Lab 03

# Introduction

This lab introduces students to essential development tools, including project management platforms, browser-based debugging tools, and API documentation tools like Swagger and Postman. Additionally, we will dive into the fundamentals of RESTful APIs, focusing on how they work, how to interact with them, and best practices for their integration. The lab will also cover the creation and use of functional documentation to support and streamline the development process.

# Objectives

1. Learn to effectively utilize Project Management tools, Chrome Developer Tools, Swagger, and Postman for debugging and API documentation.
2. Understand the fundamentals of RESTful APIs, including how they function, how to interact with them using HTTP methods, and how to work with API endpoints and data formats.
3. Develop and implement functional documentation to ensure clear communication and understanding of software requirements.
4. Understand and apply best coding practices throughout the development process.

# Description

1. **Project Management Tools:**

Project management tools are fundamental for organizing tasks, tracking progress, facilitating communication within development teams. These tools, such as JIRA and Trello, help streamline workflows by enabling teams to assign tasks, set deadlines, and monitor project status. Understanding how to effectively use these tools fosters accountability and ensures that projects remain on schedule, ultimately leading to successful outcomes.

1. **Chrome Developer Tools:**

Chrome Developer Tools are invaluable for web developers, providing features that allow for real-time inspection and debugging of web applications. The ability to inspect elements, view network requests, and analyze performance is crucial for identifying and resolving issues quickly.

## 3. RESTful API

A RESTful API is an interface that allows different applications to communicate with each other over the web by adhering to a set of constraints known as REST (Representational State Transfer). These APIs typically use HTTP (Hypertext Transfer Protocol) to send and receive data. RESTful APIs are scalable, stateless, and designed to work with the architecture of the web.

## 4. How RESTful APIs Function

RESTful APIs follow a client-server architecture, where:

* Client: The application that initiates a request (e.g., a web browser, mobile app).
* Server: The system that processes the request and sends back a response (e.g., a web server or database).

Each request made by the client to the server is independent (stateless), meaning that the server does not store any session data about the client between requests.

Middleware: Middleware acts as an intermediary layer that processes requests between the client and server. It can handle tasks like authentication, logging, error handling, or data transformation before the server responds to the client or passes the request to the –next stage of processing.

## 5. Components of a REST API

1. API Client
   * The API client initiates the request to the API.
   * Assembles and sends API requests to the API server based on user actions or external events.
2. API Request
   * *Endpoint:* The URL that specifies the resource the client is interacting with (e.g., /products in an e-commerce API).
   * *Method:* The type of operation the API should perform, such as GET (retrieve), POST (create), PUT/PATCH (update), or DELETE (remove).
   * *Parameters:* Variables included in the request to specify details, such as filters or instructions (e.g., a color parameter to filter products).
   * *Request Headers:* Key-value pairs that provide metadata, such as content format or authentication credentials.
   * *Request Body:* Contains the actual data sent by the client when creating or modifying resources, typically in formats like JSON or XML.
3. API Server
   * Processes the API request by handling authentication, validating data, interacting with the database, and returning the response.
   * *Database Interaction:* While the database itself is not part of the API, the API server serves as an intermediary, retrieving and manipulating data from the database.
4. API Response Components
   * *Status Code:* Indicates the outcome of the request, such as 200 OK (success), 201 Created, or 404 Not Found.
   * *Response Headers:* Provide additional metadata about the response (e.g., cache instructions or session cookies).
   * *Response Body:* Contains the data or content returned to the client, often in JSON format, and may include error messages if the request failed.

1. **Postman for API Testing:**

Postman simplifies the process of testing APIs by allowing developers to send requests and validate responses seamlessly. This is crucial for ensuring that APIs meet both functional and performance requirements

1. **API Documentation with Swagger:**

Swagger offers a standardized approach to API documentation, enabling developers to create clear and interactive documentation for their services.

## 8. Functional Documentation

Functional documentation in software engineering refers to a detailed description of a software system's features, behavior, and functionalities, specifying what the system should do without defining how it should be implemented. It outlines requirements, use cases, inputs, outputs, and interactions from a user’s perspective, serving as a guide for developers and stakeholders to ensure that the system meets the intended objectives

User Story:

As a customer, I want to be able to book a product from an online store so that I can ensure it is reserved for me and pick it up or have it delivered at a later time.

