**Introduction**

This document acts as a guide to understanding Caffein’s design, such as database and API design. This document is a continuation of the analysis document.

**Overview**

* Frontend: The user interface is developed in React. It is responsible for showing the product listing to the user and facilitating user interactions, such as the buttons and forms to add/ modify the products.
* Backend: Uses Node.js with Express.js to handle API requests and perform CRUD operations (create, read, update, delete).
* Database: The MySQL database stores the product information that will be retrieved.

**Database Design**

**Table Creation**

The MySQL database has a `products` table, which stores the information about the products listed on the app. This SQL query below creates the `products` table and adds some products. The queries that were used to create the table are as such:

CREATE TABLE products(

id INT NOT NULL PRIMARY KEY IDENTITY,

productname VARCHAR (100) NOT NULL,

category VARCHAR (100) NOT NULL,

roastlevel VARCHAR(20) NOT NULL,

purpose VARCHAR(100) NOT NULL,

flavor VARCHAR(100) NOT NULL,

amount VARCHAR(10) NOT NULL,

price DECIMAL (16, 2) NOT NULL,

image VARCHAR(200) NOT NULL,

);

INSERT INTO caffein.products (productname, category, roastlevel, purpose, flavor, amount, price, image)

VALUES

('Arabica Ethiopia Hambella Guji', 'Coffee Beans', 'Medium', 'Espresso', 'Lime, vanilla, coconut', '500 gr', 10.25, 'coffee.jpg'),

('Arabica Guatemala Huehuetenango Caturra Bourbon', 'Coffee Beans', 'Medium', 'Espresso, normal with or without milk', 'Intense, caramel, chocolate', '500 gr', 12.35, 'coffee.jpg'),

('Arabica Honduras Marcala Comsa', 'Coffee Beans', 'Medium', 'Espresso, filter coffee, with or without milk', 'Walnut, vanilla, milk chocolate', '500 gr', 11.95, 'coffee.jpg'),

('Arabica Costa Rica Coope Tarrazu', 'Coffee Beans', 'Medium', 'Espresso, normal with or without milk, filter coffee', 'Nutty, honey, chocolate', '500 gr', 12.10, 'coffee.jpg'),

('Arabica Brazil Yellow Bourbon', 'Coffee Beans', 'Medium', 'Espresso, filter coffee', 'Caramel, milk chocolate, mild', '500 gr', 11.80, 'coffee.jpg'),

('Arabica Peru Churupampa', 'Coffee Beans', 'Medium', 'Espresso', 'Dark chocolate, molasses', '250 gr', 9.05, 'coffee.jpg'),

('Robusta Java Mocha', 'Coffee Beans', 'Dark', 'Filter coffee', 'Chocolate, sweet tobacco', '500 gr', 8.65, 'coffee.jpg')

Notes:

* The `id` column is the primary key and is auto-incremented.
* The `price` column is of type `DECIMAL(16,2)` which store prices with 2 decimal places.

1Caffein's database diagram

**A screenshot of a computer

Description automatically generatedDatabase Diagram**

1Caffein's database diagram

The diagram above is Caffein’s intended database structure. The “messages” table is separated deliberately because it was designed to be connected to a “contact us” page, where users can anonymously ask about their orders, general questions, and even job applications. Forcing the users to create an account to ask a question will be inconvenient.

Regarding the numbers between two tables, such as between the “user” and “order” table, where there is a “1” and “0…\*”, it means that each user can have 0 to many orders. Each order can have between 1 to many items.

**API Design**

Caffein provides some RESTful API endpoints to manage the product data in the MySQL database, such as retrieving the product listing, adding a new product, deleting the product, and updating the product.

1. **Get All Products**

Endpoint: `/products`

Method: `GET`

Description: Retrieves all products

Query:

SELECT \* FROM products

Response:

{

"id": 7,

"productname": "Robusta Java Mocha",

"category": "Coffee Beans",

"roastlevel": "Dark",

"purpose": "Filter coffee",

"flavor": "Chocolate, sweet tobacco",

"amount": "500gr",

"price": 8.65,

"image": "coffee.jpg"

},

1. **Add New Product**

Endpoint: `/products`

Method: `POST`

Description: Add a new product to the database

Query:

INSERT INTO products (`productname`, `category`, `roastlevel`, `purpose`, `flavor`, `amount`, `price`, `image`) VALUES(?)

Response:

“Success!”

1. **Delete Product**

Endpoint: `/products/:id`

Method: `DELETE`

Description: Delete a product by ID from the database

Query:

DELETE FROM products WHERE id = ?

Response:

“Success!”

1. **Update Product**

Endpoint: `/products/:id`

Method: `PUT`

Description: Update a product by ID

Query:

UPDATE products SET `productname`=?, `category`=?, `roastlevel`=?, `purpose`=?, `flavor`=?, `amount`=?, `price`=?, `image`=? WHERE id=?

Response:

“Success!”

**Conclusion**

Caffein utilizes React as a frontend, Node.js as the backend, and MySQL database. The API endpoints will be responsible for the key functionalities, such as managing the products with the CRUD (create, read, update, delete) operations.