Scenario For Assessment- Module 3

A leading retail company is implementing a new Retail Management System (RMS) to manage inventory, process transactions, track customer loyalty programs, and generate sales reports. The system has both web and mobile interfaces for customer engagement and a separate backend for inventory and sales management accessible to store employees and administrators.

Objectives

1. Ensure Functional Accuracy: Validate that the system accurately processes transactions, manages inventory in real-time, and supports customer loyalty programs.
2. Maintain Quality Standards: Ensure the application aligns with industry standards for performance, security, and usability.
3. Enhance User Experience: Ensure that the application is user-friendly for both customers and employees, minimizing potential disruptions in-store operations.

Key Testing Requirements

1. Manual Testing for Functional Scenarios
   - Order Processing and Transactions: Test cases to validate order placement, payment processing (including multiple payment methods), and receipt generation.
   - Inventory Management: Verify inventory levels update correctly with each sale, return, or inventory adjustment.
   - Customer Loyalty Program: Ensure customer points are accurately calculated and redeemed across various locations.
   - User Roles and Permissions: Verify that different user roles (cashier, manager, admin) have appropriate access rights.

2. Automation Testing Scenarios
   - Regression Testing: Automate core functional tests such as transaction processing and inventory updates to ensure system stability with each release.
   - Performance Testing: Automate load testing on checkout and inventory update functionalities to ensure the application can handle high transaction volumes.

3. Software Test Levels
   - Unit Testing: Developers perform unit testing on individual functions, such as calculating loyalty points and handling inventory checks.
   - Integration Testing: Ensure the integration between the mobile app, web app, and backend systems functions smoothly.
   - System Testing: Full end-to-end testing of the entire RMS, covering workflows from inventory restocking to final sale.
   - User Acceptance Testing (UAT): Involve store employees and administrators in final testing to verify real-world usability and workflow integration.
4. Testing Types

- Functional Testing: Validate that all functional requirements like sales, returns, and inventory management work as expected.
- Non-Functional Testing: Conduct security, load, and performance testing to ensure resilience and compliance with data protection standards.

5. Security Testing:
   - Protect against SQL injections, validate user authentication, ensure secure payment processing, and verify access control on sensitive customer and sales data.

6. Bug Identification & Mock Testing
   - Identify bugs related to transaction processing, inventory discrepancies, and loyalty point errors.
   - Mock testing for high-volume transaction scenarios and scenarios where inventory levels are critically low.

Questions and Scoring Parameters

Question	Key Points to Cover	Marks Allocated	Total Marks
	Explanation of Unit Testing Level	6	
	Explanation of Integration Testing Level	6	
Q1(Concept)	Explanation of System Testing Level	6	30
	Explanation of Acceptance Testing Level	6	
	Relevant examples provided	6	
	Scenario: A customer selects multiple items and proceeds to checkout. They choose to pay half the amount using a credit card and the remaining with a gift card. **Question**: Describe the test cases you would create to ensure the transaction processes correctly, showing how each payment type (credit card and gift card) is validated and the order is confirmed.	15	
Q2 (Test Scenario)	Scenario: An inventory manager updates stock levels after a restock, but there are discrepancies in the quantities of certain products when checked on the mobile app. **Question**: Outline the steps you would take to identify and resolve these inventory discrepancies. What test cases would you develop to ensure inventory data syncs correctly across the web and mobile applications?	15	30

Q3 (Test Cases)	`package your.name;`
	
	`interface ProductDao {`
	`    Product saveProduct(Product product);`
	`    Product getProduct(int productId);`
	`    void deleteProduct(int productId)`
	`    List<Product> getProducts();`
	`}`
	
	Write the unit test cases for the above contract, and use mockito
		40	40

Submission Guidelines:

1. Name your files as per question number. Please keep all your code files (if any) and other files (if any) into a single folder, and then compress them into a zip or rar file. The zipped or rar file should be named following this specific format

<YourName>_<AssessmentName>.zip

or

<YourName>_<AssessmentName>.rar

2. Submit this file in the Lumen at designated place