Project Design

Team - 13

Aaryan Ajay Sharma, Ashutosh Srivastava, Harshvardan, Kunal Bhosikar

Introduction

A project design document (PDD) is a comprehensive blueprint that outlines the goals, objectives, and requirements of a software engineering project. The PDD defines the technical specifications, architecture, design, implementation details, testing and quality assurance, project timelines, and budgets. It serves as a critical reference point for project managers, developers, testers, and other stakeholders throughout the software development lifecycle.

The purpose of a PDD is to provide a clear, concise, and detailed plan that defines the entire software development process. The document serves as a roadmap for developers and project managers to ensure that the software development process remains on track and meets the intended goals and objectives. The PDD also acts as a communication tool for stakeholders, ensuring that everyone involved in the project has a clear understanding of the project's scope and requirements.

The client is presently utilising WordPress for hosting their website and wishes to migrate it to a headless CMS. A headless CMS is a backend content management system where the content layer is distinct from the presentation layer. To accomplish this, we need to utilise Strapi (a headless CMS) that utilises Javascript and a separate database, business, and front-end layer. The client desires to move the whole application to Strapi, which is a node js-based platform, but having a basic version of the website on the new technology stack will also suffice. However, the company wants to limit the scope of the migration to the CMS level only and not extend it to the e-commerce level.

System Overview

The software system that we are building is used for the empowerment of the nanopreneurs. We are building the backend for the website on which these small vendors and shop owners (nanopreneurs) can open their shops and expand their businesses.

For e.g., if there is a person who has a small vegetable shop and wants to expand his business. He will simply create his virtual shop on the website 'chotu.com'. Now any customer can buy anything by going to chotu.com/mobile_no_of_the_shop_owner.

This will help these nanopreneurs to expand their businesses without giving a cut to anyone as the middleman is eliminated from the scenario.

Design Overview

Architectural design

I) Our App is based on Headless CMS Strapi, where we tend to manage and store the records of Users, Rootshop and Products as different tables with correlations between them via MySQL database as the backend.

II) At last, WhatsApp Api integration is done to send the order placed on the rootshop website as a receipt to the User on his WhatsApp chat.

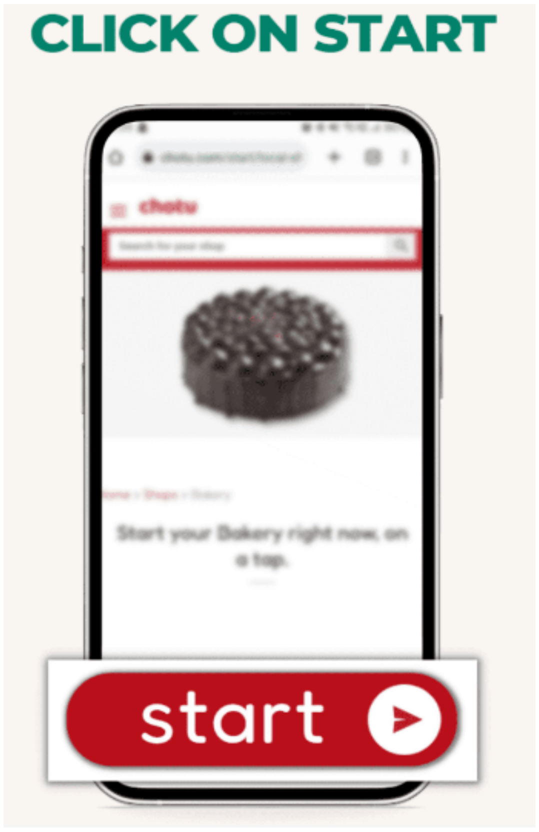
III) When a new nanopreneurs want to make his website, he simply messages "start <shoptype to be chosen from our list of shop catalog> " on the Chotu's official number, and their shop starts at https://chotu.com/<user_mobile_no> where he can add the products and its tags, which gets stored on chotu's servers. Then whenever a user tries to place an order using forms, we simply send the user the receipt of the order and let him handle all the payment and delivery part.

