**Cloud Architecture Recommendation Engine**

**Spring Boot Java Back-End Application**

**Objective: Construct a service that recommends cloud architecture based on application category (e.g., e-commerce, video streaming).**

**Database**

**- Created a MySQL database named "recommendation\_db1" using Workbench.**

**- Created a table named "recommendations" with columns: id, product\_name, requirement, reason, r\_memory, r\_storage, bandwidth, scalability, reliability, latency.**

**Spring Boot Application**

**- Created a Spring Boot application with the following dependencies: MySQL Driver, Spring Web, Lombok, and Spring Data JPA.**

**- Implemented the entity class "RecomendEntity" representing the "recommendations" table structure.**

**- Implemented the repository interface "RecomendRepo" for database operations.**

**- Implemented the service class "RecomendServices" to process user input and provide recommendations based on specified requirements.**

**- Implemented the controller class "RecomendController" to handle REST API requests and interact with the service.**

**- Created the "UserInput" class to model user input for cloud architecture requirements.**

**MySQL Database Configuration**

**- Configured application.properties file to connect to the MySQL database.**

**- Defined the database URL, username, password, Hibernate dialect, and other properties.**

**Execution Flow**

**- User input is sent to the "/recommendation" endpoint as a POST request.**

**- The input is processed by the "RecomendServices" class, which determines the suitable cloud architecture recommendation based on the specified requirements.**

**- The recommendation is saved in the database through the "RecomendRepo" interface.**

**- The response containing the recommendation is returned to the user via the REST API.**

**Conclusion**

**The Cloud Architecture Recommendation Engine Leverages Spring Boot, MySQL, and RESTful APIs to provide personalized cloud architecture recommendations based on specific application requirements.**

**- The modular design allows easy scalability and maintenance of the recommendation engine.**

**- Further enhancements can be made to improve the recommendation algorithm and integrate additional cloud architecture options.**

**Note: this report provides an overview of the key components and functionality of the Cloud Architecture Recommendation Engine. Additional details and code snippets are available in the provided code files.**

**Development Process**

**Login to AWS Console:**

**I accessed the AWS Console using my credentials to start the development process.**

**Set up MySQL database in AWS Cloud:**

**I Created an RDS instance in AWS, choosing MySQL as the database engine, Configured instance size, storage, security groups, and database credentials.**

**Connect to MySQL database with MySQL Workbench:**

**I Installed MySQL Workbench on my local machine, obtain the RDS instance's endpoint and port from the AWS Console, Use MySQL Workbench to establish a connection to the RDS instance using the endpoint, port, and database credentials.**

**Clone Spring Boot application source code:**

**I Cloned the Spring Boot application source code from the repository using a version control tool.**

**update application.properties file:**

**I Modified the application.properties file in the Spring Boot application to change the database properties, update the database connection details with the RDS instance's endpoint, port, and credentials.**

**Package Spring Boot application as a JAR file:**

**Build the Spring Boot application and package it as a JAR file using the appropriate build tools (e.g., Maven or Gradle).**

**Deploy Spring Boot JAR file on AWS using Elastic Beanstalk:**

**I had a problem with creating Elastic Beanstalk, and it was the last step for testing it successfully.**

**Ayman Sufian Jalal Musa …**

**Best Regards …**