

Peas Pte. Ltd. Presents:

SHOPPEAS

Fresh to a Fault

SC2006 SDAC Group 5

Lee Jedidiah	(U2320848H)
Winnie Koh	(U2322135J)
Yan Jun Chao	(U2321341A)
Rachel Tan	(U2320358K)
Saffron Lim	(U2321216J)
Eng Yi Xuan	(U2321305F)


Introducing the team

Peas Pte. Ltd.

Jedidiah Lee
Backend Dev

Saffron Lim
Backend Dev

Rachel Tan
Full Stack Dev

Yan Jun Chao
Frontend Dev

Winnie Koh
Frontend Dev

Eng Yi Xuan
Frontend Dev

Presentation Agenda

1. INTRODUCTION

2. SOLUTION

3. SYSTEM DESIGN

4. SOFTWARE TRACEABILITY

5. FURTHER IMPROVEMENTS

6. CONCLUSION

Presentation Agenda

1. **INTRODUCTION**
2. SOLUTION
3. SYSTEM DESIGN
4. SOFTWARE TRACEABILITY
5. FURTHER IMPROVEMENTS
6. CONCLUSION

Challenges Surrounding Wholesale Businesses

Inconvenience

Inaccessibility

Wholesalers primarily sell large quantities to businesses and F&B establishments, **limiting the opportunity** of individuals to purchase

Wholesalers often **lack user-friendly online stores** for individuals, if any, making dicrect purchases from wholesalers **inconvenient**

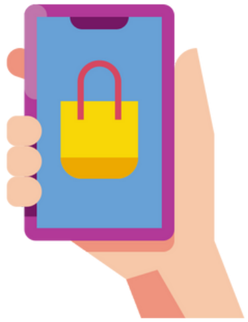
Typically focused on B2B transactions, leaving customers with **limited ways** to buy directly from wholesalers

There is **limited information** on product availability online and consumers may find it challenging to know what products are available

Consumers face significant barriers in accessing healthy food options from wholesalers due to inconvenient purchasing processes and limited access channels

Our Smart Nation Initiative

Optimizes food distribution to **promote sustainability** and r**educe barriers** to healthy options



Enables transactions between consumers and wholesalers, **supporting** Singapore's digital economy and local businesses

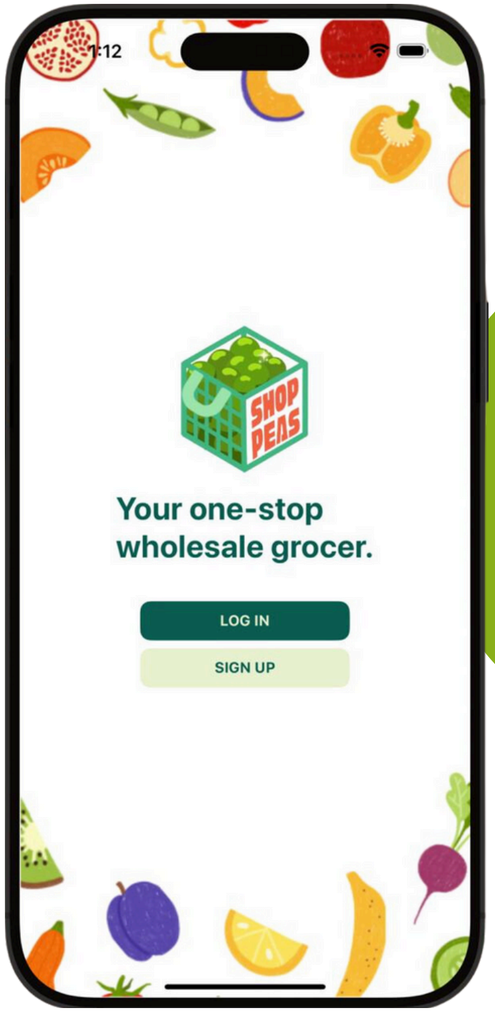
Improves access to healthier food options, **enhancing** the health and well-being of Singaporeans

Aligned with Singapore's Smart Nation Vision, our project aims to tackle key challenges in sustainability, digital accessibility and public health

Presentation Agenda

1. INTRODUCTION
2. **SOLUTION**
3. SYSTEM DESIGN
4. SOFTWARE TRACEABILITY
5. FURTHER IMPROVEMENTS
6. CONCLUSION

Shoppeas



A one-stop platform that connecting consumers with wholesalers to provide convenient access to healthy food options

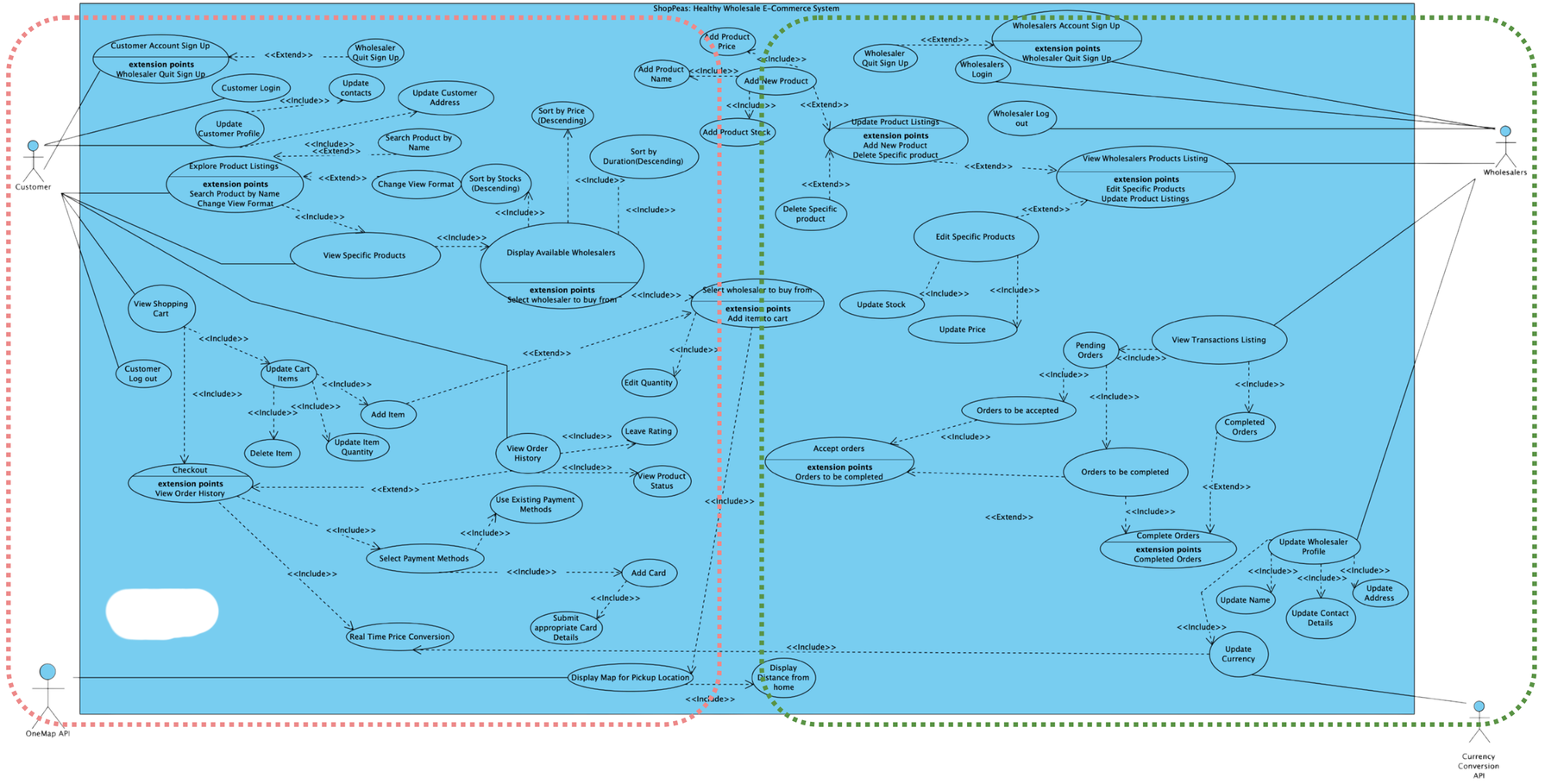
Direct Access To Wholesalers
Users can easily browse and view various wholesaler products through our app

Cross-Border Accessibility
Singaporean consumers are able to purchase from Malaysian Wholesalers

Location-Based Services
Utilize real-time data to show the distance from specific wholesalers, enhancing convenience

Data Security
All personal and payment information is safeguarded with our security protocols

Use Case Diagram



Consumers

- **Login/Register**
- **Update User Profile**
- **View Product Listings**
- **Shopping Cart**
  - View cart
  - Add Item
  - Delete Item
  - Update Item Quantity
  - Checkout Shopping Cart
- **View Specific Product Details**
- **Payment Methods**
  - View Card(s)
  - Add Card
  - Remove Card
- **Order History**
  - View Order History
  - Rate Order

Wholesalers

- **Login/Register**
- **Update User Profile**
- **Transactions**
  - View Pending/Completed Orders
  - Accept Order
- **Wholesaler's Products**
  - View Products
  - Add Product
  - Remove Product
  - Edit Product Details

