**GROCERY STORE MANAGEMENT SYSTEM**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfilment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**DHANUSH GAMIDI**

**AP23110010867**

**BHAVA SRI PULIKHANDAM**

**AP23110010929**

**VAISHNAVI KASTURI**

**AP23110010888**

**A picture containing text

Description automatically generated**

Under the Guidance of

**MS.DR. BANEE BANDANA DAS MAM**

**ASSISTANT PROFESSOR, DEPARTMENT OF CSE**

**SRM University–AP**

**Neeru Konda, Mangala Giri, Guntur**

**Andhra Pradesh – 522 240**

**[Month, Year]**

**Certificate**

Date: \_\_\_\_\_\_\_\_\_

This is to certify that the work present in this Project entitled “**GROCERY STORE MANAGEMENT SYSTEM**” has been carried out by **DHANUSH GAMIDI, BHAVYA SRI P, VASIHNAVI K** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

**MS.DR. BANEE BANDANA DAS MAM**

Designation,

Affiliation.

**Acknowledgement**

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

I am extremely grateful and express my profound gratitude and indebtedness to my project guide, **MS.DR. BANEE BANDANA DAS MAM**

Department of Computer Science & Engineering, SRM University, Andhra Pradesh, for her kind help and for giving me the necessary guidance and valuable suggestions in completing this project work.

**Table of Contents**

1. Introduction and Objective of the Project .............................1
2. Dataset – Data Collection ......................................................2
3. Methodology ..........................................................................3
   * I. ER Diagram ............................................................4
   * II. Normalization ........................................................5
   * III. Concepts Used ......................................................6
4. Results .....................................................................................7
5. Future Work ............................................................................8
6. References ...............................................................................9

## 

## 

## **Section 1: Introduction and Objective of the Project**

**Introduction**

E-commerce has revolutionized how businesses operate, offering customers convenience and efficiency in purchasing goods online. Managing data efficiently in such platforms is crucial for tracking orders, inventory, and customer details. A well-structured database ensures smooth business operations, enhances performance, and improves user experience.

### **Objective of the Project**

The main objective of this project is to design a relational database system tailored for an e-commerce platform to manage customer orders and product catalogs effectively. The system ensures:

* **Data Consistency:** Preventing duplication and maintaining integrity in customer and order records.
* **Scalability:** Designing a structure that can support future expansion with features like recommendations and analytics.
* **Efficient Query Processing:** Utilizing indexing, views, and joins for optimized data retrieval.

This database model minimizes redundancy and improves efficiency, making it adaptable for large-scale operations. It serves as the foundation for an advanced e-commerce backend system, allowing future integration of payment processing, inventory management, and customer insights.

With the growing adoption of digital transactions and online grocery shopping, a scalable and high-performance database becomes indispensable for managing real-time order processing, seamless checkout experiences, and maintaining customer loyalty programs. This project aims to create an intuitive and well-optimized relational database model that ensures smooth data handling, security, and reliability for a **comprehensive grocery store management system**.

## **Section 2: Dataset – Data Collection**

### **Data Collection Method**

The dataset for this project was designed based on the fundamental components of an e-commerce platform, ensuring it mirrors real-world scenarios. Research sources included:

1. **Online Articles & Tutorials:** Reviewing existing guides on database design and best practices.
2. **Academic Textbooks:** Learning relational database principles from renowned sources such as Database System Concepts by Silberschatz, Korth, and Sudarshan.
3. **Sample Data Sets:** Utilizing GitHub repositories with open-source schemas for inspiration.

### **Data Structure & Attributes**

Our dataset consists of five primary entities:

1. **Customers:** Contains details such as customer ID, name, email, and address.
2. **Orders:** Maintains information like order ID, customer ID, date, and total amount.
3. **Order Items:** Stores individual items within an order, linking products and quantities.
4. **Products:** Includes product ID, name, price, and available stock.
5. **Categories:** Defines classification for products (e.g., Electronics, Clothing).

Through this approach, the dataset captures essential attributes while ensuring accurate representation for a functional e-commerce platform.

## **Section 3: Methodology**

### **Overview**

The methodology of this project involves designing a structured relational database using industry-standard techniques. The key focus is on **entity-relationship modeling, normalization,** and ensuring **data integrity** through constraints and optimized queries. This approach provides a seamless foundation for managing customer orders, product catalogs, and order transactions efficiently.

