**Database Systems Project Part IV  
 End-to-End Solution Integration and Data-Driven / Database Programming**

Fangru Li Chenxing Li Yuwen Zhong

[fl1099@nyu.edu](mailto:fl1099@nyu.edu) [cl7451@nyu.edu](mailto:cl7451@nyu.edu) [yz11737@nyu.edu](mailto:yz11737@nyu.edu)

### Introduction

In Part 4 of this project, the focus shifts to the implementation of an end-to-end solution for the e-commerce food retailer's database management system. This includes designing and implementing business use cases to create a workflow-based database application that integrates machine learning models, managing data pipelines, and ensuring seamless re-training of models as unstructured data evolves. The solution also leverages database connectivity frameworks and optimizations, including query enhancements and optional Object Relational Mapping (ORM) usage, to deliver a robust and efficient system. Finally, the reference architecture is finalized, addressing key aspects of data governance, DIKW (Data-Information-Knowledge-Wisdom) frameworks, and operational transparency.

For full implementation details, the project code is available on GitHub: <https://github.com/Chenxing02/Database-Project>.

# Business Use Case (Big picture)

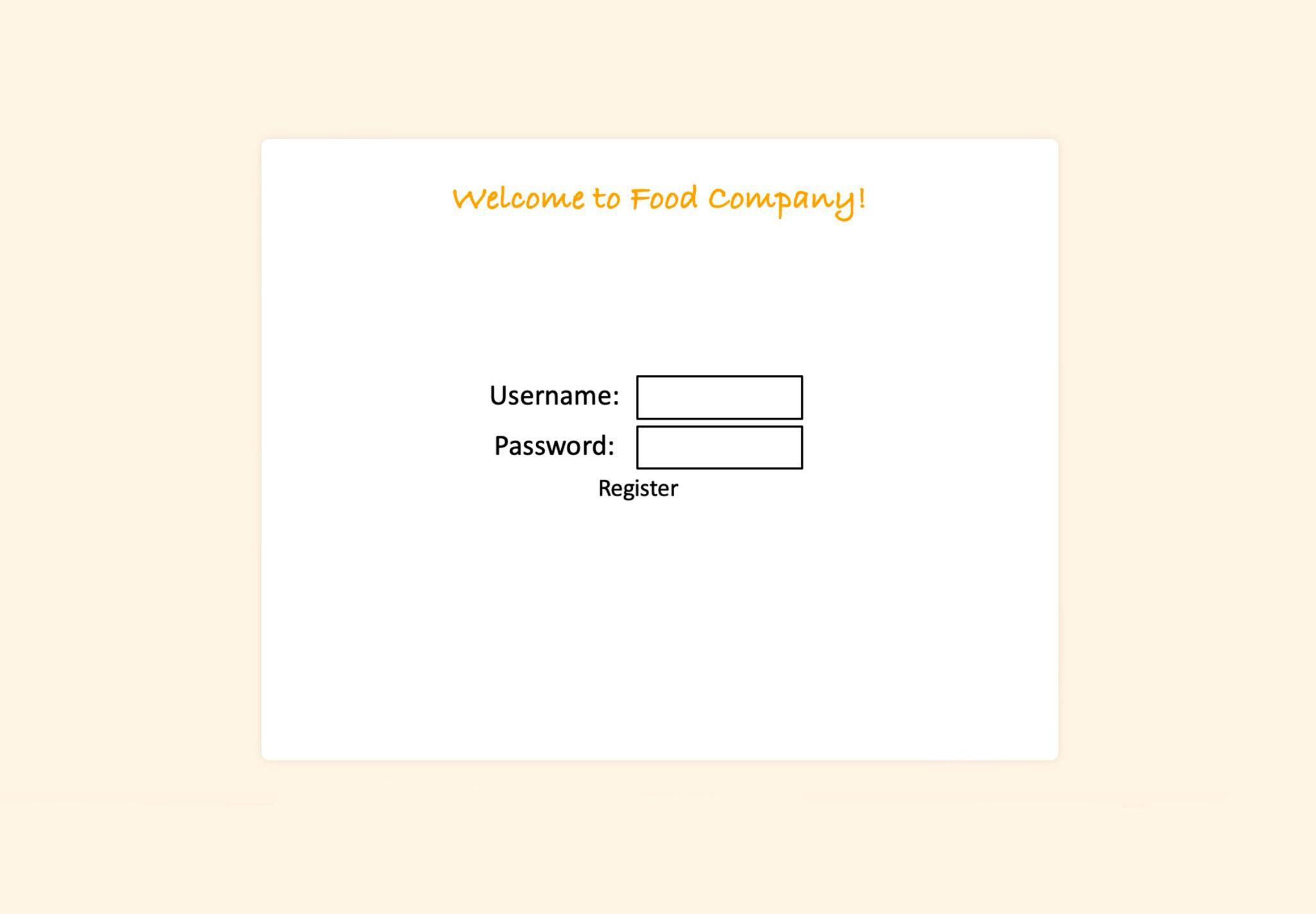
The database management system designed specifically for e-commerce food retailers. This platform offers a comprehensive solution for managing customers, products, and orders, enabling businesses to streamline their operations and enhance their service efficiency. Key features of the platform include customer relationship management tools, inventory tracking, order processing, and more. Additionally, it provides an intuitive interface for seamless data entry and retrieval.

# Process and Design Documentation

**Register**

Administrators can create a new account by providing their basic details such as name, email, phone number, and role (Senior/Junior). Passwords are securely hashed before being stored in the database. Senior administrator registration requires additional approval by an existing senior administrator.

**Login**

Administrators can log in by entering their email and password. The system validates credentials and assigns access based on their role. Failed attempts trigger appropriate error messages, and successful logins redirect users to their role-specific dashboard.

**Product page and add a product**

Page Overview:

* Search Bar: Allows users to search for products by entering keywords related to the product's name, SKU, or other attributes.
* Add Button: Opens a pop-up form to add a new product to the inventory.
* Edit Button: Enables users to modify the details of an existing product by selecting it from the list and making changes.
* Delete Button: Permanently removes the selected product from the inventory.

Adding a Product:

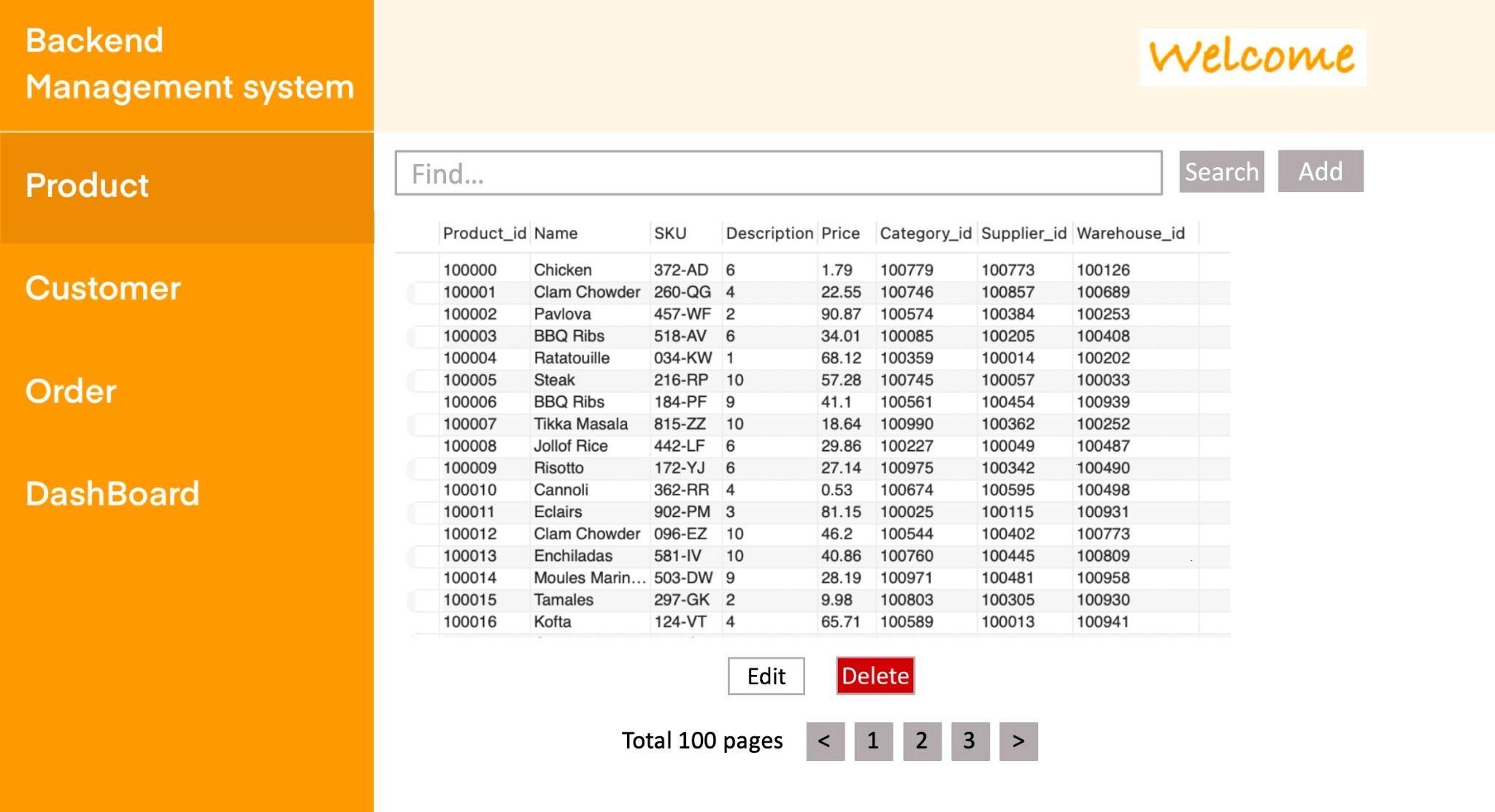
* Click the "Add" button to open the "Add Product" form.
* Fill in the following fields:
  + Name: The name of the product.
  + SKU: A unique stock-keeping unit identifier.
  + Price: The price of the product.
  + Category: The category to which the product belongs.
  + Supplier: The supplier providing the product.
  + Warehouse: Select the warehouse where the product is stored from the dropdown menu.
  + Description: Additional details or specifications about the product.
* Once all fields are completed, submit the form to add the product.

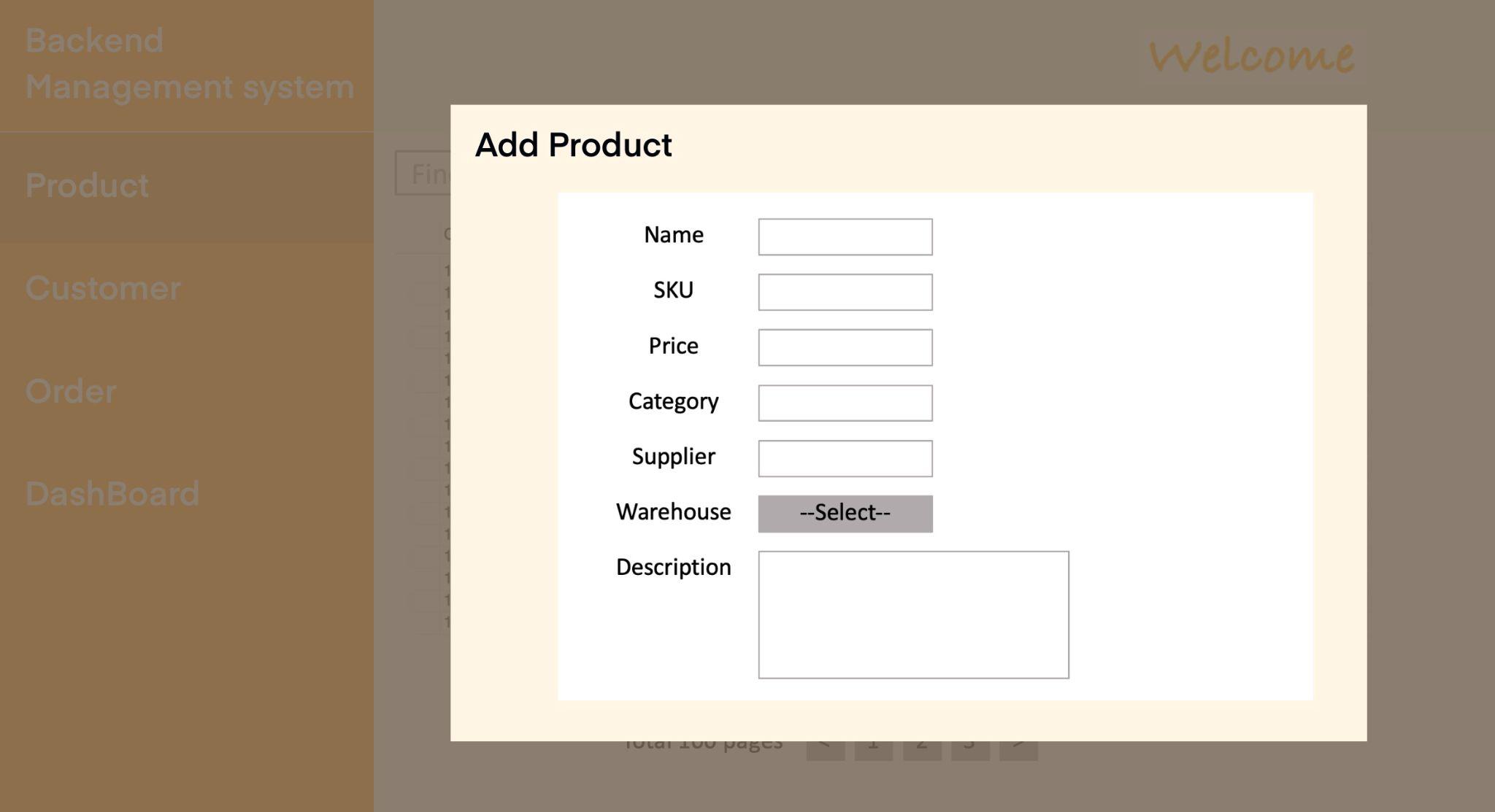
Synchronization with SQL Database:

* After submitting the form, the new product data is automatically synced with the underlying SQL database. This ensures the product is immediately available in the inventory system, visible on the product page, and accessible for search, editing, or deletion.

Pagination:

* Navigate through the product list using the pagination buttons (<, 1, 2, 3, >), which help manage large inventories by displaying a limited number of products per page.





**Customer page and add/review a customer**

Adding a Customer:

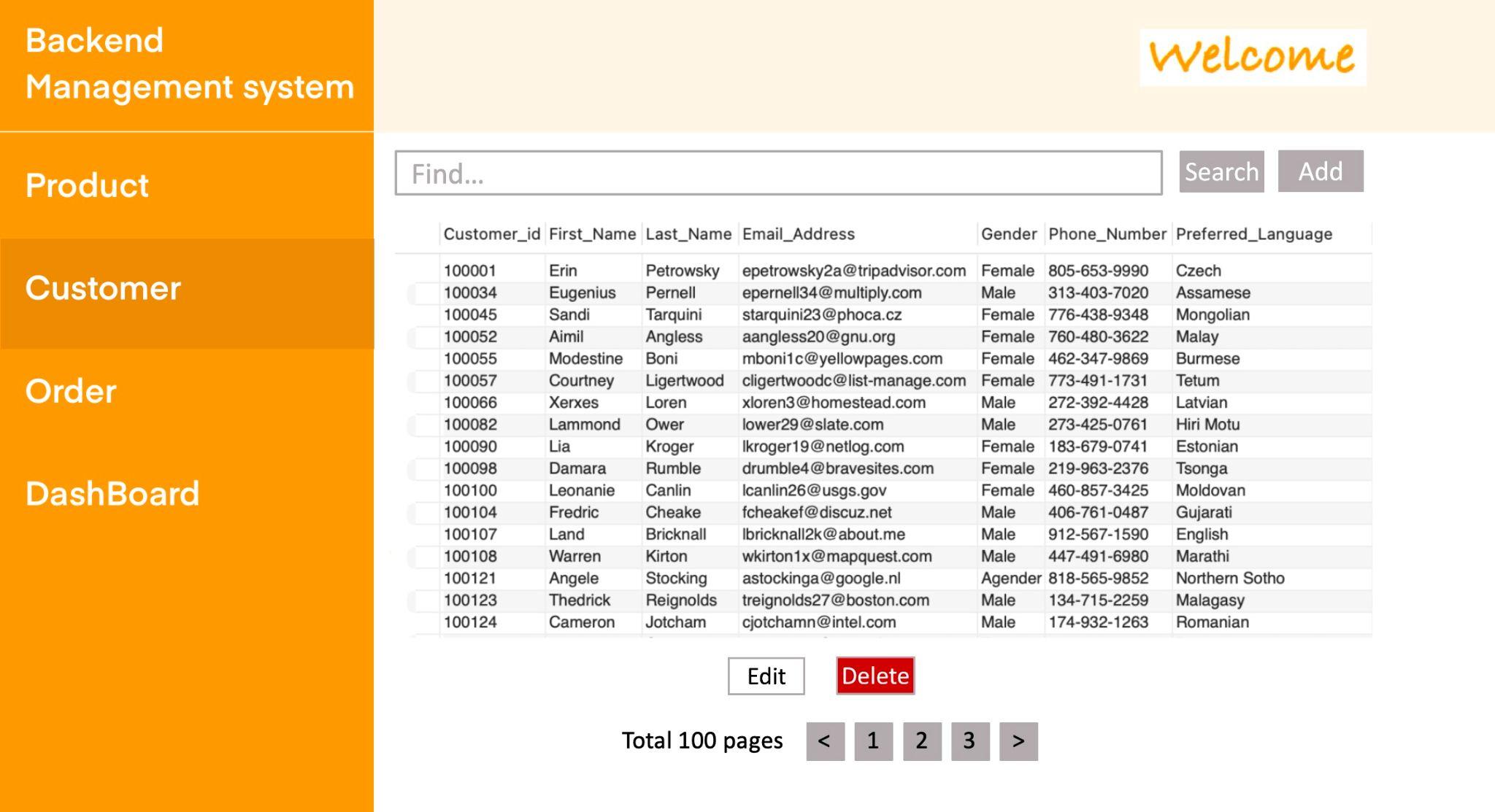
* Click the "Add" button to open the "Add Customer" form.
* Fill in the following fields:
  + First Name: The first name of the customer.
  + Last Name: The last name of the customer.
  + Gender: The customer's gender.
  + Email: The email address of the customer.
  + Phone: The customer's phone number.
  + More Detail: Additional notes or details about the customer.
* Once all fields are completed, submit the form to add the customer.

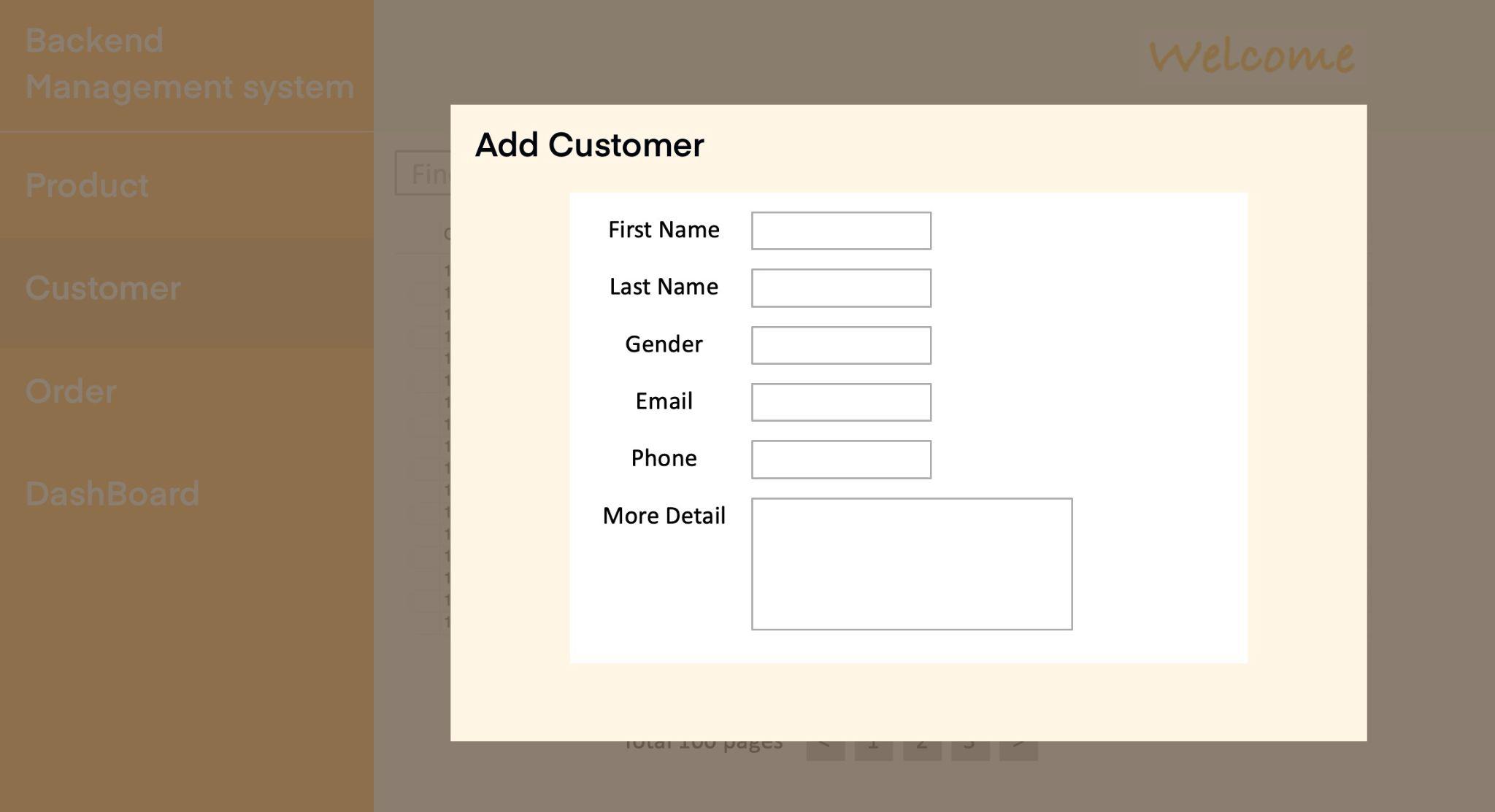
Synchronization with SQL Database:

* After submitting the form, the new customer data is automatically saved to the underlying SQL database. This ensures the customer's information is immediately available in the system for search, editing, or deletion.

