GlobalLogic®
A Hitachi Group Company

GloRepay Portal

Spring HandsOn Assessment

GlobalLogic®
A Hitachi Group Company

Assessment Flavour - **Java Developer**

Complexity - Medium

Target Band - Band x / Band 1 / Band 2

Title of the Project -

Time: 3 hours

Topics Covered:

Technology	**Topics**
Spring Web	REST API
Hibernate	Spring Data JPA
Core Java	Exception Handling

SOFTWARE REQUIREMENTS

- Java Latest Version
- **Any IDE that supports Java Application Development**
    - Intelij or STS

GlobalLogic®
A Hitachi Group Company

TABLE OF CONTENT

PURPOSE

The GloRepay is an Employee Reimbursement System and is a REST API project to be developed using Java Spring Boot. The system allows employees to manage their expense reimbursements by providing various reimbursement modes and tracking their expenditures.

PROBLEM STATEMENT

You are a backend developer working on the Employee Reimbursement System project. The project aims to develop a RESTful API using the Spring Boot framework. The Employee Reimbursement System is a web-based application that allows employees to manage their reimbursement claims for various expenses.

PROJECT IMPLEMENTATION

Database Design**:** Application schema and tables

Note: You can ignore the create table command if your application is configured to auto-create tables using Hibernate/JPA properties.

Table Structure with data types

Employee		**Description**
id	Long	It Must be of Integer data type only. It is the primary ID that is unique for each user. Example: 1

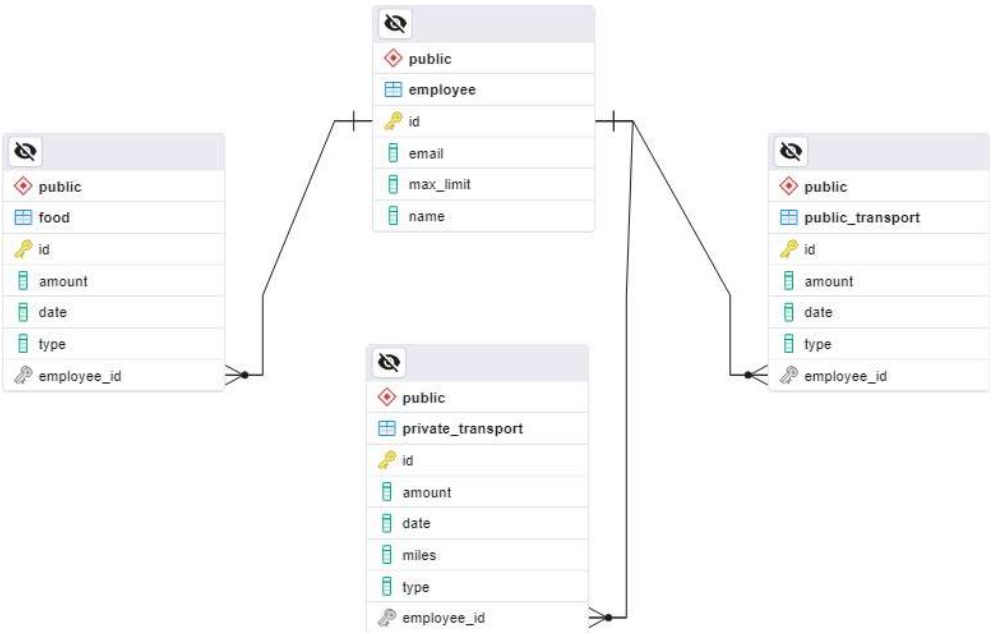
name	String	It must be of String data type. It is the name chosen by the user for the profile. Example: "JohnDoe"
email	String	It must be of String data type. It is the user's email address used for communication and login. Example: "johndoe@example.com"
maxLimit	Integer	It must be of Integer data type. It is the maximum amount allowed for reimbursement. Example: "password123"

PublicTransport		**Description**
id	Long	It must be of Integer data type only. It is the primary ID that is unique for each product. Example: 1
type	String	It must be of String data type. It is the name of the product. Example: "BUS"
date	Date	It must be of Date data type. It is the date on which the expense is made for public transport. Example: "yyyy-MM-dd"
amount	Integer	It must be of Integer data type. It is the expense that employees bore for the public transport. Example: 600

PrivateTransport		Description
id	Long	It must be of Integer data type only. It is the primary ID that is unique for each product. Example: 1
type	String	It must be of String data type. It is the name of the product. Example: "BUS"
date	Date	It must be of Date data type. It is the date on which the expense is made for public transport. Example: "yyyy-MM-dd"
miles	Integer	It must be of Integer data type. It is the distance traveled by the employee. Example: 10
amount	Integer	It must be of Integer data type. It is the expense that employees bore for the public transport. Example: 600

Food		Description
id	Long	It must be of Integer data type only. It is the primary ID that is unique for each product. Example: 1
type	String	It must be of String data type. It is the name of the product. Example: "LUNCH"
date	Date	It must be of Date data type. It is the date on which the expense is made for public transport. Example: "yyyy-MM-dd"
amount	Integer	It must be of Integer data type. It is the expense that employees bore for the public transport. Example: 600

Entity Relationship



Note:

There is bidirectional mapping which is one-to-many and many-to-one for the Entities as shown in the diagram.