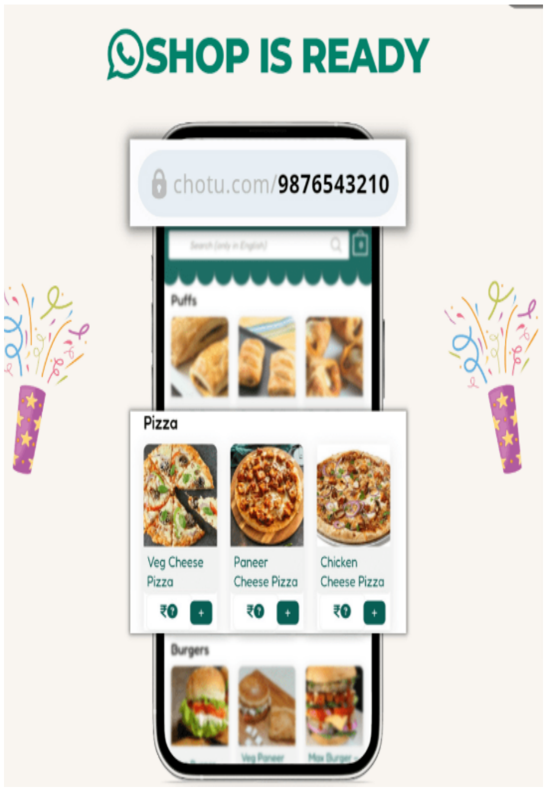
The architectural design diagram of Strapi specific to our project can be viewed here.

