ACM 321 Project Report: Inventory Management System

# Team Information

* **Section:** 3
* **Team Name:** InvenTech
* **Team Members:**
  + Rauf Kutay Akyıldız - Frontend and GUI Developer
  + Eren Acar - Backend and Database Developer

# Project Overview

## Objective

The objective of our work is to design an Inventory Management System for a kitchenware (Züccaciye) shop. With this system the store can control its products, sales and suppliers better and in a more organised and efficient way. It will enable the store to keep up stock levels, process sales very quickly and maintain reliable inventory information. Furthermore, the system will facilitate to the users to handle orders and products, which would make the shop more efficient to run

## Store Type

Our team has chosen a kitchenware store for this project. We believe that such a store is ideal for an inventory management system as it has a wide selection of products, such as cookware, dishware, and kitchen tools. This range of goods enables us to introduce various inventory categories (e.g., utensils, appliance, decor). In addition, a kitchenware store must offer excellent goods flow management so that the product is all the time available to customers, and therefore, this project is a good example of an Inventory Management System as it can function within an actual store.

# Design and Architecture

## System Architecture

Our inventory management system includes three basic operators for small retail kitchenware: managing inventory, sales and customer relations. The system also includes three basic components:

### **2.1.1. Graphical User Interface (GUI):** Built using Java Swing for user-friendly interaction.

### **2.1.2. Database Management:** SQLite is a relational database, and for reliable data storage.

### **2.1.3**. **Business Logic Layer:** Patients at the minimum gateway of standard functionalities, including CRUD functionalities, inventory control and sales control.

## Class Diagram

The class diagram depicts the main classes representing the system functionalites i.e.

**Customer:** Manages customer data, including name, address, and city.

**Product:** Stores product information (code, description, category, and price).

**Category:** Represents product categories.

**Invoice:** Handles invoice generation and customer payments.

**ListOfItems:** Tracks the products and quantities associated with each invoice.

**Class Relationships:**

* Customer is associated with Invoice via a one-to-many relationship.
* Product relates to Category to organize items.
* The invoice is linked to ListOfItems (list of purchased goods or products) and number of items purchased.

## Database Schema

The database schema is organized in a way that it allows to stage, and thus correctly associate, data for materials in stock, customers and sales. Following is a schema description for the corresponding diagram.

### **2.3.1 CustomerTable:**

* **CustomerID (integer, primary key):** Unique identifier for each customer.
* **CustomerName (text):** Name of the customer.
* **CustomerAddress (text):** Address of the customer.
* **CustomerCity (text):** City where the customer resides.
* **CustomerCounty (text):** County of the customer.

### **2.3.2 CategoryTable:**

* **CategoryID (integer, primary key):** Unique identifier for each category.
* **CategoryName (text):** Name of the product category.

### **2.3.3 ProductTable:**

* **ProductCode (text, primary key):** Unique code for each product.
* **ProductDescription (text):** Description of the product.
* **CategoryName (text, foreign key):** Links the product to its category.
* **ProductPrice (decimal):** Price of the product.

### **2.3.4 InvoiceTable:**

**InvoiceID (integer, primary key):** Unique identifier for each invoice.

**CustomerID (integer, foreign key):** Links the invoice to the customer.

**Payment (decimal):** Total payment amount for the invoice.

### **2.3.5 ListOfItemsTable:**

* InvoiceID (integer, foreign key): Links the item list to an invoice.
* ProductCode (text, foreign key): Links the item to a product.
* Quantity (integer): Quantity of the product in the invoice.

Attached Diagram: The database schema is the same as the drawing presented and it describes both the object and relational aspects of the database model, such as the use of foreign key constraints. Every table is built so as to obtain the highest data retrieval speed, as well as so as to implement the business logic tier in an appropriate manner.

metin, ekran görüntüsü, multimedya yazılımı, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

# Features and Functionality

## Key Features

Our application has 3 pages and 9 panels. Due to this we able to use these features (you can see screenshots from 4.1):

* **Login, Logout and Sing Up Operations** through LoginPage.java and RegisterPage.java
* All CRUD and listing operations. (Category, Customer, Order and Product) through **xxx**AddPanel.java and **xxx**ListPanel.java)
* Some statistical data, through **DashboardPanel.java**

**Core Features**

**Inventory Management:**

* Adding, updating, and deleting products, categories, and customers.

**Sales Management:**

* Generating invoices and tracking customer orders.
* Processing payments and handling returns.

**Customer Management:**

* Storing customer details for personalized service and future reference.

**Data Import/Export: (not ready)**

* CSV file support for bulk data import and export.

## Customization

The system has been tailored to the specific needs of a kitchenware retail environment:

* Categories can be customized for kitchenware.
* GUI elements are customized with icons and color schemes that align with a kitchenware store’s branding.

