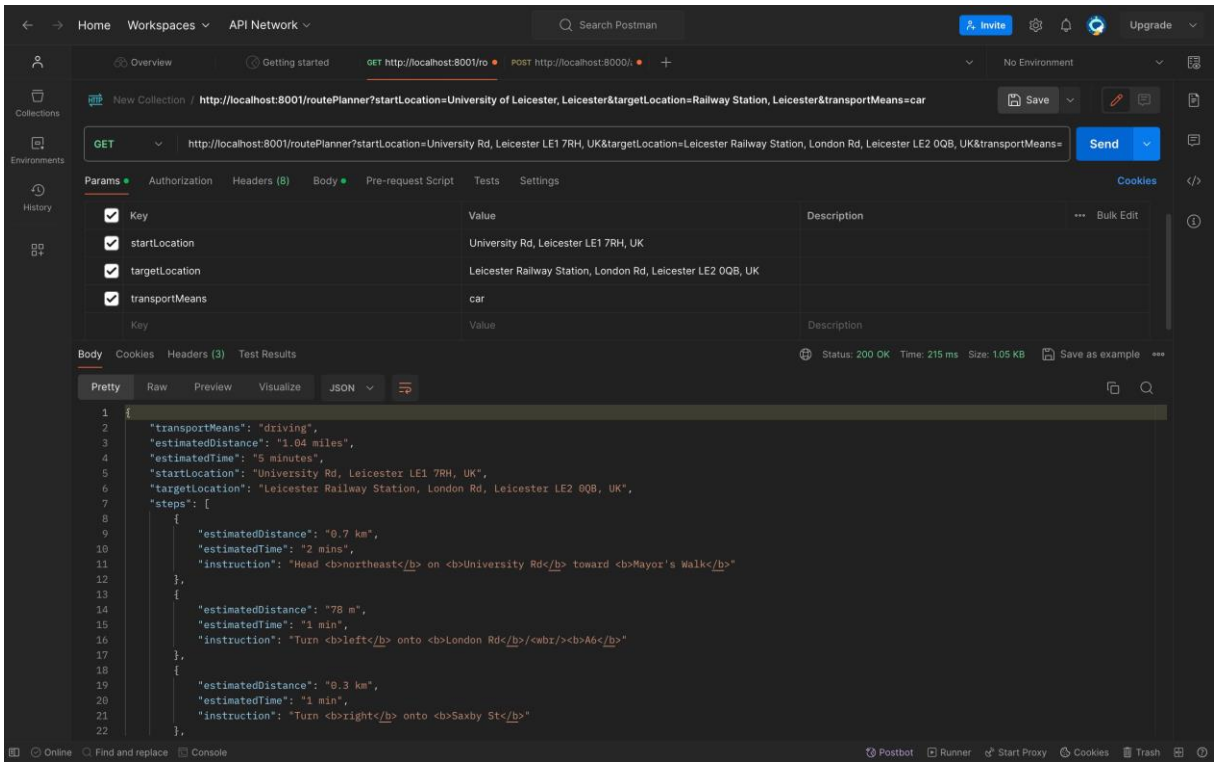
SOA: Service-oriented Architectures

Service Implementation and Testing

1. Route Planner Adapter Implementation:

- **Selection of Route Planning Service:**
  - We have Researched and selected a suitable route planning service Google Maps API.
- **Adapter Implementation:**
  - We have Developed an adapter that interfaces with the chosen route planning service to fetch route details.
  - We have Implemented methods as per the interface defined by IDUServiceRoutePlanner to request route details based on start and target locations, as well as the means of transport.
- **OpenAPI Specification (OAS):**
  - We have Created an OAS document specifying the endpoints, request parameters, and response structures for the adapter's API.

Route Planner Adapter Output:



This OpenAPI Specification (OAS) defines an API for a route planner service. The explanation of the key components:

- OpenAPI Version: This API specification follows version 3.0.0 of the OpenAPI Specification.
- Info: Provides metadata about the API, including the title, description, and version.
- Servers: Specifies the base URL(s) for the API. In this case, there's one server defined with the URL http://localhost:8001.

- Paths: Defines the available endpoints and their operations.

- /routePlanner: This endpoint has a single operation, GET, for retrieving route details. It accepts three query parameters: startLocation, targetLocation, and transportMeans.

- Parameters: Describes the parameters accepted by the GET /routePlanner operation.

    - startLocation: The starting location of the route.
    - targetLocation: The destination location of the route.
    - transportMeans: The means of transport (e.g., car, bicycle, e-bike, motorbike).

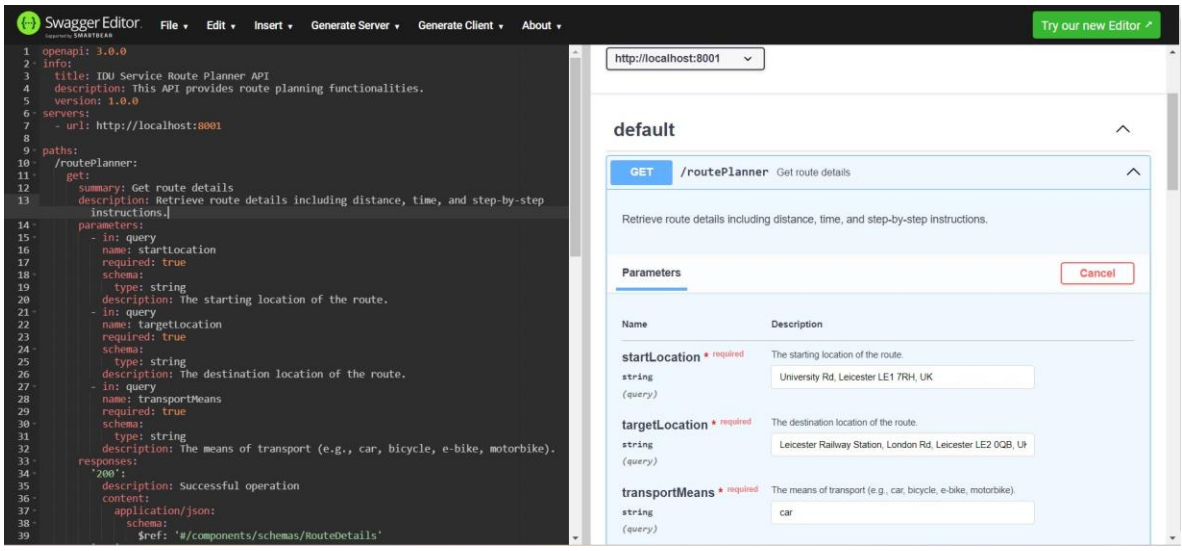
- Responses: Defines the possible responses for the GET /routePlanner operation.

