuests = numGuests + beds!['small']!;

    numGuests = numGuests + beds!['medium']! \* 2;

    numGuests = numGuests + beds!['large']! \* 2;

    return numGuests;

  }

  String getBedroomText() {

    String text = "";

    if (this.beds!["small"] != 0) {

      text = text + beds!["small"].toString() + "single/twin";

    }

    if (this.beds!["medium"] != 0) {

      text = text + beds!["medium"].toString() + "double";

    }

    if (this.beds!["large"] != 0) {

      text = text + beds!["large"].toString() + "large";

    }

    return text;

  }

  String getBathroomText() {

    String text = "";

    if (bathrooms!["full"] != 0) {

      text = text + bathrooms!["full"].toString() + "full";

    }

    if (bathrooms!["half"] != 0) {

      text = text + bathrooms!["half"].toString() + "half ";

    }

    return text;

  }

  String getFullAddress() {

    return address! + ", " + city! + ", " + country!;

  }

  getAllBookingsFromFirestore() async {

    bookings = [];

    QuerySnapshot snapshots = await FirebaseFirestore.instance

        .collection('postings')

        .doc(id)

        .collection('bookings')

        .get();

    for (var snapshot in snapshots.docs) {

      BookingModel newBooking = BookingModel();

      await newBooking.getBookingInfoFromFirestoreFromPosting(this, snapshot);

      bookings!.add(newBooking);

    }

  }

  List<DateTime> getAllBookedDates() {

    List<DateTime> dates = [];

    bookings!.forEach((booking) {

      dates.addAll(booking.dates!);

    });

    return dates;

  }

  Future<void> makeNewBooking(List<DateTime> dates, context, hostID) async {

    Map<String, dynamic> bookingData = {

      'dates': dates,

      'name': AppConstants.currentUser.getFullNameOfUser(),

      'userID': AppConstants.currentUser.id,

      'payment': bookingPrice,

    };

    DocumentReference reference = await FirebaseFirestore.instance

        .collection('posting')

        .doc(id)

        .collection('booking')

        .add(bookingData);

    BookingModel newBooking = BookingModel();

    newBooking.createBooking(

        this, AppConstants.currentUser.createUserFormContact(), dates);

    newBooking.id = reference.id;

    bookings!.add(newBooking);

    await AppConstants.currentUser

        .addBookingToFirestore(newBooking, bookingPrice!, hostID);

    Get.snackbar("Listing", "Booked succesfully");

  }

  static fromFirestore(QueryDocumentSnapshot<Object?> doc) {}

}

+ File “review\_model.dart”:

import 'package:guest\_hotel/model/contact\_model.dart';

class ReviewModel {

  ContactModel? contactModel;

  String? text;

  double? rating;

  DateTime? dateTime;

  ReviewModel();

}

+ File “user\_model.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/booking\_model.dart';

import 'package:guest\_hotel/model/contact\_model.dart';

import 'package:guest\_hotel/model/conversation\_model.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/model/review\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/cupertino.dart';

import 'package:get/get.dart';

class UserModel extends ContactModel {

  String? email;

  String? password;

  String? bio;

  String? city;

  String? country;

  bool? isHost;

  bool? isCurrentlyHosting;

  DocumentSnapshot? snapshot;

  List<BookingModel>? bookings;

  List<ReviewModel>? reviews;

  List<PostingModel>? savedPostings;

  List<PostingModel>? myPostings;

  String? role;

  UserModel({

    String super.id,

    String super.firstName,

    String super.lastName,

    super.displayImage,

    this.email = "",

    this.bio = "",

    this.city = "",

    this.country = "",

  }) {

    isHost = false;

    isCurrentlyHosting = false;

    bookings = [];

    reviews = [];

    savedPostings = [];

    myPostings = [];

  }

  createContactFromUser() {

    return ContactModel(

      id: id,

      firstName: firstName,

      lastName: lastName,

      displayImage: displayImage,

    );

  }

  Future<List<PostingModel>> getPostings() async {

    List<PostingModel> postingList = [];

    QuerySnapshot snapshot = await FirebaseFirestore.instance

        .collection('postings')

        .where('hostID', isEqualTo: id)

        .get();

    for (var doc in snapshot.docs) {

      PostingModel posting = PostingModel(id: doc.id);

      await posting.getPostingInfoFromSnapshot(doc);

      postingList.add(posting);

    }

    return postingList;

  }

  Future<List<BookingModel>> getBookings() async {

    List<BookingModel> bookingList = [];

    List<PostingModel> postings = await getPostings();

    List<String> postingIDs = postings.map((posting) => posting.id!).toList();

    for (String postingID in postingIDs) {

      QuerySnapshot snapshot = await FirebaseFirestore.instance

          .collection('bookings')

          .where('postingID', isEqualTo: postingID)

          .get();

      for (var doc in snapshot.docs) {

        BookingModel booking = BookingModel();

        booking.id = doc.id;

        booking.dates = List<DateTime>.from(

            doc['dates'].map((date) => DateTime.parse(date)));

        bookingList.add(booking);

      }

    }

    return bookingList;

  }

  addPostingToMyPostings(PostingModel posting) async {

    myPostings!.add(posting);

    List<String> myPostingIDsList = [];

    for (var element in myPostings!) {

      myPostingIDsList.add(element.id!);

    }

    await FirebaseFirestore.instance.collection("users").doc(id).update({

      'myPostingIDs': myPostingIDsList,

    });

  }

  getMyPostingsFromFirestore() async {

    List<String> myPostingIDs =

        List<String>.from(snapshot!["myPostingIDs"]) ?? [];

    for (String postingID in myPostingIDs) {

      PostingModel posting = PostingModel(id: postingID);

      await posting.getPostingInfoFromFirestore();

      await posting.getAllBookingsFromFirestore();

      await posting.getAllImagesFromStorage();

      myPostings!.add(posting);

    }

  }

  addSavePosting(PostingModel posting) async {

    for (var savedPosting in savedPostings!) {

      if (savedPosting.id == posting.id) {

        return;

      }

    }

    savedPostings!.add(posting);

    List<String> savedPostingIDs = [];

    savedPostings!.forEach((savedPosting) {

      savedPostingIDs.add(savedPosting.id!);

    });

    await FirebaseFirestore.instance.collection("user").doc(id).update({

      'savedPostingIDs': savedPostingIDs,

    });

    Get.snackbar("Marked as Favourite", "Saved to your Favourite List");

  }

  removeSavedPosting(PostingModel posting) async {

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

      if (savedPostings![i].id == posting.id) {

        savedPostings!.removeAt(i);

        break;

      }

    }

    List<String> savedPostingIDs = [];

    savedPostings!.forEach((savedPosting) {

      savedPostingIDs.add(savedPosting.id!);

    });

    await FirebaseFirestore.instance.collection("user").doc(id).update({

      'savedPostingIDs': savedPostingIDs,

    });

    Get.snackbar("Listing Removed", "Listing removed from your Favourite List");

  }

  Future<void> addBookingToFirestore(BookingModel booking,

      double totalPriceForAllNights, String hostID) async {

    Map<String, dynamic> data = {

      'dates': booking.dates,

      'postingID': booking.posting!.id!,

    };

    await FirebaseFirestore.instance

        .doc('user/${id}/bookings/${booking.id}')

        .set(data);

    String earningOld = "";

    await FirebaseFirestore.instance

        .collection("user")

        .doc(hostID)

        .get()

        .then((dataSnap) {

      earningOld = dataSnap["earnings"].toString();

    });

    await FirebaseFirestore.instance.collection("users").doc(hostID).update({

      "earnings": totalPriceForAllNights + int.parse(earningOld),

    });

    bookings!.add(booking);

    await addBookingConversation(booking);

  }

  addBookingConversation(BookingModel booking) async {

    ConversationModel conversation = ConversationModel();

    conversation.addConversationToFirestore(booking.posting!.host!);

    String textMessage =

        "Hi my name is ${AppConstants.currentUser!.firstName} and I have"

        "just booked ${booking.posting!.name} from ${booking.dates!.first} to"

        "${booking.dates!.last} if you have any questions contact me. Enjoy your"

        "stay!";

    await conversation.addMessageToFirestore(textMessage);

  }

  List<DateTime> getAllBookedDates() {

    List<DateTime> allBookedDates = [];

    myPostings!.forEach((posting) {

      posting.bookings!.forEach((booking) {

        allBookedDates.addAll(booking.dates!);

      });

    });

    return allBookedDates;

  }

}

- Folder “view”:

+ Folder “guestScreens”:

* File “account\_screen.dart”:

import 'package:guest\_hotel/global.dart';

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/view/guestScreens/user\_profile\_screen.dart';

import 'package:guest\_hotel/view/guest\_home\_screen.dart';

import 'package:guest\_hotel/view/login\_screen.dart';

import 'package:guest\_hotel/view\_model/user\_view\_model.dart';

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

import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:get/get\_core/src/get\_main.dart';

import 'host\_home\_screen.dart';

class AccountScreen extends StatefulWidget {

  const AccountScreen({super.key});

  @override

  State<AccountScreen> createState() => \_AccountScreenState();

}

class \_AccountScreenState extends State<AccountScreen> {

  String \_hostingTitle = 'Show my Host Dashboard';

  modifyHostingMode() async {

    if (AppConstants.currentUser.isHost!) {

      if (AppConstants.currentUser.isCurrentlyHosting!) {

        AppConstants.currentUser.isCurrentlyHosting = false;

        Get.to(const GuestHomeScreen());

      } else {

        AppConstants.currentUser.isCurrentlyHosting = true;

        Get.to(HostHomeScreen());

      }

    } else {

      await userViewModel.becomeHost(FirebaseAuth.instance.currentUser!.uid);

      AppConstants.currentUser.isCurrentlyHosting = true;

      Get.to(HostHomeScreen());

    }

  }

  Future<void> logout() async {

    try {

      // Xử lý đăng xuất

      Navigator.pushReplacement(

        context,

        MaterialPageRoute(builder: (context) => const LoginScreen()),

      );

    } catch (error) {

      showDialog(

        context: context,

        builder: (context) => AlertDialog(

          title: const Text('Logout Failed'),

          content: Text(error.toString()),

          actions: [

            TextButton(

              onPressed: () {

                Navigator.of(context).pop();

              },

              child: const Text('OK'),

            ),

          ],

        ),

      );

    }

  }

  Future<void> showLogoutConfirmationDialog() async {

    showDialog(

      context: context,

      builder: (context) {

        return AlertDialog(

          title: const Text('Confirm Logout'),

          content: const Text('Do you want to log out of this account?'),

          actions: [

            TextButton(

              onPressed: () {

                Navigator.of(context).pop(); // Đóng dialog

              },

              child: const Text('No'),

            ),

            ElevatedButton(

              onPressed: () {

                Navigator.of(context).pop(); // Đóng dialog

                logout(); // Thực hiện logout

              },

              style: ElevatedButton.styleFrom(backgroundColor: Colors.red),

              child: const Text('Yes'),

            ),

          ],

        );

      },

    );

  }

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    if (AppConstants.currentUser.isHost!) {

      if (AppConstants.currentUser.isCurrentlyHosting!) {

        \_hostingTitle = 'Show my Guest Dashboard';

      } else {

        \_hostingTitle = 'Show my Host Dashboard';

      }

    } else {

      \_hostingTitle = 'You are become a host';

    }

  }

  @override

  Widget build(BuildContext context) {

    return SingleChildScrollView(

        child: Padding(

            padding: const EdgeInsets.fromLTRB(20, 50, 20, 20),

            child: Column(

              mainAxisAlignment: MainAxisAlignment.start,

              crossAxisAlignment: CrossAxisAlignment.start,

              children: [

                // user infor

                Padding(

                  padding: const EdgeInsets.only(bottom: 30),

                  child: Center(

                    child: Column(

                      children: [

                        //image

                        MaterialButton(

                          onPressed: () {},

                          child: CircleAvatar(

                            backgroundColor: Colors.black,

                            radius: MediaQuery.of(context).size.width / 4.5,

                            child: CircleAvatar(

                              backgroundImage:

                                  AppConstants.currentUser.displayImage,

                              radius: MediaQuery.of(context).size.width / 4.6,

                            ),

                          ),

                        ),

                        const SizedBox(

                          height: 10,

                        ),

//name and email

                        Column(

                          crossAxisAlignment: CrossAxisAlignment.center,

                          children: [

                            Text(

                              AppConstants.currentUser.getFullNameOfUser(),

                              style: const TextStyle(

                                fontWeight: FontWeight.bold,

                                fontSize: 20,

                              ),

                            ),

                            Text(

                              AppConstants.currentUser.email.toString(),

                              style: const TextStyle(

                                fontWeight: FontWeight.bold,

                                fontSize: 15,

                              ),

                            )

                          ],

                        )

                      ],

                    ),

                  ),

                ),

                ListView(

                  shrinkWrap: true,

                  children: [

                    // Personal Information Button

                    Container(

                      decoration: const BoxDecoration(color: Colors.green),

                      child: MaterialButton(

                        height: MediaQuery.of(context).size.height / 9,

                        onPressed: () {

                          Navigator.push(

                            context,

                            MaterialPageRoute(

                                builder: (context) => UserProfileScreen()),

                          );

                        },

                        child: const ListTile(

                          contentPadding: EdgeInsets.all(0.0),

                          leading: Text(

                            "Personal Information",

                            style: TextStyle(

                              fontSize: 20,

                              fontWeight: FontWeight.normal,

                            ),

                          ),

                          trailing: Icon(

                            size: 34,

                            Icons.person\_outlined,

                          ),

                        ),

                      ),

                    ),

                    const SizedBox(

                      height: 10,

                    ),

                    //Change Hosting Button

                    Container(

                      decoration: const BoxDecoration(color: Colors.green),

                      child: MaterialButton(

                        height: MediaQuery.of(context).size.height / 9,

                        onPressed: () {

                          modifyHostingMode();

                          setState(() {});

                        },

                        child: ListTile(

                          contentPadding: const EdgeInsets.all(0.0),

                          leading: Text(

                            \_hostingTitle,

                            style: const TextStyle(

                              fontSize: 20,

                              fontWeight: FontWeight.normal,

                            ),

                          ),

                          trailing: const Icon(

                            size: 34,

                            Icons.home\_outlined,

                          ),

                        ),

                      ),

                    ),

                    const SizedBox(

                      height: 10,

                    ),

                    //Logout

                    Container(

                      decoration: const BoxDecoration(color: Colors.green),

                      child: MaterialButton(

                        height: MediaQuery.of(context).size.height / 9,

                        onPressed: () {

                          showLogoutConfirmationDialog();

                        },

                        child: const ListTile(

                          contentPadding: EdgeInsets.all(0.0),

                          leading: Text(

                            "Logout",

                            style: TextStyle(

                              fontSize: 20,

                              fontWeight: FontWeight.normal,

                            ),

                          ),

                          trailing: Icon(

                            size: 34,

                            Icons.logout\_outlined,

                          ),

                        ),

                      ),

                    ),

                    const SizedBox(

                      height: 20,

                    ),

                  ],

                )

              ],

            )));

  }

}

* File “book\_listing\_screen.dart”:

import 'dart:io';

import 'package:guest\_hotel/global.dart';

