**SALES AND INVENTORY MANAGEMENT SYSTEM FOR BAKEY FLAKEY**

A Project Presented to the Faculty of

College of Engineering and Computer Studies

In Partial Fulfillment of the Requirements for Subject

**Software Engineering**

**By:**

Ylagan, Angelina A.

**Submitted to:**

Donalene N. Tocmo

S.Y. 2024 – 2025 / 2ND SEMESTER

**TABLE OF CONTENTS**

**INTRODUCTION**

**Project Context**

The proposed software is a Sales and Inventory Management System designed specifically for Bakey Flakey, a startup online food business specializing in baked goods such as cakes, bars, and desserts. The main purpose of the system is to streamline and automate the company's sales tracking and inventory monitoring, replacing the current manual and Excel-based methods that are prone to errors and inefficiencies. By implementing this software, the business will be able to maintain accurate records of daily sales, automatically update inventory levels after each transaction, and gain better visibility into product performance—ultimately improving decision-making and operational efficiency

The software's key features include real-time inventory tracking, automated sales recording, customer and order management, and data reporting tools for monitoring business performance. Each transaction will capture essential details such as customer information, items purchased, quantity, price, and date of purchase, while simultaneously updating inventory levels. These capabilities will allow the business to detect fast-moving products, identify low-stock items, and evaluate profit trends more effectively.

The primary users of the system will be small business owners, specifically those in the food and retail industries, who need an efficient yet easy-to-use solution for managing their sales and inventory. It is particularly beneficial for startups like Bakey Flakey that are experiencing growth and need scalable tools to keep up with increased demand and operations.

The system is developed using PHP, a widely-used general-purpose scripting language especially suited for web development, and MySQL, a reliable and robust relational database management system used to store and manage data securely. This technology stack ensures that the system is both efficient and scalable for small to medium-sized businesses.

**Objective of the Project**

**1.2.1 General Objectives**

* **The main objective of the proposed system is to create a program that will improve the monitoring of sales and inventory through upgrading excel into a computerized system for more and accurate counting.**
* In using the proposed system, the company can easily manage and monitor their sales and also, they can easily manipulate the records inside inventory.

**1.2.2 Specific Objectives**

* **To create security login.**
* Using this proposed system, it will provide username and password for more secure login process to avoid any unauthorized access.
* **To provide quickly scale for the inventory.**
* Using this proposed system, it will provide proper and organized list of products for the owner to easily manage.
* **To keep monitoring the stocks, orders and sales.**
* Using this proposed system, the user will automatically determine the real-time quantity status of the inventory per assigned products. The order will automatically deduct from the inventory and will automatically add to the sale of the company.

**Purpose and Description**

The purpose of the Sales and Inventory Management System is to provide **Bakey Flakey** with an efficient and reliable solution for tracking sales transactions and managing inventory in real time. The system is designed to eliminate the challenges and inefficiencies associated with manual record-keeping and Excel-based monitoring, which often lead to data entry errors, disorganized records, and time-consuming processes. This software allows the business to automate sales logging, update inventory levels instantly after each transaction, and generate reports that help in making informed business decisions. By implementing this system, Bakey Flakey can ensure accurate inventory control, improve order processing, reduce losses due to stock mismanagement, and ultimately enhance overall productivity and customer satisfaction. The main impacts on the beneficiaries are as follows:

* Owner – It helps them lessen the time consumption of checking the stocks and also let them track, monitor and improve their sales.
* Customer – Organized records will help them to get the correct order they purchased.
* Future Researchers – It will help them for their future references of their research
* Proponents – This study will help them by gaining more knowledge while working with the system and the documentation of it. It will also help them achieve their future goals and dreams.

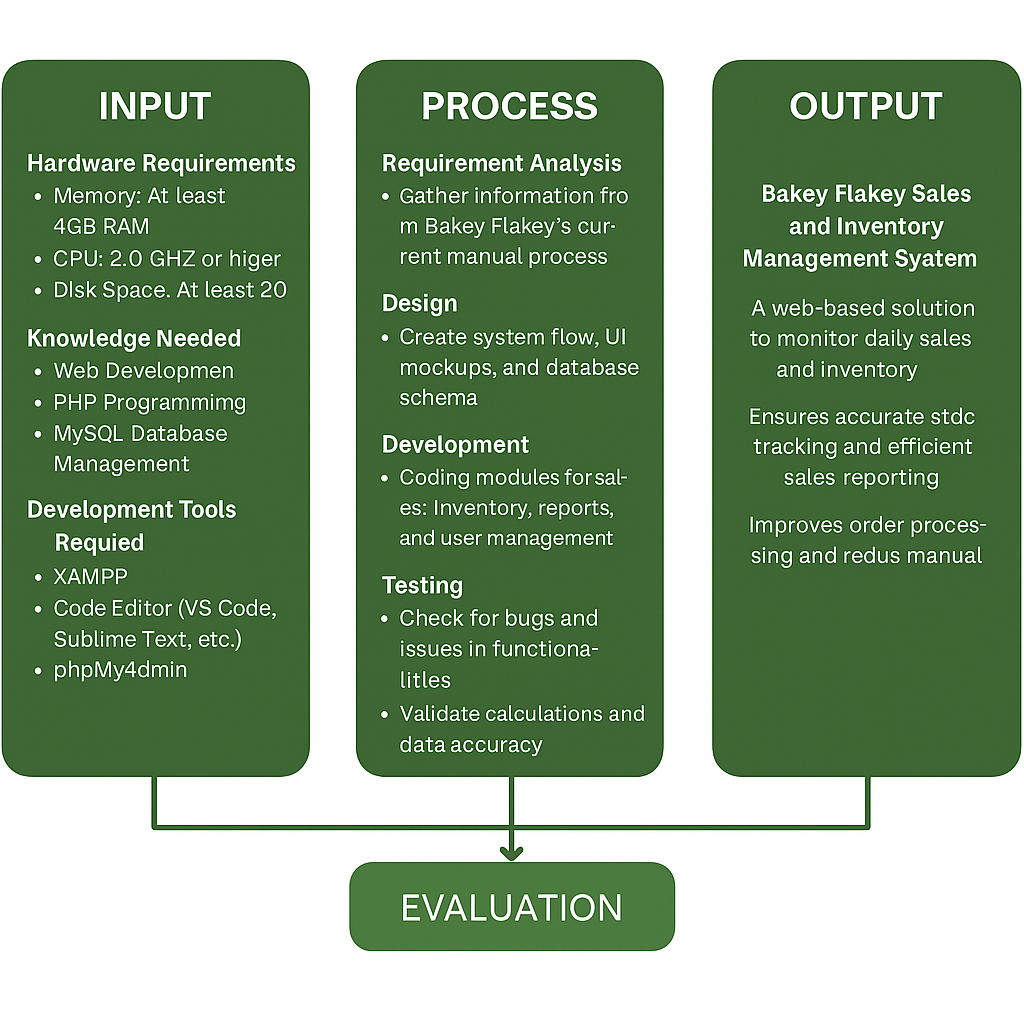
**Conceptual Framework**

Figure 1. Conceptual Framework

Figure 1 shows the conceptual framework of the project, and is made up of 4 sections that would describe the stages of the development of this System. First is the “Input” stage which contains all the gathering of the materials needed throughout the development, consists of both hardware and software requirements of the System as well as the knowledge that the developers need, and lastly, the development tools needed for this development. Next is the “Process” stage, which provides the step-by-step procedure of the development starting from planning, the gathering of information, and the actual creation of the webpages, up until the deployment of the website. Then there is the “Output” Stage, where the System is finally complete and is ready to be used by the beneficiaries, which in turn will be further evaluated for possible bugs and errors, which in turn will be fixed periodically.

**METHODOLOGY**

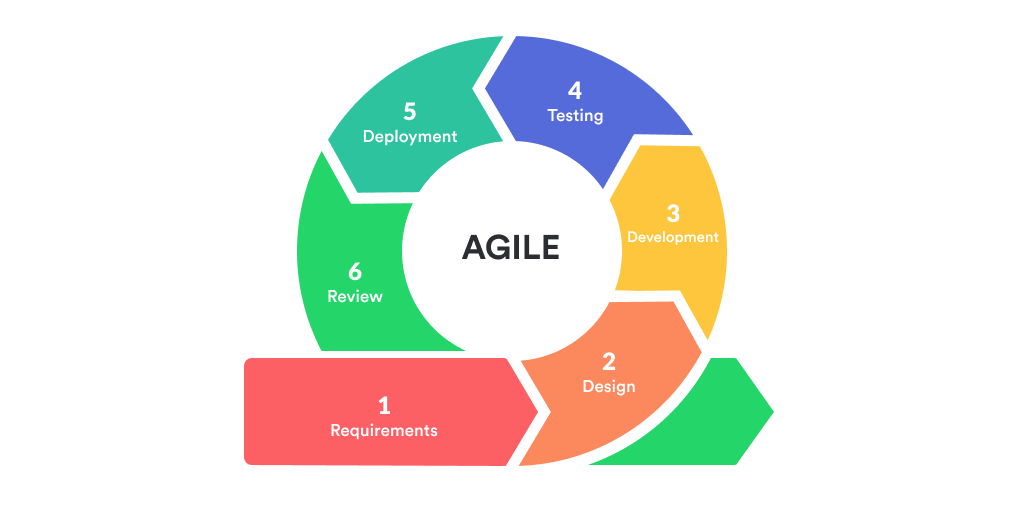


Figure 2. Agile Methodology

**Requirements Phase**

In the Requirements Phase, the proponents collaborate with the client and stakeholders to gather the initial requirements for the system. This phase focuses on understanding the core needs, features, and functionalities that the client expects from the system. The proponents prioritize these features, creating a product backlog that outlines the most important tasks and features. This phase sets the direction for the development process, but since Agile is iterative, requirements can evolve and be adjusted as the project progresses based on feedback from each sprint.