PROJECT BACKEND IMPLEMENTATION DETAILS

Application flow with functionality:

Note: Define appropriate custom exceptions for invalid values and handle the exceptions with appropriate error messages.

API Details:

Employee Entity			
API Endpoints	**HttpMethod**	**Description**	**Status Code**
/api/employee	POST	Create a new employee.	201 Created
		Throws an exception with an informative message if the data is invalid.	400 Bad Request
/api/employee/{id}	GET	Retrieve details of a specific employee.	200 OK
		Throws an exception with an informative message if the employee ID is invalid.	404 Not Found

PublicTransport Enttiy			
API Endpoints	**HttpMethod**	**Description**	**Status Code**
/api/employee/{id}/publicTransport	POST	Create a new public transport entry	201 Created
		Throws an exception with an informative message if the data is invalid.	400 Bad Request
/api/employee/{id}/publicTransport/{publicTransport}	GET	Retrieve details of a specific employee public transport	200 OK
		Throws an exception with an informative message if the employee ID or	404 Not Found

		public transport id is invalid.	

PrivateTransport Enttiy			
API Endpoints	**HttpMethod**	**Description**	**Status Code**
/api/employee/{id}/privateTransport	POST	Create a new private transport entry	201 Created
		Throws an exception with an informative message if the data is invalid.	400 Bad Request
/api/employee/{id}/publicTransport/{privateTransport}	GET	Retrieve details of a specific employee private transport	200 OK
		Throws an exception with an informative message if the employee ID or private transport id is invalid.	404 Not Found

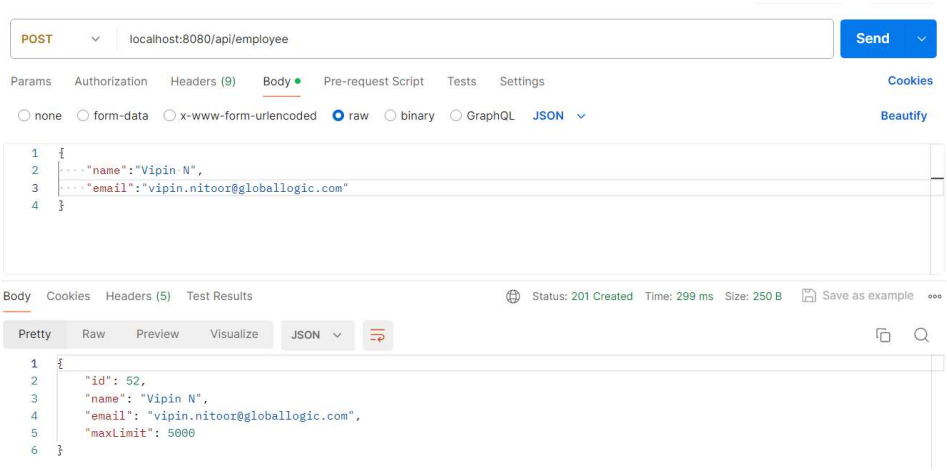
Food Enttiy			
API Endpoints	**HttpMethod**	**Description**	**Status Code**
/api/employee/{id}/food	POST	Create a new food entry	201 Created
		Throws an exception with an informative message if the data is invalid.	400 Bad Request
/api/employee/{id}/food/{foodId}	GET	Retrieve details of a specific employee food expense	200 OK
		Throws an exception with an informative message if the employee ID or food id is invalid.	404 Not Found

