1. Project Overview

The project is a simple CRUD API for a pizza store. Using Node.js and Express, I built an API that allows users to create, read, update, and delete pizza items in an online store. The API supports operations like adding a new pizza, viewing all pizzas or a specific pizza by its ID, updating pizza details, and deleting pizzas from the store

2. Objectives and Goals

The main goal of this project was to create a RESTful API that simulates the backend functionality of a pizza store. The objectives included:

- *Building CRUD operations for pizza items (Create, Read, Update, Delete).*
- *Implementing an API with the Express framework using Node.js.*
- *Testing the API using Postman to ensure the functionality is working correctly.*

3. Tech Stack

I used the following technologies for this project:

- ***Node.js**: JavaScript runtime for executing server-side code.*
- ***Express**: Framework to create RESTful APIs.*
- ***body-parser**: Middleware to parse incoming request bodies in JSON format.*
- ***Postman**: Tool for testing and sending HTTP requests to the API.*

4. Features Implemented

The application includes the following key features:

- ***POST /items**: Adds a new pizza item to the store.*
- ***GET /items**: Retrieves all pizza items from the store.*
- ***GET /items/{id}**: Retrieves a specific pizza item by its ID.*
- ***PUT /items/{id}**: Updates the details of an existing pizza item by ID.*
- ***DELETE /items/{id}**: Deletes a pizza item by its ID.*

5. Challenges and Solutions

One challenge I faced was managing in-memory data for the pizza items, as it doesn't persist after the server restarts. I solved this by using simple JavaScript arrays to

simulate a database during development. I also faced some difficulties understanding how Express handles routing and middleware, but I learned from the documentation and resolved the issues.

---

6. Code Explanation

Here is a summary of the important sections of the code:

- ***Server Setup**: I used Express to set up a basic server that listens on port 3000.*
- ***Routes**: Defined 5 routes for the CRUD operations:*
  - `POST /items` *to add a new pizza item.*
  - `GET /items` *and* `GET /items/{id}` *to retrieve all items or a specific item.*
  - `PUT /items/{id}` *to update a pizza item.*
  - `DELETE /items/{id}` *to delete a pizza item.*
- ***Error Handling**: Added basic error handling to return 404 when an item is not found and 400 when input data is invalid.*

7. Testing

I tested the API using Postman by sending HTTP requests to the different endpoints. Below are the steps I followed:

- ***POST /items**: Created a new pizza item with name "Vegetarian" and price `10.99`. The response returned the newly created pizza item with an ID.*
- ***GET /items**: Retrieved all pizza items. The response returned the list of pizza items as expected.*
- ***GET /items/{id}**: Retrieved a pizza item by ID and checked that the correct item was returned.*
- ***PUT /items/{id}**: Updated the name and price of an existing pizza item.*
- ***DELETE /items/{id}**: Deleted a pizza item by ID and confirmed successful deletion.*

8. Future Improvements

If I had more time, I would consider implementing the following improvements:

- ***Database Integration****: Integrate a database like MongoDB or MySQL for persistent storage of pizza items.*
- ***Authentication****: Add user authentication using JWT (JSON Web Token) for secure access to API endpoints.*
- ***Validation****: Add input validation to check for valid pizza names and prices.*

9. Conclusion

This project helped me improve my skills in building RESTful APIs using Node.js and Express. I learned how to handle HTTP requests, set up basic routing, and perform CRUD operations. I also gained experience testing APIs using Postman. This was a great exercise to understand backend development principles.