

CIS 4911 - Team 6

CVM Mediator

Deliverable 4



Team 6:

Ivan Olmos
Luis Bautista
Eduardo Flores
Jandry Guerra

Professor: Peter Clarke

12/07/2010

Copyright and Trademark Notices



Microsoft® HealthVault™

All contents of the Service are Copyright Microsoft Corporation and/or its suppliers, One Microsoft Way, Redmond, Washington 98052-6399 U.S.A. All rights reserved. Copyright and other intellectual property laws and treaties protect any software or content provided as part of the Service. We or our suppliers own the title, copyright, and other intellectual property rights in the software or content. Microsoft, HealthVault, Windows, Windows Live, Windows logo, MSN, MSN logo (butterfly), and/or other Microsoft products and services referenced herein may also be either trademarks or registered trademarks of Microsoft in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, places or events is intended or should be inferred. Any rights not expressly granted herein are reserved. Certain software used in certain Microsoft web sites servers is based in part on the work of the Independent JPEG Group. Copyright © 1991 -1996 Thomas G. Lane. All rights reserved. "gnuplot" software used in certain Microsoft web sites servers is copyright © 1986-1993 Thomas Williams, Colin Kelley. All rights reserved.



Teges Corp™ i-Rounds™

Copyright © 2001-2002 Teges Corp. All rights reserved.

The Content of this Site (including all software, text, displays, images, and audio) are proprietary to Teges Corp. or its Content Providers and are protected by copyright laws of the United States and other countries. The compilation of all materials on this Site is the exclusive property of Teges Corp. and protected by the copyright laws of the United States and other countries. Any reproduction, distribution, public performance, or public display of these materials, in whole or in part, is prohibited without the express prior written permission of Teges Corp. or as expressly permitted in the Terms of Use.

Teges Corp., and the associated logos, and other marks clearly identified in the Teges Corp. Web Site as Teges Corp.'s are Teges Corp. trademarks. You may not use Teges Corp. marks without Teges Corp.'s written permission. All other names, brands and marks are used for identification purposes only and may be trademarks or registered trademarks of their respective owners. This Site, and the Services made available through this Site, are protected under patents pending.



GlassFish

The GNU General Public License (GPL) Version 2, June 1991.

Copyright © 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Indiana University Extreme! Lab Software License
Version 1.1.1

Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved.



Microsoft® SQL Server® JDBC Driver 3.0

© 2007 Microsoft® Corporation. All rights reserved.



All of the deliverable code in SQLite has been dedicated to the [public domain](#) by the authors. All code authors, and representatives of the companies they work for, have signed affidavits dedicating their contributions to the public domain and originals of those signed affidavits are stored in a firesafe at the main offices of [Hwaci](#). Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

The previous paragraph applies to the deliverable code in SQLite - those parts of the SQLite library that you actually bundle and ship with a larger application. Portions of the documentation and some code used as part of the build process might fall under other licenses. The details here are unclear. We do not worry about the licensing of the documentation and build code so much because none of these things are part of the core deliverable SQLite library.

All of the deliverable code in SQLite has been written from scratch. No code has been taken from other projects or from the open internet. Every line of code can be traced back to its original author, and all of those authors have public domain dedications on file. So the SQLite code base is clean and is uncontaminated with licensed code from other projects.

Zentus

Copyright (c) 2007 David Crawshaw david@zentus.com

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF

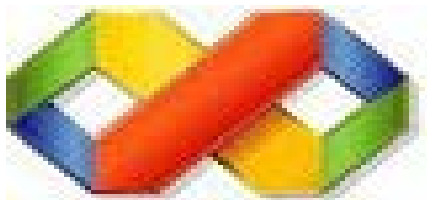
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

GNU LESSER GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License.



Visual Studio®

© 2007 Microsoft Corporation. All rights reserved

Microsoft products and services—including images, text, and software downloads (the "content")—are owned either by Microsoft Corporation or by third parties who have granted Microsoft permission to use the content. Microsoft cannot grant you permission for content that is owned by third parties. You may only copy, modify, distribute, display, license, or sell the content if you are granted explicit permission within the End-User License Agreement (EULA) or [License Terms](#) that accompany the content or are provided in the following guidelines. For more information, consult your copyright attorney.



Eclipse™

Copyright © 2010 The Eclipse Foundation. All Rights Reserved.

Unless otherwise indicated, all Content made available by the Eclipse Foundation is provided to you under the terms and conditions of the Eclipse Public License Version 1.0 ("EPL"). A copy of the EPL is provided with this Content and is also available at <http://www.eclipse.org/legal/epl-v10.html>. For purposes of the EPL, "Program" will mean the Content.

Content includes, but is not limited to, source code, object code, documentation and other files maintained in the Eclipse Foundation source code repository ("Repository") in software modules ("Modules") and made available as downloadable archives ("Downloads").

Executive Summary

The CVM-M mediator is an application that creates a flexible approach to health records retrieval. Our goal is to reduce the cost, and the time it takes to retrieve health records, which is a time consuming task with the current system. Currently, our system retrieves patient's health records from two databases, TegesICU and Health Vault. The information is then compiled and presented to users in different layouts or views selected by the users.

This document will go over the problem, flexibility study, project plan, system requirements, system design, detailed design, and system validation. Through the document the problem will be stated and a solution will be describe and presented. As you read this document there will be a chance to compare the alternative solutions describe in the flexibility section. Many hours were put into designing, integrating and deploying this system, which will be described in detailed. Testing tools were use to find problems that the user may expect as result of interaction with system. At the end of the document credit was given to all of the sources use during the composition of the system.

Table of Contents

1. Introduction	1
1.1. Problem definition	1
1.2. Scope of the System	1
1.3 Over all development methodology.	2
1.4. Definitions, Acronyms and Abbreviations.....	3
1.5. Overview of the document	5
2. Feasibility Study	6
2.1 Description of current system. Identify limitations and constraints	6
2.2 Description of alternative solutions considered.	6
2.3 Recommendation with explanation of why the solution was selected.....	7
3. Project Plan	9
3.1. Project Organization.....	9
3.1.1. Project Personnel Organization	9
3.1.2. Hardware and Software Resources.....	9
3.2. Identification of Tasks, Milestones and Deliverable (work breakdown with cost estimate for milestones)	11
3.3. Cost of the project.....	11
4. System Requirements	13
4.1. Functional and Nonfunctional Requirements	13
Nonfunctional Requirements	13
4.2 Analysis of Requirements.....	15
4.2.1 Use Case Model.....	16
4.2.2. Static Models.....	16
4.2.3. Dynamic Models.....	27

5. System Design (i.e., overall system design)	28
5.1. Overview	28
5.2. Subsystem Decomposition	29
5.3. Hardware and Software Mapping	30
5.4. Persistent Data Management	31
6. Detailed Design	33
6.1. Overview	33
6.2. Static Model	34
6.3. Dynamic Model	34
6.4 Code Specification	36
7. System Validation	39
7.1 Subsystem Test	39
7.2 System Tests	39
8. Glossary	50
9. References	52
10. Appendix	54
10.1 Appendix A - Project schedule (Gantt chart or PERT chart)	54
10.2 Appendix B – All use cases with nonfunctional requirements	55
10.3 Appendix C – User Interface designs	79
10.4 Appendix D – Analysis models (static and dynamic)	88
10.5 Appendix E – Design models (static and dynamic)	96
10.6 Appendix F – Documented Class interfaces (code) and constraints	98
10.7 Appendix G - Documented code for test drivers and stubs	103
10.8 Appendix H – Diary of meeting and tasks for the entire semester	106

1. Introduction

This chapter contains an overview of the CVM Mediator (CVM-M) design, which contains the problem definition, it incorporates the design methodology used and the terminology required to become familiar with the system. This chapter will also give a brief explanation of what to expect in the next chapters of this document.

1.1. Problem definition

In today's society, providers need to rely on patient's to retrieve medical records related to that patient. Using methods of transferring information such as paperwork, storage media, and one time electronic sending have proved to be unreliable and somewhat inaccurate to present patients current health. These methods can encompass multiple records with data repeated multiple times, which can delay time for providers. Having multiple sources of information is good, but without a proper way to organized that data can lead to erroneous results and be time consuming for providers. Providers are force to give the patient the responsibility of keeping these documents up to date, since the current system creates that dependency. The data can be lost, since it can amount to stacks of information, which needs to be kept safe and secure. Precious time is sacrifice, since records are kept disorganized and not chronological. The current system has proved very erroneous, time consuming, and unreliable.

The current problem is that the patient has to be responsible for its medical documents, which include its past and present information. Given the job of retrieving all this information with the current system to the provider has proving to be an expensive and time consuming task.

1.2. Scope of the System

The system will allow all authorized users to query patient multiple medical records without having to worry about time or day. Users are doctors and authorized medical personnel. The system will rely on medical databases to query patients' information. The system will download all compile medical information and store in a temporary folder for each user. The system will display all medical information using XML.

1.3 Over all development methodology.

For the purpose of this project the Unified Software Development Process model (Unified Process)(see Figure 1) was used to develop this system. The systems requirements were used in order to create the sequence diagrams (Olmos) which will be use for better understanding of how processes operate and interact with each other. Base on this information, a minimal class diagram, detailed class diagram and deployment diagram were created.

The minimal and detailed class diagrams were created in order to understand the behavior and interaction between each class. The minimal class diagram describes the association between all classes and the packages they are in, while the detail class diagram describes in details each class, by providing the signatures of important methods and attributes within the classes. The deployment diagram describes what hardware and software components to be use and how they interact with each other. Below there is a representation of all activities of system design.

The following diagram is the unified software development process model that was used for the development of this project. The USDP is iterative and incremental process model and since is Use Case driven it means that we can define the contents of the iteration. Each iteration uses a set of use cases which will be use for implementation.

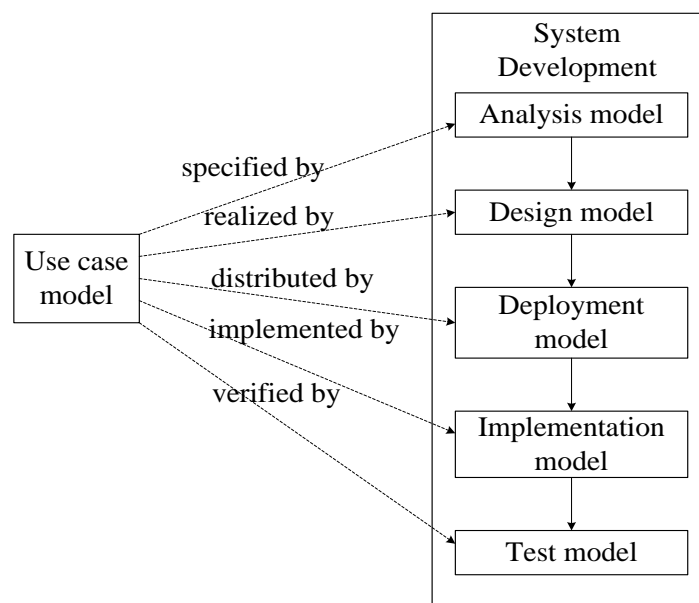


Figure 1: Unified Software Development Process (Clarke)

1.4. Definitions, Acronyms and Abbreviations

The following definition, acronyms and abbreviations will be used throughout the remainder of this document:

- CVM-M (Communication Virtual Machine Mediator): is a system that translates different data sources into usable schema for displaying.
- XML (Extensible Markup Language): is a set of rules for encoding documents electronically. It is defined in the produced by the W3C and several other related specifications; all are fee-free open standards. (XML).
- UML (Unified Modeling Language): is an open method used to specify, visualize, construct and document the artifacts of an object-oriented software-intensive system under development. It is typically used in large development teams as a bridge between process models and software development. Good process modeling tools can output the information required to develop the services required to UML, so that the development team can import the information directly into their software development tools. Some developers insist of hand crafting the UML and ignoring process inputs. However, despite their claims, it cannot replace process modeling to define business processes effectively. (Unified Modeling Language).
- USDP (United Software Development Process): The USDP or Unified Process is a popular incremental software development process framework. The Unified Process is not simply a process, but rather an extensible framework which should be customized for a specific organizations or projects. (Unified Process).
- SDK (Software Development Kit): is typically a set of development tools that allows for the creation of applications for certain software package, software framework, hardware platform, computer system or similar platform. (Software development kit).
- JVM (Java Virtual Machine): enables a set of computer software programs and data structures to use a virtual machine model for the execution of other computer programs and scripts. The model used by a JVM accepts a form of computer intermediate language commonly referred to as Java byte code. This language conceptually represents the instruction set of a stack-oriented, capability architecture. (Java Virtual Machine).
- SQL (Structured Query Language): a worldwide standard used to manage data in relational databases. SQL facilitates the sharing of data especially in large and interconnected databases. (SQL).
- GUI (Graphical User Interface): Allows users to interact with programs in more ways than typing such as computers. A GUI offers graphical icons, and visual indicator, as opposed to text-

bases interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. (Graphical user interface).

- System: The entity that takes care of all the CVM-M User input requests and return outputs base on these requests.
- CVM-M Interface: Control Page where CVM-M user will request the task that the system will get assign to.
- CVM-M User: is the Entity that request information from the system.
- Java: is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. (Java).
- Data Sources: The repository that contains information of a medical record that is use in a system.

