**Structure of Programming**

Created by: L.H.G. Yasiru Prabodha

CMU ID: st20330308

ICBT ID: CL/HDNET/CMU/51/14

Course: HD in Network Technology and Cyber Security

Lecturer: Ms. Sadali

Submit Date: 2024/03/13

# Acknowledgment

I would like to convey my sincerest gratitude to my Fundamentals in Programming lecturer, Miss Sadali, for her unselfish guidance and mentorship throughout the creation of the FURNITURE VILLEGE Furniture Ordering System. Her work ethics, patience, and commitment were key in helping me navigate the project challenges and expanding my knowledge on programming principles.

Miss Sadali's unambiguous explanations of modularization, data structures, and error handling methods gave me the confidence to create and design the system. Her positive criticism of debug exercises and her insistence on having clean and efficient code were significant in making the program functional and user-friendly. I am particularly thankful for her pushing me beyond the box and solving problems methodically, which not only enhanced this project but also enhanced my overall programming abilities.

I would also appreciate her for creating a learning atmosphere that was conducive and saw failures as opportunities to learn and questions as freely welcomed. Her enthusiasm in teaching and ensuring students' success made me more willing to reach out and provide a system that meets actual needs. Miss Sadali's mentorship enabled this project to become a possibility, and I am grateful to have been under her mentorship to learn.

With sincere appreciation,

Yasiru Prabodha

2025/03/13

# Executive summary

FURNITURE VILLEGE, a lifestyle specialty store, sought to automate its manual furniture ordering processes to reduce employee workload and enhance customer satisfaction. This project developed **a C++ menu-driven console application** addressing core requirements like secure login, furniture management, order processing, and user-friendly navigation. By replacing manual tasks with automation, the system streamlines operations, minimizes errors, and ensures real-time inventory tracking, directly aligning with the store’s goals of efficiency and improved service quality.

The system’s **key features** include a secure login interface to restrict unauthorized access, tools to add and list furniture details, and a search function for quick item retrieval. Employees can input new furniture (ID, name, category, price, quantity) while customers place orders with automatic stock validation. Data is stored persistently in text files, ensuring reliability between sessions. Input validations, such as numeric checks for prices and quantities, prevent incorrect entries, while a structured menu guides users seamlessly through tasks like viewing inventory or accessing the help section.

**Technically**, the solution employs modular functions (e.g., login(), placeOrder()) and structs to organize data, enhancing code maintainability. Files like furniture.txt and orders.txt store records, enabling easy updates and retrieval. A loop-driven menu with switch-case logic allows intuitive navigation. Rigorous testing validated scenarios like login failures, order placement with insufficient stock, and edge cases, ensuring robustness. Feedback from users highlighted the system’s ease of use and reduced operational friction.

**Outcomes** include reduced manual workload, accurate inventory management, and faster customer service. Employees benefit from automated workflows, while customers enjoy reliable stock information and smoother transactions. Future enhancements could integrate a graphical interface or expand to online sales. This project underscores how targeted automation and user-centric design can transform manual processes into scalable, efficient systems, positioning FURNITURE VILLEGE for sustained growth in a competitive market.

# Introduction

As good space management and customer-focused services occupy the top spot in today's day and age, FURNITURE VILLEGE is now a lifestyle specialty retail leader with creative solutions for furniture in small spaces. But runaway growth and increasing sales have added to operational headaches, especially due to dependence on manual processes in areas of order management, inventory monitoring, and customer care. These inefficiencies can cause burnout among employees, order error fulfillment, and dilatory follow-up on customer needs, all possibly damaging the store's image for quality and reliability.

In order to overcome such challenges, the proposed project presents an automated furniture ordering system designed in C++. The system will automate key functions—such as adding furniture, ordering, and inventory management—while also improving user experience through friendly interfaces and good error handling. By eliminating manual operations, the solution minimizes administrative workloads on staff, eliminates human errors, and improves service delivery, hence making FURNITURE VILLEGE competitive in a dynamic environment.

The system focuses on usability and data integrity. Secure login authentication, real-time inventory tracking, and search capabilities provide employees with the ability to accomplish tasks easily. Consistent visibility of stock and easy ordering are benefits to customers. Files are utilized to hold records to make them accessible during sessions. Modular programming methods enable maintenance to be simpler, and input validation and concise menus direct users through minimized learning.

This project not only fills the available operational gaps but also provides the groundwork for eventual expansion. Automating the most important processes enables FURNITURE VILLEGE to engage in strategic development, customer satisfaction, and line extension. Subsequent sections provide an overview of the system design, implementation, and testing that demonstrate how technology can convert cumbersome manual procedures to flexible, flawless workflows that bring businesses to their pinnacle of achievement.

# TASK 01

## 1.1 Introduction

### 1.1.1 Purpose

The purpose of this document is to define the **Software Requirements Specification (SRS)** for the **FURNITURE VILLEGE** Furniture Ordering System. It provides an overview of the system’s objectives, functionalities, and constraints. This SRS will guide the development team in creating a reliable, user-friendly, and efficient software solution that automates the ordering process and related activities.

