
# İstanbul Aydın University

**Project Name: Inventory System**

**Team Members: Efe Yiğit Küçük(B2305.090210)**
**Elif Canbeyli(B2305.090125)**
**Perihan Çelikoğlu(B2305.090097)**
**Umay İleri(B2305.090125)**

**Course Name: Introduction to Software Engineering**

# Project Overview

## Project Objective:

**The purpose of the this project is to use our fundamental software devolopment skills to create a user-friendly and functional Inventory System. The system aims to assist small businesses or individuals in managing their inventory. This project has provided an opportunity to learn and apply the C# programming language and object-oriented programming principles, while also allowing us to experince the software development process.**

## Problem Definition:

**Inventory management in small businesses is often carried out using notebook or simple spreadsheets.These methods can lead to issues such as incorrect data entry, data mismanagement, or loss of information.For instance:**

- **Difficulty in quickly identifying which products are in stock or running low,**
- **Inability to track stock inflow and outflow systematically,**
- **Lack of easy access to information such as the dates when specific products were added.**

**These problems result in disruptions to business processes and loss of time. To address these issues, we aimed to design an Inventory System suitable for our skill level . The system allows users to perform essential tasks such as adding, deleting, updating products , and viewing stock statuses with ease.**

# Solution Approach

•In this project, an Inventory System has been developed using the fundamental principles of object-oriented programming in C# to solve the identified problems. Our solution approach consists of the following steps:

## 1. Analysis and Design:

Initially, the inventory management needs of small businesses were analyzed. Based on this, the system's functionalities and user requirements were identified, and a design schema was created.

## 2. Modular Structure:

The system provides core functionalities such as adding, deleting, updating products, and displaying stock status in a modular structure. This allows each function to be developed and managed independently.

## 3. Database Usage:

A database structure was designed to store inventory data, and product information (e.g., product name, quantity ) is stored in this database. Communication between the database and the system has been optimized to ensure error-free data entry and reliable stock tracking.

## 4. User Interface:

The system is designed with a simple and user-friendly interface that allows users to easily access and quickly perform their tasks.

## 5. Testing and Debugging:

The developed system was tested with various scenarios, errors were fixed, and performance was improved.

This solution approach provides a hands-on experience not only in learning fundamental software engineering principles but also in solving real-world problems

# User Story

**User Profile:**
**Mr. Ahmed is the owner of a small stationery shop. He has to manage the purchase and sale of dozens of products daily. He currently keeps track of his inventory using notebooks and simple spreadsheets. However, this method falls short in tracking stock status and product history. Mr. Ahmed needs a more organized and reliable solution.**

**Problems:**
- **He cannot quickly determine which products are in stock or running low.**
- **It is difficult to track when specific products were added.**
- **Manual entries in notebooks are prone to errors or can be lost.**
- **Keeping track of stock inflows and outflows is time-consuming and complicated.**

**Needs and Requests:**
- **A system where products can be easily added and updated.**
- **A quick way to view stock levels.**
- **Notifications from the system about low or depleted stock.**
- **The ability to track stock inflow and outflow operations.**
- **A user-friendly and intuitive interface**

**Solution:**
**To meet Mr. Ahmed's needs, a simple and effective Inventory System was developed. With this system, Mr. Ahmed can easily add, edit, and monitor his stock status from a single screen. Additionally, the system provides alerts for products that are running low, making his business operations more efficient.**

**Outcome:**
**This system significantly reduces the time and effort Mr. Ahmed spends on inventory management, minimizes errors, and helps him manage his business more efficiently**

# System Design

## 🛠️System Architecture

•This document defines the system architecture of the Inventory system project. The project is developed using C# and SQLite and follows a Three-Tier Architecture model.