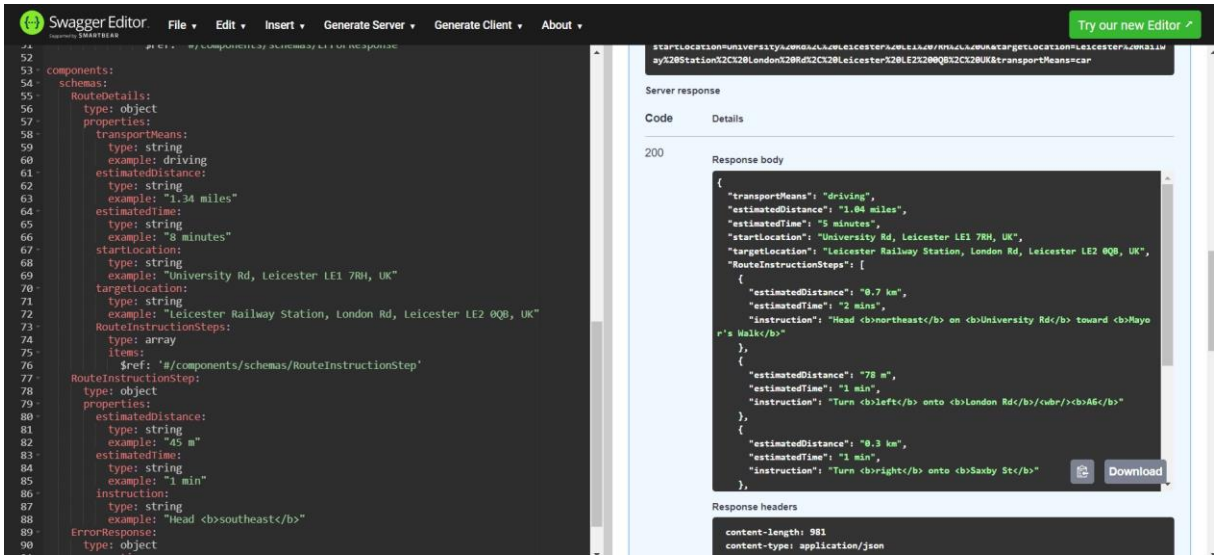
    - 200: Successful response with route details in the JSON format.
    - 400: Bad request response for invalid parameters.
    - 500: Internal server error response.

- Components: Contains reusable components such as schemas, which define the structure of the data exchanged by the API.

    - RouteDetails: Defines the structure of the route details returned by the API.
    - RouteInstructionStep: Defines the structure of a step in the route instruction.
    - ErrorResponse: Defines the structure of an error response returned by the API.

Swagger Editor Validation for OpenAPI Specification (OAS) - Route Planner Service:



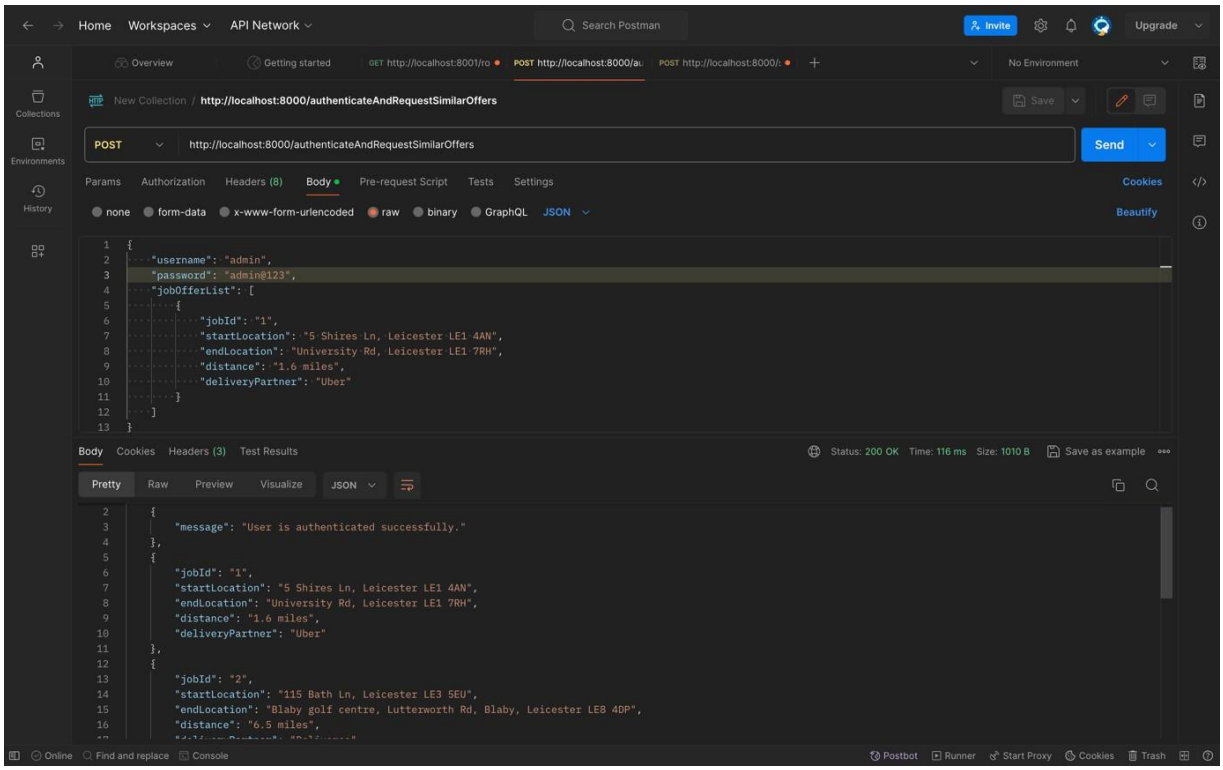


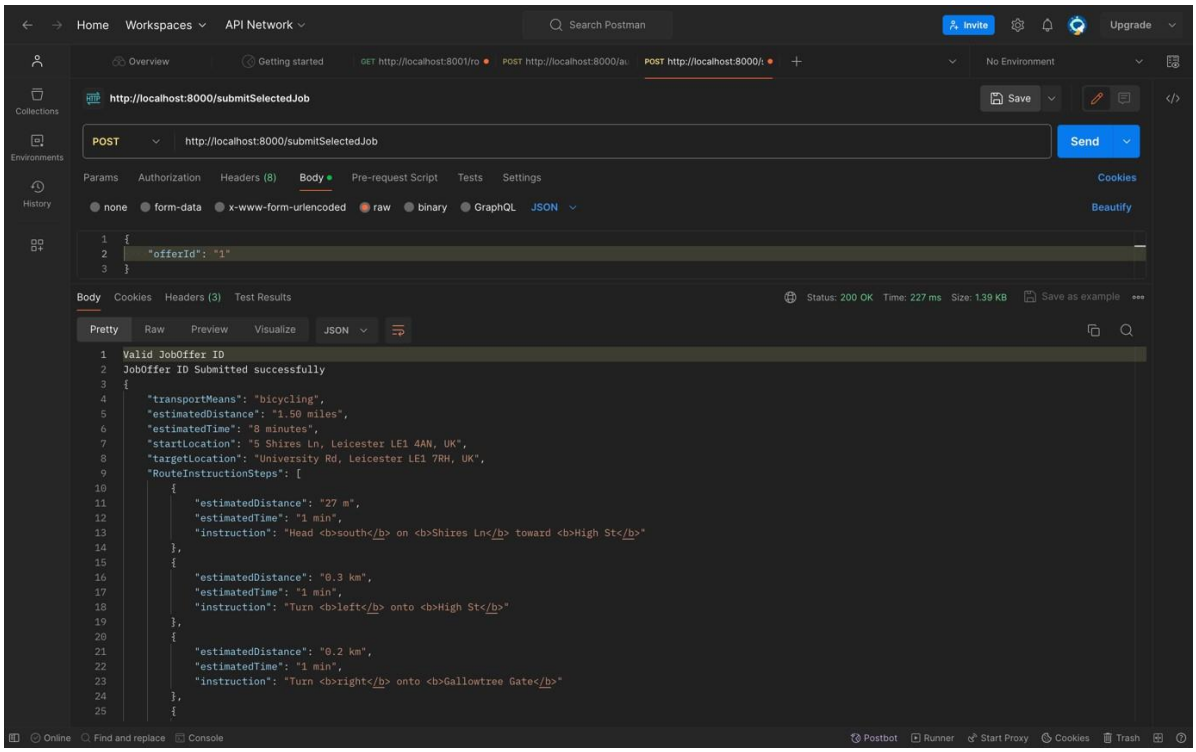
We have attached the **Route Planner Adapter code and OpenAPI Specification (OAS) code for a route planner service in the Zip File.**

