REPORT

LAB 02

Student name: Nguyễn Đăng Nhã

Student ID: 21IT033

Student email: nhand.21it@vku.udn.vn

Github Link: https://github.com/dangnha/Mobile-Multiplatform/tree/master/Lab2/mi_card

1. Introduction

- Briefly describe the purpose of the lab report.
    - Create a simple contact information display app.
    - Includes a user profile with a photo, name (Angela Yu), job title ("FLUTTER DEVELOPER"), phone number, and email address.
- Provide background information on your mobile app.
    - Written in the Dart programming language using the Flutter framework.

2. Objectives

- State the objectives of the lab.
    - Display Contact Information: Present a user's contact information, including a profile picture, name, job title, phone number, and email address in a visually appealing manner.

3. Methodology

- Describe the methodology used in the lab.
    - Widget Hierarchy:
        - The UI is constructed using a hierarchy of widgets, each responsible for a specific part of the interface.
        - The entire app is encapsulated within the MyApp widget, which returns a MaterialApp containing a Scaffold with a structured layout.
    - Reusability:
        - Flutter promotes the reuse of widgets to compose complex UIs. In this program, widgets like CircleAvatar, Text, Card, ListTile, and Divider are used for building the UI components.
    - Layout Management:

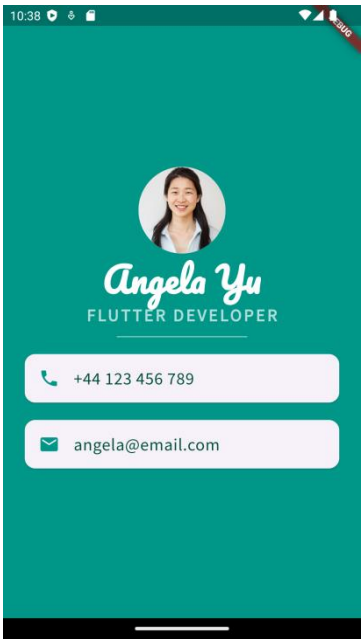
        - The Column widget is employed to vertically arrange child widgets. The SafeArea widget ensures that content is displayed within the safe area of the screen.
    - Styling and Theming:
        - Styling is achieved through the use of properties like backgroundColor, fontFamily, fontSize, color, and fontWeight.
- Explain how your app was developed.
    - Setting Up Development Environment:
        - Install Flutter SDK and Dart on the development machine.
        - Set up the development environment using an integrated development environment (IDE) like Visual Studio Code or Android Studio.
    - Creating a New Flutter Project.
    - Editing the main.dart File:
        - Open the main.dart file, which is the entry point of the Flutter application.
        - Import necessary Flutter packages, especially material.dart for Material Design components.
        - Defining the MyApp Class: Create a new class named MyApp that extends StatelessWidget. This class represents the entire application.
    - Implementing the UI:
        - Inside the build method of MyApp, use Flutter widgets to define the UI structure.
        - Nest widgets such as MaterialApp, Scaffold, SafeArea, Column, CircleAvatar, Text, Card, ListTile, and Divider to create the desired layout.
    - Styling and Theming:
        - Editing the 'pubspec.yaml' file to add fonts family
        - Apply styling by setting properties like backgroundColor, fontFamily, fontSize, color, and fontWeight to achieve the desired visual appearance.
    - Handling Contact Information:
        - Incorporate widgets like Card and ListTile to display contact information.
        - Use Flutter's layout management widgets like Column to organize the content in a structured way.
    - Running the App: on an emulator device.

4. Results

- Present the results of the lab.

    - Create a simple contact information display app.
    - Includes a user profile with a photo, name (Angela Yu), job title ("FLUTTER DEVELOPER"), phone number, and email address.

- Include screenshots of the app.



5. Discussion

- Discuss the results obtained.
    - Create a simple contact information display app.

- Analyze the strengths and weaknesses of cross-platform mobile app development.
    - Strengths:
        - Clarity and Readability: The code is well-organized, making it easy to understand and read. Proper indentation and consistent formatting contribute to code clarity.
        - Declarative UI: The program follows the declarative UI paradigm of Flutter, making it more intuitive and expressive in terms of UI design.
        - Widget Reusability: Widgets like Card, ListTile, and Divider are used effectively, promoting code reusability and modularity.
    - Weakness:
        - Limited Interactivity: The program lacks interactive elements, such as buttons or navigation, which limits its functionality. Adding interactive features could enhance user engagement.
        - Hardcoded Content: Contact information is hardcoded in the program. In a real-world scenario, data might come from dynamic sources (e.g., API calls, databases), and this program does not address dynamic data handling.

6. Conclusion

- Summarize the main findings of the lab.

    - Create a simple contact information display app.

    - Combine mayy widgets: Widgets are used extensively to compose the UI, promoting reusability and modularity. Notable widgets include Column, CircleAvatar, Text, Card, ListTile, and Divider.
- Provide recommendations for future work.
    - Add Interactivity:
        - Enhance the user experience by incorporating interactive elements like buttons or gestures. This could include navigation to other screens or actions triggered by user interactions.
    - Enhanced Design:
        - Consider adding animations or dynamic UI elements to make the application more engaging and visually appealing. This could include transitions between screens or subtle animations for a polished user experience.
    - Advanced Flutter Features:
        - Explore and integrate more advanced Flutter features and packages to showcase the full potential of Flutter, such as animations, custom paint, or complex UI interactions.