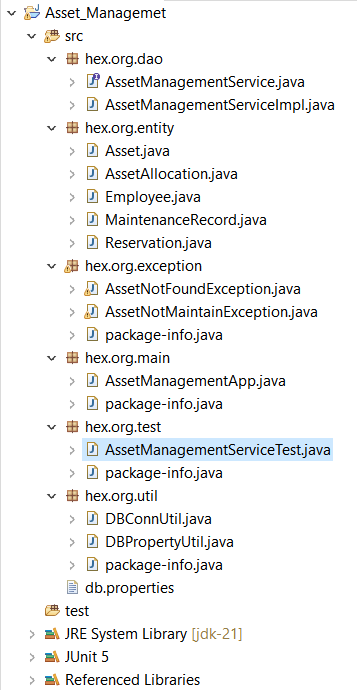
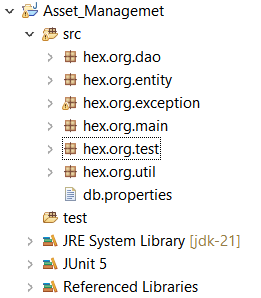
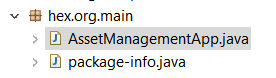
**Case Study on Digital Asset Management Application**

**Objective:**

To develop a Java-based console application for efficiently managing company-owned assets, including their allocation, maintenance, and reservation using JDBC with MySQL.

**Project Structure:**  
  


**Packages:**  
  
  
**Code:**  
  
**1)** **package hex.org.main**  
  
  
  
**AssetManagementApp.java**  
  
package hex.org.main;

import hex.org.dao.AssetManagementService;

import hex.org.dao.AssetManagementServiceImpl;

import hex.org.entity.Asset;

import hex.org.exception.AssetNotFoundException;

import hex.org.exception.AssetNotMaintainException;

import java.util.Scanner;

public class AssetManagementApp {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

AssetManagementService service = new AssetManagementServiceImpl();

while (true) {

System.out.println("\n===== DIGITAL ASSET MANAGEMENT SYSTEM =====");

System.out.println("1. Add Asset");

System.out.println("2. Update Asset");

System.out.println("3. Delete Asset");

System.out.println("4. Allocate Asset");

System.out.println("5. Deallocate Asset");

System.out.println("6. Perform Maintenance");

System.out.println("7. Reserve Asset");

System.out.println("8. Withdraw Reservation");

System.out.println("9. Exit");

System.out.print("Choose an operation: ");

int choice = sc.nextInt();

sc.nextLine(); // consume newline

try {

switch (choice) {

case 1:

System.out.println("Enter asset details:");

System.out.print("Name: ");

String name = sc.nextLine();

System.out.print("Type: ");

String type = sc.nextLine();

System.out.print("Serial Number: ");

String serial = sc.nextLine();

System.out.print("Purchase Date (yyyy-mm-dd): ");

String date = sc.nextLine();

System.out.print("Location: ");

String loc = sc.nextLine();

System.out.print("Status: ");

String status = sc.nextLine();

System.out.print("Owner ID: ");

int owner = sc.nextInt();

Asset asset = new Asset(name, type, serial, date, loc, status, owner);

if (service.addAsset(asset))

System.out.println("Asset added successfully.");

else

System.out.println("Failed to add asset.");

break;

case 2:

System.out.print("Enter Asset ID to update: ");

int updateId = sc.nextInt();

sc.nextLine();

System.out.print("New Name: ");

name = sc.nextLine();

System.out.print("New Type: ");

type = sc.nextLine();

System.out.print("New Serial Number: ");

serial = sc.nextLine();

System.out.print("New Purchase Date (yyyy-mm-dd): ");

date = sc.nextLine();

System.out.print("New Location: ");

loc = sc.nextLine();

System.out.print("New Status: ");

status = sc.nextLine();

System.out.print("New Owner ID: ");

owner = sc.nextInt();

asset = new Asset(updateId, name, type, serial, date, loc, status, owner);

if (service.updateAsset(asset))

System.out.println("Asset updated successfully.");

else

System.out.println("Failed to update asset.");

break;

case 3:

System.out.print("Enter Asset ID to delete: ");

int deleteId = sc.nextInt();

if (service.deleteAsset(deleteId))

System.out.println("Asset deleted successfully.");

else

System.out.println("Failed to delete asset.");

break;

case 4:

System.out.print("Enter Asset ID: ");

int assetId = sc.nextInt();

System.out.print("Enter Employee ID: ");

int empId = sc.nextInt();

sc.nextLine();

System.out.print("Enter Allocation Date (yyyy-mm-dd): ");

String allocDate = sc.nextLine();

if (service.allocateAsset(assetId, empId, allocDate))

System.out.println("Asset allocated successfully.");

else

System.out.println("Failed to allocate asset.");

break;

case 5:

System.out.print("Enter Asset ID: ");

assetId = sc.nextInt();

