Pharmaceutical Inventory Management System

Name: Luke Thilgen

Student ID: 921358799

GitHub Username: LucaT445

Milestone	Date Submitted
M1	9/18/2024
M2	10/2/2024
M3	10/16/2024

Table of Contents

Project Description

- The need for efficient and accurate management of pharmaceutical inventories is paramount for patients with physical and medical conditions to consistently access their needed medications and for hospitals and clinics to provide quality medical care. Using the Pharmaceutical Inventory Management System, healthcare providers can accomplish both of these with proper inventory tracking, reordering processes, and accurate records of stock levels. This will help patients be able to access their medications, reduce the detrimental impact of medical waste, and lower overall operational costs. If we can solve these issues, the Pharmaceutical Inventory Management System will help deliver high-quality care for those in need and facilitate the optimal use of available resources. To accomplish these goals, the Pharmaceutical Inventory Management System integrates with several suppliers to request information such as stock level, pricing, lead times, and delivery speed, so it allows for a consistent supply of medication and helps prevent delays. Another one of its capabilities is prioritizing the transfer of medications or supplies nearing expiration to locations where they are needed. This will greatly reduce the amount of medical waste generated, which can cause exposure to dangerous material or significant changes to the environment. Finally, it can predict possible medication shortages and take measures to solve problems that could arise.

  The Pharmaceutical Inventory management system can also improve certain software currently on the market such as FlexScanID and Sortly. Integrating the Pharmaceutical Inventory Management System with FlexScanID could improve FlexScanID's ability to anticipate shortages due Pharmaceutical Inventory Management System's machine learning features, while integrating it with Sortly could improve Sortly's ability to manage overstocking and simultaneously introduce the ability for Sortly to manage expiration dates for items tailored to the pharmaceutical industry, a feature Sortly currently lacks.

- **Use Cases**:
  - **Use Case**: Stock Replenishment - Ensuring Medication Availability
    - **Actor**: Pharmacy Manager (Jane), Supplier
    - **Description**: Jane is a pharmacy manager at a large pharmacy. She faces several challenges with critical medications being out of stock due to unpredictable delays from suppliers. As a result, she receives numerous complaints from prescribers and their patients regarding the availability of medication. She needs a system that automatically monitors stock levels, predicts when medications will run low, and places orders with appropriate suppliers to ensure sufficient stock.
    - The **Pharmaceutical Inventory Management System** will help Sarah by automatically tracking the inventory levels of all medications and predicting when stocks will run low. With real-time data on medication availability and demand, the system can place replenishment orders with

suppliers before a shortage occurs. This will prevent stockouts, ensure continuous availability of critical medications, and reduce manual workload for the pharmacy manager.

- **Use Case**: Expiry and Overstock Management - Reducing Medication Waste
  - **Actor**: Inventory Supervisor (Dave), Staff
  - **Description**: Dave is an inventory manager at a large pharmacy chain who is responsible for ensuring that the inventory is free from obsolete medication, whether they are near the expiration date or are overstocked. This is done to prevent medication from being dispensed to patients. He is becoming increasingly overwhelmed by the amount of manual checking that he has to do and expired medications are sometimes still left on the shelf which can lead to safety risks, regulatory non-compliance, and wasted stock due to other pharmacies potentially needing them. He needs a system that will provide quick, accurate, and timely alerts for medications that are expired, nearing expiration or are overstocked so he can efficiently prioritize their disposal or transfer.
  - The **Pharmaceutical Inventory Management System** will assist Dave by automatically tracking the inventory levels of medications and notifying him when they are overstocked, expired, or nearing expiration. This information will significantly reduce the amount of time Dave has to spend checking the shelves and allow him to successfully dispose of expired medications and transfer the ones that are not needed to other pharmacies that need them.
- **Use Case**: ADHD Medication Receival - Finding Alternative Medication
  - **Actor**: ADHD Patient (Paul), classmates
  - **Description**: Paul is a college student who has ADHD. While some people with ADHD can manage their condition without medication, his symptoms are much more severe than most. Although his prescribed medication makes it easier for him to pay attention and complete his work more easily in the low-stimulation environment, the worldwide ADHD medication shortage has affected his life significantly. Without his medication, he struggles with motivation on schoolwork that he isn't interested in and struggles to remember basic things such as meeting deadlines due to his poor executive functioning. He needs an alternative medication to help manage his symptoms as his Adderall is currently out of stock at many pharmacies near his location.
  - The **Pharmaceutical Inventory Management System** will help Paul find an alternative medication by contacting suppliers that supply his meds and using their information to dispense the best possible alternative for his situation. This will allow Paul to function decently well until his preferred medication is available again.

