Amazon-like E-commerce Application

A Spring Boot e-commerce application with PostgreSQL database that mimics Amazon's basic functionality.

Features

- **Product Catalog**: Browse products with fake ASINs
- **User Management**: Pre-populated with test users
- **Product Categories**: Electronics, Books, Home & Kitchen, Sports, Fashion, Beauty
- **REST API**: Full CRUD operations for products
- **Responsive UI**: Amazon-inspired frontend with Thymeleaf

Tech Stack

- Spring Boot 4.0.0
- PostgreSQL
- Spring Data JPA
- Thymeleaf
- Lombok
- Java 17

Prerequisites

1. **Java 17** or higher
2. **PostgreSQL** installed and running
3. **Gradle** (included via wrapper)

Database Setup

1. Start PostgreSQL service:

```
# macOS
brew services start postgresql

# Linux
sudo systemctl start postgresql
```

2. Create the database:

```
psql -U postgres
CREATE DATABASE ecommerce;
\q
```

3. Update database credentials in `src/main/resources/application.properties` if needed:

```
spring.datasource.username=postgres
spring.datasource.password=postgres
```

Running the Application

1. Build the project:

```
./gradlew clean build
```

2. Run the application:

```
./gradlew bootRun
```

3. Access the application:
    - **Web UI**: http://localhost:8081
    - **REST API**: http://localhost:8081/api/products

Sample Data

The application automatically initializes with:

Users (3 test users)

- john.doe@example.com
- jane.smith@example.com
- bob.jones@example.com

Products (12 items with fake ASINs)

- Electronics: Echo Dot, Fire TV Stick, Kindle
- Books: Atomic Habits, The Midnight Library
- Home & Kitchen: Instant Pot, Ninja Air Fryer
- Sports: YETI Rambler, Resistance Bands
- Fashion: Levi's Jeans, Leather Handbag
- Beauty: CeraVe Moisturizing Cream

API Endpoints

Products

- `GET /api/products` - Get all available products

- `GET /api/products/{id}` - Get product by ID
- `GET /api/products/asin/{asin}` - Get product by ASIN
- `GET /api/products/category/{category}` - Get products by category
- `GET /api/products/search?keyword={keyword}` - Search products

Project Structure

```
src/
├── main/
│   ├── java/
│   │   └── com/aerloki/personal/project/Personal/Project/
│   │       ├── model/          # Entity classes
│   │       ├── repository/     # JPA repositories
│   │       ├── service/        # Business logic
│   │       ├── controller/     # REST & Web controllers
│   │       └── config/         # Configuration & data initializer
│   └── resources/
│       ├── application.properties
│       ├── templates/          # Thymeleaf templates
│       └── static/             # Static resources
```

Development

The application uses:

- `spring.jpa.hibernate.ddl-auto=create-drop` - Database schema recreated on each run
- Auto-initialization with sample data via `DataInitializer.java`

Notes

- Default port: 8081 (configurable in application.properties)
- Database is recreated on each application restart
- CORS enabled for REST API endpoints
- Security features are currently disabled for demo purposes