gRPC-Gradle-MongoDB Application

Abishek Arumugam Thiruselvi(40218896)
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
abishekarumugam@icloud.com

Srinidhi Honakere Srinivas(40221128)
Department of Electrical and Computer Engineering
Concordia University
Montreal, Canada
srinidhihs007@gmail.com

***Abstract*— This Assignment Aims to improve our knowledge of RPC communication, MongoDB data operations, and data processing.**

Keywords—Gradle, Spring Framework, gRPC and MongoDB.

I. INTRODUCTION

This main task is to improve establishment of RPC communication, MongoDB data operations, and data processing aggregate pipelines. We'll be using data from a Kaggle dataset that details undergraduate tuition and fees. To design a distributed system architecture that can connect and function effectively, tasks include building a MongoDB collection, generating data access objects, specifying protobuf files, developing gRPC services, and putting the gRPC client and server into use.

II. APPLICATION DESIGN

The assignment is done using Gradle instead of maven as build tool and instead of pom.xml we can find dependencies in build. Gradle

Fig. 1.1. build. Gradle

The Application has source folder src/main/java and resource folder src/test/resources.

A. src/main/java

This source folder has 4 packages as shown in figure Fig. 2.1. The details of the package are described below.

1. *Package: cu. dssassignment2.controller*
   This package has the file "Controller.java". It has the RestController which use to upload the dataset to MongoDB.
2. *Package: cu. dssassignment2.model*
   This package has the file "EduCostStat.java", "EduCostStatOne.java" etc. It is the class object which has all the details about data models.
3. *Package: cu. dssassignment2.utils*
   This package has the file "DssAssignment2Application.java". It is the start of the spring boot application. Here we have started the gRPC server using @Bean annotations.
4. *Package: cu. dssassignment2.repository*
   This package has all the collections interface to perform queries and save in appropriate collections.
5. *Package: cu. dssassignment2.grpc.client*
   This package has client gRPC which is like JUnit test. This sends the requests to the gRPC server and receives the response.
6. *Package: cu. dssassignment2.grpc.server*
   This package has service implementation gRPC which is like REST Controller. Which receives the client requests and sends the response.

B. src/main/proto

1. *eduCostStat.proto*
   This file has all the proto 5 services which is required to generate service gRPC.

C. src/main/resources

The source folder contains application.properties to connect with MongoDB and elastic beanstalk.

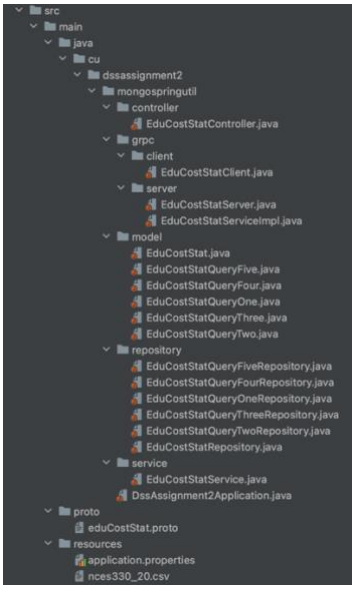

Fig. 2. 1. Project Structure

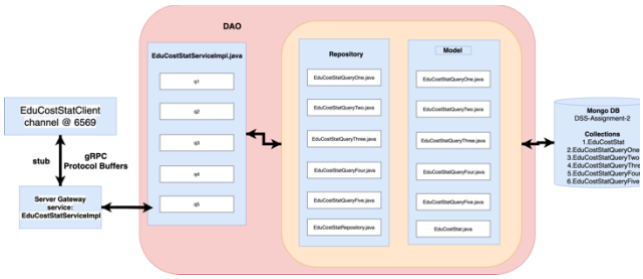

Fig. 2.2. Data Operation Architecture with RPC communication

D. *Mongodb cluster:*

Initially we created a mongodb cluster to which we connected to java application using host link. We used this host link in application properties.

E. *Mongodb Collection:*

We created mongodb collection we wrote a method in edustat controller to insert into collection and also making sure there is duplicate collection is not added when controller runs again.

F. *Models:*

We have created Educoststat, EducoststatQueryOne, EducoststatQueryTwo, EducoststatQueryThree etc. making sure to have getters and setters.

G. *Repositories:*

EducostRepository and all other repository is where all the quer and rules are writing making most part of data access object

The Diagram illustrates, the EduCostStatService which is located on the gRPC server, which is receiving requests from the gRPC client.