2. DriversUnited Service Implementation:

1. **Utilizing Route Planner Interface**:
    - We have Incorporated the interface defined by the IDUServiceRoutePlanner.yml OAS to integrate route planning functionality into the DriversUnited service.
2. **Service Implementation**:
    - We have Implemented the DriversUnited service according to the provided interface (IDUAppDUService.yml OAS).
    - We have Ensured proper handling of operations such as authentication, job submission, and offer retrieval.
3. **OAS for DriversUnited Service**:
    - We have Developed an OAS document defining the endpoints, request payloads, and response schemas for the DriversUnited service API.

DriversUnited Service Implementation Output:





This OpenAPI Specification (OAS) describes an API for an application called IDUAppDUService. The explanation of its key components:

- OpenAPI Version: This API specification follows version 3.0.0 of the OpenAPI Specification.

- Info: Provides metadata about the API, including the title and version.

- Servers: Specifies the base URL(s) for the API. In this case, there's one server defined with the URL http://localhost:8000.

- Paths: Defines the available endpoints and their operations.

- /authenticateAndRequestSimilarOffers: This endpoint has a single operation, POST, for authenticating a user and requesting similar job offers. It accepts a JSON request body containing username, password, and jobOffers.

- /submitSelectedJob: This endpoint also has a single operation, POST, for submitting a selected job offer. It accepts a JSON request body containing an offerId.

- Request Body: Describes the structure of the request body for each operation.

- For /authenticateAndRequestSimilarOffers, the request body must include username, password, and jobOffers.

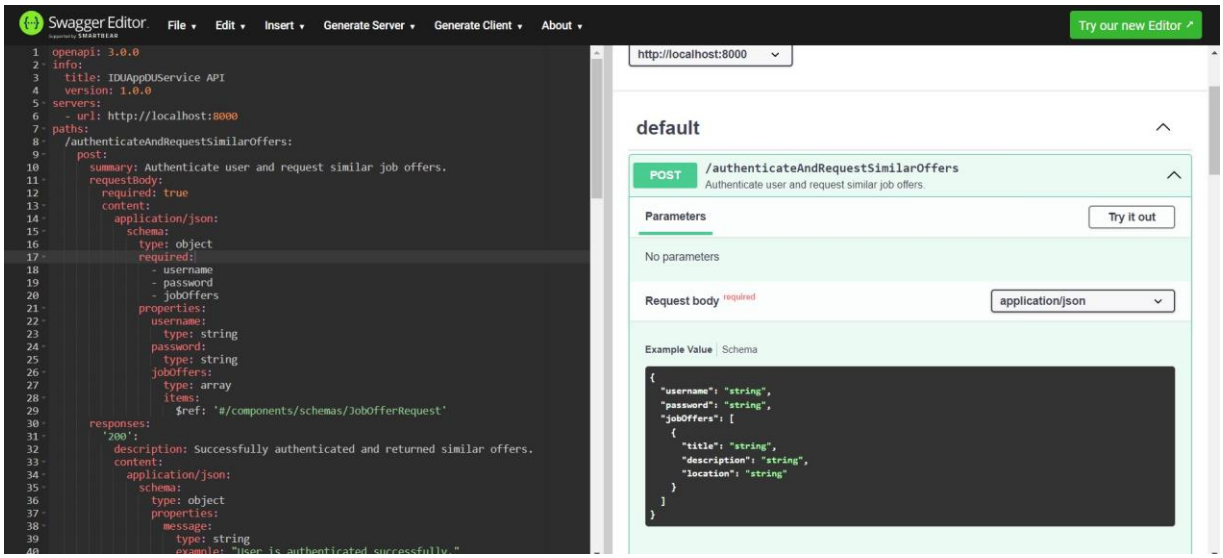
- For /submitSelectedJob, the request body must include an offerId.

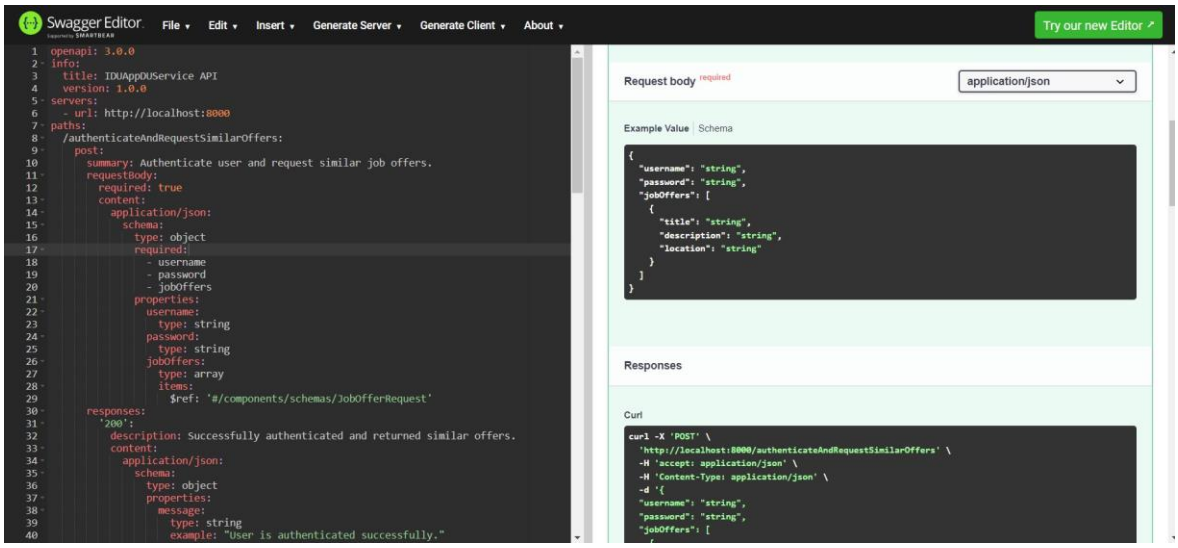
- Responses: Defines the possible responses for each operation.

