IMBY

Group number: 10
Group members: Marcel Salazar, Axel Torres, Brennan Gilbert, Jackson Davidson, Noah Loving, Emannuel Barcenas

Introduction

Our team has identified a critical gap in the market: the lack of a universally applicable, streamlined management system for data storage. In response, we have developed a Database Management System (DBMS) that is both versatile and user-friendly, catering to a wide range of users, from professionals to individuals managing personal data. Our product, IMBY (Inventory Management By Yourself), is specifically engineered to enhance the operational efficiency of businesses and self-employed individuals. IMBY simplifies inventory management by offering a solution that eliminates the usual stress associated with such tasks, without the complexity often found in advanced software. Our goal is to facilitate better inventory control and data management, thereby improving overall business efficiency.

User requirements

Typically describes (or use use-case diagram to demonstrate) who are the users, what they would like to use your system to complete their tasks. Should be as details as helping to generate what relations are to be needed. Do not just provide few general statements.
E.g.,
There are 1 types of users in this system.

The system should provide a secure and efficient means of storing sensitive personal information for both customers and employees, including identification numbers, names, phone numbers, and addresses. A pivotal feature of this system is the ability to distinguish between customer and employee profiles. In the case of employees, the system must allow for the input and tracking of salary information. Conversely, when managing customer profiles, the system should enable recording and maintenance of service history to ensure personalized customer engagement.

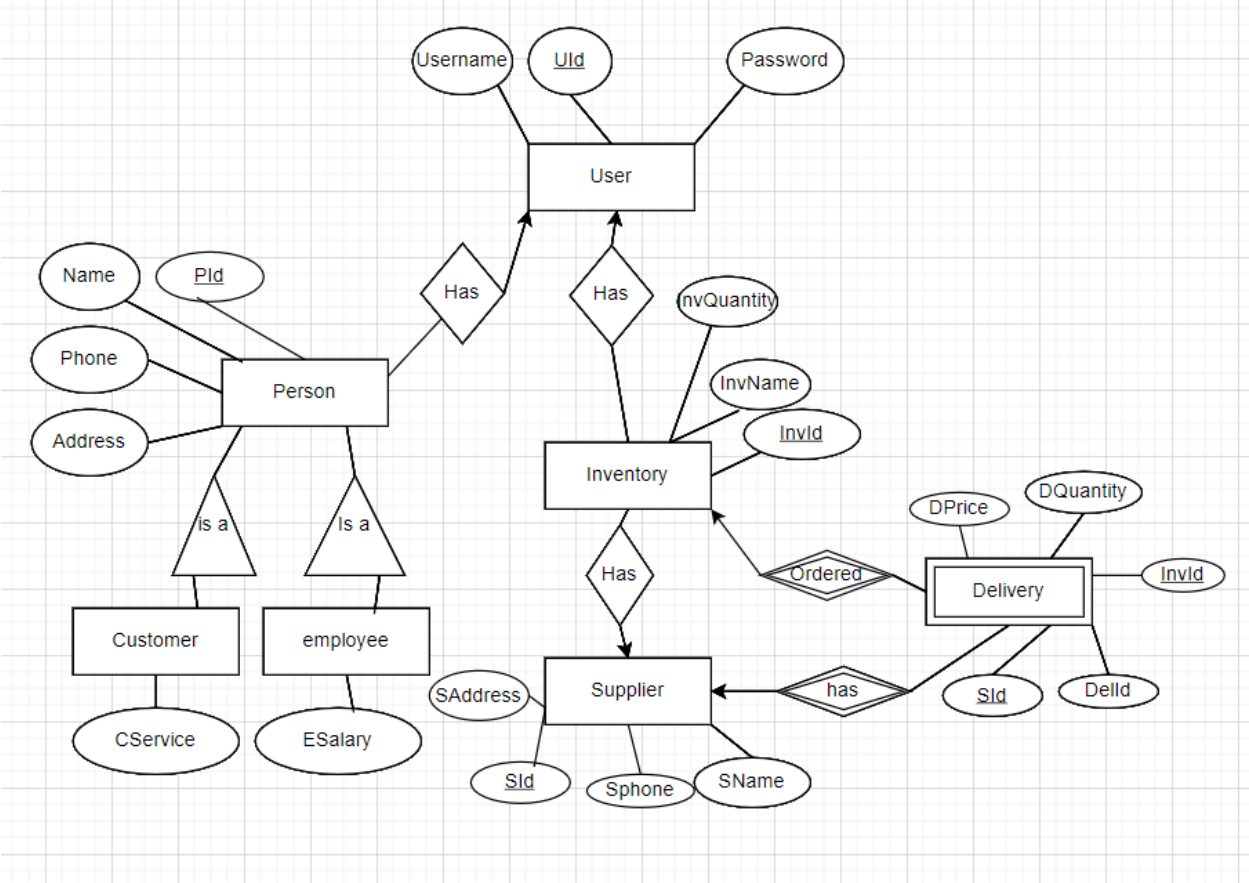
Another critical requirement is an advanced inventory management module. This module must be capable of tracking and updating inventory levels with high accuracy, documenting the flow of goods both into and out of the company's stock. Essential details such as item names, identification numbers, and quantities must be meticulously recorded to facilitate effective stock management. To enhance operational efficiency, the system must incorporate functionalities for adding, deleting, and modifying inventory records, ensuring that the data remains current and reflective of actual stock levels.