GlobalLogic®
A Hitachi Group Company

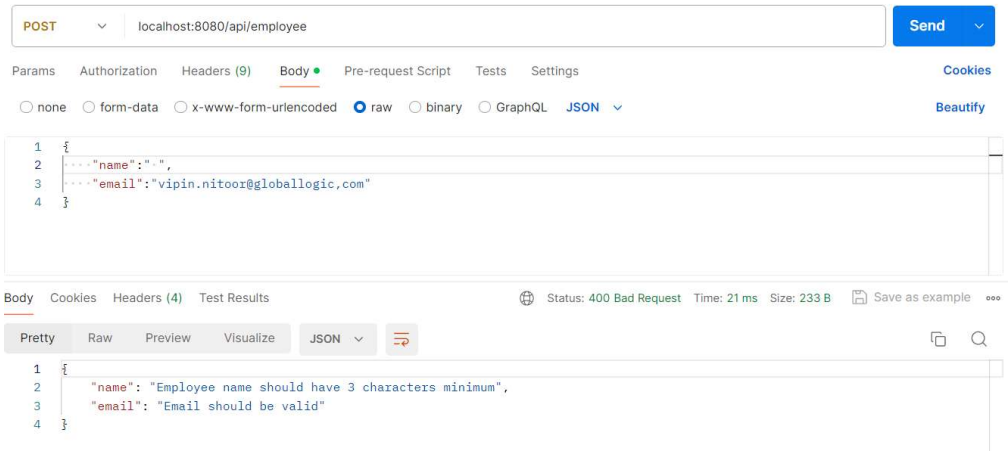
Postman Testing

POST operation for Employee Entity

Case 1 - The user provided valid data

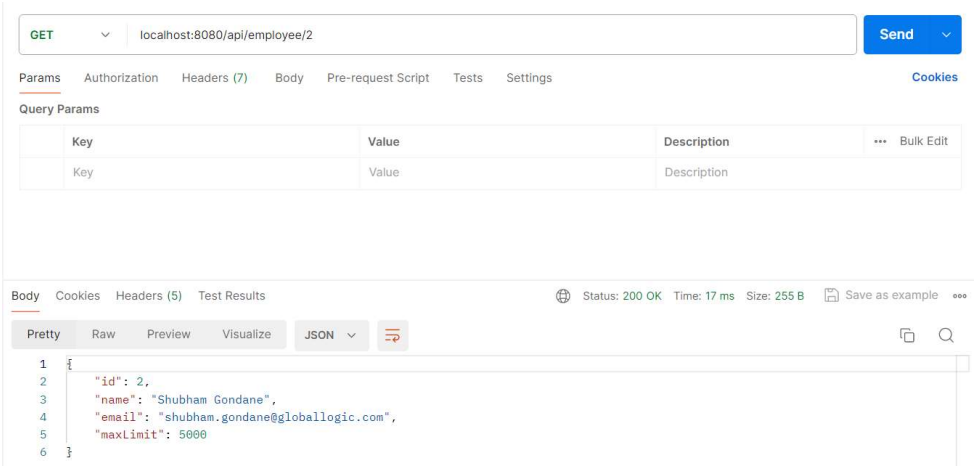


Case 2 - User provided the invalid data

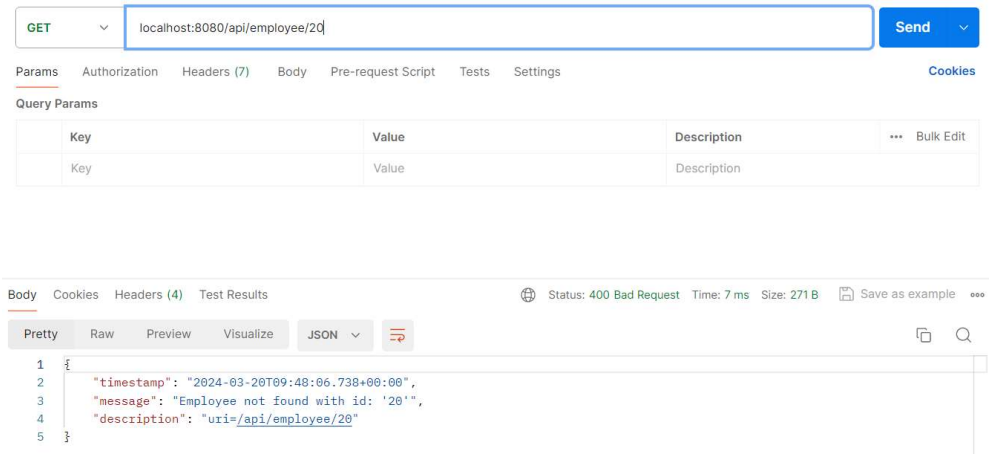


GET Operation for Employee Entity