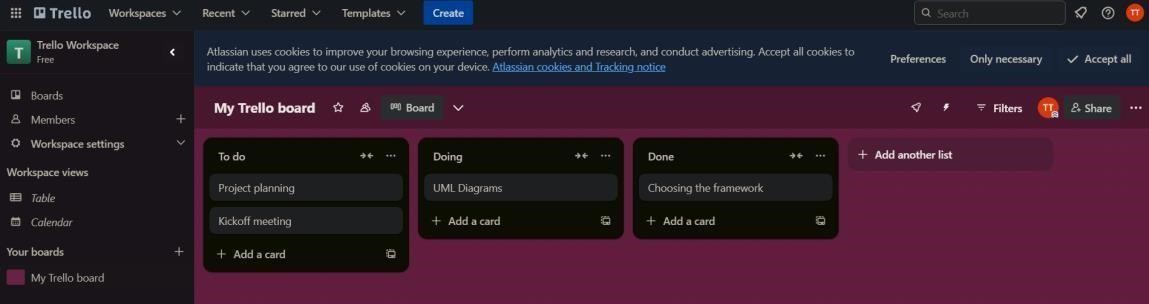
Functional Requirements:

* **FR1: Product Booking:** The system shall allow users to select a product and add it to their booking list, ensuring the product is reserved for a specific time period.
* **FR2: Confirmation of Booking:** The system shall send a booking confirmation to the user via email or SMS, including details of the product, reservation period, and pickup or delivery options.

# Document Examples

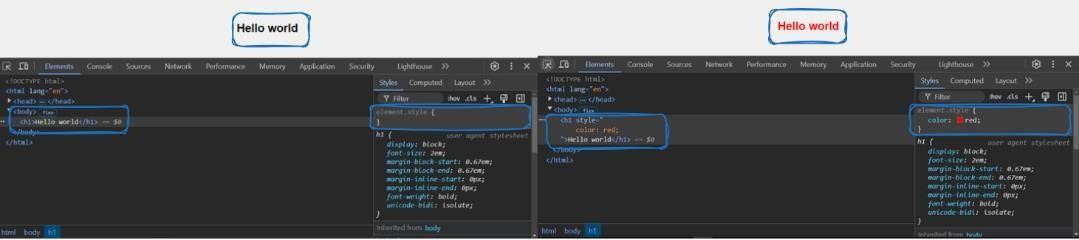
## 1- Project Management Tools [- Trello](https://trello.com/)

*Trello board with 3 lists and some cards to help you organize tasks by their current state, providing a clear visual overview of the workflow*

