****

**March 2023**

**Pharmacy Management System**

*A Project Report Submitted*

*to*

**MANIPAL ACADEMY OF HIGHER EDUCATION**

*For Partial Fulfillment of the Requirement for the*

*Award of the Degree*

*Of*

**Bachelor of Technology**

*in*

**Computer and Communication Engineering**

*by*

**Tashmit Verma, Ashish Jha, Vansh Sahay**

**210953032, 210953040, 210953144**

*Under the guidance of*

Mrs. Sumith N Dr. Raghavendra A

Associate Professor – Senior Scale Associate Professor

Department of I&CT Department of I&CT

Manipal Institute of Technology Manipal Institute of Technology

Manipal, Karnataka, India Manipal, Karnataka, India

ABSTRACT

The Pharmacy Management System project is an essential tool for the effective management of a pharmacy. It provides a comprehensive platform to manage all aspects of the pharmacy, including inventory management, customer management, and sales tracking. By utilizing this system, pharmacists can efficiently manage their inventory, track sales, and provide personalized customer service to improve customer satisfaction. The system ensures that customers receive the right medication and services at the right time, which can enhance their overall experience. The Pharmacy Management System is a valuable tool for pharmacists, enabling them to provide better healthcare services to their customers. Future work for this project could focus on incorporating advanced features such as real-time inventory tracking, data analytics, and machine learning algorithms to further optimize pharmacy operations and improve patient care outcomes.

TABLE OF CONTENTS

|  |  |
| --- | --- |
| **CHAPTER NO.** | **CHAPTER NAME** |
| 1 | INTRODUCTION |
| 2 | PROBLEM STATEMENT |
| 3 | OBJECTIVES |
| 4 | DATABASE DESIGN |
| 5 | METHODOLOGY |
| 6 | RESULTS |
| 7 | FUTURE WORK |
| 8 | SUSTAINABLE DEVELOPMENT GOAL |
| 9 | ACM TAXONOMY |

INTRODUCTION

The Pharmacy industry is a highly regulated and rapidly evolving field that requires efficient management to ensure its success. The use of technology has transformed various aspects of the healthcare industry, including the management of pharmaceutical stores. The purpose of this project is to create a Pharmacy Management System that combines database design and user interface (UI) creation to optimize inventory management, sales tracking, and customer relationship management.

The database design encompasses tables for employee data, product information, and sales transactions, among other essential elements. The UI will provide a user-friendly interface that facilitates quick access to pertinent data and enables efficient data entry and retrieval. By combining these two components, effective management of prescription tracking, inventory levels, sales analysis, and customer interactions can be achieved. This project demonstrates the potential of integrating database design and UI creation to enhance Pharmacy management.

Through the development of a Pharmacy Management System, this project provides a solution for efficient and effective management of prescription tracking, inventory levels, sales analysis, and customer interactions, among other essential functions. This report presents the development of a Pharmacy Management System that streamlines Pharmacy management, making it easier and more effective. The project emphasizes the importance of utilizing technology to improve Pharmacy management practices in the ever-changing healthcare landscape

.PROBLEM STATEMENT

Pharmacies encounter numerous obstacles in managing their inventory, sales tracking, and customer record-keeping. Many still rely on manual systems that are time-consuming and prone to errors, resulting in poor customer service, stock shortages, and loss of revenue. To overcome these challenges, a digital management system is required to automate pharmacy operations, provide real-time inventory updates, and monitor sales data. This project aims to create a Pharmacy Management System that leverages database design and UI development to boost pharmacy efficiency and effectiveness, leading to enhanced customer satisfaction, revenue growth, and cost reduction.

OBJECTIVES

Objective for Pharmacy Management System:

1. To develop a Pharmacy Management System that automates inventory management, sales tracking, and customer relations management processes.

2. To design and implement a user-friendly UI that allows pharmacy employees to easily access and input data into the system.