Case 1 - The user provided a valid Id

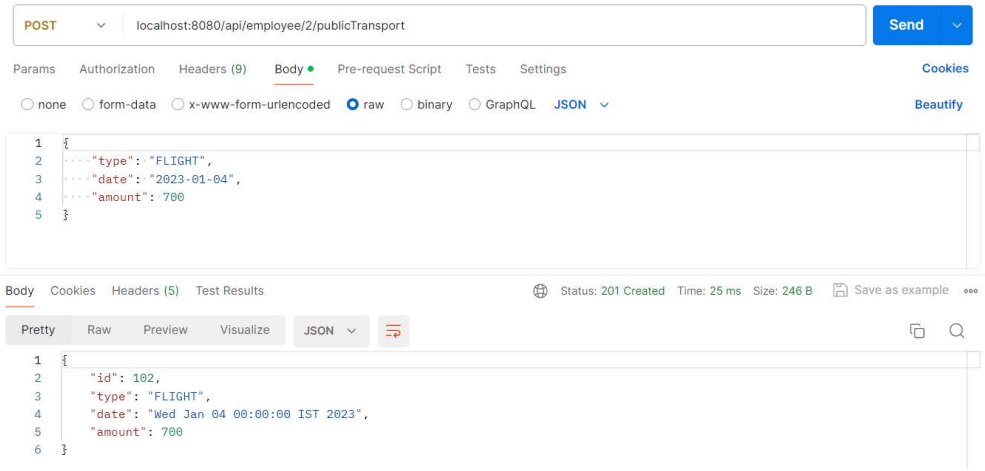


Case 2 - User-Provided the invalid Id

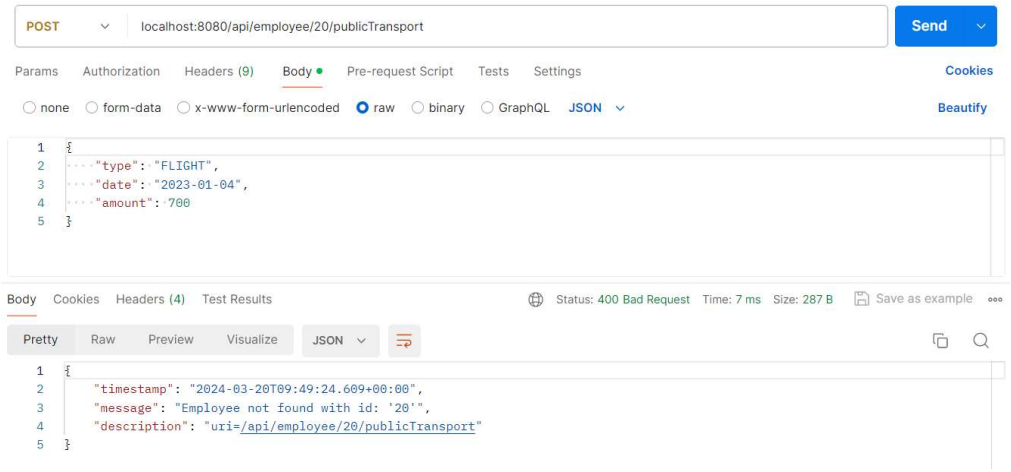


POST Operation for PublicTransport Entity

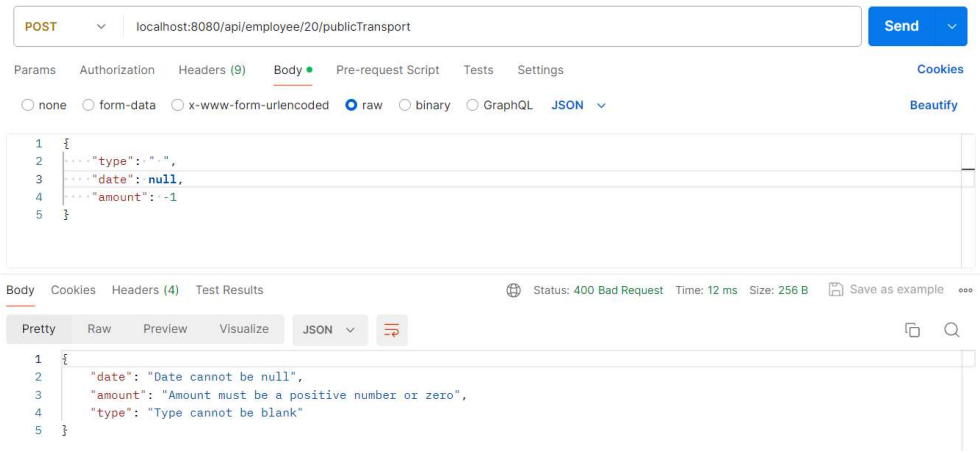
Case 1 - The user provided a valid employee Id and valid data



