# Distributed Software Systems

Assignment 2

Instructor: Yan Liu

Saman Javidnia

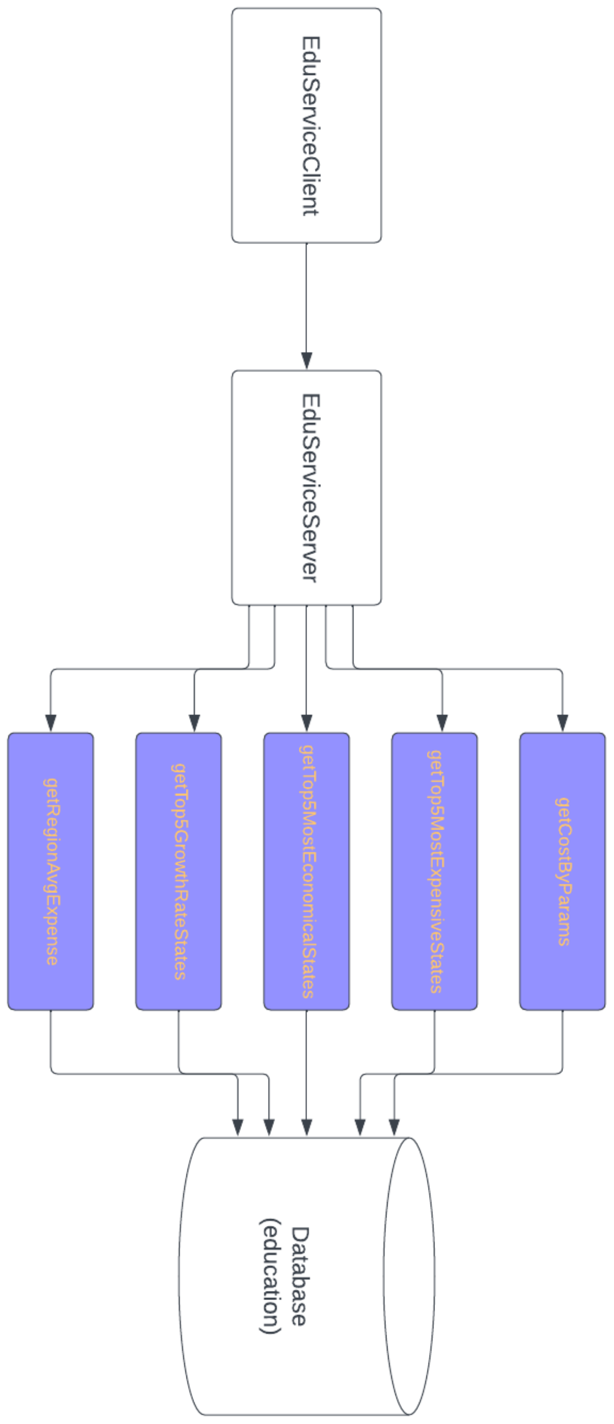
## Abstract

The growing demand for easy and efficient access to educational data has necessitated the development of resilient and scalable data storing and search systems. This report describes the design and implementation of a system that uses RPC communication, the MongoDB NoSQL database, and data aggregation pipelines to provide insightful statistics on average undergraduate tuition and fees, as well as room and board rates, for full-time students at degree-granting postsecondary institutions across the United States. The implemented system shows contemporary data storing and transmission technologies' powers in providing efficient, dependable, and scalable solutions for handling and analyzing big datasets.

