

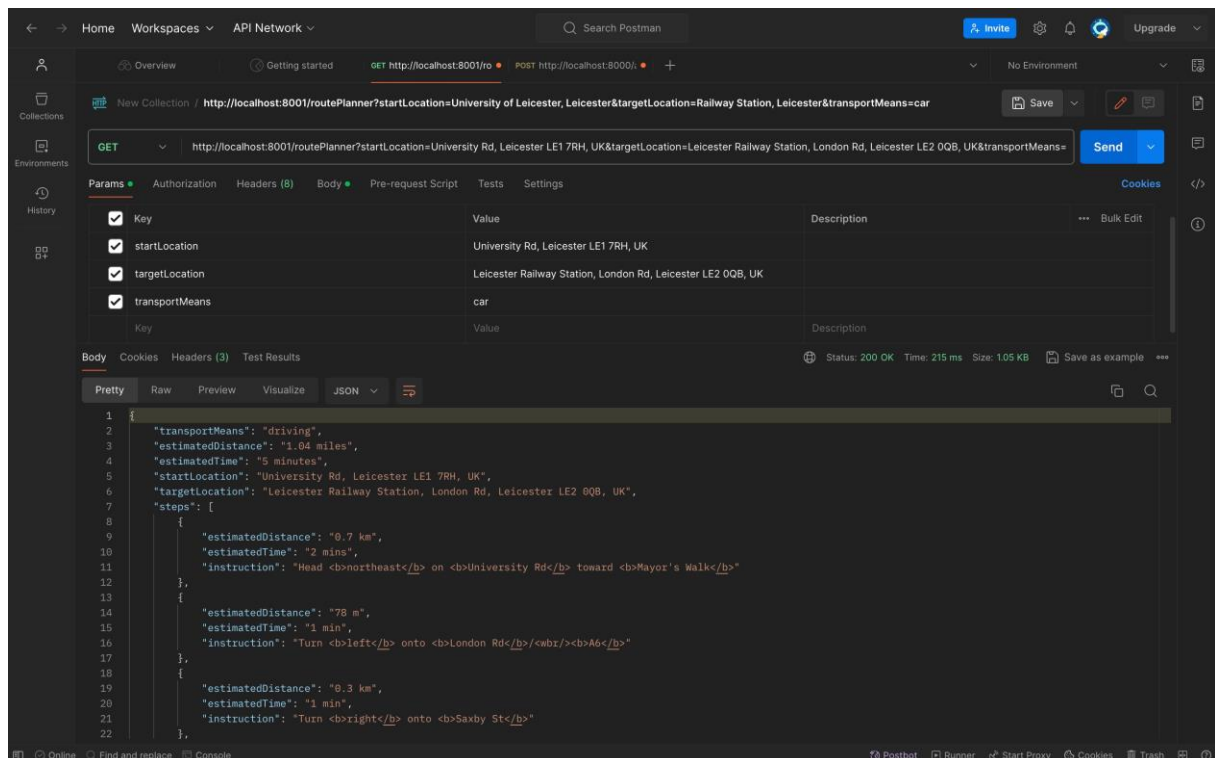
# SOA: Service-oriented Architectures

## 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:

The screenshot displays the Swagger Editor interface for the 'Route Planner Service'. The left pane shows the OpenAPI specification in YAML format, and the right pane shows the corresponding interactive UI.

