Patient- Like- Mine Design Document

# API Design Overview

The Patient-Like-Mine (PLM) System will be designed around a Client/Server architecture where the GUI portion of the system will be housed on client machines while the main analytical processes will be housed on a centralized server. This setup was decided upon because it allows the client machines to have a low resource overhead, which is advantageous on non-specialized systems. Since the server is a single entity, it is much easier to invest in specialized equipment to ensure processes are performed in a reasonable timeframe.

# System Overview Diagram

Figure 1 below describes the overall interaction between the PLM client and server as well as the various databases.

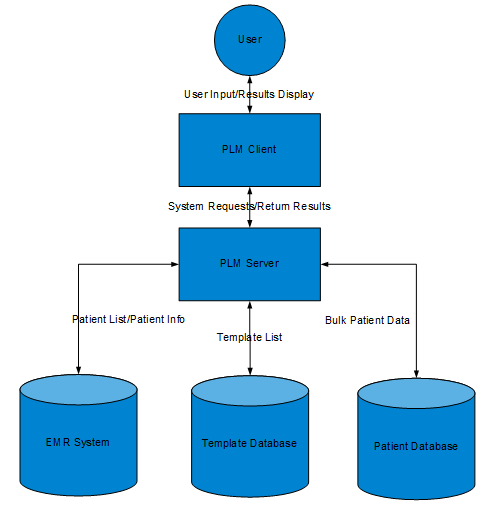


Figure 1: Overview of PLM System Interaction

# Detailed System Description

For the clinician use case, the PLM search, the GUI will initiate three functions sequentially, beginning with selectPatient(). The selectPatient() function first retrieves a list of high-level patient information (name, patient ID) about all current patients in the hospital from the EMR system using the helper function emrQueryPatientList(). The clinician is then prompted to select their patient of interest from the list using the GUI as an interface. Once the clinician selects a particular patient, the name and patient ID information is used within another helper function, emrQueryPatient(), to retrieve the full patient medical record from the EMR system. The selectPatient() function transforms the patient’s medical record into a Patient object and stores it within the GUI for later use. After the selectPatient() function completes, the plmGUI initiates the selectTemplate() function, which uses the helper function templateQueryList() to populate a list of Template objects. The template database will store templates as objects for easy retrieval. The user then selects a template from the list using the plmGUI as an interface. The GUI stores the selected Template object for later use. Using both the stored Patient object and stored Template object, the GUI finally initializes the plmSearch function. This function transforms the patient and template information into a query for the patient database using the patientDBQuery() helper function. The patientDBQuery() function finds patients that are determined to be similar to the patient of interest (as defined by the template) and returns them as a list of Patient objects. The plmSearch() function takes this list of Patient objects and exports it to the helper function plmAnalytics() for processing based on the template. The results of this helper function are returned to plmSearch() as a list of Results objects. A final helper function, plmRender(), transforms the results into graphical representations if applicable. Both the list of results and their renderings are returned to the plmGUI for display to the user.

For the administrator use case, the user is able to select from three different functions through the plmGUI interface: addTemplate(), editTemplate(), and removeTemplate(). The addTemplate() function simply allows the administrator to define the parameters of the new template, which it then converts into a Template object and stores it within the template database. The editTemplate() function allows the administrator to select a template from the database using the helper function templateQueryList() (defined above). Once selected, the administrator sets the new parameters and the template database is updated with the modified object. Finally, the deleteTemplate() function allows the administrator to select a template from the database using the helper function templateQueryList() (defined above) which is then removed from the template database.

# Detailed System Diagram

Figure 2 below describes the interaction between the various components within the PLM system.

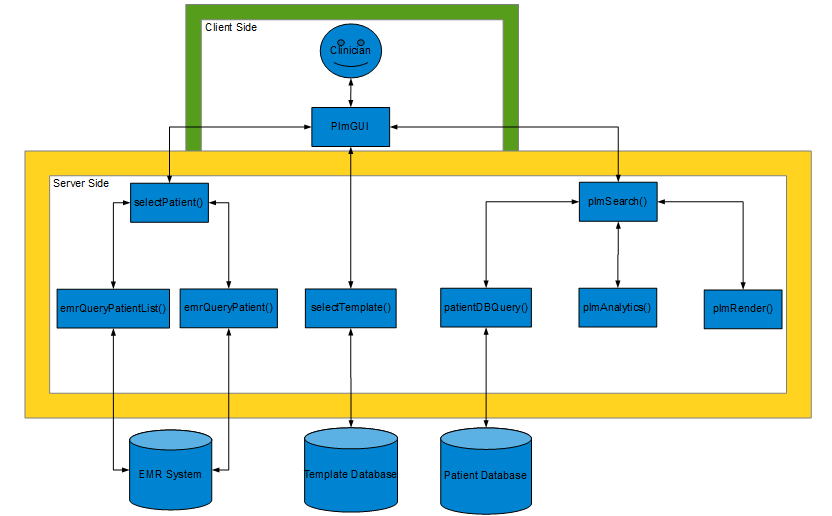


Figure 2: Detailed Overview of PLM Component Interaction

# Full System API

Class PlmGUI{

//provides interface between clinicians and administrators and the functions of the PLM System

//displays the graphical results of the PLM Search

//for the clinician to initiate the PLM search, the GUI calls selectPatient(), then selectTemplate(), and then initiates the plmSearch() function with the requisite information

//for the administrator, the template database is modified by the GUI calling either the addTemplate(), editTemplate(), or deleteTemplate() functions

}

selectPatient(){

//query the EMR system for patient list (emrQueryPatientList())

//clinician selects patient from list (through plmGUI)

//query the EMR system for specific patient information (emrQueryPatient())

//construct a Patient object from the retrieved information

//return Patient information (as a Patient object)

}

Class Patient{

//defines patient parameters

//String name

//int patientID

//List<Labs> labs

//etc.

}

emrQueryPatientList(){

//returns list of active patients from EMR system

}

emrQueryPatient(String name, int patientID){

//uses the name and patient ID to query the EMR system

//returns specific patient information

}

selectTemplate(){

//populate a list with template objects from the Template Database (templateQueryList())

//user selects a template from the list (through plmGUI)

//return the template (as a Template object)

}

templateQueryList(Object constraints){

//returns a list of template objects (constrained by parameters from selectTemplate())

}

plmSearch(Patient p, Template t){

//construct and use a query to the Patient Database using patient and template information (patientDBQuery())

//perform analytics on retrieved patient information (plmAnalytics())

//render the results graphically (plmRender())

//return both the results and rendered graphs to plmGUI to be displayed

}

patientDBQuery(Patient p, Template t){

//returns similar patients to patient p (as a list of Patient objects)

}

plmAnalytics(Patient p, Template t, List<Patient>){

//extracts the relevant information from the list of Patient objects based on information in the Template

//returns the results (as a list of Result objects)

}

Class Results{

//contains results extracted in plmAnalytics()

}

plmRender(List<Results>){

//uses the list of results to create a list of Render objects

//returns the list of Render objects

}

Class Render{

//contains all the information needed to display a graph in the GUI

//also contains information about which patient the render belongs to

}

addTemplate(){

//allows user to input template information

//create a Template object from the information

//add the Template object to the template database

}

editTemplate(){

//user selects the template to edit (selectTemplate())

//user enters updated parameters

//the updated template is saved to the database

}

deleteTemplate(){

//user selects the template to delete (selectTemplate())

//the selected template is removed from the database

}