ONLINE GROCERY SHOPPING SYSTEM

A PROJECT REPORT

Submitted by

SANTHOSH S (2303811724321097)

in partial fulfillment of requirements for the award of the course

CGB1221-DATABASE MANAGEMENT SYSTEMS

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE-2025

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"ONLINE GROCERY SHOPPING SYSTEM "** is the bonafide work of **SA N T H O S H S (2303811724321097)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE

Dr.T. AVUDAIAPPAN, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

ASSOCIATE PROFESSOR

Department of Artificial Intelligence

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

SIGNATURE

Mrs.S. GEETHA, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of Artificial Intelligence

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 04.06.2025

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I declare that the project report on **"ONLINE GROCERY SHOPPING SYSTEM"** is the result of original work done by me and best of my knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1221 – DATABASE MANAGEMENT SYSTEMS.**

Signature

SANTHOSH S

Place: Samayapuram

Date: 04.06.2025

ACKNOWLEDGEMENT

It is with great pride that I express my gratitude and in-debt to my institution **K.Ramakrishnan College of Technology (Autonomous)**, for providing me with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of my study in college.

I would like to express my sincere thanks to my beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to my project and offering adequate duration in completing my project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. AVUDAIAPPAN, M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this project.

I express my deep expression and sincere gratitude to my project supervisor **Mrs.S. GEETHA, M.E.,** Department of **ARTIFICIAL INTELLIGENCE,** for her incalculable suggestions, creativity, assistance and patience which motivated me to carry out this project.

I render my sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express my special thanks to the officials and Lab Technicians of my departments who rendered their help during the period of the work progress.

INSTITUTE

Vision:

- To serve the society by offering top-notch technical education on par with global standards.

Mission:

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all – round personalities respecting moral and ethical values.

DEPARTMENT

Vision:

- To excel in education, innovation, and research in Artificial Intelligence and Data Science to fulfil industrial demands and societal expectations.

Mission

- To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.
- To collaborate with industry and offer top-notch facilities in a conducive learning environment.
- To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.
- To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEO)

- **PEO1:** Compete on a global scale for a professional career in Artificial Intelligence and Data Science.
- **PEO2:** Provide industry-specific solutions for the society with effective communication and ethics.
- **PEO3** Enhance their professional skills through research and lifelong learning initiatives.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Capable of finding the important factors in large datasets, simplify the data, and improve predictive model accuracy.
- **PSO2:** Capable of analyzing and providing a solution to a given real-world problem by designing an effective program.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The Online Grocery Shopping System is a database-driven application designed to streamline the process of purchasing groceries through an interactive digital platform. It allows users to browse products, add items to their cart, place orders, and make payments, while administrators can manage inventory, update product listings, and view customer orders. The system bridges the gap between consumers and local grocery vendors by offering a convenient, accessible, and time-saving shopping experience. With increasing demand for online services, this system enhances user satisfaction by providing real-time updates, secure transactions, and user-role-based functionalities.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD DATABASES FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Online Grocery Shopping System is a database-driven application designed to streamline the process of purchasing groceries through an interactive digital platform. It allows users to browse products, add items to their cart, place orders, and make payments, while administrators can manage inventory, update product listings, and view customer orders. The system bridges the gap between consumers and local grocery vendors by offering a convenient, accessible, and time-saving shopping experience. With increasing demand for online services, this system enhances user satisfaction by providing real-time updates, secure transactions, and user-role-based functionalities.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -2** **PO5 -3** **PO6 -2** **PO7 -1** **PO8 -2** **PO9 -2** **PO10 -2** **PO11-1** **PO12 -2**	**PSO1 -2** **PSO2 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

The primary objective of the Online Grocery Shopping System is to develop a user-friendly platform where customers can buy groceries online and administrators can manage the product catalog and orders efficiently. The system aims to reduce manual workload, minimize human errors in inventory handling, and enhance customer convenience by offering a seamless shopping experience. It also focuses on building a robust backend using SQL for efficient data management and retrieval.

1.2 OVERVIEW

This system comprises two main modules: the Customer Module and the **Admin Module**. The Customer Module allows users to register/login, browse groceries by category, add items to a cart, place orders, and make payments. The Admin Module enables administrators to log in securely, manage product details, monitor stock levels, and view order histories. The application is developed using Python for the front-end interface and SQL for backend database operations, ensuring smooth interaction between users and the data system. The system promotes easy access to groceries and supports contactless transactions in line with modern digital needs.