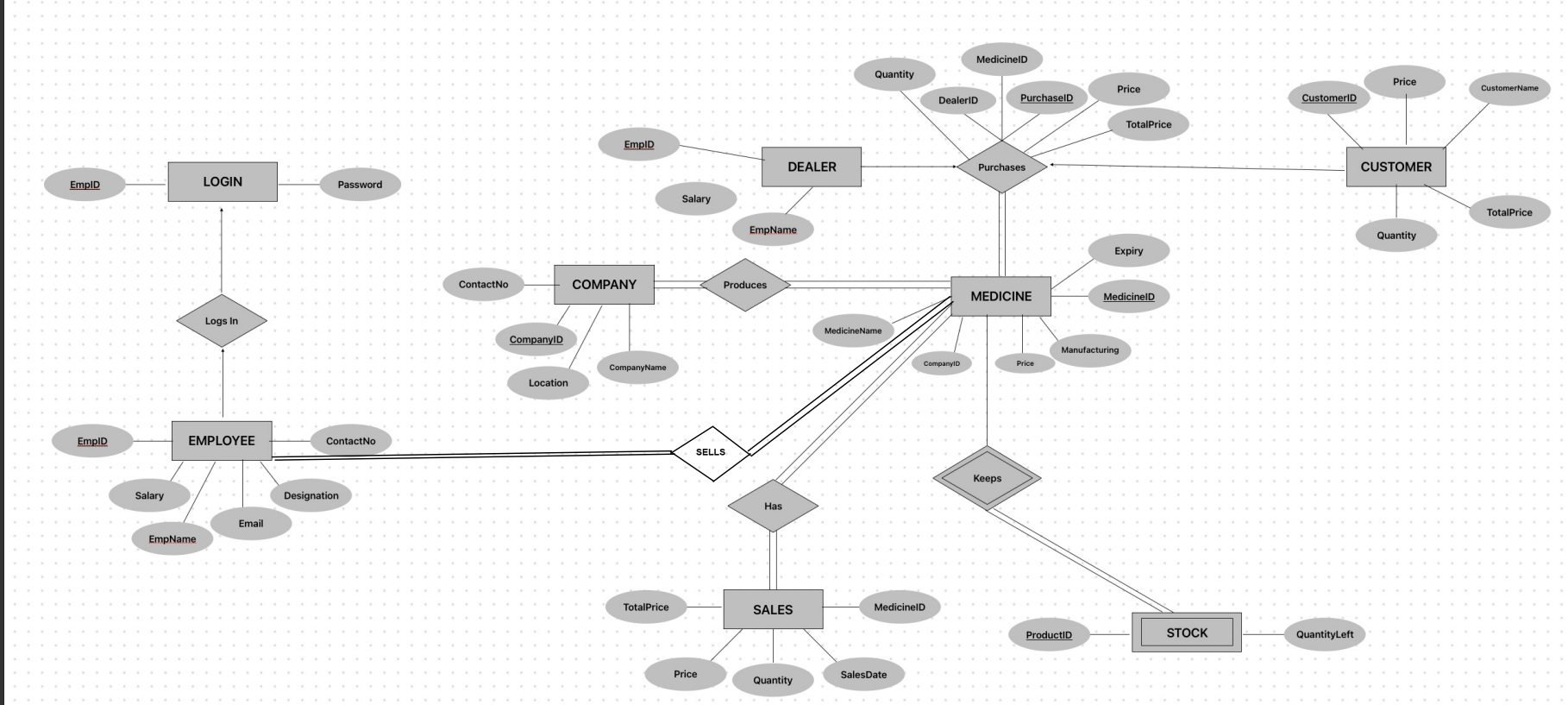
3. To create an efficient and scalable database design that accurately stores and organizes medication information, employee data, and sales transactions.

4. To provide real-time updates on medication inventory levels and sales data to improve decision-making and restocking processes.

