Department of Computer Science & Engineering
The University of Texas at Arlington

# Detailed Design Specification
# CSE 4317: Senior Design II
# Spring 2020


# Med-X
# Hands-On

Caleb Jolley
Christopher Willingham
Sansidh Kommaraju
Paras Sharma

## Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 0.1 | 2.15.2020 | CJ | document creation |
| 1.0 | 2.22.2020 | CJ, SK, PS, CW | release candidate 1 |
| 1.1 | 2.23.2020 | SK | Completed Database Layer Subsystem |
| 1.2 | 3.2.2020 | PS | Fixed few typos and added a couple subsections |
| 2.0 | 5.15.2020 | PS | Tweaks after Senior Design 2. GM was dropped from the group in Senior Design 2. |

# CONTENTS

## List of Figures

# 1 Introduction

Hands-On is a system that will allow for in-home physicians and nurses to quickly and accurately create, update, and review patient information. Hands-On will securely store patient information in accordance with HIPAA laws while providing chartering services to in-home physicians. Hands-On will possess an intuitive user interface to reduce required training and usage time, allowing more time to be spent with patients than with the system. To accomplish these goals, the system must have a secure database to store patient information, a database of drug information, an intuitive user interface, and various critical data entry forms. It will be downloadable on any android phone or tablet via the play store.

# 2 System Overview

Hands-On is going to approximately follow the Model-View-Controller (MVC) model of software design. Each part of the system will be a separate layer that can be changed independently of the others with minimal affects upon them. The three layers more specifically will be the User Interface layer, the System layer, and the Database layer. The User Interface layer will control how the user views and interacts with the system. The system will control the flow of the program, updating the UI and sending/receiving information from the database as needed. Finally, the database will handle storage of all relevant information about patients and afflictions.

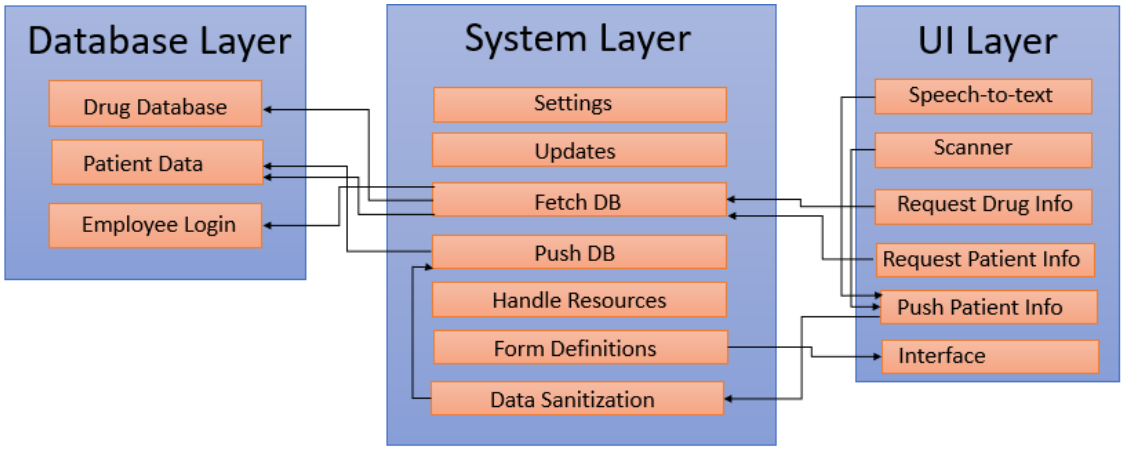


Figure 1: System architecture

# 3 System Layer Subsystems

System layer acts as a bridge between Database layer and User Interface layer. It is composed of various subsystems that are described below:

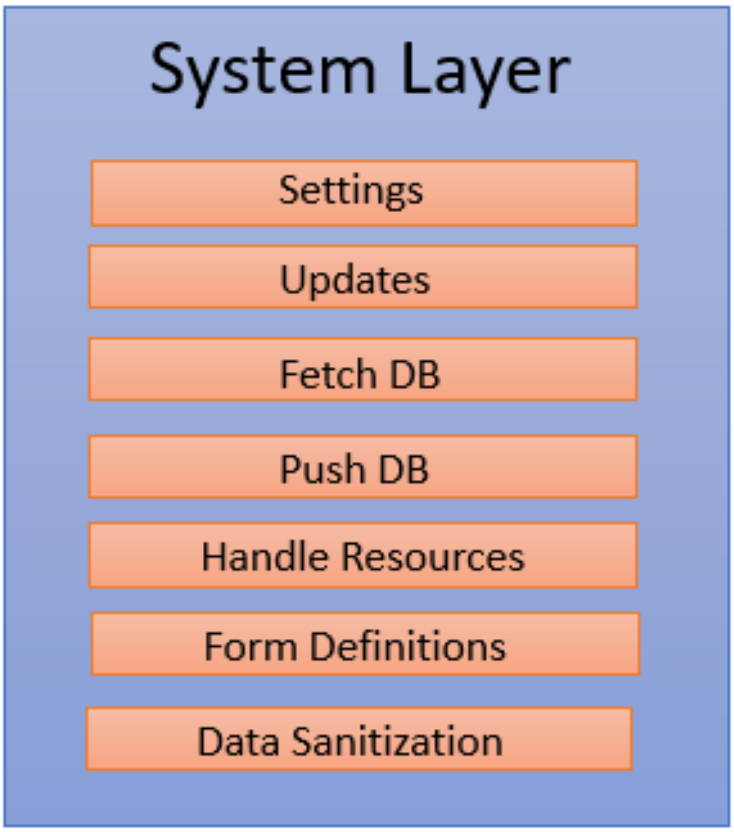


Figure 2: System layer subsystem description diagram

## 3.1 Layer Hardware

The layer is a part of a software that runs on android tablets.

## 3.2 Layer Operating System

The software runs on Android operating system with version 7 (Nougat) or greater.

## 3.3 Layer Software Dependencies

A description of any software dependencies (libraries, frameworks, etc) required by the layer.

## 3.4 Settings Subsystem

Rendering UI based on userâs preference is done via this subsystem. For instance, if the user wants larger text, this subsystem tells UI layer to make fonts bigger. Similarly, authentication and authorization is also done via this later. For example, storage of user session, remembering whoâs logged in, making sure the logged in user is authorized for the requested data, etc. The UI layer will present a form to the user and collect all the required data regarding settings. This subsystem from system layer will save that data into a JSON file and read it next time, i.e. to render components, to set the font size, to give the logged in user's information, etc.

### 3.4.1 Subsystem Software Dependencies

This subsystem will depend on UI layer to collect the data and JSONParser library to parse the JSON file to store the data or read settings from JSON file and give output as data.

### 3.4.2 Subsystem Programming Languages

Java will be used to write the settings subsystem.

### 3.4.3 Subsystem Data Structures

The JSON file will consist of key-value pairs of all the settings or components that user can change.

### 3.4.4 Subsystem Data Processing

The subsystem will parse the JSON file and then provide the contents to the UI layer for rendering. When user changes the settings, the data is passed to this subsystem, it will store the data into JSON and give out the new data to UI later for rendering.

## 3.5 Update Subsystem

This subsystem is responsible to update the application modules and provide feedback and usage statistics to us. There will be a small worker object constantly checking for updates and when update is found, replaces the old java classes with new one. Also when any abnormal usage statistics or any feedback is collected, the data is reported to us for monitoring.

### 3.5.1 Subsystem Software Dependencies

This subsystem will depend on WebClient class to check for and download updates and also post feedback to the server.

### 3.5.2 Subsystem Programming Languages

Java will be used to write the Update subsystem.

### 3.5.3 Subsystem Data Processing

There will be a small worker object constantly checking for updates and when update is found, replaces the old java classes with new one. Also when any abnormal usage statistics or any feedback is collected, such as when device freezes or the software crashes, the data (such as current state variables' values) is reported to us for analyzing and monitoring.

## 3.6 Database Access Subsystem (Push DB, Fetch DB, Data Sanitization)

These subsystems are responsible in fetching and pushing data from the database layer, and sanitizing data in the process.

### 3.6.1 Subsystem Software Dependencies

These subsystem will depend on android.database.sqlite libraries to read and write the databases.

### 3.6.2 Subsystem Programming Languages

Java will be used to write the Database Access subsystem.

### 3.6.3 Subsystem Data Processing

All the valuable and sensitive data that needs to be retrieved later will be saved on database when user confirms (i.e. clicks on "Save"). Of course, data needs to be valid and sanitized before it is stored. When the record is tried to be fetched later, the database will be read and the data is provided back to the user.

## 3.7 Handle Resources Subsystem

These subsystem is responsible for handling and monitoring system resources.

### 3.7.1 Subsystem Software Dependencies

These subsystem will depend on Android.Hardware and ActivityManager.MemoryInfo libraries to make sure hardware resources are working well and there's no memory leakage or over memory consumption.

### 3.7.2 Subsystem Programming Languages

Java will be used to write the subsystem.

### 3.7.3 Subsystem Data Processing

The program will constantly check for data and memory usage. If the memory is about to be full, the program will save the current state (such as variable values, payloads, etc) so that it won't effect much if the program crashes.

## 3.8 Form Definition Subsystem

These subsystem is responsible for making sure the form is displayed in accordance to user preference such as font size, background color, etc.

### 3.8.1 Subsystem Software Dependencies

These subsystem will depend on saved configurations of user preference.

### 3.8.2 Subsystem Programming Languages

Java will be used to write the subsystem.

### 3.8.3 Subsystem Data Processing

When user loads any form, this system will read the configuration files and throw out parameters to use in order to display or render the form.

# 4 Database Layer Subsystems

The database layer in Hands-On contains three databases that cater to all the needs of a hospice charting system. The three databases are the Drug database, Patient Data database, and the Employee Login database. The Drug database is used for identifying prescribed medication, easier access to similar medication, etc. The Patient Data database will contain all hospice records relating to a patient, including recent diagnoses, attending physician, and current prescription among other things. The Employee Login database will contain administrative information about the hospice's employees and their account which, in case of nurses are used to sign on to their device or in case of doctors are used to include them as the attending physician and as an authority for signing off on prescription medication among other things.

## 4.1 Layer Hardware

The database will initially be stored on the demo device while maintaining all methods involving the database layer as is. The database may then be moved to a standard raspberry pi with a 32 GB memory.

## 4.2 Layer Operating System

The raspberry pi requires a raspbian OS of at least debian version 9 (Stretch).

## 4.3 Layer Software Dependencies

The layer will use SQLite to manage the databases. All software packages required to manage these databases are in the android.sqlite libraries.

## 4.4 Drug Database

The drug database is hosted by RxNorm and the database that the Raspi stores will in most cases be a partial database. The data stored will be such that it can be used to access the required drugs in the hosted database.

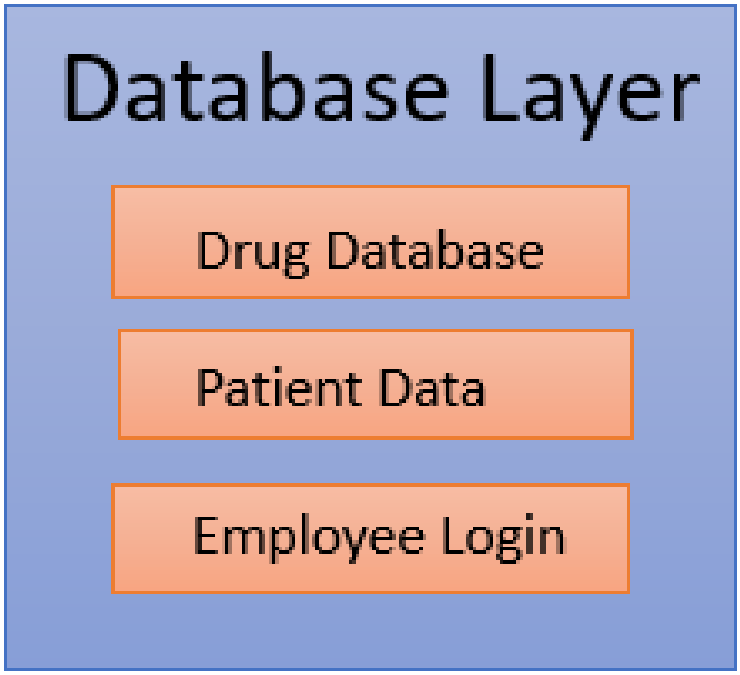


Figure 3: Example subsystem description diagram

### 4.4.1 Subsystem Software Dependencies

The Drug database will use the RxNav API (https://rxnav.nlm.nih.gov/) to access the Drug database hosted by RxNorm.

### 4.4.2 Subsystem Data Structures

The data structures in this database will hold the list of unique drug identifiers (Drug ID or bar code) for every patient.

## 4.5 Patient Data and Employee Login

The Patient Data database will hold all relevant information of the patient.

### 4.5.1 Subsystem Data Structures

The Patient Data database will hold all the data that a patient fills up in forms and all the data that in gathered while the patient is being attended to at the hospice. All data stored will be done so in as space efficient a manner as possible. That means information extracted from check-boxes will be Boolean, inputs on a scale will have a corresponding encoding, and areas of the human body will all be numbered, so that when areas of the body where the patient feels pain are marked, it is stored as a number rather than an image.

## 4.6 Patient Data and Employee Login

The Employee Login database stores all relevant information of Doctors and Nurses.

### 4.6.1 Subsystem Data Structures

The data structures used for Doctors and Nurses vary slightly. The data stored for both employee categories are the hours clocked in, salary, bio-data, etc. These data points will be stored as strings. For Doctors, additional data stored will be patients attended to, drugs prescribed, etc. which will be stored as the unique identifier of either the patients or drugs respectively. For Nurses, additional data stored will be their Login username and hash of password for signing in to Hands-On, which will be stored as a string and a SHA-256 hash.

# 5 User Interface Layer Subsystems

The User Interface Layer handles communicating with the user. It handles all user input, either via the touch screen, the camera, or the microphone. It also handles displaying all output to the user via the screen.

## 5.1 Layer Hardware

The User Interface layer requires a touchscreen to receive user input. It also uses the touch screen to show the user the activity of the application.

## 5.2 Layer Operating System

The UI layer requires Android Version 7 or greater.

## 5.3 Layer Software Dependencies

The User Interface layer requires the android.os.Bundle, android.app.Activity, android.view.View, android.widget.Button, and android.widget.TextView packages within Android studio. This handles the display and most user input (non-camera, non-microphone).

## 5.4 Subsystem Programming Languages

All User Interface subsystems use Java. If another language is used, it will be denoted in that specific entry.

### 5.5 SPEECH-TO-TEXT

This module listen's to the user's speech and converts it into text, enabling a nurse to speak to the computer rather than type notes.

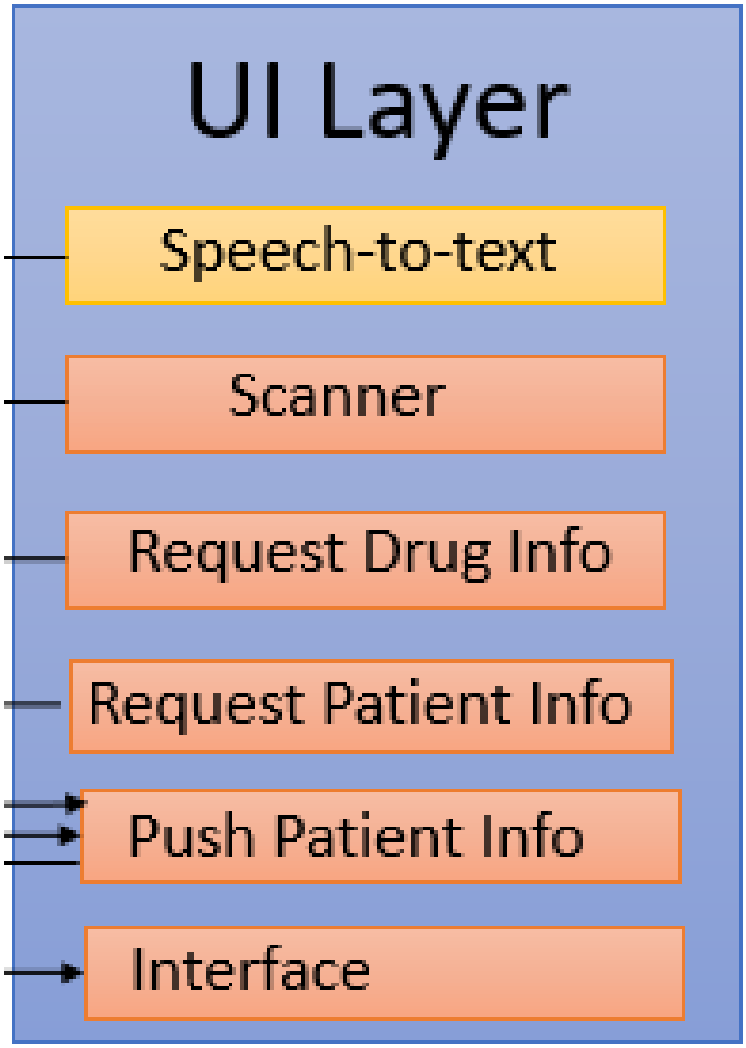


Figure 4: Speech-to-text subsystem

### 5.5.1 Subsystem Hardware

This system requires the microphone built into the tablet or phone.

### 5.5.2 Subsystem Software Dependencies

The Speech-to-text subsystem requires the android.speech package to convert the speech into text.

### 5.5.3 Subsystem Data Processing

The information will be saved into a string and passed to the System layer via the interface subsystem to update the input field with the text.

## 5.6 Scanner

This system is designed to scan a bar code and update the patient information with the associated medicine. The nurse or physician will then input the amount given if the medicine is not a prepackaged dose.

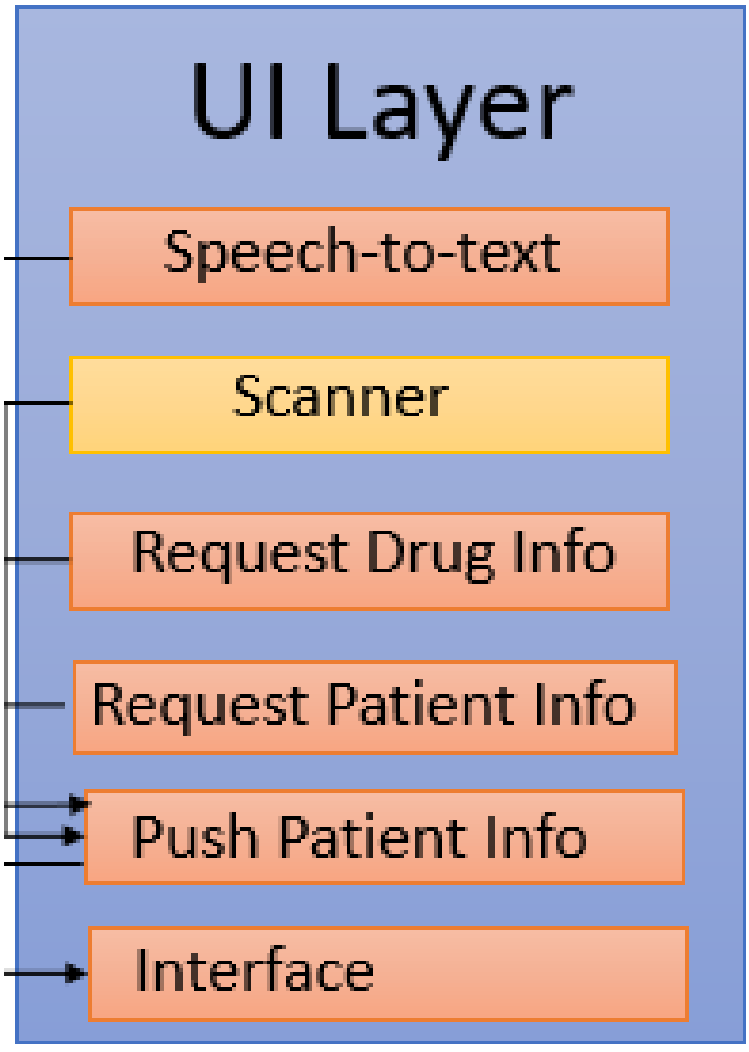


Figure 5: Scanner subsystem

### 5.6.1 Subsystem Hardware

This system requires the camera built into the tablet or phone.

### 5.6.2 Subsystem Software Dependencies

The Scanner subsystem requires the dm7.barcodescanner:zxing:1.9.13 package. This is a barcode scanner, based on zxing, using the apache version 2.0 license.

### 5.6.3 Subsystem Data Processing

The Scanner subsystem takes the barcode that is scanned in and passes it to the Request Drug Information subsystem.

## 5.7 Request Drug Information

This subsystem grabs information about a drug from the database. This information can be obtained by the scanner or entered in by the user. If entered by the user and searching by name, results will auto-populate as they type to attempt to save them from typing the entire name. Otherwise the user can also search via desired symptoms treated or possible side-effects.

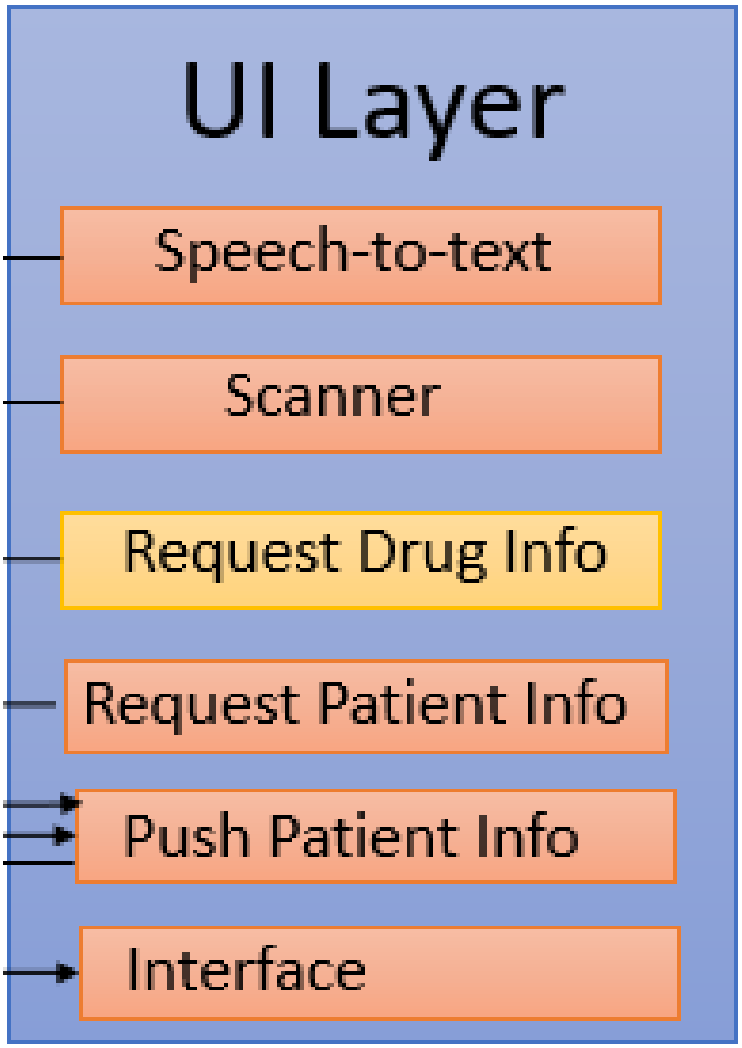


Figure 6: Request Drug Information Subsystem

### 5.7.1 Subsystem Data Processing

This subsystem takes information input by the user, either through scanning the barcode, typing in the name, or searching by symptoms treated or possible side-effects. This information is then passed to the Fetch DB subsystem on the System Layer via the interface subsystem to obtain all information about the drug. The information is then displayed on the screen. If multiple drugs are returned, because a search was done via symptoms or side-effects, all possibilities will be listed and the user will be able to select drugs to examine.

## 5.8 Patient Information

This subsystem allows the nurse or physician to select the patient who's information they wish to view. This subsystem also allows a nurse to add comments and medical details to the patient's file and push those changes to the database.

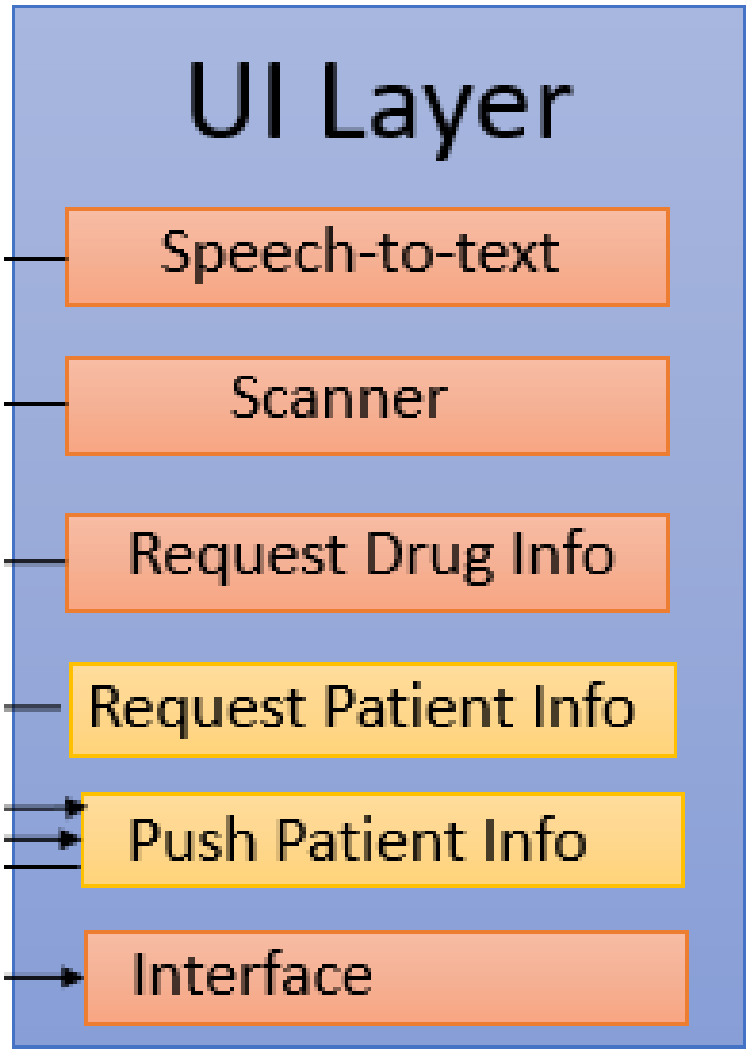


Figure 7: Patient Information subsystem

### 5.8.1 Subsystem Data Processing

When a user selects a patient to view their information the Patient Information subsystem will format a request consisting of the patient's unique ID number and send it through the interface subsystem to the system layer and eventually the database. This information will then be returned by the database and displayed to the user.
When a user edits information for the patient, the updated information will be passed to interface subsystem. The interface will then send it to the system layer which will update the database as appropriate. In the event of a new patient, the user will be able to create a new patient entry and enter information. This will then be pushed to the database.

## 5.9 Interface

This subystem is responsible for communication between the System Layer and User Interface Layer. All requests for information and pushed changes will pass through this subsystem. This subsystem will then validate input to ensure it is valid. It will also check for malicious input.

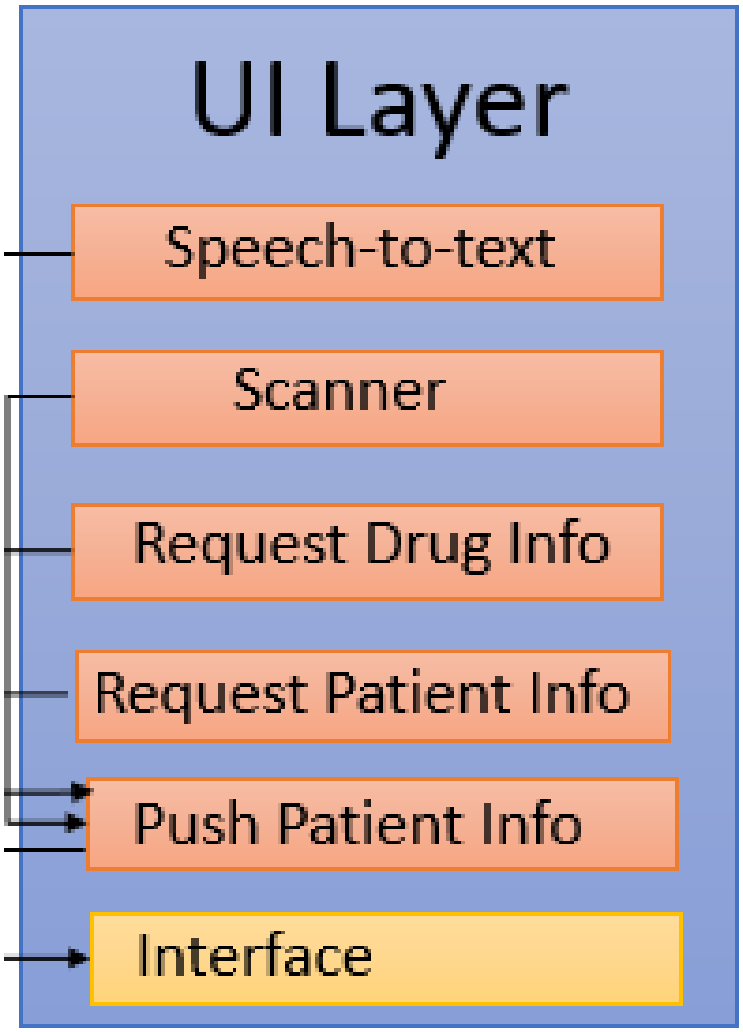


Figure 8: Interface Subsystem

### 5.9.1 Subsystem Data Structures

The Interface subsystem will communicate with the System Layer via function calls. No advanced data structures are required since the Interface subsystem does not communicate directly with the database.

### 5.9.2 Subsystem Data Processing

All database communication (push and fetch actions) will be checked for common malicious inputs and to ensure the input is valid before being passed to the system layer. This will save time and database usage. Fetch requests will be sent with one of the following: The name of the drug, the UPC of the drug, desired drug symptom treatments, desired drug side-affects, or the patient's ID number. Specific details of communicating with the database are handled via the Fetch DB subsystem in the System Layer. Information to be passed to the database will be checked to ensure validity. The user object will then be passed to the Data Sanitization subsystem on the System Layer to eventually be pushed to the database.

# 6 APPENDIX A

N/A