### 1.1.2 Scope

The scope of this system includes:

* User Authentication (login/logout)
* Add Furniture (add new furniture items to inventory)
* List Available Furniture (view all furniture items in stock)
* Search Specific Furniture (locate furniture by ID, name, or category)
* Place Orders (create orders for customers, update inventory)
* Help (display help or usage information)
* Exit (quit the application)

These functionalities are designed to reduce employee workload, streamline the ordering process, and enhance customer satisfaction. (Fretheim and Deschene, 2023)

### 1.1.3 Intended Audience and Reading Suggestions

* Management: Understand the capabilities, constraints, and business value of the system.
* System Developers: Refer to detailed requirements for implementation.
* Testers: Use the requirements to develop test cases and ensure system accuracy.
* End Users: Understand the main features and usage. (Hammer et al., 2020)

### 1.1.4 System Overview

FURNITURE VILLEGE aims to automate critical parts of its operations to handle the increased demand. By creating a menu-driven program, employees and management can efficiently handle furniture inventory and orders.

## 1.2 Overall Description

### 1.2.1 Product Perspective

Currently, operations (inventory management, order processing, etc.) are handled manually, which is time-consuming and prone to errors. This new system will replace the manual processes with automated functionalities, leveraging a simple menu-based interface and a suitable data storage mechanism (e.g., text files, CSV files, or a database). (makingofsoftware, 2024)

### 1.2.2 Product Features Summary

1. **Authentication**

* Secure login to the system.
* Different access levels (e.g., admin vs. standard user) can be assumed if needed.

1. **Inventory Management**

* Add furniture to the system (furniture ID, name, category, price, quantity).
* Update existing furniture (optional extension).
* Delete furniture (optional extension).
* List all available furniture.
* Search furniture by ID, name, or category.

1. **Order Management**

* Place new orders (capture customer details, furniture items, quantity).
* Update order status (optional extension).
* View all orders (optional extension).

1. **Help**

* Display usage instructions for each menu item.

1. **Exit**

* Terminate the system safely.

### 1.2.3 User Classes and Characteristics

* **Admin (Manager)**: Has the ability to add new furniture, remove furniture, and see detailed reports. GeeksforGeeks (2017)
* **Sales Staff**: Can place orders, search furniture, list furniture, and view order status.

### 1.2.4 Operating Environment

* **Software**: The system can be implemented in any high-level programming language (e.g., Java, C++, Python) and can use text files or a relational database (MySQL, SQLite, etc.) for data storage.
* **Hardware**: Standard desktop/laptop with a minimum recommended configuration (4 GB RAM, 1 GHz CPU, etc.). windows required. (Gomes, Rabines and Bustamante, 2016)

### 1.2.5 Constraints

* The system should be menu-driven (console or GUI).
* Must handle concurrency if multiple employees use the system (optional advanced requirement).
* Must provide secure authentication.

### 1.2.6 Assumptions and Dependencies

* All furniture items have a unique identifier.
* Data validation will be handled by the system (e.g., correct format for prices, non-empty fields, etc.).
* Network connectivity for multi-user access is assumed (if implemented in a shared environment).

## 1.3 System Features

### 1.3.1 Functional Requirements

1. **User Authentication**

* **FR1.1**: The system shall prompt for a username and password upon startup.
* **FR1.2**: The system shall verify credentials against stored user data. (Larsen, Checko and Christiansen, 2019)
* **FR1.3**: The system shall display an error message for invalid credentials.

1. **Add Furniture**

* **FR2.1**: The system shall allow the user to add new furniture by specifying an ID, name, category, price, and quantity.
* **FR2.2**: The system shall validate that the furniture ID is unique.
* **FR2.3**: The system shall confirm successful addition or display an error message if the process fails. (Chen et al., 2022)

1. **List Available Furniture**

* **FR3.1**: The system shall retrieve and display all furniture items currently in inventory.
* **FR3.2**: The system shall display relevant details (ID, name, category, price, quantity).

1. **Search Specific Furniture**

* **FR4.1**: The system shall allow searching by furniture ID, name, or category.
* **FR4.2**: The system shall display search results if the furniture item(s) exist.
* **FR4.3**: The system shall display a message if no matching furniture is found.

1. **Place Orders**

* **FR5.1**: The system shall allow the user to create an order by specifying customer details, furniture items, and quantities.
* **FR5.2**: The system shall check if sufficient stock is available before confirming the order.
* **FR5.3**: The system shall deduct the ordered quantity from the inventory upon successful order placement.
* **FR5.4**: The system shall display order confirmation with a unique order ID.

1. **Help**

* **FR6.1**: The system shall display a help menu that explains how to use each function.

1. **Exit**

* **FR7.1**: The system shall allow the user to exit the application, ensuring any in-memory data is saved to the persistent storage.

### 1.3.2 Non-Functional Requirements

1. **Performance Requirements**

* **NFR1.1**: The system should respond to user actions (e.g., listing furniture) within 2 seconds.
* **NFR1.2**: The system should handle up to 1000 furniture items efficiently in typical operations.