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view/widgets/calender\_ui.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

import 'package:get/get.dart';

class BookListingScreen extends StatefulWidget {

  final PostingModel? posting;

  final String? hostID;

  BookListingScreen({

    super.key,

    this.posting,

    this.hostID,

  });

  @override

  State<BookListingScreen> createState() => \_BookListingScreenState();

}

class \_BookListingScreenState extends State<BookListingScreen> {

  PostingModel? posting;

  List<DateTime> bookedDates = [];

  List<DateTime> selectedDates = [];

  List<CalenderUI> calendarWidgets = [];

  \_buildCalendarWidgets() {

    for (int i = 0; i < 12; i++) {

      calendarWidgets.add(CalenderUI(

        monthIndex: i,

        bookedDates: bookedDates,

        selectDate: \_selectDate,

        getSelectedDates: \_getSelectedDates,

      ));

      setState(() {});

    }

  }

  List<DateTime> \_getSelectedDates() {

    return selectedDates;

  }

  \_selectDate(DateTime date) {

    if (selectedDates.contains(date)) {

      selectedDates.remove(date);

    } else {

      selectedDates.add(date);

    }

    selectedDates.sort();

    setState(() {});

  }

  \_localBookedDates() async {

    try {

      QuerySnapshot snapshot = await FirebaseFirestore.instance

          .collection('postings')

          .doc(posting!.id)

          .collection('bookings')

          .get();

      for (var doc in snapshot.docs) {

        List<dynamic> dates = doc['dates'];

        dates.forEach((date) {

          bookedDates.add(DateTime.parse(date));

        });

      }

      \_buildCalendarWidgets();

    } catch (error) {

      print("Error fetching booked dates: $error");

    }

  }

  @override

  void initState() {

    super.initState();

    posting = widget.posting;

    \_localBookedDates();

  }

  void \_showBookingDetailsDialog() {

    final TextEditingController idController = TextEditingController();

    final TextEditingController nameController = TextEditingController();

    final TextEditingController phoneController = TextEditingController();

    final TextEditingController emailController = TextEditingController();

    final TextEditingController noteController = TextEditingController();

    String checkInDate = selectedDates.isNotEmpty

        ? selectedDates.first.toString().split(' ')[0]

        : "Not selected";

    String checkOutDate = selectedDates.length > 1

        ? selectedDates.last.toString().split(' ')[0]

        : "Not selected";

    // Tính tổng số ngày đã chọn

    int totalDays = selectedDates.length;

    // Tính tổng giá tiền

    double totalPrice = totalDays \* (posting?.price ?? 0.0);

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: Text(

            "Enter Personal Information",

            style: TextStyle(

              fontSize: 20,

              fontWeight: FontWeight.bold,

              color: Colors.black,

            ),

          ),

          content: SingleChildScrollView(

            child: Container(

              margin: const EdgeInsets.symmetric(vertical: 10.0),

              child: Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                children: [

                  \_buildTextField(

                    controller: idController,

                    hintText: "Identification Code",

                  ),

                  const SizedBox(height: 15),

                  \_buildTextField(

                    controller: nameController,

                    hintText: "Full Name",

                  ),

                  const SizedBox(height: 15),

                  \_buildTextField(

                    controller: phoneController,

                    hintText: "Phone Number",

                    keyboardType: TextInputType.phone,

                  ),

                  const SizedBox(height: 15),

                  \_buildTextField(

                    controller: emailController,

                    hintText: "Email",

                    keyboardType: TextInputType.emailAddress,

                  ),

                  const SizedBox(height: 15),

                  \_buildTextField(

                    readOnly: true,

                    hintText: "Check-in Date",

                    labelText: checkInDate,

                  ),

                  const SizedBox(height: 15),

                  \_buildTextField(

                    readOnly: true,

                    hintText: "Check-out Date",

                    labelText: checkOutDate,

                  ),

                  const SizedBox(height: 15),

                  \_buildTextField(

                    controller: noteController,

                    hintText: "Notes (optional)",

                  ),

                  const SizedBox(height: 15),

                  // Hiển thị tổng giá tiền

                  Row(

                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

                    children: [

                      Text(

                        "Total:",

                        style: TextStyle(

                          fontSize: 18,

                          fontWeight: FontWeight.bold,

                          color: Colors.black,

                        ),

                      ),

                      Text(

                        "\ ${(totalPrice % 1 == 0 ? totalPrice.toInt() : totalPrice).toString()}",

                        style: TextStyle(

                          fontSize: 18,

                          fontWeight: FontWeight.bold,

                          color: Colors.green,

                        ),

                      ),

                    ],

                  ),

                ],

              ),

            ),

          ),

          actions: <Widget>[

            TextButton(

              child: Text(

                "Close",

                style: TextStyle(color: Colors.red),

              ),

              onPressed: () {

                Navigator.of(context).pop();

              },

            ),

            ElevatedButton(

              style: ElevatedButton.styleFrom(

                backgroundColor: Colors.green,

              ),

              child: Text("Confirm"),

              onPressed: () async {

                if (idController.text.isEmpty ||

                    nameController.text.isEmpty ||

                    phoneController.text.isEmpty ||

                    emailController.text.isEmpty ||

                    selectedDates.isEmpty) {

                  Get.snackbar(

                    "Error",

                    "Please fill in all required fields.",

                    backgroundColor: Colors.white,

                    colorText: Colors.red,

                  );

                } else {

                  try {

                    // Ép kiểu userID thành String

                    String userID = AppConstants.currentUser.id ?? '';

                    // Tạo dữ liệu đặt phòng

                    Map<String, dynamic> bookingData = {

                      'dates': selectedDates

                          .map((date) => date.toIso8601String())

                          .toList(),

                      'name': nameController.text,

                      'userID': userID,

                      'postingID': widget.posting!.id,

                      'phone': phoneController.text,

                      'email': emailController.text,

                      'note': noteController.text,

                      'createdAt': DateTime.now().toIso8601String(),

                      'totalPrice':

                          totalPrice, // Lưu tổng giá tiền vào cơ sở dữ liệu

                    };

                    // Lưu thông tin đặt phòng vào bảng bookings

                    await FirebaseFirestore.instance

                        .collection('bookings')

                        .add(bookingData);

                    // Hiển thị thông báo thành công

                    Get.snackbar(

                      "Success",

                      "Your booking has been confirmed!",

                      backgroundColor: Colors.white,

                      colorText: Colors.black,

                    );

                    // Đóng dialog

                    Navigator.of(context).pop();

                  } catch (e) {

                    // Hiển thị thông báo lỗi

                    Get.snackbar(

                      "Error",

                      "Failed to save booking. Please try again.",

                      backgroundColor: Colors.red,

                      colorText: Colors.white,

                    );

                  }

                }

              },

            ),

          ],

        );

      },

    );

  }

  Widget \_buildTextField({

    TextEditingController? controller,

    String? hintText,

    String? labelText,

    TextInputType keyboardType = TextInputType.text,

    bool readOnly = false,

  }) {

    return Container(

      padding: const EdgeInsets.symmetric(horizontal: 15.0),

      decoration: BoxDecoration(

        color: const Color(0xFFececf8),

        borderRadius: BorderRadius.circular(10),

      ),

      child: TextField(

        controller: controller,

        keyboardType: keyboardType,

        readOnly: readOnly,

        decoration: InputDecoration(

          border: InputBorder.none,

          hintText: hintText,

          labelText: labelText,

          hintStyle: TextStyle(color: Colors.grey),

        ),

      ),

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        flexibleSpace: Container(

          decoration: const BoxDecoration(

            color: Colors.green,

          ),

        ),

        title: Text(

          "Book ${posting!.name}",

          style: const TextStyle(

            color: Colors.white,

            fontSize: 21,

          ),

        ),

      ),

      body: Padding(

        padding: const EdgeInsets.fromLTRB(25, 25, 25, 0),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.spaceEvenly,

          children: [

            const Row(

              mainAxisSize: MainAxisSize.max,

              mainAxisAlignment: MainAxisAlignment.spaceAround,

              children: <Widget>[

                Text('Sun'),

                Text('Mon'),

                Text('Tues'),

                Text('Wed'),

                Text('Thurs'),

                Text('Fri'),

                Text('Sat'),

              ],

            ),

            SizedBox(

              height: MediaQuery.of(context).size.height / 2,

              child: (calendarWidgets.isEmpty)

                  ? Container()

                  : PageView.builder(

                      itemCount: calendarWidgets.length,

                      itemBuilder: (context, index) {

                        return calendarWidgets[index];

                      },

                    ),

            ),

            selectedDates.isNotEmpty

                ? MaterialButton(

                    onPressed: \_showBookingDetailsDialog,

                    minWidth: double.infinity,

                    height: MediaQuery.of(context).size.height / 14,

                    color: Colors.green,

                    child: const Text(

                      'Book Now',

                      style: TextStyle(color: Colors.white),

                    ),

                  )

                : Container(),

          ],

        ),

      ),

    );

  }

}

* File “chatbot\_srceen.dart”:

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

import 'package:flutter/material.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:intl/intl.dart';

class ChatbotScreen extends StatefulWidget {

  const ChatbotScreen({super.key});

  @override

  State<ChatbotScreen> createState() => \_ChatbotScreenState();

}

class \_ChatbotScreenState extends State<ChatbotScreen> {

  final TextEditingController \_controller = TextEditingController();

  final FirebaseFirestore \_firestore = FirebaseFirestore.instance;

  final ScrollController \_scrollController = ScrollController();

  Future<void> sendMessage(String message) async {

    if (message.trim().isEmpty) return;

    String userID = FirebaseAuth.instance.currentUser?.uid ?? '';

    if (userID.isEmpty) return; // Nếu chưa đăng nhập, không gửi tin nhắn

    // 🔹 Lưu tin nhắn vào subcollection 'messages' của user

    CollectionReference messagesRef = \_firestore

        .collection('chatbot\_messages')

        .doc(userID)

        .collection('messages');

    await messagesRef.add({

      'role': 'user',

      'text': message,

      'timestamp': FieldValue.serverTimestamp(),

    });

    setState(() {

      \_controller.clear();

    });

    \_scrollToBottom();

  }

  void \_scrollToBottom() {

    Future.delayed(const Duration(milliseconds: 300), () {

      \_scrollController.animateTo(

        0.0,

        duration: const Duration(milliseconds: 300),

        curve: Curves.easeOut,

      );

    });

  }

  @override

  Widget build(BuildContext context) {

    String userID = FirebaseAuth.instance.currentUser?.uid ?? '';

    return Scaffold(

      body: Column(

        children: [

          Expanded(

            child: StreamBuilder<QuerySnapshot>(

              // 🔹 Lấy tất cả tin nhắn của user từ subcollection "messages"

              stream: \_firestore

                  .collection('chatbot\_messages')

                  .doc(userID)

                  .collection('messages')

                  .orderBy('timestamp',

                      descending: false) // Sắp xếp theo thời gian gửi

                  .snapshots(),

              builder: (context, snapshot) {

                if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {

                  return const Center(child: Text("Chưa có tin nhắn nào"));

                }

                var messages = snapshot.data!.docs;

                return ListView.builder(

                  controller: \_scrollController,

                  itemCount: messages.length,

                  itemBuilder: (context, index) {

                    var messageData =

                        messages[index].data() as Map<String, dynamic>;

                    return Align(

                      alignment: messageData['role'] == 'user'

                          ? Alignment.centerRight

                          : Alignment.centerLeft,

                      child: Container(

                        margin: const EdgeInsets.symmetric(

                            vertical: 5, horizontal: 10),

                        padding: const EdgeInsets.all(10),

                        decoration: BoxDecoration(

                          color: messageData['role'] == 'user'

                              ? Colors.blue[200]

                              : Colors.green[200],

                          borderRadius: BorderRadius.circular(10),

                        ),

                        child: Column(

                          crossAxisAlignment: CrossAxisAlignment.start,

                          children: [

                            Text(

                              messageData['text'],

                              style: const TextStyle(fontSize: 16),

                            ),

                            const SizedBox(height: 5),

                            Text(

                              DateFormat('HH:mm dd/MM/yyyy').format(

                                (messageData['timestamp'] as Timestamp?)

                                        ?.toDate() ??

                                    DateTime.now(),

                              ),

                              style: TextStyle(

                                  fontSize: 12, color: Colors.grey[700]),

                            ),

                          ],

                        ),

                      ),

                    );

                  },

                );

              },

            ),

          ),

          Padding(

            padding: const EdgeInsets.all(8.0),

            child: Row(

              children: [

                Expanded(

                  child: TextField(

                    controller: \_controller,

                    decoration:

                        const InputDecoration(hintText: "Nhập tin nhắn..."),

                  ),

                ),

                IconButton(

                  icon: const Icon(Icons.send),

                  onPressed: () {

                    sendMessage(\_controller.text);

                  },

                ),

              ],

            ),

          ),

        ],

      ),

    );

  }

}

* File “explore\_screen.dart”:

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view/view\_posting\_screen.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

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

import 'package:flutter/material.dart';

import 'package:get/get.dart';

class ExploreScreen extends StatefulWidget {

  const ExploreScreen({super.key});

  @override

  State<ExploreScreen> createState() => \_ExploreScreenState();

}

class \_ExploreScreenState extends State<ExploreScreen> {

  TextEditingController controllerSearch = TextEditingController();

  Stream stream = FirebaseFirestore.instance.collection('postings').snapshots();

  String searchType = "";

  bool isNameButtonSelected = false;

  bool isCityButtonSelected = false;

  bool isPriceButtonSelected = false;

  Future<String?> getImageUrl(String? imageName, String? postingId) async {

    if (imageName == null || postingId == null) return null;

    try {

      Reference ref = FirebaseStorage.instance

          .ref()

          .child("postingImages")

          .child(postingId)

          .child(imageName);

      String url = await ref.getDownloadURL();

      return url;

    } catch (e) {

      print("Error getting image URL: $e");

      return null;

    }

  }

  void searchByField() {

    setState(() {

      if (searchType.isNotEmpty) {

        if (searchType == "price") {

          double? price = double.tryParse(controllerSearch.text);

          if (price != null) {

            stream = FirebaseFirestore.instance

                .collection('postings')

                .where(searchType, isEqualTo: price)

                .snapshots();

          }

        } else {

          stream = FirebaseFirestore.instance

              .collection('postings')

              .where(searchType, isEqualTo: controllerSearch.text)

              .snapshots();

        }

      }

    });

  }

  void pressSearchByButton(String searchTypeStr, bool isNameButtonSelectedB,

      bool isCityButtonSelectedB, bool isPriceButtonSelectedB) {

    setState(() {

      searchType = searchTypeStr;

      isNameButtonSelected = isNameButtonSelectedB;

      isCityButtonSelected = isCityButtonSelectedB;

      isPriceButtonSelected = isPriceButtonSelectedB;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Padding(

      padding: const EdgeInsets.fromLTRB(20, 30, 20, 10),

      child: SingleChildScrollView(

        child: Column(

          children: [

            Padding(

              padding: const EdgeInsets.only(top: 0, bottom: 0),

              child: TextField(

                decoration: const InputDecoration(

                    hintText: 'Search',

                    prefixIcon: Icon(Icons.search),

                    border: OutlineInputBorder(

                      borderSide: BorderSide(

                        color: Colors.grey,

                        width: 2.0,

                      ),

                    ),

                    contentPadding: EdgeInsets.all(5.0)),

                style: const TextStyle(

                  fontSize: 20.0,

                  color: Colors.black,

                ),

                controller: controllerSearch,

                onEditingComplete: searchByField,

              ),

            ),

            SizedBox(

              height: 70,

              width: MediaQuery.of(context).size.width,

              child: ListView(

                scrollDirection: Axis.horizontal,

                padding: const EdgeInsets.all(20),

                children: [

                  MaterialButton(

                    onPressed: () {

                      pressSearchByButton("name", true, false, false);