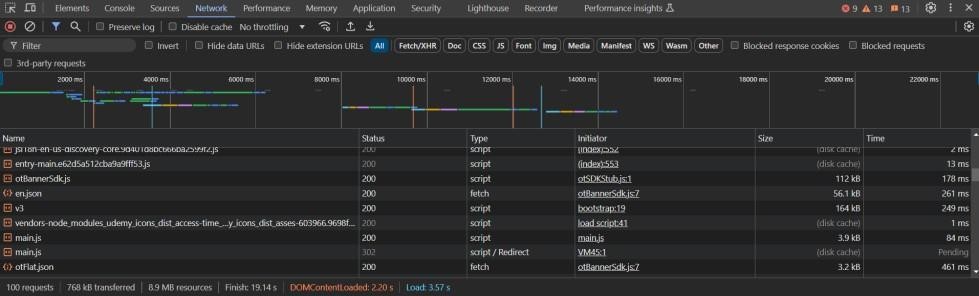


## 2- Browser-Based Debugging Tools [– Chrome DevTool](https://developer.chrome.com/docs/devtools)

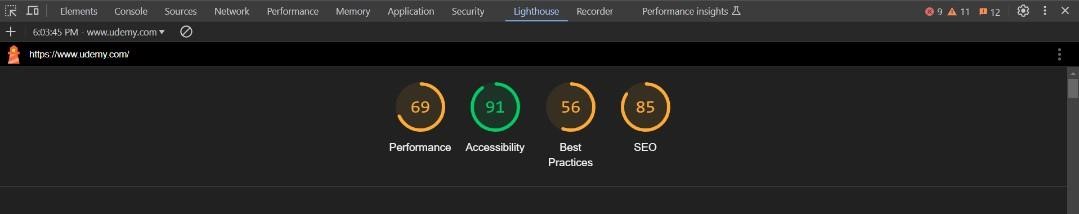
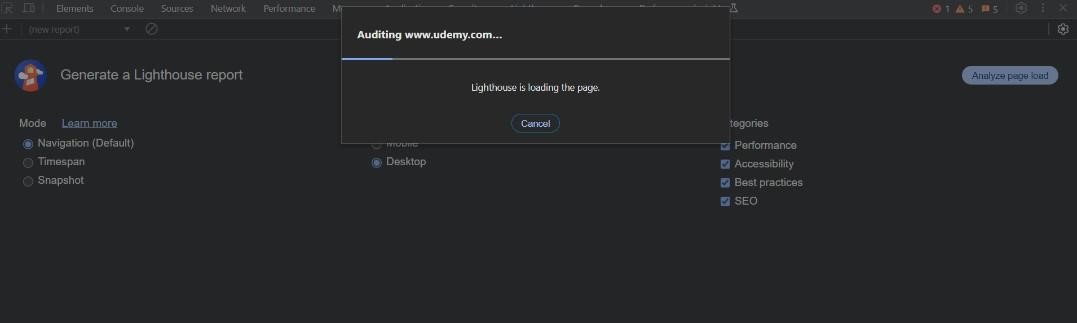
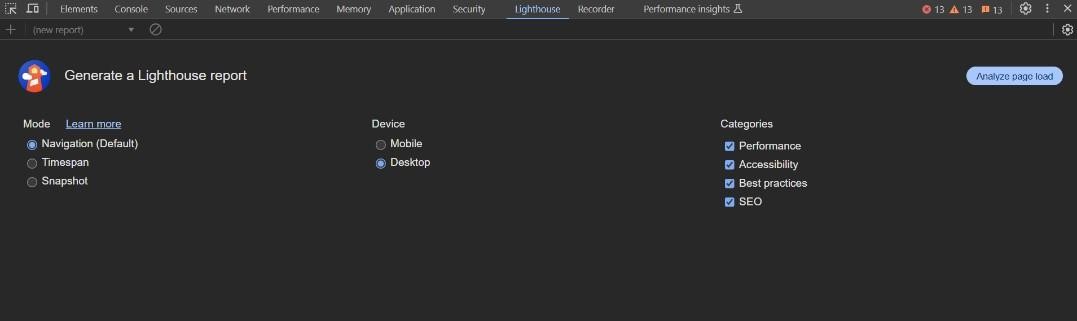
*Real-time code inspection and debugging*



*In-Depth Network Traffic Analysis*



*Comprehensive Performance Audits with Lighthouse*



## 3- Creating Simple RESTful API using Flask

***Create a basic server*** *using Flask.*

*initializing a Flask app, defines a hardcoded token ("mysecrettoken"), and creates an empty in-memory list for storing todo items. The `authenticate\_token` function acts as middleware, checking for a valid token in the Authorization header before processing any request. If the token is missing or incorrect, it responds with a 401 Unauthorized error.*

A computer screen with text on it

Description automatically generated

***GET*** *endpoint to fetch all tasks*

A screen shot of a computer

Description automatically generated

***POST*** *endpoint to create a new to-do item, provide `title` and optionally `completed` in the request body as JSON*

A computer screen with text

Description automatically generated

***PUT*** *endpoint to update an existing to-do item with the specified `id`, provide updated `title` and/or `completed` in the request body as JSON*

