Vintage Music Store Inventory System

***Abstract* – Our team is going to create a digital inventory system to service a vintage music store. This inventory system should include key inventory actions such as adding, removing items, keeping count, records, and a user login system. The team is going to build the inventory system with MySQL, JavaFX, with java being our main language. This report will include the processes and design of the inventory system. As well as plans for future development.**

I. Introduction

An inventory system can have various uses besides basic adding/removing/keeping count. Examples would be a login/user system, analytics, an interactive GUI, and an in-house repair service. The overall plan for this inventory system is to create an efficient and easy way for employees/managers to keep track of inventory.

A. Problem Description

The current situation at the vintage music store is that an outdated manual way of keeping inventory is used, given the

advanced technology of today's day and age. In the context of this store, many musical instruments are heavy, fragile, and require specific instructions/maintenance. The issue with this current system is that employees are required to do tedious tracking and counting with the manual system, leaving a bigger chance for human error. Along with error, valuable time and labor could be saved, and used for other business operations. There would be massive benefits to moving to an online inventory system that can keep items organized, records readily available, and create analytics as well.

B. Proposed Solution

The solution for the vintage music store would be to create a fully online inventory system that takes advantage of modern technology to create organization and convenience. Our team will implement an inventory application, complete with a frontend/backend and login system. Employees will be able to create an account, and add/remove items, as well as see counts of current stock, see background information of selected items, as well as serial numbers, condition, etc. There will also be an admin account that would be able to manage and oversee the employee accounts as well as the transactions they make.

II. Analysis

A. System Requirements

The inventory system's core functionality relies on first an employee/manager having an account. First an admin account must exist before any employee accounts are created, then from the admin account, employees are able to be made. Once a user logs into the login page, they are redirected to the main inventory page where both types of accounts can interact/change inventory within the user interface. The inventory system is created specifically around instruments/musical equipment. An instrument usually falls into the category of (woodwind/percussion/string/brass/electronic, other) which are the available categories within the system. Other requirements are current price, as well as retail price, to implement advanced analytics features. All transactions are recorded and kept on the database, and the option to archive items is also available. Lastly an in house repair service is an option within the inventory system for employees to keep track of.

B. Actors and Stakeholders

Our inventory system involves five main stakeholders that can be divided into Customer, Clerk, Manager, Technician, and the owner. Internal stakeholders include Clerk, Manager, Technician and owner. The Clerks, Managers, and Technicians internal stakeholders benefit from the inventory system that provides organized count/information description, along with archive/records/detailed information that can be useful to both Clerks and Technicians who need specific requests. Managers benefit by analyzing the analytics of inventory transactions and documentation of Clerks and Technicians. The owner benefits from the increased efficiency and accuracy of their employees. The external stakeholder is the Customer, who benefits by the reliability of the new inventory system compared to a manual tracking system. The customer can also request more specific information on a specific instrument/equipment that can be retrieved relatively quickly.

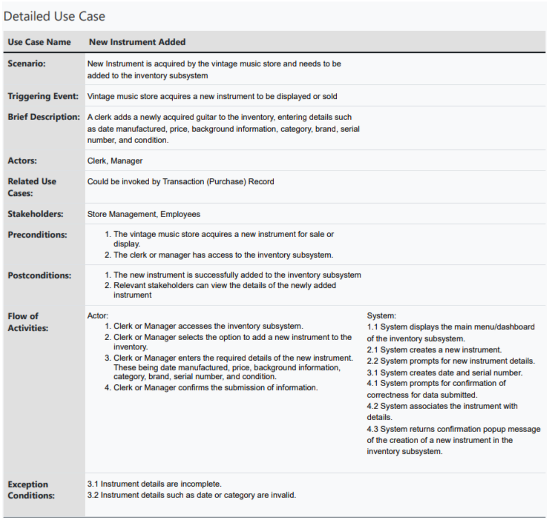
C. Analysis Model

We analyzed the actors in our Vintage Music Store Inventory System and we decided that we needed to create an analysis model for our system in order to clearly grasp the scope of this project. The process of creating an analysis model allowed us as a group to understand and plan out each component of the system before we went ahead with trying to program it. The analysis model was useful in not only showing what functionality we needed to implement into the program but also what functionalities fit into the scope of the project we were trying to accomplish so we did not get sidetracked as easily.

D. Use Cases

When we first chose our topic, we all split up in order to do our research about how different inventory subsystems work and what functionalities they should have. We came together then and proposed the requirements that we thought an inventory subsystem should have for a vintage music store. From there, it was much easier to create the use cases as we only needed to take the functionalities we had listed and figure out how they would work in regards to our program. Since the scope of our project was wide, we ended up having fourteen different use cases. These use cases being, Instrument Added, Instrument Removed, Instrument Quantity Increase, Instrument Quantity Decrease, Instrument Details Update, Instrument Deletion, Instrument Archive, Transaction (Sale) Record, Transaction (Purchase) Record, System Archiving Events, Enter Repair, Change Repair Status, Generate Inventory Analytics, and Search Inventory. Our actors are Clerk, Manager, and System. Originally we had made the mistake of putting the Customer as an actor in our system but we corrected this mistake as they would not have direct access to this subsystem.

E. Detailed Use Case



Detailed Use Case

Use Case Name	New Instrument Added	
Scenario:	New Instrument is acquired by the vintage music store and needs to be added to the inventory subsystem	
Triggering Event:	Vintage music store acquires a new instrument to be displayed or sold	
Brief Description:	A clerk adds a newly acquired guitar to the inventory, entering details such as date manufactured, price, background information, category, brand, serial number, and condition.	
Actors:	Clerk, Manager	
Related Use Cases:	Could be invoked by Transaction (Purchase) Record	
Stakeholders:	Store Management, Employees	
Preconditions:	1. The vintage music store acquires a new instrument for sale or display. 2. The clerk or manager has access to the inventory subsystem.	
Postconditions:	1. The new instrument is successfully added to the inventory subsystem 2. Relevant stakeholders can view the details of the newly added instrument	
Flow of Activities:	Actor: 1. Clerk or Manager accesses the inventory subsystem. 2. Clerk or Manager selects the option to add a new instrument to the inventory. 3. Clerk or Manager enters the required details of the new instrument. These being date manufactured, price, background information, category, brand, serial number, and condition. 4. Clerk or Manager confirms the submission of information.	System: 1.1 System displays the main menu/dashboard of the inventory subsystem. 2.1 System creates a new instrument. 2.2 System prompts for new instrument details. 3.1 System creates date and serial number. 4.1 System prompts for confirmation of correctness for data submitted. 4.2 System associates the instrument with details. 4.3 System returns confirmation popup message of the creation of a new instrument in the inventory subsystem.
Exception Conditions:	3.1 Instrument details are incomplete. 3.2 Instrument details such as date or category are invalid.	

Fig. 1 Detailed Use Case Diagram for New Instrument Added

We chose the use case New Instrument Added to use for our detailed use case diagram. This use case is utilized when the vintage music store acquires a new instrument that needs to be added to the system. The New Instrument Added use case prompts the clerk to enter details about the instrument such as date manufactured, price, background information, category, brand, serial number, and condition. The two actors for this use case are clerk and manager. This use case can be invoked by the Transaction (Purchase) Record use case. The flow of activities starts with the clerk or manager accessing the inventory subsystem. The system then displays the main menu of the inventory subsystem that the clerk then uses to add a new instrument to the inventory. This causes the system to create a new instrument and prompts the user for the instrument details. The user then enters the details required and confirms the submission is indeed correct. The exception conditions are if the instrument details are left incomplete or are invalid such as the date or category. The system then associates the instrument with the details and returns a confirmation popup message that the instrument has been populated into the inventory subsystem which ends this use case.

F. *Domain Model Diagram*

With all of our uses cases figured out, we could then move on to creating our domain model diagram. This diagram is especially useful in creating a stable foundation for the project to build off of as it helps clearly identify different key concepts and how they should be structured within the code going forward.

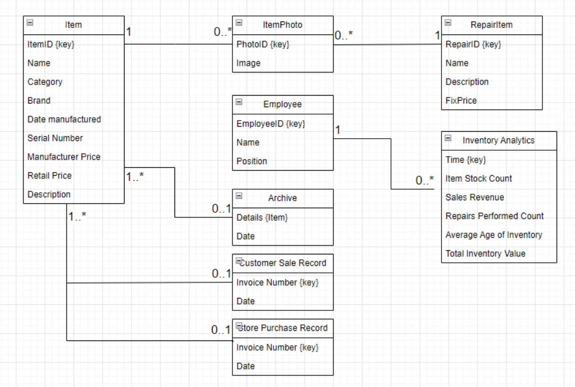


Fig 2. Domain Model Diagram

Starting on the left of our domain model diagram, we have the Item class. It is a very important class that stores the details of our instruments such as name, category, brand, date manufactured, serial number, manufacturer price, retail price, and description. The Archive, Customer Sale Record, and Store Purchase Record classes can have one or more items, but each item can only have zero or one of those classes. This is because an item is either apart of the record or archived or still active in the inventory system.

There can be zero or more connections per item with the ItemPhoto class. This is because not all items in the database will be populated with an image. The multiplicities are also the same for the RepairItem class which is implemented in the program to be able to take an image, name of the instrument, the description of the problem, and the price to fix it.