1.5. Overview of the document

The rest of this document consists of detailed information on the process of designing the CVM-M. The next chapters will contain and describe all the phases of development that lead to the CVM-M system. It presents the Feasibility Study, which contains different solutions that were presented as alternative solutions. You can also find the project organization, work breakdown and Milestones for each deliverable. The system requirements can be found in chapter 4, which gives an introduction to the proposed system, together with its functional and non-functional requirements. Chapter 5 consists of the system design, which contains the decomposition of the subsystem and hardware and software mapping. The document also gives an introduction to the detailed design and an overview of the behavior and structure of each subsystem. In chapter 7, you can find the system validation, which describes the two approaches used and the test cases used for each approach. And at the end of the chapter there is an evaluation of test. The document also contains a glossary of all the terms used for the documentation, as well as a reference chapter which contains reference to other sources used for the project. At the end of the documentation you can find the appendix, which contains all the diagrams, design models, code documentation, meetings and tasks.

2. Feasibility Study

2.1 Description of current system. Identify limitations and constraints

The system that is currently been use in the healthcare system relies on patients for the most part. Healthcare providers and Doctors access the patient's medical records when the actual patient provides them. Medical records are documented in paper and electronic media, but all records are store in the facility or given to the patient once documented or store in a media. The system puts the patient as the carrier of information, which is acquired from different facilities. Patients either keep the data or retrieve from different encounters. The information is provided in a need to know basis. The patient provides all of the information needed by a provider, so that this information can be useful for a future diagnosis.

For a provider to be able to pull a record when a patient starts a visit or is in an emergency it can take a long time before the patient gets seen or prep for surgery because of the amount of time it takes to retrieve all records from the patient. Most patients will have some of their records with them, but others either have little or none. The patient has the responsibility of keeping tabs on all records generated based on his or her visits to different facilities. Keeping up with all this information can lead to lost of data and lots of bad situations.

1. The system becomes expensive for the patient and the provider, making the cost seem ridiculous over time.
2. Many surgeries and treatments can go bad because of the information lost with this current system.
3. The lost of time cause by this current system, can lead to many avoidable deaths.

2.2 Description of alternative solutions considered.

Alternative 1: Physical Pick Up

Medical records can be picked up physically by the patient. The legal guardian of a patient under the age of 18 can physically pick up the records by showing proof of being a legal guardian of the patient. A person whom is not a legal guardian can also pick up the records with a note from the patient. This note of course has to be notarized in order to be considered valid to release the records. Medical personnel can also retrieve the medical records in person if they so

choose to after patient have clear them to do so. This method is very troublesome since medical personnel need to go to the hospital or medical facility in person and ask for them. This could take time in order to have the complete record retrieved and copy for the medical personnel requesting or patient.

Alternative 2: Email

Medical records can be send via email to any medical personnel or authorized person after patient has clear them to do so and medical personnel have been clear as an authorize person to handle the incoming information. This method takes time because you have first have to verify that the person requesting the record via email is a person authorize to do so. This method could be an easy way to violated security. Medical records could be sent to the wrong email and be read by many unauthorized people. Email also has a constraint on the amount of information you can send through it servers.

Alternative 3: Communication Virtual Mediator (CVM)

This system will provide medical personnel easier ways to get medical records on a click up a button. The CVM will authenticate parties on conference call and will provide medical records from one party to another just by clicking and dragging the information you want to send to the other medical personnel or authorize personnel. You can send partial information such as text information, pictures, or video or if you choose to all the medical record. This system will allow you to communicate with the other client in a one to one interface via video and voice or multi clients on a conference scenario to discuss any information in which they have questions on. This method would save medical personnel a lot of time in receiving medical records from hospital or medical facilities not on their database network. The facts that only authorized personnel can log in into this system cuts down on breach of medical record leaks.

2.3 Recommendation with explanation of why the solution was selected.

When the time comes for patient to retrieve their medical records, several issues have to be attended. Even though “Physical Pick Up” and “Email” solutions are better in an economic point of view, they lack of efficiency and security. Now days the patient has the responsibility to

keep their medical records up to date, and stored in a secure way. This can become a problem for the patient as well for any institution. Information can be lost, and the security of the patient's information can be easily violated.

Our system will provide medical personnel with easier ways to get medical records on a click up a button. Authorized personnel will be able to save a huge amount of time by receiving a patient's medical records from different medical facilities at the same time. Communication Virtual Machine Mediator (CVM-M) is a very efficient solution to one of the biggest problem that faces our American society.

3. Project Plan

This chapter describes the management, planning and organization aspects of the software's development. Section 3.1 covers the assignment of roles of each team members as well as a list of all the hardware and software components required. Section 3.2 breaks down the work schedule and explains all tasks and milestones (refer to Appendix A – Gantt chart and Appendix B – Meeting Dairy) created for the project.

3.1. Project Organization

This section explains the personnel organization and hardware and software components required for the software development.

3.1.1. Project Personnel Organization

The table below contains the major roles assigned to each member throughout all phases of the development process.

Name	Deliverable 1	Deliverable 2	Deliverable 3	Final Deliverable
Eduardo Flores	Minute Taker/ Business Analyst	Time Keeper/ Business Analyst	Team Leader/ Designer	Time Keeper/ Developer
Ivan Olmos	System Analyst	Minute Taker/ System Analyst	Time Keeper/ Designer	Team Leader/ QA Manager
Jandry Guerra	Team Leader	Time Keeper	Architect	Minute taker/ Developer
Luis Bautista	Time Keeper/ Business Analyst	Team Leader	Minute taker/ Integrator	QA Analyst/ Developer

Table 1: Personnel Roles

3.1.2. Hardware and Software Resources

Hardware requirements:






























- Server
 - 2.0 GHz processor speed
 - 2GB RAM or Higher
 - 500 GB available HDD space or higher

- Client
 - 2.0 GHz or equivalent
 - 2GB RAM
 - 120 GB available HDD space

Software requirements:

- Server
 - Windows server 2008 R2
- Client
 - Windows® XP or later version
- Document
 - Microsoft Project Professional
 - Star UML
- Development/Testing
 - Eclipse
 - JUnit
 - EclEmma

3.2. Identification of Tasks, Milestones and Deliverable (work breakdown with cost estimate for milestones)

ID		Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		CVM Project	73 days?	Sat 8/28/10	Tue 12/7/10		
2		Deliverable I	15 days?	Sat 8/28/10	Thu 9/16/10		
3		Project Definition	3 days?	Sat 8/28/10	Tue 8/31/10		Ivan
4		Feasibility Study	4 days	Thu 9/2/10	Tue 9/7/10		Eduardo,Luis
5		High Level Requirements	2 days?	Mon 9/6/10	Tue 9/7/10	4FS-2 days	Luis
6		Project Plan	3 days?	Mon 9/6/10	Wed 9/8/10	4FS-2 days	Jandry
7		Document Assembly	3 days?	Thu 9/9/10	Mon 9/13/10		
8		Presentation I	1 day?	Thu 9/16/10	Thu 9/16/10		Team
9		Deliverable II	16 days?	Tue 9/21/10	Tue 10/12/10	2	
10		Review Software Specification	1 day?	Tue 9/21/10	Tue 9/21/10		Team
11		Identify Use Cases	2 days?	Thu 9/23/10	Fri 9/24/10		Eduardo
12		Reviewing Use Cases	1 day?	Tue 9/28/10	Tue 9/28/10	11	Team
13		Dynamic and Static models	2 days?	Wed 9/29/10	Thu 9/30/10	12	Jandry
14		Implement Prototype	3 days?	Mon 10/4/10	Wed 10/6/10	13	Ivan,Jandry
15		Document Assembly	1 day?	Fri 10/8/10	Fri 10/8/10		Luis
16		Presentation II	1 day?	Tue 10/12/10	Tue 10/12/10		Team
17		Deliverable III	14 days?	Thu 10/14/10	Tue 11/2/10	9	
18		Software Architecture	1 day?	Thu 10/14/10	Thu 10/14/10		Eduardo
19		System Decomposition	2 days?	Fri 10/15/10	Mon 10/18/10		Jandry
20		Hardware and Software mapping	3 days?	Mon 10/18/10	Wed 10/20/10		Jandry
21		Object Design	3 days?	Wed 10/20/10	Fri 10/22/10		Team
22		Refine Class/Sequence Diagrams	1 day?	Fri 10/22/10	Fri 10/22/10	21FS-1 day	Luis
23		Solution Realization	2 days?	Tue 10/26/10	Wed 10/27/10	22	Ivan,Eduardo
24		Document Assembly	3 days?	Thu 10/28/10	Mon 11/1/10		Luis
25		Presentation	1 day?	Tue 11/2/10	Tue 11/2/10		Team
26		Final Deliverable	25 days?	Wed 11/3/10	Tue 12/7/10	17	
27		System Implementation	6 days?	Wed 11/3/10	Wed 11/10/10		Ivan,Luis
28		Integration Testing	3 days?	Thu 11/11/10	Mon 11/15/10	27	Eduardo,Jandry
29		System Testing	5 days?	Tue 11/16/10	Mon 11/22/10	28	Luis,Jandry
30		Evaluation Testing	2 days?	Tue 11/23/10	Wed 11/24/10	29	Luis
31		Test Summary Report	5 days?	Thu 11/25/10	Wed 12/1/10	30	Team
32		Document Assembly	2 days?	Thu 12/2/10	Fri 12/3/10		
33		Final Presentation	1 day?	Tue 12/7/10	Tue 12/7/10		

3.3. Cost of the project

This section will describe the cost to develop the software system using COCOMO II. This tool program is used to calculate the amount of time one person spends working on the software project for one month. We use the standard COCOMO II equation of

$$PM = A * Size^B$$

, where A = 2.94, B = 1.0997, Size = 1749 which is the size of the project and PM = is the estimated person-month. A & B are standard constants in this equation and the only thing that changes is Size, which are the thousands of single line of code in this project. COCOMO II lets you customize the cost drivers' base on a scale of very low to extra high. This scale of course depends on how much your project depends on the following criteria's. Example, DATA (Database size) is not much of concern in this project and therefore a project value of low was assigned to it and so forth. After all the inputs criteria were adjusted in COCOMO II, we got a result of 8 person-month, schedule of 4 months, and a total cost of \$17,293 for this project.

Effort = 8 Person-months

Schedule = 4 Months

Cost = \$17293

Software Size Sizing Method

Unadjusted Function Points Language

Software Scale Drivers

Precedentedness Architecture / Risk Resolution Process Maturity

Development Flexibility Team Cohesion

Software Cost Drivers

Product

Required Software Reliability Data Base Size Product Complexity Developed for Reusability Documentation Match to Lifecycle Needs

Personnel

Analyst Capability Programmer Capability Personnel Continuity Application Experience Platform Experience Language and Toolset Experience

Platform

Time Constraint Storage Constraint Platform Volatility

Project

Use of Software Tools Multisite Development Required Development Schedule

Software Labor Rates

Cost per Person-Month (Dollars)

Results

Software Engineering

Effort = 8 Person-months
Schedule = 4 Months
Cost = \$17293

Total Equivalent Size = 1590 SLOC

Phase Distribution

Phase	Effort (Person-months)	Schedule (Months)	Average Staff	Cost (Dollars)
Inception	0.5	0.6	0.8	\$1038
Elaboration	2.1	1.9	1.1	\$4151

Staffing by Phase

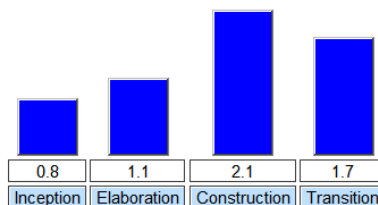


Figure 2: COCOMO Cost Report

4. System Requirements

Introduce the proposed system (one or two paragraphs).

4.1. Functional and Nonfunctional Requirements

This chapter contains information about the functional and non-functional system requirements. Each requirement contains the use case id's which makes use of those requirements.

Functional Requirements

This session highlights the system requirements to be implemented.

1. The system shall provide a Create login form (*Use Case ID: CVM-M - HL - 23*)*
2. The system shall allow user to Login (*Use Case ID: CVM-M - HL - 02*)*
3. The system shall allow Mask Password(*Use Case ID: CVM-M-SL-25*)*
4. The system shall allow users to Recovered Password (*Use Case ID: CVM-M-HL-21*)*
5. The system shall allow user to Query Information (*Use Case ID: CVM-M - HL - 07*)*
6. The system shall allow data to be compile (*Use Case ID: CMV-M-SL-10*)*
7. The system shall stored compile data into a temporary storage (*Use Case ID: CVM-M-SL-13*)*
8. The system shall display the patients information (*Use Case ID: CVM-M-SL-16*)*
9. The system shall have a session time if user become idle (*Use Case ID: CVM-M-SL-17*)*
10. The system shall allow users to log out from the system (*Use Case ID: CVM-M-HL-22*)*

*More detailed use case requirements can be found in Appendix A

Nonfunctional Requirements

Usability

- On average the user should take 20 seconds to perform the log in procedure
(*Use Case ID: CVM-M - HL - 02*).
- On Average the user should take 15 seconds to request patient's information
(*Use Case ID's: CVM-M - HL - 07*).
- On average the user shall take 3 minutes to complete the send request form
(*Use Case ID's: CVM-M - HL - 21*).