1.3 SQL AND DATABASE CONCEPTS

The Online Grocery Shopping System uses a normalized relational database to manage core entities such as customers, products, orders, payments, and administrators. Relationships between tables are enforced using foreign keys to maintain data integrity. SQLAlchemy ORM is used to handle CRUD (Create, Read, Update, Delete) operations efficiently,

allowing for cleaner code and improved security by avoiding direct SQL queries. The database schema is designed for scalability and optimized access.

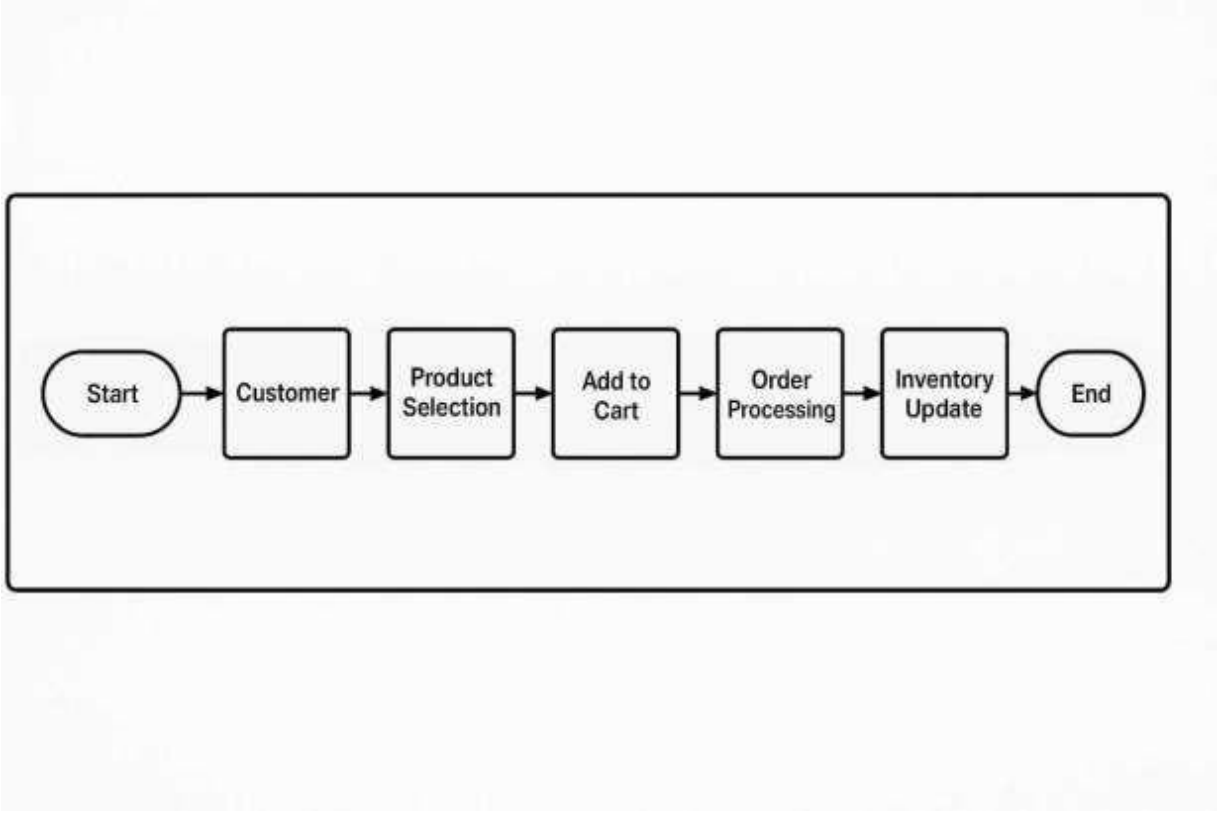
- Normalized tables for efficient and redundant-free data storage.
- Foreign key constraints to maintain relationships between entities like orders and customers.
- Passwords stored securely using hashing algorithms.
- CRUD operations supported via SQLAlchemy ORM.
- Many-to-many relationships implemented using junction tables (e.g., orders and products).

