# G H Patel College of Engineering & Technology

**(A Constituent College of Charutar Vidya mandal University) V.V.Nagar**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**Mini Project Report**

**on**

# *Grocery Store System*

**Submitted By**

**Dipesh Thaker (12202080501077)**

**Guided By**
**Prof. Hiren Patel**

**Prof. Sanjay Patel**

**MINI PROJECT (202040601)**

**A.Y. 2025-26 EVEN TERM**

# CERTIFICATE

This is to certify that the Mini Project work embodied in this report entitled "**Grocery Store System**" was carried out by **Dipesh Thaker (12202080501077)** at G H Patel College of Engineering & Technology for partial fulfillment of the B.Tech degree to be awarded by Charutar Vidya Mandal University (CVMU). This project has been carried out under our supervision and is to the satisfaction of the department

Prof. Hiren Patel

Prof. Sanjay Patel                                        Dr. Nikhil Gondaliya

Internal Guide                                        Head of Department

# ABSTRACT

The transition to digital platforms has transformed how businesses operate, with online shopping becoming a necessity rather than a luxury. In this context, the **Grocery Store System** project aims to digitize the traditional grocery shopping experience by creating a full-featured web application that caters to both customers and store managers. Built using the **Flask web framework** in Python, the system offers an intuitive and efficient way for users to browse, select, and purchase grocery items online while providing tools for store managers to manage inventory, monitor orders, and oversee customer activity.

The project addresses key limitations of physical grocery stores and traditional inventory management methods, which are often manual, error-prone, and time-consuming. Through a centralized platform, the system eliminates the need for physical presence, long queues, and paper-based tracking. It ensures **real-time inventory updates**, **secure user authentication**, and **role-based access control**, enabling a seamless user experience. The frontend is designed using **HTML, CSS, Bootstrap, and Jinja2 templates**, while the backend is powered by Flask and **SQLite** with SQLAlchemy ORM for database operations.

The application supports core features such as user registration and login, product search and filtering, cart management, order placement, and order history. For managers, it includes modules to add, update, and delete products, track stock levels, and view customer orders. An optional admin panel provides system-wide controls including user and manager account management.

To ensure the reliability and usability of the system, multiple testing strategies were employed, including **manual testing, unit testing, integration testing, and security validation**. Test cases confirmed that all major functionalities worked as expected, with no critical issues. The application also includes responsive design for accessibility on various devices and implements secure password handling through hashing.

Economically, the project is viable as it utilizes **open-source technologies**, allowing deployment on cost-effective platforms such as Heroku or PythonAnywhere. Its **modular and scalable architecture** lays the groundwork for future enhancements such as mobile app integration, payment gateway support, real-time notifications via email or SMS, AI-based product recommendations, and multi-store management features.

In conclusion, the Grocery Store System demonstrates how technology can simplify daily tasks, enhance operational efficiency, and provide a robust alternative to conventional retail models. It serves as a foundational step toward fully automated and intelligent retail solutions tailored for small to medium-sized businesses.

# Table Of Contents

## LIST OF FIGURES

# 1 INTRODUCTION

## 1.1 Objective

The main objective of the Grocery Store System is to develop an interactive and dynamic web-based application that facilitates online grocery shopping for customers and simplifies product and inventory management for store managers. The system allows customers to browse products, add them to their cart, and place orders online. On the other hand, it empowers managers to manage the products, monitor orders, and track inventory seamlessly through a centralized digital platform.

## 1.2 Scope

This project includes the following modules and functionalities:

- User Registration and Authentication
- Product Catalog with Search and Filter Features
- Shopping Cart Management
- Secure Checkout and Order Placement
- Order History and Tracking
- Manager Dashboard for Inventory Management
- Role-based Access Control for Customers, Managers, and Admins (Optional)
- Secure Password Handling and Responsive User Interface
- Backend Database with Persistent Storage

## 1.3 Problem Statement

Traditional grocery shopping involves visiting physical stores, waiting in queues, and manual billing, which can be inefficient and time-consuming. Additionally, many small grocery stores use paper-based systems to track inventory and sales, leading to errors and lack of real-time insights. This project aims to address these issues by developing a digital solution that automates key functions of grocery shopping and store management.