- On average the user shall take 20 seconds to create an account
(Use Case ID's: CVM-M - HL - 23).
- A help file should be available to describe the information been displayed
(Use Case ID's: CVM-M-SL-16).
- Training is not require to use this feature
(Use Case IDs: CVM-M - HL – 02, CVM-M - HL - 07, CVM-M-HL-21,
CVM-M-HL-22, CVM-M - HL – 23, CVM-M-SL-08, CMV-M-SL-10,
CVM-M-SL-13, CVM-M-SL-16)

Reliability

- 10% failures for every twenty four hours of operation are acceptable.
(Use Case IDs: CVM-M - HL – 02, CVM-M - HL – 23, CVM-M-SL-16).
- 5% failures for every twenty four hours of operation is acceptable
(Use Case IDs: CVM-M - HL – 07, CVM-M-HL-21, CVM-M-HL-22, CVM-M-SL-08,
CMV-M-SL-10).
- 2% failure for every 4 Hours of operation is acceptable
(Use Case IDs: CVM-M-SL-13).
- System should be available all the time except during server maintenance.
(Use Case IDs: CVM-M - HL – 02, CVM-M - HL – 07, CVM-M-HL-21,
CVM-M-HL-22, CVM-M - HL – 23, CVM-M-SL-08, CMV-M-SL-10, CVM-M-SL-13,
CVM-M-SL-16).

Performance

- User verification should be done under 1 second
(Use Case IDs: CVM-M - HL – 02, CVM-M - HL - 23).
- System should be able to handle 10 requests per second
(Use Case IDs: CVM-M-HL – 02, CVM-M - HL – 07, CVM-M - HL - 23).
- System shall base query time on the amount of current data sources
(Use Case IDs: CVM-M - HL - 07).
- Request shall be sent and saved within 7 seconds
(Use Case IDs: CVM-M - HL - 21).
- System shall be able to handle 50 requests in 1 minute
(Use Case IDs: CVM-M - HL - 21).
- Request shall be sent and confirmed within at least 10 seconds
(Use Case IDs: CVM-M - HL - 22).

- System shall be able to handle 100 requests in 1 minute
(Use Case IDs: CVM-M - HL - 22).
- Request shall be handle in less than 5 seconds
(Use Case IDs: CVM-M-SL-08).
- System shall be able to handle on average of 100 request per day.
(Use Case IDs: CVM-M-SL-08, CMV-M-SL-10).
- Request should be handle in less than 5 minutes
(Use Case IDs: CVM-M-SL-10).
- System shall be able to handle 50 I/O's per second
(Use Case IDs: CVM-M-SL-13).
- System shall take no longer than 30 seconds to layout information being display to the CVM-M user
(Use Case IDs: CVM-M-SL-16).
- File access time shall be at least at the speed of a conventional network
(Use Case IDs: CVM-M-SL-13).

Supportability

- Supported by any JVM
(Use Case IDs: CVM-M - HL – 02, CVM-M - HL – 07, CVM-M - HL – 21, CVM-M - HL – 22, CVM-M - HL - 23).
- The system shall be able to handle database engine and file systems.
(Use Case IDs: CVM-M-SL-08).
- The transferring of files needs a file allocation system
(Use Case IDs: CVM-M-SL-13).
- The display of the information needs a control that can read and layout XML data
(Use Case IDs: CVM-M-SL-16).
- The system should be able to handle database engine, data file such as documents, photos, video, and any window related file
(Use Case IDs: CVM-M-SL-10).

4.2 Analysis of Requirements

The following sections will go over the Models that describe how the user interacts with the system resources. The models included are the following: Use Case, Static, and Dynamic models.

4.2.1 Use Case Model

Different cases that the CVM-M user can encounter when entering the system are shown. The main cases occurred when the user login's, steps into CVM-M Interface, Query's Information and Compiles Information. Each case that display's information shows the word "Display" in it. Cases that have no dependencies or inclusions are use for a single purpose. Compiling and Sorting Exceptions case extends the Compile Information case by responding to all included Compile Information cases. Tree new use cases were added, Link Patient Id, TegesICU Patient and HealthVault Patient.

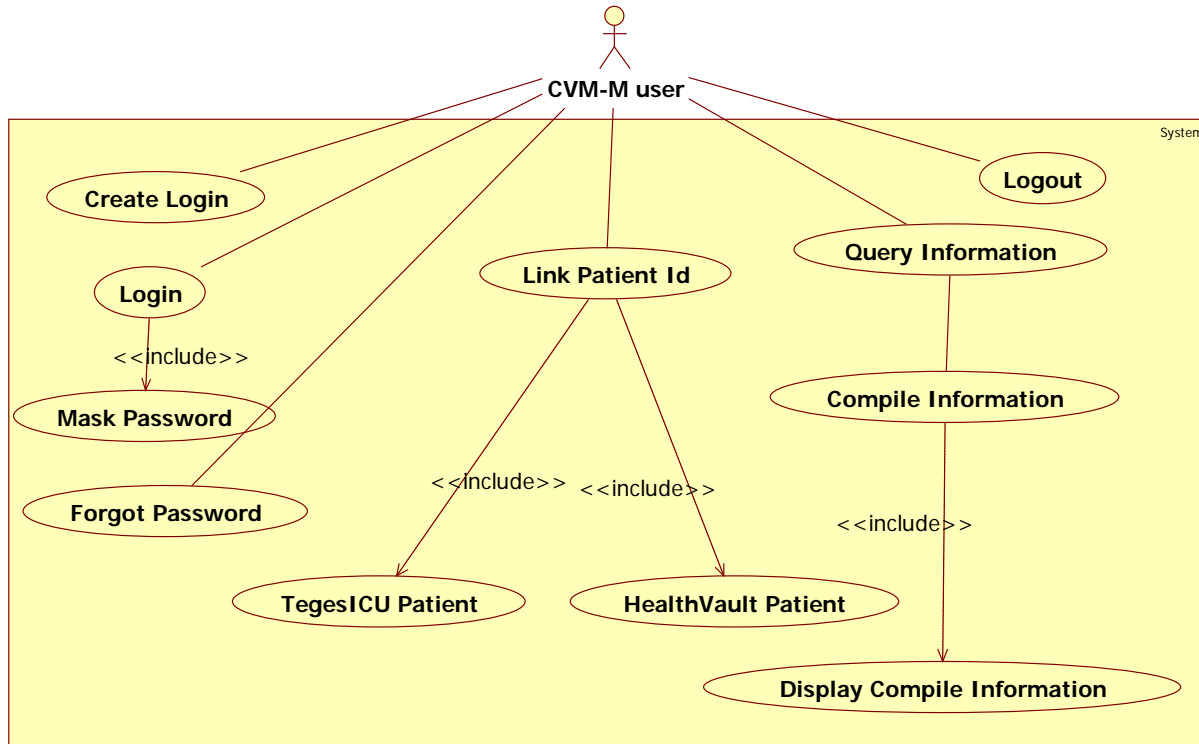


Figure 3: Use case model

4.2.2. Static Models

The CVM-M user has to be instantiated at least once to start the system cycle. Each CVM-M user that logs into the system will contain a session, which keeps tracks of the user state while in the system. Each Session in the system will submit a request to the CVM-M Mediator base on the CVM-M user input. Below you will find all the scenarios with their reference to the object diagrams.

Scenarios:

Scenario Id: CVM-M-HL-23-Create Login– Dr. John wants created a new login account.

Summary: This scenario describes a sequence of events that Dr. John follows to create a new login account.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has CVM-M up and running but is not logged in.

Description of the scenario:

Dr. John clicks on CREATE ACCOUNT button.

CVM-M redirects Dr. John to the create account page.

Dr. John enters username (drjohn123), enters password (access), selects a predefined question (What kind of access?), an answer to the question (totalaccess) and enters email address (drjohn123@yahoo.com).

Dr. John clicks on the CREATE button.

CVM-M redirects Dr. John to the login page.

Post Condition:

Dr. John has a new an active account and is ready to login.

*See Appendix C: Figure 7

Scenario Id: CVM-M-SL-25-Mask Password– Dr. John wants login onto CVM.

Summary: This scenario describes a sequence of events that CVM has to perform in order to maintain a secret password. Dr. John should be the only person who knows what password is being typed on the screen.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has CVM-M up and running but is not logged in.

Description of the scenario:

Dr. John enters username (drjohn123).

Dr. John starts to enter his password (access).

CVM-M recognizes that Dr. John is entering password and mask it so is not recognized by anyone around looking at the screen.

Post Condition:

Dr. John clicks on the LOGIN button.

*See Appendix C: Figure 4

Scenario Id: CVM-M-SL-17-Session Time Out– Dr. John is idle for a specific time.

Summary: This scenario describes a sequence of events that CVM-M follows to ensure security while system is idle.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has CVM-M up and running.

Dr. John is logged onto CVM-M.

Description of the scenario:

CVM-M is idle for 15 minutes.

CVM-M displays a timeout dialog with 30 second timer.

CVM-M logs Dr. John out the system when timer has reached 0 on the counter.

CVM-M redirects Dr. John to the login page.

Post Condition:

Dr. John is logged out of the system.

*See Appendix C: Figure 5

Scenario Id: CVM-M-SL-16-Display Compile Information– Dr. John wants to display the compile information being held in the temporary folder

Summary: This scenario describes a sequence of events Dr. John and CVM-M go through to display compile information.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has logged in and is running CVM-M.

Description of the scenario:

Dr. John enters the patient id “12345”.

Dr. John clicks on the SEARCH button.

CVM-M queries the patient data from database.

CVM-M starts to compile all data sources base on patient id.

CVM-M temporarily stores information in a folder.

CVM displays the information using a XML schema.

Post Condition:

CVM-M is displaying compile information.

*See Appendix C: Figure 11

Scenario Id: CVM-M-SL-13-Temporary Storage– Dr. John wants to store information in a temporary folder.

Summary: This scenario describes a sequence of events Dr. John and CVM-M go through to save all organized information on a temporary folder.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has logged in and is running CVM-M.

Description of the scenario:

Dr. John enters the patient id.

Dr. John clicks on the SEARCH button.

CVM-M queries the patient data from database.

CVM-M starts to compile all data sources base on patient id.

CVM-M temporarily stores information in a folder.

Post Condition:

CVM hold information on a temporary folder.

*See Appendix C: Figure 10

Scenario Id: CVM-M-SL-10- Compile Information– Dr. John wants to compile information.

Summary: This scenario describes a sequence of events that Dr. John follows to compile information from patient.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has logged in and is running CVM-M.

Description of the scenario:

Dr. John enters the patient id.

Dr. John clicks on the SEARCH button.

CVM-M queries the patient data from database.

CVM-M starts to compile all data sources base on patient id.

Post Condition:

CVM save information to temporary storage.

*See Appendix C: Figure 9

Scenario Id: CVM-M-HL-22-Logout– Dr. John wants to logout of CVM-M.

Summary: This scenario describes a sequence of events that Dr. John follows to logout or disconnects from CVM-M.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John is logged onto the system.

The CVM-M interface is being displayed.

Description of the scenario:

Dr. John clicks on the LOGOUT button.

CVM-M asked for confirmation by clicking the OK button or CANCEL button.

Dr. John clicks on the OK button.

CVM-M logs Dr. John out of the system.

CVM-M redirects Dr. John to the login page.

Post Condition:

Dr. John is no longer logged onto the system.

*See Appendix C: Figure 3

Scenario Id: CVM-M-HL-21- Forgot Password– Dr. John wants to get a new password.

Summary: This scenario describes a sequence of events that Dr. John follows to get a new password after old one, was forgotten.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has not logged onto the CVM-M

CVM-M is displaying the login page.

Description of the scenario:

Dr. John clicks on the FORGOT PASSWORD link.

CVM-M redirects Dr. John to the forgot password page.

Dr. John enters his email address (drjohn123@yahoo.com) and answers the predetermine question (totalaccess).

Dr. John clicks on the SEND button.

CVM-M will verify data was correct and issue a new password (access2).

CVM-M notifies DR. John of new password

Post Condition:

The new password replaces the old password in the CVM-M.

*See Appendix C: Figure 6

Scenario Id: CVM-M-HL-07- Query Information – Dr. John wants to query information.

Summary: This scenario describes a sequence of events that Dr. John follows to query information based on a patient id.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

Dr. John has logged in and is running CVM-M.

CVM-M has already redirected Dr. John to the CVM-M interface.

Description of the scenario:

Dr. John enters the patient id.

Dr. John checks the box next to the template desired.

Dr. John clicks on the SEARCH button.

Post Condition:

CVM-M will query the patients' data from the database.

*See Appendix C: Figure 8

Scenario Id: CVM-M-HL-02- Login – Dr. John wants to login to CVM-M.

Summary: This scenario describes a sequence of events that Dr. John follows to login to the CVM-M.

Actors: Dr. John

Media types: None

Scenario Description

Preconditions:

The login window has been activated and displayed properly.

Description of the scenario:

The Login window is displayed to Dr. John.

Dr. John enters his user id (drjohn123) and password (access).

Dr. John clicks on the LOGIN button.

CVM-M identifies and authenticates Dr. John.

CVM-M redirects Dr. John to the CVM-M Interface after a successful identification and authentication.

Post Condition:

Dr. John is now able to perform tasks in the CVM-M Interface

*See Appendix C: Figure 2

4.2.3. Dynamic Models

- The Login Sequence Diagram shows the process that the system goes through when a user attempts to login onto the CVM-M Interface.
- The Logout Sequence Diagram shows the process that the system goes through when a user attempts to logout to the CVM-M Interface.
- The Mask Password Sequence Diagram shows the process that the system goes through when a user inserts a password in the Login Page.
- The Session Time-Out Sequence Diagram shows how the user would get logged off or disconnected when it becomes inactive during the CVM-M Interface.
- The Forgot Password Sequence Diagram shows the process that the system goes through when a user forgets his password and attempts to recover it.
- The Create Login Sequence Diagram shows the process that the system goes through when a new user attempts to create a login account.
- The Query Information Sequence Diagram shows the process that the system goes through when a new query is submitted.
- The Compile Information Sequence Diagram shows the process that the system goes through when the information been query is compiled.
- The Temporary Storage Sequence Diagram shows the process that the system goes through when the information compiled is stored in a temporary folder for the user.
- The Display Compile Information Sequence Diagram shows the process that the system goes through when the information compiled is displayed for the user in the CVM-M Interface.