Lastly, the system must provide order management capabilities, designed to maintain orderly records of incoming inventory. It is imperative that the system allows for the documentation of supplier information, including names and phone numbers, as well as financial details such as the price of ordered items. This function is crucial for facilitating seamless procurement processes and for enabling accurate financial tracking and forecasting.

Entity relationship diagram

Provide the diagram and write descriptions about the entities (presenting one business flow of why each entity is important/needed, how it is related to the others).

Provide business assumptions if there are (those will impact your design, do not provide trivial ones).



User: This entity represents individuals who interact with the system, such as employees, managers, or system administrators. The User entity is vital as it forms the gateway for interaction with the business's system. It includes attributes like username, UID (User ID), and password, which are crucial for authentication and maintaining the integrity and security of the system. The User entity, through its UId, is associated with the Person entity and manages specific inventory items, as stated by the UId foreign key in the Inventory entity.

Supplier: This entity is associated with the external parties that provide products or services to the business. It is essential for maintaining the supply chain operations of the business. The Supplier entity includes the supplier's ID (SID), name (SName), and phone number (SPhone), which are important for managing orders, communication, and relationship management with the suppliers. The Supplier entity is connected to the Delivery entity through the SId foreign key.

Inventory: This entity represents the list of goods or products that are available for sale or use within the business. The Inventory entity is crucial for stock management, order fulfillment, and planning for procurement. It includes attributes such as inventory name (InvName), inventory ID (InvID), and quantity (nvQuantity), which are utilized to monitor stock levels, identify items, and manage inventory turnover. The Inventory entity is linked to the User entity via the UId foreign key, the Inventory entity also connects to Deliveries through the InvId foreign key.

Delivery: This entity allows the user to track and manage the logistics of product delivery. It contains information about each delivery, such as delivery ID (DelID), price (DPrice), and quantity (DQuantity), which are essential for ensuring timely and accurate delivery to customers. The Delivery entity integrates Inventory and Supplier entities through InvId and SId foreign keys.

Person: This entity serves as a superclass for specific types of people associated with the business, namely customers and employees. It stores basic personal information that is common to both, such as name, phone, and address, which is necessary for communication and identification purposes. As a superclass, the Person entity is linked to the User entity through the UId foreign key and gives to Customer and Employee entities via the PId foreign key.

Employee: This entity represents the staff of the business. It is a subclass of Person, inheriting personal attributes while also containing employee-specific information such as service (EService) and salary (ESalary), which are necessary for anything that fulfills the employees needs and keeping track of employees payment. The Employee entity inherits from the Person entity using the PId foreign key, which combines personal and employment information.

