**PROJECT REPORT**

**(Grocery Store)**

**Author:**

My name is **P.Y. Rajkamal Tutu**. I am currently pursuing a B.S. in Data Science at IIT Madras, and simultaneously working towards a parallel degree in B.Tech CSE from VVIT, Guntur. You can reach me through my student email: [21f2001482@ds.study.iitm.ac.in](mailto:21f2001482@ds.study.iitm.ac.in). I have a wide range of interests and I'm always eager to learn new things. I am particularly fascinated by technology, data science, and machine learning. Programming in Python and C++ is a passion of mine. In my free time, I enjoy playing games and watching movies.

**Description:**

(E-Grocery Store)

This web application, built with Flask, SQLAlchemy, and SQLite, facilitates online grocery shopping. It offers secure authentication, product management, and user-friendly dashboards for both customers and retailers. Features include cart management, order summaries, and category organization. The project aims for seamless user experience and efficient inventory management.

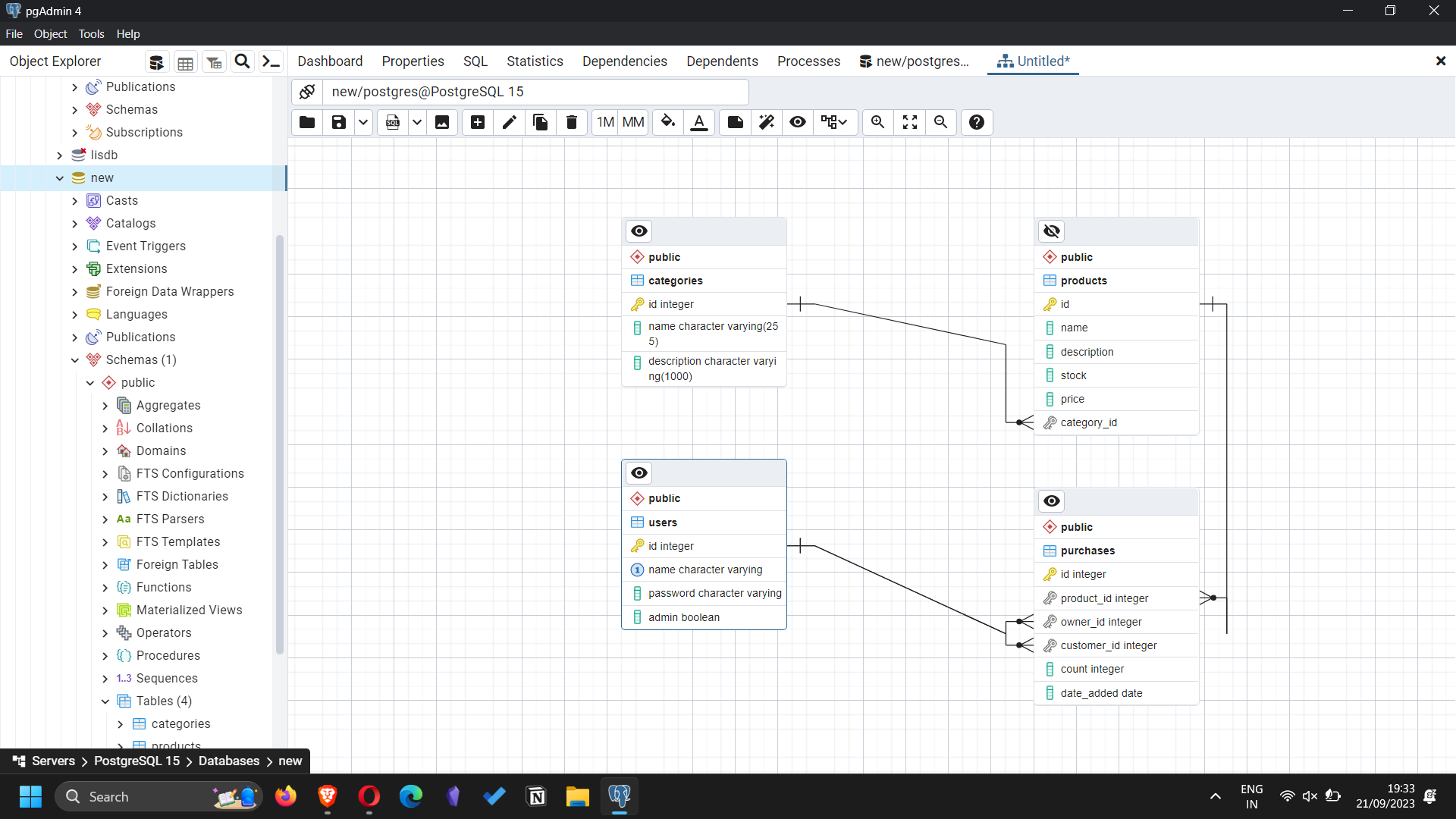
**Technologies Used:**

**Flask for Application Code:** Flask is a lightweight web framework in Python, allowing for rapid development of web applications with minimal boilerplate code. It facilitates easy routing, request handling, and integration of third-party libraries, making it ideal for building dynamic web applications.

