Question 1:

You are tasked with creating a mobile application for a hotel reservation system. The system should propose to the users alternative choices in case what they requested is not available.

1. Explain the different types of prototypes and show the above application model with a low fidelity prototype.

- Low fidelity prototypes are basic representations of a design created using simple tools like sketches or paper. They focus on overall concept and structure and are useful for quick iterations and gathering feedback.

- Medium fidelity prototypes are more detailed, created using digital tools, and incorporate design elements like colors and basic interactivity. They help refine the user experience and test specific features.

- High fidelity prototypes closely resemble the final product, with realistic visuals, interactions, and functionality. They are created using advanced software and allow for comprehensive usability testing and stakeholder approval.

For low fidelity we can:

Sketch : draw the layout of the application by hand

Sketching is quick, disposable and clear and it can be redrawn as many times to explore different paths **I drew some sketches on the questions screen**

We can use storyboards and scenarios which is a series of sketches showing the progression through a task

Brainstorm different representations

Choose a representation

Rough out an interface design

Task certain walkthrough and redesign

Screen 1: Welcome Screen

sign in or sign up.

Screen 2: Sign In/Sign Up

Screen 3: Home Screen

Display a search bar where users can input the desired location, check-in and check-out dates, and the number of guests.

Provide a "Search" button to initiate the search.

Screen 4: Search Results

Show a list of available hotels with relevant details such as name, rating, and price.

Allow users to expand each hotel item to view additional information.

Include a "Book Now" button next to each hotel for easy reservation.

Screen 5: No Results/Alternative Choices

If no hotels match the search criteria, present a message suggesting alternative choices.

Offer options like nearby hotels, different dates, or similar alternatives.

Screen 6: Hotel Details

Provide comprehensive information about the selected hotel, including amenities, photos, reviews, and a map showing the location.

Include a "Book Now" button for users to proceed with the reservation.

Screen 7: Reservation Form

Include fields to enter guest details such as name, contact information, and special requests.

Allow users to select room type, add extra services, and customize the reservation.

Display a "Confirm Reservation" button to finalize the booking.

Screen 8: Reservation Confirmation

Show a confirmation message with detailed reservation information and a unique booking reference number.

Provide an option to view or download the reservation confirmation.

Screen 9: User Profile

Display the user's profile information, including name, contact details, and reservation history.

Allow users to edit their profile details, change their password, or sign out.

b-

Comparison with Booking.com mobile app:

The proposed hotel reservation system mobile app focuses on essential features, while Booking.com offers a comprehensive set of features and polished user experience.

Booking.com app provides a user-friendly interface, detailed hotel listings, and intuitive navigation.

Both prioritize the booking process, including guest details, room selection, special requests, and secure payment options.

Booking.com app offers additional features like personalized user accounts, favorite hotel saving, booking history, and customer support options.

Question 2:

**The login screen:**

import 'package:flutter/material.dart';

class LoginScreen extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Login'),

),

body: Padding(

padding: EdgeInsets.all(16.0),

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

crossAxisAlignment: CrossAxisAlignment.stretch,

children: <Widget>[

TextFormField(

decoration: InputDecoration(

labelText: 'Email',

),

),

SizedBox(height: 16.0),

TextFormField(

obscureText: true,

decoration: InputDecoration(

labelText: 'Password',

),

),

SizedBox(height: 32.0),

RaisedButton(

child: Text('Login'),

onPressed: () {

// Perform login logic here

},

),

],

),

),

);

}

}

**home screen**

void main() {

runApp(MaterialApp(

title: 'Login App',

home: LoginScreen(),

));

}

import 'package:flutter/material.dart';

class HomeScreen extends StatefulWidget {

@override

\_HomeScreenState createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

DateTime selectedDate = DateTime.now();

List<String> rooms = ['Room 1', 'Room 2', 'Room 3', 'Room 4'];

Future<void> selectDate(BuildContext context) async {

final DateTime picked = await showDatePicker(

context: context,

initialDate: selectedDate,

firstDate: DateTime(2020, 1),

lastDate: DateTime(2101),

);

if (picked != null && picked != selectedDate) {

setState(() {

selectedDate = picked;

});

}

}

void navigateToReservationScreen(BuildContext context) {

Navigator.pushNamed(context, '/reservation');

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Home'),

),

body: Column(

children: <Widget>[

ListTile(

title: Text("Selected Date: ${selectedDate.toLocal()}"),

trailing: Icon(Icons.keyboard\_arrow\_down),

onTap: () => selectDate(context),

),

Expanded(

child: ListView.builder(

itemCount: rooms.length,

itemBuilder: (context, index) {

final room = rooms[index];

return ListTile(

title: Text(room),

onTap: () => navigateToReservationScreen(context),

);

},

),

),

],

),

);

}

}

import 'package:flutter/material.dart';

class ReservationScreen extends StatefulWidget {

@override

\_ReservationScreenState createState() => \_ReservationScreenState();

}

class \_ReservationScreenState extends State<ReservationScreen> {

late TimeOfDay startTime;

late TimeOfDay endTime;

final \_formKey = GlobalKey<FormState>();

final \_descriptionController = TextEditingController();

@override

void initState() {

super.initState();

startTime = TimeOfDay.now();

endTime = TimeOfDay.now().replacing(hour: startTime.hour + 1);

}

Future<TimeOfDay?> \_selectTime(BuildContext context, TimeOfDay initialTime) async {

return await showTimePicker(

context: context,

initialTime: initialTime,

);