Case 2 - The user provided the invalid employee Id with valid data

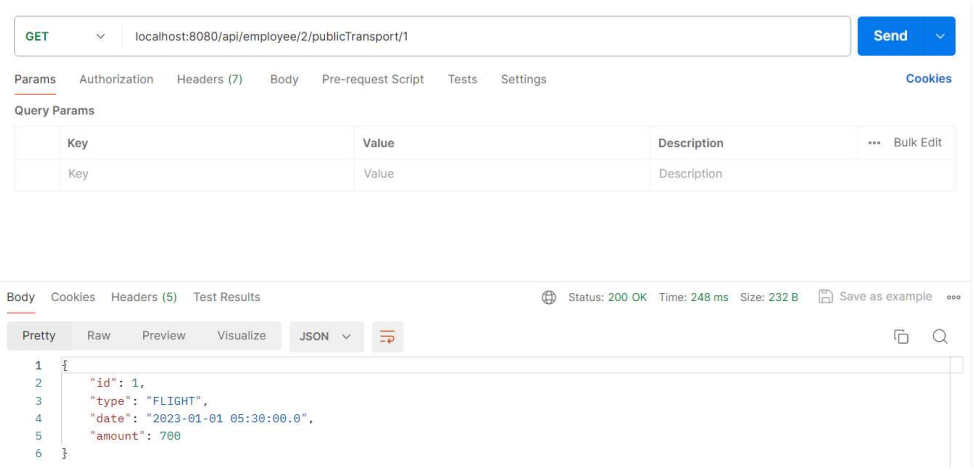


Case 3 - User provided the invalid data in the request body

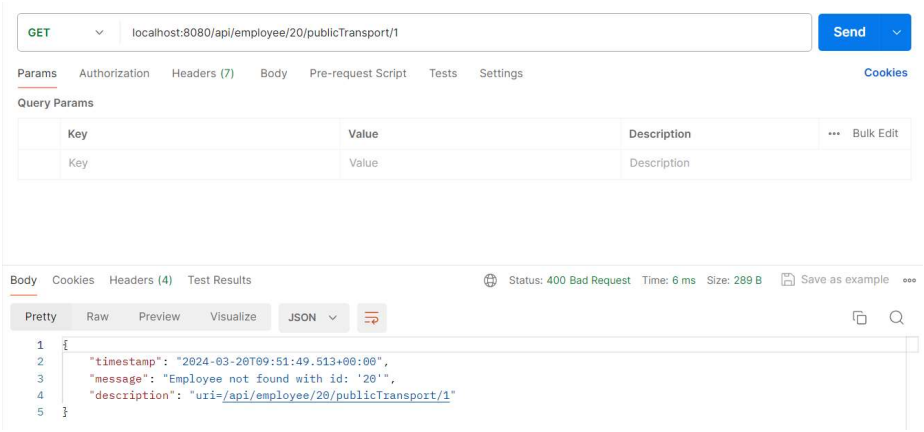


GET Operation for PublicTransport Entity

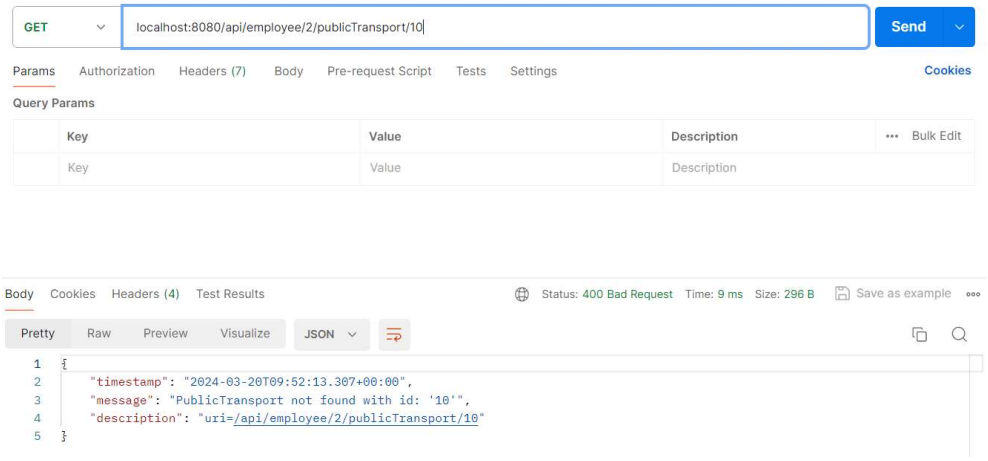
Case 1 - The user provided a valid employee Id and public transport Id



Case 2 - The user provided an invalid employee Id



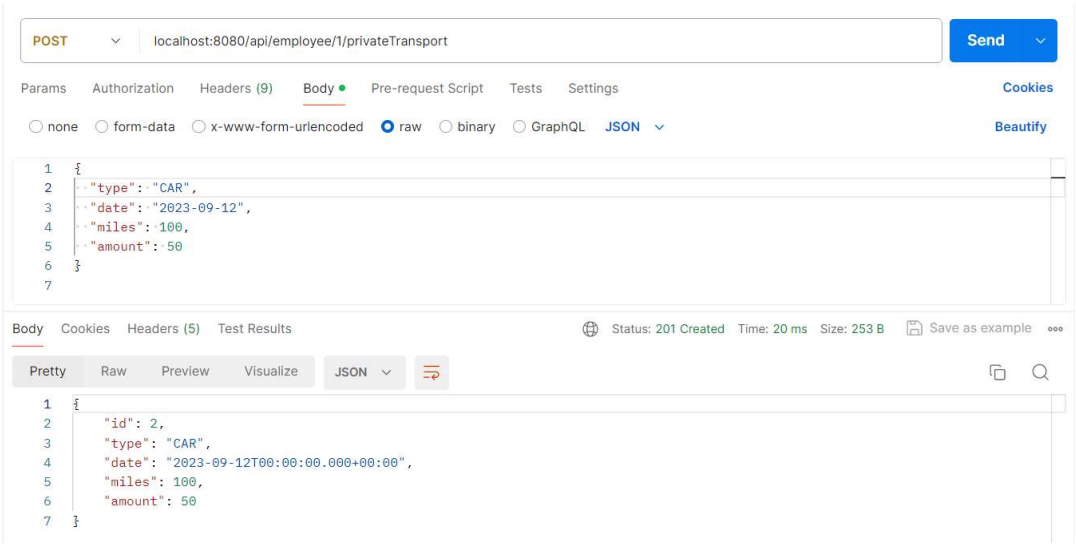
Case 3 - The user provided an invalid publicTransportId



