Project Title: Street Style Store API

1. Objective:

The purpose of this project was to develop a RESTful API using Node.js and Express with the following functionalities:

CRUD operations on items stored in a PostgreSQL database

Asynchronous file handling to store metadata

Rate-limiting middleware for API request control

Optional JWT-based authentication for secured access

2. Approach:

Setting Up the Environment:

Installed essential dependencies: Express, pg, dotenv, and cors

Initialized the project using npm init

Connected to PostgreSQL using environment variables for security

Database Integration:

Created a PostgreSQL table items using SQL script db.sql

Implemented endpoints for CRUD operations (POST, GET, PUT, DELETE)

Ensured proper database interactions using asynchronous functions

Asynchronous File Handling:

Stored metadata for each created item (timestamp) in logs.json

Used fs.promises for non-blocking file operations

Rate Limiting:

Created a middleware using an in-memory object to track user requests

Limited requests to 100 per 15 minutes

Returned status 429 Too Many Requests for exceeded limits

Optional Authentication:

Secured routes using JWT-based user login mechanism

Generated secure tokens using jsonwebtoken

3. Challenges Faced:

Database Configuration:

Challenge: Connecting PostgreSQL securely and configuring environment variables

Solution: Used dotenv to manage sensitive credentials

Error Handling:

Challenge: Managing errors from database operations and file handling

Solution: Implemented robust error-handling logic to provide meaningful messages

Rate Limiting Logic:

Challenge: Maintaining efficient rate-limit tracking without external libraries

Solution: Used efficient object-based storage to track request timestamps

Authentication:

Challenge: Implementing JWT securely for protected routes

Solution: Validated tokens on every API request to protect endpoints

4. Solutions Implemented:

Used try-catch blocks to handle errors gracefully

Followed modular architecture for route and middleware separation

Leveraged asynchronous functions for better server efficiency

Documented API routes and project setup steps in README.md

5. Conclusion:

This project provided valuable hands-on experience with RESTful API design, PostgreSQL integration, middleware development, and secure file handling practices. It showcased the importance of clean code structure, error handling, and user-friendly API responses.