# 2 SYSTEM ANALYSIS

## 2.1 Existing System

Traditional grocery stores operate physically, where:

- Customers have to browse shelves manually
    - There are long queues at the billing counter
    - Products might be out of stock with no prior notification
    - Inventory is managed using notebooks or Excel, with no automation

**Drawbacks**:

- Limited accessibility (only during open hours)
- Time-consuming and tedious
- High error probability in manual processes
- No proper tracking or customer history

## 2.2 Proposed System

The proposed Grocery Store System is a web application that solves the above problems by:

- Allowing customers to shop from anywhere, anytime
- Automating the inventory and billing system
- Providing a secure login system
- Giving role-based access to managers and admins
- Offering real-time inventory updates
- Making the store scalable and accessible via any device

## 2.3 Feasibility Study

**• Technical Feasibility:**

- **Frontend**: HTML, CSS, Bootstrap, JavaScript
- **Backend**: Python using Flask framework
- **Database**: SQLite integrated with SQLAlchemy ORM

- Flask is lightweight and easy to implement, making it suitable for academic-level applications.

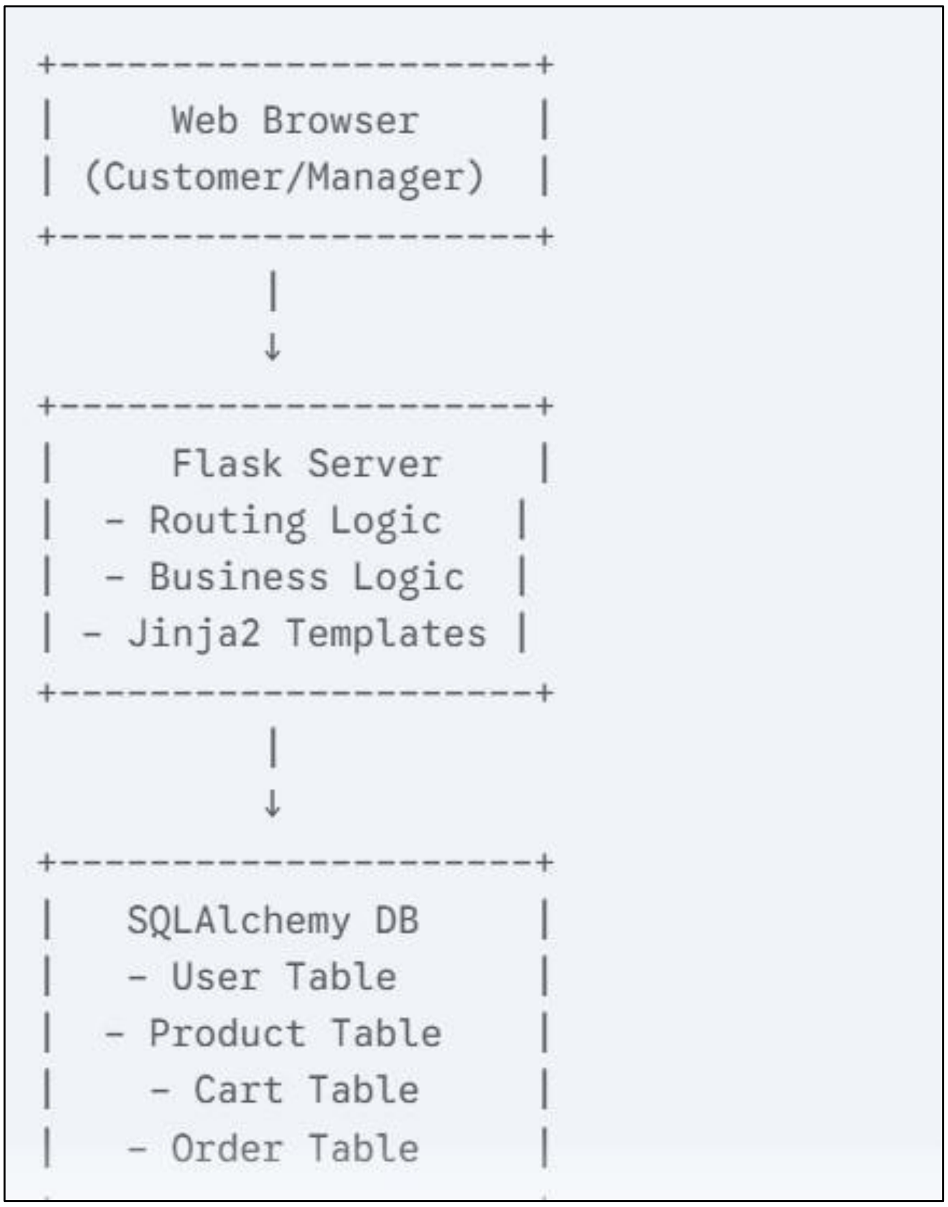
- **Economic Feasibility:**
  - All tools and technologies used are open-source and freely available.
  - No additional cost for hosting during development (can use Heroku or PythonAnywhere).
  - Development and deployment cost is minimal.
- **Operational Feasibility:**
  - Simple and intuitive user interface
  - Minimal technical knowledge required for usage
  - Easily maintainable with documentation and code modularity