Customer Management:

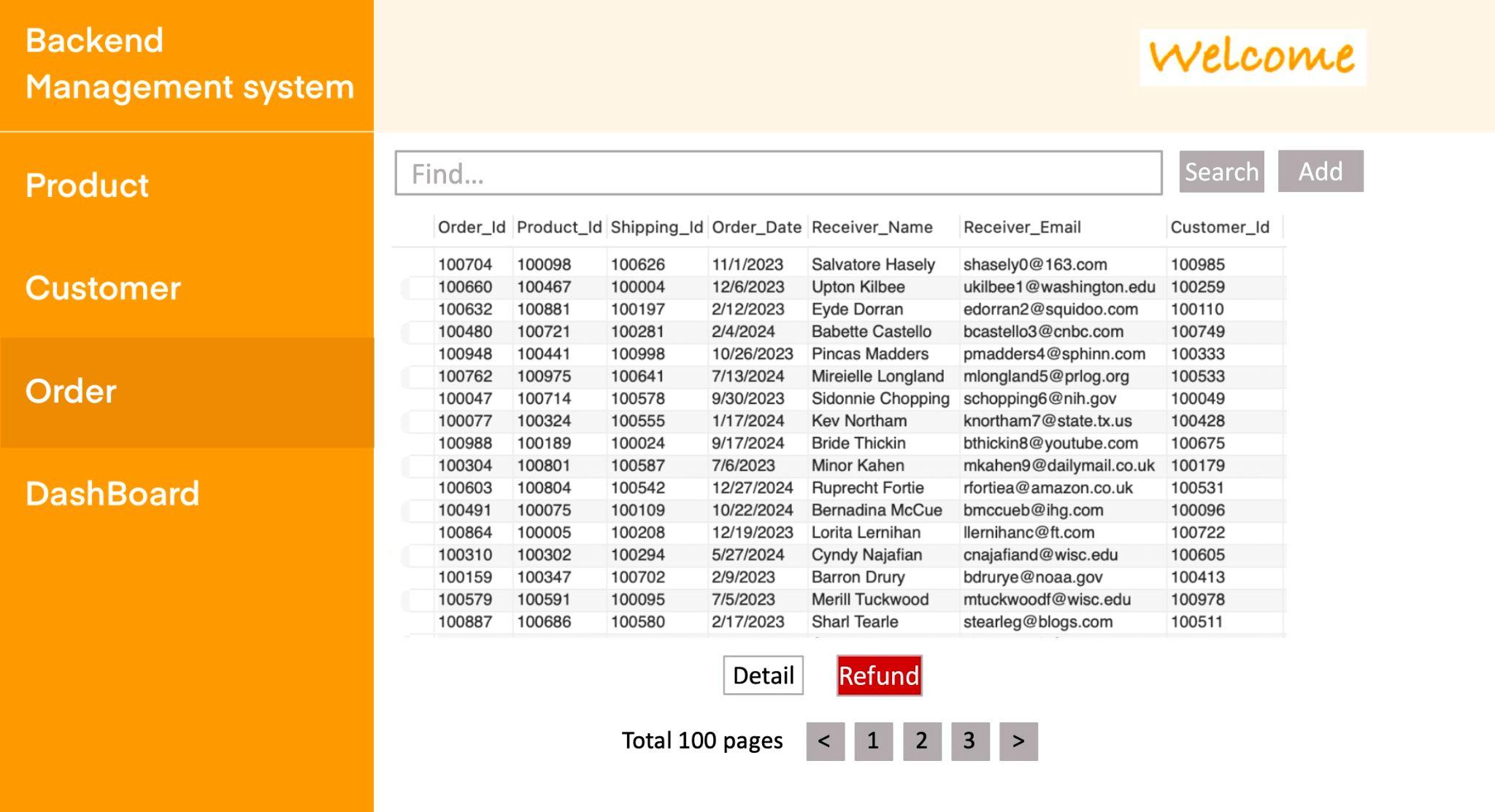
* This functionality enables users to easily maintain an up-to-date customer database, ensuring all customer-related information is stored and retrievable in an organized manner.

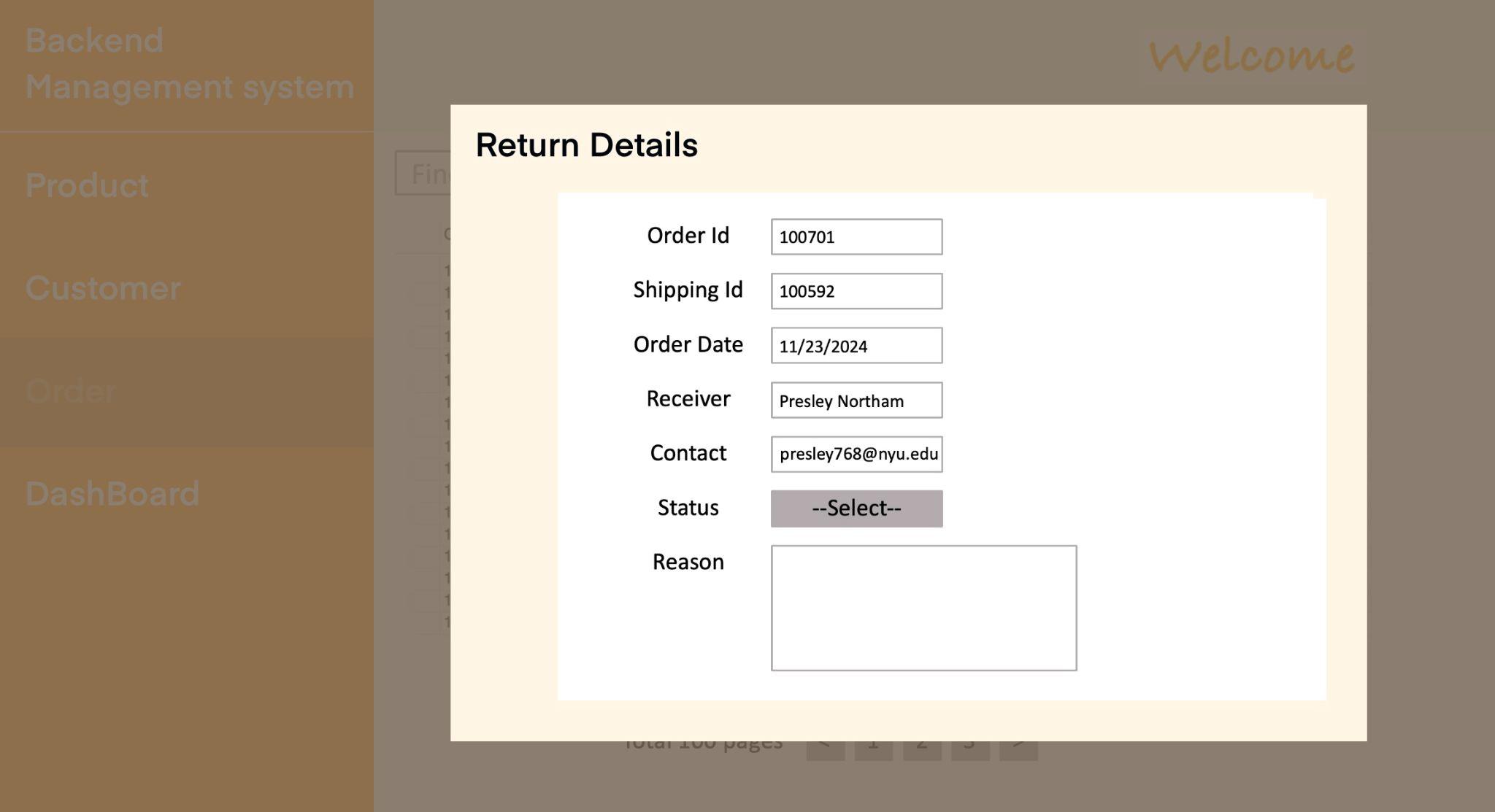




**Order Page and Return/Refund Page**

1. Overview of Buttons and Features:
   * Search Bar: Allows users to search for orders by keywords such as Order ID, Product ID, Shipping ID, or Receiver details.
   * Detail Button: Displays the detailed information of a specific order, such as the items purchased, shipping details, and status.
   * Refund Button: Opens a pop-up form to initiate the return or refund process for a specific order.
2. Initiating a Return or Refund:
   * Select an order from the list and click the "Refund" button to open the "Return Details" form.
   * Fill in the following fields in the pop-up form:
     + Order ID: Auto-filled with the selected order's ID.
     + Shipping ID: Auto-filled with the corresponding shipping information.
     + Order Date: Auto-filled with the order's placement date.
     + Receiver: Auto-filled with the customer's name.
     + Contact: Auto-filled with the customer's email or phone number.
     + Status: Select the return/refund status from a dropdown (e.g., Pending, Approved, Rejected).
     + Reason: Provide a brief explanation for the return or refund request.
   * Once the form is completed and submitted, the data is automatically synced with the SQL database, updating the order's status.
3. Synchronization with SQL Database:
   * All changes, such as return or refund requests, are automatically recorded in the SQL database. This ensures that the updated status is visible in the system for future references or actions.
4. Order and Refund Management:
   * This functionality ensures that orders are effectively tracked and managed, while returns and refunds are handled in an organized manner.
   * The integration of a refund system allows businesses to maintain transparency and provide better customer service.