System.out.print("Enter Employee ID: ");

empId = sc.nextInt();

sc.nextLine();

System.out.print("Enter Return Date (yyyy-mm-dd): ");

String returnDate = sc.nextLine();

if (service.deallocateAsset(assetId, empId, returnDate))

System.out.println("Asset deallocated successfully.");

else

System.out.println("Failed to deallocate asset.");

break;

case 6:

System.out.print("Enter Asset ID: ");

assetId = sc.nextInt();

sc.nextLine();

System.out.print("Enter Maintenance Date (yyyy-mm-dd): ");

String mDate = sc.nextLine();

System.out.print("Enter Description: ");

String desc = sc.nextLine();

System.out.print("Enter Cost: ");

double cost = sc.nextDouble();

if (service.performMaintenance(assetId, mDate, desc, cost))

System.out.println("Maintenance recorded.");

else

System.out.println("Failed to record maintenance.");

break;

case 7:

System.out.print("Enter Asset ID: ");

assetId = sc.nextInt();

System.out.print("Enter Employee ID: ");

empId = sc.nextInt();

sc.nextLine();

System.out.print("Enter Reservation Date (yyyy-mm-dd): ");

String resDate = sc.nextLine();

System.out.print("Enter Start Date (yyyy-mm-dd): ");

String start = sc.nextLine();

System.out.print("Enter End Date (yyyy-mm-dd): ");

String end = sc.nextLine();

if (service.reserveAsset(assetId, empId, resDate, start, end))

System.out.println("Reservation successful.");

else

System.out.println("Failed to reserve asset.");

break;

case 8:

System.out.print("Enter Reservation ID: ");

int resId = sc.nextInt();

if (service.withdrawReservation(resId))

System.out.println("Reservation withdrawn.");

else

System.out.println("Failed to withdraw reservation.");

break;

case 9:

System.out.println("Exiting... Thank you!");

sc.close();

System.exit(0);

break;

default:

System.out.println("Invalid choice.");

}

} catch (AssetNotFoundException | AssetNotMaintainException e) {

System.out.println("Error: " + e.getMessage());

} catch (Exception e) {

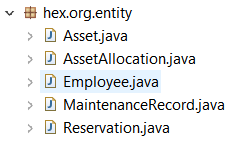
System.out.println("Unexpected Error: " + e.getMessage());

}

}

}

}

**2) package hex.org.entity:**

**Asset.java**  
package hex.org.entity;

public class Asset {

private int assetId; // Renamed from 'id'

private String name;

private String type;

private String serialNumber;

private String purchaseDate;

private String location;

private String status;

private int ownerId;

// Constructor without ID (for insert)

public Asset(String name, String type, String serialNumber, String purchaseDate,

String location, String status, int ownerId) {

this.name = name;

this.type = type;

this.serialNumber = serialNumber;

this.purchaseDate = purchaseDate;

this.location = location;

this.status = status;

this.ownerId = ownerId;

}

// Constructor with ID (for update)

public Asset(int assetId, String name, String type, String serialNumber, String purchaseDate,

String location, String status, int ownerId) {

this.assetId = assetId;

this.name = name;

this.type = type;

this.serialNumber = serialNumber;

this.purchaseDate = purchaseDate;

this.location = location;

this.status = status;

this.ownerId = ownerId;

}

// ✅ Getters and Setters

public int getAssetId() {

return assetId;

}

public void setAssetId(int assetId) {

this.assetId = assetId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getSerialNumber() {

return serialNumber;

}

public void setSerialNumber(String serialNumber) {

this.serialNumber = serialNumber;

}

public String getPurchaseDate() {

return purchaseDate;

}

public void setPurchaseDate(String purchaseDate) {

this.purchaseDate = purchaseDate;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public int getOwnerId() {

return ownerId;

}

public void setOwnerId(int ownerId) {

this.ownerId = ownerId;

}

}

**AssetAllocation.java**  
  
  
package hex.org.entity;

public class AssetAllocation {

private int allocationId;

private int assetId;

private int employeeId;

private String allocationDate;

private String returnDate;

// Constructors

public AssetAllocation() {}

public AssetAllocation(int allocationId, int assetId, int employeeId, String allocationDate, String returnDate) {

this.allocationId = allocationId;

this.assetId = assetId;

this.employeeId = employeeId;

this.allocationDate = allocationDate;

this.returnDate = returnDate;

}

// Getters and Setters

public int getAllocationId() {

return allocationId;

}

public void setAllocationId(int allocationId) {

this.allocationId = allocationId;

}

public int getAssetId() {

return assetId;

}

public void setAssetId(int assetId) {

this.assetId = assetId;

}

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getAllocationDate() {

return allocationDate;

}

public void setAllocationDate(String allocationDate) {

this.allocationDate = allocationDate;

}

public String getReturnDate() {

return returnDate;

