E-Commerce Analytics & Management

Transforming Data into Seamless Shopping Experiences

Objective

The goal of this project is to develop a **Comprehensive E-Commerce Order Management System** for managing the entire flow of product orders, customer details, payments, and delivery. The system will allow customers to browse and order products, track their orders, make payments, and interact with customer support. The back-end will utilize **MySQL** for transactional data and **MongoDB** for handling customer reviews, product feedback, and file storage (like invoices, shipping labels) stored in **AWS S3**.

This project will help attendees gain hands-on experience in building robust e-commerce systems by implementing APIs for CRUD operations, integrating with MySQL and MongoDB, and working with file storage and cloud services.

Brief Requirement

The system should allow customers to:

1. **Browse products** based on categories (e.g., electronics, clothing).
2. **Add products to cart** and proceed to checkout.
3. **Make payments** using multiple methods (Credit Card, PayPal, etc.).
4. **View order history** and track delivery status.
5. **Leave product reviews and ratings**.

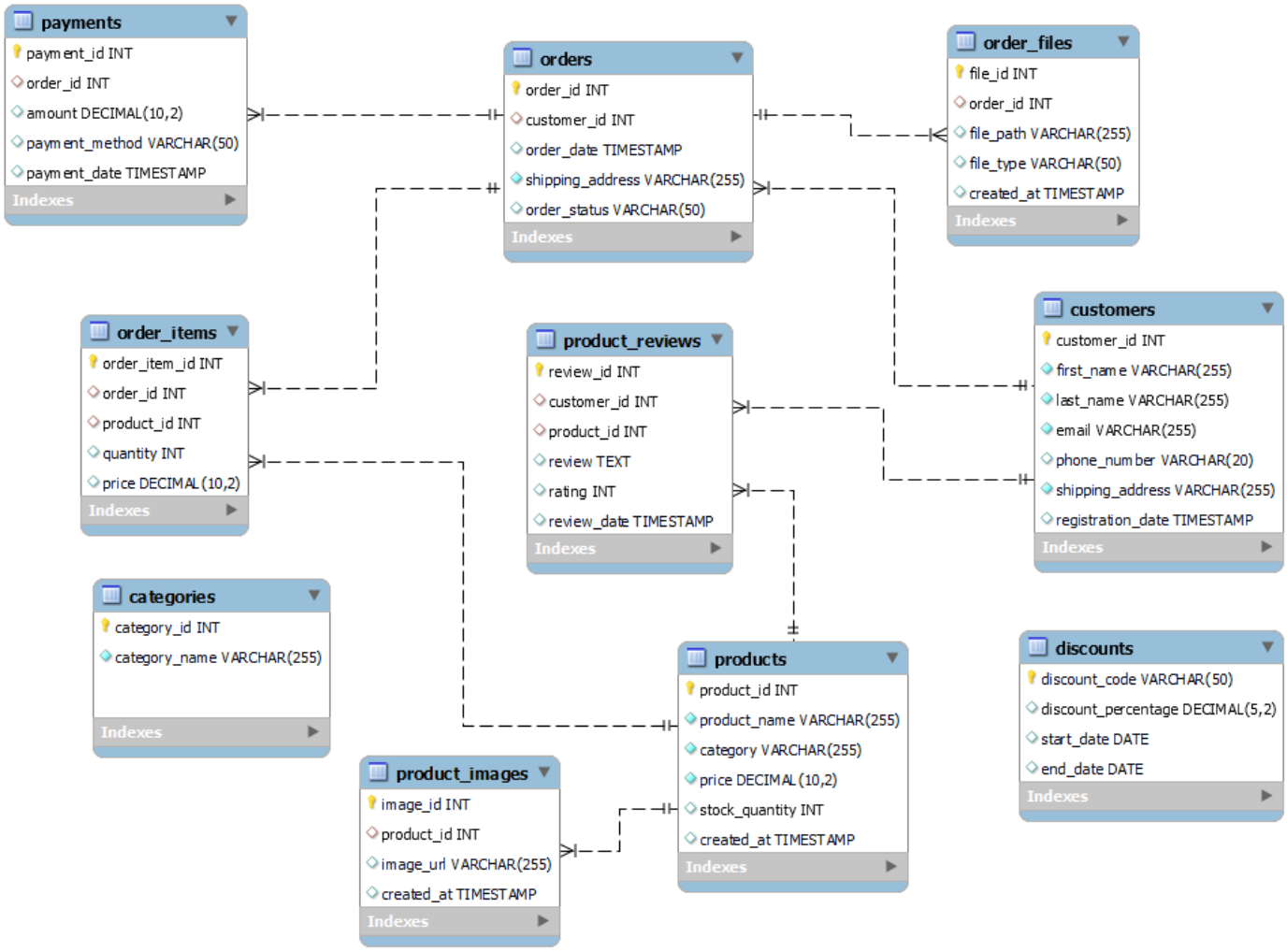
The admin section will allow:

1. **Managing products**, such as adding, updating, or deleting products.
2. **Managing orders**, including processing payments and updating order status.
3. **Viewing sales statistics and customer feedback**.

MySQL will be used for structured data, including customer details, orders, and payment information. **MongoDB** will be used for unstructured data, including product reviews and file paths for invoices and delivery labels stored in **AWS S3**.