External Services

- **OneMap API**
- **Currency Conversion API**

Dataset

DATA.GOV.SG

Healthier Choice Symbol Product List from HPB

- 2 fields: product name & package size

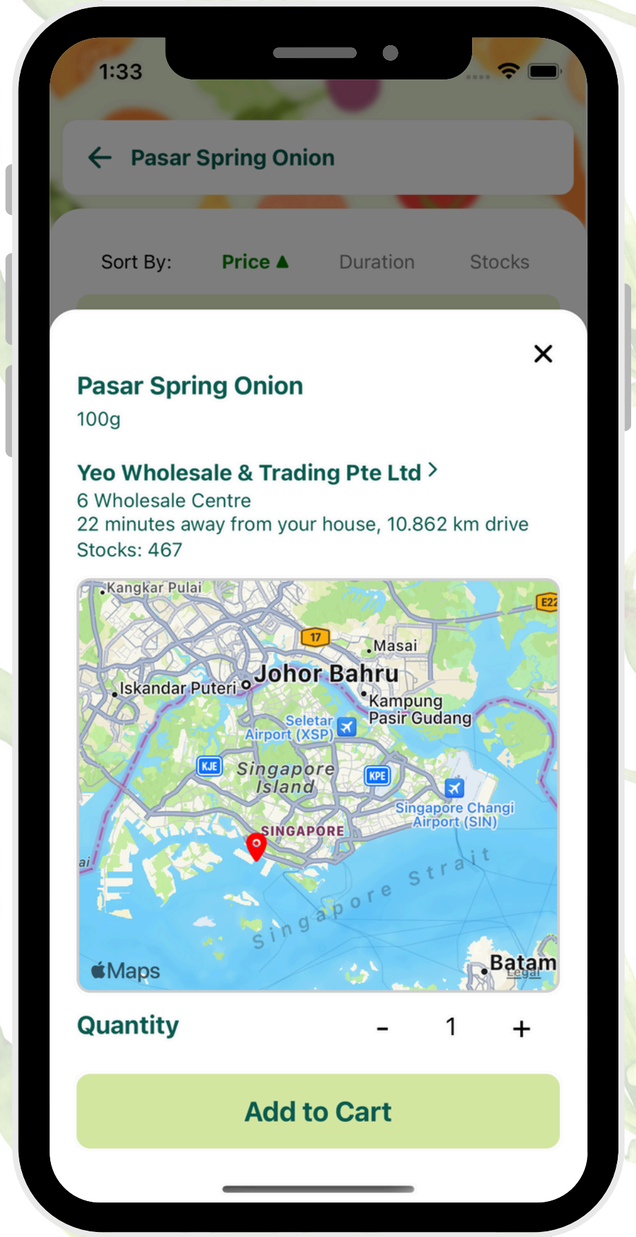
Real-Time APIs

Singapore's OneMapAPI

- Live mapping for user and wholesaler locations
- Calculate exact distance and travel time

Currency Conversion API

- Real time currency conversion for accurate pricings



Tech Stack

PRESENTATION

React Native

Expo

APPLICATION

NPM

SpringBoot

Firebase Authentication

Axios

BUSINESS

Javascript

Java

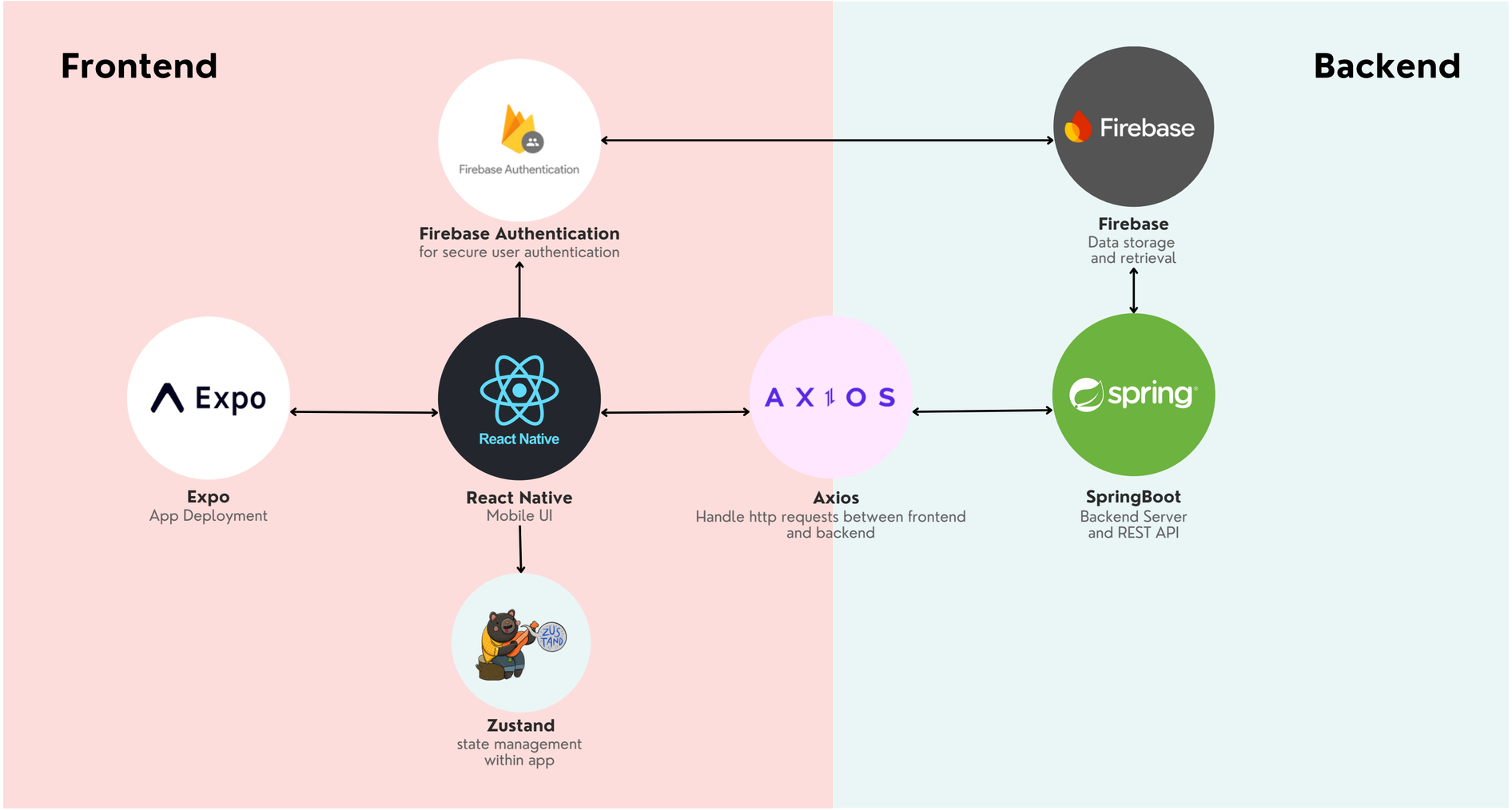
PERSISTENCE

Zustand

DATABASE

Firebase

Tech Stack


Frontend
Backend
Firebase Authentication
Firebase Authentication
for secure user authentication
Firebase
Firebase
Data storage
and retrieval
Expo
Expo
App Deployment
React Native
React Native
Mobile UI
AXIOS
Axios
Handle http requests between frontend
and backend
spring
SpringBoot
Backend Server
and REST API
Zustand
state management
within app

Live Demo

Presentation Agenda

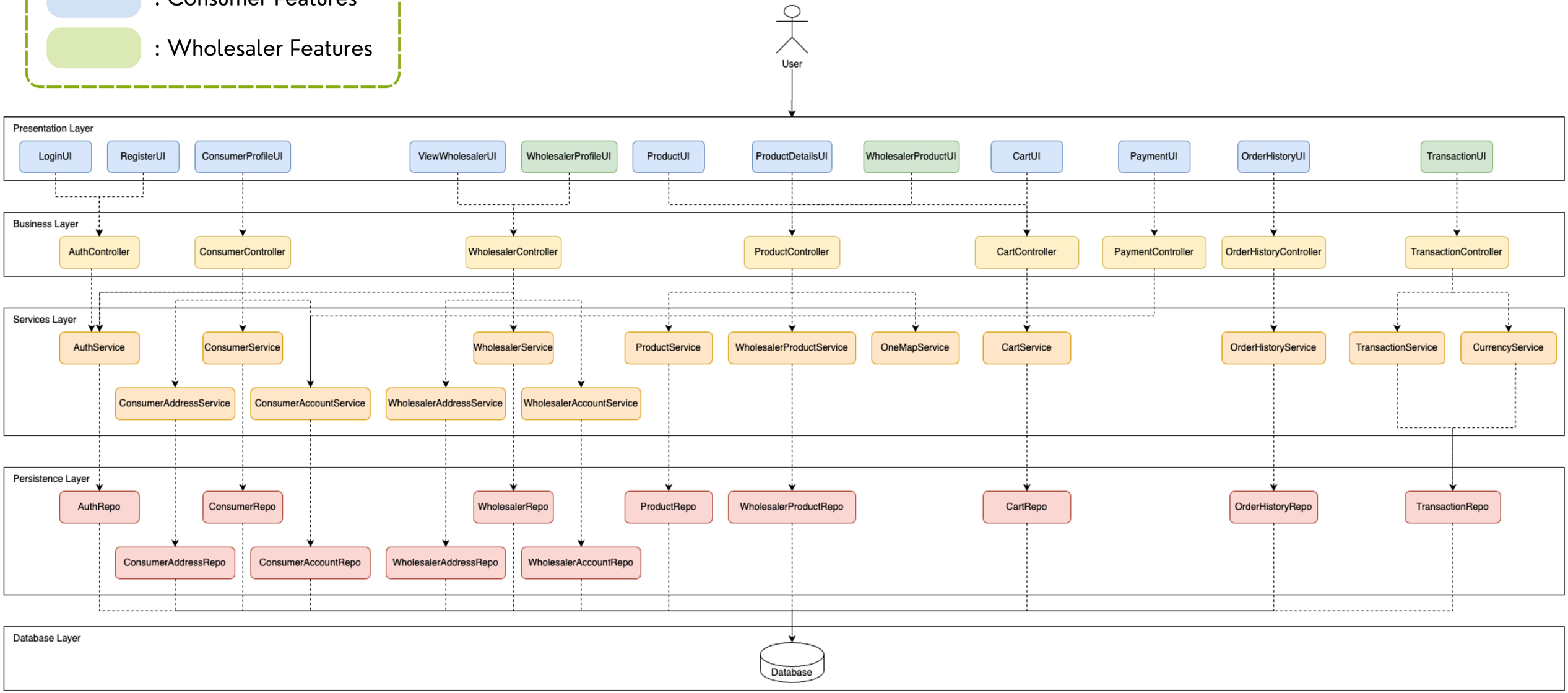
1. INTRODUCTION
2. SOLUTION
3. **SYSTEM DESIGN**
4. SOFTWARE TRACEABILITY
5. FURTHER IMPROVEMENTS
6. CONCLUSION