*See Pages 41-49 for the Sequence Diagrams.

5. System Design (i.e., overall system design)

This chapter introduces the system decomposition and gives a high level description of the system design architecture. It also describes the subsystem decomposition and identifies the requirements associated with each subsystem. In section 2.3 there is a diagram describing the hardware and software mapping, which shows the association between the subsystems and hardware. This chapter will also identify persistent data management in order to identify data that needs to be stored.

5.1. Overview

The CVM-Mediator system uses two-tier architecture which is composed of a client and multiple servers. The client interface consists of a Mediator and Proxy architecture. The Reason for the Mediator is to control the interaction between all the data sources and to allow the system to become extensible. The system has two servers at the moment, which are TegesICU and Microsoft Vault (Microsoft HealthVault).

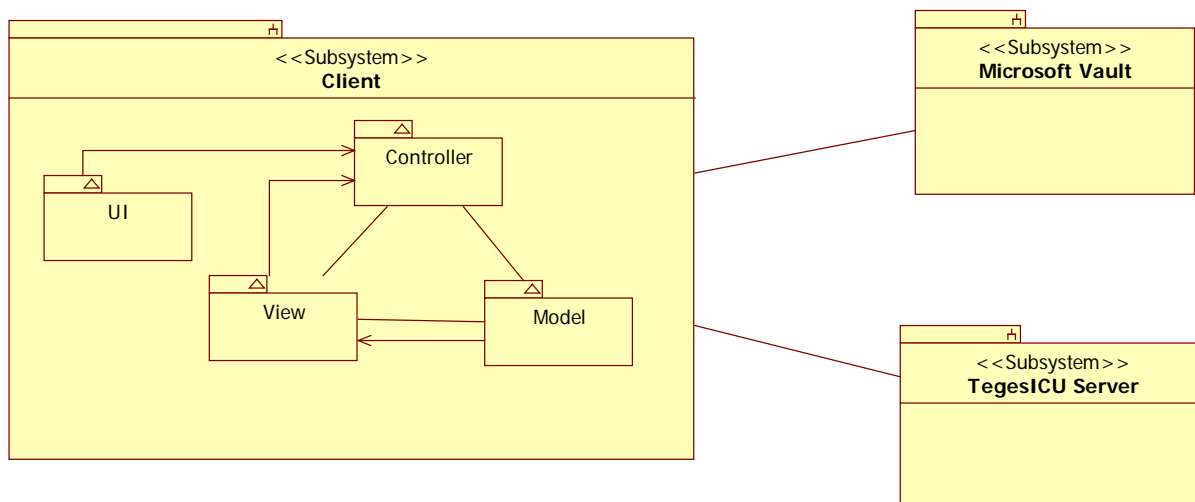


Figure 4: Package Diagram

The client interface subsystem is in charge of presenting the user interface, and application logic that will interact with the servers. It allows users to login and log out of the system; however, the main functionality is that it allows users to query patient's health records.

The mediator in the client interface will compile the patient's information, which will later be displayed to the CVM-M user. Inside the Client we use a model view controller architecture which will help us query the data, compile it, and display the information on different views determined by the controller. Inside the client we also have a package called the UI which contains all the forms and interfaces of the system.

The Servers subsystem consists of multiple servers, which contain the patient's health records. For the TegesICU server, SQL (SQL) commands are used to query the data, while for the Microsoft Vault, an SDK (Software development kit) library will be used to obtain the data from the data source.

5.2. Subsystem Decomposition

Mediator Subsystem (client)

The client is composed of a Model-View-Controller (MVC), which controls the different templates or layouts to be displayed. Also the client has a mediator design pattern which controls the interactions between all the data sources and provides an interface for user interaction. This subsystem provides CVM-M users with the capability to log in and log out of the system. It also allows CVM-M users to search patient's health records using a unique patient ID. The mediator connects to multiple data sources in order to retrieve the patient's information, which is to be displayed to the CVM-M users.

Data Sources Subsystem (servers)

This subsystem consists of several databases which will be used to retrieve patients health record information. No data will be stored in this database, the main purpose, is to retrieve information only.

5.3. Hardware and Software Mapping

The CVM-Mediator software will consist of two-tier architecture. The system is composed of a client, which connects to several servers in order to retrieve patient's health records. The client is supported by all windows operating system and will require at least 512 MB of RAM. In order to interact with the system it is required to have Firefox (Mozilla Firefox) or Google chrome (Google Chrome) installed. The mediator will be able to connect to several servers such as the TegesICU server and Microsoft Vault.

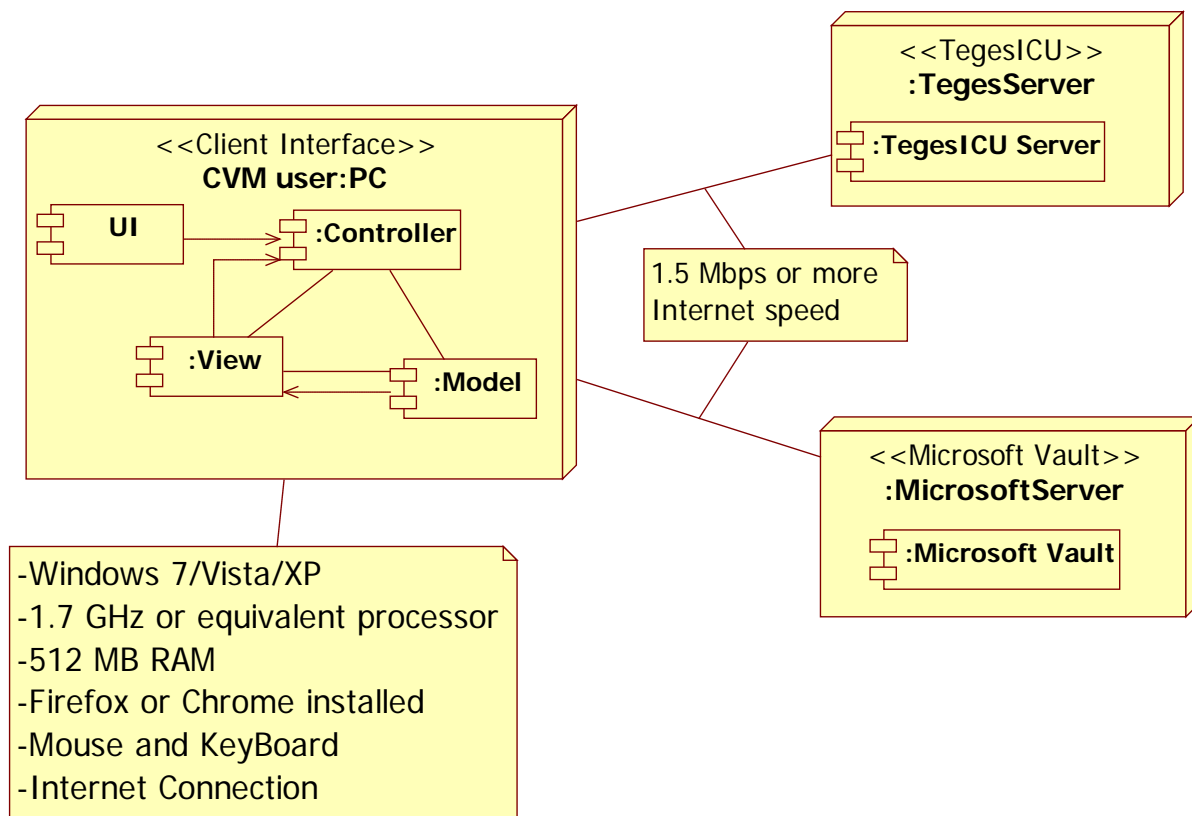


Figure 5: Deployment Diagram

5.4. Persistent Data Management

5.4.1. Data Retrieval

The System is accessing two data sources, which contain information that will be retrieve and process through the CVM-Mediator System.

The first data source (see Figure 4) is the Microsoft Vault system. The data contained in this system, is a collection of health care system summaries. The information that is extracted from the Microsoft Health Vault will be transferred to the CVM-Mediator as a file, which will be mapped, store and display within the CVM-Mediator system.

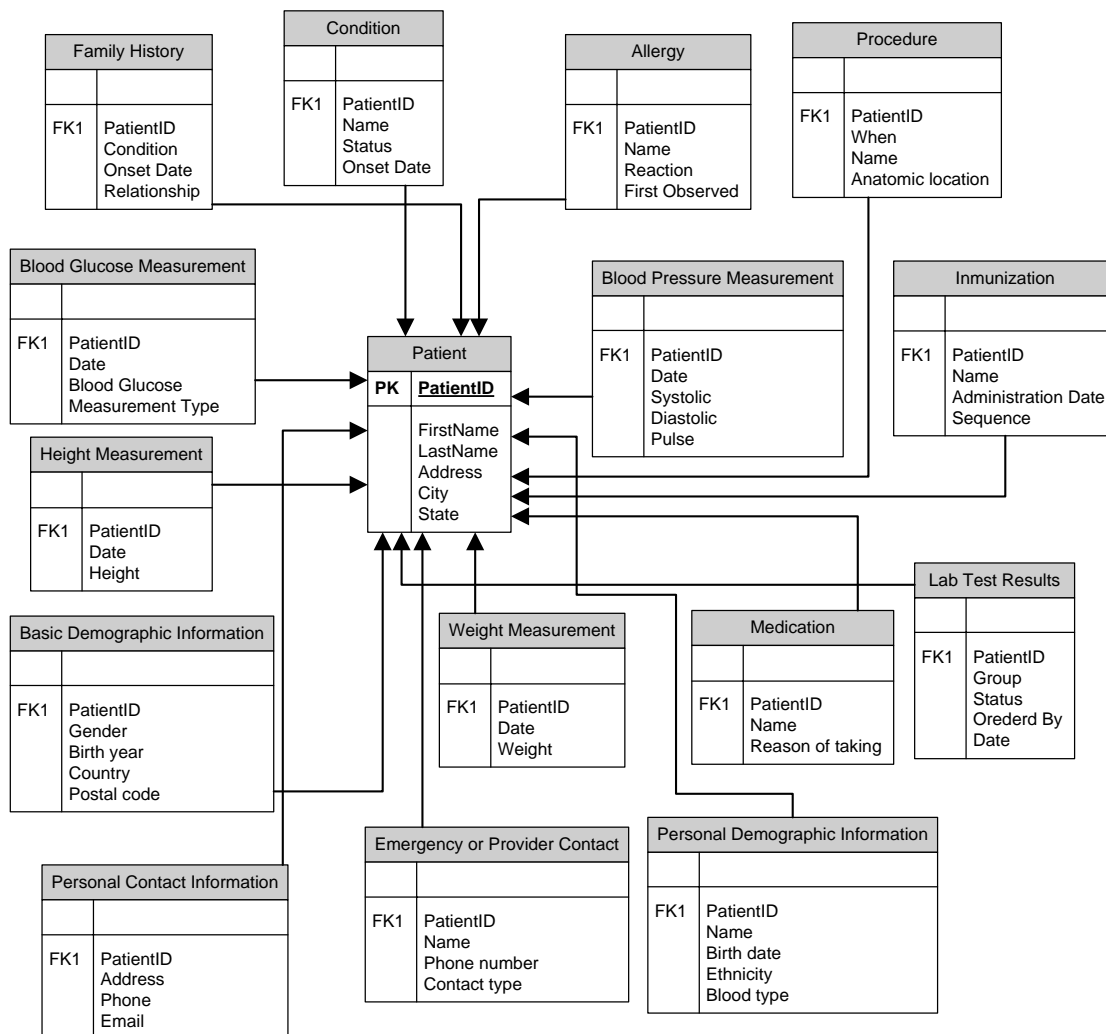


Figure 6 – Microsoft Vault

The second data source (see Figure 5) is the TegesICU system. The data contained within this system is composed of different tables that are related by a patientID, HospitalizationID, and MedicationID. Some of the tables in the TegesICU system will contain references to tables that will be retrieved by the CVM-Mediator, which will be use as part of the retrieval. The Data will be retrieved by creating a relational database connection to the TegesICU data source. The data retrieved will not be stored until is ready to be display.

