Department of Electrical and Computer Engineering
Gina Cody School of Engineering and Computer Science

ASSIGNMENT 1
COEN 6731- Distributed Software System
COURSE INSTRUCTOR:
Yan Liu

Submitted by:

Adit Shah
ID- 40172848
Email- adit.shah@mail.concordia.ca

Navaldeep Sandhu
ID- 40170635
Email- navaldeep.sandhu@mail.concordia.ca

Table of Contents

Objective

The Objective of this assignment is to enhance skills in:
a. RPC communication for resources and services
b. Data operations on MongoDB, a NoSQL database
c. Implementation of aggregation pipelines for data processing

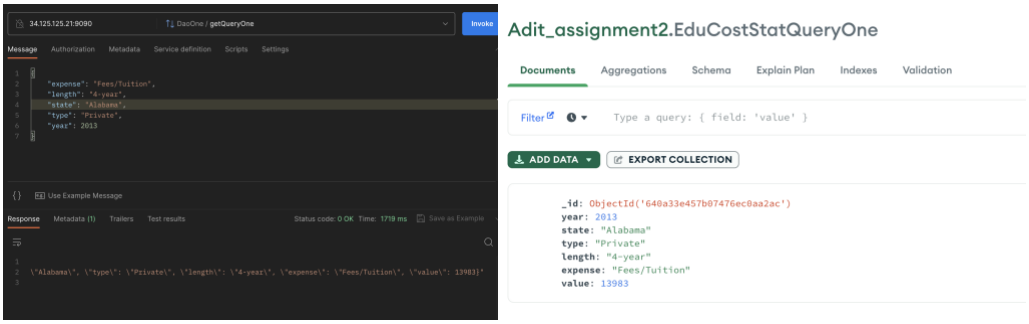
Introduction

The client/server architecture is implemented using Spring. This process involves using Spring initializer and selecting the maven project and Java language, adding dependencies, and generating the framework that stores all dependencies in the pom.xml file. MongoDB collection EduCostStat has been generated utilizing public data source compiled from the National Center of Education Statistics Annual Digest, USA which is used to store data in MongoDB instance that is operating on MongoDB Atlas. Five queries have been written to fetch the data for different requirements Protobuff definition file has been defined to illustrate the request, response, and service for each query. For each service defined, the gRPC service has been established for invoking the corresponding DAO classes. Ultimately, GRPC client-server code has been generated that oversees the RPC calls sent from the client to the server and handles the server's response to each call.

MongoDB queries

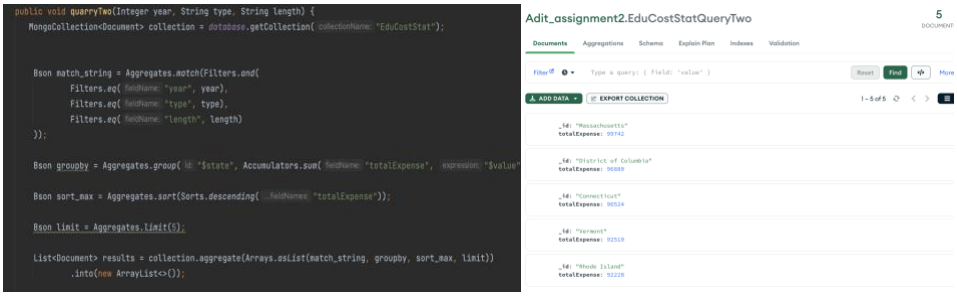
Quarry 1

a. Query the cost given specific year, state, type, length, expense; and save the query as a document in a collection named EduCostStatQueryOne.



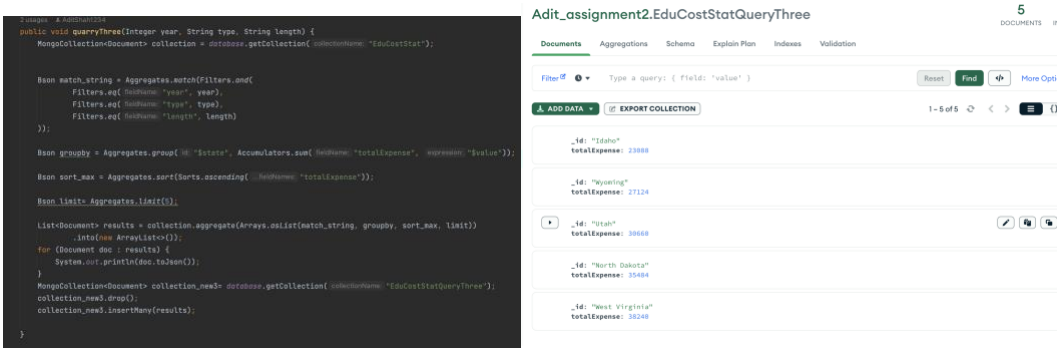
Quarry 2

b. Query the top 5 most expensive states (with overall expense) given a year, type, length; and save the query as a document in a collection named EduCostStatQueryTwo.