CHAPTER 2
PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed work involves designing and developing a Python-based application integrated with an SQL database to simulate a fully functional online grocery store. The application will include separate login systems for customers and administrators, with each having role-specific interfaces. Customers will be able to search and purchase groceries online, while administrators will handle inventory and order management. The system will ensure real-time updates, a secure login system, and smooth user interactions. Focus will be given to intuitive UI design, optimized database queries, and scalable architecture to accommodate future enhancements.

2.2 BLOCK DIAGRAM



CHAPTER 3
MODULE DESCRIPTION

3.1 User Module

Allows users to register, log in, update profiles, and manage their account settings. Supports role-based access control to differentiate between normal users, teachers, and admins.

Key Methods:

- register_user(user_data) — Creates a new user account.
- login_user(credentials) — Authenticates a user and initiates a session.
- update_profile(user_id, profile_data) — Updates user information.
- delete_user(user_id) — Removes or deactivates a user account.
- get_user_role(user_id) — Retrieves the role and permissions of a user.

3.2 Admin Module

Allows admins to manage system-wide settings, user roles, and monitor platform activities. Provides tools for user moderation and content oversight.

Key Methods:

- add_user(user_data) — Adds a new user with specific roles.
- remove_user(user_id) — Deletes or suspends a user account.
- assign_role(user_id, role) — Updates user permissions.
- view_activity_logs(filter_criteria) — Retrieves logs for auditing.
- manage_system_settings(settings_data) — Updates global system configurations.

3.3 Product Management Module

Enables adding, updating, organizing, and removing products in the inventory with detailed metadata like name, price, description, and stock status. Supports categorization and product availability control.

Key Methods:

- add_product(product_data) — Inserts a new product into the system.
- update_product(product_id, updated_data) — Modifies existing product details.
- delete_product(product_id) — Removes or archives a product.
- list_products(filter_criteria) — Retrieves a list of products based on filters.
- manage_stock(product_id, quantity) — Updates inventory stock levels.

3.4 Search and Filter Module

Provides functionality to search products or content and filter results based on various criteria like category, price range, rating, or availability. Supports sorting and pagination.

Key Methods:

- search_items(query) — Searches products or content matching the query.
- filter_items(filters) — Applies multiple filter criteria to narrow results.
- sort_items(criteria) — Sorts results by price, rating, or popularity.
- paginate_results(page, size) — Handles result pagination.
- autocomplete_suggestions(prefix) — Provides live search suggestions.

3.5 Billing and Payment Module

Manages all billing operations including invoice generation, payment processing, discount

application, and transaction verification. Ensures secure and reliable financial workflows.
Key Methods:

- generate_invoice(order_id) — Creates billing details for an order.
- process_payment(payment_info) — Handles payment authorization and capture.
- apply_discount(order_id, coupon_code) — Applies discount coupons to orders.
- verify_transaction(transaction_id) — Confirms payment success.
- refund_payment(transaction_id) — Processes payment refunds when needed.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

The Online Grocery Shopping System provides a convenient way for customers to buy groceries from home. I t simplifies product search, inventory management, and order processing. The system ensures fast billing and secure payments. Overall, it improves the shopping experience for both customers and store owners.

FUTURE SCOPE

- Implement AI-driven personalized recommendations to improve user engagement.
- Integrate multiple secure payment gateways to offer flexible payment options.
- Enhance search functionality with natural language processing for smarter queries.
- Develop mobile applications to increase accessibility and user convenience.
- Add detailed analytics dashboards to provide real-time insights for admins.