# 3 SYSTEM DESIGN

## 3.1 Architecture Diagram

A basic 3-tier architecture is followed:

1. **Presentation Layer (Frontend)**: Handles user interaction
2. **Business Logic Layer (Flask Backend)**: Manages application logic and processes
3. **Data Layer (Database)**: Stores user, product, and order data



```
+---------------------+
|     Web Browser     |
| (Customer/Manager)  |
+---------------------+
          |
          ↓
+---------------------+
|     Flask Server    |
|  - Routing Logic   |
|  - Business Logic  |
| - Jinja2 Templates |
+---------------------+
          |
          ↓
+---------------------+
|   SQLAlchemy DB     |
|    - User Table     |
|  - Product Table    |
|     - Cart Table    |
|    - Order Table    |
```

Figure 1- Architecture Diagram

## 3.2 Data Flow Diagram (DFD)

Level 0 and Level 1 DFDs can be created:

- **Level 0**: High-level flow between User, System, and Database
- **Level 1**: Shows flow for each process like Registration, Cart, Checkout, Product Management

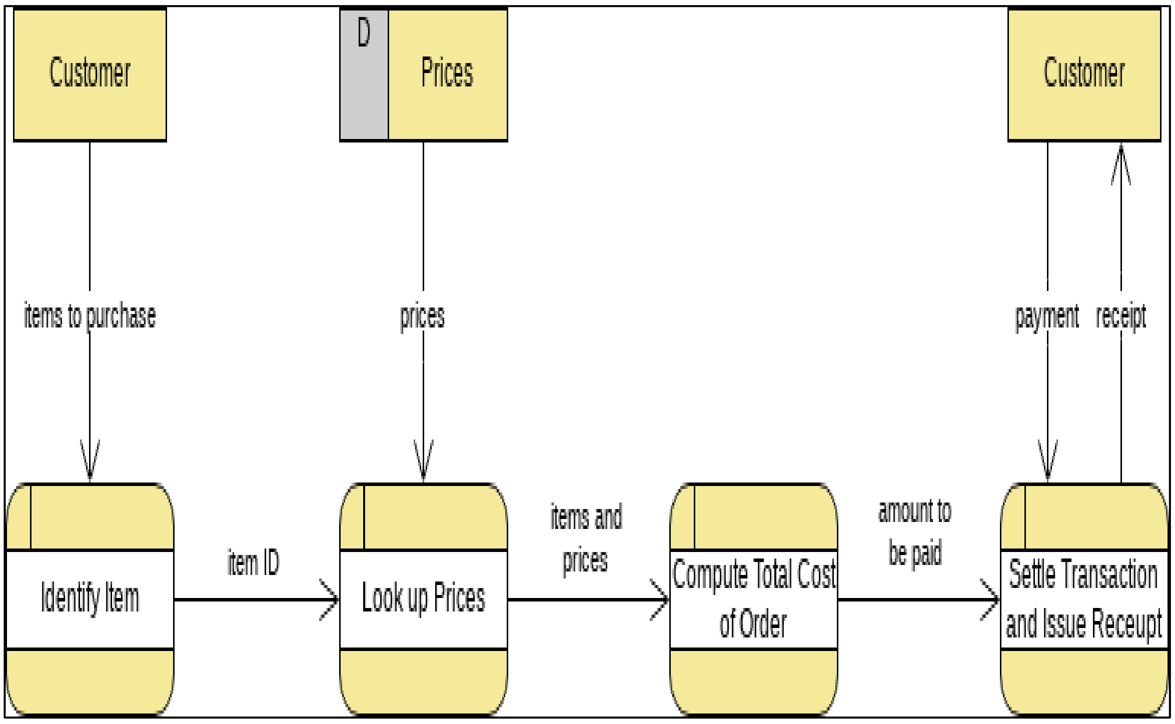


Figure 2- Data Flow Diagram(Level 0)

