## **Furniture Inventory & Management System (FIMS)**

Group Name: Placeholders

Team Members : Arian Vares , Jared Nathan Batallones , Jilun Liang , Wei Cui , Guillermo Rebolledo

**Furniture Inventory & Management System (FIMS)**

|  |  |
| --- | --- |
| Team Members | Role |
| Arian Vares | Project Manager & Documentation Lead , Frontend |
| Jared Nathan Batallones | Backend Developer (Database & Authentication)   |  | | --- | |  | |
| Jilun Liang | Frontend Developer (UI & System Interface)   |  | | --- | |  | |
| Wei Cui | Search & Filtering System Developer   |  | | --- | |  | |
| Guillermo Rebolledo | Quality Assurance and testing |

## 

## 

## 

**1. Project Summary**

A detailed document titled "Project Overview – Furniture Inventory & Management System" was submitted to the instructors. This document outlines the minimum viable product (MVP) for the project, covering features developed and progress made so far.

**2. Introduction**

The Furniture Inventory & Management System (FIMS) is a desktop application built to help furniture store owners and managers track and manage their inventory efficiently. Initially, it was a command-line program, but we upgraded it to a Tkinter-based graphical user interface (GUI) for better usability.

FIMS includes a secure login system with five user roles:

* Admin
* Manager
* Employee
* Inventory Manager
* Warehouse Employee

Users can add, edit, delete, sort, and search furniture items based on keywords like type or color. The system runs on a SQLite database, ensuring easy storage and retrieval of inventory data. Since it is designed for internal use, customer checkout features were not included, but managers have access to various tools to streamline operations.

**3. First Iteration Summary**

**Key Developments:**

1. **Project Setup & Repository:**
   * Created a GitHub repository with a main branch and README.
   * Chose Python as the primary language and SQLite for database storage.
   * Designed an initial SQLite database structure for users and inventory data.
   * Established team collaboration tools using Trello and Discord.
2. **GUI & Core Functionalities:**
   * Built a Tkinter-based GUI with a secure login system featuring role-based access control.
   * Added inventory management features (add, edit, remove, sort, search).
   * Implemented keyword search (by type or color).
   * Integrated a database with two tables:
     + User table: staff\_id, level, username, password.
     + Furniture table: furniture\_id, type, color, price.
3. **Basic Inventory Management:**
   * Users can view all inventory items or search for specific ones.
   * Admins can manage user access (add/remove users).
   * Implemented sorting by type and price.
4. **Testing & Debugging:**
   * Conducted manual tests to verify login functionality, inventory operations, sorting, and searching for all roles:
     + **Admin:** Full access, including user management.
     + **Manager:** Add, edit, remove, sort, search, view inventory.
     + **Employee:** Search and view only.
     + **Inventory Manager:** Sort, search, and view.
     + **Warehouse Employee:** View-only access.

**4.User Stories & Prioritization**

**Must-Have Stories (Core Features)**

|  |  |  |  |
| --- | --- | --- | --- |
| **User Role** | **User Story** | **Estimated Hours** | **Priority** |
| Administrator | I want a secure login system with authentication, so that only authorized users can access and manage the inventory. | 5 hours | 10 |
| Store Manager | I want to add new furniture products, so that I can keep the inventory up to date. | 5 hours | 10 |
| Employee | I need to view products in a list format, so that I can efficiently scan inventory for updates and stock checks. | 5 hours | 10 |
| Inventory Manager |  | 5 hours | 10 |
| Warehouse Employee |  | 5 hours | 10 |

**We have implemented all these User Stories considered Core to our MVP.**

1. **UML Diagrams**

* **Admin Modifying Furniture:** Shows how the Administrator selects and modifies furniture using the Tkinter GUI while interacting with database functions.

A diagram of a furniture system

AI-generated content may be incorrect.

* **General UML for Core Features:** Illustrates the GUI generating user interfaces, the furniture module handling inventory, and the database module managing SQLite queries.

A screenshot of a computer program

AI-generated content may be incorrect.

**5. Gaps and Limitations**

1. First, right now employees, admins, and managers can only log in using their Staff ID. We would like them to be able to log in using their Usernames, to be implemented in the next iteration using the Tkinter GUI.
2. The database needs more entries. At least 5 items of each type, and 5 items of each color. We could overhaul the database to include columns like aisle, so you would know what aisle the item is in. A mass column, so one knows how heavy each item is. A size column, so one knows whether at least the size is "small", "medium" or "large". Just price column alone provides too little information for a real business. Real businesses aren't just accountants.
3. The current Tkinter GUI, while functional, could be enhanced for a better user experience. Future iterations may explore a Django-based interface for improved scalability and aesthetics.

4. We are not sure how secure the system is; we need to test it, so that employees or others can't just hack themselves into Manager or Admin roles. Security testing will be a priority in Iteration 2.

**6. Video Demo**

The following link provides access to the older video demo (text-based). The included screenshots showcase the latest version of the application, updated with a Tkinter-based interface.(You can also find them in Screenshots folder)

[<https://drive.google.com/file/d/1L5aqbwsF2PvUD36yNJV0Z59IvJ0cVnMC/view?usp=sharing>]

A screenshot of a computer

AI-generated content may be incorrect.