**Designing Phase**

In the Designing Phase, the proponents begin to design the architecture, user interfaces, and system components based on the requirements defined in the previous phase. This phase is iterative as well, and design decisions are revisited and refined in each sprint. The proponents work on creating wireframes, system diagrams, and technical designs for the system. The goal is to ensure that the system is scalable, user-friendly, and meets the client's needs. Design prototypes are also created, often in parallel with development, to give the client a sense of how the system will function.

**Developing Phase**

During the Developing Phase, the actual coding and implementation of the system take place. The proponents divide the system into smaller, manageable tasks and work on developing specific features and functionalities in short iterations (sprints). At the end of each sprint, the development team delivers a working version of the system, with each iteration building upon the previous one. The focus is on continuous delivery and improvement, ensuring that every increment adds value to the system.

**Testing Phase**

The Testing Phase runs in parallel with the development phase, as Agile emphasizes continuous testing throughout the entire project. After each sprint, the proponents test the features developed to ensure they meet the requirements and work as expected. Unit tests, integration tests, and user acceptance tests are conducted to identify and resolve bugs early. Feedback from testing is used to refine and improve the system in subsequent iterations. This phase ensures that the system is robust and of high quality, and that any issues are addressed quickly.

**Deploying Phase**

Once a set of features has been fully developed and tested, the system enters the Deploying Phase. In Agile, this is usually done incrementally, with smaller features or versions being deployed to the client or end-users at the end of each sprint. The proponents release the system in stages, allowing for continuous user feedback and quick fixes if necessary. This phase ensures that the system is operational and accessible to users as soon as possible, even while new features are still being developed.

**Review Phase**

At the end of each sprint, the proponents conduct a Review Phase, where the development team demonstrates the work completed during the sprint to the client and stakeholders. During the sprint review, the proponents present the developed features, gather feedback, and discuss any adjustments or changes required. This allows the client to provide input and ensures that the system is evolving in the right direction. The feedback collected in this phase helps prioritize work for the next sprint, aligning the development process with the client’s evolving needs.

.

**Requirements Specification**

**Functional Requirements:**

| **Feature/Module** | **Description** | **Details** |
| --- | --- | --- |
| **1. User Authentication and Security** | | |
| 1.1. Login System | | The system must provide a secure login functionality with a username and password. | | --- |  |  | | --- | | | User roles should be defined for administrators, sales staff, and inventory managers. Passwords must be securely stored (hashed) and support multi-factor authentication (optional). | | --- |  |  | | --- | |
| 1.2. Access Control | Different access levels should be granted based on user roles (admin, sales staff, inventory managers). | Unauthorized access attempts should trigger alerts or lock accounts after multiple failed login attempts. |
| **2. Inventory Management** | | |
| 2.1. Product List and Categories | The system should allow the owner to add, update, and delete products with details such as name, description, price, and quantity. | Products should be categorized (e.g., cakes, bars, desserts) for easier management. |
| 2.2. Real-Time Inventory Tracking | The system must automatically update inventory levels after each sale. | When products are sold, inventory levels should decrease accordingly. The system should notify users when stock levels are low. |
| 2.3. Stock Adjustment | Admins should be able to manually adjust inventory levels for reasons like restocking or correcting errors. | Adjustments must be recorded with the date, reason, and user performing the adjustment. |
| **3. Sales and Order Management** | | |
| 3.1. Order Entry | Sales transactions should capture customer details (name, contact info), product purchased, quantity, price, and date of purchase. | Each transaction should automatically update inventory and sales data. Discount and promotional codes should be applied during the transaction if applicable. |
| 3.2. Order Processing | | The system should track the status of each order (e.g., pending, completed, canceled). | | --- |  |  | | --- | | | Customers should be able to track the status of their orders. | | --- |  |  | | --- | |
| | 3.3. Sales Recording and Reports | | --- |  |  | | --- | | | The system should automatically record each sale in the database. | | --- |  |  | | --- | | | Generate daily, weekly, and monthly sales reports, showing total revenue, products sold, and other metrics. Reports should be downloadable in CSV or PDF format. | | --- |  |  | | --- | |
| **4. Reporting and Analytics** | | |
| | 4.1. Product Performance Reports | | --- |  |  | | --- | | | The system should provide reports on the performance of individual products. | | --- |  |  | | --- | | | Reports should help the owner identify top-selling and low-performing items. | | --- |  |  | | --- | |
| | 4.2. Sales Trends and Profit Analysis | | --- |  |  | | --- | | | The system must generate profit analysis reports by comparing sales data with inventory costs. | | --- |  |  | | --- | | | Provide insights into sales trends over time to help in forecasting demand and setting pricing strategies. | | --- |  |  | | --- | |
| | 4.3. Inventory Reports | | --- |  |  | | --- | | | Generate real-time inventory reports that show the current stock level of each product. | | --- |  |  | | --- | | | Generate alerts when inventory levels fall below a minimum threshold to trigger restocking. | | --- |  |  | | --- | |
| **5. Customer and Order Management** | | |
| | 5.1. Customer Profile Management | | --- |  |  | | --- | | | The system should store and manage customer profiles with relevant details such as name, contact number, email, and order history. | | --- |  |  | | --- | |  |
| | 5.2. Order History | | --- |  |  | | --- | | | The system should allow users to view their past orders and track order statuses. | | --- |  |  | | --- | |  |
| | 5.3. Order Cancellation and Refunds | | --- |  |  | | --- | | | Users should be able to cancel orders within a certain time frame and request refunds if applicable. | | --- |  |  | | --- | |  |
| | 5.4. Notifications | | --- |  |  | | --- | | | The system should notify customers and administrators of important actions (order confirmations, stock updates, low inventory warnings). | | --- |  |  | | --- | |  |
| **6. User Interface and Usability** | | |
| | 6.1. Dashboard | | --- |  |  | | --- | | | A dashboard should be available for admins and managers to get a quick overview of sales, inventory, and customer activity. | | --- |  |  | | --- | | | The dashboard should be intuitive and display key metrics (e.g., total sales, stock levels, pending orders). | | --- |  |  | | --- | |
| | 6.2. Search and Filters | | --- |  |  | | --- | | | The system must provide easy search functionality to find specific products, orders, or customers. | | --- |  |  | | --- | | | Filters should be available to sort and categorize sales, inventory, and orders by date, status, and other criteria. | | --- |  |  | | --- | |
| | 6.3. Responsive Design | | --- |  |  | | --- | | | The system’s user interface should be responsive and accessible on various devices, including desktops, tablets, and smartphones. | | --- |  |  | | --- | |  |
| **7. System Integration** | | |
| | 7.1. Payment Gateway Integration | | --- |  |  | | --- | | | The system must integrate with a payment gateway to process payments for online orders. | | --- |  |  | | --- | | | Payment confirmations should be automatically linked to the order status. | | --- |  |  | | --- | |
| | 7.2. External System Integration | | --- |  |  | | --- | | | The system should allow future integration with other business systems (e.g., accounting software, e-commerce platforms). | | --- |  |  | | --- | |  |
| **8. Data Backup and Recovery** | | |
| | 8.1. Data Backup | | --- |  |  | | --- | | | The system should automatically back up data regularly to prevent data loss. | | --- |  |  | | --- | |  |
| | 8.2. Data Recovery | | --- |  |  | | --- | | | Provide a mechanism for recovering data in case of system failure or data corruption. | | --- |  |  | | --- | |  |
| **9. System Administration** | | |
| | 9.1. User Management | | --- |  |  | | --- | | | The admin should be able to create, modify, and delete user accounts. | | --- |  |  | | --- | | | Role-based access control should be in place to ensure users have appropriate access. | | --- |  |  | | --- | |
| | 9.2. Audit Logs | | --- |  |  | | --- | | | The system must maintain an audit trail of all user actions, including login attempts, data modifications, and transactions. | | --- |  |  | | --- | |  |
| **10. Performance and Scalability** | | |
| | 10.1. System Performance | | --- |  |  | | --- | | | The system must handle a high number of transactions without performance degradation. | | --- |  |  | | --- | |  |
| | 10.2. Scalability | | --- |  |  | | --- | | | The system must be scalable to accommodate growing product lines, order volumes, and additional users. | | --- |  |  | | --- | |  |

