Git Hub url - https://github.com/SHIVAKUMARGOWDAHN222/student-fee-management-microservice.git

Prerequisite

- Intellij IDE
- Docker Desktop
- Java 17

Steps

Clone the project in local and then import all the microservices as modules inside base container called “school platform project”.

Navigate to Project Folder where Docker-compose file is present and Run Docker-compose up command. This will start the Kafka server and zookeeper that is being used for event driven architecture.

Navigate to the pom.xml file located in folder “schoolplatform” and run “mvn clean install” command to build the project

Start Individual Microservices. Eureka Server needs to be started first.

There are 5 microservices

- API Gateway
- Eureka Server
- Fee Process
- Receipt
- Student Management

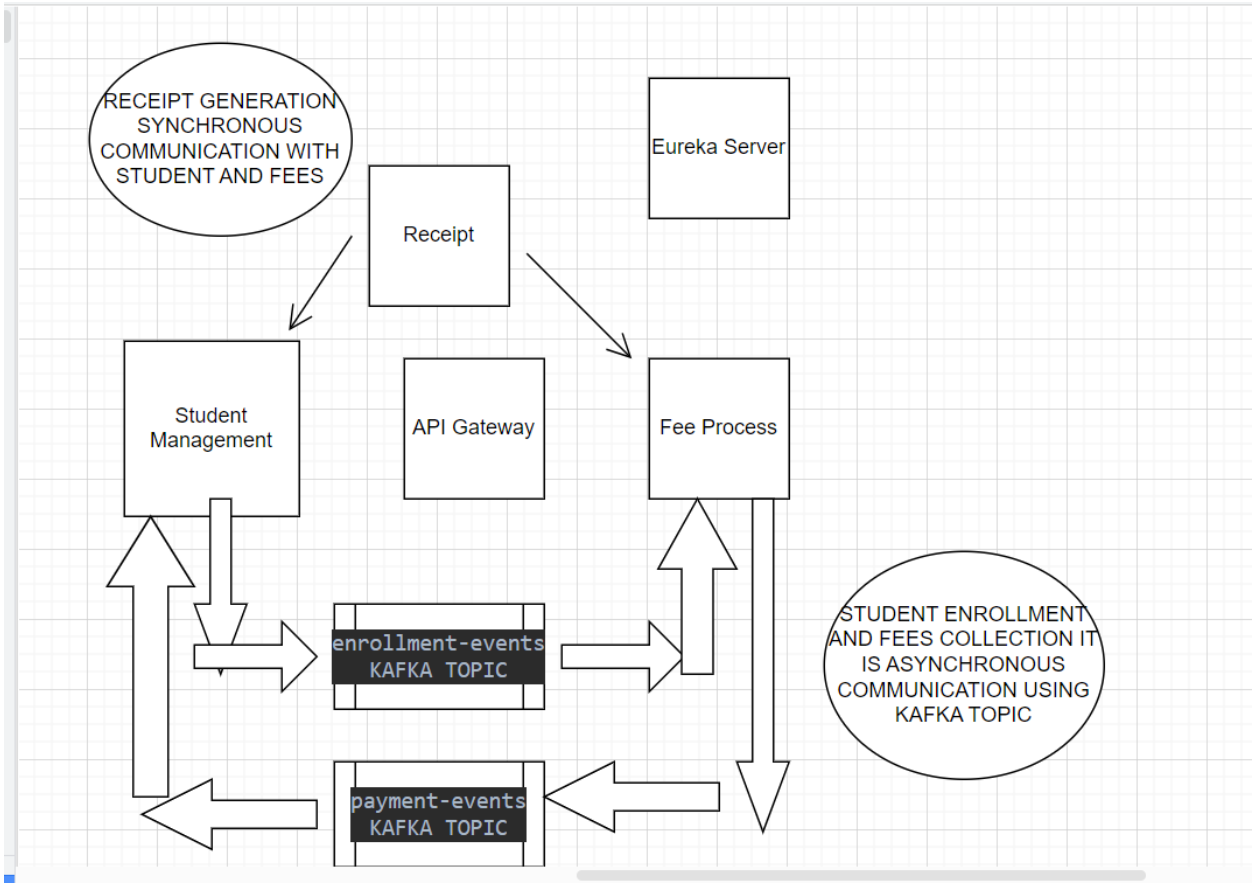
DATA BASE TABLE DESIGN QUERIES FOR MICRO SERVICES

Student Management

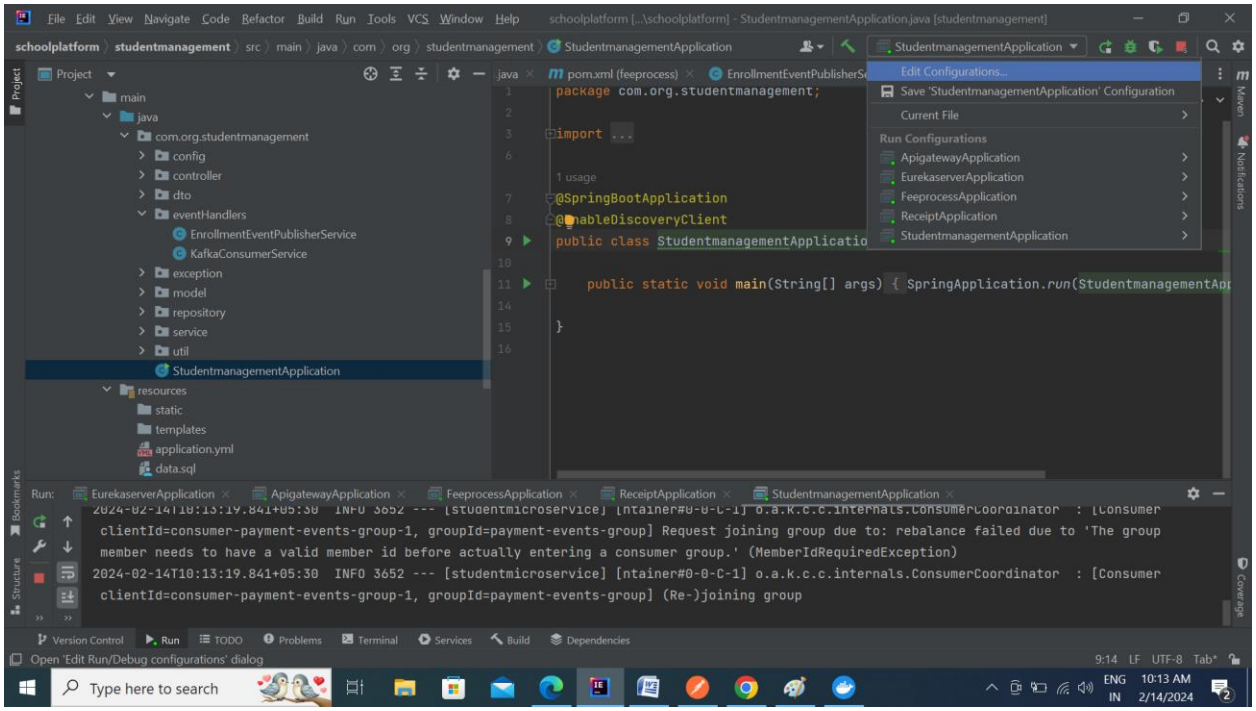
```
CREATE TABLE Student (
  student_id INT AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(255) NOT NULL,
  last_name VARCHAR(255) NOT NULL,
  school_name VARCHAR(255) NOT NULL,
  grade INT,
  email VARCHAR(255),
  mobile_number VARCHAR(20)
);

CREATE TABLE Address (
  address_id INT AUTO_INCREMENT PRIMARY KEY,
  student_id INT NOT NULL,
  street VARCHAR(255) NOT NULL,
  city VARCHAR(255) NOT NULL,
  state VARCHAR(255) NOT NULL,
  postal_code VARCHAR(20),
  CONSTRAINT fk_address_student FOREIGN KEY (student_id) REFERENCES
Student(student_id)
);

CREATE TABLE Guardian (
  guardian_id INT AUTO_INCREMENT PRIMARY KEY,
  student_id INT NOT NULL,
