**REAL-TIME DATABASE**

**Project Requirements Breakdown:**

1. Utilizing an external API to populate a real-time database.
2. Following the MVC architecture.
3. Implementing a start-up routine for initial data fetching.
4. Completing all CRUD operations.
5. Testing via Postman.

**Implementing:**

1. **Select an API:** chose an API that is **simple** but serves some detailed information about fruits and their nutritionalvalues that could be a part of Nutrition calculator application.
2. **Database Schema Design:** Created a database schema “fruits” using MySQL, to reflect the type of data returned from the selected API (with a small twist).
3. **Start-up Routine:** Implemented a startup routine to automatically fetch data using “Axios” and populate the database(if the database is empty) when the app starts.
4. **MVC Structure:** The project is following an MVC pattern, creating separate files for models, routes, and controllers.
5. **CRUD Operations:** Successfully implemented in my controllers.
6. **Postman Testing:** Used Postman to test each CRUD operation, ensuring functionality.

**Application Overview:**

The application is a server-side **RESTful API** developed using Node.js, Express, to interact with a **MySQL database**. The project aims to **centralize** and **manage** information about different types of fruits and their nutritional values. It **fetches** initial data from an **external API** and populates the **MySQL** database as a **start-up routine**.

**Features Include:**

* **Initial Data Fetch and Insert**: Using Axios, and into MySQL Table.
* **CRUD Operations**: Create, Read, Update, Delete and Upsert.
* **Data Modelling**: Utilizes Sequelize as an ORM to define a **Fruit** model.
* **Error Handling**: Integrated **error handling** to return appropriate status codes and/or **confirmation messages**.
* **Data Validation**: Used Sequelize validators to enforce data integrity, ensuring fields like **name**, **family**, etc., are properly filled.
* **Logging**: Detailed logs of database transactions for traceability.

**Some Benefits of this Database vs External API:**

**Speed**: Reading from this database was generally faster than making an external API call.

**Request Limits:** Some APIs I came across accepted limited number of requests, so generally databases would eliminate this limit.

**Data Structure**: I could structure the data in a different way that is more relevant to me, which will be very useful for more complex projects.

**Security**: a database can provide more security control over the stored data.

**Possible Expantion:**

1. Expanding **CRUD** operations to **Search/ Filter** fruits by **Family** or **Nutritional Values** and integrate it into **a frond end app.**
2. **Caching**: integrate it with Redis for quicker data access.