## Introduction

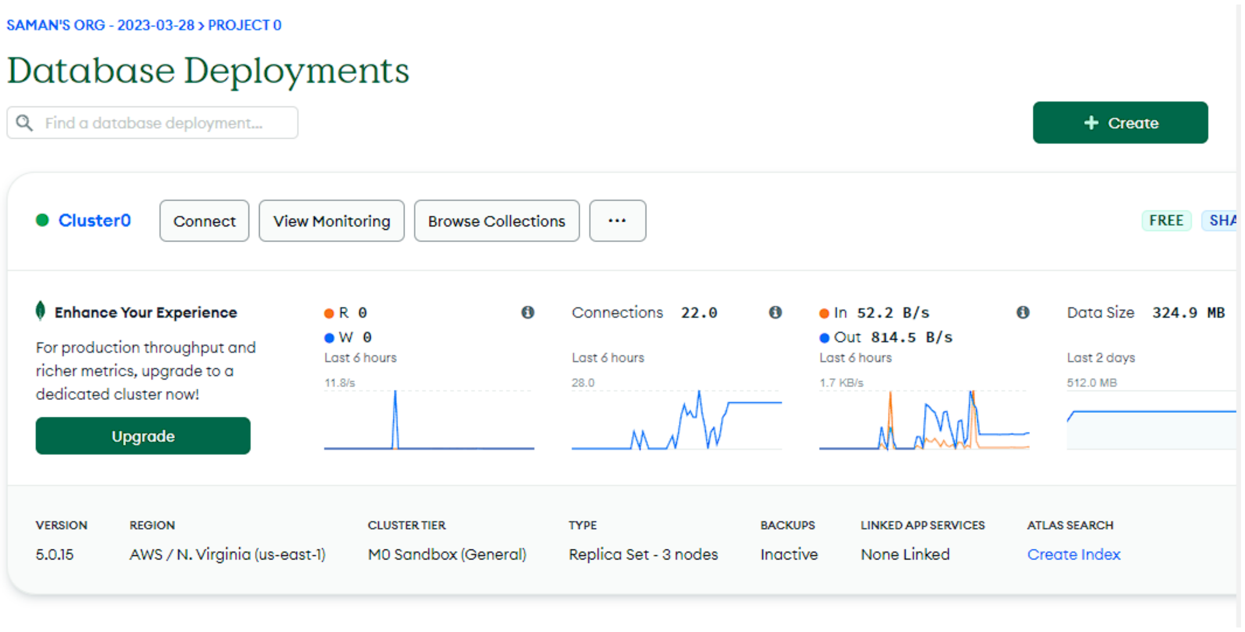
The National Center for Education Statistics (NCES) gathers and disseminates information about education in the United States, such as tuition and room and board prices. With the growing amount of data produced and the need for effective analysis, a system that can effectively store, manage, and process this data is required.

This report describes the design and execution of a system that stores education-related data gathered by the NCES using MongoDB, a NoSQL database. To conduct searches and derive useful insights from data, the system employs data access objects and aggregation processes. Furthermore, the system makes use of gRPC, a high-performance RPC framework, to enable contact between clients and the server, enabling efficient and scalable data access.


EduServiceClient
EduServiceServer
getRegionAvgExpense
getTop5GrowthRateStates
getTop5MostEconomicalStates
getTop5MostExpensiveStates
getCostByParams
Database
(education)