  first_name VARCHAR(255) NOT NULL,
  last_name VARCHAR(255) NOT NULL,
  relationship VARCHAR(255),
  email VARCHAR(255),
  mobile_number VARCHAR(20),
  CONSTRAINT fk_guardian_student FOREIGN KEY (student_id) REFERENCES
Student(student_id)
);

CREATE TABLE Course (
  course_id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description VARCHAR(255),
  cost DECIMAL(10,2) NOT NULL,
  students_limit INT NOT NULL,
  CONSTRAINT CHECK_cost CHECK (cost > 0)
);

CREATE TABLE Enrollment (
  enrollment_id INT AUTO_INCREMENT PRIMARY KEY,
  student_id INT NOT NULL,
  course_id INT NOT NULL,
  joining_year INT NOT NULL,
  status VARCHAR(255) NOT NULL,
  reason VARCHAR(255) NOT NULL,
  CONSTRAINT fk_enrollment_student FOREIGN KEY (student_id) REFERENCES
Student(student_id),
  CONSTRAINT fk_enrollment_course FOREIGN KEY (course_id) REFERENCES
Course(course_id)
);
```

```
FEES PROCESS

CREATE TABLE Fee (
  fee_id INT AUTO_INCREMENT PRIMARY KEY,
  totalFees DECIMAL(10,2) NOT NULL
);


CREATE TABLE Admissions (
  admission_id INT AUTO_INCREMENT PRIMARY KEY,
  enrollment_id INT NOT NULL,
  fee_id INT NOT NULL,
  student_id INT NOT NULL,
  course_id INT NOT NULL,
  joining_year INT NOT NULL,
  cost DECIMAL(10,2) NOT NULL,
  CONSTRAINT fk_admissions_fee FOREIGN KEY (fee_id) REFERENCES Fee(fee_id)
);



CREATE TABLE Payment (
  payment_id INT AUTO_INCREMENT PRIMARY KEY,
  fee_id INT NOT NULL,
  transaction_date_time DATETIME NOT NULL,
  amount DECIMAL(10,2) NOT NULL,
  transaction_status VARCHAR(255) NOT NULL,
  reference_number INT NOT NULL,
  CONSTRAINT fk_payment_fee FOREIGN KEY (fee_id) REFERENCES Fee(fee_id)
);
```