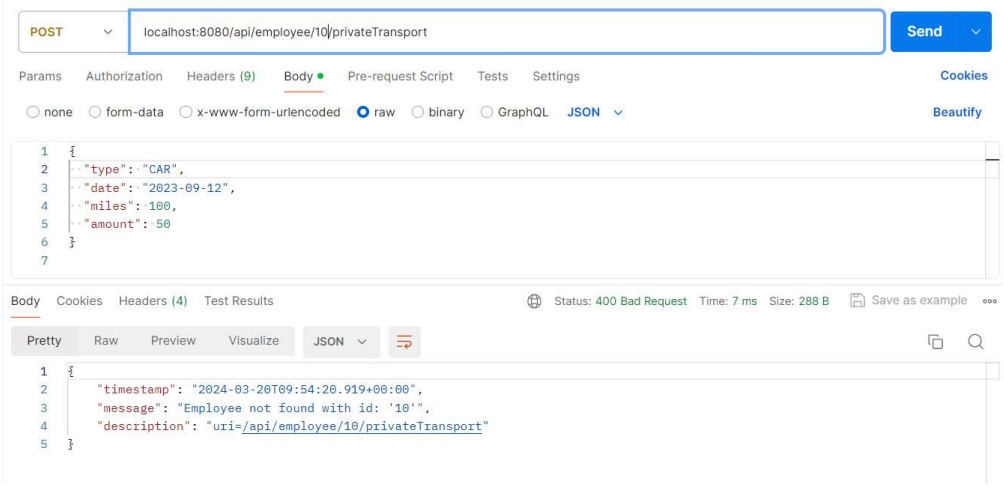
GlobalLogic®
A Hitachi Group Company

POST Operation for PrivateTransport Entity

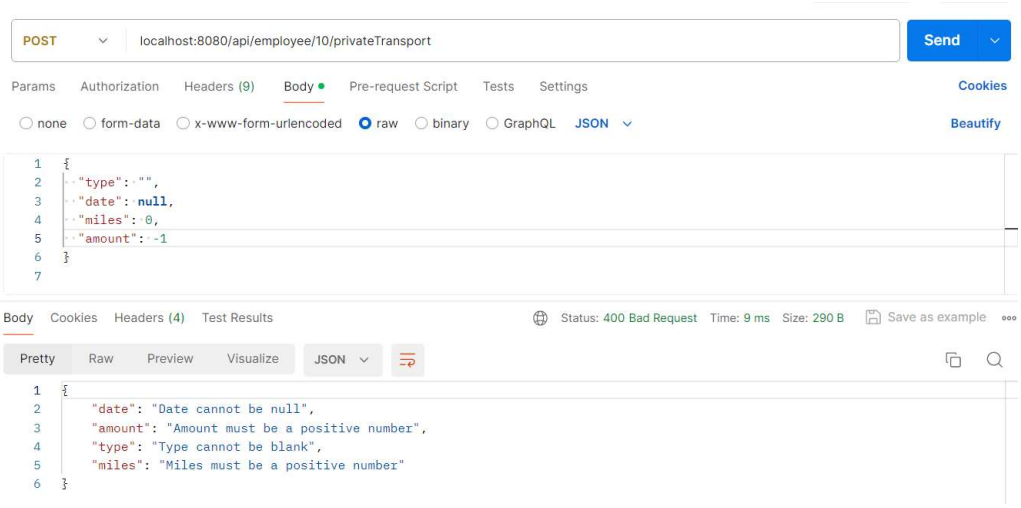
Case 1 - The user provided a valid employee Id and valid data