Database Architecture

- **MySQL**: Used for structured, transactional data like customers, orders, products, and payments.
  - **Customers**: Store customer details (name, address, contact info).
  - **Products**: Store product details like name, category, price, and stock.
  - **Orders**: Store orders placed by customers.
  - **Payments**: Track payments made for each order.
- **MongoDB**: Used to store unstructured feedback data and file paths for invoices and shipping labels.
  - **Product Reviews**: Store customer reviews and ratings for products.
  - **Order Files**: Store file paths for invoices and shipping labels.


payments
payment_id INT
order_id INT
amount DECIMAL(10,2)
payment_method VARCHAR(50)
payment_date TIMESTAMP
Indexes
orders
order_id INT
customer_id INT
order_date TIMESTAMP
shipping_address VARCHAR(255)
order_status VARCHAR(50)
Indexes
order_files
file_id INT
order_id INT
file_path VARCHAR(255)
file_type VARCHAR(50)
created_at TIMESTAMP
Indexes
order_items
order_item_id INT
order_id INT
product_id INT
quantity INT
price DECIMAL(10,2)
Indexes
product_reviews
review_id INT
customer_id INT
product_id INT
review TEXT
rating INT
review_date TIMESTAMP
Indexes
customers
customer_id INT
first_name VARCHAR(255)
last_name VARCHAR(255)
email VARCHAR(255)
phone_number VARCHAR(20)
shipping_address VARCHAR(255)
registration_date TIMESTAMP
Indexes
categories
category_id INT
category_name VARCHAR(255)
Indexes
products
product_id INT
product_name VARCHAR(255)
category VARCHAR(255)
price DECIMAL(10,2)
stock_quantity INT
created_at TIMESTAMP
Indexes
discounts
discount_code VARCHAR(50)
discount_percentage DECIMAL(5,2)
start_date DATE
end_date DATE
Indexes
product_images
image_id INT
product_id INT
image_url VARCHAR(255)
created_at TIMESTAMP
Indexes

Explanation of Key Constraints:

1. **Foreign Keys**:
    - orders, order_items, payments, product_reviews, and order_files reference other tables such as customers, products, and orders.
2. **Unique Keys**:
    - discounts table has a unique discount_code.
    - categories table has a unique category_name.
3. **Default Values**:
    - order_files table has a default file_type of 'invoice' for all file records.
4. **Generated Columns**:
    - created_at in tables like order_files is automatically generated with the current timestamp.

MongoDB Schema for Product Reviews

Id	Field Name	Datatype	Default Value	Additional Fields
1	customer_email	String	-	required: true
2	product_title	String	-	required: true
3	review	String	-	required: true, minlength: 10, maxlength: 500
4	rating	Number	-	required: true, min: 1, max: 5
5	review_date	Date	Date.now()	-
6	file_path	String	null	-
7	file_type	String	null	-
8	tags	[String]	-	-

API Endpoints Table

Sr.No	API Title	API Endpoint	Sample Input	Sample Output	Status Code
1	Flask nodejs signin Admin	POST/signin	{“email”:”admin@admin.com, “password”:”123456”}	{“message”:”success”, “credentials’:{“id”:1,”email”:admin@admin.com,” token”:”lkajfdslj3os”}	200
2	Flask validate token	POST/auth/validate	{“token”:” lkajfdslj3os”}	{“message”:”success”}	200
3	Flask Create Order	POST /orders	{ "customer_id": "60e8e72b85a59e0d1f8f8b0f", "shipping_address": "123 Elm St" }	{ "message": "Order created successfully", "order_id": 1 }	201
4	nodejs Create Product Review	POST /reviews	{ "customer_id": "60e8e72b85a59e0d1f8f8b0f", "product_id": "60e8e72b85a59e0d1f8f8b0d", "review": "Great product!", "rating": 5 }	{ "message": "Review added successfully", "review_id": 1 }	201
5	nodejs Get Product Reviews	GET /reviews/{product_id}	{ "product_id": "60e8e72b85a59e0d1f8f8b0d" }	{ "reviews": [{ "customer_id": "60e8e72b85a59e0d1f8f8b0f", "review": "Great product!", "rating": 5, "review_date": "2024-12-13T10:30:00Z" }] }	200
6	Flask Get Order Details	GET /orders/{order_id}	{ "order_id": 1 }	{ "order_id": 1, "customer_id": "60e8e72b85a59e0d1f8f8b0f", "shipping_address": "123 Elm St", "order_status": "Processing", "items": [{ "product_id": 1, "quantity": 1 }] }	200
7	Flask Get Customer Orders	GET /customers/{customer_id}/orders	{ "customer_id": "60e8e72b85a59e0d1f8f8b0f" }	{ "orders": [{ "order_id": 1, "shipping_address": "123 Elm St", "order_status": "Processing" }] }	200

Sr.No	API Title	API Endpoint	Sample Input	Sample Output	Status Code
8	Flask Get Product Details	GET /products/{product_id}	{ "product_id": "60e8e72b85a59e0d1f8f8b0d" }	{ "product_id": 1, "product_name": "Smartphone", "category": "Electronics", "price": 699.99, "stock_quantity": 50 }	200
9	Flask Update Order Status	PUT /orders/{order_id}/status	{ "order_status": "Shipped" }	{ "message": "Order status updated successfully", "order_id": 1, "order_status": "Shipped" }	200
10	nodejs Delete Product Review	DELETE /reviews/{review_id}	{ "review_id": "60e8e72b85a59e0d1f8f8b0d" }	{ "message": "Review deleted successfully" }	200
11	Flask Delete Product	DELETE /products/{product_id}	{ "product_id": "60e8e72b85a59e0d1f8f8b0d" }	{ "message": "Product deleted successfully" }	200
12	Flask Get Payment Details	GET /payments/{payment_id}	{ "payment_id": "1" }	{ "payment_id": 1, "order_id": 1, "amount": 699.99, "payment_method": "Credit Card", "payment_date": "2024-12-13T10:30:00Z" }	200

Authentication using NodeJS vs flask

nodejs **flask**