# MongoDB

In the first step I created a MongoDB database and collection



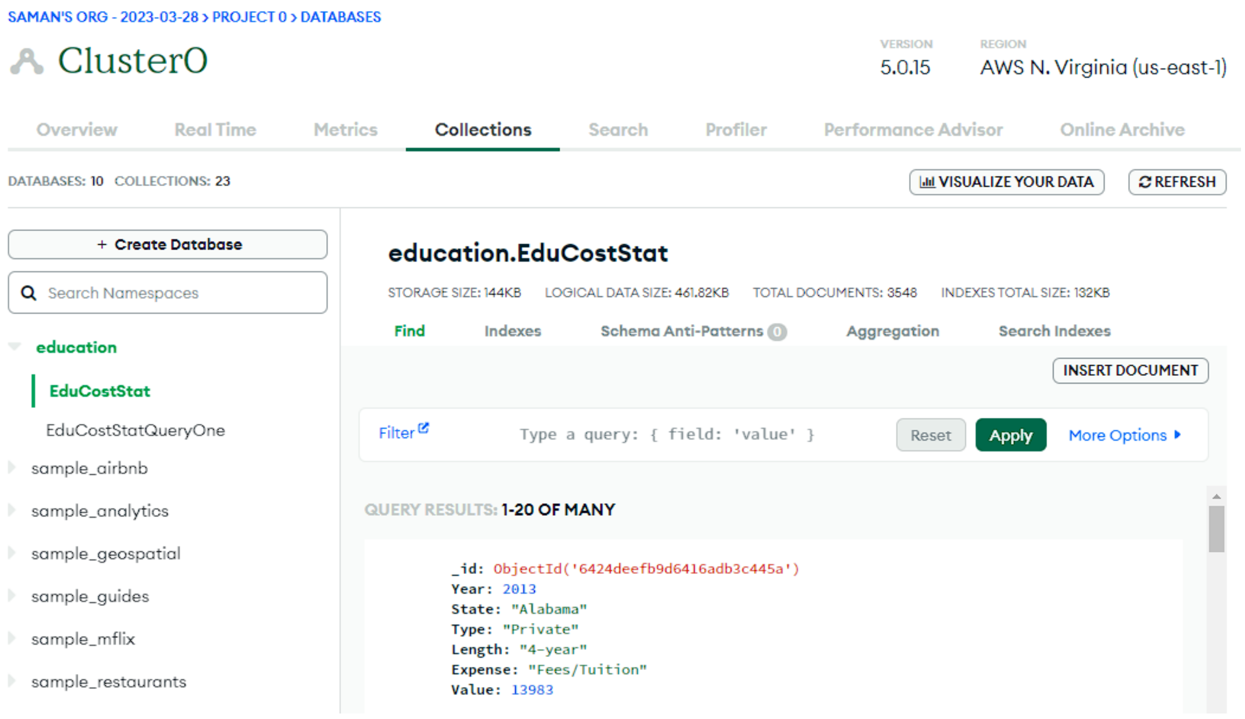
SAMAN'S ORG - 2023-03-28 > PROJECT 0

Database Deployments

For production throughput and richer metrics, upgrade to a dedicated cluster now!

```
R 0
W 0
Last 6 hours
11.8/s

Connections 22.0
Last 6 hours
28.0

In 52.2 B/s
Out 814.5 B/s
Last 6 hours
1.7 KB/s

Data Size 324.9 MB
Last 2 days
512.0 MB
```

| VERSION | REGION | CLUSTER TIER | TYPE | BACKUPS | LINKED APP SERVICES | ATLAS SEARCH |
|---|---|---|---|---|---|---|
| 5.0.15 | AWS / N. Virginia (us-east-1) | M0 Sandbox (General) | Replica Set - 3 nodes | Inactive | None Linked | Create Index |



SAMAN'S ORG - 2023-03-28 > PROJECT 0 > DATABASES

Cluster0

VERSION 5.0.15 REGION AWS N. Virginia (us-east-1)

DATABASES: 10 COLLECTIONS: 23

- education
  - EduCostStat
  - EduCostStatQueryOne
- sample_airbnb
- sample_analytics
- sample_geospatial
- sample_guides
- sample_mflix
- sample_restaurants

education.EduCostStat

STORAGE SIZE: 144KB LOGICAL DATA SIZE: 461.82KB TOTAL DOCUMENTS: 3548 INDEXES TOTAL SIZE: 132KB

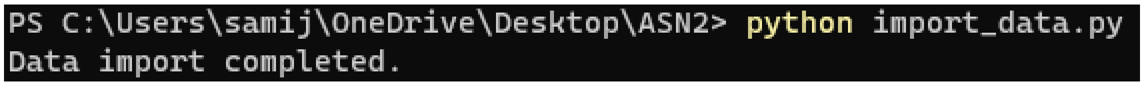
```
Type a query: { field: 'value' }
```

QUERY RESULTS: 1-20 OF MANY

```
_id: ObjectId('6424deefb9d6416adb3c445a')
Year: 2013
State: "Alabama"
Type: "Private"
Length: "4-year"
Expense: "Fees/Tuition"
Value: 13983
```

A database named education and a collection named EduCostStat

Then I wrote a program in python to fill the collection by reading the data from .csv file provided