System Architecture

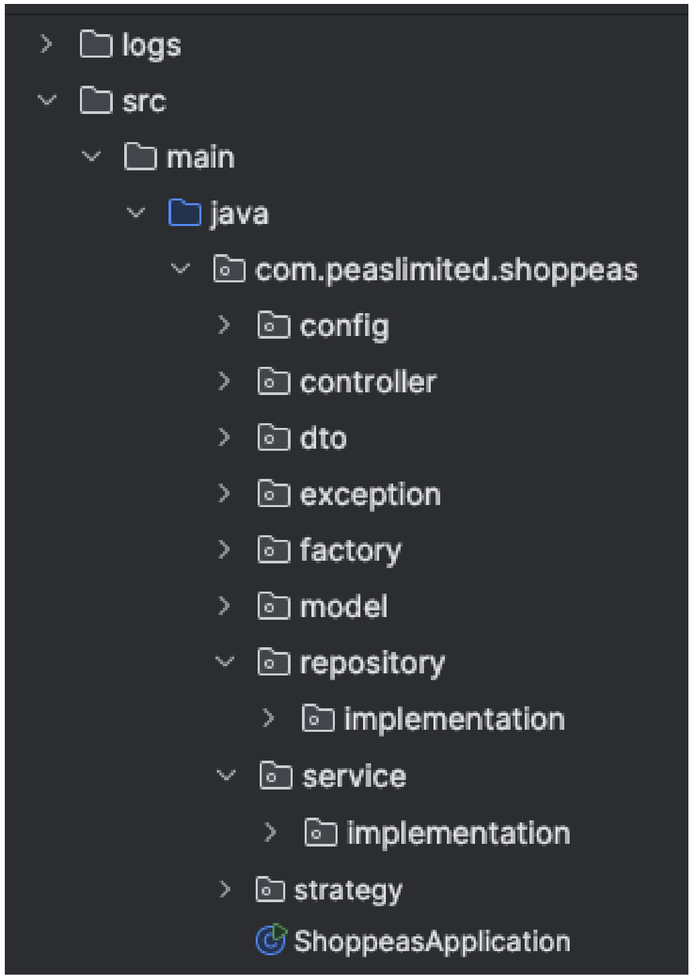
Legend:

: Consumer Features

: Wholesaler Features



Design Patterns



MVC, CSR & DTO Pattern

MVC Pattern

- **Model**: models folder
- **View**: frontend pages & components
- **Controller**: controller folder

Controller-Service-Repository Pattern

- **Purpose**: Separation of concerns, Easier Testability, Scalability, Reusability
- **Controller**: Handles incoming HTTP requests
- **Service**: Application's business logic, processes data from the controller and communicates with the repository to fetch or persist data
- **Repository**: Data access and persistence, encapsulates querying, adding, and updating logic

Data Transfer Object (DTO) Pattern

- **Situation**: Encapsulation; prevent sending confidential information (e.g., user id) to the client
- **Implementation**: Build different views based on our model design

Consumer Model

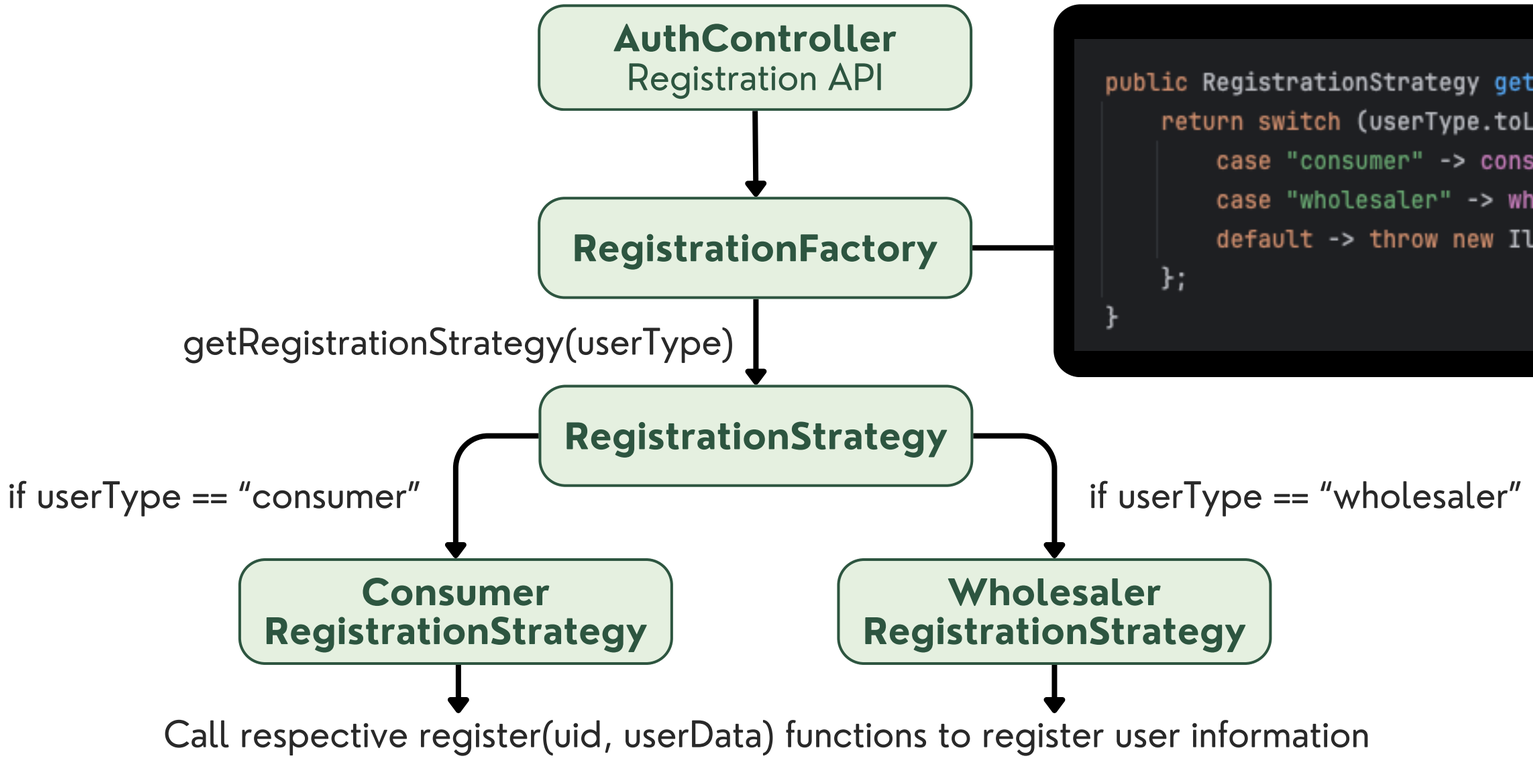
```
public class Consumer {
    @Id
    private String UID;
    private String first_name;
    private String last_name;
    private String email;
    private String phone_number;
```

```
public class ConsumerDTO {
    private String first_name;
    private String last_name;
    private String email;
    private String phone_number;
}
```

Consumer DTO

Design Patterns

Strategy & Factory Pattern




```
public RegistrationStrategy getRegistrationStrategy(String userType) {  4 usages  Rachel Tan
    return switch (userType.toLowerCase()) {
        case "consumer" -> consumerRegistrationStrategy;
        case "wholesaler" -> wholesalerRegistrationStrategy;
        default -> throw new IllegalArgumentException("Unknown user type: " + userType);
    };
}
```

Factory Pattern

- **Situation**: Multiple user types with potentially more users to add
- **Implementation**: Instantiate strategy object during runtime when a registration request is created

Strategy Pattern

- **Situation:** Registration process varies based on the user type
- **Implementation:** Switch registration strategy based on user type

Design Principles

Single Responsibility Principle (SRP)

- Each entity has its own class (service, repository etc)
- Minimise the number of reasons to change a class

Open-Closed Principle (OCP)

- Factory & Strategy Pattern allows for easy extension without modification upon the addition of new user roles

Interface Segregation Principle (ISP)

- Created specific interfaces for each service, repository, strategy for clients to access only the relevant interfaces
- Reduce issues caused by changes

Dependency Injection Principle (DIP)

- Usage of @Autowired across services, repository to separate concerns between object construction and usage
- Loosely coupled application

Non-Functional Requirements

Efficiency | Reliability | Robustness | Maintainability | Security

```java
// Get wholesaler info
List<WholesalerDTO> wholesalers = wholesalerRepository.findWholesalers(uen_list);

// Get wholesaler address info
List<WholesalerAddress> wholesalerAddresses = wholesalerAddressRepository.findAllWholesalerAddress(wholesalerProducts);
// Combine product and wholesaler data into DTOs
List<CompletableFuture<WholesalerProductDetailsDTO>> futures = new ArrayList<>();

for (int i = 0; i < wholesalerProducts.size(); i++) {
    WholesalerProducts product = wholesalerProducts.get(i);
    WholesalerDTO wholesaler = wholesalers.get(i);
    WholesalerAddress address = wholesalerAddresses.get(i);
    String swp_id = swp_id_list.get(i);
    String uen = uen_list.get(i);

    CompletableFuture<WholesalerProductDetailsDTO> future = CompletableFuture.supplyAsync(() -> {
        try {
            String userCoordinates = oneMapService.getCoordinates(userPostalCode);
            String wholesalerCoordinates = oneMapService.getCoordinates(address.getPostal_code());
            LocationDTO travelStats = oneMapService.calculateDrivingTime(userCoordinates, wholesalerCoordinates);
```