}

public void setReturnDate(String returnDate) {

this.returnDate = returnDate;

}

}

**Employee.java**package hex.org.entity;

public class Employee {

private int employeeId;

private String name;

private String department;

private String email;

private String password;

// Constructors

public Employee() {}

public Employee(int employeeId, String name, String department, String email, String password) {

this.employeeId = employeeId;

this.name = name;

this.department = department;

this.email = email;

this.password = password;

}

// Getters and Setters

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}  
  
**MaintenanceRecord.java**

package hex.org.entity;

public class MaintenanceRecord {

private int maintenanceId;

private int assetId;

private String maintenanceDate;

private String description;

private double cost;

// Constructors

public MaintenanceRecord() {}

public MaintenanceRecord(int maintenanceId, int assetId, String maintenanceDate, String description, double cost) {

this.maintenanceId = maintenanceId;

this.assetId = assetId;

this.maintenanceDate = maintenanceDate;

this.description = description;

this.cost = cost;

}

// Getters and Setters

public int getMaintenanceId() {

return maintenanceId;

}

public void setMaintenanceId(int maintenanceId) {

this.maintenanceId = maintenanceId;

}

public int getAssetId() {

return assetId;

}

public void setAssetId(int assetId) {

this.assetId = assetId;

}

public String getMaintenanceDate() {

return maintenanceDate;

}

public void setMaintenanceDate(String maintenanceDate) {

this.maintenanceDate = maintenanceDate;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public double getCost() {

return cost;

}

public void setCost(double cost) {

this.cost = cost;

}

}  
  
  
 **Reservation.java**package hex.org.entity;

public class Reservation {

private int reservationId;

private int assetId;

private int employeeId;

private String reservationDate;

private String startDate;

private String endDate;

private String status;

// Constructors

public Reservation() {}

public Reservation(int reservationId, int assetId, int employeeId, String reservationDate, String startDate, String endDate, String status) {

this.reservationId = reservationId;

this.assetId = assetId;

this.employeeId = employeeId;

this.reservationDate = reservationDate;

this.startDate = startDate;

this.endDate = endDate;

this.status = status;

}

// Getters and Setters

public int getReservationId() {

return reservationId;

}

public void setReservationId(int reservationId) {

this.reservationId = reservationId;

}

public int getAssetId() {

return assetId;

}

public void setAssetId(int assetId) {

this.assetId = assetId;

}

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

public String getReservationDate() {

return reservationDate;

}

public void setReservationDate(String reservationDate) {

this.reservationDate = reservationDate;

}

public String getStartDate() {

return startDate;

}

public void setStartDate(String startDate) {

this.startDate = startDate;

}

public String getEndDate() {

return endDate;

}

public void setEndDate(String endDate) {

this.endDate = endDate;

}

public String getStatus() {

return status;

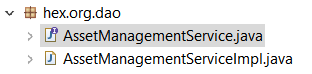
}

public void setStatus(String status) {

this.status = status;

}

}

**3) package hex.org.dao**  
 **AssetManagementService.java**  
  
package hex.org.dao;

import hex.org.entity.Asset;

import hex.org.exception.AssetNotFoundException;

import hex.org.exception.AssetNotMaintainException;

public interface AssetManagementService {

boolean addAsset(Asset asset);

boolean updateAsset(Asset asset);

boolean deleteAsset(int assetId);

boolean allocateAsset(int assetId, int employeeId, String allocationDate) throws AssetNotFoundException;

boolean deallocateAsset(int assetId, int employeeId, String returnDate) throws AssetNotFoundException;

boolean performMaintenance(int assetId, String maintenanceDate, String description, double cost) throws AssetNotMaintainException;

boolean reserveAsset(int assetId, int employeeId, String reservationDate, String startDate, String endDate);

boolean withdrawReservation(int reservationId);

boolean addAssetToMaintenance(int assetId);

boolean reserveAsset(int assetId, int empId);

}  
  
**AssetManagementServiceImpl.java**

package hex.org.dao;

import hex.org.entity.Asset;

import hex.org.exception.AssetNotFoundException;

import hex.org.exception.AssetNotMaintainException;

import hex.org.util.DBConnUtil;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class AssetManagementServiceImpl implements AssetManagementService {

@Override

public boolean addAsset(Asset asset) {

String query = "INSERT INTO assets (name, type, serial\_number, purchase\_date, location, status, owner\_id) VALUES (?, ?, ?, ?, ?, ?, ?)";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setString(1, asset.getName());

ps.setString(2, asset.getType());

ps.setString(3, asset.getSerialNumber());

ps.setString(4, asset.getPurchaseDate());

ps.setString(5, asset.getLocation());

ps.setString(6, asset.getStatus());

ps.setInt(7, asset.getOwnerId());

return ps.executeUpdate() > 0;

} catch (SQLException e) {

System.out.println("Error adding asset: " + e.getMessage());

}

return false;

