Project Report

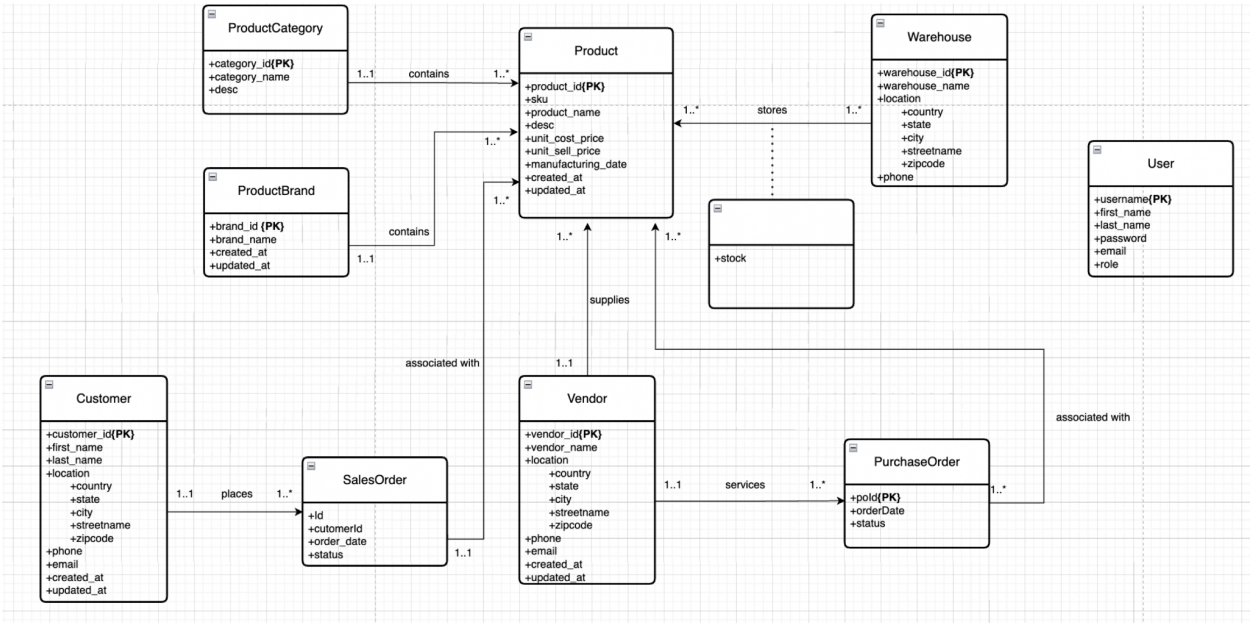
Project Group Name: AhmadASoniS

Authors: Ayman Mushtaq Ahmad, Shivam Soni

Project Title: Inventory Management System

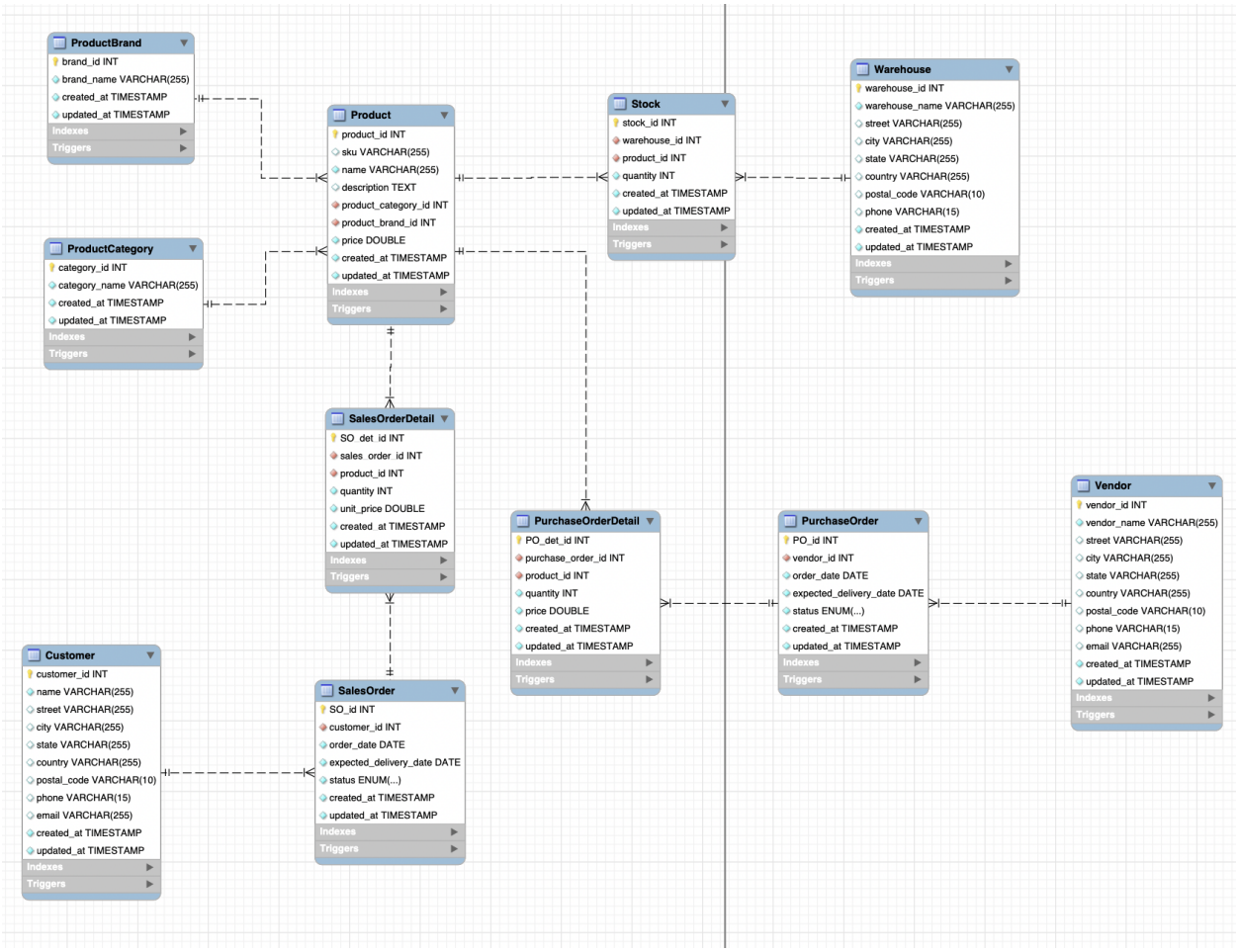
Description: The proposed project is an inventory management system designed to manage the inventory of a business. The system provides a centralized platform to manage inventory levels, track sales, and automate purchasing orders. The system will enable businesses to optimize inventory management, increase efficiency, reduce costs, and streamline their operations, thereby improving customer satisfaction.

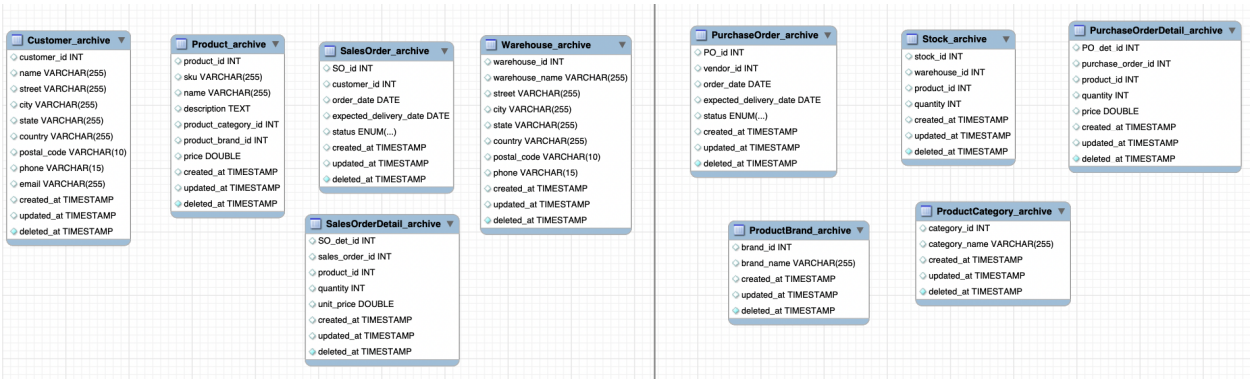
Project Conceptual Design



Project ERD in UML

Project Logical Design





README

The project has been created in Python, and MySQL has been used as a database.
Please ensure that Python 3 is installed on your machine. This would be required to run the program.

To run the program, you would need the following applications to be installed to view and run the code.
1. MySQL Workbench
2. Visual Studio Code

Installation

Install Python

- Go to the official Python website and download the latest stable release of Python for your operating system. You can download Python from https://www.python.org/downloads/.

- Follow the installation instructions for your operating system to complete the installation process. Make sure to add Python to your system's PATH environment variable, so you can run Python from any directory on the command line.

Install MySQL Workbench

- Download the MySQL Workbench installer for your operating system from the official [MySQL website] (https://dev.mysql.com/downloads/workbench/).