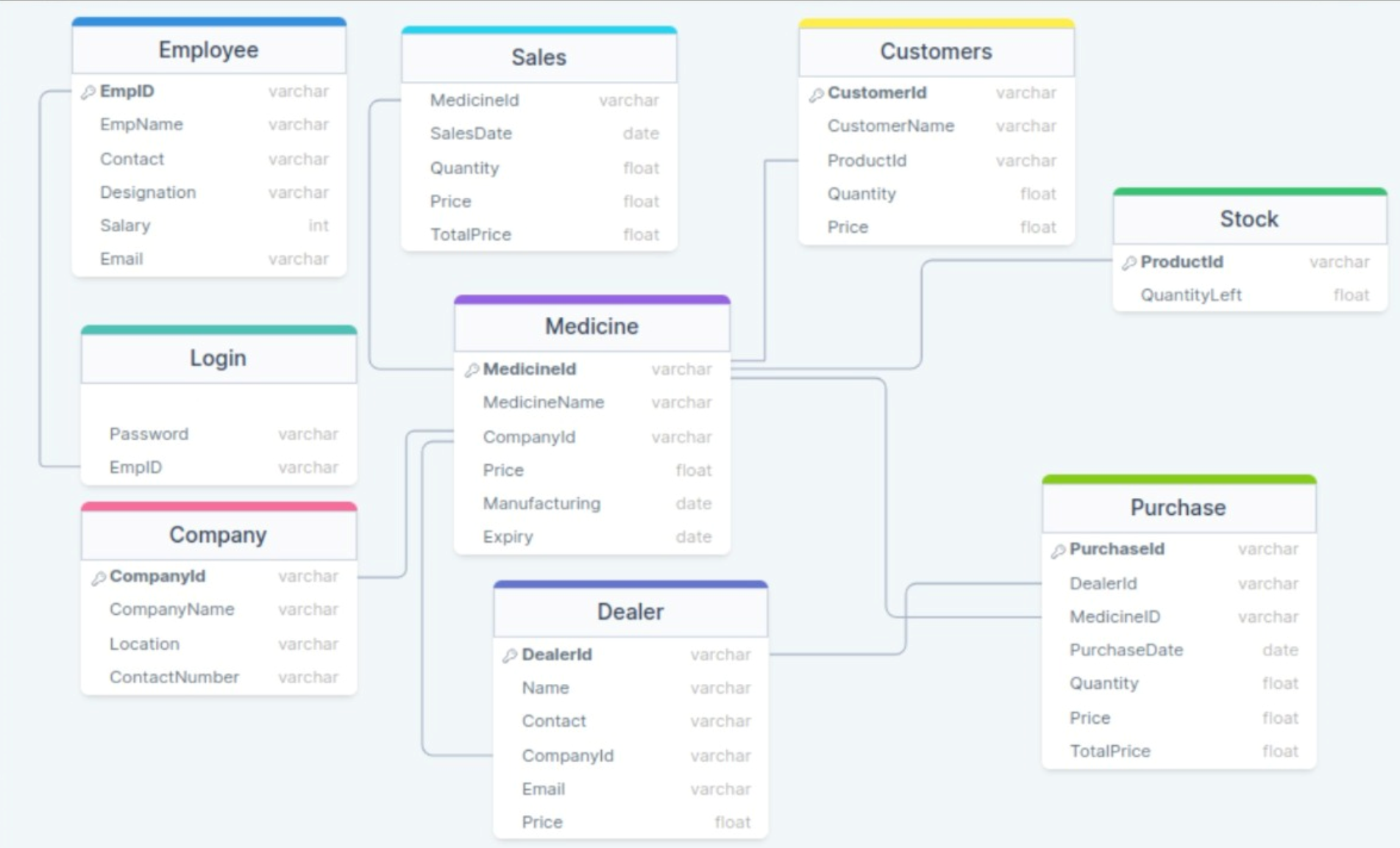
5. To improve overall pharmacy efficiency and profitability by reducing errors and streamlining operations through the implementation of the digital management system.

DATABASE DESIGN

ER DIAGRAM



SCHEMA DIAGRAM



REDUCTION

Company(**CompanyID**, CompanyName, Location, ContactNumber)

Medicine(**MedicineID**, MedicineName, CompanyID, Price, ManufacturingDate, ExpiryDate)

Dealer(**DealerID**, Name, Contact, CompanyID, Email, Price)

Employ(**EmpID**, EmpName, Contact, House, Designation, Salary, Email)

Login (Password, **EmpID**)

Sales(**MedicineID**, SalesDate, Quantity, Price, Totalprice)

Purchase(**PurchaseID**, **DealerID**, **Medicineid**, PurchaseDate, Quantity, Price, Totalprice)

Stock(**Productid**, Quantityleft)

Customers(**CustomerID**, CustomerName, Productid, Quantity, Price)

NORMALIZATION

The functional dependencies are as follows:

- In the Company table, CompanyID → CompanyName, Location, ContactNumber (since a company is uniquely identified by its ID, and its name, location, and contact number depend on the company ID).

- In the Medicine table, MedicineID → MedicineName, CompanyID, Price, ManufacturingDate, ExpiryDate, and CompanyID → CompanyName, Location, ContactNumber (since a medicine is uniquely identified by its ID, and its name, company details, price, manufacturing date, and expiry date depend on the medicine ID, and a company can manufacture multiple medicines).

- In the Dealer table, DealerID → Name, Contact, CompanyID, Email, Price, and CompanyID → CompanyName, Location, ContactNumber (since a dealer is uniquely identified by its ID, and its name, contact, company details, email, and price depend on the dealer ID, and a company can have multiple dealers).

- In the Employ table, EmpID → EmpName, Contact, House, Designation, Salary, Email (since an employee is uniquely identified by its ID, and its name, contact, house, designation, salary, and email depend on the employee ID).

- In the Login table, Password → EmpID (since a password is associated with a specific employee ID).

- In the Sales table, MedicineID, SalesDate → Quantity, Price, Totalprice (since the quantity, price, and total price of a medicine sold depend on both the medicine ID and the sales date).