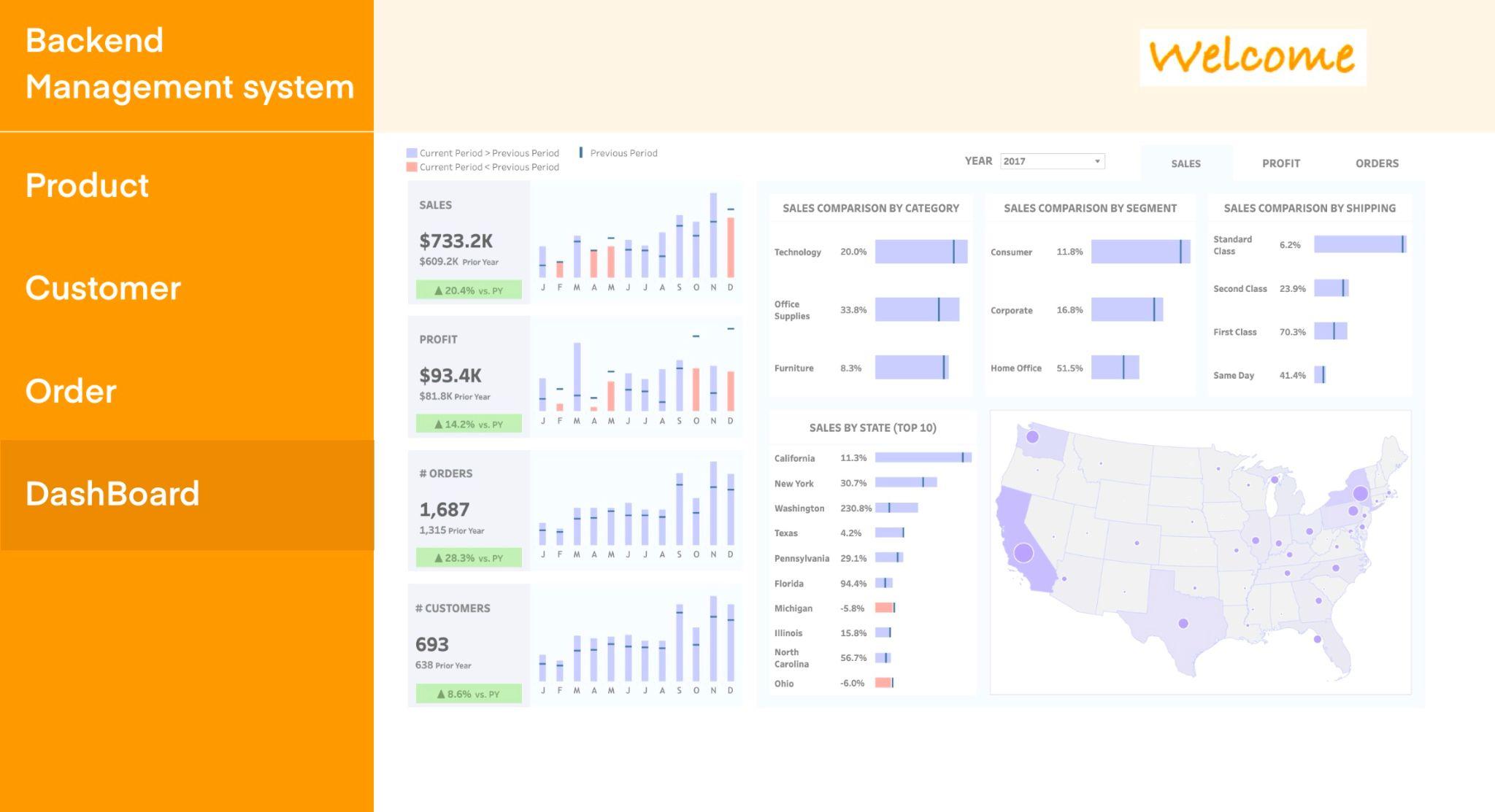
**Data Analysis Page (Connect to Tableau)**

Connecting to Tableau:

* Database Integration: Tableau connects to the backend SQL database of the e-commerce management system.
* Real-Time Data: Enable live connection in Tableau for real-time updates or set up an extract to periodically sync data.
* Data Preparation: Use Tableau Prep Builder or SQL queries to clean and structure the data into formats suitable for visualization.

E-Commerce Use Cases:

* Identify Trends: Spot monthly or seasonal trends in sales, profits, and orders to adjust inventory and marketing strategies.
* Optimize Product Offerings: Use category and segment data to determine which products or customer types drive revenue.
* Improve Logistics: Analyze shipping method performance to reduce costs and improve customer satisfaction.
* Target Marketing: Use customer growth and geographical sales data to target high-value markets



# 

# Data-Driven Module Development

# We developed a web-based application for predicting the membership level of customers in a food company using their demographic information and yearly spend. It leverages a Decision Tree Classifier to classify customers into five levels based on their spending patterns. Detailed instructions on running the application locally and implementation details are in the Github link readme file.

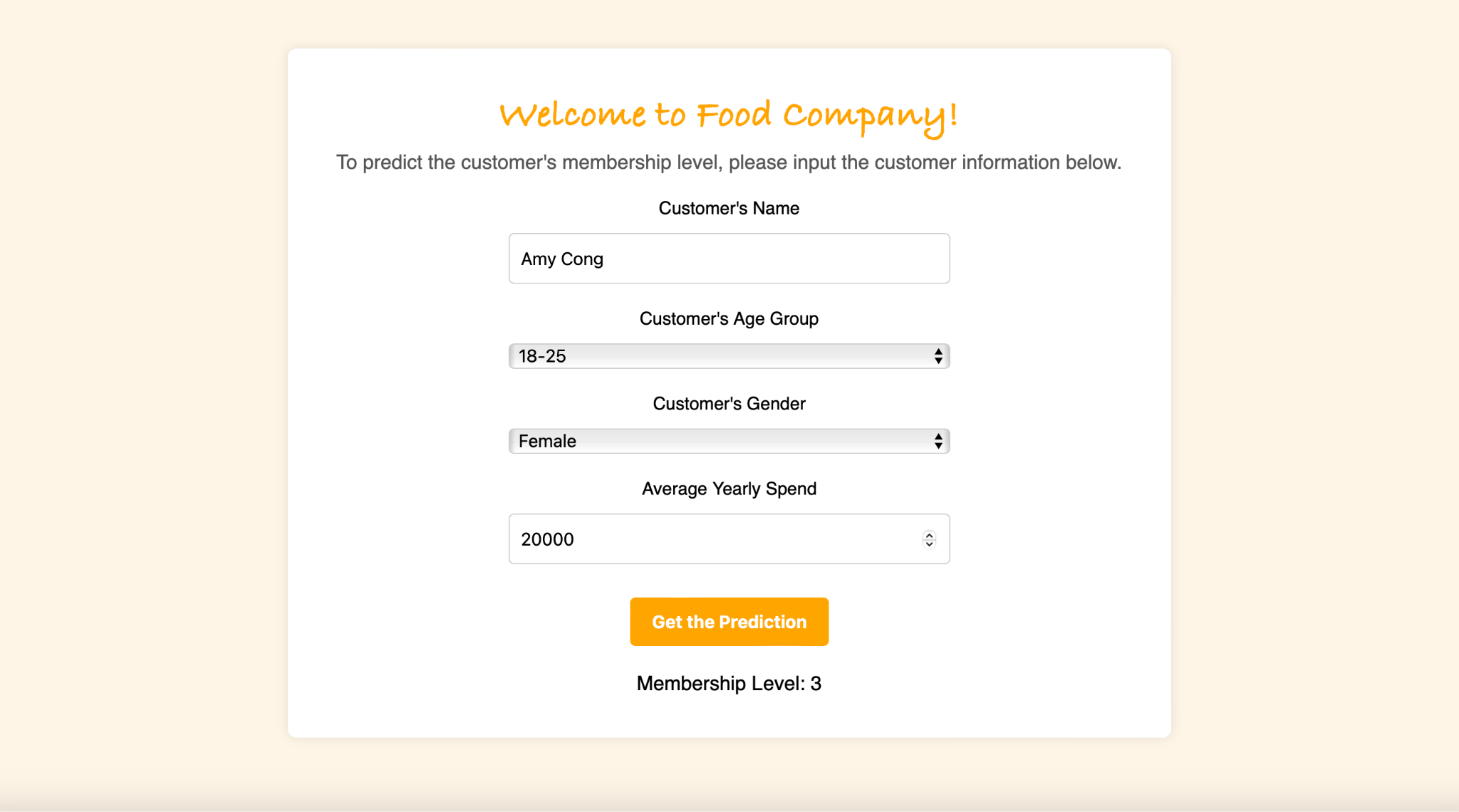
# The application leverages data-driven techniques by extracting and processing relevant features from a MySQL database. For the Decision Tree Classifier, we discretize the yearly spending feature into quantiles to enable a classification-friendly format, optimizing the predictions of accurate membership level. The model adapts dynamically to customer spending trends, and ensures actionable insights derived from robust data. The membership-level prediction application supports critical business functions such as targeted marketing, inventory management, dynamic pricing, and customer service prioritization. By identifying high-spending customers, the company can create personalized promotions and loyalty programs to boost retention, while incentivizing engagement from lower-tier members. Insights into spending patterns enable accurate demand forecasting, optimizing inventory and reducing waste, enhancing overall customer satisfaction and driving revenue growth.