The Employee and Inventory Analytics classes are tied together as we have implemented the functionality of requiring a certain permission level to generate the inventory analytics. The position of the employee must be set to a manager in order to not only access existing inventory analytics but also to generate new ones.

G. *Activity Diagram*

Vintage Music Store Inventory System

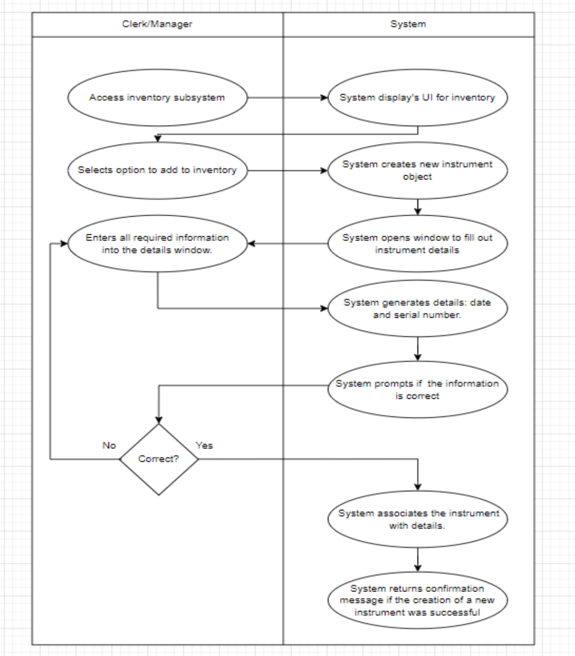

Clerk/Manager
System
Access inventory subsystem
System display's UI for inventory
Selects option to add to inventory
System creates new instrument object
Enters all required information into the details window.
System opens window to fill out instrument details
System generates details: date and serial number.
System prompts if the information is correct
No
Yes
Correct?
System associates the instrument with details.
System returns confirmation message if the creation of a new instrument was successful

Fig 3. Activity Diagram

An activity diagram is a useful tool when designing a project as it allows one to visualize every step of the user and system interaction process. This diagram is as useful as the domain model diagram in its own right as it shows the flow of activities instead of the attributes and multiplicities of each class.

Our activity diagram walks us through the flow of activities for the New Instrument Added use case. The clerk or manager would first access the inventory subsystem which leads the system to display the UI for our inventory. The clerk or manager would then select the option to add an instrument to our inventory and from there the system would create a new instrument object and open a window for the clerk or manager to fill out the instrument details. After the clerk or manager fills out the required information into the details window, the system would generate details such as date and serial number and then prompt the user to see if the information was correct. If not, the details window would pop up again to repeat that process, if the information was indeed correct, the system proceeds with associating the instrument with the details provided and returns a confirmation upon the successful creation of a new instrument.

H. *System Sequence Diagram*

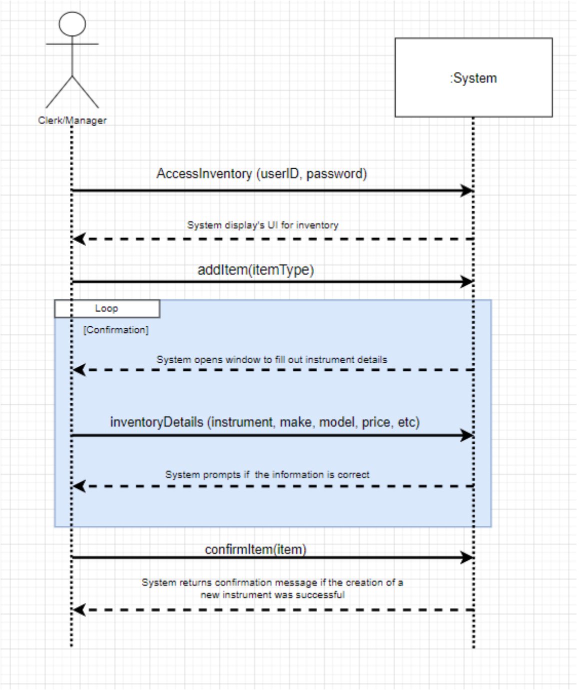

Clerk/Manager
:System
AccessInventory (userID, password)
System display's UI for inventory
addItem(itemType)
Loop
[Confirmation]
System opens window to fill out instrument details
inventoryDetails (instrument, make, model, price, etc)
System prompts if the information is correct
confirmItem(item)
System returns confirmation message if the creation of a new instrument was successful

Fig 4. System Sequence Diagram

The system sequence diagram is very similar to the activity diagram but instead of focusing on the flow of activities, it focuses on how objects interact with each other. In this diagram, you can see the two actors are once again the clerk or manager and the system. The user accesses the inventory system with their userID and password while the system responds by displaying the UI for the inventory. The user then adds an item with the corresponding item type. We then enter a confirmation loop where the system pops out a details window where the user fills out the inventory details such as instrument, make, model, price, etc. The system then prompts if the information is correct, and we leave the loop. The user confirms the item, and we are returned a confirmation message from the system.

