BINUS University

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Academic Career:**  ***Undergraduate / ~~Master / Doctoral / International / BASE / BINUS~~ ~~Online~~\*)*** | | | | | **Class Program:**  ***Regular / ~~Global Class~~\*)*** | | |
| ☒ **Mid Exam**   * **Final Exam** | * **Compact Term Exam** * **Others Exam :** | | | | **Term : Odd / ~~Even~~** / **~~Compact~~ \*)**  **Period (Only for *BINUS Online*): 1 / 2 \*)** | | |
| ☒ **Kemanggisan**   * **Alam Sutera** * **Bekasi** | * **Senayan** * **Bandung** * **Malang** | |  | * **Semarang** | **Academic Year :**  **2024/2025** | | |
| Exam Type\* | : | ~~Onsite~~ / Online (1 week) | | | Faculty / Dept. | : | Engineering / Computer Engineering |
| Day / Date\*\* | : | Friday / 08 November 2024 | | | Code - Course | : | CPEN6222010 - Mobile  Application Development for Engineering |
| Time\*\* | : | 13:00 |  |  | Code - Lecturer | : | D5855 - Johannes, S.Kom., M.T. |
| Exam Specification\*\*\* | : | * Open Book * Close Book * Open E-Book | | * Open Notes * Submit Project * Oral Test | BULC (Only for BINUS Online) | : |  |
|  |  | Class | : | **LB40** |
| Equipment\*\*\* | : |  |  |  | Student ID \*\*\* | : | **2702407514** |
| * Exam Booklet * Calculator * Dictionary | ☒ Laptop   * Tablet * Smartphone | | * Drawing Paper – A3 * Drawing Paper – A2 * Notes | | Name \*\*\* | : | **Elvin Aurelius Yamin** |
| Signature \*\*\* | : | flvin |
| 🞸) *Strikethrough the unnecessary items \*\*) For Online Exam, this is the due date* \*\*\*) *Only for Onsite Exam* | | | | | | | |
| ***Please insert the test paper into the exam booklet and submit both papers after the test.***  ***The penalty for CHEATING is DROP OUT!*** | | | | | | | |

Learning Outcome for

☒ Mid Exam ☐ Final Exam

LO 1: Explain fundamental concepts of software design and mobile application development LO 2: Solve problem related to software design and mobile application development

# A screenshot of a phone Description automatically generatedConcept (30%)

* 1. [LO1 – 10 Points]

Explain what is the difference between Stateless and Stateful widget?

When do you use them? Provide an example for each.

* Stateless Widgets are immutable; once they are created, they cannot change. They're ideal for UI components that don’t require interaction or state changes.
  + Example: A simple text display widget or a static button.

class Greeting extends StatelessWidget {

final String text;

Greeting(this.text);

@override

Widget build(BuildContext context) {

return Text(text);

}

}

* Stateful Widgets maintain a mutable state, allowing them to update based on user interaction or other events. These widgets are used for dynamic elements in the UI.
  + Example: A counter that increments on button press.

class Counter extends StatefulWidget {

@override

\_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

int count = 0;

void increment() {

setState(() {

count++;

});

}

@override

Widget build(BuildContext context) {

return Column(

children: [

Text('Count: $count'),

ElevatedButton(onPressed: increment, child: Text('Increment')),

],

);

}

}

* 1. [LO1 – 10 Points]

Figure 1 - Flutter layout Demo

From Flutter layout tutorial page [here,](https://docs.flutter.dev/ui/layout/tutorial) draw a diagram of the widget tree for that UI.

See Figure 1.

Column

├── AppBar

│ └── Text("Flutter layout demo")

├── Image.asset(‘images/lake.jpg’)

├── Padding

│ └── Column

│ ├── Text("Oeschinen Lake Campground")

│ ├── Text("Kandersteg, Switzerland")

│ ├── Row

│ │ ├── Icon(Icons.star, color: Colors.red)

│ │ └── Text("41")

│ ├── Row

│ │ ├── Column

│ │ │ ├── IconButton(Icons.call)

│ │ │ └── Text("CALL")

│ │ ├── Column

│ │ │ ├── IconButton(Icons.near\_me)

│ │ │ └── Text("ROUTE")

│ │ ├── Column

│ │ │ ├── IconButton(Icons.share)

│ │ │ └── Text("SHARE")

│ └── Padding

│ └── Text("Lake Oeschinen lies at the foot of the Blüemlisalp the… ")

└── BottomNavigationBar

Explanation

1. Column: The main layout structure that stacks the widgets vertically.
2. AppBar: Displays the title of the app ("Flutter layout demo").
3. Image.asset: Displays the main image of the location.
4. Padding: Adds padding around the Column widget containing the campground details.