Java CompletableFuture:

- Asynchronous computation of intensive API requests
- Reduced computation time to ensure quicker loading of resources

Non-Functional Requirements

Efficiency | **Reliability** | **Robustness** | Maintainability | Security

```java
@RestControllerAdvice  Rachel Tan +1
public class ApiExceptionHandler extends ResponseEntityExceptionHandler {

    private static final Logger log = LogManager.getLogger(ApiExceptionHandler.class);  3 usages

    // Handle firebase authentication errors
    @ExceptionHandler(FirebaseAuthException.class)  no usages  Rachel Tan
    public ResponseEntity<Object> handleBadRequestException(FirebaseAuthException exception) {
        // Response Status
        HttpStatus badRequest = HttpStatus.BAD_REQUEST;
        // Response Body
        ApiException errorResponse = new ApiException(exception.getMessage(), badRequest, LocalDateTime.now());
        // Log error
        log.error( message: "{} Error due to unsupported request: {}", badRequest, exception.getMessage());
        // Return response entity
        return new ResponseEntity<>(errorResponse, badRequest);
    }
```

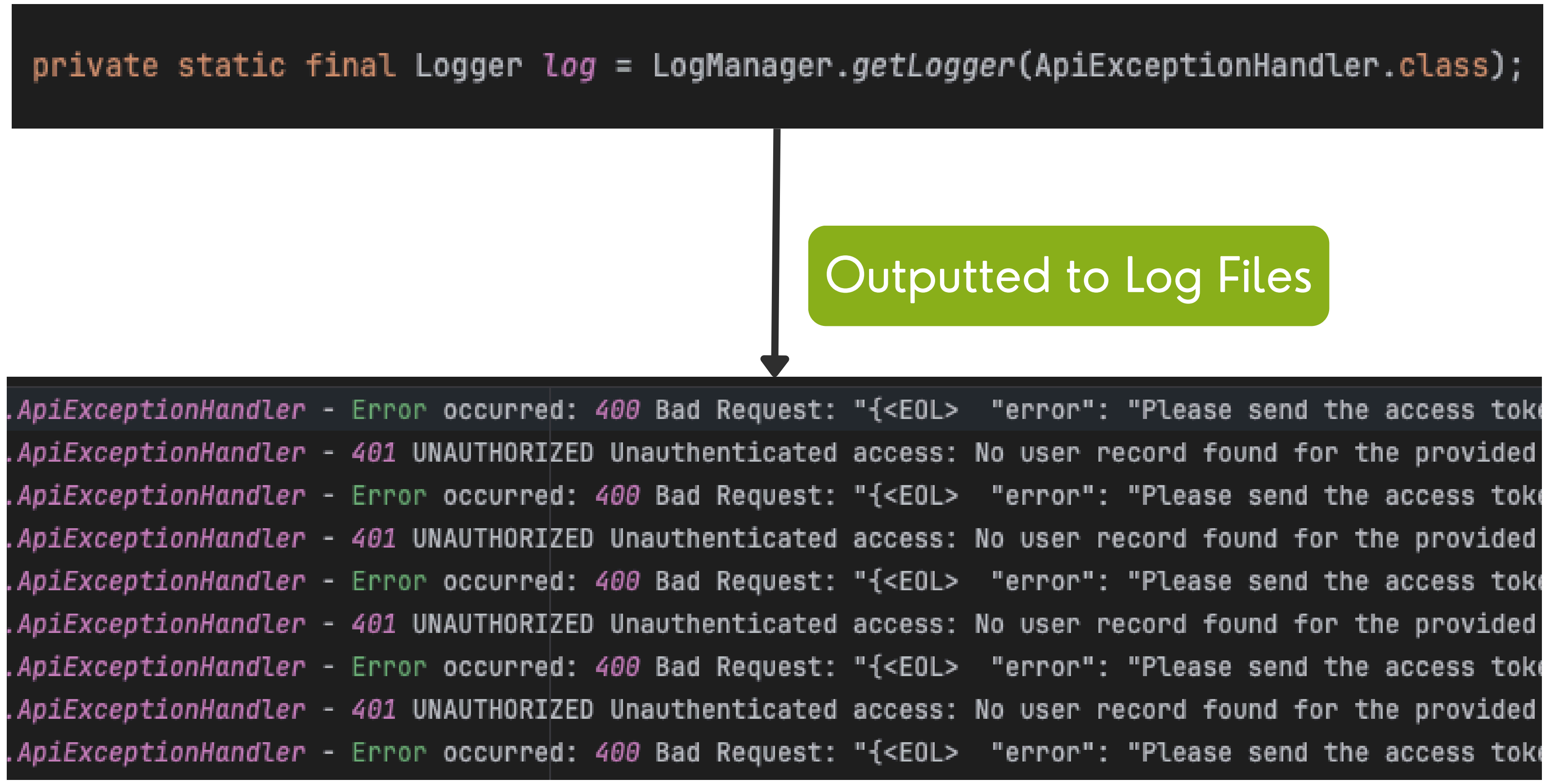
Handling a specific Firebase Authentication Exception

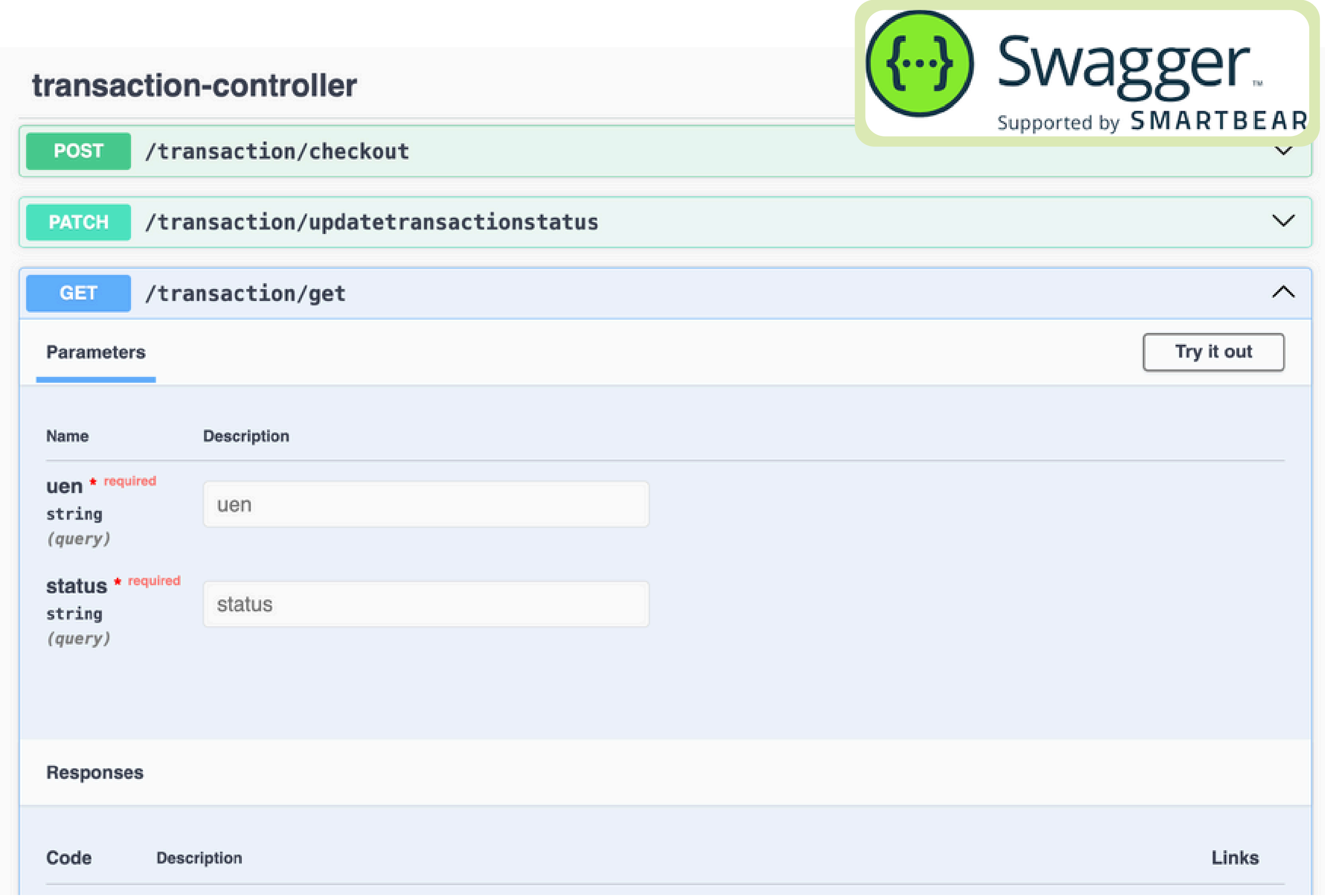
- **Exception Handling (both front & back-ends)**: handle specific and general exceptions to ensure that the program is fault tolerant & does not crash

Non-Functional Requirements

Efficiency | Reliability | Robustness | **Maintainability** | Security

- **Logging with Apache Log4j 2 (back-end)**: to capture errors for troubleshooting, performance monitoring, and understanding the flow of data
- **Documentation:** Javadocs (back-end functions) and Swagger UI (API documentation)





Non-Functional Requirements

Efficiency | Reliability | Robustness | Maintainability | **Security**

- **Stored the API Keys in env files**
- **Role-based Authorisation** with **Spring Method Security**: validated the user's role through the Authorization header to ensure only users of a certain role can access certain APIs to minimise the risk of data breaches

```java
/**
 * get wholesalers by pid for consumers
 * @return List of wholesalers
 */
@GetMapping("/{pid}") no usages  Rachel Tan
@PreAuthorize("hasRole('CONSUMER')")
@ResponseStatus(code = HttpStatus.OK)
public List<WholesalerProductDetailsDTO> getWholesalersByPid(@PathVariable String pid) throws ExecutionException, InterruptedException,
    // Get UID
    Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
    String uid = (String) authentication.getPrincipal();
    return wholesalerProductService.findByPid(pid, uid);
}
```