**OpenAPI Specification (YAML):**

```

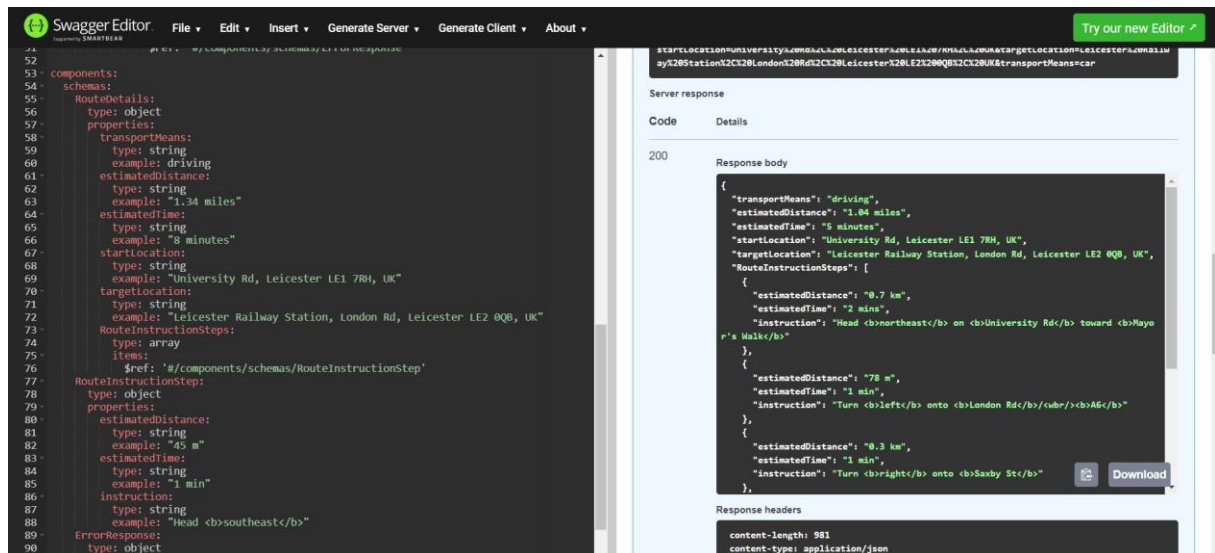
1 openapi: 3.0.0
2 info:
3   title: IDU Service Route Planner API
4   description: This API provides route planning functionalities.
5   version: 1.0.0
6 servers:
7   - url: http://localhost:8001
8
9 paths:
10  /routePlanner:
11    get:
12      summary: Get route details
13      description: Retrieve route details including distance, time, and step-by-step instructions.
14      parameters:
15        - in: query
16          name: startLocation
17          required: true
18          schema:
19            type: string
20            description: The starting location of the route.
21        - in: query
22          name: targetLocation
23          required: true
24          schema:
25            type: string
26            description: The destination location of the route.
27        - in: query
28          name: transportMeans
29          required: true
30          schema:
31            type: string
32            description: The means of transport (e.g., car, bicycle, e-bike, motorbike).
33      responses:
34        '200':
35          description: Successful operation
36          content:
37            application/json:
38              schema:
39                $ref: '#/components/schemas/RouteDetails'
  
```

**Interactive UI:**

The UI shows the endpoint `GET /routePlanner` with the description: "Retrieve route details including distance, time, and step-by-step instructions." The parameters section lists three required query parameters:

Name	Description
<b>startLocation</b> * required	The starting location of the route.
<b>targetLocation</b> * required	The destination location of the route.
<b>transportMeans</b> * required	The means of transport (e.g., car, bicycle, e-bike, motorbike).

Each parameter has a text input field with example values: "University Rd, Leicester LE1 7RH, UK" for startLocation, "Leicester Railway Station, London Rd, Leicester LE2 0QB, UK" for targetLocation, and "car" for transportMeans. A "Cancel" button is visible next to the parameters section.



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:

Postman interface showing a successful POST request to `http://localhost:8000/authenticateAndRequestSimilarOffers`. The request body is a JSON object with the following structure:

```
1 {
2   "username": "admin",
3   "password": "admin@123",
4   "jobOfferList": [
5     {
6       "jobId": "1",
7       "startLocation": "5 Shires Ln, Leicester LE1 4AN",
8       "endLocation": "University Rd, Leicester LE1 7RH",
9       "distance": "1.6 miles",
10      "deliveryPartner": "Uber"
11    }
12  ]
13 }
```

The response body (Status: 200 OK, Time: 116 ms, Size: 1010 B) is a JSON object with the following structure:

```
2 {
3   "message": "User is authenticated successfully.",
4 },
5 {
6   "jobId": "1",
7   "startLocation": "5 Shires Ln, Leicester LE1 4AN",
8   "endLocation": "University Rd, Leicester LE1 7RH",
9   "distance": "1.6 miles",
10  "deliveryPartner": "Uber"
11 },
12 {
13   "jobId": "2",
14   "startLocation": "115 Bath Ln, Leicester LE3 5EU",
15   "endLocation": "Blaby golf centre, Lutterworth Rd, Blaby, Leicester LE8 4DP",
16   "distance": "6.5 miles",
17   "deliveryPartner": "Uber"
18 }
```

