

iEHR Web Application System Design Document



Version 1.4

October 2011

**Prepared by:
Pacific Telehealth & Technology Hui
459 Patterson Road
4th Floor, E-Wing
Honolulu, HI 96819-1522**

iEHR Web Application	Version 1.4
System Design Document	October 2011

Revision History

Date	Version	Description of Change	Author
6/17/11	1.0	Initial Draft	Danette Amoy
8/3/11	1.1	Changed title of document from "iEHR Presentation Layer System Design Document" to "iEHR Web Application System Design Document." Corrected number of clinical domain widgets and added descriptions of provider and patient widgets for clinical domains. Changed reference from "iEHR" to "iEHR Web Application."	Danette Amoy
9/11/11	1.2	Edits	Danette Amoy
9/14/11	1.3	Changed patient portal widgets: from "Outpatient Notes" to "Progress Notes"; from "Inpatient Notes" to "Discharge Summaries"	Danette Amoy
10/28/11	1.4	Removed "session ID" from session management. Changed patient portal widget: from "Discharge Summaries" to "Discharge/Essentris Notes"	Danette Amoy

iEHR Web Application	Version 1.4
System Design Document	October 2011

Table of Contents

1. Introduction.....	5
2. Project Scope.....	5
3. Presentation Layer Overview	6
3.1 User Interface Components.....	6
3.2 User Interface Process Components.....	6
4. GUI Framework Technologies	7
4.1 Client-side Technologies	7
4.2 Server-side Technologies	9
5. Design Concepts, Definitions, and Implementation.....	9
5.1 Provider Portal Widgets.....	12
5.2 Patient Portal Widgets	12
6. Security.....	13
7. Communication Between the Layers	14

iEHR Web Application	Version 1.4
System Design Document	October 2011

List of Figures

Figure 2-1. High-level View of System Architecture	5
Figure 4-1. Client/Server Technologies in the Stack	8
Figure 5-1. Provider Portal Diagram	11
Figure 5-2. Patient Portal Diagram.....	11
Figure 6-1. Sequence Diagram of Session Management	13
Figure 6-2. Sequence Diagram of DoD Citrix Single Sign-On	14
Figure 6-3. Sequence Diagram of VA Sentillion Bridge Single Sign-On	15
Figure 7-1. Sequence Diagram of the Request/Response Relationship	16

iEHR Web Application	Version 1.4
System Design Document	October 2011

1. Introduction

This document discusses the graphical user interface (GUI) framework of the iEHR Web Application that supports an electronic health record system to be jointly utilized by the Veterans Administration (VA) and the Department of Defense (DoD). It provides the design decisions and usability factors that determined the GUI infrastructure, as well as the technologies, design concepts, interface functions, and communications necessary for data to be presented and viewed on the presentation layer.

2. Project Scope

To provide effective quality care, VA and DoD physicians must be able to access clinical data for a patient regardless of the location of the data. A VA provider seeing a DoD patient must have proper access to the patient's clinical history within the DoD Composite Health Care System (CHCS) to provide contiguous care. Likewise, a DoD provider seeing a VA patient must have proper access to the patient's clinical history within the Veterans Health Information Systems and Technology Architecture (VistA) to provide contiguous care.

The primary goal of this project is to accomplish a complete VA/DoD health care data systems interoperability whereby data for patient records between the VA and the DoD are shared and complete on both hospital information systems, creating a unified view of the patient information. Part of achieving this goal is the design, development, and implementation of a GUI for the iEHR Web Application.