Customer: This entity represents the clients who purchase from the business. As a subclass of Person, it includes customer-specific information such as service preferences (CService), which is used to monitor and continue service to our customers Coming from the Person entity with the PId foreign key, the Customer entity allows for the access to services.

Relational schemas

Provide the relational schemas. Describe where some special situations/cases occur while mapping from ERD to this.

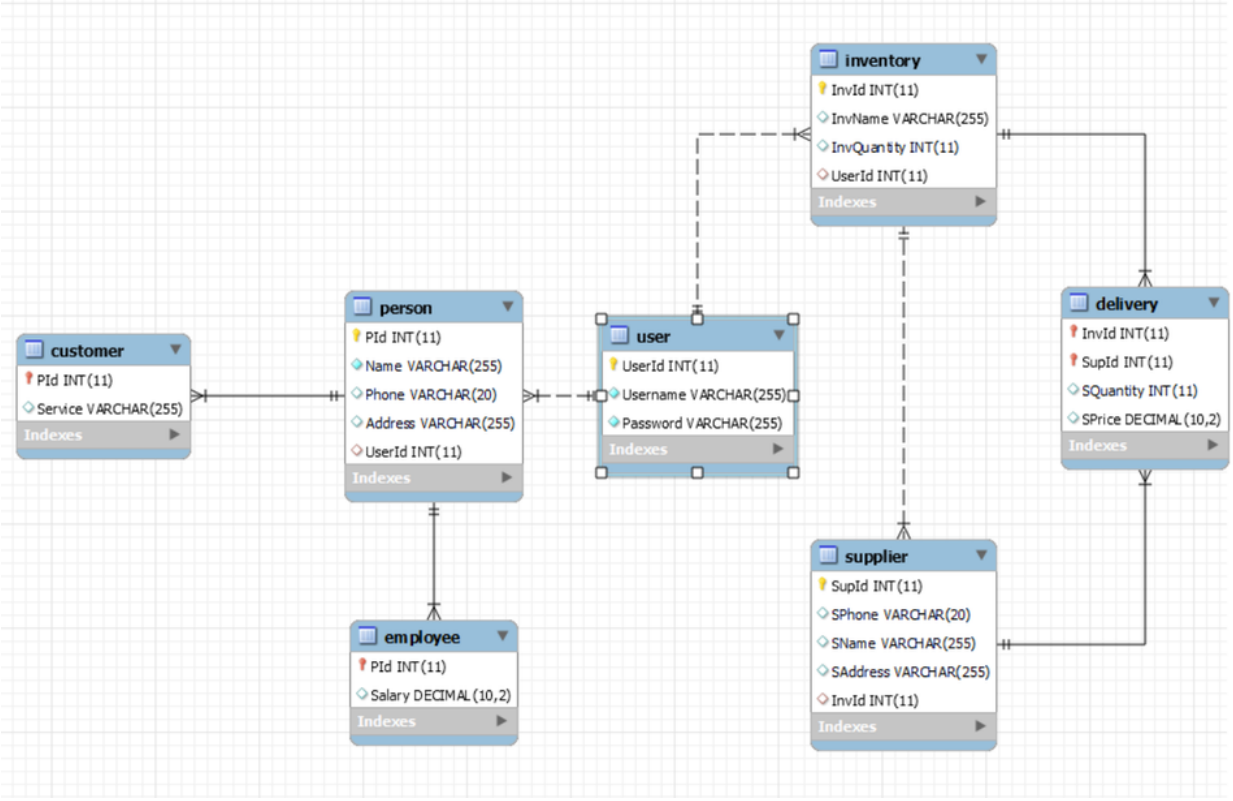
User(UId, Username, Password)
Person(PId, Name, Phone, Address, UId*) UId is a FK referencing User(UId)
Customer(Cservice, PId*) PId is a FK which references the superclass Person(PId)
Employee(Esalary, PId*) PId is a FK which references the superclass Person(PId)
Inventory(InvId, InvQuantity, InvName, UId*) UId is a FK which references User(UId)
Supplier(SId, SAddress, SPhone, SName)
Delivery(DPrice, DelId, Dquantity, InvId*,SId*) InvId* is a FK which references Inventory(InvID), SId* is a FK which references Supplier(SId)

Customer and Employee inherit from Person: This is a case of inheritance, where Customer and Employee are subclasses of Person. This means that they inherit all the attributes of Person, and they can also have their own unique attributes.