A computer screen shot of text

Description automatically generated

***DELETE*** *endpoint to remove an existing todo item with the specified `id`*

A computer code on a black background

Description automatically generated

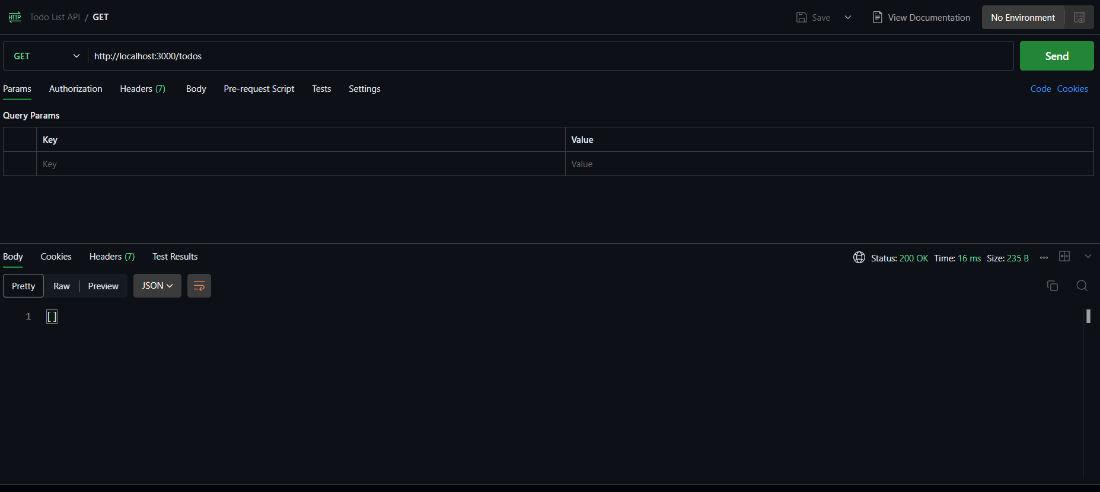
***Middleware*** *to handle 404 errors by returning a JSON response with an error message*