Function in ProductController.java: handles API calls related to products

Presentation Agenda

1. INTRODUCTION
2. SOLUTION
3. SYSTEM DESIGN
4. **SOFTWARE TRACEABILITY**
5. FURTHER IMPROVEMENTS
6. CONCLUSION

Traceability

Add to Cart

Functional Requirements

Use case diagrams and descriptions

Requirements Analysis

Class and sequence diagrams

Implementation

Design principles

Testing

Basis Path Testing

Add to Cart

Functional Requirements

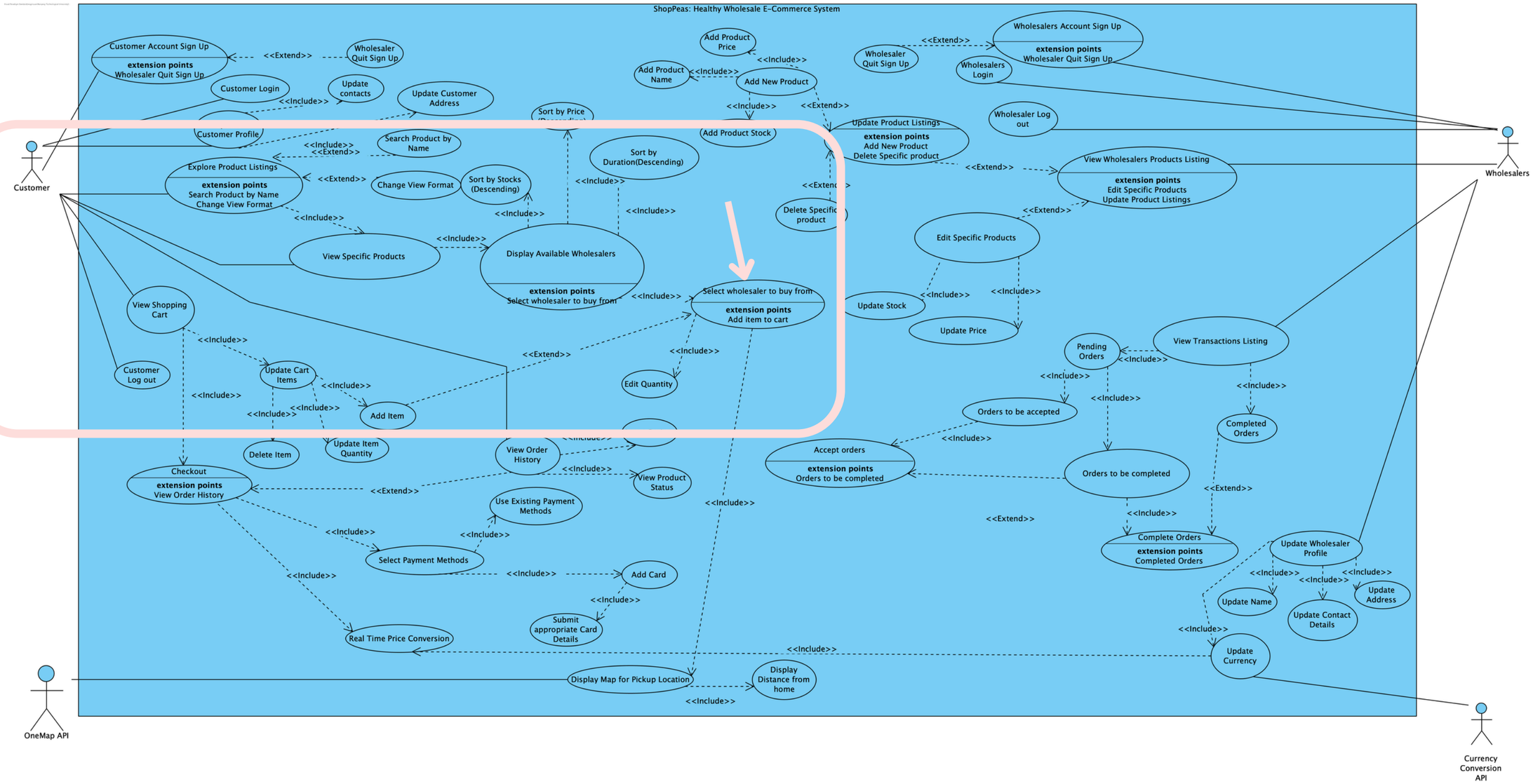
Use Case Description

Use Case ID:	CUS04		
Use Case Name:	Product Details		
Created By:	Rachel	Last Updated By:	Saffron Lim
Date Created:	27/08/2024	Date Last Updated:	20/10/2024

Actor:	Consumer
Description:	Displays the details of the healthy food product the user selected, such as the available wholesalers, price, and quantity for them to make an informed decision before purchasing the product. Through this page, the user can add products to their shopping cart.
Preconditions:	1. The user must be logged in. 2. The user selects a product from the Product page.
Postconditions:	
Priority:	High
Frequency of Use:	High
Flow of Events:	1. The system queries the database to get all wholesalers sellin the product based on the product selected on the Product page. a. For each wholesaler, the system will call the OneMap API to get the tota pickup location and t i. The time will 2. The user reviews the list of v a. The user can sort the i. The system v chosen field. ii. The system w b. For each option, wholesaler's name t wholesaler. i. The wholesal ii. The user wi "Wholesaler
	3. The user selects an option. a. The option is highlighted in a different colour. b. A pop-up screen appears and the user indicates the quantity they want to add to the cart. A map of the wholesaler's location is also shown using the OneMap API. i. If the user selects the "Add to Cart" button, the Shopping Cart page is updated with the addition of this product. ii. The shopping cart and transaction records in the database are updated accordingly 4. The user clicks the return button to continue shopping on the Product page.
Alternative Flows:	NIL
Exceptions:	EX01-CUS04: System unable to retrieve information 1. The System displays the message “Unable to retrieve products, check your connection”.
Includes:	NIL
Special Requirements:	NIL
Assumptions:	NIL
Notes and Issues:	NIL