- Run the installer and follow the prompts to complete the installation process.
Run Locally
- **Update the database username and password in the ".env" file in the project directory.**

Database Application in Python:

Please download the zip file from the canvas submission and open the project directory "New Backend" in VS Code.

Follow the below steps to ensure all installations are complete and a virtual environment is successfully created.
To setup please create a virtual env - run following commands
1 python3 -m venv env
2 source env/bin/activate
3 pip3 install -r requirements.txt - to install all packages
All required libraries have been added to the file 'requirements.txt'.(included in the project submission)

After successful installations, the application is ready to run. Please complete the database schema import below before trying to run the application.

To run the application:

```bash
  python main.py
```

Or just press run in main.py file in Visual Studio Code

MySQL Database schema import:

The database schema will be part of the downloaded zip.
To import the database schema, please follow the below steps:

1. Open MySQL Workbench.
2. In MySQL connections, either create a new connection to MySQL or open an existing connection.
3. Once connected, go to the “Server” tab on the MySQL Workbench menu bar and click on “Data Import”.
4. From Import Options, select the option “Import from self-contained file”. Choose the path of the Schema File and click on the “start import” button .

Once the import has been completed, you can connect to the “inventorymanagement” database in the MySQL Workbench and start querying the database.

Project Requirements / Technical Description

Software Requirements:

1. Operating System: Windows or MacOS
2. DBMS: MySQL
3. Programming Language: Python

Why SQL Over NO SQL?

1. **Data Consistency:** Inventory management systems require high data consistency because it is important to have accurate information about inventory levels, order statuses, and other important data. SQL databases are designed to ensure data consistency, while NoSQL databases are optimized for scalability and performance, which can sometimes result in inconsistent data.

2. **Data Relationships:** Inventory management systems often require complex data relationships between different types of data, such as products, suppliers, orders, and customers. SQL databases use structured tables that can be easily related to one another using keys and foreign keys, while NoSQL databases use unstructured documents that are more difficult to relate to one another.

3. **Querying Capabilities:** Inventory management systems often require complex queries to retrieve data for reporting and analysis purposes. SQL databases have more advanced querying capabilities than NoSQL databases, making it easier to retrieve and analyze data.

4. **Transactions and ACID Compliance:** Inventory management systems require transactional support to ensure that inventory levels are accurately reflected in real-time. SQL databases have built-in transaction support and are ACID (Atomicity, Consistency, Isolation, and Durability) compliant, which ensures that all transactions are processed reliably and consistently. NoSQL databases may not offer ACID compliance or transaction support, which can lead to data inconsistencies.

Project Database Focus (Technical Features):

- **22 Tables** in the system including the archive tables to maintain history of data for audit purposes.
- Archival happens automatically whenever a delete happens through triggers.
- **20 plus Stored Procedures** for handling the CRUD operations in the system. Some with complexity where cursors and transactions are used along with data retrieval queries involving multiple tables.
- **10 Functions** created for various purposes to use alongside our Stored Procedures.
- **Error handling done** in our Application to ensure smooth User Experience.

Program Flow

Here is the user flow of the system:

1. User runs the CLI application
2. CLI prompts the user to select a screen among the options provided. The available options are: Inventory,Warehouse, Reports, SalesOrder, Purchase Order.
3. Each of these screens provide users with their respective CRUD operations on them.
4. Fetch: The user is provided the option to fetch all records, filter them individually based on an identifier or filter based on specific column values.
5. Update: Similarly the user is provided an option to update either Individual columns or all the fields for an identifier.
6. Create: The user is provided the option to create records as required based on the screen he had chosen. Once he's done, a new record is saved in the database.
7. Delete: the delete option asks the user to enter the respective identifier based on the screen he had chosen. Upon entering a valid identifier, the respective tuple is deleted from the database.
8. The Reports screen is different in a way that it only gives a readme option to the user to generate reports based on specific criteria.
9. The user will be prompted to input the database username and password while running the application to connect to the database.

Lessons Learned

In this section, we discuss the key lessons learned from the development and execution of the inventory management system project. The project was implemented using MySQL as the database management system and Python for the application layer. Throughout the project, we as a team faced various challenges and gained valuable insights. Some of these have been categorized below:

Technical expertise gained:

Throughout the project, the team acquired and enhanced various technical skills, including:

1. Proficiency in MySQL: Developing complex queries, stored procedures, triggers and functions to manipulate and retrieve data efficiently.
2. Advanced Python programming: Implementing object-oriented programming principles, utilizing Python libraries, and integrating the application layer with the database.
3. Version Control through Git: Maintaining version control through Git helped us in collaborating efficiently on the project.