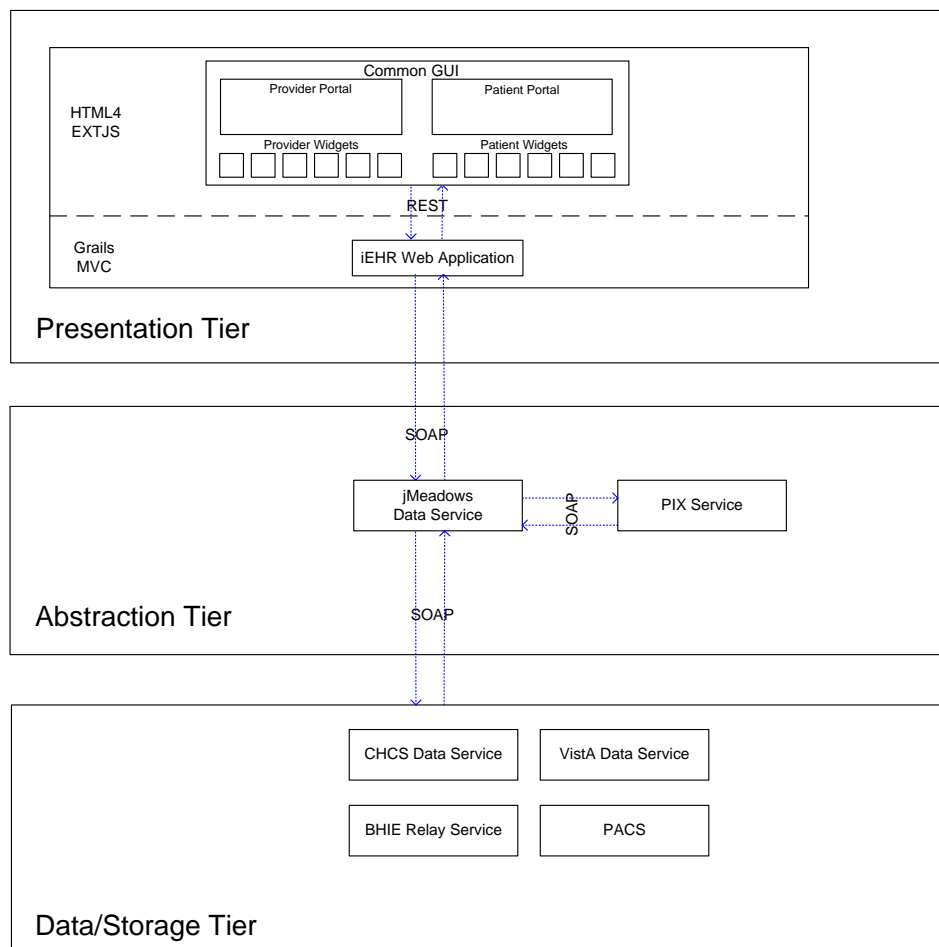


Figure 2-1. High-level View of iEHR Web Application System Architecture

iEHR Web Application	Version 1.4
System Design Document	October 2011

3. Presentation Layer Overview

Most, if not all, applications require some level of user interaction. In today's distributed applications, the code that manages this user interaction is in the presentation layer.

Most simple presentation layers contain user interface components. These components typically contain code to perform functions such as configuring the visual appearance of controls; accepting and validating user input; and acquiring and rendering data from data access components.

The presentation layer can also include user interface process components. User interface process components perform presentation layer tasks that are not directly concerned with user interactions. For example, user interface process components orchestrate the flow of control between forms in the presentation layer and coordinate background tasks such as state management and handling of concurrent user activities.

The presentation layer includes the following types of software components that perform specific tasks:

User interface components—These components make up the user interface of the application. Users see and interact with these components.

User interface process components—These components orchestrate the user interface elements and control user interaction. Users do not see user interface process components; however, these components perform a vital supportive role to user interface components.

The combination of these two types of components forms the presentation layer of the application. The presentation layer interoperates with the data access layers to form the overall solution.

3.1 User Interface Components

User interface components make up the subset of presentation layer components that interact with the user. Think of it as a layer in itself. They are generally referred to as “views” in presentation design patterns. User interface components are responsible for:

- Acquiring data from the user
- Rendering data to the user

The following characteristics determine other responsibilities for user interface components:

- Validation and data input control
- Managing visual layouts, styles, and the general appearance and navigation of the application
- Formatting data and displaying it in useful visual styles
- Browsing, searching, and organizing displayed data

Regardless of the specific technology used to implement them, user interface components are used to present information to the user and to accept user input, thereby enabling interaction with the business process embodied in the application.

3.2 User Interface Process Components

User interface components manage data rendering and acquisition with the user, but these responsibilities do not cover the full spectrum of issues that presentation layers must handle.

A user interacts with an application executing use cases. Each use case requires a set of interactions with the user and the data layers to complete. Applications that have use cases involving multiple user interface

iEHR Web Application	Version 1.4
System Design Document	October 2011

components, or that have to implement multiple user interfaces, have to decide how to maintain data or state across all user interactions and how the user control flows across multiple user interface components.

User interface process components are responsible for managing interactions between multiple user interactions in a use case. User interface process components are referred to as application controllers.

User interface process components are responsible for:

- Managing control flow through the user interface components involved in a use case
- Encapsulating how exceptions affect the user process flow
- Separating the conceptual user interaction flow from the implementation or device where it occurs
- Maintaining internal state that is affected by the user interaction

This means they also manage:

- Accumulating data taken from many user interface components to perform a batch update
- Keeping track of the progress of a user in a certain process
- Exposing functionality that user interface components can invoke to receive data they must render to affect the state for the process

To help separate the tasks performed by user interface process components from the tasks performed by user interface components, the following guidelines apply:

- Identify the process that the user interface process helps to accomplish. Identify how the user sees this as a task.
- Identify the data. The user process needs to be able to submit this data when necessary.
- Identify additional state needed to maintain throughout the user activity to assist rendering and data capture in the user interface.

4. GUI Framework Technologies

To build the presentation layer for the iEHR Web Application, current state-of-the-art web application framework designs and tools were researched and reviewed. The information derived from this research included markup languages, programming interfaces and languages, and standards for document identification and display. It was determined after this review that the following technologies would be used to build the presentation layer of the iEHR Web Application. To further define their roles in the GUI framework, the technologies have been designated as either client-side or server-side (see Figure 4-1).

4.1 Client-side Technologies

HyperText Markup Language (HTML) is the predominant markup language for web pages. It provides a means to create structured documents using semantic tags. Images and other media objects can be embedded and can be used to create interactive forms.

Cascading Style Sheets (CSS) is the language for describing the presentation (i.e. the formatting and layout) of an HTML document. CSS is designed to enable the separation of document control from the details of how it should be presented, including the typography, positioning, colors, and margins. This separation improves content accessibility and provides more flexibility in controlling presentation characteristics.

iEHR Web Application	Version 1.4
System Design Document	October 2011

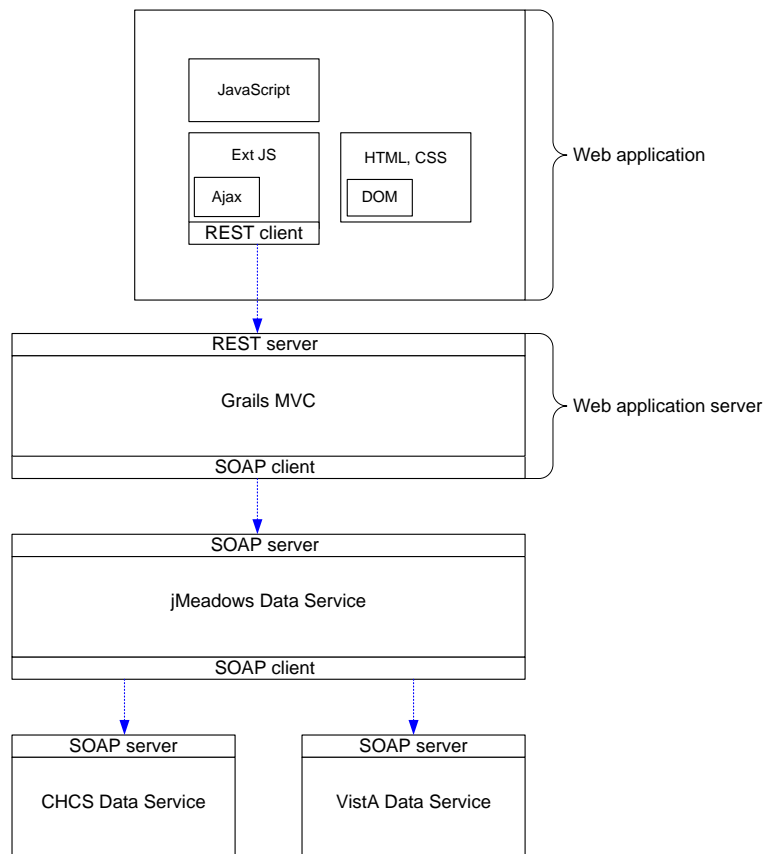


Figure 4-1. Client/Server Technologies in the Stack

JavaScript is an object-oriented scripting language. Although JavaScript has other uses, it is client-side JavaScript—the version that runs inside a user’s browser and manipulates HTML page elements—that is being used. Client-side JavaScript is used to turn static HTML documents into interactive web applications.

Asynchronous JavaScript and XML (Ajax) is the set of techniques used to create web pages with elements that can be independently updated with new content in response to a user’s mouse click or some other event without having to reload the entire page.

eXtensible Markup Language (XML) is a set of rules for marking up documents. It is widely used to transmit arbitrarily structured data in mixed client/server environments. XML and HTML are compatible members of a family of markup languages called Standard Generalized Markup Language (SGML). HTML is an SGML language with a specific Document Object Model (DOM) focused on describing hypertext documents.

Document Object Model (DOM) is a dictionary and grammar for interpreting HTML. A DOM describes HTML elements and their attributes and properties and how they are used to create web pages.

JavaScript Object Notation (JSON) is a language-independent system for representing data objects, although it is based on JavaScript. It is simpler than XML and is often used as an alternative to XML in Ajax applications to transfer data objects between a server and a script running in a user’s browser.

Ext JS is a JavaScript UI library that uses HTML and CSS to build its UI controls and widgets. Ext JS has a complete suite of layout management tools that allow full control over organizing and manipulating the UI as requirements dictate.

iEHR Web Application	Version 1.4
System Design Document	October 2011

4.2 Server-side Technologies

Grails MVC is an open source web application framework that has been designed according to the model-view-controller (MVC) paradigm. MVC is a software architectural pattern that isolates domain logic (i.e. the application logic for the user) from the user interface (i.e. input and presentation).

REST (Representational State Transfer) relies on a stateless, client-server, cacheable communications protocol and, in virtually all cases, the HTTP protocol is used. RESTful applications use HTTP requests to post data (create and/or update), read data (e.g., make queries), and delete data. REST uses HTTP for all four create/read/update/delete operations.

5. Design Concepts, Definitions, and Implementation

The GUI framework is built on a very simple architecture consisting of portals, widgets, tokens, and sessions. The design concepts, the definition and/or purpose of each, and how they are used in the GUI are provided in the table below. Diagrams that show the layout of the provider portal and the patient portal are displayed in Figure 5-1 and Figure 5-2, respectively.

Design Concept	Definition	Implementation
Portal	Is a term, synonymous with “gateway,” for a web site or web that is, or proposes to be, a major starting site for users when they get connected to the web or that users tend to visit as an anchor	<p>The GUI has two portals: a provider portal (see Figure 5-1) and a patient portal (see Figure 5-2). Each portal does the following:</p> <ul style="list-style-type: none"> • Pertains to a particular subject or topic • Includes a library of unique widgets • Provides a column-based widget layout and layout customization • Provides a tabular layout design and the ability to have any number of widget layouts
Token	Is an object that represents something else, such as another object (either physical or virtual), or an abstract concept	<p>The GUI uses two types of tokens: a patient token and a record token.</p> <p>A patient token:</p> <ul style="list-style-type: none"> • Consists of the following: <ul style="list-style-type: none"> - patient ID - patient site code - timestamp • Is tied to an active session that is initiated by the provider when the provider logs in to the system • Is generated in Grails and encrypted. Data encryption is provided by the Advanced Encryption Standard. <p>A record token:</p> <ul style="list-style-type: none"> • Is used to retrieve specific details

iEHR Web Application	Version 1.4
System Design Document	October 2011

Design Concept	Definition	Implementation
Widget (see “Provider Portal Widgets” and “Patient Portal Widgets” below for widget descriptions)	Is an element of a GUI that displays information or provides a specific way for a user to interact with the operating system and application. Widgets include icons, pull-down menus, buttons, selection boxes, progress indicators, on-off checkmarks, scroll bars, windows, window edges (that allow the resizing of a window), toggle buttons, forms, and many other devices for displaying information and for inviting, accepting, and responding to user actions.	<p>The GUI has 9 clinical domain widgets on the provider portal (see Figure 5-1) and 15 clinical domain widgets on the patient portal (see Figure 5-2). Each widget does the following:</p> <ul style="list-style-type: none"> • Is a mini-application running on top of a larger application • Is the launching ground for all windows in the GUI • Is a generic container to which provider data or clinical data can be ported • Contains data coming from one source; in this case, all of the data is coming from the REST layer • Requires a patient token to retrieve data
User profile	Is a collection of personal data associated with a specific user	See Figure 5-1 and Figure 5-2.
Tabbed document interface	Allows multiple documents to be contained within a single window, using tabs as a navigational widget for switching between sets of documents. It is an interface style most commonly associated with web browsers, web applications, text editors, and preference panes.	See Figure 5-1 and Figure 5-2.
Session management	Is the process of keeping track of a user’s activity across sessions of interaction with a computer system. Session management allows the state of applications that are running to be saved and remembered.	This is a server-side component.
Session state	Is a server-side tool for managing state. Each time a web application goes to the server to get the next request, the server has to know much of the last web page needs to be “remembered” when the new information is sent to the web page. The process of knowing the values of controls and variables is known as state management. Session state is server side. In session state, a special session ID is stored on the server. This session ID identifies a specific application. The session ID is assigned to the calling browser.	This is a server-side component.

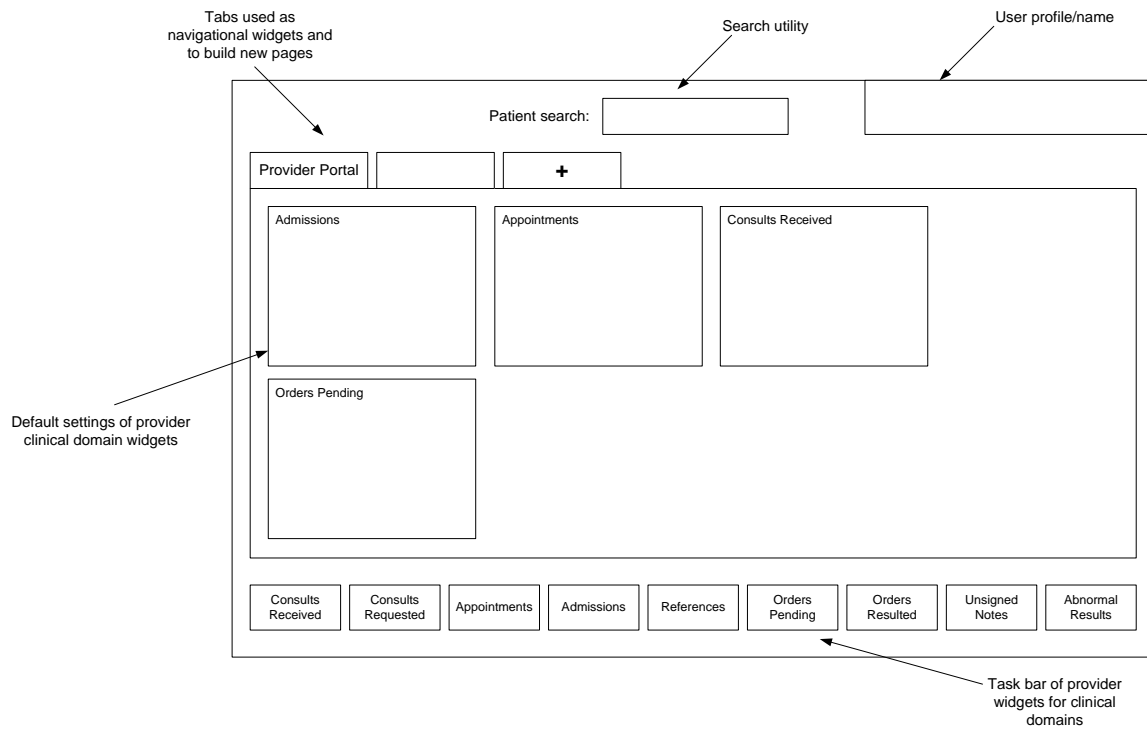


Figure 5-1. Provider Portal Diagram

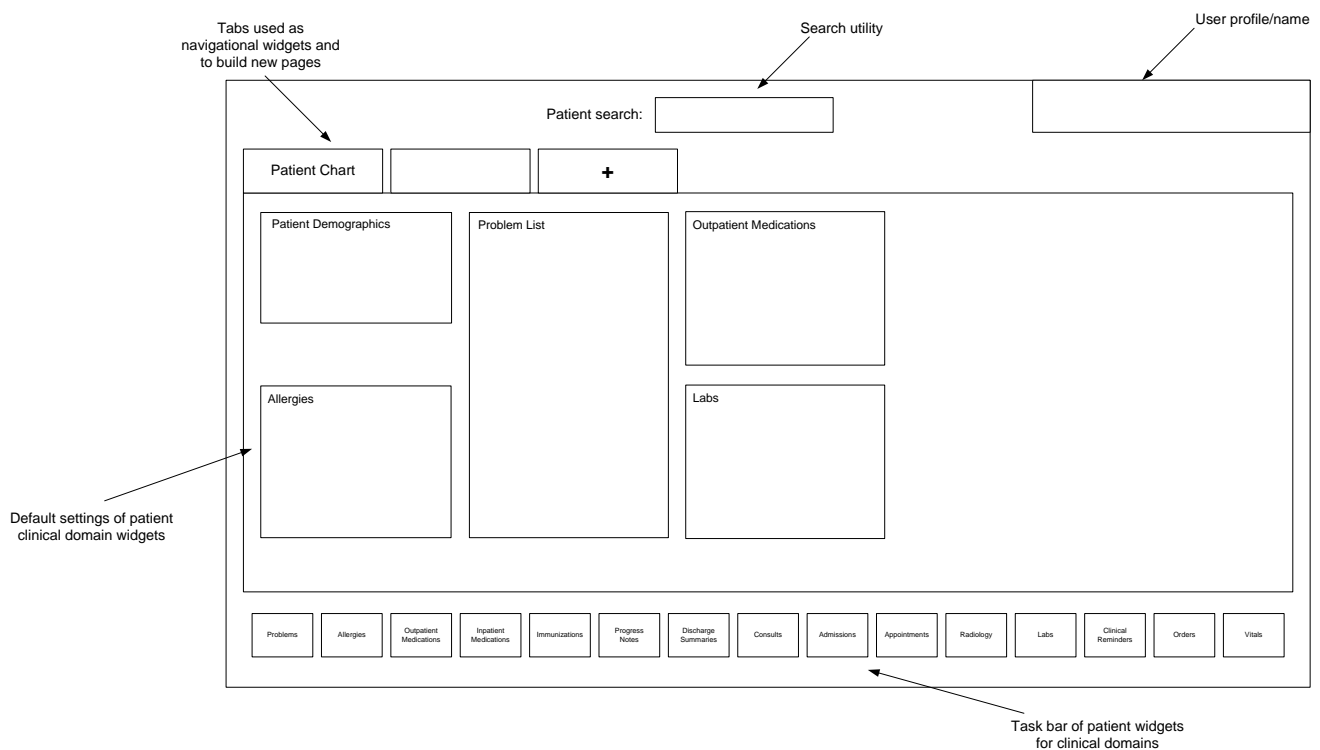


Figure 5-2. Patient Portal Diagram

iEHR Web Application	Version 1.4
System Design Document	October 2011

5.1 Provider Portal Widgets

The following are brief descriptions of the nine clinical domain widgets on the provider portal:

- **Consults Received:** The provider's consults received information is displayed in reverse chronological order by consult date and includes site, consult date, patient name, and consult service.
- **Consults Requested:** The provider's consults requested information is displayed in reverse chronological order by consult date and includes site, consult date, patient name, and consult service.
- **Appointments:** The provider's appointments information is displayed in reverse chronological order by appointment date and includes site, appointment date, clinic, and patient name.
- **Admissions:** The provider's admissions information is displayed in reverse chronological order by admission date and includes site, admission date, ward, and patient name.
- **References:** The provider's references to online clinical medicine resources are provided as hyperlinks.
- **Orders Pending:** The provider's orders pending information is displayed in reverse chronological order by order date and includes site, order date, patient name, and description.
- **Orders Resulted:** The provider's orders resulted information is displayed in reverse chronological order by order date and includes site, order date, patient name, and description.
- **Unsigned Notes:** The provider's unsigned notes information is displayed in reverse chronological order by note date and includes site, note date, patient name, and type.
- **Abnormal Results:** The provider's abnormal results information is displayed in reverse chronological order by date and includes site, date, patient name, and lab test.

5.2 Patient Portal Widgets

The following are brief descriptions of the 15 clinical domain widgets on the patient portal:

- **Problems:** The patient's problem list information is displayed in reverse chronological order by the last modified date and includes site, updated date, description, and status.
- **Allergies:** The patient's allergies information is displayed and includes site and allergy name.
- **Outpatient Medications:** The patient's outpatient active medications information is displayed in reverse chronological order by the last fill order date and includes site, last fill order date, prescription, and status.
- **Inpatient Medications:** The patient's inpatient active medications information is displayed in reverse chronological order by the last fill order date and includes site, last fill order date, and drug name.
- **Immunizations:** The patient's immunization history is displayed in reverse chronological order by the immunization administration date and includes site, administered date, and vaccine name.
- **Progress Notes:** The patient's progress notes information is displayed in reverse chronological order by note date and includes site, note date, type, and provider name.
- **Discharge/Essentris Notes:** The patient's discharge/Essentris notes information is displayed in reverse chronological order by note date and includes site, note date, and type.
- **Consults:** The patient's outpatient consult information is displayed in reverse chronological order by consult date and includes site, consult date, and consult service.
- **Admissions:** The patient's admissions information is displayed in reverse chronological order by admission date and includes site, admission date, discharge date, ward, and registration number.
- **Appointments:** The patient's appointments information is displayed in reverse chronological order by appointment date and includes site, appointment date, and clinic.
- **Radiology:** The patient's radiology exam information is displayed in reverse chronological order by exam date and includes site, exam date, study, and image.
- **Labs:** The patient's lab test information is displayed in reverse chronological order by lab test date and includes site, lab test date, and lab test.

iEHR Web Application	Version 1.4
System Design Document	October 2011

- **Clinical Reminders:** The patient’s clinical reminders information is displayed in reverse chronological order by due date and includes site, due date, clinical reminder, and last recorded date/time taken.
- **Orders:** The patient’s orders information is displayed in reverse chronological order by the order date and includes site, order date, description, status, and type.
- **Vitals:** The patient’s vitals information is displayed in reverse chronological order by the date/time taken and includes site, type of vital sign, histogram, result, date/time taken.

6. Security

Session management security in the iEHR Web Application (see Figure 6-1) is outlined in the following procedure:

1. The provider logs in to the system by using his Access Code and Verify Code.
2. The Access Code and Verify Code authenticate the provider.
3. The provider performs a patient search.
4. The patient search returns a list of GUI patient objects.
5. Each GUI patient object contains a patient token, which is generated in Grails MVC and is encrypted. Symmetric key encryption is used. A unique security/encrypted key is generated for each provider session. The encryption key is stored in the provider’s session object on the server.
6. Each patient token is comprised of a patient ID, patient site code, and timestamp.
7. The patient token is included in every patient-centric request that is available from a clinical domain.

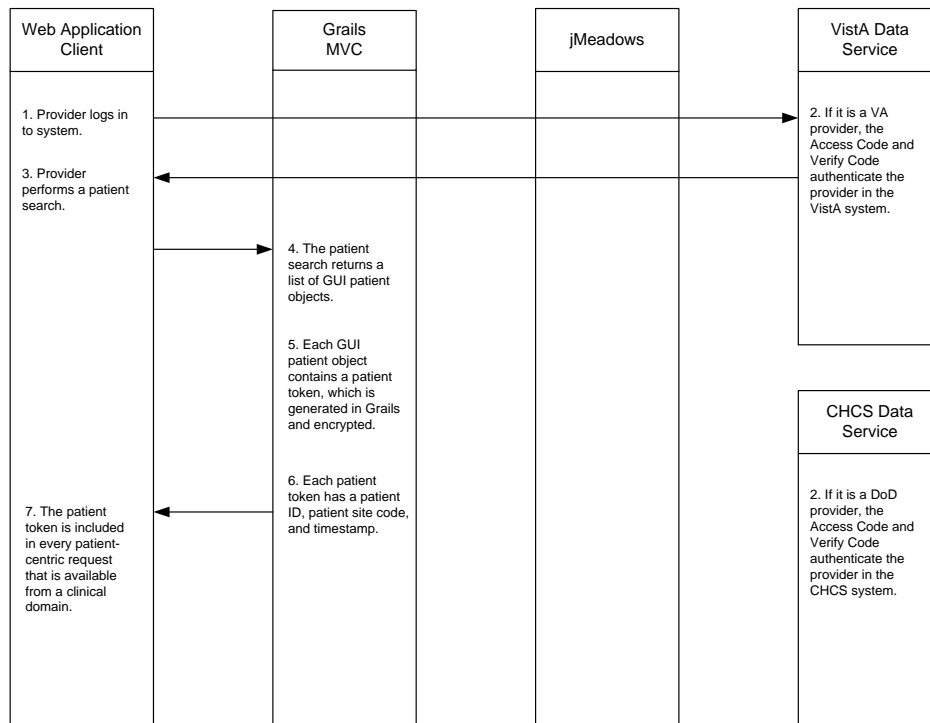


Figure 6-1. Sequence Diagram of Session Management

iEHR Web Application	Version 1.4
System Design Document	October 2011

6.1 Single Sign-On, Context Management, and PKI Authentication

The iEHR Web Application supports single sign-on (SSO), context management, and public key infrastructure (PKI) authentication.

Single sign-on (SSO) is a property of access control of multiple related but independent software applications. With this property, a user logs in once and gains access to all systems without being prompted to log in again at each system. As different applications and resources support different authentication mechanisms, SSO has to internally translate to and store different credentials compared to what is used for initial authentication.

Context management is a dynamic computer process that uses subjects of data in one application to point to data resident in a separate application containing the same subject. Context management allows a user to choose a subject once in one application, and have all other applications containing information on that same subject tune to the data they contain, thereby removing the need to redundantly select the subject in the varying applications.

Public key infrastructure (PKI) authentication is a system that provides for trusted third-party user identity inspection and assurance. This is done by a Certificate Authority and uses cryptography.

The iEHR Web Application supports two SSO procedures at local Vista:

- The DoD Citrix SSO for MHS workstations (see Figure 6-2)
- The VA Sentillion SSO for VA workstations (see Figure 6-3)

