Technical Design Document

For an ecommerce REST API

TABLE OF CONTENTS

1 System Design

1.1 Design Method and Standards

1.2 Naming conventions

1.3 Software development tools

2 Component Description

3 CONCLUSION

**Note:** The Api documentation of this app can be found in<https://explore.postman.com/templates/4772/wootlab-test>

# System Design

## Design Method and Standards

##### The design methodology used in the project is the waterfall model.

##### The processes includes;

##### Requirement gathering and analysis

##### System design

##### Implementation

##### Testing

##### Deployment (not part of this project)

##### Maintenance (not part of this project)

##### Reason: waterfall model happens to clarify the process into a linear flow with a specified sequence to let the users understand that further level is made progressive on completion of the previous one

##### 

## Programming Standards

##### This project was built using the industry standards for writing JavaScript which includes;

##### ***Formatting***

##### Use of tabs to achieve 4 spaces indentation

##### No trailing white spaces in the .js files

##### Continuous use of semicolons to end statements

##### Use of single quotes to differentiate object strings from normal strings

##### Use of opening braces on the same line as the statement

##### ***Naming Conventions***

##### The use of lowerCamelCase for my variables, functions names and properties

##### ***Variables/functions***

##### Use of descriptive conditions

##### Use of small/short functions

##### Placing all required packages at the top of the file

## Software development tools

##### ***Node js***

##### Node js was used as the backend language for this API. This was used because I am familiar with it :D \*smiles\* and also it helps to build APIs that are scalable. It render the app with speed and since it is a javascript library is help easier integration of front end frameworks like React and Vue

##### ***Express***

##### Express is a node,js framework that helps to handle server side request. This was used because it is faster and its flexibility and scalability.

##### ***MySQL***

##### The database used for this project is MySQL as stated in the technical requirement docs.

##### ***Xampp***

##### Xampp is an application that help you get started with MySQL database on your local machine.

##### ***Postman***

##### The routes was tested with Postman application and published on <https://explore.postman.com/templates/4772/wootlab-test>

##### ***Vs Code***

##### I LOVE VSCODE! :D

# Component Description

## Server

##### The server component which is the server.js file holds the entire backend interaction of our app. At the top of the file i imported all our packages needed to run the app. Next is use of middleware (body-parser) which allows express to read the body of requests and then parse that into a json object.

##### Then i specify our port and connect using out app.listen() function which makes our site accessible in the browser on the specified port; in this case localhost:5000.

##### In the server.js file our database configuration also goes here. I use the mysql.createConnection() function to interact setup our database configuration.

##### **ROUTES**

##### The routes in our application are built with the react-router node package. First i specify the route in our server.js file with app.use() function and include our api/routes path in our server.js file.In our api/routes file i have three files which serve as endpoints for our items, user authentication and payment gateway.

##### ***Items Route***

##### I employ the use of express.Router() function to get our endpoints using Router.get(/\*rest of the code goes here\*/). There are three instances in this component; the module to get all items in our shop, the module to add an item in our shop and the module to get a single item in our shop. At the end of this file i export our router so as to be able to use it outside the file in our application.

##### ***User authentication***

##### Our app integrates a simple user auth system, although for a bigger project i can use a more strict authentication modules like passport.js and JWT since mysql is very porous when it comes to security on user auth.

##### I use express-session module to store login credentials of users after login. The login and register paths interacts and get data from the database using the sql command.

##### ***Payment gateway***

##### The payment gateway used for this project is paypal because it has a more flexible and easier approach in implementing online payment with REST api using a MySQL backend. A better approach would be paystack because of the Nigerian effect but I would prefer to use this over a mongo database.

##### ***Mail***

SMTP was used in sending mail to user after checkout in the api. The [Simple Mail Transfer Protocol](https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol) (SMTP) is a protocol for sending e-mail messages between servers. Most e-mail systems that send mail over the Internet supports SMTP based sending. For testing purpose I used Mailtrap. Mailtrap is a "fake SMTP server" used for development purposes. Instead of having to test your code with your own email account, and potentially flooding your inbox with test emails, you can instead use Mailtrap as the endpoint.

# CALL BACK

##### This is a REST API built using Node js, MySQL and Express. It can be integrated into any front end frame work like React, Vue or Angular. Node js serves as the backend language, express is used to handle server request and MySQL is the database technology used in building the application store.

##### The server starts up on the specified port and database connection is created. When a user hits any route the user is taken to chosen path as specified in the server.js file and redirected to the api file where the request is taken to the particular endpoint.

##### The API documentation can be found in <https://explore.postman.com/templates/4772/wootlab-test>