Postman interface showing a successful POST request to `http://localhost:8000/submitSelectedJob`. The request body is a JSON object with the following structure:

```
1 {
2   "offerId": "1"
3 }
```

The response body (Status: 200 OK, Time: 227 ms, Size: 139 KB) is a JSON object with the following structure:

```
1 Valid JobOffer ID
2 JobOffer ID Submitted successfully
3 {
4   "transportMeans": "bicycling",
5   "estimatedDistance": "1.50 miles",
6   "estimatedTime": "8 minutes",
7   "startLocation": "5 Shires Ln, Leicester LE1 4AN, UK",
8   "targetLocation": "University Rd, Leicester LE1 7RH, UK",
9   "RouteInstructionSteps": [
10     {
11       "estimatedDistance": "27 m",
12       "estimatedTime": "1 min",
13       "instruction": "Head <b>south</b> on <b>Shires Ln</b> toward <b>High St</b>"
14     },
15     {
16       "estimatedDistance": "0.3 km",
17       "estimatedTime": "1 min",
18       "instruction": "Turn <b>left</b> onto <b>High St</b>"
19     },
20     {
21       "estimatedDistance": "0.2 km",
22       "estimatedTime": "1 min",
23       "instruction": "Turn <b>right</b> onto <b>Gallowtree Gate</b>"
24     }
25   ]
26 }
```

**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:

The screenshot shows the Swagger Editor interface. On the left, the OpenAPI specification is displayed in a code editor. The specification includes the following details:

- OpenAPI:** 3.0.0
- Info:**
  - Title:** IDUAppUserService API
  - Version:** 1.0.0
- Servers:**
  - url:** http://localhost:8000
- Paths:**
  - /authenticateAndRequestSimilarOffers:**
    - post:**
      - summary:** Authenticate user and request similar job offers.
      - requestBody:**
        - required:** true
        - content:**
          - application/json:**
            - schema:**
              - type:** object
              - required:**
                - username
                - password
                - jobOffers
              - properties:**
                - username:**
                  - type:** string
                - password:**
                  - type:** string
                - jobOffers:**
                  - type:** array
                  - items:**
                    - \$ref:** '#/components/schemas/JobOfferRequest'
          - responses:**
            - 200:**
              - description:** Successfully authenticated and returned similar offers.
              - content:**
                - application/json:**
                  - schema:**
                    - type:** object
                    - properties:**
                      - message:**
                        - type:** string
                        - example:** "User is authenticated successfully."

On the right, the default endpoint details are shown:

      - default**
      - POST /authenticateAndRequestSimilarOffers**
      - Authenticate user and request similar job offers.**
      - Parameters:** No parameters
      - Request body:** required, application/json
      - Example Value:**

```
{
  "username": "string",
  "password": "string",
  "jobOffers": [
    {
      "title": "string",
      "description": "string",
      "location": "string"
    }
  ]
}
```

The screenshot shows the Swagger Editor interface. On the left, the OpenAPI specification is displayed in a code editor. The specification includes the following details:

- OpenAPI:** 3.0.0
- Info:**
  - Title:** IDUAppUserService API
  - Version:** 1.0.0
- Servers:**
  - url:** http://localhost:8000
- Paths:**
  - /authenticateAndRequestSimilarOffers:**
    - post:**
      - summary:** Authenticate user and request similar job offers.
      - requestBody:**
        - required:** true
        - content:**
          - application/json:**
            - schema:**
              - type:** object
              - required:**
                - username
                - password
                - jobOffers
              - properties:**
                - username:**
                  - type:** string
                - password:**
                  - type:** string
                - jobOffers:**
                  - type:** array
                  - items:**
                    - \$ref:** '#/components/schemas/JobOfferRequest'
            - responses:**
              - 200:**
                - description:** Successfully authenticated and returned similar offers.
                - content:**
                  - application/json:**
                    - schema:**
                      - type:** object
                      - properties:**
                        - message:**
                          - type:** string
                          - example:** "User is authenticated successfully."