Add to Cart Use Case Diagram

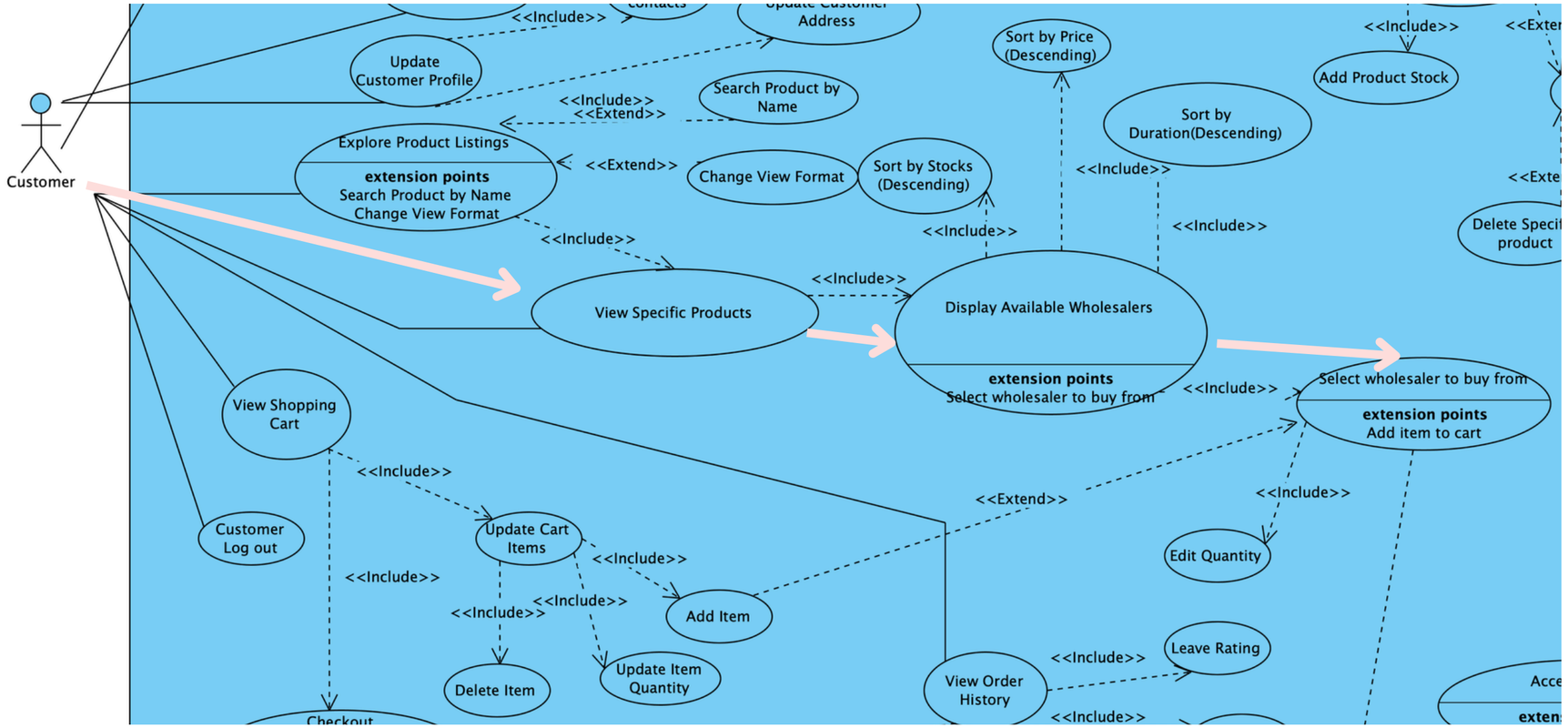
Functional Requirements



Add to Cart

Functional Requirements

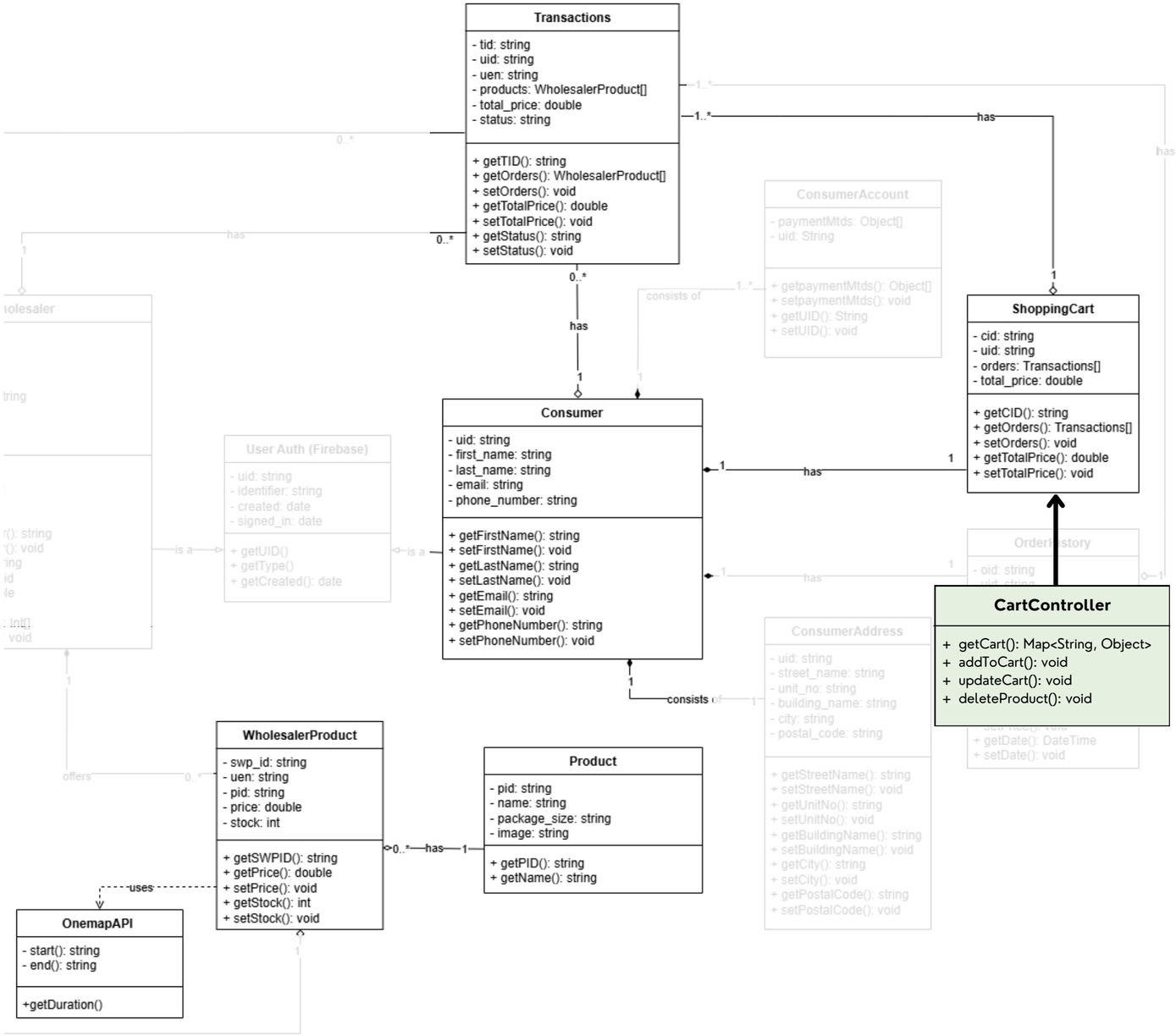
Use Case Diagram

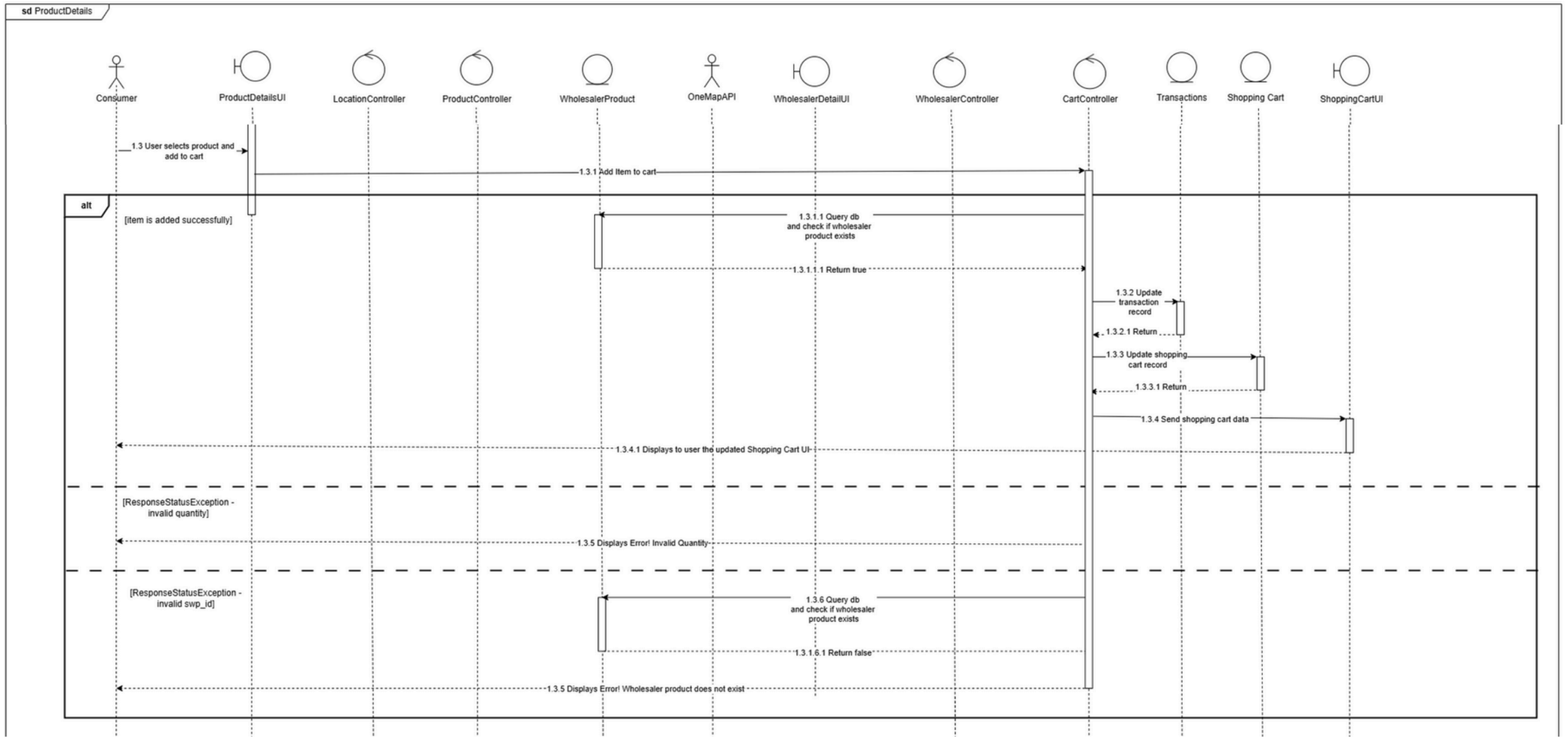


Add to Cart

Requirements Analysis

Class Diagram

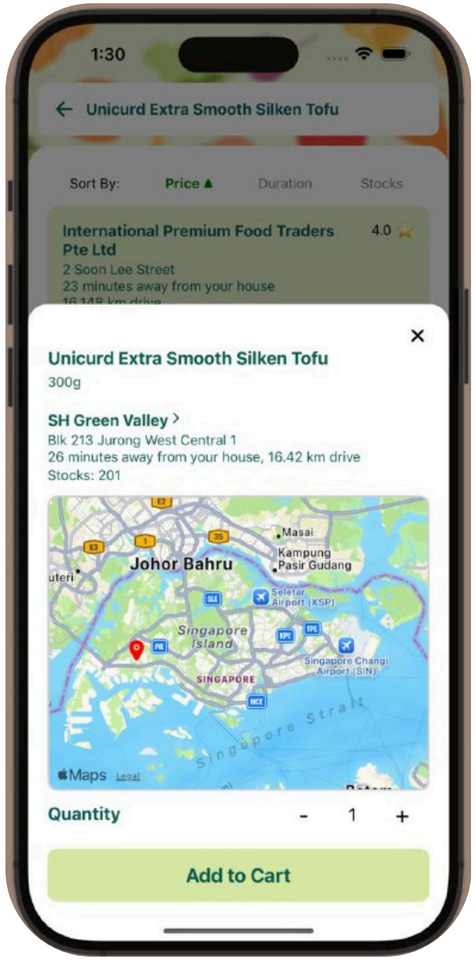


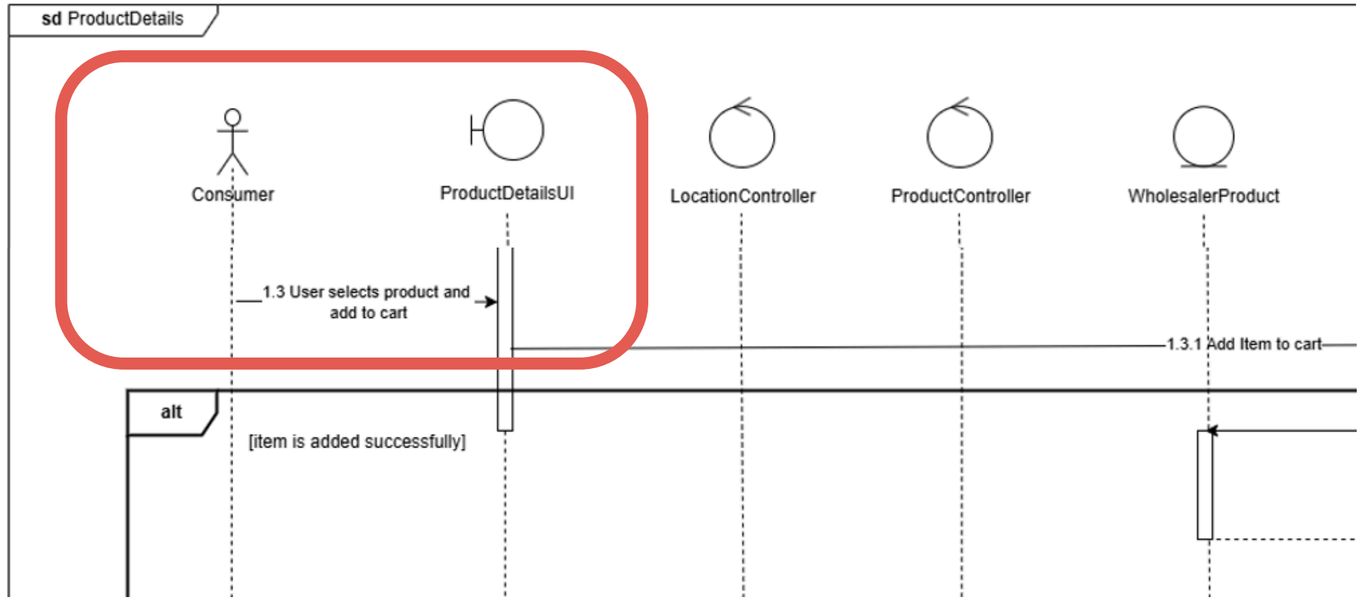


Add to Cart

Sequence Diagrams

Requirements Analysis

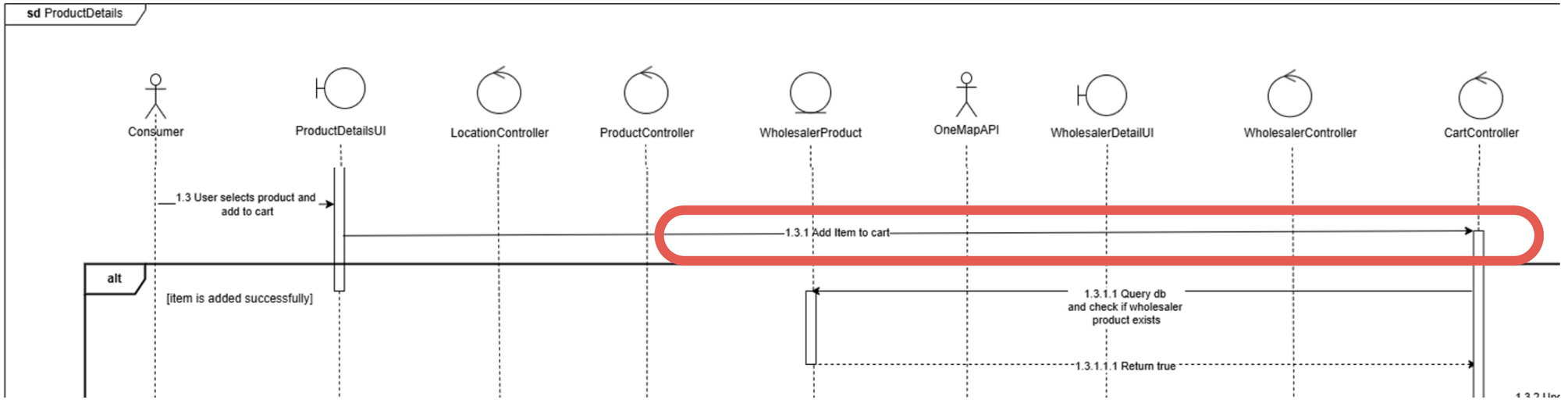




Add to Cart

Sequence Diagrams

Requirements Analysis

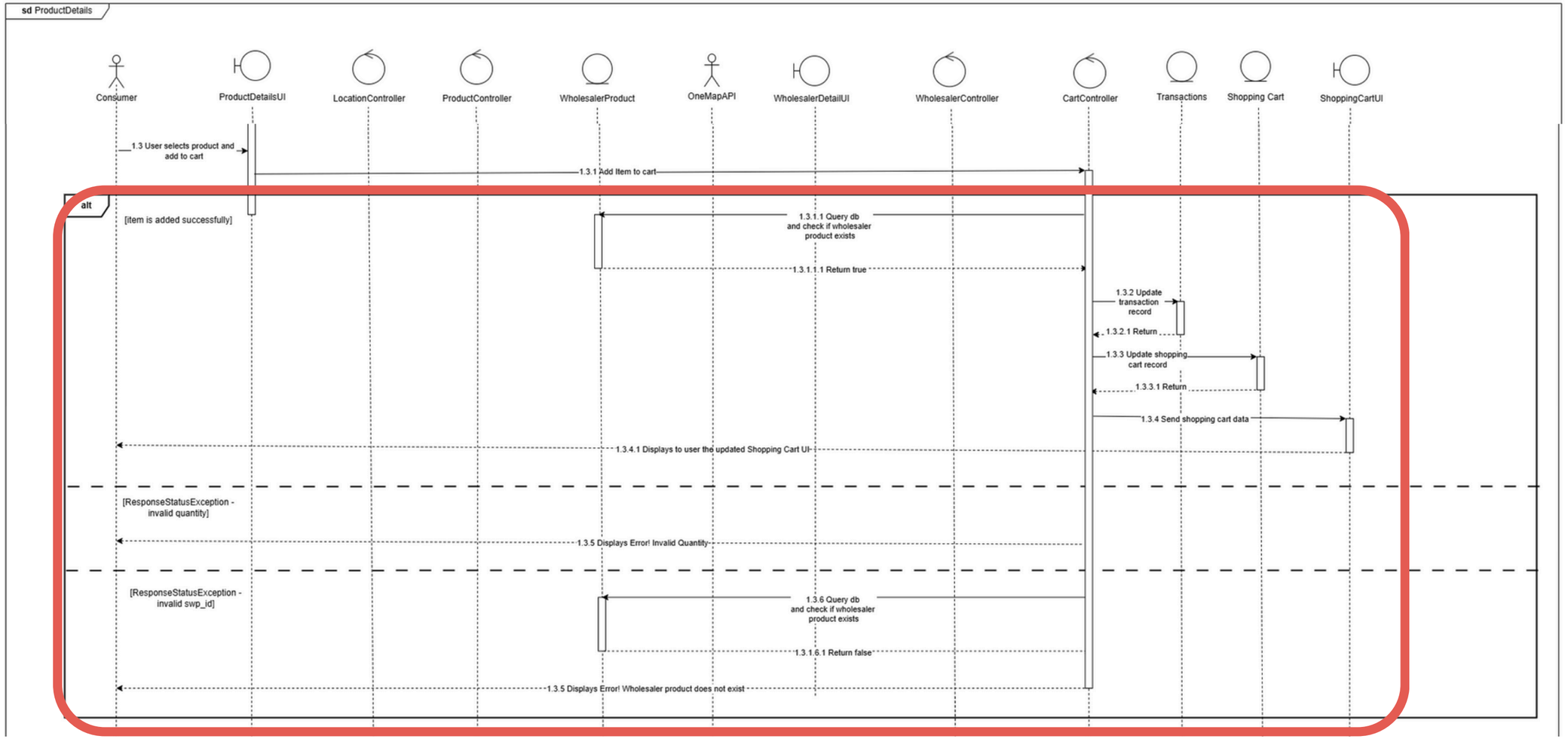


Add to Cart

Sequence Diagrams

Requirements Analysis

```
{
    "swp_id": "3q1VqrKuNAkpFxIvziTd",
    "uen": "53474231M",
    "quantity": 10,
    "total_price": 304.6,
    "price": 30.46
}
```



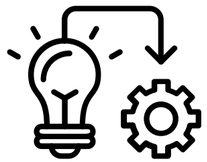
Add to Cart

Sequence Diagrams

Requirements Analysis

SpringBoot

Firebase



Add to Cart

Implementation

Design Principles

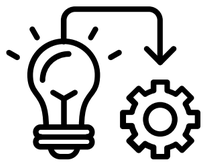
Controller-Service-Repository Pattern

- CartController
- CartService
- CartServiceImpl
- CartRepository
- CartRepositoryImpl

Single Responsibility Principle

Interface Segregation Principle

Add to Cart



Implementation

Design Principles

Data Transfer Object

```java
public class ShoppingCart {
    @Id
    String cid;
    ArrayList<String> orders;
    String uid;
    double total_price;
}
```

```java
public class ShoppingCartDTO {
    ArrayList<String> orders;
    String uid;
    double total_price;
}
```

Dependency Injection Principle

```java
public class CartController {

    @Autowired  4 usages
    private CartService cartService;

    @Autowired  1 usage
    private WholesalerService wholesalerService;

    @Autowired  1 usage
    private CartRepository cartRepository;

    @Autowired  2 usages
    private WholesalerProductService wholesalerProductService;
```