Functional Database Requirements

- **Entities**:
    1. **User**
        a. A user shall only have one account.
        b. A user shall have a role (Patient, Pharmacist, Prescriber).
        c. A user shall have a unique user ID.
    2. **Account**
        a. An account can only have one user.
        b. An account shall have a valid email address and the option to add a phone number.
        c. An account shall have the user's date of birth.
        d. An account shall have a creation date.
    3. **Address**
        a. An address can be shared by many users.
        b. An address shall have a unique address ID.
        c. An address shall include a street name, city, state, and country.
    4. **Notification**
        a. A notification shall be associated with a specific user or account.
        b. A notification shall be delivered to the user's email address or phone number.
        c. A notification shall reflect the most recent changes to the user's prescription status.
    5. **Medication**
        a. A medication shall have a name and the name shall be indicative if it is brand or generic.
        b. A medication shall have a drug class.
        c. A medication shall have an SKU (stock-keeping unit).
        d. A medication shall have an expiration date.
    6. **Drug Interactions**
        a. An interaction shall involve at least two substances.
        b. An interaction shall have a severity level.
        c. An interaction shall have a description of what its effects are.
    7. **Supplier**
        a. A supplier shall have a name
        b. A supplier shall have a list of medications they manufacture and supply.
        c. A supplier shall be able to update contact information if needed.
        d. A supplier shall be able to deliver multiple orders to multiple pharmacies.
    8. **Patient**
        a. A patient shall have a name.
        b. A patient shall have a patient ID linked to the user.
        c. A patient shall have their insurance information linked to them.
        d. A patient shall be able to view their prescription history.
    9. **Pharmacy**

a. A pharmacy shall have an address.
b. A pharmacy shall be able to place orders and receive deliveries from suppliers.
c. A pharmacy shall have a list of pharmacists who work there.

10. Pharmacist

a. A pharmacist shall have a name.
b. A pharmacist shall have a pharmacist ID linked to the user.
c. A pharmacist shall be able to access and update inventory records.
d. A pharmacist shall be able to communicate with prescribers or patients.

11. Prescriber

a. A prescriber shall have a name.
b. A prescriber shall have a prescriber ID linked to the user.
c. A prescriber shall be able to have multiple patients.
d. A prescriber shall have a valid medical license.
e. A prescriber shall be able to view the medical history of their patients.

12. Prescription

a. A prescription shall have a unique ID.
b. A prescription shall have the medication's details such as the dosage and frequency of taking it.
c. A prescription shall be able to track the number of refills available.

13. Order

a. An order shall have a unique ID.
b. An order shall either be associated with a pharmacy and a supplier or two or more pharmacies.
c. An order shall have the details of the medications ordered.
d. An order shall have a status.

14. Inventory

a. An inventory shall have stock levels for each medication.
b. An inventory shall be updated when medications are dispensed to patients or received by suppliers.
c. An inventory shall notify the staff when stock levels fall below a certain level.

15. Drug Class

a. A drug class shall have a name.
b. A drug class shall have a unique ID
c. A drug class shall be able to group multiple medications under a name.
d. A drug class shall have a list of conditions each is designed to treat.

16. Insurance

a. The insurance shall have a name.
b. The insurance shall be able to be linked to multiple patients.
c. The insurance shall be able to authorize prescription coverage.
d. The insurance shall be able to track claims filed by the pharmacy or petitions by the patient or prescriber.

17. Renewal

a. A renewal shall have a unique ID.
b. A renewal shall notify the prescriber or patient when a renewal is available.
c. A renewal shall track the amount of times a prescription has been renewed.

18. Payment

a. A payment shall have a method of payment.
b. A payment shall be associated with a specific prescription or order.
c. A payment shall have a status.

19. Inventory Audit

a. An inventory audit shall have a period in which it is regularly conducted.
b. An inventory shall compare the actual and recorded inventory and see if they match.
c. An inventory audit shall include the date and the name of the pharmacist who conducted the audit.