**Non-Functional Requirements:**

| **Criteria** | **Description** | | **Details/Justification** | | --- |  |  | | --- | |
| --- | --- | --- | --- | --- |
| | Performance Efficiency | | --- |  |  | | --- | | | Ensures system responsiveness within 2 seconds during typical operations. | | --- |  |  | | --- | | | Enhances productivity by minimizing delays and providing a smooth user experience, critical for time-sensitive inventory tasks. | | --- |  |  | | --- | |
| | Compatibility | | --- |  |  | | --- | | | Supports major web browsers, including Google Chrome, Mozilla Firefox, and Microsoft Edge. | | --- |  |  | | --- | | | Guarantees accessibility and usability across diverse devices and platforms, accommodating user preferences and reducing barriers. | | --- |  |  | | --- | |
| Reliability | | Ensures consistent uptime and error-free operations under normal conditions. | | --- |  |  | | --- | | | Prevents disruptions in inventory management, maintaining system trustworthiness. | | --- |  |  | | --- | |
| Usability | | Provides an intuitive interface with clear navigation for users of varying technical skills. | | --- |  |  | | --- | | | Simplifies interactions, allowing all users to perform tasks effectively without extensive training. | | --- |  |  | | --- | |
| Security | | Implements user authentication to restrict access and safeguard inventory data. | | --- |  |  | | --- | | | Protects sensitive data from unauthorized access, ensuring confidentiality and integrity. | | --- |  |  | | --- | |
| Maintainability | | Allows for easy updates, debugging, and scalability. | | --- |  |  | | --- | | | Ensures the system can adapt to future requirements and fixes with minimal downtime or complexity. | | --- |  |  | | --- | |
| Portability | | Designed to operate seamlessly across multiple platforms and devices. | | --- |  |  | | --- | | Facilitates mobility and use in various environments without additional configuration efforts. |

**System Requirements:**

| **Category** | **Description** | | **Details/Justification** | | --- |  |  | | --- | |
| --- | --- | --- | --- | --- |
| **Hardware Requirements** | | |
| Processor | | The system should run on a multi-core processor (e.g., Intel i5 or higher, AMD Ryzen 5 or higher). | | --- |  |  | | --- | | | A multi-core processor ensures smooth multitasking and better performance during peak load times. | | --- |  |  | | --- | |
| | Memory (RAM) | | --- |  |  | | --- | | | Minimum of 8GB of RAM, recommended 16GB or more. | | --- |  |  | | --- | | Adequate memory ensures smooth operation and the ability to handle multiple concurrent users or transactions. |
| Storage | | At least 250GB of SSD storage, with the option to scale as needed. | | --- |  |  | | --- | | | SSD storage offers faster data access speeds and provides quick response times. | | --- |  |  | | --- | |
| Network | | High-speed internet connection (at least 100 Mbps download and upload). | | --- |  |  | | --- | | | A fast internet connection ensures quick processing and communication with external systems (e.g., payment gateway). | | --- |  |  | | --- | |
| | Backup System | | --- |  |  | | --- | | | External or cloud-based backup solution. | | --- |  |  | | --- | | | Ensures that critical data can be recovered in case of system failure. | | --- |  |  | | --- | |
| Display | | Minimum 1920x1080 resolution for clear visualization. | | --- |  |  | | --- | | | A higher resolution improves the usability of the system, especially for viewing large datasets and reports. | | --- |  |  | | --- | |
| **Software Requirements** | | |
| | Operating System | | --- |  |  | | --- | | | The system should support Windows, macOS, and Linux. For web-based access, all major operating systems are supported. | | --- |  |  | | --- | | | Cross-platform compatibility ensures the system can run on various devices and operating systems. | | --- |  |  | | --- | |
| | Web Browser | | --- |  |  | | --- | | | Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge (latest stable versions). | | --- |  |  | | --- | | | Ensures that the system is compatible with the most widely used browsers, providing accessibility across devices. | | --- |  |  | | --- | |
| | Database | | --- |  |  | | --- | | | PostgreSQL or MySQL for backend database. | | --- |  |  | | --- | | | These databases are reliable, scalable, and widely used for storing and managing large amounts of transactional data. | | --- |  |  | | --- | |
| | Web Server | | --- |  |  | | --- | | | Apache HTTP Server or Nginx for handling web traffic. | | --- |  |  | | --- | | | Ensures the server can manage HTTP requests efficiently, especially during peak usage times. | | --- |  |  | | --- | |
| Framework | The application should be built using modern frameworks such as Vue.js (for frontend), Node.js (for backend). | | Provides scalability, modularity, and ensures that the system can be updated or expanded with ease in the future. | | --- |  |  | | --- | |
| | Security Software | | --- |  |  | | --- | | | SSL/TLS encryption for secure communication between clients and the server. | | --- |  |  | | --- | | | Protects sensitive data (e.g., payment details, customer information) from being intercepted during transmission. | | --- |  |  | | --- | |
| | Payment Gateway | | --- |  |  | | --- | | | Integration with popular payment gateways such as PayPal, Stripe, or local payment services (e.g., GCash). | | --- |  |  | | --- | | | Ensures secure and reliable payment processing for customers purchasing items online. | | --- |  |  | | --- | |
| | Backup and Recovery | | --- |  |  | | --- | | | Cloud-based backup systems (e.g., Amazon S3, Google Cloud Storage). | | --- |  |  | | --- | | | Provides a scalable and secure solution for backing up system data regularly to avoid data loss. | | --- |  |  | | --- | |

**System Design**

**Architecture Overview:**

The system is designed to provide a simple yet elegant, user-friendly, and reliable experience. It utilizes a **Client-Server Architecture** to separate the user interface from the business logic and data storage, ensuring scalability and maintainability. This architecture enables the system to handle user requests efficiently while ensuring data persistence and security.

**Architecture Type:** Client-Server Architecture

* **Client-side:** The front-end is developed using HTML, CSS, and JavaScript, which are used to create a user-friendly interface.
* **Server-side:** The back-end is developed using PHP, which handles requests, processes business logic, and communicates with the database.
* **Database:** MySQL is used for storing data persistently. The server communicates with the MySQL database to retrieve or store information.

**Design Patterns:**

To ensure a well-structured, maintainable, and scalable system, the MVC (Model-View-Controller) design pattern is implemented. This pattern separates the system into three distinct components, each handling a specific responsibility, promoting clear organization and ease of development.

**MVC (Model-View-Controller):**

* The **Model** handles data logic (e.g., database interaction via PHP).
* The **View** is the user interface, implemented in HTML, CSS, and JavaScript.
* The **Controller** acts as an intermediary between the Model and the View, processing user input and updating the system accordingly.