1. **Reliability & Availability**

* **NFR2.1**: The system should back up data (e.g., after each successful transaction) to prevent data loss.
* **NFR2.2**: The system should handle minor data corruption by restoring from the latest backup. (Shooman, 2003)

1. **Usability**

* **NFR3.1**: The user interface should be menu-driven with clear prompts.
* **NFR3.2**: Error messages should be descriptive and guide users to correct input.

1. **Security**

* **NFR4.1**: User passwords should not be displayed in plain text.
* **NFR4.2**: Only authenticated users can perform inventory modifications.

1. **Maintainability**

* **NFR5.1**: Code should follow modular design principles for easy updates.
* **NFR5.2**: Documentation (comments, user guides) should be provided.

## 1.4. System Modules (High-Level)

1. **Authentication Module**

* Handles login functionality.
* Validates user credentials.

1. **Inventory Management Module**

* Add furniture (add item details, validate, store in database or file).
* Search furniture (by ID, name, or category).
* List furniture (fetch all items from storage).

1. **Ordering Module**

* Place orders (capture order details, update inventory).
* Potentially view/update order status (optional extension).

1. **Help Module**

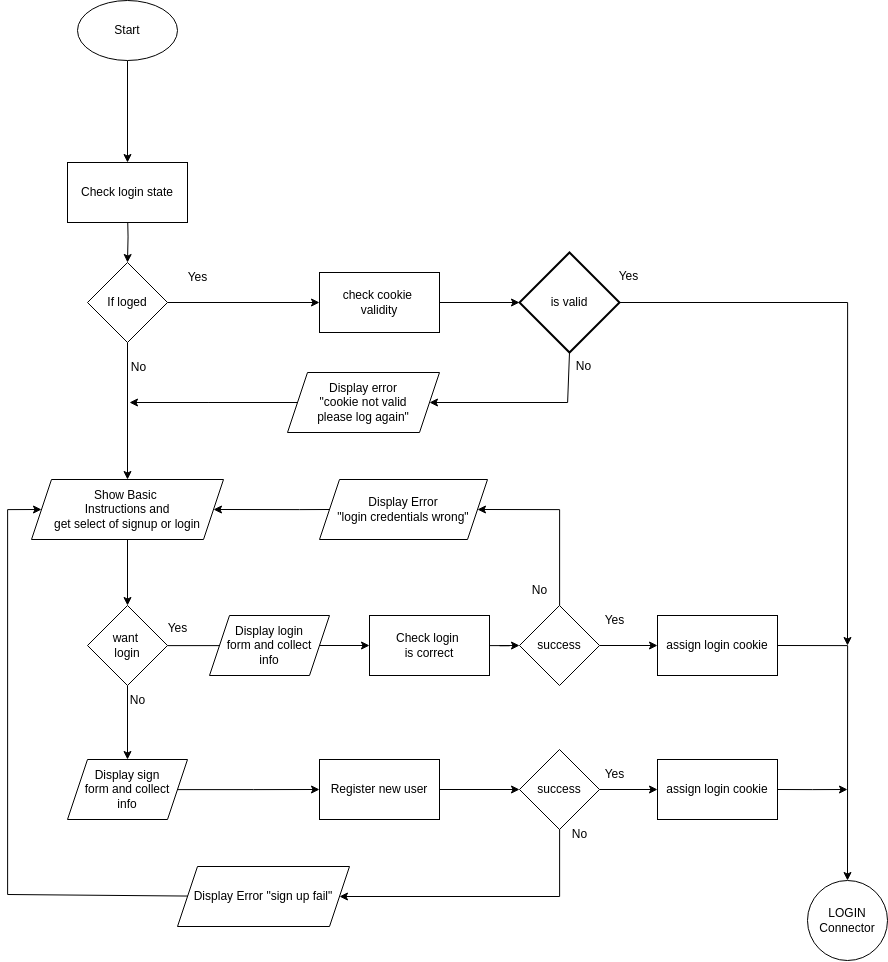
* Display usage information.

1. **Main Menu / Controller**

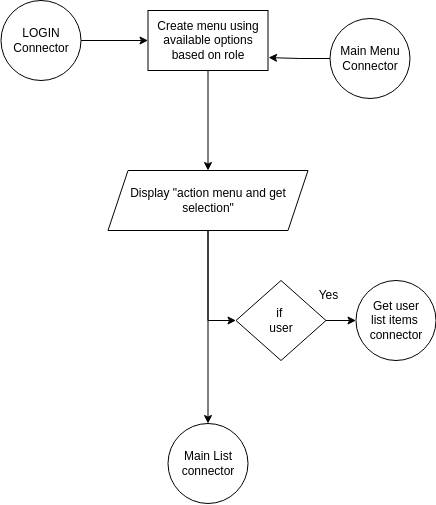
* Coordinates the calls to each module based on user choice.
* Handles navigation and exit functionality. (Staib, Dörrhöfer and Rosenthal, 2013)