                    },

                    shape: RoundedRectangleBorder(

                        borderRadius: BorderRadius.circular(20)),

                    color: isNameButtonSelected ? Colors.green : Colors.white,

                    child: const Text("Name"),

                  ),

                  const SizedBox(width: 10),

                  MaterialButton(

                    onPressed: () {

                      pressSearchByButton("city", false, true, false);

                    },

                    shape: RoundedRectangleBorder(

                        borderRadius: BorderRadius.circular(20)),

                    color: isCityButtonSelected ? Colors.green : Colors.white,

                    child: const Text("City"),

                  ),

                  const SizedBox(width: 10),

                  MaterialButton(

                    onPressed: () {

                      pressSearchByButton("price", false, false, true);

                    },

                    shape: RoundedRectangleBorder(

                        borderRadius: BorderRadius.circular(20)),

                    color: isPriceButtonSelected ? Colors.green : Colors.white,

                    child: const Text("Price"),

                  ),

                  const SizedBox(width: 10),

                  MaterialButton(

                    onPressed: () {

                      controllerSearch.clear();

                      pressSearchByButton("", false, false, false);

                    },

                    shape: RoundedRectangleBorder(

                        borderRadius: BorderRadius.circular(20)),

                    color: Colors.white,

                    child: const Text("Clear"),

                  ),

                ],

              ),

            ),

            StreamBuilder(

                stream: stream,

                builder: (context, dataSnapshots) {

                  if (dataSnapshots.hasData) {

                    return GridView.builder(

                      physics: const ScrollPhysics(),

                      shrinkWrap: true,

                      itemCount: dataSnapshots.data.docs.length,

                      gridDelegate:

                          const SliverGridDelegateWithFixedCrossAxisCount(

                        crossAxisCount: 2,

                        crossAxisSpacing: 12,

                        mainAxisSpacing: 15,

                        childAspectRatio: 3 / 4,

                      ),

                      itemBuilder: (context, index) {

                        DocumentSnapshot snapshot =

                            dataSnapshots.data.docs[index];

                        PostingModel cPosting = PostingModel(id: snapshot.id);

                        cPosting.getPostingInfoFromSnapshot(snapshot);

                        return FutureBuilder<String?>(

                          future: getImageUrl(

                              cPosting.imageNames?.first, cPosting.id),

                          builder: (context, snapshot) {

                            if (snapshot.connectionState ==

                                ConnectionState.done) {

                              return InkResponse(

                                onTap: () {

                                  Get.to(ViewPostingScreen(

                                    posting: cPosting,

                                  ));

                                },

                                enableFeedback: true,

                                child: Column(

                                  crossAxisAlignment: CrossAxisAlignment.start,

                                  children: [

                                    snapshot.hasData &&

                                            snapshot.data!.isNotEmpty

                                        ? Image.network(

                                            snapshot.data!,

                                            fit: BoxFit.cover,

                                            height: 100,

                                            width: double.infinity,

                                          )

                                        : const Text("No Image"),

                                    Text(

                                      cPosting.name ?? "Unknown",

                                      style: const TextStyle(

                                        fontWeight: FontWeight.bold,

                                      ),

                                    ),

                                    Text(

                                      "${cPosting.price ?? 0} VNĐ/night",

                                    ),

                                    Row(

                                      children: List.generate(5, (starIndex) {

                                        return Icon(

                                          starIndex < (cPosting.rating ?? 0)

                                              ? Icons.star

                                              : Icons.star\_border,

                                          color: Colors.green,

                                        );

                                      }),

                                    ),

                                  ],

                                ),

                              );

                            } else {

                              return const CircularProgressIndicator();

                            }

                          },

                        );

                      },

                    );

                  } else {

                    return const Center(

                      child: CircularProgressIndicator(),

                    );

                  }

                }),

          ],

        ),

      ),

    );

  }

}

* File “host\_home\_screen.dart”:

import 'package:guest\_hotel/view/guestScreens/account\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/explore\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/statistics\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/saved\_listings\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/trips\_screen.dart';

import 'package:guest\_hotel/view/hostScreens/bookings\_screen.dart';

import 'package:guest\_hotel/view/hostScreens/my\_postings\_screen.dart';

import 'package:flutter/material.dart';

class HostHomeScreen extends StatefulWidget {

  int? index;

  HostHomeScreen({super.key, this.index});

  @override

  State<HostHomeScreen> createState() => \_HostHomeScreenState();

}

class \_HostHomeScreenState extends State<HostHomeScreen> {

  int selectedIndex = 0;

  final List<String> screenTitles = [

    'Bookings',

    'My Postings',

    'Statistics',

    'Profile',

  ];

  final List<Widget> screens = [

    const BookingsScreen(),

    const MyPostingsScreen(),

    const StatisticScreen(),

    const AccountScreen(),

  ];

  BottomNavigationBarItem customNavigationBarItem(

      int index, IconData iconData, String title) {

    return BottomNavigationBarItem(

      icon: Icon(

        iconData,

        color: Colors.black,

      ),

      activeIcon: Icon(

        iconData,

        color: Colors.green,

      ),

      label: title,

    );

  }

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    selectedIndex = widget.index ?? 3;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        flexibleSpace: Container(

          decoration: const BoxDecoration(

            color: Colors.green,

          ),

        ),

        title: Text(

          screenTitles[selectedIndex],

          style: const TextStyle(

            fontSize: 18,

            color: Colors.white,

          ),

        ),

        automaticallyImplyLeading: false,

        centerTitle: true,

      ),

      body: screens[selectedIndex],

      bottomNavigationBar: BottomNavigationBar(

        onTap: (i) {

          setState(() {

            selectedIndex = i;

          });

        },

        currentIndex: selectedIndex,

        type: BottomNavigationBarType.fixed,

        items: <BottomNavigationBarItem>[

          customNavigationBarItem(

              0, Icons.calendar\_month\_outlined, screenTitles[0]),

          customNavigationBarItem(1, Icons.home\_outlined, screenTitles[1]),

          customNavigationBarItem(2, Icons.stacked\_bar\_chart, screenTitles[2]),

          customNavigationBarItem(3, Icons.person\_outlined, screenTitles[3]),

        ],

      ),

    );

  }

}

* File “saved\_listings\_screen.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view/view\_posting\_screen.dart';

import 'package:guest\_hotel/view/widgets/posting\_grid\_tile\_ui.dart';

import 'package:flutter/material.dart';

import 'package:get/get.dart';

class SavedListingsScreen extends StatefulWidget {

  const SavedListingsScreen({super.key});

  @override

  State<SavedListingsScreen> createState() => \_SavedListingsScreenState();

}

class \_SavedListingsScreenState extends State<SavedListingsScreen> {

  @override

  Widget build(BuildContext context) {

    return Padding(

      padding: const EdgeInsets.fromLTRB(25, 15, 25, 0),

      child: GridView.builder(

        physics: const ScrollPhysics(),

        shrinkWrap: true,

        itemCount: AppConstants.currentUser.savedPostings?.length ?? 0,

        gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

          crossAxisCount: 2,

          crossAxisSpacing: 15,

          mainAxisSpacing: 15,

          childAspectRatio: 3 / 4,

        ),

        itemBuilder: (context, index) {

          PostingModel currentPosting =

              AppConstants.currentUser.savedPostings![index];

          return Stack(

            children: [

              InkResponse(

                enableFeedback: true,

                child: PostingGridTileUi(posting: currentPosting),

                onTap: () {

                  Get.to(ViewPostingScreen(

                    posting: currentPosting,

                  )); // Add your onTap logic here

                },

              ),

              Align(

                alignment: Alignment.topRight,

                child: Padding(

                  padding: const EdgeInsets.only(right: 10.0),

                  child: Container(

                    width: 30,

                    decoration: const BoxDecoration(

                      shape: BoxShape.circle,

                      color: Colors.white,

                    ),

                    child: IconButton(

                      onPressed: () {

                        AppConstants.currentUser

                            .removeSavedPosting(currentPosting);

                        setState(() {});

                      },

                      padding: const EdgeInsets.all(0),

                      icon: const Icon(

                        Icons.clear,

                        color: Colors.black,

                      ),

                    ),

                  ),

                ),

              ),

            ],

          );

        },

      ),

    );

  }

}

* File “statistics\_screen.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/booking\_model.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:flutter/material.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

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

import 'package:intl/intl.dart';

import 'package:percent\_indicator/circular\_percent\_indicator.dart';

class StatisticScreen extends StatefulWidget {

  const StatisticScreen({super.key});

  @override

  State<StatisticScreen> createState() => \_StatisticScreenState();

}

class \_StatisticScreenState extends State<StatisticScreen> {

  int postingCount = 0;

  int bookingCount = 0;

  List<BookingModel> allBookings =

      []; // Thêm danh sách để lưu thông tin đặt phòng

  @override

  void initState() {

    super.initState();

    \_fetchData();

  }

  Future<void> \_fetchData() async {

    var userModel = AppConstants.currentUser;

    // Lấy danh sách bài đăng

    List<PostingModel> postings = await userModel.getPostings();

    List<BookingModel> bookings = await userModel.getBookings();

    // Lấy danh sách đặt phòng cho từng bài đăng

    for (var posting in postings) {

      QuerySnapshot snapshot = await FirebaseFirestore.instance

          .collection('postings')

          .doc(posting.id)

          .collection('bookings')

          .get();

      for (var doc in snapshot.docs) {

        BookingModel booking = BookingModel();

        await booking.getBookingInfoFromFirestoreFromPosting(posting, doc);

        bookings.add(booking);

      }

    }

    setState(() {

      postingCount = postings.length;

      bookingCount = bookings.length;

      allBookings = bookings; // Lưu thông tin đặt phòng vào danh sách

    });

  }

  Future<List<Map<String, dynamic>>> \_fetchUserBookings() async {

    String currentUserID = AppConstants.currentUser.id ?? '';

    // Truy vấn danh sách bài đăng của người dùng hiện tại

    QuerySnapshot postingSnapshot = await FirebaseFirestore.instance

        .collection('postings')

        .where('hostID', isEqualTo: currentUserID)

        .get();

    List<String> postingIDs =

        postingSnapshot.docs.map((doc) => doc.id).toList();

    // Truy vấn danh sách đặt phòng liên quan đến các bài đăng

    QuerySnapshot bookingSnapshot = await FirebaseFirestore.instance

        .collection('bookings')

        .where('postingID', whereIn: postingIDs)

        .get();

    List<Map<String, dynamic>> bookings = bookingSnapshot.docs

        .map((doc) => doc.data() as Map<String, dynamic>)

        .toList();

    // Lấy thông tin chi tiết từng bài đăng

    for (var booking in bookings) {

      DocumentSnapshot postingSnapshot = await FirebaseFirestore.instance

          .collection('postings')

          .doc(booking['postingID'])

          .get();

      booking['posting'] = postingSnapshot.data();

    }

    return bookings;

  }

  Future<String> \_getImageUrl(String postingID, String imageName) async {

    Reference ref = FirebaseStorage.instance

        .ref()

        .child('postingImages')

        .child(postingID)

        .child(imageName);

    return await ref.getDownloadURL();

  }

  String \_formatDate(dynamic date) {

    if (date is Timestamp) {

      DateTime dateTime = date.toDate();

      return DateFormat('dd/MM/yyyy').format(dateTime);

    } else if (date is String) {

      try {

        DateTime dateTime = DateTime.parse(date);

        return DateFormat('dd/MM/yyyy').format(dateTime);

      } catch (e) {

        return date;

      }

    } else {

      return 'Invalid date';

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: SingleChildScrollView(

        child: Padding(

          padding: const EdgeInsets.all(16.0),

          child: Column(

            crossAxisAlignment: CrossAxisAlignment.stretch,

            children: [

              // Phần hiển thị thống kê tổng quan (Postings, Bookings)

              Row(

                mainAxisAlignment: MainAxisAlignment.spaceBetween,

                children: [

                  \_buildInfoCard(

                      postingCount.toString(), "Postings", Colors.purple),

                  \_buildInfoCard(

                      bookingCount.toString(), "Bookings", Colors.blue),

                ],

              ),

              const SizedBox(height: 20),

              Row(

                mainAxisAlignment: MainAxisAlignment.spaceBetween,

                children: [

                  \_buildCircularIndicator(

                      "Booking", bookingCount / 100.0, Colors.blue),

                  \_buildCircularIndicator(

                      "Posting", postingCount / 100.0, Colors.green),

                ],

              ),

              const SizedBox(height: 20),

              // Phần hiển thị danh sách thông tin đặt phòng

              const Text(

                "Booking Details",

                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

              ),

              const SizedBox(height: 10),

              FutureBuilder<List<Map<String, dynamic>>>(

                future: \_fetchUserBookings(),

                builder: (context, snapshot) {

                  if (snapshot.connectionState == ConnectionState.waiting) {

                    return const Center(child: CircularProgressIndicator());

                  } else if (snapshot.hasError) {

                    return const Center(

                        child: Text('Error loading statistics.'));

                  } else if (!snapshot.hasData || snapshot.data!.isEmpty) {

                    return const Center(child: Text('No bookings found.'));

                  } else {

                    List<Map<String, dynamic>> bookings = snapshot.data!;

                    return ListView.builder(

                      shrinkWrap: true, // Đảm bảo không bị lỗi kích thước

                      physics: const NeverScrollableScrollPhysics(),

                      itemCount: bookings.length,

                      itemBuilder: (context, index) {

                        var booking = bookings[index];

                        var posting = booking['posting'];

                        return FutureBuilder<String>(

                          future: \_getImageUrl(

                            booking['postingID'],

                            posting['imageNames'].first,

                          ),

                          builder: (context, imageSnapshot) {

                            if (imageSnapshot.connectionState ==

                                ConnectionState.waiting) {

                              return const Center(

                                  child: CircularProgressIndicator());

                            } else if (imageSnapshot.hasError) {

                              return const Center(

                                  child: Text('Error loading image.'));

                            } else {

                              return Card(

                                margin: const EdgeInsets.symmetric(vertical: 8),

                                child: Column(

                                  crossAxisAlignment: CrossAxisAlignment.start,

                                  children: [

                                    ClipRRect(

                                      borderRadius: BorderRadius.circular(10),

                                      child: Image.network(

                                        imageSnapshot.data!,

                                        width: double.infinity,

                                        height: 200,

                                        fit: BoxFit.cover,

                                      ),

                                    ),

                                    Padding(

                                      padding: const EdgeInsets.all(10.0),

                                      child: Column(

                                        crossAxisAlignment:

                                            CrossAxisAlignment.start,

                                        children: [

                                          Text(

                                            posting['name'] ?? 'No Name',

                                            style: const TextStyle(

                                              fontSize: 18,

                                              fontWeight: FontWeight.bold,

                                            ),

                                          ),

                                          const SizedBox(height: 5),

                                          Text(

                                              'Price: \ ${posting['price'] ?? 'N/A'}/night'),

                                          Text(

                                              'Booked by: ${booking['name'] ?? 'No Name'}'),

                                          Text(

                                              'Email: ${booking['email'] ?? 'No Email'}'),

                                          Text(

                                              'Phone: ${booking['phone'] ?? 'No Phone'}'),

                                          Text(

                                              'Check-in: ${\_formatDate(booking['dates'].first)}'),

                                          Text(

                                              'Check-out: ${\_formatDate(booking['dates'].last)}'),

                                        ],

                                      ),

                                    ),

                                  ],

                                ),

                              );

                            }

                          },

                        );

                      },

                    );

                  }

                },

              ),

            ],

          ),

        ),

      ),

    );

  }

  Widget \_buildInfoCard(String value, String label, Color color) {

    return Expanded(

      child: Container(

        margin: const EdgeInsets.all(8),

        padding: const EdgeInsets.all(16),

        decoration: BoxDecoration(

          gradient: LinearGradient(colors: [color.withOpacity(0.7), color]),

          borderRadius: BorderRadius.circular(16),

          boxShadow: [

            BoxShadow(

              color: color.withOpacity(0.4),

              blurRadius: 10,

              spreadRadius: 5,

              offset: const Offset(0, 5),

            ),

          ],

        ),

        child: Column(

          children: [

            const Icon(Icons.insert\_chart, color: Colors.white, size: 32),

            const SizedBox(height: 8),

            Text(

              value,

              style: const TextStyle(

                fontSize: 24,

                color: Colors.white,

                fontWeight: FontWeight.bold,

              ),

            ),

            const SizedBox(height: 4),

            Text(label,

                style: const TextStyle(color: Colors.white70, fontSize: 16)),

          ],

        ),

      ),

    );

  }

  Widget \_buildCircularIndicator(String label, double percentage, Color color) {

    return Expanded(

      child: Column(

        children: [

          CircularPercentIndicator(

            radius: 50.0,

            lineWidth: 8.0,

            percent: percentage,

            center: Text(

              "${(percentage \* 100).round()}%",

              style: const TextStyle(fontWeight: FontWeight.bold, fontSize: 16),

            ),

            progressColor: color,

            backgroundColor: Colors.grey[300]!,

          ),

          const SizedBox(height: 8),

          Text(label,

              style: const TextStyle(color: Colors.black54, fontSize: 14)),

        ],

      ),

    );

  }

}

* File “trips\_screen.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:flutter/material.dart';

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

import 'package:intl/intl.dart'; // Import package intl

class TripsScreen extends StatefulWidget {

  const TripsScreen({super.key});

  @override

  State<TripsScreen> createState() => \_TripsScreenState();

}

class \_TripsScreenState extends State<TripsScreen> {

  Future<List<Map<String, dynamic>>> \_fetchUserBookings() async {

    String currentUserID = AppConstants.currentUser.id ?? '';

    QuerySnapshot bookingSnapshot = await FirebaseFirestore.instance

        .collection('bookings')

        .where('userID', isEqualTo: currentUserID)

        .get();

    List<Map<String, dynamic>> bookings = bookingSnapshot.docs

        .map((doc) => doc.data() as Map<String, dynamic>)

        .toList();

    for (var booking in bookings) {

      DocumentSnapshot postingSnapshot = await FirebaseFirestore.instance

          .collection('postings')

          .doc(booking['postingID'])

          .get();

      booking['posting'] = postingSnapshot.data();

    }

    return bookings;

  }

  Future<String> \_getImageUrl(String postingID, String imageName) async {

    Reference ref = FirebaseStorage.instance

        .ref()

        .child('postingImages')

        .child(postingID)

        .child(imageName);

    return await ref.getDownloadURL();

  }

  String \_formatDate(dynamic date) {

    if (date is Timestamp) {

      DateTime dateTime = date.toDate();

      return DateFormat('dd/MM/yyyy').format(dateTime);

    } else if (date is String) {

      try {

        DateTime dateTime = DateTime.parse(date);

        return DateFormat('dd/MM/yyyy').format(dateTime);

      } catch (e) {

        return date;

      }

    } else {

      return 'Invalid date';

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: FutureBuilder<List<Map<String, dynamic>>>(

        future: \_fetchUserBookings(),

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return const Center(child: CircularProgressIndicator());

          } else if (snapshot.hasError) {

            return const Center(child: Text('Error loading bookings.'));

          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {

            return const Center(child: Text('No bookings found.'));

          } else {

            List<Map<String, dynamic>> bookings = snapshot.data!;

            return ListView.builder(

              itemCount: bookings.length,

              itemBuilder: (context, index) {

                var booking = bookings[index];

                var posting = booking['posting'];

                return FutureBuilder<String>(

                  future: \_getImageUrl(

                    booking['postingID'],

                    posting['imageNames'].first,

                  ),

                  builder: (context, imageSnapshot) {

                    if (imageSnapshot.connectionState ==

                        ConnectionState.waiting) {

                      return const Center(child: CircularProgressIndicator());

                    } else if (imageSnapshot.hasError) {

                      return const Center(child: Text('Error loading image.'));

                    } else {

                      return Card(

                        margin: const EdgeInsets.all(10),

                        child: Column(

                          crossAxisAlignment: CrossAxisAlignment.start,

                          children: [

                            // Hình ảnh lớn hơn

                            ClipRRect(

                              borderRadius: BorderRadius.circular(10),

                              child: Image.network(

                                imageSnapshot.data!,

                                width: double.infinity,

                                height: 200,

                                fit: BoxFit.cover,

                              ),

                            ),

                            Padding(

                              padding: const EdgeInsets.all(10.0),

                              child: Column(

                                crossAxisAlignment: CrossAxisAlignment.start,

                                children: [

                                  Text(

                                    posting['name'] ?? 'No Name',

                                    style: const TextStyle(

                                      fontSize: 18,

                                      fontWeight: FontWeight.bold,

                                    ),

                                  ),

                                  const SizedBox(height: 5),

                                  Text(

                                      'Price: \ ${posting['price'] ?? 'N/A'}/night'),

                                  Text(

                                      'Booked by: ${booking['name'] ?? 'No Name'}'),

                                  Text(

                                      'Email: ${booking['email'] ?? 'No Email'}'),

                                  Text(

                                      'Phone: ${booking['phone'] ?? 'No Phone'}'),

                                  Text(

                                      'Check-in: ${\_formatDate(booking['dates'].first)}'),

                                  Text(

                                      'Check-out: ${\_formatDate(booking['dates'].last)}'),

                                ],

                              ),

                            ),

                          ],

                        ),

                      );

                    }

                  },

                );

              },

            );

          }

        },

      ),

    );

  }

}

* File “user\_profile\_screen.dart”:

import 'package:flutter/material.dart';

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

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

import 'package:get/get.dart';

import 'package:image\_picker/image\_picker.dart';

class UserProfileScreen extends StatefulWidget {

  @override

  \_UserProfileScreenState createState() => \_UserProfileScreenState();

}

class \_UserProfileScreenState extends State<UserProfileScreen> {

  final TextEditingController firstNameController = TextEditingController();

  final TextEditingController lastNameController = TextEditingController();

  final TextEditingController emailController = TextEditingController();

  final TextEditingController cityController = TextEditingController();

  final TextEditingController countryController = TextEditingController();

  final TextEditingController bioController = TextEditingController();

  ImageProvider displayImage = AssetImage('assets/default\_avatar.png');

  @override

  void initState() {

    super.initState();

    \_loadUserData();

  }

  Future<void> \_loadUserData() async {

    setState(() {

      firstNameController.text = AppConstants.currentUser.firstName ?? '';

      lastNameController.text = AppConstants.currentUser.lastName ?? '';

      emailController.text = AppConstants.currentUser.email ?? '';

      cityController.text = AppConstants.currentUser.city ?? '';

      countryController.text = AppConstants.currentUser.country ?? '';

      bioController.text = AppConstants.currentUser.bio ?? '';

      displayImage = AppConstants.currentUser.displayImage ?? displayImage;

    });

  }

  Future<void> \_updateUserData() async {

    String currentUserID = AppConstants.currentUser.id!;

    await FirebaseFirestore.instance

        .collection('users')

        .doc(currentUserID)

        .update({

      'firstName': firstNameController.text,

      'lastName': lastNameController.text,

      'email': emailController.text,

      'city': cityController.text,

      'country': countryController.text,

      'bio': bioController.text,

    });

    AppConstants.currentUser.firstName = firstNameController.text;

    AppConstants.currentUser.lastName = lastNameController.text;

    AppConstants.currentUser.email = emailController.text;

    AppConstants.currentUser.city = cityController.text;

    AppConstants.currentUser.country = countryController.text;

    AppConstants.currentUser.bio = bioController.text;

    ScaffoldMessenger.of(context).showSnackBar(

      SnackBar(content: Text('Profile updated successfully')),

    );

  }

  Future<void> \_pickImage() async {

    final ImagePicker \_picker = ImagePicker();

    final XFile? image = await \_picker.pickImage(source: ImageSource.gallery);

    if (image == null) return;

    final bytes = await image.readAsBytes();

    String currentUserID = AppConstants.currentUser.id!;

    Reference storageReference = FirebaseStorage.instance

        .ref()

        .child("userImages")

        .child(currentUserID)

        .child(currentUserID + ".png");

    await storageReference.putData(bytes);

    setState(() {

      displayImage = MemoryImage(bytes);

      AppConstants.currentUser.displayImage = displayImage as MemoryImage?;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('User Profile'),

        backgroundColor: Colors.green,

      ),

      body: SingleChildScrollView(

        child: Padding(

          padding: const EdgeInsets.all(16.0),

          child: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            children: [

              GestureDetector(

                onTap: \_pickImage,

                child: Center(

                  child: CircleAvatar(

                    backgroundImage: displayImage,

                    radius: 60,

                  ),

                ),

              ),

              const SizedBox(height: 20),

              \_buildTextField(

                controller: firstNameController,

                hintText: 'First Name',

              ),

              const SizedBox(height: 15),

              \_buildTextField(

                controller: lastNameController,

                hintText: 'Last Name',

              ),

              const SizedBox(height: 15),

              \_buildTextField(

                controller: emailController,

                hintText: 'Email',

                readOnly: true,

              ),

              const SizedBox(height: 15),

              \_buildTextField(

                controller: cityController,

                hintText: 'City',

              ),

              const SizedBox(height: 15),

              \_buildTextField(

                controller: countryController,

                hintText: 'Country',

              ),

              const SizedBox(height: 15),

              \_buildTextField(

                controller: bioController,

                hintText: 'Bio',

              ),

              const SizedBox(height: 20),

              Center(

                child: ElevatedButton(

                  onPressed: () {

                    \_updateUserData();

                    Get.snackbar(

                      "Success",

                      "Your personal information has been updated.",

                    );

                  },

                  child: const Text(

                    "Update",

                    style: TextStyle(

                      fontWeight: FontWeight.bold,

                      fontSize: 20,

                      color: Colors.green,

                    ),

                  ),

                ),

              ),

            ],

          ),

        ),

      ),

    );

  }

  Widget \_buildTextField({

    required TextEditingController controller,

    required String hintText,

    bool readOnly = false,

    TextInputType keyboardType = TextInputType.text,

  }) {

    return TextField(

      controller: controller,

      readOnly: readOnly,

      keyboardType: keyboardType,

      decoration: InputDecoration(

        hintText: hintText,

        border: OutlineInputBorder(),

      ),

    );

  }

}

+ Folder “hostScreens”:

* File “bookings\_screen.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view/widgets/calender\_ui.dart';

import 'package:guest\_hotel/view/widgets/posting\_list\_tile\_ui.dart';

import 'package:flutter/material.dart';

class BookingsScreen extends StatefulWidget {

  const BookingsScreen({super.key});

  @override

  State<BookingsScreen> createState() => \_BookingsScreenState();

}

class \_BookingsScreenState extends State<BookingsScreen> {

  List<DateTime> \_bookedDates = [];

  List<DateTime> \_allBookedDates = [];

  PostingModel? \_selectedPosting;

  List<DateTime> \_getSelectedDates() {

    return [];

  }

  \_selectDate(DateTime date) {}

  \_selectAPosting(PostingModel posting) {

    \_selectedPosting = posting;

    posting.getAllBookingsFromFirestore().whenComplete(() {

      \_bookedDates = posting.getAllBookedDates();

    });

    setState(() {});

  }

  \_clearSelectedPosting() {

    \_bookedDates = \_allBookedDates;

    \_selectedPosting = null;

    setState(() {});

  }

  void initState() {

    super.initState();

    \_bookedDates = AppConstants.currentUser.getAllBookedDates();

    \_allBookedDates = AppConstants.currentUser.getAllBookedDates();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: SingleChildScrollView(

        child: Padding(

          padding: const EdgeInsets.fromLTRB(25, 25, 25, 0),

          child: Column(

            mainAxisAlignment: MainAxisAlignment.spaceEvenly,

            children: [

              const Row(

                mainAxisSize: MainAxisSize.max,

                mainAxisAlignment: MainAxisAlignment.spaceAround,

                children: <Widget>[

                  Text('Sun'),

                  Text('Mon'),

                  Text('Tues'),

                  Text('Wed'),

                  Text('Thus'),

                  Text('Fri'),

                  Text('Sat'),

                ],

              ),

              //calender

              Padding(

                padding: const EdgeInsets.only(top: 15, bottom: 35),

                child: SizedBox(

                  height: MediaQuery.of(context).size.height / 1.8,

                  child: PageView.builder(

                      itemCount: 12,

                      itemBuilder: (context, index) {

                        return CalenderUI(

                          monthIndex: index,

                          bookedDates: \_bookedDates,

                          selectDate: \_selectDate,

                          getSelectedDates: \_getSelectedDates,

                        );

                      }),

                ),

              ),

              //reset

              Padding(

                padding: const EdgeInsets.fromLTRB(25, 25, 0, 0),

                child: Row(

                  mainAxisAlignment: MainAxisAlignment.spaceBetween,

                  children: [

                    const Text(

                      'Fiter by Listing',

                      style: TextStyle(

                        fontSize: 20,

                        fontWeight: FontWeight.bold,

                      ),

                    ),

                    MaterialButton(

                      onPressed: () {

                        \_clearSelectedPosting();

                      },

                      child: const Text(

                        'Reset',

                        style: TextStyle(

                          fontSize: 15,

                          fontWeight: FontWeight.bold,

                        ),

                      ),

                    ),

                    Padding(

                      padding: const EdgeInsets.only(top: 25, bottom: 25),

                      child: Container(),

                    ),

                  ],

                ),

              ),

              //display host listings

              ListView.builder(

                shrinkWrap: true,

                itemCount: AppConstants.currentUser.myPostings!.length,

                itemBuilder: (context, index) {

                  return Padding(

                    padding: const EdgeInsets.only(bottom: 26.0),

                    child: InkResponse(

                      onTap: () {

                        \_selectAPosting(

                            AppConstants.currentUser.myPostings![index]);