# Application Walkthrough

## GUI Overview

Some screenshots and their descriptions:

ekran görüntüsü, metin, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Login and Registration Pages

metin, ekran görüntüsü, yazılım, web sitesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

First Panel After Logging In (Dashboard.java)

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Product Section (Adding, Deleting and Listing)

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Categories Section (Adding, Deleting and Listing)

metin, ekran görüntüsü, yazılım, web sitesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

Customers Section (Adding, Deleting and Listing)

\*Orders not ready.

Also, our classes are here:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

## Sample Workflow

A admin wants to add a new customer to the system. They follow these steps:

* The user navigates to the Customer Add Panel from the main menu.
* In the panel, the user fills out the fields: Name, Address, City, and County (e.g., “Eren Acar,” “Eren Acar’s Adress,” “Eren Acar’s City,” “Eren Acar’s County”).
* Clicking the Add button triggers an action that validates the input. If valid, the customer details are added to the table within the panel for review.

# Object-Oriented Principles

## Use of Classes and Objects

Some of the classes are CustomerAddPanel, OrderAddPanel, and ProductListPanel, which contain their own functions. For example, CustomerAddPanel controls the insertion of customer information using DefaultTableModel, which would dynamically change the displayed data.

## Inheritance and Polymorphism

* **Inheritance:** CategoryAddPanel and CategoryListPanel extend JPanel and reuse its layout management and event handling capabilities
* **Polymorphism:** Different panels (e.g., OrderAddPanel, ProductAddPanel) implement ActionListener for button actions. This allows the same addActionListener method to perform specific tasks based on the context​

## Interfaces and Abstract Classes

* **Abstract Classes:** Panels inherit common behavior (layout settings) from **JPanel**.
* **DAO Classes:** Interfaces for database operations such as **addCategory** and **deleteCategory** ensure standard behavior across different DAO implementations​.

# Database Integration

## Database Operations

The system uses CRUD operations:

* **Create:** **addCategory** adds a new category to the database.
* **Read:** **getAllCategories** retrieves all category entries.
* **Update:** **refreshTable** dynamically updates UI tables after CRUD operations.
* **Delete:** Removes records from tables using **DAO methods.**

## Sample Queries

* Adding a new category:
  + INSERT INTO CategoryTable (CategoryName) VALUES (?);
* Retrieving all categories:
  + SELECT \* FROM CategoryTable;
* Deleting a category:
  + DELETE FROM CategoryTable WHERE CategoryID = ?;

# File I/O

## Import/Export Functionality

The project supports **categories**, **customers**, **orders**, and **products** import/export operations in **CSV** format.

**File Format:**

* CSV format is used. Data is written in rows separated by commas. The first row contains the column headers.

**Import Process:**

* Files are selected using **JFileChooser.**
* BufferedReader reads the file line by line.
* Each line is split based on the CSV format, and the data is added to the table.
* For example, in the **CategoryListPanel** file, the categoryDAO method checks whether the category already exists and either updates it or inserts a new one​.
* Sample Import Code:
* private void importCategories() {
* JFileChooser fileChooser = new JFileChooser();
* fileChooser.setDialogTitle("Import Categories");
* int userSelection = fileChooser.showOpenDialog(this);
* if (userSelection == JFileChooser.APPROVE\_OPTION) {
* File file = fileChooser.getSelectedFile();
* try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
* String line = reader.readLine(); // Ignore the header line
* int updatedCount = 0;
* int insertedCount = 0;
* while ((line = reader.readLine()) != null) {
* String[] values = line.split(",");
* if (values.length >= 2) { // Ensure ID and Name are present
* int categoryId = Integer.parseInt(values[0]);
* String categoryName = values[1];
* if (categoryDAO.isCategoryExists(categoryName)) {
* categoryDAO.updateCategory(categoryId, categoryName);
* updatedCount++;
* } else {
* categoryDAO.addCategoryWithId(categoryId, categoryName);
* insertedCount++;
* }
* refreshTableRow(categoryId, categoryName);
* } else {
* JOptionPane.showMessageDialog(this, "Invalid data format in file.", "Error", JOptionPane.ERROR\_MESSAGE);
* }
* }

**Export Process:**

* **BufferedWriter** writes data in CSV format during export.
* Column names and data from the table are written to the file. For example, customer information is exported row by row​
* Sample Import Code:
* private void exportCategories() {
* JFileChooser fileChooser = new JFileChooser();
* fileChooser.setDialogTitle("Export Categories");
* int userSelection = fileChooser.showSaveDialog(this);
* if (userSelection == JFileChooser.APPROVE\_OPTION) {
* File file = fileChooser.getSelectedFile();
* try (PrintWriter writer = new PrintWriter(file)) {
* writer.println("CategoryID,CategoryName");
* for (int i = 0; i < tableModel.getRowCount(); i++) {
* writer.println(tableModel.getValueAt(i, 0) + "," + tableModel.getValueAt(i, 1));
* }

## Error Handling

**Types of Errors:**

* File read/write errors (IOException).
* Data format errors (NumberFormatException).
* Database connection errors (SQLException).