Architecture



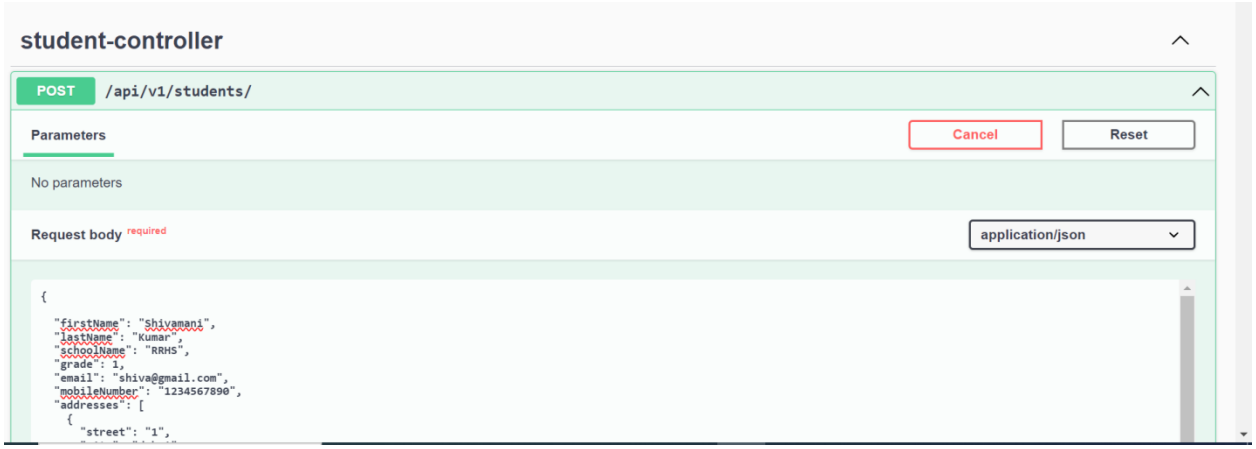
Once all the services are up

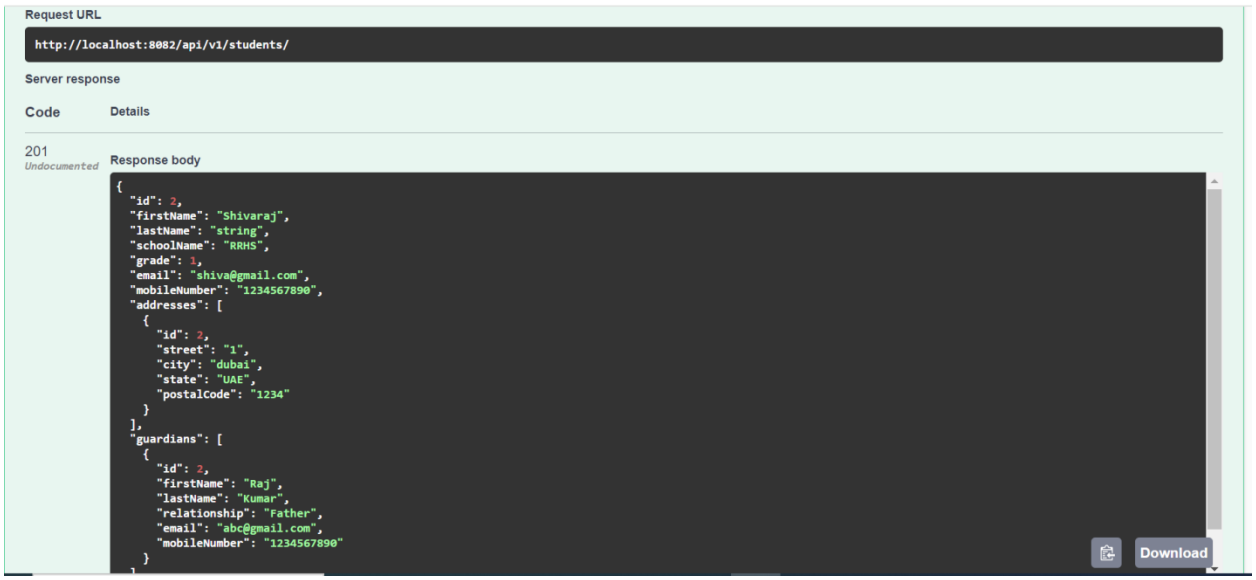


Student Service Swagger URL - http://localhost:8082/swagger-ui/index.html

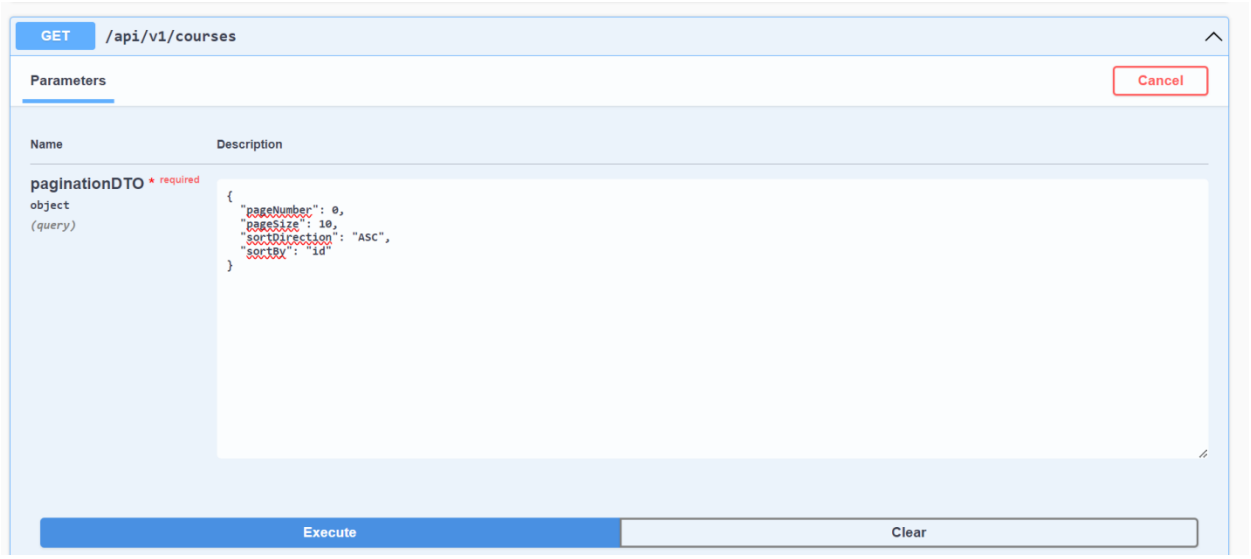
Eureka server Url - http://localhost:8761/

