DoseInfo

JID 4204

Alexander Yoon, Robert Williams Scott Powers, Suhyun Lee

Client: Dr. Rebecca Steinberg, Resident Physician Emory University School of Medicine

Repository: https://github.com/sue0-si/Medication_Conversion_4204

Table of Contents

Table of Contents 2

Table of Figures 4

Terminology 5

Introduction 7

System Architecture 9

- Goals and Objectives 9

Static System Architecture 10

- Diagram 10
- Description 10

Dynamic System Architecture 12

- Diagram 12
- Description 13
  - Interaction Flow 13

Component Design 15

- Static Design 16
  - Diagram 16
  - Description 16
- Dynamic Design 18
  - Diagram 18

Data Design 20

- Introduction 20
- Diagram 20

UI Design 22

- Introduction 22
- UI Walkthrough 22

Appendix 26

- OpenFDA API 26

Resource URL: 26
Resource Information: 26
Parameters: 26
Example Request: 26
Example Response: 26
Response Fields 27
Wikipedia API 28
Resource URL: 28
Resource Information: 28
Parameters: 28
Example Request: 28
Example Response: 28
Response Fields 28
CancerCalc 29
Resource URL: 29
Resource Information: 29
Parameters: 29
Table Example: 29
Response Fields 30
MDCalc 30
Resource URL: 30
Resource Information: 30
Parameters: 30
Table Example: 30
Response Fields 31

Table of Figures

Figure Number	Figure Title	Page Number
1	Static System Architecture	9
2	Dynamic System Architecture	11
3	Static Component Design	15
4	Dynamic Component Design	17
5	Data Design	19
6	UI Design	22
7	Medication Search	23
8	Medication Information	23
9	PO-IV Conversion Page Screen	24
10	Conversion Result Page Screen	25

Terminology

Term	**Definition**	**Context**
API	Application Programming Interface; a set of rules and protocols for building and interacting with software applications.	Software Development
CDN	Content Delivery Network; A system of web hosting designed to provide fast and reliable service. By caching content on multiple servers located closer to end-users, CDNs reduce latency, improve load times, and enhance the overall performance and scalability of web applications.	Software Development
Hosting	The service of providing storage and access to websites or applications on a server, making them available on the internet.	Software Development
IV	Intravenous administration of medication	Medical Field
JSON	JavaScript Object Notation; a lightweight data interchange format that's easy for	Software Development

	humans to read and write, and easy for machines to parse and generate.	
PO	Per Os: Oral administration of medication	Medical Field
React	A JavaScript library for building user interfaces, particularly single-page applications, by creating reusable UI components	Software Development
SC	Subcutaneous administration of medication	Medical Field
Static	Refers to content that doesn't change or doesn't involve server-side processing; typically used in the context of web pages that are delivered to the user exactly as stored.	Software Development
HTTPS	Hypertext Transfer Protocol Secure; an extension of HTTP used for secure communication over a computer network.	Software Development
Conversion Factor	A ratio used to convert between different units or forms of medication administration.	Medical Field
Equi-analgesic Dose	The dosage of one drug that is pharmacologically equivalent in analgesic effect to another drug.	Medical Field

Introduction

Introduction

The DoseInfo application is a React-based web tool designed to assist healthcare professionals in efficiently converting medication dosages between oral (PO) and intravenous (IV) / subcutaneous (SC) forms. By leveraging external APIs like OpenFDA and DrugBank, the application provides access to detailed medication information, including adverse events, side effects, dosing, and conversion ratios. This ensures users can make informed decisions, even in cases of allergies or adverse events, where alternative medications may be necessary.

Background

The DoseInfo application operates as a static React web application hosted on a Content Delivery Network (CDN) for optimal performance, scalability, and reliability. All dynamic content is fetched in real-time using APIs, eliminating the need for server-side logic or dedicated databases. Internal JSON files serve as a fallback mechanism, containing essential conversion data and warnings to ensure uninterrupted functionality in the event of API unavailability. This design enables seamless medication conversions, user-friendly interfaces, and reliable data retrieval, catering to the needs of healthcare providers. With its modular architecture, DoseInfo combines efficiency, accuracy, and scalability in a single platform.

Document Summary

The *System Architecture* section provides a high-level view of the system and its major component interaction. Static system architecture diagram depicts logical relationships between major components. Dynamic system architecture diagram shows how each component behaves in runtime.

The *Component Design* section provides an in-depth exploration of the system's internal structure and behavior at the component level. Static design details the structural arrangement of components, their interfaces, and their relationships. Dynamic design illustrates how these components interact during system operation.

The *Data Storage Design* section outlines the mechanisms used for storing and exchanging data within the system. It contains a description of the data storage approach, covering the type of database, the structure of stored data, and any relevant technologies used. Also, it explains how data is transferred between system components by highlighting protocols, formats, and communication methods.

The *UI Design* section demonstrates how users interact with the application. It introduces the visuals of the application by showcasing all major screens, key features, and navigation flows.

Color Coding

Frontend Components (Light Blue): These represent the user interface and logic that the user directly interacts with. They handle data input, display results, and communicate with backend services.

Backend Services (Light Green): These represent the external APIs and data sources that provide necessary information to the frontend components. They handle data retrieval, conversions, and other backend logic that supports the frontend.

Internal Files (Purple): These represent the internal information files the application uses in the event API calls cannot return the necessary information. They contain conversion calculation and warning data for various drug types.

System Architecture

Goals and Objectives

Our application is a web-based tool designed for healthcare professionals to access medication information and perform dosage conversions. It operates as a static website built with modern technologies like React, leveraging external APIs for real-time data retrieval. To ensure reliability, the system also incorporates internal information files that provide conversion and warning data as a fallback when APIs are unavailable.
Hosted on a content delivery network (CDN), the application delivers fast loading times, scalability, and high availability. The React front end manages user interactions and directly communicates with external APIs, dynamically fetching medication and medical information. This serverless architecture eliminates the need for dedicated servers or databases, simplifying maintenance while maintaining flexibility. The inclusion of internal fallback files further enhances system resilience, ensuring uninterrupted functionality. With its static site setup, the application achieves superior performance, enhanced security, and optimal efficiency.

Static System Architecture

Diagram

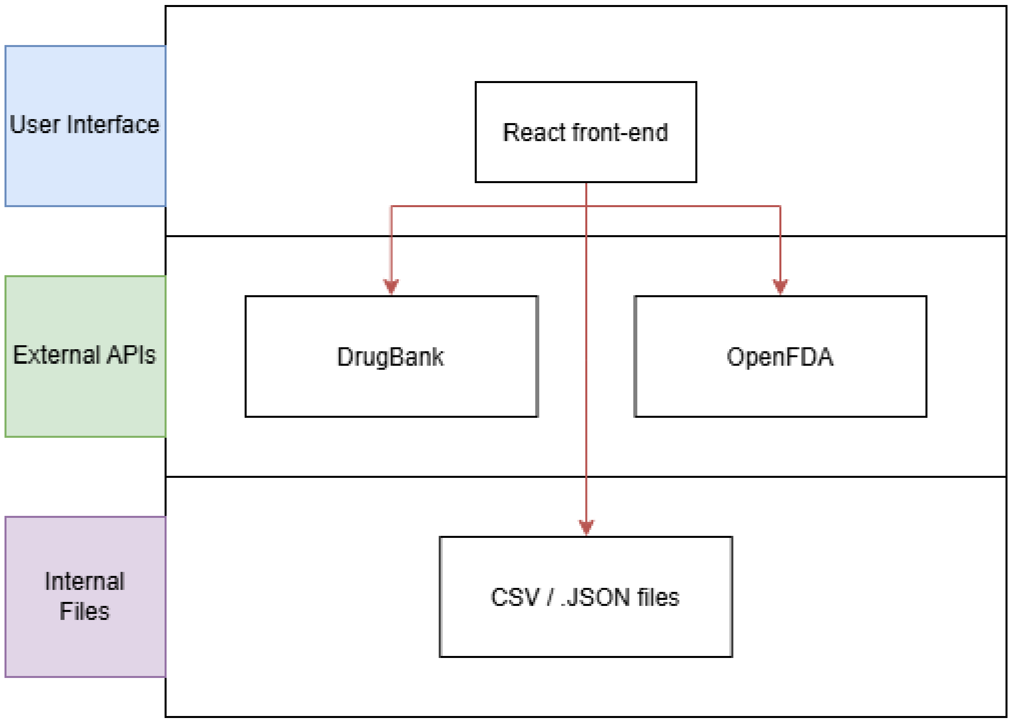


Figure 1

Description

The static architecture of our application is composed of three primary elements: the React front-end, external APIs, and local storage. These elements interact in a dependency-based structure to manage medication data retrieval.

The React front end is the core of the system, responsible for rendering the user interface and handling user interactions. All dynamic content is retrieved via API calls, but the React front-end is a self-sufficient system, operating independently once hosted.

DrugBank and OpenFDA are external data sources that supply critical medication and regulatory data. The React front-end depends on these APIs to provide the information needed to populate the user interface. These APIs are accessed directly from the client without a need for intermediary server-side logic. The system is designed to be flexible—if necessary, different APIs can be swapped in without changing the core front-end architecture, as the front-end is built to depend on external APIs for all dynamic data.

Internal files contain necessary conversion information such as warning criteria and conversion formulae for the application to use as defaults in the event of API failure.

The UML Component Diagram was chosen to represent the static architecture of the system because it clearly illustrates the high-level structure and dependencies between the main components. In this diagram:

- Arrows represent dependencies, showing how the React front-end relies on external APIs for data and on local storage for retaining user-specific information.
- The diagram effectively captures the fact that the system is static and front-end heavy, with dynamic content driven by API dependencies rather than internal server-side logic.

By using a component diagram, we focus on the structural organization and dependency relationships between the front end and its external systems, which is key to understanding the system’s architecture and its reliance on external data sources for dynamic content.

Dynamic System Architecture

Diagram

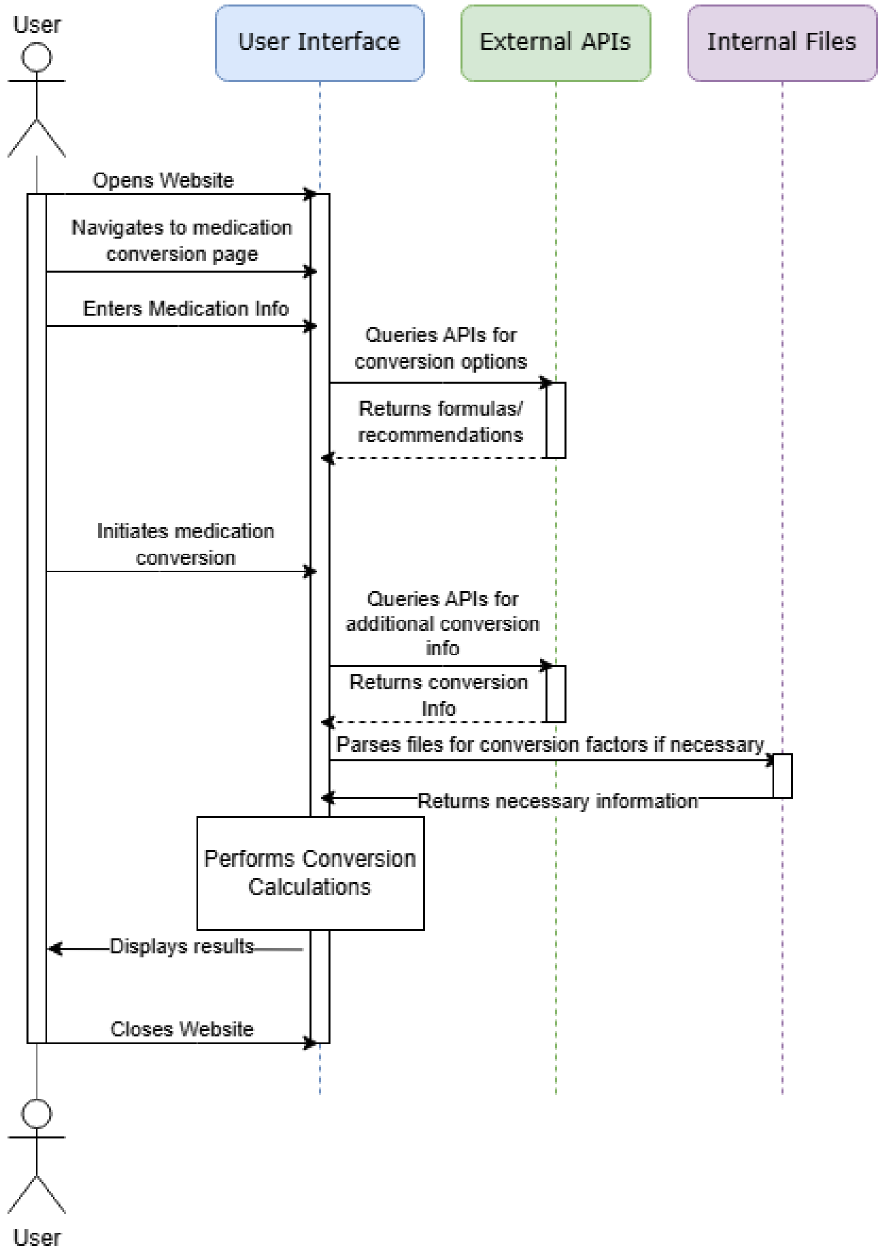

User
User Interface
External APIs
Internal Files
Opens Website
Navigates to medication conversion page
Enters Medication Info
Queries APIs for conversion options
Returns formulas/ recommendations
Initiates medication conversion
Queries APIs for additional conversion info
Returns conversion Info
Parses files for conversion factors if necessary
Returns necessary information
Performs Conversion Calculations
Displays results
Closes Website
User

Figure 2

Description

The dynamic architecture of the system is depicted in the System Sequence Diagram (SSD), which outlines the flow of interactions when a user performs a medication conversion using the application. The diagram illustrates the key interactions between the User, the User Interface, External APIs, and the User Device (local storage) throughout this process.

Interaction Flow

1. User Opens Website:
    - The user accesses the website, triggering the loading of the User Interface (React front-end). The system renders the main interface where the user can interact with the medication conversion functionality.
2. Navigation to Medication Conversion Page:
    - The user navigates to the medication conversion page within the User Interface, where they are prompted to input relevant medication information (such as dosage or type).
3. Entering Medication Information:
    - The user enters the medication details required for conversion. This data is processed by the front end, and the system prepares to query the necessary external APIs for conversion recommendations and options.
4. Querying External APIs for Conversion Options:
    - The User Interface sends requests to the External APIs (DrugBank, OpenFDA, etc.) to fetch conversion options or dosage recommendations based on the medication information provided by the user.
    - The external APIs respond with potential conversion formulas, which are sent back to the front end.
5. Initiating Medication Conversion:
    - The user initiates the conversion process by selecting one of the recommendations or entering further data, which prompts the User Interface to make another query to the External APIs for additional conversion information. If the APIs can’t provide necessary conversion information, internal files are consulted.
6. Performing Conversion Calculations:

    - After receiving the necessary data from the APIs, the User Interface performs the required conversion calculations based on the API results. The results are displayed to the user, providing accurate medication conversions or dosage information.
7. User Closes Website:
    - After reviewing or saving the conversion results, the user closes the website, ending the session.

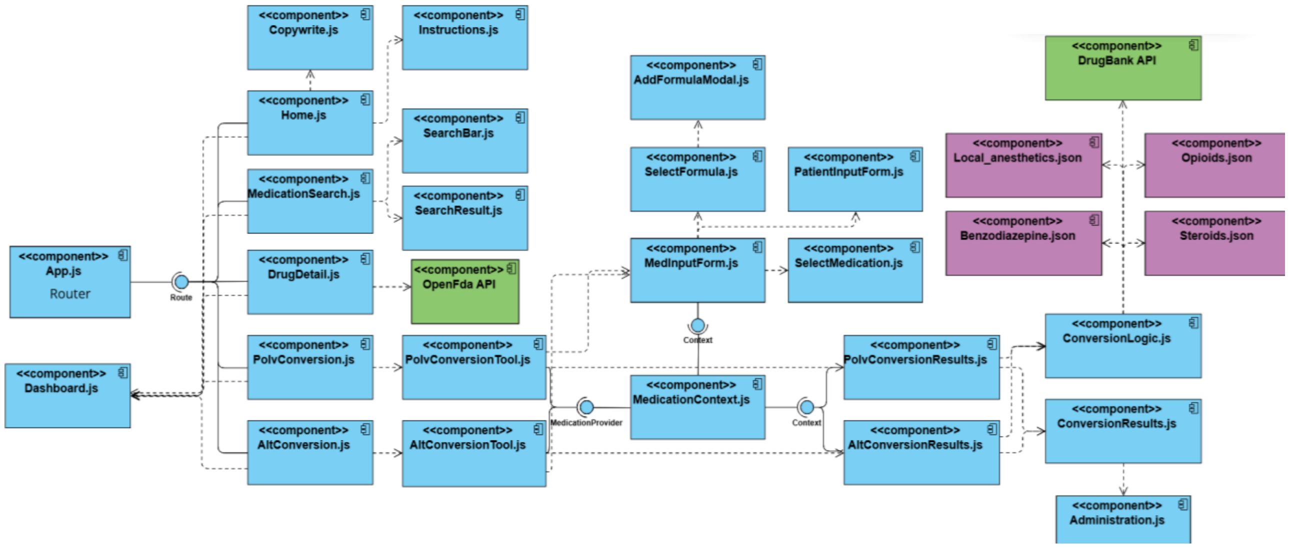
The System Sequence Diagram (SSD) was chosen to depict this dynamic interaction because it effectively captures the flow of communication between the user and the system, as well as the dependencies on external APIs for data retrieval. The SSD highlights the key steps in the process, from inputting medication information to performing calculations and saving results, making it an ideal choice for representing the system's runtime behavior.

Component Design

This section provides a detailed design of the medication conversion tool, building upon the system-level architecture. The goal here is to "zoom in" to the lower-level components of the system, presenting both the static structure (the organization and relationships between components) and the dynamic behavior (the interaction between these components at runtime). This section reinforces the conceptual integrity of the system and ensures coherence between the higher-level architecture and the detailed component design.

Static Design

Diagram



<<component>> Copywrite.js
<<component>> Instructions.js
<<component>> AddFormulaModal.js
<<component>> DrugBank API
<<component>> Home.js
<<component>> SearchBar.js
<<component>> SelectFormula.js
<<component>> PatientInputForm.js
<<component>> Local_anesthetics.json
<<component>> Opioids.json
<<component>> MedicationSearch.js
<<component>> SearchResult.js
<<component>> Benzodiazepine.json
<<component>> Steroids.json
<<component>> App.js
Router
Route
<<component>> DrugDetail.js
<<component>> OpenFda API
<<component>> MedInputForm.js
<<component>> SelectMedication.js
Context
<<component>> ConversionLogic.js
<<component>> Dashboard.js
<<component>> PolvConversion.js
<<component>> PolvConversionTool.js
<<component>> MedicationContext.js
<<component>> PolvConversionResults.js
MedicationProvider
Context
<<component>> ConversionResults.js
<<component>> AltConversion.js
<<component>> AltConversionTool.js
<<component>> AltConversionResults.js
<<component>> Administration.js

Figure 3

Description

This component diagram illustrates the static architecture of our application. Dashed lines and arrows represent dependency relationships, primarily in the form of component implementation and data dependency. Ball and socket relationships represent more complex data and state exchange interactions. The system is modular, with distinct components handling routing, user interaction, state management, and conversion logic. At the core, App.js serves as the entry point, using react-router-dom to display pages like Home, MedicationSearch, DrugDetail, PolvConversion, and AltConversion. Each page is composed of reusable components to manage user inputs and display data. For instance, MedicationSearch.js implements SearchBar.js for input

and SearchResult.js for display, while MedInputForm.js relies on SelectFormula.js, SelectMedication.js, and PatientInfoForm.js for handling complex input.
The application's conversion tools (PoIvConversionTool.js and AltConversionTool.js) rely on MedicationContext.js, which acts as a centralized state management system, ensuring seamless communication between input forms (e.g., MedInputForm.js) and result display components (e.g., PoIvConversionResults.js). Conversion calculations are managed by ConversionLogic.js, which processes data using user inputs, external APIs like DrugBank API, and internal fallback files such as steroids.json and opioids.json. All pages also implement the Dashboard.js component to handle navigation by triggering the react-router. This architecture, with its clean separation of concerns, enables dynamic and reliable functionality while maintaining scalability and ease of maintenance.

Dynamic Design

Diagram

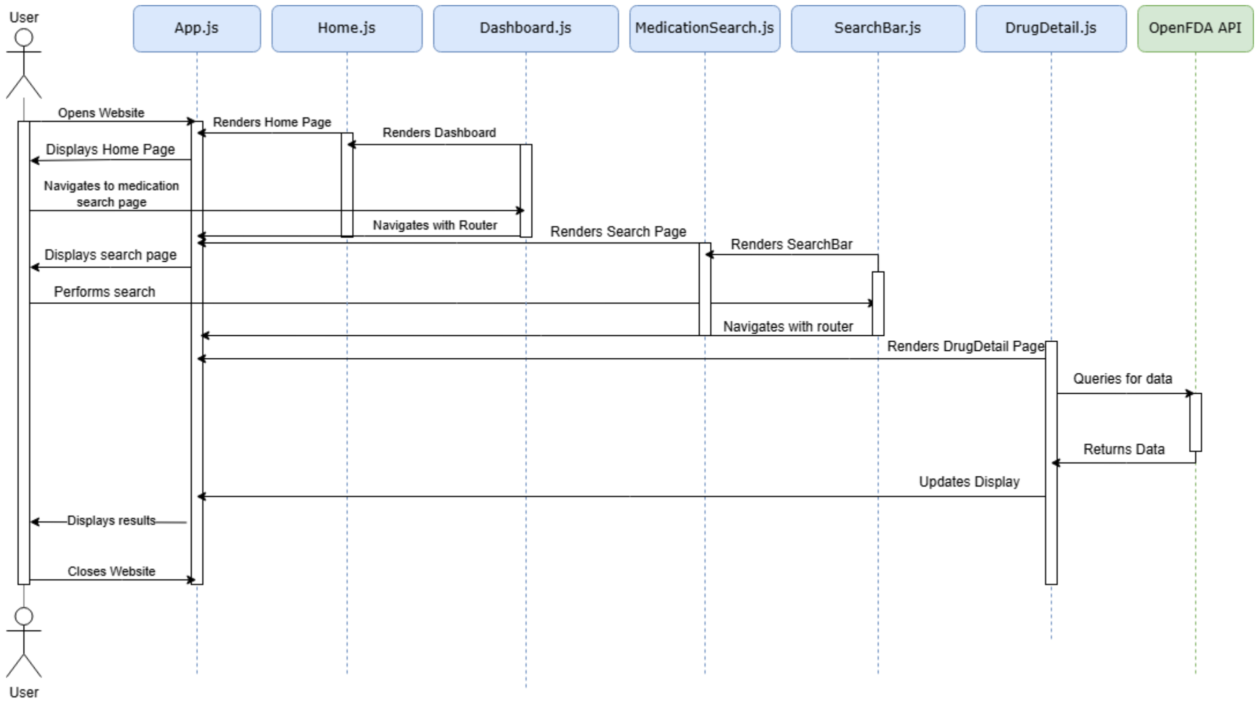


Figure 4

Description

This system sequence diagram demonstrates the dynamic interaction within the application when using the Medication Search feature. It showcases the modular architecture, with App.js serving as the central controller, dynamically rendering pages like Home.js, MedicationSearch.js, and DrugDetail.js using react-router. User navigation is facilitated through the Dashboard.js component, which triggers route updates, rendering components such as MedicationSearch.js for input and DrugDetail.js for displaying detailed information. The system efficiently integrates the OpenFDA API, enabling real-time retrieval of medication data, which is dynamically displayed to the user. This diagram highlights the clean separation of concerns, with each component handling specific tasks, ensuring maintainability and scalability, while delivering a seamless and interactive user experience.

Interaction Flow

1. Opening the Application:
    a. The user opens the application, initiating the App.js component.
    b. The Home.js page is rendered through App.js, which includes the Dashboard.js component as part of its layout.
2. Navigating to the MedicationSearch Page:
    a. The user navigates to the MedicationSearch feature via the Dashboard.js component.
    b. This triggers the react-router within App.js, updating the route to display the MedicationSearch.js page.
3. Rendering the MedicationSearch Page:
    a. The MedicationSearch.js page renders the SearchBar.js component, enabling the user to input search queries.
4. Performing a Search:
    a. The user interacts with the SearchBar.js to perform a search. The input from the user triggers a route update through react-router to display the Drug Detail page.
5. Navigating to the Drug Detail Page:
    a. The DrugDetail.js page is rendered via App.js, displaying details about the selected medication.
6. Querying OpenFDA API:
    a. The DrugDetail.js component queries the OpenFDA API for detailed information about the selected drug.
    b. The API returns the requested data, which is used to update the display dynamically within the DrugDetail.js component.
7. Closing the Application:
    a. After viewing the required information, the user closes the application, concluding the interaction.

Data Design

Introduction

The diagram below shows how we interact with JSON files. This method of data storage does not involve SQL but resembles key-value pair data schema. The data is stored locally in the application and passed into the corresponding JSON file to retrieve the conversion data in JSON file.

Diagram

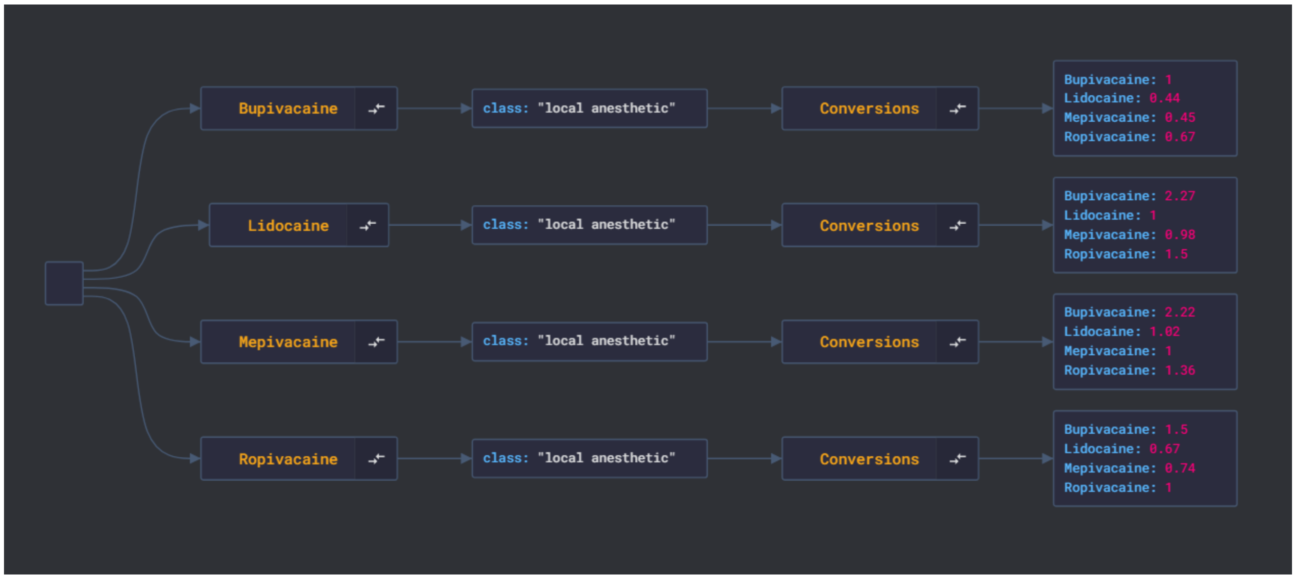


Figure 5

The above diagram shows the drug conversion JSON file structure where the initial key is a drug name which points to a class element providing the class of the drug. This allows the application to retrieve the class of two drugs to determine if a drug conversion is possible. Each drug object indexed by the drug names contains the drug class and the conversion ratios to every other drug in the same class.

File Use

Our application relies on medication warning and conversion data and utilizes JSON files to store this information. They were created manually using online medication conversion datasets and are stored within our projects source, so that they can be accessed by the application without use of an API.

Data Exchange

Our application uses OpenFDA API to retrieve medication data for the medication look-up page. We use Axios, which is a promised-based HTTP (Hypertext Transfer Protocol) client for node.js to retrieve desired data from the API. With this data, we convert it to actual data with JSON (JavaScript Object Notation) and pass it to the client.

Security Concerns

Our application collects patient data, including Personally Identifiable Information (PII) such as height, medical conditions, age, and gender, to process queries. To ensure privacy and security, the application does not store patient data at any stage, and it is never linked to patient names or other identifiers. All data transmitted between the client and server is secured using HTTPS to prevent interception or unauthorized access. Additionally, patient data is encrypted during transmission to safeguard it from potential breaches.

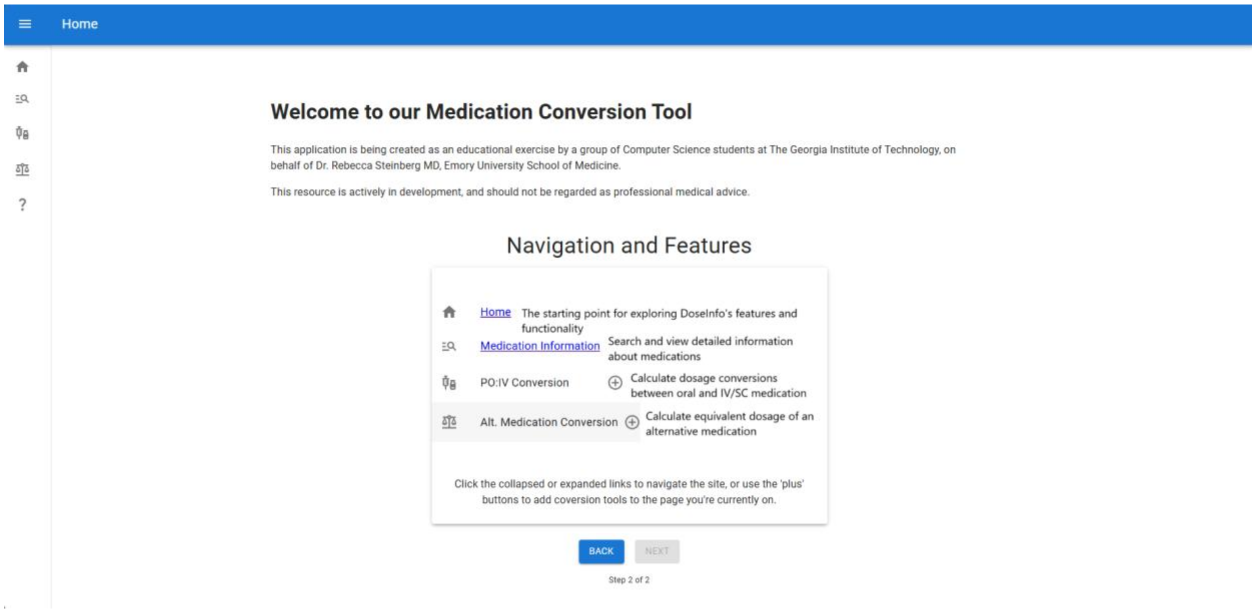
UI Design

Introduction

In this section, we present the User Interface (UI) of our Medication Conversion App, designed to provide users with an intuitive and efficient way to convert dosages across different medications. We discuss our UI design decisions, ensuring ease of use and clarity, and provide an overview of the major screens that users will interact with throughout their experience.

UI Walkthrough

When a user first opens the application, they are greeted with a Welcome Screen (Figure 6) that provides a straightforward overview of the app's purpose: converting medication dosages from intravenous (IV) to oral and vice versa. This screen adheres to Nielsen's Visibility of system status heuristic by prominently displaying the title and the description of the website indicating that the system is functioning and ready to proceed.



Welcome to our Medication Conversion Tool

This application is being created as an educational exercise by a group of Computer Science students at The Georgia Institute of Technology, on behalf of Dr. Rebecca Steinberg MD, Emory University School of Medicine.

This resource is actively in development, and should not be regarded as professional medical advice.

Navigation and Features

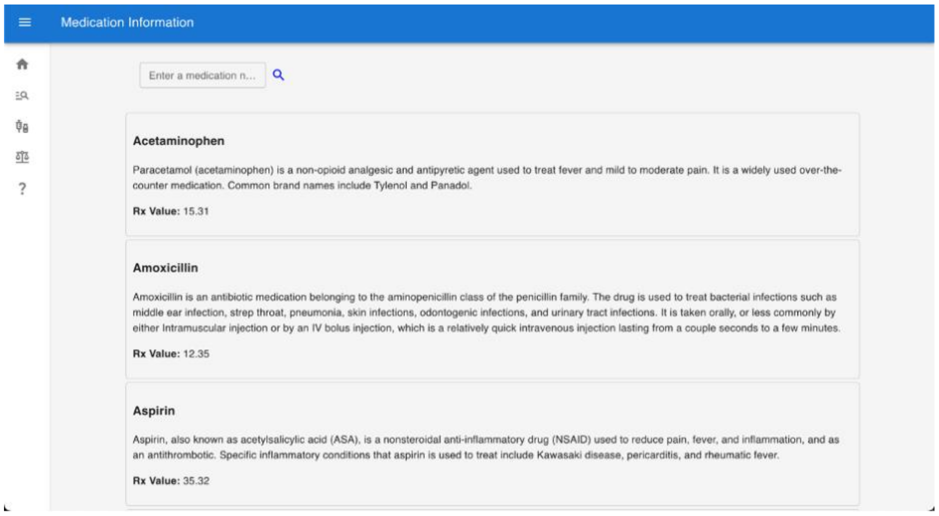
Home	The starting point for exploring DoseInfo's features and functionality
Medication Information	Search and view detailed information about medications
PO:IV Conversion	Calculate dosage conversions between oral and IV/SC medication
Alt. Medication Conversion	Calculate equivalent dosage of an alternative medication

Click the collapsed or expanded links to navigate the site, or use the 'plus' buttons to add coversion tools to the page you're currently on.

Figure 6: Home Page Screen

Users can expand the navigation tab to view the available resources and change pages or add a conversion component directly to their current page.

Users can navigate to the page below (Figure 7) using the second icon on the navigation bar. From this page users can search for specific drugs to get general information about medications. After clicking a box corresponding to a specific drug (or searching for the drug in the search bar at the top), users are presented with a page with information about the specific drug including dosage guidelines, side effects, drug interactions, and treatment indication (Figure 8).



Acetaminophen

Paracetamol (acetaminophen) is a non-opioid analgesic and antipyretic agent used to treat fever and mild to moderate pain. It is a widely used over-the-counter medication. Common brand names include Tylenol and Panadol.

Rx Value: 15.31

Amoxicillin

Amoxicillin is an antibiotic medication belonging to the aminopenicillin class of the penicillin family. The drug is used to treat bacterial infections such as middle ear infection, strep throat, pneumonia, skin infections, odontogenic infections, and urinary tract infections. It is taken orally, or less commonly by either Intramuscular injection or by an IV bolus injection, which is a relatively quick intravenous injection lasting from a couple seconds to a few minutes.

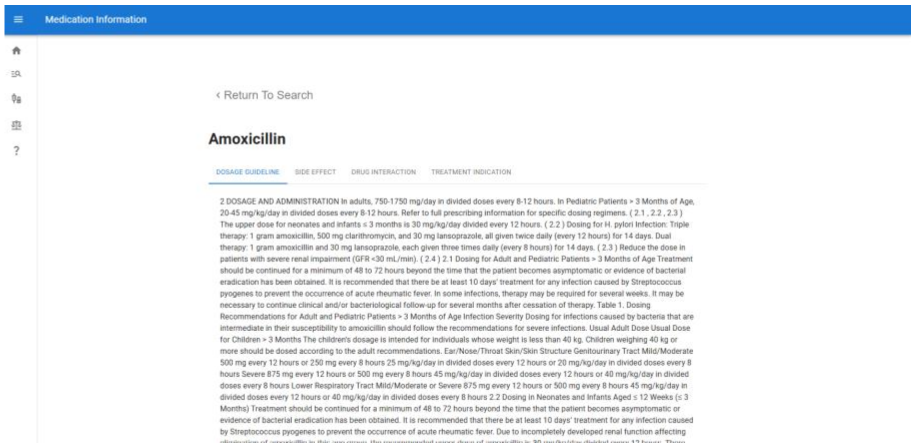
Rx Value: 12.35

Aspirin

Aspirin, also known as acetylsalicylic acid (ASA), is a nonsteroidal anti-inflammatory drug (NSAID) used to reduce pain, fever, and inflammation, and as an antithrombotic. Specific inflammatory conditions that aspirin is used to treat include Kawasaki disease, pericarditis, and rheumatic fever.

Rx Value: 35.32

Figure 7: Medication Search



Amoxicillin

2 DOSAGE AND ADMINISTRATION In adults, 750-1750 mg/day in divided doses every 8-12 hours. In Pediatric Patients > 3 Months of Age, 20-45 mg/kg/day in divided doses every 8-12 hours. Refer to full prescribing information for specific dosing regimens. (2.1 , 2.2 , 2.3) The upper dose for neonates and infants ≤ 3 months is 30 mg/kg/day divided every 12 hours. (2.2) Dosing for H. pylori Infection: Triple therapy: 1 gram amoxicillin, 500 mg clarithromycin, and 30 mg lansoprazole, all given twice daily (every 12 hours) for 14 days. Dual therapy: 1 gram amoxicillin and 30 mg lansoprazole, each given three times daily (every 8 hours) for 14 days. (2.3) Reduce the dose in patients with severe renal impairment (GFR <30 mL/min). (2.4) 2.1 Dosing for Adult and Pediatric Patients > 3 Months of Age Treatment should be continued for a minimum of 48 to 72 hours beyond the time that the patient becomes asymptomatic or evidence of bacterial eradication has been obtained. It is recommended that there be at least 10 days' treatment for any infection caused by Streptococcus pyogenes to prevent the occurrence of acute rheumatic fever. In some infections, therapy may be required for several weeks. It may be necessary to continue clinical and/or bacteriological follow-up for several months after cessation of therapy. Table 1. Dosing Recommendations for Adult and Pediatric Patients > 3 Months of Age Infection Severity Dosing for infections caused by bacteria that are intermediate in their susceptibility to amoxicillin should follow the recommendations for severe infections. Usual Adult Dose Usual Dose for Children > 3 Months The children's dosage is intended for individuals whose weight is less than 40 kg. Children weighing 40 kg or more should be dosed according to the adult recommendations. Ear/Nose/Throat Skin/Skin Structure Genitourinary Tract Mild/Moderate 500 mg every 12 hours or 250 mg every 8 hours 25 mg/kg/day in divided doses every 12 hours or 20 mg/kg/day in divided doses every 8 hours Severe 875 mg every 12 hours or 500 mg every 8 hours 45 mg/kg/day in divided doses every 12 hours or 40 mg/kg/day in divided doses every 8 hours Lower Respiratory Tract Mild/Moderate or Severe 875 mg every 12 hours or 500 mg every 8 hours 45 mg/kg/day in divided doses every 12 hours or 40 mg/kg/day in divided doses every 8 hours 2.2 Dosing in Neonates and Infants Aged ≤ 12 Weeks (≤ 3 Months) Treatment should be continued for a minimum of 48 to 72 hours beyond the time that the patient becomes asymptomatic or evidence of bacterial eradication has been obtained. It is recommended that there be at least 10 days' treatment for any infection caused by Streptococcus pyogenes to prevent the occurrence of acute rheumatic fever. Due to incompletely developed renal function affecting

Figure 8: Medication Information

Selecting the third icon takes the user to the Conversion Selection Screen (Figure 9), where they can choose between "IV to Oral" or "Oral to IV" conversion formulas. Each formula appears in a drop-down menu for easy selection, ensuring the interface aligns with the User Control and Freedom heuristic. Additionally, this screen includes a “Back” button and a confirmation pop-up to support Error Prevention in case the user mistakenly navigates away.

Users can also enter information such as dosage, weight, and medication type, and the screen incorporates familiar medical terms and units to create a Match Between System and the Real World, enhancing usability for medical professionals. A feature to add patient-specific information, such as height, weight, gender, and conditions, further customizes the experience, allowing warnings to align with Recognition Rather Than Recall for specific patient populations.

When users click the "Select Target Medication" input field, a filtered list of valid target medications appears, showing only those compatible with the selected source medication. This complies with the Error Prevention heuristic, minimizing error-prone conditions. The conversion formula section also offers Flexibility and Efficiency of Use by allowing advanced users to enter custom formulas for specific medication pairs or administration routes.

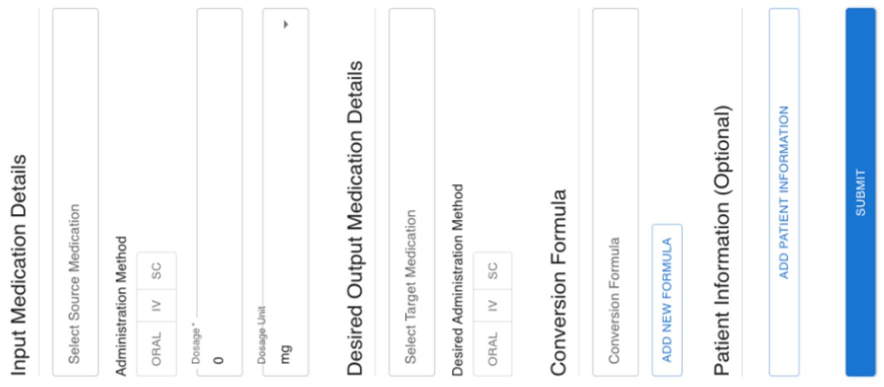
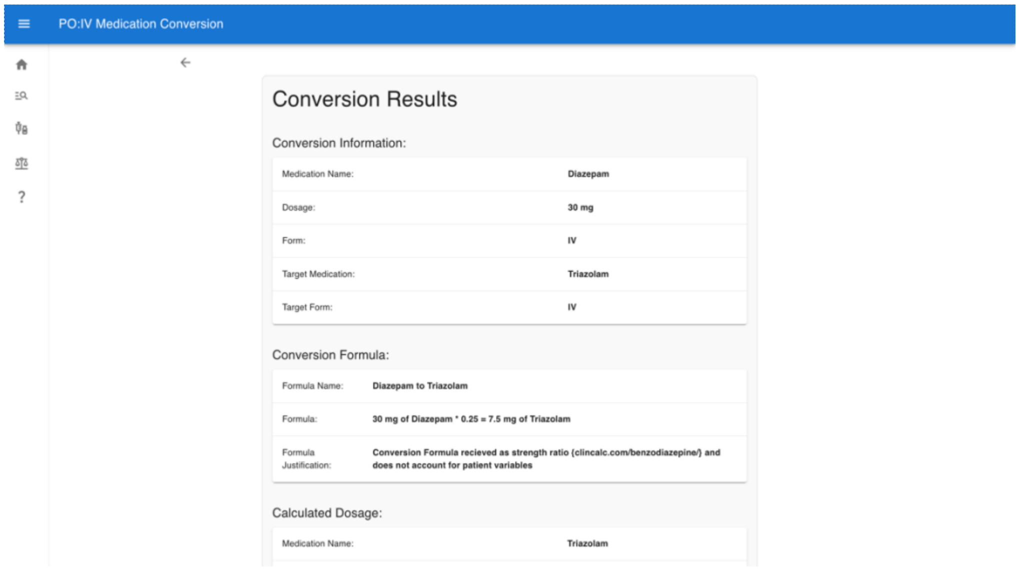


Figure 9: PO-IV Conversion Page Screen
Rotated 90° for formatting



Conversion Results

Conversion Information:

Medication Name:	Diazepam
Dosage:	30 mg
Form:	IV
Target Medication:	Triazolam
Target Form:	IV

Conversion Formula:

Formula Name:	Diazepam to Triazolam
Formula:	30 mg of Diazepam * 0.25 = 7.5 mg of Triazolam
Formula Justification:	Conversion Formula recieved as strength ratio (clincalc.com/benzodiazepine/) and does not account for patient variables

Calculated Dosage:

Medication Name:	Triazolam

Figure 10: Conversion Result Page Screen

After submitting data, the Result Screen (Figure 10) displays the converted dosage prominently in a clear, bold font. This screen includes a summary of the input parameters and the calculated conversion for easy review, supporting the Visibility of System Status heuristic. Below the result, sections labeled “Warnings” and “Administration Instructions” highlight potential risks and provide guidelines for administering the medication safely, aligning with the Help Users Recognize, Diagnose, and Recover from Errors heuristic.

Additionally, the application further supports this heuristic by offering a comprehensive summary of the calculation, enabling users to review and confirm the accuracy of the input data before proceeding, thus minimizing potential errors in the conversion process.

Appendix

OpenFDA API

Resource URL:

https://open.fda.gov/apis/

Resource Information:

- Response Format: JSON
- Requires Authentication: No
- Rate Limit: 1,000 calls per day (varies depending on the API endpoint)

Parameters:

- Drug Name / NDC (National Drug Code): The name or NDC number of the drug to retrieve information for.
- Selected Data Fields: Specific data fields you wish to retrieve, such as drug labeling, adverse events, or product recalls.

Example Request:

https://api.fda.gov/drug/label.json?search=morphine

Example Response:

```
{
  "meta": {
    "disclaimer": "FDA provides this information for educational purposes only."
  },
  "results": [
    {
      "openfda": {
        "ndc": ["12345-6789"],
        "brand_name": "BrandName",
        "generic_name": "GenericName",
        "manufacturer_name": "ManufacturerName"
      },
      "drug_interactions": [
        "DrugInteraction1",
        "DrugInteraction2"
      ],
      "pregnancy_category": "Category C",
      "dosage_and_administration": "Take one tablet daily."
    }
  ]
}
```

Response Fields

Parameter	Description
meta.disclaimer	A disclaimer stating that the information provided is for educational purposes only.
openfda.ndc	National Drug Code (NDC) number for the drug.
openfda.brand_name	The brand name of the drug.
openfda.generic_name	The generic name of the drug.
openfda.manufacturer_name	The name of the drug’s manufacturer.
drug_interactions	List of known drug interactions for the specified drug.
pregnancy_category	The drug’s pregnancy category (e.g., "Category C", "Category D").
dosage_and_administration	Instructions for the drug’s usage and administration (e.g., dosage).

Wikipedia API

Resource URL:

https://en.wikipedia.org/api/rest_v1/

Resource Information:

- **Response Format**: JSON
- **Requires Authentication**: No
- **Rate Limit**: Limited by Wikipedia’s usage policy

Parameters:

- **Medication Name**: The name of the medication for which a summary is requested.
- **Description Length**: Maximum length of the description to be retrieved.

Example Request:

https://en.wikipedia.org/api/rest_v1/page/summary/ Acetaminophen

Example Response:

```
{
 "type": "standard",
  "title": "Acetaminophen",
  "displaytitle": "Acetaminophen",
  "extract": "Acetaminophen, also known as paracetamol, is a medication used to treat
pain and fever..."
}
```

Response Fields

Parameter	Description
type	The type of page returned, usually "standard" for basic summary pages.

title	The title of the Wikipedia page for the medication.
displaytitle	The display title for the page, often similar to or the same as the title.
extract	The summary text for the medication, limited to a certain number of characters (e.g., 630 characters).

CancerCalc

Resource URL:

https://www.cancercalc.com/

Resource Information:

- **Response Format**: Table
- **Requires Authentication**: No
- **Rate Limit**: Not applicable (online website)

Parameters:

- **Original Drug Name and Administration**: The name of the original drug and administration method.
- **Desired Drug Name and Administration**: The name of the desired drug and administration method.
- **Dosage**: dosage administered of original format.

Table Example:

Drug	Route	Conversion factor to oral morphine	Equi-analgesic dose to 100mg oral morphine
Morphine	Oral	1	100mg
Morphine	SC	2	50mg
Morphine	IV	3	33.3mg

Response Fields

Parameter	Description
Drug	name of the drug for conversion and dosage.
Route	Specifies the administration route of the drug (e.g., oral, IV, etc.).
Conversion factor to oral morphine	The factor used to convert the drug's dose to an equivalent oral morphine dose.
Equi-analgesic dose to 100mg oral morphine	The dosage of the drug with same analgesic effect as 100mg of oral morphine.

MDCalc

Resource URL:

https://www.medcalc.org/

Resource Information:

- **Response Format**: Table
- **Requires Authentication**: No
- **Rate Limit**: Not applicable (online website)

Parameters:

- **Original Drug Name and Administration**: The name of the original drug and administration method.
- **Desired Drug Name and Administration**: The name of the desired drug and administration method.
- **Dosage**: dosage administered of original format.

Table Example:

Compound	Route	Equivalent Dose (mg)	Duration of Action
Betamethasone	IV	0.75	Long (36-72 Hours)

Cortisone	PO	25	Short (8-12 Hours)
Dexamethasone (Decadron)	IV or PO	0.75	Long (36-72 Hours)
Hydrocortisone	IV or PO	20	Short (8-12 Hours)

Response Fields

Parameter	Description
Compound	The name of the chemical compound or drug being analyzed.
Route	The administration method of the compound (IV / PO).
Equivalent Dose (mg)	The dose of the compound that is considered equivalent to a reference dose in milligrams.
Duration of Action	The period of time the compound remains effective in the body after administration.