BÁO CÁO DỰ ÁN CUỐI KỲ

HỌC KỲ 2, NĂM HỌC 2023-2024

CT484: PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

- **Tên dự án/ứng dụng: Todo app**
- **Link GitHub mã nguồn:** 23-24Sem2-Courses/CT48401_Project
- **MSSV 1: B2014584**
- **Họ tên SV 1: Trương Nhật Minh**
- **MSSV 2: B2014627**
- **Họ tên SV 2: Ngô Huỳnh Công Trứ**
- **Lớp học phần: CT484-01**

I. Tổng quan

- **Miêu tả dự án/ứng dụng:** Ứng dụng đơn giản giúp người dùng ghi chú lại các công việc sẽ làm trong tương lai.

- Bảng phân công công việc:

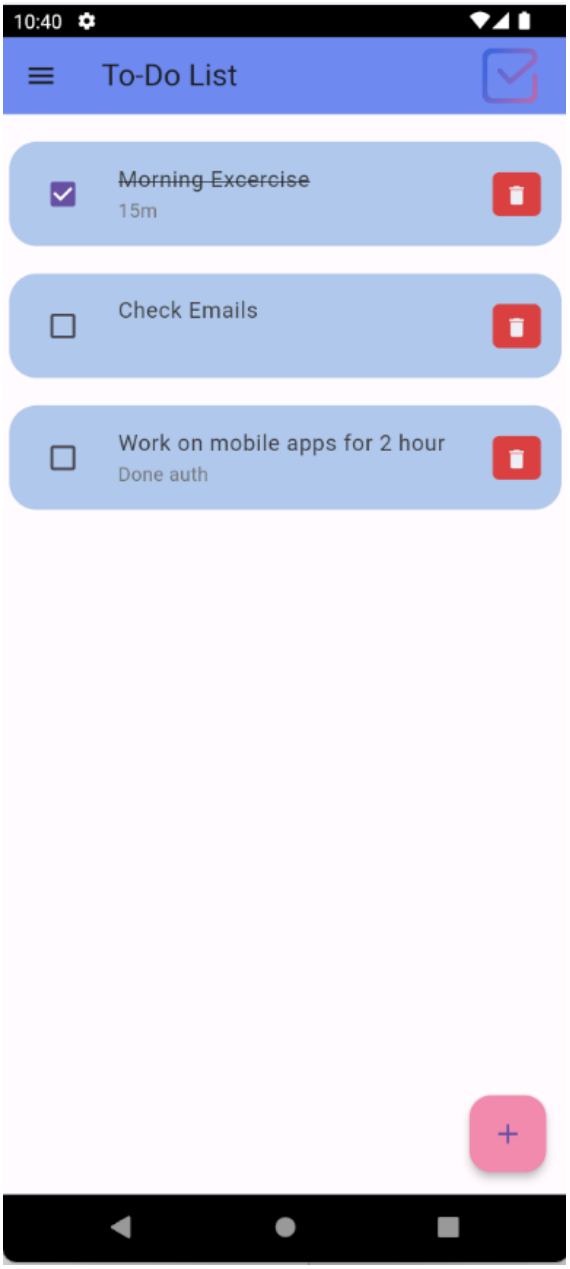
Trương Nhật Minh	Ngô Huỳnh Công Trứ
Thiết kế Front-End, viết báo cáo	Thiết kế Back-End, cơ sở dữ liệu

II. Chi tiết các chức năng

1. Giao diện: Giao diện trang chủ

- **Miêu tả giao diện:** Nơi để người dùng xem các công việc, đánh dấu hoàn thành hoặc xóa công việc họ đã nhập

