### ****Project Description: CraveCouriers****

**CraveCouriers** is a feature-rich food delivery app developed using **Flutter** for the frontend and **Firebase** for backend services. The app offers users a seamless and intuitive platform to browse menus, customize food orders, and track their deliveries in real-time. Aimed at providing convenience, CraveCouriers combines responsive design, secure payment processing, and real-time order updates, making it an ideal solution for busy individuals looking for a reliable food delivery service.

#### ****Key Features:****

* **User Authentication**: Users can create accounts and log in using secure email-based Firebase authentication.
* **Restaurant Menu Browsing**: A clean, categorized menu interface allowing users to browse food items like burgers, salads, desserts, and drinks, with options to add extras to their meals.
* **Cart Management**: Users can add items to the cart, modify the quantity, and view the total price before checkout.
* **Payment Integration**: Secure card payment functionality is integrated to ensure safe transactions. The system is designed to handle payment errors, ensuring a smooth experience.
* **Order Tracking**: Real-time order status updates are synced with Firebase Firestore, providing users with up-to-date delivery progress.
* **Light Mode**: The app supports light mode, allowing users to switch between themes based on their preferences.

#### ****Technology Stack:****

* **Frontend**: Built with **Flutter**, ensuring smooth performance and compatibility across both Android and iOS devices.
* **Backend**: Powered by **Firebase Firestore** and **Firebase Authentication** for secure user data handling and real-time database syncing.
* **Payment Processing**: Secure payments
* **State Management**: Implemented using the **Provider** package to efficiently manage app-wide states like user authentication and cart updates.

#### ****Target Audience:****

**CraveCouriers** caters to users who desire a fast, easy, and secure way to order food from nearby restaurants. The app is ideal for working professionals, students, and anyone seeking a reliable food delivery service without complicated procedures.

#### ****Objective:****

The primary goal of **CraveCouriers** is to offer a highly functional food delivery solution with a focus on user experience, security, and real-time order tracking, making the food ordering process hassle-free and enjoyable.

### ****1. Real World Problem Identification****

The need for fast, convenient, and reliable food delivery services has risen in recent years. Users expect apps that allow them to browse a wide selection of restaurants, customize orders, track deliveries in real-time, and pay securely. Many existing solutions either have complicated user interfaces or lack real-time updates on orders. **CraveCouriers** addresses these issues by providing a smooth user experience with an intuitive interface, live order tracking, and secure payment options.

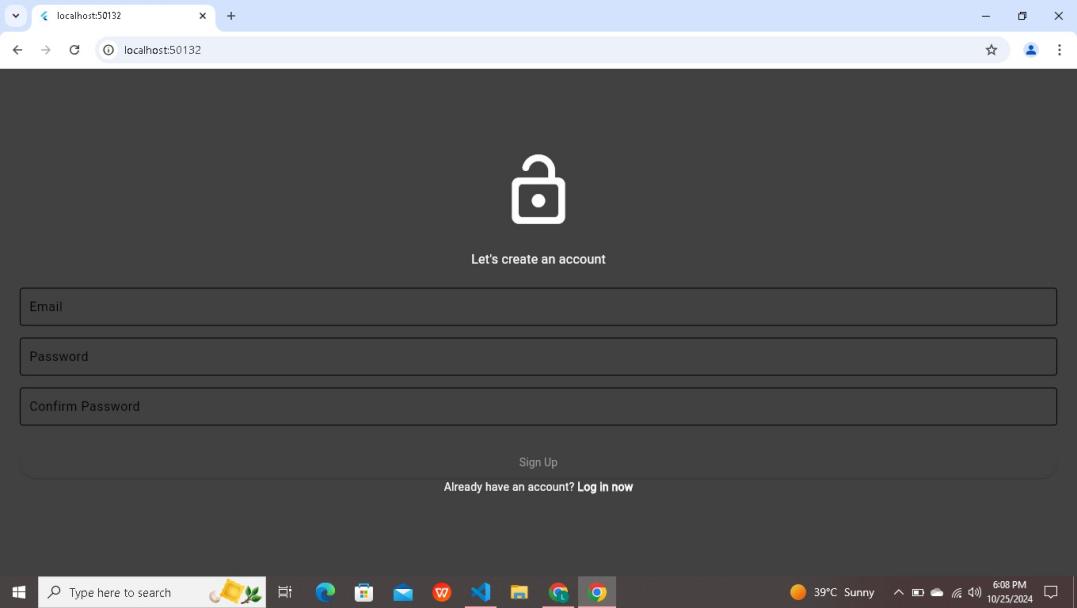
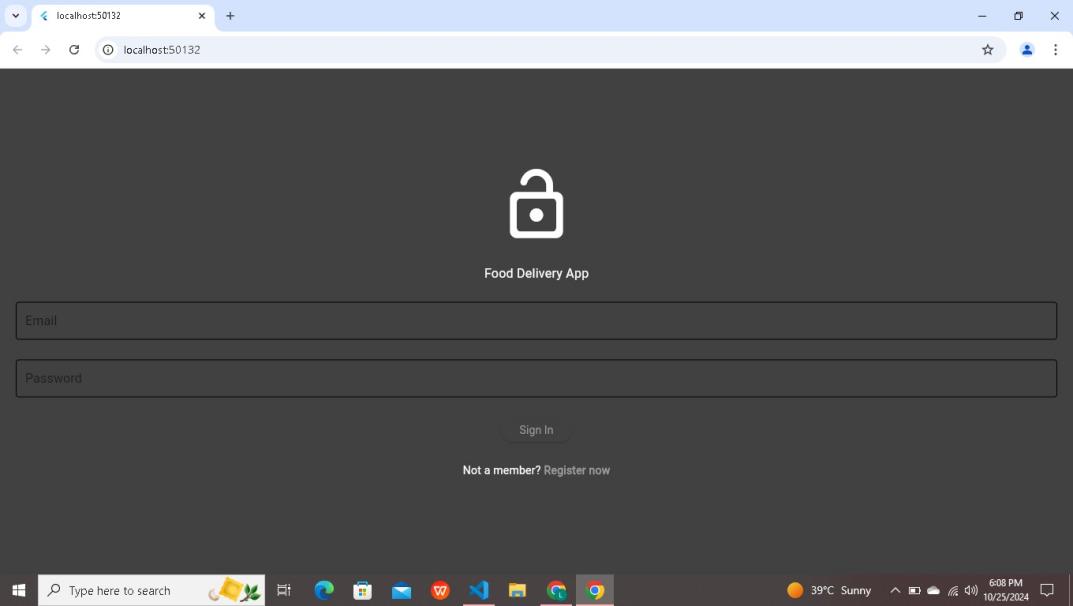
### ****2. Proposed Solution****

