**Assignment Solutions**

## Assignment-1 Marks: 10

Q-1: Describe the Embedder layer of Flutter Architecture

**Solutions:**

Flutter layered architecture based on Framework (Top Layer), Engine (Middle Layer), and Embedder (last or down layer).  A platform-specific embedder provides an entry point; coordinates with the underlying operating system for access to services like rendering surfaces, accessibility, and input; and manages the message event loop. The embedder is written in a language that is appropriate for the platform: currently Java and C++ for Android, Objective-C/Objective-C++ for iOS and macOS, and C++ for Windows and Linux. Using the embedder, Flutter code can be integrated into an existing application as a module, or the code may be the entire content of the application. Flutter includes a number of embedders for common target platforms.

## Assignment-2 Marks: 10

Q-1: Describe the difference between normal Class and Interface

**Solutions:**

A class describes the attributes and behaviours of an object. It contains abstract methods, concrete methods. Members of a class can be public, private, protected or default.

An interface contains behaviours that a class implements. An interface contains only abstract methods. All the members of the interface are public by default.

Dart does not have a syntax for declaring interfaces. Class declarations are themselves interfaces in Dart. Classes should use the implements keyword to be able to use an interface. It is mandatory for the implementing class to provide a concrete implementation of all the functions of the implemented interface. In other words, a class must redefine every function in the interface it wishes to implement.

**Implementing an Interface**

*Class identifier implements interface\_name*

In the following program, we are declaring a class Printer. The ConsolePrinter class implements the implicit interface declaration for the Printer class. The main function creates an object of the ConsolePrinter class using the new keyword. This object is used to invoke the function print\_data defined in the ConsolePrinter class.

|  |
| --- |
| void main() {  ConsolePrinter cp= new ConsolePrinter();  cp.print\_data();  }  class Printer {  void print\_data() {  print(“Printing Data”);  }  }  class ConsolePrinter implements Printer {  void print\_data() {  print(“Printing to Console”);  }  } |

**Implementing Multiple Interfaces**

A class can implement multiple interfaces. The interfaces are separated by a comma. The syntax for the same is given below the Flutter application.

## Assignment-3 Marks: 10

Q-1: **Implement the following widgets in flutter Application.**

1. **Icons**: use Icons Class for implementing different icons
2. **Image:** Use Images widget in your project that will show image from
3. Direct from Network by providing URL
4. Access image from project folder. use Assets concepts here.
5. **TextField :** Use this widget in your form designing and know how to get value from textfield.
6. **Buttons :** Use multiple type of button in you code

## Solution

**Icons**: use Icons Class for implementing different icons

1. Widget build(BuildContext context) {
2. return MaterialApp(
3. home: Scaffold(
4. appBar: AppBar(
5. title: Text(
6. 'Triocders',
7. style: TextStyle(
8. color: Colors.black,
9. fontSize: 20.0,
10. fontWeight: FontWeight.bold,
11. ),
12. ),
13. leading: Icon(
14. Icons.menu,
15. ),
16. backgroundColor: Colors.red.shade400,
17. ),

**Image:** Use Images widget in your project that will show image from

1. Direct from Network by providing URL
2. Access image from project folder. use Assets concepts here.

Code:

1. body:Column(children: [
2. Image.asset('assets/1.png', height: 300.0,),
3. Image.network('https://images.unsplash.com/photo-1637181932973-2f156f331d74?ixid=MnwxMjA3fDB8MHxlZGl0b3JpYWwtZmVlZHw5fHx8ZW58MHx8fHw%3D&ixlib=rb-1.2.1&auto=format&fit=crop&w=500&q=60',
4. height: 300.0,width: 600.0,),
5. ]),

**TextField :** Use this widget in your form designing and know how to get value from textfield.

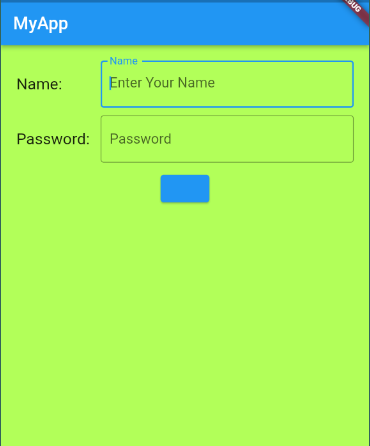
1. body:Column(children: [
2. Center(
3. child:
4. Text(
5. 'User Name',
6. style: TextStyle(fontSize:22.0 ),),
7. ),
8. TextField(
9. decoration: InputDecoration(border:null,hintText:"Enter Email"),
10. ),
11. Center(
12. child:
13. Text(
14. 'Password',
15. style: TextStyle(fontSize:22.0 ),),
16. ),
17. TextField(
18. decoration: InputDecoration(border:null,hintText:"Enter Password"),
19. ),
21. ElevatedButton(child:Text("Login Me"),onPressed:(){
23. }),
25. ],
27. ),

