Cleaning Store Backend API Documentation

This document describes the endpoints exposed by the Cleaning Store backend server (`server.js`).

Authentication

Register Customer

- **URL:** `/api/customers/register`
- **Method:** POST
- **Fields (JSON):**
  - `username` (string) – required
  - `email` (string) – required, unique
  - `password` (string) – required, minimum 6 characters
- **Behavior:** hashes password with bcrypt, stores customer in SQLite.
- **Response:** `201` with JSON `{ message, customer }` (customer excludes password).

Login Customer

- **URL:** `/api/customers/login`
- **Method:** POST
- **Fields (JSON):**
  - `email` (string) – registered address
  - `password` (string)
- **Behavior:** verifies password and returns a JWT if valid.
- **Response:** `200` with JSON `{ token }`.

Get Current Customer

- **URL:** `/api/customers/me`
- **Method:** GET
- **Headers:** `Authorization: Bearer <token>`
- **Behavior:** verifies JWT token via middleware and returns customer info.
- **Response:** `200` with customer object (id, username, email, createdAt).

Admin (existing functionality)

Register Admin

- **URL:** `/api/auth/register`
- **Method:** POST
- **Fields:** `username`, `email`, `password`
- **Note:** first registration creates superadmin.

Login Admin

- **URL:** `/api/auth/login`
- **Method:** POST
- **Fields:** `username`, `password`
- **Response:** JWT token and admin details.

Get Admin Profile

- **URL:** `/api/auth/me`
- **Method:** GET
- **Headers:** `Authorization: Bearer <token>`

Categories

- `GET /api/categories` – list all categories

- `GET /api/categories/:id` – retrieve category
- `POST /api/categories` – create (protected)
- `PUT /api/categories/:id` – update (protected)
- `DELETE /api/categories/:id` – delete (protected, no products)

Products

- `GET /api/products` – list products (optional `?category=` filter)
- `GET /api/products/:id` – retrieve product
- `POST /api/products` – create (protected, multipart/form-data with `image`)
- `PUT /api/products/:id` – update (protected)
- `DELETE /api/products/:id` – delete (protected)

Orders

- `GET /api/orders` – list orders
- `POST /api/orders` – create order
- `PUT /api/orders/:id/status` – update status

Database Schema

The SQLite database (`customers.db`) contains a `customers` table:

```
CREATE TABLE customers (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT NOT NULL,
  email TEXT NOT NULL UNIQUE,
  password TEXT NOT NULL,
  createdAt TEXT NOT NULL
);
```

Other data is stored as JSON files in `data/`.

Running the Server

1. Install all dependencies:

   ```
   npm install
   ```

2. (Optional) set `JWT_SECRET` environment variable.
3. Start the server:

   ```
   npm start
   ```

4. Database file and tables are created automatically.

Notes

- CORS enabled.
- `express.json()` used for JSON parsing.
- Errors propagate to global error handler.

This document may be converted to PDF via the provided npm script or external tools.