**A black screen with white text

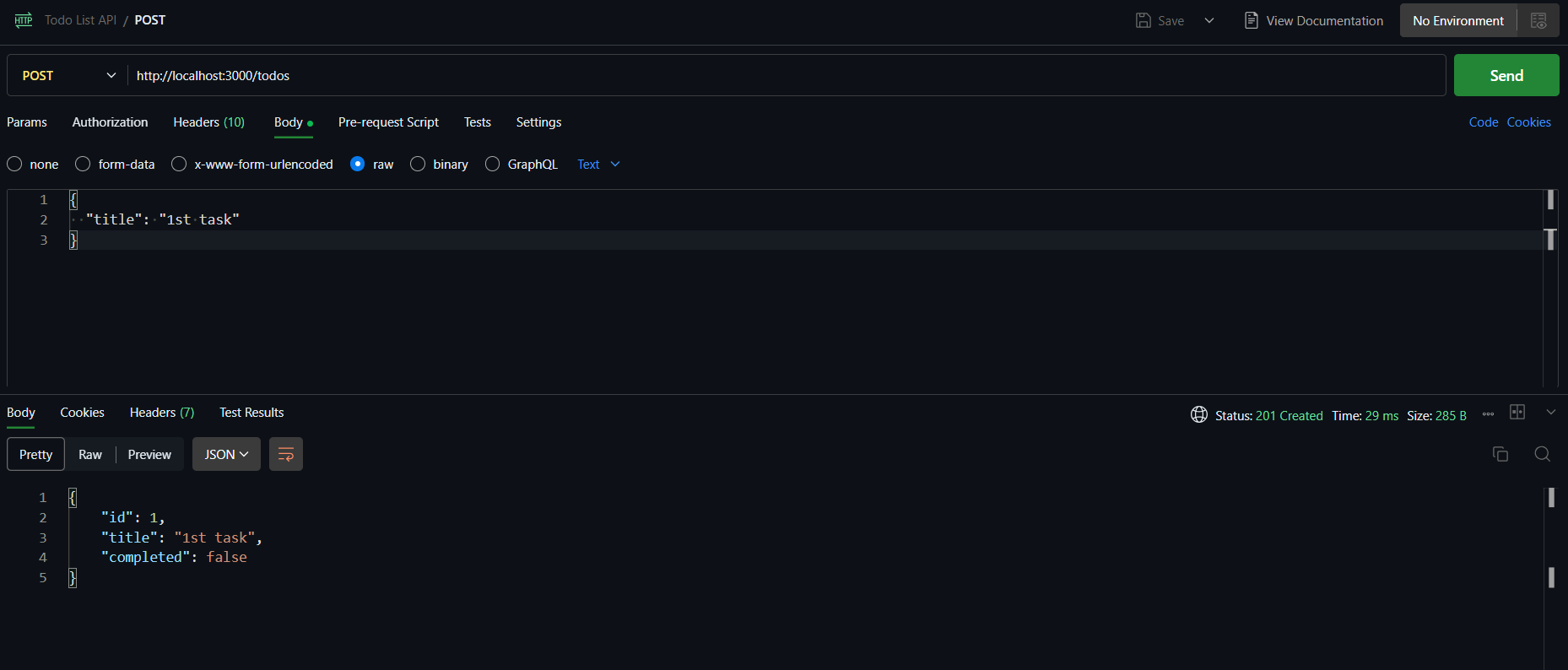
Description automatically generated**

## 4- [Postman f](https://www.postman.com/)or API Testing

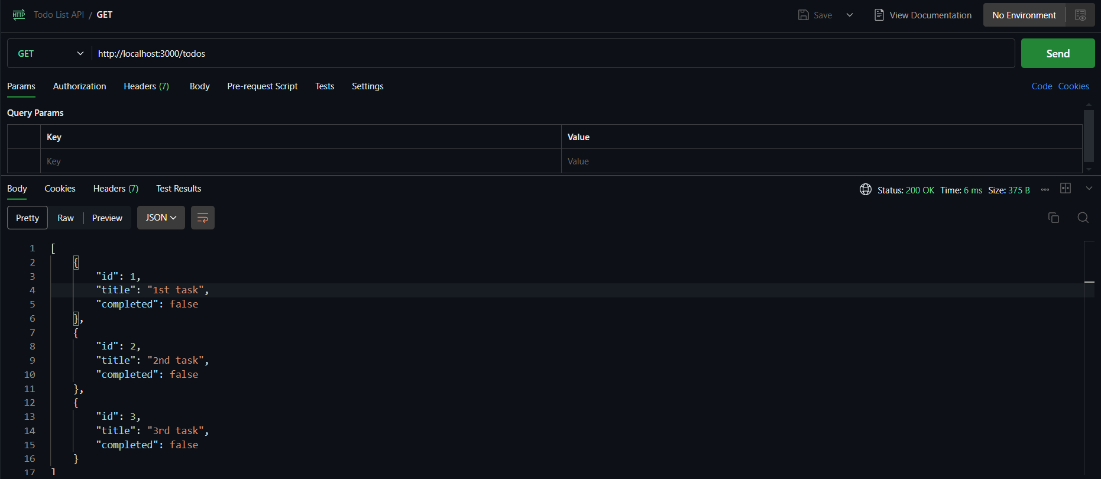
***GET /todos*** *- Fetch all todo items (empty list)*



