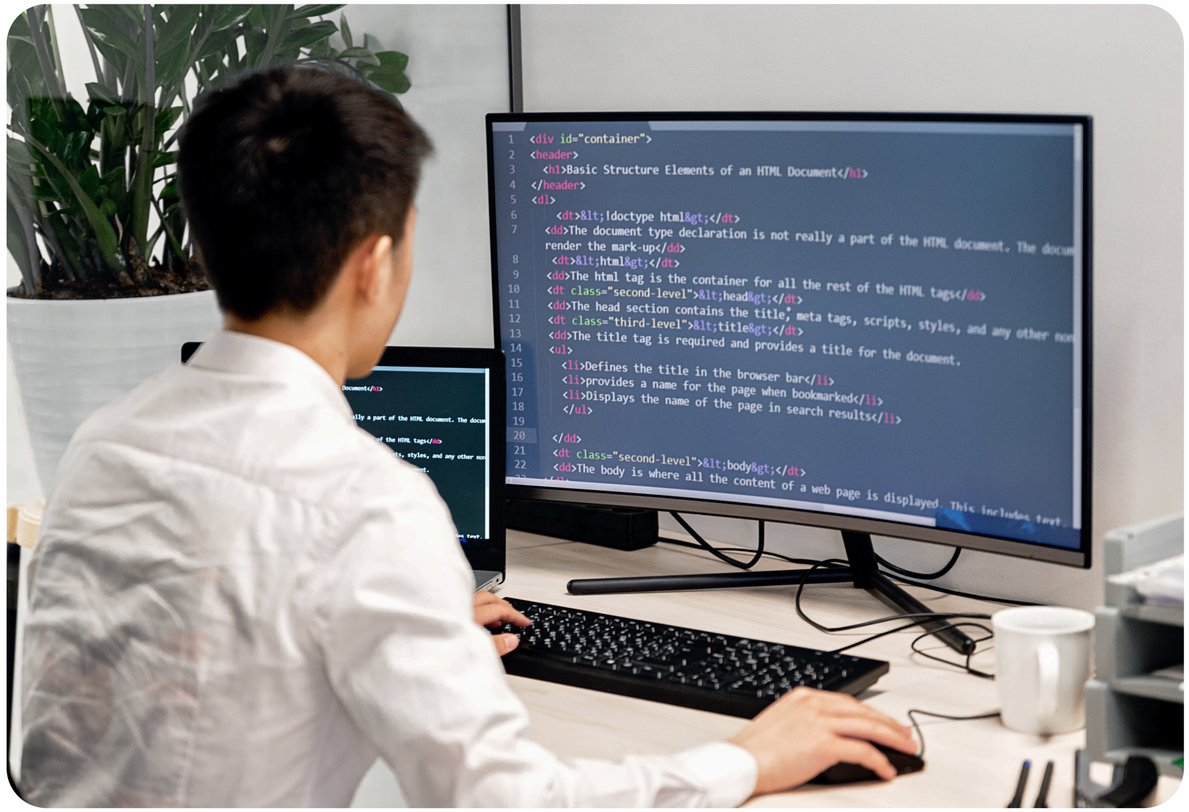
Sanlam

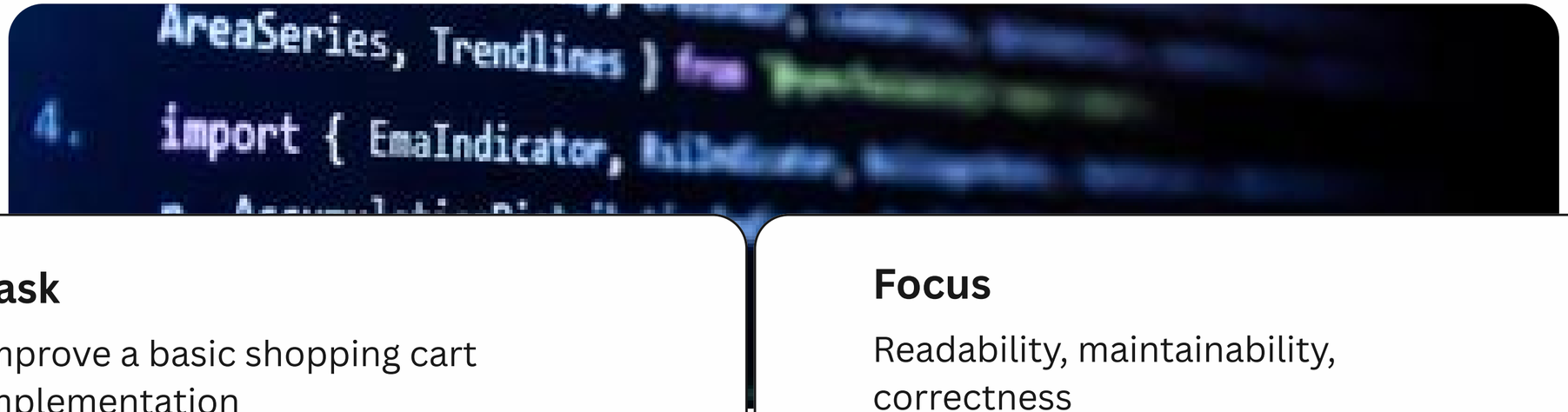
Overview | Approach | Architecture | Improvements

