- Rivision Lab: Grocery App
    - Objective:
    - Project Structure
    - Requirements:
    - Task Breakdown:
        - Part A: Setting Up Models and API Integration
        - Part B: Building the UI and Navigation
        - Part C: Final Setup
    - Instructions:

Rivision Lab: Grocery App

Objective:

You are required to build a Flutter application that fetches product data from a web API, displays the products in a grid format, and shows detailed information about a selected product. Your project should follow the folder layout provided below.

Project Structure

```
main.dart
├── model
│   ├── product.dart
├── providers
│   ├── product_provider.dart
├── repositories
│   └── product_repository.dart
├── routes
│   └── app_router.dart
└── screens
    ├── product_screen.dart
    ├── product_details_screen.dart
```

Requirements:

1. **Fetch and Display Products:**

   - You are required to fetch product data from the endpoint `https://dummyjson.com/products`.
   - Display the products in a grid format with relevant information, such as an image, name, and price.

2. **Product Details Screen:**

   - When a user clicks on a product in the grid, navigate to a new screen and display detailed information about the selected product, such as the product name, description, image, and price.

Task Breakdown:

Part A: Setting Up Models and API Integration

1. Add the following dependencies

```
go_router: ^14.3.0
flutter_riverpod: ^2.5.1
dio: ^5.7.0
```

2. Create a model class in the `model` folder to represent the structure of a product.

```
-------------------------------
|          Product            |
-------------------------------
| - id: int                   |
| - title: String             |
| - description: String       |
| - price: double             |
| - image: String             |
| - rating: double            |
-------------------------------
| + Product()                 |
| + fromJson(Map<String,      |
|   dynamic> json): Product   |
-------------------------------
```

3. Create a Product Repository using Dio:

   - Implement a ProductRepository class in the repositories folder.

    - This class should use the Dio package to interact with the API.
    - Add methods to fetch all products and fetch product details by ID.

4. Set up a provider in the `providers` folder to fetch product data and manage the state.

Part B: Building the UI and Navigation

1. Implement a screen named `ProductScreen` to display products in a grid view layout.
2. Implement a screen named `ProductDetailsScreen` to show detailed information when a product is selected.
3. Create routes using a router class in the `routes` folder to manage navigation between screens.

Part C: Final Setup

1. Use the `main.dart` file to set up the main app with providers and routes.
2. Ensure that all components work together to achieve the required functionality.

Instructions:

- **Time**: 2 hours
- **Task**: Build the app based on the structure and requirements mentioned above.
- **Note**: Ensure that your app is fully functional and follows the provided structure.

Bonus: Add a search feature to filter products by name.