}

@Override

public boolean updateAsset(Asset asset) {

String query = "UPDATE assets SET name=?, type=?, serial\_number=?, purchase\_date=?, location=?, status=?, owner\_id=? WHERE asset\_id=?";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setString(1, asset.getName());

ps.setString(2, asset.getType());

ps.setString(3, asset.getSerialNumber());

ps.setString(4, asset.getPurchaseDate());

ps.setString(5, asset.getLocation());

ps.setString(6, asset.getStatus());

ps.setInt(7, asset.getOwnerId());

ps.setInt(8, asset.getAssetId()); // <-- Correct method from Asset.java

return ps.executeUpdate() > 0;

} catch (SQLException e) {

System.out.println("Error updating asset: " + e.getMessage());

}

return false;

}

@Override

public boolean deleteAsset(int assetId) {

String query = "DELETE FROM assets WHERE asset\_id=?";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setInt(1, assetId);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

System.out.println("Error deleting asset: " + e.getMessage());

}

return false;

}

@Override

public boolean allocateAsset(int assetId, int employeeId, String allocationDate) throws AssetNotFoundException {

String query = "INSERT INTO asset\_allocations (asset\_id, employee\_id, allocation\_date) VALUES (?, ?, ?)";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setInt(1, assetId);

ps.setInt(2, employeeId);

ps.setString(3, allocationDate);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

throw new AssetNotFoundException("Allocation failed: " + e.getMessage());

}

}

@Override

public boolean deallocateAsset(int assetId, int employeeId, String returnDate) throws AssetNotFoundException {

String query = "UPDATE asset\_allocations SET return\_date=? WHERE asset\_id=? AND employee\_id=?";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setString(1, returnDate);

ps.setInt(2, assetId);

ps.setInt(3, employeeId);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

throw new AssetNotFoundException("Deallocation failed: " + e.getMessage());

}

}

@Override

public boolean performMaintenance(int assetId, String maintenanceDate, String description, double cost) throws AssetNotMaintainException {

String query = "INSERT INTO maintenance\_records (asset\_id, maintenance\_date, description, cost) VALUES (?, ?, ?, ?)";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setInt(1, assetId);

ps.setString(2, maintenanceDate);

ps.setString(3, description);

ps.setDouble(4, cost);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

throw new AssetNotMaintainException("Maintenance failed: " + e.getMessage());

}

}

@Override

public boolean reserveAsset(int assetId, int employeeId, String reservationDate, String startDate, String endDate) {

String query = "INSERT INTO reservations (asset\_id, employee\_id, reservation\_date, start\_date, end\_date) VALUES (?, ?, ?, ?, ?)";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setInt(1, assetId);

ps.setInt(2, employeeId);

ps.setString(3, reservationDate);

ps.setString(4, startDate);

ps.setString(5, endDate);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

System.out.println("Error reserving asset: " + e.getMessage());

}

return false;

}

@Override

public boolean withdrawReservation(int reservationId) {

String query = "DELETE FROM reservations WHERE reservation\_id=?";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setInt(1, reservationId);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

System.out.println("Error withdrawing reservation: " + e.getMessage());

}

return false;

}

@Override

public boolean addAssetToMaintenance(int assetId) {

String query = "UPDATE assets SET status='Maintenance' WHERE asset\_id=?";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setInt(1, assetId);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

System.out.println("Error adding asset to maintenance: " + e.getMessage());

}

return false;

}

@Override

public boolean reserveAsset(int assetId, int empId) {

// Just a dummy basic reservation with current date

String query = "INSERT INTO reservations (asset\_id, employee\_id, reservation\_date) VALUES (?, ?, CURRENT\_DATE)";

try (Connection con = DBConnUtil.getDbConnection();

PreparedStatement ps = con.prepareStatement(query)) {

ps.setInt(1, assetId);

ps.setInt(2, empId);

return ps.executeUpdate() > 0;

} catch (SQLException e) {

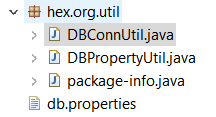
System.out.println("Error reserving asset with simplified method: " + e.getMessage());

}

return false;

}

}

**4) package hex.org.util**  
  
  
  
  
**DBConnUtil.java**

package hex.org.util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.io.IOException;

public class DBConnUtil {

private static final String fileName = "db.properties";

public static Connection getDbConnection() {

Connection con = null;

String connString = null;

try {

connString = DBPropertyUtil.getConnectionString(fileName);

} catch (IOException e) {

System.out.println("Connection String Creation Failed");

e.printStackTrace();

}

if (connString != null) {

try {

con = DriverManager.getConnection(connString);

} catch (SQLException e) {

System.out.println("Error While Establishing DBConnection........");

e.printStackTrace();

}

}

return con;