}

void \_handleStartTimeButton() async {

final picked = await \_selectTime(context, startTime);

if (picked != null && picked != startTime) {

setState(() {

startTime = picked;

});

}

}

void \_handleEndTimeButton() async {

final picked = await \_selectTime(context, endTime);

if (picked != null && picked != endTime) {

setState(() {

endTime = picked;

});

}

}

void \_saveReservation() {

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

// Save reservation

final description = \_descriptionController.text;

print('Description: $description');

print('Start Time: ${startTime.format(context)}');

print('End Time: ${endTime.format(context)}');

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Reservation'),

),

body: Padding(

padding: EdgeInsets.all(16.0),

child: Form(

key: \_formKey,

child: Column(

children: <Widget>[

TextFormField(

controller: \_descriptionController,

decoration: InputDecoration(labelText: 'Purpose/Description'),

),

ElevatedButton(

onPressed: \_handleStartTimeButton,

child: Text('Start Time: ${startTime.format(context)}'),

),

ElevatedButton(

onPressed: \_handleEndTimeButton,

child: Text('End Time: ${endTime.format(context)}'),

),

ElevatedButton(

onPressed: \_saveReservation,

child: Text('Reserve'),

),

],

),

),

),

);

}

}

**A reservation room that shows a list of the user’s upcoming reservations including the room name reservation date and time and purpose**

import 'package:flutter/material.dart';

class ReservationScreen extends StatefulWidget {

@override

\_ReservationScreenState createState() => \_ReservationScreenState();

}

class \_ReservationScreenState extends State<ReservationScreen> {

late TimeOfDay startTime;

late TimeOfDay endTime;

final \_formKey = GlobalKey<FormState>();

final \_descriptionController = TextEditingController();

@override

void initState() {

super.initState();

startTime = TimeOfDay.now();

endTime = TimeOfDay.now().replacing(hour: TimeOfDay.now().hour + 1);

}

Future<TimeOfDay?> selectTime(BuildContext context, TimeOfDay initialTime) async {

final picked = await showTimePicker(

context: context,

initialTime: initialTime,

);

return picked;

}

void saveReservation() {

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

// Perform reservation saving logic here

final description = \_descriptionController.text;

// Print reservation details for demonstration

print('Description: $description');

print('Start Time: ${startTime.format(context)}');

print('End Time: ${endTime.format(context)}');

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Reservation'),

),

body: Padding(

padding: EdgeInsets.all(16.0),

child: Form(

key: \_formKey,

child: Column(

children: <Widget>[

TextFormField(

controller: \_descriptionController,

decoration: InputDecoration(labelText: 'Purpose/Description'),

validator: (value) {

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

return 'Please enter a description';

}

return null;

},

),

ElevatedButton(

onPressed: () async {

final picked = await selectTime(context, startTime);

if (picked != null && picked != startTime) {

setState(() {

startTime = picked;

});

}

},

child: Text('Start Time: ${startTime.format(context)}'),

),

ElevatedButton(

onPressed: () async {

final picked = await selectTime(context, endTime);

if (picked != null && picked != endTime) {

setState(() {

endTime = picked;

});

}

},

child: Text('End Time: ${endTime.format(context)}'),

),

ElevatedButton(

onPressed: saveReservation,

child: Text('Reserve'),

),

],

),

),

),

);

}

}

**Question 3:**

• Initial error: The class Todo contains errors in field initialization. There are two possible solutions:

Include "required" in the constructor of the Todo class, such as: Todo({required this.title, required this.isDone}).

Make the "title" and "isDone" fields nullable by adding a question mark to their types, like: String? and bool?

• Subsequent error: The issue lies with the constructor of the TodoListApp class. The solutions are as follows:

Add "todo" as a required argument in the TodoListApp constructor and pass the value of the "todos" list in the main function of the file.

Make the "todos" list nullable.

• Third error: The error lies within the "onChanged" function of the ListViewBuilder. To ensure that the checkbox triggers an update, the "onChanged" function should invoke the setState function of Flutter's StatefulWidget. Consequently, TodoListApp should extend StatefulWidget instead of StatelessWidget. Additionally, within the setState function, the value of the todoListItem should be updated.

• The last minor syntax error is regarding the "value" argument in onChanged, which should be nullable. Therefore, we should use bool? instead of just bool. Consequently, the last line of code should be modified to => todos[index].isDone = value! (adding an exclamation mark to assert that the value is not null).The correct code is:

import 'package:flutter/material.dart';

class Todo {

String title;

bool isDone;

Todo({

required this.title,

required this.isDone,

});

}

class TodoListApp extends StatefulWidget {

TodoListApp();

@override

\_TodoListAppState createState() => \_TodoListAppState();

}

class \_TodoListAppState extends State<TodoListApp> {

List<Todo> todos;

@override

void initState() {

super.initState();

todos = [

Todo(title: 'Buy groceries', isDone: false),

Todo(title: 'Do laundry', isDone: true),

Todo(title: 'Walk the dog', isDone: false),

];

}

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Todo List',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: Scaffold(

appBar: AppBar(

title: Text('Todo List'),

),

body: ListView.builder(

itemCount: todos.length,

itemBuilder: (BuildContext context, int index) {

return ListTile(

title: Text(todos[index].title),

trailing: Checkbox(

value: todos[index].isDone,

onChanged: (bool? value) {

if (value != null) {

setState(() {

todos[index].isDone = value;

});

}

},

),

);

},

),

),

);

}

}

void main() {

runApp(TodoListApp());

}B- No errors