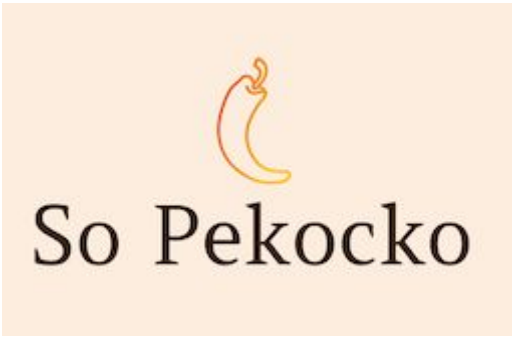

Application API Requirements

Project Context

So Pekocko is a family business with 10 employees. Its main activity is the creation of spicy sauces whose composition is kept secret. Building on its success, the company wants to grow and create a web application in which users can add their favorite sauces and like or dislike the sauces offered by others.

API Requirements

Verb	**Endpoint**	**Request body (where applicable)**	**Expected response type**	**Function**
POST	/api/auth/signup	{ email: string, password: string }	{ message: string }	Hashes user password, adds user to database.
POST	/api/auth/login	{ email: string, password: string }	{ userId: string token: string }	Checks user credentials, returning the user's _id from the database and a signed JSON web token (also containing the user's _id).
GET	/api/sauces	-	Array of sauces	Returns array of all sauces in the database.
GET	/api/sauces/:id	-	Single sauce	Returns the sauce with the provided _id.
POST	/api/sauces	{ sauce: String, image: File }	{ message: String }	Captures and saves image, parses stringified sauce and saves it to the database, setting its imageUrl correctly. Initializes sauces likes and dislikes to 0, and usersLiked and usersDisliked to empty arrays.

PUT	/api/sauces/:id	EITHER Sauce as JSON OR { sauce: String, image: File }	{ message: String }	Updates the sauce with the provided _id. If an image is uploaded, capture it and update the sauces imageUrl. If no file is provided, the sauce details are directly within the request body (req.body.name, req.body.heat etc). If a file is provided, the stringified sauce is in req.body.sauce.
DELETE	/api/sauces/:id	-	{ message: String }	Deletes the sauce with the provided _id.
POST	/api/sauces/:id/like	{ userId: String, like: Number }	{ message: String }	Sets "like" status for the userId provided. If like = 1, the user likes the sauce. If like = 0, the user is cancelling their like or dislike. If like = -1, the user dislikes the sauce. The user's ID must be added to or removed from the appropriate array, keeping track of their preferences and preventing them from liking or disliking the same sauce multiple times. Total number of likes and dislikes to be updated with each like.

API Errors

Any errors must be returned as they are thrown, with no modification or additions. If needed, use new Error().

API Routes

All sauce-related routes must require an authenticated request (containing a valid token in its Authorization header: "Bearer <token>").

Data Models

Sauce

- **_id**: *String* — the unique identifier created by MongoDB
- **userId**: *String* — the MongoDB unique identifier for the user who created the sauce
- **name**: *String* — name of the sauce
- **manufacturer**: *String* — manufacturer of the sauce
- **description**: *String* — description of the sauce
- **mainPepper**: *String* — the main pepper ingredient in the sauce
- **imageUrl**: *String* — the URL for the picture of the sauce uploaded by the user
- **heat**: *Number* — number between 1 and 10 describing the sauce
- **likes**: *Number* — number of users liking the sauce
- **dislikes**: *Number* — number of users disliking the sauce
- **usersLiked**: *[String]* — array of user IDs of users having liked the sauce
- **usersDisliked**: *[String]* — array of user IDs of users having disliked the sauce

User

- **userId**: *String* — the MongoDB unique identifier for the user who created the sauce
- **email**: *String* — the user's email address **[unique]**
- **password**: *String* — hash of the user's password

Security Requirements

- User password must be encrypted.
- Transport and storage of PII should be secure.
- Two types of database administrator must be defined: access to delete and modify tables, and access to modify the database contents.
- Authentication must be reinforced on the required routes;
- Email addresses in the database are unique and an appropriate Mongoose plugin is used to ensure their uniqueness and report errors.
- Security of the MongoDB database (from a service such as MongoDB Atlas) must not impede the application from launching on a user’s machine;

GitHub Repository

Pull the front-end app code from this project repo and take the following steps:

1. Clone repo
2. Open a terminal (Linux/Mac) or command prompt/PowerShell (Windows)
3. Run npm install from within the project directory
4. Run npm start
5. Run your API on http://localhost:3000