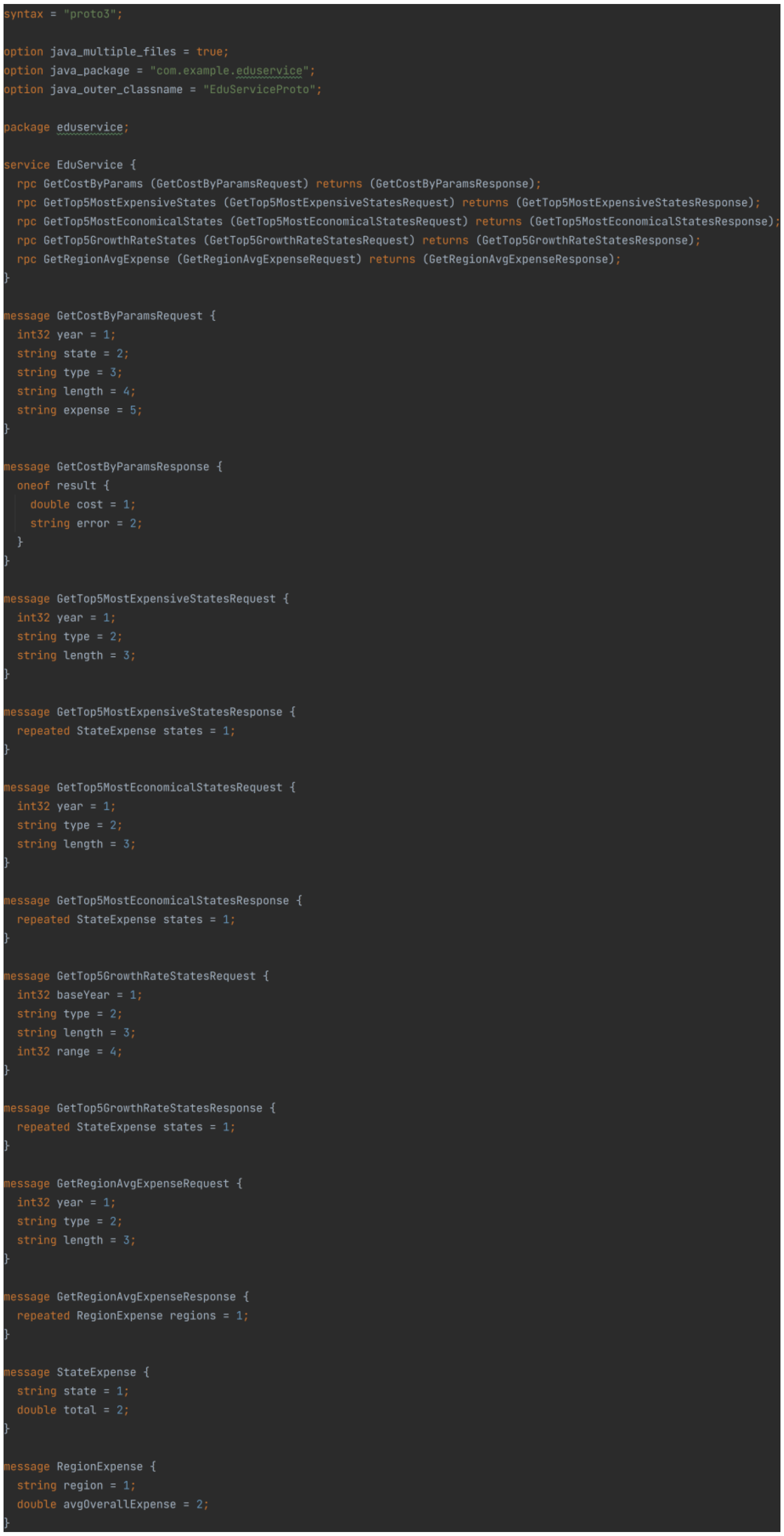


```
PS C:\Users\samij\OneDrive\Desktop\ASN2> python import_data.py
Data import completed.
```

Then a new class for making queries named of EduCostStatQueries which makes the all 5 queries to the database