- In the Purchase table, PurchaseID → DealerID, Medicineid, PurchaseDate, Quantity, Price, Totalprice (since a purchase is uniquely identified by its ID, and its details depend on the purchase ID).

- In the Stock table, Productid → Quantityleft (since the quantity left of a product in stock depends on the product ID).

- In the Customers table, CustomerID, Productid → CustomerName, Quantity, Price (since the customer name, quantity, and price of a product purchased by a customer depend on both the customer ID and the product ID).

1NF

Company(CompanyID, CompanyName, Location, ContactNumber)

- The Company table is already in 1NF because each attribute contains only atomic values.

Medicine(MedicineID, MedicineName, CompanyID, Price, ManufacturingDate, ExpiryDate)

- The Medicine table is already in 1NF because each attribute contains only atomic values.

Dealer(DealerID, Name, Contact, CompanyID, Email, Price)

- The Dealer table is already in 1NF because each attribute contains only atomic values.

Employ(EmpID, EmpName, Contact, House, Designation, Salary, Email)

- The Employ table is already in 1NF because each attribute contains only atomic values.

Login (Password, EmpID)

- The Login table is already in 1NF because each attribute contains only atomic values.

Sales(MedicineID, SalesDate, Quantity, Price, Totalprice)

- The Sales table is already in 1NF because each attribute contains only atomic values.

Purchase(PurchaseID, DealerID, Medicineid, PurchaseDate, Quantity, Price, Totalprice)

- The Purchase table is already in 1NF because each attribute contains only atomic values.

Stock(Productid, Quantityleft)

- The Stock table is already in 1NF because each attribute contains only atomic values.

Customers(CustomerID, CustomerName, Productid, Quantity, Price)

- To convert the Customers table into 1NF, we need to split the Productid column into two columns: ProductID and ProductName, because ProductName is not an atomic value. The new Customers table is as follows:

Customers(CustomerID, CustomerName, ProductID, ProductName, Quantity, Price)

Now, all the tables are in 1NF, and each attribute contains only atomic values.

2NF

To convert into 2NF, we need to follow these steps:

1. Identify the functional dependencies (FDs) that exist within the data.

2. Eliminate any partial dependencies by creating separate tables for the dependent data.

3. Use primary keys and foreign keys to relate the new tables to each other.

To eliminate any partial dependencies, we need to move some columns to separate tables:

1. Company(CompanyID, CompanyName, Location, ContactNumber)

2. Medicine(MedicineID, MedicineName, CompanyID, Price, ManufacturingDate, ExpiryDate)

3. Dealer(DealerID, Name, Contact, CompanyID, Email)

4. Employ(EmpID, EmpName, Contact, House, Designation, Salary, Email, Password)

5. Sales(SalesID, MedicineID, SalesDate, Quantity, Price, Totalprice)

6. Purchase(PurchaseID, DealerID, Medicineid, PurchaseDate, Quantity, Price, Totalprice)

7. Stock(Productid, Quantityleft)

8. Customers(CustomerID, CustomerName)

New Tables:

9. Customer\_Product(CustomerID, Productid, Quantity, Price)

10. Employee\_Login(EmpID, Password)

The Customer\_Product table separates the product information from the Customers table, which eliminates partial dependencies.

The Employee\_Login table separates the login information from the Employees table, which eliminates partial dependencies.

The final set of tables, in 2NF, would be:

1. Company(CompanyID, CompanyName, Location, ContactNumber)

2. Medicine(MedicineID, MedicineName, CompanyID, Price, ManufacturingDate, ExpiryDate)

3. Dealer(DealerID, Name, Contact, CompanyID, Email)

4. Employ(EmpID, EmpName, Contact, House, Designation, Salary, Email)

5. Sales(SalesID, MedicineID, SalesDate, Quantity, Price, Totalprice)

6. Purchase(PurchaseID, DealerID, Medicineid, PurchaseDate, Quantity, Price, Totalprice)

7. Stock(Productid, Quantityleft)

8. Customers(CustomerID, CustomerName)

9. Customer\_Product(CustomerID, Productid, Quantity, Price)

10. Employee\_Login(EmpID, Password)

3NF

To convert the 2NF table structure into 3NF, we need to identify and remove any transitive dependencies. In other words, we need to ensure that every non-key attribute in a table is dependent only on the primary key of that table, and not on any other non-key attributes.

Using the tables provided in the 2NF structure, we can identify the following transitive dependencies:

- In the Medicine table, Price and CompanyID depend on MedicineID. However, CompanyID also determines the CompanyName and Location, which are not directly related to MedicineID. Therefore, we can create a separate table for Company that includes CompanyID, CompanyName, and Location. The Medicine table would then reference the Company table via the foreign key CompanyID.