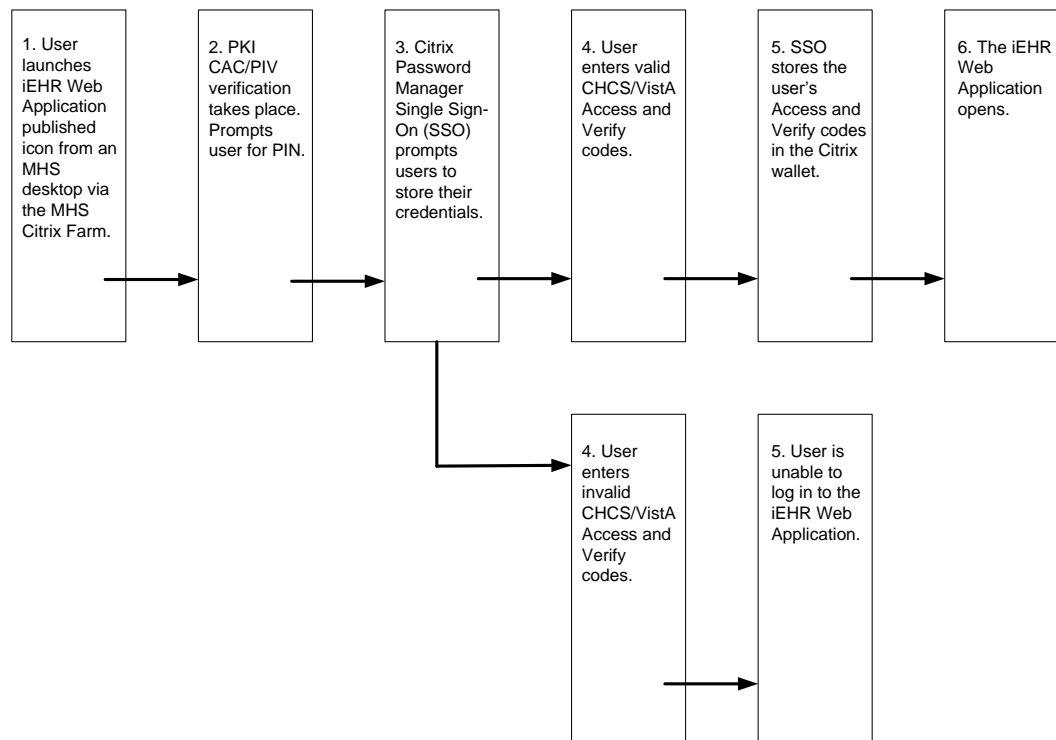


Figure 6-2. Sequence Diagram of DoD Citrix Single Sign-On

iEHR Web Application	Version 1.4
System Design Document	October 2011

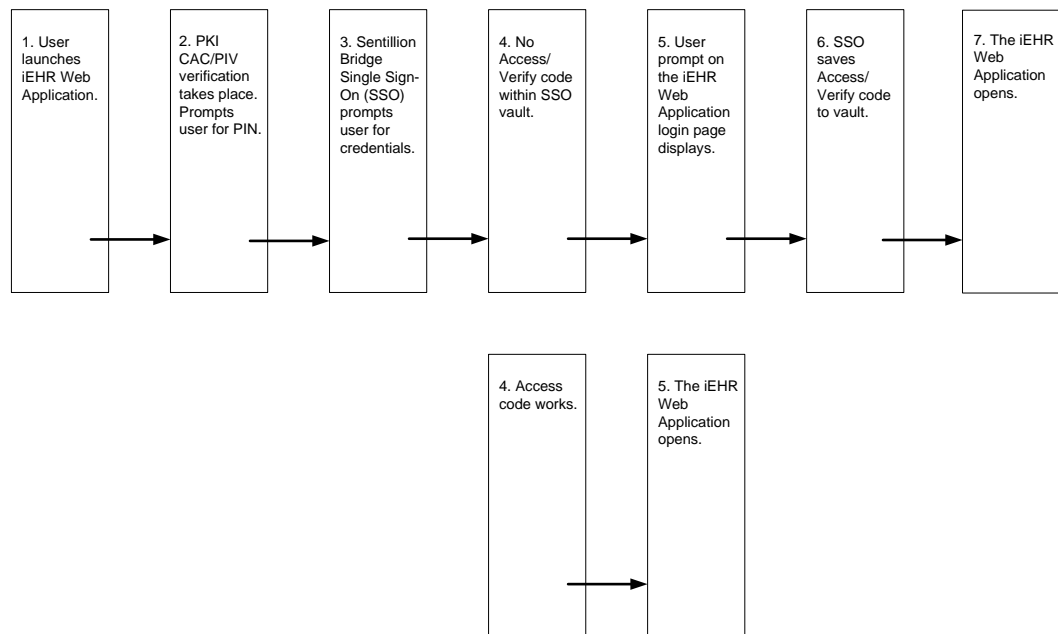


Figure 6-3. Sequence Diagram of VA Sentillion Bridge Single Sign-On

7. Communication Between the Layers

The process of requesting data and the resulting response (see Figure 7-1) is as follows:

1. A widget requests data for a clinical domain from the REST service.
2. The REST service calls a corresponding SOAP service.
3. The jMeadows SOAP service layer makes the corresponding clinical domain request from jMeadows.
4. jMeadows returns a SOAP response that contains a VistA bean.
5. The VistA bean is mapped to a GUI bean.
6. The REST service returns the GUI bean to the widget.
7. The GUI bean is communicated back to the GUI, which then returns the response to the widget.

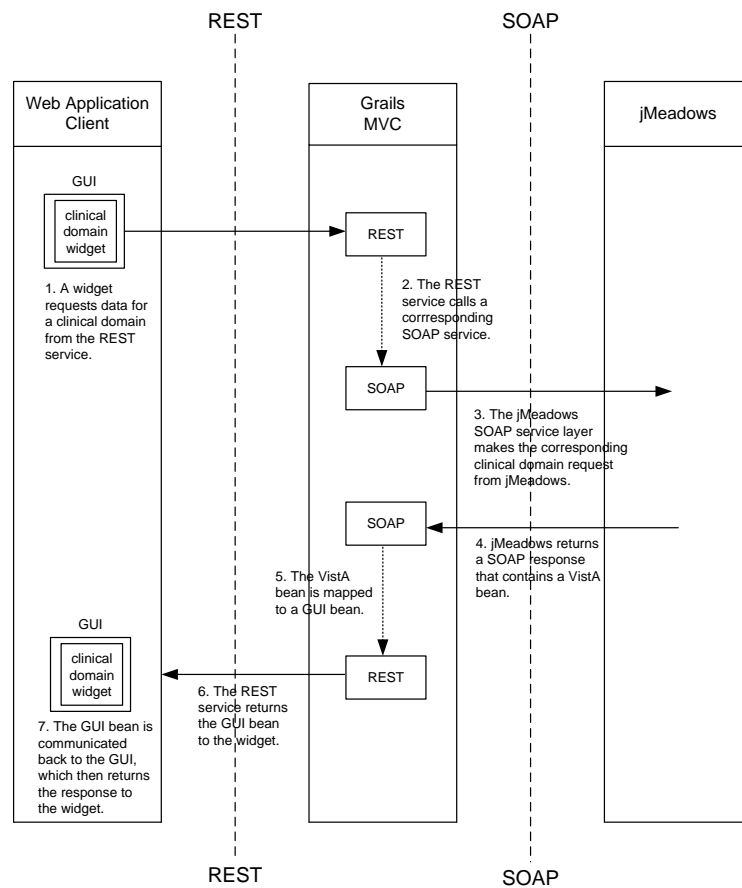


Figure 7-1. Sequence Diagram of the Request/Response Relationship