                      },

                      child: Container(

                        decoration: BoxDecoration(

                          border: Border.all(

                            color: \_selectedPosting ==

                                    AppConstants.currentUser.myPostings![index]

                                ? Colors.blue

                                : Colors.grey,

                            width: 1.5,

                          ),

                          borderRadius: BorderRadius.circular(6.0),

                        ),

                        child: PostingListTileUI(

                          posting: AppConstants.currentUser.myPostings![index],

                        ),

                      ),

                    ),

                  );

                },

              ),

            ],

          ),

        ),

      ),

    );

  }

}

* File “create\_posting\_screen.dart”:

import 'dart:io';

import 'package:guest\_hotel/global.dart';

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view/guestScreens/host\_home\_screen.dart';

import 'package:guest\_hotel/view/widgets/amenities\_ui.dart';

import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:image\_picker/image\_picker.dart';

class CreatePostingScreen extends StatefulWidget {

  PostingModel? posting;

  CreatePostingScreen({

    Key? key,

    this.posting,

  }) : super(key: key);

  @override

  State<CreatePostingScreen> createState() => \_CreatePostingScreenState();

}

class \_CreatePostingScreenState extends State<CreatePostingScreen> {

  final formKey = GlobalKey<FormState>();

  TextEditingController \_nameTextEditingController = TextEditingController();

  TextEditingController \_priceTextEditingController = TextEditingController();

  TextEditingController \_descriptionTextEditingController =

      TextEditingController();

  TextEditingController \_addressTextEditingController = TextEditingController();

  TextEditingController \_cityTextEditingController = TextEditingController();

  TextEditingController \_countryTextEditingController = TextEditingController();

  TextEditingController \_amenitiesTextEditingController =

      TextEditingController();

  final List<String> residenceTypes = [

    'Detatched House',

    'Villa',

    'Apartment',

    'Condo',

    'Flat',

    'Town House',

    'Studio',

  ];

  late String residenceTypeSelected;

  Map<String, int> \_beds = {'small': 0, 'medium': 0, 'large': 0};

  Map<String, int> \_bathrooms = {'full': 0, 'half': 0};

  List<MemoryImage> \_imagesList = [];

  Future<void> \_selectImageFromGallery(int index) async {

    var imageFilePickedFromGallery =

        await ImagePicker().pickImage(source: ImageSource.gallery);

    if (imageFilePickedFromGallery != null) {

      MemoryImage imageFileInBytesForm = MemoryImage(

          (File(imageFilePickedFromGallery.path)).readAsBytesSync());

      if (index < 0) {

        \_imagesList.add(imageFileInBytesForm);

      } else {

        \_imagesList[index] = imageFileInBytesForm;

      }

      setState(() {});

    }

  }

  void initializeValues() {

    if (widget.posting == null) {

      \_nameTextEditingController.text = "";

      \_priceTextEditingController.text = "";

      \_descriptionTextEditingController.text = "";

      \_addressTextEditingController.text = "";

      \_cityTextEditingController.text = "";

      \_countryTextEditingController.text = "";

      residenceTypeSelected = residenceTypes.first;

      \_beds = {'small': 0, 'medium': 0, 'large': 0};

      \_bathrooms = {'full': 0, 'half': 0};

      \_imagesList = [];

    } else {

      \_nameTextEditingController =

          TextEditingController(text: widget.posting!.name);

      \_priceTextEditingController =

          TextEditingController(text: widget.posting!.price.toString());

      \_descriptionTextEditingController =

          TextEditingController(text: widget.posting!.description);

      \_addressTextEditingController =

          TextEditingController(text: widget.posting!.address);

      \_cityTextEditingController =

          TextEditingController(text: widget.posting!.city);

      \_countryTextEditingController =

          TextEditingController(text: widget.posting!.country);

      \_amenitiesTextEditingController =

          TextEditingController(text: widget.posting!.getAmenititesString());

      \_beds = widget.posting!.beds!;

      \_bathrooms = widget.posting!.bathrooms!;

      \_imagesList = widget.posting!.displayImages!;

      residenceTypeSelected = residenceTypes.contains(widget.posting!.type!)

          ? widget.posting!.type!

          : residenceTypes.first;

    }

    setState(() {});

  }

  @override

  void initState() {

    super.initState();

    initializeValues();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        flexibleSpace: Container(

          decoration: const BoxDecoration(

            color: Colors.green,

          ),

        ),

        title: const Text(

          "Create/Update a Listing",

          style: TextStyle(

            color: Colors.white,

          ),

        ),

        actions: [

          IconButton(

            onPressed: () async {

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

                return;

              }

              if (\_imagesList.isEmpty) {

                return;

              }

              postingModel.name = \_nameTextEditingController.text;

              postingModel.price =

                  double.tryParse(\_priceTextEditingController.text) ?? 0.0;

              postingModel.description = \_descriptionTextEditingController.text;

              postingModel.address = \_addressTextEditingController.text;

              postingModel.city = \_cityTextEditingController.text;

              postingModel.country = \_countryTextEditingController.text;

              postingModel.amenities =

                  \_amenitiesTextEditingController.text.split(",");

              postingModel.beds = \_beds;

              postingModel.bathrooms = \_bathrooms;

              postingModel.displayImages = \_imagesList;

              postingModel.host =

                  AppConstants.currentUser.createUserFormContact();

              postingModel.setImagesName();

              if (widget.posting == null) {

                postingModel.rating = 3.5;

                postingModel.bookings = [];

                postingModel.reviews = [];

                await postingViewModel.addListingInfoToFirestore();

                await postingViewModel.addImagesToFirebaseStorage();

                Get.snackbar("New Listing",

                    "Your New Listing is uploaded successfully.");

              } else {

                postingModel.rating = widget.posting!.rating;

                postingModel.bookings = widget.posting!.bookings;

                postingModel.reviews = widget.posting!.reviews;

                postingModel.id = widget.posting!.id;

                for (int i = 0;

                    i < AppConstants.currentUser.myPostings!.length;

                    i++) {

                  if (AppConstants.currentUser.myPostings![i].id ==

                      postingModel.id) {

                    AppConstants.currentUser.myPostings![i] = postingModel;

                    break;

                  }

                }

                await postingViewModel.updatePostingInfoToFirestore();

                Get.snackbar(

                    "Update Listing", "Your Listing is updated successfully.");

              }

              postingModel = PostingModel();

              Get.to(HostHomeScreen());

            },

            icon: const Icon(Icons.upload\_outlined),

          )

        ],

      ),

      body: Center(

        child: SingleChildScrollView(

          child: Padding(

            padding: const EdgeInsets.fromLTRB(20, 20, 20, 10),

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.start,

              mainAxisAlignment: MainAxisAlignment.center,

              children: [

                Form(

                  key: formKey,

                  child: Column(

                    crossAxisAlignment: CrossAxisAlignment.start,

                    children: [

                      // Listing name

                      Padding(

                        padding: const EdgeInsets.only(top: 5),

                        child: TextFormField(

                          decoration:

                              const InputDecoration(labelText: "Listing name"),

                          style: const TextStyle(

                            fontSize: 25.0,

                          ),

                          controller: \_nameTextEditingController,

                          validator: (textInput) {

                            if (textInput == null || textInput.isEmpty) {

                              return "Please Enter Your Name";

                            }

                            return null;

                          },

                        ),

                      ),

                      //Select property type

                      Padding(

                        padding: const EdgeInsets.only(top: 28),

                        child: DropdownButton<String>(

                          items: residenceTypes.map((item) {

                            return DropdownMenuItem<String>(

                              value: item,

                              child: Text(

                                item,

                                style: const TextStyle(

                                  fontSize: 21,

                                ),

                              ),

                            );

                          }).toList(),

                          onChanged: (valueItem) {

                            if (valueItem != null &&

                                residenceTypes.contains(valueItem)) {

                              setState(() {

                                residenceTypeSelected = valueItem;

                              });

                            }

                          },

                          isExpanded: true,

                          value: residenceTypeSelected,

                          hint: const Text(

                            "Select property type",

                            style: TextStyle(

                              fontSize: 21,

                            ),

                          ),

                        ),

                      ),

                      //Price /night

                      Padding(

                        padding: const EdgeInsets.only(top: 21),

                        child: Row(

                          crossAxisAlignment: CrossAxisAlignment.end,

                          children: <Widget>[

                            Expanded(

                              child: TextFormField(

                                decoration:

                                    const InputDecoration(labelText: "Price"),

                                style: const TextStyle(

                                  fontSize: 25.0,

                                ),

                                keyboardType: TextInputType.number,

                                controller: \_priceTextEditingController,

                                validator: (text) {

                                  if (text == null || text.isEmpty) {

                                    return "Please Enter Price";

                                  }

                                  return null;

                                },

                              ),

                            ),

                            const Padding(

                              padding:

                                  EdgeInsets.only(left: 10.0, bottom: 10.0),

                              child: Text(

                                " / night",

                                style: TextStyle(

                                  fontSize: 18,

                                ),

                              ),

                            ),

                          ],

                        ),

                      ),

                      //Description

                      Padding(

                        padding: const EdgeInsets.only(top: 21.0),

                        child: TextFormField(

                          decoration:

                              const InputDecoration(labelText: "Description"),

                          style: const TextStyle(

                            fontSize: 25.0,

                          ),

                          controller: \_descriptionTextEditingController,

                          maxLines: 3,

                          minLines: 1,

                          validator: (text) {

                            if (text == null || text.isEmpty) {

                              return "Please Enter Description";

                            }

                            return null;

                          },

                        ),

                      ),

                      //Address

                      Padding(

                        padding: const EdgeInsets.all(21.0),

                        child: TextFormField(

                          decoration:

                              const InputDecoration(labelText: "Address"),

                          maxLines: 3,

                          style: const TextStyle(

                            fontSize: 25.0,

                          ),

                          controller: \_addressTextEditingController,

                          validator: (text) {

                            if (text == null || text.isEmpty) {

                              return "Please Enter Address";

                            }

                            return null;

                          },

                        ),

                      ),

                      //Beds

                      const Padding(

                        padding: EdgeInsets.only(top: 38.0),

                        child: Text(

                          'Beds',

                          style: TextStyle(

                            fontSize: 20.0,

                            fontWeight: FontWeight.bold,

                          ),

                        ),

                      ),

                      Padding(

                        padding:

                            const EdgeInsets.only(top: 20, left: 20, right: 20),

                        child: Column(

                          children: <Widget>[

                            AmenitiesUI(

                              type: 'Twin/Single',

                              startValue: \_beds['small'] ?? 0,

                              decreaseValue: () {

                                setState(() {

                                  \_beds['small'] = (\_beds['small'] ?? 0) - 1;

                                  if (\_beds['small']! < 0) \_beds['small'] = 0;

                                });

                              },

                              increaseValue: () {

                                setState(() {

                                  \_beds['small'] = (\_beds['small'] ?? 0) + 1;

                                });

                              },

                            ),

                            AmenitiesUI(

                              type: 'Double',

                              startValue: \_beds['medium'] ?? 0,

                              decreaseValue: () {

                                setState(() {

                                  \_beds['medium'] = (\_beds['medium'] ?? 0) - 1;

                                  if (\_beds['medium']! < 0) \_beds['medium'] = 0;

                                });

                              },

                              increaseValue: () {

                                setState(() {

                                  \_beds['medium'] = (\_beds['medium'] ?? 0) + 1;

                                });

                              },

                            ),

                            AmenitiesUI(

                              type: 'Queen/King',

                              startValue: \_beds['large'] ?? 0,

                              decreaseValue: () {

                                setState(() {

                                  \_beds['large'] = (\_beds['large'] ?? 0) - 1;

                                  if (\_beds['large']! < 0) \_beds['large'] = 0;

                                });

                              },

                              increaseValue: () {

                                setState(() {

                                  \_beds['large'] = (\_beds['large'] ?? 0) + 1;

                                });

                              },

                            ),

                          ],

                        ),

                      ),

                      // Bathrooms

                      const Padding(

                        padding: EdgeInsets.only(top: 20.0),

                        child: Text(

                          'Bathrooms',

                          style: TextStyle(

                            fontSize: 20.0,

                            fontWeight: FontWeight.bold,

                          ),

                        ),

                      ),

                      Padding(

                        padding: const EdgeInsets.fromLTRB(20, 30, 20, 5),

                        child: Column(

                          children: <Widget>[

                            AmenitiesUI(

                              type: 'Full',

                              startValue: \_bathrooms['full'] ?? 0,

                              decreaseValue: () {

                                setState(() {

                                  \_bathrooms['full'] =

                                      (\_bathrooms['full'] ?? 0) - 1;

                                  if (\_bathrooms['full']! < 0)

                                    \_bathrooms['full'] = 0;

                                });

                              },

                              increaseValue: () {

                                setState(() {

                                  \_bathrooms['full'] =

                                      (\_bathrooms['full'] ?? 0) + 1;

                                });

                              },

                            ),

                            AmenitiesUI(

                              type: 'Half',

                              startValue: \_bathrooms['half'] ?? 0,

                              decreaseValue: () {

                                setState(() {

                                  \_bathrooms['half'] =

                                      (\_bathrooms['half'] ?? 0) - 1;

                                  if (\_bathrooms['half']! < 0)

                                    \_bathrooms['half'] = 0;

                                });

                              },

                              increaseValue: () {

                                setState(() {

                                  \_bathrooms['half'] =

                                      (\_bathrooms['half'] ?? 0) + 1;

                                });

                              },

                            ),

                          ],

                        ),

                      ),

                      // Extra Amenities

                      Padding(

                        padding: const EdgeInsets.only(top: 21),

                        child: TextFormField(

                          decoration: const InputDecoration(

                              labelText: "Amenities (comma separated)"),

                          style: const TextStyle(

                            fontSize: 25.0,

                          ),

                          controller: \_amenitiesTextEditingController,

                          validator: (text) {

                            if (text == null || text.isEmpty) {

                              return "Please enter valid amenities (comma separated)";

                            }

                            return null;

                          },

                          maxLines: 3,

                          minLines: 1,

                        ),

                      ),

                      //Photo

                      Padding(

                        padding: const EdgeInsets.only(top: 21, bottom: 21),

                        child: GridView.builder(

                          shrinkWrap: true,

                          itemCount: \_imagesList.length + 1,

                          gridDelegate:

                              const SliverGridDelegateWithFixedCrossAxisCount(

                            crossAxisCount: 2,

                            mainAxisSpacing: 25,

                            crossAxisSpacing: 25,

                            childAspectRatio: 3 / 2,

                          ),

                          itemBuilder: (context, index) {

                            if (index == \_imagesList.length) {

                              return IconButton(

                                icon: const Icon(Icons.add),

                                onPressed: () {

                                  \_selectImageFromGallery(-1);

                                },

                              );

                            }

                            return MaterialButton(

                              onPressed: () {},

                              child: Image(

                                image: \_imagesList[index],

                                fit: BoxFit.fill,

                              ),

                            );

                          },

                        ),

                      ),

                    ],

                  ),

                ),

              ],

            ),

          ),

        ),

      ),

    );

  }

}

* File “my\_postings\_screen.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view/hostScreens/create\_posting\_screen.dart';

import 'package:guest\_hotel/view/widgets/posting\_list\_tile\_button.dart';

import 'package:guest\_hotel/view/widgets/posting\_list\_tile\_ui.dart';

import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

class MyPostingsScreen extends StatefulWidget {

  const MyPostingsScreen({super.key});

  @override

  State<MyPostingsScreen> createState() => \_MyPostingsScreenState();

}

class \_MyPostingsScreenState extends State<MyPostingsScreen> {

  List<PostingModel> userPostings = [];

  @override

  void initState() {

    super.initState();

    \_filterUserPostings();

  }

  void \_filterUserPostings() {

    final currentUserID = AppConstants.currentUser.id;

    userPostings = AppConstants.currentUser.myPostings!