### **I. ER Diagram**

An **Entity-Relationship Diagram (ERD)** visually represents the logical structure of our database. ERDs help define relationships between entities, ensuring a clear and scalable design. The following components are included:

**Entities**

* **CUSTOMERS:** Stores customer details such as name, contact information, and address.
* **ORDERS:** Represents purchase transactions with an associated customer and timestamp.
* **ORDER\_ITEMS:** Tracks individual products within an order, mapping to the corresponding product.
* **PRODUCTS:** Maintains product details including stock availability, price, and category.
* **CATEGORIES:** Groups products into broader classifications.

 Relationships

* A customer places multiple orders.
* Each order consists of multiple order items.
* An order item corresponds to one product.
* Each product belongs to a specific category.

 ER **Diagram Representation**

A diagram of a company

AI-generated content may be incorrect.

### **Entities and Their Attributes**

**1.CUSTOMERS** (Stores customer information)

* + **customer\_id** (PK) – Unique identifier for each customer
  + **name** – Customer’s full name
  + **contact\_number** – Phone number
  + **email** – Email address
  + **address** – Residential address

**2.ORDERS** (Tracks purchases made by customers)

* + **order\_id** (PK) – Unique order identifier
  + **customer\_id** (FK) – Links the order to a specific customer
  + **order\_date** – Date and time of purchase
  + **total\_amount** – Final bill amount

**3. ORDER\_ITEMS (*Stores details of individual products in an order*)**

* **order\_item\_id (PK) – Unique identifier for each order item**
* **order\_id (FK) – Links the order item to a specific order**
* **product\_id (FK) – References the purchased product**
* **quantity – Number of units purchased**
* **price – Cost per unit**

**4.PRODUCTS (*Stores grocery items available for purchase*)**

* **product\_id (PK) – Unique identifier for each product**
* **name – Product name**
* **category\_id (FK) – Defines product category (e.g., Vegetables, Dairy, Snacks)**
* **price – Cost per unit**
* **stock\_quantity – Available units in inventory**

**5.CATEGORIES (*Defines product classifications*)**

* **category\_id (PK) – Unique identifier for each category**
* **category\_name – Name of the category (e.g., Fruits, Beverages)**

**6.SUPPLIERS (*Tracks suppliers for inventory replenishment*)**

* **supplier\_id (PK) – Unique identifier for each supplier**
* **name – Supplier’s name**
* **contact\_number – Phone number**
* **email – Email address**
* **address – Supplier’s business location**

**Relationships in ER Diagram**

* **A Customer places multiple Orders (One-to-Many Relationship)**
* **An Order contains multiple Order Items (One-to-Many Relationship)**
* **Each Order Item references a Product (Many-to-One Relationship)**
* **Each Product belongs to one Category (Many-to-One Relationship)**
* **A Supplier provides multiple Products (One-to-Many Relationship)**

### **Constraints & Keys**

* **Primary Keys (PK)** ensure unique identification of each record.
* **Foreign Keys (FK)** establish relationships between different tables.
* **Referential Integrity** is enforced to maintain data consistency.

**TABLE:PRODUCTS**

A screenshot of a computer

AI-generated content may be incorrect.

**TABLE:CUSUTOMER**

**A screenshot of a computer

AI-generated content may be incorrect.**

**TABLE:ORDERSA screenshot of a computer

AI-generated content may be incorrect.**

**TABLE:ORDER\_ITEM**

A screenshot of a computer

AI-generated content may be incorrect.

ENTITY RELATIONAL CARDINALITY

| **Relationship** | **From Entity** | **To Entity** | **Cardinality** | **Description** |
| --- | --- | --- | --- | --- |
| Places | customers | orders | 1:N | A customer can place multiple orders, but each order belongs to one customer. |
| Contains | orders | Order\_items | 1:N | An order can contain multiple items, each item is linked to one order. |
| Includes Product | order\_items | products | M:1 | Many order items can reference the same product. |
| Belongs To | products | categories (as category field) | M:1 | Many products can belong to one category (as a string or FK in a normalized form). |

**Primary & Foreign Keys**

* **Primary Keys (PK):** Unique identifiers, ensuring each record is distinct (e.g., customer\_id, order\_id).
* **Foreign Keys (FK):** Establish dependencies between tables (e.g., customer\_id in ORDERS references CUSTOMERS).