* Column: Holds the campground details.
  + Text Widgets: Display the title ("Oeschinen Lake Campground")

and location ("Kandersteg, Switzerland").

* + Row (Rating Section): Contains the star icon and rating count ("41").
  + Row (Action Buttons): Contains the three buttons (CALL, ROUTE, SHARE),

each Column with an IconButton and Text label.

* + Padding with Text: Displays the description of the lake and activities.

1. BottomNavigationBar: (Not explicitly visible here but often used in this layout).
   1. [LO1 – 10 Points]

Explain what is responsive and adaptive in Flutter?

* **Responsive design:** adjust layout and UI elements according to the screen size to provide an ideal experience for users across various devices (e.g., phone, tablet, desktop).
* **Adaptive design:** the UI and components adapt based on the device's platform (example, Android or iOS) to give a native feel on each system. Flutter has widgets like LayoutBuilder for responsive layouts & uses platform-specific widgets like Cupertino for adaptive behavior.

# Project (70%)

1. [LO2 – 20 Points]

Imagine that you get a Flutter project to build Fashion shop apps that sells shoes, with many variation of shoes. Create a Dart file that contains a class named Shoes, that has properties of id, size, shoe\_color, gender, brand, type, and limited\_edition. Below are the requirements:

* + The class must be constructed using a named arguments.
  + The id of each new item should be managed using Dart uuid package.
  + Within the same model file, create an enum of ShoeColors that contains at least 4 colors of your choosing.
  + Within the same model file, create an enum of Gender that contains male and female.
  + Within the same model file, create an enum of Brands that contains at least 4 brands of your choosing.
  + The limited\_edition parameter is a boolean type.

|  |  |
| --- | --- |
| **Project Structure:**  lib/  ├── main.dart  ├── models/  │ └── shoes.dart  ├── providers/  │ └── shoe\_provider.dart  ├── screens/  │ ├── home\_screen.dart  │ └── shoe\_detail\_screen.dart  └── widgets/  ├── shoe\_card.dart  └── shoe\_list.dart | **main.dart**  import 'package:flutter/material.dart';  import 'package:flutter\_riverpod/flutter\_riverpod.dart';  import 'screens/home\_screen.dart';    void main() {  runApp(ProviderScope(child: MyApp()));  }    class MyApp extends StatelessWidget {  @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'Fashion Shop',  theme: ThemeData(  primarySwatch: Colors.blue,  ),  home: HomeScreen(),  );  }  } |
| **models/shoes.dart**  import 'package:uuid/uuid.dart';    final uuid = Uuid();    enum ShoeColors { red, blue, green, black }  enum Gender { male, female }  enum Brands { Nike, Adidas, Puma, Reebok }    class Shoes {  final String id;  final double size;  final ShoeColors shoeColor;  final Gender gender;  final Brands brand;  final String type;  final bool limitedEdition;    Shoes({  String? id,  required this.size,  required this.shoeColor,  required this.gender,  required this.brand,  required this.type,  required this.limitedEdition,  }) : id = id ?? uuid.v4();  } | **providers/shoe\_provider.dart**  import 'package:flutter\_riverpod/flutter\_riverpod.dart';  import '../models/shoes.dart';    final shoeProvider = StateProvider<List<Shoes>>((ref) => []); |

|  |  |
| --- | --- |
| **screens/home\_screen.dart**  import 'package:flutter/material.dart';  import '../widgets/shoe\_list.dart';    class HomeScreen extends StatelessWidget {  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: Text('Fashion Shop'),  ),  body: ShoeList(),  );  }  } | **screens/shoe\_detail\_screen.dart**  import 'package:flutter/material.dart';  import '../models/shoes.dart';    class ShoeDetailScreen extends StatelessWidget {  final Shoes shoe;    ShoeDetailScreen({required this.shoe});    @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: Text(shoe.brand.toString()),  ),  body: Padding(  padding: const EdgeInsets.all(16.0),  child: Column(  crossAxisAlignment: CrossAxisAlignment.start,  children: [  Text('Size: ${shoe.size}'),  Text('Color: ${shoe.shoeColor}'),  Text('Gender: ${shoe.gender}'),  Text('Limited Edition: ${shoe.limitedEdition ? 'Yes' : 'No'}'),  ],  ),  ),  );  }  } |
| **widgets/shoe\_card.dart**  import 'package:flutter/material.dart';  import '../models/shoes.dart';  import '../screens/shoe\_detail\_screen.dart';    class ShoeCard extends StatelessWidget {  final Shoes shoe;    ShoeCard({required this.shoe});    @override  Widget build(BuildContext context) {  return Card(  child: ListTile(  title: Text(shoe.brand.toString()),  subtitle: Text('Size: ${shoe.size} - Color: ${shoe.shoeColor}'),  onTap: () {  Navigator.of(context).push(  MaterialPageRoute(  builder: (ctx) => ShoeDetailScreen(shoe: shoe),  ),  );  },  ),  );  }  } | **widgets/shoe\_list.dart**  import 'package:flutter/material.dart';  import 'package:flutter\_riverpod/flutter\_riverpod.dart';  import '../providers/shoe\_provider.dart';  import 'shoe\_card.dart';    class ShoeList extends ConsumerWidget {  @override  Widget build(BuildContext context, WidgetRef ref) {  final shoes = ref.watch(shoeProvider).state;  return ListView.builder(  itemCount: shoes.length,  itemBuilder: (ctx, index) => ShoeCard(shoe: shoes[index]),  );  }  } |

1. [LO2 – 50 Points]

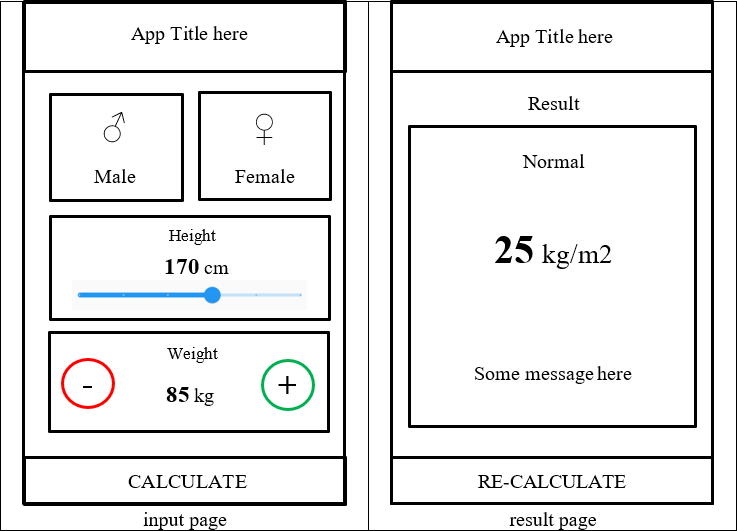
Create a simple BMI calculator app which has two pages for navigation. You must following the same application layout as presented in Figure 2 below, but you may customize all the widgets styles and colors.

Figure 2 - Application Layout

App description:

The app consists of 2 pages for navigation. Both page should have a title on top of it and a button for navigation at the bottom. On the input page, user should choose his/her gender, then input their height using a Slider Widget ([Slider class](https://api.flutter.dev/flutter/material/Slider-class.html)

[- material library - Dart API (flutter.dev)),](https://api.flutter.dev/flutter/material/Slider-class.html) and finally input their weight with the help of 2 buttons. When pressed, the “CALCULATE” button will navigate the user to the result page showing all the calculation result with some messages at the bottom. The “RE-CALCULATE” button will take the user back to the input page.

The BMI classifications are as follow :

Table 1 – BMI Classification based on gender

|  |  |  |  |
| --- | --- | --- | --- |
| **Female** | | **Male** | |
| Underweight | < 17 kg/m2 | Underweight | < 18 kg/m2 |
| Normal | 17 - 23 kg/m2 | Normal | 18-25 kg/m2 |
| Overweight | 23-27 kg/m2 | Overweight | 25-27 kg/m2 |
| Obese | > 27 kg/m2 | Obese | > 27 kg/m2 |

A black text with a black line

Description automatically generated

The formula for BMI calculation is :

**Note** : The formula calculate height in meter while the user input is in centimeter. A conversion must be made within your code.

Project Structure :

* main.dart
* screens
  + home\_page.dart
  + result\_page.dart
* widgets
  + gender\_selection.dart
  + height\_selection.dart
  + weight\_selection.dart

Scoring will be based on the following criteria :

1. [LO2 – 10 Points] Able to navigate between pages.
2. [LO2 – 10 Points] App layout with complete widgets are presented.
3. [LO2 – 10 Points] Widgets functionality (buttons and slider).
4. [LO2 – 10 Points] Able to do calculations based on user input.
5. [LO2 – 10 Points] Following the project structure requirements.

Your task :

1. (Mandatory) Video recording
   * Record a short demonstration video ( < 5 min ) to demonstrate all your app functionality.
   * Demonstrate all variations in Table 1 classifications.
   * The recording must be made available online for scoring. Provide the link to your video repository in your exam answer sheet.
   * Recording upload date must be clearly visible. Any upload after the exam submission date will not be graded.
2. (Mandatory) Attach only your code (not the whole project) as a zip file.