Delivery has two foreign keys: one referencing Inventory (InvId) and one referencing Supplier (SId): This is a many-to-many relationship, where a Delivery can be associated with multiple Inventory items, and an Inventory item can be associated with multiple Deliveries. Similarly, a Delivery can be associated with multiple Suppliers, and a Supplier can be associated with multiple Deliveries.

Generated database diagram

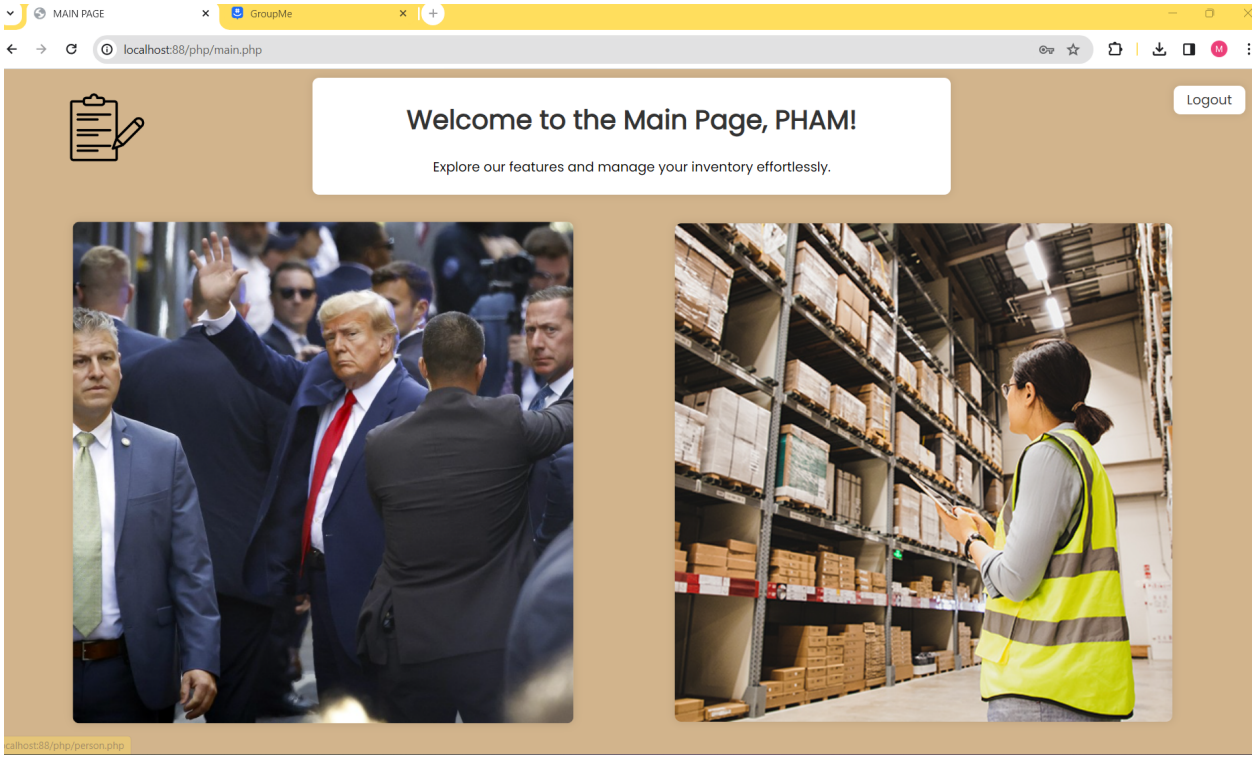
See the PPTX template for the steps for generating generated database diagram and place it here. Write explanations regarding discrepancies or confirm this diagram vs. the designed relational schemas.