- **Ảnh giao diện:**



- **Chi tiết cài đặt:**
  + Liệt kê các widget sử dụng:
    - AppBar: Hiển thị tiêu đề "To-Do List" và logo ở góc phải.
    - RefreshIndicator: Cho phép người dùng làm mới danh sách công việc bằng cách kéo xuống.
    - Consumer: Sử dụng để lắng nghe sự thay đổi trong ToDosManager và tái tạo danh sách công việc khi có sự thay đổi.
    - ListView.builder: Hiển thị danh sách các công việc.
    - ListTile: Hiển thị thông tin chi tiết của mỗi công việc, bao gồm tiêu đề, mô tả và checkbox.
    - AlertDialog: Dùng để tạo hộp thoại cho việc thêm và cập nhật công việc.
    - TextField: Để người dùng nhập tiêu đề và mô tả cho công việc.
  + Thư viện/plugin sử dụng:
    - Provider: Dùng để quản lý trạng thái và chia sẻ dữ liệu giữa các widget trong ứng dụng.
    - Flutter Material Components: Cung cấp các widget và hộp thoại chuẩn của Flutter như AppBar, AlertDialog, FloatingActionButton, và các widget khác.
  + Giải pháp quản lý trạng thái chia sẻ:
    - Sử dụng Provider để quản lý trạng thái chia sẻ giữa các widget trong ứng dụng. ToDosManager là một lớp được cung cấp cho các widget khác nhau trong ứng dụng để quản lý danh sách các công việc.

  + Trang thực hiện đọc, thêm, sửa, xóa, cập nhật các to-do thông qua ToDosManager và ToDosService.

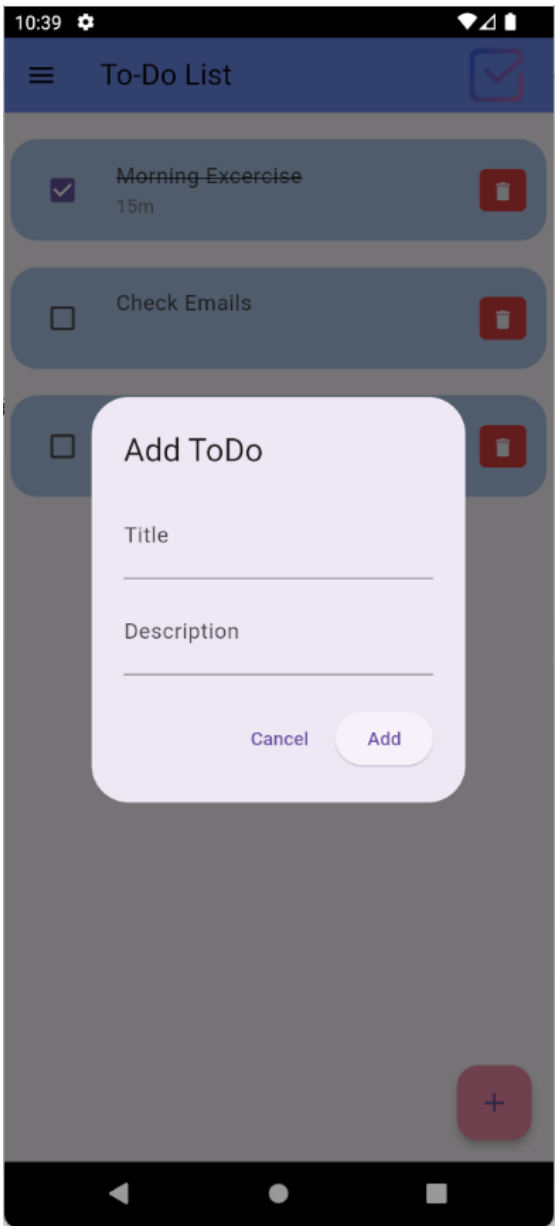
2. **Chức năng: ToDosManager và ToDosService**