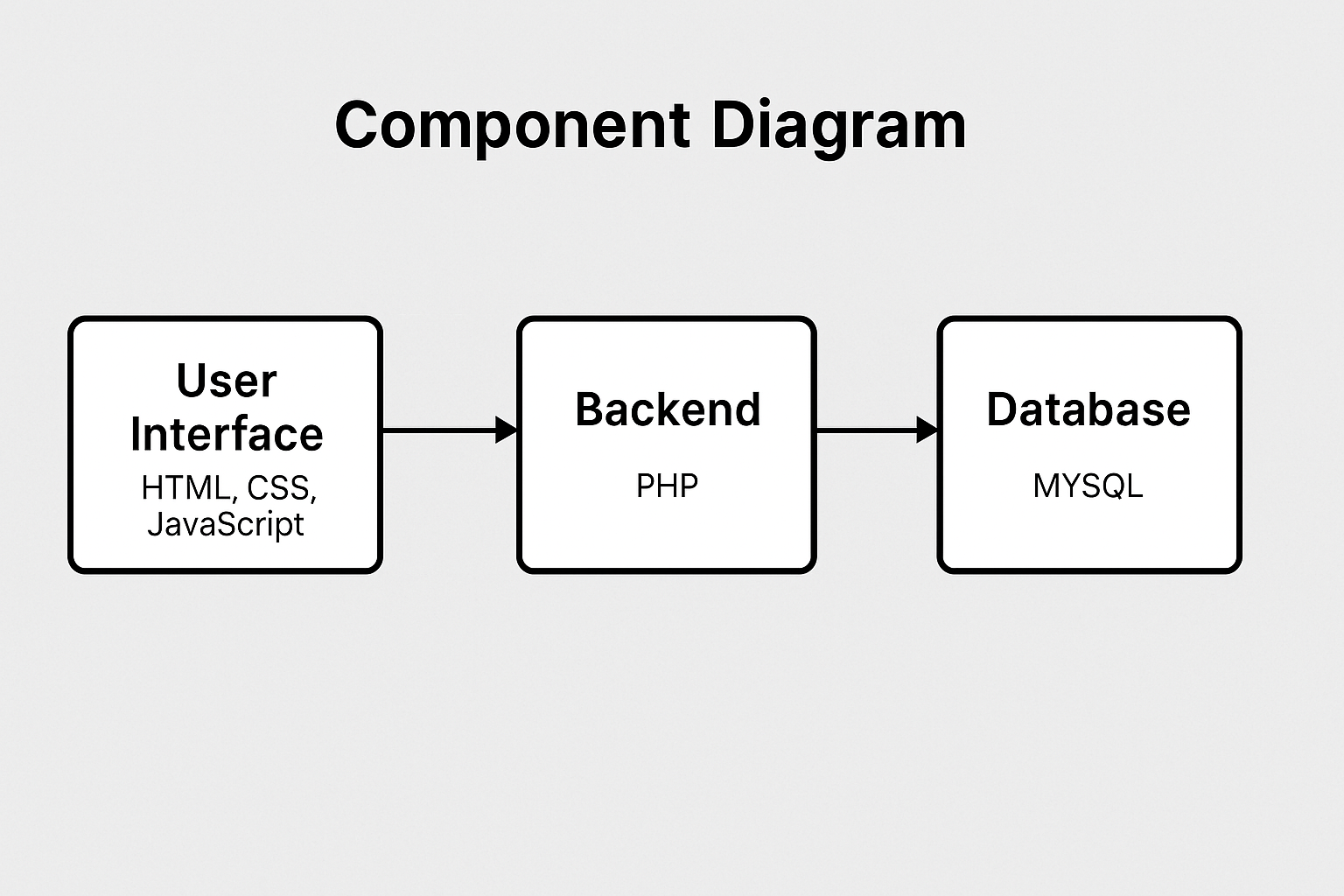


Figure 3. Component Diagram of Proposed System

**Database Design:**

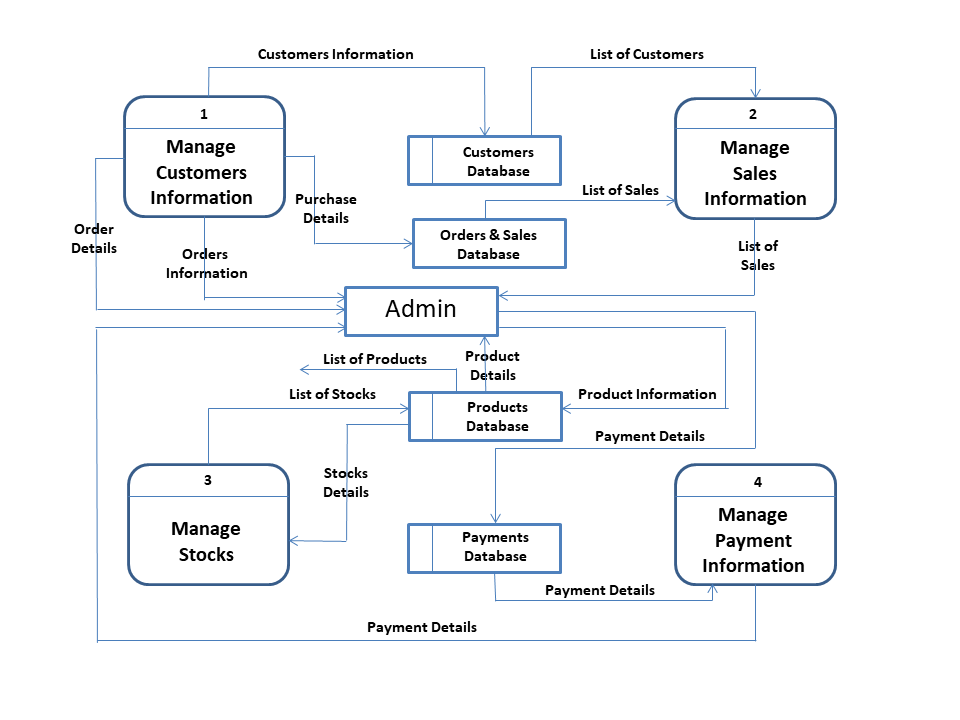


Figure 3. Data Flow Diagram of Proposed System

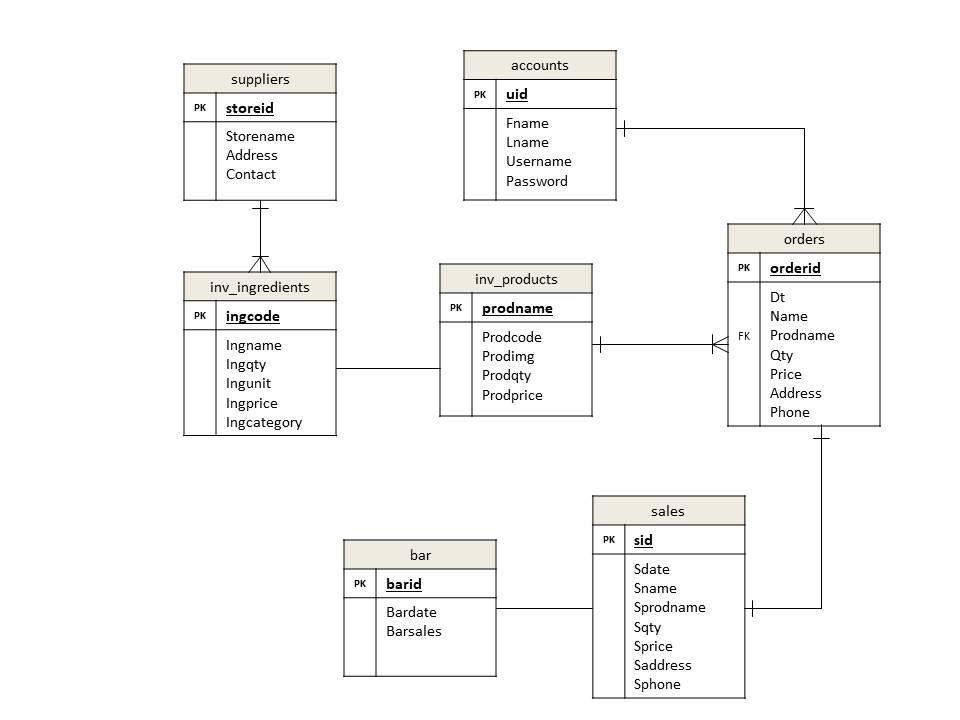


Figure 4. Entity Relationship Diagram of Proposed System

**User Interface Design:**

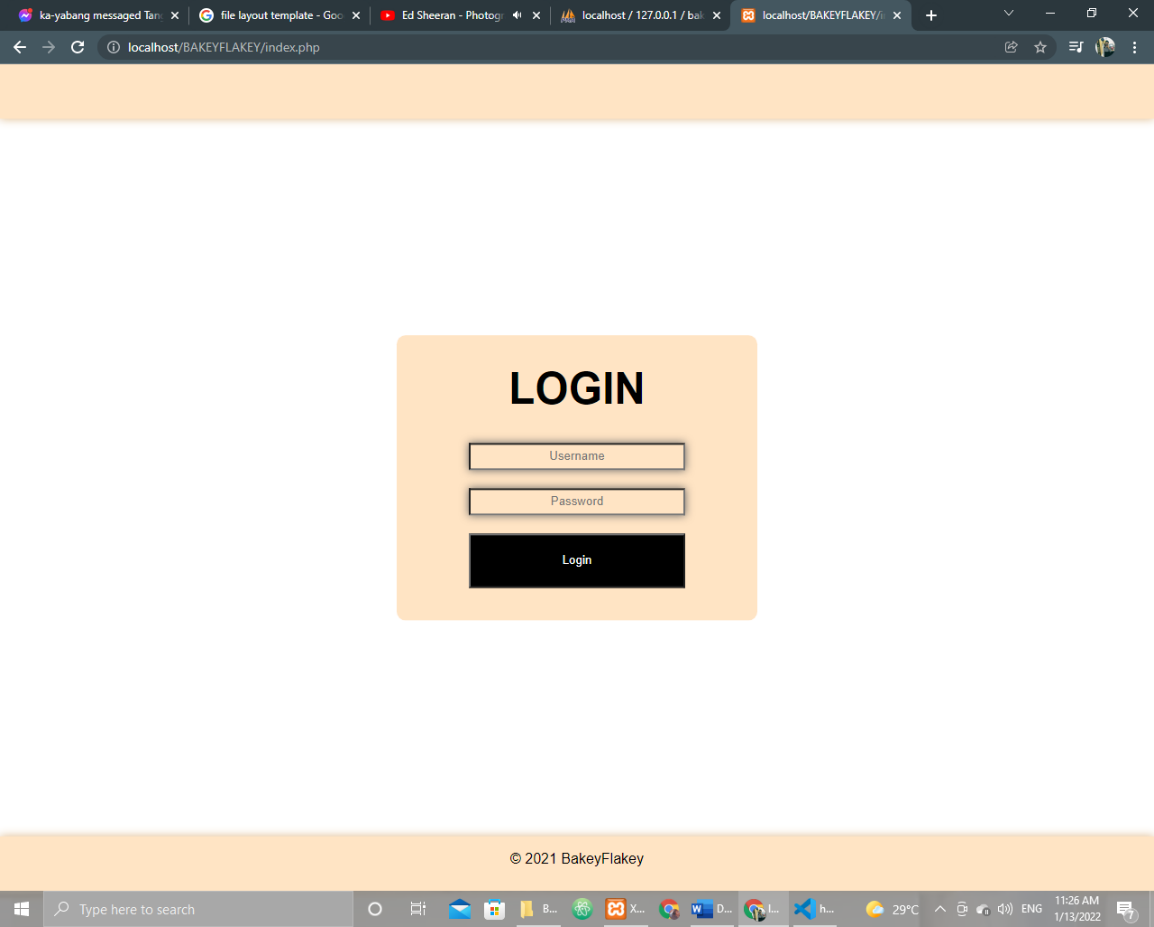


Figure 12. Admin Login Module Form

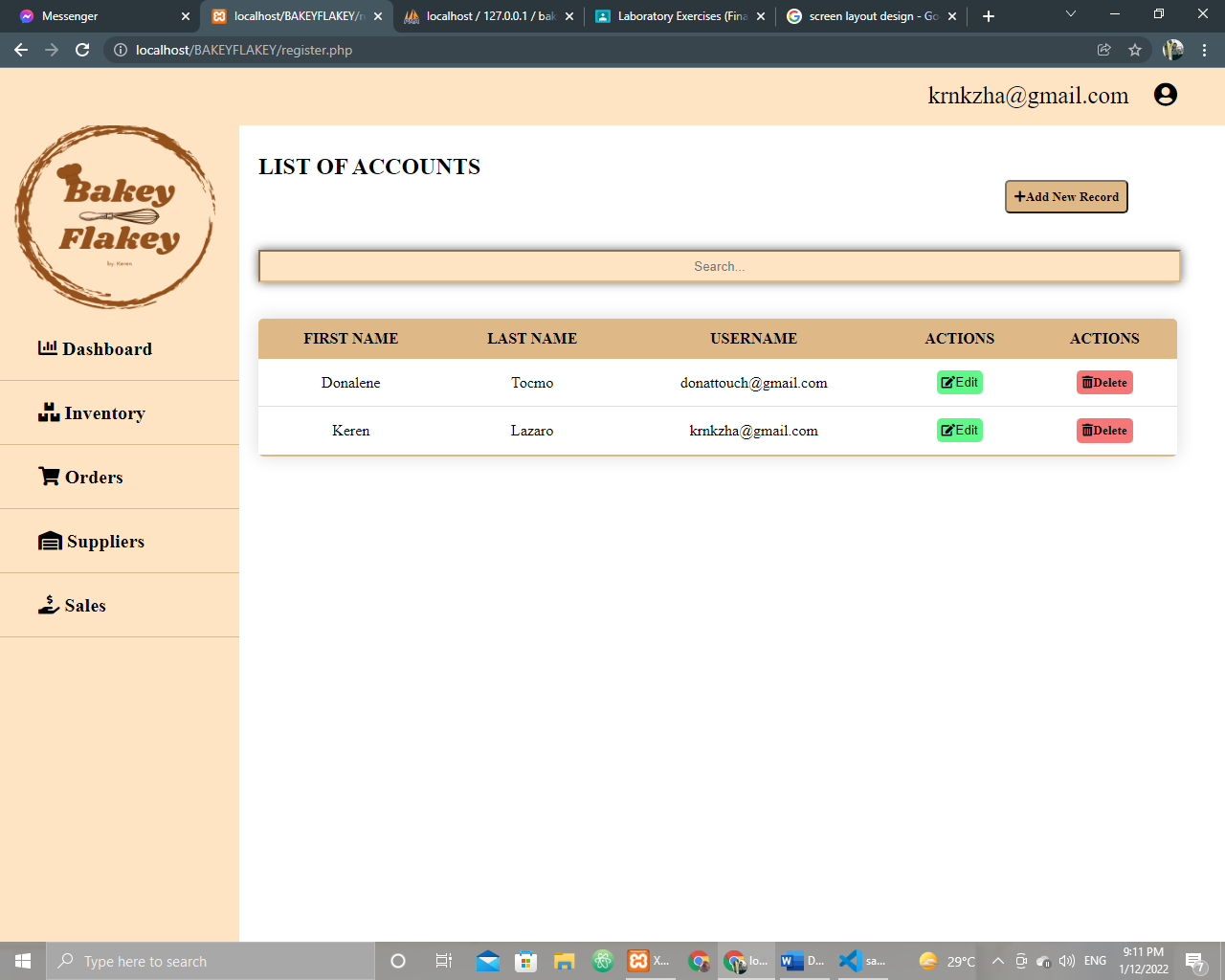


Figure 13. Accounts Module Form

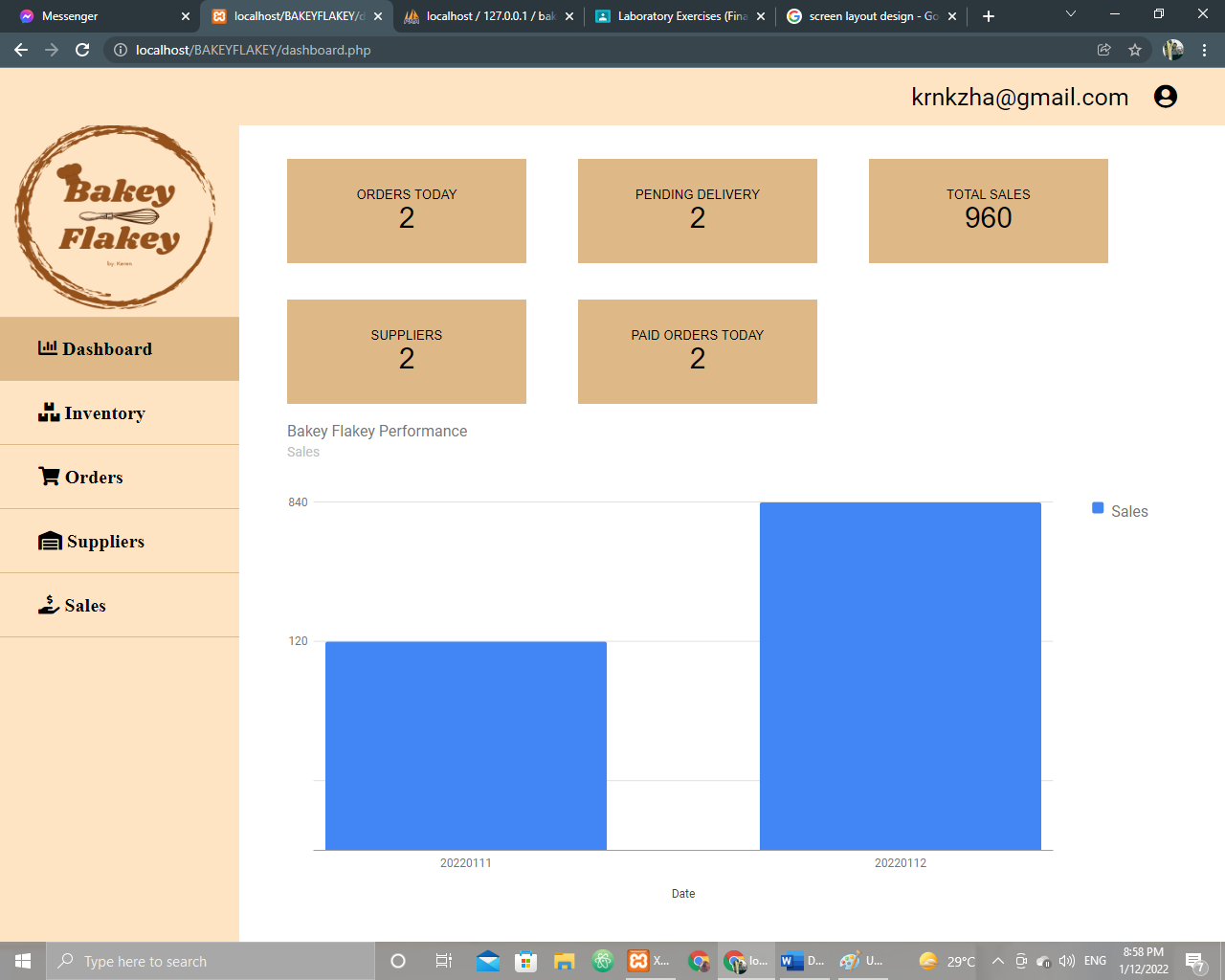


Figure 14. Dashboard Module Form

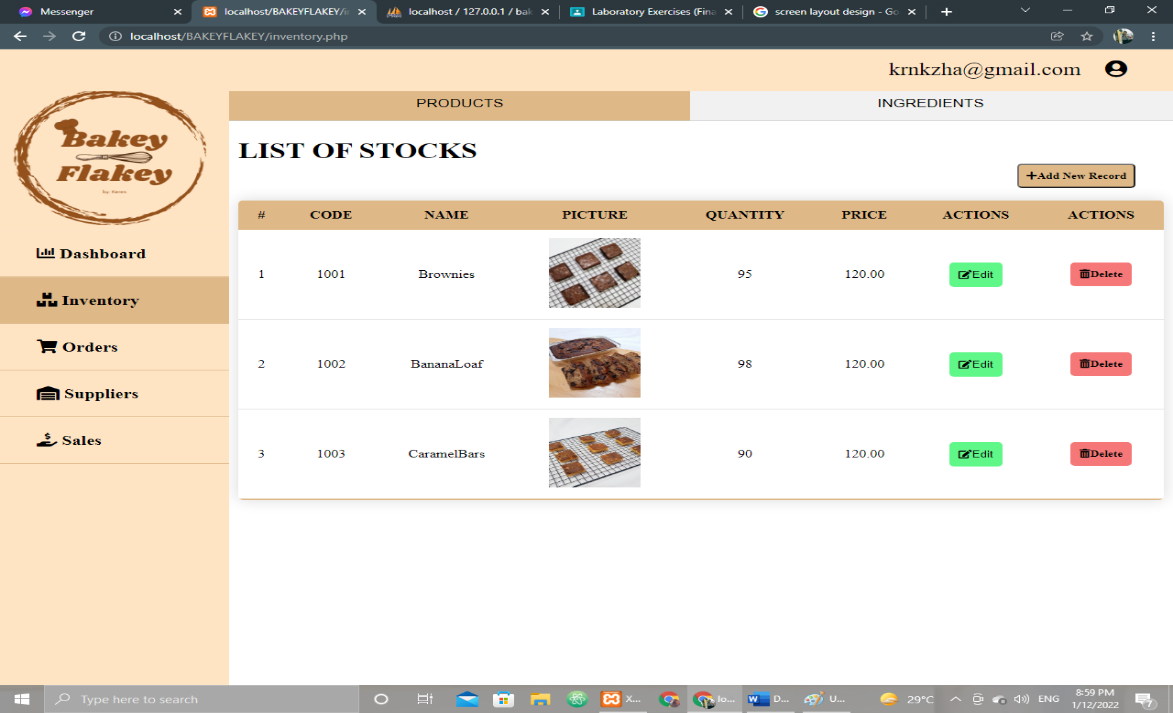


Figure 15. Inventory Module Form

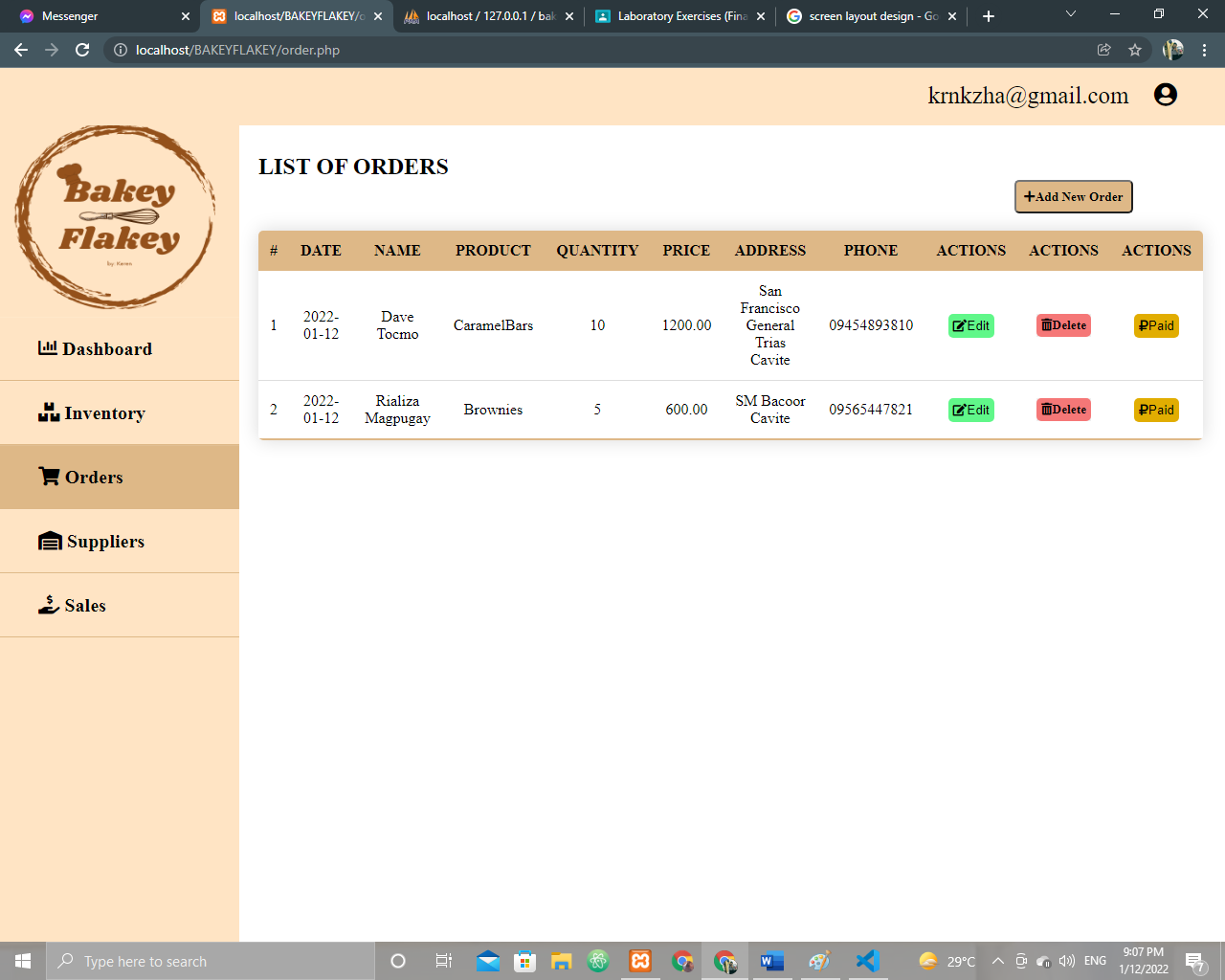


Figure 16. Orders Module Form

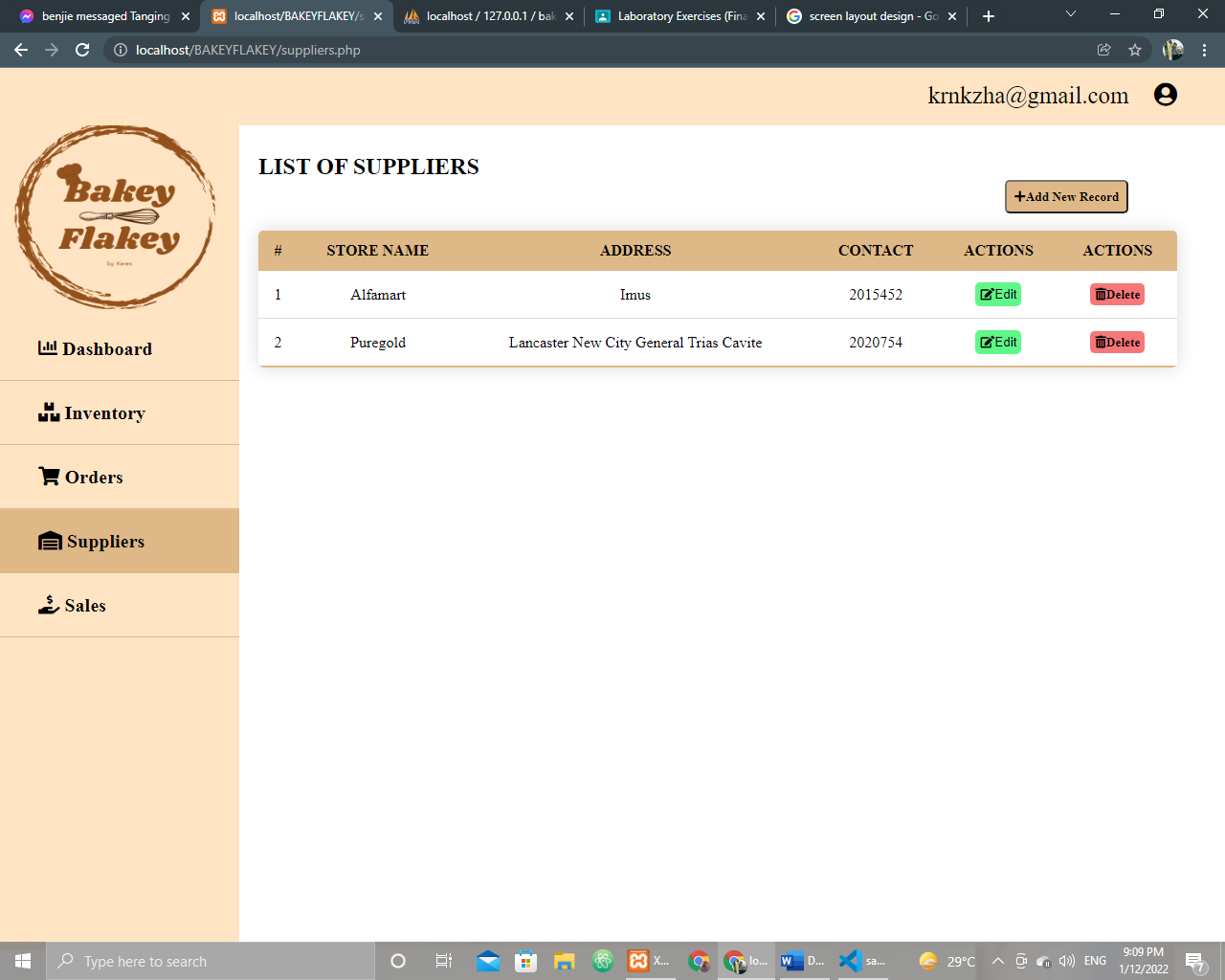


Figure 17. Suppliers Module Form

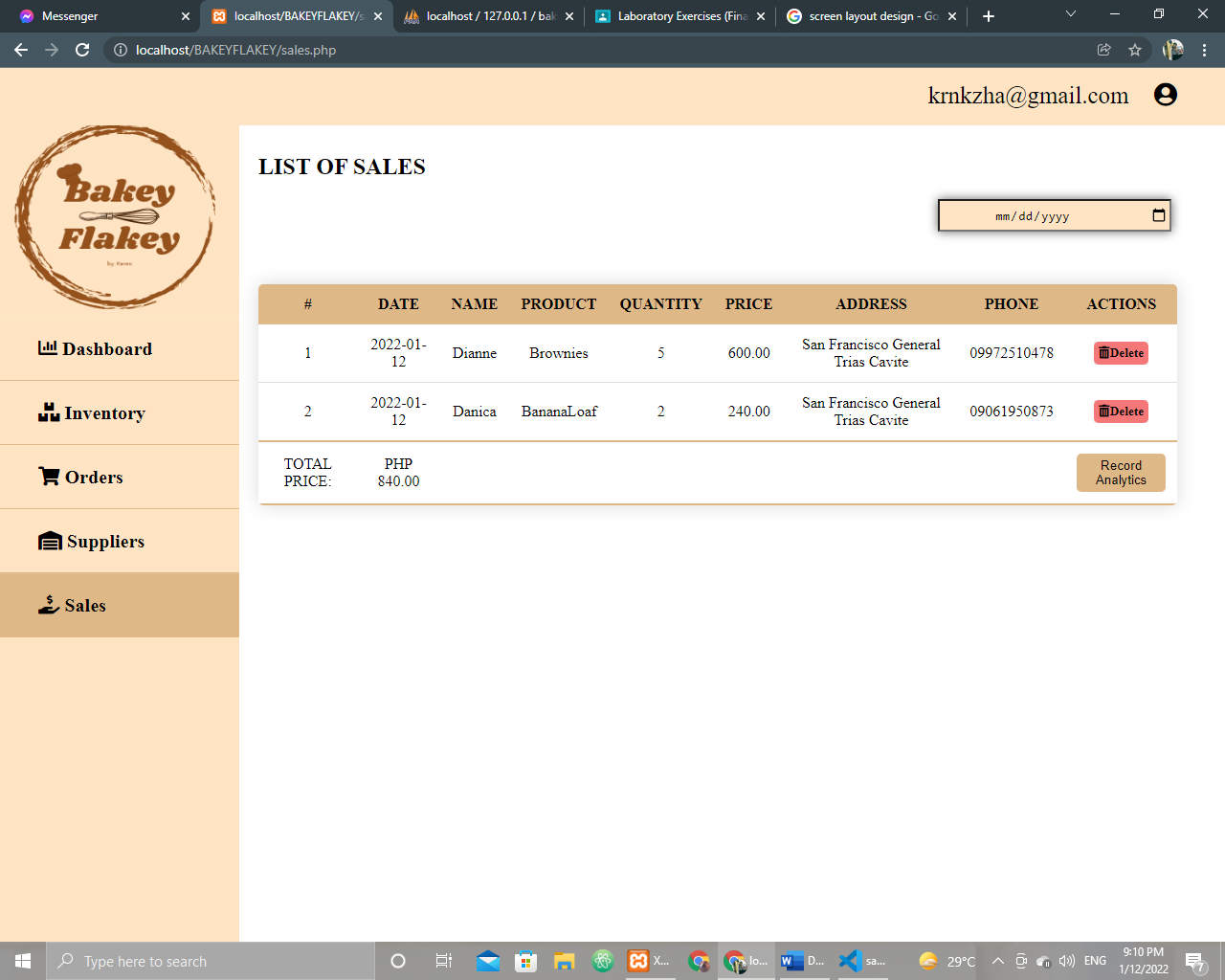


Figure 18. Sales Module Form

**Implementation Details**

**Module Overview:**

* **Sales Module**: Handles all sales transactions, including customer details, items purchased, quantities, and payment processing. This module is responsible for capturing the details of each sale and updating inventory levels accordingly.
* **Inventory Module**: Manages stock levels and updates inventory after each transaction. This module also helps track low-stock items and triggers alerts when restocking is necessary.
* **Order Management Module**: Handles order details, including processing new orders, updating the status of orders, and generating invoices for customers.
* **Dashboard/Reporting Module**: Provides data analysis tools to monitor sales trends, product performance, and overall business health, enabling informed decision-making.
* **Account Module:** Manages User accounts and Admin accounts details.
* **Suppliers Module:** Handles supplier information and manages supply chain processes.

**Class Diagrams:**

Show the main classes and their relationships (for object-oriented design).

**Code Structure:**

* **/assets**: Contains static files such as images, CSS files, and JavaScript files used on the client-side.
* **/controllers**: Includes PHP files that act as the controller in the MVC pattern. These files handle the user input and interact with the models to update the views.
* **/models**: Contains PHP classes that represent the data structures (e.g., Sales, Inventory, Customer, etc.) and the business logic for interacting with the database.
* **/views**: Contains HTML templates and views, which are the user interfaces that display data to the end-user.
* **/config**: Stores configuration files, such as database connection settings, API keys, and environment variables.
