Order Management System

> - *Order Management System is an application to track the details of orders placed by customers.*
> - *It is implemented with Core Java Features with JDBC and JUnit 4.*
> - *It is a console based java application that provides functionalities such as customer management, orders management and stock management.*

Application Functionalities

> 1. *Add New Customer*
> 2. *Add new Purchase Order*
> 3. *Add new Stock Item*
> 4. *Fetch customer based on id*
> 5. *Fetch orders placed by specific customers*
> 6. *Fetch orders placed for a specific duration*
>
>   > *a. fetch all orders placed between from and to date inclusive of both dates*
>   > *b. fetch all the orders placed for the given date*
>
> 7. *Update order status to ship and update ship date based on order id.*
> 8. *Fetch delayed orders*
>
>   > *a. By default, every order placed by the customer should get dispatched within the four days after placing the order(inclusive placed date). If an order is not dispatched within the four days after placing date, it should be considered as delayed order*
>
> 9. *Fetch all Stock Items*
> 10. *Find month-wise total orders shipped.*
> 11. *Find total amount collected based on months*
> 12. *Find customer who has made maximum orders.*
> 13. *Generate bill for customer for a specific order. (create a file under bills/customerid directory)*

Application Structure

> - *Model Layer- Model classes defines the class models.*
>
>   > 1. *Customer*
>   > 2. *PurchaseOrder*
>   > 3. *OrderItems*
>   > 4. *StockItem*
>
> - *Data Access Object(DAO) Layer- Dao classes are used to isolate business layer(logic) from database layer. It provides an abstract interface to a database.*
>
>   > 1. *CustomerDao*
>   > 2. *PurchaseOrderDao*
>   > 3. *OrderItemsDao*
>   > 4. *StockItemDao*

> - *Service Layer- A Service class/interface provides a way for users to interact with the functionalities of the application.*
>
>   > 1. *CustomerService*
>   > 2. *PurchaseOrderService*
>   > 3. *OrderItemsService*
>   > 4. *StockItemService*
>
> - *Util - Contains the utility classes such as database connections, enums, etc.*
>
>   > 1. *ConnectionManager*
>   > 2. *OrderStatus(Enum)*
>
> - *TestOrderManagementSystem Class - For using the system functionalities.*
> - *Also the logs are created and saved under 'logs/' directory.*

Setup Database

> - *Replicate the database using queries from **'SQL_Script.txt'** to create Database, Tables, And Populate them with Test Data. (**Run SQL queries one by one**).*

Setup Application

> - *Create a Eclipse Java Project with Name "OrderManagementSystem".*
> - *Download and copy all the folder contents to eclipse "OrderManagementSystem" project directory.*
> - *Add JDBC jar files to eclipse project to work with JDBC functions.*
> - *Add JUnit 4 Library in eclipse for this project to test DAO classes.*
> - *Modify config.properties based on your system for DB connection*
>
> ```
> mysql.url=jdbc:mysql://localhost:3306/order-management_db
> mysql.username=user_name
> mysql.password=user_password
> ```
>
> - *Make sure your MySQL Database connection is up and running.*
> - *Run TestOrderManagementSystem.java to start executing the application.*

Application Tested On

> - *Eclipse IDE Photon*
> - *JDK version: 1.8*
> - *MySQL Server 8.0*
> - *JDBC 4.0*
> - *JUnit 4*