III. Design

A. System Architecture

The system architecture of our inventory system is divided into three main layers.

1. Presentation: This layer includes the user interface which we built using JavaFX to provide a way for users and managers to interact with the inventory system. It includes forms for adding and updating inventory items, viewing item details, and managing user accounts.

2. Business Logic: This layer includes all the business logic that is required for managing inventory items and handling user authentication. Here, we make sure that all the operations performed by users and managers are validated and processed before interacting with the database.

3. Data Access: This layer is for interacting with the MySQL database. We included methods for creating, reading, updating, and deleting operations on the inventory items, user accounts, and transaction records. We abstract the database operations from the business logic to make sure that the data is stored securely.

B. Database Schema

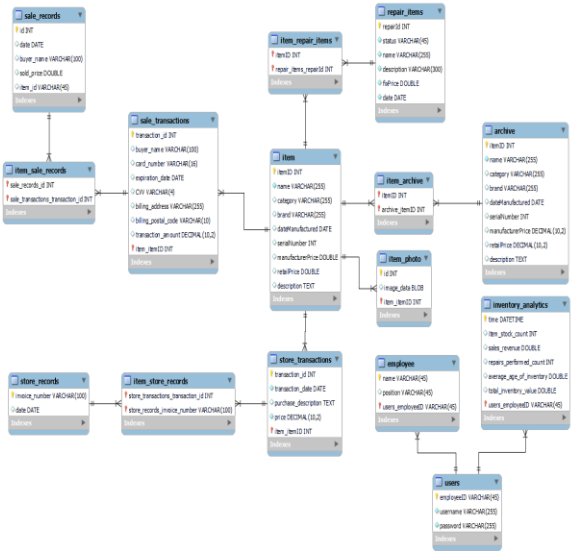


Fig. 5 Database Schema

Our database schema defines tables and their relationships in our MySQL database. The tables included data regarding sales, inventory, employee details, and items. It links sales and transactions to specific items and store records. We also used specialized tables for managing repairs, archiving, and user authentication.

IV. Suggestions

To improve the useability and efficiency of our inventory management system, there are some improvements and features that we can considered. First off, we could transition from a Java application to a web-based interface which could be very beneficial as this would allow users to browse the inventory online. Modern web development tools also offer a very flexible and user-friendly experience compared to JavaFX. Integrating a payment and billing system would make sales transaction much smoother. This subsystem could connect to external billing systems, which allows for easier handling of payments and financial records. Including fields for tracking purchases made by customers, payment statuses, and invoices would make the system much more comprehensive.

V. Conclusion

A. Challenges

When we were developing our inventory system, we encountered some significant challenges that tested our skills in problem solving. One of the main challenges we faced was time management. Balancing this project with our other academic and personal responsibilities required us to do careful planning and prioritization. We had to make sure that each member was contributing equally while also keeping track of our progress and the deadlines.

Another major challenge was setting SQL. Many of our team members had little to no experience with database programming, and the syntax of SQL was new to us. We had to rely on trial and error, along with research, to implement the database correctly.

We also had to learn how to use classes that handle java database connections. Having a stable connection between our Java application and the MySQL database was very important for the functionality of our system, and this required us to learn Java Database Connectivity (JDBC) and understand it fully.

Lastly, calculating the average lifetime of inventory items was a challenging aspect. Developing an algorithm to track and predict the lifespan of items based on usage and sales data required us to do thorough analysis and testing.

B. What we learned

Facing these challenges helped us learn valuable lessons and skills throughout the project. First off, we learned the importance of effective planning. The first planning phase is very important, as it sets the foundation for the entire project. Making sure that our domain model, use cases, and system diagrams were accurate saved us from potential issues during the implementation phase. We also learned about the significance of scope management. Understanding the scope of the project and setting realistic goals helped us stay focused. The project taught us the value of dividing labor. By splitting tasks based on what each individual was good at, we were able to overcome many obstacles. We gained a deep understanding of database implementation and the power SQL. Using SQL queries allowed us to manage data more efficiently, and we learned how a relational database can improve systems.

C. Recap

Our inventory system for the vintage music store aims to help users manage inventory more efficiently, save time and labor cost, and to reduce errors that could happen when trying to manage inventory items without a dedicated system. We introduce a user-friendly interface for employees to manage inventory efficiently and to keep inventory records reliable. Our implementation provides a significant improvement over the outdated manual methods, offering many benefits to both the store and the customers.