# SHOPPING CART



Fanelesibonge Mbuyazi

# OVERVIEW




## Task

Improve a basic shopping cart implementation

## Focus

Readability, maintainability, correctness

## Out of scope

Database, testing, security

## Approach

Refactor into clean layered architecture

# OVERVIEW

## The original code functionality:

- Add items (name, price, quantity) to a cart
- Track multiple carts using unique cart IDs
- Calculate the total cost of all items

Sanlam



**FOCUS**: Readability, maintainability, correctness

1. Separated code into Controller, Service, Model, DTO layers

2. Introduced strong typing with Cart & CartItem classes

3. Replaced doubles with BigDecimal for monetary accuracy

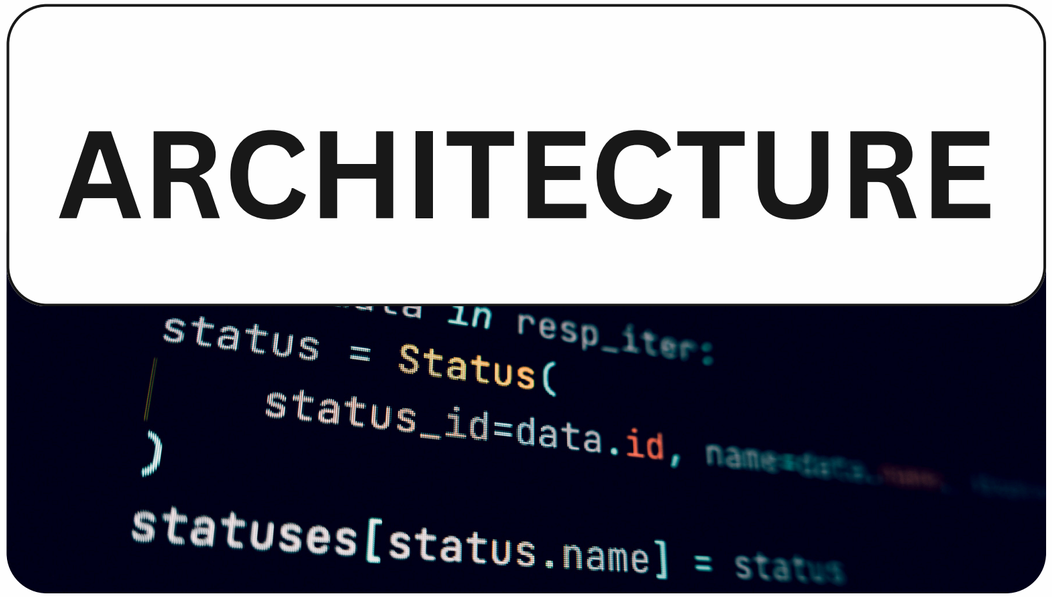
4. Added validation and structured error handling