20. Feedback

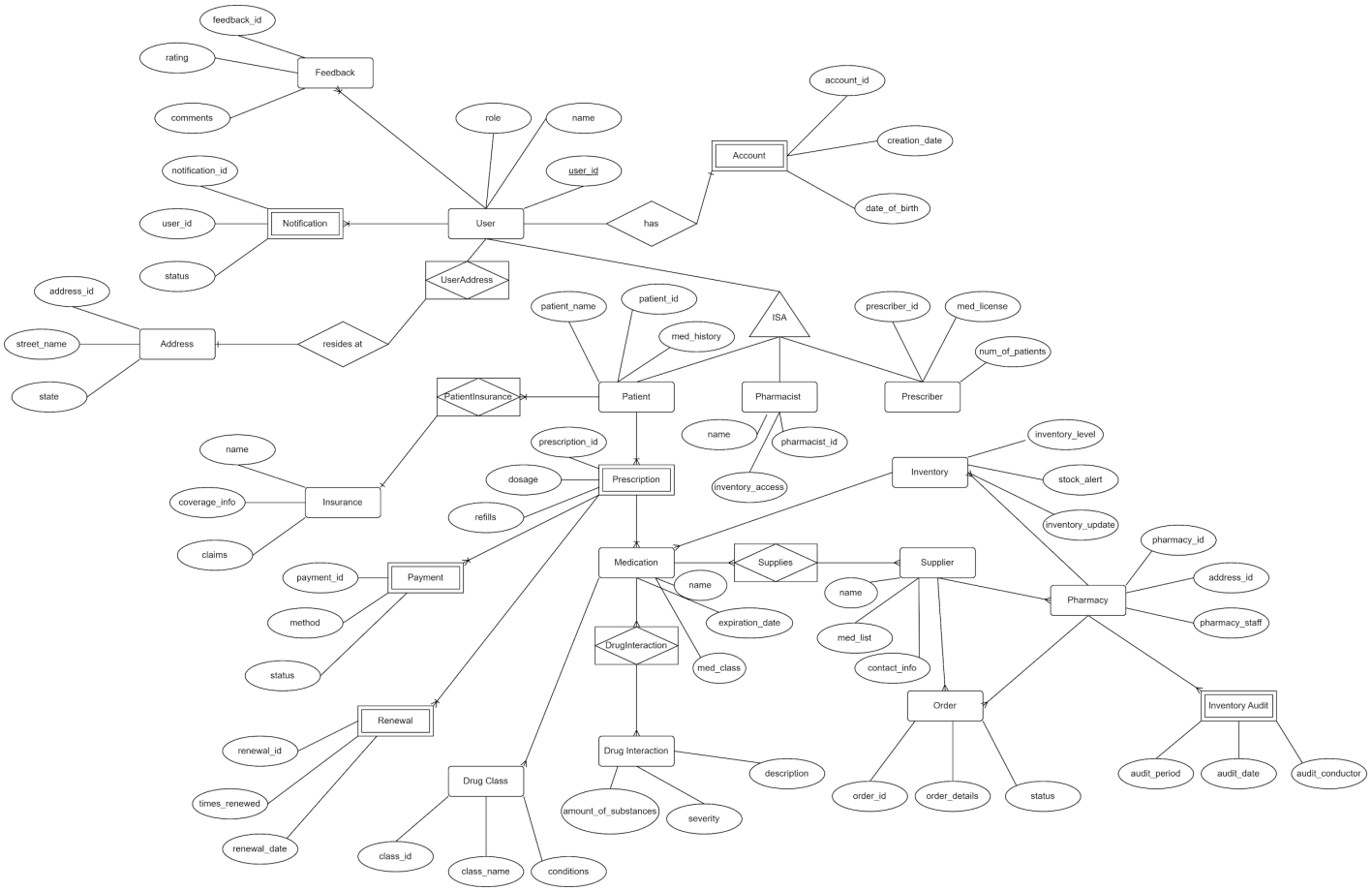
a. The feedback shall be linked to a user.
b. The feedback shall have a rating and the option for comments from other users.
c. The feedback shall allow pharmacies or prescribers to respond to address concerns.