On the right, the request body details are shown:

          - Request body:** required, application/json
          - Example Value:**

```
{
  "username": "string",
  "password": "string",
  "jobOffers": [
    {
      "title": "string",
      "description": "string",
      "location": "string"
    }
  ]
}
```
          - Responses:**
            - 200:**
              - description:** Successfully authenticated and returned similar offers.
              - content:**
                - application/json:**
                  - schema:**
                    - type:** object
                    - properties:**
                      - message:**
                        - type:** string
                        - example:** "User is authenticated successfully."

Below the responses, the curl command is shown:

```
curl -X 'POST' \
  'http://localhost:8000/authenticateAndRequestSimilarOffers' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "username": "string",
    "password": "string",
    "jobOffers": [
      {
        "title": "string",
        "description": "string",
        "location": "string"
      }
    ]
  }'
```

Swagger Editor

```
50- description: Method not allowed.
51-
52- /submitSelectedJob:
53-   post:
54-     summary: Submit a selected job offer.
55-     requestBody:
56-       required: true
57-       content:
58-         application/json:
59-           schema:
60-             type: object
61-             required:
62-               - offerId
63-             properties:
64-               offerId:
65-                 type: string
66-     responses:
67-       '200':
68-         description: Job offer successfully submitted with route details.
69-         content:
70-           application/json:
71-             schema:
72-               $ref: '#/components/schemas/RouteDetailsResponse'
73-       '400':
74-         description: Bad request (e.g., invalid input data).
75-       '404':
76-         description: OfferId does not correspond to a valid job offer.
77-       '405':
78-         description: Method not allowed.
79-
80- components:
81-   schemas:
82-     JobOfferRequest:
83-       type: object
84-       properties:
85-         title:
86-           type: string
87-         description:
88-           type: string
89-         location:
90-           type: string
```

POST /submitSelectedJob Submit a selected job offer.

Parameters

No parameters

Request body **required** application/json

Example Value Schema

```
{
  "offerId": "string"
}
```

Responses

| Code | Description  | Links    |
|------|--|----------|
| 200  | Job offer successfully submitted with route details. | No links |

Swagger Editor

```
51-
52- /submitSelectedJob:
53-   post:
54-     summary: Submit a selected job offer.
55-     requestBody:
56-       required: true
57-       content:
58-         application/json:
59-           schema:
60-             type: object
61-             required:
62-               - offerId
63-             properties:
64-               offerId:
65-                 type: string
66-     responses:
67-       '200':
68-         description: Job offer successfully submitted with route details.
69-         content:
70-           application/json:
71-             schema:
72-               $ref: '#/components/schemas/RouteDetailsResponse'
73-       '400':
74-         description: Bad request (e.g., invalid input data).
75-       '404':
76-         description: OfferId does not correspond to a valid job offer.
77-       '405':
78-         description: Method not allowed.
79-
80- components:
81-   schemas:
82-     JobOfferRequest:
83-       type: object
84-       properties:
85-         title:
86-           type: string
87-         description:
88-           type: string
89-         location:
90-           type: string
```

Responses

| Code | Description  | Links    |
|------|--|----------|
| 200  | Job offer successfully submitted with route details. | No links |
| 400  | Bad request (e.g., invalid input data).              | No links |
| 404  | OfferId does not correspond to a valid job offer.    | No links |