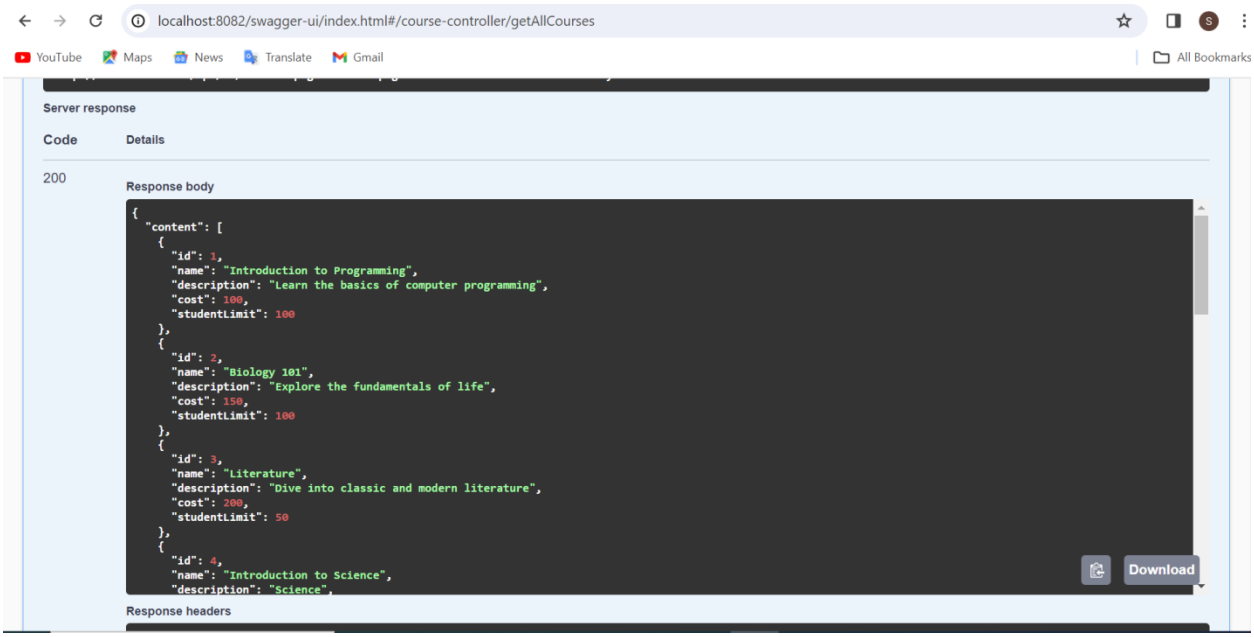
1st API CALL to Register the Student



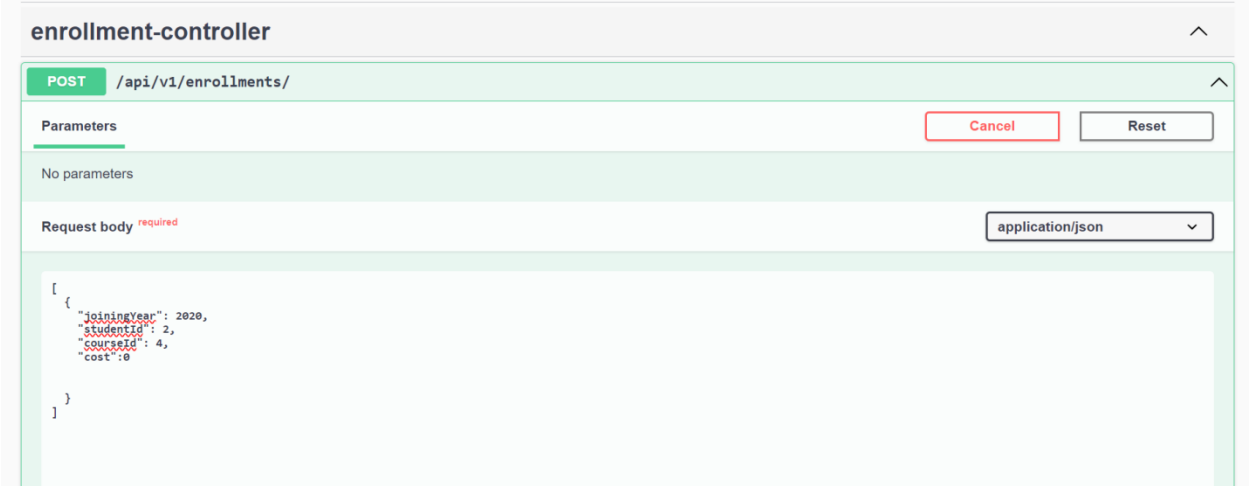