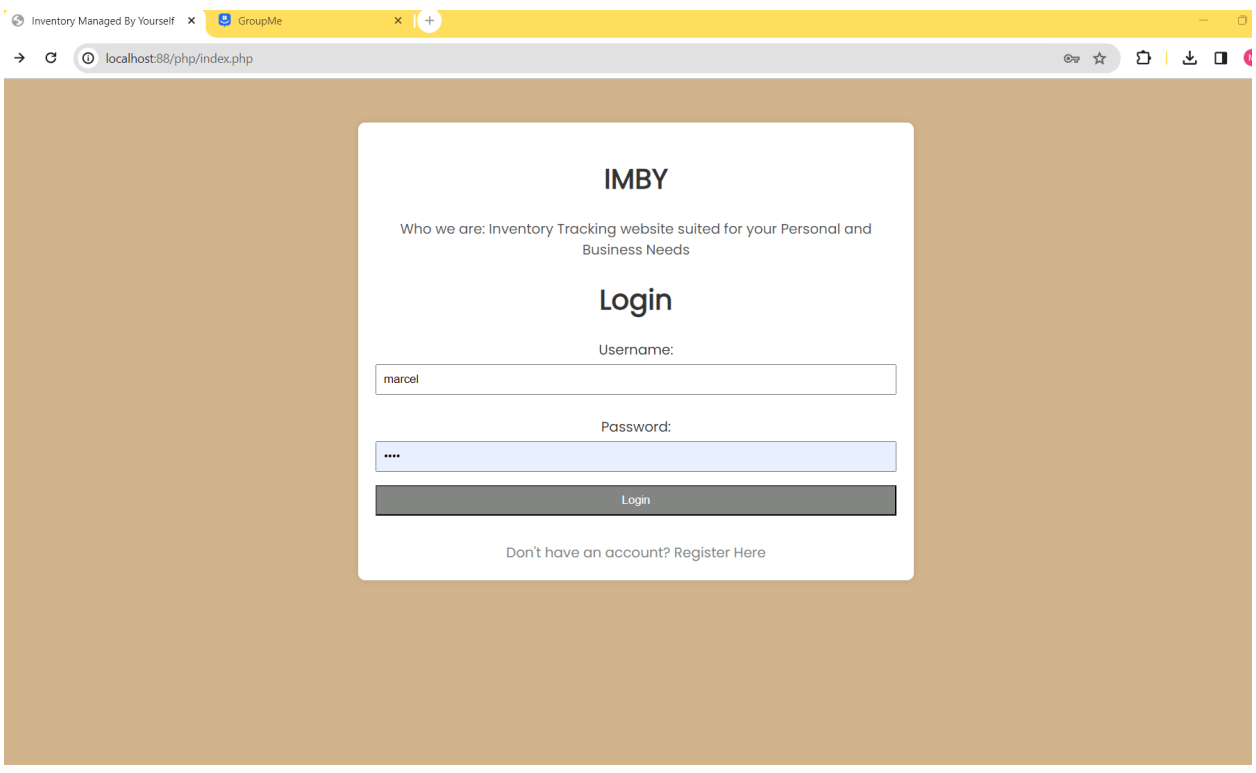


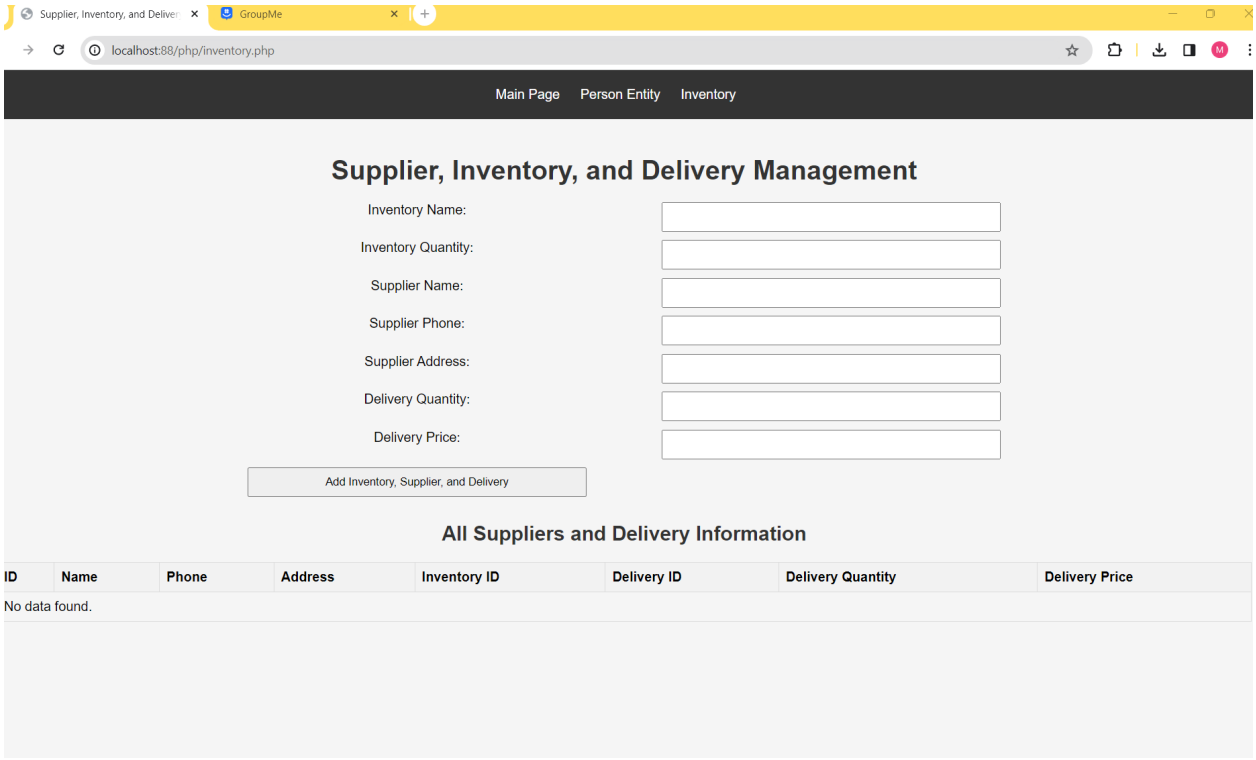
This Generated database diagram is 100% correctly represented compared to our designed relational schemas

Database application

Technologies used: HTML, PHP, CSS, JavaScript, and MySQL







Security concerns

Discuss about concerns such as Authentication; Authorization; Password encryption (e.g., MD5); SQL Injection (e.g., using PreparedStatement or do not allow multiple queries).

1. Protecting user passwords was an important concern of ours and we wanted to make sure our users felt secure when using the site. To achieve this, we utilized PHP's password_hash() function. This function is designed to convert the password entered by the user into a hashed format, which is far more secure. This is crucial because it means that the passwords are not stored in a format that can be easily deciphered. Once hashed, these passwords are then stored in our database. There is a big significance in implementing this because if there were ever a security breach, the attackers would encounter only the hashed passwords. These hashes are designed to be challenging to reverse-engineer making it unlikely for the original passwords to be compromised.

2. The process of verifying a user upon login is important. We achieve this through PHP's password_verify() function. During the login attempt, this function takes the password entered by the user and checks it against the stored hash in the database. It's a way to authenticate users without having to expose actual passwords at any point in the

process. This method ensures that user verification can be completed with confidence and without the risk of password interception or exposure during the login process.

3. To address the possible use of SQL injection, we implemented multiple layers of security. These attacks typically involve inserting malicious code into databases through user input fields, which leads to unauthorized access or even destruction of data. To prevent this, we ensure that all user inputs are validated. This means we examine every piece of information users enter on our site to confirm that it is legitimate and does not contain malicious code. This step is crucial in ensuring that only clean, expected data is processed by our system.