# Workflow-Based Application

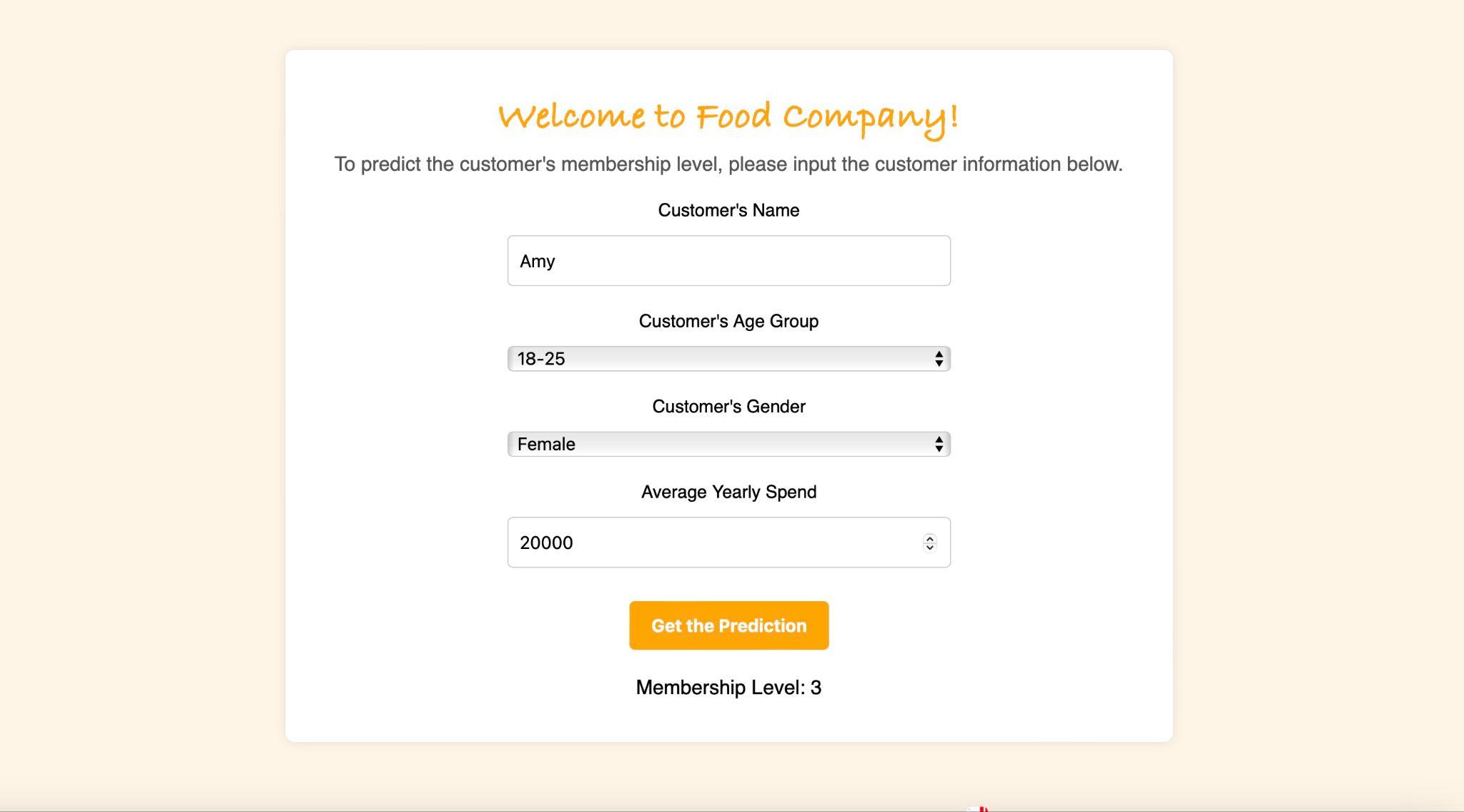
The workflow integrates user-friendly front-end input forms with a Flask-based back-end API. Users submit spending and demographic data via the interface, which is validated and processed in real-time. The back-end retrieves pre-trained models, performs predictions, and dynamically returns results to the interface. This seamless workflow ensures efficient interaction between front-end and back-end systems, providing an end-to-end solution.

# Different test cases:

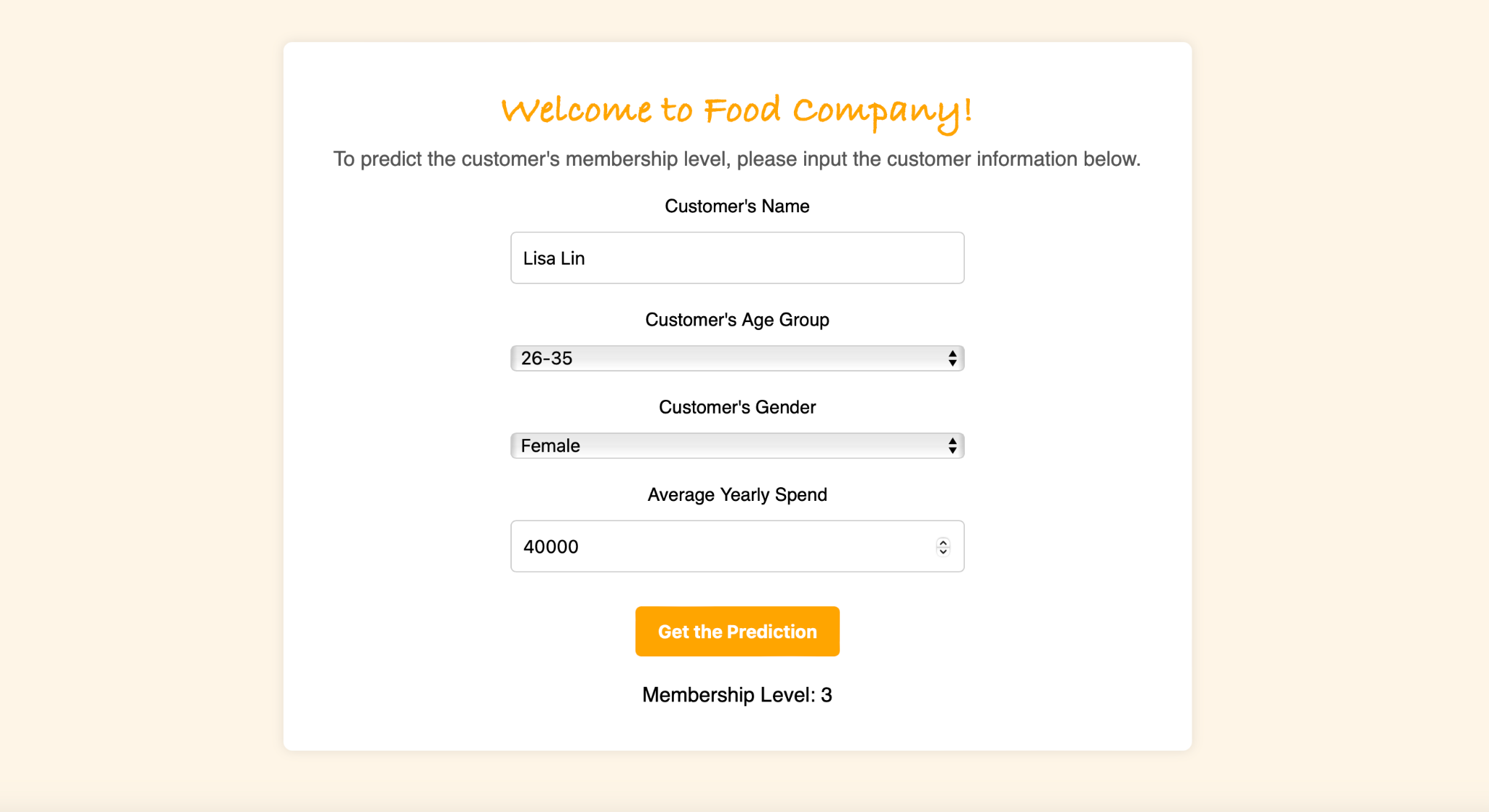
1. Customer Amy Cong, Age 18-25, Female, yearly spending 20000 on related product



