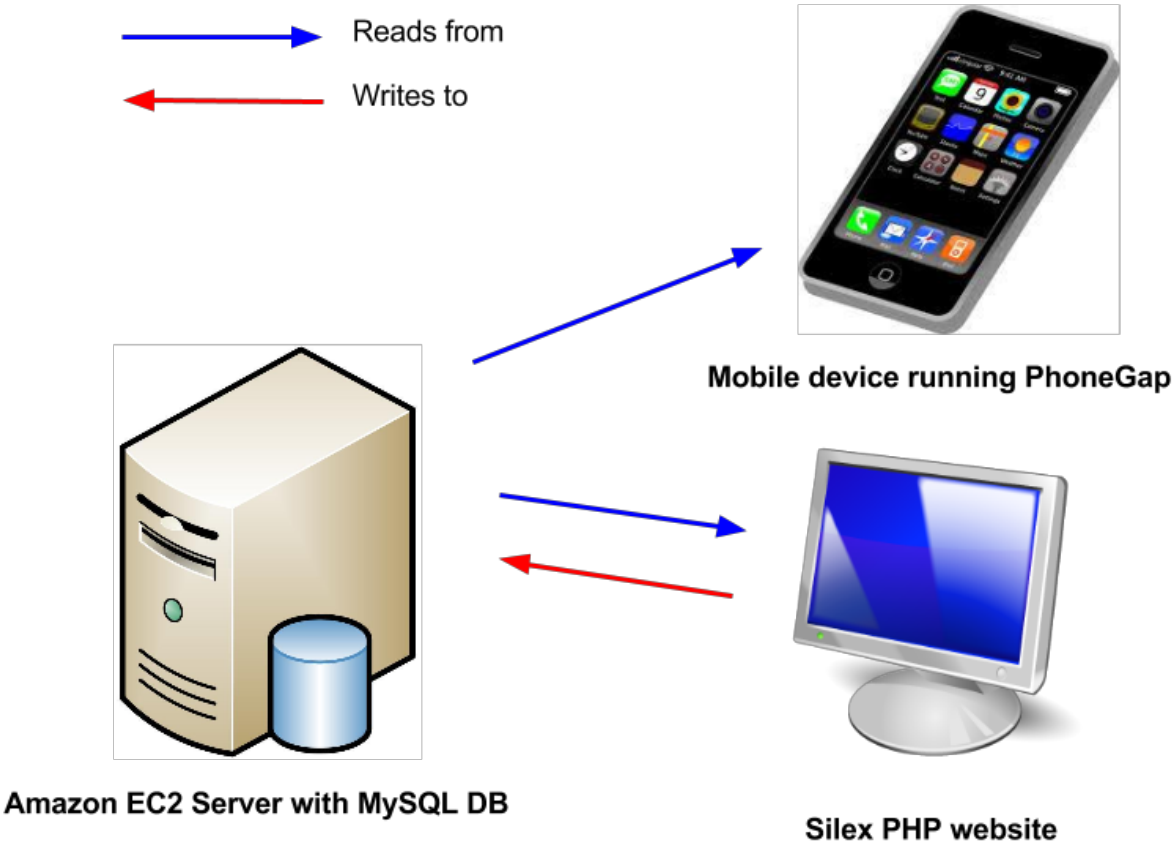
Architecture Document

James Cadek
Zachary Lubinski
James Moore
Kyle Ross
Tanvir Sajed
Jeremy Smereka
Scott Vig

Contents:

Introduction

The AntiC project contains three main components which communicate with each other in order to display the oral chemotherapeutic information to the user. Two of the components are user facing, in the form of a mobile application and a website. The mobile application and the website each communicate with the server database in order to read and display the medical information. The website also has an admin console which allows an authenticated user to edit the information in the database, thus making it a read and write relationship with the server database, whereas the mobile app is read only.



Data exchange is done primarily via PHP requests through the server API. The server dispatcher has a variety of GET, POST, PUT and DELETE request handlers which are used to interface with the user-facing environments. The mobile application uses jQuery ajax calls to download the medical information in the form of JSON strings from the MySQL database. Dose chart images are also downloaded to the mobile device using PhoneGap's FileTransfer API, which directly transfers the image files from the remote

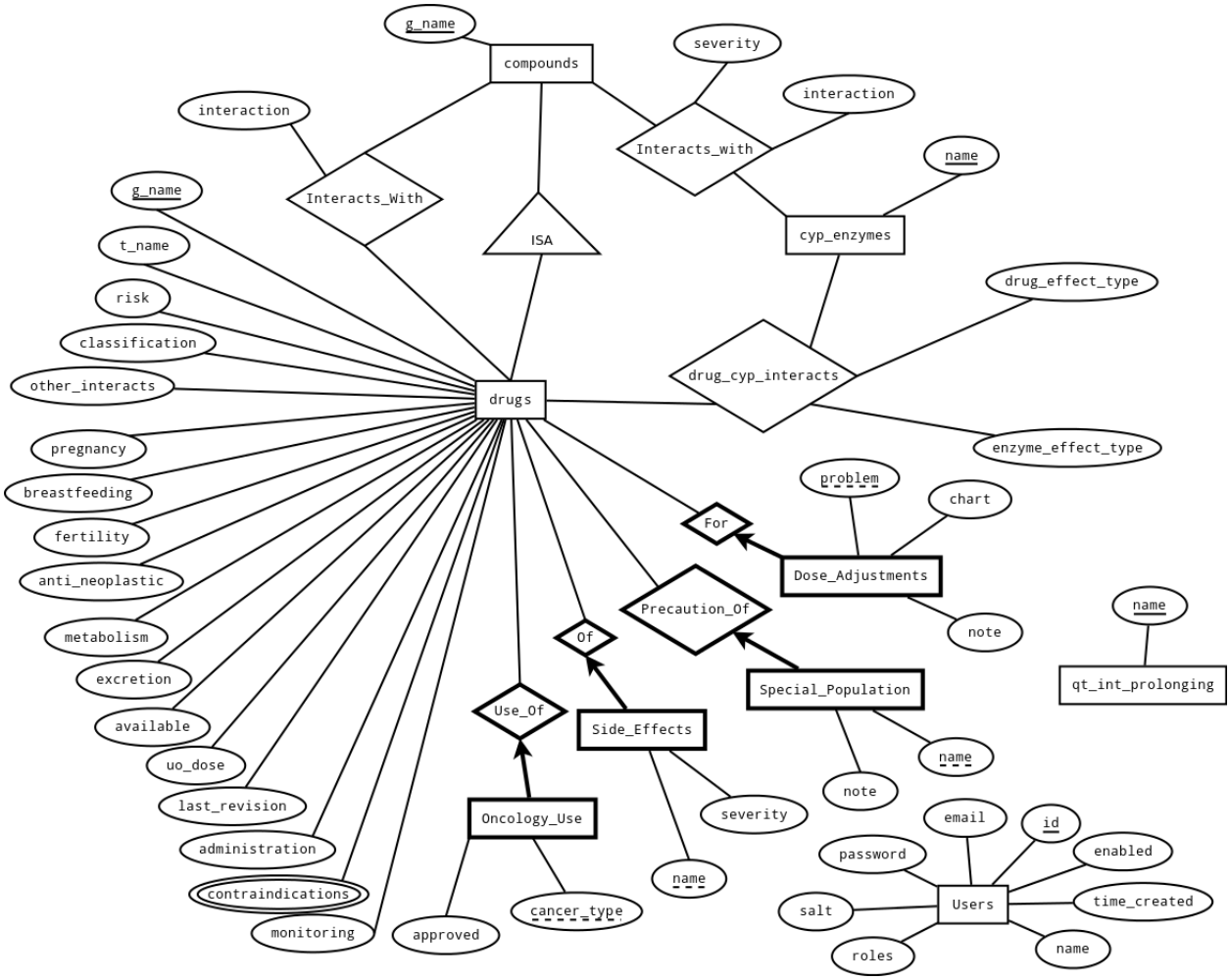
server onto the phone's persistent storage. All of the data which is transferred to the mobile application is stored permanently on the phone so that the user can access the medical information even if an internet connection is not present.

The website interacts with the server by interacting with an API. The API accesses the server database which in turn is used by the PHP code to edit, add, show, and hide data on the system. This is done by directly calling different methods within the API rather than making requests out to the API. The reason for doing this is because the website and API are on the same server, so it is faster to call a function, than wait for a cURL or similar mechanism to respond from the API itself. The different drug properties and enzyme properties presented as columns and tables can all be updated, deleted or added via POST, DELETE or PUT. Drugs are never deleted. Instead users have the ability to hide them or show them from the admin console. The API is also used for getting drugs, enzymes and their properties via GET. Currently, both the website and the app is using the REST GET to get all the required information to display in their respectable interfaces.

System Design

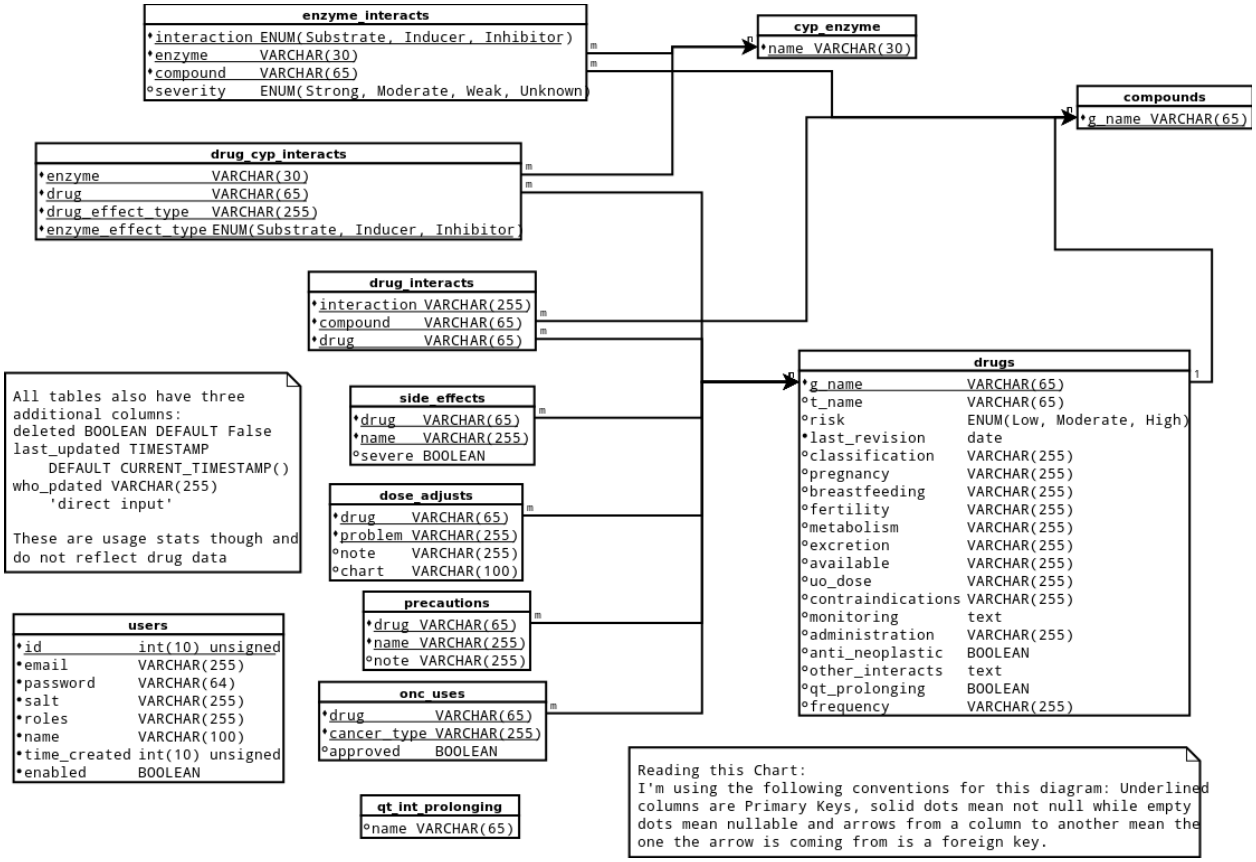
Database Design

The database is formatted primarily to store information about drugs and CYP enzymes, and to represent interactions between these two entities. Several weak entities of drug also received their own tables so as to better represent the data.



A compounds entity was added as a super entity to drug so that drug and enzymes could interact with both drugs and compounds without having to create separate tables. Finally, there are two additional tables qt_int_prolonging, and users. qt_int_prolonging stores a list of qt_int_prolonging agents, which though never referenced explicitly in other tables, is needed as a reference for some drug interactions. users stores user data for use priviledge purposes.

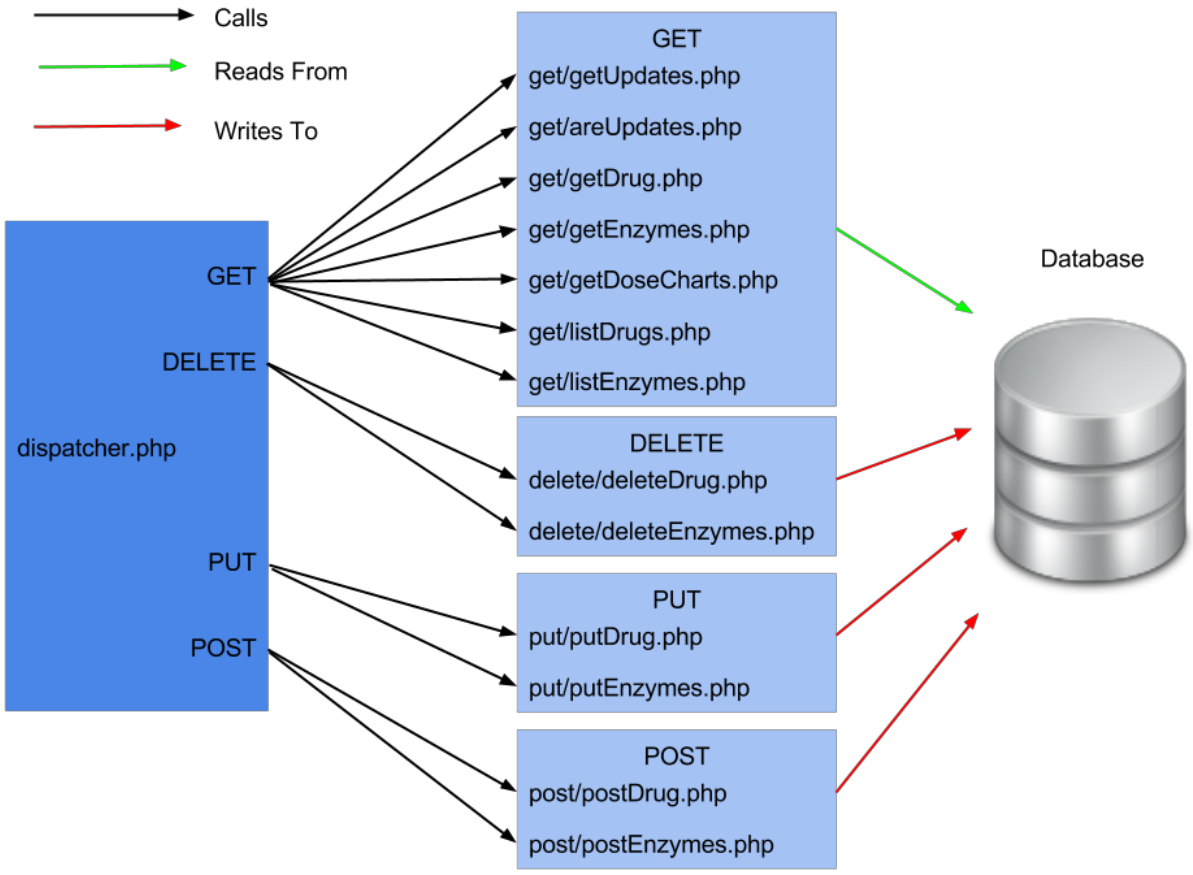
So the final database structure is as follows:



Complete information about a drug can be accessed by combining information from the drugs, drug_cyp_interacts, drug_interacts, side_effects, dose_adjusts, precautions, and onc_uses tables. Complete information about an enzyme can be accessed by combining information from the cyp_enzymes and enzymes_interacts tables.

Server API

All requests made to the server API are routed through the dispatcher.php file which is found in the web/api/ folder. This folder is the base folder for all api files. dispatcher.php interprets the request and then passes it to the appropriate file for handling.



All requests require a json associative array sent with them as data, and individual requests of the same type (GET, PUT, etc) are distinguished based on the key in the json array. Individual files contain all the functions necessary to execute the requests, and all files are php and return json objects.

Also of note is that there are two different database users used in the connection to the database. The user “reader” has only select privileges on all tables, except users which it cannot access, and so “reader” is used with most of the GET functions to add security to the database. The other user, “dbAdmin” has full privileges and is used in the other functions.

Individual functions and their parameters and return values in the api are explained in more detail in separate API documentations, such as on the github wiki.

Website - Overview

The website is a culmination of a variety of open source tools. Namely, the Silex Micro Framework, which is a MVC framework with some basic components to handle routing, rendering, and security. Below is a list of open source components used to create the web management console back-end components:

- Silex PHP Micro-Framework
- Twig
- Symfony-Twig Bridge
- Symfony-Security
- Doctrine DBal
- IRCMaxell Password Compat Library
- Composer
- Silex SimpleUser
- PHPUnit
- PHPDoc (Comments are made using PHPDoc style syntax)

Note: Silex SimpleUser has been heavily modified as many features, such as user registration, are handled via Administrators adding users to the system. Thus, we have removed many functionalities provided by Silex SimpleUser, and it is no longer imported as an external library by Composer.

For the front-end of the website (management console, and live website), we used several libraries as well. These technologies and libraries include:

- AngularJS
- JQuery
- Bootstrap 3
- OpenSans Font
- HTML
- CSS
- JavaScript
- Twig

Website - Application Structure

Application structure for the website/api is shown below. Since Silex has no set standard way as it is scalable to structure websites, it is best to show its architecture and explain its reasoning. Below we have listed the directory structure, and important files within said directories.

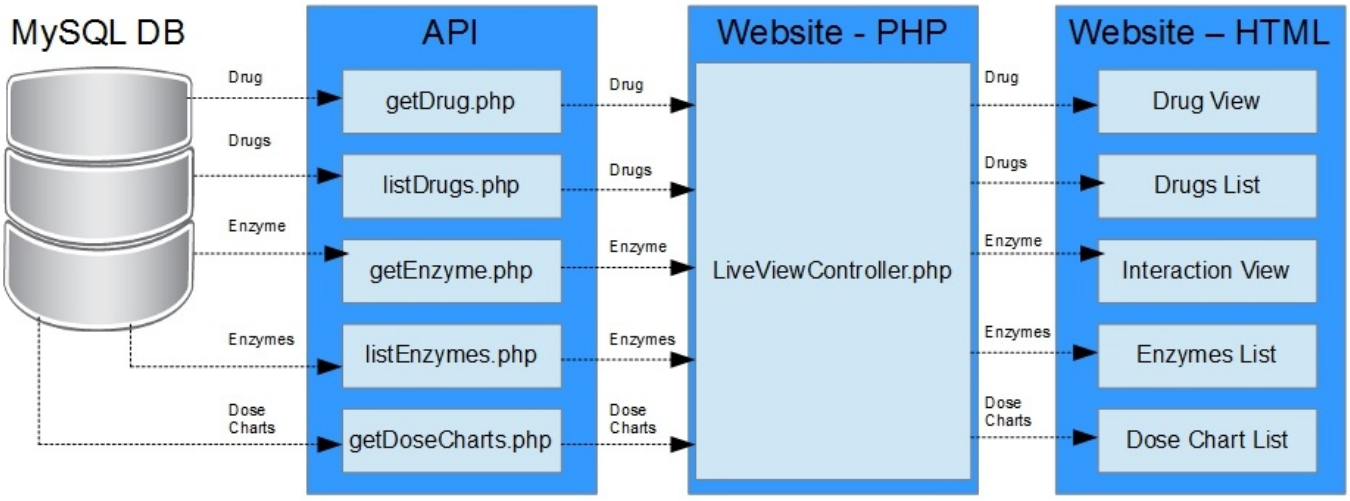
app/ - Contains configuration files for starting Silex
- **app.php** - Registers routes and begins
- **bootstrap.php** - Registers user service providers, external libraries, DB connection

src/ - Contains our own Application logic for Console, User, and LiveView
vendor/ - Contains external libraries imported by Composer
web/ - Contains front facing UI common elements such as CSS, JavaScript etc.
- **api/** - Contains the API which the mobile application communicates with
- **css/**
- **scripts/**
- **font/**
- **img/**
- **.htaccess** - Routes everything through index.php UNLESS its for the API directory
- **index.php** - Requires bootstrap.php from app directory and starts Silex

composer.json - Composer required json file for importing libraries
LICENSE - License for website and SimpleUser
phpunit.xml - PHPUnit XML file for specifying directory of tests
README.md - Contains some basic installation instructions for setting up website

Website - Front-End Overview

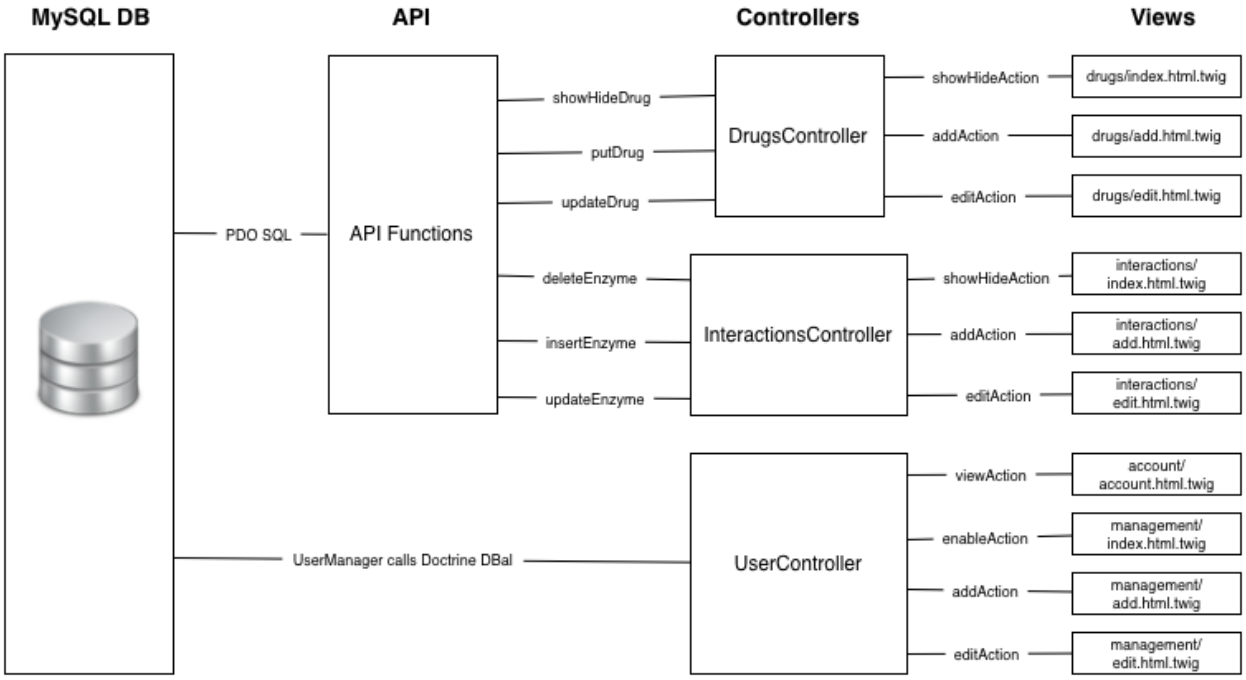
Front-End Website Information Flow Diagram



Below is a chart showing where the the html file for each page is to be found, as well as the website URL path to use to get to it.

Role	**File**	**Path**
Drug View	/livedrugs/view.html.twig	/drug/{ID}
Drugs List	/livedrugs/index.html.twig	/drugs/
Interaction View	/liveinteractions/view.html.twig	/interactions/
Enzymes List	/liveinteractions/index.html.twig	/interactions/{ID}
Dose Chart List	/livedoseadjust/index.html.twig	/doseadjust/
About Page	/liveabout/index.html.twig	/about
Home Page	/index.html.twig	/

Website - Back-End Overview



The web management console is divided into 3 bundles: User, LiveView, and Console. The User bundle is used for user management, and contains Controllers, Models, Providers, Tests, Views, and the Manager. Below I go into the functionality provided by each of the files in these controllers.

Controllers	Contains the functions which are called when a user goes to specified routes defined in the Providers.
Models	A class which is used to pass around data, for ease of persisting users to the database. This is used primarily by the Manager.
Providers	Contains a service provider to allow Silex to obtain the logged in use from the user manager easily. Additionally, defined the routes which ar used when executing controller methods.
Manager	Largely unmodified User Manager which is used to load users from the database into a class. This is executed primarily by calling upon it with the service provider that is registered in Silex.
Views	Contains the front facing HTML twig files which are rendered by the Symfony-Twig bridge library.

Tests	Contains some black box testing primarily focused on the User Manage which is largely unmodified. The tests are done using PHPUnit.

The web management console contains several pages for management which call select functions. These are defined as below:

Role	**View Path**	**URL**
List Drugs	Console/Views/drugs/index.html.twig	/console
Add Drug	Console/Views/drugs/add.html.twig	/console/drugs/add
Edit Drug with ID	Console/Views/drugs/edit.html.twig	/console/drugs/{ID}
Show/Hide Drug with ID	*Not Applicable*	/console/drugs/{ID}/showhide
List Interactions	Console/Views/interactions/index.html.twig	/console/interactions
Add Interaction	Console/Views/interactions/add.html.twig	/console/interactions/add
Edit Interaction with ID	Console/Views/interactions/edit.html.twig	/console/interactions/{ID}
Show/Hide Drug with ID	*Not Applicable*	/console/interactions/{ID}/showhide
User Account	User/views/account/settings.html.twig	/console/account
Show User List	User/views/management/index.html.twig	/console/user
Add User	User/views/management/add.html.twig	/console/user/add
Edit User with ID	User/views/management/edit.html.twig	/console/user/{ID}
Enable/Disable User with ID	*Not Applicable*	/console/user/{ID}/enable
I Forgot My Password	User/views/iforgot/iforgot.html.twig	/console/iforgot
Login	User/views/authenticate.html.twig	/console/login
Logout Route	*Not Applicable*	/console/logout
Check if Authenticated	*Not Applicable*	/console/login_check
Install User DB	*Not Applicable*	/console/install

Note: Routes with Not Applicable are only used by JQuery AJAX calls and a corresponding controller function. The logout and login_check are used only by a controller and redirect elsewhere once completed.

Website - Website to API Interface

The back-end console interfaces with the API that we built. This is done via including the files that are needed and then calling the functions with the appropriate parameters. This is listed below in the following table:

Role	**Controller - Method()**	**API Function**
List Drugs	Drugs - indexAction() LiveView - drugsListAction()	getDrugList()
Add Drug	Drugs - addAction()	putDrug()
Edit Drug	Drugs - editAction()	updateDrug()
View Drug	LiveView - viewDrugAction()	getDrug()
List Interactions	Interactions - indexAction() LiveView - interactionsListAction()	getEnzymeList()
Add Interaction	Interactions - addAction()	insertEnzyme()
Edit Interaction	Interactions - editAction()	updateEnzyme()
View Interaction	LiveView - viewInteractionAction()	getEnzyme()
View Dose Adjusts	LiveView - doseAdjustListAction()	getCharts()

This happens on both the front-end live view of the website, and the management console. The user system does not interface with the API in anyway. This is because the user system is managed primarily by an external library that has been heavily modified called Silex SimpleUser.

Mobile - Overview

The mobile app was developed using the PhoneGap mobile development framework. Development was primarily done in the Android environment, but PhoneGap allows for the porting of the application to iPhone, BlackBerry, and webOS. Specifically PhoneGap version 2.3.0 was used in conjunction with the cordova 2.3.0 library. The front-end views are HTML and are mostly comprised with jQuery mobile elements, and the backend logic of the application was primarily written in Javascript.

Both iOS and Android versions were developed, the sole difference being that the iOS version has one minor change that it hides the iOS7 status bar, as the theme does not play well with the UI.

Data transfer from the remote server was done through two methods. The medical JSON text was downloaded off the server via an ajax call, and the drug dosage chart images were downloaded via PhoneGap's FileTransfer API. In both cases, this data is stored permanently on the phone. The medical data is kept as a string value of the JSON object and is stored via PhoneGap's localStorage object, which provides access to a W3C persistent storage interface.

Upon loading the mobile application, the user is first presented with the medical disclaimer. The user must (for legal reasons) scroll down and select the accept button every time he or she loads the application. Upon accepting the disclaimer, the user is redirected to the application. After redirection, the device checks if an internet access to the remote server is present, and if so, presents the server with the timestamp of the last time that the medical update was downloaded to the phone. If new updates are available since the last download timestamp, the phone will initiate a download of the medical information and dose charts to save on the phone. If an internet connection to the remote server cannot be established, or if there are no new updates to download, the phone will load the relevant medical information off of the phone's permanent storage for offline use.

The medical data is used to populate a variety of Javascript objects (defined below), which are used inside the different tabs available for the user. Currently, there are 4 tabs which present the medical information for the user: Drugs, Doses, Interactions, and About:

- **Drugs:** This is the first tab which is loaded after the user accepts the disclaimer after loading the application. This tab presents the list of oral chemotherapeutics to the user. The list can be sorted alphabetically (by default) or by risk level. Selecting an individual drug will load a new page which contains the relevant information of that drug.
- **Doses:** This tab contains the list of dose adjustments that can be viewed by the user. Each dose adjustment is presented in the form of an image of the chart of different adjustments, because each adjustment has a very distinct form.

- **Interactions:** The interactions tab contains a list of the drug-drug interactions that are contained in the medical information. The interactions will display the substrate, inducer, or inhibitors relevant to the drug interaction.
- **About:** The About tab contains all further reference information for the app. The same disclaimer that appears on the splash screen is the forefront of the about tab, with a symbol legend,a word glossary, additional resources, version information and the development credits available below.

Each of the major HTML pages has an accompanying javascript file which contains the utility functions used to populate listviews, load data into objects, and perform other page transition features, in order to prevent HTML pages from becoming too bloated with back-end logic. Logic for saving the image files on the phone and for interfacing with the remote server is also separated into two helper modules, fileStorage.js and remoteServer.js.

The repository also contains references to an obsolete version of the protocols tab and options page, which was discontinued by request of the client, but remains in the code repository incase development on these features continues.

Documentation for the application code was done through Yuidocs, which uses typical documentation markup to generate HTML documentation of the system components for the user.

External Libraries and Packages:

- Cordova 2.3.0
- PhoneGap 2.3.0
- JQuery Mobile 1.4.0
- JQuery 1.8.3
- JQuery NiceScroll
- Android Support .v4
- Yuidocs
- HTML
- Javascript
- CSS
- Composer (For Behat)
- JSON

Mobile - Application Structure

Application structure for the mobile is shown below. Below we have listed the directory structure, and important files within said directories.

assets/		
	www/	The location of all files for compiling the application
		classes/ Class definitions that match the database tables
		Icons/ Icons used in the application
		jquery/ JQuery file libraries
		js/ Javascript scripts called by the html
		tests/ Some basic HTML test scripts
		/ The root contains all of the .html and .css files
bin/		Android Binaries
cordova/		Files for the cordova library
docs/		App Documentation
features/		Behat test files
gen/		Android buildpath info
libs/		Libraries used
res/		Icons and images used by android system
src/		Android files
test/		Our testing document

Front-End - Mobile Application Design

Models

Javascript Objects	Description
Drug (drug.js)	Creates a Drug object. Holds Drug values for use in mobile application. Attributes of object are: Trade Name (tradeName), Risk (risk), Scientific Name (genName), Classification (classification),Last Revision (lastRev), Monitoring (monitoring), Administration (admin), Pregnacy Info (preg) Breastfeeding Info (breastfeed), fertility info (fertility), metabolism (metabolism), Usual Oral Dose (usualoraldose), Available Info - available, Excretion (excretion), Contraindications (contrain), Special Populations (specialpop) Oncology Use (oncology), Side Effects (sideeffects), Dose Adjustments (doseadjust), Other Interactions (otherinteractions), QT Prolonging (qtprolonging), Anti-Neoplastic (neo), Frequency (frequency)
Cyp Enzyme (cypenzyme.js)	Creates Cyp Enzyme object. Holds Cyp Enzyme values for use in mobile application. Attributes of object: Name (name)
Interaction (interaction.js)	Creates Interaction object. Holds Interaction values for use in mobile application. Attributes of object: Substance Name (subname), Cyp Enzyme Name (cypName), Substance Severity (severity), Substance Type (type)
Oncology Use (oncologyuse.js)	Creates Oncology Use object. Holds Oncology Use values for use in mobile application. Attributes of object:

	Approved (approved), Cancer Type (type)
Side Effect (sideeffect.js)	Creates Side Effect object. Attributes of object: Name (name), Severity (severity)
Special Population (specialpopulation.js)	Creates Side Effect Object. Attributes of object: Name (name), Severity (severity)

Views

View	Description
Disclaimer Page (index.html)	Disclaimer page for mobile application. User must accept this to use application beyond this page.
Drug Menu Page (drugmenu.html)	Drug menu page for mobile application. User can select drugs on this page to view. Can also choose to navigate to other tabs.
Drug Information Page (drug.html)	Drug information page for information. Loads specified drug information and presents that to the user.
Dose Adjustment Page (doses.html)	Dose adjustment list for mobile application. User can select dose adjustments to view.
Dose Adjustment Chart (dosechart.html)	Individual dose chart for dose adjustment. Is loaded when dose adjustment is select in Dose Adjustment Page.
Interactions (interaction.html)	Displays list of interactions for cyp enzymes. Users can select the interaction for the cyp enzyme they are looking for.
Cyp Enzyme Page (cyppage.html)	Displays interactions for cyp enzyme as substrate, inhibitor, or inducer.
About Page (about.html)	About page for mobile application has information about the disclaimer, the legend, glossary, the team, and the resources.

Controllers

Controller	Description
File Storage (filestorage.js)	This module handles the interaction between the application and the offline storage system, which utilizes the FileSystem API plugin provided by PhoneGap. Currently, the fileStorage consists of the handling of a single file, located on the sdcard at anticData/medicalData.txt.
Drug Menu JavaScript (drugmenu.js)	Contains helper functions used by drugmenu.html to populate the list of drugs
Cyp Page Javascript (cyppage.js)	Contains helper functions for cyppage.html to display the drug interaction information to the user
Dose Adjustment Javascript (dose.js)	Contains helper functions used to display the relevant drug information to the user
Drug Information Page Javascript (drug.js)	Contains helper functions used to display the relevant drug information to the user
Init.js	init.js handles the initialization of the application's event handlers This follows the proper initialization process defined at PhoneGap's official website documentation
Interaction Page Javascript (interaction.js)	Contains helper functions for the interactions page to display the list of drug-drug interactions
Remote Server (remoteServer.js)	The remoteServer module provides an interface for the phone to access the remote

	amazon server where the medical data is hosted.The module can be used to check the connection to the server, check if there are valid medical updates, and download those updates

Front-End - Mobile Application - UML