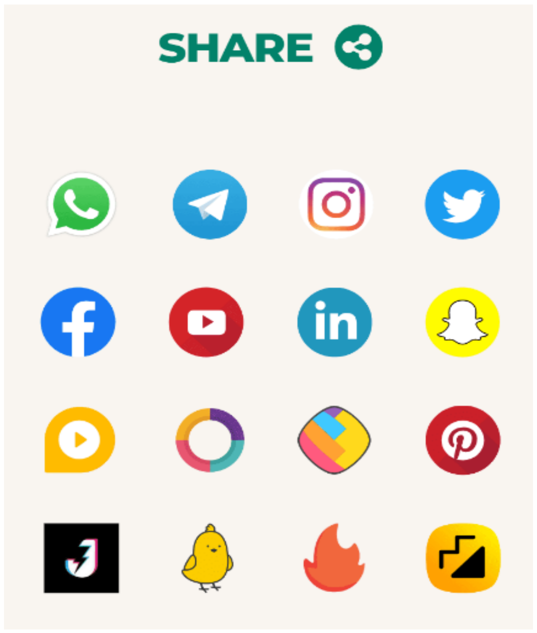
System interfaces

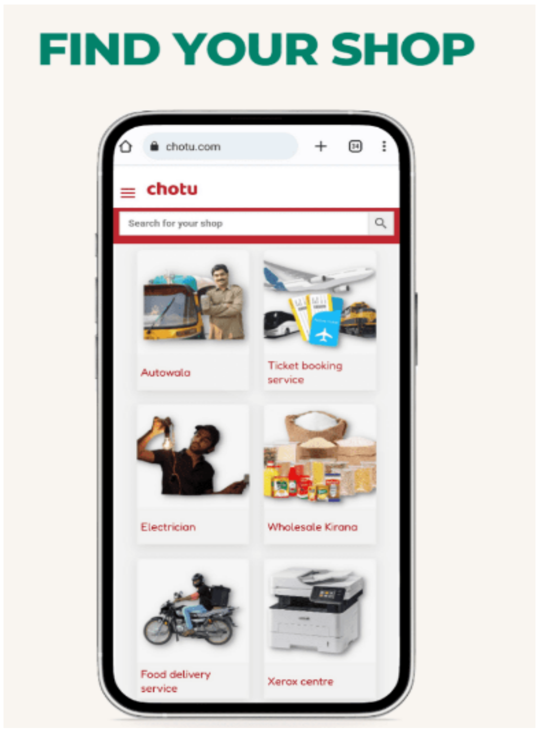
User Interface

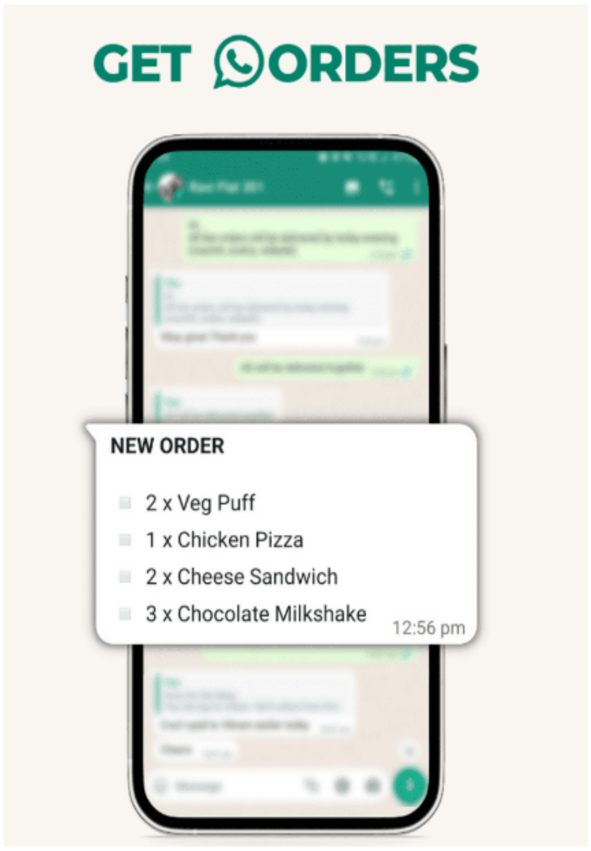
A sample UI of the project is shown via the screenshots below:




SHOP IS READY
chotu.com/9876543210
Puffs
Pizza
Veg Cheese Pizza
Paneer Cheese Pizza
Chicken Cheese Pizza
₹0
Burgers


SHARE





NEW ORDER

- 2 x Veg Puff
- 1 x Chicken Pizza
- 2 x Cheese Sandwich
- 3 x Chocolate Milkshake

