**E-Commerce System Documentation**

**Introduction**

The E-Commerce System is a comprehensive web application developed to facilitate online shopping activities. It comprises three primary services: Authentication, Product Management, and Order Management. These services are built using Flask, a lightweight Python web framework, and MongoDB, a NoSQL database, to provide a scalable, secure, and efficient solution.

**Architecture**

The system architecture follows a microservices approach, where each service is responsible for a specific aspect of the application. This allows for modular development, easier maintenance, and scalability. The architecture consists of the following components:

* Authentication Service: Handles user registration and login.
* Product Service: Manages product creation, retrieval, and updates.
* Order Service: Facilitates order creation, retrieval, and status updates.

Additionally, each service includes a set of routes for handling HTTP requests and a corresponding implementation to process these requests.

**Authentication Service**

**Routes**

* **/register (POST)**: Registers a new user with a username and password.
* **/login (POST)**: Logs in an existing user with username and password.

**Implementation Details**

* Utilizes Flask for routing and request handling.
* Stores user data in MongoDB for persistence.
* Implements password hashing using the hashlib library to enhance security.

**Product Service**

**Routes**

* **/product/create (POST)**: Creates a new product with a name, price, and quantity.
* **/products/search (GET)**: Searches for products by name.
* **/product/<product\_id> (GET)**: Retrieves product details by ID.
* **/product/update/<product\_id> (PUT)**: Updates product details by ID.
* **/products (GET)**: Retrieves a list of all products.

**Implementation Details**

* Utilizes Flask for HTTP routing and request handling.
* Stores product data in MongoDB for persistence.
* Implements text indexing to enable efficient product searching.
* Supports product versioning to prevent conflicts during updates.

**Order Service**

**Routes**

* **/order/create (POST)**: Creates a new order for a product.
* **/order/<order\_id> (GET)**: Retrieves order details by ID.
* **/order/update/<order\_id> (PUT)**: Updates order status by ID.
* **/orders (GET)**: Retrieves a list of all orders.

**Implementation Details:**

* Flask is used for HTTP request routing and handling.
* Order information is stored in MongoDB.
* Upon order creation, the product quantity is deducted from the inventory (assuming quantity tracking).
* Order status updates are supported (e.g., pending, shipped, delivered).

**Testing**

The system employs a comprehensive testing strategy using:

* **pytest:** A popular Python framework for writing and running unit tests.
* **pytest-flask:** Provides fixtures specifically designed for testing Flask applications.
* **pymongo:** Offers functionalities to interact with MongoDB during testing database operations.

Test cases cover various functionalities of each service, ensuring proper user registration, login, product management, and order processing logic.

**Challenges Faced**

**1. Password Security:**

Ensuring secure password storage and robust authentication mechanisms is crucial. Password hashing algorithms are implemented to securely store user credentials.

**2. Database Integration:**

Careful integration between Flask and MongoDB is essential for data persistence and manipulation. Proper testing ensures data consistency and integrity.

**3. Testing Complexity:**

Writing comprehensive test suites requires addressing potential dependencies between tests and achieving thorough coverage. Utilizing techniques like pytest fixtures and modularizing tests can help mitigate this challenge.

**Conclusion**

Through careful planning, implementation, and testing, we have developed a robust e-commerce system capable of handling user authentication, product management, and order processing. The project demonstrates effective utilization of Flask, MongoDB, and various Python libraries to build scalable and secure web applications.