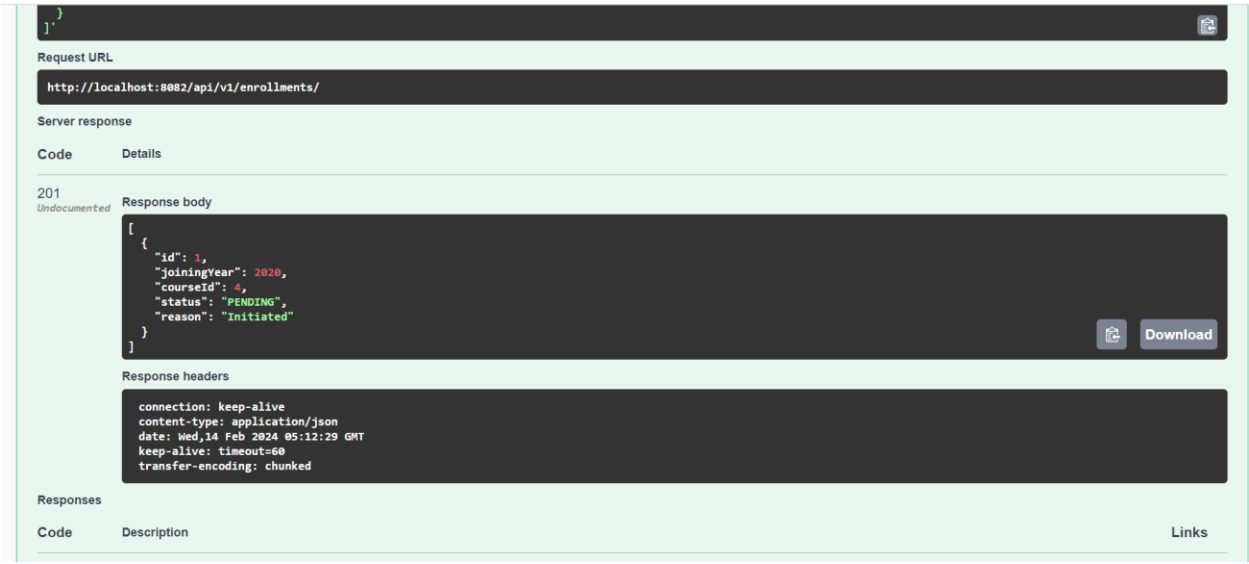
2nd Service to check which courses are available to enroll for student.