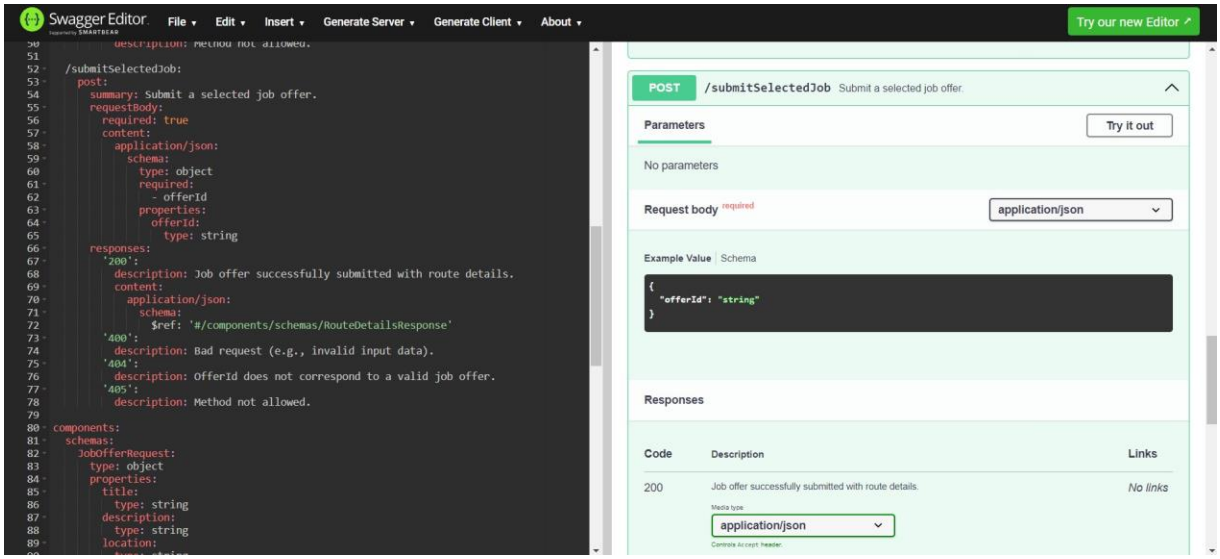
- For /authenticateAndRequestSimilarOffers:
    - 200: Successful response with a message indicating successful authentication and an array of similar job offers.
    - 401: Authentication failed.
    - 400: Bad request response for invalid request body.
    - 405: Method not allowed.
- For /submitSelectedJob:
    - 200: Successful response with route details for the submitted job offer.
    - 400: Bad request response for invalid input data.
    - 404: OfferId does not correspond to a valid job offer.
    - 405: Method not allowed.
- Components: Contains reusable components such as schemas, which define the structure of the data exchanged by the API.
    - JobOfferRequest: Defines the structure of a job offer request.

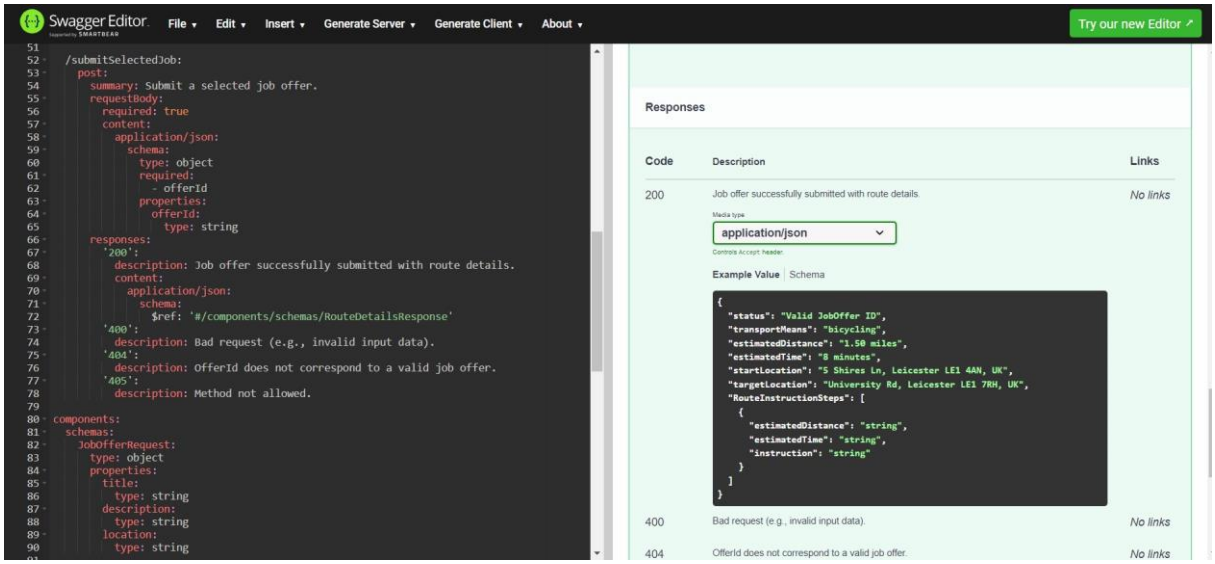
- JobOfferResponse: Defines the structure of a job offer response.
- RouteDetailsResponse: Defines the structure of a response containing route details.
- RouteInstructionStep: Defines the structure of a step in the route instruction.

Swagger Editor Validation for OpenAPI Specification (OAS) - DriversUnitedService:



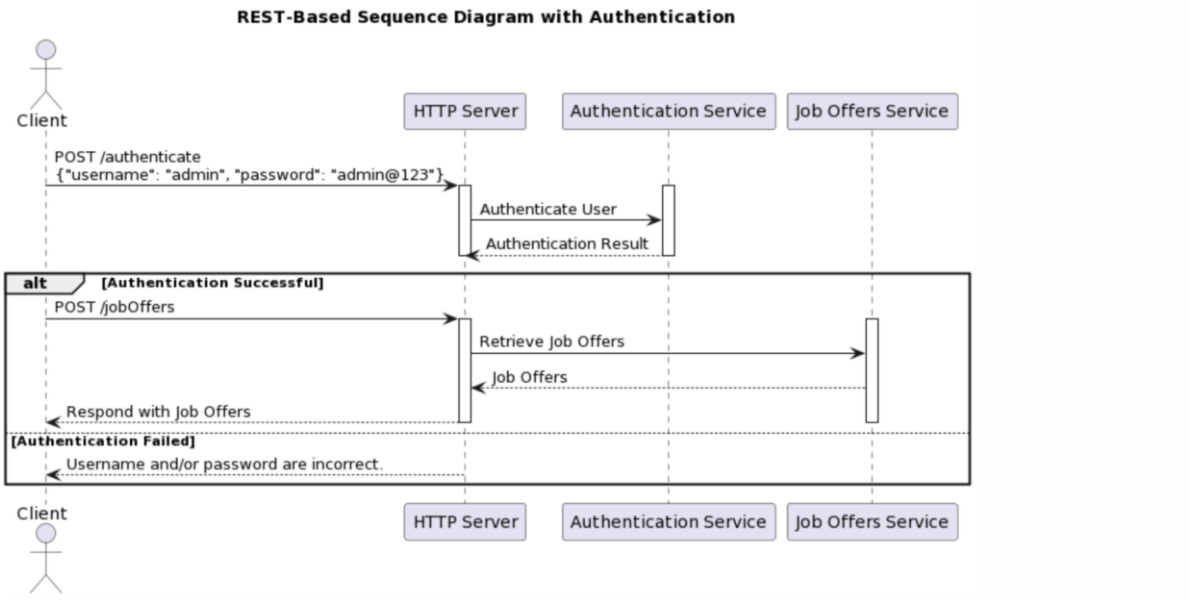






We have attached **DriversUnitedService code and OpenAPI Specification (OAS) code for IDUAppDUService in the Zip File.**

3. **Three concrete REST-specific sequence diagrams describing the execution of test cases with their concrete inputs and outputs covering different interactions.**



Client Authentication Request:

- The client initiates the interaction by sending a POST request to the server, containing the user's credentials for authentication.