**Jinja2 Templates + Bootstrap for HTML Generation and Styling:** Jinja2 is a powerful templating engine that seamlessly integrates with Flask, enabling dynamic content rendering in HTML templates. Bootstrap is a popular front-end framework that provides pre-designed CSS styles and JavaScript components, ensuring a responsive and visually appealing user interface.

**SQLite for Data Storage:** SQLite is a self-contained, serverless, and file-based relational database system, making it suitable for small to medium-sized web applications.It offers simplicity, portability, and ease of setup, allowing for efficient data storage and retrieval within the application.

**DB Scheme Design:**

****

This database appears to be designed for managing a basic e-commerce system. It consists of four tables: User, Category, Product, and Purchases. Each table represents a key entity in an e-commerce system, and the relationships between them are defined to organize and connect the data.  
  
**User Table**:

Purpose: This table is essential for authentication and authorization. It allows the system to identify and differentiate between users.

**Category Table**:

Relationships:

* One-to-Many relationship with the Product table (one category can have multiple products).
* Purpose: Categories help organize products and allow users to browse and filter products by type.

**Product Table**:

This table stores information about individual products.

* Many-to-One relationship with the Category table (each product belongs to one category).
* One-to-Many relationship with the Purchases table (one product can have multiple purchases).

Purpose: This table represents the actual products available in the system, along with their details.

**Purchases Table**: This table records information about purchases made by users.

Purpose: This table tracks the purchases made by users, including details about the products, owners, customers, and quantities.

The database follows a relational model, which is well-suited for structured data like users, products, and categories. It leverages foreign keys to establish relationships between tables, ensuring data integrity. The design separates entities (e.g., users, categories, products) into distinct tables, which allows for efficient querying and maintenance. The relationships between tables (One-to-Many, Many-to-One) mirror real-world associations (e.g., a product belongs to a category; a user can make multiple purchases). The inclusion of an admin field in the User table suggests that there may be different user roles with varying levels of access. Overall, this design facilitates the management of products, purchases, and users within an e-commerce system, providing a structured and organized way to store and retrieve relevant information.

**Architecture and Features:**This is a Flask application that implements a basic e-commerce system with various features. Let's break down the key aspects of this application:

***1. Architecture:***

**Applications Folder**:

* controllers.py: Handles incoming requests, processes data, and decides which templates to render.
* models.py: Defines the database models, outlining the structure of data and its storage in the database (e.g., users, categories, products).
* database.py: Manages the database connection and setup, potentially using SQLAlchemy for object-relational mapping.

**Instance Folder**:

* db.sqlite3: Likely an SQLite database, suitable for small to medium-sized applications, often used during development.

**Static Folder**:

* images: Contains static content like logos or other images.
* products: Holds images specifically related to products, likely associated with database records.

**Templates Folder**:

* Contains HTML templates for rendering the user interface, dynamically populated with data from controllers before being presented to users.

This architecture adheres to the Model-View-Controller (MVC) design pattern, separating concerns in the application:

* **Models** handle data modeling and interaction with the database.
* **Controllers** process requests, apply business logic, and determine which views to render.
* **Views** (HTML templates) present data to end-users.

The application uses SQLite, a self-contained, file-based relational database engine, suitable for small to medium-sized applications.

The static folder stores static files like images, which are served directly to the client.

The instance folder may be used for storing sensitive or environment-specific configurations.

This architecture provides a structured and organized approach to web application development, allowing for separation of concerns and easy scalability.

***2. Features:***

Here are the main features of the application:

* User Authentication:
* Allows users to register, log in, and log out.
* Passwords are securely hashed using the passlib library

Homepage (/):

* Displays a list of products and supports search functionality to filter products by name.
* Product Operations: Admin users can add, edit, and delete products.
* Product details include name, description, stock, price, and category.

Category Management:

* Admin users can add new product categories.

Shopping Cart (/cart):

* Users can add products to their cart and specify quantities.
* Users can remove items from the cart.
* Supports a checkout process that finalizes purchases and updates product stock.

Order History (/history):

* Logged-in users can view their purchase history, which includes details about the products they've bought.

Admin Dashboard (/dashboard):

* Accessible only to admin users.
* Provides an overview of all products, allowing for management (add, edit, delete).

Summary (/summary):

* Displays aggregated data about product sales.
* Features include top-selling products and total revenue.

Search (/search): Enables users to search for products by name.