1. Tom Elmond, Age 36-50, Male, yearly spend 40000 on related products



1. Lisa Lin, Age 26-35, Female, yearly spend 40000 on related products



# Application Documentation

The application developed for the e-commerce food retailer’s database management system (DBMS) is designed to meet the specific needs of two user roles: senior administrators and junior administrators. This section provides detailed documentation of how the application fulfills the requirements set out in the user cases, showcasing its features, workflows, and query optimizations. Due to the time constraints, we were unable to implement all the functionality with the code, which leaves an opportunity for future work to complete and finalize.Below are the core functionalities and their implementations:

## User Management and Access Control

The system allows administrators to securely register and log in, ensuring appropriate role-based access. Senior administrators have extended privileges for managing strategic tasks, while junior administrators handle day-to-day operations.

* **Registration**: Senior administrators require approval from an existing senior administrator before their account is activated.
* **SQL Query**:

INSERT INTO Admin\_Account (Admin\_ID, Name, Email, Role, Approval\_Status)

VALUES (1, 'Alice', 'alice@example.com', 'Senior', 'Pending');

* **Login**: After entering credentials, the system validates the user’s email and password, granting access to role-specific dashboards.
* **SQL Query**:

SELECT Role

FROM Admin\_Account

WHERE Email = 'alice@example.com' AND Password\_Hash = 'hashed\_password';

## Customer Account Management

Administrators can manage customer accounts, including creation, updates, and retrieving details such as shipping and billing addresses.

* **Create a New Customer Account:** Both administrators can create a new customer account by entering customer details.
* **SQL Query:**

INSERT INTO Account\_Info (Customer\_id, First\_Name, Last\_Name, Email\_address, Gender, Phone\_number, Preferred\_Language)

VALUES (1, 'John', 'Doe', 'john.doe@example.com', 'Male', '1234567890', 'English');

* **Update Customer Information (Junior Administrators):** When a customer updates their email or phone number, junior administrators can modify the record.
* **SQL Query:**

UPDATE Account\_Info

SET Email\_address = 'new.email@example.com', Phone\_number = '0987654321'

WHERE Customer\_id = 1;

* **View Customer Order History (Both Administrators):** Administrators can review complete customer order histories for better service or analysis.
* **SQL Query:**

SELECT \* FROM Order\_History

WHERE Customer\_id = 1;

* **View Membership Status and Tier Information**: Both administrators can check detailed membership levels and statuses to provide personalized customer support.
* **SQL Query:**

SELECT \*

FROM Customer\_Info

WHERE Customer\_id = 1;

* **Track Expired Memberships**: Senior administrators can identify customers with expired memberships to target them for renewal campaigns, improving customer retention.
* **SQL Query:**

SELECT Customer\_id, Membership\_Status

FROM Customer\_Info

WHERE Membership\_Status = 'Expired';

## Order Processing and Management:

## Junior administrators manage tasks such as order creation, refunds, and shipping tracking, while senior administrators analyze trends and campaign effectiveness.

* **Process a New Order**: Junior administrators record new orders, including payment and shipping details.
* **SQL Query:**

INSERT INTO Order\_History (Order\_id, Customer\_id, Order\_Date, Total\_Amount, Item\_Amount, Order\_Status)

VALUES (101, 1, '2024-12-19', 150.00, 3, 'Processing');

* **Cancel an Order and Update Inventory**: Administrators cancel an order and restock the inventory.
* **SQL Query:**

DELETE FROM Order\_History

WHERE Order\_id = 101;

UPDATE Inventory

SET Quantity = Quantity + 3

WHERE Product\_id = 202;

* **Track Shipping Details：**Junior administrators monitor shipping information to ensure timely delivery of orders.
* **SQL Query:**

SELECT \*

FROM Shipping\_Information

WHERE Order\_id = 101;

## Product and Inventory Management

Administrators handle product additions, inventory updates, and promotional campaigns.

* **Add a New Product**: Administrators can add products along with their categories and supplier details.
* **SQL Query:**

INSERT INTO Product (Product\_id, Name, Price, Category\_id, Supplier\_id)

VALUES (301, 'Organic Apples', 3.50, 101, 201);

* **Update Inventory Levels** (Junior Administrators): Adjust inventory quantities to maintain accurate stock levels.
* **SQL Query:**

UPDATE Inventory

SET Quantity = Quantity - 10

WHERE Product\_id = 301 AND Warehouse\_id = 1;

* **Add Product Discounts**: Senior administrators can assign discounts to products for promotional events and campaigns.
* **SQL Query:**

INSERT INTO Discount (Discount\_id, Product\_id, Percentage)

VALUES (401, 301, 10);

## Reporting and Insights

Senior administrators generate insights for operational improvements, such as customer segmentation and analyzing discount effectiveness.

* **Analyze Yearly Average Spending for Segmentation**:Senior administrators assess customer spending to group them into tiers.
* **SQL Query:**

SELECT Customer\_id, AVG(Total\_Amount) AS Avg\_Yearly\_Spend

FROM Order\_History

GROUP BY Customer\_id;

* **Generate a Discount Report:** Evaluate the total discounts applied across all orders during a specific campaign.
* **SQL Query:**

SELECT SUM(Discount\_Amount) AS Total\_Discount

FROM Order\_Discount

WHERE Order\_Date BETWEEN '2024-01-01' AND '2024-12-31';

* **Analyze Return Trends**: Senior administrators can analyze return reasons to improve product quality and reduce returns.
* **SQL Query:**

SELECT Return\_Reason, COUNT(\*) AS Occurrences

FROM Return\_Refund

GROUP BY Return\_Reason

ORDER BY Occurrences DESC;

* **Identify Top-Rated Products**: Administrators can determine the most popular products based on customer reviews and ratings.
* **SQL Query:**

SELECT Product\_id, AVG(Rating) AS Avg\_Rating

FROM Product\_Review

GROUP BY Product\_id

ORDER BY Avg\_Rating DESC

LIMIT 10;

## Query Optimization and Integration

Queries are optimized for performance, ensuring fast and efficient operations.

* **Indexing:** Indexes are created on frequently queried fields like Customer\_id, Order\_id, and Product\_id to reduce execution time.
* **SQL Query Example:**

CREATE INDEX idx\_customer\_id ON Account\_Info(Customer\_id);

* **Partitioning:** Partitioning is employed to handle large datasets effectively by dividing tables into smaller, more manageable segments.
* **SQL Query Example:** This query scans only the canada partition, optimizing the query execution for region-specific data retrieval.