- In the Dealer table, Email and Price depend on DealerID, but CompanyID also determines CompanyName, which is not directly related to DealerID. Therefore, we can create a separate table for Company (if it does not already exist) and include CompanyID, CompanyName, and Location. The Dealer table would then reference the Company table via the foreign key CompanyID.

- In the Employ table, Contact and Email depend on EmpID, but Designation also determines Salary. Therefore, we can create a separate table for Designation that includes Designation and Salary. The Employ table would then reference the Designation table via the foreign key Designation.

- In the Sales table, Quantity and Totalprice depend on MedicineID and SalesDate, but Price is already determined by MedicineID. Therefore, we can remove Price from the Sales table and assume that it can be looked up in the Medicine table. If necessary, we can also create a view that joins Sales and Medicine to display all relevant information.

- In the Purchase table, Quantity and Totalprice depend on DealerID, Medicineid, and PurchaseDate, but Price is already determined by Medicineid. Therefore, we can remove Price from the Purchase table and assume that it can be looked up in the Medicine table. If necessary, we can also create a view that joins Purchase and Medicine to display all relevant information.

With these changes, the 3NF table structure would look like:

- Company(CompanyID, CompanyName, Location)

- Medicine(MedicineID, MedicineName, CompanyID, ManufacturingDate, ExpiryDate)

- Dealer(DealerID, Name, Contact, CompanyID, Email)

- Designation(Designation, Salary)

- Employ(EmpID, EmpName, Contact, House, Designation, Email)

- Login(Password, EmpID)

- Sales(MedicineID, SalesDate, Quantity, Totalprice)

- Purchase(PurchaseID, DealerID, Medicineid, PurchaseDate, Quantity, Totalprice)

- Stock(Productid, Quantityleft)

- Customers(CustomerID, CustomerName, Productid, Quantity, Price)

Note that the Sales and Purchase tables no longer include the Price attribute, as it can be looked up in the Medicine table. Similarly, the CompanyID attribute in the Dealer table now serves only as a foreign key, with the CompanyName and Location attributes moved to a separate Company table.

BCNF

To convert the schema to BCNF, we need to ensure that all functional dependencies are dependencies on the primary key, and there are no non-trivial functional dependencies between non-key attributes.

There are no non-trivial functional dependencies between non-key attributes in the current schema, so it is already in BCNF.

METHODOLOGY

The development of the Pharmacy Management System involved the use of C# for the UI and MS SQL for the backend, with Visual Studio serving as the development environment. A structured development methodology was adopted, which encompassed project planning, system analysis and design, implementation, and testing.

During the planning phase, project goals, requirements, and constraints were identified. System analysis and design followed, which entailed creating a detailed system design covering system architecture, database schema, and UI design. Agile methodologies were utilized to ensure the project's smooth and timely progression.

Subsequently, the implementation phase commenced, with the coding of the UI and backend components in C# and MS SQL ,respectively. The team paid attention to coding standards to guarantee the code's maintainability, scalability, and readability.

In summary, the Pharmacy Management System development utilized C# and MS SQL for the UI and backend, respectively, while Visual Studio Studio served as the development environment. The project team followed a structured development methodology that included planning, system analysis and design, implementation, and testing, with agile methodologies utilized to ensure timely progress.

RESULTS

<screenshots>

FUTURE WORK

Future work for the Pharmacy Management System project could include various areas for expansion and improvement. One such area is the integration of electronic prescribing and medication dispensing to streamline the medication ordering process and reduce errors, thereby increasing efficiency.

Another potential development area is the implementation of a real-time inventory tracking system, which can ensure accurate stock levels and prevent stockouts. In addition, incorporating data analytics and machine learning algorithms can help predict trends and improve decision-making in inventory management, purchasing, and pricing strategies, leading to better outcomes.

The system could also be upgraded with features for prescription drug adherence monitoring and patient engagement to provide patients with more personalized care and support. Moreover, integrating the system with mobile applications could give patients convenient access to their medical records, appointment schedules, and medication reminders, thereby enhancing patient engagement.

Moreover, extending the system to include billing and insurance processing could simplify the payment process for patients and healthcare providers, improving the overall user experience. Finally, enhancing the UI and user experience can lead to better usability and adoption by healthcare professionals and patients, thereby increasing system effectiveness.

In conclusion, these potential future developments can significantly enhance the efficiency and effectiveness of managing pharmacy operations, leading to improved patient care outcomes.