**CraveCouriers** is a food delivery app developed using **Flutter** and **Firebase**. It provides users with a streamlined experience to browse restaurant menus, add food to their cart, customize items with addons, and securely pay through an integrated card payment system. The app’s backend is powered by Firebase, ensuring real-time updates on order progress for users to track their transactions.

### ****3. Responsive User Interfaces****

The app has been designed to adapt to **light mode** to cater to user preferences. Key interface components include:

* **Login Page**: Users can sign in or create an account using Firebase authentication.
* **Register Page**: New users can create an account by providing an email and password.
* **Menu Drawer**: Includes navigation links for different sections like the settings page, orders, and more.
* **Settings Page**: Users can toggle between light/dark modes and adjust other preferences.
* **Sliver App Bar and Tab Bar**: Used for smooth scrolling and easy navigation between food categories.
* **Food Page**: Displays food items categorized into burgers, salads, sides, desserts, and drinks.
* **Cart Page**: Users can review their orders, modify quantities, and proceed to checkout.
* **Delivery Progress Page**: Shows real-time updates of the delivery.



### ****4. Data Storage****

The app uses **Firebase Firestore** as the primary database to store user data, including authentication details, order history, food categories, and live delivery updates. Firebase was chosen due to its seamless integration with Flutter, real-time data sync capabilities, and robust security features. Firestore allows scalability, which is essential for growing user bases.

### ****5.APIs/Packages/Plug-ins****

Several packages and plug-ins are used in **CraveCouriers** to extend its functionality:

* **Firebase Auth**: For managing user authentication securely.
* **Firebase Firestore**: For real-time data storage and retrieval.
* **Provider**: A state management package for Flutter to manage app-wide data efficiently.
* **HTTP**: For making network requests to external APIs (if applicable).

These were chosen for their compatibility with Flutter, ease of integration, and robust community support.

### ****6. Issues and Bugs Encountered and Resolved during Development****

During the development of **CraveCouriers**, several challenges were encountered:

**Authentication Bugs**: Initially, there were issues with Firebase email authentication where login attempts failed due to improper token handling. This was resolved by ensuring proper configuration of Firebase rules and updating the authentication state management logic.

**Cart Quantity Management**: A bug where the quantity of items in the cart would not update correctly was identified. This was fixed by correctly updating the state using the **Provider** package.

**Payment Processing Errors**: Some issues were found when processing card payments, particularly with handling invalid cards. Implementing error-checking and validation logic on the payment page resolved this issue.

**Real-time Updates for Delivery**: Synchronizing real-time updates between the user's app and Firebase posed some challenges, especially with network latency. This was improved by optimizing Firestore queries and caching techniques to ensure smoother updates.

### ****Project Link****

https://github.com/Lajwantiharani/pro