# **Cardiff Metropolitan University**

**B.Sc. (Hons) in Software Engineering**

**Assignment Cover Sheet**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Student Details ( Student should fill the content)** | | | | | | | | |
| Name | | Ramakrishnan Gayalan Kishor | | | | | | |
| Student ID | | ST20212760 / CL/BSC/SE-CMU-02/02 | | | | | | |
| **Scheduled unit details** | | | | | | | | |
| Unit code | | CIS6003 | | | | | | |
| Unit title | | Advanced Programming | | | | | | |
| Unit enrolment details | | Year | | 3 | | | | |
| Study period | |  | | | | |
| Lecturer | | Mr. Bhagya Rathnayake | | | | | | |
| Mode of delivery | | Full Time | | | | | | |
| **Assignment Details** | | | | | | | | |
| Nature of the Assessment | | Online Billing Symtem Pahana Edu | | | | | | |
| Topic of the Case Study | |  | | | | | | |
| Learning Outcomes covered | | LO1 , LO2, LO3, LO4 | | | | | | |
| Word count | | 4000 | | | | | | |
| Due date / Time | |  | | | | | | |
| Extension granted? | | Yes | No | Extension Date | | |  | |
| Is this a resubmission? | | Yes | No | Resubmission Date | | |  | |
| **Declaration** | | | | | | | | |
| I certify that the attached material is my original work. No other person’s work or ideas have been used without acknowledgement. Except where I have clearly stated that I have used some of this material elsewhere, I have not presented it for examination / assessment in any other course or unit at this or any other institution | | | | | | | | |
| Name/Signature | | R. Gayalan Kishor | | | Date | | 21/08/2025 | |
| **Submission** | | | | | | | | |
| Return to: | |  | | | | | | |
| **Result** | | | | | | | | |
| Marks by 1st  Assessor |  | Name & Signature of the 1st Assessor | | | |  | | **Agreed Mark** |
| Marks by2nd  Assessor |  | Name & Signature of the 2nd Assessor | | | |  | |
| Comments on  the Agreed mark |  | | | | | | | |

Table of Contents

[1](#_Toc206675953)

[**Introduction** 1](#_Toc206675954)

[**Class Diagram Explanation** 2](#_Toc206675955)

[**Use Case Diagram Explanation** 3](#_Toc206675956)

[**Sequence Diagram Explanation** 4](#_Toc206675957)

[**Technologies Used** 5](#_Toc206675958)

[**Database Design** 7](#_Toc206675959)

[**Java Classes** 10](#_Toc206675960)

[**Github Repository** 21](#_Toc206675961)

[**Conclusion** 22](#_Toc206675962)

# **Introduction**

**Project Overview**:

The Pahana Edu Bookshop Web Service is a web-based system designed to facilitate the management of bookshop operations. The service aims to provide functionalities for managing customers, items (books, products, etc.), and users (authentication and registration). It enables administrators to manage a bookshop’s inventory and customer information, while allowing customers to browse and purchase items. The system supports basic CRUD (Create, Read, Update, Delete) operations for both customers and items, ensuring efficient management of the bookshop's resources.

The service is built using modern web technologies, including Java EE (JAX-RS) for the backend, MySQL for database storage, and NetBeans IDE for development. It runs on Apache Tomcat as the web server and supports unit testing with JUnit.

**Purpose of the Project:**

The Pahana Edu Bookshop Web Service serves as a comprehensive solution for managing a bookshop's core operations in a digital environment. The key objectives of this project include:

1. Streamlining Operations: Simplifying the management of customers and items by providing an easy-to-use web interface to perform essential operations such as adding, retrieving, updating, and deleting data.
2. Centralized Data Management: Storing and managing customer and item data in a relational database (MySQL), ensuring that all information is accessible and up to date in one central location.
3. Improving User Experience: Enabling secure user authentication and registration, allowing both administrators and customers to interact with the system effectively while ensuring that only authorized users have access to certain functionalities.
4. Scalability and Security: Ensuring that the web service can handle a growing number of users and data, and implementing secure login and data validation to protect user information and transactions.

The project is significant because it addresses common challenges faced by small to medium-sized bookshops, such as inventory management, customer data storage, and secure user authentication. By providing a digital solution for these tasks, the bookshop can enhance its efficiency and scalability.

**Key Functionalities:**

The Pahana Edu Bookshop Web Service provides the following key functionalities:

1. User Authentication (Login and Registration):  
   This functionality allows users (both customers and admins) to register for an account and securely log into the system. User authentication ensures that sensitive operations,such as managing customer data or placing orders, are only accessible to authorized users.  
   Key Operations:
   * Login: Users can authenticate using a username and password.
   * Registration: New users can create an account with a unique username and password.
2. Customer Management (CRUD):

This functionality enables administrators to manage customer information in the system. The ability to perform CRUD operations ensures that customer data is kept up to date and accurate.

**Key Operations:**

* + Create: Add a new customer to the system.
  + Read: Retrieve customer details by account number.
  + Update: Modify existing customer information, such as address and contact details.
  + Delete: Remove customer records when necessary.

1. Item Management (CRUD):  
   This functionality enables administrators to manage the inventory of items available for sale in the bookshop. It provides the ability to add new items, update details, view item information, and delete items from the inventory.  
   **Key Operations:**
   * Create: Add new items to the inventory, including books and other products.
   * Read: Retrieve item details by item ID.
   * Update: Modify the details of an existing item, such as the price or description.
   * Delete: Remove items from the inventory when no longer needed.

# **Class Diagram Explanation**

In a class diagram, the relationships between the CustomerService and Customer classes, as well as the ItemService and Item classes, define the structure of the system and its interactions.

1. CustomerService and Customer Classes:  
   Relationship: The CustomerService class has a one-to-many relationship with the Customer class, meaning that one instance of the CustomerService class can manage multiple Customer objects.

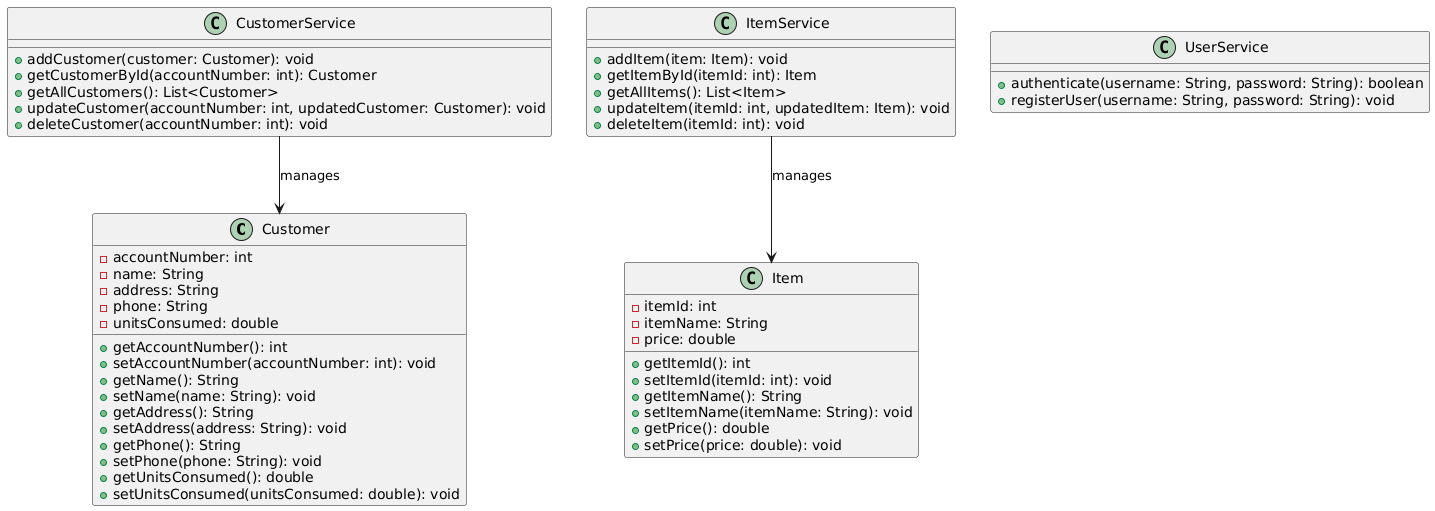
* The CustomerService is responsible for handling operations like adding, updating, or deleting customers. It provides a set of business logic that enables the manipulation of customer data.
* The Customer class stores the customer data, which includes attributes such as name, contact information, and order history.
* Example: The CustomerService can call methods such as addCustomer(), removeCustomer(), or updateCustomer(), and each operation will affect a specific Customer object in the system.

1. ItemService and Item Classes:

* Relationship: Similarly, the ItemService class has a one-to-many relationship with the Item class. This means the ItemService manages a collection of Item objects, which represent the products in the system.
* The ItemService is responsible for managing the catalog of items. It may have operations like adding a new item, updating an item’s details, or deleting an item from the catalog.
* The Item class represents individual products with attributes such as name, price, description, and stock quantity.
* Example: The ItemService can call methods such as addItem (), updateItem (), or removeItem (), affecting a particular Item object within the system.

1. UserService and Authentication Handling:

* Authentication Process: The UserService class handles user authentication by verifying credentials (such as username and password) against the database. It ensures that only authorized users can access certain system functionalities.
* The UserService interacts with a Database class or layer to query stored user information and verify the entered credentials. If the credentials are correct, the UserService allows the user to proceed; otherwise, it raises an error or prompts the user to retry.
* Example: The UserService class will have methods like login () and logout (). The login () method receives the user’s credentials, queries the database for a matching user, and either grants access or denies it based on the outcome of the query.



# **Use Case Diagram Explanation**

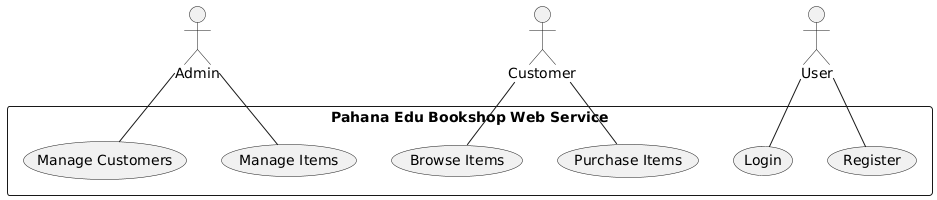
In a use case diagram, the various roles (Admin, Customer, and User) interact with the system to achieve specific goals. Here is a breakdown of each role and its use cases:

1. Roles:

* Admin: The Admin is responsible for managing the system. The admin can perform a variety of tasks, such as adding and removing customers, managing inventory, and overseeing user activities.
* Customer: The Customer interacts with the system by browsing and purchasing items. They are typically end-users who are focused on selecting products, placing orders, and managing their personal information.
* User: The User represents anyone who has access to the system. This could be a customer, admin, or any other authorized individual. The primary responsibility of a User is to log in to the system.

1. Use Cases:

* Admin’s Use Cases:
* Managing Customers: The admin can add new customers, update their information, or remove them from the system. This ensures that the customer database remains up to date.
* Managing Items: The admin has control over the product catalog, allowing them to add, update, or delete items. This helps maintain the accuracy and availability of products.
* Customer’s Use Cases:
* Browsing Items: The customer can view and search for items in the catalog. This includes looking through various product categories, reading descriptions, and checking prices.
* Placing Orders: Customers can place orders by selecting items, adding them to their shopping cart, and proceeding to checkout.
* Managing Profile: Customers can update their personal details, view past orders, and track their current orders.
* User’s Use Case:
* Logging In: The user must provide credentials to authenticate and gain access to the system. Once logged in, they can access their personal dashboard and perform other relevant actions.



# **Sequence Diagram Explanation**

A sequence diagram illustrates how objects interact in a particular scenario, step by step. For this explanation, we will focus on the login process.

1. Login Process Flow:

Step 1: User Enters Credentials:

* The user enters their username and password into the login form.
* The front-end system (User Interface) captures these credentials and sends them to the UserService for authentication.

Step 2: UserService Queries the Database:

* The UserService receives the credentials and queries the Database to check if a user with the provided username exists.
* The UserService sends a request to the Database for matching credentials.

Step 3: Database Response:

* The Database responds with the relevant user data. If the credentials match, the UserService proceeds with authentication; otherwise, it raises an error (e.g., "Incorrect credentials").

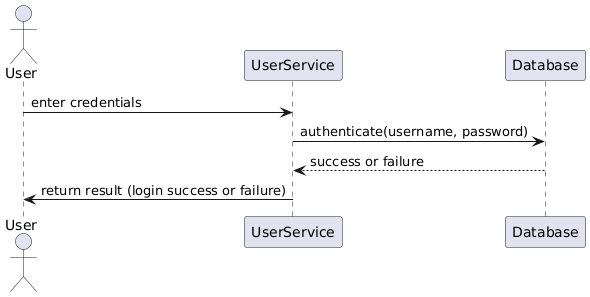
Step 4: User Authentication:

* If the credentials are valid, the UserService grants access to the system and logs the user in.
* The system may return a success message or redirect the user to the main dashboard.

Step 5: Access Granted or Denied:

* If authentication is successful, the user is granted access. If unsuccessful, an error message is displayed, prompting the user to try again.

The sequence diagram visually shows the interaction between the User, UserService, and Database, capturing the request, validation, and response steps.



# **Technologies Used**

The implementation of the Pahana Edu Bookshop Web Service involves a range of technologies and tools that help build, manage, and deploy the web service effectively. Below is an overview of the technologies used in this project and their roles in implementing the system:

**1. Java EE (JAX-RS)**

* **Role**: Java EE (Enterprise Edition) is a robust, industry-standard platform for building enterprise-level applications. Specifically, the JAX-RS (Java API for RESTful Web Services) module is used to create the RESTful APIs in this project. JAX-RS allows the development of web services that can be accessed over HTTP. These APIs are responsible for handling various operations like:
* User Authentication: Verifying user credentials through POST requests.
* CRUD Operations for Customers: Performing Create, Read, Update, and Delete actions on customer records.
* CRUD Operations for Items: Managing the inventory of books and products through similar CRUD operations.

**Benefits:**

* Standardized and simplified handling of HTTP requests and responses.
* Integration with Java-based technologies for backend logic.
* Scalable and secure RESTful web service management.

2. MySQL

* **Role**: MySQL is used as the relational database to store and manage data for the system. It is used to store the customer records, item details (books), and other system-related data. MySQL's role is crucial in maintaining data consistency and integrity.
* Customer Data: Customer information like names, contact details, order history, and authentication credentials (username and password).
* Item Data: Information about each item in the bookshop, such as title, price, author, description, stock quantity, etc.

**Benefits:**

* Fast and efficient data storage and retrieval.
* Support for complex queries and relationships between entities (e.g., customers and orders, items and categories).
* Scalability to handle a growing database as the system expands.

3. NetBeans IDE

* Role: NetBeans is the Integrated Development Environment (IDE) used for developing the entire project. It provides a convenient interface for writing Java code, managing the project structure, and integrating various components.
* Project Management: Organizing files, managing dependencies, and handling project configurations.
* Code Writing: Writing Java classes, interfaces, and RESTful endpoints for the system.
* Debugging and Testing: Debugging the code and running unit tests to ensure the system works as expected.

**Benefits:**

* Efficient code management and navigation.
* Built-in tools for Java development, including Maven support, version control, and debugging.
* Integrated support for Java EE projects, including JAX-RS web services.

4. Apache Tomcat

* **Role**: Apache Tomcat is the web server used to deploy and run the Java EE application. It serves as the container for the Java web application, handling HTTP requests and responses from users. Tomcat is used to deploy the JAX-RS-based RESTful services and ensure they are accessible over the web.
* Deployment: Hosting the web service on the server.
* Request Handling: Listening for incoming HTTP requests, forwarding them to the appropriate endpoints, and returning responses to the client.

**Benefits:**

* Open-source and widely used in Java-based applications.
* Lightweight and efficient for deploying web services.
* Easy to configure and scale for handling growing traffic.

# **Database Design**

The database schema for the Pahana Edu Bookshop Web Service is structured to efficiently store and manage customer and item data. The database is designed using MySQL, and the key entities are represented by two main tables: Customers and Items.

**Database Name**: BookshopDB

The database used to store the information related to customers, items, and transactions is named BookshopDB. This database contains two primary tables: Customers and Items, which store customer details and item/product information, respectively.

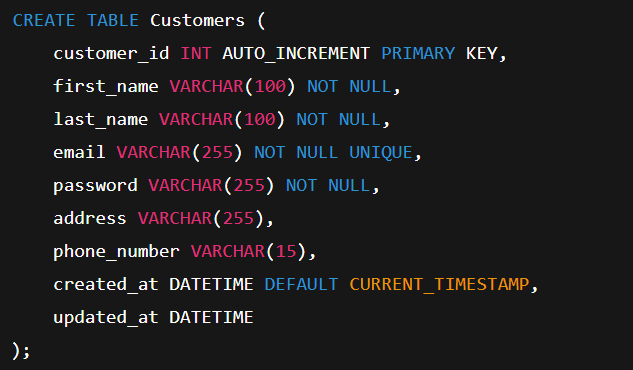
Tables:

1. Customers Table

The Customers table holds information about the customers, including personal details and credentials for authentication. Here is the schema for the Customers table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| customer\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for each customer. |
| first\_name | VARCHAR(100) | NOT NULL | Customer's first name. |
| last\_name | VARCHAR(100) | NOT NULL | Customer's last name. |
| email | VARCHAR(255) | NOT NULL, UNIQUE | Customer's email address (for authentication). |
| password | VARCHAR(255) | NOT NULL | Hashed password for authentication. |
| address | VARCHAR(255) | NULL | Customer's shipping address. |
| phone\_number | VARCHAR(15) | NULL | Customer's phone number. |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | The date and time when the customer account was created. |
| updated\_at | DATETIME | NULL | The date and time when the customer account was last updated. |

SQL Query to Create Customers Table:

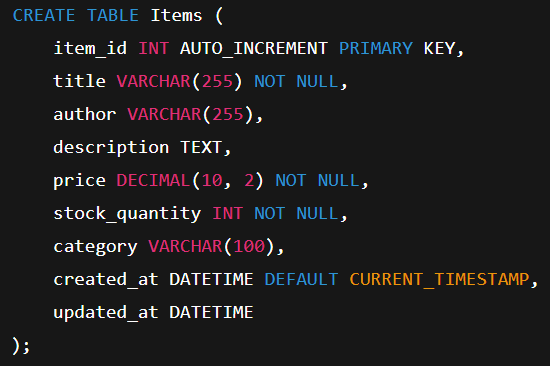


1. Items Table

The Items table stores details about the items available in the bookshop. This includes the product name, price, description, stock quantity, and other relevant information. Here's the schema for the Items table:

Table Structure for Items:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** | **Description** |
| item\_id | INT | PRIMARY KEY, AUTO\_INCREMENT | Unique identifier for each item. |
| title | VARCHAR(255) | NOT NULL | The title/name of the item (book, product, etc.). |
| author | VARCHAR(255) | NULL | The author or creator of the item. |
| description | TEXT | NULL | A detailed description of the item. |
| price | DECIMAL(10, 2) | NOT NULL | The price of the item. |
| stock\_quantity | INT | NOT NULL | The available quantity in stock. |
| category | VARCHAR(100) | NULL | The category or genre of the item (e.g., fiction, non-fiction). |
| created\_at | DATETIME | DEFAULT CURRENT\_TIMESTAMP | The date and time when the item was added to the inventory. |
| updated\_at | DATETIME | NULL | The date and time when the item details were last updated. |

SQL Query to Create Items Table:  
 

Relationships Between Tables

* Customers: Each customer is uniquely identified by the customer\_id and can be associated with multiple orders (though order details are not captured in the schema above, this can be expanded in future iterations). The email field is used for authentication purposes when users log in.
* Items: Each item is uniquely identified by the item\_id and has attributes like title, price, author, and stock\_quantity. This table provides all the necessary information to display and manage items in the bookshop.

Additional Considerations

* Indexes: For better performance, especially with larger datasets, the email column in the Customers table could be indexed, as it is frequently used for user authentication. Similarly, the title column in the Items table can be indexed for quick searching.
* Foreign Keys: In a more complex system, you may need foreign keys to associate customers with their orders, or items with specific categories or suppliers. These can be added as needed.

# **Java Classes**

**Customer.java**

The Customer class represents a customer in the system. It contains essential attributes for storing a customer's personal information, such as name, address, phone number, and units consumed. This class provides getters and setters for these attributes to allow easy access and modification of customer details.

**Purpose:**

The purpose of the Customer class is to model a customer’s data in the system. It acts as an entity that holds information about the customer, which can be stored in a database or passed around within the service layer of the application.

Attributes:

1. accountNumber:

* A unique identifier for each customer (primary key).
* Data Type: int
* Purpose: This uniquely identifies a customer in the system.

1. name:

* The customer's full name.
* Data Type: String
* Purpose: Holds the name of the customer.

1. address:

* The address of the customer.
* Data Type: String
* Purpose: Represents the physical address where the customer resides or receives deliveries.

1. phone:

* The phone number of the customer.
* Data Type: String
* Purpose: Holds the customer's contact phone number.

1. unitsConsumed:

* The total number of units consumed by the customer (e.g., for billing or usage purposes).
* Data Type: double
* Purpose: Used for tracking the customer's usage, whether for billing or statistical purposes.



The Customer class encapsulates the core attributes of a customer and provides the necessary methods to manipulate and retrieve customer information. This class is a central part of the system, allowing for easy management of customer data in the Pahana Edu Bookshop Web Service

**Item.java**

The **Item** class represents an item in the bookshop. It contains essential details about the item, such as its ID, name, price, and other related attributes. This class provides getter and setter methods to access and modify these attributes.

**Purpose:**

The purpose of the Item class is to model the details of an item in the bookshop. Each item could represent a product like a book, a digital product, or other inventory items. The class allows the system to store, retrieve, and modify item information.

Attributes:

1. itemId:

* A unique identifier for each item (primary key).
* Data Type: int
* Purpose: This uniquely identifies the item in the system.

1. itemName:

* The name of the item (e.g., "Java Programming Book").
* Data Type: String
* Purpose: Represents the title or name of the item.

1. price:

* The price of the item.
* Data Type: double
* Purpose: Holds the cost of the item, which may be used for billing or displaying on the bookshop's interface.



The Item class encapsulates the core attributes and behavior of items in the bookshop, such as the item ID, name, and price. By providing getter and setter methods, the class enables flexible management of item data. The toString () method provides a user-friendly representation of the item’s details, making it easier to debug and log item information. This class is a fundamental part of the bookshop’s inventory management system.

**CustomerService.java**

The CustomerService class is responsible for managing all operations related to customers within the Pahana Edu Bookshop Web Service. It provides functionality for performing CRUD (Create, Read, Update, Delete) operations on customer data. The class interacts directly with the database to handle customer records.

**Purpose:**

The purpose of the CustomerService class is to manage customer-related operations such as adding new customers, retrieving customer details, updating customer information, and deleting customer records. This class acts as the business logic layer between the user interface (or API endpoints) and the database.



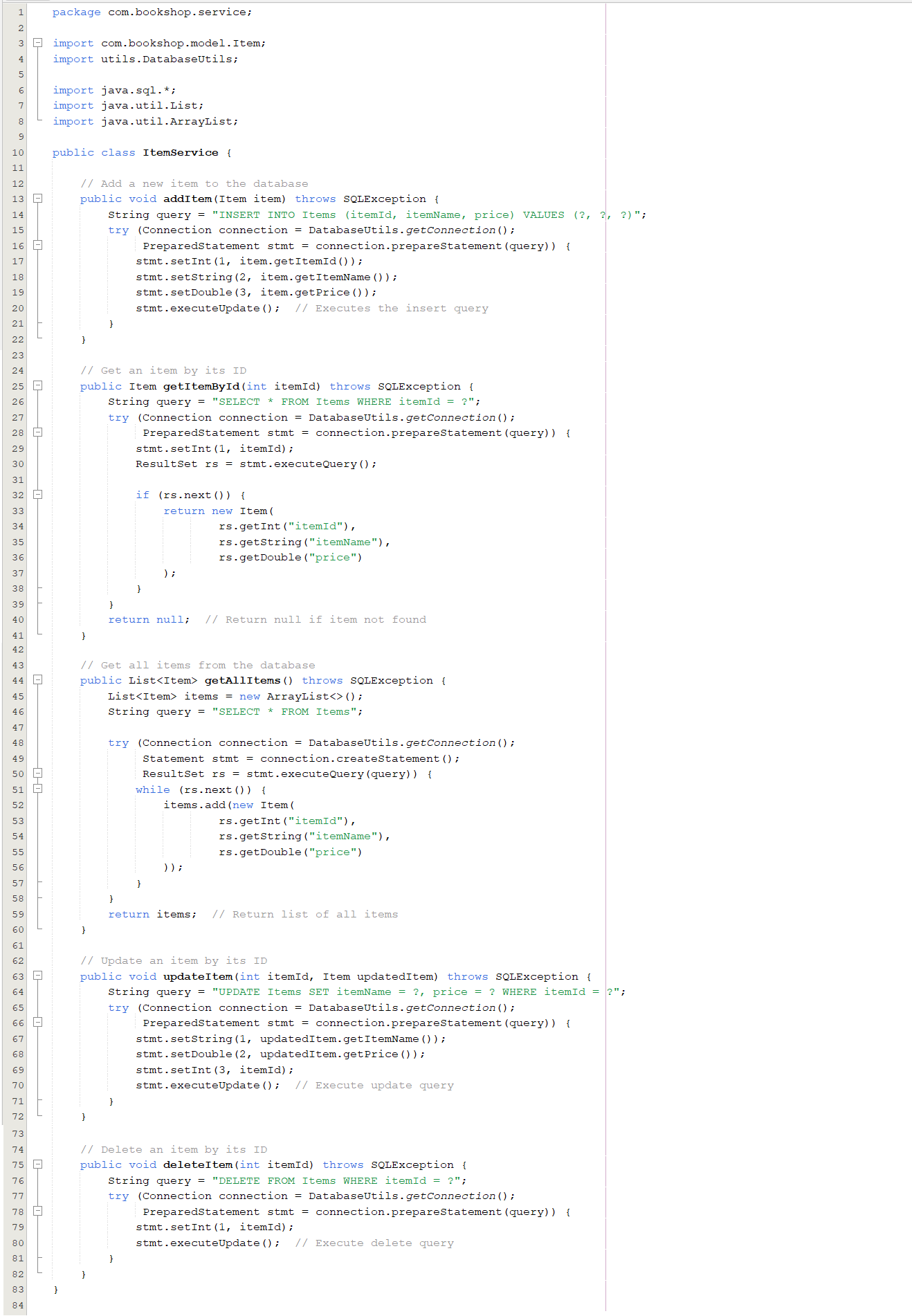
The CustomerService class handles all customer-related operations in the Pahana Edu Bookshop Web Service. By interacting with the database, it provides functionality for adding, retrieving, updating, and deleting customer records. This class is essential for maintaining customer data and ensuring the system runs smoothly.

**ItemService.java**

The ItemService class is responsible for handling all CRUD (Create, Read, Update, Delete) operations related to items in the Pahana Edu Bookshop Web Service. This class interacts with the database to manage item data, including adding, retrieving, updating, and deleting items.

**Purpose:**

The ItemService class serves as the business logic layer for managing items in the bookshop. It provides methods that perform database operations related to items, ensuring that the inventory is managed efficiently and accurately.



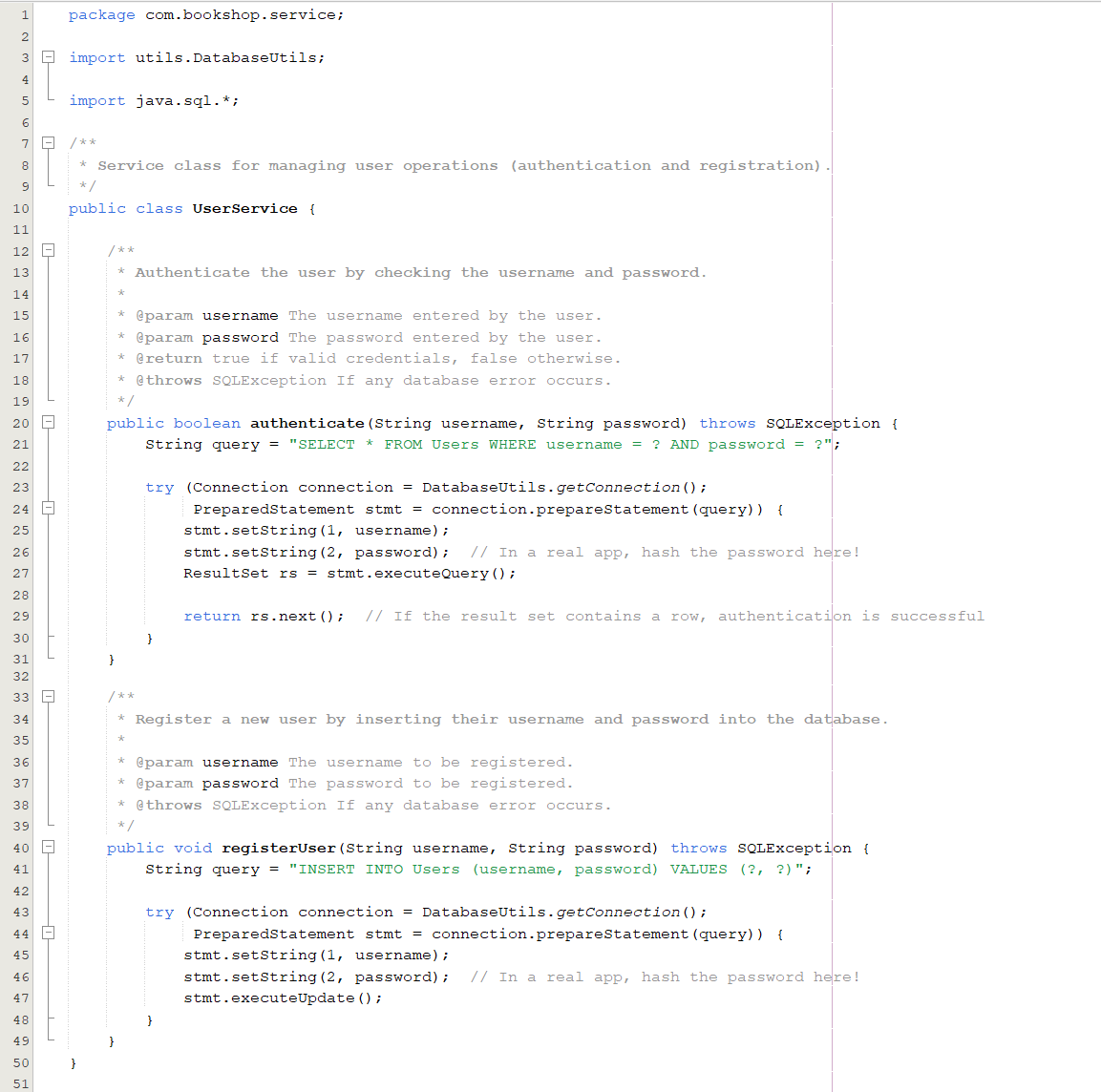
The ItemService class provides an essential set of methods for managing items in the Pahana Edu Bookshop Web Service. It allows the system to handle various operations related to items, such as adding, retrieving, updating, and deleting inventory. By interacting with the database, the ItemService class ensures that the inventory data is accurately managed and can be efficiently accessed and modified.

**UserService.java**

The UserService class is responsible for handling user authentication operations such as login and registration in the Pahana Edu Bookshop Web Service. This class provides the necessary methods to authenticate users by validating their credentials and register new users by storing their information in the database.

**Purpose**:

The purpose of the UserService class is to manage user authentication processes. It provides functionality for verifying user credentials during login and handling user registration. The service ensures secure handling of user data, enabling only authorized users to access specific system functionalities.



The UserService class plays a crucial role in the Pahana Edu Bookshop Web Service by handling user authentication and registration processes. It ensures that only valid users can log in and gain access to protected parts of the system. By interacting with the database, it securely verifies credentials and registers new users, providing a solid foundation for managing user access.

# **Github Repository**

**Overview**

The GitHub repository for this project serves as the central hub for all source code, documentation, and version control activities. It allows developers, collaborators, and reviewers to work together efficiently, facilitating collaboration and ensuring that changes to the codebase are well-documented and tracked over time. The repository helps manage different versions of the code, making it easy to review past modifications, manage branches for feature development, and merge changes seamlessly. Additionally, it provides a clear history of the project's progress, including commit messages, pull requests, and issue tracking, which contributes to effective project management.

**Key Features and Benefits:**

* Version Control: GitHub uses Git for version control, which allows developers to keep track of all changes made to the codebase. Each change is logged with detailed commit messages, which describe the modification made, making it easier to understand the evolution of the project over time.
* Collaboration: With features like branches, pull requests, and merging, GitHub enables multiple developers to work on different features or fixes simultaneously. This helps maintain a clean and stable main branch, while new features or bug fixes can be developed in isolated branches and then reviewed and merged when ready.
* Issue Tracking: GitHub provides an integrated issue tracking system that allows developers to report bugs, request features, or track progress on tasks. Each issue can be assigned to a team member, given a priority, and linked to relevant pull requests for clarity.
* Documentation: The repository can include documentation files, such as README.md, which provide a detailed description of the project, installation instructions, usage guidelines, and any additional notes for developers or users. This helps new contributors get up to speed with the project quickly.
* Code Review: GitHub's pull request feature makes it easy to review code changes before they are merged into the main branch. Developers can discuss the proposed changes, suggest improvements, and ensure that the code follows the project's standards and best practices.
* Continuous Integration (CI): GitHub integrates with various CI/CD tools, such as GitHub Actions, Travis CI, or CircleCI, which can automatically run tests on each commit or pull request. This ensures that new code does not break existing functionality and meets the quality standards set by the project.
* Access Control and Permissions: GitHub provides flexible access control options, allowing administrators to control who can view, edit, or commit to the project. This ensures that only authorized contributors can make changes to the codebase.
* Project Management Tools: GitHub includes Projects and Milestones to organize tasks, track progress, and set deadlines. This makes it easy to manage project timelines, track deliverables, and coordinate efforts between different team members.
* Open Source Contributions: If the project is open-source, the repository enables external contributors to fork the project, make changes, and submit their contributions via pull requests. This fosters collaboration beyond the core team and can lead to faster development and improvement of the project.
* **How to Use the Repository:**
* Clone the Repository: Developers can clone the repository locally to work on the project using Git, allowing them to make changes and test them before committing.
* Create Branches: For every new feature or bug fix, developers can create a new branch. This allows them to work independently of the main branch and avoids conflicts.
* Commit Changes: After making changes, developers commit them to their local branch with detailed commit messages describing what was done. These commits are then pushed to the repository.
* Pull Requests: Once a branch is ready to be merged into the main branch, a pull request is created. This triggers a review process where other developers can examine the changes, suggest improvements, and approve the merge.
* Merge: Once the pull request is reviewed and approved, it can be merged into the main branch. The main branch will always reflect the most stable version of the code.

**Repository Link:**

The GitHub repository for this project can be accessed at the following link:

Repository Link: <https://github.com/GayalanK/PahanaEduBookshopWebService/tree/main>

# **Conclusion**

Despite the challenges encountered during the development process, each issue was met with thoughtful and effective solutions, contributing to the creation of a stable and robust Pahana Edu Bookshop Web Service. Below is a summary of the challenges faced, the solutions implemented, and the overall outcomes:

* Database Privileges: Initially, issues arose due to insufficient database privileges, preventing the application from performing CRUD operations. The problem was resolved by adjusting the MySQL user privileges, ensuring the application could interact with the database seamlessly.
* Data Validation: We encountered issues with handling null values and empty fields in the database. This was addressed by implementing proper validation for both incoming data and database schema adjustments to allow for nullable fields when appropriate. This ensured that the system could handle incomplete data without causing errors or interruptions.
* Database Connection Issues: At times, the connection between the web service and the database was unreliable. This was resolved by reviewing and correcting the JDBC connection URL and ensuring that MySQL was running correctly. These fixes provided a stable and consistent connection between the web service and the database.
* JSON Validation: We faced challenges with ensuring proper data format and validation when handling JSON inputs in API endpoints. By implementing stricter validation checks and handling invalid data appropriately, we improved the stability and reliability of the API.
* Test Data Management: Issues were encountered during testing due to inconsistent or missing test data. We addressed this by inserting correct test data and resetting the database to a known state before running each test, which improved test accuracy and reliability.

These thoughtful solutions not only resolved the challenges but also contributed to the web service becoming more reliable and efficient. The careful handling of test data, JSON validation, and database access has made the final product ready for real-world use.

The lessons learned from tackling these challenges will serve as a foundation for further enhancements and optimizations in the future. As the Pahana Edu Bookshop Web Service evolves, these improvements will ensure that the system remains robust, scalable, and adaptable to new requirements, providing secure and efficient management of customers, items, and user authentication.