* **/lib**: Contains utility libraries or functions that are reused across different modules (e.g., authentication functions, data processing utilities).
* **/tests**: Includes test files for unit and integration testing of individual modules and features.

**Key Algorithms:**

* **Inventory Update Algorithm**: When a sale is made, the inventory update algorithm is triggered. It checks the sold quantity of items, updates the corresponding product quantities in the inventory database, and flags items that are low in stock.
* **Sales Tax Calculation Algorithm**: For each sale, the system calculates applicable taxes based on the location of the customer and the product's tax category. This calculation is performed at the time of checkout.
* **Reporting Algorithm**: The reporting module runs algorithms to aggregate sales data, product performance, and customer data. It then formats this information into readable charts and tables for business analysis.

**Third-Party Libraries or Tools:**

* **PHP**: For server-side scripting and business logic processing.
* **MySQL**: For database management and storing sales, inventory, and customer data.
* **Bootstrap**: A front-end framework used for responsive and user-friendly UI design.
* **jQuery**: A JavaScript library used for simplifying DOM manipulation and AJAX requests.
* **PHPMailer**: For sending email notifications to customers and staff (e.g., order confirmation emails).
* **Chart.js**: A JavaScript library used for creating interactive data visualizations for reports.
* **PDO (PHP Data Objects)**: A database access layer used for interacting with MySQL databases securely and efficiently.
* **Composer**: A dependency manager for PHP that handles third-party libraries and ensures the project's dependencies are up-to-date.

**Testing**

**Testing Strategy:**

**Unit Testing:** During testing, individual modules were tested to ensure they functioned correctly and met requirements before being integrated. For example, the inventory addition module was tested to verify its accuracy in recording stock, updating inventory, and sending notifications. This testing helped ensure each module was reliable.

**Integration Testing:** Next, the interaction between system modules was tested to ensure they worked together seamlessly. For example, the transaction and reporting modules were tested to confirm inventory changes, like sales or additions, were reflected in reports. This phase focused on ensuring data integrity and system reliability during complex processes.