## 📐1.Architectural Approach: Three-Tier Architecture

The project is divided into three layers:

1. Presentation Layer – User interface and user interactions
2. Business Logic Layer – Application rules and data processing
3. Data Access Layer – Database connections and data managemen

## 📊 2. Layer Descriptions

### 🖥️ 1. Presentation Layer

• Responsibility: Provides a visual interface to the user and collects inputs.
• Technologies: Windows Forms (C#)
• Components:
• Login Screen: User login screen (Correct/incorrect login validation)
• Homepage: Main menu, navigation, and exit button (Exit confirmation screen)
• Manage Items: Product add, edit, and delete screen
• List Items: Product listing screen

Tasks:

• Pass data received from the user to the Business Logic Layer (BLL)
• Display processing results to the use

## ⚙️ 2. Business Logic Layer (BLL)

• Responsibility: Manages the application's business rules and logic.
• Technologies: C# (Backend coding)
• Components:
• User Authentication: Username and password validation
• Product Management: Add, edit, and delete product operations
• Data Validation: Checking for empty fields and verifying data types
Tasks:
• Validate and process data received from the Presentation Layer
• Forward necessary data requests to the Data Access Layer
• Process results from the Data Access Layer and send them back to the Presentation Laye

## 🗃️ 3. Data Access Layer (DAL)

Responsibility: Manages the database connection and performs data access operations.
• Technologies: SQLite
• Components:
• Database Connection: Management of database connections
• CRUD Operations: Create (Add), Read (View), Update (Edit), Delete (Remove)
• Table Management: Product table, User table
Tasks:
• Manage database connections
• Process data requests from the Business Logic Layer
• Return results of database operations back to the Business Logic Layer

## 🔄 3. Communication Flow Between Layers

**1. User Action: The user logs in or initiates an action via the Presentation Layer.**

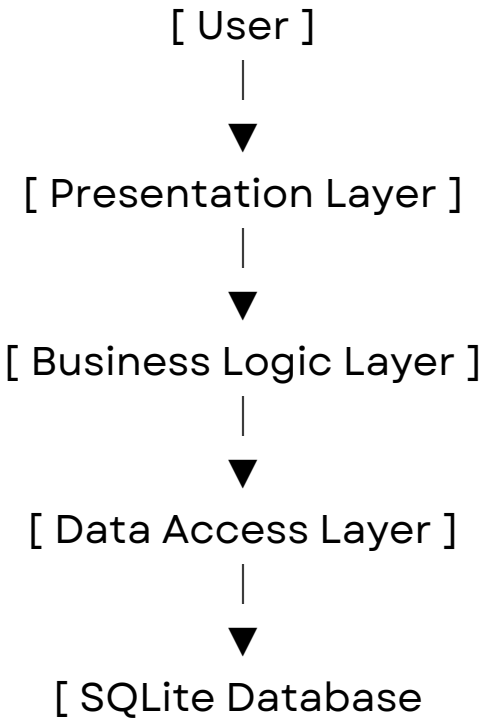
**2. Presentation → Business Logic: Data received from the user is sent to the Business Logic Layer.**

**3. Business Logic → Data Access: The Business Logic Layer calls the Data Access Layer for data operations.**

**4. Data Access → Business Logic: Results from the database are returned to the Business Logic Layer.**

**5. Business Logic → Presentation: Processed data is passed to the Presentation Layer and shown to the user**

## 📝 4. Data Flow Diagram



## 🛠️ 5. Database Design

### 📊 Tables:

#### 1.Users

| Column | Data Type | Description |
|---|---|---|
| UserID | INTEGER | Primary Key |
| Username | TEXT | User Name |
| Password | TEXT | User Password |

#### 2.Items

| Column | Data Type | Description |
|---|---|---|
| ItemID | INTEGER | Primary Key |
| Name | TEXT | Product Name |
| Category | TEXT | Product Category |
| Quantity | INTEGER | Product Quantity |
| Unit | TEXT | Unit Type |
| Description | TEXT | Product Description |

## ✅ 6. Technologies and Tool

| Layer | Technologies |
|---|---|
| Presentation Layer | Windows Forms (C#) |
| Business Logic Layer | C#,.NET |
| Data Access Layer | SQLite |
| Development Environment | Visual Studio |