3rd service to enroll into course based on the course id

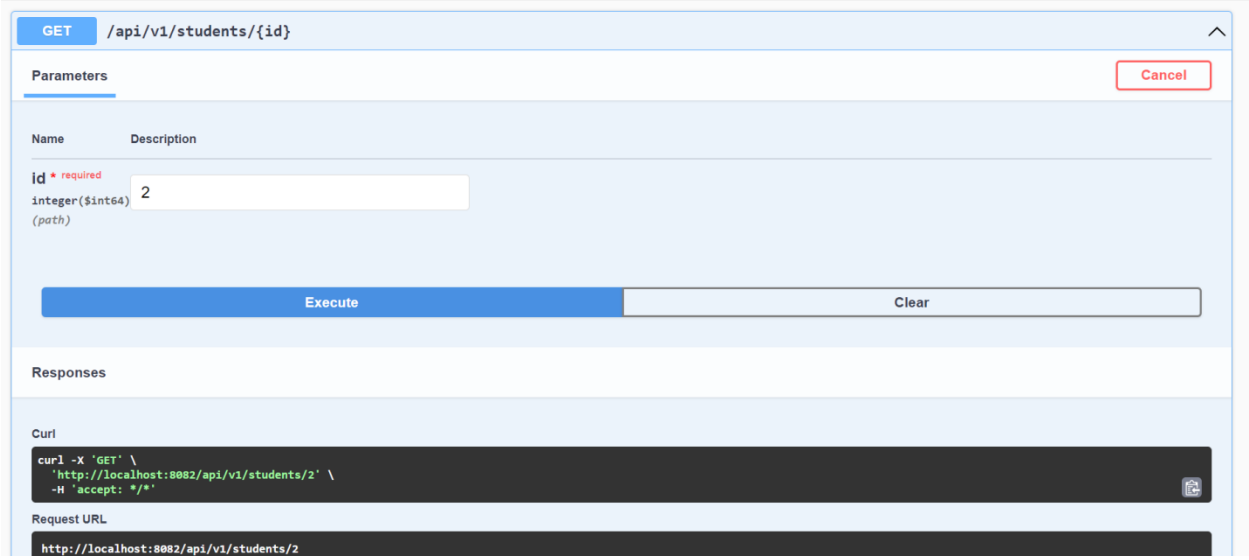


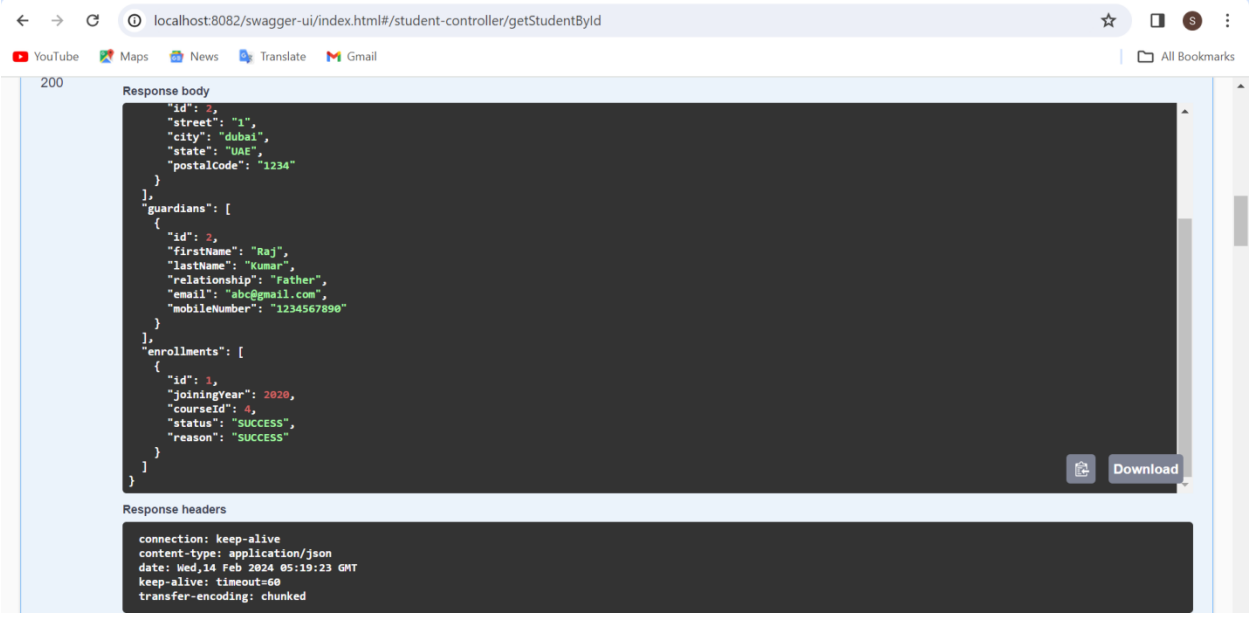


When the student is enrolled for the course. Intial Status will be pending until we get response from fee process microservice. This communication is done using Kafka broker.

Once we get response from Kafka Broker , about the status. The status will be updated. We can verify the status by calling getStudent api based on Student id. The status will be updated to Success. This is event driven communication.

4th service to check enrollment status based on student Id

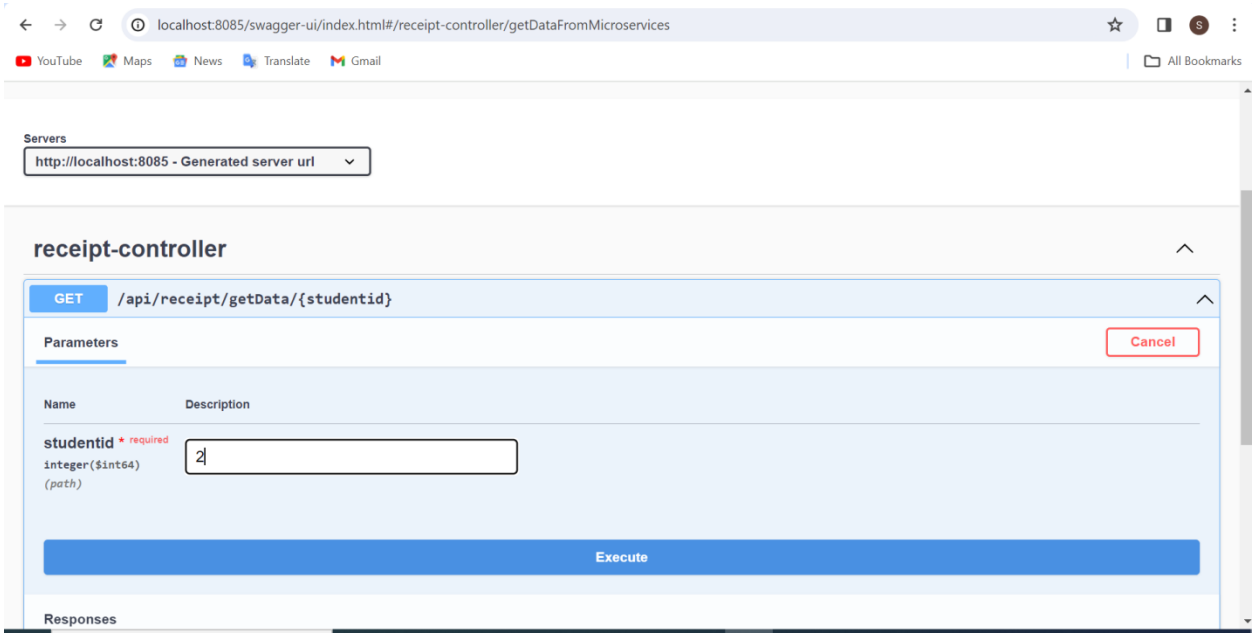


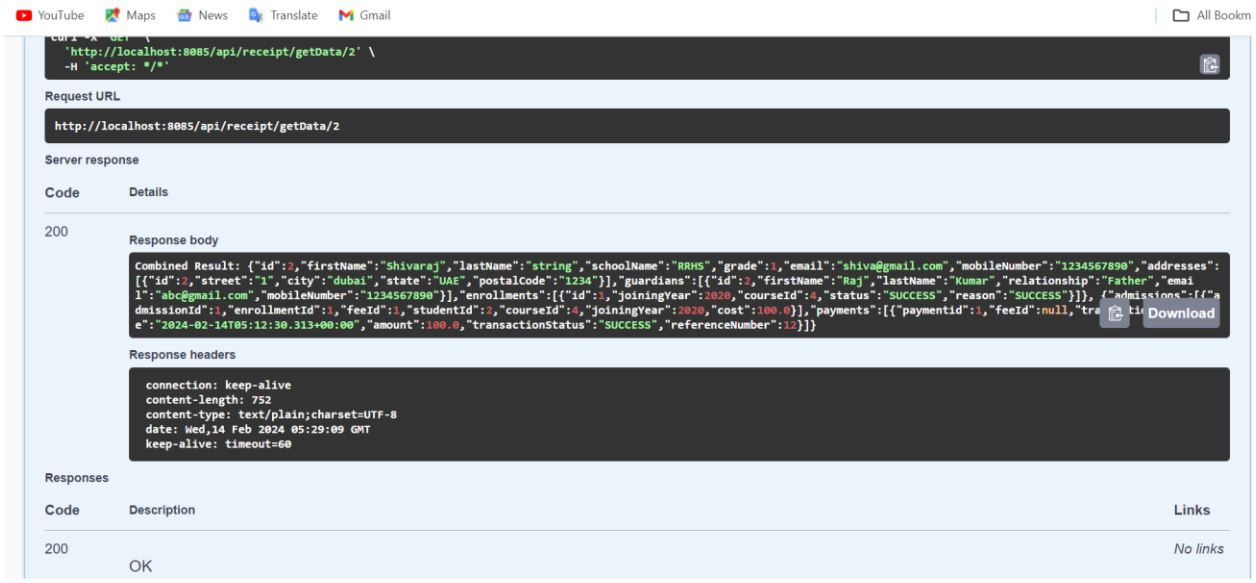


Student Management service sends enrollment event to enrollment event Kaka topic. FeeProcessService listens to the event and initiates Payment and sends the status to payment event Kaka topic. The Student Management service listens to this topic and updates status in its database in enrollment table.

Swagger URL - http://localhost:8085/swagger-ui/index.html

5th Service to generate Receipt Based on the Student ID. This is done using webclient