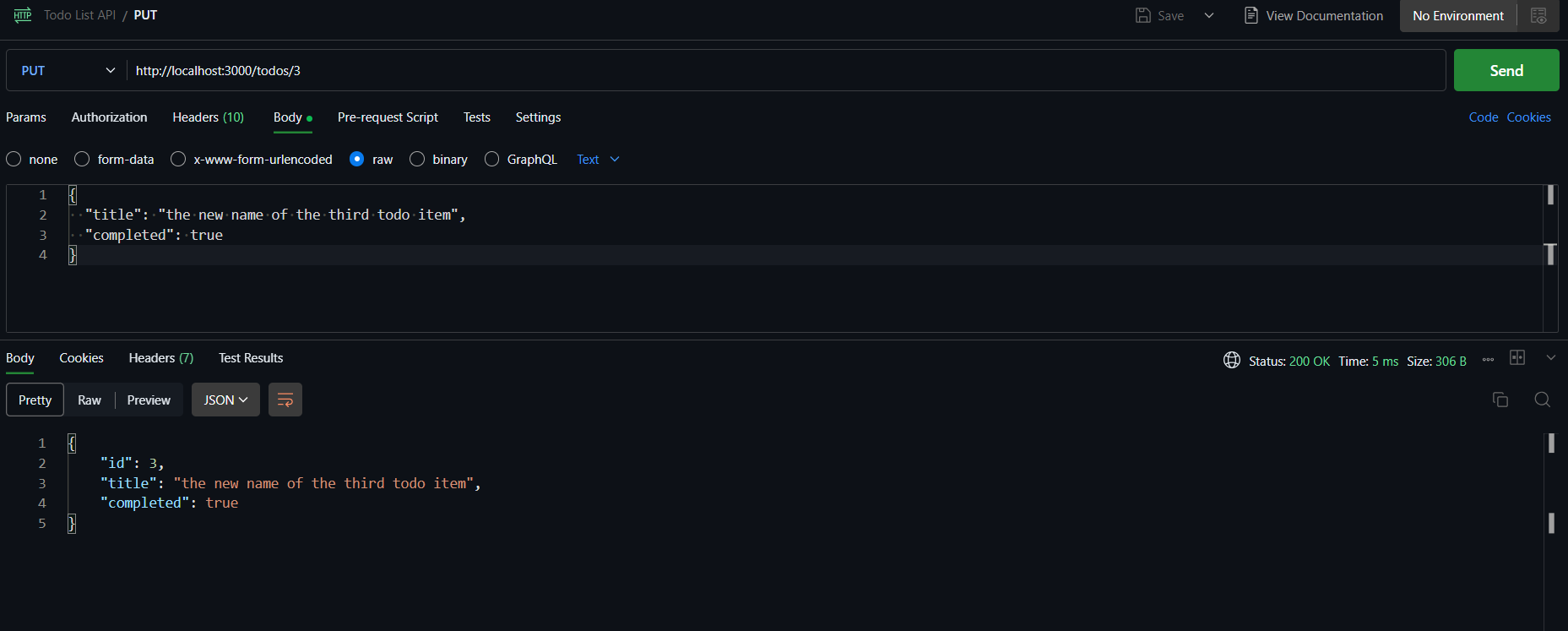
***POST /todos*** *- Create new todo items with titles 1st todo, 2nd todo, 3rd todo*



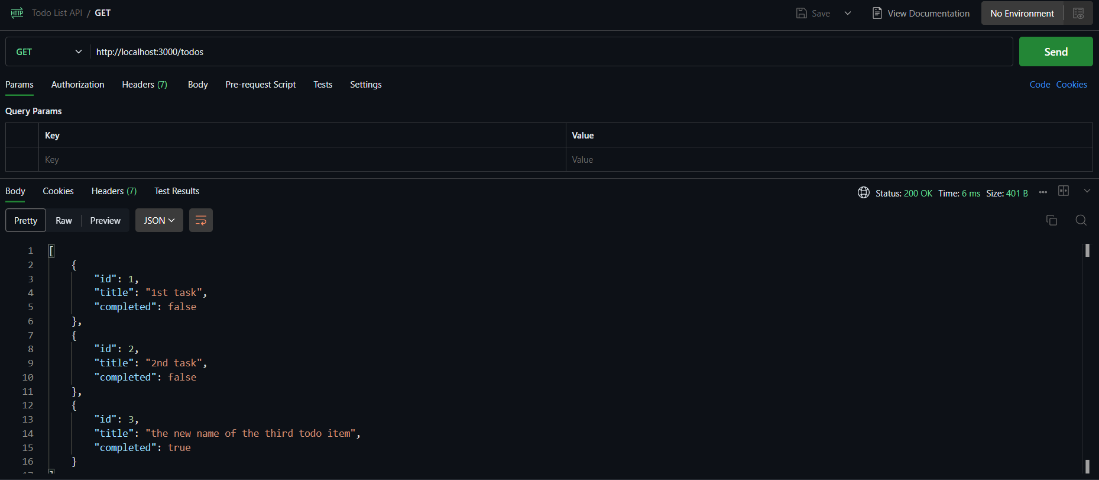
***GET /todos*** *- Fetch all todo items (list of 3 todo items)*