5. Designed RESTful APIs with proper HTTP responses

# APPROACH

Java spring boot architecture

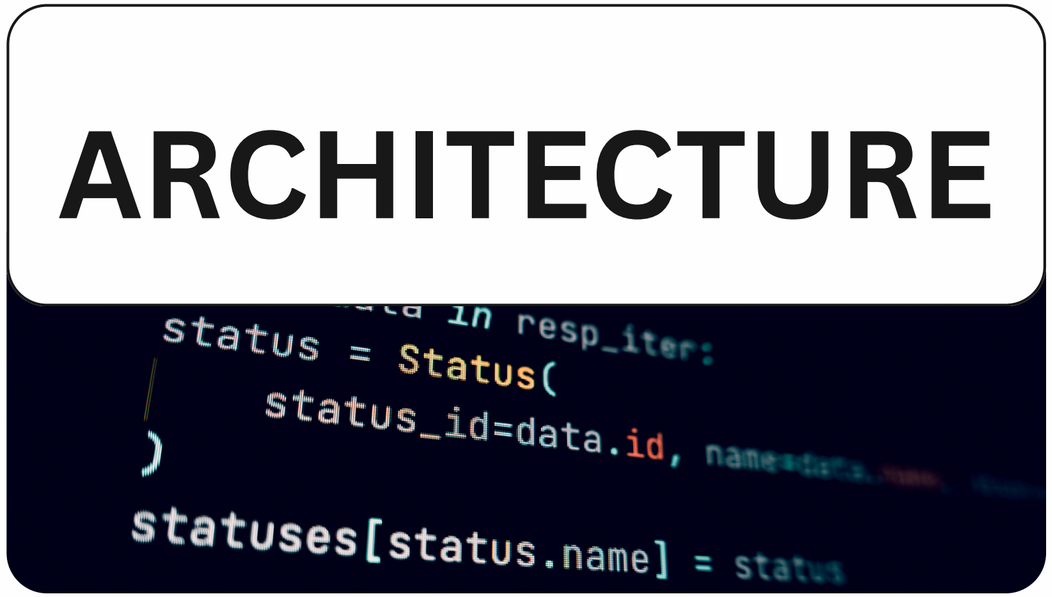
```
com.examemple.projectname
├─ controller  // API endpoints (REST)
├─ service  // Business logic
├─ repository  // Database layer (Spring Data JPA)
├─ model  // Entity/DTO classes
├─ config  // Custom configurations (e.g. security, CORS)
├─ exception  // Custom exceptions & handlers
├─ security  // JWT/OAuth2 setup
└─ utils  // Helper classes (converters, validators)
```




- **Benefits**
- **Clear separation of concerns (maintainability & correctness)**
- **Easy onboarding for new developers (readability)**
- **Scalable for microservices or monoliths**
- **Cleaner dependency injection and testing**

Refactored the code into a layered architecture
***(Controller, Service, Model, DTO)***

```
shopping-cart/
├── controller/           # REST controllers
├── service/              # Business logic
├── model/                # Domain models (Cart, CartItem)
│   └── exception/        # Custom exceptions
├── dto/                  # Response DTOs
├── resources/            # Application resources
└── ShopApplication.java # Spring Boot entry point
```




- **Controller Layer** – Handles HTTP requests (**ShoppingCartController**)
- **Service Layer** – Core business logic (**ShoppingCartService**)
- **Model Layer** – Domain objects (**Cart, CartItem**)
- **DTO Layer** – API responses (**CartResponse**)
- **Exception Layer** – Centralized error handling (**CartNotFoundException**)

# KEY IMPROVEMENTS

## Improvements

- Clean separation of concerns
- Immutability with Java records
- RESTful API design with proper HTTP codes
- Centralized exception handling
- Extensible for future features (e.g., remove items, discounts)


## Key Changes

- **Endpoints renamed:**
  - From addItem → /shop/{cartId}/addItems
  - From getTotal → /shop/{cartId}/total
- Path Variables introduced: cartId is now part of the URL path (RESTful style).
- **Responses changed:**
  - Originally: plain strings ("Item added. Total: X")
  - Now: structured JSON DTOs (CartResponse).
- **HTTP status codes:**
  - 201 Created for adding items
  - 200 OK for retrieving totals
  - 400 Bad Request for invalid inputs
  - 404 Not Found if the cart doesn’t exist

# FUTURE IMPROVEMENTS

## Improvements

- Add persistence (database integration)
- Introduce unit and integration testing
- Add Swagger/OpenAPI documentation
- Implement discount/promotion engine
- Improve concurrency handling further

Overview | Approach | Architecture | Improvements

# THANK YOU


***Get In Touch***

+27 67 229 8553

fanelesibongembuyazi0@gmail.com

https://www.linkedin.com/in/fanelesibonge-mbuyazi-036934193/