}

}  
  
**DBPropertyUtil.java**

package hex.org.util;

import java.io.IOException;

import java.io.InputStream;

import java.util.Properties;

public class DBPropertyUtil {

// This method loads the db.properties file from the classpath and builds the connection string

public static String getConnectionString(String fileName) throws IOException {

String connStr = null;

Properties props = new Properties();

// Load properties file from classpath

InputStream is = DBPropertyUtil.class.getClassLoader().getResourceAsStream(fileName);

if (is == null) {

throw new IOException("Property file '" + fileName + "' not found in the classpath");

}

props.load(is);

String user = props.getProperty("user");

String password = props.getProperty("password");

String protocol = props.getProperty("protocol");

String system = props.getProperty("system");

String database = props.getProperty("database");

String port = props.getProperty("port");

connStr = protocol + "//" + system + ":" + port + "/" + database + "?user=" + user + "&password=" + password;

return connStr;

}

}

**Db.properties**user=root

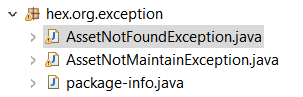
password=Shraddha@2003

protocol=jdbc:mysql:

system=localhost

database=asset\_management\_app

port=3306

**5) package hex.org.exception**  
  
**AssetNotFoundException.java**  
  
package hex.org.exception;

public class AssetNotFoundException extends Exception {

public AssetNotFoundException(String message) {

super(message);

}

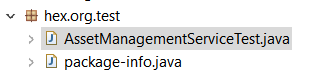
}  
  
**AssetNotMaintainException.java**  
  
package hex.org.exception;

public class AssetNotMaintainException extends Exception {

public AssetNotMaintainException(String message) {

super(message);

}

}  
  
**6) package hex.org.test**  
 **AssetManagementServiceTest.java**  
package hex.org.test;

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

import org.junit.jupiter.api.Test;

import hex.org.dao.AssetManagementServiceImpl;

import hex.org.entity.Asset;

import hex.org.exception.AssetNotFoundException;

import hex.org.exception.AssetNotMaintainException;

public class AssetManagementServiceTest {

AssetManagementServiceImpl service = new AssetManagementServiceImpl();

@Test

void testAddAsset() {

Asset asset = new Asset("Mouse", "Peripheral", "SN12350", "2023-10-01", "Office", "Available", 1);

assertTrue(service.addAsset(asset));

}

@Test

void testMaintenance() throws AssetNotMaintainException {

assertTrue(service.performMaintenance(1, "2024-03-01", "Routine check", 150.0));

}

@Test

void testReservation() {

assertTrue(service.reserveAsset(1, 1, "2024-04-01", "2024-04-05", "2024-04-10"));

}

@Test

void testAssetNotFoundException() {

assertThrows(AssetNotFoundException.class, () -> {

service.allocateAsset(999, 1, "2024-04-01");

});

