**Requirement Gathering and Analysis Phase**

**Solution Architecture**

|  |  |
| --- | --- |
| Date | 6-07-2024 |
| Team ID | SWTID1719933594 |
| Project Name | Project - SHOPEZ |
| Maximum Marks | 4 |

**Solution Architecture**

**Comparison feature for an E-Commerce Website**

**1. Functional Requirements**

* **User Actions:** Users can add, delete or modify multiple products for comparison.
* **Comparison Attributes:** Users can compare attributes like price, specifications, availability, ratings, and reviews.
* **Functional Buttons:** User should be able to add to cart or buy the products.

**2. Non-Functional Requirements**

* **Performance:** Fast response times for loading comparison results.
* **Scalability:** Able to handle increasing numbers of products and users.
* **Security:** Secure handling of user data and comparisons.
* **User Appeal:** Should be appealing to the user.

**3. System Components**

**Frontend:**

* **Framework:** **React.js** - Provides a robust frontend framework for building dynamic user interfaces.
* **UI Library:** **Material-UI** - Provides ready-to-use UI components for a consistent and responsive design.

**Backend:**

* **Framework:** **Node.js**- Efficient for handling HTTP requests and serving APIs.
* **APIs:** **RESTful APIs** for CRUD operations on products and comparisons.
* **Database:** **MongoDB**- Depending on the structure and complexity of product data. MongoDB is flexible for unstructured data.

**Infrastructure:**

* **Cloud Platform:** **AWS** or **Google Cloud Platform (GCP)** - Provides scalability, reliability, and managed services.
* **Database Hosting: MongoDB Atlas** - Managed database services for scalability and reliability.

**4. Architecture Diagram**

* **User Interface (UI):** React.js frontend with Material-UI components.
* **Application Layer:** Node.js with Express serving RESTful APIs.
* **Data Layer:** MongoDB for storing product data and comparison results.
* **Infrastructure:** Deployed on AWS with Atlas as a database

**5. Data Flow**

* Users select products and initiate comparison in the frontend.
* Frontend sends API requests to backend services.
* Backend retrieves product data from the database.
* Backend processes comparison logic and returns results to the frontend.

**6. Security Considerations**

* **Data Encryption:** HTTPS for secure data transmission.
* **Input Validation:** Validate user inputs to prevent injection attacks.

**7. Scalability and Performance**

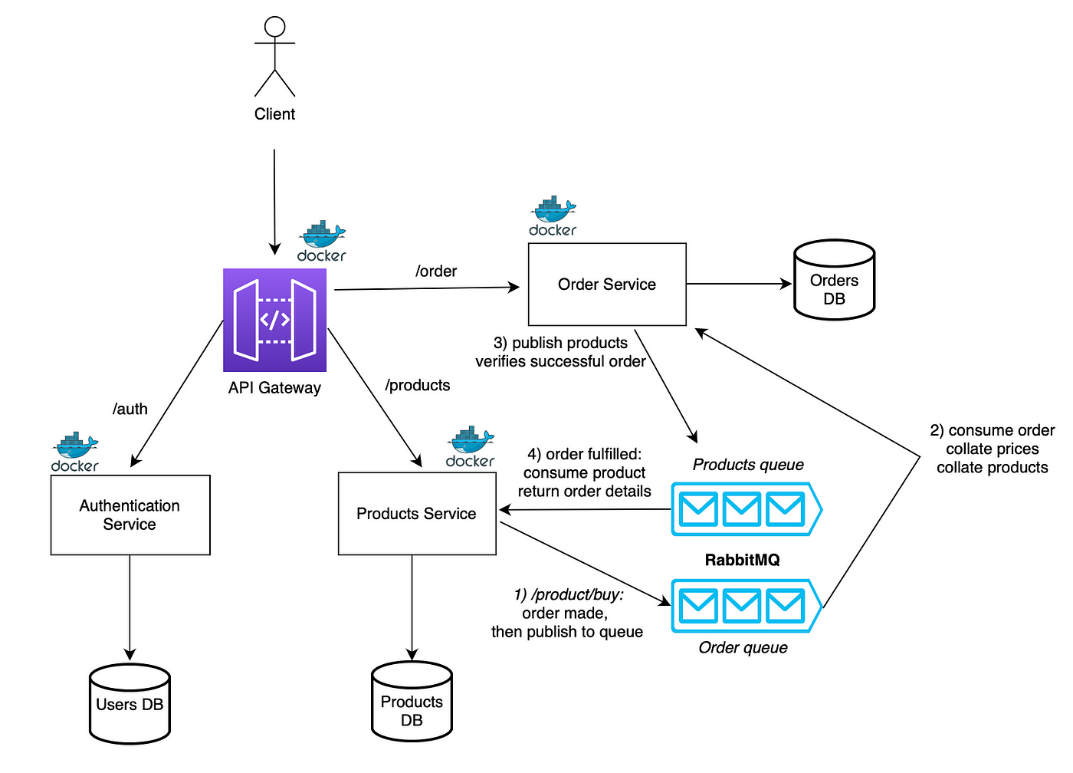
* **Horizontal Scaling:** Autoscaling on AWS EC2 instances based on traffic.
* **Database Scaling:** Vertical scaling with Amazon RDS or MongoDB Atlas.
* **Caching:** **Redis** for caching frequently accessed data and queries.
* **Load Balancing:** **AWS Elastic Load Balancing** for distributing traffic across multiple EC2 instances.

**8. Deployment Strategy**

* **Continuous Integration/Continuous Deployment (CI/CD):** **GitHub Actions** for automated builds and deployments.

**9. Documentation**

* **API Documentation:** **Swagger** or **Postman** for documenting API endpoints and usage.
* **System Architecture:** Detailed architecture diagrams and component descriptions.



A diagram of a customer service

Description automatically generated with medium confidence