        .where((posting) => posting.host!.id == currentUserID)

        .toList();

    setState(() {});

  }

  Future<void> \_deletePosting(PostingModel posting) async {

    try {

      await FirebaseFirestore.instance

          .collection('postings')

          .doc(posting.id)

          .delete();

      setState(() {

        userPostings.remove(posting);

      });

      Get.snackbar("Success", "Posting deleted successfully.");

    } catch (e) {

      Get.snackbar("Error", "Failed to delete posting.");

    }

  }

  void \_showDeleteConfirmationDialog(PostingModel posting) {

    showDialog(

      context: context,

      builder: (BuildContext context) {

        return AlertDialog(

          title: Text("Delete Posting"),

          content: Text("Are you sure you want to delete this post?"),

          actions: [

            TextButton(

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

              child: Text("No"),

            ),

            TextButton(

              onPressed: () {

                Navigator.of(context).pop();

                \_deletePosting(posting);

              },

              child: Text("Delete"),

            ),

          ],

        );

      },

    );

  }

  @override

  Widget build(BuildContext context) {

    return Padding(

      padding: const EdgeInsets.only(top: 25),

      child: ListView.builder(

        itemCount: userPostings.length + 1,

        itemBuilder: (context, index) {

          return Padding(

            padding: const EdgeInsets.fromLTRB(20, 0, 20, 20),

            child: InkResponse(

              onTap: () {

                if (index == userPostings.length) {

                  Get.to(CreatePostingScreen(posting: null));

                } else {

                  Get.to(CreatePostingScreen(posting: userPostings[index]));

                }

              },

              child: Container(

                decoration: BoxDecoration(

                  border: Border.all(

                    color: Colors.green,

                    width: 1.2,

                  ),

                ),

                child: (index == userPostings.length)

                    ? const PostingListTileButton()

                    : Stack(

                        children: [

                          PostingListTileUI(

                            posting: userPostings[index],

                          ),

                          Positioned(

                            top: 0,

                            right: 0,

                            child: IconButton(

                              icon: Icon(Icons.close, color: Colors.red),

                              onPressed: () => \_showDeleteConfirmationDialog(

                                  userPostings[index]),

                            ),

                          ),

                        ],

                      ),

              ),

            ),

          );

        },

      ),

    );

  }

}

+ Folder “widgets”:

* File “amenities\_ui.dart”:

import 'package:flutter/material.dart';

class AmenitiesUI extends StatefulWidget

{

  String type;

  int startValue;

  Function decreaseValue;

  Function increaseValue;

  AmenitiesUI({super.key, required this.type, required this.startValue, required this.decreaseValue, required this.increaseValue});

  @override

  State<AmenitiesUI> createState() => \_AmenitiesUIState();

}

class \_AmenitiesUIState extends State<AmenitiesUI>

{

  int? \_valueDigit;

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    \_valueDigit = widget.startValue;

  }

  @override

  Widget build(BuildContext context) {

    return Row(

      mainAxisAlignment: MainAxisAlignment.spaceBetween,

      children: [

        Text(

          widget.type,

          style: const TextStyle(

            fontSize: 21.0,

          ),

        ),

        Row(

          children: <Widget>[

            IconButton(

              icon: const Icon(Icons.remove),

              onPressed: ()

              {

                widget.decreaseValue();

                \_valueDigit = \_valueDigit! - 1;

                if(\_valueDigit! <0 ){

                  \_valueDigit = 0;

                }

                setState(() {

                });

              },

            ),

            Text(

              \_valueDigit.toString(),

              style: const TextStyle(

                fontSize: 21,

              ),

            ),

            IconButton(

              icon: const Icon(Icons.add),

              onPressed: ()

              {

                widget.increaseValue();

                \_valueDigit = \_valueDigit! + 1;

                setState(() {

                });

              },

            ),

          ],

        ),

      ],

    );

  }

}

* File “calender\_ui.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:flutter/material.dart';

class CalenderUI extends StatefulWidget {

  int? monthIndex;

  List<DateTime>? bookedDates;

  Function? selectDate;

  Function? getSelectedDates;

  CalenderUI({

    super.key,

    this.getSelectedDates,

    this.selectDate,

    this.monthIndex,

    this.bookedDates,

  });

  @override

  State<CalenderUI> createState() => \_CalenderUIState();

}

class \_CalenderUIState extends State<CalenderUI> {

  List<DateTime> \_selectedDates = [];

  List<MonthTileWidget> \_monthTiles = [];

  int? \_currentMonthInt;

  int? \_currentYearInt;

  \_setUpMonthTiles() {

    \_monthTiles = [];

    int daysInMonth = AppConstants.daysInMonths![\_currentMonthInt]!;

    DateTime firstDayOfMonth = DateTime(\_currentYearInt!, \_currentMonthInt!, 1);

    int firstWeekOfMonth = firstDayOfMonth.weekday;

    if (firstWeekOfMonth != 7) {

      for (int i = 0; i < firstWeekOfMonth; i++) {

        \_monthTiles.add(MonthTileWidget(

          dateTime: null,

        ));

      }

    }

    for (int i = 1; i <= daysInMonth; i++) {

      DateTime date = DateTime(\_currentYearInt!, \_currentMonthInt!, i);

      \_monthTiles.add(MonthTileWidget(dateTime: date));

    }

  }

  \_selectDate(DateTime date) {

    if (\_selectedDates.contains(date)) {

      \_selectedDates.remove(date);

    } else {

      \_selectedDates.add(date);

    }

    widget.selectDate!(date);

    setState(() {});

  }

  void initState() {

    super.initState();

    \_currentMonthInt = (DateTime.now().month + widget.monthIndex!) % 12;

    if (\_currentMonthInt == 0) {

      \_currentMonthInt = 12;

    }

    \_currentYearInt = DateTime.now().year;

    if (\_currentMonthInt! < DateTime.now().month) {

      \_currentYearInt = \_currentYearInt! + 1;

    }

    \_selectedDates.sort();

    \_selectedDates.addAll(widget.getSelectedDates!());

    \_setUpMonthTiles();

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      children: [

        Padding(

          padding: const EdgeInsets.only(bottom: 20.0),

          child: Text(

            "${AppConstants.monthDict[\_currentMonthInt]} - ${\_currentYearInt}",

          ),

        ),

        GridView.builder(

            itemCount: \_monthTiles.length,

            shrinkWrap: true,

            gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

              crossAxisCount: 7,

              childAspectRatio: 1 / 1,

            ),

            itemBuilder: (context, index) {

              MonthTileWidget monthTile = \_monthTiles[index];

              if (monthTile.dateTime == null) {

                return const MaterialButton(

                  onPressed: null,

                  child: Text(""),

                );

              }

              if (widget.bookedDates!.contains(monthTile.dateTime)) {

                return MaterialButton(

                  onPressed: null,

                  color: Colors.yellow,

                  disabledColor: Colors.yellow,

                  child: monthTile,

                );

              }

              return MaterialButton(

                onPressed: () {

                  \_selectDate(monthTile.dateTime!);

                },

                color: (\_selectedDates.contains(monthTile.dateTime))

                    ? Colors.green

                    : Colors.white,

                child: monthTile,

              );

            }),

      ],

    );

  }

}

class MonthTileWidget extends StatelessWidget {

  DateTime? dateTime;

  MonthTileWidget({

    super.key,

    this.dateTime,

  });

  @override

  Widget build(BuildContext context) {

    return Text(

      dateTime == null ? "" : dateTime!.day.toString(),

      style: const TextStyle(

        fontSize: 10,

      ),

    );

  }

}

* File “posting\_grid\_tile\_ui.dart”:

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:custom\_rating\_bar/custom\_rating\_bar.dart';

import 'package:flutter/material.dart';

class PostingGridTileUi extends StatefulWidget {

  final PostingModel? posting;

  PostingGridTileUi({Key? key, this.posting}) : super(key: key);

  @override

  State<PostingGridTileUi> createState() => \_PostingGridTileUiState();

}

class \_PostingGridTileUiState extends State<PostingGridTileUi> {

  PostingModel? posting;

  @override

  void initState() {

    super.initState();

    posting = widget.posting;

    updateUI();

  }

  Future<void> updateUI() async {

    if (posting != null) {

      await posting!.getFirstImageFromStorage();

      setState(() {});

    }

  }

  @override

  Widget build(BuildContext context) {

    return Column(

      mainAxisAlignment: MainAxisAlignment.spaceAround,

      crossAxisAlignment: CrossAxisAlignment.start,

      children: [

        AspectRatio(

          aspectRatio: 3 / 2,

          child: (posting?.displayImages?.isEmpty ?? true)

              ? Container(

                  color: Colors.grey,

                  child: const Center(child: Text('No Image')),

                )

              : Image(

                  image: posting!.displayImages!.first,

                  fit: BoxFit.cover,

                ),

        ),

        Text(

          "${posting?.type ?? ''} ${posting?.city ?? ''}, ${posting?.country ?? ''}",

          maxLines: 2,

          style: const TextStyle(

            fontSize: 12,

            fontWeight: FontWeight.bold,

          ),

        ),

        Text(

          posting?.name ?? '',

          maxLines: 1,

          style: const TextStyle(

            fontSize: 12,

            fontWeight: FontWeight.bold,

          ),

        ),

        Text(

          '${posting?.price?.toString() ?? '0'}/night',

          style: const TextStyle(

            fontSize: 12,

            fontWeight: FontWeight.bold,

          ),

        ),

        Row(

          children: [

            RatingBar.readOnly(

              size: 28,

              maxRating: 5,

              initialRating: posting?.getCurrentRating() ?? 0,

              filledIcon: Icons.star,

              emptyIcon: Icons.star\_border,

              filledColor: Colors.green,

            ),

          ],

        ),

      ],

    );

  }

}

* File “posting\_info\_tile\_ui.dart”:

import 'package:flutter/material.dart';

class PostingInfoTileUI extends StatefulWidget {

  IconData? iconData;

  String? category;

  String? categoryInfo;

  PostingInfoTileUI(

      {super.key, this.iconData, this.category, this.categoryInfo});

  @override

  State<PostingInfoTileUI> createState() => \_PostingInfoTileUIState();

}

class \_PostingInfoTileUIState extends State<PostingInfoTileUI>

    with SingleTickerProviderStateMixin {

  late AnimationController \_controller;

  @override

  void initState() {

    super.initState();

    \_controller = AnimationController(vsync: this);

  }

  @override

  void dispose() {

    \_controller.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return ListTile(

      leading: Icon(

        widget.iconData,

        size: 30,

      ),

      title: Text(

        widget.category!,

        style: const TextStyle(

          fontWeight: FontWeight.bold,

          fontSize: 25,

        ),

      ),

      subtitle: Text(

        widget.categoryInfo!,

        style: const TextStyle(

          fontSize: 21,

        ),

      ),

    );

  }

}

* File “posting\_list\_tile\_button.dart”:

import 'package:flutter/material.dart';

class PostingListTileButton extends StatelessWidget {

  const PostingListTileButton({super.key});

  @override

  Widget build(BuildContext context) {

    return SizedBox(

      height: MediaQuery.of(context).size.height / 11,

      child: const Row(

        mainAxisAlignment: MainAxisAlignment.center,

        children: [

          Icon(Icons.add),

          Text(

            'Create a listing',

            style: TextStyle(

              fontWeight: FontWeight.bold,

              fontSize: 21,

            ),

          ),

        ],

      ),

    );

  }

}

* File “posting\_list\_tile\_ui.dart”:

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:flutter/material.dart';

class PostingListTileUI extends StatefulWidget

{

  PostingModel? posting;

  PostingListTileUI({super.key, this.posting,});

  @override

  State<PostingListTileUI> createState() => \_PostingListTileUIState();

}

class \_PostingListTileUIState extends State<PostingListTileUI>

{

  PostingModel? posting;

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    posting = widget.posting;

  }

  @override

  Widget build(BuildContext context) {

    return Padding(

        padding: const EdgeInsets.all(15),

      child: ListTile(

        leading: Padding(

            padding: const EdgeInsets.only(left: 10),

          child: Text(

            posting!.name!,

            maxLines: 2,

            style: const TextStyle(

              fontSize: 18,

              fontWeight: FontWeight.bold,

            ),

          ),

        ),

        trailing: AspectRatio(

            aspectRatio: 3/2,

          child: Image(

              image: posting!.displayImages!.first,

            fit: BoxFit.fitWidth,

          ),

        ),

      ),

    );

  }

}

+ File “guest\_home\_screen.dart”:

import 'package:guest\_hotel/view/guestScreens/account\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/chatbot\_srceen.dart';

import 'package:guest\_hotel/view/guestScreens/explore\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/statistics\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/saved\_listings\_screen.dart';

import 'package:guest\_hotel/view/guestScreens/trips\_screen.dart';

import 'package:flutter/material.dart';

class GuestHomeScreen extends StatefulWidget {

  const GuestHomeScreen({super.key});

  @override

  State<GuestHomeScreen> createState() => \_GuestHomeScreenState();

}

class \_GuestHomeScreenState extends State<GuestHomeScreen> {

  int selectedIndex = 0;

  final List<String> screenTitles = [

    'Explore',

    'Saved',

    '',

    'Trips',

    'Profile',

  ];

  final List<Widget> screens = [

    const ExploreScreen(),

    const SavedListingsScreen(),

    const ChatbotScreen(),

    const TripsScreen(),

    const AccountScreen(),

  ];

  BottomNavigationBarItem customNavigationBarItem(

      int index, IconData iconData, String title) {

    return BottomNavigationBarItem(

      icon: Icon(

        iconData,

        color: Colors.black,

      ),

      activeIcon: Icon(

        iconData,

        color: Colors.green,

      ),

      label: title,

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        flexibleSpace: Container(

          decoration: const BoxDecoration(

            color: Colors.green,

          ),

        ),

        title: Text(

          screenTitles[selectedIndex],

          style: const TextStyle(

            fontSize: 18,

            color: Colors.white,

          ),

        ),

        automaticallyImplyLeading: false,

        centerTitle: true,

      ),

      body: screens[selectedIndex],

      bottomNavigationBar: BottomNavigationBar(

        onTap: (i) {

          setState(() {

            selectedIndex = i;

          });

        },

        currentIndex: selectedIndex,

        type: BottomNavigationBarType.fixed,

        items: <BottomNavigationBarItem>[

          customNavigationBarItem(0, Icons.search, screenTitles[0]),

          customNavigationBarItem(1, Icons.favorite, screenTitles[1]),

          customNavigationBarItem(2, Icons.question\_answer, screenTitles[2]),

          customNavigationBarItem(3, Icons.home\_outlined, screenTitles[3]),

          customNavigationBarItem(4, Icons.person\_outlined, screenTitles[4]),

        ],

      ),

    );