4. We also employed the use of prepared statements and parameterized queries when interacting with our database. Prepared statements separate SQL code from user data. We first set up the SQL command structure, essentially telling the database what kind of commands to expect and where the user's input should fit within that structure. Only then do we insert the user input into the pre-prepared placeholders. This method is highly effective because it allows the database to distinguish between code and data, regardless of what the user inputs. Through these measures, we provide a system that significantly reduces the risk of unauthorized data access through SQL injection, maintaining the security of our inventory management system.

Future work

What is missing in this current version, and you would like to further develop in the future to make the application more deployable if you would have a longer time.

1. We are planning to develop a mobile application that will seamlessly integrate with our inventory management system. This app will be designed to provide users with the flexibility to access and modify their account details on-the-go. With real-time access, users can update or retrieve any piece of information within the system instantly, enhancing the efficiency of managing inventory levels, order tracking, and data accuracy from any location at any time.

2. To fortify the security of our inventory management system, we will implement a policy requiring users to reset their password every six months. Alongside this, we will introduce two-step verification for an additional layer of security. This will reduce the risk of unauthorized access, ensuring that inventory data remains secure and that only authenticated users can make changes or view sensitive information.

3. Each user of our inventory management system will be assigned a unique private key. This key will serve as a failsafe for account recovery, allowing users to regain access to their accounts in the event they forget their login details. This measure ensures that users have a secure method to recover their accounts without compromising the integrity and security of the overall system.

4. Recognizing the dynamic nature of inventory data, we will add a feature that allows users to edit their information within the inventory management system. This functionality will be critical in correcting any incorrect entries or updating information to reflect real-time changes in inventory, ensuring that the system maintains high accuracy and reliability.

5. We aim to enhance our inventory management system by implementing a robust feature that enables comprehensive tracking and management of sales directly through our website. This integration will not only streamline the sales process but also provide valuable insights into sales trends and inventory demands. Concurrently, we are committed to updating the website's interface to prioritize user-friendliness. We envision a design that is intuitive, responsive, and accessible, ensuring that users can navigate and interact with the system with ease and efficiency. This redesign will focus on simplifying workflows, improving the layout, and incorporating visually engaging elements that align with our commitment to providing a superior user experience.

Conclusion

What you have done.

In this project, we successfully implemented an inventory management database system utilizing a combination of PHP, HTML, CSS, JavaScript, and SQL. The system effectively manages inventory items, including their quantity, name, and associated user. The project involved several key phases, such as database design utilizing SQL, backend and frontend development employing PHP to integrate with HTML, CSS, and JavaScript, and unified system integration. Overall, the system effectively manages inventory data, provides a user-friendly interface, and ensures data security and integrity.

What you have learnt.

Through comprehensive instruction, we acquired the skills necessary to effectively manage databases. By adhering to relational table structures, we established a well-organized data architecture. Furthermore, we delved into the intricacies of data distribution, ensuring optimal data placement within the database. This expertise enabled us to optimize data retrieval efficiency and maintain database integrity.

What you would do differently if you could start over.

While PHP served as an adequate development language for this project, adopting a more stronger framework could have potentially streamlined the learning process. Considering the project's extended semester-long duration, integrating Entity-Relationship Diagramming (ERD) principles earlier in the semester would have allowed ample time for refining data models and ensuring their suitability for the project's requirements. Introducing ERD concepts at the tail end of the semester constrained the ability to fully capitalize on its benefits and potentially led to inefficiencies in the development process.

Work distribution

Please write this section carefully and have all group members agree with it. Normally students would have the same grades for the project. However, if some is not responsive to group meetings or does not contribute the grades will be impacted (reduced).

Every member contributed equally to the project. Brennan worked on the ERD and relational schema to ensure correctness. Emmanuel and Noah worked on the report and HTML portion of the database. Jackson worked on the presentation and PHP portion of the database code. Marcel excelled in working on and explaining the HTML, CSS, SQL, and JavaScript portion of the code. Axel worked on the presentation and HTML portion of the database code as well.