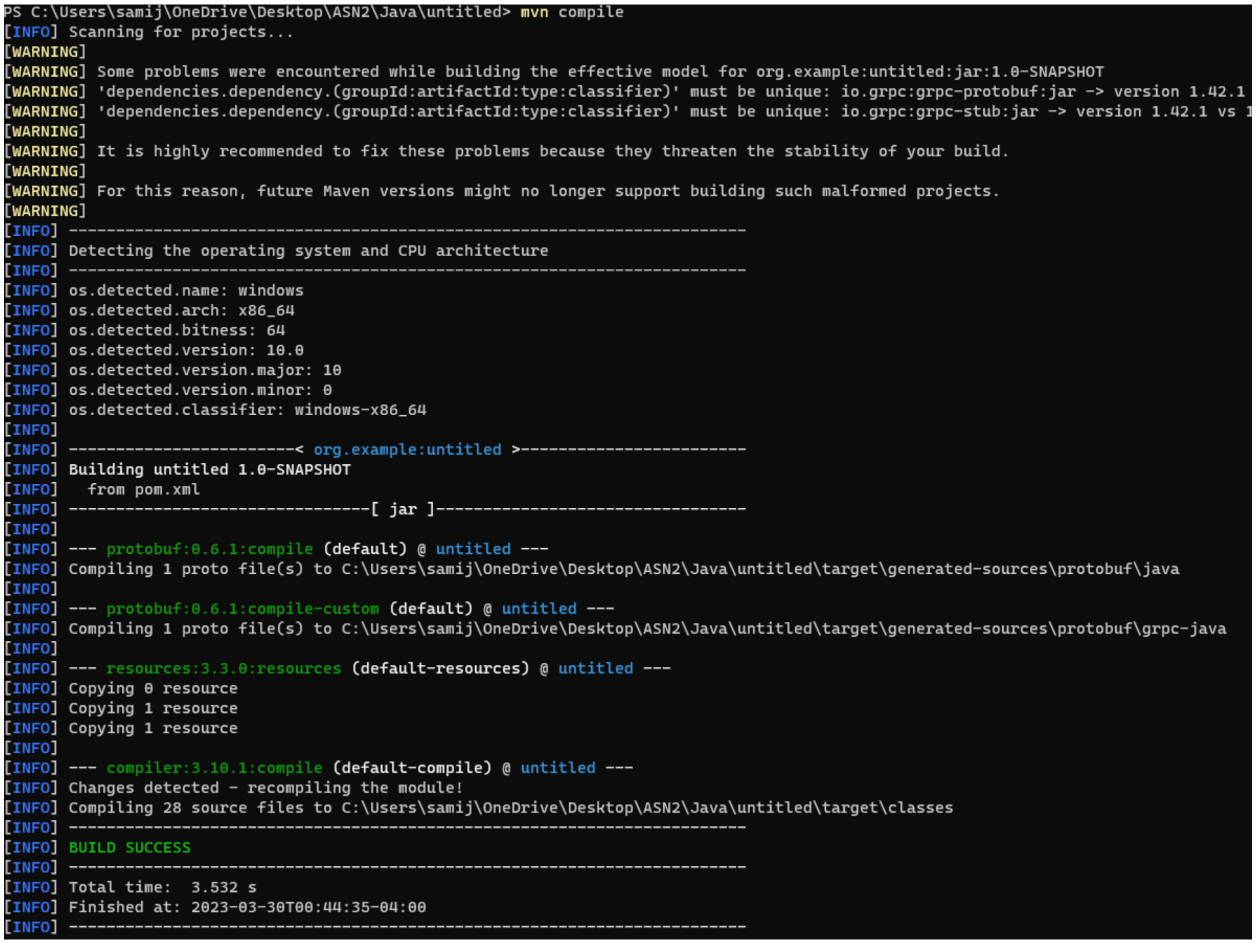
For the task 2 first we need a protobuff file to generate the files we need

I wrote it in edu_service.proto



```proto
syntax = "proto3";

option java_multiple_files = true;
option java_package = "com.example.eduservice";
option java_outer_classname = "EduServiceProto";

package eduservice;

service EduService {
  rpc GetCostByParams (GetCostByParamsRequest) returns (GetCostByParamsResponse);
  rpc GetTop5MostExpensiveStates (GetTop5MostExpensiveStatesRequest) returns (GetTop5MostExpensiveStatesResponse);
  rpc GetTop5MostEconomicalStates (GetTop5MostEconomicalStatesRequest) returns (GetTop5MostEconomicalStatesResponse);
  rpc GetTop5GrowthRateStates (GetTop5GrowthRateStatesRequest) returns (GetTop5GrowthRateStatesResponse);
  rpc GetRegionAvgExpense (GetRegionAvgExpenseRequest) returns (GetRegionAvgExpenseResponse);
}

message GetCostByParamsRequest {
  int32 year = 1;
  string state = 2;
  string type = 3;
  string length = 4;
  string expense = 5;
}

message GetCostByParamsResponse {
  oneof result {
    double cost = 1;
    string error = 2;
  }
}

message GetTop5MostExpensiveStatesRequest {
  int32 year = 1;
  string type = 2;
  string length = 3;
}

message GetTop5MostExpensiveStatesResponse {
  repeated StateExpense states = 1;
}

message GetTop5MostEconomicalStatesRequest {
  int32 year = 1;
  string type = 2;
  string length = 3;
}

message GetTop5MostEconomicalStatesResponse {
  repeated StateExpense states = 1;
}

message GetTop5GrowthRateStatesRequest {
  int32 baseYear = 1;
  string type = 2;
  string length = 3;
  int32 range = 4;
}

message GetTop5GrowthRateStatesResponse {
  repeated StateExpense states = 1;
}

message GetRegionAvgExpenseRequest {
  int32 year = 1;
  string type = 2;
  string length = 3;
}

message GetRegionAvgExpenseResponse {
  repeated RegionExpense regions = 1;
}

message StateExpense {
  string state = 1;
  double total = 2;
}

message RegionExpense {
  string region = 1;
  double avgOverallExpense = 2;
}
```