Non-Functional Database Requirements

1. **Performance**
    - The database system shall support concurrent transactions.
    - The database system shall use query logging to determine the most frequently accessed data.
    - The database system shall implement caching for the most frequently accessed data.
    - The database system shall have fast query execution, with 95% of queries executing within 2 seconds.
2. **Security**
    - The database system shall only accept encrypted passwords.
    - The database system shall be automatically backed up every day at 11:59 pm.
    - The database system shall automatically log out users if there are 20 minutes of inactivity.
3. **Scalability**
    - The database system shall allow integration with third-party suppliers.
    - The database system shall be able to allow additional servers or instances to be added.
    - The database system shall use dynamic memory allocation to ensure it only uses the memory that it needs.
4. ***Capability***
    - The database system shall ensure that users can only access features and data relevant to their role.
    - The database system shall be able to support real-time updates to the inventory.
    - The database system shall allow pharmacies to batch-process multiple medication orders.
5. **Environmental**
    - The database system shall be able to run on AWS or Google Cloud.
    - The database system shall be compatible with Windows and Linux.
    - The database system shall support data compression to reduce the impact on users with limited Internet connectivity.
6. **Coding Standards**
    - The database system shall follow PEP 8.
    - The database system shall use Git for version control.
    - The database system shall have docstrings for its functions and classes.
7. **Storage**
    - The database system shall support persistent storage.
    - The database system shall be automatically backed up daily.
    - The database system shall support indexing for fast data retrieval.
8. **Privacy**
    - The database system shall comply with HIPAA.
    - The database system shall ensure that users provide consent before any of their data is stored.

- The database system shall implement data anonymization to protect highly sensitive data.

Entity Relationship Diagram



Entity Set Description

1.) User (Strong)

- user_id: Numeric, primary key.
- name: Alphanumeric.
- role: Alphanumeric (Patient, Pharmacist, Prescriber).

2.) Account (Weak)

- account_id: Numeric, primary key.
- user_id: Numeric, foreign key (references User entity set).
- creation_date: Date.
- date_of_birth: Date.

3.) Address (Strong)

- address_id: Numeric, primary key.
- street_name: Alphanumeric.
- state: Alphanumeric.

4.) Notification (Weak)

- notification_id: Numeric, primary key.
- user_id: Numeric, foreign key (references User entity set).
- message: Alphanumeric.

5.) Medication (Strong)

- medication_id: Numeric, primary key.
- name: Alphanumeric.
- med_class: Alphanumeric.

6.) Drug Interaction (Strong)

- drug_interaction_id: Numeric, primary key.
- amount_of_substances: Numeric.
- severity: Alphanumeric.

7.) Supplier (Strong)

- supplier_id: Numeric, primary key.
- name: Alphanumeric.
- contact_info: Alphanumeric.

8.) Patient (Strong)

- patient_id: Numeric, primary key.
- patient_name: Alphanumeric.
- med_history: Alphanumeric.

9.) Pharmacy (Strong)

- pharmacy_id: Numeric, primary key.
- address_id: Numeric, foreign key (references Address entity set).
- pharmacy_staff: Alphanumeric

10.) Pharmacist (Strong)

- pharmacist_id: Numeric, primary key.
- name: Alphanumeric.
- inventory_access: Boolean.

11.) Prescriber (Strong)

- prescriber_id: Numeric, primary key.
- med_license: Alphanumeric.
- num_of_patients: Numeric

12.) Prescription (Strong)

- prescription_id: Numeric, primary key.
- dosage: Alphanumeric.
- refills: Numeric.

13.) Order (Strong)

- order_id: Numeric, primary key.
- order_details: Alphanumeric.
- status: Alphanumeric.

14.) Inventory (Strong)

- inventory_id: Numeric, primary key.
- inventory_level: Numeric.
- stock_alert: Boolean.

15.) Drug Class (Strong)

- class_id: Numeric, primary key.
- class_name: Alphanumeric.
- conditions: Alphanumeric.

16.) Insurance (Strong)

- insurance_id: Numeric, primary key.
- name: Alphanumeric.
- coverage_info: Alphanumeric.

17.) Renewal (Weak)

- renewal_id: Numeric, primary key.
- times_renewed: Numeric.
- renewal_date: Date.

18.) Payment (Weak)

- payment_id: Numeric, primary key.
- method: Alphanumeric.
- status: Alphanumeric.

