Title: Integration of MongoDB, Redis, and Neo4j for E-commerce Data Management

Project Overview: In this project, we aim to integrate three types of NoSQL databases: MongoDB (Document Model), Redis (Key-Value Store), and Neo4j (Graph Database), into a unified e-commerce platform to manage user and product data efficiently. Each database type has been chosen for a specific purpose to enhance the performance, scalability, and ease of querying.

Team Members:

- Member 1: Sravan Kumar Mudireddy
- Member 2: Vaishnavi Rajasekaran

Database Usage:

1. **MongoDB (Document Model):**
   - ➢ MongoDB stores user and product information in a flexible document format.
   - ➢ MongoDB is used to store detailed information such as user profiles, product details, etc.
2. **Redis (Key-Value Store):**
   - ➢ Redis is used as a caching layer for faster access to frequently requested user data.
   - ➢ The user data is stored in Redis using a unique key and can be quickly retrieved when needed.
3. **Neo4j (Graph Database):**
   - ➢ Neo4j is used to store and manage relationships between users and products.
   - ➢ For example, users can "view" products, and these relationships are stored in Neo4j as graph nodes and edges.

Use Case: In this e-commerce use case, we simulate a user browsing through products. The system records user details in MongoDB, caches the user data in Redis for fast access, and stores user-product interaction data in Neo4j as graph relationships.

Data Example:

- **Users:**
  1. Alice (user ID: u1)
  2. Bob (user ID: u2)
  3. Sravan (user ID: u3)
  4. Eve (user ID: u4)
  5. John (user ID: u5)

- **Products:**
    1. Laptop (product ID: 1)
    2. Phone (product ID: 2)
    3. Headphones (product ID: 3)
    4. Smartwatch (product ID: 4)
    5. Keyboard (product ID: 5)

Queries and Operations:

- **MongoDB Queries:** Insert user and product data into MongoDB, query products and users.
- **Redis Operations:** Cache user data for faster retrieval.
- **Neo4j Queries:** Create nodes for users and products, establish relationships (e.g., user views product).

Challenges:

- Connecting and managing different types of NoSQL databases together.
- Handling data consistency between the databases.
- Ensuring efficient retrieval of data with Redis caching.

Conclusion: This project demonstrates the power of combining multiple NoSQL databases to handle large-scale data for an e-commerce platform. MongoDB provides flexible document storage, Redis offers fast caching, and Neo4j helps with complex relationships between users and products.