}

@Test

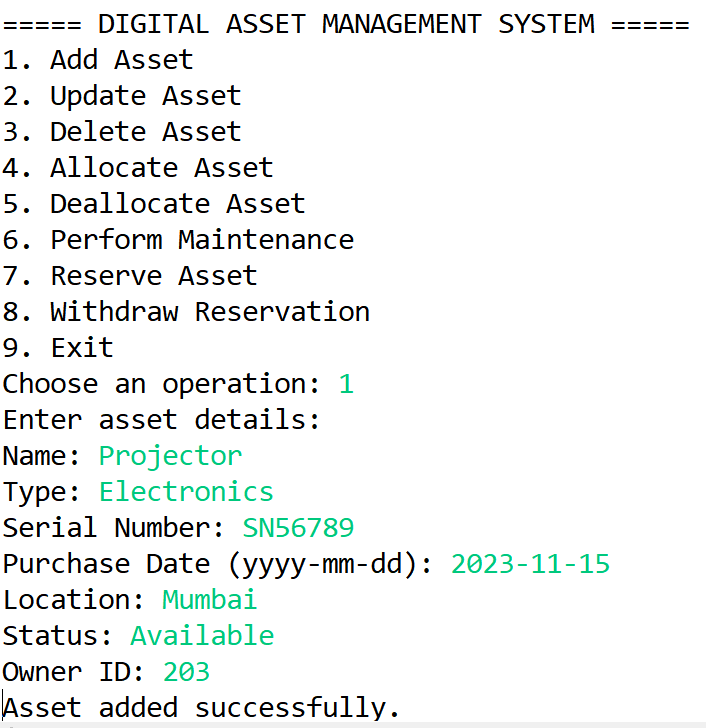
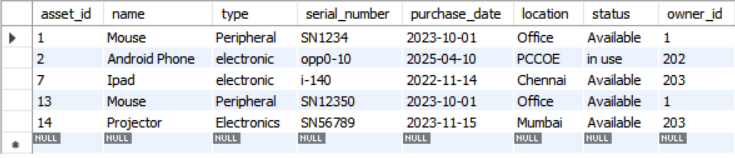
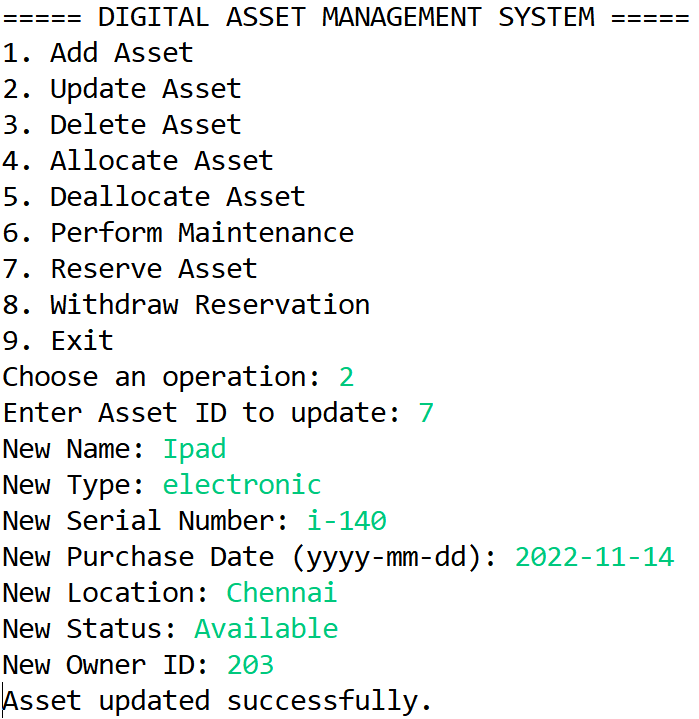
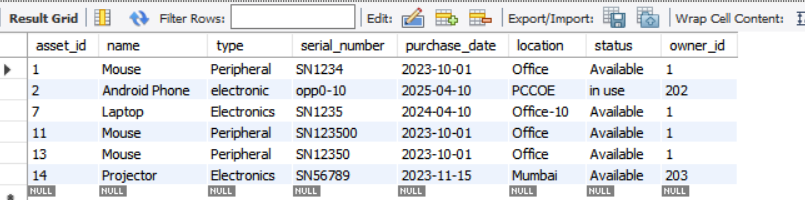
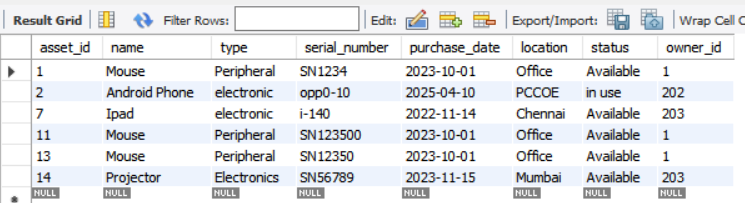
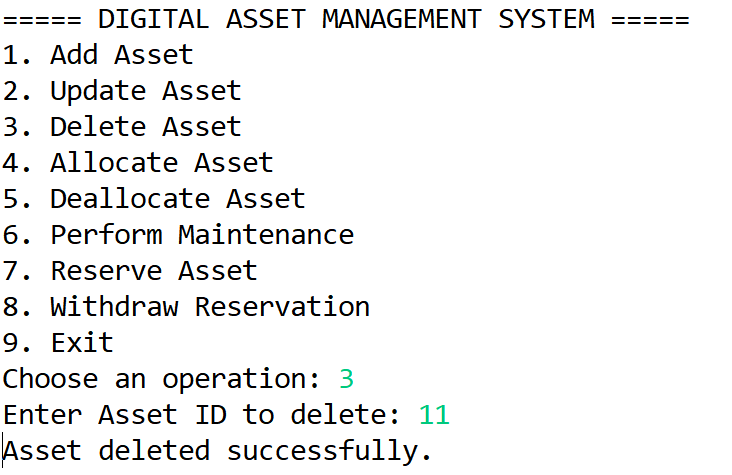
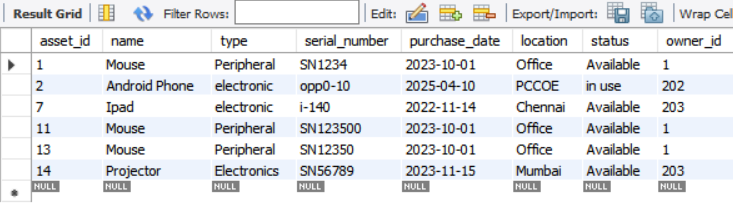
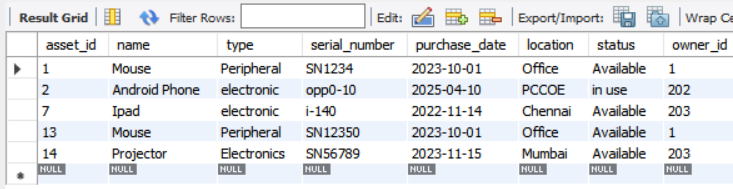
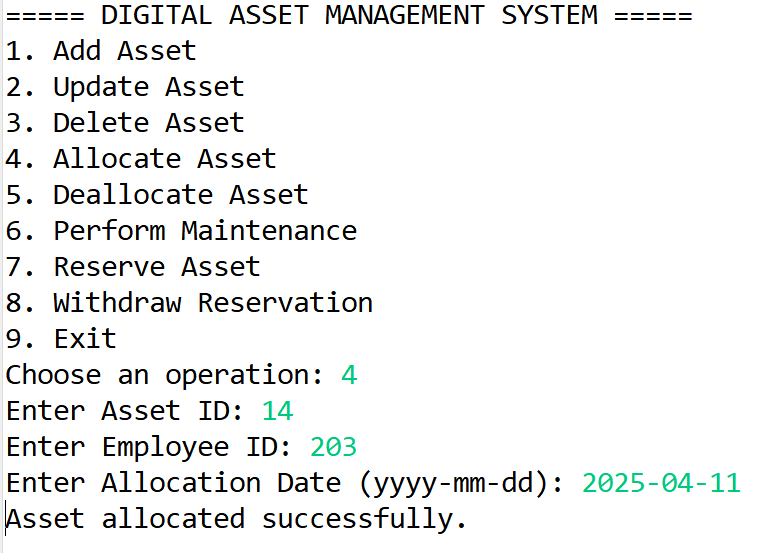
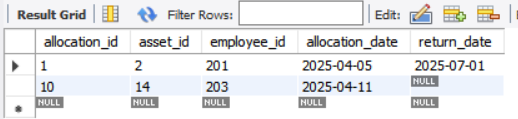