**Buttons :** Use multiple type of button in you code

1. ElevatedButton(child:Text("Login Me"),onPressed:(){
3. }),
4. floatingActionButton: FloatingActionButton(
5. onPressed: (){},
6. child:Text(
7. 'Click ',
8. style: TextStyle(
9. color: Colors.black,
10. ),
11. ),
12. backgroundColor: Colors.red[600],
13. ),

## Assignment-4 Marks: 10

Q-1: Design the following Login interface

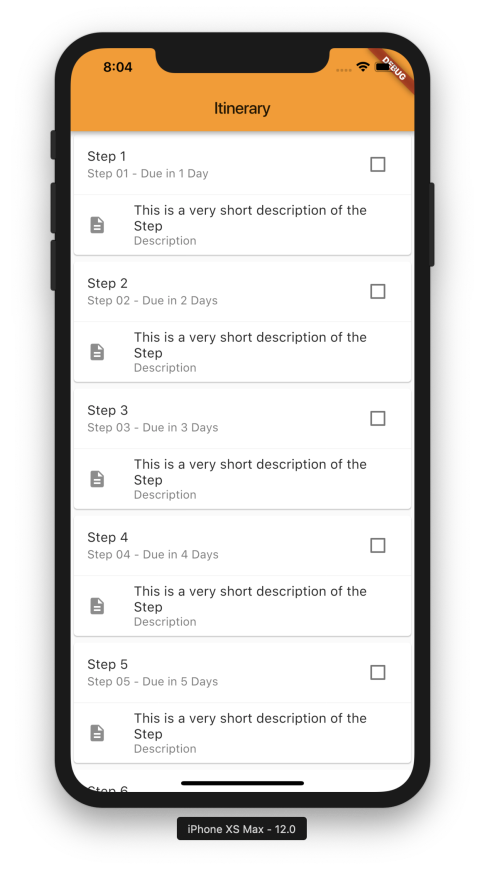


## Solution

|  |
| --- |
| import 'package:flutter/cupertino.dart'; import 'package:flutter/material.dart'; void main() => runApp(MaterialApp(  home: Scaffold(  appBar: AppBar(  title: Text('MyApp'),  ),  body: Padding(  padding: EdgeInsets.all(20),  child: Column(  children: [  //------------- First Row ----------------------------  Container (  margin: EdgeInsets.fromLTRB(0, 0, 0, 10),  child: Row(  children: <Widget>[  Expanded(  flex: 1,  child: Text('Name:',  style: TextStyle(fontSize: 18,),  )   ),  Expanded(  flex: 3,  child: TextField(  obscureText: true,  decoration: InputDecoration(  border: OutlineInputBorder(),  labelText: 'Name',  hintText: 'Enter Your Name'  onChanged  ),  ),  ),  ]  ),  ),  *//----------------Second Row-------------------------* Container(  margin: EdgeInsets.fromLTRB(0, 0, 0, 10),  child: Row(  children: <Widget>[  Expanded(  flex: 1,  child: Text('Password:',  style: TextStyle(fontSize: 18,),  )  ),  Expanded(  flex: 3,  child: TextField(  decoration: InputDecoration(  border: OutlineInputBorder(),  labelText: 'Password',  hintText: 'Enter Your Name'  ),  ),  ),  ]  ),  ),  *//-----------------------Third Row--------------------* Container(  margin: EdgeInsets.fromLTRB(0, 0, 0, 10),  child: ElevatedButton(  onPressed: () { },  child: Text('Sign In'),  style: ElevatedButton.*styleFrom*(  onPrimary: Colors.*white*,  shadowColor: Colors.*blue*,  onSurface: Colors.*grey*,  elevation: 8,  ),  )  ],  )  ),  backgroundColor: Colors.*lightGreenAccent*,  ),  ), ); |