  }

}

+ File “login\_screen.dart”:

import 'package:guest\_hotel/global.dart';

import 'package:guest\_hotel/view/signup\_screen.dart';

import 'package:flutter/material.dart';

import 'package:get/get.dart';

class LoginScreen extends StatefulWidget {

  const LoginScreen({super.key});

  @override

  State<LoginScreen> createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  final \_formkey = GlobalKey<FormState>();

  final TextEditingController \_emailTextEditingController =

      TextEditingController();

  final TextEditingController \_passwordTextEditingController =

      TextEditingController();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        padding: const EdgeInsets.all(20),

        decoration: const BoxDecoration(color: Colors.green),

        child: ListView(

          children: [

            Image.asset(

              "images/login.png",

            ),

            const SizedBox(height: 20),

            const Center(

              child: Text(

                "LOGIN PAGE",

                style: TextStyle(

                  fontSize: 30,

                  fontWeight: FontWeight.bold,

                  color: Colors.white,

                  letterSpacing: 3,

                ),

                textAlign: TextAlign.center,

              ),

            ),

            const SizedBox(height: 20),

            Form(

              key: \_formkey,

              child: Container(

                margin: const EdgeInsets.symmetric(vertical: 10.0),

                child: Column(

                  crossAxisAlignment: CrossAxisAlignment.start,

                  children: [

                    \_buildTextField(

                      controller: \_emailTextEditingController,

                      hintText: "Email",

                      keyboardType: TextInputType.emailAddress,

                      validator: (valueEmail) {

                        if (valueEmail == null || valueEmail.isEmpty) {

                          return "Please enter your Email";

                        }

                        return null;

                      },

                    ),

                    const SizedBox(height: 15),

                    \_buildTextField(

                      controller: \_passwordTextEditingController,

                      hintText: "Password",

                      obscureText: true,

                      validator: (valuePassword) {

                        if (valuePassword!.length < 5) {

                          return "Password too weak";

                        }

                        return null;

                      },

                    ),

                  ],

                ),

              ),

            ),

            const SizedBox(height: 20),

            Padding(

              padding: const EdgeInsets.only(top: 20),

              child: ElevatedButton(

                onPressed: () async {

                  if (\_formkey.currentState!.validate()) {

                    await userViewModel.login(

                      \_emailTextEditingController.text.trim(),

                      \_passwordTextEditingController.text.trim(),

                    );

                  }

                },

                style: ElevatedButton.styleFrom(

                  backgroundColor: Colors.white,

                  padding: const EdgeInsets.symmetric(horizontal: 30),

                ),

                child: const Text(

                  "LOGIN",

                  style: TextStyle(

                    fontWeight: FontWeight.bold,

                    fontSize: 30,

                    color: Colors.green,

                  ),

                ),

              ),

            ),

            TextButton(

              onPressed: () {

                Get.to(const SignupScreen());

              },

              child: const Text(

                "Don't have an account? Create here",

                style: TextStyle(

                  color: Colors.white,

                  fontSize: 18,

                  fontWeight: FontWeight.bold,

                ),

              ),

            ),

          ],

        ),

      ),

    );

  }

  Widget \_buildTextField({

    required TextEditingController controller,

    required String hintText,

    bool obscureText = false,

    TextInputType keyboardType = TextInputType.text,

    String? Function(String?)? validator,

  }) {

    return TextFormField(

      controller: controller,

      obscureText: obscureText,

      keyboardType: keyboardType,

      validator: validator,

      decoration: InputDecoration(

        hintText: hintText,

        border: OutlineInputBorder(),

        fillColor: Colors.white,

        filled: true,

      ),

      style: const TextStyle(

        fontSize: 18,

        color: Colors.black,

      ),

    );

  }

}

+ File “signup\_screen.dart”:

import 'dart:io';

import 'package:guest\_hotel/global.dart';

import 'package:guest\_hotel/view/login\_screen.dart';

import 'package:flutter/material.dart';

import 'package:get/get.dart';

import 'package:image\_picker/image\_picker.dart';

class SignupScreen extends StatefulWidget {

  const SignupScreen({super.key});

  @override

  State<SignupScreen> createState() => \_SignupScreenState();

}

class \_SignupScreenState extends State<SignupScreen> {

  final TextEditingController \_emailTextEditingController =

      TextEditingController();

  final TextEditingController \_passwordTextEditingController =

      TextEditingController();

  final TextEditingController \_firstNameTextEditingController =

      TextEditingController();

  final TextEditingController \_lastNameTextEditingController =

      TextEditingController();

  final TextEditingController \_cityTextEditingController =

      TextEditingController();

  final TextEditingController \_countryTextEditingController =

      TextEditingController();

  final TextEditingController \_bioTextEditingController =

      TextEditingController();

  final \_formkey = GlobalKey<FormState>();

  File? imageFileOfUser;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        flexibleSpace: Container(

          decoration: const BoxDecoration(color: Colors.green),

        ),

        title: const Text(

          "SIGN UP PAGE",

          style: TextStyle(color: Colors.white),

        ),

      ),

      body: Container(

        padding: const EdgeInsets.only(top: 20),

        decoration: const BoxDecoration(color: Colors.green),

        child: ListView(

          children: [

            Image.asset(

              "images/signup.png",

            ),

            const SizedBox(height: 20),

            const Center(

              child: Text(

                "SIGN UP PAGE",

                style: TextStyle(

                  fontSize: 30,

                  fontWeight: FontWeight.bold,

                  color: Colors.white,

                  letterSpacing: 3,

                ),

                textAlign: TextAlign.center,

              ),

            ),

            Padding(

              padding: const EdgeInsets.all(10.0),

              child: Form(

                key: \_formkey,

                child: Container(

                  margin: const EdgeInsets.symmetric(vertical: 10.0),

                  child: Column(

                    crossAxisAlignment: CrossAxisAlignment.start,

                    children: [

                      \_buildTextField(

                        controller: \_emailTextEditingController,

                        hintText: "Email",

                        keyboardType: TextInputType.emailAddress,

                      ),

                      const SizedBox(height: 15),

                      \_buildTextField(

                        controller: \_passwordTextEditingController,

                        hintText: "Password",

                        obscureText: true,

                      ),

                      const SizedBox(height: 15),

                      \_buildTextField(

                        controller: \_firstNameTextEditingController,

                        hintText: "First Name",

                      ),

                      const SizedBox(height: 15),

                      \_buildTextField(

                        controller: \_lastNameTextEditingController,

                        hintText: "Last Name",

                      ),

                      const SizedBox(height: 15),

                      \_buildTextField(

                        controller: \_cityTextEditingController,

                        hintText: "City",

                      ),

                      const SizedBox(height: 15),

                      \_buildTextField(

                        controller: \_countryTextEditingController,

                        hintText: "Country",

                      ),

                      const SizedBox(height: 15),

                      \_buildTextField(

                        controller: \_bioTextEditingController,

                        hintText: "Bio",

                        maxLines: 3,

                      ),

                    ],

                  ),

                ),

              ),

            ),

            Padding(

              padding: const EdgeInsets.all(20.0),

              child: MaterialButton(

                onPressed: () async {

                  var imageFile = await ImagePicker()

                      .pickImage(source: ImageSource.gallery);

                  if (imageFile != null) {

                    imageFileOfUser = File(imageFile.path);

                    setState(() {

                      imageFileOfUser;

                    });

                  }

                },

                child: imageFileOfUser == null

                    ? const Icon(Icons.add\_a\_photo\_outlined)

                    : CircleAvatar(

                        radius: MediaQuery.of(context).size.width / 5.0,

                        backgroundColor: Colors.green,

                        backgroundImage: FileImage(imageFileOfUser!),

                      ),

              ),

            ),

            Padding(

              padding: const EdgeInsets.only(top: 20, left: 40),

              child: ElevatedButton(

                onPressed: () async {

                  if (!\_formkey.currentState!.validate() ||

                      imageFileOfUser == null) {

                    Get.snackbar(

                      "Field Missing",

                      "Please choose an image and fill out the complete sign-up form.",

                    );

                    return;

                  }

                  try {

                    bool isSuccess = await userViewModel.signUp(

                      \_emailTextEditingController.text.trim(),

                      \_passwordTextEditingController.text.trim(),

                      \_firstNameTextEditingController.text.trim(),

                      \_lastNameTextEditingController.text.trim(),

                      \_cityTextEditingController.text.trim(),

                      \_countryTextEditingController.text.trim(),

                      \_bioTextEditingController.text.trim(),

                      imageFileOfUser!,

                    );

                    if (isSuccess) {

                      Get.snackbar(

                        "Success",

                        "Account created successfully! Redirecting to login page...",

                      );

                      Get.off(() => const LoginScreen());

                    } else {

                      Get.snackbar(

                        "Error",

                        "Sign-up failed. Please try again.",

                      );

                    }

                  } catch (e) {

                    Get.snackbar(

                      "Error",

                      "An unexpected error occurred: $e",

                    );

                  }

                },

                style: ElevatedButton.styleFrom(

                  backgroundColor: Colors.white,

                  padding: const EdgeInsets.symmetric(horizontal: 30),

                ),

                child: const Text(

                  "SIGN UP",

                  style: TextStyle(

                    fontWeight: FontWeight.bold,

                    fontSize: 30,

                    color: Colors.green,

                  ),

                ),

              ),

            ),

            const SizedBox(height: 60),

          ],

        ),

      ),

    );

  }

  Widget \_buildTextField({

    required TextEditingController controller,

    required String hintText,

    bool obscureText = false,

    TextInputType keyboardType = TextInputType.text,

    int maxLines = 1,

  }) {

    return TextField(

      controller: controller,

      obscureText: obscureText,

      keyboardType: keyboardType,

      maxLines: maxLines,

      decoration: InputDecoration(

        hintText: hintText,

        border: OutlineInputBorder(),

        fillColor: Colors.white,

        filled: true,

      ),

    );

  }

}

+ File “spalash\_screen.dart”:

import 'dart:async';

import 'package:guest\_hotel/view/login\_screen.dart';

import 'package:get/get.dart';

import 'package:flutter/material.dart';

class SpalashScreen extends StatefulWidget {

  const SpalashScreen({super.key});

  @override

  State<SpalashScreen> createState() => \_SpalashScreenState();

}

class \_SpalashScreenState extends State<SpalashScreen> {

  @override

  void initState() {

    // TODO: implement initState

    super.initState();

    Timer(const Duration(seconds: 2), () {

      Get.to(const LoginScreen());

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Container(

        decoration: const BoxDecoration(

          color: Colors.green,

        ),

        child: Center(

          child: Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              Image.asset("images/login.png"),

              const Padding(

                padding: EdgeInsets.only(top: 20.0),

                child: Text(

                  "Welcome to Booking Hotel System!",

                  style: TextStyle(

                    fontSize: 30,

                    fontWeight: FontWeight.bold,

                    color: Colors.white,

                  ),

                ),

              )

            ],

          ),

        ),

      ),

    );

  }

}

+ File “view\_posting\_screen.dart”:

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view/guestScreens/book\_listing\_screen.dart';

import 'package:guest\_hotel/view/widgets/posting\_info\_tile\_ui.dart';

import 'package:flutter/material.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

import 'package:get/get.dart';

class PostingReport {

  final String userId;

  final String postId;

  PostingReport({required this.userId, required this.postId});

  Future<void> submitReport() async {

    try {

      await FirebaseFirestore.instance.collection('postReports').add({

        'userId': userId,

        'postId': postId,

        'timestamp': Timestamp.now(),

      });

    } catch (error) {

      print("Error reporting post: $error");

    }

  }

}

class ViewPostingScreen extends StatefulWidget {

  // final PostingModel? posting;

  // ViewPostingScreen({Key? key, this.posting}) : super(key: key);

  PostingModel? posting;

  ViewPostingScreen({super.key, this.posting});

  @override

  State<ViewPostingScreen> createState() => \_ViewPostingScreenState();

}

class \_ViewPostingScreenState extends State<ViewPostingScreen> {

  PostingModel? posting;

  getRequiredInfo() async {

    if (posting != null) {

      await posting!.getAllImagesFromStorage();

      await posting!.getHostFromFirestore();

      setState(() {});

    }

  }

  @override

  void initState() {

    super.initState();

    posting = widget.posting;

    getRequiredInfo();

  }

  void reportPost() {

    final report = PostingReport(

      userId: AppConstants.currentUser.id!,

      postId: posting!.id!,

    );

    report.submitReport();

    ScaffoldMessenger.of(context).showSnackBar(

      SnackBar(content: Text('Post reported successfully.')),

    );

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        flexibleSpace: Container(

          decoration: const BoxDecoration(

            color: Colors.green,

          ),

        ),

        title: Text('Posting Information'),

        actions: [

          IconButton(

            icon: const Icon(Icons.favorite\_outline, color: Colors.white),

            onPressed: () {

              AppConstants.currentUser.addSavePosting(posting!);

              Get.snackbar(

                "Success",

                "Added to favorites!",

                backgroundColor: Colors.white,

                colorText: Colors.black,

              );

            },

          ),

          PopupMenuButton<String>(

            onSelected: (value) {

              if (value == 'Report') {

                reportPost();

              }

            },

            itemBuilder: (BuildContext context) {

              return [

                PopupMenuItem<String>(

                  value: 'Report',

                  child: Text('Report'),

                ),

              ];

            },

          ),

        ],

      ),

      body: posting == null

          ? Center(child: CircularProgressIndicator())

          : SingleChildScrollView(

              child: Column(

                children: [

                  // Listing Image

                  AspectRatio(

                    aspectRatio: 3 / 2,

                    child: PageView.builder(

                      itemCount: posting!.displayImages!.length,

                      itemBuilder: (context, index) {

                        MemoryImage currentImage =

                            posting!.displayImages![index];

                        return Image(image: currentImage, fit: BoxFit.fill);

                      },

                    ),

                  ),

                  Padding(

                    padding: const EdgeInsets.fromLTRB(20, 20, 20, 5),

                    child: Column(

                      crossAxisAlignment: CrossAxisAlignment.start,

                      children: [

                        // Posting Name, Price and Book Now Button

                        Row(

                          mainAxisAlignment: MainAxisAlignment.spaceBetween,

                          crossAxisAlignment: CrossAxisAlignment.start,

                          children: [

                            Expanded(

                              child: Text(

                                posting!.name!.toUpperCase(),

                                style: const TextStyle(

                                  fontWeight: FontWeight.bold,

                                  fontSize: 16,

                                ),

                                maxLines: 3,

                                overflow: TextOverflow.ellipsis,

                              ),

                            ),

                            Column(

                              crossAxisAlignment: CrossAxisAlignment.end,

                              children: [

                                Container(

                                  decoration:

                                      const BoxDecoration(color: Colors.green),

                                  child: MaterialButton(

                                    onPressed: () {

                                      Get.to(BookListingScreen(

                                          posting: posting,

                                          hostID: posting!.host!.id!));