Case 2 - The user provided the invalid employee Id with valid data

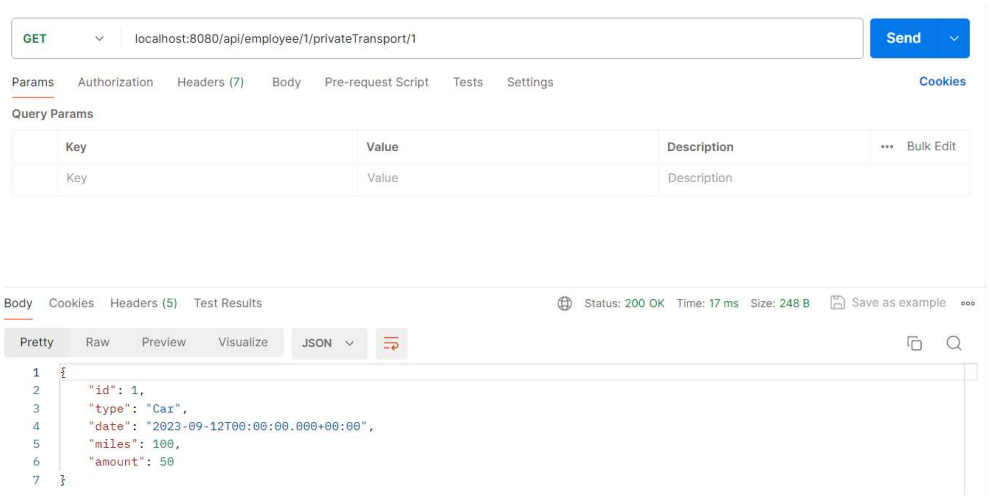


Case 3 - User provided the invalid data in the request body

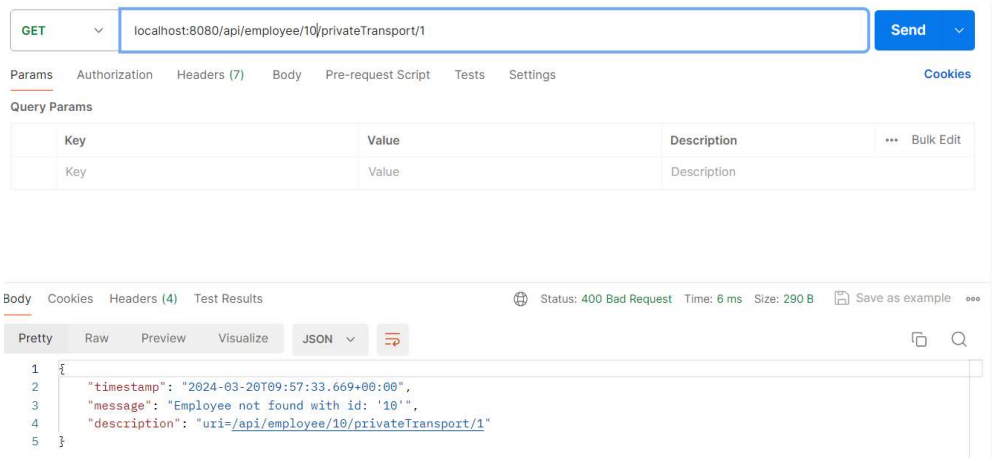


GET Operation for PrivateTransport Entity

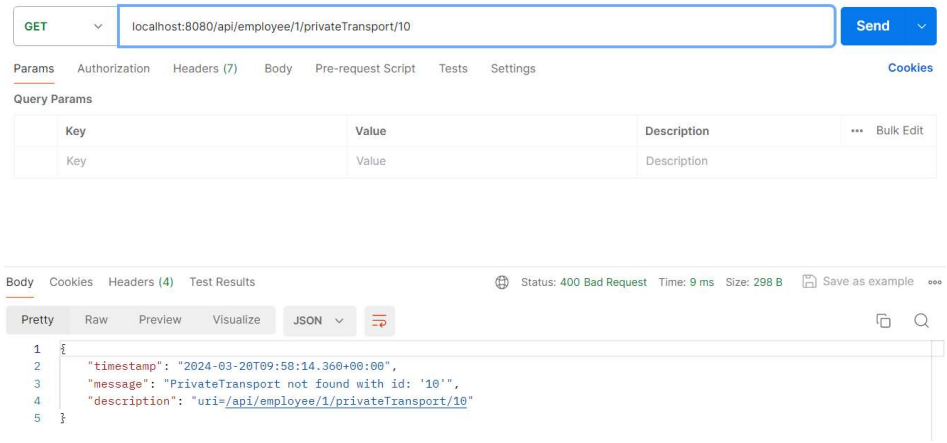
Case 1 - The user provided a valid employee Id and private transport Id



Case 2 - The user provided an invalid employee Id



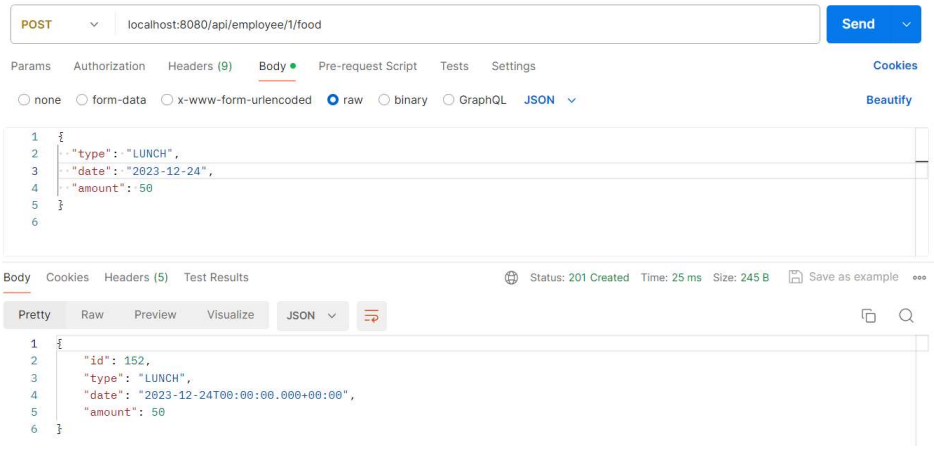
Case 3 - The user provided an invalid privateTransporttId



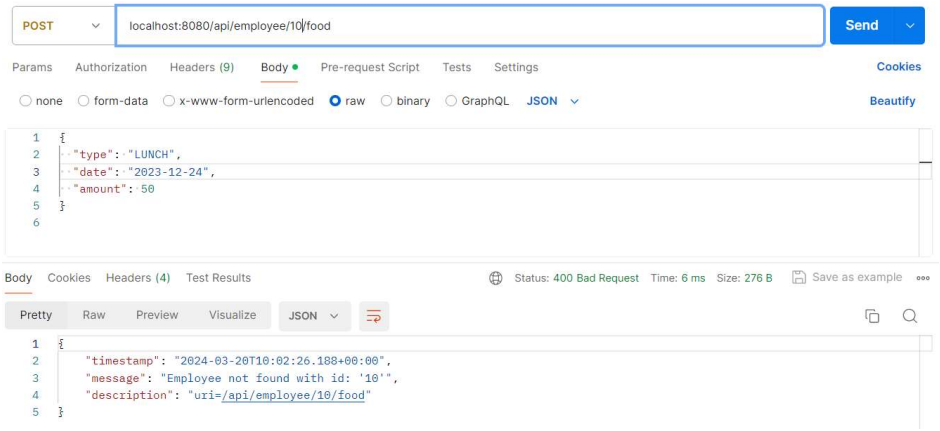
GlobalLogic®
A Hitachi Group Company

POST Operation for Food Entity

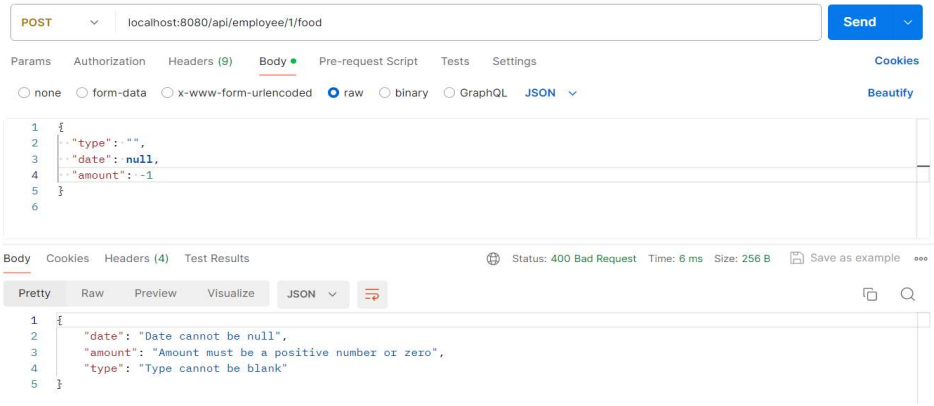
Case 1 - The user provided a valid employee Id and valid data