### **II. Normalization**

To maintain data consistency and remove redundancy, the database schema is **normalized to 3rd Normal Form (3NF)**. This ensures optimized storage and efficient query execution.

1. **First Normal Form (1NF)** – Eliminates duplicate columns and ensures atomic data fields.
2. **Second Normal Form (2NF)** – Removes partial dependencies, ensuring attributes depend entirely on primary keys.
3. **Third Normal Form (3NF)** – Eliminates transitive dependencies, preventing redundant data duplication.

Normalization enhances the overall **efficiency, scalability, and consistency** of the database.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a form

AI-generated content may be incorrect.

* Here, **customer\_name** depends on order\_id, but product\_name depends on product\_id.
* This creates redundancy because product\_name does not need to exist in the Orders table.

A screenshot of a computer

AI-generated content may be incorrect.

This ensures **attributes depend entirely on their respective primary keys**, reducing redundant storage.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a store location

AI-generated content may be incorrect.

**Benefits of Normalization for Grocery Store Management**

* **Optimized Storage:** Reduces duplicate information and saves memory space.
* **Faster Queries:** Efficient **JOIN operations** for retrieving customer orders and inventory status.
* **Data Integrity:** Prevents errors due to inconsistent updates in dependent tables.
* **Scalability:** Future system enhancements can be implemented without disrupting existing data.

### **III. Concepts Used**

* **Views:** Created to fetch consolidated reports, e.g., customer order summary.
* **Triggers:** Implemented to auto-update stock when a new order is placed.
* **Joins:** Used INNER and LEFT JOINs for retrieving complex relational data efficiently.
* **Foreign Key Constraints:** Ensures referential integrity across related tables.

## **Section 4: Results**

### **Overview**

The successful implementation of the e-commerce order management database ensures **efficient data organization, integrity, and optimization** for handling customer orders, products, and categories. To validate the effectiveness of the system, we tested various database queries, relationships, and constraints. The results demonstrate **smooth execution, minimal redundancy, and accurate data retrieval** through SQL queries.

### **Database Implementation & Testing**

The database was tested by executing SQL queries, ensuring correctness and efficiency in operations such as data retrieval, insertion, updates, and deletions. Below are key outcomes:

**1. Creating Tables**

Each entity was successfully structured with appropriate primary keys, foreign key constraints, and attributes. The following SQL commands were executed:

* CREATE TABLE CUSTOMERS (...) – Defines customer details.
* CREATE TABLE ORDERS (...) – Stores purchase details.
* CREATE TABLE ORDER\_ITEMS (...) – Tracks order items.
* CREATE TABLE PRODUCTS (...) – Lists products and categories.

#### **2. Inserting Sample Data**

To validate the database schema, sample records were inserted into tables using SQL INSERT statements. This allowed testing of the relationships between orders, customers, and products.

#### **3. Query to List Orders with Customer Names & Total Amount**

SQL queries using JOIN operations successfully retrieved consolidated reports, ensuring orders are correctly linked to customers. Example:

SELECT CUSTOMERS.name, ORDERS.order\_id, SUM(ORDER\_ITEMS.quantity \* PRODUCTS.price) AS TotalAmount