Media type: application/json

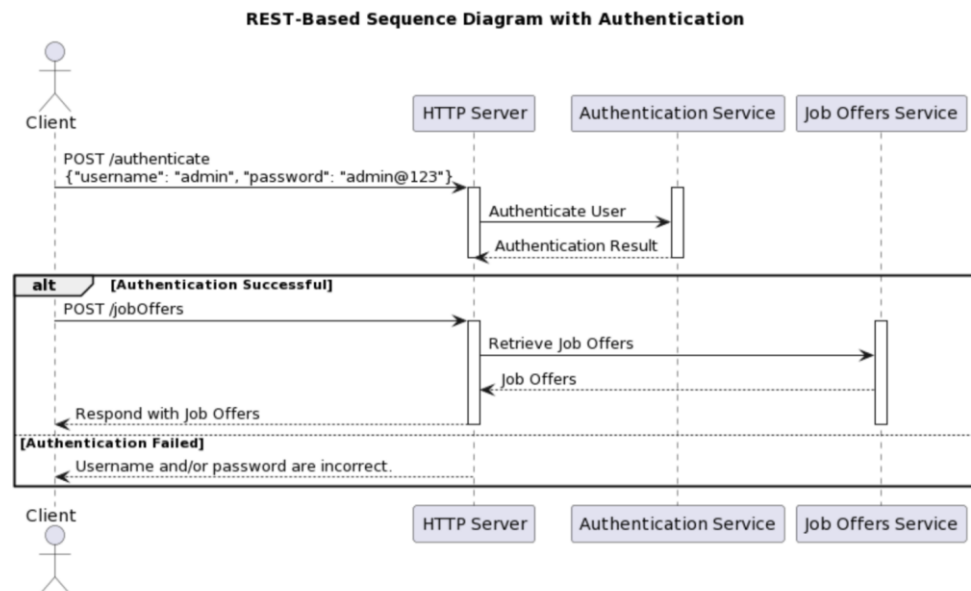
Example Value Schema

```
{
  "status": "Valid JobOffer ID",
  "transportMeans": "bicycling",
  "estimatedDistance": "11.50 miles",
  "estimatedTime": "8 minutes",
  "startLocation": "5 Shires Ln, Leicester LE1 4AN, UK",
  "targetLocation": "University Rd, Leicester LE1 7RH, UK",
  "RouteInstructionSteps": [
    {
      "estimatedDistance": "string",
      "estimatedTime": "string",
      "instruction": "string"
    }
  ]
}
```

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:**

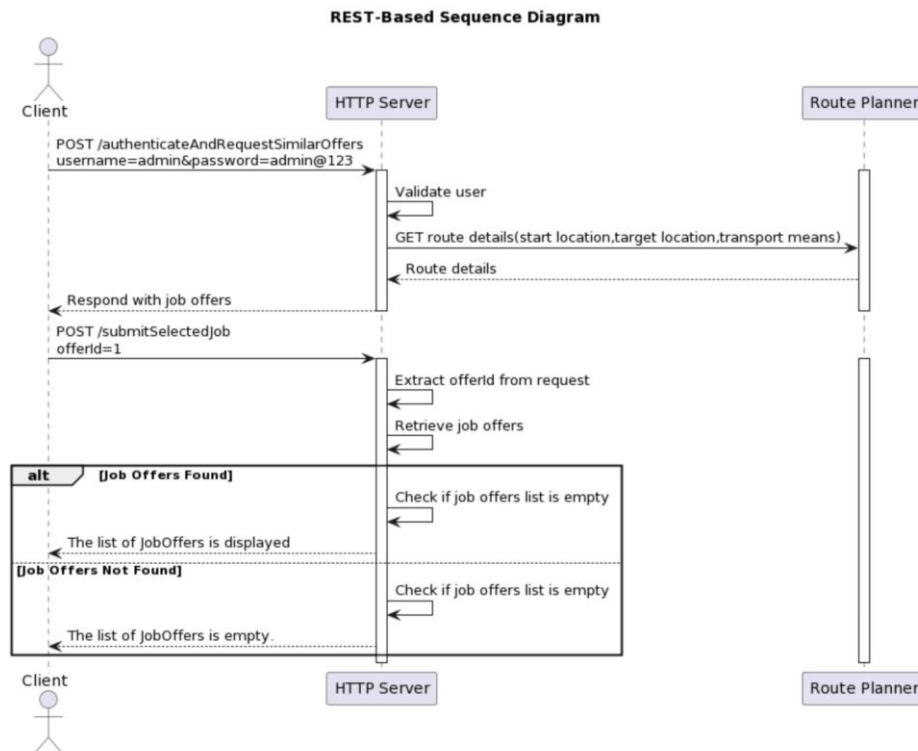
- The Authentication Service verifies the provided credentials and responds to the server with the authentication result.