Server Authentication Handling:

- Upon receiving the authentication request, the server activates the Authentication Service to validate the user credentials.

Authentication Service Processing:

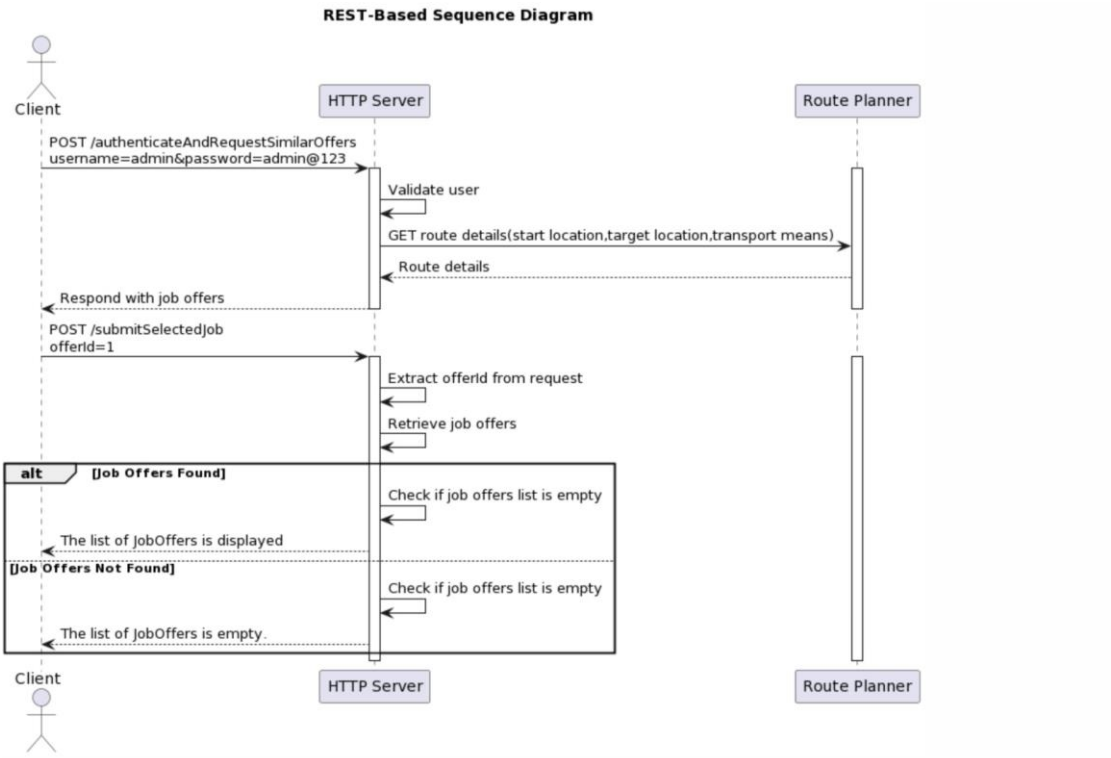
- The Authentication Service verifies the provided credentials and responds to the server with the authentication result.

Successful Authentication:

- If the authentication is successful, indicating valid user credentials, the server proceeds to fetch job offers from the Job Offers Service.

Job Offers Retrieval and Response:

- The server sends a request to the Job Offers Service to retrieve the available job offers. Upon receiving the job offers from the Job Offers Service, the server responds to the client with the list of job opportunities.



Client Authentication and Job Offer Request:

- The client initiates the process by sending a POST request to the HTTP server /authenticateAndRequestSimilarOffers with the username admin and password admin@123.
- Upon receiving the request, the server activates and validates the user's credentials.
- The server then communicates with the external service to obtain route details based on the provided addresses.
- After obtaining the route details, the server responds to the client with the requested job offers.

Client Submission of Selected Job:

- Following the authentication and retrieval of job offers, the client proceeds to submit a selected job by sending a POST request to the HTTP server /submitSelectedJob with the offer ID 1.
- Upon receiving this request, the server activates again and extracts the offer ID from the request.

Valid Job Offer Found:

- If a valid job offer corresponding to the provided offer ID is found:
- The server retrieves the job offers.
- It responds to the client with the message "Job offer submitted successfully."

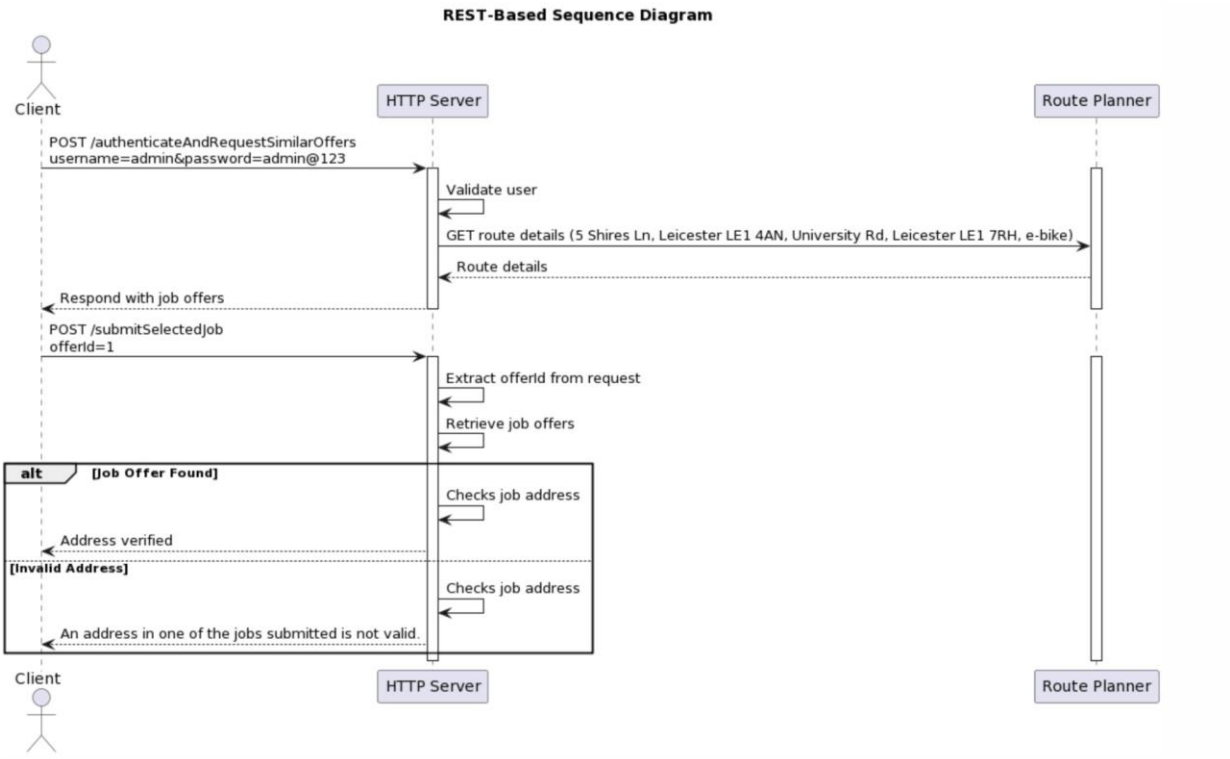
Invalid Address Detected:

- In case an invalid address is detected during the process:
- The server identifies the issue by checking the address of the job offer.