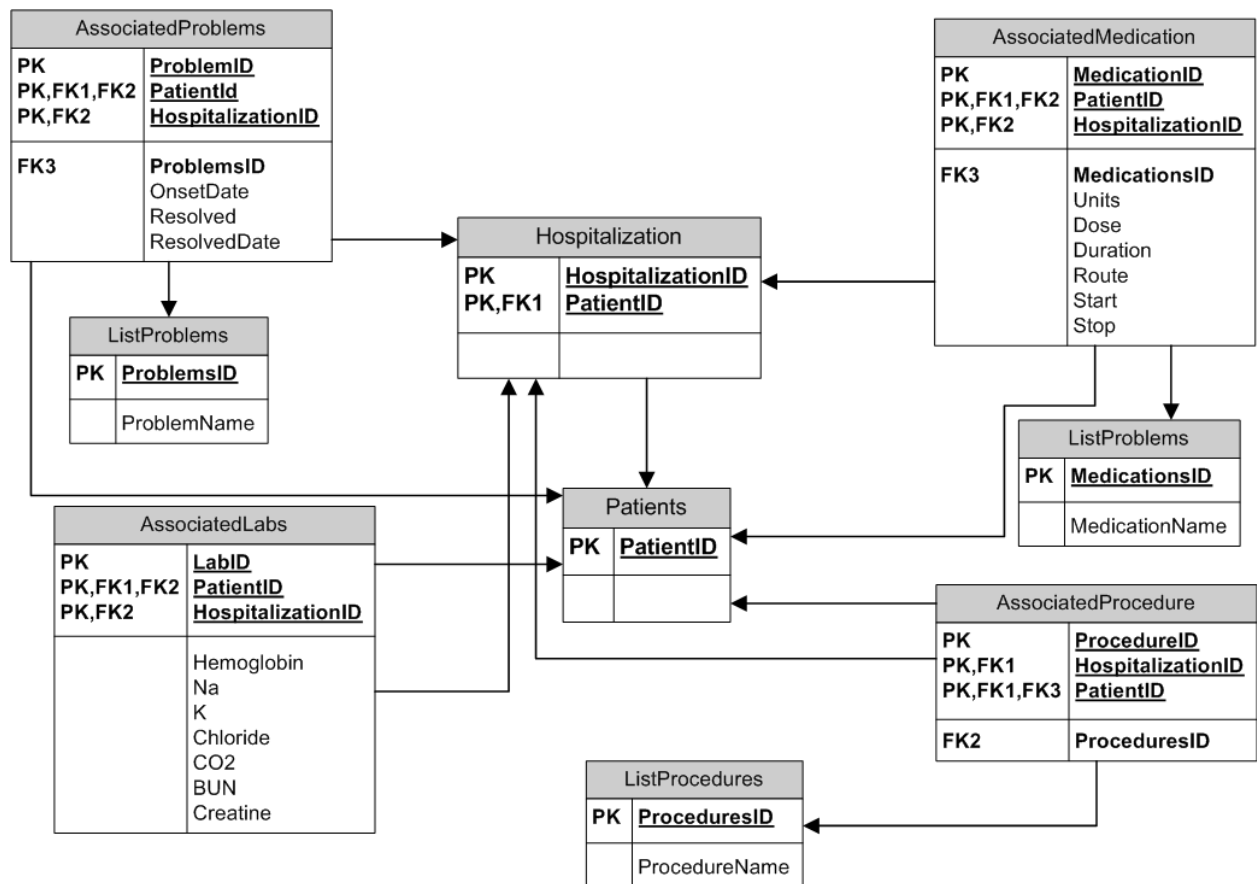


Figure 7 - TegesICU Database

6. Detailed Design

In this chapter a deeper and more complete description of the system will be discussed. The classes use and the interactions between them will be described. The main control will be viewed in a state diagram, which will show the different actions taken by the user in. The algorithm use in the problem solution will be discussed. The refined sequence diagrams will be outlined. The main control class interfaces will described by its functions and constraints.

6.1. Overview

The two subsystems are the Client Interface and the Server Interface. In the Client Interface the CVM-M User will interact with the application that will drive the information, which is compiled and display to the user. The Client Interface is composed of a Model-View –Controller architecture pattern in order to presents different layouts of the patient’s health records. The controller will be in charge of retrieving the data, which the model will compile to later be presented to the user.

6.2. Static Model

The Client Interface is composed of MVC architecture pattern. Therefore we have three packages inside the client subsystem. Inside of the Client there are pages, users, and buttons (Command Pattern). The pages are a Login_Page, Lost_Password_Page, Create_Login_Page, CVM-Interface, and DS_Settings_Page. The users will be aggregated in many of the pages with the CVM-M_User object (Singleton Pattern). The buttons will be used in many of the pages to perform the redirect command, which loads new pages for the user.

Inside of the Controller we have a manager, Mediator Pattern, two data source structures, Proxy Pattern, and a display structure. The manager (Interface_Manager) controls the flow of information between the model and the view. The mediator controls the communication between the two data source structures by relaying their status to the system and providing useful information from the system. The proxy (data source structures) creates a binding between the Server Interface and the display structure. The display structure (Organized_DS) stores and organizes the data retrieved from the proxy, which is inside the Model package.

*See Appendix E for minimal class diagram

6.3. Dynamic Model

The state diagram describes how everything in the system revolves around the CVM Display control. The state machine represents the state of the Controller from the MVC architecture. We have the Session Time Out time to the Logout which leads to a final state. The search done by the user will trigger a query, which can send a bad status, sending the system into the CVM Display control to try again. If the query is successful depending on which is the next query, the system will continue to query items. Once all queries are done, the information is sent to be displayed, which once displayed, will return control to the CVM Display control.

*See Appendix E for state machine

Algorithm:

The main algorithm used in the system is a Depth First Search. It was implemented using an Adjacency Matrix, which makes the runtime $O(V^2)$.

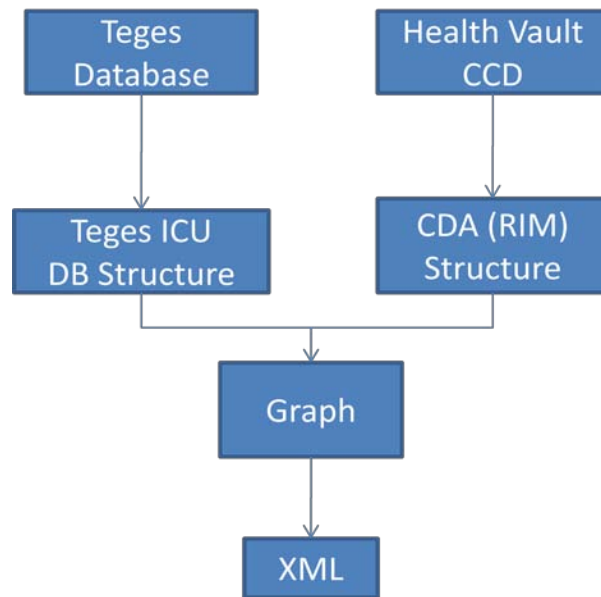


Figure 8: Data Retrieval and Compilation

Once the patients information has being retrieved and compiled into a graph. The algorithm is used to traverse the graph and make calls to print the nodes information into a XML file. Given a root index (i.e. Lab Results) the depth first search will traverse the tree and print each of the children's of lab result.

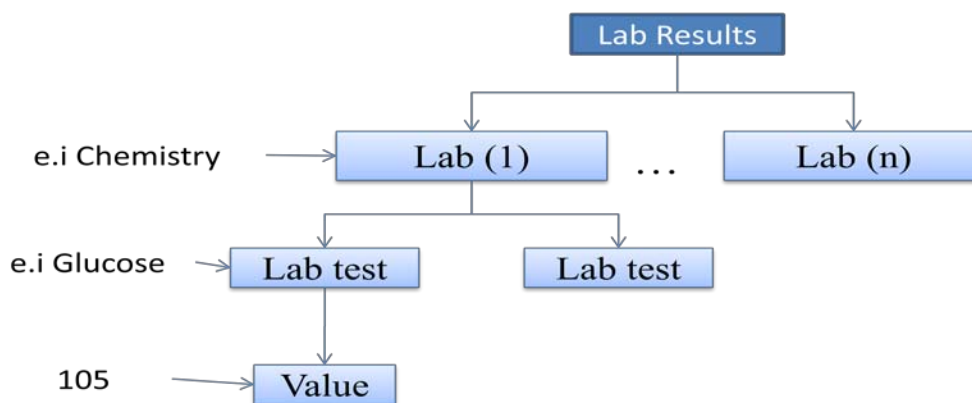


Figure 9: Lab Results Tree

6.4 Code Specification

Interface Manager

In the Client Interface the main control object is the Interface Manager. This object controls the flow of information between the user interface and the business services aspect of the application. It executes functions in a timely manner through the use of instantiated objects, which help control the application's flow of information.

The objects use in this object are User_Account, dataMediator, teges, ccd, dsTypes, and patientID. The User_Account is use to control the Session Time Outs of the CVM-M user. The dataMediator is use to create a communication mechanism between the data sources objects. The teges is a structured object, which stores the information extracted from the TegesICU database. The ccd is a list of structured CCD objects, which depending on the amount of CCD data sources will get bigger or contained nothing. The dsTypes object is a TreeMap, which contains a collection of index data sources name's that are use in the system to be able to retrieve necessary parameters for a specific datasource. The patientID is use to store a patient ID submitted by the user, so that it can be further process.

The functions use in the Interface Manager are the sessionTimeout, checkProcessStatus, retrieveDS_Type, and setPatientID.

- For the retrievePatients constraints are the following:
 - **Pre-Condition:** System has started running and the patients have been taking from the administration database.
 - **Invariants:** No patient may have been created at the time of retrieval, probably database issues could have occur also that were ignored.
 - **Post-Conditions:** The list is populated on the dropdown list the window that will link the data source to the patient.
- For the setCVMUser the constraints are the following:
 - **Pre-Condition:** The user is in the process of login into the system.
 - **Invariants:** None
 - **Post-Conditions:** The information of the user is displayed on the search window.
- For the retrieveDS the constraints are the following:
 - **Pre-Condition:** User has logged into the system and patient being used must have being link to all data sources; were the information, should be pulled from.
 - **Invariants:** While this function is running the system might be place on hold for a couple of seconds, so no process could be executed at this time.
 - **Post-Conditions:** The health information for the patient has been compile and displayed.

- For the findPatient the constraints are the following:
 - **Pre-Condition:** Patients have been retrieved and store in the system.
 - **Invariants:** No patients have been taking into the structure, so nothing can be look for.
 - **Post-Conditions:** A patient has been found and can be either use for processing or be a sign to not enter new patients.
- For the openGeneratedXML the constraints are the following:
 - **Pre-Condition:** Information about a patient has already been compile and been expanded into a XML file.
 - **Invariants:** Files IO and search errors can come up as the process is going.
 - **Post-Conditions:** An HTML file is displayed on a reliable HTML browser.
- For the getPatients the constraints are the following:
 - **Pre-Condition:** System has started running and the patients have been taking from the administration database.
 - **Invariants:** There could be database errors retrieving the items or patients could not exist, since they never were added.
 - **Post-Conditions:** System can now use this list of patients to perform any kind of search, retrieval or update on information, regarding the patients.
- For the convertToHTML the constraints are the following:
 - **Pre-Condition:** Information about a patient has already been compile and been exported into an XML file.
 - **Invariants:** A file could not be open, or found. The conversion came up with errors so no HTML document could generate it. The HTML was not save properly.
 - **Post-Conditions:** The generated HTML file is saved in a folder.
- For the retrievePatientList the constraints are the following:
 - **Pre-Condition:** The patients have been retrieved from the administrative database and populated into a structure in the system, which are showing a dropdown, user has log in into the system.
 - **Invariants:** The TegesICU database is down, or there was errors retrieving the necessary database on the SQL syntax.
 - **Post-Conditions:** A dropdown list is populated in the window where the TegesICU data source is link to the patient.

Source Interface

In the Server Interface, the Source Interface in the main control. This Object will be use to control the flow of information between the data sources and the application. It will retrieve

information base on calls send by the system, either by querying or using some other retrieval commands.

There is only one object call Category, which is use to categorized different parts of the query. All methods pre-conditions are that the Client Interface has initiated a call asking about a certain patient existence in the data sources. There are no invariants in these functions, since they don't really alter anything in the system. All methods post-conditions are that a query from multiple data sources has been compile and send back to the Client Interface.

7. System Validation

The system being tested is the CVM-Mediator application. For the purpose of this project only the following approaches will be use; System Testing and Subsystem testing.

7.1 Subsystem Test

Features to be tested: the Model package from the model view controller inside the client tier.

- Cvm.mediator.model.Orginizer
- Cvm.mediator.model.Graph
- Cvm.mediator.model.Vertex
- Cvm.mediator.model.MakeXML
- Cvm.mediator.model.CreateStandardGraph
- Cvm.mediator.model.CreateCCDHealthVault
- Cvm.mediator.model.CreateCCDTeges
- Cvm.mediator.model.GraphDFS

Only three test cases were needed to test the subsystem. The organizer class execute all the require procedures to compile the data. To test if the data was compiled we checked to see if the compiled file was created. In order to test if the Graph had added the correct number of nodes, the size of the graph was printed in every test case.

Test cases:

7.2 System Tests

This section contains the test cases used to test and validate the CVM-Mediator. To test the system, the user's scenarios were used to test for errors. The system test will test if the system meets with the requirements defined in the requirements document.