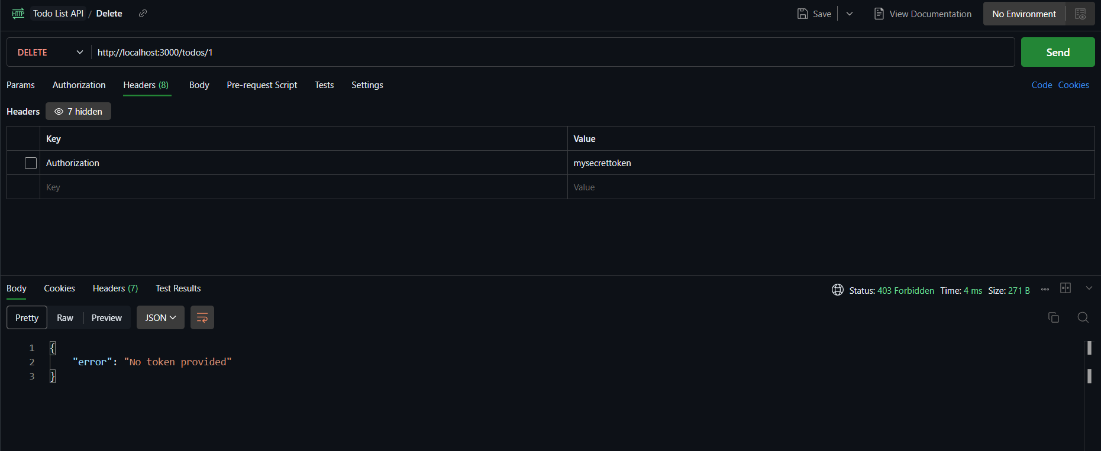
***PUT /todos/1*** *- Update the 3rd with title 'the new name of the third todo item' and completed true*



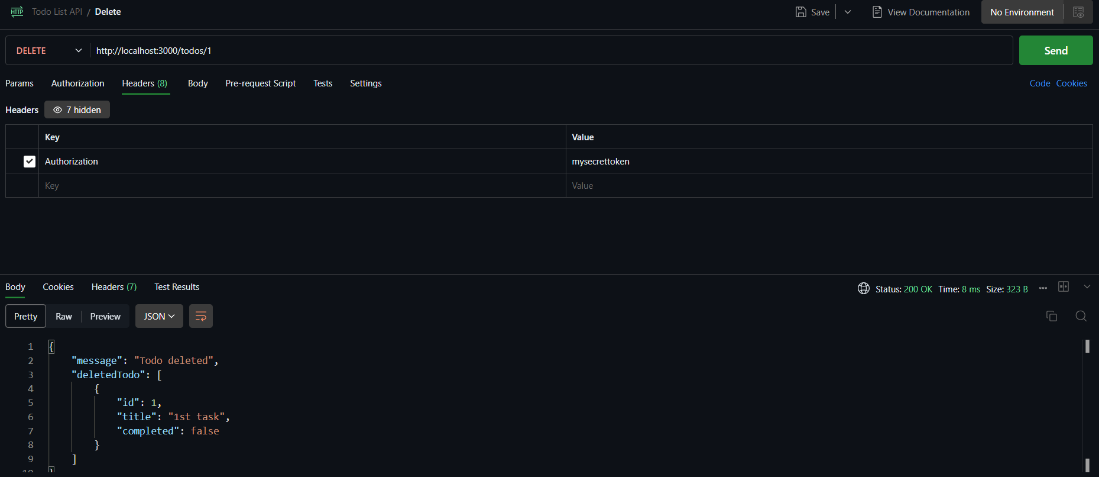
***GET /todos*** *- Fetch all todo items (list of 3 todo items with the updated 3rd todo item)*