## 1.5. Flow Charts for Core Functions

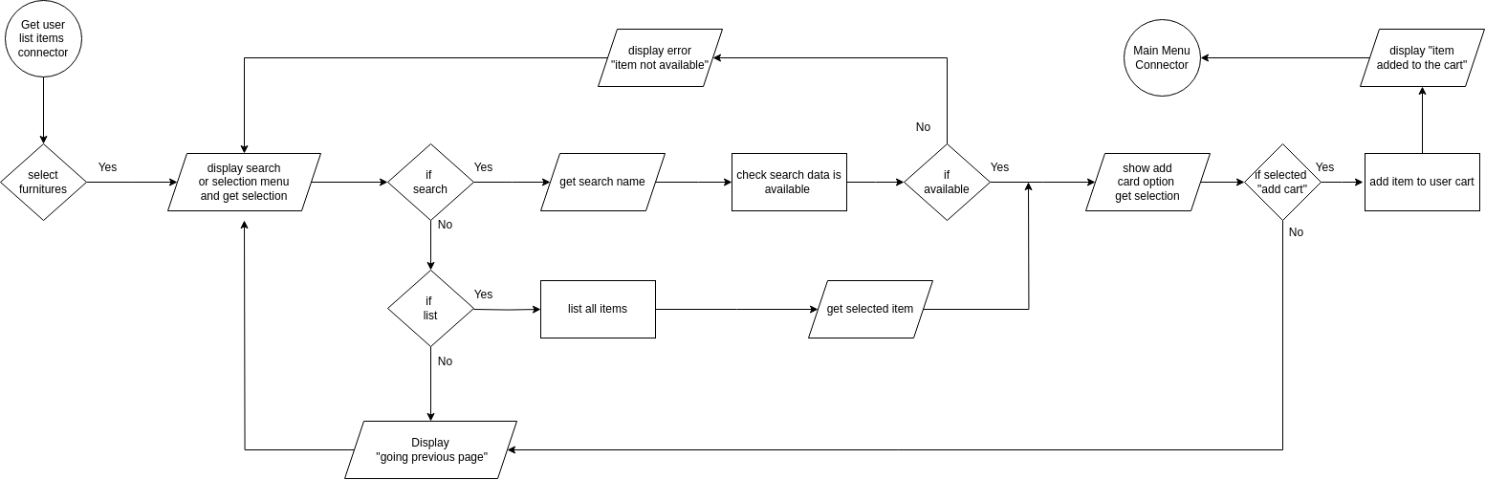
Login system



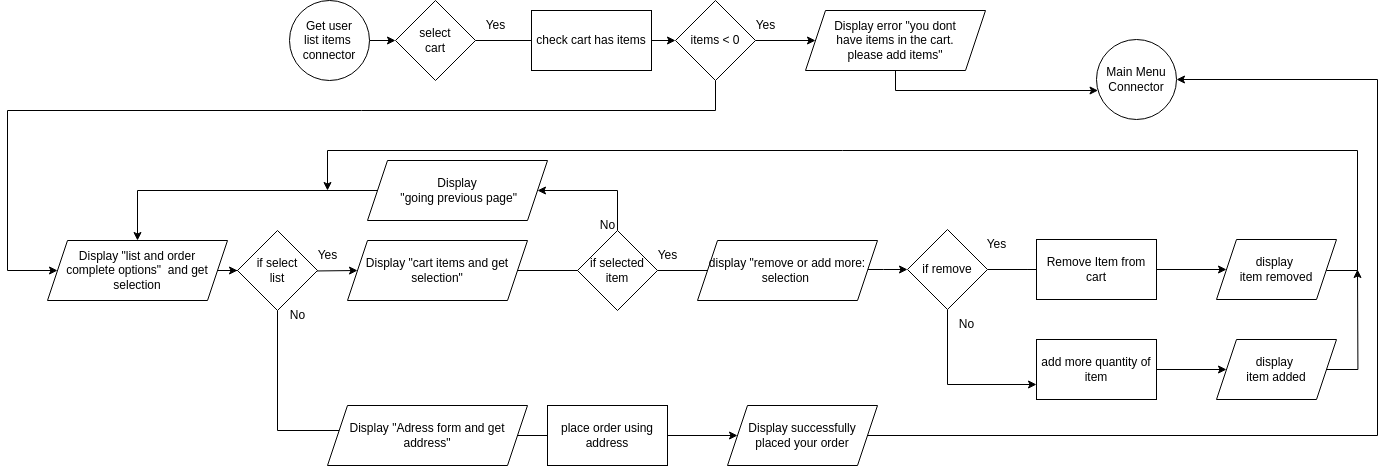
Selection menu with connectors



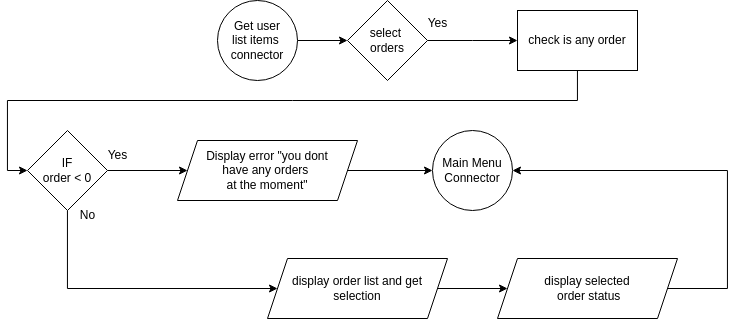
Furnitures menu



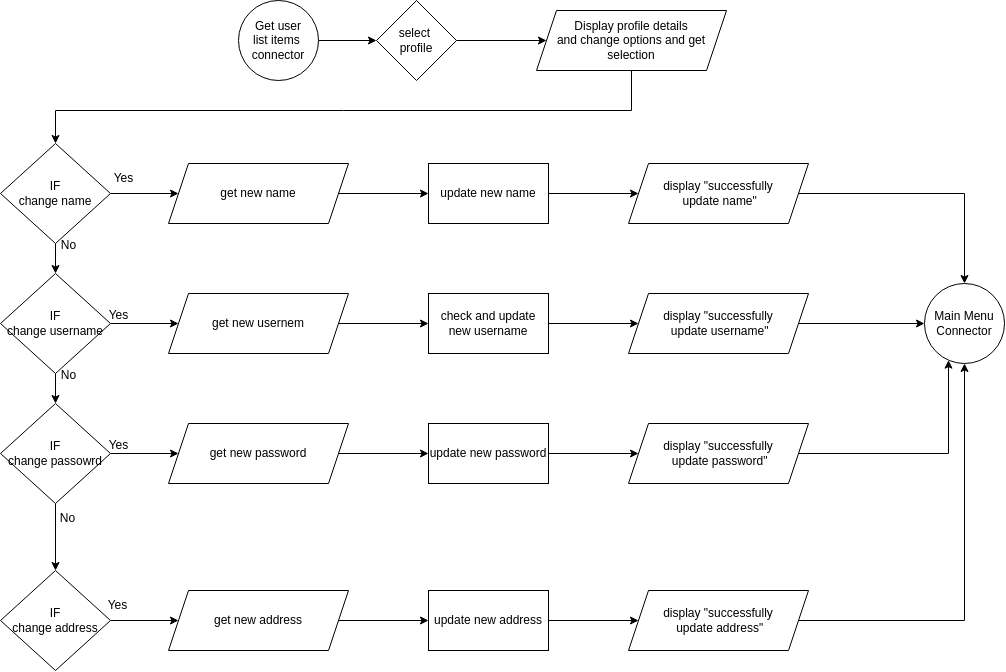
