Sharin Nikita
501281914

Technical Report

Introduction:

This report explains the design choices and testing methods used to develop a parts inventory system web application. The application allows users to add and list parts in an inventory using Express, HTML, JavaScript, MongoDB, and Node.js.

Design Choices:

Backend:

Express.js: Used to create a server and handle HTTP requests.
Mongoose: Used to connect to MongoDB and define the schema for parts.
Frontend:

HTML and JavaScript: Used to create the user interface.
Bootstrap: Used for styling the web pages to make them look good and responsive.
Database:

MongoDB: Chosen for its ability to store JSON-like documents, making it easy to store part details.
Endpoints:

Add a Part:

URI: /inventory/part/add
Method: POST
Description: Allows users to add a new part to the inventory by sending a JSON object with name and description.
List All Parts:

URI: /inventory/part/list
Method: GET
Description: Allows users to retrieve a list of all parts in the inventory.
User Interface:

Add Part Page:

Contains a form to input the part's name and description.
On form submission, it sends the data to the /inventory/part/add endpoint.
List Parts Page:

Displays all parts in a table by fetching data from the /inventory/part/list endpoint.
Testing RESTful Endpoints:

Add Part Endpoint:

Tested using Postman by sending POST requests with sample part data.
Verified that parts are added correctly by checking the MongoDB database.
List Parts Endpoint:

Tested using Postman by sending GET requests.
Verified that the returned JSON contains all the parts stored in the database.
Consuming and Parsing JSON:

Fetching Data:

Used the Fetch API to send requests to the backend endpoints.
Parsed the JSON responses using response.json() to convert them into JavaScript objects.
Displaying Data:

Updated the HTML to display parts data in a table format on the list page.
Conclusion:
This web application provides a simple way to manage a parts inventory, allowing users to add and list parts using a user-friendly interface. The backend is powered by Express and MongoDB, ensuring efficient data handling and storage.