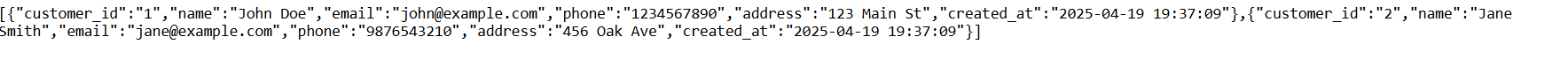
FROM ORDERS

JOIN CUSTOMERS ON ORDERS.customer\_id = CUSTOMERS.customer\_id

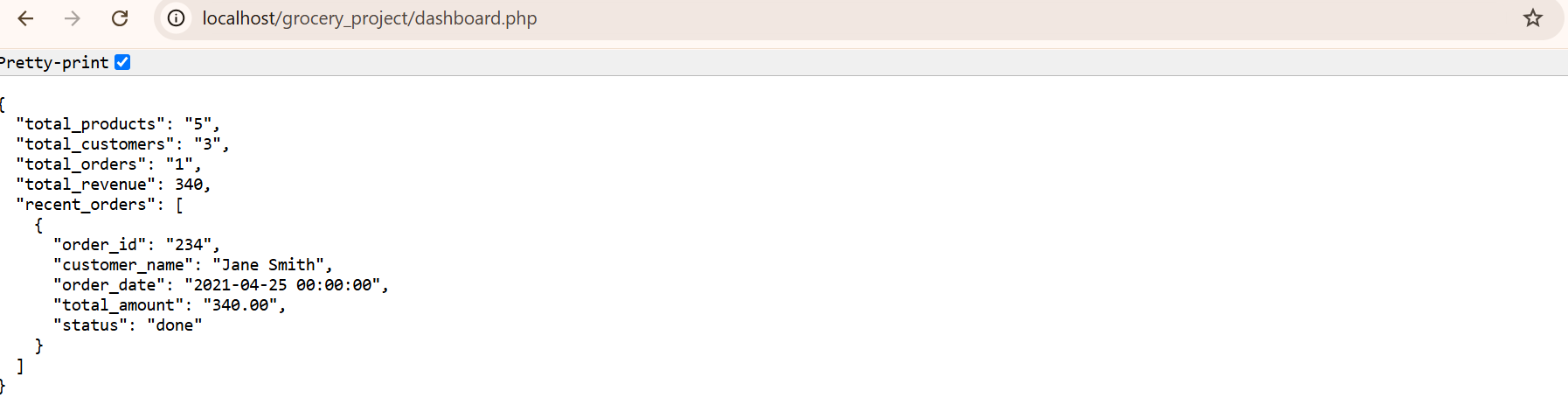
JOIN ORDER\_ITEMS ON ORDERS.order\_id = ORDER\_ITEMS.order\_id

JOIN PRODUCTS ON ORDER\_ITEMS.product\_id = PRODUCTS.product\_id

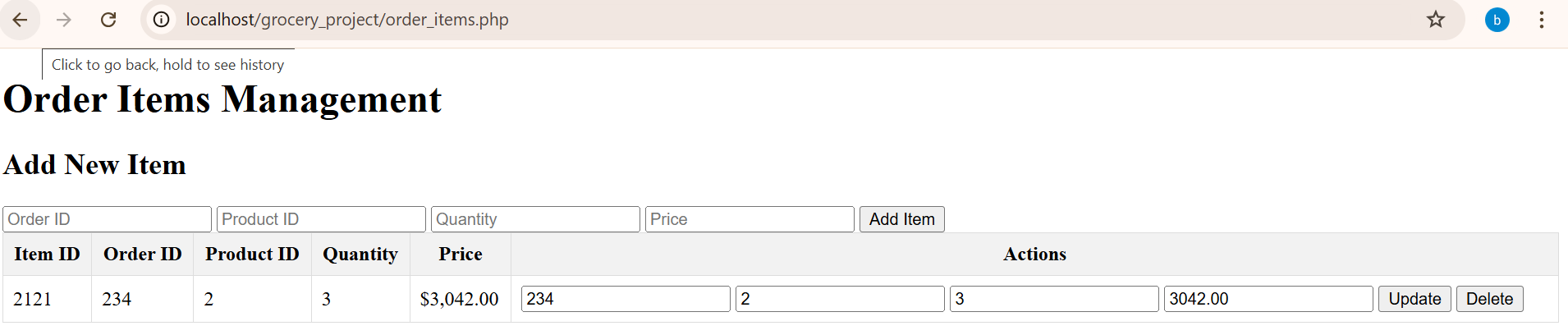
GROUP BY ORDERS.order\_id;



