**Project Directory Structure**

graphql

Copy code

myproject/

│

├── app.py # Main Flask application file

├── requirements.txt # Python dependencies

├── config.py # Configuration file (Firebase, API keys, etc.)

├── README.md # Project documentation

├── static/ # Static files (CSS, JavaScript, Images)

│ ├── css/

│ │ └── styles.css # CSS for styling

│ ├── js/

│ │ ├── admin.js # JavaScript for admin dashboard

│ │ ├── user.js # JavaScript for user interaction

│ │ └── reviews.js # JavaScript for handling reviews

│ ├── images/ # Image assets

│ │ └── logo.png # Project logo

│

├── templates/ # HTML templates

│ ├── index.html # Homepage for users

│ ├── register.html # Registration page

│ ├── login.html # Login page for users/admins

│ ├── admin\_dashboard.html # Admin dashboard page

│ ├── products.html # Products listing page

│ ├── branches.html # Branches listing page

│ ├── reviews.html # Reviews submission page

│ └── error.html # Error handling page

│

├── firebase/ # Firebase-specific files

│ ├── service\_account.json # Firebase Admin SDK credentials

│ ├── firebase\_admin.py # Firebase initialization and helpers

│

├── api/ # Flask API routes for dynamic functionality

│ ├── admin\_routes.py # Routes for admin operations (products, branches)

│ ├── user\_routes.py # Routes for user operations (registration, login)

│ └── review\_routes.py # Routes for customer reviews

│

├── utils/ # Utility functions and helpers

│ ├── auth.py # Authentication and authorization logic

│ ├── sms\_notifier.py # SMS sending functionality

│ ├── validators.py # Input validation functions

│

├── migrations/ # Database migrations (if applicable)

│

└── logs/ # Log files

└── app.log # Application logs

**File/Folder Details**

**1. app.py**

* Entry point for the Flask application.
* Initializes the app, loads configurations, and registers routes.

**2. requirements.txt**

* Lists Python dependencies, e.g.:

Copy code

Flask

firebase-admin

Twilio

requests

* Install dependencies with:

Copy code

pip install -r requirements.txt

**3. config.py**

* Contains configuration settings like Firebase URLs, API keys, and environment variables:

python

Copy code

FIREBASE\_CONFIG = {

"type": "service\_account",

"project\_id": "your\_project\_id",

...

}

SMS\_API\_KEY = "your\_sms\_api\_key"

**4. static/**

* Holds static assets:
  + css/styles.css: Main stylesheet.
  + js/: Contains JavaScript files for dynamic behavior.
  + images/: Stores image files, such as logos or banners.

**5. templates/**

* Stores HTML templates:
  + index.html: User homepage.
  + register.html: Registration form.
  + login.html: Login form for users and admins.
  + admin\_dashboard.html: Dashboard for managing products and branches.
  + products.html: Displays product listings.
  + branches.html: Displays branch locations.
  + reviews.html: Form for submitting customer reviews.
  + error.html: Generic error page.

**6. firebase/**

* service\_account.json: Private key file for Firebase Admin SDK.
* firebase\_admin.py: Initializes Firebase:

python

Copy code

import firebase\_admin

from firebase\_admin import credentials

cred = credentials.Certificate("firebase/service\_account.json")

firebase\_admin.initialize\_app(cred, {

"databaseURL": "https://your-database.firebaseio.com"

})

**7. api/**

* **admin\_routes.py:**
  + Contains Flask routes for:
    - Adding, updating, and deleting products.
    - Managing branches.
  + Example route:

python

Copy code

@app.route("/admin/add-product", methods=["POST"])

def add\_product():

# Add product logic

* **user\_routes.py:**
  + Handles user-related functionality like registration and login.
  + Example route:

python

Copy code

@app.route("/register", methods=["POST"])

def register():

# User registration logic

* **review\_routes.py:**
  + Routes for collecting customer reviews and sending them via SMS.
  + Example route:

python

Copy code

@app.route("/submit-review", methods=["POST"])

def submit\_review():

# Review submission logic

**8. utils/**

* **auth.py:** Authentication helpers, e.g., verifying admin accounts.
* **sms\_notifier.py:** Sends SMS notifications to admins using Twilio or similar API.
* **validators.py:** Contains reusable validation functions for input fields.

**9. migrations/**

* Placeholder for database migrations if you decide to integrate a relational database like PostgreSQL or SQLite.

**10. logs/**

* Store application logs for debugging and monitoring.
* Example app.log:

yaml

Copy code

[2025-01-15 12:00:00] INFO: User registered successfully.

[2025-01-15 12:01:00] ERROR: Failed to add product.