The RPCs that the client can call, Query1 to Query 5 which are all included in the EduCostStatService. One or more protobuf messages may be used by each RPC for input and output.

From EducoststatService the data is sent / received from mongodb through repository and model .

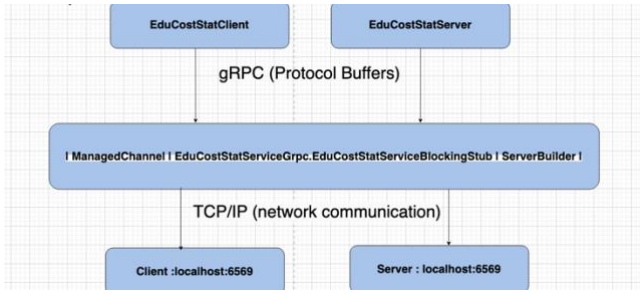

Fig. 2.3. Data Operation Architecture with RPC communication

III. TASKS

A. *Task 1.1*

Created MongoDB collection named EduCostStat to store the data to the MongoDB instance running on MongoDB online cluster.

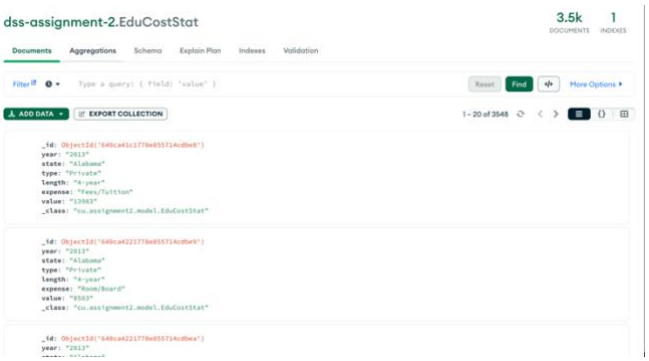

Fig. 3.1. EduCostStat Collection with 3.5K records

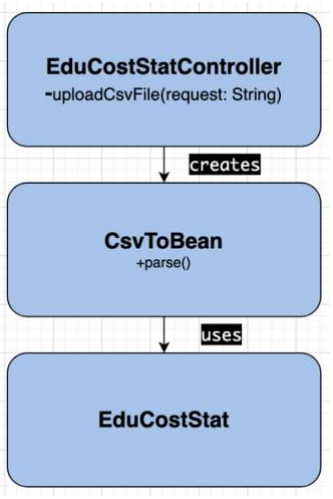

Fig. 3.2. Code Structure to create Mongodb Collection

B. *Task 1.2*

Entails creating Java data access objects to represent various data queries. We calculated the cost using the specified year, state, kind, length, and expense; we then saved the query as a document in the EduCostStatQueryOne collection.

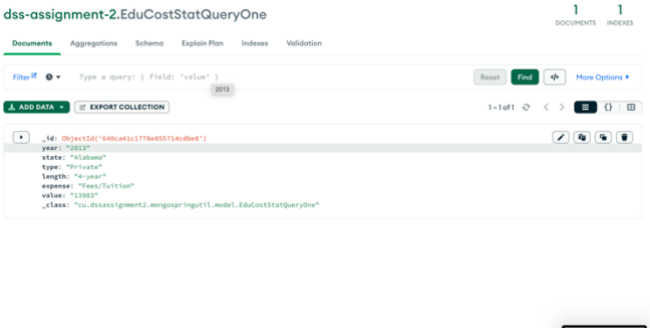

Fig. 3.3. Output EduCostStatQueryOne

- The following task was to query the top 5 cost-prohibitive states (with overall expense) given a year, kind, and length, and record the query as a document in a collection called EduCostStatQueryTwo.

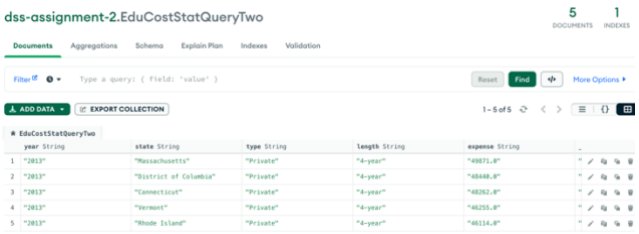

Fig. 3.4. Output EduCostStatQueryTwo

- The following task was to save the query as a document in a collection called EduCostStatQueryThree and query the top 5 most economically prosperous states given a year, type, and length.