Case 2 - The user provided the invalid employee Id with valid data

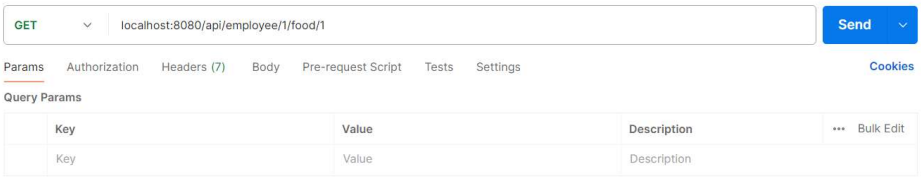


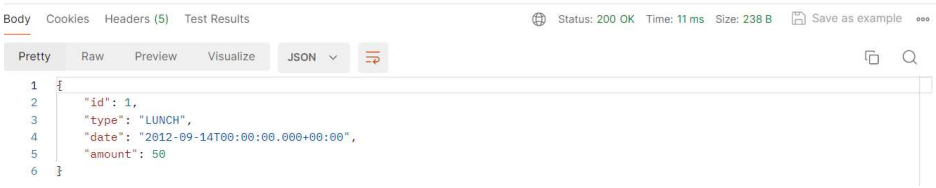
Case 3 - User provided the invalid data in the request body



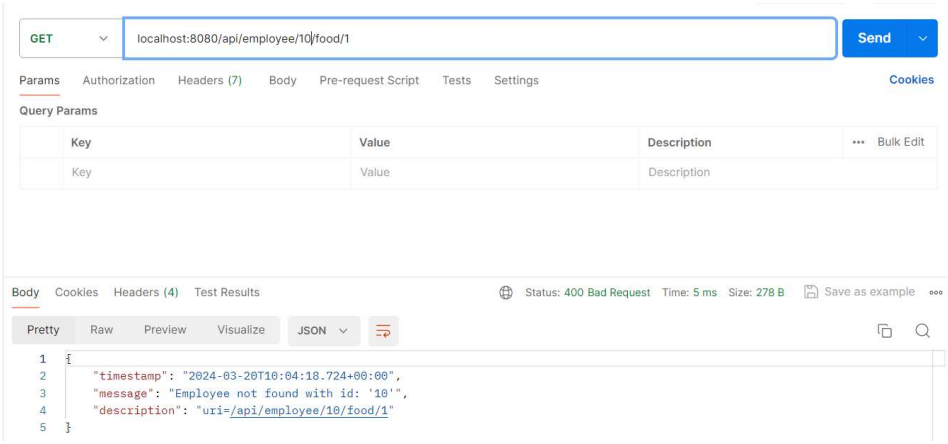
GET Operation for Food Entity

Case 1 - The user provided a valid employee Id and private transport Id

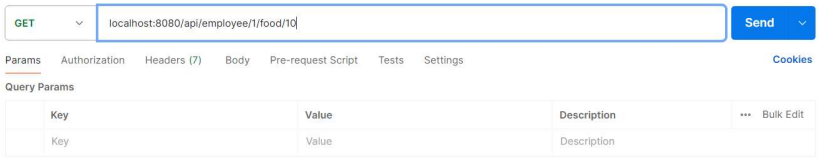


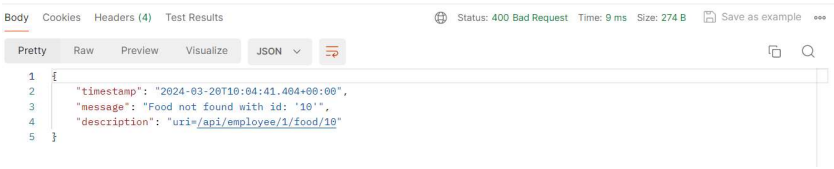


Case 2 - The user provided an invalid employee Id



Case 3 - The user provided an invalid privateTransportId





Note

- Use the Utility class provided in the codebase for mapping DTO to Entity and vice versa
- Use the Exception handling utility class to handle the exception in the application as and when needed

GlobalLogic
A Hitachi Group Company

RUNNING THE BACKEND APPLICATION

Test the application by running the main file in the starter code.

Note: Main file code is commented. Uncomment the code, test the application, and comment it back before submitting the assessment.

EVALUATION METRICS

Sl. No.	Assessing Parameter	Marks
1	Creating the well-defined Controller Layer	5
2	Creating the JPA Entity with appropriate Entity Mapping	5
3	Creating the Dto class for all the entities	5
4	Handling the invalid data in the Dto class using the Hibernate Validator	5
5	Creating the Repository layer	5
6	Creating the Business Logic for all the entities with proper exception handling.	20
7	Coding standards and best practices	5
Total Marks		50 Marks