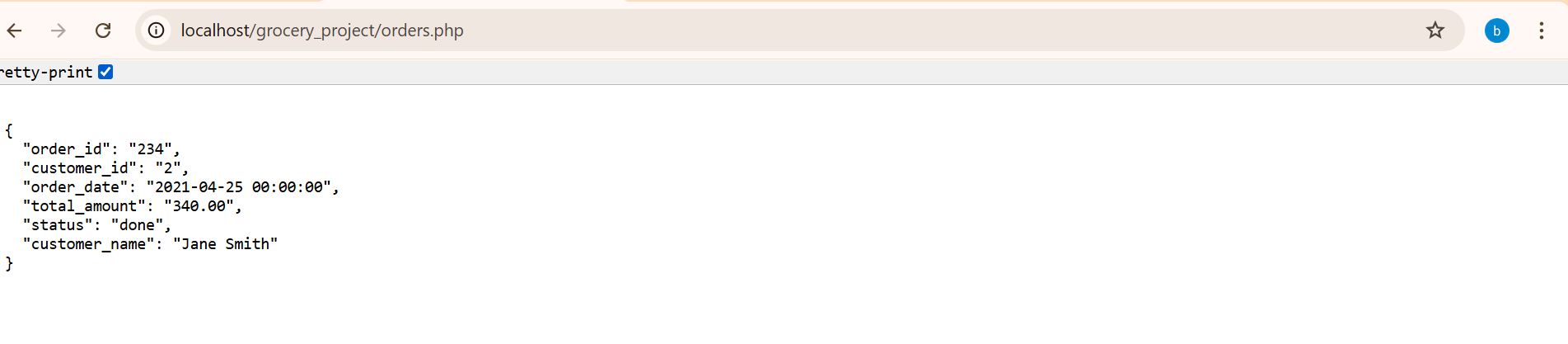
**CUSTOMERS**

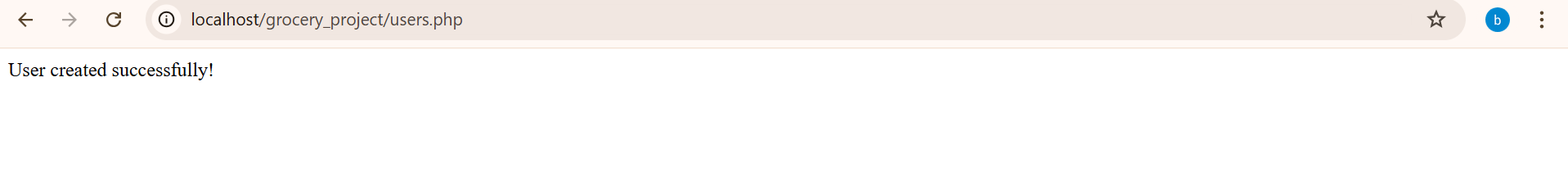


**dashboard.php**

****

**Order\_items.php**

**orders.php**

****

**Users.php**

## **Section 5: Future Work**

### **Enhancements & Expansion**

This project serves as the **foundational framework** for an e-commerce order management system. Future improvements focus on refining features, enhancing user experience, and integrating automation mechanisms for a complete backend system.

**Admin Panel Interface:**

* A graphical user interface (GUI) will allow administrators to **manage orders, customers, and product inventory** with ease.
* Features may include **order approval workflows, inventory restocking alerts, and real-time customer queries**.

**Inventory Management System:**

* Real-time stock updates **upon order placement** will ensure accuracy.
* **Low-stock alerts** to prevent overselling.

**Payment Processing Integration:**

* **Secure payment gateways** (such as PayPal, Stripe, or Razorpay) will be linked to enable transactions.
* **Transaction logs** for tracking successful and failed payments.

**User Review & Ratings System:**

* Customers can leave **feedback & ratings**, enhancing product recommendations.
* Reviews can be **validated using AI-based filtering** for spam detection.

**Recommendation Engine:**

* Using customer **purchase history & browsing behavior**, personalized product recommendations will be generated.
* AI-based algorithms will analyze trends to **predict user preferences**.

## **Section 6: References**

### **Academic & Technical References**

To ensure a **well-informed and reliable design**, the following references were studied:

1. **Silberschatz, Korth, Sudarshan – *Database System Concepts***

* A foundational textbook covering relational database principles, entity relationships, normalization and indexing.

2. **Online Articles & Tutorials on Database Design**

* Tutorials on **MySQL best practices**, including normalization techniques, indexing methods, and query optimization.

3.**Official MySQL Documentation**

* Guidelines on **triggers, views, foreign key constraints, indexing**, and structured query language (SQL) operations.