ALTER TABLE customer\_shipping\_info

PARTITION BY LIST (Country) (

PARTITION usa VALUES ('USA'),

PARTITION canada VALUES ('Canada'),

PARTITION uk VALUES ('UK'),

PARTITION other VALUES (DEFAULT)

);

SELECT \*

FROM customer\_shipping\_info

WHERE Country = 'Canada';

* **Clustering：**Clustering optimizes the physical storage of related data to minimize disk I/O operations during frequent joins and sequential data access.
* **SQL Query Example:** The clustering ensures that both Orders and Order\_Item data for Order\_ID = 101 are accessed together, significantly reducing the time needed for the join operation.

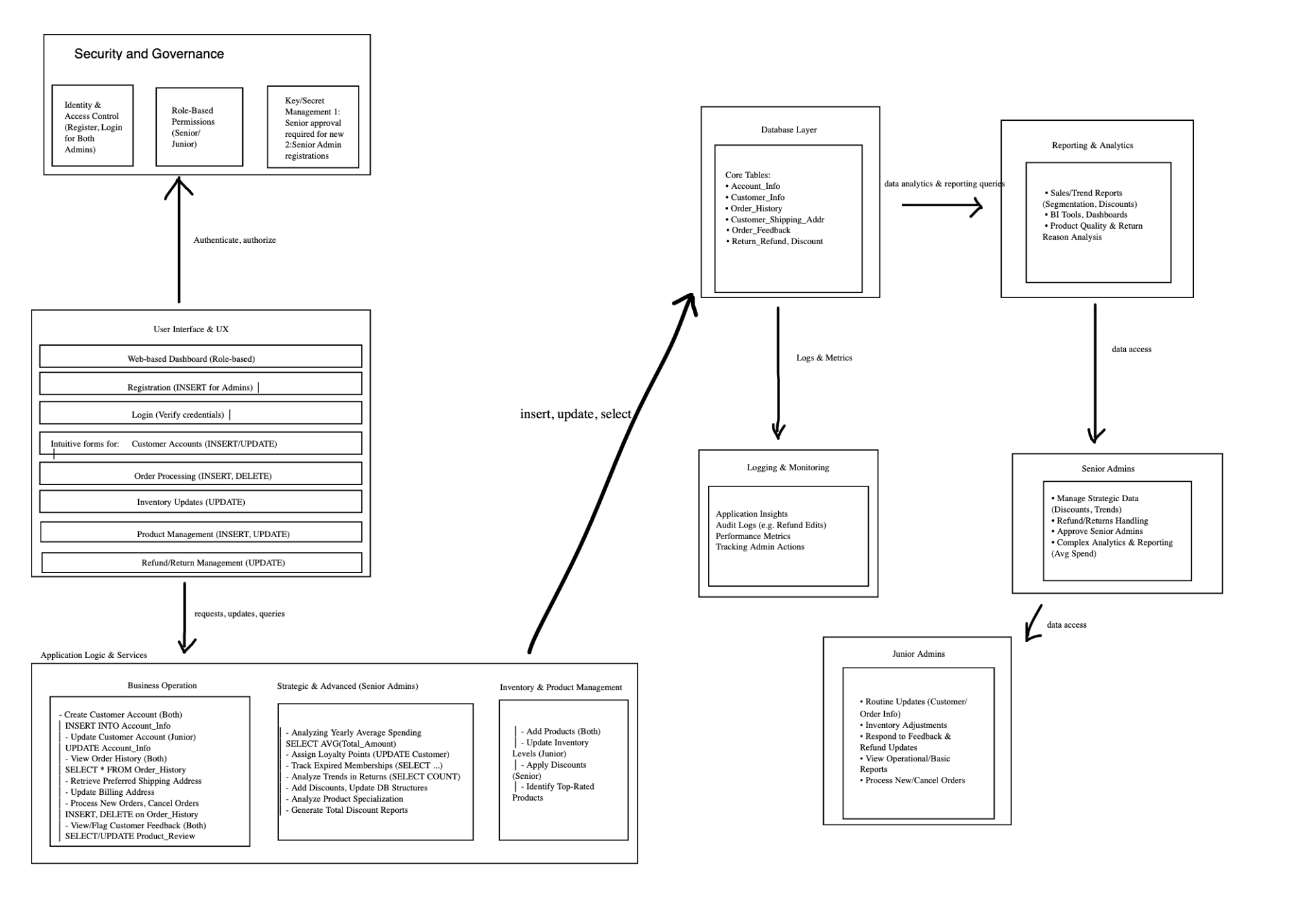
SELECT O.Order\_ID, O.Order\_Date, O.Total\_Amount, I.Product\_ID, I.Quantity, I.Price

FROM Orders O

JOIN Order\_Item I ON O.Order\_ID = I.Order\_ID

WHERE O.Order\_ID = 101;

## 6. Reference Architecture



RA Foundation Principle

1. Business Alignment

Our framework ensures efficient order processing, seamless customer account management, and powerful analytics that must improve commerce retailers' competitiveness, operational efficiency, and customer satisfaction.

1. Security and Privacy by Design

Integrate security, identity management, and privacy protection into every layer. For example, refunds and discount management are performed only by authorized personnel, thus protecting customer data and maintaining compliance with regulations.

1. Transparency and Accountability

The framework gives us a clear view of every action, decision and data change. Logging and monitoring systems maintain detailed audit trails. These safeguards support internal accountability and external compliance.

Organizational Framework:

We have five major layers, namely the Business Domain Layer, Application Layer, Data Layer, Technology Layer, and Cross-Services Layer.

1. Business Domain Layer:

Focus on managing customer accounts, processing orders, and handling product inventory. This maps Senior Admins to Junior Admins and Customer Registration, Order Management, Refunds, Returns, Analytics.

1. Application Layer:

Accommodates the logic and services that implement business functions. Applications handle customer relationship management functions, order fulfillment, inventory tracking, and generating management reports. Ensure that senior administrators have access to more strategic and structural functions such as adding products and scheduling discounts, while junior administrators focus on operational tasks such as updating customer records and adjusting inventory.

1. Data Layer:

Manage structured and unstructured data including customer profiles, order history, inventory records, product details, feedback and return/refund records. Data models and schemas are designed for accuracy, normalization, and referential integrity.

1. Technology Layer:

Includes infrastructure, database, and identity management. It ensures high availability, performance optimization, and scalability.

1. Cross-cutting services:

Security, identity and access management, logging, monitoring, and data governance services cover all layers. These ensure consistent enforcement of policies, standards, and best practices.

Data Governance

1. Data Quality Management: Implement validation, cleansing and enrichment processes at the point of data entry. Regularly review customer, product and order data to ensure completeness and accuracy.
2. Prevent Data Loss and Leakage: Utilize encryption, access control, and thus maintenance of backups to maintain data integrity.
3. Data Lifecycle Management: Define retention policies for different data sets. Automate archiving and deletion processes to minimize storage costs and comply with privacy regulations.
4. Limit decision-making power of automated systems: Route high-risk decisions such as major refunds, database structure changes, and volume discounts to senior administrators for approval.
5. Accountability and transparency: Document all administrative actions (customer record updates, order cancellations, loyalty point assignments). Implement a reporting tool that generates audit reports, enabling internal and external stakeholders to verify the integrity of the system.
6. Compliance: Comply with data protection laws (GDPR, CCPA) and industry standards (PCI-DSS for payment data), implement proper consent management, secure data processing and timely breach reporting.

# **Conclusion**

The end-to-end solution developed in Part 4 successfully integrates data-driven insights, machine learning models, and real-time database operations into a cohesive and scalable application. By addressing business use cases through workflow-based design, the solution enhances operational efficiency and user experience. Advanced database optimizations and adherence to comprehensive governance principles ensure data integrity, security, and fairness across the system. This implementation not only meets the project requirements but also establishes a foundation for future enhancements and strategic decision-making in the e-commerce food retail domain.