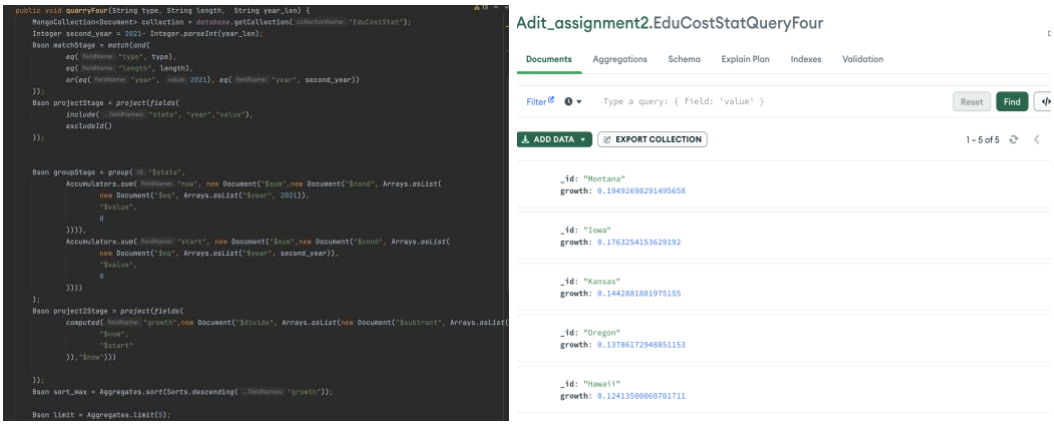
Quarry 3

c. Query the top 5 most economic states (with overall expense) given a year, type, length; save the query as a document in a collection named EduCostStatQueryThree.



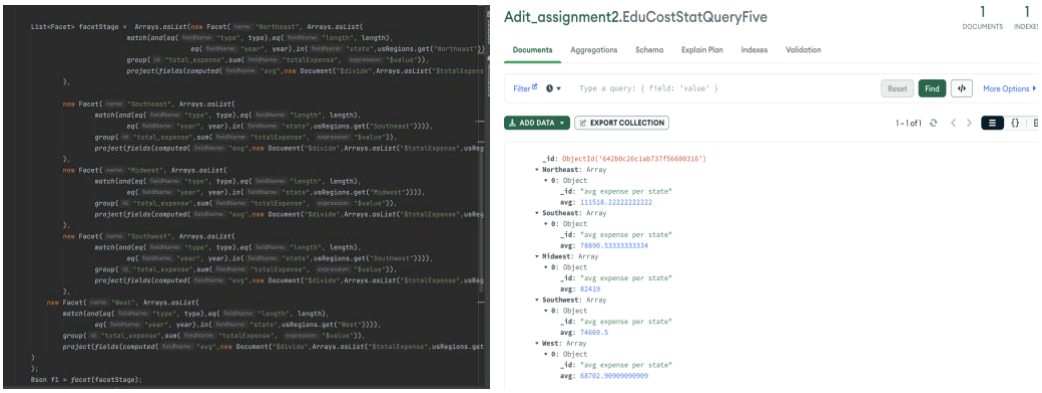
Quarry 4

d. Query the top 5 states of the highest growth rate of overall expense given a range of past years, one year, three years and five years (using the latest year as the base) , type and length; and save the query as a document in a collection named EduCostStatQueryFour.



Quarry 5

e. Aggregate region‘s average overall expense for a given year, type and length; and save the query



Architectural diagram

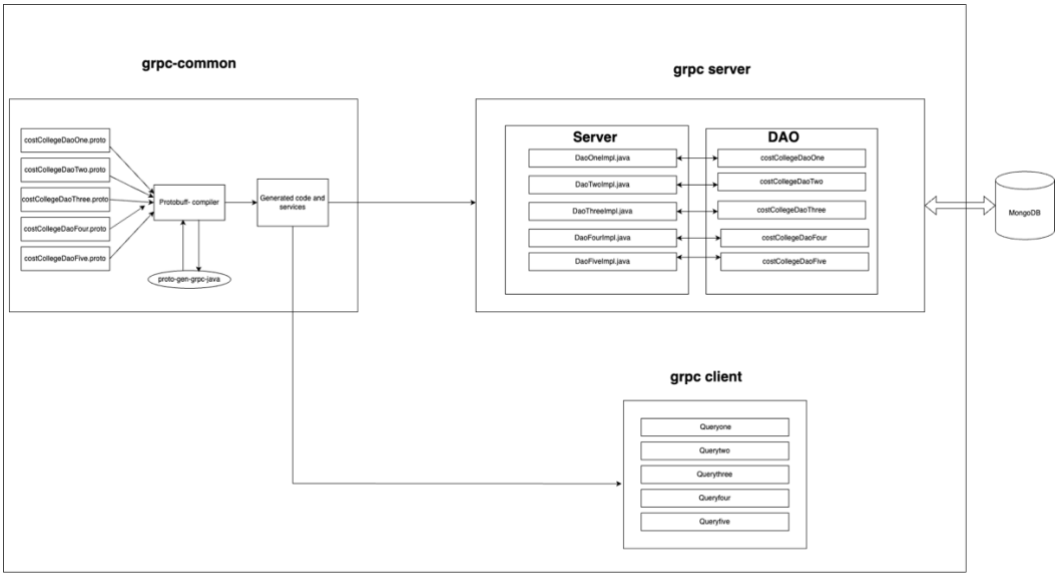


Figure 1 Architecture Diagram

The architecture diagram above shows the structure and implementation of the required system. The whole structure has 4 major components:

a. gRPC Common
b. gRPC Service
c. gRPC Client

a. **gRPC Common:** gRPC Common is one of the modules of the gRPC system which is used as an interface. It contains all the **Protobuff** files for the 5 queries. We defined request, response, and services for each query under proto files for the respective query. After the compilation of proto files and code for request, the response is auto-generated under the target of the grpc-common package.

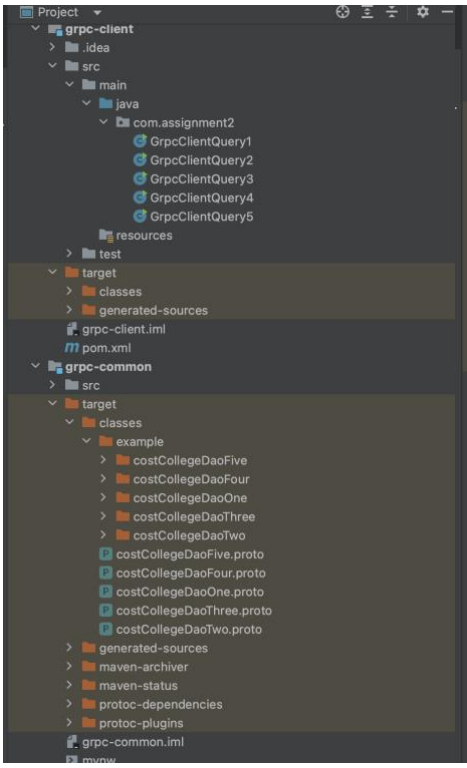


Figure 2 Client Code Structure

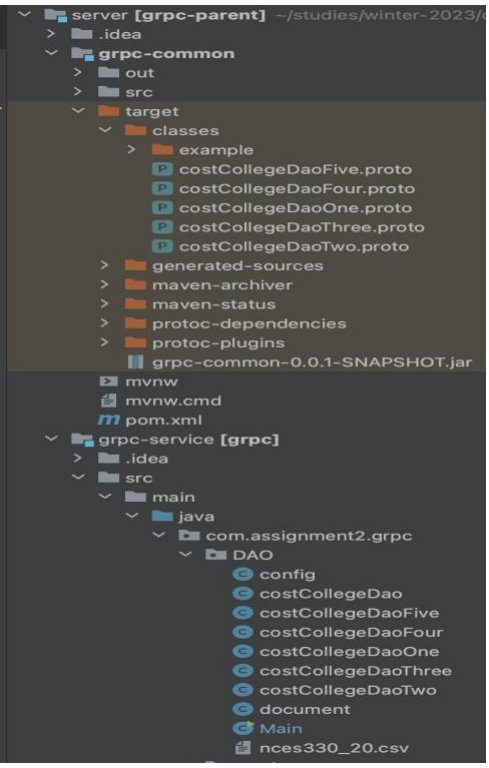


Figure 3Server Code Structure

b. **gRPC Service:** The second module under the gRPC system is the gRPC service which contains **DAO** services as well as **service implementation** files. On compilation of proto files code for request, response, and service is generated and for each query gRPC class is generated which contains the base class for services to implement servers. The service class created is made operational by overriding the service base class generated and by running the gRPC server run to listen to requests from the server and provide service responses in return. There are 5 DAO files and each file contains the java objects for different queries to the data.

c. **gRPC Client**: gRPC client module contains the client code for each query and it calls the required service directly. The code contains the stub classes that clients use to communicate to the server. It has all the java code for all the queries required.

Deployment

The code is deployed on the GCP server as a dockized application.
Public IP: 34.125.125.21

PORT: 9090
GitHub : https://github.com/AditShah1234/distributed_ass2

Summary

The requirement of executing RPC communication for resources and services, implementing aggregation pipeline and conducting data operation on mongoDB has been fulfilled. A protocol buff file has been created for every query to depict its request, response, and service. Upon construction, this file is transformed into a Java file. gRPC client has been created for each query which sends RPC request to server using stub and hence server responds to the client's request.

References

- https://github.com/AditShah1234/distributed_ass2
- https://grpc.io/docs/languages/java/basics/
- https://www.youtube.com/watch?v=2CWYorTWyGs&t=1560s&ab_channel=TechPrimers
- https://adityagoel123.medium.com/developing-microservices-apis-using-grpc-6ee97805861