**FIMS MVP Walkthrough**

1. **Start Up**: Run python main.py—you’ll see a login window.
   * Screenshot: screenshots/login.png
2. **Log In**: Use Staff ID 1, Password pass123 for admin access.
   * Screenshot: screenshots/main\_menu.png
3. **View All**: Click "View All Furniture" to see the list.
   * Screenshot: screenshots/view\_all.png
4. **View Specific**: Click "View Specific Furniture", enter 101.
   * Screenshot: screenshots/view\_specific.png
5. **Sort**: Click "Sort Furniture", pick Type or Price.
   * Screenshot: screenshots/sort\_furniture.png
6. **Search**: Click "Search Furniture", type black.
   * Screenshot: screenshots/search\_furniture.png
7. **Add**: Click "Add Furniture", enter ID: 103, Type: Sofa, Colour: Red, Price: 199.99.
   * Screenshot: screenshots/add\_furniture.png
8. **Edit**: Click "Edit Furniture", enter 103, change Colour to Blue.
   * Screenshot: screenshots/edit\_furniture.png
9. **Remove**: Click "Remove Furniture", enter 103.
   * Screenshot: screenshots/remove\_furniture.png
10. **Log Out**: Click "Log Out" to go back to login.
    * Screenshot: screenshots/login.png

A screenshot of a login screen

Description automatically generated

login

A screenshot of a computer screen

Description automatically generated

main\_menu\_admin

A screenshot of a black and white screen

Description automatically generated

main\_menu\_manager

A screenshot of a black screen

Description automatically generated

main\_menu\_employee

A screenshot of a black screen

Description automatically generated

main\_menu\_inventory\_manager

A screenshot of a phone

Description automatically generated

main\_menu\_inventory\_manager

A screenshot of a computer

Description automatically generated

main\_menu\_admin

**A screenshot of a computer

Description automatically generated**

view\_specific

**A screenshot of a computer

Description automatically generated**

sort\_furniture

A screenshot of a computer

Description automatically generated

search\_furniture

A screenshot of a computer

Description automatically generated

edit\_furniture

A screenshot of a computer

Description automatically generated

add\_furniture

A screenshot of a computer screen

Description automatically generated

remove\_furniture

**7. Repo Code and Trello**

The following link should that you to our Github with the code for our MVP:

[ https://github.com/jaredBatallones/csci2040Uproj ]

The following link should that you to our Trello:

[ https://trello.com/b/nlMoDgif/placeholder-group ]

A screenshot of a chat

AI-generated content may be incorrect.

**8. Challenges**

- Fortunately, we encountered no significant technical challenges with the coding or implementation of the SQLite database and Tkinter GUI. However, we did face some organizational hurdles.

- Organizational Issues: The team experienced difficulties in coordinating effectively, resulting in several tasks being completed at the last minute due to poor initial planning and communication.

- Issues Adding a New Member: A newly added team member initially faced challenges connecting to Discord and receiving emails. This has since been resolved, with all members now active on Discord, and we anticipate smoother onboarding in future iterations.

- Underestimation of the Time: The team initially underestimated the complexity of certain tasks, assuming they would be straightforward, which led to last-minute work and time constraints.

Below is a burndown chart, with our ideal work pace vs our actual real life work pace for Iteration1. Green is the ideal pace. The red is the actual pace, best fitted on to our chart, using Python : A graph with a red line and a black line

AI-generated content may be incorrect.

We thought we could get 7 hours of work done a day. But the actual velocity of work as about 5.441 hours. We didn’t miss the deadline, because we work a lot during the last day, completing Iteration 1 on day 12 as expected.

**9. Next steps and Iteration 2:**

**Iteration 1 was complete now here is our plan of Iteration 2**

**Sprint Plan**

|  |  |  |
| --- | --- | --- |
| **Iteration 1** | **Iteration 2** | **Iteration 3** |
| Set up GitHub Repo & Project Structure | Implement Sorting & Filtering | Final Bug Fixes & Testing |
| Secure authentication for users | Implement Add/Edit/Remove Product Feature | Prepare the System for Demo |
| Basic UI for Product List & Individual Products | Basic Search System (Keyword-Based) | Final Run-through & Documentation Updates |

* As mentioned before we need to overhaul the database and add more entries. There needs to be more columns like which aisle the items are in. What is their size, how much it weights etc.
* Users should be able to login with their Usernames
* The functionalities need to be in GUI, with a better UI, implemented in Django.

**Below is a more detailed plan for Iteration 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Estimated Hours** | **Assigned Members** | **Date** |
| Add users/permissions | 4 | *Jared, Arian, Wei* | March 07 |
| Get started on GUI UI | 5 | *Jared, Arian, Wei* | March 07 |
| Statechart Diagram | 4 | *Jared, Guillermo* | March 08 |
| More UI work | 5 | *Jared, Arian, Wei* | March 08 |
| Iteration 1 deliverables/estimates | 9 | *Jilun* | March 09 |
| Database overhaul | 5 | *Wei* | March 10 |
| More Database entries | 4 | *Wei* | March 10 |
| Work on GUI UI | 5 | *Whole Team* | March 11 |
| Finish DataBase and GUI | 5 | *Whole Team* | March 11 |
| Work on report | 6 | *Whole Team* | March 12 |
| Test database | 3 | *Wei* | March 12 |
| Unit test methods | 5 | *Jared, Jilun* | March 13 |
| Test GUI | 5 | *Arian, Jilun, Guillermo* | March 13 |
| Customer meet 3 | 1 | *Whole Team* | March 14 |

**10. Completed Code & Functionalities**

* Implemented the application using Python and SQLite.
* Developed a role-based login system with five user roles.
* The main.py file contains a demo SQLite database with sample furniture products.
* Verified secure login using Staff ID and password, with plans to introduce username-based login.

**11. Conclusion**

The MVP for FIMS provides a functional inventory management system with essential features for store owners and employees. Future iterations will focus on expanding database functionality and improving user experience.