APPENDIX A – SOURCE CODE

```
import sqlite3
from flask import Flask, render_template, request, jsonify, redirect, url_for, session
from werkzeug.security import generate_password_hash, check_password_hash
import json

app = Flask(__name__)
app.secret_key = 'your-secret-key'  # Replace with a secure key in production

# Database setup
def init_db():
    conn = sqlite3.connect('grocery.db')
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS products
                 (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, price REAL,
stock INTEGER)''')
    c.execute('''CREATE TABLE IF NOT EXISTS users
                 (id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT
UNIQUE, password TEXT, is_admin INTEGER)''')
    c.execute('''CREATE TABLE IF NOT EXISTS orders
                 (id INTEGER PRIMARY KEY AUTOINCREMENT, user_id INTEGER,
products TEXT, total REAL, status TEXT)''')

    # Insert sample admin user if not exists
    c.execute("SELECT * FROM users WHERE username = 'admin'")
```

```
    if not c.fetchone():
        c.execute("INSERT INTO users (username, password, is_admin) VALUES (?, ?, ?)",
                  ('admin', generate_password_hash('admin123'), 1))

    # Insert sample products if table is empty
    c.execute("SELECT COUNT(*) FROM products")
    if c.fetchone()[0] == 0:
        sample_products =
            [ ('Apple', 0.5,
            100),
            ('Banana', 0.3, 150),
            ('Milk', 3.0, 50),
            ('Bread', 2.0, 30)
        ]
        c.executemany("INSERT INTO products (name, price, stock) VALUES (?, ?, ?)",
sample_products)

    conn.commit()
    conn.close()

# Initialize database
init_db()

# Routes
@app.route('/')
def index():
        return redirect(url_for('login'))
```

```
if 'user_id' not in session:

    return redirect(url_for('login'))
```

```
    if session.get('is_admin'):
        return redirect(url_for('admin'))
    conn = sqlite3.connect('grocery.db')
    c = conn.cursor()
    c.execute("SELECT * FROM products")
    products = c.fetchall()
    conn.close()
    return render_template('index.html', products=products)

@app.route('/search', methods=['GET'])
def search():
    if 'user_id' not in session:
        return redirect(url_for('login'))
    if session.get('is_admin'):
        return redirect(url_for('admin'))
    query = request.args.get('query', '')
    conn = sqlite3.connect('grocery.db')
    c = conn.cursor()
    c.execute("SELECT * FROM products WHERE name LIKE ?", ('%' + query + '%',))
    products = c.fetchall()
    conn.close()
    return jsonify(products)

@app.route('/cart', methods=['GET', 'POST'])
def cart():
    if 'user_id' not in session:
        return redirect(url_for('login'))
```

```
if session.get('is_admin'):
    return redirect(url_for('admin'))

if 'cart' not in session:
    session['cart'] = {}

if request.method == 'POST':
    product_id = request.form['product_id']
    quantity = int(request.form['quantity'])
    if quantity <= 0:
        session['cart'].pop(product_id, None)
    else:
        session['cart'][product_id] = quantity
    session.modified = True
    return jsonify({'status': 'success'})

conn = sqlite3.connect('grocery.db')
c = conn.cursor()
cart_items = []
total = 0
for product_id, quantity in session['cart'].items():
    c.execute("SELECT * FROM products WHERE id = ?", (product_id,))
    product = c.fetchone()
    if product:
        cart_items.append({'product': product, 'quantity': quantity})
        total += product[2] * quantity
conn.close()
```

```
    return render_template('cart.html', cart_items=cart_items, total=total)

@app.route('/checkout', methods=['GET', 'POST'])
def checkout():
    if 'user_id' not in session:
        return redirect(url_for('login'))
    if session.get('is_admin'):
        return redirect(url_for('admin'))
    if 'cart' not in session or not session['cart']:
        return render_template('cart.html', error='Cart is empty')

    if request.method == 'POST':
        # Simulate payment processing
        card_number = request.form['card_number']
        expiry = request.form['expiry']
        cvv = request.form['cvv']

        # Basic validation (for simulation)
        if len(card_number) < 16 or not expiry or len(cvv) < 3:
            return render_template('payment.html', error='Invalid payment details')

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

        # Save order
        user_id = session['user_id']
        products = json.dumps(session['cart'])
```