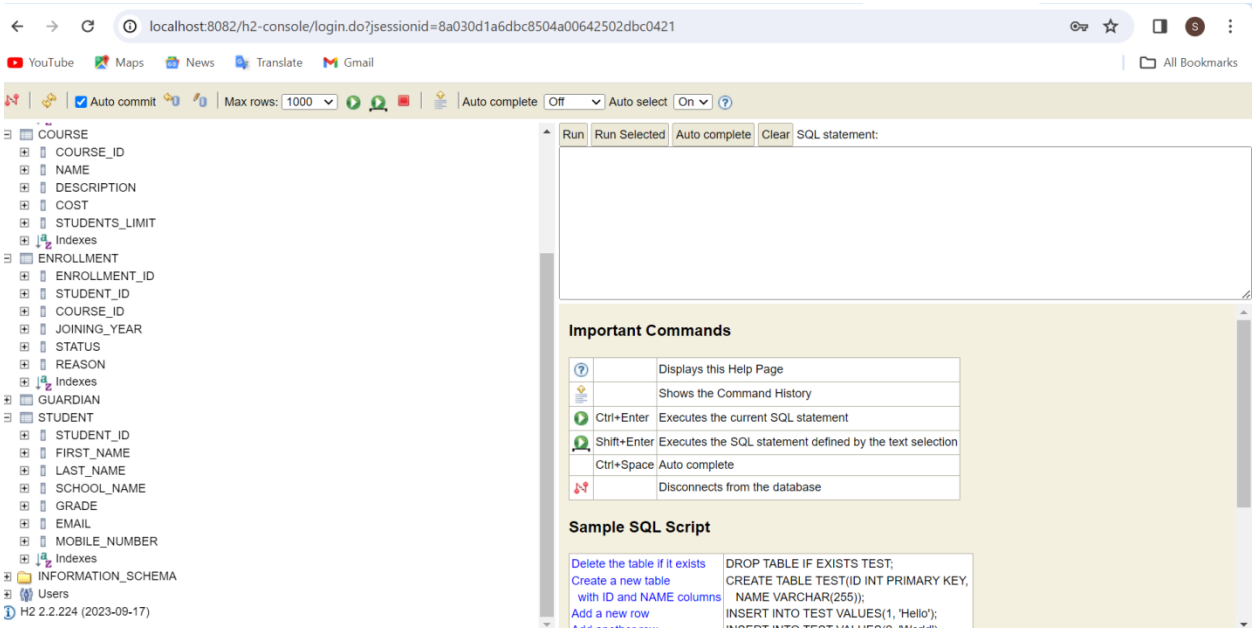




These are the major services for workflow. Apart from these there are services to perform CRUD operations

H2 DataBase Table Screenshots

http://localhost:8082/h2-console/login.jsp?jsessionid=8a030d1a6dbc8504a00642502dbc0421
org.h2.Driver
DB : jdbc:h2:mem:student_information
username : user
password : password



http://localhost:8084/h2-console/login.jsp?jsessionid=8a030d1a6dbc8504a00642502dbc0421
org.h2.Driver
DB : jdbc:h2:mem:payment_information
username : user
password : password