Cart



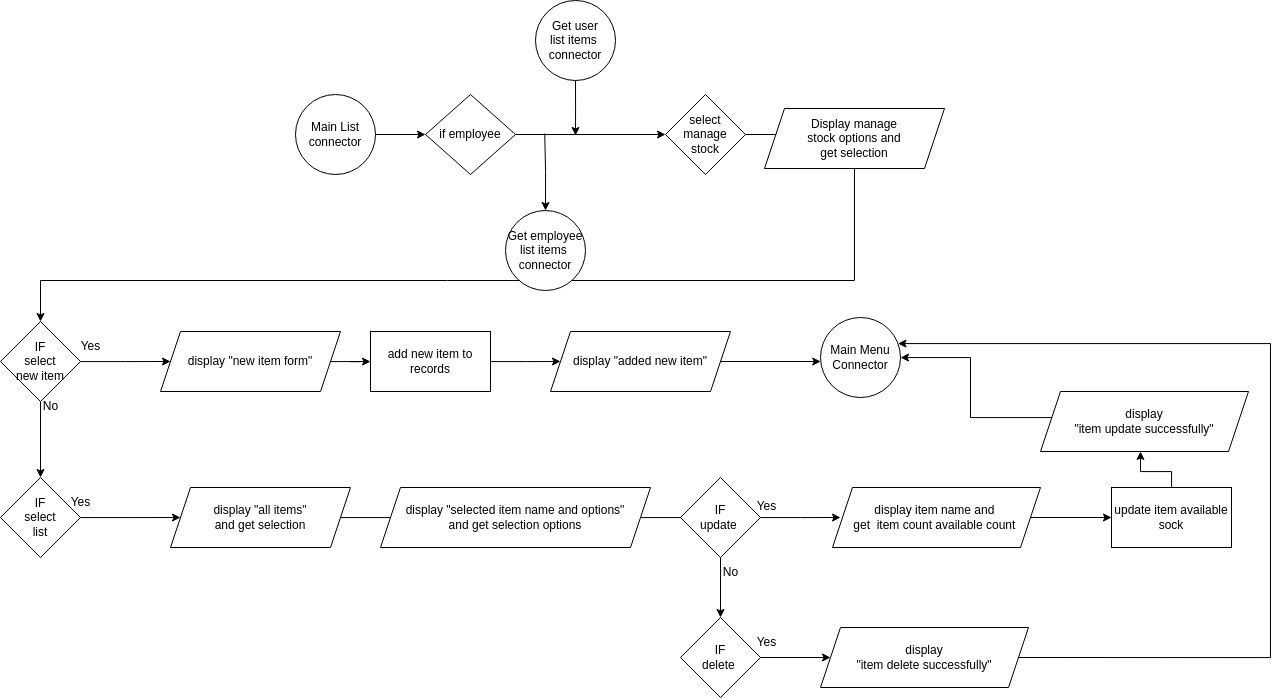
Order



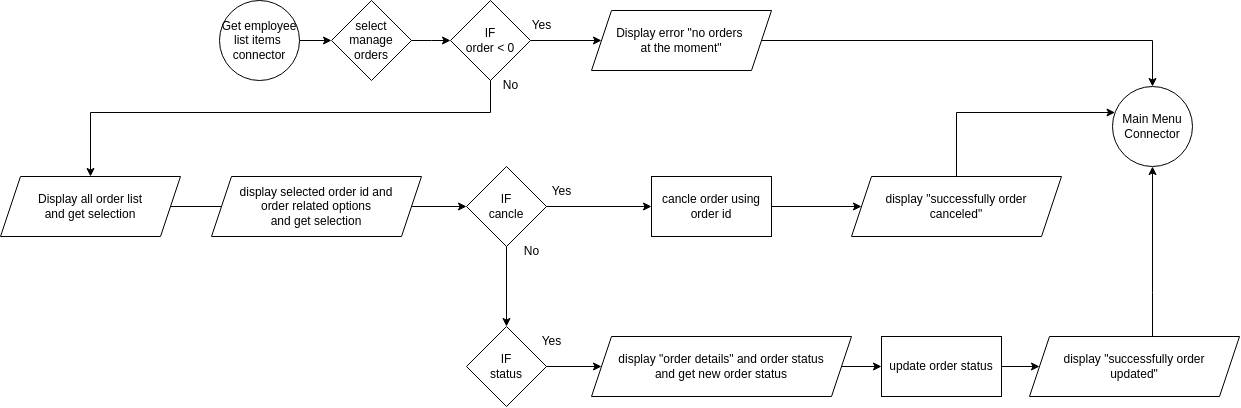
Profile



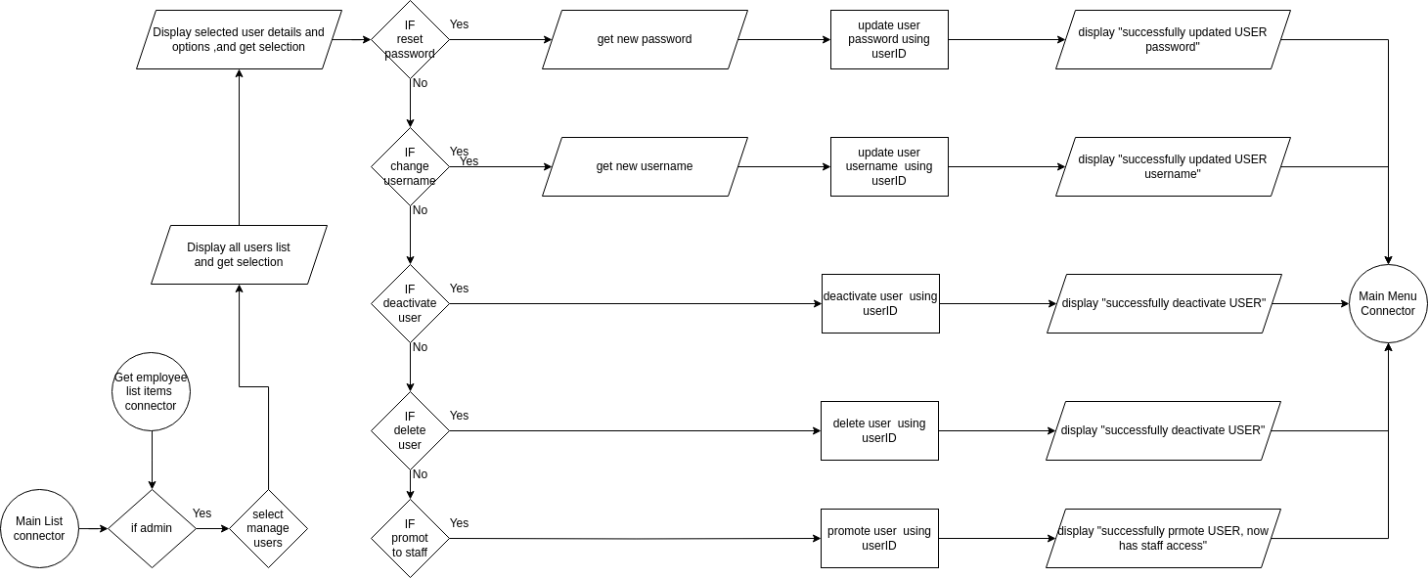
Employee furniture related area



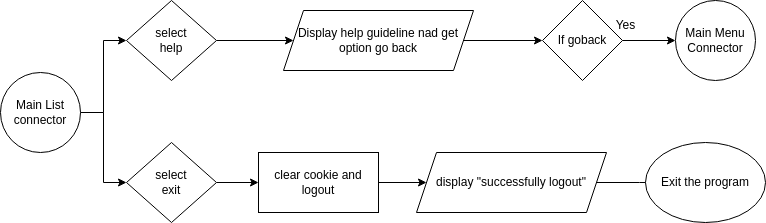
Employee order related area



Admin related area



Basic options



## 1.6 Tools and Technologies

* **Programming Language**: C++
* **Data Storage**: Text files / CSV files for simplicity
* **IDE/Editor**: Visual Studio Code
* **Version Control**: Git (recommended for backup and versioning)