- **Miêu tả chức năng:** thực hiện đọc, thêm, sửa, xóa, cập nhật các to-do theo yêu cầu của Home.
- **Chi tiết cài đặt ToDosManager:**
  Chịu trách nhiệm quản lý danh sách các công việc. Nó triển khai ChangeNotifier để có thể thông báo cho người nghe về bất kỳ sự thay đổi nào trong danh sách công việc.
  + Các Phương thức:
    - fetchToDos: Lấy danh sách công việc từ dịch vụ to-do service và cập nhật danh sách công việc hiện tại.
    - addToDo: Thêm một công việc mới vào danh sách công việc và thông báo cho người nghe về sự thay đổi.
    - updateToDo: Cập nhật thông tin của một công việc trong danh sách và thông báo cho người nghe về sự thay đổi.
    - toggleDoneStatus: Chuyển đổi trạng thái hoàn thành của một công việc và lưu trữ trạng thái mới vào dịch vụ.
    - deleteToDo: Xóa một công việc khỏi danh sách và cập nhật dịch vụ tương ứng.
  + Thuộc tính:
    - _items: Danh sách các công việc được quản lý bởi ToDosManager.
    - _todosService: Dịch vụ to-do service được sử dụng để tương tác với dữ liệu công việc (fetch, add, update, delete).
  + ToDosManager phụ thuộc vào một dịch vụ ToDosService để thực hiện các thao tác CRUD với danh sách công việc. Các thao tác này bao gồm lấy, thêm, cập nhật và xóa công việc.
  + Sau mỗi thay đổi trong danh sách công việc, ToDosManager thông báo cho các người nghe bằng cách gọi notifyListeners(), giúp cập nhật giao diện người dùng khi cần thiết.
- **Chi tiết cài đặt ToDosService:**
  Chịu trách nhiệm gửi yêu cầu HTTP đến cơ sở dữ liệu Firebase để thực hiện các thao tác CRUD trên danh sách công việc.
  + Các Phương thức:
    - fetchToDos: Lấy danh sách công việc từ cơ sở dữ liệu Firebase. Nếu được chỉ định, các công việc được lọc theo người dùng hiện tại.
    - addToDo: Thêm một công việc mới vào cơ sở dữ liệu Firebase và cập nhật lại danh sách công việc.
    - updateToDo: Cập nhật thông tin của một công việc trong cơ sở dữ liệu Firebase.
    - deleteToDo: Xóa một công việc khỏi cơ sở dữ liệu Firebase.

        - saveDoneStatus: Lưu trạng thái hoàn thành của một công việc trong cơ sở dữ liệu Firebase.
    + Gửi yêu cầu HTTP: Sử dụng phương thức httpFetch để gửi yêu cầu HTTP đến cơ sở dữ liệu Firebase. Phương thức này được triển khai bên ngoài và được cung cấp thông qua lớp cha FirebaseService.
    + Xử lý Dữ liệu: Khi nhận được dữ liệu từ cơ sở dữ liệu Firebase, đoạn mã sử dụng JSON để chuyển đổi dữ liệu thành các đối tượng ToDo và ngược lại.
    + Bảo mật: Các yêu cầu HTTP được thực hiện với token xác thực để đảm bảo tính bảo mật.
    + Xử lý Lỗi: Xử lý các trường hợp lỗi bằng cách in ra lỗi trong chế độ Debug hoặc trả về giá trị false để báo cáo lỗi.
- **Cấu trúc của một ToDo:**
    + Thuộc tính:
        - id: Định danh duy nhất của công việc.
        - todoText: Tiêu đề của công việc.
        - todoDescription: Mô tả chi tiết về công việc.
        - _isDone: Một ValueNotifier để theo dõi trạng thái hoàn thành của công việc
    + Constructor:
        - Constructor của ToDo chấp nhận các tham số bắt buộc là id, todoText, todoDescription và một tham số không bắt buộc isDone (mặc định là false).
        - ValueNotifier được sử dụng để tạo _isDone, mặc định là false.
    + Getter và Setter:
        - isDoneListenable: Trả về ValueNotifier để có thể theo dõi trạng thái hoàn thành của công việc.
        - isDone: Getter truy cập trực tiếp vào giá trị của _isDone.
        - isDone: Setter để cập nhật giá trị _isDone.
    + Phương thức:
        - copyWith: Tạo một bản sao mới của đối tượng ToDo với các thuộc tính được cung cấp.
        - toJson: Chuyển đổi ToDo thành một đối tượng Map để lưu trữ hoặc gửi dữ liệu qua mạng.
        - fromJson: Phương thức tĩnh để tạo một đối tượng ToDo từ một đối tượng Map.