- It responds to the client with the message "An address in one of the jobs submitted is not valid."

Process Completion:

- After handling the job submission and address validation, the server and external service are deactivated, marking the completion of the interaction between the client, server, and external service.



Client Authentication and Job Offer Request:

- The client sends a POST request to the HTTP server /authenticateAndRequestSimilarOffers with the username admin and password admin@123.
- The server validates the user and retrieves the route details from the external service.
- The server responds to the client with the requested job offers.

Client Submission of Selected Job:

- The client sends a POST request to the HTTP server /submitSelectedJob with the selected job offer ID 1.
- The server extracts the offer ID from the request and retrieves the job offers.

Job Offer Found:

- If a valid job offer is found based on the provided offer ID:
- The server checks the address of the job offer.
- The server responds to the client with the message "Address verified."

Invalid Address:

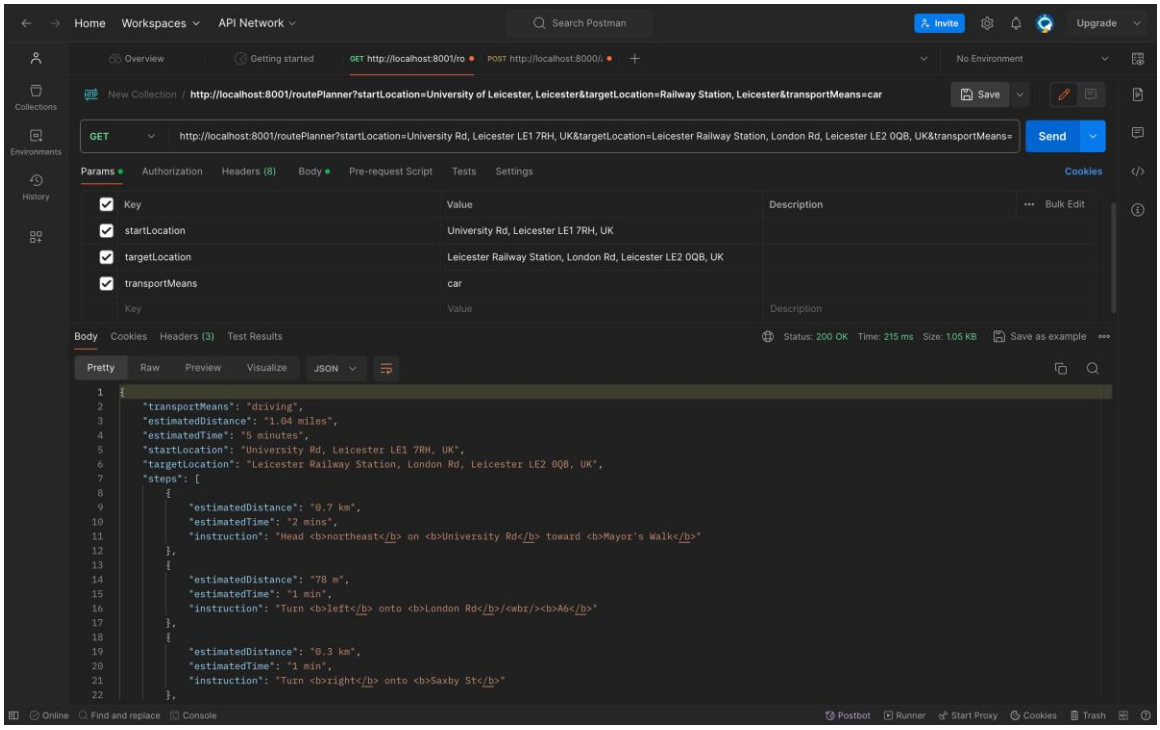
- If an address in one of the submitted jobs is not valid:
- The server checks the address of the job offer.
- The server responds to the client with the message "An address in one of the jobs submitted is not valid."

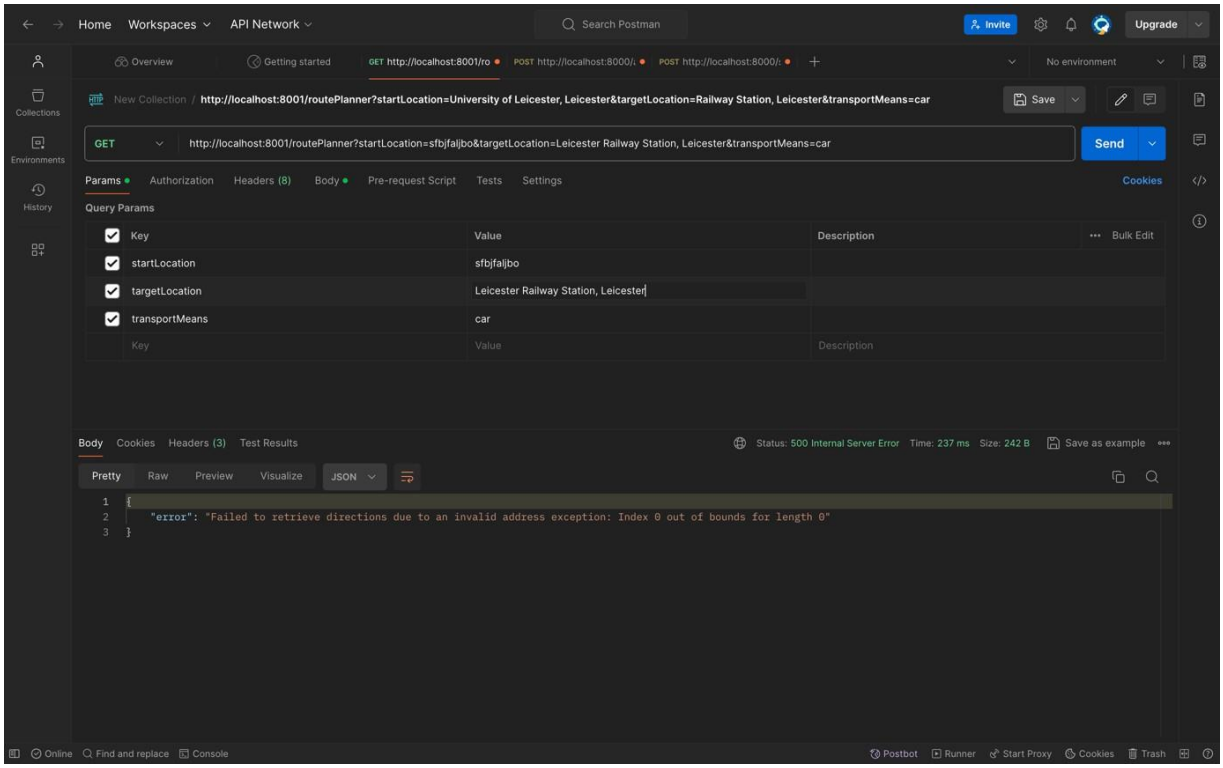
Process Completion:

- The interaction between the client, server, and external service concludes, and all components are deactivated.