Add to Cart 

Testing Test Case Description & CFG

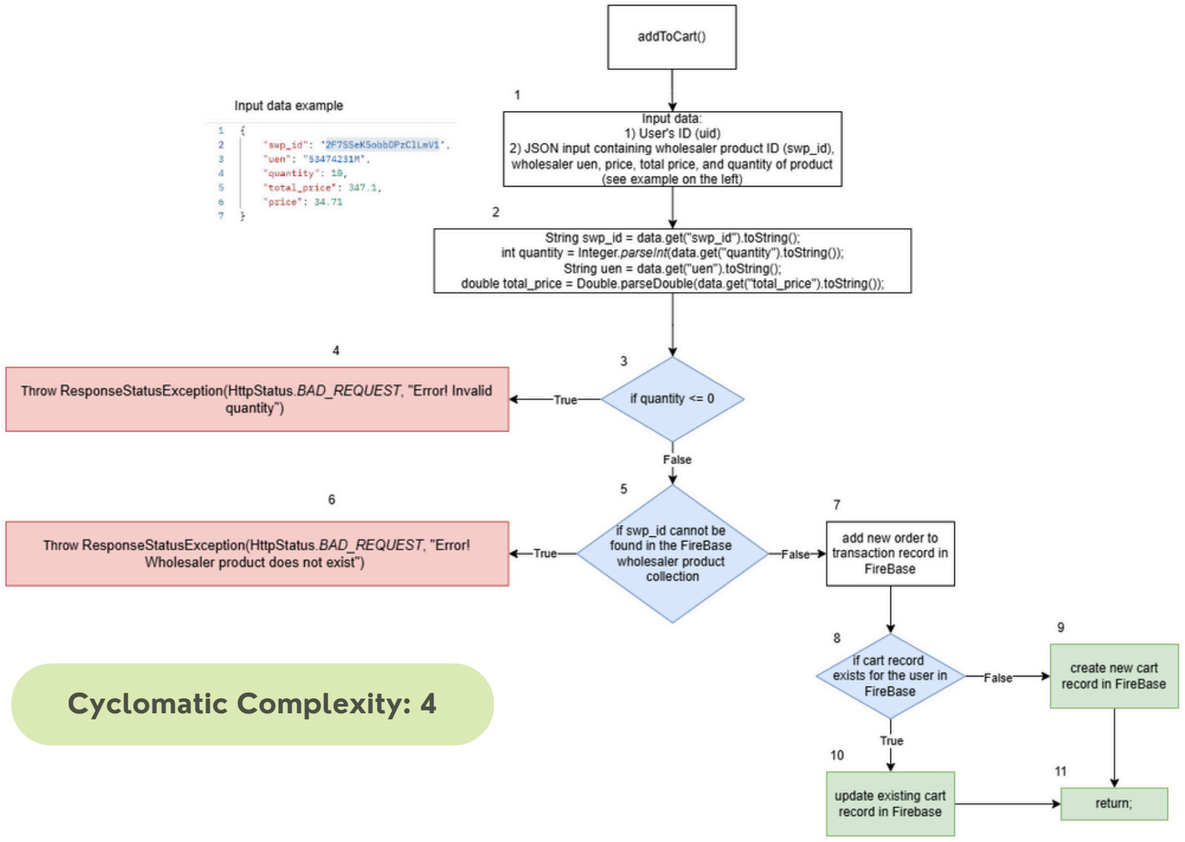
Test id	Description
2.1a	1. Valid JSON input that represents the addition of one product to cart without any discrepancies for any fields. ○ The product must have a positive quantity. ○ Valid authorization key (consumer ID, typical case). 2. A cart record for the consumer does not exist in FireBase (i.e., this is the first time the consumer is adding a product to cart). 3. After calling addToCart(), a new transaction record in FireBase will be created with the status “IN-CART”. ○ The transaction record will reflect the new product added. 4. After calling addToCart(), a new shopping cart record in FireBase will be created. ○ The shopping cart record will reflect the new product added. 5. HTTP Status of 201 will be received
2.1b	1. Valid JSON input that represents the addition of one product to cart without any discrepancies for any fields. ○ The product must have a positive quantity. ○ Valid authorization key (consumer ID, typical case). 2. A cart record for the consumer already exists in FireBase (i.e., the consumer has added a product to cart before). 3. After calling addToCart(), the transaction records in FireBase for the consumer will be updated. ○ The transaction record will reflect the new product added. 4. After calling addToCart(), the shopping cart record in FireBase will be updated. ○ The shopping cart record will reflect the new product added. 5. HTTP Status of 201 will be received
2.1c	1. Invalid JSON input that represents the addition of one product to cart with a discrepancy in the “quantity” field only. ○ The product has a negative or zero quantity. 2. HTTP Status of 400 will be received 3. No updates will be made to the consumer’s shopping cart and transaction record in FireBase
2.1d	1. Invalid JSON input that represents the addition of one product to cart with a discrepancy in the “swp_id” field only. ○ The wholesaler product represented by the swp_id does not exist. 2. HTTP Status of 400 will be received 3. No updates will be made to the consumer’s shopping cart and transaction record in FireBase

Test case	swp_id	uen	quantity	total_price	price	Expected HTTP status	Actual HTTP status
Valid Input Cart DNE	3q1VqrKuNAkpFxlvziTd	53474231M	10	304.6	30.46	201	201
Valid Input Cart Exists	3q1VqrKuNAkpFxlvziTd	53474231M	10	304.6	30.46	201	201
Invalid quantity	3q1VqrKuNAkpFxlvziTd	53474231M	-10	304.6	30.46	400	400
Invalid swp_id	123	53474231M	10	304.6	30.46	400	400

Add to Cart

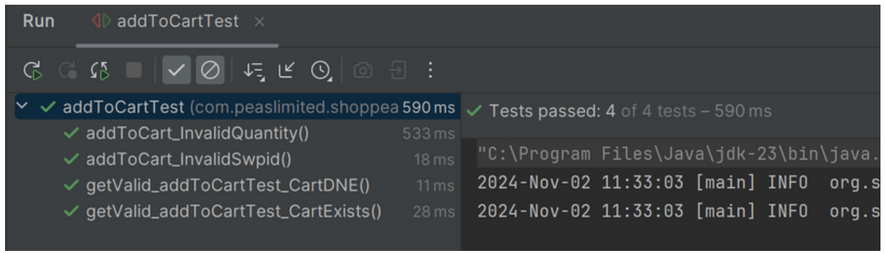
Testing

Basis Path Testing



Cyclomatic Complexity: 4

	Test Script	backend/src/test/java/com/peaslimited/shoppeas/controller/addToCartTest.java		
Test id	Basis Path	**Basis Path Testing** Inputs	Expected output	Actual output
		JSON Input Data		
2.1a	1,2,3,5,8,9,11	{ "swp_id": "3q1VqrKuNAkpFxlvziTd", "uen": "53474231M", "quantity": 10, "total_price": 304.6, "price": 30.46 } • Default correct JSON input (shown above) • Cart record does not exist for the user	201 Cart and transaction records in FireBase reflect the new product added.	201 Cart and transaction records in FireBase reflect the new product added.
2.1b	1,2,3,5,8,10,11	{ "swp_id": "3q1VqrKuNAkpFxlvziTd", "uen": "53474231M", "quantity": 10, "total_price": 304.6, "price": 30.46 } • Default correct JSON input (shown above) • Cart record already exists for the user	201 Cart and transaction records in FireBase reflect the new product added.	201 Cart and transaction records in FireBase reflect the new product added.
2.1c	1,2,3,4	{ "swp_id": "3q1VqrKuNAkpFxlvziTd", "uen": "53474231M", **"quantity": -10,** "total_price": 304.6, "price": 30.46 } • Invalid JSON input (shown above) • Quantity of product is less than or equal to zero	400 Throws Response Status Exception: Error! Invalid quantity	400 Throws Response Status Exception: Error! Invalid quantity
2.1d	1,2,3,5,6	{ **"swp_id": "123",** "uen": "53474231M", "quantity": -10, "total_price": 304.6, "price": 30.46 } • Invalid JSON input (shown above) • swp_id is invalid (wholesaler product does not exist)	400 Throws Response Status Exception: Error! Wholesaler product does not exist	400 Throws Response Status Exception: Error! Wholesaler product does not exist



Presentation Agenda

1. INTRODUCTION
2. SOLUTION
3. SYSTEM DESIGN
4. SOFTWARE TRACEABILITY
5. **FURTHER IMPROVEMENTS**
6. CONCLUSION

Future Improvements

Consumers

- Filter for dietary preferences
- AI-powered product recommendations
- Allow users to share product reviews with photos

Wholesalers

- Analytics Dashboard
- Inventory Management - Automated Stocks Alert
- Integration with logistics providers for deliveries
- Business verification

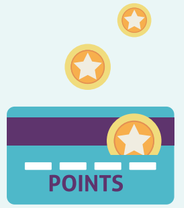
PEA LTD.

User Experience

- In-app chat between customers and wholesalers
- Notifications (order status updates)
- Multi language support
- Virtual tour for new users

Business Features

- Loyalty Program and point system for purchases
- Referral Program for new customers



Presentation Agenda

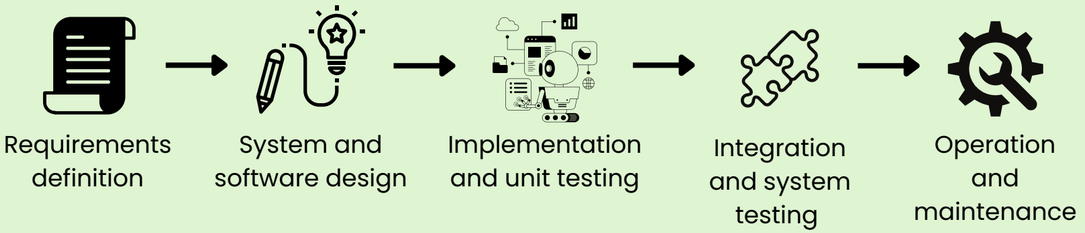
1. INTRODUCTION
2. SOLUTION
3. SYSTEM DESIGN
4. SOFTWARE TRACEABILITY
5. FURTHER IMPROVEMENTS
6. **CONCLUSION**

Conclusion

Takeaways

- Processes: Specification through SRS document, Use Case Diagram, Design and implementation, Validation and Evolution.

SDLC



Practices

- Adopted the best software engineering practices through OOP, design patterns, and principles to fulfil both functional & non-functional requirements.

SMART Nation

- Digital Society : Digitalisation of traditional wholesale businesses
- Supporting Singaporeans in adopting healthy lifestyles

Thank You!

Shoppeas: Your one-stop wholesaler grocer