Delivery App Simulator

Overview

The Delivery App Simulator is designed to emulate a delivery platform where customers can place orders, track deliveries, and view order history. It uses Spring Boot REST APIs and a relational database Exception handling ensures proper error management for invalid inputs, incorrect addresses, or unavailable items. Git is used for version control and Postman for API testing.

- Allows customers to place and track delivery orders.
- Maintains order and delivery status history.
- Validates customer details and delivery addresses.
- Manages order items and pricing accurately.
- Implements OOP principles for customers, orders, and delivery objects.
- Supports modular API design using Spring Boot.
- Facilitates development and testing with Git and Postman.

Goals

The goal of the Delivery App Simulator is to create a reliable platform for managing deliveries, ensuring accurate order processing, delivery tracking, and customer satisfaction.

- Enable customers to place new delivery orders.
- Track delivery status in real-time.
- Maintain accurate order history for users.
- Validate inputs and handle exceptions properly.
- Use Spring Boot for API development and modular design.
- Ensure proper database persistence for orders and deliveries.
- Support Git version control and API testing via Postman.

Specifications

Below is a list of **proposed endpoints** with a brief summary of each. You can extend or modify these as you see fit, but the core functionality must be covered.

Endpoint	HTTP Method	Description
/orders	POST	Place a new delivery order.
/orders/{id}	GET	Retrieve details of a specific order.
/orders/{id}	PUT	Update order details.
/orders/{id}	DELETE	Cancel an order.
/orders/history/{userId}	GET	Fetch all past orders for a customer.
/delivery/{orderId}/status	GET	Check the delivery status of an order.
/delivery/{orderId}/update	PUT	c
/items	GET	Retrieve available delivery items.
/items/add	POST	Add a new delivery item (admin functionality).
/items/{id}	PUT	Update an existing delivery item.

Additional Guidelines

- Ensure validation of inputs (e.g., no negative amounts, valid dates).
- Use Maven to manage dependencies and build lifecycle.
- Use Git for source control with clear commit messages.
- Write clean, readable code with proper comments where needed.
- Bonus: Implement pagination for data retrieval if time permits.

Milestones

- Complete source code in a GitHub repository
- README explaining setup, build, and run instructions
- Postman collection or API documentation for testing endpoints