Insights (time management, data, etc.):

1. Database Design and Normalization: During the initial phase of the project, we learned the importance of carefully designing and normalizing the database schema. Investing time in creating a well-structured, normalized schema allowed us to minimize redundancy and improve data integrity, which led to more efficient database operations and simplified application development.
2. Data Validation and Sanitization: Ensuring that user inputs are properly validated and sanitized before being stored in the database was a crucial lesson. By implementing strict validation rules and techniques, we ensured that only accurate and relevant data was stored.
3. Modular and Scalable Application Design: One of the key lessons learned during the project was the importance of designing a modular and scalable application. We followed an MVC architecture for our application for clear segregation of roles in our code. By dividing the application into smaller, independent modules, we were able to simplify development and maintenance. This approach also made it easier to scale the application in response to future growth or changes in requirements.
4. Effective Version Control and Collaboration: We used Git as a version control tool which allowed us to efficiently manage the project's source code and keep track of changes. This facilitated collaboration among us as a team, improved code quality, and made it easier to identify and resolve bugs or issues.
5. Thorough Testing and Debugging: We learned that rigorous testing and debugging are essential for delivering a reliable and robust application. By investing time in reviewing modules worked on by the other team partner, we identified and fixed various bugs and issues, ensuring a more stable and reliable application.

6. Time Management and Prioritization: Effective time management and prioritization of tasks were crucial to the project's success. We learned to break down tasks into smaller, manageable subtasks and to prioritize them based on their importance and dependencies. This approach allowed us to meet deadlines and manage the workload efficiently.
7. Communication and Documentation: Throughout the project, we realized the importance of maintaining clear and open communication among us as team members. We had regular meetings and status updates which helped keep us on the same page and facilitated the smooth execution of the project.

Realized/Contemplated Alternate Design/Approaches:

During the project, we considered several alternative designs and approaches:

1. Simplification of the conceptual schema: We as a team discussed thoroughly on the initial project proposal report and after thorough analysis and brainstorming, we came up with a better ,simplified version of the conceptual schema of our application which helped us in building the application with the idea being more clear to understand.
2. Tech Stack selection: After careful thought on the application layer technology to use, we decided to go ahead with Python instead of Java. We thought this would be a good time to learn Python as well and become proficient while building this application. As a result, our collective knowledge of Python and its libraries increased greatly. By using inbuilt python libraries like Pedantic Models which ensure that values entered by a user are of correct data type. We even utilized Python's Panda library for fetching data for the database. We even created a MySQL connector which returns a data frame as an output.
3. Implementation of GUI: We initially decided to have a GUI front end of the application in React. Since both of us were not familiar with React, we started off with learning the implementation of the same for our project. Although we did start building on it, gaining significant knowledge while at it, keeping the deadline in mind, we decided to instead shift the focus comprehensively on the database aspect of the application and making complex database objects and trying to make it a feature heavy database application for the user.

Any code not working:

By shifting our focus entirely on the database aspect of our application, we ensured that we completed all the features of the application with error handling for a smooth user experience. As such, our application is working fully without any issues.

Future Work

The inventory management system has laid a strong foundation for managing and optimizing inventory processes. However, there is a lot of future work that will help make the application more complete in terms of the use cases we plan to accommodate in the application:

1. Introducing a GUI: Developing a user-friendly graphical user interface (GUI) for the system can significantly improve its usability and visualization. A well-designed GUI would allow users to interact with the system more intuitively, perform tasks more efficiently, and access information more readily.
2. Display Report Info in the Form of a Dashboard and Bar Graphs: To facilitate better understanding and analysis of the data, the system can be enhanced to display report information in the form of a dashboard and visualizations, such as bar graphs, pie charts, and line graphs. These visualizations can help users quickly identify trends, patterns, and anomalies in the data, leading to more informed decision-making.
3. Introducing More Reports to Help Businesses Make Better Decisions: The system's reporting capabilities can be expanded to include additional reports that provide valuable insights for businesses.
   These additional reports can help businesses make more informed decisions about procurement, pricing, promotions, and inventory optimization.
4. Ability to Upload Data Through CSV Files: To simplify the process of importing data into the system, the functionality to upload data through CSV files can be introduced. This will enable users to easily import large volumes of data, such as product information, stock levels, or supplier details, without having to manually input each record.

In conclusion, the future scope of the inventory management system includes enhancements that focus on improving usability, visualization, reporting capabilities, and data handling. By addressing these aspects, the system can become even more valuable and effective in supporting businesses' inventory management processes and decision-making.