3. **Chức năng: Chức năng thêm công việc**

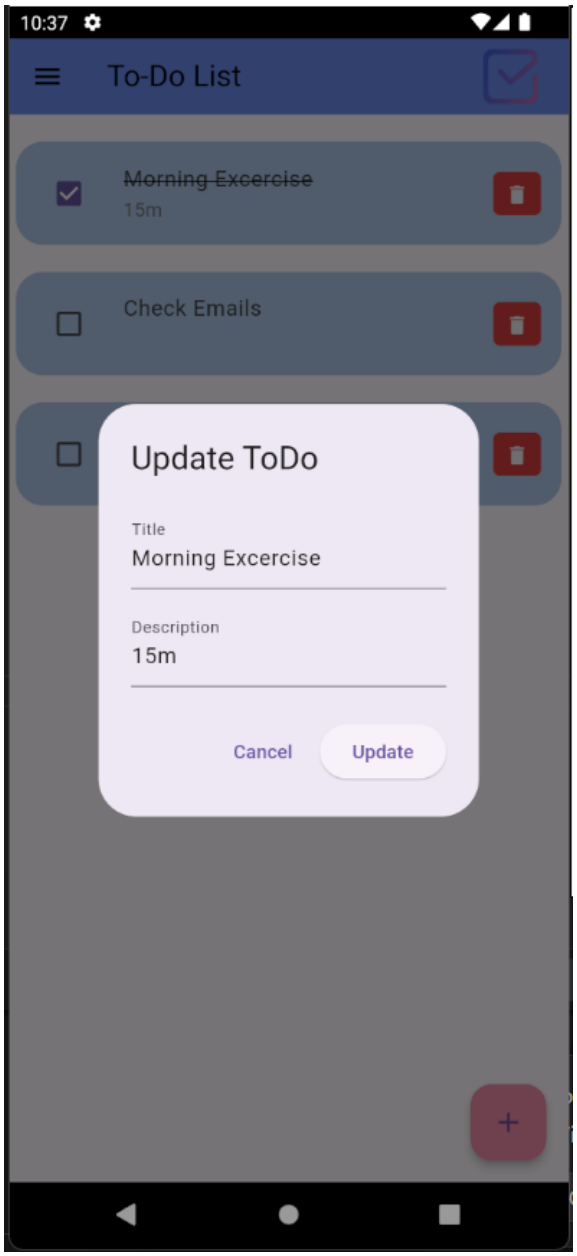
- **Miêu tả chức năng:** dùng để hiển thị cửa sổ để người dùng có thể thêm một công việc mới.
- **Ảnh chức năng:**



- **Chi tiết cài đặt:**
  + showDialog: Đây là một phương thức của lớp BuildContext để hiển thị một hộp thoại. Nó nhận vào một builder function, một hàm mà trả về một widget để hiển thị trong hộp thoại.
  + AlertDialog: Đây là một widget trong Flutter để hiển thị một hộp thoại cảnh báo. Nó chứa một tiêu đề và nội dung.
  + TextField: Đây là một widget để người dùng nhập văn bản. Ở đây, có hai TextField, một để nhập tiêu đề công việc và một để nhập mô tả công việc.
  + TextButton và ElevatedButton: Đây là các nút trong hộp thoại để thực hiện các hành động. Nút "Cancel" để đóng hộp thoại mà không thêm công việc mới. Nút "Add" để thêm công việc mới và đóng hộp thoại.
  + Navigator.pop(context): Dùng để đóng hộp thoại và quay lại màn hình trước đó.
  + _addNewToDo(): Hàm này được gọi khi người dùng nhấn nút "Add". Nó lấy thông tin từ hai TextField và tạo một công việc mới có tiêu đề và mô tả tương ứng. Sau đó, công việc mới này được thêm vào danh sách công việc thông qua _toDosManager.addToDo(newToDo).
    Cuối cùng, dữ liệu nhập của hai TextField được xóa để chuẩn bị cho việc nhập công việc mới tiếp theo.

4. **Chức năng: Chức năng cập nhật công việc:**

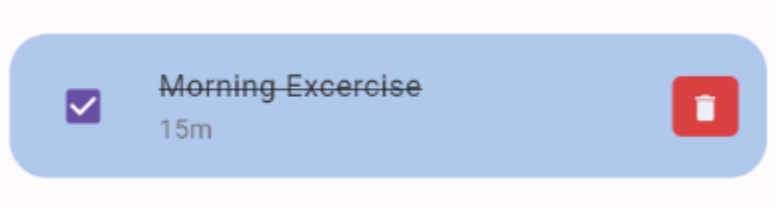
- **Miêu tả chức năng:** dùng để hiển thị cửa sổ thoả thuận để người dùng có thể cập nhật thông tin của một công việc đã tồn tại.
- **Ảnh chức năng:**