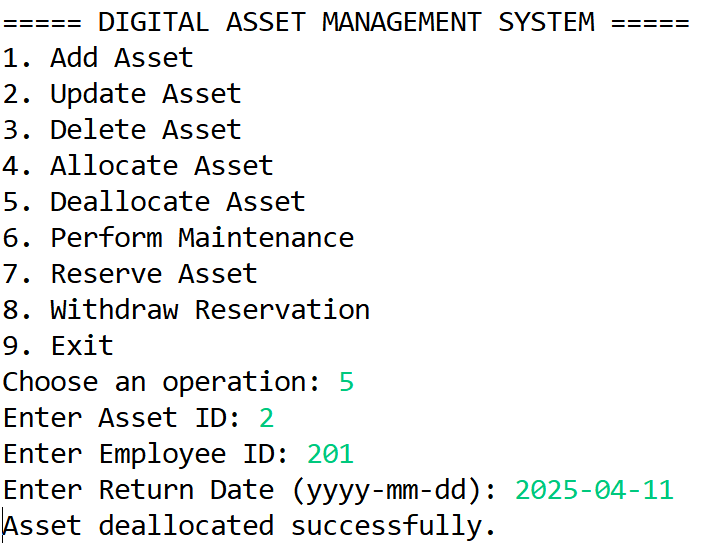
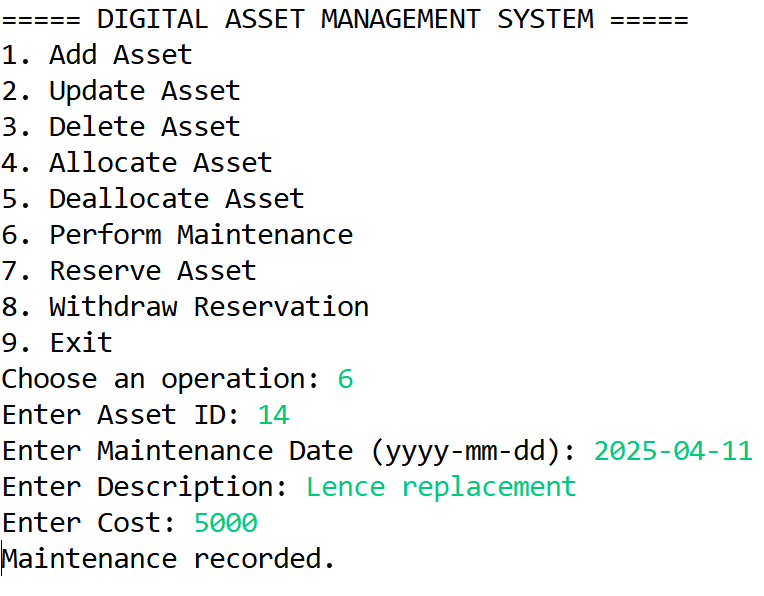
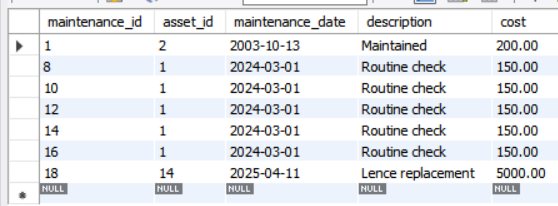
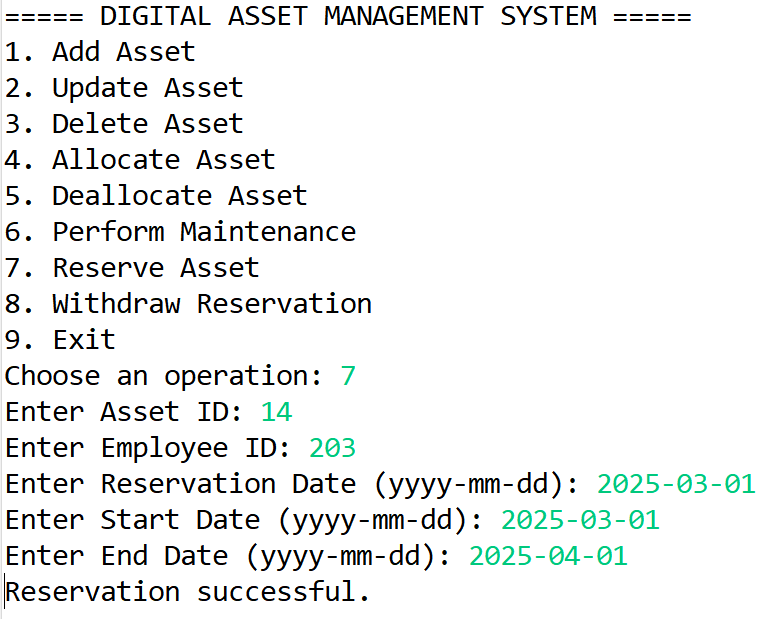
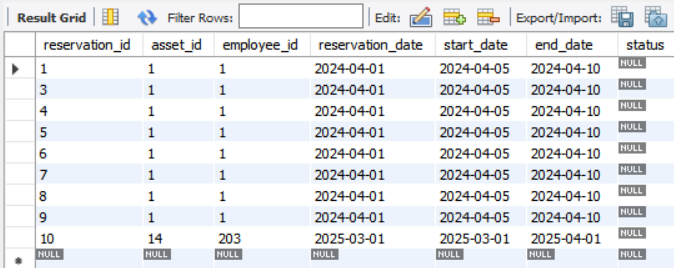
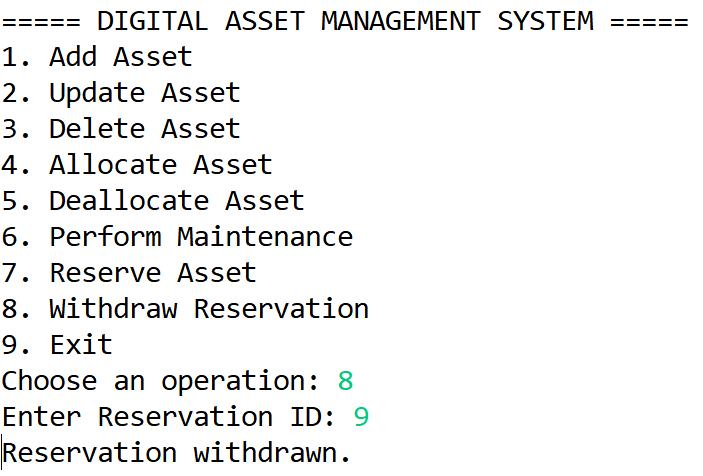
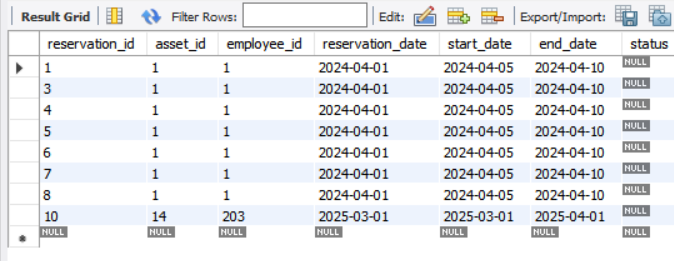
void testAssetNotMaintainException() {

assertThrows(AssetNotMaintainException.class, () -> {

service.performMaintenance(999, "2024-04-01", "Invalid Asset", 100);

});

}

}  
  
**Output Screenshot:**  
1. Add Asset:  
  
  
  
  
2. Update Asset:  
  
  
  
Before:  
  
  
  
  
After updation:  
  
  
  
3. Delete Asset:  
  
  
  
  
After deletion:  
  
  
4. Allocate Asset:  
  
  
  
  
  
5. Deallocate Asset:  
  
  
  
6. Perform Maintenance:  
  
  
  
  
  
7. Reserve Asset:  
  
  
  
  
  
8. Withdraw Reservation:  
  
  
  
  
9. Exit:  