## Assignment-5 Marks: 10

**Q-1: Design the following UI Interface**



|  |
| --- |
| import 'package:flutter/cupertino.dart'; import 'package:flutter/material.dart';  void main() => runApp(MaterialApp(  title: 'Itinerary',  home: Scaffold(  body: SafeArea(child:  Container(  padding: EdgeInsets.fromLTRB(15, 15, 25, 15),   child: Column(  children: [   Card(  elevation: 5,  child: Column(  children: [  ListTile(  title: Text('Step 1'),  subtitle: Text('Step 01 - Due in 2 Day'),  trailing: Checkbox(value: false, onChanged: (bool? value) { },),  ),  Divider(color: Colors.*black12*),  ListTile(  title: Text('This is very Short Description of the Step'),  subtitle: Text('Description'),  leading: Icon(Icons.*file\_copy\_sharp*),  ),  ]  ),  ),  Card(  elevation: 5,  child: Column(  children: [  ListTile(  title: Text('Step 2'),  subtitle: Text('Step 02 - Due in 2 Day'),  trailing: Checkbox(value: false, onChanged: (bool? value) { },),  ),  Divider(color: Colors.*black12*),  ListTile(  title: Text('This is very Short Description of the Step'),  subtitle: Text('Description'),  leading: Icon(Icons.*file\_copy\_sharp*),  ),  ]  ),  ),   Card(  elevation: 5,  child: Column(  children: [  ListTile(  title: Text('Step 3'),  subtitle: Text('Step 03 - Due in 3 Day'),  trailing: Checkbox(value: false, onChanged: (bool? value) { },),  ),  Divider(color: Colors.*black12*),  ListTile(  title: Text('This is very Short Description of the Step'),  subtitle: Text('Description'),  leading: Icon(Icons.*file\_copy\_sharp*),  ),  ]  ),  ),   Card(  elevation: 5,  child: Column(  children: [  ListTile(  title: Text('Step 4'),  subtitle: Text('Step 04 - Due in 4 Day'),  trailing: Checkbox(value: false, onChanged: (bool? value) { },),  ),  Divider(color: Colors.*black12*),  ListTile(  title: Text('This is very Short Description of the Step'),  subtitle: Text('Description'),  leading: Icon(Icons.*file\_copy\_sharp*),  ),  ]  ),  ),   Card(  elevation: 5,  child: Column(  children: [  ListTile(  title: Text('Step 5'),  subtitle: Text('Step 05 - Due in 5 Day'),  trailing: Checkbox(value: false, onChanged: (bool? value) { },),  ),  Divider(color: Colors.*black12*),  ListTile(  title: Text('This is very Short Description of the Step'),  subtitle: Text('Description'),  leading: Icon(Icons.*file\_copy\_sharp*),  ),  ]  ),  ),  ],  ),    ),   ),     )  )); |

## Assignment-6 Marks: 10

Q-1: Write the difference between Hero and Curved animation with examples

A hero animation is a type of animation where an element of one screen **flies** to a new screen when the app goes to the next page.

The curved animation is very useful when you need to apply a non-linear curve with an animation object. Thus, it defines the animation's progress as a non-linear curve.

Curved Animation (parent: animation Controller, curve: Curves.bounceOut));

## Assignment:7 Marks: 10

Q-1: Design the form of your own choice in flutter for your mobile application. It should contain all type of controls like TextFields, Buttons, Checkboxes, RadioButton etc. User will submit data from Form 1 and system should redirect to Form 2. All the submitted data will be shown on Form 2. Please implement App state techniques to achieve this.

|  |
| --- |
| import 'package:flutter/material.dart';  import '../models/user.dart';  class HomeMaterial extends StatefulWidget {  @override  \_HomeMaterialState createState() => \_HomeMaterialState();  }  class \_HomeMaterialState extends State <homematerial>{  final \_formKey = GlobalKey<formstate>();  final \_user = User();  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(title: Text('Profile')),  body: Container(  padding:  const EdgeInsets.symmetric(vertical: 16.0, horizontal: 16.0),  child: Builder(  builder: (context) => Form(  key: \_formKey,  child: Column(  crossAxisAlignment: CrossAxisAlignment.stretch,  children: [  TextFormField(  decoration:  InputDecoration(labelText: 'First name'),  validator: (value) {  if (value.isEmpty) {  return 'Please enter your first name';  }  },  onSaved: (val) =>  setState(() => \_user.firstName = val),  ),  TextFormField(  decoration:  InputDecoration(labelText: 'Last name'),  validator: (value) {  if (value.isEmpty) {  return 'Please enter your last name.';  }  },  onSaved: (val) =>  setState(() => \_user.lastName = val)),  Container(  padding: const EdgeInsets.fromLTRB(0, 50, 0, 20),  child: Text('Subscribe'),  ),  SwitchListTile(  title: const Text('Monthly Newsletter'),  value: \_user.newsletter,  onChanged: (bool val) =>  setState(() => \_user.newsletter = val)),  Container(  padding: const EdgeInsets.fromLTRB(0, 50, 0, 20),  child: Text('Interests'),  ),  CheckboxListTile(  title: const Text('Cooking'),  value: \_user.passions[User.PassionCooking],  onChanged: (val) {  setState(() =>  \_user.passions[User.PassionCooking] = val);  }),  CheckboxListTile(  title: const Text('Traveling'),  value: \_user.passions[User.PassionTraveling],  onChanged: (val) {  setState(() => \_user  .passions[User.PassionTraveling] = val);  }),  CheckboxListTile(  title: const Text('Hiking'),  value: \_user.passions[User.PassionHiking],  onChanged: (val) {  setState(() =>  \_user.passions[User.PassionHiking] = val);  }),  Container(  padding: const EdgeInsets.symmetric(  vertical: 16.0, horizontal: 16.0),  child: RaisedButton(  onPressed: () {  final form = \_formKey.currentState;  if (form.validate()) {  form.save();  \_user.save();  \_showDialog(context);  }  },  child: Text('Save'))),  ])))));  }  \_showDialog(BuildContext context) {  Scaffold.of(context)  .showSnackBar(SnackBar(content: Text('Submitting form')));  }  }</formstate></homematerial> |

## Assignment-8 Marks: 10

Q-1: Write the configuration steps required to implement Firebase Cloud Messaging Service.