                                    },

                                    child: const Text(

                                      'Book now',

                                      style: TextStyle(color: Colors.white),

                                    ),

                                  ),

                                ),

                                const SizedBox(height: 10),

                                Text(

                                  '${posting!.price} /night',

                                  style: const TextStyle(fontSize: 14.0),

                                ),

                              ],

                            ),

                          ],

                        ),

                        // Description, Host Profile Picture, and Name

                        Padding(

                          padding:

                              const EdgeInsets.only(top: 25.0, bottom: 25.0),

                          child: Row(

                            crossAxisAlignment: CrossAxisAlignment.start,

                            mainAxisAlignment: MainAxisAlignment.spaceBetween,

                            children: [

                              SizedBox(

                                width: MediaQuery.of(context).size.width / 1.75,

                                child: Text(

                                  posting!.description!,

                                  textAlign: TextAlign.justify,

                                  style: const TextStyle(fontSize: 14),

                                  maxLines: 5,

                                ),

                              ),

                              Column(

                                children: [

                                  GestureDetector(

                                    onTap: () {},

                                    child: CircleAvatar(

                                      radius:

                                          MediaQuery.of(context).size.width /

                                              12.5,

                                      backgroundColor: Colors.black,

                                      child: CircleAvatar(

                                        backgroundImage:

                                            posting!.host!.displayImage,

                                        radius:

                                            MediaQuery.of(context).size.width /

                                                13,

                                      ),

                                    ),

                                  ),

                                  Padding(

                                    padding: const EdgeInsets.only(top: 20),

                                    child: Text(

                                      posting!.host!.getFullNameOfUser(),

                                      style: const TextStyle(

                                          fontWeight: FontWeight.bold),

                                    ),

                                  ),

                                ],

                              ),

                            ],

                          ),

                        ),

                        // Apartments, Beds, Bathrooms Information

                        Padding(

                          padding: const EdgeInsets.only(bottom: 20),

                          child: ListView(

                            shrinkWrap: true,

                            physics: NeverScrollableScrollPhysics(),

                            children: [

                              PostingInfoTileUI(

                                iconData: Icons.home,

                                category: 'Apartment',

                                categoryInfo:

                                    '${posting!.getGuestsNumber()} guests',

                              ),

                              PostingInfoTileUI(

                                iconData: Icons.hotel,

                                category: 'Beds',

                                categoryInfo: posting!.getBedroomText(),

                              ),

                              PostingInfoTileUI(

                                iconData: Icons.wc,

                                category: 'Bathrooms',

                                categoryInfo: posting!.getBathroomText(),

                              ),

                            ],

                          ),

                        ),

                        // Amenities

                        const Text(

                          'Amenities: ',

                          style: TextStyle(

                              fontWeight: FontWeight.bold, fontSize: 25),

                        ),

                        Padding(

                          padding: const EdgeInsets.only(top: 5.0, bottom: 25),

                          child: GridView.count(

                            shrinkWrap: true,

                            crossAxisCount: 2,

                            childAspectRatio: 3.6,

                            children: List.generate(

                              posting!.amenities!.length,

                              (index) {

                                String currentAmenity =

                                    posting!.amenities![index];

                                return Chip(

                                  label: Padding(

                                    padding: const EdgeInsets.symmetric(

                                        horizontal: 20.0),

                                    child: Text(

                                      currentAmenity,

                                      style: const TextStyle(

                                        color: Colors.black45,

                                        fontSize: 16,

                                        fontWeight: FontWeight.bold,

                                      ),

                                    ),

                                  ),

                                  backgroundColor: Colors.white10,

                                );

                              },

                            ),

                          ),

                        ),

                        // Location

                        Padding(

                          padding: const EdgeInsets.only(top: 20, bottom: 10),

                          child: Text(

                            posting!.getFullAddress(),

                            style: const TextStyle(fontSize: 18),

                          ),

                        ),

                      ],

                    ),

                  ),

                ],

              ),

            ),

    );

  }

}

- Folder “viewmodel”:

+ File “posting\_view\_model.dart”:

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

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

import '../global.dart';

import '../model/app\_constants.dart';

class PostingViewModel {

  addListingInfoToFirestore() async {

    postingModel.setImagesName();

    Map<String, dynamic> dataMap = {

      "address": postingModel.address,

      "amenities": postingModel.amenities,

      "bathrooms": postingModel.bathrooms,

      "description": postingModel.description,

      "beds": postingModel.beds,

      "city": postingModel.city,

      "country": postingModel.country,

      "hostID": AppConstants.currentUser.id,

      "imageNames": postingModel.imageNames,

      "name": postingModel.name,

      "price": postingModel.price,

      "rating": 3.5,

      "type": postingModel.type,

    };

    DocumentReference ref =

        await FirebaseFirestore.instance.collection("postings").add(dataMap);

    postingModel.id = ref.id;

    await AppConstants.currentUser.addPostingToMyPostings(postingModel);

  }

  updatePostingInfoToFirestore() async {

    postingModel.setImagesName();

    Map<String, dynamic> dataMap = {

      "address": postingModel.address,

      "amenities": postingModel.amenities,

      "bathrooms": postingModel.bathrooms,

      "description": postingModel.description,

      "beds": postingModel.beds,

      "city": postingModel.city,

      "country": postingModel.country,

      "hostID": AppConstants.currentUser.id,

      "imageNames": postingModel.imageNames,

      "name": postingModel.name,

      "price": postingModel.price,

      "rating": 3.5,

      "type": postingModel.type,

    };

    FirebaseFirestore.instance

        .collection("postings")

        .doc(postingModel.id)

        .update(dataMap);

  }

  addImagesToFirebaseStorage() async {

    PostingModel posting = PostingModel();

    for (int i = 0; i < postingModel.displayImages!.length; i++) {

      Reference ref = FirebaseStorage.instance

          .ref()

          .child("postingImages")

          .child(postingModel.id!)

          .child(postingModel.imageNames![i]);

      await ref

          .putData(postingModel.displayImages![i].bytes)

          .whenComplete(() {});

    }

  }

}

+ File “user\_view\_model.dart”:

import 'dart:io';

import 'package:guest\_hotel/model/app\_constants.dart';

import 'package:guest\_hotel/view/admin\_screen/adminbottomnav.dart';

import 'package:guest\_hotel/view/guestScreens/account\_screen.dart';

import 'package:guest\_hotel/view/guest\_home\_screen.dart';

import 'package:cloud\_firestore/cloud\_firestore.dart';

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

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

import 'package:flutter/cupertino.dart';

import 'package:get/get.dart';

import 'package:get/get\_core/src/get\_main.dart';

import '../model/user\_model.dart';

class UserViewModel {

  UserModel userModel = UserModel();

  signUp(email, password, firstName, lastName, city, country, bio,

      imageFileOfUser) async {

    Get.snackbar("Please wait", "Your account is being created.");

    try {

      await FirebaseAuth.instance

          .createUserWithEmailAndPassword(

        email: email,

        password: password,

      )

          .then((result) async {

        String currentUserID = result.user!.uid;

        AppConstants.currentUser.id = currentUserID;

        AppConstants.currentUser.firstName = firstName;

        AppConstants.currentUser.lastName = lastName;

        AppConstants.currentUser.city = city;

        AppConstants.currentUser.country = country;

        AppConstants.currentUser.bio = bio;

        AppConstants.currentUser.email = email;

        AppConstants.currentUser.password = password;

        await saveUserToFirestore(

                bio, city, country, email, firstName, lastName, currentUserID)

            .whenComplete(() async {

          addImageToFirebaseStorage(imageFileOfUser, currentUserID);

        });

        Get.snackbar("Congratulations", "Your account has been created.");

      });

    } catch (e) {

      Get.to(const GuestHomeScreen());

      Get.snackbar("Error!!!", e.toString());

    }

  }

  Future<void> saveUserToFirestore(

      bio, city, country, email, firstName, lastName, id) async {

    Map<String, dynamic> dataMap = {

      "bio": bio,

      "city": city,

      "country": country,

      "email": email,

      "firstName": firstName,

      "lastName": lastName,

      "isHost": false,

      "myPostingIDs": [],

      "savedPostingIDs": [],

      "earnings": 0,

      "role": "user" // Added role field with default value 'user'

    };

    await FirebaseFirestore.instance.collection("users").doc(id).set(dataMap);

  }

  addImageToFirebaseStorage(File imageFileOfUser, currentUserID) async {

    Reference referenceStorage = FirebaseStorage.instance

        .ref()

        .child("userImages")

        .child(currentUserID)

        .child(currentUserID + ".png");

    await referenceStorage.putFile(imageFileOfUser).whenComplete(() {});

    AppConstants.currentUser.displayImage =

        MemoryImage(imageFileOfUser.readAsBytesSync());

  }

  login(email, password) async {

    Get.snackbar("Please wait", "Checking your credentials...");

    try {

      FirebaseAuth.instance

          .signInWithEmailAndPassword(

        email: email,

        password: password,

      )

          .then((result) async {

        String currentUserID = result.user!.uid;

        AppConstants.currentUser.id = currentUserID;

        await getUserInfoFromFirestore(currentUserID);

        await getImageFromStorage(currentUserID);

        await AppConstants.currentUser.getMyPostingsFromFirestore();

        // Check role and navigate accordingly

        if (AppConstants.currentUser.role == "user") {

          Get.snackbar("Login Successful", "You have logged in successfully");

          Get.to(const GuestHomeScreen());

        } else if (AppConstants.currentUser.role == "admin") {

          Get.snackbar("Login Successful", "Welcome, Admin");

          Get.to(Bottomnav());

        }

      });

    } catch (e) {

      Get.snackbar("Error!!!", e.toString());

    }

  }

  Future<void> getUserInfoFromFirestore(String userID) async {

    DocumentSnapshot snapshot =

        await FirebaseFirestore.instance.collection('users').doc(userID).get();

    AppConstants.currentUser.snapshot = snapshot;

    AppConstants.currentUser.firstName = snapshot['firstName'] ?? " ";

    AppConstants.currentUser.lastName = snapshot['lastName'] ?? "";

    AppConstants.currentUser.email = snapshot['email'] ?? "";

    AppConstants.currentUser.bio = snapshot['bio'] ?? "";

    AppConstants.currentUser.city = snapshot['city'] ?? "";

    AppConstants.currentUser.country = snapshot['country'] ?? "";

    AppConstants.currentUser.isHost = snapshot['isHost'] ?? false;

    AppConstants.currentUser.role = snapshot['role'] ?? "";

  }

  getImageFromStorage(userID) async {

    if (AppConstants.currentUser.displayImage != null) {

      return AppConstants.currentUser.displayImage;

    }

    final imageDataInBytes = await FirebaseStorage.instance

        .ref()

        .child("userImages")

        .child(userID)

        .child(userID + ".png")

        .getData(1024 \* 1024);

    AppConstants.currentUser.displayImage = MemoryImage(imageDataInBytes!);

    return AppConstants();

  }

  becomeHost(String userID) async {

    userModel.isHost = true;

    Map<String, dynamic> dataMap = {

      "isHost": true,

    };

    await FirebaseFirestore.instance

        .collection("users")

        .doc(userID)

        .update(dataMap);

  }

  modifyCurrentlyHosting(bool isHosting) {

    userModel.isCurrentlyHosting = isHosting;

  }

}

- File “firebase\_options.dart”:

// File generated by FlutterFire CLI.

// ignore\_for\_file: type=lint

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

import 'package:flutter/foundation.dart'

    show defaultTargetPlatform, kIsWeb, TargetPlatform;

/// Default [FirebaseOptions] for use with your Firebase apps.

///

/// Example:

/// ```dart

/// import 'firebase\_options.dart';

/// // ...

/// await Firebase.initializeApp(

///   options: DefaultFirebaseOptions.currentPlatform,

/// );

/// ```

class DefaultFirebaseOptions {

  static FirebaseOptions get currentPlatform {

    if (kIsWeb) {

      return web;

    }

    switch (defaultTargetPlatform) {

      case TargetPlatform.android:

        return android;

      case TargetPlatform.iOS:

        return ios;

      case TargetPlatform.macOS:

        throw UnsupportedError(

          'DefaultFirebaseOptions have not been configured for macos - '

          'you can reconfigure this by running the FlutterFire CLI again.',

        );

      case TargetPlatform.windows:

        throw UnsupportedError(

          'DefaultFirebaseOptions have not been configured for windows - '

          'you can reconfigure this by running the FlutterFire CLI again.',

        );

      case TargetPlatform.linux:

        throw UnsupportedError(

          'DefaultFirebaseOptions have not been configured for linux - '

          'you can reconfigure this by running the FlutterFire CLI again.',

        );

      default:

        throw UnsupportedError(

          'DefaultFirebaseOptions are not supported for this platform.',

        );

    }

  }

  static const FirebaseOptions web = FirebaseOptions(

    apiKey: 'AIzaSyCVHERtwpEresAA5yxgM92bv9dmJQmCRzc',

    appId: '1:601096685939:web:4432afaadb1ff183ccdd06',

    messagingSenderId: '601096685939',

    projectId: 'thigiuaky-bf69f',

    authDomain: 'thigiuaky-bf69f.firebaseapp.com',

    storageBucket: 'thigiuaky-bf69f.appspot.com',

    measurementId: 'G-6MC9QCV1LH',

  );

  static const FirebaseOptions android = FirebaseOptions(

    apiKey: 'AIzaSyAag3\_gRIbRwLdE0D7e6c13RaTDs3beydo',

    appId: '1:601096685939:android:7fa9e5288c18d5e8ccdd06',

    messagingSenderId: '601096685939',

    projectId: 'thigiuaky-bf69f',

    storageBucket: 'thigiuaky-bf69f.appspot.com',

  );

  static const FirebaseOptions ios = FirebaseOptions(

    apiKey: 'AIzaSyCA3-J6D3IslqvXi9RU0\_dGNymgyYlZZnA',

    appId: '1:601096685939:ios:adf0b3e9067c996eccdd06',

    messagingSenderId: '601096685939',

    projectId: 'thigiuaky-bf69f',

    storageBucket: 'thigiuaky-bf69f.appspot.com',

    iosBundleId: 'com.example.bookinghotel',

  );

}

- File “global.dart”:

import 'package:guest\_hotel/model/posting\_model.dart';

import 'package:guest\_hotel/view\_model/posting\_view\_model.dart';

import 'package:guest\_hotel/view\_model/user\_view\_model.dart';

PostingModel postingModel = PostingModel();

UserViewModel userViewModel = UserViewModel();

PostingViewModel postingViewModel = PostingViewModel();

String paymentResult = "";

double? bookingPrice = 0;

String hostID = "";

- File “main.dart”:

import 'package:guest\_hotel/view/spalash\_screen.dart';

import 'package:flutter/material.dart';

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

import 'package:get/get.dart';

import 'firebase\_options.dart';

// ...

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(

    options: DefaultFirebaseOptions.currentPlatform,

  );

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return GetMaterialApp(

      title: 'Booking Hotel',

      theme: ThemeData(

        appBarTheme: const AppBarTheme(

          iconTheme: IconThemeData(color: Colors.white),

          color: Colors.black,

        ),

        colorScheme: ColorScheme.fromSeed(seedColor: Colors.green),

        useMaterial3: true,

      ),

      home: const SpalashScreen(),

    );

  }

}