Test Cases:

Test Case ID	CVM-M-Login-SD-01
Purpose	To test if a user is able to log in with an existing account
Test Setup	The CVM-M application is running The login interface is being displayed User: "dr john" Password: "access". Exist
Test Inputs	User enters in the: Username box: "drjohn" Password box: "access" User click the "Login" Button
Expected Output	User is logged into the system

Test Case ID	CVM-M-Login-SD-02
Purpose	To test if a user is able to log in with an existing account that was just created.
Test Setup	<p>The CVM-M application is running</p> <p>The login interface is being displayed</p> <p>User Creates an account (see Manual for create account)</p> <p>Account Created:</p> <p>Username: "jan345"</p> <p>Password: "j6789"</p>
Test Inputs	<p>User enters in the:</p> <p>Username box: "jan345"</p> <p>Password box: "j6789"</p> <p>User clicks the "Login" button</p>
Expected Output	User is logged into the system

Test Case ID	CVM-M-Login-RD-01
Purpose	To test the system allows non-existing user to log in
Test Setup	<p>The CVM-M application is running</p> <p>The login interface is being displayed</p> <p>User: "nurseJoy" Password: "n789654". Does not exist</p>
Test Inputs	<p>Enter in the :</p> <p>Username box: "nurseJoy"</p> <p>Password box: "n789654"</p> <p>User clicks the "Login" button</p>
Expected Output	A message is display"The username or password is not valid"

Test Case ID	CVM-M-Create Login-SD-01
Purpose	To test is the system allows the creation of a new login account of a DOCTOR user that does not exist
Test Setup	The CVM-M application is running The login interface is being displayed Click on the “File” menu and click on “Create Account” Create account interface is displayed
Test Inputs	User enters the following data in the Create Account fields: Username: gwen Password: ascalon Confirm: ascalon Email: gwen@gmail.com Question: name a fort Answer: evonhacke Type: Doctor User clicks “Create” button
Expected Output	Account is created, and the create account interface closes

Test Case ID	CVM-M-Create Login-SD-02
Purpose	To test is the system allows the creation of a new login account of a NURSE user that does not exist
Test Setup	The CVM-M application is running The login interface is being displayed Click on the “File” menu and click on “Create Account” Create account interface is displayed
Test Inputs	User enters the following data in the Create Account fields: Username: test1 Password: t123456 Confirm: t123456 Email: test1@gmail.com Question: test subject name Answer: test1 Type: Nurse

	User clicks “Create” button
Expected Output	Account is created, and the create account interface closes

Test Case ID	CVM-M-Create Login-RD-01
Purpose	To test if the system allows the creation of a new login account of a user that already exist.
Test Setup	<p>The CVM-M application is running</p> <p>The login interface is being displayed</p> <p>Click on the “File” menu and click on “Create Account”</p> <p>User with username: gwen password: ascalon already exists</p> <p>Create account interface is displayed</p>
Test Inputs	<p>User enters the following data in the Create Account fields:</p> <p>Username: gwen</p> <p>Password: ascalon</p> <p>Confirm: ascalon</p> <p>Email: gwen@gmail.com</p> <p>Question: name a fort</p> <p>Answer: evonhacke</p> <p>Type: Doctor</p> <p>User clicks “Create” button</p>
Expected Output	User account cannot be created and user is informed that username already exist.

Test Case ID	CVM-M-Mask Password-SD-01
Purpose	To test if when users enter their password. The password is masked
Test Setup	<p>The CVM-M application is running</p> <p>The login interface is being displayed</p> <p>User: “drjames” Password: “access1”. Exist</p>
Test Inputs	<p>User enters in the:</p> <p>Username box: “drjames”</p> <p>Password box: “access1”</p>
Expected Output	Password is masked.

Test Case ID	CVM-M-Mask Password-SD-02
Purpose	To test if when users enter their password. The password is masked on a newly created account
Test Setup	The CVM-M application is running The login interface is being displayed User Creates an account (see Manual for create account) Account Created: Username: "john345" Password: "j9876"
Test Inputs	User enters in the: Username box: "john345" Password box: "j9876"

Test Case ID	CVM-M-Mask Password-RD-01
Purpose	To test if password is masked with a user that does not exist
Test Setup	The CVM-M application is running The login interface is being displayed
Test Inputs	Enters Username box: "mask1" Password box: "m123456"
Expected Output	Password is masked.
Expected Output	Password is masked.

Test Case ID	CVM-M-Forgot Password-SD-01
Purpose	To test is a user is able to retrieve its password on an existing account of a Doctor
Test Setup	The CVM-M application is running The login interface is being displayed Click on the "File" menu and click on "Forgot Password" Forgot password interface is display
Test Inputs	User Enters Email: drjohn456@yahoo.com When Secret Question is displayed interface is displayed

	User enters Answer: totalaccess
Expected Output	User's password is displayed.

Test Case ID	CVM-M-Forgot Password-SD-02
Purpose	To test is a user is able to retrieve its password on an existing account of a Nurse
Test Setup	The CVM-M application is running The login interface is being displayed Click on the "File" menu and click on "Forgot Password" Forgot password interface is display
Test Inputs	User Enters Email: nurse@yahoo.com When Secret Question is displayed interface is displayed User enters Answer: whataccess
Expected Output	User's password is displayed.

Test Case ID	CVM-M-Forgot Password-RD-01
Purpose	To test users are able to retrieve a password of an account that does not exist.
Test Setup	The CVM-M application is running The login interface is being displayed Click on the "File" menu and click on "Forgot Password" Forgot password interface is display
Test Inputs	User Enters: Email: nouser@gmail.com
Expected Output	A message is display "No user had email: nouser@gmail.com"

Test Case ID	CVM-M-Query Information-SD-01
Purpose	To test is the system allows to search an existing patient
Test Setup	The CVM-M application is running
Test Inputs	Patient ID: 12345 is entered Click on "Search" button

Expected Output	Patient health information is retrieved and displayed
------------------------	---

Test Case ID	CVM-M-Query Information-SD-02
Purpose	To test is the system allows to search an existing patient that is not linked to the Teges Database
Test Setup	The CVM-M application is running Patient ID: 54321 is not linked to the Teges Database
Test Inputs	Patient ID: 54321 is entered Click on “Search” button
Expected Output	Patient health information is retrieved and displayed

Test Case ID	CVM-M-Query Information-RD-01
Purpose	To test is the system allows to search an non-existing patient
Test Setup	The CVM-M application is running
Test Inputs	Patient ID: wrongID is entered Click on “Search” button
Expected Output	Message is displayed “User Does not Exist in the System”

Test Case ID	CVM-M-Compile Information-SD-01
Purpose	To test is data is compiled, when a doctor is logged on the system
Test Setup	The CVM-M application is running User “drjohn” is logged onto the system
Test Inputs	Patient ID: 12346 is entered Click on “Search” button
Expected Output	Patient health information is compiled and displayed

Test Case ID	CVM-M-Compile Information-SD-02
Purpose	To test is data is compiled, when a nurse is logged on the system
Test Setup	The CVM-M application is running User “nurse1” is logged onto the system
Test Inputs	Patient ID: 54326 is entered Click on “Search” button
Expected Output	Patient health information is compiled and displayed

Test Case ID	CVM-M-Compile Information-RD-01
Purpose	To test is data is compiled, when patient Id does not exist
Test Setup	The CVM-M application is running
Test Inputs	Patient ID: 963258 is entered Click on “Search” button
Expected Output	Message is displayed “User Does not Exist in the System”

Test Case ID	CVM-M- Display Compile Information-SD-01
Purpose	To test is the System displays compiled information when a doctor is logged on the system
Test Setup	The CVM-M application is running User “drjohn” is logged onto the system Patient Id: “patient1” exists and was entered in the patient Id textbox
Test Inputs	Clicks on Search button.
Expected Output	Patients Health Records are displayed

Test Case ID	CVM-M- Display Compile Information-SD-02
Purpose	To test is the System displays compiled information when a Nurse is logged on the system
Test Setup	The CVM-M application is running User “nurse1”is logged onto the system Patient Id: “patient2” exists and was entered in the patient Id textbox
Test Inputs	Clicks on Search button.
Expected Output	Patients Health Records are displayed

Test Case ID	CVM-M- Display Compile Information-RD-01
Purpose	To test is Compiled Information is displayed when user does not exist, and a nurse is logged on the system.
Test Setup	The CVM-M application is running User “nurse1”is logged onto the system

	Patient Id: "patient3" does not exists and was entered in the patient Id textbox
Test Inputs	Clicks on Search button.
Expected Output	A message is prompted "User Does not Exist in the System."

Test Case ID	CVM-M- Logout-SD-01
Purpose	To test is users type doctor are able to log out
Test Setup	The CVM-M application is running User "drjohn" is logged onto the system Clicks on "Logout" button. A message is displayed "Do you really want to Disconnect?"
Test Inputs	Clicks on "Yes" button.
Expected Output	User is disconnected, and the login interface is displayed.

Test Case ID	CVM-M- Logout-SD-02
Purpose	To test if users type nurse are able to log out
Test Setup	The CVM-M application is running User "nurse1" is logged onto the system Clicks on "Logout" button. A message is displayed "Do you really want to Disconnect?"
Test Inputs	Clicks on "Yes" button.
Expected Output	User is disconnected, and the Login interface is displayed.

Test Case ID	CVM-M- Logout-RD-01
Purpose	To test if system logs you out if the user cancels the confirmation
Test Setup	The CVM-M application is running User "nurse1" is logged onto the system Clicks on "Logout" button. A message is displayed "Do you really want to Disconnect?"
Test Inputs	Clicks on "No" button.
Expected Output	User is not disconnected and the confirmation dialog goes away.

7.3 Evaluation of Test

Subsystem test cases were performed using the JUnit Tester. The coverage tool used was Eclemma – java code coverage for Eclipse.

Subsystem Test cases results:

Test Case ID	Expected Output	Pass	Fail
CVM-M-Compile Data-SD-01	Data is compiled, and the data was written to a file	X	
CVM-M-Compile Data-SD-02	Data is compiled, and the data was written to a file	X	
CVM-M-Compile Data-RD-01	The File is created, but will only have 15 vertexes.	X	

The Model package line code coverage was 95.4%

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Subsystem Testing	99.0 %	284	3	287
CVM Mediator2	24.6 %	8627	26500	35127
src	50.3 %	5744	5672	11416
cvm.mediator.view	0.0 %	0	109	109
cvm.mediator.controller	39.6 %	1019	1557	2576
cvm.mediator.client	0.0 %	0	3780	3780
cvm.mediator.model	95.4 %	4725	226	4951
Vertex.java	50.0 %	12	12	24
Organizer_DS.java	92.2 %	119	10	129
CreateStandardGraph.java	100.0 %	200	0	200
Graph.java	97.8 %	218	5	223
GraphDFS.java	100.0 %	243	0	243
CreateCCDTeges.java	98.7 %	1137	15	1152
MakeXML.java	89.9 %	1098	124	1222
CreateCCDHealthVault.java	96.6 %	1698	60	1758
lib	12.2 %	2883	20828	23711

Figure 10: Model Package Coverage

System test cases were performed manually.

System Test cases results:

Test Case ID	Expected Output	Pass	Fail
CVM-M-Login-SD-01	User is logged into the system	X	
CVM-M-Login-SD-02	User is logged into the system	X	
CVM-M-Login-RD-01	A message is display “The username or password is not valid”	X	
CVM-M-Create Login-SD-01	Account is created, and the create account interface closes	X	
CVM-M-Create Login-SD-02	Account is created, and the create account interface closes	X	
CVM-M-Create Login-RD-01	User account cannot be created and user is informed that username already exist.		X
CVM-M-Mask Password-SD-01	Password is masked.	X	
CVM-M-Mask Password-SD-02	Password is masked.	X	

CVM-M-Mask Password-RD-01	Password is masked.	X	
CVM-M-Forgot Password-SD-01	User's password is displayed.	X	
CVM-M-Forgot Password-SD-02	User's password is displayed.	X	
CVM-M-Forgot Password-RD-01	A message is display "No user had email: nouser@gmail.com"	X	
CVM-M-Query Information-SD-01	Patient health information is retrieved and displayed	X	
CVM-M-Query Information-SD-02	Patient health information is retrieved and displayed	X	
CVM-M-Query Information-RD-01	Message is displayed "User Does not Exist in the System"	X	
CVM-M-Compile Information-SD-01	Patient health information is compiled and displayed	X	
CVM-M-Compile Information-SD-02	Patient health information is compiled and displayed	X	
CVM-M-Compile Information-RD-01	Message is displayed "User Does not Exist in the System"	X	
CVM-M- Display Compile Information-SD-01	Patients Health Records are displayed	X	
CVM-M- Display Compile Information-SD-02	Patients Health Records are displayed	X	
CVM-M- Display Compile Information-RD-01	A message is prompted "User Does not Exist in the System."	X	
CVM-M- Logout-SD-01	User is disconnected, and the Login interface is displayed.	X	
CVM-M- Logout-SD-02	User is disconnected, and the Login interface is displayed.	X	
CVM-M- Logout-RD-01	User is not disconnected and the confirmation dialog goes away.	X	

8. Glossary

Compile Information: Information that is manage, resource, and collected by the system.

CVM-M: Communication Virtual Machine-Mediator, is a system that translate different data sources into usable schema for displaying.

CVM-M Interface: Control Page where CVM-M user will request the task that the system will get assign to.

CVM-M User: Is the Entity that request information from the system.

Data Source: The repository that contains information of a medical record that is use in a system

File System a system of classifying into files (usually arranged alphabetically).

Google Chrome: Is a web browser developed by Google that uses the WebKit layout engine and application framework. It was first released as s beta version for Microsoft Windows. (Google Chrome).

GUI (Graphical User Interface) : Allows users to interact with programs in more ways than typing such as computers. A GUI offers graphical icons, and visual indicator, as opposed to text-bases interfaces, typed command labels or text navigation to fully represent the information and actions available to a user. (Graphical user interface).

Java: is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. (Java)

JVM (Java Virtual Machine) enables a set of computer software programs and data structures to use a virtual machine model for the execution of other computer programs and scripts. The model used by a JVM accepts a form of computer intermediate language commonly referred to as Java byte code. This language conceptually represents the instruction set of a stack-oriented, capability architecture. (Java Virtual Machine).

