Mobile Applications and Web Development - IS4904(Practical)		
Student Name:	**Student ID:**	**Section:**
Assignment 8 (Flutter Application Development)		
Date: 5th May 2024	**Max Points:**	

Part 1:

Task: Reverse Engineering and Flowchart Creation

Objective

The goal of this task is to enhance your understanding of reverse engineering and flowchart design. You will analyze a simple Flutter mini application (in the attachments) , identify its components, and create flowcharts to represent its processes.

Instructions

1. **Application Overview**:
   - You will receive a Flutter mini application (provided as an attachment).
   - Study the application carefully to understand its functionality, user interface, and behavior.
2. **Reverse Engineering**:
   - Reverse engineer the application by examining its code.
   - Identify the key components, such as widgets, state management, and navigation.
   - Understand how data flows within the app.
3. **Flowchart Creation**:
   - Create flowcharts to describe the following processes within the application:
     - The flow of data and processes.
     - How data will be processed and displayed

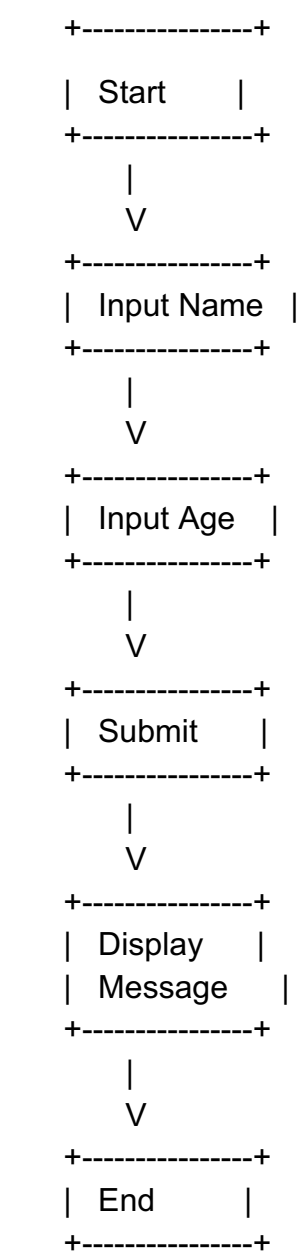
Additional Notes

- You can use any tool or software to create the flowcharts (e.g., draw.io, Lucidchart, pen and paper).
- Be concise and clear in your flowchart design.
- Consider edge cases and error handling in your analysis.