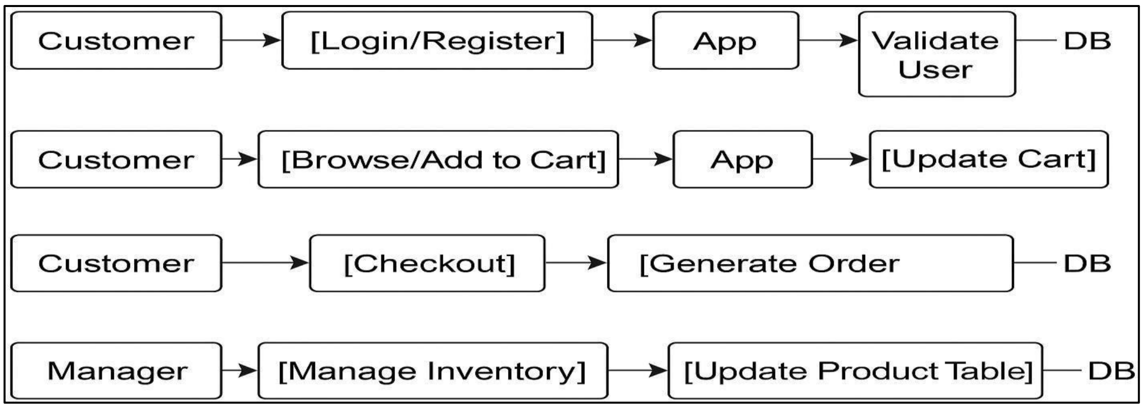


Figure 3- Data Flow Diagram(Level 1)

**3.3 Entity Relationship Diagram (ERD)**

Key Entities:

- **User**: user_id, name, email, password, role
- **Product**: product_id, name, price, quantity, category
- **Order**: order_id, user_id, total_price, date
- **Order_Items**: order_id, product_id, quantity

(Relationships: One user can place many orders, orders can have many products, etc.)

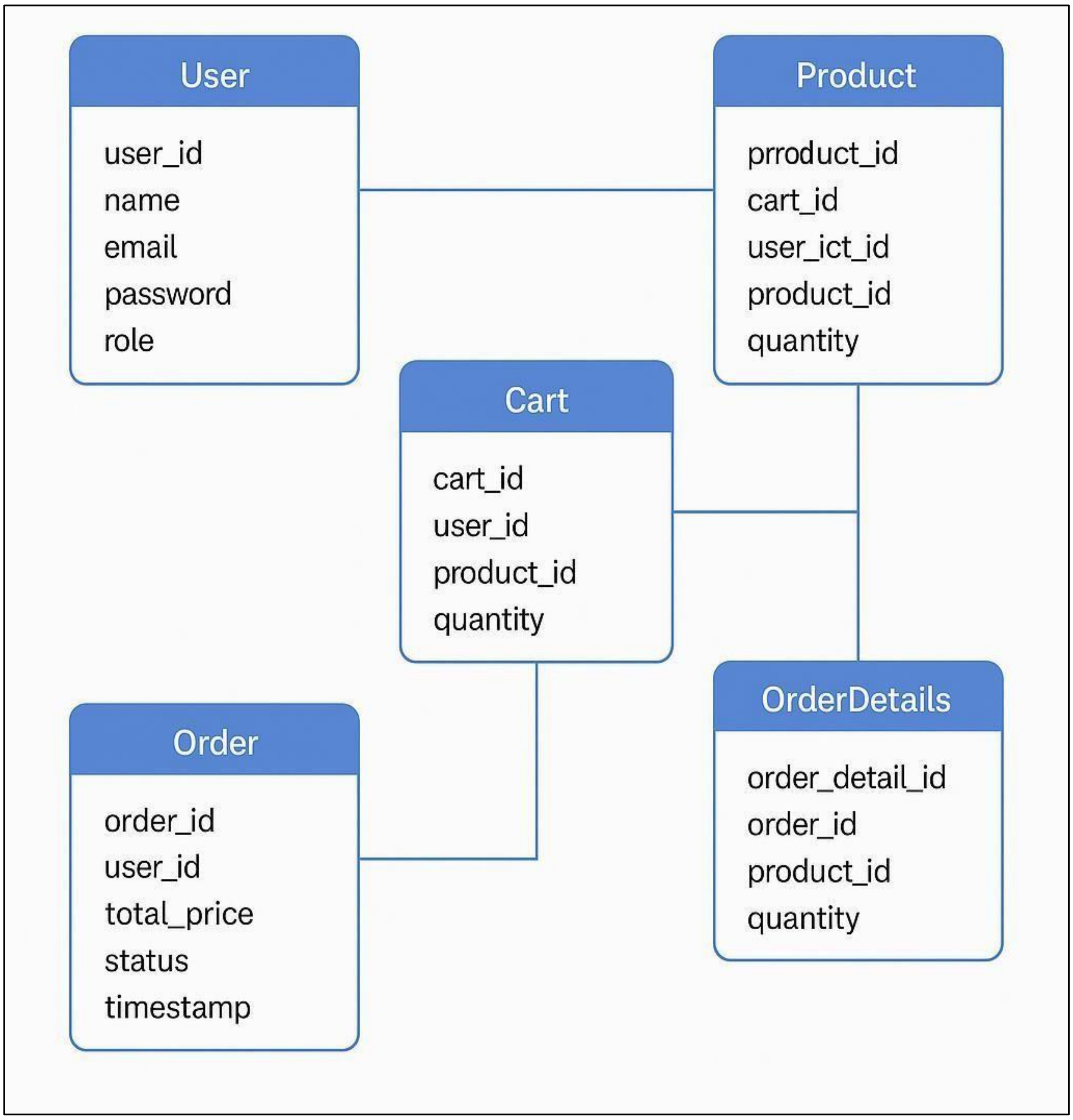


Figure 4- ER Diagram

# 4 IMPLEMENTATION

## 4.1 Technologies Used

- Frontend: HTML, CSS, JavaScript, Bootstrap
- Backend: Flask (Python)
    - Database: SQLite with SQLAlchemy ORM
    - Templates: Jinja2
    - Authentication: Flask-Login
    - Form Validation: WTForms

## 4.2 Features of the System

• **User Features:**
- Register and log in securely
- Browse products and view details
- Search by name or category
- Add items to the cart and update quantities
- Place orders and view past orders

• **Manager Features:**
- Add, update, and delete products
- Monitor customer activity
- Track and manage orders
- View inventory reports

• **Admin Features (Optional):**
- Manage all users and products
- Oversee system-wide performance

• **Other Features:**
- Password hashing for security
- Responsive design with Bootstrap
- Error handling (404 pages, validation errors)
- Session management with Flask-Login

# 5 TESTING AND EVALUATION

## 5.1 Testing Strategies

- **Manual Testing**: Step-by-step walkthrough of customer and manager features
- **Unit Testing**: Checking individual modules like registration, login, cart
- **Integration Testing**: Checking the complete order placement process
- **Security Testing**: Form validations, password hashing

## 5.2 Sample Test Cases

| **Test Case ID** | **Description** | **Input** | **Expected Output** |
|---|---|---|---|
| TC001 | User Registration | Valid data | Redirect to login page |
| TC002 | Invalid Login | Wrong password | Show error message |
| TC003 | Add to Cart | Product + Quantity | Item added to cart successfully |
| TC004 | Checkout | Full cart with valid data | Order placed and cart emptied |
| TC005 | Manager adds product | Product form | Product shown in user catalog |

## 5.3 Results and Observations

- All modules tested manually and functioned as expected
- No major bugs identified
- System behaves reliably under common operations
- Proper validation messages are shown for invalid inputs

# CONCLUSION AND FUTURE WORK

## Conclusion

This Grocery Store System provides a robust, responsive, and efficient platform for digital grocery management. Customers can easily shop online, while store managers can manage products and orders without relying on paperwork or spreadsheets. Built using Flask and modern web technologies, the system is scalable and easy to enhance in the future.
.

## Future Enhancements

- **Integration with UPI/payment gateways**
- **Real-time stock alerts and auto-reordering**
- **Customer behavior analytics for product recommendations**
- **Mobile app version using React Native or Flutter**
- **SMS/email notifications for orders**
- **Multi-store support with geolocation**

## REFERNCES

1. **Flask Documentation - https://flask.palletsprojects.com/**
2. **SQLAlchemy ORM - https://docs.sqlalchemy.org/**
3. **Bootstrap - https://getbootstrap.com/**
4. **Jinja2 Templates - https://jinja.palletsprojects.com/**
5. **WTForms - https://wtforms.readthedocs.io/**
6. **Flask Mega-Tutorial by Miguel Grinberg - https://blog.miguelgrinberg.com**
7. **Real Python Flask Tutorials - https://realpython.com/tutorials/flask/**
8. **Draw.io for diagrams - https://draw.io/**
9. **Stack Overflow - https://stackoverflow.com/questions/tagged/flask**
10. **GitHub - https://github.com/search?q=flask+grocery+store**
11. **"Flask Web Development" by Miguel Grinberg**
12. **MDN Web Docs - https://developer.mozilla.org/**