Microsoft HealthVault: Is a platform from Microsoft to store and maintain health and fitness information. Started in October 2007, the website is accessible at www.healthvault.com. HealthVault record stores an individual's health information. Access to a record is through a HealthVault account, which may be authorized to access for multiple individuals. (Microsoft HealthVault).

Mediator Design Pattern: One of 23 design patterns described in Design Patterns: Elements of Reusable Object oriented Software, provides a unified interface to a set of interfaces in a subsystem. This pattern is considered to be a behavioral pattern due to the way it can alter the program's running behavior. (Mediator pattern)

Medical Record: Is an item in the form of a written or electronic document that shows information about a particular person's health diagnosis.

Mozilla Firefox: Is a free and open source web browser descended from the Mozilla Application Suite and managed by Mozilla Corporation. (Mozilla Firefox).

Proxy Design Pattern: A proxy, in its most general form, is a class functioning as an interface to something else. The proxy could interface to anything: a network connection, a large object in memory, a file, or some other sources that is expensive or impossible to duplicate. (Proxy pattern)

Query Information: Task that CVM-M User assigns to system to compile information.

SDK (Software Development Kit): Is typically a set of development tools that allows for the creation of applications for certain software package, software framework, hardware platform, computer system or similar platform. (Software development kit).

Session Time-Out: Times the amount of time a CVM-M User is kept alive while idle.

SQL (Structured Query Language) a worldwide standard used to manage data in relational databases. SQL facilitates the sharing of data especially in large and interconnected databases. (SQL).

System: The entity that takes care of all the CVM-M User input requests and return outputs based on these requests.

TegesICU: A system used in Miami Children Hospital for internal medical information.

Temporary Storage: Space used to store compile information while system is processing.

USDP (Unified Software Development Process): The USDP or Unified Process is a popular incremental software development process framework. The Unified Process is not simply a process, but rather an extensible framework which should be customized for a specific organizations or projects. (Unified Process).

UML (Unified Modeling Language) is an open method used to specify, visualize, construct and document the artifacts of an object-oriented software-intensive system under development. It is typically used in large development teams as a bridge between process models and software development. Good process modeling tools can output the information required to develop the services required to UML, so that the development team can import the information directly into their software development tools. Some developers insist on hand crafting the UML and ignoring process inputs. However, despite their claims, it cannot replace process modeling to define business processes effectively. (Unified Modeling Language).

XML (Extensible Markup Language) is a set of rules for encoding documents electronically. It is defined in the produced by the W3C and several other related specifications; all are fee-free open standards. (XML)

XML Schema is a way to define and validate the XML file.

9. References

- “Proxy”. .NET Design Pattern and Architecture. n.d. Web. October 28, 2010.
<http://www.dofactory.com/Patterns/PatternProxy.aspx>.
- “Mediator”. .NET Design Pattern and Architecture. n.d. Web. October 28, 2010.
<http://www.dofactory.com/Patterns/PatternMediator.aspx>.
- “Mediator pattern”. Wikipedia. October 17, 2010. Web. October 28, 2010.
http://en.wikipedia.org/wiki/Mediator_pattern.
- “Proxy pattern”. Wikipedia. October 24, 2010. Web. October 28, 2010.
http://en.wikipedia.org/wiki/Proxy_design_pattern.
- “XML”. Wikipedia. October 26, 2010. Web. October 28, 2010. <http://en.wikipedia.org/wiki/Xml>.
- “Unified Modeling Language”. Wikipedia. October 28, 2010. Web. October 28, 2010.
http://en.wikipedia.org/wiki/Unified_Modeling_Language#Overview.
- “Java Virtual Machine”. Wikipedia. October 24, 2010. Web. October 28, 2010.
<http://en.wikipedia.org/wiki/JVM>.
- “SQL”. Wikipedia. October 24, 2010. Web. October 28, 2010. <http://en.wikipedia.org/wiki/SQL>.
- “Graphical user interface”. Wikipedia. October 27, 2010. Web. October 28, 2010.
<http://en.wikipedia.org/wiki/GUI>.
- “Java (programming language)”. Wikipedia. October 25, 2010. Web. October 28, 2010.
[http://en.wikipedia.org/wiki/Java_\(programming_language\)](http://en.wikipedia.org/wiki/Java_(programming_language)).
- “Unified Process”. Wikipedia. September 20, 2010. Web. October 29, 2010.
http://en.wikipedia.org/wiki/Unified_Software_Development_Process.
- Olmos, Ivan, et al. “CVM Mediator Deliverable 2”. BS senior project. Florida International University, 2010.
- “Microsoft HealthVault”. Wikipedia. August 13, 2010. Web. October 29, 2010.
http://en.wikipedia.org/wiki/Microsoft_HealthVault.
- “Software development kit” Wikipedia. October 11, 2010. Web. October 29, 2010.
<http://en.wikipedia.org/wiki/SDK>.
- “Mozilla Firefox”. Wikipedia. October 31, 2010. Web. October 31, 2010.
<http://en.wikipedia.org/wiki/Firefox>.
- “Google Chrome”. Wikipedia. October 31, 2010. Web. October 31, 2010.
http://en.wikipedia.org/wiki/Google_chrome.
- “Java”. Wikipedia. October 25, 2010. Web. October 31, 2010.
[http://en.wikipedia.org/wiki/Java_\(programming_language\)](http://en.wikipedia.org/wiki/Java_(programming_language)).

Clarke, Peter J. *Evolution of Software Design, Overview of MDSD*. Class Lecture. CEN 5064/4021. Florida International University, Miami, FL. January 2009.

“HealthVault Information Site Service Agreement.” Microsoft® HealthVault™. August 2009. Web. November 29, 2010. <https://www.healthvault.com/terms-of-use.aspx>.

“Privacy Policy”. Teges™ i-Rounds™. Web. November 29, 2010. <https://irounds.mch.com/tegesICU/privacypolicy.html>.

“JAXB”. GlassFish. November 27, 2010. Web. December 2, 2010. <https://jaxb.dev.java.net/>

“xpp3 1.1.4c.jar”. 12 Demo Source and Support. Web. December 2, 2010. <http://www.java2s.com/Code/Jar/STUVWXYZ/Downloadxpp3114cjar.htm>.

“Microsoft® SQL Server JDBC Driver 3.0”. Microsoft®. Web. December 2, 2010. <http://www.microsoft.com/downloads/en/details.aspx?FamilyID=a737000d-68d0-4531-b65d-da0f2a735707&displaylang=en>.

“SQLite Download Page”. SQLite. Web. December 2, 2010. <http://www.sqlite.org/download.html>.

“SqliteJDBC”. Zentus. Web. December 2, 2010. <http://www.zentus.com/sqlitejdbc/>.

“JCom(Java-COM Bridge)”. Source Forge. Web. December 2, 2010. <http://sourceforge.net/projects/jcom/>.

“GNU Lesser General Public License”. GNU Operating System. October 9, 2010. Web. December 2, 2010. <http://www.gnu.org/copyleft/lesser.html>.

“Eclipse Foundation Software User Agreement”. Eclipse. April 14, 2010. Web. December 5, 2010. <http://www.eclipse.org/legal/epl/notice.php>.

10. Appendix

10.1 Appendix A - Project schedule (Gantt chart or PERT chart).

The following figure describes the work schedule only for the last deliverable.

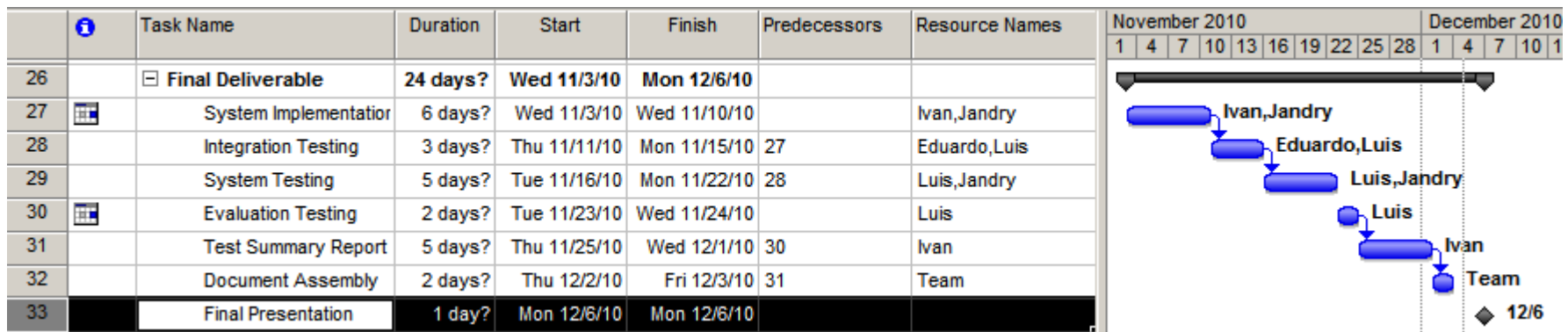


Figure 11: Project Schedule

10.2 Appendix B – All use cases with nonfunctional requirements.

Use Case ID: CVM-M - HL - 02 - Login

Use Case Level: High-level

Details:

Actor: CVM-M user

Pre-conditions:

1. The login window (see Appendix F: Figure 22) has been enabled and display's properly.

Description:

1. Use case begins when CVM-M user enters the following data: **user id, and password**.
2. The CVM-M user clicks on the **Login** button.
3. The system verifies the logon information.
4. Use case ends when the CVM- Interface (see Appendix F: Figure 26) is displayed.

Relevant requirements: None.

Post-conditions:

1. The CMM-M user is logged onto the system.

Alternative Courses of Action:

1. In step D.2 (step 2 of Description section) the user has the option to click on the cancel button, which will cause the program to close.
2. In step D.3 if any of the required fields are blank or invalid the system shall request the user to input valid data.

Extensions: None.

Exceptions:

1. After the user enters the required data the **login** button is disabled.

Related Uses Case:

CVM-M – SL – 25 – Mask Password

Concurrent Uses: None.

Decision Support:

Frequency: On average, 100 are made daily by CVM-M users.

Criticality: High. Allows CVM-M users to login to the system

Risk: Low. Implementing this use case employs basic java code.

Constraints:

1. Usability:
 - a) No previous Training Time
 - b) On average the user should take 20 seconds to login.
2. Reliability:
 - a) Mean time to Failure – 10% failures for every twenty four hours of operation is acceptable.
 - b) Availability – System should be available all the time except during server maintenance.
3. Performance:
 - a) User verification should be done under 1 second.
 - b) System should be able to handle 10 requests per second.
4. Supportability:
 - a) The login page shall be correctly handled by any JVM

Modification History:

Owner: Jandry Guerra

Initiation date: 09/27/10

Date last modified: 10/01/10

Use Case ID: CVM-M - HL - 07 - Query Information

Use Case Level: High-level

Details:

Actor: CVM-M user

Pre-conditions:

1. The CVM-M user has successfully logged onto the system.
2. The CVM-M Interface has been displayed.

Description:

1. Use case begins when CVM-M user enters the following data: **patient id**.
2. Use case ends when the CVM- user clicks on the **Search** button (see Appendix F: Figure 26).

Relevant requirements: None.

Post-conditions:

1. The patient's data will be query from the database.

Alternative Courses of Action:

1. In step D.2 (step 2 of Description section) if the patient's ID does not exist, the CVM-M user is notified.

Extensions: None.

Exceptions:

1. The CVM-M user submitted invalid values.

Related Uses Case:

CVM-M – SL – 10 – Compile Information

Concurrent Uses:

CVM-M - SL - 14 - Display Progress

Decision Support:

Frequency: On average, 600 queries are made daily by CVM-M users.

Criticality: High. Allows CVM-M users to request patient's health records

Risk: Medium. Implementing this use case requires the knowledge of SQL Syntax

Constraints:

1. Usability:
 - a) No previous Training Time
 - b) On average the user should take 15 seconds to request patient's information.
2. Reliability:
 - a) Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
 - b) Availability – System should be available all the time except during server maintenance.
3. Performance:
 - a) System shall base query time on the amount of current data sources.
 - b) System shall be able to handle 10 requests per second.
4. Supportability:
 - a) The textbox and search button shall be correctly handled by any JVM.

Modification History:

Owner: Jandry Guerra

Initiation date: 09/26/10

Date last modified: 10/01/10

Use Case ID: CVM-M-HL-21 – Forgot Password.

Use Case Level: High Level

Details:

Actor: CVM-M User

Pre-conditions:

1. CVM-M User has logged onto the system.
2. The CVM-M interface is displayed.
3. Forgot password link was enabled

Description:

1. Use case begins when user clicks on the “forgot password” link in the login window (see Appendix F: Figure 22).
2. The system shall provide the user with a new page with the lost password request.
3. The user shall enter the following data: **user email** (see Appendix F: Figure 24).
4. The user clicks submit.
5. A new dialog is display asking the user a question.
6. The user enters the **answer** to a predefine question that was created with the account (see Appendix F: Figure 25).
7. The user shall then send the request by selecting the **send** button.
8. System shall confirm that the data that was entered is correct.
9. When the request is received the system shall generate a **new password**.
10. Use case ends when the new password is delivered to the user, and the user is notified.

Post-conditions:

1. The new password replaces the old password in the system.

Alternative Courses of Action:

1. In step D.5 if the data entered is incorrect, then the system shall not generate a new password and will notify the user which data entered was incorrect.

Extensions: None

Exceptions:

1. The “forgot password” link on the web page is not active.
2. After the user enters the required data the **send** button is not active.
3. The data entered may not be on database, and the system might crash.

Concurrent Uses: None

Related Uses Case:

CVM-M - HL - 23 – Create Login

Decision Support:

Frequency: On average 1 out of 30 users may request this feature.

