Online Grocery Store Project Documentation

# 1. Project Overview

Project Name: Online Grocery Store

Objective: To develop a fully functional online grocery store with user registration, login, product browsing, cart, order placement, and admin management.

Technologies Used: Python (Flask), SQLite Database, HTML, CSS, JavaScript.

Key Features:

- Customer registration and login system.

- Admin login and management system.

- Product catalog with details.

- Shopping cart and order placement.

- Stylish UI with animations and 3D effects.

- Password recovery ('Forgot Password') feature.

- Order history for customers.

- Admin dashboard analytics.

- Email notifications for orders.

Detailed Overview:

- Each feature has been designed for a seamless user experience.

- Frontend and backend communicate via Flask routes.

- Database design ensures data integrity and fast retrieval.

- Animations and UI effects enhance the look and feel of the store.

# 2. System Architecture

Frontend: HTML, CSS, JavaScript with animations and 3D hover effects.

Backend: Flask handles requests, session management, and database operations.

Database: SQLite database with relational tables for Users, Products, Admin, and Orders.

Detailed Architecture:

- Frontend sends HTTP requests to Flask backend.

- Flask backend interacts with SQLite database to fetch/store data.

- Data is rendered dynamically on the HTML templates using Jinja2.

- Admin panel provides full control over products and user management.

Flow Diagram: [Insert detailed flowchart showing homepage -> login/register -> product pages -> cart -> order placement -> admin management]

# 3. Database Design

Tables and Fields:

- Users: id (PK), name, email, password, address, phone, registration\_date

- Admin: id (PK), username, password, email

- Products: id (PK), name, description, price, quantity, category, image

- Orders: id (PK), user\_id (FK), product\_id (FK), quantity, total\_price, order\_date, status

ER Diagram: [Insert detailed ER diagram with relationships, primary and foreign keys]

Sample SQL Queries:

CREATE TABLE Users (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

email TEXT UNIQUE NOT NULL,

password TEXT NOT NULL,

address TEXT,

phone TEXT,

registration\_date TEXT

);

# 4. Backend (app.py) Explanation

Initialization and Imports:

from flask import Flask, render\_template, request, redirect, url\_for, session

import sqlite3

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

Database Connection:

def get\_db\_connection():

conn = sqlite3.connect('grocery.db')

conn.row\_factory = sqlite3.Row

return conn

Routes:

- Home Page (/)

- Customer Registration (/register)

- Customer Login (/login)

- Admin Login (/admin\_login)

- Products Page (/products)

- Place Order (/place\_order)

- Logout (/logout)

Additional Backend Features:

- Password hashing

- Input validation

- Session management

- Error handling and feedback messages

Full code is provided in Appendix A.

# 5. Frontend Explanation

HTML Pages:

- index.html – Home page with categories, featured products, and search bar.

- login.html – Login form for customers.

- register.html – Registration page with input validation.

- admin\_login.html – Admin login form.

- products.html – Catalog with add-to-cart buttons and filters.

- place\_order.html – Cart page with order confirmation.

CSS Features:

- 3D hover effects on product cards.

- Smooth transitions for buttons.

- Responsive design for desktop, tablet, and mobile.

JavaScript Features:

- Cart quantity adjustment.

- Dynamic form validation.

- Interactive UI effects like modal pop-ups.

Example CSS Snippet:

.product-card:hover {

transform: scale(1.05) rotateY(10deg);

transition: 0.5s;

}

Example JS Snippet:

function updateCart(productId, quantity){

// Update cart dynamically without refreshing page

}

# 6. User Flow

1. Customer visits home page.

2. Registers or logs in.

3. Browses products and adds items to cart.

4. Proceeds to place order and confirms payment.

5. Admin can manage products and view analytics.

Flow Diagram: [Insert detailed stepwise diagram for user and admin flows]

# 7. How to Run Project

1. Install Python 3.9+ and Flask.

2. Clone or download the project folder.

3. Open terminal and navigate to project directory.

4. Install dependencies: pip install flask

5. Run app: python app.py

6. Open browser: http://127.0.0.1:5000/

7. Test all functionalities (login, register, products, cart, order placement).

# 8. Future Improvements

- Integration with online payment gateways.

- Advanced search and filter options.

- User profile with order history.

- Enhanced admin analytics dashboard.

- Mobile-first responsive design improvements.

- Email/SMS notifications for orders.

# 9. Screenshots / Mockups

Homepage: [Insert detailed image]

Login Page: [Insert image]

Register Page: [Insert image]

Products Page: [Insert image]

Place Order Page: [Insert image]

Admin Dashboard: [Insert image]

(Include multiple images with descriptions to expand documentation)

# 10. Conclusion

This project demonstrates a complete e-commerce grocery store application with dynamic frontend, secure backend, database integration, and admin management functionalities. It serves as a foundation for advanced online shopping platforms and can be scaled further with payment integrations and analytics.

# Appendices

Appendix A: Full source code for app.py

Appendix B: HTML, CSS, and JS files with detailed comments

Appendix C: Database schema and sample data

Appendix D: Flowcharts, mockups, and screenshots

Note: Including screenshots, diagrams, code snippets, and expanded explanations will make the document reach 45–50 pages.