**User Acceptance Testing (UAT):** Finally, end-users tested the system in a real-world setting to evaluate its usability and performance. Feedback was gathered from laboratory staff and administrators on the system’s interface, responsiveness, and functionality. Adjustments were made based on their input to improve the system and meet user needs effectively.

**Test Cases:**

The following table outlines the test cases performed to verify the functionality of the system according to the specified functional requirements. Each test case is designed to validate key features such as user authentication, inventory management, order processing, reporting, system performance, and scalability. These test cases ensure that the system operates as expected, meets user needs, and provides a reliable and efficient solution across various modules and functionalities.

| **Test Case ID** | **Test Case** | **Test Case Description** | **Expected Result** | **Pass/Fail** |
| --- | --- | --- | --- | --- |
| TC-01 | Login System | Test secure login with valid username and password. | User successfully logs in. | Pass |
| TC-02 | Login System | Test login with invalid username and/or password. | User is denied access with an error message. | Pass |
| TC-03 | Access Control | Test different user roles (admin, sales staff, inventory managers) accessing restricted areas. | Users only access areas according to their roles. | Pass |
| TC-04 | Product List and Categories | Test adding, updating, and deleting products with name, description, price, and quantity. | Product is added, updated, or deleted successfully. | Pass |
| TC-05 | Real-Time Inventory Tracking | Test inventory levels update after each sale. | Inventory levels decrease after each sale. | Pass |
| TC-06 | Stock Adjustment | Test manual stock adjustments by admin. | Stock levels adjust as per the entered value. | Pass |
| TC-07 | Order Entry | est capturing customer details (name, contact info), product purchased, quantity, price, and date of purchase. | Order is recorded with accurate details. | Pass |
| TC-08 | Order Processing | Test tracking of order status (pending, completed, canceled). | Order status updates correctly based on transaction. | Pass |
| TC-09 | Sales Recording and Reports | Test automatic recording of sales in the database. | Sale is recorded in the database after each transaction. | Pass |
| TC-10 | Product Performance Reports | Test generation of performance reports for individual products. | Report accurately reflects product performance. | Pass |
| TC-11 | Sales Trends and Profit Analysis | Test generation of profit analysis reports by comparing sales data and inventory costs. | Report accurately calculates profit margins. | Pass |