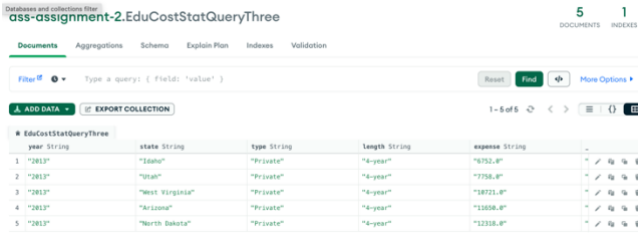

Fig. 3.5. Output EduCostStatQueryThree

- The following task was to query the top 5 states with the highest increase rate of overall expense given a range of prior years, including the most recent year as the base, one year, three years, and five years; and save the query as a document in a collection called EduCostStatQueryFour.

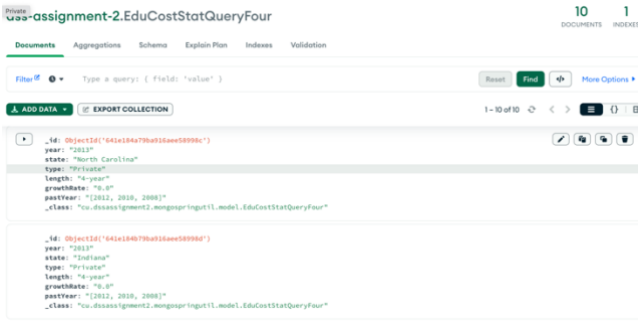

Fig. 3.6. Output EduCostStatQueryFour

- A collection called EduCostStatQueryFive aggregates the region's average overall expense for a specific year, type, and duration.

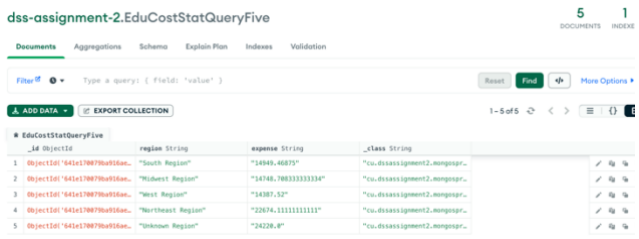

Fig. 3.7. Output EduCostStatQueryFive

C. *Task 2.1*

Defines a Protobuf definition file to represent the request, response and service for reach query in Task 1

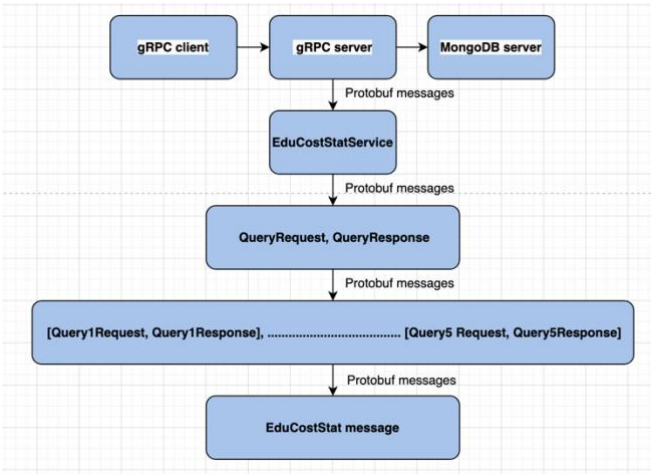

Fig. 3.8. Higher Level Architecture of Task 2.1

This code is a Protocol Buffers file defining a service called "EduCostStatService" with five RPC methods: "test", "Q1", "Q2", "Q3", and "Q4", and one additional RPC method "Q5" that returns a "Query5Response".

It also specifies some options for Java, including the Java package name and to generate multiple Java files. Finally, it imports the empty proto file from the Google Protobuf library.

IV. RESULTS

1. In This Assignment, we connect client and server and use data query to extract insights from data, we developed a distributed system utilizing gRPC.
2. In MongoDB, we successfully carried out several searches and stored data. Cost calculations, state-by-state expense queries, state-by-state expense growth rate identification, and regional expense average aggregation were among the tasks.
3. Ultimately, we were able to archive the results successful.

V. CONCLUSION

In conclusion, the COEN 6731 Winter 2023 Assignment Two has provided us with the opportunity to practice RPC communication, data operations on MongoDB, and implementation of data pipelines.

REFERENCES

[1] https://moodle.concordia.ca/moodle/mod/lti/view.php?id=3433629

[2] https://github.com/abishekat/dss-assignment-2