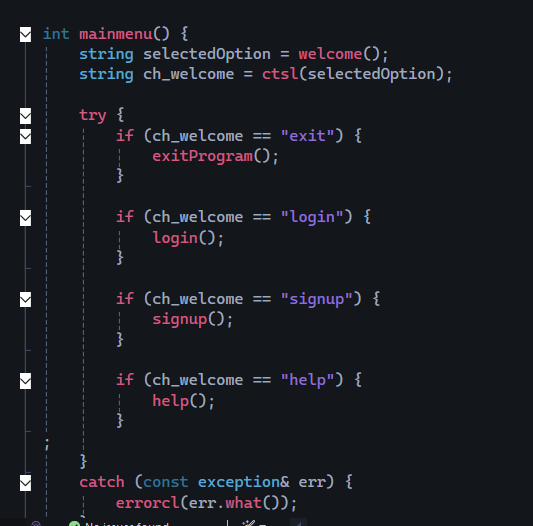
## 1.7 Conclusion

This document outlines the core requirements and design flow for the FURNITURE VILLEGE Furniture Ordering System. By adhering to this SRS, developers can implement a robust, user-friendly, and maintainable system. The flow charts and modular design help reduce complexity, ensuring each feature is well-defined and straightforward to implement.

# TASK 2

## Key Code Components:

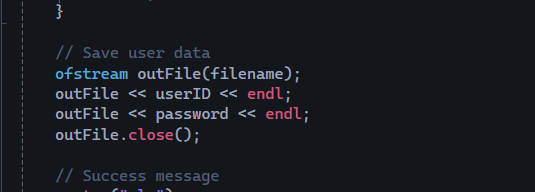
1. **Control Structures Implementation:**



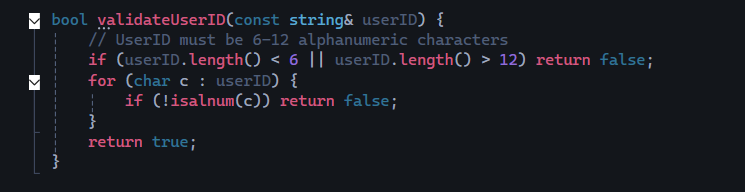
1. **Modularization & Data Passing:**



1. **Data Storage Implementation:**



1. **Input Validation & UX:**



1. **Navigation & System Flow:**



# TASK 3

## Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Test Case ID | Test Case name | Steps | Results |
| 1 | TC\_001 | **Verify Login (Valid Credentials)** | 1. Launch the system 2. Enter valid username & password 3. Click **Login** 4. Verify the main menu is displayed | pass |
| 2 | TC\_002 | **Verify Login (Invalid Credentials)** | 1. Launch the system 2. Enter invalid username or password 3. Click **Login** 4. Verify an error message is displayed | Pass |
| 3 | TC\_003 | **Add Furniture Item** | 1. Log in as admin 2. Select **Add Furniture** from main menu 3. Enter new furniture details (ID, name, category, price, quantity) 4. Confirm entry | Pass |
| 4 | TC\_004 | **List All Available Furniture** | 1. Log in 2. Select **List Furniture** from main menu 3. Verify all furniture items are displayed in a list/table | Pass |
| 5 | TC\_005 | **Search Furniture (By ID)** | 1. Log in 2. Select **Search** from main menu 3. Enter a valid furniture ID 4. Verify the correct item details are displayed | Pass |
| 6 | TC\_006 | **Place Order (Sufficient Stock)** | 1. Log in 2. Select **Place Order** 3. Enter customer details, furniture ID, and quantity 4. Confirm order 5. Check if inventory is updated | Pass |
| 7 | TC\_007 | **Exit System** | 1. Log in 2. Select **Exit** from main menu 3. Verify the system closes and saves data properly | Pass |
| 8 | TC\_008 | **Search Furniture (By Name)** | 1. Log in 2. Select **Search** from main menu 3. Enter a valid furniture name 4. Verify the correct item details are displayed | Pass |
| 9 | TC\_009 | **Place Order (Insufficient Stock)** | 1. Log in 2. Select **Place Order** 3. Enter customer details, furniture ID, and quantity that exceeds available stock 4. Attempt to confirm order | Pass |
| 10 | TC\_010 | **Display Help Menu** | 1. Log in 2. Select **Help** from main menu 3. Verify that the help instructions and guidance information are displayed | Pass |