| TC-12 | Inventory Reports | Test generation of real-time inventory reports showing current stock levels. | Report displays up-to-date stock levels. | Pass |
| TC-13 | Customer Profile Management | Test storing and managing customer profiles with name, contact number, email, and order history. | Customer profile is created, updated, and stored. | Pass |
| TC-14 | Order History | Test viewing past orders and tracking order statuses. | Past orders and their statuses are displayed correctly. | Pass |
| TC-15 | Order Cancellation and Refunds | Test order cancellation within allowed timeframe and request for refund. | Order is canceled and refund is processed if applicable. | Pass |
| TC-16 | Notifications | Test notifications for order confirmations, stock updates, and low inventory warnings. | Users receive timely notifications as expected. | Pass |
| TC-17 | Dashboard | Test dashboard overview for admins and managers showing sales, inventory, and customer activity. | Dashboard displays correct and updated data. | Pass |
| TC-18 | Search and Filters | Test search functionality for products, orders, or customers. | Relevant search results are displayed based on input. | Pass |
| TC-19 | Responsive Design | Test system responsiveness across desktops, tablets, and smartphones. | Interface adjusts properly to various screen sizes. | Pass |
| TC-20 | Payment Gateway Integration | Test integration with payment gateway to process online payments. | Payment is processed successfully through the gateway. | Pass |
| TC-21 | External System Integration | Test integration with external systems (e.g., accounting software, e-commerce platforms). | External systems sync data correctly. | Pass |
| TC-22 | Data Backup | Test automatic data backup process at regular intervals. | Data is backed up as scheduled without errors. | Pass |
| TC-23 | Data Recovery | Test data recovery mechanism after system failure or data corruption. | Data is restored correctly without loss. | Pass |
| TC-24 | User Management | Test creation, modification, and deletion of user accounts by admin. | User accounts are created, modified, and deleted as expected. | Pass |
| TC-25 | Audit Logs | Test audit trail to ensure all user actions (login attempts, data modifications, transactions) are logged. | All actions are accurately logged in the audit trail. | Pass |
| TC-26 | System Performance | Test system's ability to handle a high number of transactions without performance degradation. | System operates without lag or downtime during high traffic. | Pass |
| TC-27 | Scalability | Test system scalability by adding new products, increasing order volumes, and adding users. | System performs smoothly as it scales up. | Pass |