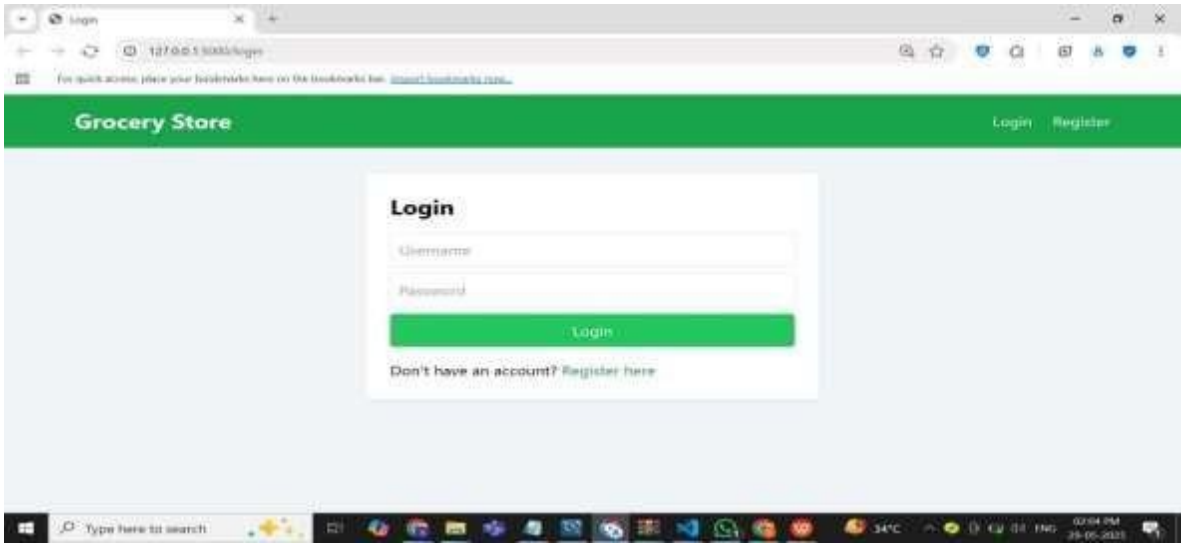
```
        total = sum(
            c.execute("SELECT price FROM products WHERE id = ?", (pid,)).fetchone()[0] *
qty
            for pid, qty in session['cart'].items()
        )
        c.execute("INSERT INTO orders (user_id, products, total, status) VALUES (?, ?, ?, ?)",
                  (user_id, products, total, 'Completed'))

        # Update stock
        for product_id, quantity in session['cart'].items():
            c.execute("UPDATE products SET stock = stock - ? WHERE id = ?", (quantity,
product_id))

        conn.commit()
        conn.close()

        session['cart'] = {}  # Clear cart
        return render_template('payment_success.html', total=total)

    return render_template('payment.html')

@app.route('/admin', methods=['GET', 'POST'])
def admin():
    if 'user_id' not in session or not session.get('is_admin'):
        return redirect(url_for('login'))

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

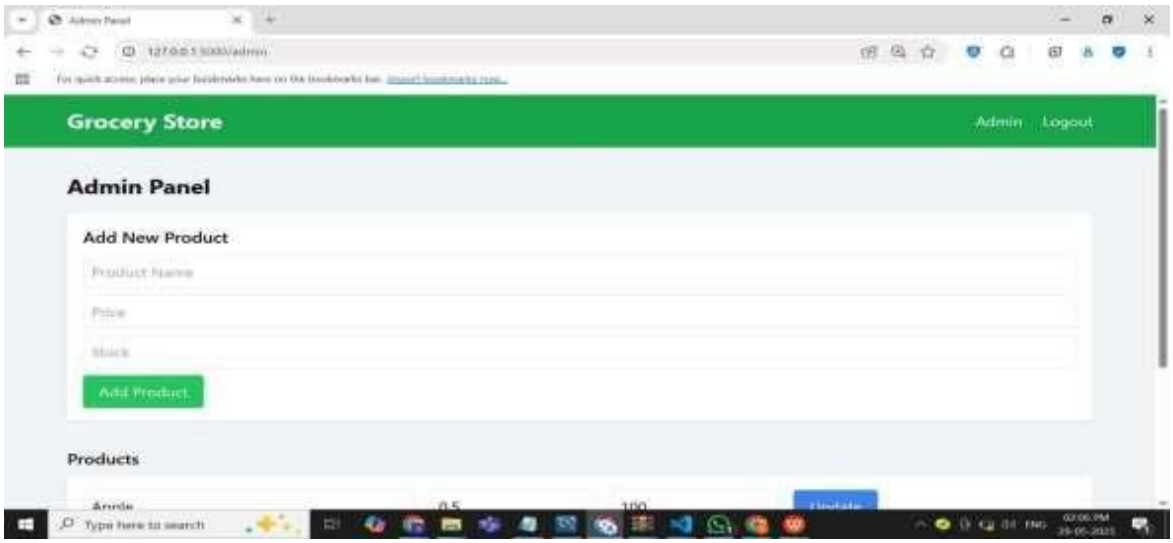
```
    c = conn.cursor()

    if request.method == 'POST':
        if 'add_product' in request.form:
            name = request.form['name']
            price = float(request.form['price'])
            stock = int(request.form['stock'])
            c.execute("INSERT INTO products (name, price, stock) VALUES (?, ?, ?)", (name,
price, stock))
        elif 'edit_product' in request.form:
            product_id = request.form['product_id']
            name = request.form['name']
            price = float(request.form['price'])
            stock = int(request.form['stock'])
            c.execute("UPDATE products SET name = ?, price = ?, stock = ? WHERE id = ?",
                    (name, price, stock, product_id))
        conn.commit()

    c.execute("SELECT * FROM products")
    products = c.fetchall()
    conn.close()
    return render_template('admin.html', products=products)

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
```

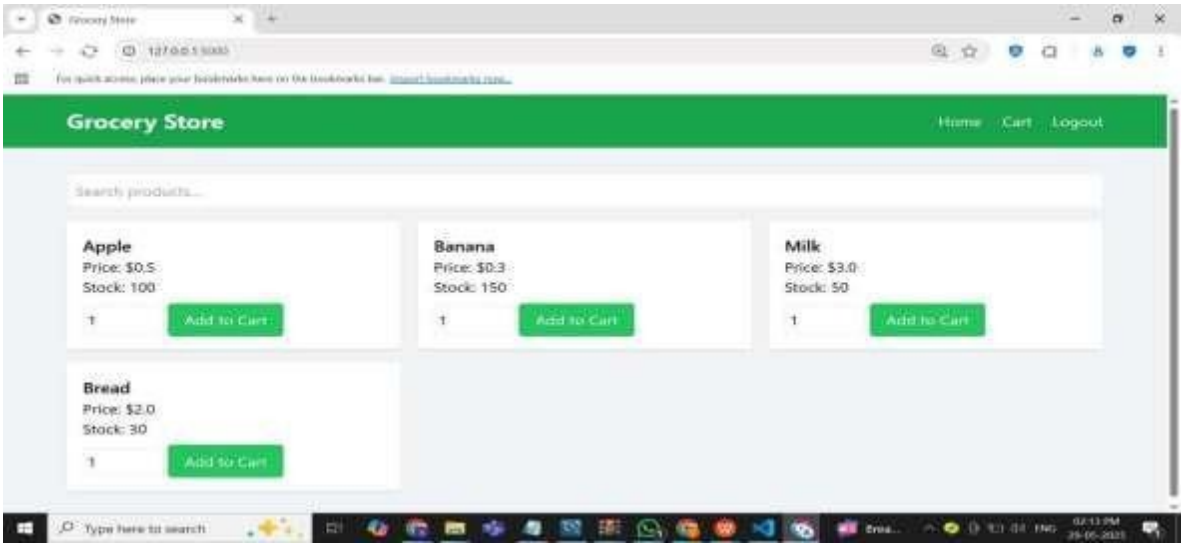
```
        password = request.form['password']
        conn  =  sqlite3.connect('grocery.db')
        c = conn.cursor()

        # Check if username exists
        c.execute("SELECT * FROM users WHERE username = ?", (username,))
        if c.fetchone():
            conn.close()
            return render_template('register.html', error='Username already exists')

        # Create new user
        c.execute("INSERT INTO users (username, password, is_admin) VALUES (?, ?, ?)",
                  (username, generate_password_hash(password), 0))
        conn.commit()
        conn.close()
        return redirect(url_for('login'))

    return render_template('register.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        conn = sqlite3.connect('grocery.db')
        c = conn.cursor()
        c.execute("SELECT * FROM users WHERE username = ?", (username,))
```

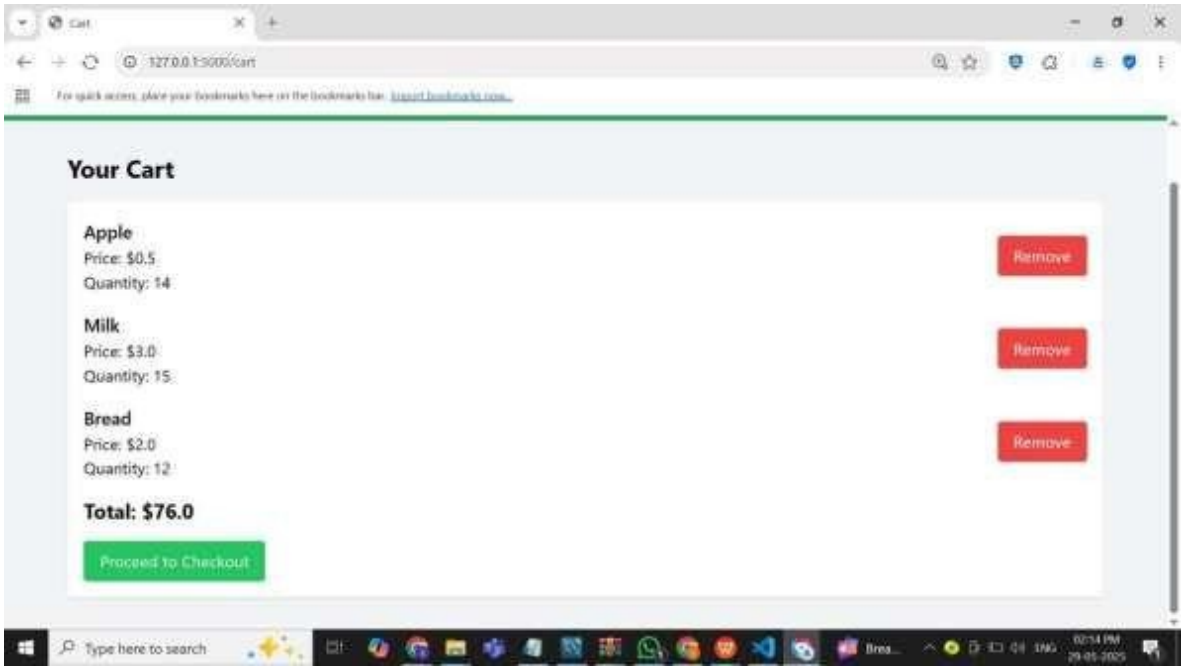
```
        user = c.fetchone()
        conn.close()

        if user and check_password_hash(user[2], password):
            session['user_id'] = user[0]
            session['is_admin'] = user[3]
            return redirect(url_for('admin') if user[3] else url_for('index'))
        return render_template('login.html', error='Invalid credentials')

    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))

if __name__ == '__main__':
    app.run(debug=True)
```

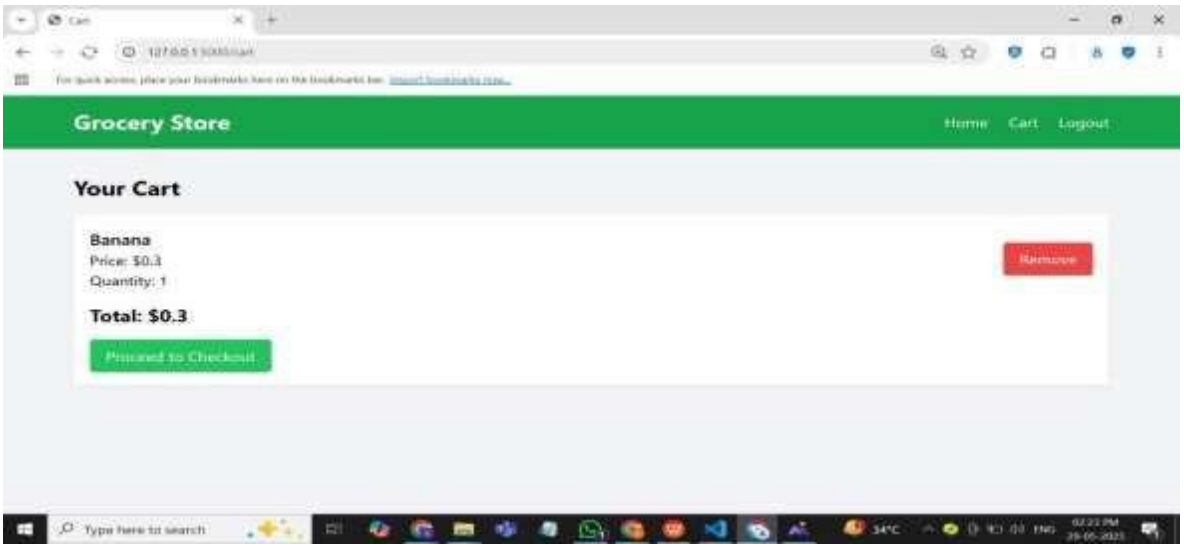
"APPENDIX B – SCREENSHOTS

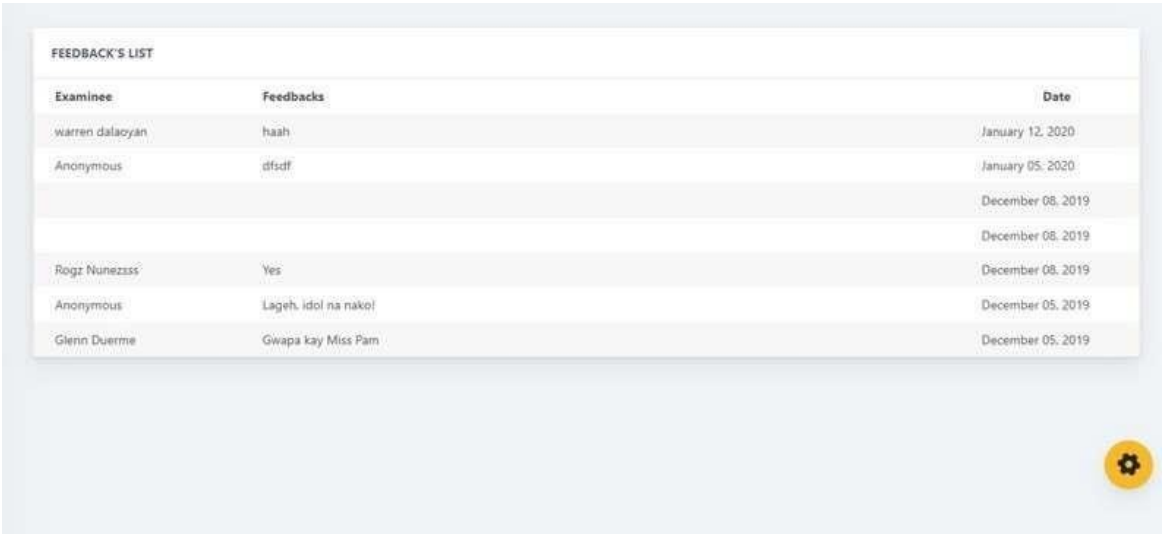





Grocery Store
Home
Cart
Logout
Search products...
Apple
Price: $0.5
Stock: 100
1
Add to Cart
Banana
Price: $0.3
Stock: 150
1
Add to Cart
Milk
Price: $3.0
Stock: 50
1
Add to Cart
Bread
Price: $2.0
Stock: 30
1
Add to Cart


Your Cart
Apple
Price: $0.5
Quantity: 14
Remove
Milk
Price: $3.0
Quantity: 15
Remove
Bread
Price: $2.0
Quantity: 12
Remove
Total: $76.0
Proceed to Checkout


Grocery Store
Home
Cart
Logout
Your Cart
Banana
Price: $0.3
Quantity: 1
Remove
Total: $0.3
Proceed to Checkout
Type here to search


FEEDBACK'S LIST
Examinee
Feedbacks
Date
warren dalaoyan
haah
January 12, 2020
Anonymous
dfsdf
January 05, 2020
December 08, 2019
December 08, 2019
Rogz Nunezsss
Yes
December 08, 2019
Anonymous
Lageh, idol na nako!
December 05, 2019
Glenn Duerme
Gwapa kay Miss Pam
December 05, 2019

REFERENCES

1. Gradio Documentation: An open-source Python library for creating customizable user interfaces.
   URL: https://www.gradio.app
2. SQLite Documentation: A lightweight, serverless SQL database engine used in local applications.
   URL: https://www.sqlite.org/docs.html
3. Pandas Documentation: Used for data manipulation and retrieval from the database.
   URL: https://pandas.pydata.org/docs/
4. Matplotlib Documentation: A visualization library for Python used to generate charts and graphs.
   URL: https://matplotlib.org/stable/contents.html
5. Python Official Documentation: Comprehensive reference for Python language features and standard libraries.
   URL: https://docs.python.org/3/