User Story: Browsing and Managing Products as a User
Title: User can browse and interact with product listings

*As a
Customer of the fresh-cart platform

I want to
View available products and their details

So that I can
Browse, make decisions, and potentially add them to my cart

////////////////////////////////
** Acceptance Criteria:
   1- I can view a list of all available products.
   2- I can see detailed information for each product (e.g., name, price, image, stock).
   3-I can filter or sort products (optional/advanced).
   4-I can add products to a cart (if the feature exists).
   5-I can view my cart and proceed to checkout (if implemented).

///////////////////////////////
**Product data is fetched from MongoDB via Mongoose models and exposed through RESTful APIs.

///////////////////////////////
**Technical Notes:
Product data is retrieved via GET /products

Data served from MongoDB using Mongoose models

we need Cart, orders, and user authentication can also be built using MongoDB

JWT is not required for public product viewing, but needed for cart/checkout if user accounts are used

///
///
///
///
///
///
///
///
///

User Story: Admin Management of Products
Title: Admin can manage (CRUD) product listings from a secure admin dashboard

As an
Site Administrator

I want to
Register, log in, and access a protected dashboard using a secure authentication system

So that I can

Create, read, update, and delete product data stored in the MongoDB database via a PostgreSQL-powered admin backend

**Acceptance Criteria:

1-Admin can register with a unique username and secure password.

2-Admin can log in and receive a JWT token.

3-Admin routes are protected using a JWT-based middleware.

///////////////////////////////////////
**Admin can:

1-Create a new product (POST /products/item)

2-View all products (GET /products)

3-Update a product by ID (PUT /products/:id)

4-Delete a product by ID (DELETE /products/:id)

*** All product operations interact with MongoDB through a service layer (bridge).

*** Admin credentials and authentication details are stored and validated using PostgreSQL with Sequelize ORM.

//
*** Technical Notes:
Admin module (auth, dashboard) → PostgreSQL + Sequelize

Product data → MongoDB + Mongoose

Middleware handles JWT validation

Communication bridge built in Express services/routes

Tested via Thunder Client