19.) Inventory Audit (Weak)

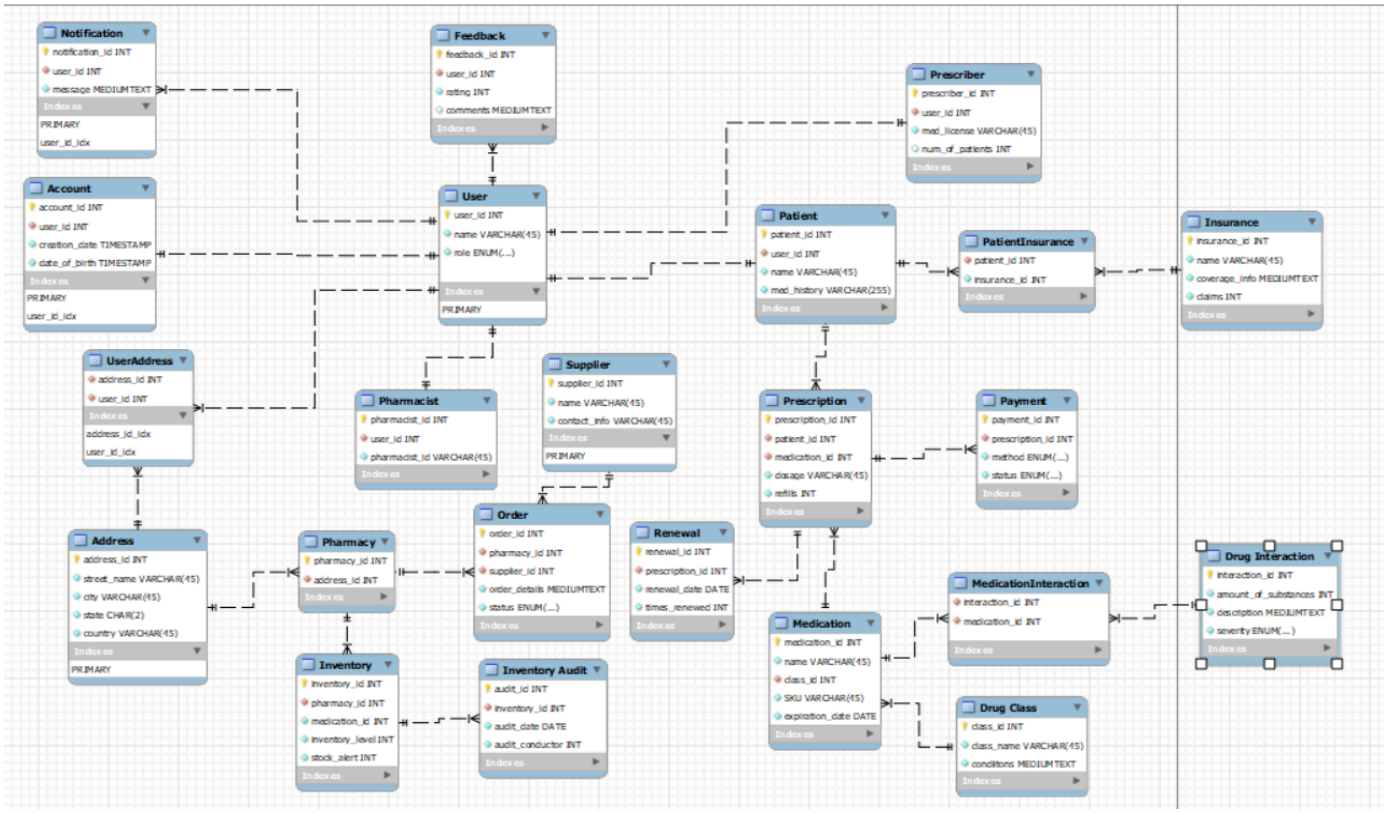
- audit_id: Numeric, primary key.
- audit_period: Date.
- audit_conductor: Alphanumeric.

20.) Feedback (Strong)

- feedback_id: Numeric, primary key.

- rating: Numeric.
- comments: Alphanumeric.

Enhanced Entity Relationship



Normalization Techniques Used

Throughout this milestone, I implemented:

- 1NF normalization by having primary keys in each entity table. This allows for more efficient querying since each element is only stored once in a specific table.
- 2NF normalization is achieved by making each attribute in each table dependent solely on the primary key, eliminating any partial dependency, which, in turn, reduces redundancy.
- 3NF normalization by ensuring that no non-key attributes depend on other non-key attributes (e.g. the `dosage` attribute in `Prescription` is not dependent on another non-key attribute that may exist in another entity table, such as `Medication`). This compartmentalizes and organizes data and prevents duplicate data storage.