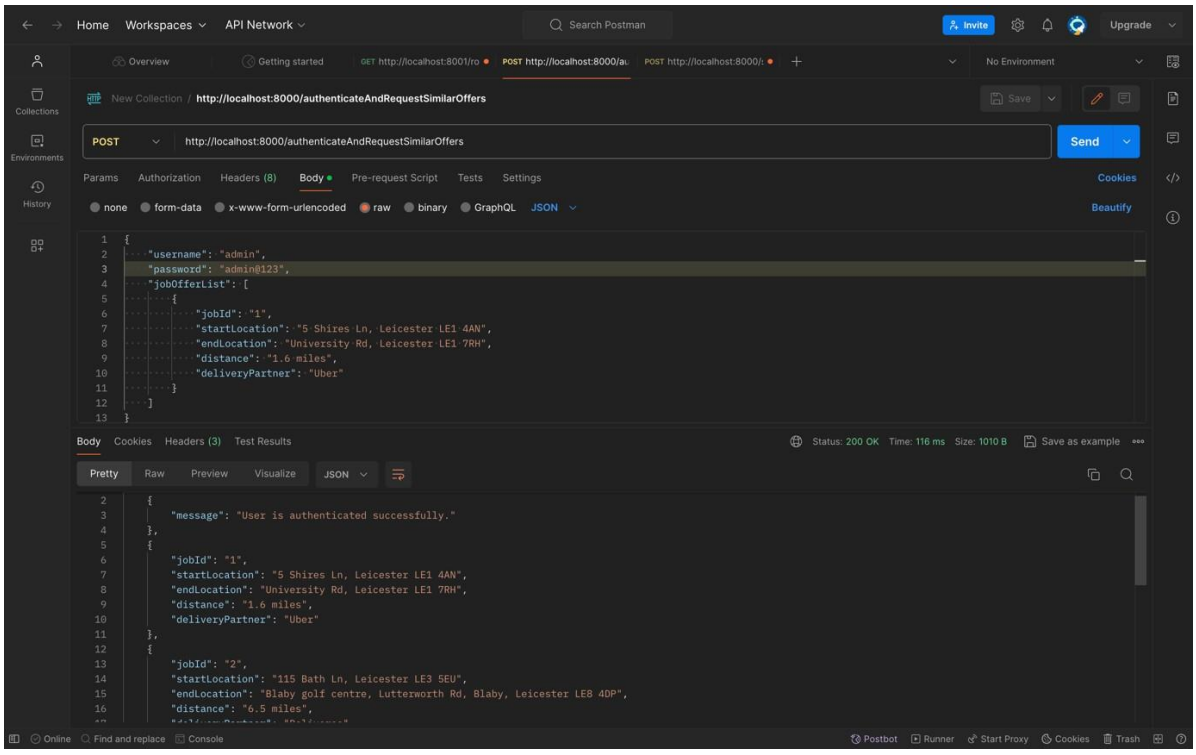
4. Postman Validation Screenshots:

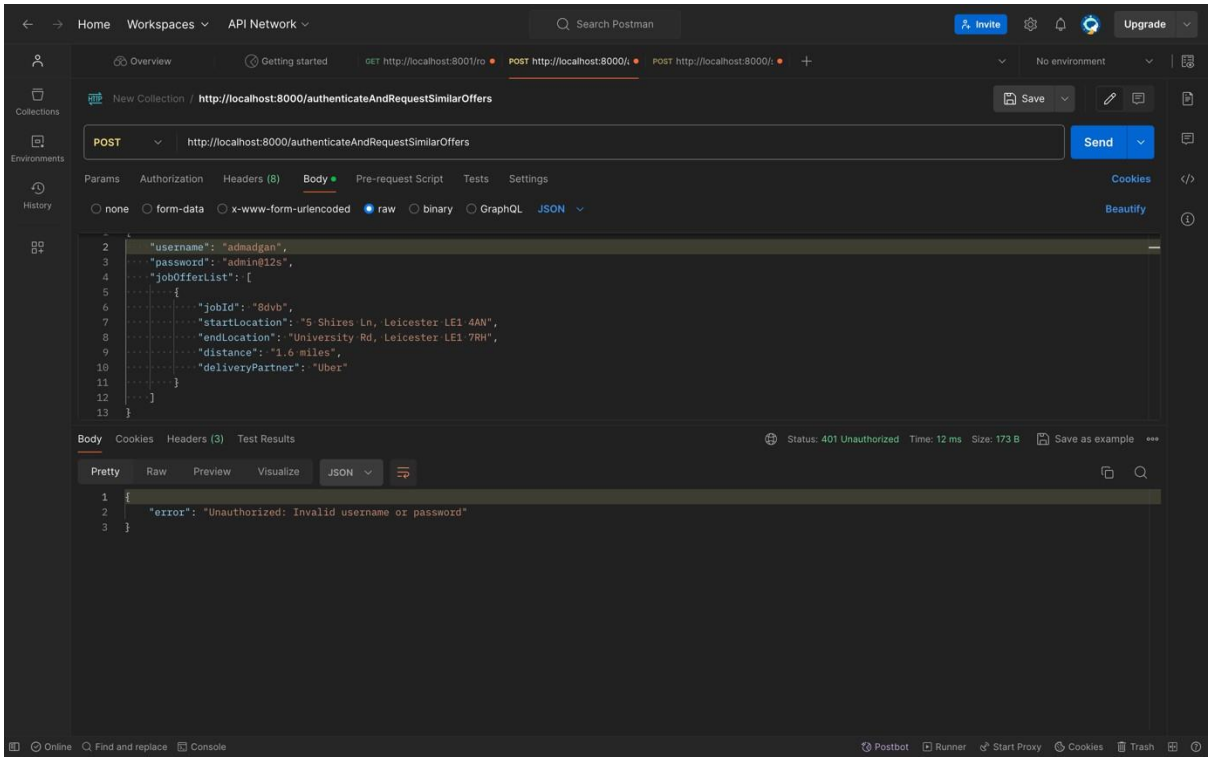
IDUServiceRoutePlanner Validation:

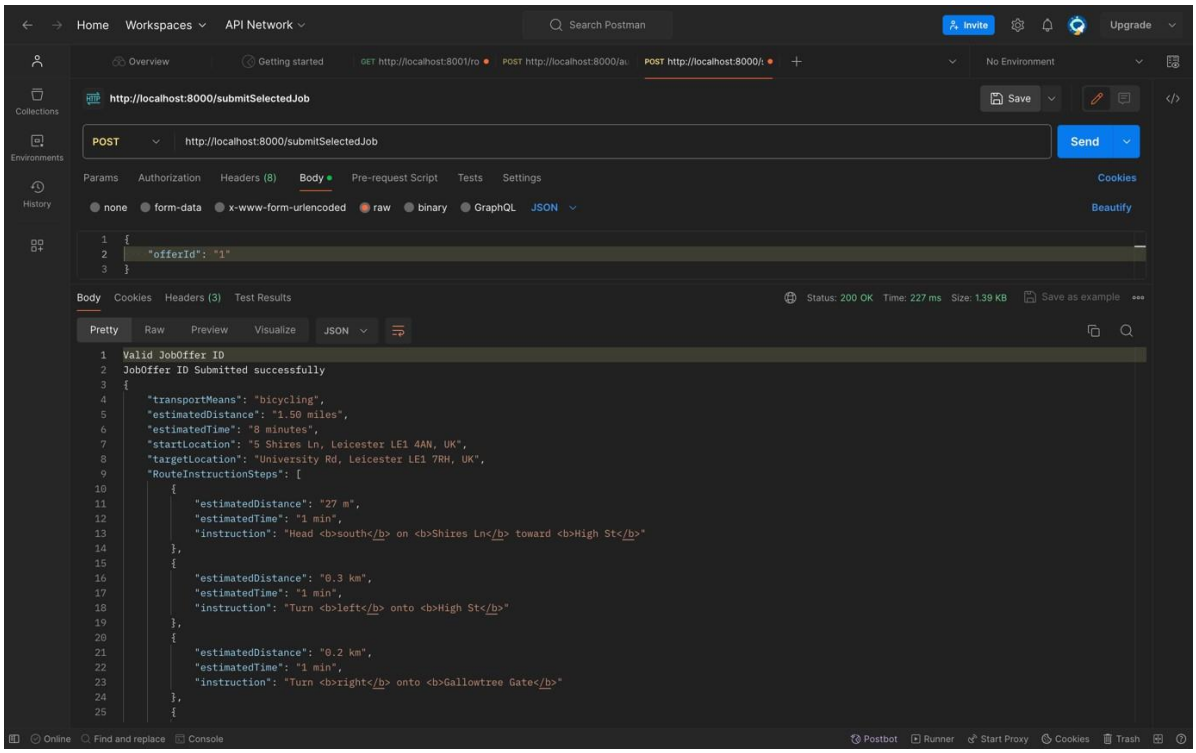


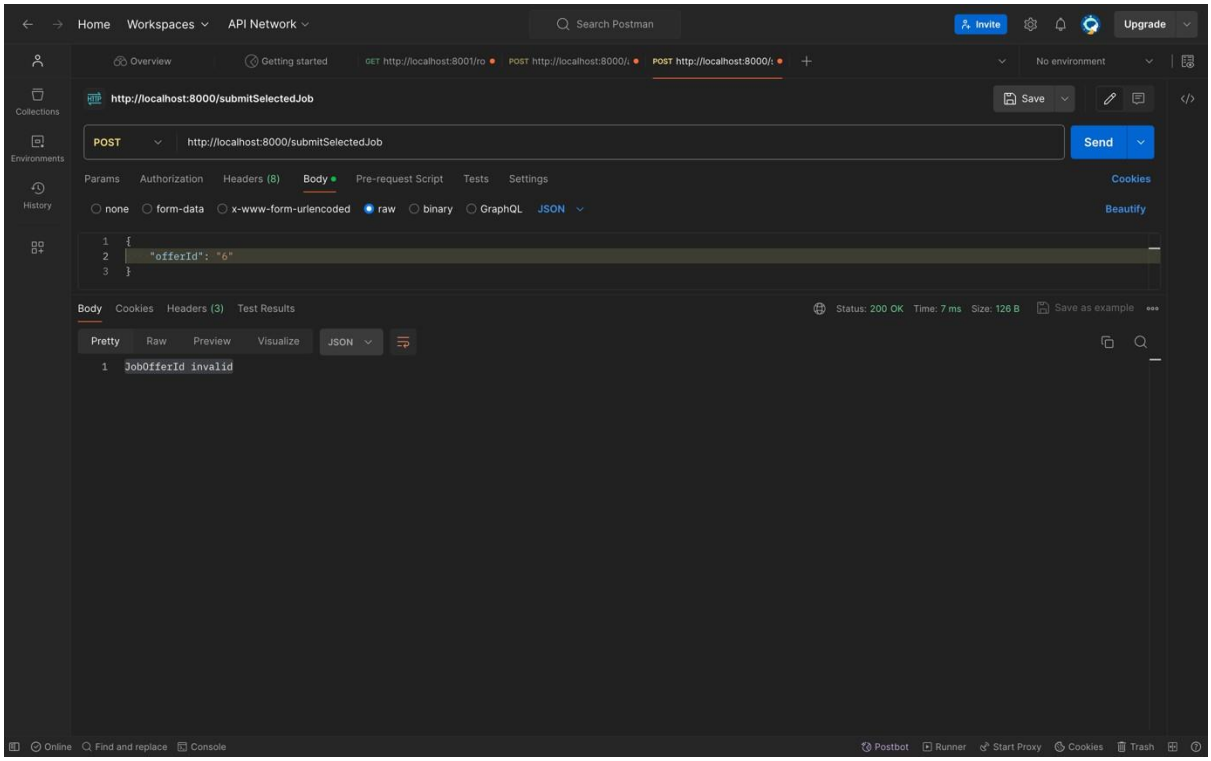


IDUAppDUService Vaildation:




Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Overview
Getting started
GET http://localhost:8001/ro
POST http://localhost:8000/
POST http://localhost:8000/
No environment
Collections
Environments
History
New Collection / http://localhost:8000/authenticateAndRequestSimilarOffers
Save
POST
http://localhost:8000/authenticateAndRequestSimilarOffers
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"username": "admadgan",
"password": "admin@12s",
"jobOfferList": [
{
"jobId": "8dvb",
"startLocation": "5 Shires Ln, Leicester LE1 4AN",
"endLocation": "University Rd, Leicester LE1 7RH",
"distance": "1.6 miles",
"deliveryPartner": "Uber"
}
]
}
Body
Cookies
Headers (3)
Test Results
Status: 401 Unauthorized
Time: 12 ms
Size: 173 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"error": "Unauthorized: Invalid username or password"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Overview
Getting started
GET http://localhost:8001/ro
POST http://localhost:8000/au
POST http://localhost:8000/
No Environment
Collections
Environments
History
http://localhost:8000/submitSelectedJob
Save
POST
http://localhost:8000/submitSelectedJob
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"offerId": "1"
}
Body
Cookies
Headers (3)
Test Results
Status: 200 OK
Time: 227 ms
Size: 1.39 KB
Save as example
Pretty
Raw
Preview
Visualize
JSON
Valid JobOffer ID
JobOffer ID Submitted successfully
{
"transportMeans": "bicycling",
"estimatedDistance": "1.50 miles",
"estimatedTime": "8 minutes",
"startLocation": "5 Shires Ln, Leicester LE1 4AN, UK",
"targetLocation": "University Rd, Leicester LE1 7RH, UK",
"RouteInstructionSteps": [
{
"estimatedDistance": "27 m",
"estimatedTime": "1 min",
"instruction": "Head south on Shires Ln toward High St"
},
{
"estimatedDistance": "0.3 km",
"estimatedTime": "1 min",
"instruction": "Turn left onto High St"
},
{
"estimatedDistance": "0.2 km",
"estimatedTime": "1 min",
"instruction": "Turn right onto Gallowtree Gate"
},
{
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Overview
Getting started
GET http://localhost:8001/ro
POST http://localhost:8000/
POST http://localhost:8000/
No environment
Collections
Environments
History
http://localhost:8000/submitSelectedJob
Save
POST
http://localhost:8000/submitSelectedJob
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"offerId": "6"
}
Body
Cookies
Headers (3)
Test Results
Status: 200 OK
Time: 7 ms
Size: 126 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
JobOfferId invalid
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash