**Documentation**

Task: JDK setup, MYSQL installation and GitHub Repo Setup

Student Name(s): Mohd Faizan

Date Completed: 09-sept-25

1.⁠ ⁠Objective:- The objective of this task is to successfully install and configure the Java Development Kit (JDK) , MYSQL on my local machine to prepare the development environment for the project and push the project structure to GitHub.

2.⁠ ⁠Steps Taken:-

* Downloaded the JDK installer from the official Oracle website.
* Ran the installer and followed the on-screen instructions.
* Verified the installation by opening a command prompt and running java -version.
* Downloaded MYSQL from its official website and setup it on local machine.
* Made new GitHub repository named as Inventory Management System.
* Push the code from local repository using git commands.

3.⁠ ⁠Challenges Encountered (if any):- N/A

4.⁠ ⁠Verification:-

* I ran the java -version command in my terminal, and it displayed the correct JDK version number, confirming that the installation was successful.
* Made some tables in MYSQL workbench and run some query to fetch the result.
* Verified pushed repository from GitHub website.

**Documentation**

Task: Create Product class , Main Class for storing created products.

Student Name(s): Mohd Faizan

Date Completed: 10-sept-25

1.⁠ ⁠Objective:-

1. The objective of this task is to create a product class using appropriate naming conventions inside appropriate package.
2. Create a Main class and take input from user and store the products into appropriate data structures like array list.

2.⁠ ⁠Steps Taken:-

* Created a simple Product class with attributes like productId,name,type and quantity.
* Written getter and setter methods for each field also a parametrized constructor.
* Created a Main class where I created an ArrayList to store the products.
* Taken the input from user and combine all the input fields and make a product and then stored that product to ArrayList.
* Learned basics of Unit Testing.

3.⁠ ⁠Challenges Encountered (if any):- N/A

4.⁠ ⁠Verification:-

* By printing all the products that are stored in ArrayList I successfully verified that my code is perfectly working both for creating object and storing it into ArrayList.

**Documentation**

Task: Create Inventory Management Class with CRUD support.

Student Name(s): Mohd Faizan

Date Completed: 11-sept-25

1.⁠ ⁠Objective:-

1.The objective of this task is to create a Inventory Management Class which is used to add,update,delete and read the Products.

2.Create a Main class and call the desired CRUD methods according the input of user.

2.⁠ ⁠Steps Taken:-

* Created a Main class which has UI in which user can input which functionality he wants to use with a continuous while loop.
* Created a Inventory-Management class.
* Added a Helper Method which is used to take input from users the details of product he wants to add or update.
* Added a static HashMap which will store the product-Id as key and Product detail for fast access.
* Implemented Five different methods for full CRUD Support.

3.⁠ ⁠Challenges Encountered (if any):- N/A

4.⁠ ⁠Verification:-

* One by one calling all the functions from main method and print the output of all the methods I successfully verified the working of our CRUD functionality.

**Documentation**

Task: Add Exceptional Handling to Inventory Management Class.

Student Name(s): Mohd Faizan

Date Completed: 12-sept-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add Exceptional Handling support to the previously made Inventory Management class as well as in Main class from where I am calling all the CRUD methods.

2.Push the code to GitHub.

2.⁠ ⁠Steps Taken:-

* In my Input Helper which I was using to take input from user for new product for protecting from undesired input I write all the input fields into try block and with the help of catch block I handled Input Mismatch Exception.
* In Read by id suppose if the id is not present in our HashMap then it will return null value , So I handled null value using if and else statement.
* Also follows the same previous step for delete and update method.
* In update method if the user inputs quantity less than zero , then I also handled that exception using if and else statement.
* At last, after verifying all the code pushed it to GitHub.

3.⁠ ⁠Challenges Encountered (if any):- N/A

4.⁠ ⁠Verification:-

* One by one calling all the functions from main method and print the output of all the methods I successfully verified the working of our CRUD functionality.
* By passing different undesired input I checked that my code is able to handle that exception or not and successfully verified that my code is able to handle that exception correctly.

**Documentation**

Task: Add Database connection.

Student Name(s): Mohd Faizan

Date Completed: 15-sept-25

1.⁠ ⁠Objective:-

1.The objective of this task is to replace our HashMap which was previously used to store products with our MySQL database to store product.

2.⁠ ⁠Steps Taken:-

* I made a Util package which contains the which we have to use repeatedly.
* In Util package created a class for making connection with database.
* Created a DAO package in which created a ProductDAO class which contains CRUD methods with all the SQL query we need to make with database.
* Connected all these CRUD methods with Inventory Management class.

3.⁠ ⁠Challenges Encountered (if any):- N/A

4.⁠ ⁠Verification:-

* One by one calling all the functions from main method and print the output of all the methods I successfully verified the working of our CRUD functionality.
* By passing different queries from my Java Program to MySQL using JDBC connections and fetching the result , I verified the JDBC connection working perfectly.

**Documentation**

Task: Replace Predefined Java Exception with Custom Exception.

Student Name(s): Mohd Faizan

Date Completed: 17-sept-25

1.⁠ ⁠Objective:-

1.The objective of this task is to replace predefined java runtime exceptions with our Custom Exception with meaningful message so as to get the correct information what actually is going wrong in our program instead of some predefined messages.

2.⁠ ⁠Steps Taken:-

* Made a Package Named as Exceptions which will contain all our custom exceptions .
* Created some Custom Exceptions like ProductNotFound, InvalidInputException, DuplicateDataException etc.
* Replaced predefined exceptions with my custom exception with meaningful messages.

3.⁠ ⁠Challenges Encountered (if any):- N/A

4.⁠ ⁠Verification:-

* By passing different undesired input I checked that my code can handle that exception or not and successfully verified that my code is able to handle that exception and give meaningful message about what is going wrong in code.

**Documentation**

Task: Add read and write support with CSV file.

Student Name(s): Mohd Faizan

Date Completed: 22-sept-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add support for read the data from CSV file and write it to database and read data from database and generate report in csv file.

2.⁠ ⁠Steps Taken:-

* I added dependency for CSV i.e. opencsv for handling CSV files instead of using basic File reader and writer classes
* In Util package , I made a CSV Helper class file.
* Added two methods inside this class i.e. readDataFromCSV method used for reading data from csv and write it to database.
* Generate Report method used to copy the data of database to CSV file.
* Updated Main class for supporting this feature.
* Updated Inventory Management class with two methods.

3.⁠ ⁠Challenges Encountered :-

* Initially File Handling was a new concept for me , also I find it difficult how to connect it with my CRUD functionality.
* Got some bugs in my code.

4.⁠ ⁠Verification:-

* After Calling generate Report Method I looked into Product.csv file and successfully verified that the data of database has been copied here
* After running the readDataFromCSV method , ran the query for fetching all the rows of table and successfully verified the result.

**Documentation**

Task: Add read product by Category and write JUnit Test Cases.

Student Name(s): Mohd Faizan

Date Completed: 29-sept-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add support for read the products from database based on their category.

2. Write Unit test cases for product model class.

2.⁠ ⁠Steps Taken:-

* In DAO interface added one method named as Search by Category which return a list of product.
* Later on implemented that method.
* Updated Main as well as Inventory Management class for supporting this feature.
* Wrote some basic JUnit test case in the test folder.

3.⁠ ⁠Challenges Encountered :-

4.⁠ ⁠Verification:-

* Calling the search by category method and printing the result, successfully verified the result.
* By running the test cases successfully verified the result.

**Documentation**

Task: Add Mock test using Mockito.

Student Name(s): Mohd Faizan

Date Completed: 04-oct-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add mock tests for all the methods present in product DAO .

2.⁠ ⁠Steps Taken:-

* Created One test class for Mock testing.
* In that first start mocking the behaviour of connection, prepared statement etc. and then injected mocks to Product DAO.
* Then created different methods for testing like test\_getProduct ,test\_addProduct and start mimicking the behaviour of database.

3.⁠ ⁠Challenges Encountered :-

* Initially this was a new thing for me so I found it hard to write testcases because of some connection issues .

4.⁠ ⁠Verification:-

* By running all testcases one by one successfully verified the working of mock tests.

**Documentation**

Task: Create User Class and UserDAO.

Student Name(s): Mohd Faizan

Date Completed: 7-oct-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add a user class which will store the users data.

2.Create a UserDAO and its implementation for manipulating and storing the user’s data into database.

2.⁠ ⁠Steps Taken:-

* In Model Package created a User class with different attributes like username, role etc.
* In DAO package added a class UserDAO and its implementation class.
* Inside this added methods like add user , get user by username etc.

3.⁠ ⁠Challenges Encountered :-

4.⁠ ⁠Verification:-

* One by one calling all the methods of UserDAO successfully verified the functionality.

**Documentation**

Task: Add Unit test and Mock test for User.

Student Name(s): Mohd Faizan

Date Completed: 8-oct-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add Junit tests and mock tests using mockito for different functionalities of UserDAO.

2.⁠ ⁠Steps Taken:-

* Created two different classes one for integration test and one for mocktesting.
* In integration test added test cases for different methods of user like get user by name, add user etc.
* In mock testing first I mocked the behaviour of databases and then added mock test cases for different methods of UserDAO.
* .

3.⁠ ⁠Challenges Encountered :-

4.⁠ ⁠Verification:-

* One by one running all the test cases successfully completed the testing of UserDAO class.

**Documentation**

Task: Add role based access.

Student Name(s): Mohd Faizan

Date Completed: 10-oct-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add role based access to inventory management i.e., for user there will be different menu and for admin there will be more functionality to access.

2.⁠ ⁠Steps Taken:-

* Created a login menu where first user have to input its username and password .
* Based on users credentials we fetch the users data from database and check his role and based on his role we display different types of menu like for admin there is admin menu and for user there is view-only functionality menu.
* In login menu there is also option for new user to register.

3.⁠ ⁠Challenges Encountered :-

4.⁠ ⁠Verification:-

* By login through different credentials of user and admin successfully verified two different menu is visible to two different roles.
* By registering as a new user with new credentials successfully verified the functionality of registration.

**Documentation**

Task: Add Send report by Email Functionality.

Student Name(s): Mohd Faizan

Date Completed: 12-oct-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add functionality for sending email to user the report of our inventory as a csv file.

2.⁠ ⁠Steps Taken:-

* Created a Email-Util class which have a method to send email to user.
* Refactored Menu as well as Main class and added option for generate csv report and send it to user.
* First called the method in CSV Helper class to generate report and then used Email Util class to send this report to the email of user.

3.⁠ ⁠Challenges Encountered :-

4.⁠ ⁠Verification:-

* By calling this method and verification by my mail I successfully verified this functionality.

**Documentation**

Task: Add OTP Authentication.

Student Name(s): Mohd Faizan

Date Completed: 14-oct-25

1.⁠ ⁠Objective:-

1.The objective of this task is to add authentication through otp sent to their email and allow only verified user to access the inventory.

2.⁠ ⁠Steps Taken:-

* In database table of user added two columns for email and status.
* In the Login/SignUp menu for existing user first check if they are verified or not and if they are not verified , first they need to enter their email and based on OTP verification they get access to inventory.
* For new user they need to enter one extra entry i.e. their email address and verify it based on otp and then they will be added to user list and can access.
* If new user does not want to verify their email they can still added to user list but they need to verify their identity later.

3.⁠ ⁠Challenges Encountered :-

4.⁠ ⁠Verification:-

* By login through existing user and verification by otp successfully verified this functionality.
* By registering through email and without email successfully verified registration functionality.

**Documentation**

Task: Add Stock Alert .

Student Name:- Mohd Faizan

Date Completed: 22-oct-25

1.⁠ ⁠Objective:-

The objective of this task is to add stock alert system which will notify admin which stock of product is getting low.

2.⁠ ⁠Steps Taken:-

* In database table of product added one column for threshold value and put its default value to 10.
* Created one stock alert service class which will contains the logic for checking the database and sending alert to admin.
* In Main class, created a daemon thread (Scheduler) which will always checking our database from the background after certain time duration and notify admin. This thread starts as soon as admin logins to inventory.
* In the Email Util class added one method to send email about stocks getting low.

3.⁠ ⁠Challenges Encountered :- N/A

4.⁠ ⁠Verification:-

* By login through admin’s credentials and then checking email I successfully verified that stock alert system is working perfectly.