- **Chi tiết cài đặt:**
  + showDialog: là phương thức của lớp BuildContext để hiển thị một hộp thoại. Nó nhận vào một builder function để tạo nội dung của hộp thoại.
  + AlertDialog: Đây là một widget để hiển thị hộp thoại cảnh báo, chứa tiêu đề và nội dung.
  + TextField: Widget này cho phép người dùng nhập văn bản. Ở đây, có hai TextField, một để nhập tiêu đề công việc mới và một để nhập mô tả công việc mới. Dữ liệu hiện tại của công việc đang được cập nhật được hiển thị trong TextField.
  + TextButton và ElevatedButton: Hai nút này được sử dụng để thực hiện các hành động. Nút "Cancel" để đóng hộp thoại mà không cập nhật công việc. Nút "Update" để cập nhật thông tin công việc và đóng hộp thoại.
  + Navigator.pop(context): Dùng để đóng hộp thoại và quay lại màn hình trước đó.
  + _updateToDo(ToDo todo): Hàm này được gọi khi người dùng nhấn nút "Update". Nó lấy thông tin mới từ TextField và cập nhật công việc tương ứng trong danh sách công việc.

5. **Chức năng: Thay đổi trạng thái và xóa một công việc**

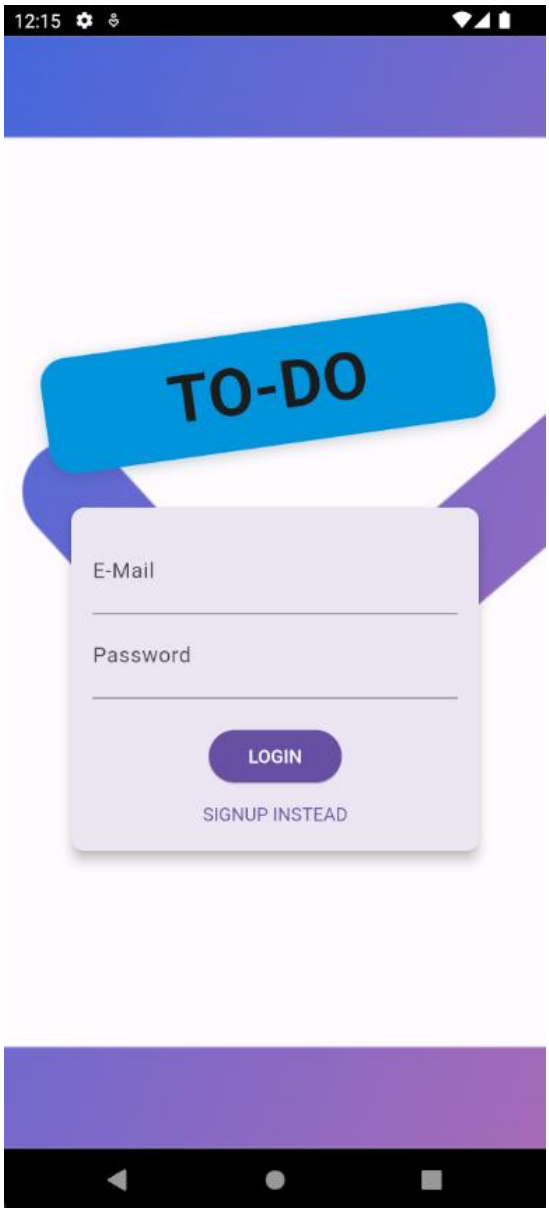
- **Miêu tả chức năng:** được sử dụng để thay đổi trạng thái của một công việc và xóa một công việc khỏi danh sách, tương ứng.
- **Ảnh chức năng:**