## Test Cases

Test case 1

|  |  |
| --- | --- |
| **Test Case ID** | TC\_001 |
| **Test Objective** | Ensure a user can log in with valid credentials. |
| **Test Data** | - Username: user - Password: pass123 |
| **Expected Result** | - System accepts credentials - Main menu is displayed |
| **Actual Result** | Main menu displayed successfully |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 2

|  |  |
| --- | --- |
| **Test Case ID** | TC\_002 |
| **Test Objective** | Prevent login with invalid credentials. |
| **Test Data** | - Username: wrongUser - Password: 123wrong |
| **Expected Result** | - Error message is displayed: “Invalid credentials” - User is prompted to re-try |
| **Actual Result** | Error message displayed, user not granted access |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 3

|  |  |
| --- | --- |
| **Test Case ID** | TC\_003 |
| **Test Objective** | Verify a new furniture item can be added to the system. |
| **Test Data** | - Furniture ID: F001 - Name: Chair - Category: Living Room - Price: 100 - Quantity: 10 |
| **Expected Result** | - System stores the item - Confirmation message: “Furniture added successfully” |
| **Actual Result** | Furniture added, confirmation message shown |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 4

|  |  |
| --- | --- |
| **Test Case ID** | TC\_004 |
| **Test Objective** | Ensure the system can list all available furniture. |
| **Test Data** | Select furniture list |
| **Expected Result** | - System displays all furniture in the inventory - Shows ID, name, category, etc. |
| **Actual Result** | List of furniture items displayed correctly |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 5

|  |  |
| --- | --- |
| **Test Case ID** | TC\_005 |
| **Test Objective** | Check search functionality for a specific furniture by ID. |
| **Test Data** | - Furniture ID: **F54** |
| **Expected Result** | - System displays details of item with ID F001 - If not found, shows “Item not found” |
| **Actual Result** | Correct item displayed |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 6

|  |  |
| --- | --- |
| **Test Case ID** | **TC\_006** |
| **Test Objective** | Place an order when stock is sufficient. |
| **Test Data** | - Customer Name: John Doe - Furniture ID: F001 - Quantity: 2 |
| **Expected Result** | - Order is created - Inventory for F001 is reduced by 2 - Confirmation message |
| **Actual Result** | Order placed, inventory updated, confirmation shown |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 7

|  |  |
| --- | --- |
| **Test Case ID** | TC\_007 |
| **Test Objective** | Verify the system exits and saves data properly. |
| **Test Data** | No specific input data; user simply selects Exit |
| **Expected Result** | - System closes application - Data is saved/persisted |
| **Actual Result** | Application closed successfully, data persisted |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 8

|  |  |
| --- | --- |
| **Test Case ID** | TC\_008 |
| **Test Objective** | Verify the search functionality for a specific furniture by name. |
| **Test Data** | - Furniture Name: Chair |
| **Expected Result** | - System displays details of the furniture item matching “Chair” |
| **Actual Result** | Correct item details displayed based on name |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 9

|  |  |
| --- | --- |
| **Test Case ID** | TC\_009 |
| **Test Objective** | Test place order when requested quantity exceeds stock. |
| **Test Data** | - Customer Name: Jane Smith - Furniture ID: F001 - Quantity: 20 (assuming only 2 in stock) |
| **Expected Result** | - System should display an error: “Insufficient stock available” and reject the order |
| **Actual Result** | Error message displayed; order not processed due to insufficient stock |
| **Conclusion** | Pass |
| **Screenshot** |  |

Test case 10

|  |  |
| --- | --- |
| **Test Case ID** | TC\_010 |
| **Test Objective** | Verify the Help menu displays clear guidance and instructions. |
| **Test Data** | No specific data input; simply select Help from the main menu |
| **Expected Result** | - Help instructions and usage guidelines are clearly displayed |
| **Actual Result** | Help menu displayed with clear instructions |
| **Conclusion** | Pass |
| **Screenshot** |  |

* The **test plan** and **test cases** show that the core functionalities (login, add furniture, list, search, place order, exit) are working as expected.
* **UAT** feedback indicates the system is generally user-friendly and meets business needs, with minor suggestions for enhanced user messages.

# Conclusion

The finished project offers a comprehensive outline for creating an end-to-end plan for deploying a strong Furniture Ordering System at FURNITURE VILLEGE. Below are the key highlights of the work:

**Clear Requirements and System Design:**

The SRS document describes all the key system features from secure log in to inventory control to order processing. It makes sure that each module is described clearly, which results in a system with minimal human workload and higher customer satisfaction.

**Modular and Structured Approach:**

The system has been kept modular in its design. Fundamental activities such as adding furniture, displaying products available, and ordering are well defined, making maintenance and later on upgrading quite easy. Use of flowcharts in defining the processes also reduces complexity, with the design not so complex that it is difficult to understand for developers and even stakeholders.

**Stringent Testing Methodology:**

They have also had a rigorous test schedule and test scenarios so that every function is working as it should. Adding further test cases like name search, insufficiency stock management in placing an order, and assistance menu display checking shows a highly strict quality check process. There is also massive testing, User Acceptance Testing (UAT), and gathering feedback to confirm that the system meets specified requirements and user demands.

**Scalable and User-Centric Design:**

With usability as a prime focus, the system boasts an end-user-driven, menu-supported interface that is intuitive and quick to use. The design facilitates easy learning and use by the employees, ensuring greater operational performance and an overall improved user experience.

In short, the project is able to successfully integrate requirement analysis, system design, and extensive testing to deliver a sustainable and stable Furniture Ordering System. This systematic approach not only ensures that the system meets today's business needs but also lays a solid foundation for future development and integration of additional functionalities.