**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."

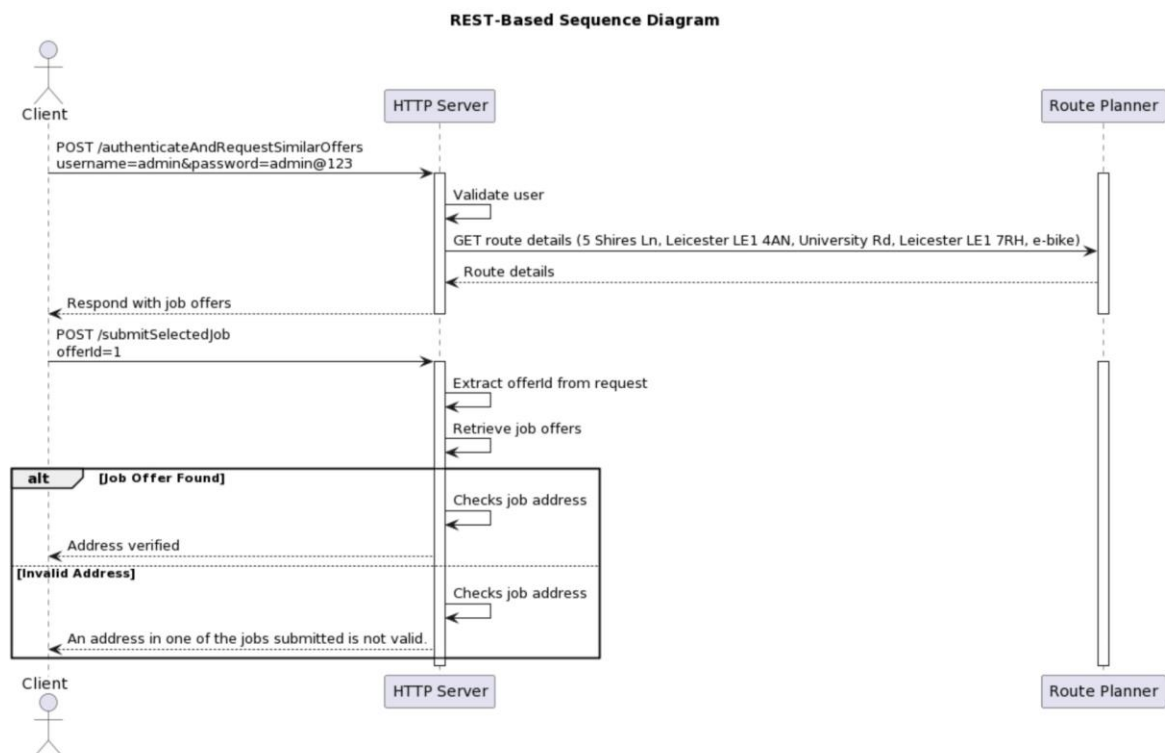
#### Invalid Address Detected:

- 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:

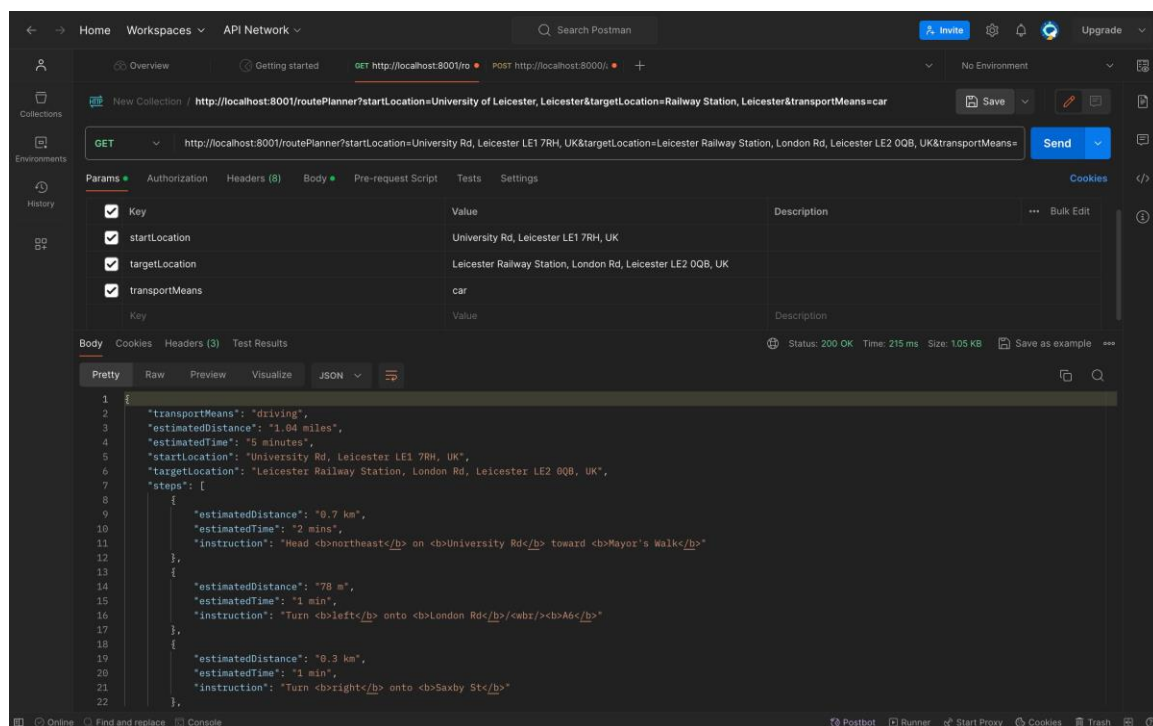
- If an address in one of the submitted jobs is not valid:
- The server checks the address of the job offer.
- The server responds to the client with the message "An address in one of the jobs submitted is not valid."

