

Clean Coding and Concurrent Programming

COMP63038

Assignment 2

Gampala Waduge Shemeera Siyumi Fonseka

CB013452

Submitted to

The School of Computing

Of

Asia Pacific Institute of Information Technology

Sri Lanka

Table of Contents

Abstract	3
Chapter 1 - Introduction	4
Chapter 2 – Clean Architecture.....	5
2.1 Multi-tier client-server Architecture	5
2.2 MVC Architecture	6
2.3 Allocation of Clean Architecture components	6
Chapter 3 - Concurrency	8
3.1 Server Side Concurrency	8
3.2 Client Side Concurrency	10
3.3 Summary	12
Chapter 5 - Automatic Test Clients.....	24
Chapter 6 - Test Cases	26
6.1 Junit Test Cases.....	26
6.2 Test Coverage	38
Chapter 7 – Postman Tests.....	39
Chapter 8 - Conclusion	42
Chapter 9 – Acknowledgement.....	43

Abstract

SYOS Billing System is a web based solution that is designed to automate a traditional billing process and enhance the efficiency through a structured modern software architecture and it enables the cashiers to process sales using unique item codes and quantities, automatically calculating totals, managing stock reductions and generating bills. The system supports both instore and online sales maintaining separate inventories and generating detailed reports for each. SYOS follows a clean architectural design with a multi-tier client server model containing presentation, application and data layers. It uses Model View Controller (MVC) pattern by using React for the frontend, Java Servlets for the controllers and MySQL for persistent data storage. And also the use of SOLID principles and key design patterns ensures modularity, maintainability and scalability of the system. Concurrency is managed effectively on both server and client sides and the server uses Java EE's built in multithreading and transaction handling to process simultaneous client requests securely while the react based frontend utilizes asynchronous operations and state management to maintain a responsive user interface. Even though it is a strong architectural foundation, the system's design highlights the need for careful boundary management, scalability planning and continuous optimization to support long term growth and maintain system robustness under high concurrency scenarios.

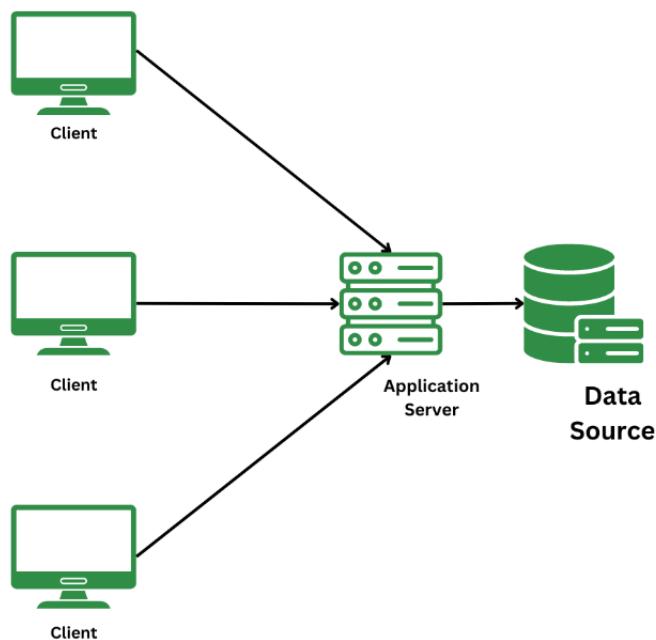
Chapter 1 - Introduction

In today's fast paced retail environment, grocery stores require efficient and accurate billing systems to streamline sales and inventory management. The SYOS billing system is developed to address these needs by replacing manual billing method with a modern automated solution. This web based solution enables cashiers to process transactions using unique item codes and quantities while automatically updating inventory and generating bills. And also it is designed to support both instore and online operations by ensuring seamless inventory tracking and sales reporting across multiple mediums and its structured architecture and the use of modern technologies make it a scalable and maintainable solution for grocery retailers aiming to enhance customer service and operational efficiency.

Chapter 2 – Clean Architecture

2.1 Multi-tier client-server Architecture

SYOS automated billing system follows a multi-tier client-server architecture specifically a three-tier architecture where the system is divided into 3 main layers such as presentation, application and data. The presentation layer is the client side web interface running in the browser where users interact with the system and this layer sends requests to the server to perform actions like saving a bill or registering a user. The application layer is the middle tier which acts as the brain of the system which receives requests from the client, applies business rules, process data and coordinates operations such as generating unique bill numbers or calculating totals and this layer ensures that the logic and processes of the system are centralized and managed efficiently. The data layer is responsible for storing and retrieving information in the database and it handles tasks like saving bill information, accessing customer records and updating inventory. By using this type of architecture in the SYOS system, improves maintainability, scalability and security and also each layer can be developed, tested and updated independently by making the system more flexible and robust.



2.2 MVC Architecture

The SYOS Billing system is built on a layered architecture with a strict separation of concerns. The Model Layer using the entity classes like Batch, Bill, Item, User and Order take care of data management and business logic. The Model Layer interacts with the MySQL database via DAOs that encapsulate the SQL calls. The View Layer is in React to provide a dynamic user interface which handles user interactions and performs updates based on responses from the backend. The Controller Layer uses Java Servlets to handle HTTP requests from the user interface that validate inputs and call the model methods. Service Layer contains core business logic for order processing and stock management and acts as a bridge between the controller layer and DAO layer.

2.3 Allocation of Clean Architecture components

The allocation of clean architecture components within the SYOS automated billing system's multi-tier client-server structure identifies a well considered effort to balance modularity and practical implementation but some critical observations can be seen when examining the approach in depth. While the clear separation between the presentation layer, controller layer, service layer and data access objects aligns well with the clean architecture's principles, some boundaries can be seen between these layers by risking unintended tight coupling. For example, the controller layer's dual role of parsing input and handling HTTP responses sometimes can lead to the mixing of concerns by violating the single responsibility principle if not carefully managed. Although the service layer effectively centralizes the business logic, the system's reliance on concrete DAO implementations may reduce flexibility in swapping out data sources or mocking dependencies for testing.

Integrating frontend react components with backend java servlets using JSON and Gson adapters presents a practical yet tightly coupled solution and this approach allows efficient data exchange but it introduces a strong dependency on specific data formats. Both frontend and the backend must agree on the structure and naming of data fields which means that any changes in the data model on one side require corresponding updates on the other. Without clear API versioning or abstraction layers such as defined interfaces or schema management evolving the system becomes

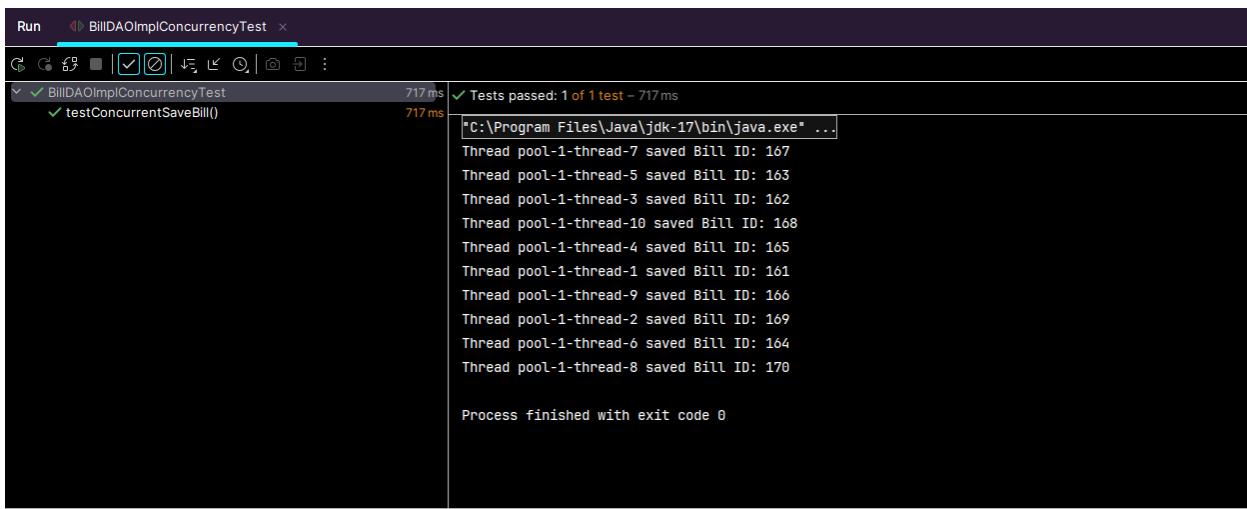
increasingly difficult and the absence of these things can lead to fragile communication, reduce flexibility and scalability as the application grows or requirements change.

The use of several design patterns like DAO, Factory and Strategy, the overall complexity of the system may increase which could pose a learning curve for new developers and affect maintainability if the documentation or coding standards are not properly followed. Lastly, while SOLID principles are mostly respected, the system's scalability and adaptability depend heavily on ongoing discipline in maintaining strict layering and avoiding the leakage of infrastructure concerns into business logic or UI code. In summary, the allocation of clean architecture components in SYOS demonstrates a strong foundational design with many benefits but it also requires a proper management of layer boundaries, dependencies and complexity to fully realize the theoretical advantages of clean architecture in practice.

Chapter 3 - Concurrency

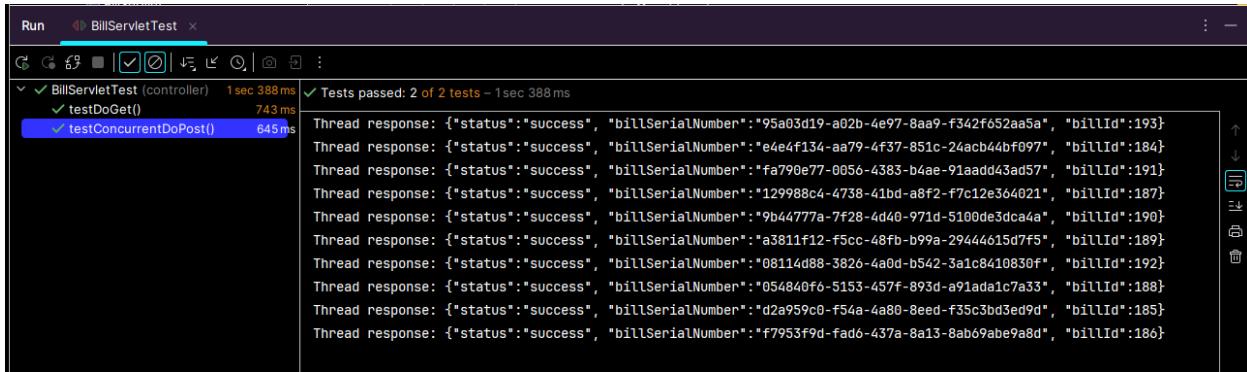
3.1 Server Side Concurrency

SYOS automated billing system has been implemented as a servlet based application that relies on the built in concurrency mechanisms of the Java EE servlet container such as Apache Tomcat to handle multiple simultaneous client requests efficiently through a thread per request model. When multiple clients send requests to save a bill, log in, register or insert a batch, the server uses a thread pool to process them concurrently by creating a separate thread invoking methods like doPost() or doGet() independently for each incoming request and this thread per request model ensures that one request does not block or interfere with another by maintaining high performance and responsiveness even under heavy load. Thread safety is maintained by using local variables and avoiding shared mutable state and additionally, each database operation creates a new PreparedStatement and obtains its own database connection by ensuring isolated transactions and also the underlying database management system handles concurrency at the data level through locking and transaction isolation by guaranteeing data integrity even when multiple inserts or updates happen concurrently. This combination of server side multi threading and database level concurrency control enables the application to safely and efficiently support multiple client interactions at the same time.



```
Run BillDAOImplConcurrencyTest ×
C:\Program Files\Java\jdk-17\bin\java.exe" ...
Tests passed: 1 of 1 test - 717ms
717ms
BillDAOImplConcurrencyTest
  ✓ testConcurrentSaveBill()
    ✓ Thread pool-1-thread-7 saved Bill ID: 167
    ✓ Thread pool-1-thread-5 saved Bill ID: 163
    ✓ Thread pool-1-thread-3 saved Bill ID: 162
    ✓ Thread pool-1-thread-10 saved Bill ID: 168
    ✓ Thread pool-1-thread-4 saved Bill ID: 165
    ✓ Thread pool-1-thread-1 saved Bill ID: 161
    ✓ Thread pool-1-thread-9 saved Bill ID: 166
    ✓ Thread pool-1-thread-2 saved Bill ID: 169
    ✓ Thread pool-1-thread-6 saved Bill ID: 164
    ✓ Thread pool-1-thread-8 saved Bill ID: 170

Process finished with exit code 0
```



```
BillServletTest (controller) 1 sec 388 ms Tests passed: 2 of 2 tests - 1 sec 388 ms
  ✓ testDoGet() 743 ms
  ✓ testConcurrentDoPost() 645 ms

Thread response: {"status":"success", "billSerialNumber":"95a03d19-a02b-4e97-8aa9-f342f652aa5a", "billId":193}
Thread response: {"status":"success", "billSerialNumber":"e4e4f134-aa79-4f37-851c-24acb44bf097", "billId":184}
Thread response: {"status":"success", "billSerialNumber":"fa790e77-0056-4383-b4ae-91aadd43ad57", "billId":191}
Thread response: {"status":"success", "billSerialNumber":"129988c4-4738-41bd-a8f2-f7c12e364021", "billId":187}
Thread response: {"status":"success", "billSerialNumber":"9b44777a-7f28-4d40-971d-5100de3dca4a", "billId":190}
Thread response: {"status":"success", "billSerialNumber":"a3811f12-f5cc-48fb-b99a-29444615d7f5", "billId":189}
Thread response: {"status":"success", "billSerialNumber":"08114d88-3826-4a0d-b542-3a1c8410830f", "billId":192}
Thread response: {"status":"success", "billSerialNumber":"054840f6-5153-457f-893d-a91ada1c7a33", "billId":188}
Thread response: {"status":"success", "billSerialNumber":"d2a959c0-f54a-4a80-8eed-f35c3bd3ed9d", "billId":185}
Thread response: {"status":"success", "billSerialNumber":"f7953f9d-fad6-437a-8a13-8ab69abe9a8d", "billId":186}
```

While the SYOS automated billing system depends on the Java EE servlet container's built in multithreading and database transaction mechanisms to handle concurrent client requests efficiently, this approach has several limitations. The use of thread per request and local variables promotes thread safety and prevents shared state issues but sometimes scalability can be challenged under very high loads due to thread pool exhaustion. And also by relying entirely on database locking and transaction isolation can lead to bottlenecks or deadlocks if not carefully managed especially with complex or long running transactions. The system's performance and reliability also depend heavily on the servlet container's configuration and the database's concurrency capabilities. Overall while the server side follows standard best practices for concurrency, continuous monitoring and optimization are essential to ensure consistent performance and data integrity under real world high concurrency scenarios.

3.2 Client Side Concurrency

The client side application included in the SYOS system uses concurrency mainly by using state management and asynchronous operations to keep the UI as smooth and responsive as possible. And also it uses reactive state updates that schedules changes very efficiently by executing multiple state changes without locking the main thread preventing the UI from freezing up during rapid user interactions. For example, when a user adds items to the cart it executes the `setItems([...items, newItem])` function and react schedules the state change without blocking the main thread so the user can smoothly interact with other modals and update input fields.

Asynchronous side effects are employed for the operations like data fetching or computations based on state changes run in the background so the UI can continue rendering and responding to inputs without waiting for each to complete synchronously by ensuring the application reacts immediately to user actions. As an example, operations such as saving a bill or order items use asynchronous fetch calls and this non blocking behavior ensures the UI to stay responsive which means the user can still scroll or interact with other components while the request is processed in the background.

```
try {
  // First, save the bill
  const billResponse = await fetch('http://localhost:8080/SYOS%20Backend/bill', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(billData)
  });

  if (!billResponse.ok) {
    throw new Error('Failed to save bill');
  }

  const billResult = await billResponse.json();
  const billSerialNumber = billResult.billSerialNumber;
  const billId = billResult.billId;

  // Then save each order item
  const orderPromises = billData.orders.map(order => {
    const orderData = new URLSearchParams();
    orderData.append('orderID', order.orderID);
    orderData.append('itemCode', order.itemCode);
    orderData.append('quantity', order.quantity);
    orderData.append('amount', order.amount);
    orderData.append('billSerialNumber', billSerialNumber);

    return fetch('http://localhost:8080/SYOS%20Backend/orders', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        ...order,
        billSerialNumber
      })
    });
  });

  await Promise.all(orderPromises);
}
```

For longer running operations such as submitting orders or fetching data, asynchronous processing avoids locking up the interface by running these tasks independently and updating the UI only once results are ready. As an example, when the user clicks the “Checkout” button, both the bill and all related order items are submitted and this parallel execution ensures fast completion without freezing the UI since order submission won’t wait for one another sequentially.

```
//  
//  
await Promise.all(orderPromises);
```

Routing is also handled concurrently by enabling smooth page transitions and view updates without interrupting ongoing user activities. By coordinating these mechanisms, the application achieves efficient concurrency, improving performance and user experience by allowing multiple processes to run and update the interface seamlessly.

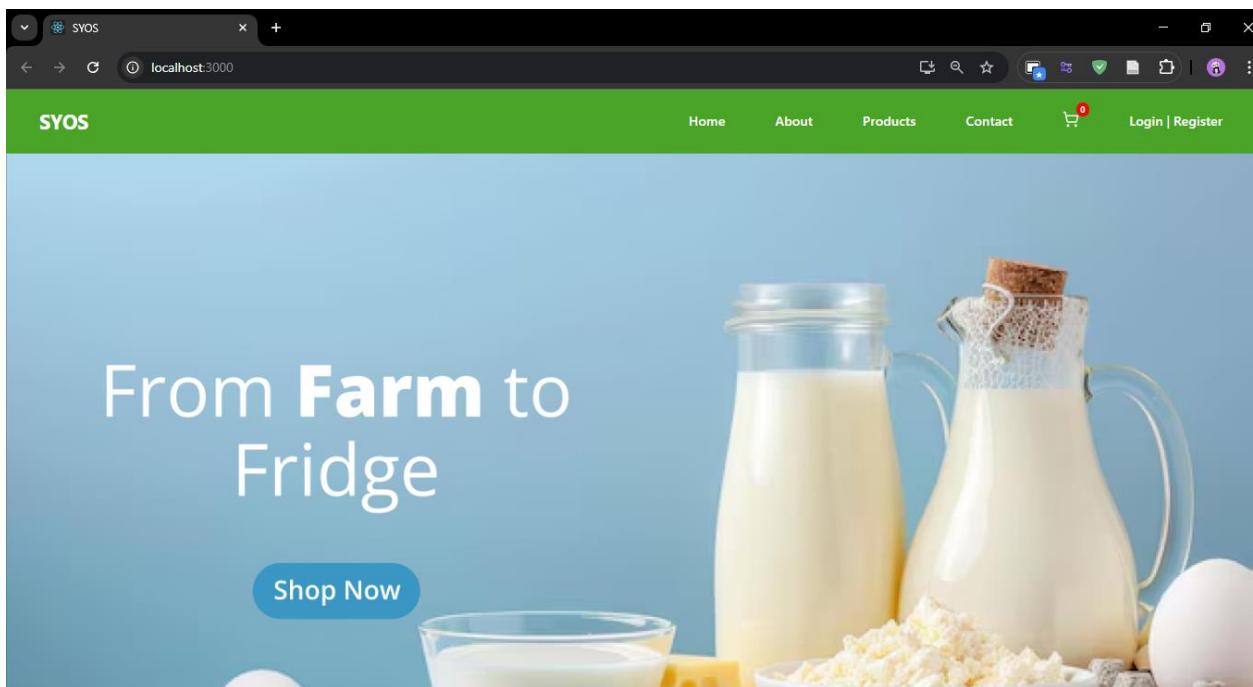
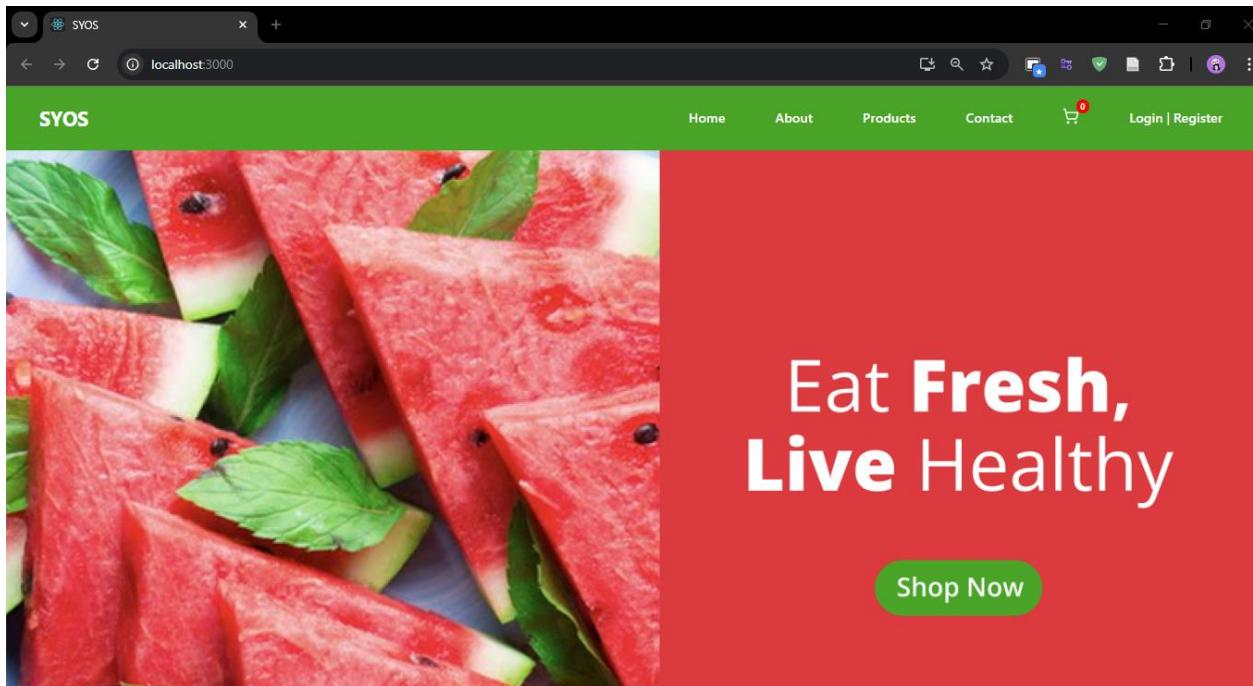
While the SYOS system effectively handles client side concurrency through React’s state management and asynchronous operations to ensure a smooth user experience, there are several architectural limitations that need to be paid attention. The system assumes that all asynchronous tasks such as adding items to a cart or submitting orders will execute reliably and in the expected order which may not hold true under real world conditions like poor network connectivity or high server load. This could lead to delayed UI updates, failed requests or data inconsistencies if not properly managed with error handling and fallback mechanisms. And also while the system supports multiple clients sending asynchronous requests simultaneously, it relies on client side state which can become stale in multi user scenarios without real time synchronization technologies like WebSockets. For example, one client may update a shared resource while another remains unaware which will lead to conflicts or incorrect data rendering. Therefore, while the SYOS shows client side responsiveness and concurrency handling, it would benefit from enhanced real time synchronization, better error recovery and robust backend concurrency control to maintain integrity and reliability in high traffic multi user environments.

3.3 Summary

By using the SYOS system, more clients can simultaneously send asynchronous requests in rapid succession and the system is designed to handle such concurrency efficiently. On the client side the react application uses asynchronous operations (async/await, fetch) which allow multiple requests to be happen without blocking the user interface by ensuring a smooth and responsive experience even during heavy interaction. For example, while one client is submitting an order, another can also perform checkout operations without any delay or interference. On the backend, servlets handle each incoming requests from different clients in parallel and the MySQL database will manage these operations by maintaining data integrity and consistency even when multiple clients interact with the system simultaneously. This coordination between client side asynchronous behavior and server side multi threaded processing ensures that the SYOS system can handle concurrent actions from different users without performance issues or data conflicts.

Chapter 4 - GUI

Image Carousel

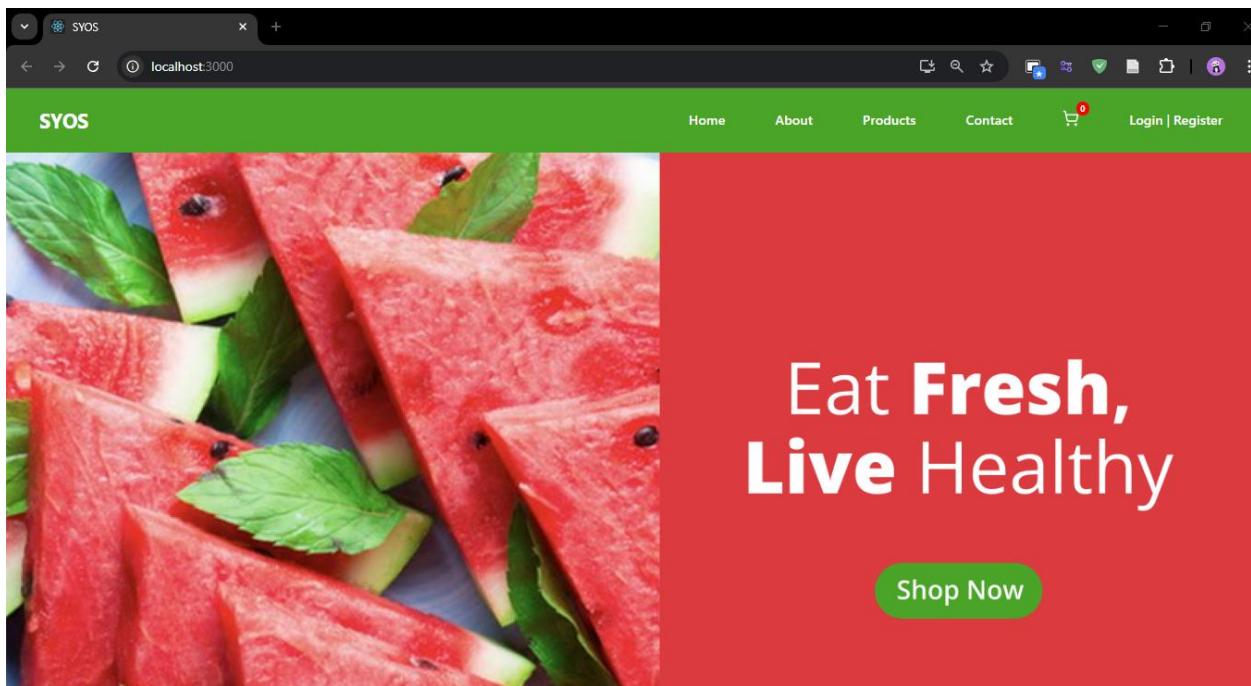




Navigation Bar



Home Page



Fruits

 Papaya 1.25kg Rs. 235 Add To Cart	 Mango 1kg Rs. 570 Add To Cart	 Watermelon 2kg Rs. 180 Add To Cart	 Apple 500g Rs. 710 Add To Cart	 Grapes 500g Rs. 1590 Add To Cart	 Orange 600g Rs. 990 Add To Cart
--	--	---	---	---	--

Vegetables

 Pumpkin 500g Rs. 70 Add To Cart	 Carrot 500g Rs. 225 Add To Cart	 Leeks 250g Rs. 70 Add To Cart	 Tomato 500g Rs. 70 Add To Cart	 Capicum 250g Rs. 145 Add To Cart	 Cabbage 500g Rs. 165 Add To Cart
--	--	--	---	---	---

Dairy

 Kotmale Curd 900g Rs. 544 Add To Cart	 Puredale Kirthe 400g Rs. 830 Add To Cart	 Kotmale Ball Cheese 325g Rs. 1485 Add To Cart	 Ambewela Fresh Milk 1L Rs. 550 Add To Cart	 Cheese Spread 175g Rs. 180 Add To Cart	 Happy Cow Cheese 120g Rs. 448 Add To Cart
--	---	--	---	---	--

SYOS

[Home](#) [Abouts Us](#) [Products](#) [Contact Us](#)

Stay in Touch

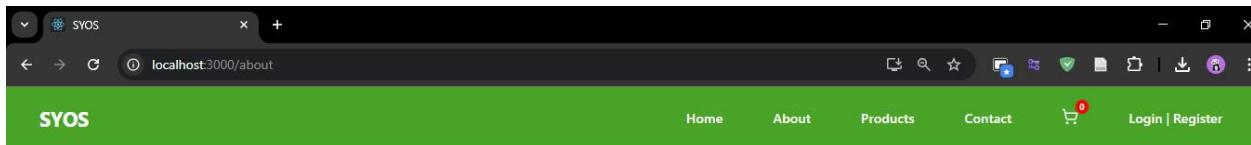
Footer

SYOS

[Home](#) [Abouts Us](#) [Products](#) [Contact Us](#)

Stay in Touch

About Page



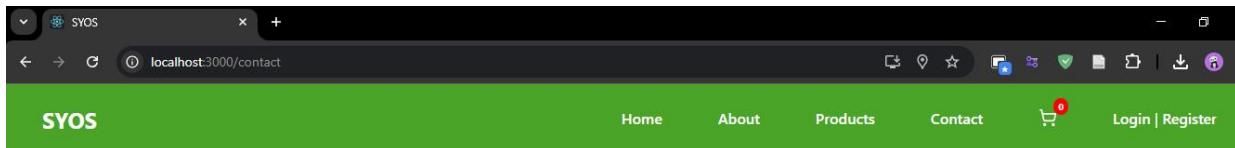
The screenshot shows a web browser window with the URL `localhost:3000/about`. The page has a green header bar with the logo "SYOS". Below the header is a navigation bar with links for "Home", "About", "Products", "Contact", and a shopping cart icon with a "0" notification. The main content area features four images related to fresh produce and dairy products.



About Us

Welcome to our website! We are dedicated to delivering high-quality services and fresh, wholesome products that enhance your everyday life. Our mission is to create meaningful digital experiences while connecting you with the finest selection of fruits, vegetables, and dairy products. At the heart of our platform is a commitment to freshness and quality. We carefully source a wide variety of seasonal fruits, nutrient-rich vegetables, and farm-fresh dairy to ensure that your family gets the best. Whether you're planning a healthy meal, stocking your pantry, or making a quick grocery run, we make the process simple, convenient, and reliable. We believe in innovation, sustainability, and putting our customers first. With a passionate team and years of expertise, we aim to exceed expectations and provide exceptional value. Your health and satisfaction are our top priorities, and we're proud to support your journey to a better lifestyle—one fresh product at a time.

Contact Page



The screenshot shows a web browser window with the URL `localhost:3000/contact`. The page has a green header bar with the logo "SYOS". Below the header is a navigation bar with links for "Home", "About", "Products", "Contact", and a shopping cart icon with a "0" notification. The main content area includes sections for "Contact Information" and "Send Us a Message" with form fields for Name, Email, and Message, along with a map and a "Send Message" button.

Contact Information

Phone: +1 234 567 890

Email: contact@freshcart.com

Address: 123 Fresh Street, Green City, Country



Send Us a Message

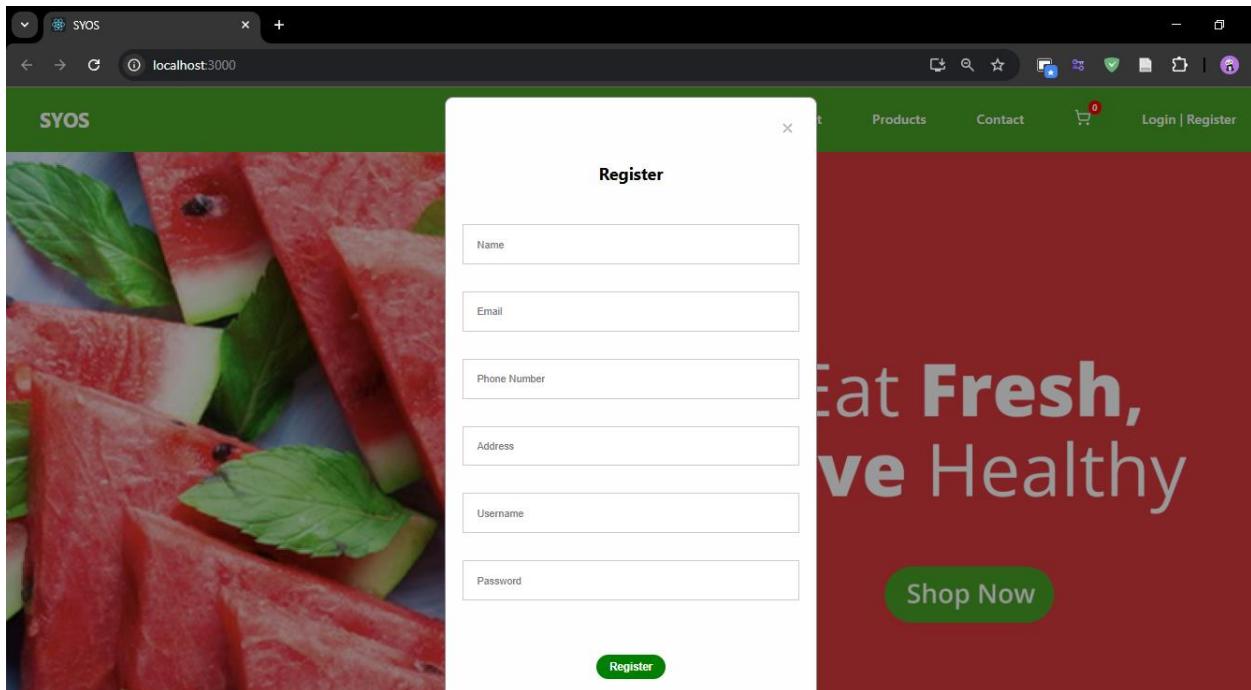
Name:

Email:

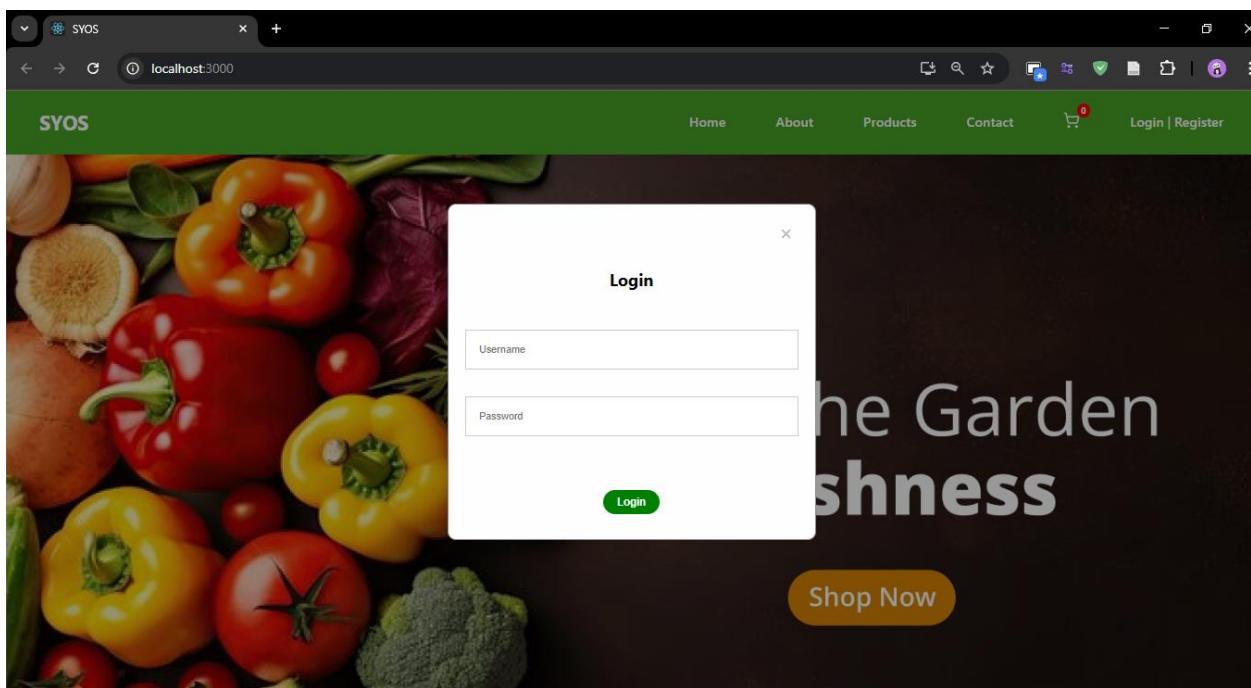
Message:

Send Message

Register Modal



Login Modal



My Cart – Sidebar

The screenshot shows a web browser window with a green header bar labeled "SYOS". On the left, there's a large promotional image with the text "From Farm to Fridge" and a "Shop Now" button. On the right, a sidebar titled "My Cart" lists three items: Apple 500g (Rs. 710), Tomato 500g (Rs. 70), and Carrot 500g (Rs. 225). Each item has a quantity selector (minus, 1, plus) and a "Check Out" button at the bottom.

Product	Quantity	Price
Apple 500g	1	Rs. 710.00
Tomato 500g	1	Rs. 70.00
Carrot 500g	1	Rs. 225.00

Checkout Page

The screenshot shows a web browser window with a green header bar labeled "SYOS". The header includes links for Home, About, Products, Contact, a shopping cart icon with a red notification (3), and Login | Register. Below the header, the page is divided into two main sections: "Bill Summary" and "Delivery Details".

Bill Summary

Product	Quantity	Price
Apple 500g	1	Rs. 710.00
Tomato 500g	1	Rs. 70.00
Carrot 500g	1	Rs. 225.00

Total Amount: Rs. 1005.00

Delivery Details

Name:

Address:

City:

Phone:

Email:

Place Order

Place Order Page

The screenshot shows the 'Place Order' page of a web application. On the left, a sidebar menu lists various reports and actions. The main area contains input fields for placing an order, including Item Code, Item Name, Unit Price, Quantity, and Amount. It also includes fields for Discount, Amount After Discount, Cash Tendered, and Cash Change. A red 'Add' button is located below the input fields, and a green 'Place Order' button is on the right.

Item Code	Item Name	Unit Price	Quantity	Action	Amount
<input type="text"/>					

Item Name:
Unit Price:
Quantity:
Amount:

Discount: 0.00
Amount After Discount: 0.00
Cash Tendered:
Cash Change:

Add **Place Order**

Bill Summary

The screenshot shows a 'Bill Summary' modal window. It displays the Bill Serial Number (195), Transaction Type (ONSITE), and a table of Ordered Items. The table shows one item: Watermelon 2kg at 180 per unit, quantity 3, and total amount 540.00. Below the table, it shows the Total Amount (Rs. 540.00), Discount (Rs. 0.00), Amount After Discount (Rs. 540.00), Cash Tendered (Rs. 600.00), and Cash Change (Rs. 60.00). A 'Close' button is at the bottom of the modal.

Item Code	Item Name	Unit Price	Quantity	Amount
fru003	Watermelon 2kg	180	3	540.00

Bill Summary

Bill Serial Number: 195
Transaction Type: ONSITE

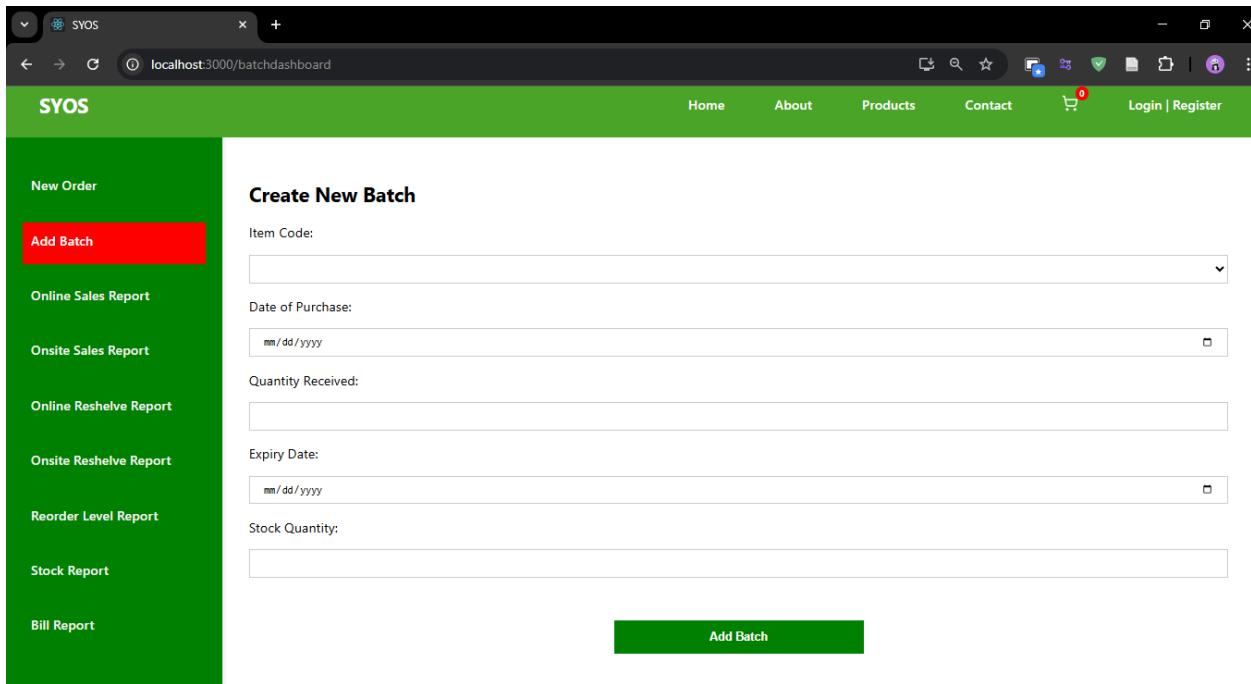
Ordered Items

Item Code	Item Name	Unit Price	Quantity	Amount
fru003	Watermelon 2kg	180	3	540.00

Total Amount: Rs. 540.00
Discount: Rs. 0.00
Amount After Discount: Rs. 540.00
Cash Tendered: Rs. 600.00
Cash Change: Rs. 60.00

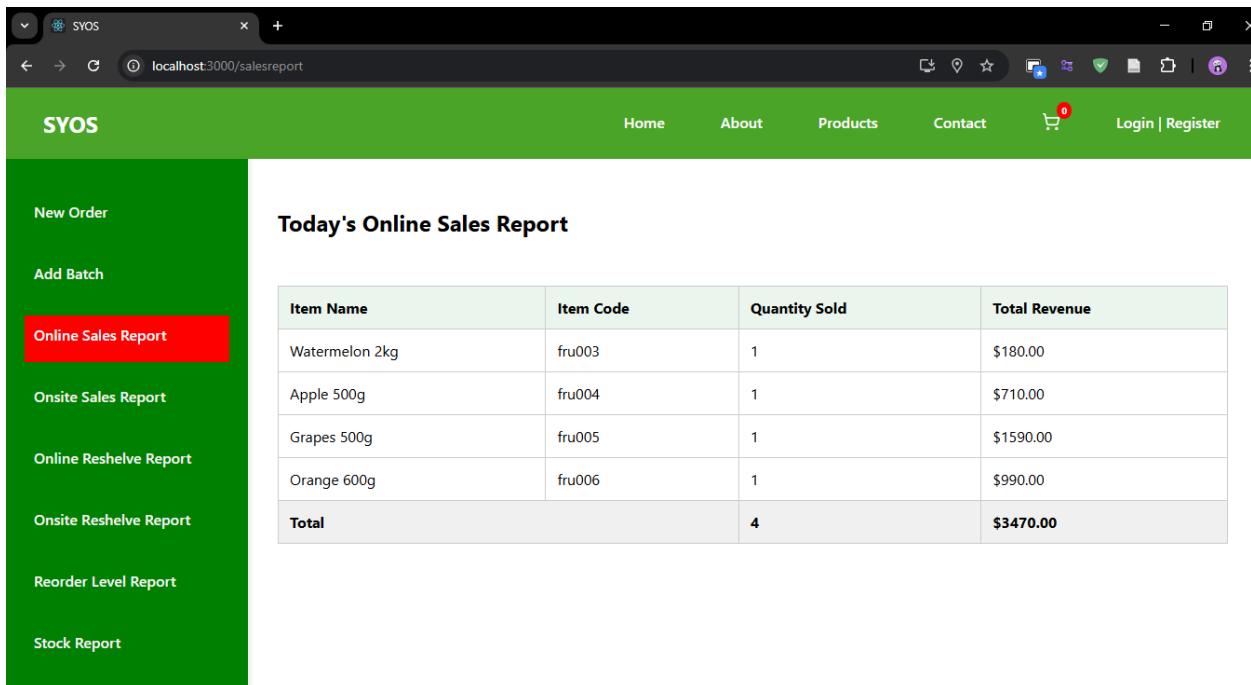
Close

Create New Batch Page



The screenshot shows a web browser window for the SYOS application. The URL is `localhost:3000/batchdashboard`. The page has a green header with the SYOS logo and navigation links for Home, About, Products, Contact, and a shopping cart icon with 0 items. A red button labeled "Add Batch" is highlighted. The main content area is titled "Create New Batch" and contains fields for Item Code (a dropdown menu), Date of Purchase (a date input field), Quantity Received (a text input field), Expiry Date (a date input field), and Stock Quantity (a text input field). A green "Add Batch" button is at the bottom.

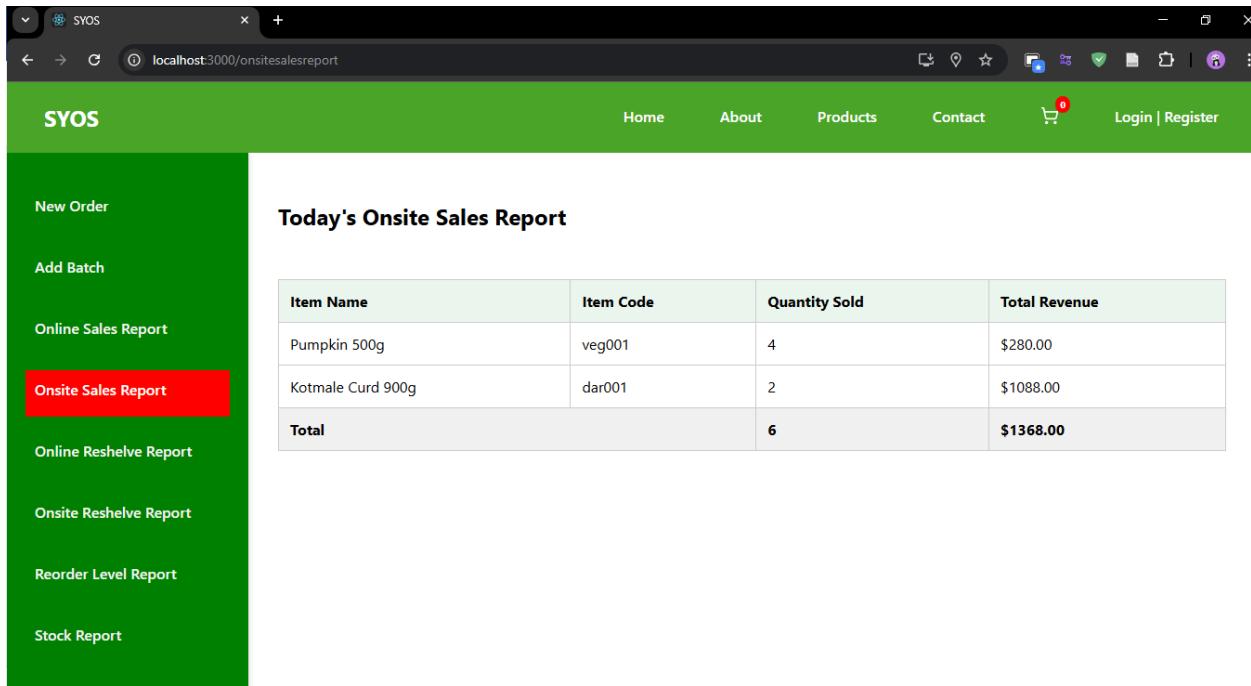
Online Sales Report



The screenshot shows a web browser window for the SYOS application. The URL is `localhost:3000/salesreport`. The page has a green header with the SYOS logo and navigation links for Home, About, Products, Contact, and a shopping cart icon with 0 items. A red button labeled "Online Sales Report" is highlighted. The main content area is titled "Today's Online Sales Report" and displays a table of sales data.

Item Name	Item Code	Quantity Sold	Total Revenue
Watermelon 2kg	fru003	1	\$180.00
Apple 500g	fru004	1	\$710.00
Grapes 500g	fru005	1	\$1590.00
Orange 600g	fru006	1	\$990.00
Total		4	\$3470.00

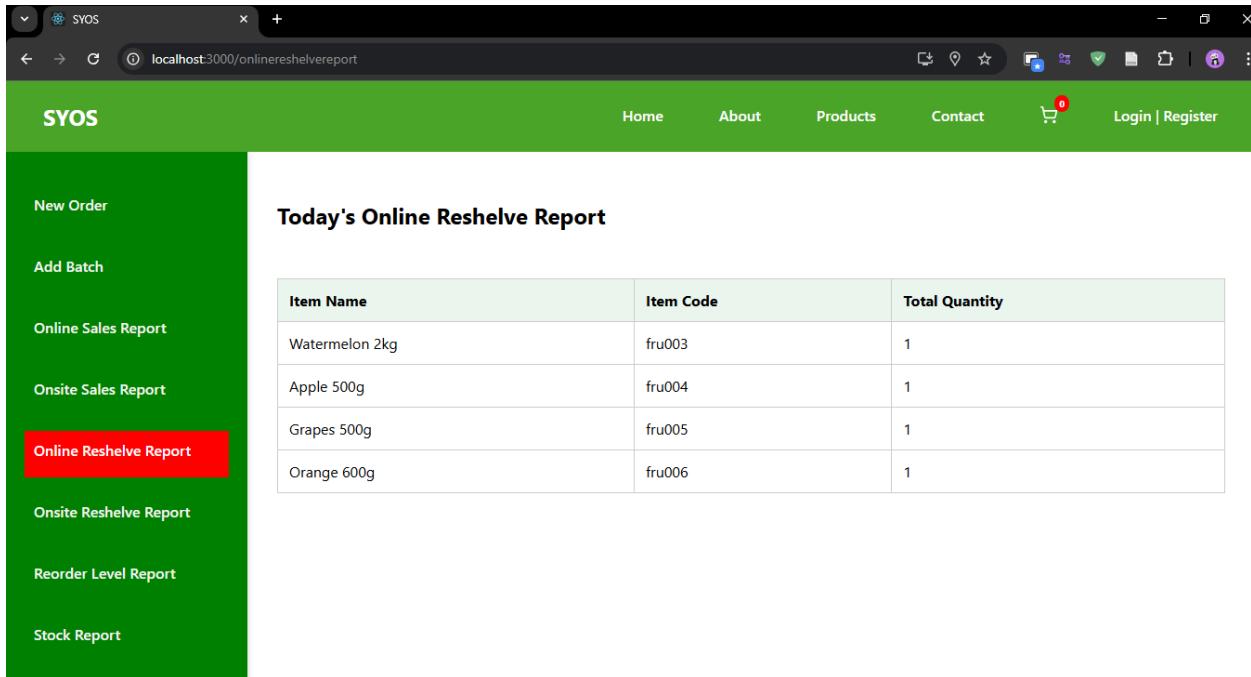
Onsite Sales Report



The screenshot shows a web application interface for 'SYOS'. The left sidebar has a dark green background with white text links: 'New Order', 'Add Batch', 'Online Sales Report', 'Onsite Sales Report' (which is highlighted in red), 'Online Reshelve Report', 'Onsite Reshelve Report', 'Reorder Level Report', and 'Stock Report'. The main content area has a light gray background with a title 'Today's Onsite Sales Report'. Below it is a table with four columns: Item Name, Item Code, Quantity Sold, and Total Revenue. The table contains three rows of data and a summary row at the bottom.

Item Name	Item Code	Quantity Sold	Total Revenue
Pumpkin 500g	veg001	4	\$280.00
Kotmale Curd 900g	dar001	2	\$1088.00
Total		6	\$1368.00

Online Reshelve Report



The screenshot shows a web application interface for 'SYOS'. The left sidebar has a dark green background with white text links: 'New Order', 'Add Batch', 'Online Sales Report', 'Onsite Sales Report', 'Online Reshelve Report' (which is highlighted in red), 'Onsite Reshelve Report', 'Reorder Level Report', and 'Stock Report'. The main content area has a light gray background with a title 'Today's Online Reshelve Report'. Below it is a table with three columns: Item Name, Item Code, and Total Quantity. The table contains five rows of data.

Item Name	Item Code	Total Quantity
Watermelon 2kg	fru003	1
Apple 500g	fru004	1
Grapes 500g	fru005	1
Orange 600g	fru006	1

Onsite Reshelve Report

The screenshot shows a web browser window for the 'SYOS' application. The URL is 'localhost:3000/onsitereshelvereport'. The page has a green header with the 'SYOS' logo and navigation links for Home, About, Products, Contact, a shopping cart icon with a red '0', and Login | Register. A sidebar on the left contains links for New Order, Add Batch, Online Sales Report, Onsite Sales Report, Online Reshelve Report, Onsite Reshelve Report (which is highlighted in a red box), Reorder Level Report, and Stock Report. The main content area displays a table titled 'Today's Onsite Reshelve Report' with three rows:

Item Name	Item Code	Total Quantity
Pumpkin 500g	veg001	4
Kotmale Curd 900g	dar001	2

Reorder Level Report

The screenshot shows a web browser window for the 'SYOS' application. The URL is 'localhost:3000/reorderlevelreport'. The page has a green header with the 'SYOS' logo and navigation links for Home, About, Products, Contact, a shopping cart icon with a red '0', and Login | Register. A sidebar on the left contains links for New Order, Add Batch, Online Sales Report, Onsite Sales Report, Online Reshelve Report, Onsite Reshelve Report, Reorder Level Report (which is highlighted in a red box), and Stock Report. The main content area displays a table titled 'Reorder Level Report' with six rows:

Item Name	Item Code	Total Stock Quantity
Papaya 1.25kg	fru001	44
Pumpkin 500g	veg001	32
Watermelon 2kg	fru003	29
Orange 600g	fru006	39
Apple 500g	fru004	19

Stock Report

The screenshot shows a web browser window for the 'SYOS' application at localhost:3000/stockreport. The left sidebar has a red 'Stock Report' button. The main content area displays a table of stock items:

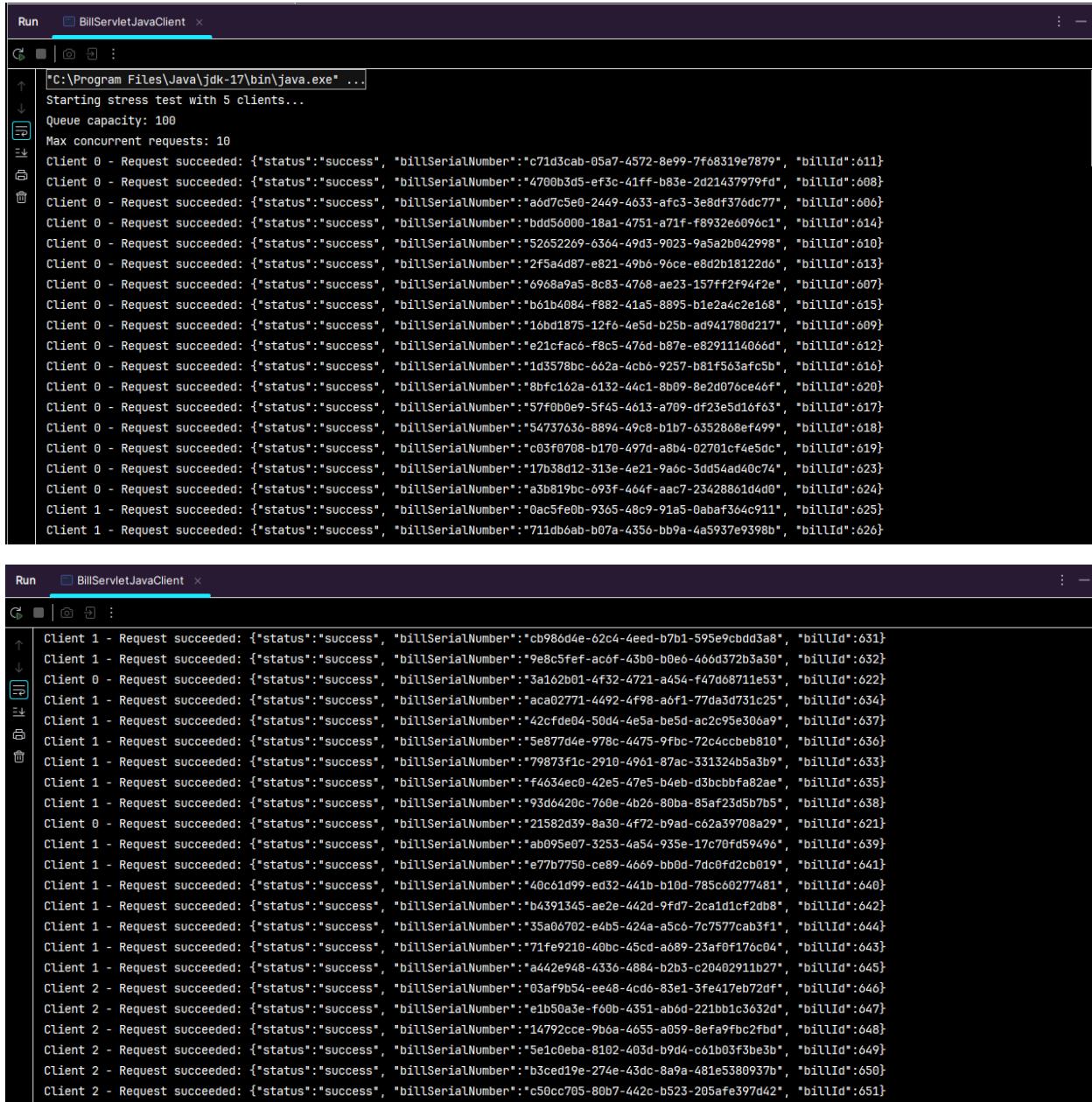
Item Code	Date of Purchase	Quantity Received	Expiry Date
fru001	2024-10-12	44	2025-01-01
fru004	2023-10-02	20	2025-10-07
fru003	2025-05-04	30	2025-07-12
fru002	2025-04-07	50	2025-06-26
veg004	2025-04-30	66	2025-06-03
veg003	2025-05-01	50	2025-06-01
veg002	2025-05-04	60	2025-05-30
fru005	2025-05-01	58	2025-06-20

Bill Report

The screenshot shows a web browser window for the 'SYOS' application at localhost:3000/billreport. The left sidebar has a red 'Stock Report' button. The main content area displays a table of bill transactions:

Bill Serial No	Bill Date	Transaction Type	Total Amount	Discount	Cash Tendered	Cash Change	Customer ID
757	2025-05-24	ONLINE	3470	0	3470	0	3
756	2025-05-24	ONSITE	1368	0	2000	632	8

Chapter 5 - Automatic Test Clients



```
[C:\Program Files\Java\jdk-17\bin\java.exe* ...]
Starting stress test with 5 clients...
Queue capacity: 100
Max concurrent requests: 10
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"c71d3cab-05a7-4572-8e99-7f68319e7879", "billId":611}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"4700b3d5-ef3c-41ff-b83e-2d21437979fd", "billId":608}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"a0d7c5e0-2449-4633-afc3-3e8df376dc77", "billId":606}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"bdd56000-18a1-4751-a71f-f8932e6096c1", "billId":614}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"52652269-6364-49d3-9023-9a5a2b042998", "billId":610}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"2f5a4d87-e821-49b0-96ce-e8d2b18122d6", "billId":613}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"6968a9a5-8c83-4768-ae23-157ff2f94f2e", "billId":607}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"b61b4084-f882-41a5-8895-b1e2a4c2e168", "billId":615}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"16bd1875-12f6-4e5d-b25b-ad941780d217", "billId":609}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"e21cfac6-f8c5-476d-b87e-e8291114066d", "billId":612}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"1d3578bc-662a-4cbo-9257-b81f563afc5b", "billId":616}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"8bfc162a-6132-44c1-b809-8e2d07ce46f", "billId":620}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"57f0b0e9-5f45-4613-a709-df23e5d16f63", "billId":617}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"54737636-8894-49c8-b1b7-6352868ef499", "billId":618}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"c03f0708-b170-497d-a8b4-02701cf4e5dc", "billId":619}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"17b38d12-313e-4e21-9a6c-3dd54ad40c74", "billId":623}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"a3b819bc-693f-464f-aac7-23428861dd0", "billId":624}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"0ac5fe0b-9365-48c9-91a5-0abaf304c911", "billId":625}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"711db6ab-b07a-4356-bb9a-4a5937e9398b", "billId":626}

Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"cb986d4e-62c4-4eed-b7b1-595e9cbdd3a8", "billId":631}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"98e8c5fef-ac0f-43b0-b0e6-46d372b3a30", "billId":632}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"3a162b01-4f32-4721-a454-f47d68711e53", "billId":622}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"aca02771-4492-4f98-a6f1-77da3d731c25", "billId":634}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"42cfde04-50d4-4e5a-be5d-ac2c95e306a9", "billId":637}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"5e877d4e-978c-4475-9fb-72c4ccbeb810", "billId":636}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"79873f1c-2910-4961-87ac-331324b5a3b9", "billId":633}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"f4634ec0-42e5-47e5-b4eb-d3bcbfffa82ae", "billId":635}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"93d6420c-760e-4b26-80ba-85af23d5b7b5", "billId":638}
Client 0 - Request succeeded: {"status":"success", "billSerialNumber":"21582d39-8a30-4f72-b9ad-c62a39708a29", "billId":621}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"ab095e07-3253-4e54-935e-17c70fd59490", "billId":639}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"e77b7750-ce89-4669-bb0d-7dc0fd2cb019", "billId":641}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"40c61d99-ed32-441b-b10d-785c60277481", "billId":640}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"b4391345-ae2e-442d-9fd7-2ca11cf2dbb", "billId":642}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"35a06702-e4b5-424a-a5c8-7c7577cab3f1", "billId":644}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"71fe9210-40bc-45cd-a689-23af0f176c04", "billId":643}
Client 1 - Request succeeded: {"status":"success", "billSerialNumber":"a442e948-4336-4884-b2b3-c2040291b27", "billId":645}
Client 2 - Request succeeded: {"status":"success", "billSerialNumber":"03af9b54-ee48-4cd6-83e1-3fe417eb72df", "billId":646}
Client 2 - Request succeeded: {"status":"success", "billSerialNumber":"1lb50a3e-f60b-4351-ab0d-221bb1c3632d", "billId":647}
Client 2 - Request succeeded: {"status":"success", "billSerialNumber":14792cce-9b6a-4655-a059-8efa9fb2fb0", "billId":648}
Client 2 - Request succeeded: {"status":"success", "billSerialNumber":5e1c0eba-8102-403d-b9d4-c61b03f3be3b, "billId":649}
Client 2 - Request succeeded: {"status":"success", "billSerialNumber":b3ced19e-274e-43dc-8a9a-481e5380937b, "billId":650}
Client 2 - Request succeeded: {"status":"success", "billSerialNumber":c50cc705-80b7-442c-b523-205afe397d42, "billId":651}
```

```
Run BillServletJavaClient x
G | : 

↑ Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "c8291ale-59c4-4533-8422-ff75f1c06913", "billId": 692}
↓ Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "ca3a4993-211b-4432-ace3-2df10fbc7714", "billId": 693}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "ec4b384e-6e68-4772-a0d8-cf0e0efc7bca", "billId": 694}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "cb3fc0b04-6fde-4ca6-9870-1dc955826bcc", "billId": 695}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "b878d359-c728-4600-93e0-87a10875adb0", "billId": 696}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "b2360d69-5019-419c-b862-ea503e5a862a", "billId": 697}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "eb33f872-6bae-4643-9406-4eb87c0ccb1ae", "billId": 698}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "4b8def52-7f35-4919-af40-e5af3af299d5", "billId": 699}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "25a43650-0961-4d5b-83a8-1a402d02cb2a", "billId": 700}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "f04531f5-291a-4638-801f-05c5e7bdf559", "billId": 701}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "1cec400e-04ed-42e6-80dd-0a98dd047417", "billId": 702}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "835684e2-7748-4e76-863f-bcd5baed10b3", "billId": 703}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "5d7ef192-c0ec-4437-b80f-7332e44ff2af", "billId": 704}
Client 4 - Request succeeded: {"status": "success", "billSerialNumber": "0c69522d-524b-4e98-918e-2f9a374c7501", "billId": 705}

Test completed in 1044 ms
Successful requests: 100
Failed requests: 0
Requests per second: 95.79
Final queue size: 0

Process finished with exit code 0
```

Chapter 6 - Test Cases

6.1 Junit Test Cases

The screenshot shows a Java IDE interface with the following details:

- Project View:** Shows a package named "Test" containing various test classes like BatchTest, BillDAOImplConcurrencyTest, BillReportTest, BillTest, DBConnectionTest, ItemDAOImplTest, ItemTest, LoginDAOImplTest, OnlineInventoryTest, OnlineReshelveReportTest, OnsiteReshelveReportTest, OnsiteSalesReportTest, OrderTest, and OrderValidatorTest.
- Code Editor:** The DBConnectionTest.java file is open, showing code related to JDBC connections and assertions.
- Run Tab:** The "Run" tab is selected, showing the output of the DBConnectionTest run. It indicates 2 tests passed in 508 ms, with individual test times for testSingletonInstance() (24 ms) and testGetConnection() (484 ms).
- Console:** The console shows the command "C:\Program Files\Java\jdk-17\bin\java.exe" ... and the message "Process finished with exit code 0".
- Bottom Status:** Shows the file path "SYOS Backend > Test > DBConnectionTest", and the status bar with "15:16 CRLF UTF-8 4 spaces".

The screenshot shows a Java IDE interface with the following details:

- Project View:** Shows a package named "Test" containing various test classes like BatchTest, BillReportServletTest, BillServletTest, BillDAO, BillDAOImplConcurrencyTest, BillTest, and OrderTest.
- Code Editor:** The BatchTest.java file is open, showing test cases for the Batch class.
- Run Tab:** The "Run" tab is selected, showing the output of the BatchTest run. It indicates 6 tests passed in 27 ms, with individual test times for testDateOfPurchaseGetterSetter (23 ms), testExpiryDateGetterSetter (1 ms), testQuantityReceivedGetterSetter (1 ms), testItemCodeGetterSetter (1 ms), testStockQuantityGetterSetter (1 ms), and testConstructor (1 ms).
- Console:** The console shows the command "C:\Program Files\Java\jdk-17\bin\java.exe" ... and the message "Process finished with exit code 0".
- Bottom Status:** Shows the file path "SYOS Backend > Test > BatchTest > testExpiryDateGetterSetter", and the status bar with "45:40 CRLF UTF-8 4 spaces".

SYS Backend Version control

Project

- RegisterDAOImpl 0% methods, 0% lines
- ReorderLevelReport 0% methods, 0%
- SalesReport 0% methods, 0% lines covered
- StockReport 0% methods, 0% lines covered
- db 100% classes, 100% lines covered
 - DBConnection 100% methods, 100% lines covered
- model 14% classes, 15% lines covered
 - Batch 0% methods, 0% lines covered
 - Batch 87% methods, 61% lines covered
 - Bill 87% methods, 61% lines covered
 - Item 0% methods, 0% lines covered
 - OnlineInventory 0% methods, 0% lines covered
 - Order 0% methods, 0% lines covered

BatchTest.java

```

class BillTest {
    @Test
    void testConstructor() {
        String serial = "BILL-001";
        Date billDate = new Date();
        String transactionType = "Cash";
        float totalAmount = 1000f;
        float discount = 50f;
    }
}

```

Run BillTest ×

BillTest 38 ms Tests passed: 12 of 12 tests - 38 ms

- ✓ testTotalAmount() 23 ms
- ✓ testAmountAfterDiscount() 2 ms
- ✓ testBillSerialNumber() 1 ms
- ✓ testCashChange() 1 ms
- ✓ testDiscount() 2 ms
- ✓ testTransactionType() 1 ms
- ✓ testBillID() 2 ms
- ✓ testCashTendered() 1 ms
- ✓ testBillDate() 1 ms
- ✓ testOrders() 2 ms
- ✓ testCustomerID() 1 ms
- ✓ testConstructor() 1 ms

C:\Program Files\Java\jdk-17\bin\java.exe ...

Process finished with exit code 0

119:1 CRLF UTF-8 4 spaces

SYS Backend Version control

Project

- User 0% methods, 0% lines covered
- service 66% classes, 4% lines covered
 - ItemValidator 0% methods, 0% lines covered
 - OrderProcessingService 100% methods
 - StockMovementService
 - StockMovementServiceImpl 0% methods
- view
- java
- Main 0% methods, 0% lines covered
- Test
 - BatchTest
 - BillDAOImplConcurrencyTest
 - BillTest
 - ItemTest
- WEB-INF
- .gitignore

Item.java

```

import static org.junit.jupiter.api.Assertions.*;

class ItemTest {
    @Test
    void testConstructorAndGetters() {
        Item item = new Item(itemCode: "I001", itemName: "Apple", pricePerUnit: 2.5f, path: "/images/apple.jpg");
        assertEquals(expected: "I001", item.getItemCode());
        assertEquals(expected: "Apple", item.getItemName());
        assertEquals(expected: 2.5f, item.getPricePerUnit());
        assertEquals(expected: "/images/apple.jpg", item.getPath());
        assertEquals(expected: "Fruit", item.getCategory());
    }
}

```

Run ItemTest ×

ItemTest 24 ms Tests passed: 6 of 6 tests - 24 ms

- ✓ testSetItemCode() 19 ms
- ✓ testSetItemName() 1 ms
- ✓ testConstructorAndGetters() 1 ms
- ✓ testSetPricePerUnit() 1 ms
- ✓ testSetCategory() 1 ms
- ✓ testSetPath() 1 ms

C:\Program Files\Java\jdk-17\bin\java.exe ...

Process finished with exit code 0

28:37 CRLF UTF-8 4 spaces

SYS Backend

```

st.java      Item.java    ItemTest.java  OnlineInventory.java  OnlineInventoryTest.java  BillDAOImplConc...
1 import model.OnlineInventory;
2 import org.junit.jupiter.api.Test;
3
4 import java.util.Date;
5
6 import static org.junit.jupiter.api.Assertions.*;
7
8 class OnlineInventoryTest {
9
10    @Test
11    void testConstructorAndGetters() {
12        String id = "01123";
13        Date date = new Date();
14        int qtySold = 10;
    }

```

Run OnlineInventoryTest

Tests passed: 7 of 7 tests – 33ms

- testSetItemCode() 28ms
- testConstructorAndGetters() 1ms
- testSetQuantitySold() 1ms
- testSetOnlineInventoryID() 1ms
- testSetOrderID() 1ms
- testSetDate() 1ms
- testSetBatchID() 1ms

Process finished with exit code 0

SYS Backend

```

Project Alt+1
a      OnlineInventory.java  OnlineInventoryTest.java  Order.java  OrderTest.java  BillDAOImplConc...
1 import model.Order;
2 import org.junit.jupiter.api.Test;
3
4 import static org.junit.jupiter.api.Assertions.*;
5
6 class OrderTest {
7
8    @Test
9    void testConstructorAndGetters() {
10        String orderId = "0100";
11        String itemCode = "I200";
12        int quantity = 3;
13        float totalPrice = 45.5f;
14        String billSerialNumber = "B123";
    }

```

Run OrderTest

Tests passed: 2 of 2 tests – 24ms

- testConstructorAndGetters() 23ms
- testSetters() 1ms

Process finished with exit code 0

```
import model.ShelfInventory;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class ShelfInventoryTest {

    @Test
    void testConstructorAndGetters() {
        String shelfInventoryID = "S001";
        int shelfQuantity = 25;
        String itemCode = "I100";

        ShelfInventory shelfInventory = new ShelfInventory(shelfInventoryID, shelfQuantity, itemCode);
    }
}
```

Run ShelfInventoryTest

ShelfInventoryTest 23 ms Tests passed: 2 of 2 tests – 23 ms

testConstructorAndGetters() 22 ms

testSetters() 1ms

C:\Program Files\Java\jdk-17\bin\java.exe ...

Process finished with exit code 0

```
import model.User;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class UserTest {

    @Test
    void testConstructorAndGetters() {
        int userID = 1;
        String name = "John Doe";
        String email = "john@example.com";
        String phone = "1234567890";
        String address = "123 Main St";
        String username = "johndoe";
    }
}
```

Run UserTest

UserTest 24 ms Tests passed: 2 of 2 tests – 24 ms

testConstructorAndGetters() 23 ms

testSetters() 1ms

C:\Program Files\Java\jdk-17\bin\java.exe ...

Process finished with exit code 0

SYS Backend > Test > ItemDAOImplTest

```

import static org.junit.jupiter.api.Assertions.*;

class ItemDAOImplTest {

    @BeforeEach
    void setup() {
        itemDAO = new ItemDAOImpl();
    }

    @Test
    void testGetItemPrice_validItemCode() {
    }
}

```

Run ItemDAOImplTest

Tests passed: 5 of 5 tests - 604 ms

- ✓ ItemDAOImplTest 604 ms
- ✓ testItemCodeValid_invalidCode 504 ms
- ✓ testItemCode_validCode 24 ms
- ✓ testGetItemPrice_invalidItemCode 25 ms
- ✓ testGetAllItems() 27 ms
- ✓ testGetItemPrice_validItemCode 24 ms

C:\Program Files\Java\jdk-17\bin\java.exe* ...

Process finished with exit code 0

SYS Backend > Test > LoginDAOImplTest

```

import org.junit.jupiter.api.*;
import static org.junit.jupiter.api.Assertions.*;

class LoginDAOImplTest {

    private LoginDAO loginDAO;

    @BeforeEach
    void setUp() {
        loginDAO = new LoginDAOImpl(DBConnection.getInstance());
    }

    @Test
    void testLoginCustomer_validCredit() {
    }

    @Test
    void testLoginCustomer_invalidCredit() {
    }
}

```

Run LoginDAOImplTest

Tests passed: 2 of 2 tests - 566 ms

- ✓ LoginDAOImplTest 566 ms
- ✓ testLoginCustomer_validCredit 537 ms
- ✓ testLoginCustomer_invalidCredit 29 ms

C:\Program Files\Java\jdk-17\bin\java.exe* ...

Process finished with exit code 0

SB SYOS Backend Version control

Project: StockMovementService

t.java

```

import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;

class ItemValidatorTest {

    private ItemDAO itemDAO;
    private ItemValidator itemValidator;

    @BeforeEach
    void setUp() {
        itemDAO = mock(ItemDAO.class);
        itemValidator = new ItemValidator(itemDAO);
    }

    @Test
    void testIsValidItemCode_ReturnsFalse() {
        // Given
        String invalidCode = "1234567890";
        // When
        boolean result = itemValidator.isValidItemCode(invalidCode);
        // Then
        assertFalse(result);
    }

    @Test
    void testIsValidItemCode_ReturnsTrue() {
        // Given
        String validCode = "1234567890";
        // When
        boolean result = itemValidator.isValidItemCode(validCode);
        // Then
        assertTrue(result);
    }
}

```

Run: ItemValidatorTest

Output:

```

ItemValidatorTest 633 ms
  ✓ testsIsValidItemCode_ReturnsFalse 630 ms
  ✓ testsIsValidItemCode_ReturnsTrue 3 ms
[C:\Program Files\Java\jdk-17\bin\java.exe* ...]
Process finished with exit code 0

```

25:1 CRLF UTF-8 4 spaces

SB SYOS Backend Version control

Project: OrderProcessingService

OrderProcessingServiceTest.java

```

import service.StockMovementService;
import java.util.Arrays;
import java.util.List;

import static org.mockito.Mockito.*;

class OrderProcessingServiceTest {

    private StockMovementService stockMovementService;
    private OrderProcessingService orderProcessingService;

    @BeforeEach
    void setUp() {
        stockMovementService = mock(StockMovementService.class);
        orderProcessingService = new OrderProcessingService(stockMovementService);
    }

    @Test
    void testProcessOrders_CallsMove() {
        // Given
        String[] itemCodes = {"1234567890", "2345678901", "3456789012"};
        List<String> items = Arrays.asList(itemCodes);
        // When
        orderProcessingService.processOrders(items);
        // Then
        verify(stockMovementService).moveItems(items);
    }
}

```

Run: OrderProcessingServiceTest

Output:

```

OrderProcessingServiceTest 651 ms
  ✓ testProcessOrders_CallsMove 651 ms
[C:\Program Files\Java\jdk-17\bin\java.exe* ...]
Process finished with exit code 0

```

17:23 CRLF UTF-8 4 spaces

```
import static org.junit.jupiter.api.Assertions.*;  
class BillReportTest {  
    @Test  
    void testGenerateBillReport() {  
        BillReport billReport = new BillReport();  
        List<BillReport.BillEntry> report = billReport.generateBillReport();  
  
        // Check that the list is not null  
        assertNotNull(report, message: "Report list should not be null");  
  
        // Optionally check that the list is not empty  
        assertFalse(report.isEmpty(), message: "Report list should not be empty if the DB has data");  
    }  
}
```

Run BillReportTest

BillReportTest 582 ms Tests passed: 1 of 1 test – 582 ms

testGenerateBillReport() 582 ms

C:\Program Files\Java\jdk-17\bin\java.exe ...
Process finished with exit code 0

```
import static org.junit.jupiter.api.Assertions.*;  
class OnlineReshelveReportTest {  
    @Test  
    void testGenerateReshelveReport() {  
        OnlineReshelveReport report = new OnlineReshelveReport();  
  
        List<OnlineReshelveReport.ReshelveEntry> reshelveEntries = report.generateReshelveReport();  
  
        assertNotNull(reshelveEntries, message: "Report list should not be null");  
        // You can add further checks depending on expected DB state, e.g.:  
        // assertFalse(reshelveEntries.isEmpty(), "Report should not be empty if there is data for today");  
    }  
}
```

Run OnlineReshelveReportTest

OnlineReshelveReportTest 555 ms Tests passed: 1 of 1 test – 555 ms

testGenerateReshelveReport() 555 ms

C:\Program Files\Java\jdk-17\bin\java.exe ...
Process finished with exit code 0

Screenshot of a Java test run in an IDE. The project navigation bar shows 'OnsiteReshelveReportTest' selected. The code editor displays the 'OnsiteReshelveReportTest.java' file. The run output shows a single test case, 'testGenerateReshelveReport()', passed in 536ms. The terminal window shows the command 'java -jar C:\Program Files\Java\jdk-17\bin\java.exe ...' and the message 'Process finished with exit code 0'. The status bar at the bottom indicates the file is 27.2 CRLF, UTF-8, 4 spaces.

```
import dao.OnsiteReshelveReport;
import org.junit.jupiter.api.Test;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;
class OnsiteReshelveReportTest {
    @Test
    void testGenerateReshelveReport() {
        OnsiteReshelveReport report = new OnsiteReshelveReport();
    }
}
```

Screenshot of a Java test run in an IDE. The project navigation bar shows 'OnsiteSalesReportTest' selected. The code editor displays the 'OnsiteSalesReportTest.java' file. The run output shows a single test case, 'testGetOnsiteSalesReport()', passed in 500ms. The terminal window shows the command 'java -jar C:\Program Files\Java\jdk-17\bin\java.exe ...' and the message 'Process finished with exit code 0'. The status bar at the bottom indicates the file is 14:19 CRLF, UTF-8, 4 spaces.

```
import dao.OnsiteSalesReport;
import org.junit.jupiter.api.Test;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;
class OnsiteSalesReportTest {
    @Test
    void testGetOnsiteSalesReport() {
        OnsiteSalesReport report = new OnsiteSalesReport();
    }
}
```

SalesReportTest.java

```
import java.util.List;
import dao.SalesReport;
import org.junit.jupiter.api.Test;

public class SalesReportTest {

    @Test
    public void testGetSalesReport() {
        SalesReport salesReport = new SalesReport();
        List<SalesReport.ReportEntry> reportEntries = salesReport.getSalesReport();
    }
}
```

Run > SalesReportTest

Tests passed: 1 of 1 test – 562 ms

C:\Program Files\Java\jdk-17\bin\java.exe* ...

Process finished with exit code 0

StockReportTest.java

```
import java.util.List;
import dao.StockReport;
import org.junit.jupiter.api.Test;

public class StockReportTest {

    @Test
    public void testGenerateStockReport() {
        StockReport stockReport = new StockReport();
        List<StockReport.StockEntry> entries = stockReport.generateStockReport();
    }
}
```

Run > StockReportTest

Tests passed: 1 of 1 test – 540 ms

C:\Program Files\Java\jdk-17\bin\java.exe* ...

Process finished with exit code 0

```

import static org.junit.jupiter.api.Assertions.*;
import java.util.List;
import org.junit.jupiter.api.Test;

public class ReorderLevelReportTest {
    @Test
    void testGenerateReorderLevelReport() {
        ReorderLevelReport report = new ReorderLevelReport();
        List<ReorderLevelReport.ReorderEntry> entries = report.generateReorderLevelReport();
        // Basic assertions - adjust based on your test DB data
        assertNotNull(entries, message: "Report list should not be null");
    }
}

```

Run ReorderLevelReportTest

ReorderLevelReportTest 534 ms Tests passed: 1 of 1 test – 534 ms

testGenerateReorderLevelReport 534 ms

C:\Program Files\Java\jdk-17\bin\java.exe* ...

Process finished with exit code 0

```

import org.mockito.ArgumentCaptor;
import javax.servlet.http.*;
import java.io.PrintWriter;
import java.io.StringWriter;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

public class BatchServletTest {
    private BatchServlet batchServlet;
    private HttpServletRequest request;
    private HttpServletResponse response;
    private PrintWriter writer;
    private ArgumentCaptor<String> argumentCaptor;

    @Test
    void testValidBatchInsertion() {
        when(batchServlet.insertBatch(argumentCaptor.capture())).thenReturn(true);
        when(argumentCaptor.getValue()).thenReturn("Batch inserted successfully");
        assertEquals(true, batchServlet.insertBatch("Batch inserted successfully"));
        assertEquals("Batch inserted successfully", argumentCaptor.getValue());
    }

    @Test
    void testInvalidDateFormat() {
        when(batchServlet.insertBatch(argumentCaptor.capture())).thenReturn(false);
        when(argumentCaptor.getValue()).thenReturn("Batch insertion failed due to invalid date format");
        assertEquals(false, batchServlet.insertBatch("Batch insertion failed due to invalid date format"));
        assertEquals("Batch insertion failed due to invalid date format", argumentCaptor.getValue());
    }
}

```

Run BatchServletTest

BatchServletTest (controller) 1sec 908 ms Tests passed: 2 of 2 tests – 1sec 908 ms

testValidBatchInsertion() 1sec 903 ms

testInvalidDateFormat() 5ms

C:\Program Files\Java\jdk-17\bin\java.exe* ...

May 24, 2025 4:19:06 PM dao.BatchDAOImpl insertBatch

INFO: Batch inserted successfully:

Process finished with exit code 0

```

import java.util.Arrays;
import java.util.List;

public class ItemServletTest {
    private ItemServlet servlet;
    private ItemDAO itemDAO;
    private Gson gson;

    @BeforeEach
    public void setup() {
        // Create a mock ItemDAO
        itemDAO = mock(ItemDAO.class);
    }
}

```

Run ItemServletTest

itemServletTest (controller) 888ms Tests passed: 5 of 5 tests – 888 ms

- ✓ testGetPriceNotFound()
- ✓ testGetAllItems() 22 ms
- ✓ testGetPriceFound() 3 ms
- ✓ testInvalidAction() 2 ms
- ✓ testValidateCode() 6 ms

Process finished with exit code 0

```

import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

class StockReportServletTest {

    private StockReportServlet servlet;
    private StockReport stockReport;
    private HttpServletRequest request;
    private HttpServletResponse response;
    private MockMvc mockMvc;

    @BeforeEach
    public void setup() {
        servlet = new StockReportServlet();
        stockReport = mock(StockReport.class);
        request = mock(HttpServletRequest.class);
        response = mock(HttpServletResponse.class);
        mockMvc = MockMvcBuilders.standaloneSetup(servlet).build();
    }

    @Test
    public void doGet_ShouldReturnJsonResponse() {
        when(stockReport.getStockReport()).thenReturn("Stock Report Data");
        mockMvc.perform(get("/stock-report"))
            .andExpect(status().isOk())
            .andExpect(content().json("Stock Report Data"));
    }
}

```

Run StockReportServletTest

StockReportServletTest (controller) 732ms Tests passed: 1 of 1 test – 732 ms

- ✓ doGet_ShouldReturnJsonResponse() 732 ms

Process finished with exit code 0

SalesReportServletTest.java

```

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.mockito.Mockito.*;

class SalesReportServletTest {

    1 usage
    private SalesReportServlet servlet;
    2 usages
    private HttpServletRequest request;
    8 usages
    private HttpServletResponse response;

    @BeforeEach
    void setUp() {
        servlet = new SalesReportServlet();
        request = mock(HttpServletRequest.class);
        response = mock.HttpServletResponse.class);
    }

    @Test
    void doGet_ShouldReturnJsonResp() {
        // Given
        when(servlet.getEntries()).thenReturn(dummyEntries);

        // When
        String json = servlet.doGet(request, response);

        // Then
        assertEquals("{\"entries\": " + dummyEntries, json);
    }
}

```

Run SalesReportServletTest

Tests passed: 1 of 1 test – 796 ms

doGet_ShouldReturnJsonResp 796 ms

C:\Program Files\Java\jdk-17\bin\java.exe* ...

Process finished with exit code 0

ReorderLevelReportServletTest.java

```

private void override servlet's doGet to inject dummy data instead of calling DAO
ReorderLevelReportServlet testServlet = new ReorderLevelReportServlet();
override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
    resp.setHeader("Access-Control-Allow-Origin", "*");
    resp.setHeader("Access-Control-Allow-Methods", "GET, POST, OPTIONS");
    resp.setHeader("Access-Control-Allow-Headers", "Content-Type");

    resp.setContentType("application/json");
    resp.setCharacterEncoding("UTF-8");

    String json = new Gson().toJson(dummyEntries);
    resp.getWriter().write(json);
}

```

Run ReorderLevelReportServletTest

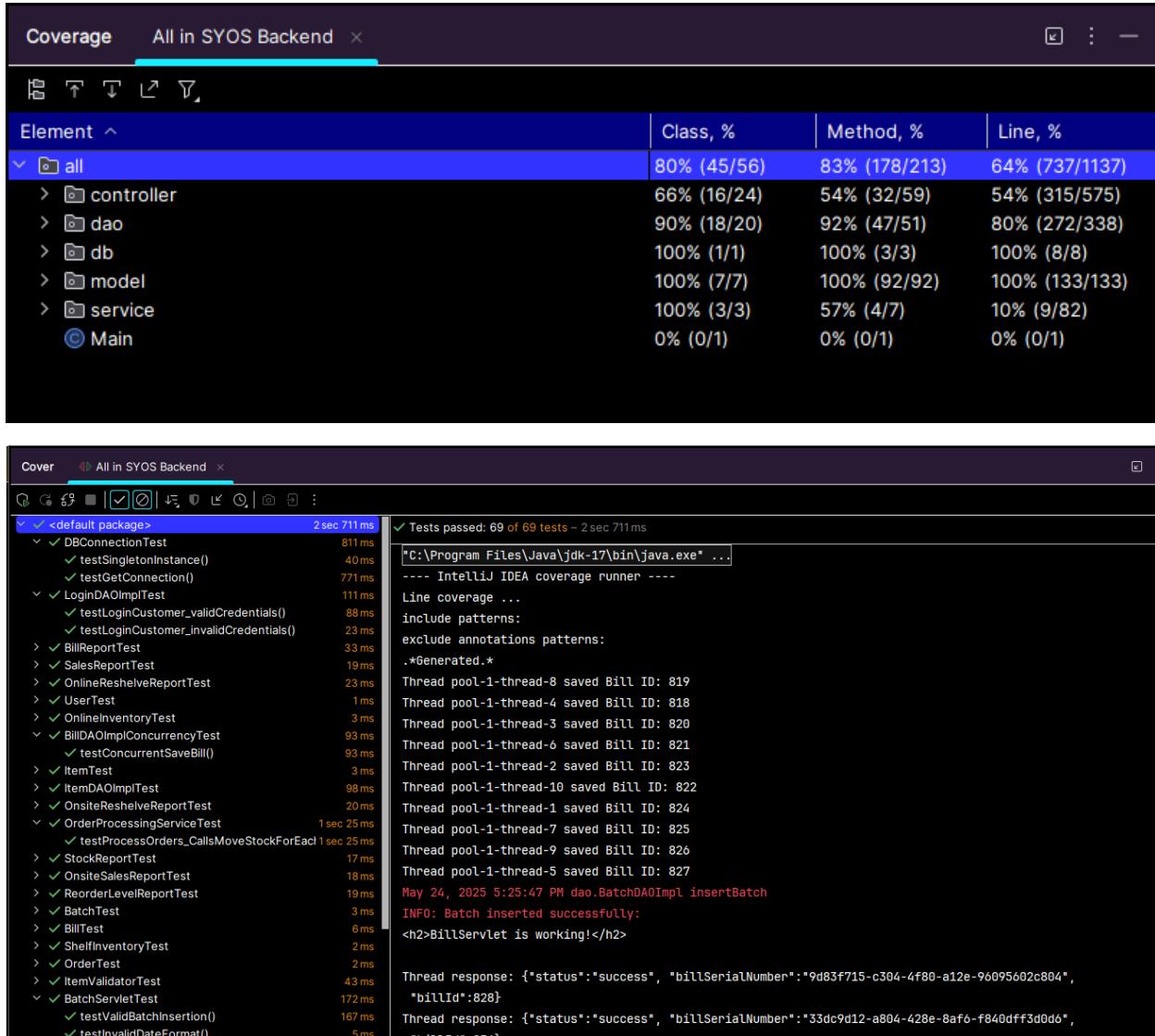
Tests passed: 1 of 1 test – 711 ms

doGet_ShouldReturnJsonResp 711 ms

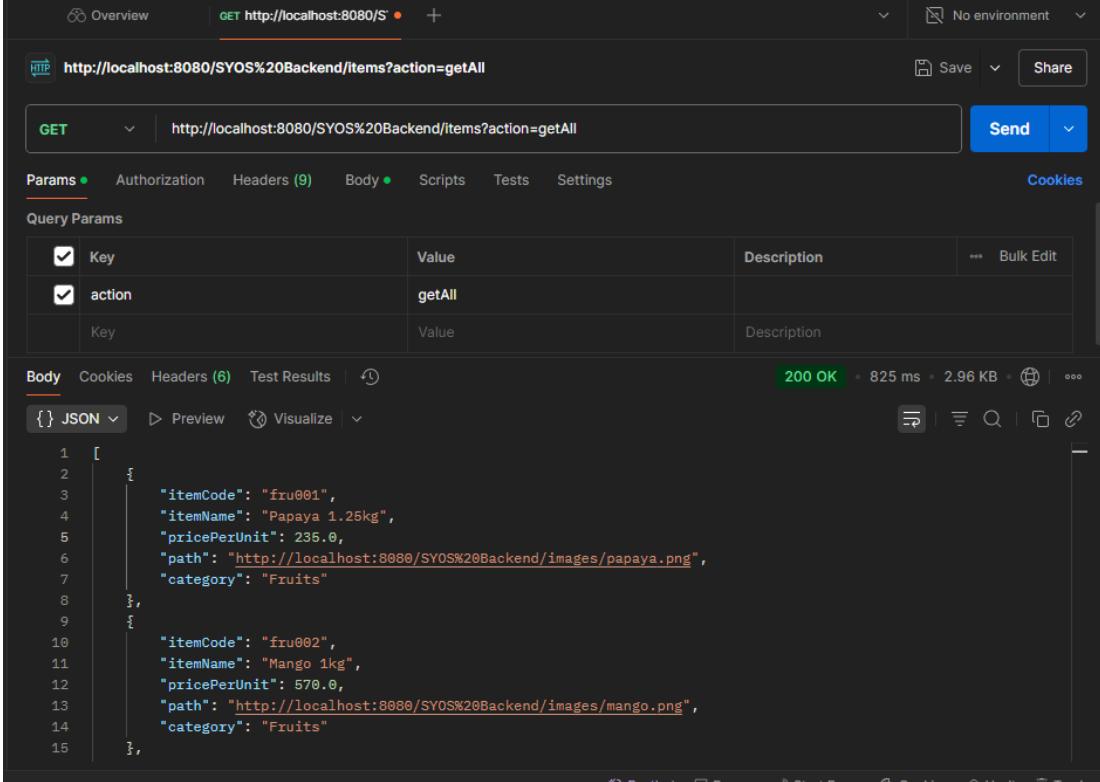
C:\Program Files\Java\jdk-17\bin\java.exe* ...

Process finished with exit code 0

6.2 Test Coverage



Chapter 7 – Postman Tests



HTTP <http://localhost:8080/SYOS%20Backend/items?action=getAll>

GET <http://localhost:8080/SYOS%20Backend/items?action=getAll>

Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

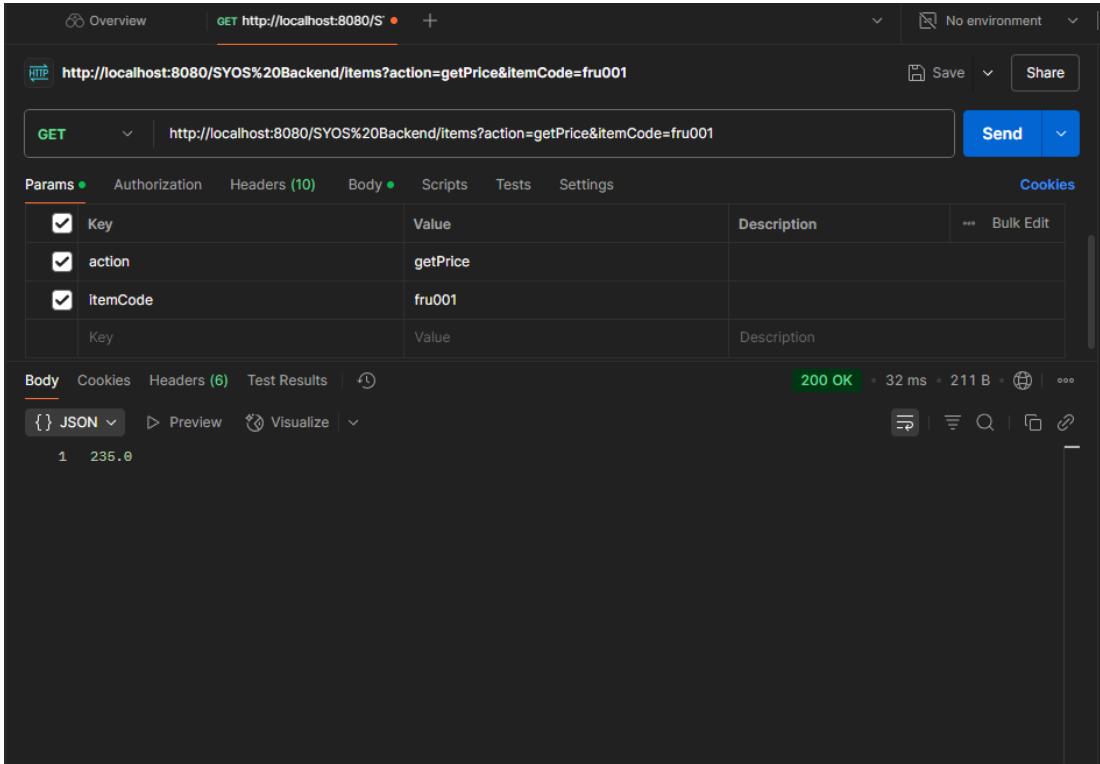
Query Params

Key	Value	Description	Bulk Edit
action	getAll		
Key	Value	Description	

Body Cookies Headers (6) Test Results

200 OK • 825 ms • 2.96 KB

```
1 [  
2 {  
3   "itemCode": "fru001",  
4   "itemName": "Papaya 1.25kg",  
5   "pricePerUnit": 235.0,  
6   "path": "http://localhost:8080/SYOS%20Backend/images/papaya.png",  
7   "category": "Fruits"  
8 },  
9 {  
10   "itemCode": "fru002",  
11   "itemName": "Mango 1kg",  
12   "pricePerUnit": 570.0,  
13   "path": "http://localhost:8080/SYOS%20Backend/images/mango.png",  
14   "category": "Fruits"  
15 }]
```



HTTP <http://localhost:8080/SYOS%20Backend/items?action=getPrice&itemCode=fru001>

GET <http://localhost:8080/SYOS%20Backend/items?action=getPrice&itemCode=fru001>

Send

Params Authorization Headers (10) Body Scripts Tests Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
action	getPrice		
itemCode	fru001		
Key	Value	Description	

Body Cookies Headers (6) Test Results

200 OK • 32 ms • 211 B

```
1 235.0
```

Overview | GET http://localhost:8080/SYOS%20Backend/items?action=validateCode&itemCode=fru001 | No environment

HTTP http://localhost:8080/SYOS%20Backend/items?action=validateCode&itemCode=fru001

Send

GET http://localhost:8080/SYOS%20Backend/items?action=validateCode&itemCode=fru001

Params Authorization Headers (10) Body Scripts Tests Settings Cookies

Key	Value	Description	Bulk Edit
action	validateCode		
itemCode	fru001		
Key	Value	Description	

Body Cookies Headers (6) Test Results

200 OK 45 ms 222 B

```
{ } JSON Preview Visualize
1 {
2   "valid": true
3 }
```

Overview | POST http://localhost:8080/ | No environment

HTTP http://localhost:8080/SYOS%20Backend/login?action=login

POST http://localhost:8080/SYOS%20Backend/login?action=login

Send

POST http://localhost:8080/SYOS%20Backend/login?action=login

Params Authorization Headers (10) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description	Bulk Edit
itemCode	fru001		
username	shsh		
password	ssss		
Key	Value	Description	

Body Cookies Headers (8) Test Results

200 OK 27 ms 345 B

```
{ } JSON Preview Visualize
1 {
2   "message": "Login successful",
3   "userID": 3
4 }
```

Overview | POST http://localhost:8080/ | + | No environment | Save | Share

HTTP http://localhost:8080/SYOS%20Backend/register?action=register

POST http://localhost:8080/SYOS%20Backend/register?action=register

Send

Params • Authorization Headers (10) Body • Scripts Tests Settings Cookies

Body x-www-form-urlencoded

None form-data x-www-form-urlencoded raw binary GraphQL

Key	Description
itemCode	fru001
<input checked="" type="checkbox"/> name	amali
<input checked="" type="checkbox"/> email	amali@gmail.com
<input checked="" type="checkbox"/> phone	0774645433
<input checked="" type="checkbox"/> address	Colombo
<input checked="" type="checkbox"/> username	amali
<input checked="" type="checkbox"/> password	amali123

Body Cookies Headers (8) Test Results | ⚡

200 OK • 23 ms • 344 B • ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂

{ } JSON ▾ ▶ Preview ⚡ Visualize | ⌂

```
1 {  
2   "message": "User registered successfully"  
3 }
```

Overview | POST http://localhost:8080/ | + | No environment | Save | Share | ⌂

HTTP http://localhost:8080/SYOS%20Backend/bill

POST http://localhost:8080/SYOS%20Backend/bill

Send

Params Authorization Headers (10) Body • Scripts Tests Settings Cookies

Body raw JSON

None form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {  
2   "billSerialNumber": "auto-generated-or-empty",  
3   "transactionType": "ONLINE",  
4   "totalAmount": 7000,  
5   "discount": 350,  
6   "cashTendered": 8000,  
7   "cashChange": 650,  
8   "customerID": 3,  
9   "amountAfterDiscount": 6650,  
10  "orders": [  
11    {  
12      "orderID": "generated-uuid",  
13      "itemCode": "ITEM001",  
14      "name": "Product A",  
15      "quantity": 2.  
16    }  
17  ]  
18 }  
19 {  
20   "status": "success",  
21   "billSerialNumber": "877eadfd-116a-4b68-826e-3639bfabd810",  
22   "billId": 838  
23 }
```

Body Cookies Headers (9) Test Results | ⚡

200 OK • 114 ms • 462 B • ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂ | ⌂

{ } JSON ▾ ▶ Preview ⚡ Visualize | ⌂

```
1 {  
2   "status": "success",  
3   "billSerialNumber": "877eadfd-116a-4b68-826e-3639bfabd810",  
4   "billId": 838  
5 }
```

Chapter 8 - Conclusion

The SYOS Billing System successfully automates the traditional billing process by integrating efficient stock management, cash handling and transaction recording within a well structured software architecture. By adopting a three-tier client-server model combined with MVC principles and SOLID design guidelines, the system achieves a clear separation of concerns that enhances maintainability, flexibility and scalability. The implementation of concurrency control at both server and client sides ensures that the system can handle multiple simultaneous users effectively by providing a smooth and responsive experience. Although the system demonstrates solid architectural foundations and practical design patterns, ongoing attention to managing layer boundaries, improving real-time synchronization and optimizing concurrency handling will be critical to maintain performance and reliability as usage scales. Overall, SYOS lays a strong groundwork for a robust billing solution that supports both instore and online sales with potential for further enhancements to meet evolving business and technical requirements.

Chapter 9 – Acknowledgement

I would like to express my heartfelt gratitude to everyone who supported and encouraged me throughout the successful completion of this project. A special thank you goes to my lecturer for the continuous guidance, insightful feedback and motivation that helped me stay focused and improve at every stage. I am also deeply grateful to my class members for their support. Furthermore, I appreciate my family and friends for their unwavering support, patience and encouragement, which played a vital role in helping me stay positive and driven. Their contributions, both direct and indirect, were truly invaluable in making this project a success.