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps.It is built on Google’s infrastructure and provides developers with a variety of tools and services to help them develop quality apps, grow their user base.

Firebase Cloud Messaging is a service offered by Firebase which lets you send these notifications to your users. You can set up various configurations to send different notifications to different audiences based on time and routine.

Here we are going to learn how to add push notifications to a Flutter app using Firebase Cloud Messaging.

Push Notifications are a sort of pop-up messaging medium that alerts app users to what's going on in the app.

# Here are Steps to configure Push Notifications

## Step-1: Create a Flutter Project

## Step-2 Integrate Firebase Configuration with Flutter

## Step 3: Register Firebase to Your Android App

## Step 4: Add Firebase Configurations to Native Files in your Flutter Project

First in our **root-level (project-level)** Gradle file (**android/build.gradle**), we need to add rules to include the Google Services Gradle plugin. We need to check if the following configurations are available or not:

|  |
| --- |
| buildscript {  repositories {  // Check that you have the following line (if not, add it):  google() // Google's Maven repository  }  dependencies {  ...  // Add this line  classpath 'com.google.gms:google-services:4.3.4'  }  }  allprojects {  ...  repositories {  // Check that you have the following line (if not, add it):  google() // Google's Maven repository  ...  }  } |

## Step 5: Integrate Firebase Messaging with Flutter

|  |
| --- |
| dependencies {  implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin\_version"  implementation 'com.google.firebase:firebase-messaging:20.1.0'  } |

Next, we need to add an action and a category as an intent-filter within the activity tag in the **./android/app/src/main/AndroidManifest.xm**l file:

|  |
| --- |
| <intent-filter>  <action android:name="FLUTTER\_NOTIFICATION\_CLICK" />  <category android:name="android.intent.category.DEFAULT" />  </intent-filter> |

Now, we need to create a Java file called **Application.java** in the path **/android/app/src/main/java/<app-organization-path>**.

Then, we need to add the code from the following code snippet inside it:

|  |
| --- |
| package io.flutter.plugins.pushNotification;  import io.flutter.app.FlutterApplication;  import io.flutter.plugin.common.PluginRegistry;  import io.flutter.plugin.common.PluginRegistry.PluginRegistrantCallback;  import io.flutter.plugins.GeneratedPluginRegistrant;  import io.flutter.plugins.firebasemessaging.FirebaseMessagingPlugin;  import io.flutter.plugins.firebasemessaging.FlutterFirebaseMessagingService;  public class Application extends FlutterApplication implements PluginRegistrantCallback {  @Override  public void onCreate() {  super.onCreate();  FlutterFirebaseMessagingService.setPluginRegistrant(this);  }  @Override  public void registerWith(PluginRegistry registry) {  FirebaseMessagingPlugin.registerWith(registry.registrarFor("io.flutter.plugins.firebasemessaging.FirebaseMessagingPlugin"));  }  } |

Now, we need to assign this Application activity to the application tag of the **AndroidManifest.xml** file as shown in the code snippet below:

|  |
| --- |
| <application  android:name=".Application" |

## Step 6: Install the Firebase Messaging Package

We need to add the following line of code to the dependencies option:

|  |
| --- |
| firebase\_messaging: ^7.0.3 |

## Step 7: Implement a Simple UI Screen

Now, inside the MyHomePage stateful widget class of the **main.dart** file, we need to initialize the FirebaseMessaging instance and some constants as shown in the code snippet below:

|  |
| --- |
| String messageTitle = "Empty";  String notificationAlert = "alert";  FirebaseMessaging \_firebaseMessaging = FirebaseMessaging(); |

The messageTitle variable will receive the notification message title and notificationAlert will be assigned the action that's been completed once the notification comes up.

Now, we need to apply these variables to the build function inside the Scaffold widget body as shown in the code snippet below:

|  |
| --- |
| Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: Text(widget.title),  ),  body: Center(  child: Column(  mainAxisAlignment: MainAxisAlignment.center,  children: <Widget>[  Text(  notificationAlert,  ),  Text(  messageTitle,  style: Theme.of(context).textTheme.headline4,  ),  ],  ),  ),  );  } |