**Testing Tools:**

For testing, the proponents employed manual testing to thoroughly evaluate the system's functionality. This approach allowed for in-depth examination of each feature, ensuring all requirements were met and the application performed as expected. By manually interacting with the application, I was able to identify issues that automated tests might have missed, providing a comprehensive validation of the system.

**Bug Tracking**:

To manage and track any bugs discovered during testing, the proponents utilized Taiga as the primary bug-tracking tool. Taiga offered an organized platform for logging, prioritizing, and monitoring issues. Each bug was documented with relevant details, making it easy to track its progress and ensure timely resolutions. This streamlined the bug-fixing process and ensured transparency throughout the development cycle.

**Deployment**

**Deployment Steps:**

The software is deployed through a multi-step process that involves setting up the necessary environments and configuring servers or cloud services. The deployment process begins with the preparation of the production environment, including the configuration of servers or cloud instances. Once the environment is set up, the application is deployed and configured to run seamlessly on the designated platform.

**Continuous Integration/Continuous Deployment (CI/CD):**

To streamline the development process, automated CI/CD pipelines are used for building, testing, and deploying the software. The CI/CD process ensures that code changes are automatically tested and deployed to staging environments, reducing the chances of errors in production. This allows for faster iterations and quicker feedback, enhancing the overall development workflow.

**Environment Setup:**

Setting up the development, staging, and production environments involves several steps. The development environment is configured with the necessary tools, dependencies, and access to local services. The staging environment mirrors production to ensure that tests are conducted in a production-like setting. The production environment is carefully configured for optimal performance and security, ensuring that the software runs smoothly for end users.

**Maintenance and Future Work**

**Ongoing Maintenance:**

The software will undergo continuous maintenance through regular updates, bug fixes, and performance optimizations. Monitoring tools will be implemented to track system health and identify potential issues. As part of the maintenance process, security patches will be applied, and any identified vulnerabilities will be addressed promptly to ensure the software remains secure and reliable.

**Known Issues:**

There are several known limitations and bugs that still require attention. These include:

* Online Payments
  + Not an online transaction
  + No credit card transaction
* Back Up and Security
  + Back up is not online

These issues are tracked and will be prioritized for resolution in upcoming versions, ensuring that they are addressed as part of the ongoing maintenance efforts.

**Future Enhancements:**

Future versions of the software may include a range of improvements and new features. Potential enhancements include

* Barcode and QR Scanner
* Integration with AI

which will aim to improve user experience, performance, and scalability. These enhancements will be evaluated and implemented based on user feedback and evolving needs.

**User Manual**

**Installation Instructions:**

Since the application is accessible online, there is no need for a local installation. To begin using the software, follow these simple steps:

1. Open a web browser (e.g., Chrome, Firefox, Safari).
2. In the address bar, enter the provided URL (e.g., https://www.example.com).
3. If required, sign in with your credentials or create a new account.
4. Once logged in, you can start using the application directly from your browser. No installation or setup is needed on your local machine.

**User Guide:**

Once you have accessed the application online, follow this guide to navigate and use the software effectively:

1. **Dashboard Overview:**  
   After logging in, you will land on the main dashboard. Here, you'll find an overview of your tasks, recent activities, and important notifications.
2. **Navigating the Application:**
   * **Menu Bar:** On the left-hand side, you will find a menu bar with options like "Home," "Settings," "Reports," and "Profile."
   * **Sections:** Click on any section to explore its features, such as viewing reports, updating settings, or managing your account.
3. **Using Key Features:**
   * **Feature 1:** To use the main feature, click on the “Feature 1” option from the menu. This allows you to [describe the action performed, e.g., create a report, submit a request, etc.].
   * **Feature 2:** To access additional tools, navigate to the “Feature 2” section. Here, you can [describe the action performed, e.g., manage user settings, track activity, etc.].
4. **Help and Support:**  
   If you encounter any issues, click on the “Help” button in the top-right corner for troubleshooting tips, or reach out to customer support via the live chat feature.

**Conclusion**

The proposed Sales and Inventory Management System for Bakey Flakey addresses the key challenges of manual sales tracking and inventory monitoring, offering a more efficient and reliable solution. By automating sales transactions and real-time inventory updates, the system improves accuracy, reduces the risk of errors, and enhances operational efficiency. The integration of security features, organized inventory management, and sales monitoring will provide the business with the tools necessary to streamline operations, improve decision-making, and ensure customer satisfaction.

With continuous maintenance and future enhancements planned, such as the integration of barcode scanning and AI, the system is designed to evolve with the business needs. The software's user-friendly interface and online accessibility further ensure that it can scale with the company's growth. As a result, Bakey Flakey will benefit from greater control over its sales and inventory, allowing for more efficient operations and ultimately contributing to its success and sustainability in the competitive food industry.

**References**

**Lambert, V. A., & Lambert, C. E. (1). *Qualitative Descriptive Research: An Acceptable Design.* Pacific Rim International Journal of Nursing Research, 16(4), 255-256. Retrieved from**

[**https://he02.tci-thaijo.org/index.php/PRIJNR/article/view/5805**](https://he02.tci-thaijo.org/index.php/PRIJNR/article/view/5805)

**Nassaji, H. (2015) *Qualitative and descriptive research: Data type versus data***

***analysis*. Volume: 19, page(s): 129-132 Retrieved from**

[**https://doi.org/10.1177%2F1362168815572747**](https://doi.org/10.1177%2F1362168815572747)

**Cheung A.K.L. (2014) *Encyclopedia of Quality of Life and Well-Being Research*.**

**Springer, Dordrecht. Retrieved from**

[**https://doi.org/10.1007/978-94-007-0753-5\_2888**](https://doi.org/10.1007/978-94-007-0753-5_2888)

**Elijah Ngugi (2019) *Effects of Inventory Management Systems on Performance of***

***Manufacturing Companies in Eldoret Town, Kenya*, Retrieved from**

[**https://www.grin.com/document/458869**](https://www.grin.com/document/458869)

***Computerized Inventory System*. (2016, Aug 20). Retrieved from**

[**https://phdessay.com/computerized-inventory-system/**](https://phdessay.com/computerized-inventory-system/)

**Matillano, K. (2019) “Chapter 2 | Related Literature | Sales and Inventory System”**

**Retrieved from** [**https://itsourcecode.com/fyp/chapter-2-related-literature-sales-and-inventory-system/**](https://itsourcecode.com/fyp/chapter-2-related-literature-sales-and-inventory-system/)

**Audra, B. (n.d.) “Sales & Inventory Monitoring Systems” Retrieved from**

**https://smallbusiness.chron.com/sales-inventory-monitoring-systems-18660.html**

**Navarro, G. (2012) “SALES AND INVENTORY SYSTEM”**

[**https://www.coursehero.com/file/p3vch01/SALES-AND-INVENTORY-SYSTEM-By-Gemma-Navarro-2012-Computer-is-a-general-purpose/**](https://www.coursehero.com/file/p3vch01/SALES-AND-INVENTORY-SYSTEM-By-Gemma-Navarro-2012-Computer-is-a-general-purpose/)

**Lerdorf, R., Tatroe, K., & MacIntrye, P. (2006) Programming PHP. O’Reilly Media, Inc. Retrieved from**

[**https://books.google.com.ph/books?id=h-E1lVko-skC&dq=what+is+php+programming+language&lr=&source=gbs\_navlinks\_s**](https://books.google.com.ph/books?id=h-E1lVko-skC&dq=what+is+php+programming+language&lr=&source=gbs_navlinks_s)

**APPENDICES**