**Store Inventory Management System**

**CIS 152: Data Structures**

**Hugo Alvarez Valdivia**

*Introduction*

A company that deals with a diverse range of products requires an efficient inventory management system to keep track of its product stock, manage restocking, and provide real-time information on product availability. This IMS serves as the backbone for businesses dealing with diverse product portfolios. This project was created with that issue in mind. It tries to address the pressing need of a company navigating both physical and online retail spaces. The challenge at hand involved designing and developing an IMS capable of seamlessly managing product inventory, tracking orders, and optimizing restocking processes. The overarching goal was to unify the inventory system across physical stores and online platforms, ensuring real-time visibility into product availability. The emphasis was on creating a user-friendly interface for employees, empowering them to execute tasks such as adding, updating, and removing products with ease. The system also needed to incorporate a robust search mechanism based on various criteria like product name, category, and supplier. A critical component was an order tracker that would automatically update inventory, ensuring accurate and up-to-date information.

To address the complexity of handling large datasets, I made the decision to implement the merge sort algorithm for sorting purposes. This project was developed using Qt for the GUI, and leveraged the versatility of std::list as the underlying data structure. While the initial scope outlined a clear path, the journey was not without obstacles. Challenges emerged in the form of Qt errors related to a corrupted .pro file and intricacies in GUI logic. This report elucidates the evolution of the project, detailing the scope adjustments made in response to unforeseen challenges, and the solutions devised to overcome them.

*Time-logs*

|  |  |
| --- | --- |
| Feature | Date of Implementation |
| Repository creation (Initial Commit) | 10/17/2023 |
| Created Qt project template | 10/17/2023 |
| Project proposal | 10/17/2023 |
| Added base customer class | 10/30/2023 |
| Added base product class | 10/30/2023 |
| Created database csv files | 10/30/2023 |
| Created header file to handle writing to database files | 11/08/2023 |
| Migrating project to Qt 6.4.2 (Issues with previous version) | 11/20/2023 |
| Starting to add GUI | 11/25/2023 |
| Adding navigation between windows using QPushButtons | 11/30/2023 |
| Implemented “next-available” files to help keep track of IDs. | 12/05/2023 |
| Implemented order class and its own GUI window. | 12/05/2023 |
| Moved from label-based csv parsing to utilizing QTableView widgets for clarity purposes. | 12/05/2023 |
| Implemented the mergesort algorithm using a C++ template so that any object can be sent as long as it can be comparable (< >) | 12/05/2023 |
| Migrated the entire project to Qt 6.6, as well as transforming it into a Subdirs Qt Project, which allowed me to implement unit testing for my model classes (customer, product, order) while having my widgets application separate. | 12/06/2023 |
| Implemented binary operator overrides for each model so that mergesort can be utilized. | 12/06/2023 |
| Finished GUI | 12/07/2023 |
| Added app icon | 12/07/2023 |

*Lessons Learned*

As with any complicated project, the scope of the Inventory Management System underwent several transformations throughout development. The initial vision outlined an inventory management with a particular emphasis on user-friendly interfaces and efficient data retrieval. However, the challenges encountered during implementation led to unexpected adjustments in the project's scope.

One of the first notable alterations pertained to the aesthetics of the home screen. Recognizing the importance of user experience, I revisited the design to enhance visual appeal and usability. This adjustment, while seemingly cosmetic, played a crucial role in aligning the system with modern design principles, thereby ensuring an intuitive and engaging interface for end-users.

A significant challenge emerged during the implementation phase when certain features were not supported by the Qt version initially employed. This pushed me to transition to the latest Qt version (6.6) to leverage the full spectrum of features essential for the seamless integration of the system.

Another important decision made in the development journey of this system was shifting the approach to data storage. Initially thought with the intention of utilizing an std::map for storing data, recognizing the dynamic nature of the system and the need for a more adaptable solution, a decision was made to move to a std::list at runtime, coupled with CSV files for persistent data storage, meaning that the user can still have access to their data even after a full app restart.

The initial inclination towards std::map came from the desire for an efficient data structure that facilitated quick and structured access to product information. But as the project unfolded, it became apparent that the flexibility offered by a linked list structure was better suited to handle the dynamic nature of inventory management. The switch to std::list allowed for seamless addition, removal, and modification of products, aligning more closely with the evolving requirements of the system.

Furthermore, implementing CSV files, with their simplicity and widespread compatibility, provided an effective means of storing and retrieving product, customer, and order information. This shift not only ensured that data persisted across sessions but also allowed for straightforward integration.

Another hurdle presented itself in the integration of QTests into the existing project structure. The need to incorporate testing for the merge sort algorithm and various classes, including Product, Customer, and Order. To overcome this challenge, a strategic decision was made to migrate the entire project into a Subdirs configuration. This change allowed for the inclusion of a separate Qt Test Application, dedicated exclusively to testing components like the merge sort algorithm and the core classes of the system. This modular approach facilitated the testing process.

For the latest updates, source code, and project documentation, please refer to the GitHub repository: [Inventory Management System Repository](https://github.com/hugoalval-code/cis152-final-project). Instructions are included the repo’s README file.

Conclusion

1. Conclusion/Summary
   1. A paragraph describing MERUSE principles of good programming applied to your code
   2. A paragraph summarizing your project
   3. A paragraph of future versions