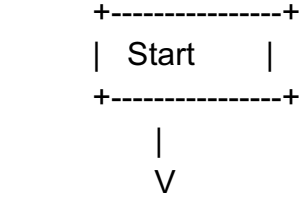
Attachment

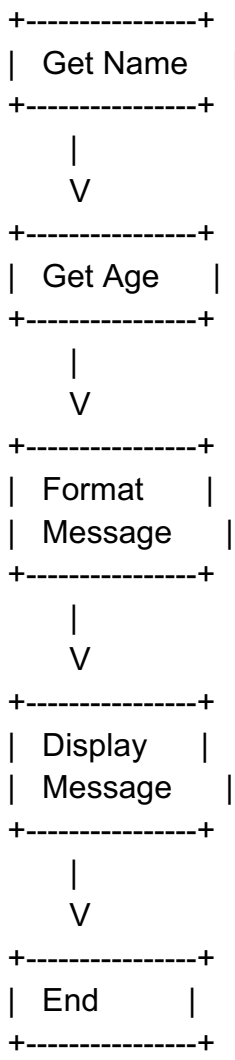
Download the Flutter Mini Application

1. Input Process Flowchart:

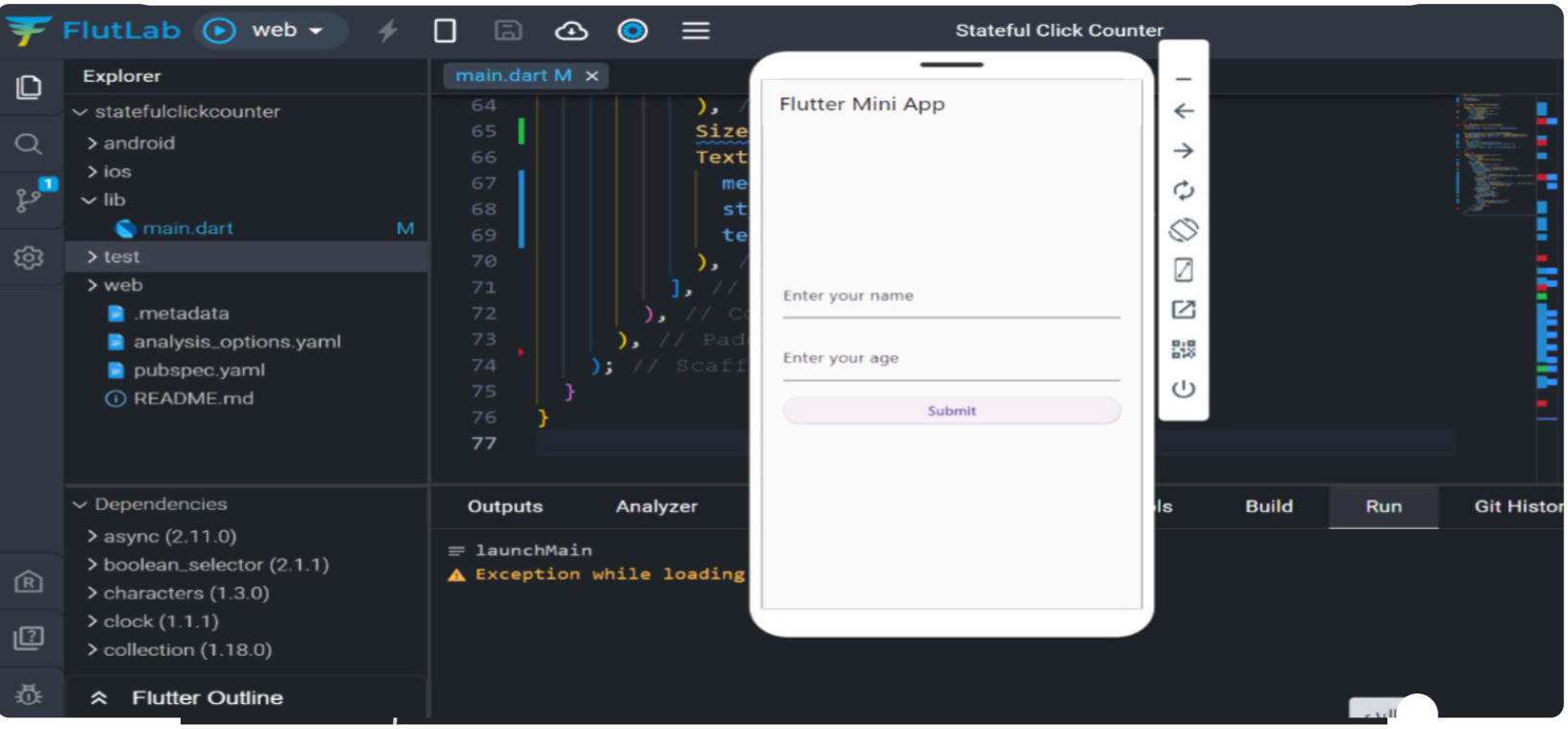


2. Data Processing and Display Flowchart




Get Name
Get Age
Format Message
Display Message
End

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Mini App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  TextEditingController nameController = TextEditingController();
  TextEditingController ageController = TextEditingController();
  String message = '';

  void _submit() {
    String name = nameController.text;
    int age = int.tryParse(ageController.text) ?? 0;
    setState(() {
      message = 'Hello, $name! You are $age years old.';
    });
  }
```


FlutLab
web
Stateful Click Counter
Explorer
statefulclickcounter
android
ios
lib
main.dart
M
test
web
.metadata
analysis_options.yaml
pubspec.yaml
README.md
main.dart M
64
65
66
67
68
69
70
71
72
73
74
75
76
77
Flutter Mini App
Enter your name
Enter your age
Submit
Dependencies
async (2.11.0)
boolean_selector (2.1.1)
characters (1.3.0)
clock (1.1.1)
collection (1.18.0)
Flutter Outline
Outputs
Analyzer
Build
Run
launchMain
Exception while loading

Part II:

Task 1: Image Display and Random Order

1. Create a **StatefulWidget** named `ImageDisplayApp`.
2. In the `build` method of the widget, display an image centered on the screen.
3. Initially, use the first image from the assets folder.
4. Add a button below the image.
5. When the user presses the button, change the displayed image to the next one in the sequence.
6. If the user reaches the last image, the image should cycle back to the first one.

Task 2: Display Images in Random Order

1. Reformat the code from Task 1.
2. Instead of displaying images sequentially, display them in a random order.
3. Every time the user presses the button, the images should shuffle randomly.

Additional Notes

```
import 'package:flutter/material.dart';
import 'dart:math';

void main() {
  runApp(ImageDisplayApp());
}

class ImageDisplayApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Image Display App',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: ImageDisplay(),
    );
  }
}

class ImageDisplay extends StatefulWidget {
  @override
  _ImageDisplayState createState() => _ImageDisplayState();
}

class _ImageDisplayState extends State<ImageDisplay> {
  int _currentIndex = 0;
  List<String> _images = [
    'assets/image1.jpg',
    'assets/image2.jpg',
    'assets/image3.jpg',
    'assets/image4.jpg',
    'assets/image5.jpg',
  ];

  void _nextImage() {
```

```
void _nextImage() {
  setState(() {
    _currentIndex = (_currentIndex + 1) % _images.length;
  });
}

void _shuffleImages() {
  setState(() {
    _images.shuffle();
    _currentIndex = 0;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Image Display App'),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Image.asset(
            _images[_currentIndex],
            width: 300,
            height: 300,
          ),
          SizedBox(height: 20.0),
          ElevatedButton(
            onPressed: _shuffleImages,
            child: Text('Shuffle Images'),
          ),
          SizedBox(height: 20.0),
          ElevatedButton(
            onPressed: _nextImage,
```

```
            child: Text('Next Image'),
          ),
        ],
      ),
    ),
  );
 }
}
```