Then we compile the code with maven to generate the required files



```
PS C:\Users\samij\OneDrive\Desktop\ASN2\Java\untitled> mvn compile
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for org.example:untitled:jar:1.0-SNAPSHOT
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: io.grpc:grpc-protobuf:jar -> version 1.42.1
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: io.grpc:grpc-stub:jar -> version 1.42.1 vs 1
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO] ------------------------------------------------------------------------
[INFO] Detecting the operating system and CPU architecture
[INFO] ------------------------------------------------------------------------
[INFO] os.detected.name: windows
[INFO] os.detected.arch: x86_64
[INFO] os.detected.bitness: 64
[INFO] os.detected.version: 10.0
[INFO] os.detected.version.major: 10
[INFO] os.detected.version.minor: 0
[INFO] os.detected.classifier: windows-x86_64
[INFO]
[INFO] -----------------------< org.example:untitled >------------------------
[INFO] Building untitled 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- protobuf:0.6.1:compile (default) @ untitled ---
[INFO] Compiling 1 proto file(s) to C:\Users\samij\OneDrive\Desktop\ASN2\Java\untitled\target\generated-sources\protobuf\java
[INFO]
[INFO] --- protobuf:0.6.1:compile-custom (default) @ untitled ---
[INFO] Compiling 1 proto file(s) to C:\Users\samij\OneDrive\Desktop\ASN2\Java\untitled\target\generated-sources\protobuf\grpc-java
[INFO]
[INFO] --- resources:3.3.0:resources (default-resources) @ untitled ---
[INFO] Copying 0 resource
[INFO] Copying 1 resource
[INFO] Copying 1 resource
[INFO]
[INFO] --- compiler:3.10.1:compile (default-compile) @ untitled ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 28 source files to C:\Users\samij\OneDrive\Desktop\ASN2\Java\untitled\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.532 s
[INFO] Finished at: 2023-03-30T00:44:35-04:00
[INFO] ------------------------------------------------------------------------
```

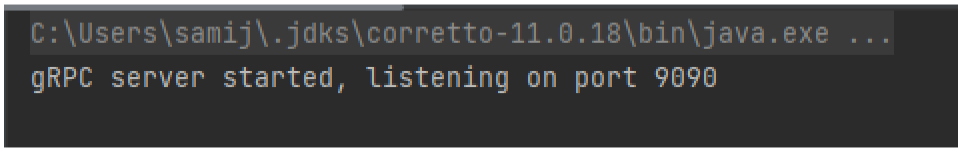
For the Task 2.2 I implemented a new java class named EduServiceImpl to extend EduServiceImplBase

The EduServiceImpl class is a gRPC service implementation class that implements the functions specified in the EduService proto file. Its task is to accept client requests, retrieve the requested data from the database using the appropriate data access object classes, and then deliver the answer to the client. It acts as a connection point between the gRPC client and the data access object layer.

The gRPC service specified in the draft file is implemented as a client by EduServiceClient. It enables a client to link to the gRPC server and send requests to the RPC methods that have been developed. Using the generated stub methods supplied by gRPC, the client makes queries and gets replies from the server. EduServiceClient's primary purpose is to allow the client to interact with the EduService gRPC server and utilize its functionality.

The EduServiceServer is in charge of setting up and running the gRPC server, which waits for inbound client queries. It starts the EduServiceImpl and connects it to the server. When a client makes a request, the server calls the proper function on the EduServiceImpl to process it and return a response to the client.

And finally The program's starting point is Main.java. It starts and initializes the gRPC server and generates a gRPC client to communicate with it. It also contains a basic command line interface that allows the user to enter questions and receive responses from the server.



```
C:\Users\samij\.jdks\corretto-11.0.18\bin\java.exe ...
gRPC server started, listening on port 9090
```