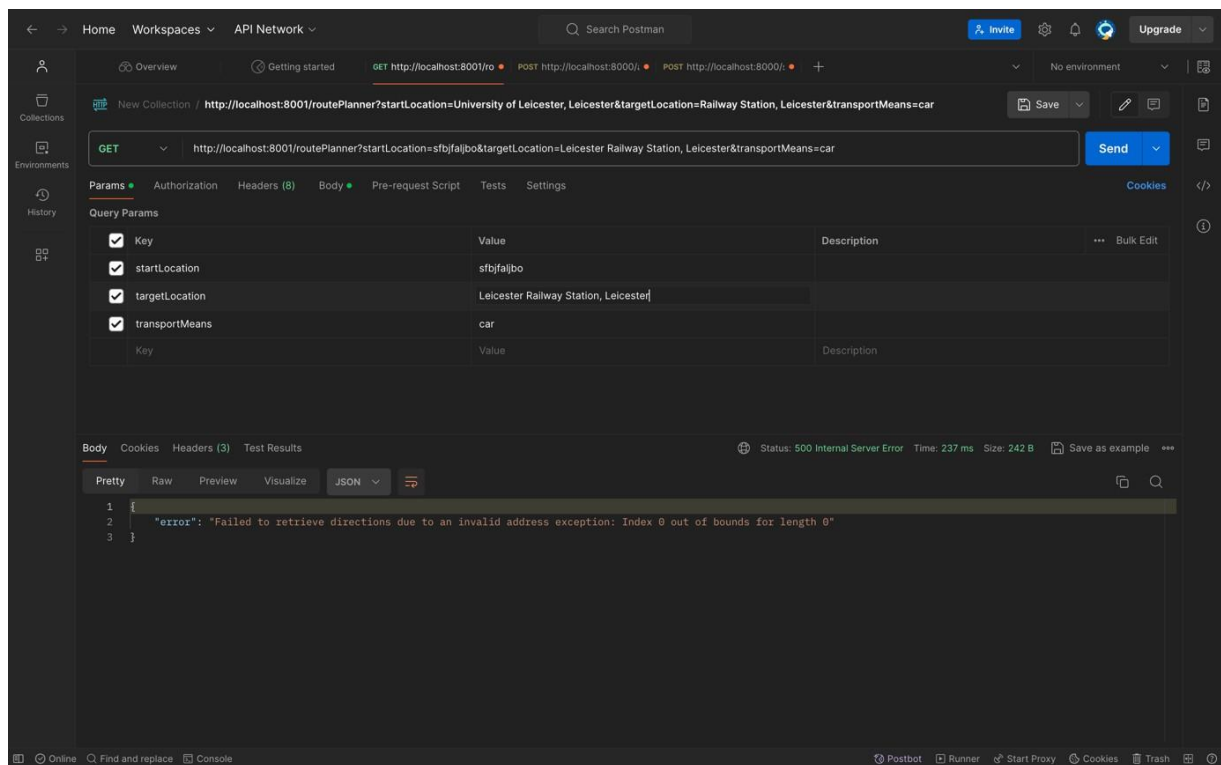
### Process Completion:

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

## 4. Postman Validation Screenshots:

### IDUServiceRoutePlanner Validation:





## IDUAppDUService Vaildation:

