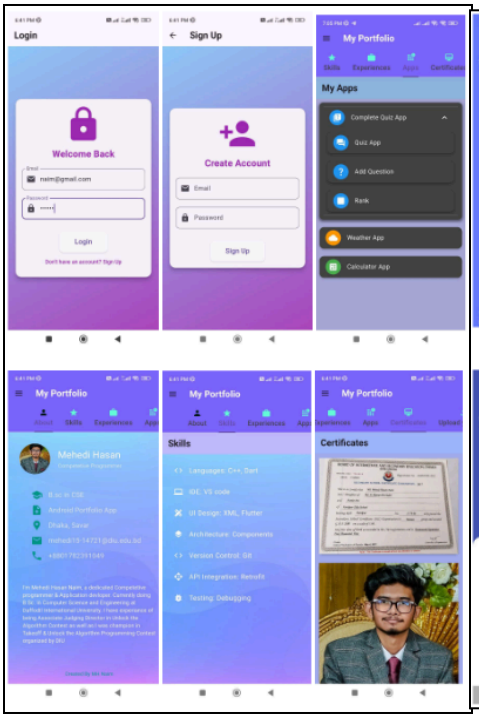
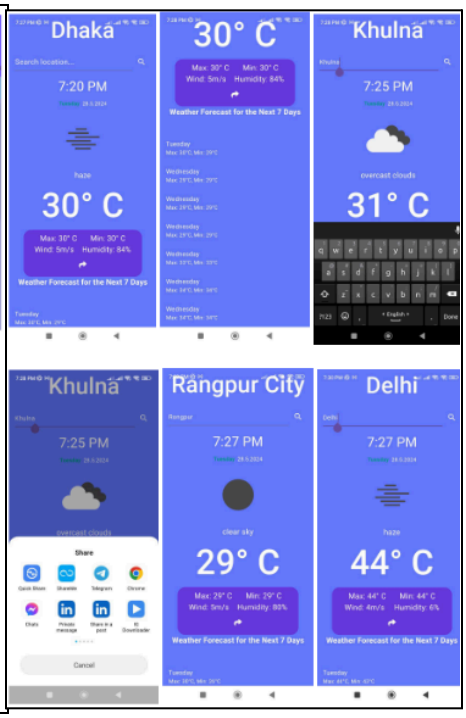
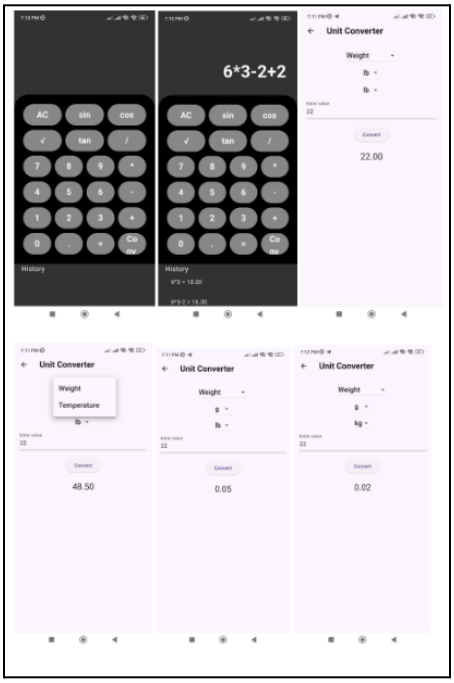
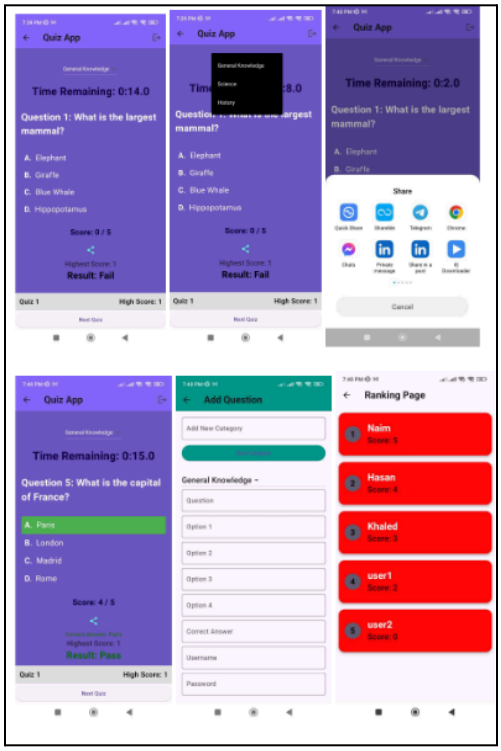
Github: https://github.com/naim79992/Intregrated_Application
Video: https://youtu.be/W1Ho53tt30M?si=Fpnl3FL6HGzDwlVJ

User Interface(UI):









1. Setting Up Your Flutter Development Environment

i. Download and Install Flutter SDK
- Visit the [official Flutter website](https://flutter.dev/docs/get-started/install) and download the Flutter SDK for your operating system.
- Extract the downloaded Flutter SDK to a preferred location on your computer.

ii. Set Up Environment Variables
- Windows:
- Search for "Environment Variables" in the Start Menu & edit the `PATH` variable to include the Flutter `bin` directory (e.g., `C:\path\to\flutter\bin`).

iii. Install Visual Studio Code (VS Code)
- Download and install VS Code from the [official website](https://code.visualstudio.com/).

iv. Install Essential Extensions in VS Code
- Open VS Code and go to the Extensions view & search for and install the following extensions:
  - Flutter
  - Dart
  - Flutter Helpers
  - Flutter Icons
  - Flutter Widget Snippets
  - Flutter Widget Wrap

v**. Set Up an Emulator (Without Android Studio)**
- Download and install a standalone emulator.
- Ensure the emulator is properly installed and added to your system's `PATH`.

vi. Verify Installation flutter doctor

2. Step-by-Step Guide to Creating and Organizing a Flutter Project

I. Create a New Flutter Project:

Step 1: Check Flutter Installation

Open a terminal or command prompt and run: flutter doctor

Step 2: Enable Desktop Support

Run the following command to enable desktop support (specifically for Windows):

flutter config --enable-windows-desktop

Step 3: Create a New Flutter Project

Command to create a new Flutter project: flutter create my_desktop_app

Navigate to the project directory: cd port

Step 4: Run the Project on Desktop

First, add necessary dependencies by running: flutter pub get

Then, run the project on your desktop: flutter run (checks Flutter installation & displays missing dependencies.)

2. Organize Project Structure: Create Folders inside the lib directory, create folders for each section of app: profile, weather, quiz, calculator

3. Personal Profile with Portfolio

a) Design the User Interface for the Personal Profile and Portfolio Sections

Personal Profile:

- Create a visually appealing form for users to input their personal information.

- Design a display area that shows the user's name, photo, bio and contact details in a clean layout.

- Use consistent colors and styling to ensure a cohesive look throughout the profile section.

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


class About extends StatelessWidget {
  const About({super.key});


  @override
  Widget build(BuildContext context) {
```

```
return Scaffold(
  body: Container(
    decoration: const BoxDecoration(
      image: DecorationImage(
        image: AssetImage("lib/Assets/images/BG.jpg"),
        fit: BoxFit.cover,
      ),
    ),
    child: const Padding(
      padding: EdgeInsets.only(top: 30.0, left: 30),
      child: Column(
        children: <Widget>[
          Row(
            children: <Widget>[
              CircleAvatar(
                radius: 40,
                backgroundImage:
                    AssetImage("lib/Assets/images/MY PHOTO.jpg"),
              ),
              SizedBox(
                width: 20,
              ),
              Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: <Widget>[
                  Text(
                    "Mehedi Hasan",
                    style: TextStyle(
                        fontSize: 25,
                        color: Colors.white,
                        fontFamily: "Roboto"),
                  ),
                  Text(
                    "Competetive Programmer",
                    style: TextStyle(
                        fontSize: 14,
                        color: Color.fromARGB(179, 255, 255, 255),
                        fontFamily: "Robotor"),
                  )
                ],
              )
            ],
          ),
          SizedBox(
            height: 40,
          ),
          Padding(
            padding: EdgeInsets.only(left: 30),
            child: Column(
              children: <Widget>[
                Row(
                  children: <Widget>[
                    Icon(
                      Icons.school,
                      size: 30,
```

```
        color: Colors.teal,
      ),
      SizedBox(
        width: 15,
      ),
      Text(
        "B.sc in CSE",
        style: TextStyle(
            fontSize: 18,
            color: Colors.white70,
            fontFamily: "Robotor"),
      ),
    ],
  ),
  SizedBox(
    height: 10,
  ),
  Row(
    children: <Widget>[
      Icon(
        Icons.note_add_rounded,
        size: 30,
        color: Colors.teal,
      ),
      SizedBox(
        width: 15,
      ),
      Text(
        "Android Portfolio App",
        style: TextStyle(
            fontSize: 18,
            color: Colors.white70,
            fontFamily: "Robotor"),
      ),
    ],
  ),
  SizedBox(
    height: 10,
  ),
  Row(
    children: <Widget>[
      Icon(
        Icons.location_pin,
        size: 30,
        color: Colors.teal,
      ),
      SizedBox(
        width: 15,
      ),
      Text(
        "Dhaka, Savar",
        style: TextStyle(
            fontSize: 18,
            color: Colors.white70,
            fontFamily: "Robotor"),
```

```
            ),
          ],
        ),
        SizedBox(
          height: 10,
        ),
        Row(
          children: <Widget>[
            Icon(
              Icons.email,
              size: 30,
              color: Colors.teal,
            ),
            SizedBox(
              width: 15,
            ),
            Text(
              "mehedi15-14721@diu.edu.bd",
              style: TextStyle(
                  fontSize: 18,
                  color: Colors.white70,
                  fontFamily: "Robotor"),
            ),
          ],
        ),
        SizedBox(
          height: 10,
        ),
        Row(
          children: <Widget>[
            Icon(
              Icons.phone,
              size: 30,
              color: Colors.teal,
            ),
            SizedBox(
              width: 15,
            ),
            Text(
              "+8801782391049",
              style: TextStyle(
                  fontSize: 18,
                  color: Colors.white70,
                  fontFamily: "Robotor"),
            ),
          ],
        ),
      ],
    ),
  ),
  SizedBox(
    height: 50,
  ),
  Padding(
    padding: EdgeInsets.all(8),
```

```
            child: Text(
              "I'm Mehedi Hasan Naim, a dedicated Competetive programmer & Application devloper. Currently doing B.Sc. in Computer Science and Engineering at Daffodil International University. I have experiance of being Associate Judging Director in Unlock the Algorithm Contest as well as I was champion in Takeoff & Unlock the Algorithm Programming Contest organized by DIU",
               style: TextStyle(
                  fontSize: 14,
                  color: Color.fromARGB(255, 250, 250, 250),
                  fontFamily: "Robotor"),
            ),
          ),
          SizedBox(
            height: 60,
          ),
          Text(
            "Created By MH Naim",
            style: TextStyle(
                fontSize: 13, color: Colors.teal, fontFamily: "Robotor"),
          ),
        ],
      ),
    ),
  ),
 );
 }
}
```

Portfolio:

- Design a section to showcase user projects using cards or tiles for each project.

- Each card should include the project name, description, and a link to the project.

- Ensure the layout is scrollable and responsive for a good user experience.

b) Allow Users to Input Their Personal Information

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


class Portfolio extends StatelessWidget {
 const Portfolio({super.key});
```

```
@override
Widget build(BuildContext context) {
  return Container(
    decoration: BoxDecoration(
      gradient: LinearGradient(
        begin: Alignment.topCenter,
        end: Alignment.bottomCenter,
        colors: [const Color.fromARGB(255, 100, 90, 249), Colors.grey[900]!],
      ),
    ),
    child: SingleChildScrollView(
      padding: const EdgeInsets.all(16),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            "Project list",
            style: Theme.of(context).textTheme.titleLarge!.copyWith(
                  color: Colors.white,
                  fontWeight: FontWeight.bold,
                ),
          ),
          const SizedBox(height: 16),
          const PortfolioItem(
            projectName: "FoodMate",
            projectLink: "https://github.com/naim79992/Food-Mate",
            projectDescription:
                "In this project I have made a food delivary website",
          ),
          const SizedBox(height: 16),
          const PortfolioItem(
            projectName: "Student Management System",
            projectLink:
                "https://github.com/naim79992/Student-management-system",
            projectDescription: "This is a Java based project.",
          ),
          const SizedBox(height: 16),
          const PortfolioItem(
            projectName: "Weather App",
            projectLink: "https://github.com/naim79992/Weather-app",
            projectDescription: "This is a Flutter based project.",
          ),
          const SizedBox(height: 16),
          const PortfolioItem(
            projectName: "Scientific Calculator",
            projectLink:
                "https://github.com/naim79992/Scientific-Calculator_",
            projectDescription: "This is a fully functional calculator app.",
          ),
          // Add more PortfolioItem widgets as needed
        ],
      ),
    ),
  );
}
```

```
}


class PortfolioItem extends StatelessWidget {
  final String projectName;
  final String projectLink;
  final String projectDescription;


  const PortfolioItem({
    super.key,
    required this.projectName,
    required this.projectLink,
    required this.projectDescription,
  });


  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 3,
      margin: const EdgeInsets.symmetric(vertical: 8),
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(15),
      ),
      color: const Color.fromARGB(255, 255, 23, 23),
      child: ListTile(
        title: Text(
          projectName,
          style: const TextStyle(
            color: Colors.white,
            fontSize: 18,
            fontWeight: FontWeight.bold,
          ),
        ),
        subtitle: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            const SizedBox(height: 8),
            Text(
              projectDescription,
              style: const TextStyle(color: Colors.white),
            ),
            const SizedBox(height: 8),
            Text(
              projectLink,
              style: const TextStyle(
                color: Colors.teal,
                decoration: TextDecoration.underline,
              ),
            ),
          ],
        ),
        onTap: () {
          // Add actions for when the portfolio item is tapped
```

```
      },
    ),
  );
 }
}
```

- Implement text fields and image pickers in the profile form for users to input their name, upload a photo, write a bio, and add contact details.

- Use form validation to ensure all necessary information is provided before submission.

- Store the input data and display it dynamically in the profile section.

c) Create a Section for Users to Showcase Their Portfolio

- Use a list or grid view to display the portfolio items.

- Each portfolio item should be represented as a card with the project name, description, and a link.

- Allow users to add new projects through an input form and update the list dynamically.

d) Additional Functions

i. Allow Users to Upload Their Resume/CV

- Implement a file picker to enable users to upload their resume/CV.

- Store the uploaded file and provide a link or button to download/view the resume from the profile section.

ii. Implement a Social Media Links Section

- Create a navigation drawer or a dedicated section in the profile where users can add their social media links.

- Include icons and links for popular social media platforms like LinkedIn, GitHub, Twitter, etc.

- Use a consistent style and ensure the links open in the appropriate apps or browser.

iii. Add a Blog Section Where Users Can Write and Share Their Articles:

- Implement a blog section with a list of blog posts.

- Allow users to create new blog posts using a form with fields for title, excerpt, and content.

- Display the blog posts with options to read the full content and share via social media.

iv. Provide an Option for Users to Upload and Display Their Achievements and Certifications:

- Allow users to upload images or PDFs of their achievements and certifications.

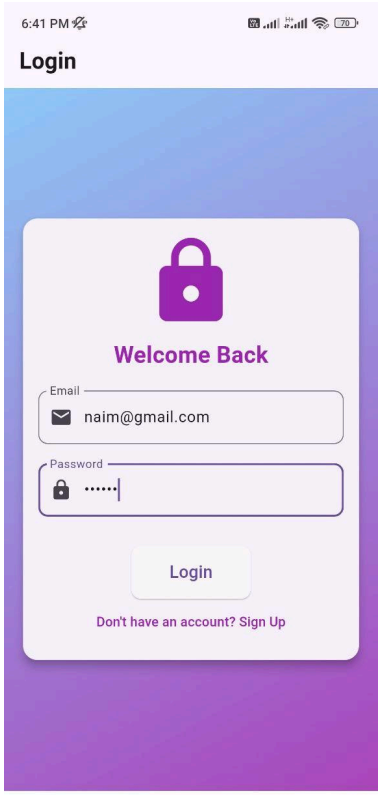
- Display the uploaded files in a scrollable list or grid view.

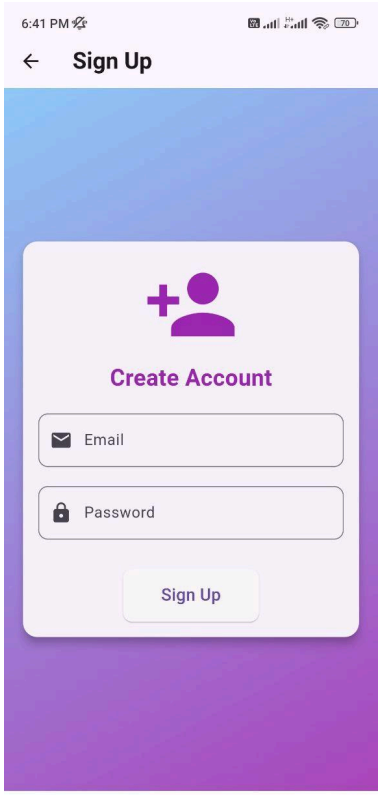
- Use cards or similar UI elements to showcase each certificate with options to view in full size.

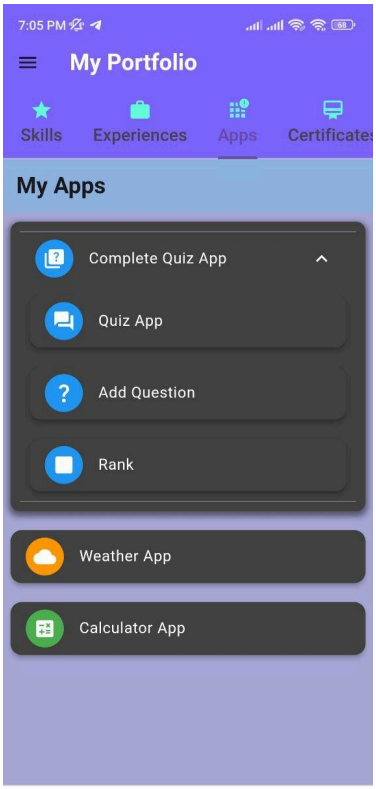
```
import 'package:flutter/material.dart';
import 'package:share/share.dart';
import 'BlogWrite.dart';

class Blog extends StatelessWidget {
  const Blog({super.key});

  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Blog'),
        backgroundColor: Theme.of(context).primaryColor,
      ),
      body: ListView(
        children: const [
          BlogPost(
            title: 'First Blog Post',
            excerpt: 'This is the summary of the first blog post...',
            content:
                'This is the full content of the first blog post. Here you can add more details about your
topic...',
          ),
          BlogPost(
            title: 'Second Blog Post',
            excerpt: 'This is the summary of the second blog post...',
            content:
```

```
            'This is the full content of the second blog post. Here you can add more details about
your topic...',
          ),
          // Add more BlogPost widgets as needed
        ],
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => const BlogWrite(),
            ),
          );
        },
        backgroundColor: Theme.of(context).primaryColor,
        child: const Icon(Icons.add),
      ),
    );
  }
}

class BlogPost extends StatelessWidget {
  final String title;
  final String excerpt;
  final String content;


  const BlogPost({
    required this.title,
    required this.excerpt,
    required this.content,
  });

  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.all(10),
      child: ListTile(
        title: Text(
          title,
          style: Theme.of(context).textTheme.titleMedium,
        ),
        subtitle: Text(
          excerpt,
          style: Theme.of(context).textTheme.titleSmall,
        ),
        trailing: const Icon(Icons.arrow_forward),
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => BlogDetail(
                title: title,
                content: content,
              ),
```

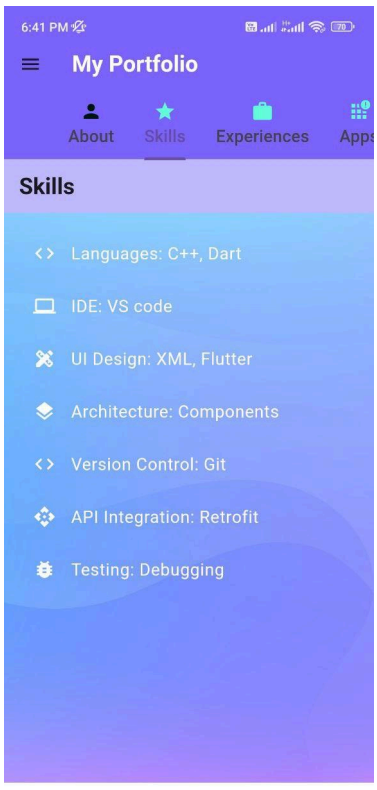
```
          ),
        );
      },
    ),
  );
 }
}

class BlogDetail extends StatelessWidget {
 final String title;
 final String content;
 const BlogDetail({
  required this.title,
  required this.content,
 });

 Widget build(BuildContext context) {
  return Scaffold(
   appBar: AppBar(
    title: Text(title),
    backgroundColor: Theme.of(context).primaryColor,
    actions: [
     IconButton(
      icon: const Icon(Icons.share),
      onPressed: () {_shareContent(context, title, content);},
     ),
    ],
   ),
   body: Padding(
    padding: const EdgeInsets.all(16.0),
    child: SingleChildScrollView(
     child: Text(
      content,
      style: Theme.of(context).textTheme.bodyLarge,
     ),
    ),
   ),
  );
 }
 void _shareContent(BuildContext context, String title, String content) {
  final RenderBox box = context.findRenderObject() as RenderBox;
  Share.share(
   '$title\n\n$content',
   subject: title,
   sharePositionOrigin: box.localToGlobal(Offset.zero) & box.size,
  );
 }
}
```

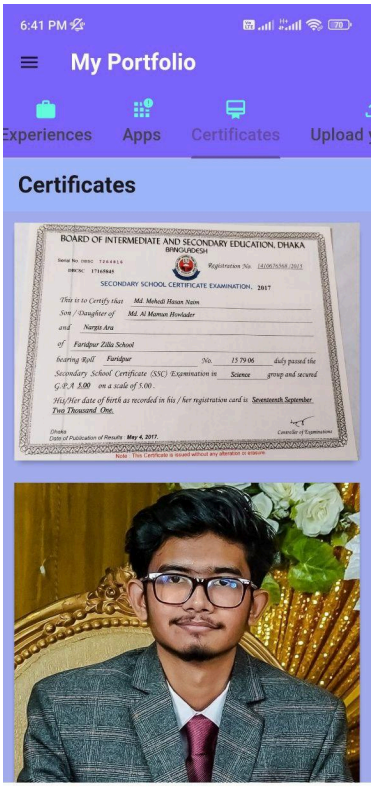

6:41 PM
Login
Welcome Back
Email
naim@gmail.com
Password
Login
Don't have an account? Sign Up

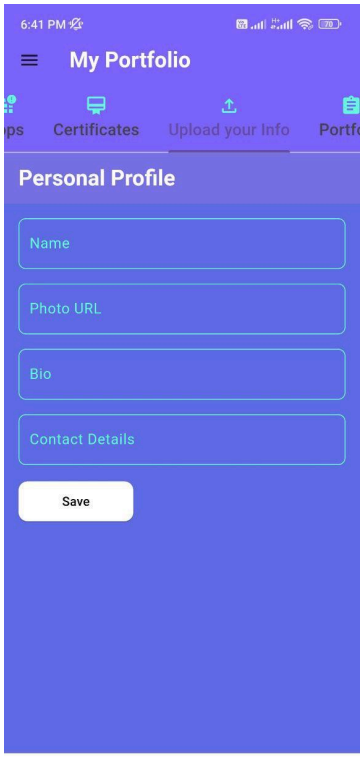

6:41 PM
Sign Up
Create Account
Email
Password
Sign Up

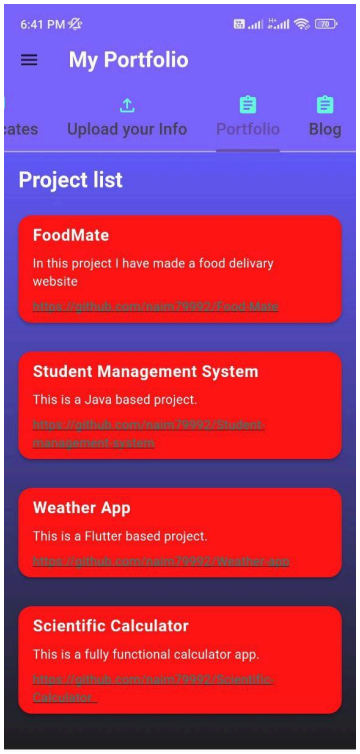

7:05 PM
My Portfolio
Skills
Experiences
Apps
Certificates
My Apps
Complete Quiz App
Quiz App
Add Question
Rank
Weather App
Calculator App

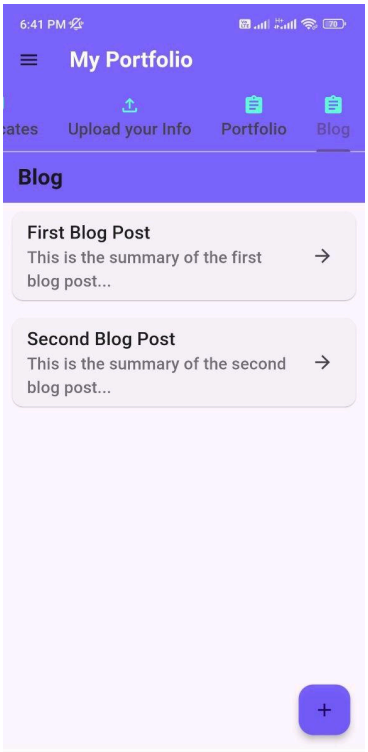

6:41 PM
My Portfolio
About
Skills
Experiences
Mehedi Hasan
Competetive Programmer
B.sc in CSE
Android Portfolio App
Dhaka, Savar
mehedi15-14721@diu.edu.bd
+8801782391049
I'm Mehedi Hasan Naim, a dedicated Competetive programmer & Application devloper. Currently doing B.Sc. in Computer Science and Engineering at Daffodil International University. I have experiance of being Associate Judging Director in Unlock the Algorithm Contest as well as I was champion in Takeoff & Unlock the Algorithm Programming Contest organized by DIU
Created By MH Naim

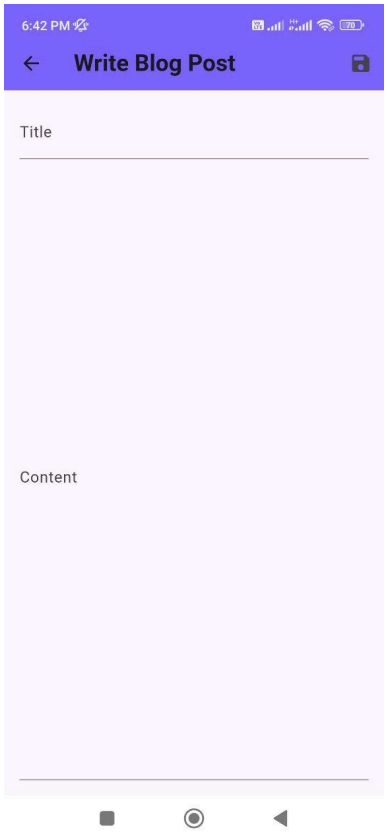

6:41 PM
My Portfolio
About
Skills
Experiences
Skills
Languages: C++, Dart
IDE: VS code
UI Design: XML, Flutter
Architecture: Components
Version Control: Git
API Integration: Retrofit
Testing: Debugging

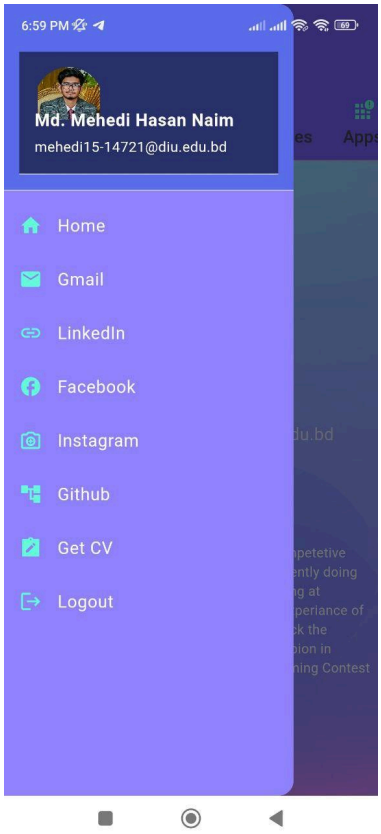

6:41 PM
My Portfolio
Experiences
Apps
Certificates
Certificates
BOARD OF INTERMEDIATE AND SECONDARY EDUCATION, DHAKA
BANGLADESH
SECONDARY SCHOOL CERTIFICATE EXAMINATION, 2017

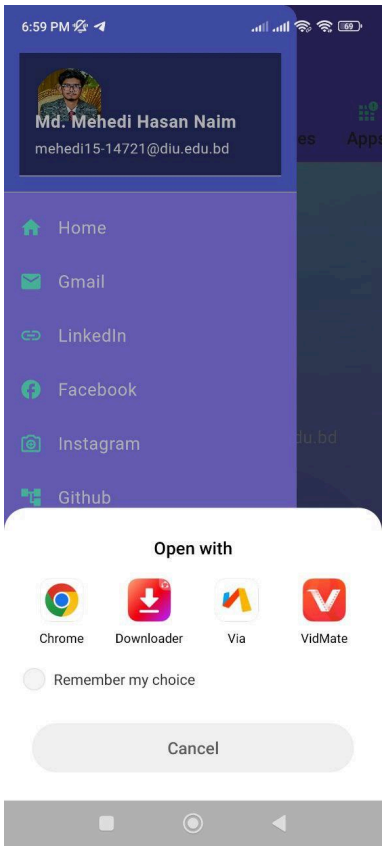

6:41 PM
My Portfolio
Certificates
Upload your Info
Personal Profile
Name
Photo URL
Bio
Contact Details
Save


6:41 PM
My Portfolio
Upload your Info
Portfolio
Blog
Project list
FoodMate
In this project I have made a food delivery website
Student Management System
This is a Java based project.
Weather App
This is a Flutter based project.
Scientific Calculator
This is a fully functional calculator app.


6:41 PM
My Portfolio
Upload your Info
Portfolio
Blog
Blog
First Blog Post
This is the summary of the first blog post...
Second Blog Post
This is the summary of the second blog post...


6:42 PM
Write Blog Post
Title
Content


6:59 PM
Md. Mehedi Hasan Naim
mehedi15-14721@diu.edu.bd
Home
Gmail
LinkedIn
Facebook
Instagram
Github
Get CV
Logout


6:59 PM
Md. Mehedi Hasan Naim
mehedi15-14721@diu.edu.bd
Home
Gmail
LinkedIn
Facebook
Instagram
Github
Open with
Chrome
Downloader
Via
VidMate
Remember my choice
Cancel

4. Navigation in Flutter App

a) Implement Navigation Within Your App

1. Create a Central Navigation Widget:
    - Define a main navigation widget that will serve as the entry point for navigating between different sections of the app (Calculator, Weather, Quiz, Profile).
2. Utilize Navigator for Route Management:
    - Use Flutter's Navigator to manage routes and facilitate navigation between different screens.
    - Define named routes for each section for easier and more readable navigation code.
3. Bottom Navigation Bar:
    - Implement a BottomNavigationBar for seamless navigation between the main sections of the app.
    - Ensure each navigation item in the bar corresponds to a different screen.

b) Utilize Flutter's Navigation Widgets

1. Navigator:
    - Use Navigator.push and Navigator.pop for stack-based navigation.
    - Utilize Navigator.pushNamed for named route navigation.
2. BottomNavigationBar:
    - Define a BottomNavigationBar widget at the bottom of the main screen to provide quick access to different sections.

5. Scientific Calculator

a) Design the User Interface

1. Calculator Layout:

- Create a visually appealing layout using Flutter widgets to design the calculator interface.

- Utilize buttons for digits, arithmetic operators, and advanced functions.

- Organize the buttons in a grid layout to optimize space and improve usability.

```
import 'package:flutter/material.dart';
import 'package:math_expressions/math_expressions.dart';
import 'package:port/Fragment/UnitConverterScreen.dart';


class CalculatorScreen extends StatefulWidget {
  const CalculatorScreen({super.key});


  @override
  _CalculatorScreenState createState() => _CalculatorScreenState();
}


class _CalculatorScreenState extends State<CalculatorScreen> {
  String _input = '';
  String _output = '';


  // Define a list to store history
  List<String> _history = [];


  // Add method to handle button press
  void _handleButtonPressed(String buttonText) {
    setState(() {
      if (buttonText == '=') {
        _output = _calculateResult();
        // Add the calculation to history
        _history.add('$_input = $_output');
      } else if (buttonText == 'AC') {
        _input = '';
        _output = '';
      } else if (buttonText == '√') {
        _input = _calculateSquareRoot();
      } else if (buttonText == 'sin') {
        _input = 'sin($_input)';
      } else if (buttonText == 'cos') {
        _input = 'cos($_input)';
      } else if (buttonText == 'tan') {
        _input = 'tan($_input)';
      } else if (buttonText == '*') {
        _input += '*';
      } else if (buttonText == '/') {
        _input += '/';
      } else if (buttonText == 'Convert') {
        _navigateToUnitConverter();
      } else {
        _input += buttonText;
      }
    });
  }


  void _navigateToUnitConverter() {
```

```
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => const UnitConverterScreen()),
  );
}


String _calculateResult() {
  try {
    Parser p = Parser();
    Expression exp = p.parse(_input);
    ContextModel cm = ContextModel();
    double result = exp.evaluate(EvaluationType.REAL, cm);
    if (result.isNaN) {
      return 'Error';
    }
    return result.toStringAsFixed(2);
  } catch (e) {
    return 'Error';
  }
}


String _calculateSquareRoot() {
  try {
    Parser p = Parser();
    Expression exp = p.parse('sqrt($_input)');
    ContextModel cm = ContextModel();
    double result = exp.evaluate(EvaluationType.REAL, cm);
    if (result.isNaN) {
      return 'Error';
    }
    return result.toStringAsFixed(2);
  } catch (e) {
    return 'Error';
  }
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color(0xFF333333),
    body: Column(
      children: [
        Expanded(
          flex: 2,
          child: Container(
            padding: const EdgeInsets.all(32),
            alignment: Alignment.bottomRight,
            child: Text(
              _input,
              style: const TextStyle(
                fontSize: 48,
                fontWeight: FontWeight.bold,
```

```
          color: Colors.white,
        ),
      ),
    ),
  ),
  Expanded(
    flex: 4,
    child: Container(
      padding: const EdgeInsets.all(20),
      decoration: const BoxDecoration(
        color: Colors.black,
        borderRadius: BorderRadius.vertical(top: Radius.circular(40)),
      ),
      child: Column(
        children: [
          _buildButtonRow(['AC', 'sin', 'cos']),
          _buildButtonRow(['√', 'tan', '/']),
          _buildButtonRow(['7', '8', '9', '*']),
          _buildButtonRow(['4', '5', '6', '-']),
          _buildButtonRow(['1', '2', '3', '+']),
          _buildButtonRow(['0', '.', '=', 'Convert']),
        ],
      ),
    ),
  ),
  Expanded(
    child: Container(
      padding: const EdgeInsets.symmetric(horizontal: 20),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: [
          const Text(
            'History',
            style: TextStyle(
              color: Colors.white,
              fontSize: 20,
            ),
          ),
          Expanded(
            child: ListView.builder(
              itemCount: _history.length,
              itemBuilder: (context, index) {
                return ListTile(
                  title: Text(
                    _history[index],
                    style: const TextStyle(
                      color: Colors.white,
                    ),
                  ),
                );
              },
            ),
          ),
        ],
      ),
```

```
          ),
        ),
      ],
    ),
  );
}


Widget _buildButtonRow(List<String> buttonLabels) {
  return Expanded(
    child: Row(
      children: buttonLabels
          .map((label) => Expanded(child: _buildButton(label)))
          .toList(),
    ),
  );
}


Widget _buildButton(String buttonText) {
  return Container(
    margin: const EdgeInsets.all(5),
    child: ElevatedButton(
      onPressed: () => _handleButtonPressed(buttonText),
      style: ButtonStyle(
        backgroundColor: MaterialStateProperty.all(Colors.white54),
        shape: MaterialStateProperty.all(
          RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(50),
          ),
        ),
      ),
      child: SizedBox(
        width: 60,
        height: 60,
        child: Center(
          child: Text(
            buttonText,
            style: const TextStyle(
              fontSize: 24,
              fontWeight: FontWeight.bold,
              color: Colors.white,
            ),
          ),
        ),
      ),
    ),
  );
 }
}
```

2. Advanced Functions:

- Include buttons for advanced mathematical functions such as square root, exponentiation, trigonometric functions (sin, cos, tan), logarithms, etc.

- Ensure that the layout is intuitive and user-friendly, with clear labels and descriptions for each button.

b) Implement Calculator Logic

1. Basic Arithmetic Operations:

- Develop Dart logic to handle basic arithmetic operations such as addition, subtraction, multiplication, and division.

- Implement error handling to prevent division by zero and handle invalid input.

2. Advanced Functions:

- Code the logic for advanced mathematical functions according to mathematical principles.

- Utilize Dart's math library or custom functions to calculate square roots, exponents, trigonometric functions, logarithms, etc.

c) Handle User Input and Display Results

1. Input Handling:

- Capture user input from button clicks and translate it into mathematical expressions.

- Implement a parser to evaluate the input expression and perform the corresponding calculations.

2. Result Display:

- Display the calculated result in a clear and readable format on the calculator screen.

- Ensure that the result area is large enough to accommodate long numbers and scientific notation.

d) Additional Functions

- Implement a feature to store previous calculations in a history log.

- Allow users to view and scroll through past calculations for reference.

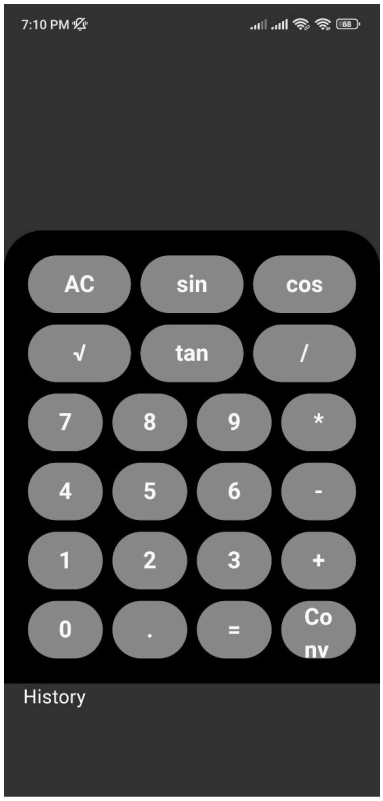
- Extend the calculator's capabilities by adding unit conversion functionalities for weight, temperature, etc.

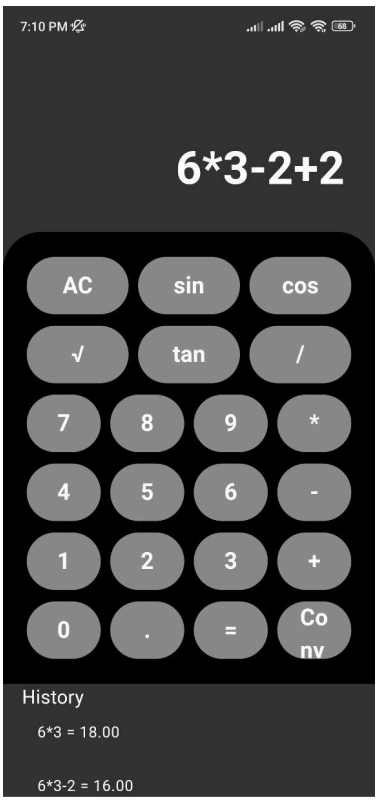
- Integrate conversion formulas and logic to handle different units and conversions.

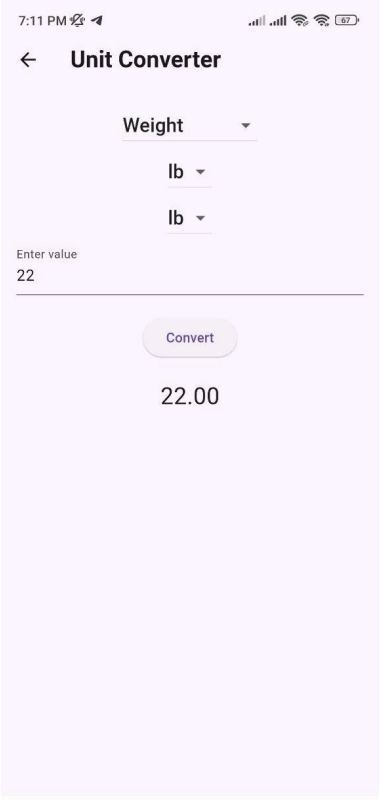
```
class UnitConverterScreen extends StatefulWidget {
  const UnitConverterScreen({super.key});
  _UnitConverterScreenState createState() => _UnitConverterScreenState();
}
class _UnitConverterScreenState extends State<UnitConverterScreen> {
  String _input = '';
  String _output = '';
  String _selectedConversion = 'Weight';
  String _fromUnit = 'kg';
  String _toUnit = 'lb';
  final Map<String, List<String>> _units = {
    'Weight': ['kg', 'lb', 'g'],
    'Temperature': ['C', 'F', 'K']
  };
  final Map<String, Map<String, double>> _conversionFactors = {
    'Weight': {
      'kg_to_lb': 2.20462,
      'kg_to_g': 1000,
      'lb_to_kg': 0.453592,
      'lb_to_g': 453.592,
      'g_to_kg': 0.001,
      'g_to_lb': 0.00220462,
    },
    'Temperature': {
      'C_to_F': 1.8,
      'F_to_C': -0.555556,
      'C_to_K': 1,
      'K_to_C': 1,
      'F_to_K': 0.555556,
      'K_to_F': -1.8,
    }
  };
  void _convert() {
    setState(() {
      if (_input.isNotEmpty) {
        double value = double.parse(_input);
        if (_selectedConversion == 'Weight') {
          _output = _convertWeight(value).toStringAsFixed(2);
```

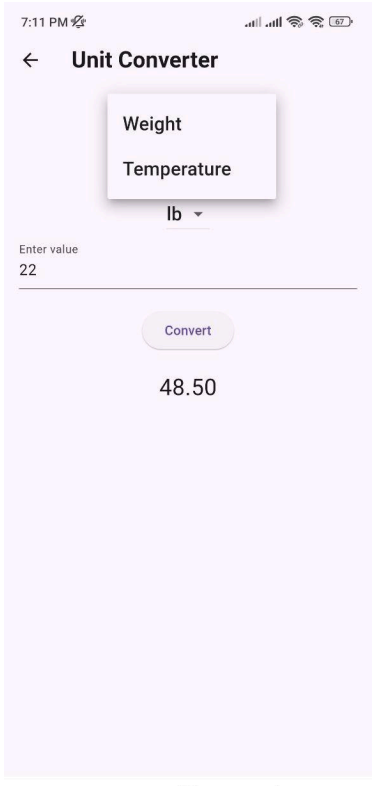
```
      } else if (_selectedConversion == 'Temperature') {
        _output = _convertTemperature(value).toStringAsFixed(2);
      }
    }
  });
}
double _convertWeight(double value) {
  String key = '${_fromUnit}_to_$_toUnit';
  return value * (_conversionFactors['Weight']![key] ?? 1);
}

double _convertTemperature(double value) {
  if (_fromUnit == 'C' && _toUnit == 'F') {
    return (value * _conversionFactors['Temperature']!['C_to_F']!) + 32;
  } else if (_fromUnit == 'F' && _toUnit == 'C') {
    return (value - 32) * _conversionFactors['Temperature']!['F_to_C']!;
  } else if (_fromUnit == 'C' && _toUnit == 'K') {
    return value + 273.15;
  } else if (_fromUnit == 'K' && _toUnit == 'C') {
    return value - 273.15;
  } else if (_fromUnit == 'F' && _toUnit == 'K') {
    return (value - 32) * _conversionFactors['Temperature']!['F_to_K']! +
        273.15;
  } else if (_fromUnit == 'K' && _toUnit == 'F') {
 return (value - 273.15) * _conversionFactors['Temperature']!['K_to_F']! +  32;
  }
  return value;
}
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Unit Converter'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Column(
        children: [
          DropdownButton<String>(
            value: _selectedConversion,
            onChanged: (value) {
              setState(() {
                _selectedConversion = value!;
                _fromUnit = _units[_selectedConversion]![0];
                _toUnit = _units[_selectedConversion]![1];
              });
            },
            items: _units.keys.map<DropdownMenuItem<String>>((String key) {
              return DropdownMenuItem<String>(
                value: key,
                child: Text(key),
              );
            }).toList(),
          ),
          DropdownButton<String>(
            value: _fromUnit,
```

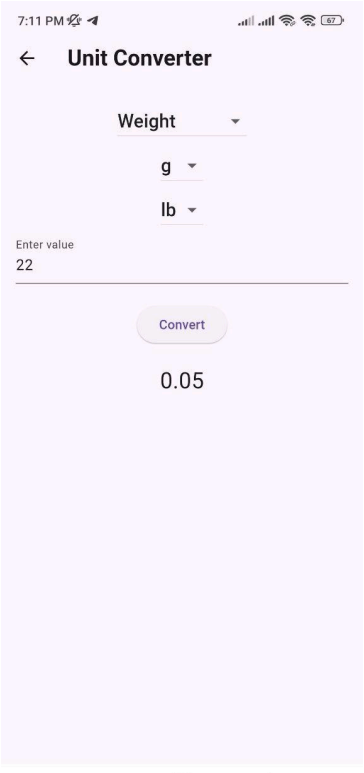
```
            onChanged: (value) {
              setState(() {
                _fromUnit = value!;
              });
            },
            items: _units[_selectedConversion]!
                .map<DropdownMenuItem<String>>((String unit) {
              return DropdownMenuItem<String>(
                value: unit,
                child: Text(unit),
              );
            }).toList(),
          ),
          DropdownButton<String>(
            value: _toUnit,
            onChanged: (value) {
              setState(() {
                _toUnit = value!;
              });
            },
            items: _units[_selectedConversion]!
                .map<DropdownMenuItem<String>>((String unit) {
              return DropdownMenuItem<String>(
                value: unit,
                child: Text(unit),
              );
            }).toList(),
          ),
          TextField(
            keyboardType: TextInputType.number,
            decoration: const InputDecoration(
              labelText: 'Enter value',
            ),
            onChanged: (value) {
              _input = value;
            },
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            onPressed: _convert,
            child: const Text('Convert'),
          ),
          const SizedBox(height: 20),
          Text(
            _output,
            style: const TextStyle(fontSize: 24),
          ),
        ],
      ),
    ),
  );
 }
}
```

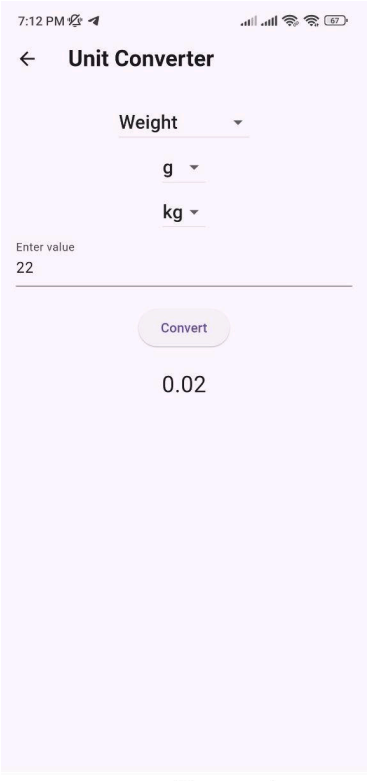

7:10 PM
AC
sin
cos
√
tan
/
7
8
9
*
4
5
6
-
1
2
3
+
0
.
=
Co
nv
History

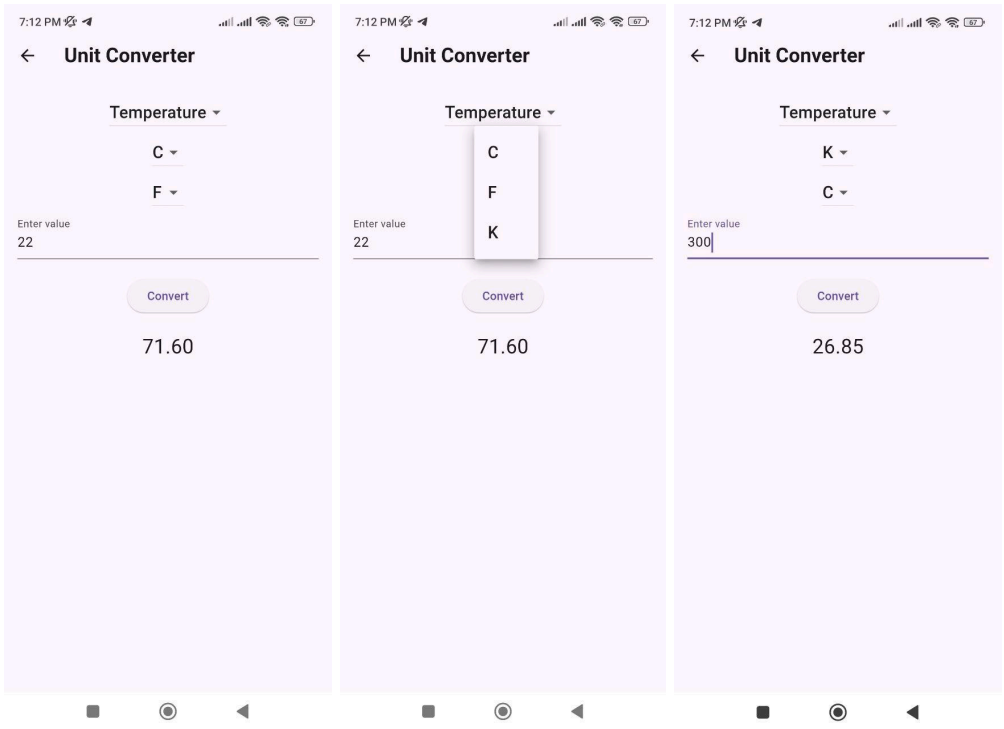

7:10 PM
6*3-2+2
AC
sin
cos
√
tan
/
7
8
9
*
4
5
6
-
1
2
3
+
0
.
=
Co
nv
History
6*3 = 18.00
6*3-2 = 16.00


7:11 PM
Unit Converter
Weight
lb
lb
Enter value
22
Convert
22.00


7:11 PM
Unit Converter
Weight
Temperature
lb
Enter value
22
Convert
48.50


7:11 PM
Unit Converter
Weight
g
lb
Enter value
22
Convert
0.05


7:12 PM
Unit Converter
Weight
g
kg
Enter value
22
Convert
0.02



6. Weather App:

a) User Interface Design:

- Header: Shows current location.
- Search Bar: For adding new locations.
- Date and Time: Current date and time.
- Weather Icon and Description: Current weather conditions.
- Temperature: Current temperature.
- Additional Weather Info: Wind speed, humidity, max/min temperatures.
- Share Button: To share weather info.
- Forecast List: Scrolling forecast for next 7 days.

b) API Integration: Use OpenWeatherMap API for weather data.

c) User Experience:

- **Real-Time Updates:** Instantly reflect weather changes.
- **Location-Based Forecasting:** Accurate forecasts based on user's location.
- **Simple Design:** Easy to use and navigate.

d) Additional Functions:

i. 7-Day Forecast: Detailed forecast for upcoming week.

ii. Multiple Locations: Save and switch between different places.

iii. Weather Alerts: Notify about severe weather conditions.

iv. Weather Maps: Interactive maps and radar views.

v. Air Quality Index: Include AQI data for user's area.

```
import 'package:flutter/material.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import 'package:intl/intl.dart';
import 'package:share/share.dart';
import 'package:weather/weather.dart';
import 'package:port/consts.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  final WeatherFactory _wf = WeatherFactory(OPENWEATHER_API_KEY);

  Weather? _weather;
  List<Weather> _forecast = [];
  TextEditingController _searchController = TextEditingController();

  void initState() {
    super.initState();
    _fetchWeather("Dhaka");
    _fetchWeatherForecast("Dhaka");
  }
```

```
void _fetchWeather(String location) {
  _wf.currentWeatherByCityName(location).then((w) {
    setState(() {
      _weather = w;
    });
  });
}
void _fetchWeatherForecast(String location) {
  _wf.fiveDayForecastByCityName(location).then((forecast) {
    setState(() {
      _forecast = forecast.take(7).toList();
    });
  });
}

Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: const Color.fromARGB(255, 100, 120, 253),
    body: _buildUI(),
  );
}

Widget _buildUI() {
  if (_weather == null || _forecast.isEmpty) {
    return const Center(
      child: CircularProgressIndicator(),
    );
  }
  return SingleChildScrollView(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        _locationHeader(),
        _searchLocation(),
        _dateTimeInfo(),
        _weatherIcon(),
        _currentTemp(),
        _extraInfo(),
        _weatherForecast(),
      ],
    ),
  );
}
Widget _locationHeader() {
  return Text(
    _weather?.areaName ?? "",
    style: const TextStyle(
      fontSize: 60,
      fontWeight: FontWeight.w500,
      color: Colors.white,
    ),
  );
}
```

```
Widget _searchLocation() {
  return Padding(
    padding: const EdgeInsets.all(16.0),
    child: Row(
      children: [
        Expanded(
          child: TextField(
            controller: _searchController,
            style: const TextStyle(color: Colors.white),
            decoration: const InputDecoration(
              hintText: 'Search location...',
              hintStyle: TextStyle(color: Colors.white70),
            ),
          ),
        ),
        IconButton(
          icon: const Icon(Icons.search, color: Colors.white),
          onPressed: () {
            String searchLocation = _searchController.text.trim();
            if (searchLocation.isNotEmpty) {
              _fetchWeather(searchLocation);
              _fetchWeatherForecast(searchLocation);
            }
          },
        ),
      ],
    ),
  );
}
Widget _dateTimeInfo() {
  DateTime now = _weather!.date!;
  return Column(
    children: [
      Text(
        DateFormat("h:mm a").format(now),
        style: const TextStyle(
          fontSize: 35,
          color: Color.fromARGB(255, 255, 255, 255),
        ),
      ),
      const SizedBox(
        height: 10,
      ),
      Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Text(
            DateFormat("EEEE").format(now),
            style: const TextStyle(
              fontWeight: FontWeight.w700,
              color: Colors.teal,
            ),
          ),
```

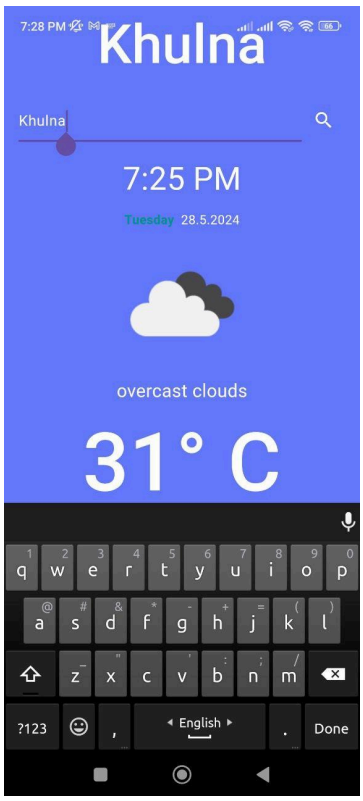
```
          Text(
            "  ${DateFormat("d.M.y").format(now)}",
            style: const TextStyle(
              fontWeight: FontWeight.w400,
              color: Colors.white,
            ),
          ),
        ],
      ),
    ],
  );
}
Widget _weatherIcon() {
  return Column(
    mainAxisSize: MainAxisSize.min,
    children: [
      Container(
        height: MediaQuery.of(context).size.height * 0.20,
        decoration: BoxDecoration(
          image: DecorationImage(
            image: NetworkImage(
               "http://openweathermap.org/img/wn/${_weather?.weatherIcon}@4x.png"),
          ),
        ),
      ),
      Text(
        _weather?.weatherDescription ?? "",
        style: const TextStyle(
          color: Color.fromARGB(255, 255, 255, 255),
          fontSize: 20,
        ),
      ),
    ],
  );
}

Widget _currentTemp() {
  return Text(
    "${_weather?.temperature?.celsius?.toStringAsFixed(0)}° C",
    style: const TextStyle(
      color: Colors.white,
      fontSize: 90,
      fontWeight: FontWeight.w500,
    ),
  );
}
Widget _extraInfo() {
  return Container(
    height: MediaQuery.of(context).size.height * 0.15,
    width: MediaQuery.of(context).size.width * 0.80,
    decoration: BoxDecoration(
      color: const Color.fromARGB(255, 103, 57, 220),
      borderRadius: BorderRadius.circular(20),
    ),
    padding: const EdgeInsets.all(8.0),
```

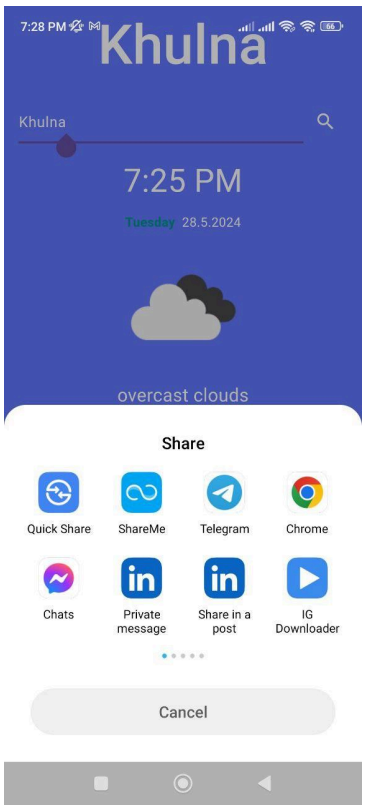
```
child: Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  crossAxisAlignment: CrossAxisAlignment.center,
  children: [
    Row(
      mainAxisSize: MainAxisSize.max,
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Text(
          "Max: ${_weather?.tempMax?.celsius?.toStringAsFixed(0)}° C",
          style: const TextStyle(
            color: Colors.white,
            fontSize: 20,
          ),
        ),
        Text(
          "Min: ${_weather?.tempMin?.celsius?.toStringAsFixed(0)}° C",
          style: const TextStyle(
            color: Colors.white,
            fontSize: 20,
          ),
        ),
      ],
    ),
    Row(
      mainAxisSize: MainAxisSize.max,
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Text(
          "Wind: ${_weather?.windSpeed?.toStringAsFixed(0)}m/s",
          style: const TextStyle(
            color: Colors.white,
            fontSize: 20,
          ),
        ),
        Text(
          "Humidity: ${_weather?.humidity?.toStringAsFixed(0)}%",
          style: const TextStyle(
            color: Colors.white,
            fontSize: 20,
          ),
        ),
      ],
    ),
    Flexible(
      child: IconButton(
        icon: const Icon(
          FontAwesomeIcons.share,
          color: Colors.white,
          size: 20,
        ),
        onPressed: () {
          String shareText =
```

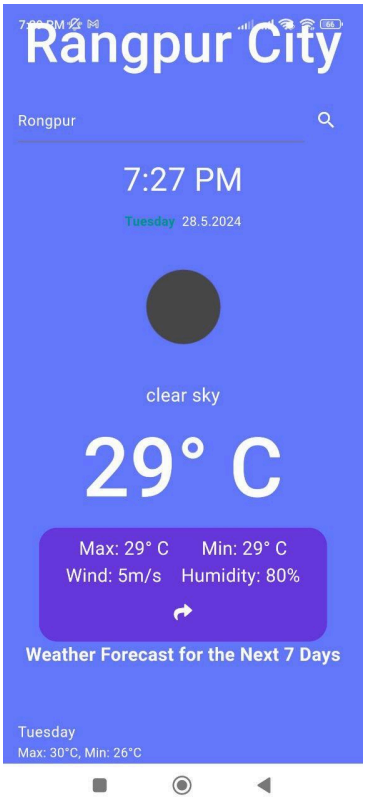
```
                "Current weather in ${_weather?.areaName}:\nMax:
${_weather?.tempMax?.celsius?.toStringAsFixed(0)}° C\nMin:
${_weather?.tempMin?.celsius?.toStringAsFixed(0)}° C\nWind:
${_weather?.windSpeed?.toStringAsFixed(0)}m/s\nHumidity:
${_weather?.humidity?.toStringAsFixed(0)}%";
              Share.share(shareText);
            },
          ),
        ),
      ],
    ),
  );
 }
 Widget _weatherForecast() {
  return Column(
    children: [
      const Text(
        "Weather Forecast for the Next 7 Days",
        style: TextStyle(
          fontSize: 20,
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ),
      ),
      const SizedBox(height: 10),
      ListView.builder(
        shrinkWrap: true,
        physics: const NeverScrollableScrollPhysics(),
        itemCount: _forecast.length,
        itemBuilder: (context, index) {
          Weather forecastItem = _forecast[index];
          return ListTile(
            title: Text(
              DateFormat("EEEE").format(forecastItem.date!),
              style: const TextStyle(
                color: Colors.white,
                fontSize: 16,
              ),
            ),
            subtitle: Text(
              "Max: ${forecastItem.tempMax?.celsius?.toStringAsFixed(0)}°C, "
              "Min: ${forecastItem.tempMin?.celsius?.toStringAsFixed(0)}°C",
              style: const TextStyle(
                color: Colors.white,
                fontSize: 14,
              ),
            ),
          );
        },
      ),
    ],
  );
 }
}
```


Dhaka
Search location...
7:20 PM
Tuesday 28.5.2024
haze
30° C
Max: 30° C Min: 30° C
Wind: 5m/s Humidity: 84%
Weather Forecast for the Next 7 Days
Tuesday
Max: 30°C, Min: 29°C


30° C
Max: 30° C Min: 30° C
Wind: 5m/s Humidity: 84%
Weather Forecast for the Next 7 Days
Tuesday
Max: 30°C, Min: 29°C
Wednesday
Max: 29°C, Min: 29°C
Wednesday
Max: 29°C, Min: 29°C
Wednesday
Max: 29°C, Min: 29°C
Wednesday
Max: 33°C, Min: 33°C
Wednesday
Max: 34°C, Min: 34°C
Wednesday
Max: 34°C, Min: 34°C


Khulna
Khulna
7:25 PM
Tuesday 28.5.2024
overcast clouds
31° C
Done


Khulna
Khulna
7:25 PM
Tuesday 28.5.2024
overcast clouds
Share
Quick Share
ShareMe
Telegram
Chrome
Chats
Private message
Share in a post
IG Downloader
Cancel


Rangpur City
Rongpur
7:27 PM
Tuesday 28.5.2024
clear sky
29° C
Max: 29° C Min: 29° C
Wind: 5m/s Humidity: 80%
Weather Forecast for the Next 7 Days
Tuesday
Max: 30°C, Min: 26°C


Delhi
Delhi
7:27 PM
Tuesday 28.5.2024
haze
44° C
Max: 44° C Min: 44° C
Wind: 4m/s Humidity: 6%
Weather Forecast for the Next 7 Days
Tuesday
Max: 44°C, Min: 43°C

7. Quiz App:

- Design a user-friendly interface for the Quiz App, encompassing questions, answer choices, and score tracking, with intuitive navigation and clear presentation of quiz elements.
- Develop the quiz logic, incorporating scoring and organization by categories or topics, ensuring immediate feedback to users after each question.
- Create diverse quiz modes, like timed quizzes and practice mode, to accommodate varied user preferences.
- Enable users to generate and share their quizzes, fostering community engagement and user-generated content.
- Integrate a leaderboard to track top scores and inspire friendly competition among users.

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

class QuizScreen extends StatefulWidget {
  const QuizScreen({super.key});
  _QuizScreenState createState() => _QuizScreenState();
}

class _QuizScreenState extends State<QuizScreen> {
  SharedPreferences? sharedPreferences;
  int highestScore = 0;
  int quizNumber = 1;
  int questionIndex = 0;
  int score = 0;
  bool isAnswered = false;
  String selectedCategory = "General Knowledge"; // Default category
  String userRole = 'student'; // Default role
  double quizTimeInSeconds = 0.25 * 60; // 15 minutes in seconds
  double timeRemaining = 0.25 * 60; // Initially set to quiz time
  List<String> categories = ["General Knowledge", "Science", "History"];
  Map<String, List<String>> categoryQuestions = {
    "General Knowledge": [
      'What is the capital of France?',
      'Who painted the Mona Lisa?',
      'What is the largest planet in our solar system?',
      'How many continents are there in the world?',
      'What is the largest mammal?',
    ],
    "Science": [
      'What is the largest planet in our solar system?',
      'Which gas do plants absorb from the atmosphere?',
      'What is the powerhouse of the cell?',
      'Who developed the theory of relativity?',
      'What is the chemical symbol for water?',
```

```
  ],
  "History": [
    'In which year did Christopher Columbus discover America?',
    'Who is known as the "Father of Modern Physics"?',
    'Who was the first President of the United States?',
    'When did World War II end?',
    'Who wrote "Romeo and Juliet"?',
  ],
};
Map<String, List<List<String>>> categoryOptions = {
  "General Knowledge": [
    ['Paris', 'London', 'Madrid', 'Rome'],
    ['Leonardo da Vinci',
      'Pablo Picasso',
      'Vincent van Gogh',
      'Claude Monet'
    ],
    ['Saturn', 'Mars', 'Earth', 'Jupiter'],
    ['5', '6', '7', '8'],
    ['Elephant', 'Giraffe', 'Blue Whale', 'Hippopotamus'],
  ],
  "Science": [
    ['Jupiter', 'Mars', 'Earth', 'Saturn'],
    ['Oxygen', 'Carbon Dioxide', 'Nitrogen', 'Hydrogen'],
    ['Mitochondria', 'Nucleus', 'Ribosome', 'Endoplasmic Reticulum'],
    ['Albert Einstein', 'Isaac Newton', 'Galileo Galilei', 'Niels Bohr'],
    ['H2O', 'CO2', 'O2', 'N2'],
  ],
  "History": [
    ['1492', '1607', '1776', '1789'],
    ['Albert Einstein', 'Isaac Newton', 'Galileo Galilei', 'Niels Bohr'],
    ['George Washington', 'John Adams', 'Thomas Jefferson', 'James Madison'],
    ['1945', '1940', '1949', '1939'],
    ['William Shakespeare', 'Jane Austen', 'Charles Dickens', 'Mark Twain'],
  ],
};

Map<String, List<String>> categoryCorrectAnswers = {
  "General Knowledge": [ 'Paris',
    'Leonardo da Vinci',
    'Jupiter',
    '7',
    'Blue Whale',
  ],
  "Science": [ 'Jupiter',
    'Carbon Dioxide',
    'Mitochondria',
    'Albert Einstein',
    'H2O',
  ],
  "History": [
    '1492',
    'Albert Einstein',
    'George Washington',
    '1945',
```

```
    'William Shakespeare',
  ],
};

List<String> questions = [];
List<List<String>> options = [];
List<String> correctAnswers = [];
List<String> selectedAnswers = [];

void shuffleQuestionsAndOptions() {
  final random = Random();
  for (var i = questions.length - 1; i > 0; i--) {
    final j = random.nextInt(i + 1);

    // Swap questions
    final tempQuestion = questions[i];
    questions[i] = questions[j];
    questions[j] = tempQuestion;

    // Swap options
    final tempOptions = options[i];
    options[i] = options[j];
    options[j] = tempOptions;

    // Swap correct answers
    final tempAnswer = correctAnswers[i];
    correctAnswers[i] = correctAnswers[j];
    correctAnswers[j] = tempAnswer;
  }
}

void initializeQuestions() {
  questions = categoryQuestions[selectedCategory]!;
  options = categoryOptions[selectedCategory]!;
  correctAnswers = categoryCorrectAnswers[selectedCategory]!;
}

@override
void initState() {
  super.initState();
  initializeSharedPreferences();
  initializeQuestions();
  shuffleQuestionsAndOptions();
  startQuizTimer();
}

void initializeSharedPreferences() async {
  sharedPreferences = await SharedPreferences.getInstance();
  setState(() {
    highestScore = sharedPreferences?.getInt('highestScore') ?? 0;
    userRole = sharedPreferences?.getString('role') ?? 'student';
  });
}

void updateHighestScore() async {
```

```
  final currentScore = sharedPreferences?.getInt('highestScore');
  if (currentScore != null) {
    if (score > currentScore) {
      await sharedPreferences?.setInt('highestScore', score);
      setState(() {
        highestScore = score;
      });
    }
  } else {
    await sharedPreferences?.setInt('highestScore', score);
    setState(() {
      highestScore = score;
    });
  }
}

void checkAnswer(String selectedOption) {
  if (isAnswered) {
    return; // Prevent multiple answer selections
  }

  String correctAnswer = correctAnswers[questionIndex];
  bool isCorrect = selectedOption == correctAnswer;

  setState(() {
    selectedAnswers.add(selectedOption);
    isAnswered = true;

    if (isCorrect) {
      score++;
      sharedPreferences?.setInt('highestScore', score);
    }
  });

  Future.delayed(const Duration(seconds: 1), () {
    setState(() {
      if (questionIndex < questions.length - 1) {
        questionIndex++;
        isAnswered = false;
        timeRemaining =
            quizTimeInSeconds; // Reset timer for the next question
      } else {
        // Quiz completed, perform any desired actions
        timeRemaining = quizTimeInSeconds; // Reset timer for the next quiz
        shuffleQuestionsAndOptions(); // Shuffle questions and options for the next quiz
      }
    });
  });
}

void startQuizTimer() {
  Future.delayed(const Duration(seconds: 1), () {
    setState(() {
      if (timeRemaining > 0) {
        timeRemaining--;
```

```
        startQuizTimer(); // Recursively call to update the timer
      } else {
        // Time's up, perform any desired actions here
        timeRemaining =
            quizTimeInSeconds; // Reset the timer for the next quiz
        shuffleQuestionsAndOptions(); // Shuffle questions and options for the next quiz
        // You can add actions to proceed to the next question or end the quiz
        // based on your requirements when the time is up.
      }
    });
  });
}

void shareScore() {
  String message =
      'I scored $score out of ${questions.length} in the quiz app!';
  Share.share(message);
}

void resetQuiz() {
  setState(() {
    selectedAnswers.clear();
    questionIndex = 0;
    quizNumber++;
    score = 0;
    isAnswered = false;
    timeRemaining = quizTimeInSeconds; // Reset timer for the next quiz
    shuffleQuestionsAndOptions(); // Shuffle questions and options for the next quiz
  });
}

void updateHighScore() {
  if (score > highestScore) {
    setState(() {
      highestScore = score;
    });
  }
}

String getQuizResult() {
  if (score >= 3) {
    return "Pass";
  } else {
    return "Fail";
  }
}

Color getResultColor() {
  if (score >= 3) {
    return const Color.fromARGB(255, 3, 112, 7);
  } else {
    return const Color.fromARGB(255, 0, 0, 0);
  }
}
```

```
Widget build(BuildContext context) {
  String result = getQuizResult();
  Color resultColor = getResultColor();

  return Scaffold(
   appBar: AppBar(
    backgroundColor: const Color.fromARGB(
       255, 134, 105, 251), // Change the app bar color to black
    title: const Text('Quiz App'),
    actions: [
     IconButton(
      icon: const Icon(Icons.logout),
      onPressed: () async {
       SharedPreferences prefs = await SharedPreferences.getInstance();
       await prefs.remove('role');
       Navigator.pushReplacementNamed(context, '/');
      },
     ),
    ],
   ),
   body: Container(
    color: const Color.fromARGB(
       255, 108, 87, 194), // Change the background color to black
    child: userRole == 'admin'
       ? AdminScreen()
       : Column(
          children: [
           const SizedBox(height: 30),
           DropdownButton<String>(
            value: selectedCategory,
            items: categories.map((String category) {
             return DropdownMenuItem<String>(
              value: category,
              child: Text(
               category,
               style: const TextStyle(
                color: Colors.white, // Change text color
               ),
              ),
             );
            }).toList(),
            onChanged: (String? value) {
             setState(() {
              selectedCategory = value!;
              initializeQuestions(); // Reload questions for the selected category
              shuffleQuestionsAndOptions(); // Shuffle questions and options
              questionIndex = 0; // Reset question index
              score = 0; // Reset score
              isAnswered = false; // Reset answer status
              timeRemaining = quizTimeInSeconds; // Reset timer
             });
            },
            style: const TextStyle(
             color: Colors.white, // Change button text color
            ),
```

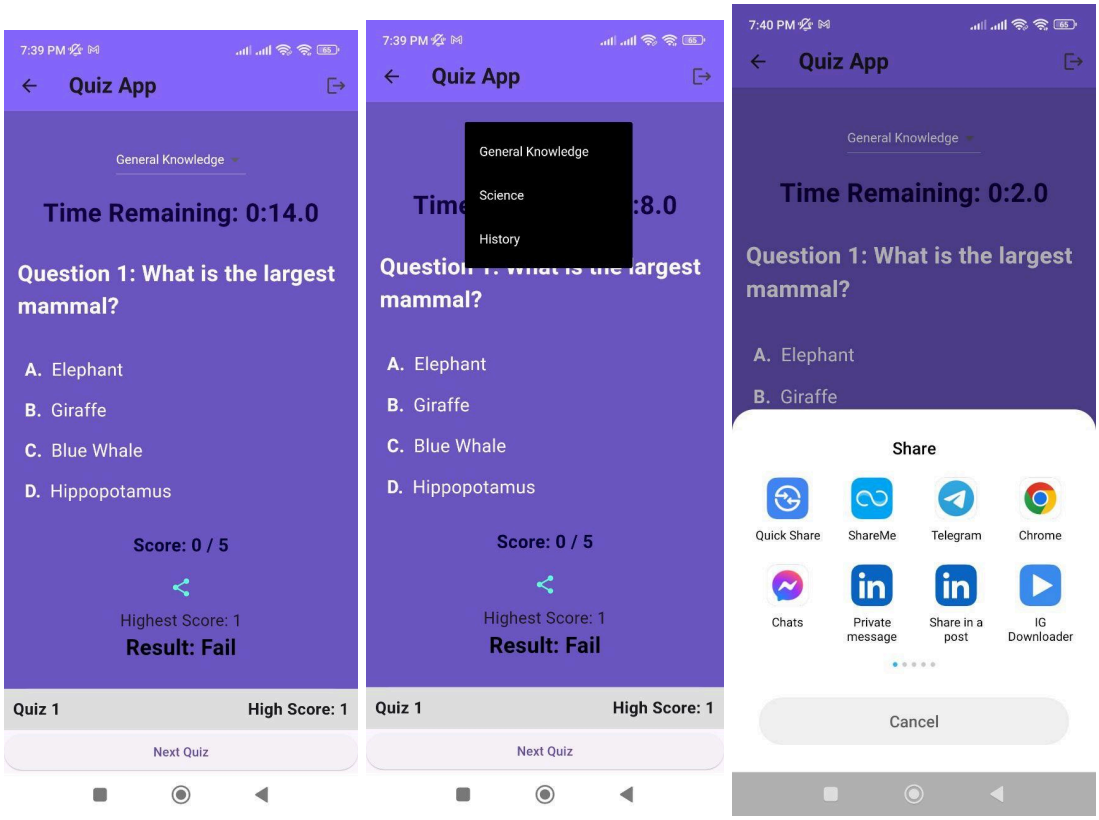
```
                elevation: 0, // Remove the shadow
                dropdownColor:
                    Colors.black, // Change the dropdown box color
              ),
              const SizedBox(height: 15),
              Text(
                'Time Remaining: ${(timeRemaining ~/ 60)}:${(timeRemaining %
60).toString().padLeft(2, '0')}',
                style: const TextStyle(
                    fontSize: 28,
                    fontWeight: FontWeight.bold,
                    color: Color.fromARGB(255, 17, 2, 64)),
              ),
              const SizedBox(height: 15),
              Padding(
                padding: const EdgeInsets.all(15.0),
                child: Text(
                  'Question ${questionIndex + 1}: ${questions[questionIndex]}',
                  style: const TextStyle(
                      fontSize: 25,
                      fontWeight: FontWeight.bold,
                      color: Colors.white),
                ),
              ),
              Padding(
                padding: const EdgeInsets.all(15.0),
                child: ListView.builder(
                  shrinkWrap: true,
                  itemCount: options[questionIndex].length,
                  itemBuilder: (context, index) {
                    bool isSelected = selectedAnswers
                        .contains(options[questionIndex][index]);
                    bool isCorrect = options[questionIndex][index] ==
                        correctAnswers[questionIndex];
                    bool showCorrectAnswer = isAnswered && isCorrect;

                    Color backgroundColor = Colors.transparent;
                    if (isSelected) {
                      backgroundColor =
                          isCorrect ? Colors.green : Colors.red;
                    } else if (showCorrectAnswer) {
                      backgroundColor = Colors.green;
                    }

                    return GestureDetector(
                      onTap: () {
                        if (!isSelected) {
                          checkAnswer(options[questionIndex][index]);
                        }
                      },
                      child: Container(
                        color: backgroundColor,
                        padding: const EdgeInsets.all(8),
                        child: Row(
                          children: [
```

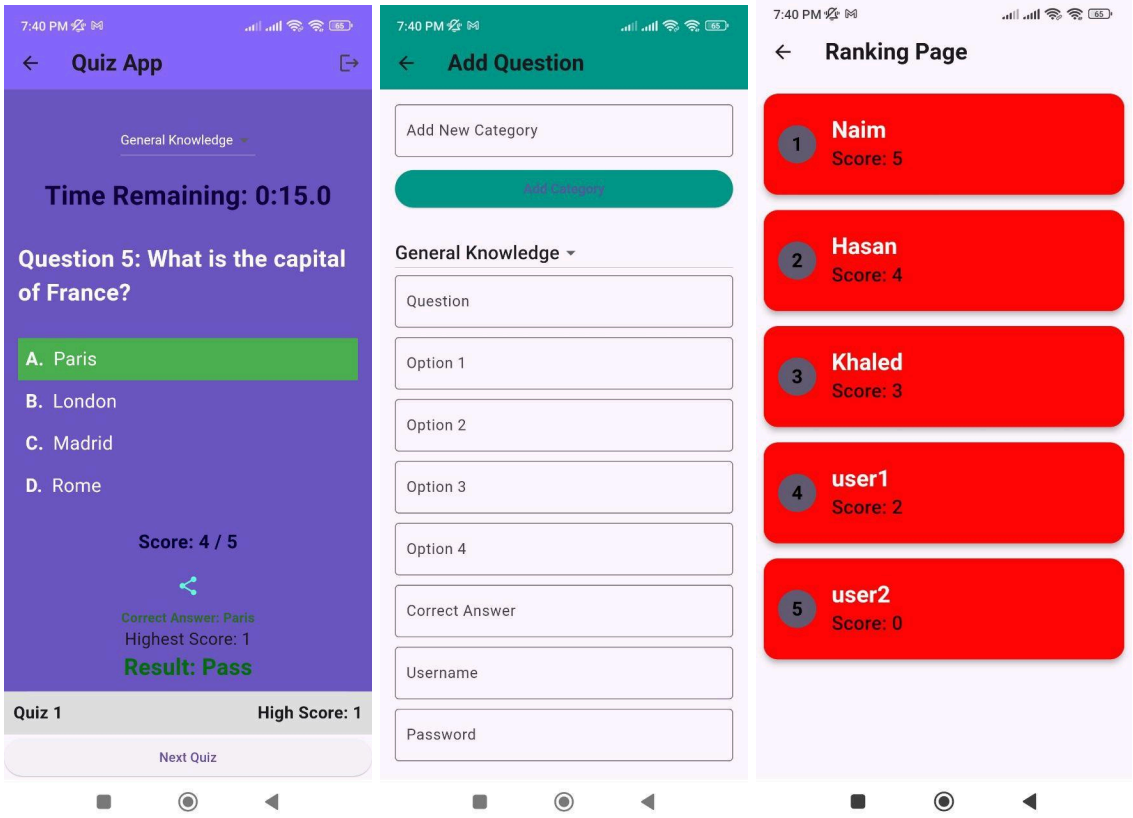
```
                Text(
                  '${String.fromCharCode(65 + index)}.',
                  style: const TextStyle(
                      fontWeight: FontWeight.bold,
                      fontSize: 20,
                      color: Color.fromARGB(255, 255, 255,
                          255) // Increase the font size
                      ),
                ),
                const SizedBox(width: 10),
                Text(
                  options[questionIndex][index],
                  style: TextStyle(
                    color: isSelected || showCorrectAnswer
                        ? Colors.white
                        : const Color.fromARGB(
                            255, 254, 254, 254),
                    fontSize: 20, // Increase the font size
                  ),
                ),
              ],
            ),
          ),
        );
      },
    ),
  ),
  const SizedBox(height: 8),
  Text(
    'Score: $score / ${questions.length}',
    style: const TextStyle(
        fontSize: 20,
        fontWeight: FontWeight.bold,
        color: Color.fromARGB(255, 3, 0, 54)),
  ),
  const SizedBox(height: 8),
  IconButton(
    icon: const Icon(Icons.share, color: Colors.tealAccent),
    onPressed: shareScore,
  ),
  const SizedBox(height: 0),
  if (selectedAnswers.contains(correctAnswers[questionIndex]))
    Text(
      'Correct Answer: ${correctAnswers[questionIndex]}',
      style: const TextStyle(
        color: Color.fromARGB(255, 47, 107, 49),
        fontWeight: FontWeight.bold,
      ),
    ),
  Text(
    'Highest Score: $highestScore',
    style: const TextStyle(
      fontSize: 18,
    ),
  ),
```

```
              // Display result and color signal
              Text(
                'Result: $result',
                style: TextStyle(
                  color: resultColor,
                  fontSize: 24,
                  fontWeight: FontWeight.bold,
                ),
              ),
            ],
          ),
    ),
    bottomNavigationBar: ElevatedButton(
      onPressed: () {
        updateHighScore();
        resetQuiz();
      },
      child: const Text('Next Quiz'),
    ),
    bottomSheet: Container(
      padding: const EdgeInsets.all(10),
      color: Colors.grey[300],
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Text(
            'Quiz $quizNumber',
            style: const TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
          Text(
            'High Score: $highestScore',
            style: const TextStyle(
              fontSize: 18,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    ),
  );
 }
}
class AdminScreen extends StatelessWidget {
 final TextEditingController _questionController = TextEditingController();
 final TextEditingController _option1Controller = TextEditingController();
 final TextEditingController _option2Controller = TextEditingController();
 final TextEditingController _option3Controller = TextEditingController();
 final TextEditingController _option4Controller = TextEditingController();
 final TextEditingController _correctAnswerController =
    TextEditingController();
 void addQuestion() {
  print('Question added');
```

```
  }
  Widget build(BuildContext context) {
   return Column(
    children: [
     TextField(
      controller: _questionController,
      decoration: const InputDecoration(labelText: 'Question'),
     ),
     TextField(
      controller: _option1Controller,
      decoration: const InputDecoration(labelText: 'Option 1'),
     ),
     TextField(
      controller: _option2Controller,
      decoration: const InputDecoration(labelText: 'Option 2'),
     ),
     TextField(
      controller: _option3Controller,
      decoration: const InputDecoration(labelText: 'Option 3'),
     ),
     TextField(
      controller: _option4Controller,
      decoration: const InputDecoration(labelText: 'Option 4'),
     ),
     TextField(
      controller: _correctAnswerController,
      decoration: const InputDecoration(labelText: 'Correct Answer'),
     ),
     ElevatedButton(
      onPressed: addQuestion,
      child: const Text('Add Question'),
     ),
    ],
   );
  }
}
class LoginScreen extends StatefulWidget {
  const LoginScreen({super.key});
  _LoginScreenState createState() => _LoginScreenState();
}
class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController _passwordController = TextEditingController();
  bool _isAdmin = false;

  void _login() async {
   String password = _passwordController.text;

   if (_isAdmin) {
    if (password == 'admin_password') {
     SharedPreferences prefs = await SharedPreferences.getInstance();
     await prefs.setString('role', 'admin');
     Navigator.pushReplacementNamed(context, '/quiz');
    } else {
     // Show an error message or handle incorrect password
    }
```

```
    } else {
      SharedPreferences prefs = await SharedPreferences.getInstance();
      await prefs.setString('role', 'student');
      Navigator.pushReplacementNamed(context, '/quiz');
    }
  }
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Login'),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            SwitchListTile(
              title: const Text('Login as Admin'),
              value: _isAdmin,
              onChanged: (bool value) {
                setState(() {
                  _isAdmin = value;
                });
              },
            ),
            if (_isAdmin)
              TextField(
                controller: _passwordController,
                decoration: const InputDecoration(labelText: 'Admin Password'),
                obscureText: true,
              ),
            ElevatedButton(
              onPressed: _login,
              child: const Text('Login'),
            ),
          ],
        ),
      ),
    );
  }
}
class MyApp extends StatelessWidget {
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Quiz App',
      theme: ThemeData(
        primarySwatch: Colors.teal,
      ),
      initialRoute: '/',
      routes: {
        '/': (context) => const LoginScreen(),
        '/quiz': (context) => const QuizScreen(),
      },
    );
  }
}
```


7:39 PM
Quiz App
General Knowledge
Time Remaining: 0:14.0
Question 1: What is the largest mammal?
A. Elephant
B. Giraffe
C. Blue Whale
D. Hippopotamus
Score: 0 / 5
Highest Score: 1
Result: Fail
Quiz 1
High Score: 1
Next Quiz
7:39 PM
Quiz App
General Knowledge
Science
History
Question 1: What is the largest mammal?
A. Elephant
B. Giraffe
C. Blue Whale
D. Hippopotamus
Score: 0 / 5
Highest Score: 1
Result: Fail
Quiz 1
High Score: 1
Next Quiz
7:40 PM
Quiz App
General Knowledge
Time Remaining: 0:2.0
Question 1: What is the largest mammal?
A. Elephant
B. Giraffe
Share
Quick Share
ShareMe
Telegram
Chrome
Chats
Private message
Share in a post
IG Downloader
Cancel


7:40 PM
Quiz App
General Knowledge
Time Remaining: 0:15.0
Question 5: What is the capital of France?
A. Paris
B. London
C. Madrid
D. Rome
Score: 4 / 5
Correct Answer: Paris
Highest Score: 1
Result: Pass
Quiz 1
High Score: 1
Next Quiz
7:40 PM
Add Question
Add New Category
Add Category
General Knowledge
Question
Option 1
Option 2
Option 3
Option 4
Correct Answer
Username
Password
7:40 PM
Ranking Page
1 Naim Score: 5
2 Hasan Score: 4
3 Khaled Score: 3
4 user1 Score: 2
5 user2 Score: 0

8. User Authentication:

Implement user authentication for saving quiz scores and personalizing profiles. As well as Implement two-factor authentication for security & have password recovery and profile management options.

9. Styling and Theming in Flutter App:

Maintain consistent styling and theming throughout the app for a unified visual experience. Implement dark mode and light mode themes, allow users to customize themes with color pickers, and ensure accessibility features like text size adjustment and high contrast themes.

10. Testing:

Make sure to check your app on different devices and screen sizes to make sure it works right. Also, use Flutter's testing tool to test automatically. And ask people to try your app and tell you what they think so you can make it better.

11. Deployment:

To get app ready for Android and iOS devices, follow Flutter's instructions to publish it on Google Play Store. Also, consider adding app analytics to track user behavior, incorporating in-app purchases for monetization, and ensuring compliance with app store guidelines and data privacy regulations.

12. Maintenance and Updates:

Keep app updated and maintain it regularly to address user feedback and fix bugs. Schedule feature updates based on user requests, implement an in-app feedback system, and monitor app performance and usage statistics for improvements.

Conclusion:

This Flutter app development process covers everything from setting up the environment to deploying and maintaining the app. It includes step-by-step instructions for creating different sections like portfolio, calculator, weather app and quiz app along with additional features like user authentication and theming. Each section provides clear guidance, making the development process straightforward. Overall, it's an easy-to-follow guide for anyone interested in Flutter app development.