AI-Enhanced Pharmacy Procurement

Taha H. Ababou, Manuel Segimón, Joel Akerman, Bora Bulut, Zaiyan Muhammad

**Executive Summary**—The nature of the problem at hand revolves around the client's manual and time-consuming drug procurement processes, characterized by the need for constant evaluation of alternatives based on price, quantity, and packaging. Additionally, the criticality of specific drugs necessitates a precise and timely restocking strategy. Our final deliverable aims to address these challenges comprehensively through an AI-driven procurement solution. This solution comprises a three-tiered technical approach: continuous stock monitoring and replenishment based on live feed analysis, intelligent recommendations for alternative drug replacements drawn from the pharmacy's database, and the development of a user-friendly interface based on client feedback. The proposed model offers innovative features such as dynamic stock monitoring, intelligent alternative recommendations prioritized by cost, dosage preference, and packaging, and a user-centric interface tailored to streamline the buyer's decision-making process. This solution is poised to significantly optimize the client's drug procurement workflow, ensuring efficiency, accuracy, and responsiveness to critical drug restocking requirements.

**Index Terms**— Artificial Intelligence, Machine Learning, Optimization, Procurement

————————————————

* *Taha H. Ababou is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail: hababou@bu.edu*
* *Manuel Segimón is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail: manuelsp@bu.edu*
* *Joel Akerman is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail: akermanj@bu.edu.*
* *Bora Bulut is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail: bbulut@bu.edu.*
* *Zaiyan Muhammad is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215. E-mail: monem@bu.edu*

—————————— ◆ ——————————

# 1 Introduction

In contemporary pharmacy procurement systems, the prevalent issue revolves around the labor-intensive and time-consuming nature of manual order placements and inventory updates. This process often leads to challenges such as overstocking and stockouts, resulting in inefficiencies and potential disruptions in patient care. Our team has developed an AI-enhanced pharmacy procurement solution to address these concerns. This project aims to leverage artificial intelligence (AI) to streamline the procurement process, making it more efficient, cost-effective, and responsive to the needs of both healthcare providers and patients.

Customer's Problem:

Hospital pharmacies face intricate challenges related to drug shortages, necessitating complex buying decisions. Buyers must navigate package sizes, drug availability, potential alternatives, and budget constraints, requiring a deep understanding and experience. The reliance on individual expertise may lead to inconsistencies and inefficiencies in the procurement process. The overarching customer problem is the need for a system that can intelligently orchestrate procurement operations, ensuring optimal medication levels, reducing the risk of stockouts, and containing healthcare costs.

Purpose of the Project:

The primary purpose of our project is to conceptualize and develop an AI-driven system dedicated to refining and augmenting the pharmaceutical procurement process. This initiative addresses the multifaceted challenges of replenishing medicine stocks from various suppliers, each operating with distinct paradigms. Our project aims to alleviate stock-out risks by intelligently orchestrating procurement operations based on real-time demand analytics and supplier availability assessments.

General Approach:

Our team's general approach involves the development of a machine learning model trained using historical purchase data, considering drug package sizes and availability. The AI model will automatically determine the optimal amount of medication to order, considering alternatives when the primary choice is unavailable. The project comprises a responsive dashboard using ReactJS, a Flask-powered Python-based backend, a secure SQL database, and a TensorFlow and Keras-backed machine learning model.

How the Approach Solves User's Problem/Needs:

Our approach ensures that the procurement system becomes proactive and data-driven by considering purchase data and training the AI model to consider critical factors. The system monitors current drug stocks, automates restocking decisions based on supplier data, and provides intelligent recommendations for alternative drug replacements. This streamlines the procurement process and empowers procurement managers and pharmacists with actionable insights to optimize strategies, reduce administrative burdens, and ensure a seamless patient care experience.

Highlights and Special Features:

Comprehensive Data Analysis: The project starts with a thorough data collection and analysis phase, considering past data, current trends, demand, and supplier details.

AI Model Development: The machine learning models are developed to optimize the procurement process, determining optimal quantities based on extensive data collected during the analysis phase.

User-Friendly Interface: An intuitive dashboard is created for effective interaction with the AI-powered procurement system, allowing human overseers to monitor, override, or tweak system recommendations.

Seamless Integration: The final phase focuses on ensuring seamless integration with external systems, combining internal data with external wholesaler APIs for efficient data exchange.

Feedback Loop: The system incorporates a feedback loop for real-time adjustments, ensuring adaptability to unpredictable scenarios in a pharmacy environment.

Our AI-enhanced pharmacy procurement solution is poised to revolutionize the pharmaceutical supply chain, ensuring optimal inventory levels, cost savings, and a responsive approach to patient care.

*Written by:* Bora Bulut

# 2 Concept Development

## 2.1 Problem Statement

Our project, AI-Driven Pharmacy Procurement Software, aims to enhance the decision-making process in pharmacy procurement by utilizing advanced AI technologies. The primary objective of our project extends beyond merely streamlining the ordering process. Through a comprehensive site visit and interactions with industry professionals, we have identified a more complex and nuanced need: to assist procurement managers in making informed choices among a wide range of pharmaceutical products, particularly in scenarios where direct product replacements are not readily available. This revised objective guides our development strategy, ensuring that our software is not only technologically sophisticated but also closely aligned with the real-world challenges faced in pharmaceutical procurement.

Our journey began with an understanding of the procurement process's intricacies and the challenges faced by procurement managers in their daily operations. This understanding was pivotal in guiding our technical approach and ensuring that the software we develop is not only technologically advanced but also practically relevant and user-friendly.

## 2.2 Engineering Understanding of the Customer's Problem

Initially, our team's understanding of the customer’s problem was somewhat straightforward – to create an application that would simplify the ordering process for pharmacy procurement managers. This application was intended to aid in identifying required replenishment sizes, automatically add them to a shopping cart, and facilitate direct ordering within our interface. This understanding guided the early stages of our project.

However, a critical turning point in our project was the site visit conducted on November 8th. During this visit, we had the invaluable opportunity to engage closely with the end-users of our software. We observed the procurement process firsthand, interacted with OmniCell, and navigated the supplier's website alongside seasoned procurement managers. One such interaction was with Mike, an experienced procurement manager, whose insights were particularly enlightening.

From these interactions, we gleaned two critical insights. Firstly, we realized that the client sought to minimize dependency on experienced managers for decision-making. This was a significant revelation, as it confirmed the direction our system needed to take – aiding in decision-making processes rather than merely automating them. Secondly, and more importantly, we discovered that the existing problem was not just about streamlining the order process. The real challenge lay in distinguishing between a myriad of supplier choices and determining the best alternative when the exact replacement for a drug was not available. This could be due to various reasons, such as availability in different form factors (e.g., tablets vs. liquid) or under different brand names.

This deeper understanding of the customer's problem was a game-changer. It shifted our focus from a simple order-processing application to a more complex decision-aid system. Our software needed to not only facilitate ordering but also provide insightful recommendations, especially in scenarios where direct replacements were not available. This requirement led us to develop a solution that could navigate through these complexities, offering cost-effective alternatives and maintaining the integrity of the procurement process.

## 2.3 Conceptual Approach to Solve the Problem

Recognizing the nuanced challenges identified during our site visit, we conceptualized a dual-output system for our AI-Driven Pharmacy Procurement Software. This innovative approach was designed to address the two-fold problem identified: making cost-effective choices among various suppliers and determining the best alternatives when exact replacements are unavailable.

Dual-Output System Design:

1.Best Matches Sorted by Price:

The first output of the system provides a list of the top three best matches for a requested pharmaceutical product, sorted by unit price. This feature is particularly useful in scenarios where cost-effectiveness is the primary criterion. Furthermore, the system offers flexibility by allowing the user to change sorting criteria and view additional results, thereby enhancing user control and adaptability.

2.Best Price Alternative for Different Form Factors:

The second output is a more nuanced solution, maintaining the same drug name but offering alternatives in different form factors. This output is tailored for critical decision-making scenarios, where experienced procurement managers might consider switching to a different form factor if it proves to be more cost-effective or readily available.

## 2.4 Alternative Solutions Considered

In the course of our project, we evaluated several alternative solutions before settling on our current conceptual approach. Two notable alternatives that were considered and subsequently abandoned are:

1. Single-Output, Price-Only Focus: Initially, we considered a simpler, single-output system focusing solely on finding the cheapest available option for a given product. While this approach had the merit of simplicity, it lacked the nuanced decision-making support needed in complex procurement scenarios, such as finding alternatives in different form factors.

2. Integration with Existing Ordering Software: Another approach was to integrate our solution directly with the existing ordering software used by the pharmacy. This integration promised a seamless user experience. However, gaining approval and access to modify existing systems proved to be a formidable challenge, especially considering our status as students. The existing software contains sensitive data that requires handling by professionals, posing a significant limitation for us.

These alternatives, though viable in their own right, were ultimately abandoned in favor of a more comprehensive solution that aligned closely with the client’s evolving needs and the complex nature of pharmaceutical procurement.

*Written by:* Zaiyan Muhammad

# 3 System Description

**3.1 System Architecture**

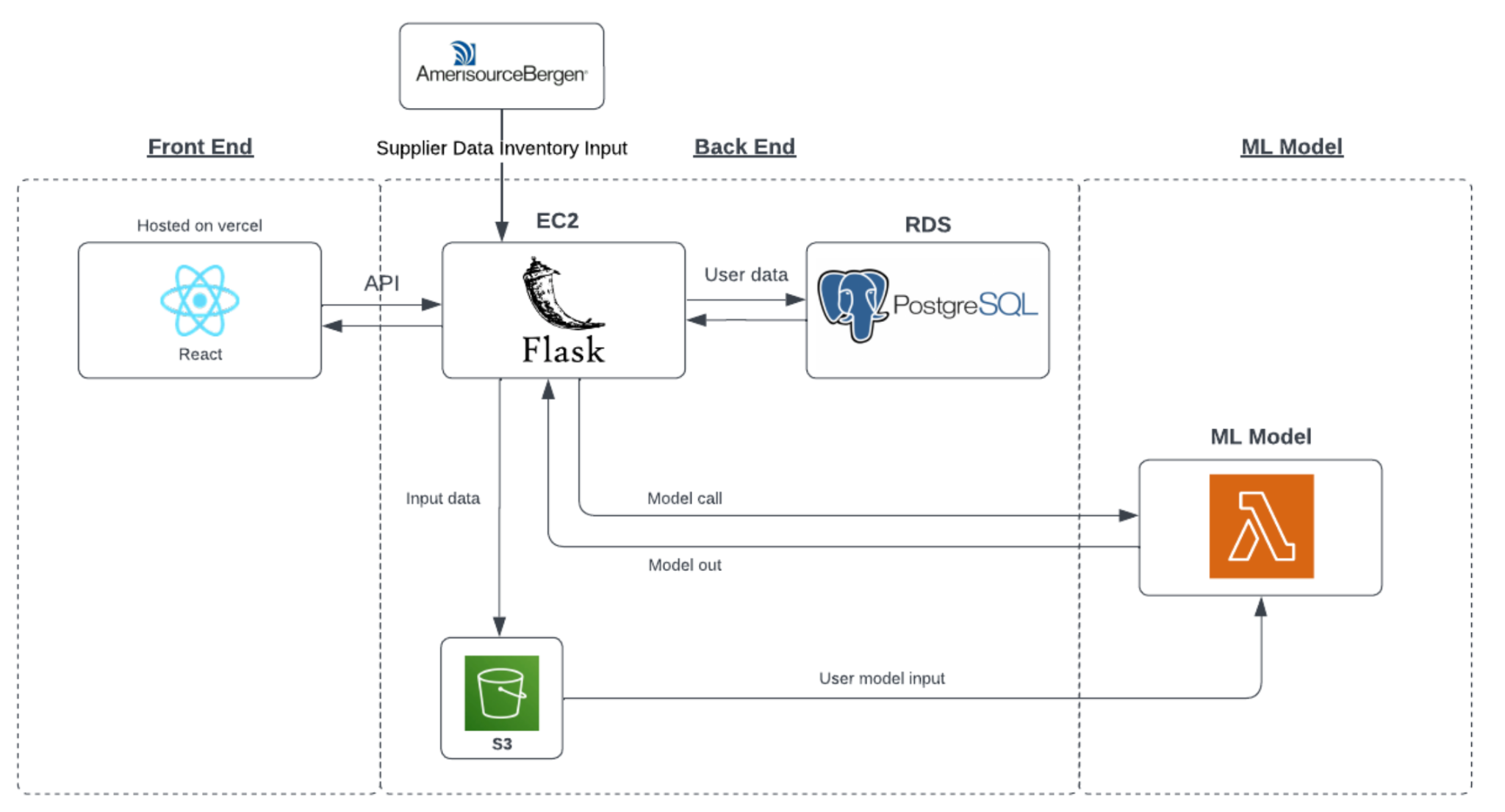


Fig. A. This image shows the overall architecture of the application.

Figure A showcases the architecture of our AI-driven pharmacy procurement system, specifically designed to optimize the interaction with our wholesaler supplier, AmerisourceBergen. The architecture is divided into three main components: Front End, Back End, and Machine Learning Model, each serving a distinct purpose in the procurement process. The Front End, built with React and hosted on Vercel, provides a user-friendly interface that allows seamless engagement with the system, making it accessible for users to navigate through AmerisourceBergen's data.

The Back End is powered by a Flask application on an Amazon EC2 instance, indicating a flexible and scalable server setup capable of managing complex workflows. The integration with AmerisourceBergen's supplier data is critical, and it is here where data is processed and prepared for analysis. PostgreSQL within Amazon RDS is employed for reliable data storage, ensuring that all supplier data from AmerisourceBergen is accurately maintained and readily available for transactional processing.

AWS S3 is utilized for the storage of large datasets, likely including AmerisourceBergen's inventory data, order history, and real-time stock levels. The Machine Learning Model, operating on AWS Lambda, is designed to analyze this data efficiently, enabling dynamic and responsive stock management based on the latest information provided by AmerisourceBergen.

## 3.2 System Block Diagram

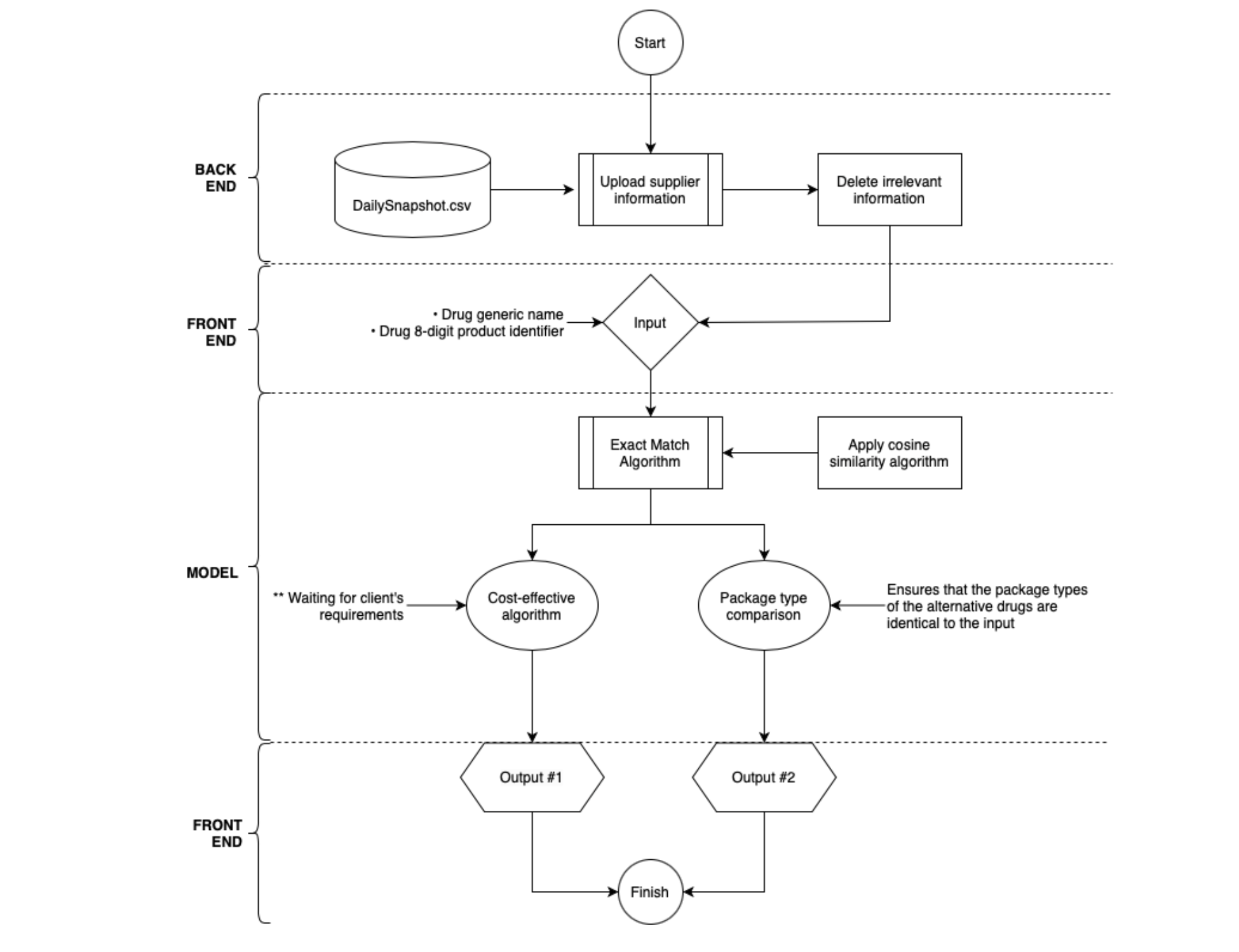


Fig. B. This image shows the block diagram of the application.

Figure B details the workflow of our system through a comprehensive block diagram, which begins with the Back End's uploading and processing of supplier data from AmerisourceBergen. The streamlined initial stage ensures that only pertinent data from the 'DailySnapshot.csv'—presumably a regular data extract from AmerisourceBergen—is carried forward for analysis. This data is the bedrock upon which procurement decisions are based, underpinning the accuracy of the subsequent steps.

Upon user input of drug information into the Front End, the system activates its core algorithms—the Exact Match and the Cosine Similarity algorithms—to process AmerisourceBergen's data. These algorithms are vital for identifying the best possible matches and alternatives within AmerisourceBergen's vast inventory, providing procurement managers with precise and actionable insights.

The Model's dual-output system then generates cost-effective procurement options and alternative drug form factors, each leveraging the comprehensive data provided by AmerisourceBergen. This ensures that procurement decisions are not only economically advantageous but also in alignment with actual inventory availability.

*Written by:* Taha H. Ababou

# 4 First Semester Progress

In the first semester of our senior design project, our team embarked on an ambitious journey to develop an AI-enhanced pharmacy procurement software. Our initial phase involved extensive interactions with the client to define the scope and objectives of the project. These meetings were instrumental in setting clear expectations and understanding the client’s unique requirements. Following client discussions, we delved into rigorous research on machine learning models. Our focus was on identifying algorithms that would be useful for us. Regular check-ins with the client were a cornerstone of our project. These meetings allowed us to refine our ideas and approaches based on client feedback. A significant milestone this semester was the development and testing of our prototype. The testing phase was crucial in identifying areas for improvement and fine-tuning the software's functionality. As we are wrapping up this semester, we are establishing a clear vision for the next semester, which includes enhancing the capabilities of our software, creating the front-end with a user-friendly interface, setting up the back-end, and carrying the hospital buyer user data over to the database for scalability. Our complete first semester progress and accomplishments are provided below in detail, organized chronologically by month.

## 4.1 September 2023

Our team initiated the senior design project by thoroughly reviewing the client's documentation to grasp the project's overview. This comprehensive analysis helped us understand the client's vision and specific requirements. We then formulated an initial roadmap. For the current semester, our concentration would be on Data Collection and Analysis, AI Model Development, Testing, and Validation. The subsequent semester would shift focus towards User Interface and Integration aspects. A key decision made during this period was the selection of our primary tools for version control and project management. We chose GitHub for its robust version control capabilities. For project management and communication, Jira was selected for its comprehensive tracking and organizational features. In preparation for our initial meeting with the client, we dedicated time to formulate pertinent questions. These questions were crafted to clarify ambiguities in the project document and were focused on understanding the type of data the client could provide, identifying key features to extract from the data, and specifying the deliverables' requirements.

## 4.2 October 2023

On October 16th, our team had the initial meeting with the client. This meeting brought to light significant discrepancies between the client's actual expectations and the initial document they provided. A key insight from this meeting was the understanding that the client already operates a procurement system. However, this system relies heavily on human intervention, particularly in selecting alternative medications for ordering. Consequently, our revised objective became clear: to significantly minimize the time and effort involved in making decisions about alternative medication orders. While awaiting the client data necessary for building our model, we proactively scheduled a meeting with the client's procurement team that is responsible for manually selecting and ordering alternative medications. We prepared critical questions for the team, which revolved around the role of cost in their decision-making, the relationship between cost and bulk quantity, and the specific criteria they consider when selecting substitute medications. In the meantime, we engaged in research on the machine learning algorithms that we got recommended by alumni during Shark Tank. These included weak learning decision random forest, clustering algorithms, and regression method.

## 4.3 November 2023

In early November, we received the anticipated dataset from our client, which primarily consisted of purchase history records spanning from November 1, 2022, to October 31, 2023. However, upon analysis, it became evident that the data did not align with our initial expectations. Recognizing the constraints of our project timeline, we made a strategic decision to commence the prototype development phase. Concurrently, we started researching the client's suppliers and their pharmaceutical products, which were the types of information we needed for this project.

## 4.3.1 Prototype Development

To build the prototype, we had to dive into the contents of the client data, extract features, and think about which features to use. To do that, we wrote a python script and used pandas and matplotlib libraries to analyze the dataset. We specifically looked into 1) dimension of the data, 2) number of unique drugs ordered 3) common primary ingredients 4) average cost of each drug. We drew bar graphs to see the patterns in orders and identify any outliers. Our overarching objective was to learn how fast the drugs move, how long they stay in storage, in which seasons the demand increases for some of the drugs. In the midst of our data analysis process, we had another meeting with the client, where we all observed all of the operations that the hospital's procurement system is capable of. Consequently, our focus shifted a little bit. We created a more formal scheme of what our algorithm will do. In the end, our model would call the supplier data, and if the drug exists in the desired amount, which is 80% of the cases, it would pass the order in their system. If the drug isn't available in minimum quantity, following client's orders, model would wait until the inventory is updated, which is several times a day. If the model is not in stock at all, it would find replacements taking into account cost, active ingredient, and size. For the first prototype, we focused on the last possibility. Since the supplier data wasn't available, which was the most critical data for the team, the professors asked us to generate sample data for use in our first prototype. We devised our plan moving forwards, which was to identify key drugs, create a dummy order using these drugs, which would be used to run the model. To identify key drugs, we wrote a Python script that finds the most ordered active ingredients, and, for each active ingredient, it provides and maps to the drug name, package size, and average price. It stores them in a dictionary. We then saved these key drugs in a JSON file. While we were in the process, we had a trip to the client's office in which we saw the procurement system in person. We noticed the client had overlooked some of the important software that they use when considering alternatives and placing orders. We learned that there is a seperate software called ABC that provides alternatives with a click of a button based on active ingredient and cross-checks prices of alternatives from different suppliers. However, their software didn't take into account form, size, or packaging when providing alternatives. With that, our focus slightly changed once again. We created the diagram provided in the System Description section to follow for our final design. Due to the noisiness of the data, the cost-effective and package type comparison algorithms couldn't be created for the first prototype as the data didn't reflect cost and package type information accurately. Therefore, for the first prototype, we only focused on the exact match algorithm. The code for the prototype does the following:

1. The code begins by assigning generic\_name and reference\_item\_number for the drug we want to find alternatives containing the active ingredient of interest. The user sets the variables generic\_name and reference\_item\_number for the aforementioned desired product.
2. The dataset is loaded from a CSV file named 'Daily Snapshot.csv' using pandas, a Python data analysis library. The initial part of the code is dedicated to cleaning and reorganizing the dataset. This includes removing specific columns, reordering columns, and splitting column values for better clarity and analysis.
3. The 'Generic Description' column is split into 'Generic Name' and 'Form', and the original column is dropped. Similar processing is done for the 'Description' column, splitting it into 'Name' and 'Size'. The dataset is filtered to remove rows with empty 'Generic Name'.
4. The dataset is further filtered to focus on items with the specified generic name.
5. The code employs TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and cosine similarity measures to find drug form and size similarities. A reference item is selected based on the provided reference item number, and its form and size are used for comparison. The dataset is filtered to retain only those highly similar in form and size to the reference item.
6. Finally, the items with high similarity scores are sorted based on their size similarity, providing a clear view of the most similar items to the reference drug.

## 4.3.2 Testing and Discussion of Key Results

After successfully developing the first prototype, we conducted various tests on 3 different active ingredients and 9 different drugs in total. The active ingredient-drug code combinations we tested are as follows: lidocaine - 10090846, lidocaine - 10104126, lidocaine - 10136471, cefepime - 10175647, cefepime - 10267015, cefepime - 10027431, ondansetron - 10096566, ondansetron - 10112532, ondansetron - 10126451. In our tests, we aimed to assess data cleaning and preprocessing Accuracy, efficiency in data transformation, similarity score precision, filtering and sorting capability. To run our tests, we followed the steps given below:

1. Pick an active ingredient from the client data and set the ‘generic\_name’ variable to the chosen active ingredient (Test cases are provided at the top of the Jupyter notebook).
2. Pick the 8-digit product code of the drug that contains the aforementioned active ingredient that you want to find alternatives to and set the ‘reference\_item\_number’ to this value.
3. Press run all.
4. The algorithm filters the drugs initially by the active ingredient and then the administration method. It then uses cosine similarity to calculate the similarity based on size, which contains information about the concentration and dosage.
5. The table at the bottom shows the drug we’re finding alternatives to in the first row, followed by the details of the alternatives with form and size similarities above 0.9 in descending order.

As per the outputs we observed, we concluded that we were able to achieve our aforementioned goals. The outputs showed that the prototype is able to filter out non-matching drugs, filter by form and size, display alternative drug options with very similar specifications, and is modular in the sense that it can be used with different drugs from the dataset. Our findings suggested that, due to the stringent size similarity criteria, in many cases, the algorithm was unable to find a valid alternative. This indicated that the majority of the supplier’s drugs are not extremely similar to each other in size and that a reasonable similarity margin should be determined for each drug before applying cosine similarity. Therefore, a machine learning model must be built to intelligently calculate a size similarity threshold for each drug so that the algorithm yields a viable range of alternatives to consider.

## 4.4 December 2023

As the first semester of our senior design project drew to a close, the month of December 2023 was focused on evaluating the testing results of our initial prototype and setting clear, achievable milestones for the upcoming semester. These milestones are as follows:

1. Create a platform to include our product
2. Make the supplier data available in real time and automatically
3. Gain access to client software’s API
4. Create a cost-effective algorithm
5. Create the package effective algorithm
6. Run accuracy checks with the procurement team

*Written by:* Bora Bulut

# 5 Technical Plan

## 5.1 Proposed Technical Solution

The proposed technical solution outlined in this section represents a strategically designed architecture (Fig. A from Section 3.1) to implement our application. This concise overview will dissect the rationale behind each component choice and their collaborative function to meet and adapt to the dynamic needs of our client.

**5.1.1 Front-End:**

The choice of React for front-end development is motivated by its component-based architecture, which allows for creating dynamic user interfaces with high performance and responsive design. React's virtual DOM (Document Object Model) ensures efficient updates and rendering, making the UI highly interactive and smooth, which is crucial for maintaining an optimal user experience in complex systems like pharmaceutical procurement platforms where accurate data representation is vital.

Hosting the front end on Vercel provides benefits such as edge caching and global distribution, which minimizes latency and ensures that the application loads quickly from anywhere in the world. Vercel's continuous deployment capabilities also allow for seamless updates to the production environment, essential for maintaining the integrity and up-to-dateness of the application in a field where compliance and regulation changes can necessitate frequent updates.

**5.1.2 Back-End API:**

Utilizing Flask, a micro web framework written in Python, for the API layer offers simplicity and flexibility. It's lightweight and easy to start yet powerful enough to support complex applications. Flask applications are known for their scalability and quick time to market, which align with our client's demands for rapid deployment and the ability to adapt to changing requirements.

Amazon EC2 is famous for hosting such APIs due to its scalable computing capacity. It allows systems to increase or decrease resource allocation based on demand, which is particularly useful in the pharmaceutical industry, where procurement needs can vary drastically based on factors like epidemic outbreaks, seasonal demand, and supply chain disruptions.

**5.1.3 Back-End Data Storage:**

PostgreSQL is a robust open-source relational database system that provides strong ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring the reliability and integrity of user data — a non-negotiable requirement in the pharmaceutical sector. Hosting the database on Amazon RDS offers managed services like automated backups, patch management, and high availability. These features help maintain the system's resilience and data integrity, reducing the risk of data loss or downtime, which can have severe implications in healthcare settings.

An S3 bucket stores the supplier data that feeds the machine-learning model. The choice of S3 is likely due to its durability, availability, and scalability. It's designed for 99.999999999% (11 9's) of durability, which is crucial for ensuring that critical procurement data is not lost. Moreover, the object storage service is highly scalable, allowing for storing and retrieving any amount of data, which is essential for machine learning models that may require large datasets for accurate predictions.

**5.1.4 Model:**

Encapsulating the machine learning model within an AWS Lambda function is a strategic move toward serverless architecture. This approach has several advantages, including cost-effectiveness, as it runs code in response to events and automatically manages the computing resources, charging only for the computing time consumed. This can be particularly cost-efficient for applications with variable workloads, such as pharmaceutical procurement, where demand can fluctuate.

Lambda also offers rapid scaling, which is essential when the model may need to handle a surge in data processing requests, such as during health crises when predictive analytics become critical for decision-making. The serverless model further abstracts away infrastructure management tasks like server or cluster provisioning, patching, operating system maintenance, and capacity provisioning, allowing the pharmaceutical organization to focus on core business activities rather than IT infrastructure management.

**5.2 Technical Implementation Plan**

**5.2.1 Task 1. Machine Learning Model Development**

The machine learning model will be developed and trained to find drug replacements based on historical and possibly real-time data. It will be encapsulated in an AWS Lambda function for serverless execution. The deliverable is a trained, tested, and deployed model within AWS Lambda, ready for integration.

Lead: Bora Bulut

**5.2.2 Task 2. Back-End API Construction**

A robust Flask-based API will be constructed, documented, and tested to handle data transactions between the front-end, the database, and the machine learning model. It will be designed for scalability and security, with well-defined endpoints for each system interaction. The deliverable will be a well-documented API deployed to an Amazon EC2 instance.

Lead: Manuel Segimón

**5.2.3 Task 3. Database Setup and Integration**

A PostgreSQL database will be set up on Amazon RDS, with schemas designed to store and retrieve user data efficiently. Integration testing with the Flask API will ensure data integrity and transaction reliability. The deliverable will be a fully functional database with a connected API.

Lead: Taha Ababou

**5.2.4 Task 4. Data Storage and Management**

An Amazon S3 bucket will be configured for the secure and scalable storage of input data for the machine-learning model. Data management procedures will be established, including automated backups and lifecycle policies. The deliverable is a secure, accessible S3 bucket with governance policies in place.

Lead: Manuel Segimón

**5.2.5 Task 5. Front-End Interface Development**

The front-end user interface will be designed, developed, and user-tested to ensure an intuitive and responsive experience. It will be implemented using React and must adhere to accessibility and usability standards. The deliverable will be a fully functional front-end application ready for integration with the back-end API.

Lead: Taha Ababou

**5.2.6 Task 6. System Integration and Testing**

The front-end, back-end, database, and machine learning components will be integrated into a cohesive system. End-to-end testing will be conducted to ensure all parts work harmoniously. The deliverable is a fully integrated system that passes all functional tests.

Lead: Bora Bulut

**5.2.7 Task 7. Performance Optimization**

System performance will be analyzed and optimized for speed, efficiency, and cost-effectiveness. Load testing will be conducted to ensure the system can handle expected traffic volumes. The deliverable is an optimized system that meets performance benchmarks.

Lead: Joel Akerman

**5.2.8 Task 8. Security Audit and Compliance Check**

A comprehensive security audit will be performed to ensure the system's security measures are robust and meet industry standards. A compliance check will also ensure the system adheres to relevant healthcare regulations. The deliverable is an audit report and a list of remediations if necessary.

Lead: Zayian Muhammad

**5.2.9 Task 9. User Training and Documentation**

Training materials will be developed, and sessions will be conducted to ensure end-users can use the system effectively. Comprehensive system documentation will also be created. The deliverable is a set of training materials and system documentation.

Lead: Joel Akerman

**5.2.10 Task 10. Final Review and Deployment**

A final review of the entire system will be conducted to ensure all components are functioning as expected. The system will then be deployed to the production environment. The deliverable is a deployed system ready for live use.

Lead: Manuel Segimón

**5.2.11 Task 11. Maintenance Documentation and Technical Handoff**

Comprehensive maintenance documentation will be created, detailing the system architecture, codebase, APIs, and third-party services. This will include routine maintenance schedules, troubleshooting guides, and escalation protocols. Additionally, a technical handoff will be conducted to transfer knowledge to the operational team responsible for ongoing maintenance. The deliverable is a complete set of maintenance documentation and a successful knowledge transfer to the hospital IT team.

Lead: Bora Bulut

**5.3 Technical Implementation Milestones**

Task 1. Machine Learning Model Development by February 5, 2024.

Task 2. Back-End API Construction by February 22, 2024.

Task 3. Database Setup and Integration by February 22, 2024.

Task 4. Data Storage and Management by February 22, 2024.

Task 5. Front-End Interface Development by March 7, 2024.

Task 6. System Integration and Testing by March 18, 2024.

Task 7. Performance Optimization by March 25, 2024.

Task 8. Security Audit and Compliance Check by April 1, 2024.

Task 9. User Training and Documentation by April 8, 2024.

Task 10. Final Review and Deployment by April 15, 2024.

Task 11. Maintenance Documentation and Technical Handoff by May 1, 2024.

*Written by:* Manuel Segimón

# 6 Budget Estimate

1. **React and Node.js for Front End Development:**
   * Both React and Node.js are open-source and free to use.
2. **Hosting for React Application (GitHub Pages or Netlify):**
   * **GitHub Pages:** Offers free hosting for public repositories, which is ideal for our open-source projects.
   * **Netlify:** Its free tier is typically sufficient for small to medium-sized projects, aligning well with our initial requirements.
3. **Flask for Back End Development:**
   * Flask, being an open-source micro web framework, is free to use. Our primary focus here is on the hosting aspect.
4. **Heroku for Hosting Flask Application and PostgreSQL Database:**
   * **Heroku:** The free tiers for application hosting and PostgreSQL databases are limited in terms of dyno hours and database size. These should serve our initial needs well, but we'll keep an eye on potential scaling.
5. **AWS Services:**
   * **Amazon EC2:** The free tier offers limited compute hours suitable for low-traffic applications or development environments.
   * **AWS S3:** Provides a free tier with some storage and request limitations.
   * **AWS Lambda:** The free tier includes a number of requests and computation time, which should be adequate for our early stages.
6. **Google Colab for Machine Learning Model Development:**
   * Google Colab's free environment, with its GPU and TPU usage limits, will be a valuable asset for our development and small-scale model training.
7. **Database (PostgreSQL or MySQL):**
   * Both PostgreSQL and MySQL are open-source and free. We'll need to be mindful of potential costs if we host them on services like Heroku or AWS and exceed the free limits.
8. **Miscellaneous**:
   * Contingency Fund: $100

### **Total Estimated Budget: 100$**

### **Additional Notes:** The budget estimate is based on a 6-month timeline, considering the costs for cloud services, software development tools, and other essentials. Our approach focuses on leveraging free tiers and open-source tools to minimize costs. We understand that while these tools themselves may not add to our budget, associated hosting or operational services could. We are prepared to regularly review our usage and the latest service terms to ensure our project remains within budget while considering potential future scaling needs.

*Written by:* Joel Akerman

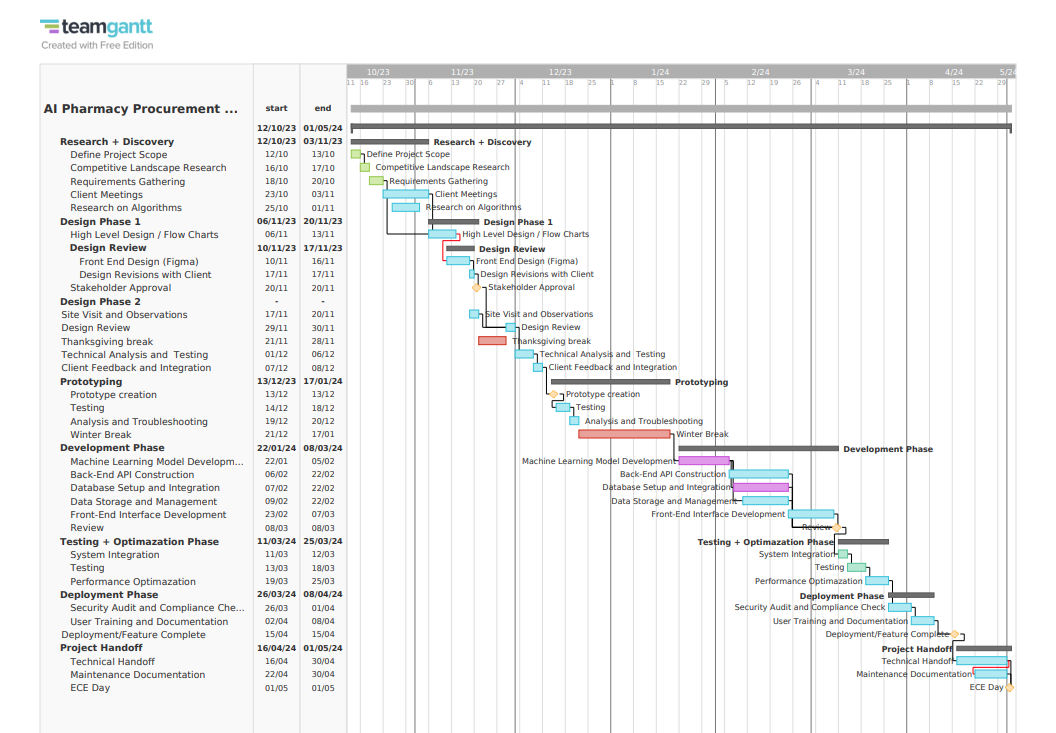
# 7 Attachments

# 7.1 Appendix 1 – Engineering Requirements

1. **Product Similarity Analysis Accuracy**
   * Requirement: The software shall accurately analyze product similarity.
   * Value: Error rate less than 10^-11 for normal product matching.
   * Tolerance: None.
   * Units: Error rate (mismatched drug identifiers).
2. **System Response Time**
   * Requirement: The software shall have a maximum response time for processing queries.
   * Value: Not more than 3 seconds for standard queries.
   * Tolerance: +/- 0.5 seconds.
   * Units: Seconds.
3. **User Interface Load Time**
   * Requirement: The user interface shall load within a specific time frame.
   * Value: Less than 2 seconds.
   * Tolerance: +/- 0.5 seconds.
   * Units: Seconds.
4. **Integration Compatibility**
   * Requirement: The software shall be compatible with specific external systems.
   * Value: Integration with the hospital's internal OmniCell data and external wholesaler APIs.
   * Tolerance: None.
   * Units: N/A (compatibility requirement).
5. **User Interface Usability Score**
   * Requirement: The user interface shall achieve a minimum usability score.
   * Value: Score of 80/100 based on standardized usability testing.
   * Tolerance: -5 points.
   * Units: Usability score (out of 100).
6. **Cloud Data Processing and Management**
   * **Requirement:** The software shall be capable of efficiently processing and managing data in a cloud-based environment.
   * **Value:** The software must support continuous data processing with minimal latency, handling up to 1 TB of data with cloud-based operations.
   * **Tolerance:** +200 GB, -100 GB.
   * **Units:** Terabytes (TB) for data volume; milliseconds (ms) for latency.
   * **Additional Specification:**
     + **Cloud Integration:** Ensure seamless integration with cloud services for data storage, retrieval, and real-time data processing.
     + **Data Security and Compliance:** Implement robust security protocols and comply with relevant data protection regulations for cloud-based data handling.
     + **Scalability:** The system should be scalable to accommodate increases in data volume and complexity without significant degradation in performance.

*Written by:* [Zaiyan Muhammad](mailto:zbu22@icloud.com)

# 7.2 Appendix 2 – Gantt Chart



*Written by:* Zaiyan Muhammad

**Acknowledgment**

The authors wish to thank Professors Osama Alshaykh, Alan Pisano, Michael Hirsch, and Gordon Hubbard of the Beth Israel Deaconess Medical Center for their continuous support throughout the project.

**References**

[1] G. Ledger, “Facing Pharmaceutical Supply Challenges With Artificial Intelligence,” HealthTrust - Performance Improvement For Healthcare, Feb. 01, 2021. https://healthtrustpg.com/thesource/cqo/clinically-integrated-supply-chain/facing-pharmaceutical-supply-challenges-with-artificial-intelligence/ (accessed Oct. 20, 2023).

[2] Arkestro, “Global pharma firm leverages AI first offers to drive savings and shorten cycle time,” Arkestro, Mar. 13, 2023. https://arkestro.com/case-studies/global-pharma-firm-leverages-ai-first-offers-to-drive-savings-and-shorten-cycle-time/ (accessed Oct. 20, 2023).

‌[3] “7 Ways AI is Transforming Procurement in Healthcare,” *Vamstar*. https://vamstar.io/my-resources/7-ways-ai-is-transforming-procurement-in-healthcare/ (accessed Oct. 20, 2023).

‌[4] D. Owczarek, “Inefficiencies in pharmaceutical supply chain cost. Leveraging AI in drug supply chain management - nexocode,” nexocode, Mar. 30, 2021. Available: https://nexocode.com/blog/posts/ai-in-drug-supply-chain/

‌