## 🌟 7. Advantages

- **Modularity: Each layer is independent and can be easily updated.**
- **Portability: The database can run on different devices thanks to SQLite.**
- **Ease of Maintenance: Issues can be easily identified and fixed.**
- **Reduced Code Redundancy: A reusable code structure is ensured**

## 🚀 8. Future Enhancements

- **API Integration: Expanding the application with REST API**
- **Reporting Module: Creating user-friendly reports**
- **Authorization System: User management based on roles and permission**

# Development process

## Code review processes

First, the interface is created on the login screen and the password and username are entered. Sends a feedback message based on correct or incorrect entry

The interface is created on the Homepage screen. When we click on the Manage Items and List Stocks images, it directs us to those pages. Exit is available from the application.

List stocks interface is created. It checks the connection with Datebase. It connects the Items datebase to its own interface. It fills the table in the interface, and if it is not connected, it shows an error message as feedback. When you press the Cancel button, it returns to the main screen.

Manage items interface is created. It adds temporary texts to empty textboxes and loads the data from the datebase into the interface. When you press the Cancel button, you return to the main menu. When you press the save button, it saves the texts in the textbook to the datebase and returns the textbooks to their old appearance. It refreshes the view of the datebase by refreshing the screen. When you click on the Delete button, it deletes the data from the datebase and reflects the selected data to the textbooks. When you click on the Edit button, the changes are transferred to the table.

Sql Variables is connected to the inventory database with Sqlite. If the table does not exist, a table is created and provides feedback about the situation.

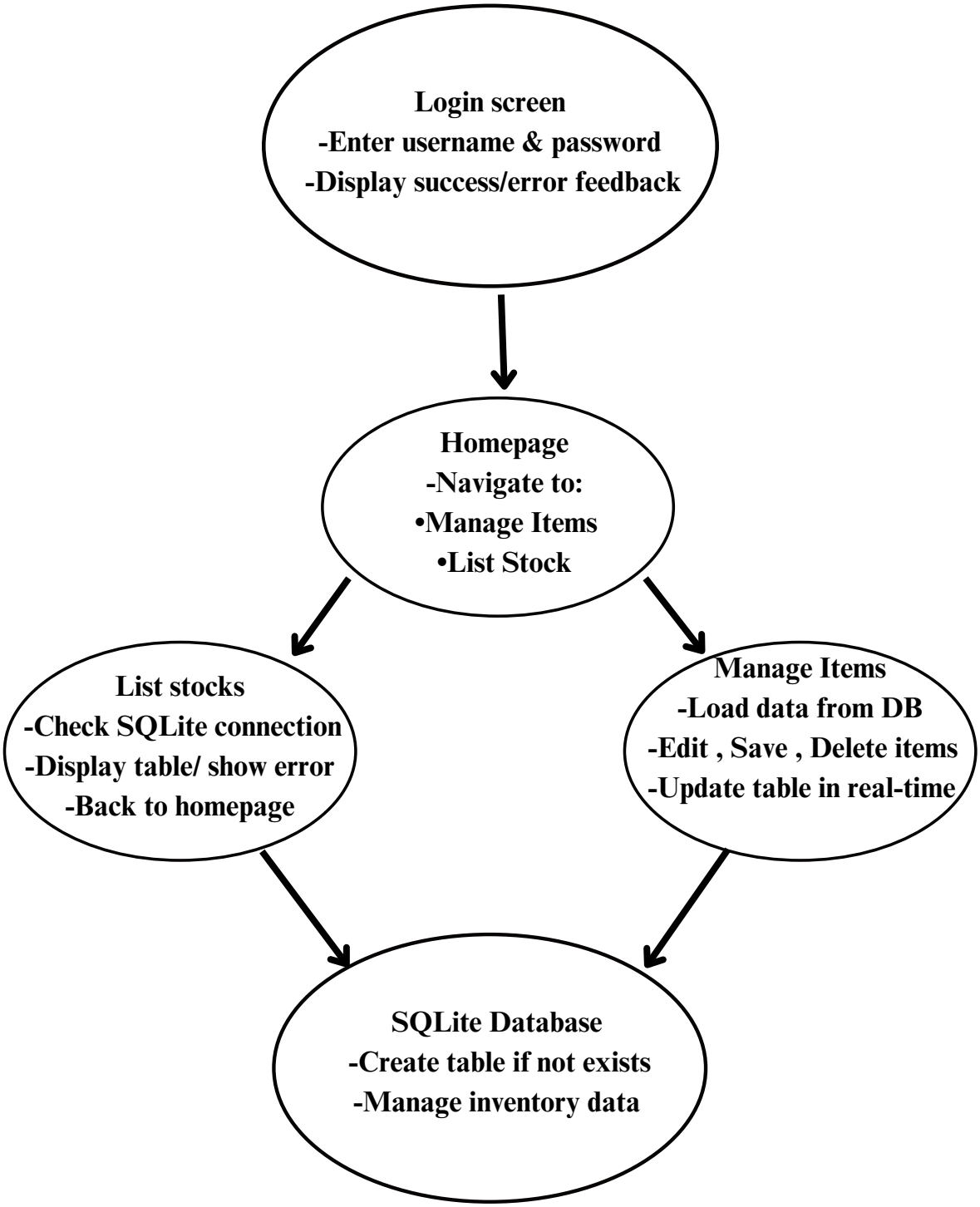
It calls the database creation method in the program and initializes the settings, themes, general configurations, and starts the program by reflecting the login interface to the screen.