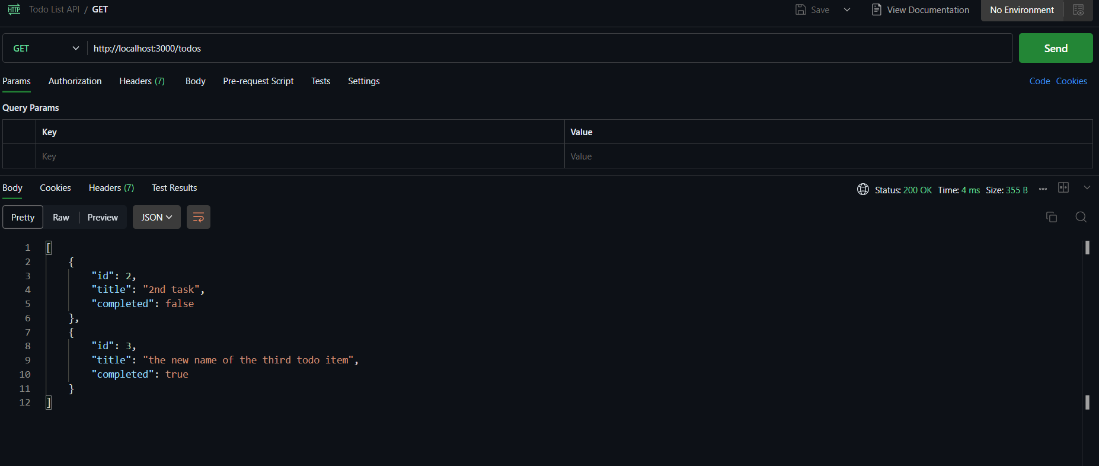
***DELETE /todos/2*** *- Delete the 1st todo item, unauthorized, to test the DELETE endpoint, you need to provide the token in the request headers*



***DELETE /todos/2*** *- Delete the 1st todo item, authorized after providing the token*



***GET /todos*** *- Fetch all todo items (list of 2 todo items after deleting the 1st todo item)*

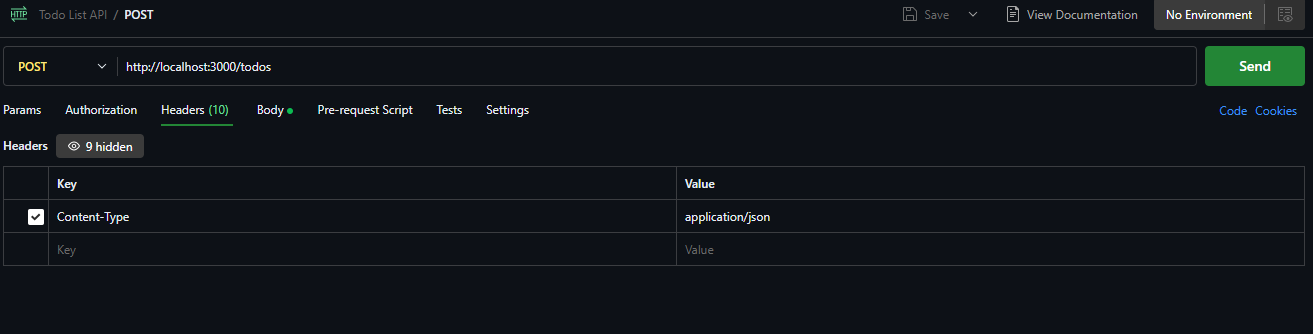


***IMPORTANT NOTE***

*without providing* ***Content-Type*** *in the* ***headers****, the request will* ***not work****. To test the POST, PUT, and DELETE endpoints, you need to provide the Content-Type in the request headers.*

*You can add the Content-Type in the Headers section in Postman as follows:*

*Key: Content-Type | Value: application/json*



## 5- API Documentation wit[h Swagger](https://swagger.io/)

**A screenshot of a computer

Description automatically generated**

# Reading (IMP)

 Code Conventions for the Java Programming Language: [Introduction (oracle.com)](https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html)

 Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin.

# Questions

* What are the primary functions of Chrome Developer Tools, and how can they assist in debugging web applications?
* Describe the benefits of using Swagger for API documentation. How does it enhance communication between developers and users?
* What features of Postman make it a valuable tool for API testing?
* Why is it important to follow coding style best practices, and how can they impact team collaboration and code maintenance?
* How do project management tools like JIRA contribute to the overall success of a software development project? Give examples of features that facilitate this.

# References

<https://developer.chrome.com/docs/devtools/><https://swagger.io/docs/><https://learning.postman.com/docs/>