**Error Prevention:**

* When a file cannot be found or a format error occurs, an error message is shown using JOptionPane.
* Example Usage:
* public void actionPerformed(ActionEvent e) {
* String email = emailTextField.getText();
* String password = new String(passwordField.getPassword());
* // Check if the user exists in the database
* boolean success = UserDAO.loginUser(email, password);
* if (success) {
* // Get user information from the database
* String[] userInfo = UserDAO.getUserInfo(email); // Name and surname
* if (userInfo != null) {
* MainPage mainPage = new MainPage(userInfo[0], userInfo[1]); // Send name and surname to the main page
* mainPage.setVisible(true);
* dispose(); // Close the login
* }
* } else {
* JOptionPane.showMessageDialog(null, "Invalid email or password.");
* }
* }
* });

# Challenges and Solutions

**Database Connection:**

* **Challenge:** Connection timeout issues occurred when loading data.
* **Solution:** Connections were refreshed, and try-with-resources was used to ensure proper resource management.

**CSV Format Compatibility:**

* **Challenge:** Different CSV files had varying column counts.
* **Solution:** A validation mechanism was added to ensure correct formatting and display an error message if columns were missing.

**GUI Issues:**

* **Challenge:** Some buttons triggered multiple actions due to double-click events.
* **Solution:** consume() method was added to ignore double-clicks​

# Future Improvements

**Database Backup:** An automatic data backup feature can be added.

**Support for Multiple File Formats:** Support for JSON and XML import/export can be implemented.

**Reporting Module:** A feature for graphical reports and statistics can be included.

**User Access Control:** Role-based access control for different user types can be introduced.

# Conclusion

This project developed a database-driven application for managing categories, customers, orders, and products. File I/O operations facilitated data import/export and the design pattern of the DAO was adopted to guarantee a modular structure. In the course of this process, the team acquired experience in graphical user interface (GUI) design, file management, error handling and SQL integration.

# Appendix

## User Manual

**Login:**

* Open the application. The login screen (LoginPage.java) prompts you to enter your username and password.

**Register:**

* If you are a new user, use the RegisterPage.java to create an account.

**Dashboard (MainPage.java):**

* After successful login, you are redirected to the main dashboard.
* From here, you can navigate to various panels:
  + **Categories:** Add, edit, or list categories.
  + **Customers:** Add, edit, or list customer information.
  + **Orders:** View invoices and payment information.
  + **Products:** Manage product information.

**Panels for CRUD Operations:**

* Each section (categories, customers, orders, and products) has a dedicated panel for:
  + **Add Panel:** Allows adding new entries (e.g., CategoryAddPanel.java, CustomerAddPanel.java).
  + **Edit Panel:** Enables updating existing entries (e.g., OrderEditPanel.java).
  + **List Panel:** Displays a table of entries with import/export options (e.g., CategoryListPanel.java).

**Import Data:**

* Click the "Import" button in list panels (CategoryListPanel.java, ProductListPanel.java, etc.) to upload a CSV file.
* Data is imported and added to the database after validation.

**Export Data:**

* Click the "Export" button to save the table data as a CSV file.
* The file will contain columns such as Product Code, Category Name, etc.

**Error Handling:**

* If an invalid file is selected, an error message will be shown.
* In case of a database connection issue, a dialog box displays the error details.

## References

* <https://www.baeldung.com/sha-256-hashing-java>
* <https://www.geeksforgeeks.org/java-swing-jcombobox-examples/>
* <https://stackoverflow.com/questions/67073072/java-simple-log-in-register-application>
* <https://docs.oracle.com/javase/tutorial/jdbc/basics/jdbcswing.html>
* <https://www.quora.com/How-do-I-connect-a-database-with-Java-GUI>
* <https://www.geeksforgeeks.org/data-access-object-pattern/>
* <https://www.youtube.com/watch?v=wa3rzZdkVrE>
* <https://www.geeksforgeeks.org/mouselistener-mousemotionlistener-java/>
* <https://www.youtube.com/watch?v=640M9E26sZg>
* <https://coderanch.com/t/342905/java/JOptionPane-JFrame>
* <https://www.geeksforgeeks.org/java-joptionpane/>
* <https://docs.oracle.com/javase/8/docs/api/javax/swing/table/DefaultTableModel.html>