## Github usage and commits

**•My teammates opened a program file from Github and sent an invitation link to three other people to actively use this file.**

https://github.com/ileriumay/InventorySystem/commits/master/

## Flow diagram



# Features of the project

## 1. User Authentication and Login

- Includes fields for entering a username and password.
- Displays a success message for correct login credentials and an error message for incorrect ones

## 2. Homepage

- Designed as the application's control panel.
- Includes quick access buttons for "Manage Items" and "List Stocks" pages.
- Allows the user to log out from the homepage.

## 3. Stock Listing

- Connects to the SQLite database and lists stock data.
- Displays an error message if the database connection fails.
- Users can return to the homepage

## 4. Item Management

- Provides empty text fields for entering new item data.
- Loads and displays existing item data from the database
- Functions:
- Save: Adds the entered item data to the database.
- Delete: Removes the selected item record from the database.
- Edit: Updates the selected item data and saves the changes
- Updates the table in real time.

## 5. Error Messages and User Feedback

- Provides clear and concise error messages if incorrect data is entered or a database connection issue occurs

## 6. SQLite Database Management

- Integrated with SQLite for storing stock and item data.
- Automatically creates the required table if it doesn't exist and informs the user

## 7. Modern and User-Friendly Interface

- Features a simple and intuitive design, prioritizing user experience.
- Includes a basic navigation system for easy transitions between pages

# Lessons learned

## 1. Planning and Organization

• The importance of setting clear goals and making step-by-step plans at the beginning of the project.

• Understanding that poor or incomplete planning can lead to bigger problems later on.

## 2. Communication and Teamwork

• The importance of clear communication and how exchanging ideas contributes to more efficient progress.

• Realizing that mutual feedback is necessary for achieving higher-quality results

## 3. Identifying Problems Early

• Recognizing issues early in the project makes the problem-solving process easier.

• Understanding the critical role of testing and feedback mechanisms during the process

## 4. Technical Skills

• Learning what works and what doesn't when using a specific technology or method.

• Discovering better coding standards or more effective tools.

## 5. Time Management

• Understanding that using time effectively is crucial to successfully completing the project.

• The importance of allocating buffer time for unexpected situations

## 6. Documentation and Learning

• Realizing that documenting experiences after the project helps in finding quick solutions when facing similar problems in the future.

• Noticing the advantages of thinking more clearly while documenting the process.