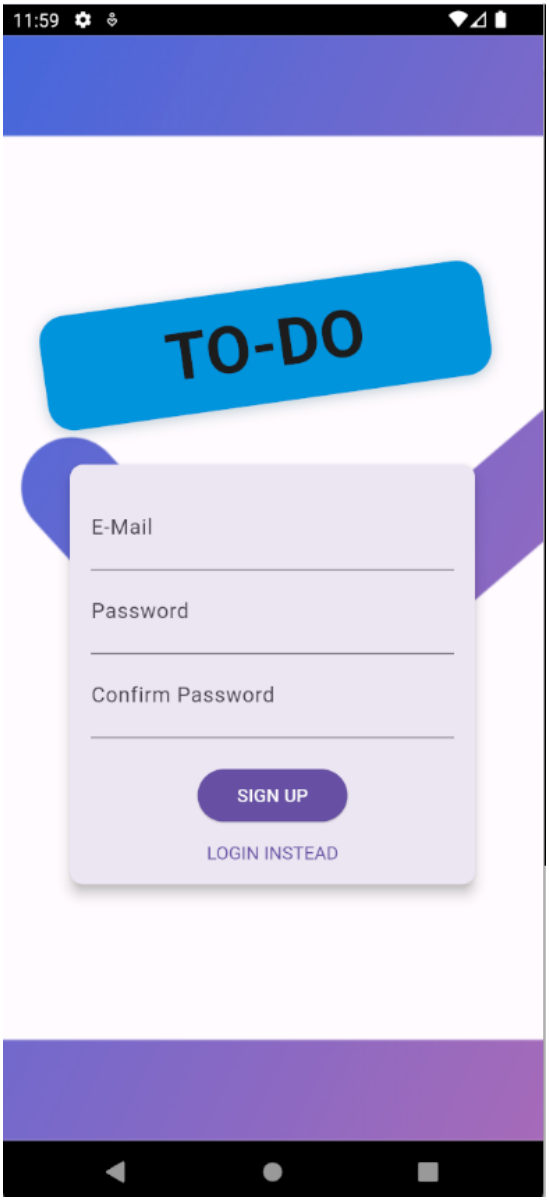


- **Chi tiết cài đặt thay đổi trạng thái của một công việc:**
  _toggleToDoStatus(ToDo todo, bool newValue)
  + Hàm này được gọi khi người dùng thay đổi trạng thái của một công việc bằng cách chọn hoặc bỏ chọn checkbox.
  + Nhận vào hai tham số: todo là công việc cần cập nhật trạng thái và newValue là giá trị mới cho trạng thái (đã hoàn thành hoặc chưa hoàn thành).
  + Tạo một bản sao của công việc với trạng thái mới bằng cách sử dụng phương thức copyWith của lớp ToDo.
  + Gọi _toDosManager.updateToDo(updatedToDo) để cập nhật công việc với trạng thái mới.
- **Chi tiết cài đặt xóa một công việc:**
  _deleteToDoItem(String id):
  + Hàm này được gọi khi người dùng muốn xóa một công việc khỏi danh sách.
  + Nhận vào một tham số id là ID của công việc cần xóa.
  + Gọi _toDosManager.deleteToDo(id) để yêu cầu _toDosManager xóa công việc có ID tương ứng khỏi danh sách.

6. **Chức năng/giao diện: Giao diện Đăng nhập và Đăng ký**

- **Miêu tả chức năng/giao diện:** nhiệm vụ quản lý thông tin xác thực người dùng và thực hiện các hành động liên quan đến việc đăng nhập và đăng ký.
- **Ảnh chức năng/giao diện:**





- **Chi tiết cài đặt giao diện:**
  + Danh sách các widget sử dụng:
    - Scaffold: Định nghĩa cấu trúc cơ bản cho trang, bao gồm app bar, bottom navigation bar, drawer, và nhiều hơn nữa.
    - Stack: Dùng để xếp các widget lên nhau.