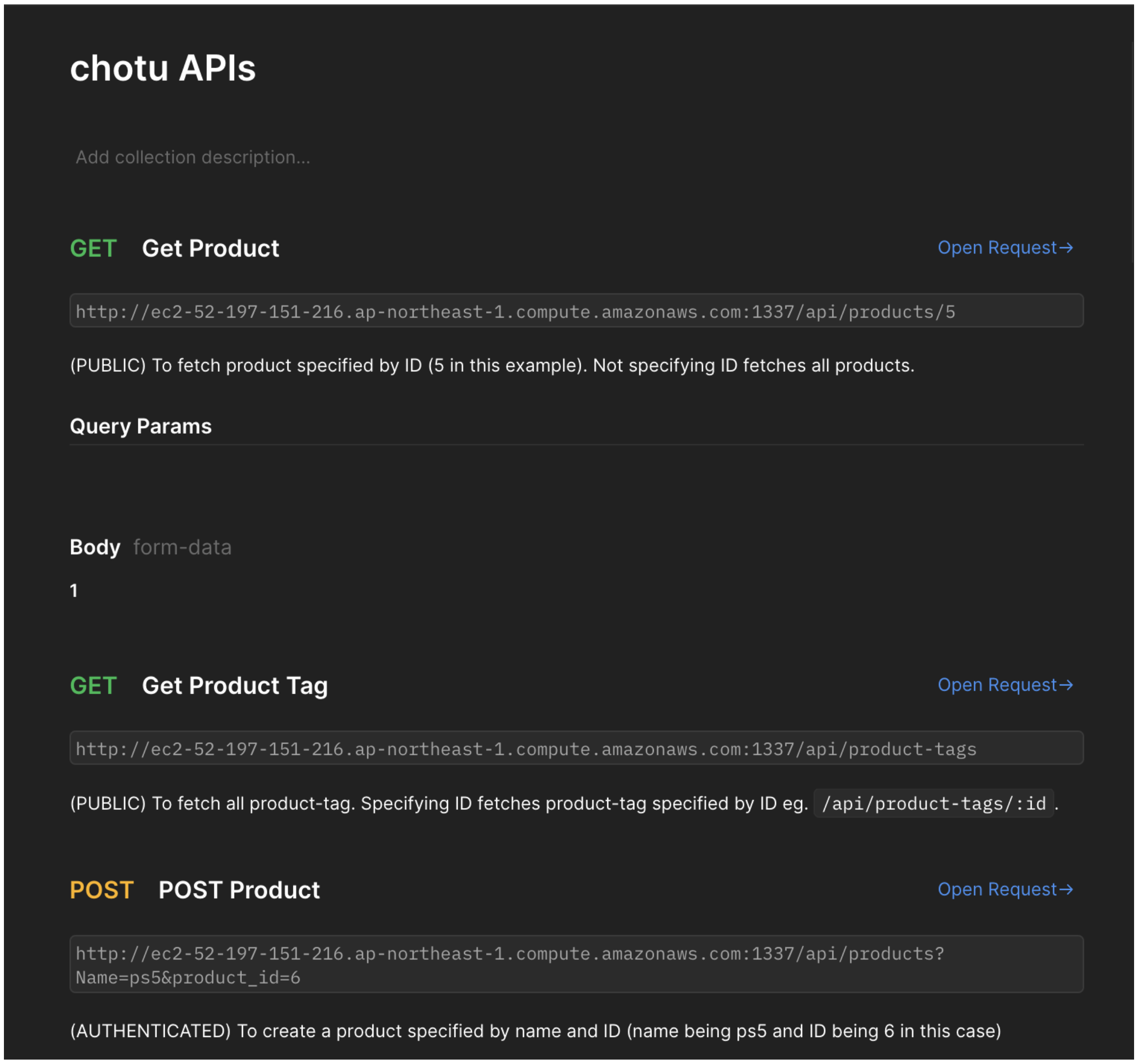
12:56 pm

APIs

We will be designing and creating the APIs for the client's frontend team to use. There will be API to view products & rootshops for public users and API for creating, deleting and updating products and rootshop for authenticated users.

Postman will be used for testing and documenting the APIs that are created.

The API design process is still in progress. The screenshot of the documentation of the working APIs is as follows:



chotu APIs

Add collection description...

GET Get Product

```
http://ec2-52-197-151-216.ap-northeast-1.compute.amazonaws.com:1337/api/products/5
```

(PUBLIC) To fetch product specified by ID (5 in this example). Not specifying ID fetches all products.

Query Params

Body form-data

1

GET Get Product Tag

```
http://ec2-52-197-151-216.ap-northeast-1.compute.amazonaws.com:1337/api/product-tags
```

(PUBLIC) To fetch all product-tag. Specifying ID fetches product-tag specified by ID eg. `/api/product-tags/:id`.

POST POST Product

```
http://ec2-52-197-151-216.ap-northeast-1.compute.amazonaws.com:1337/api/products?
Name=ps5&product_id=6
```

(AUTHENTICATED) To create a product specified by name and ID (name being ps5 and ID being 6 in this case)

Design Rationale

We faced a lot of difficulties during the initial understanding of the headless CMS and while comparing between several options we had . In this regard, our meeting with the client turned out to be fruitful in the sense they provided us with an easy understanding of how headless cms work and what are the different parameters we can look for while comparing between the options like Open Source/paid, Community support, plugins etc. After a long analysis, we chose to move on with Strapi.

Moving on we are faced with difficulties in finding out similar analogies of WordPress and Strapi while porting the website from the former to the latter, here also our regular meetings with the client came to our rescue as the client gave a brief overview of what different terminologies in WordPress exactly meant, which made our lives easier while finding out the similar features in Strapi and implementing them as per our use case.

The form is a major part of our project, and we were faced with many difficulties as to how to implement forms in Strapi. The client requested for us to use Plugins to create forms, but no plugins are available as of now that do the same.

Therefore, we decided to use collection types as a substitute for it and be implementing major functionalities of form, like sending form data using the collection types API in Strapi.