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

[Describe the file formats used and how data import/export is implemented.]

## Error Handling

[Discuss how your application handles errors during file operations.]

# Challenges and Solutions

[Describe the challenges your team faced during the project and how you addressed them.]

# Future Improvements

[List potential improvements or additional features that could be added to the project.]

# Conclusion

[Summarize your experience working on the project and the skills your team developed.]

# Appendix

## User Manual

[Provide a detailed user manual, including installation instructions and usage guidelines.]

## References

[List any references, tools, or resources used during the project.]