        - Image.asset: Hiển thị hình ảnh từ tài nguyên được đặt trong thư mục assets.
        - SingleChildScrollView: Cho phép cuộn nội dung khi không đủ không gian.
        - Column: Sắp xếp các widget theo chiều dọc.
        - Flexible: Cho phép các widget con mở rộng hoặc co lại để điều chỉnh kích thước.
    + Quản lý trạng thái chia sẻ: Trong mã này, không có sử dụng giải pháp quản lý trạng thái chia sẻ như Provider.
- **Chi tiết cài đặt thẻ đăng nhập hoặc đăng ký:**
    + Danh sách các widget sử dụng:
        - Card: Định nghĩa một card có góc bo tròn.
        - Container: Đóng gói các widget con và quản lý kích thước của chúng.
        - Form: Được sử dụng để quản lý trạng thái và xác thực các trường đầu vào trong một biểu mẫu.
        - TextFormField: Để nhập dữ liệu từ người dùng, như email và mật khẩu.
        - ElevatedButton và TextButton: Để tạo các nút "Login" và "Sign up", và nút chuyển đổi giữa chế độ đăng nhập và đăng ký.
        - ValueListenableBuilder: Sử dụng để xây dựng UI dựa trên giá trị của một ValueNotifier, trong trường hợp này là _isSubmitting, để hiển thị một tiến trình khi đang thực hiện đăng nhập hoặc đăng ký.
    + Thư viện hay plugin sử dụng: provider: Được sử dụng để cung cấp và truy cập các đối tượng trong cả cây widget.
    + Quản lý trạng thái chia sẻ: Sử dụng Provider để cung cấp và truy cập vào AuthManager, một lớp quản lý trạng thái liên quan đến việc đăng nhập và đăng ký.
    + Đọc hoặc lưu trữ dữ liệu: Mã này sử dụng AuthManager để thực hiện việc đăng nhập và đăng ký, nhưng không có chi tiết cụ thể về cách dữ liệu được lưu trữ hoặc xử lý. Có thể giả định rằng AuthManager tương tác với một dịch vụ nào đó, chẳng hạn như một API hoặc cơ sở dữ liệu, để thực hiện các thao tác này. Đối với lỗi xảy ra trong quá trình đăng nhập hoặc đăng ký, mã sử dụng HttpException để xác định và hiển thị thông báo lỗi cho người dùng.
- **Chi tiết cài đặt AuthManager:**
    + Các thành phần chính:
        - AuthToken: Một lớp đại diện cho thông tin về mã xác thực người dùng, bao gồm token chính và thời gian hết hạn.
        - AuthService: Một dịch vụ cung cấp các phương thức để thực hiện đăng nhập, đăng ký và quản lý thông tin xác thực của người dùng.

+ Các phương thức:
    - signup: Phương thức để thực hiện việc đăng ký người dùng mới, gọi phương thức tương ứng từ AuthService và sau đó cập nhật AuthToken.
    - login: Phương thức để thực hiện việc đăng nhập, tương tự như signup.
    - tryAutoLogin: Phương thức để thử tự động đăng nhập người dùng dựa trên thông tin đã được lưu trữ trước đó.
    - logout: Phương thức để đăng xuất người dùng, xóa thông tin xác thực và hủy bỏ bất kỳ hẹn giờ tự động nào đang chạy.
    - _autoLogout: Phương thức được gọi để thiết lập hẹn giờ tự động đăng xuất dựa trên thời gian hết hạn của token xác thực.
+ Quản lý trạng thái và thông báo thay đổi:
    - ChangeNotifier được mở rộng để thông báo về bất kỳ thay đổi nào trong thông tin xác thực và trạng thái của người dùng.
    - Khi thông tin xác thực thay đổi (ví dụ: người dùng đăng nhập hoặc đăng xuất), notifyListeners() được gọi để thông báo cho các widget liên quan cập nhật giao diện người dùng.

7. Chức năng: FirebaseService

- **Miêu tả chức năng:** Để tương tác với dịch vụ Firebase Realtime Database.
- **Chi tiết cài đặt:**
    + Các thành phần chính:
        - httpFetch: Phương thức chính để thực hiện các yêu cầu HTTP đến Firebase Realtime Database, bao gồm các phương thức như GET, POST, PUT, PATCH và DELETE. Phương thức này nhận vào một URL, một phương thức yêu cầu, các tiêu đề tùy chọn và một thân yêu cầu (nếu cần). Sau đó, nó gửi yêu cầu HTTP tương ứng và trả về dữ liệu phản hồi.
        - token và userId: Dùng để lưu trữ thông tin về token xác thực và ID người dùng, được sử dụng để xác thực yêu cầu đến Firebase.
    + Thư viện và gói phụ thuộc:
        - http: Được sử dụng để thực hiện các yêu cầu HTTP đến Firebase Realtime Database.
        - dotenv: Sử dụng để tải các biến môi trường từ tệp .env, bao gồm cấu hình URL của Firebase Realtime Database.
        - json: Sử dụng để mã hóa và giải mã dữ liệu JSON.
    + Lớp trừu tượng và phương thức trừu tượng:
        - FirebaseService được định nghĩa là một lớp trừu tượng, giúp tái sử dụng và mở rộng mã.

- Phương thức httpFetch được định nghĩa là một phương thức trừu tượng, để các lớp con có thể triển khai lại để cung cấp các chức năng cụ thể cho mỗi yêu cầu HTTP.