Criticality: High. If the user forgets his password, the system allows the user to have access to his account.

Risk: Low. Implementing this use case requires a part of implementation of the GUI.

Constraints:

1. Usability:
 - a) No previous Training Time
 - b) On average the user shall take 3 minutes to complete the send request form.
 - c) User friendly.
2. Reliability:
 - a) Mean time to Failure – 5% failure for every twenty four hours of operation is acceptable.
 - b) Availability – System shall be available all the time except during server maintenance.
3. Performance:
 - a) Request shall be sent and saved within 7 seconds.
 - b) System shall be able to handle 50 request in 1 minute
4. Supportability:
 - a) The forgot password page shall be correctly handled by any JVM.

Modification History:

Owner: Luis Bautista.

Initiation date: 09/15/2010

Date last modified: 09/27/2010

Use Case ID: CVM-M-HL-22 – Logout

Use Case Level: High Level

Details:

Actor: CVM-M User

Pre-conditions:

4. CVM-M User has logged onto the system.
5. The CVM-M interface is displayed.
6. Logout link was enabled.

Description:

11. Use case begins when user clicks on the Logout link in the CVM-M interface (see Appendix F: Figure 26).
12. The system shall provide the user with a new interface with Logout confirmation dialog.
13. The user shall then send the request by selecting: **OK or CANCEL**.
14. System shall confirm that the request was sent.
15. Systems shall logout or disconnect the session if the request is received.
16. Use case ends when the user is logged out, and the user is notified.

Post-conditions:

1. The login page is displayed.

Relevant Requirements: None

Alternative Courses of Action:

2. In step D.3 if the user presses the cancel button, then the system shall not logout the user and shall go back to the CVM-M interface.

Exceptions:

4. The Logout link on the CVM-M interface is not enabled.
5. The Logout confirmation dialog won't pop up.
6. The **OK** button in the Logout confirmation dialog is not enabled.
7. The **CANCEL** button in the Logout confirmation dialog is not enabled.

Concurrent Uses: None.

Related Uses Case:

CVM-M-SL-17

Decision Support:

Frequency: This will depend on the amount of users that sign in. On average 500 users may request this feature daily.

Criticality: High. If the user does not want to continue the service, the system allows the user to delete his account.

Risk: Low. Implementing this use case requires a part of implementation of the GUI.

Constraints:

1. Usability:
 - a) No Previous Training Time required.
2. Reliability:
 - a) Mean time to Failure – 5% failures for every twenty four hours of operation is acceptable.
 - b) Availability – System should be available all the time except during server maintenance.
3. Performance:
 - a) Request shall be sent and confirmed within at least 10 seconds.
 - b) System shall be able to handle 100 requests in 1 minute.
4. Supportability:
 - a) The Logout Link shall be correctly handled by any JVM.

Modification History:

Owner: Luis Bautista.

Initiation date: 09/15/2010

Date last modified: 09/27/2010

Use Case ID: CVM-M - HL - 23 – Create Login

Use Case Level: High-level

Details:

Actor: CVM-M user

Pre-conditions:

1. The login window (see Appendix F: Figure 22) has been activated and displayed properly.

Description:

1. Use case begins when CVM-M user clicks on the **create account** link.
2. The create Login dialog interface is display (see Appendix F: Figure 23).
3. The CVM-M user enters the following account information data: **username, password,** select a **predefined question, question answer,** and **email.**
4. Use case ends when the CVM-M user clicks on the **create** button.

Relevant requirements: None.

Post-conditions:

1. The CMM-M user will be taken back to the login interface to submit his login information.
2. The number of accounts in the system is increased by one.

Alternative Courses of Action:

1. In step D.3 (step 3 of Description section) the user has the option to click on the cancel button, which will cause the create account interface to close.
2. In step D.4 if the submitted information is already in the system, they user will be notify that the user is already in the system.

Extensions: None.

Exceptions:

1. After the user enters the required data the **create** button is not active.

Related Uses Case:

CVM-M - HL - 02 – Login

Concurrent Uses: None.

Decision Support:

Frequency: On average, 10 accounts are created daily.

Criticality: High. Allows CVM-M users to create an account in order to log on the system

Risk: Low. Implementing this use case employs basic java code.

Constraints:

1. Usability:
 - a) No previous Training Time
 - b) On average the user shall take 20 seconds to create an account.
2. Reliability:
 - a) Mean time to Failure – 10% failures for every twenty four hours of operation is acceptable.
 - b) Availability – System shall be available all the time except during server maintenance.
3. Performance:
 - a) User verification shall be done under 1 second.
 - b) System shall be able to handle 10 requests per second.
4. Supportability:
 - a) The create login page shall be correctly handled by any JVM.

Modification History:

Owner: Jandry Guerra

Initiation date: 09/26/10

Date last modified: 10/01/10

Use Case ID: **CVM-M-SL-09-Verify Patient**

Use Case Level: System level end to end

Details:

Actor: CVM-M User

Pre-conditions:

1. CVM-M user has logged onto system.
2. CVM-M user is in CVM-M Interface

Description:

1. Use case begins when CVM-M user clicks on **Search** button (Refer to Use Case ID CVM-M-HL-12)
2. System shall check if the patient exist
3. Use case ends if system cannot confirm patient or if system can confirm patient.

Post-conditions:

1. If patient ID exist, system shall compile information.

Alternative Courses of Action:

1. In step 1 of description the user has the option to search for patient ID by clicking the search button or to logout by clicking the logout button.

Extensions: None.

Exceptions:

1. After user types ID and click the search button nothing happens.

Concurrent Uses: None.

Related Use Cases: CVM-M-HL-07, and CVM-M_SL-08

Decision Support:

Frequency: On average of 100 request are made daily by user.

Criticality: High. User needs to know if patient has any medical history available.

Risk: High. Without user ID information can't be accessed.

Constraints:

1. Usability:
 - a) Training is not require to use this feature.
2. Reliability:
 - a) System should work corrently at least 95% of the time in a 24 hours expand.
 - b) System should be available 24 hours a day exept when is being maintained.
3. Performance:
 - a) Request should be handle in less than 5 seconds
 - b) System should be able to handle on average of 100 request per day.
4. Supportablitiy: None

Modification History:

Owner: Eduardo Flores

Initiation date: October 1, 2010

Date last modified: October 1, 2010

Use Case ID: **CMV-M-SL-10-Compiling Information**

Use Case Level: System level end to end

Details:

Actor: CVM-M User

Pre-conditions:

1. CVM-M user has logged onto system.
2. CVM-M user is in CVM-M Interface

Description:

1. Use case begins when CVM-M user clicks on **Search** button (Refer to Use Case ID CVM-M-HL-12)
2. System shall start to compile all data sources base on patient ID.
3. Use case ends when compile information is completely put in temporary storage.

Post-conditions:

1. System save information to file.

Alternative Courses of Action:

1. In step 2 of description the user has the option to search for ID by clicking search or logout by clicking logout button.

Extensions: CVM-M-SL-15

Exceptions:

1. Information does not get displayed when progress is done.
2. System doesn't have access to data sources.

Concurrent Uses: CVM-M-SL-13, CVM-M-SL-14

Related Use Cases: CVM-M-SL-18, CVM-M-SL-11, CVM-M-SL-12, CVM-M-SL-14

Decision Support:

Frequency: On average of 100 request are made daily by user.

Criticality: High. User needs to know if patient has any medical history available.

Risk: High. This implements data base searches and saving.

Constraints:

1. Usability:
 - a. Training is not require to use this feature
2. Reliabilty:
 - a. System should work corrently at least 95% of the time in a 24 hours expand.
 - b. System should be available 24 hours a day exept when is being maintained.
3. Performance:
 - a. Request should be handle in less than 5 minutes
 - b. System should be able to handle on average of 100 request per day.
4. Supportablitiy:
 - a. The system should be able to handle database engine, data file such as documents, photos, video, and any window related file.

Modification History:

Owner: Eduardo Flores

Initiation date: October 1, 2010

Date last modified: October 1, 2010

Use Case ID: **CVM-M-SL-13** **Temporary Storage**

Use Case Level: System level end to end

Details:

Actor: CVM-M user

Pre-conditions:

1. CVM-M user has logged onto system.
2. CVM-M user is in CVM-M Interface

Description:

1. Use case begins when CVM-M user clicks on **Search** button (Refer to Use Case ID CVM-M-HL-12)
2. System shall temporarily store information while is being compiled, categorize, and organize.
3. Use case ends when CVM-M user has disconnected from the system.

Relevant Requirements:

Standard I/O Commands

Post-conditions:

1. CVM-M user is out of the system.
2. Temporary data is no longer available

Alternative Courses of Action:

1. In description step 2, the system is retrieving data and structuring it.

Extensions:

None

Exceptions:

1. The storage system is inaccessible.
2. Storage location is out of space.

Concurrent Uses:

CVM-M-HL-11, CVM-M-SL-16, CVM-M-SL-12, CVM-M-SL-15, CVM-M-SL-14

Related Use Cases:

CVM-M-SL-11, CVM-M-SL-12, CVM-M-SL-22

Decision Support:

Frequency: On average 600 request are made daily by CVM-M user.

Criticality: High, the processes of the system cannot get done.

Risk: Medium, implementation of file system commands.

Constraints:

1. Usability:
 - a) Training is not require to use this feature
2. Reliability:
 - a) Meantime to failure: 2% failure for every 4 Hours of operation is acceptable.
 - b) System shall be available 24 hours a day except when is being maintained.
3. Performance:
 - a) System shall be able to handle 50 I/O's per second.
 - b) File access time shall be at least at the speed of a conventional network.
4. Supportability:
 - a) The transferring of files needs a file allocation system.

Modification History:

Owner: Ivan Olmos.

Initiation date: 09/29/2010

Date last modified: 10/04/2010

Use Case ID: CVM-M - SL - 14 - Display Progress

Use Case Level: System level end to end

Details:

Actor: CVM-M user

Pre-conditions:

1. The CVM-M user has successfully logged onto the system.
2. The CVM-M Interface has been displayed.

Description:

1. Use case begins when CVM-M user clicks on the **Search** button (see CVM-M – HL – 7 – Query Information).
2. The system displays the progress bar on the interface (see Appendix F: Figure 19)
3. Use case ends when the patient's information is compiled and displayed. (see CVM-M – SL – 6 – Display Information, CVM-M – SL – 10 – Compile Information)

Relevant requirements:

None.

Post-conditions:

1. The information compiled is displayed.

Alternative Courses of Action:

1. In step D.2 (step 2 of Description section) if the patient is not found, they progress bar will not be display and the user will be notify that the patient does not exist.

Extensions: None.

Exceptions:

2. The progress bar is not visible.

Related Uses Case:

CVM-M – HL – 07 – Query Information

CVM-M – SL – 10 – Compile Information

Concurrent Uses:

CVM-M – HL – 11, CVM-M-SL-16, CVM-M-SL-12, CVM-M-SL-15, CVM-M-SL-13

Decision Support:

Frequency: On average, the progress bar will be display 600 times daily.

Criticality: Medium. Gives users a time estimate of how long it will take.

Risk: Low. Implementing this use case employs basic java code.

Constraints:

1. Usability:
 - a. No previous Training Time
2. Reliability:
 - a. Mean time to Failure – 10% failures for every twenty four hours of operation is acceptable.
 - b. Availability – System should be available all the time except during server maintenance.
3. Performance:
 - a. System should be able to handle 50 requests per hour.
4. Supportability:
 - a. The Display Progress windows shall be correctly handled by any JVM.

Modification History:

Owner: Jandry Guerra

Initiation date: 10/01/10

Date last modified: 10/01/10

Use Case ID: **CVM-M-SL-15-Compiling and Sorting Exceptions**

Use Case Level: System level end to end

Details:

Actor: CVM-M User

Pre-conditions:

1. The CVM-M user has successfully logged onto the system.
2. The CVM-M Interface has been displayed.

Description:

1. Use case begins when CVM-M user clicks on **Search** button (Refer to Use Case ID CVM-M-HL-12)
2. System shall report any exceptions that are found while compiling, categorizing and sorting the information.
3. Use case end when data errors is thrown or compile data is displayed.

Post-conditions:

1. If system throws an error, it should go back to search for ID page.
2. If system find ID, it should continue to compile.

Alternative Courses of Action:

None

Extensions:

None.

Exceptions:

1. Error does not display when exception is throwned.

Concurrent Uses:

CVM-M-SL-10, CVM-M-SL-11, CVM-M-SL-12, CVM-M-SL-13

Related Use Cases:

CVM-M-SL-10, CVM-M-SL-16

Decision Support:

Frequency: On average of 100 request are made daily by user.

Criticality: High. The purpose of system is to have data sources available to the user.

Risk: High. Accessing many different data sources

Constraints:

1. Usability:
 - a) Training is not require to use this feature.
2. Reliabilty:
 - a) System shall work corrently at least 95% of the time in a 24 hours expand.
 - b) System shall be available 24 hours a day exepct when is being maintained.
3. Performance:
 - a) System shall throw exceptions within 2 seconds after occuring.
4. Supportablitiy:
 - a) The Exception window shall be correctly handle by any JVM.

Modification History:

Owner: Eduardo Flores

Initiation date: October 1, 2010

Date last modified: October 1, 2010

Use Case ID: **CVM-M-SL-16** **Display Compile Information**

Use Case Level: System level end to end

Details:

Actor: CVM-M user

Pre-conditions:

1. CVM-M user has logged onto system.
2. CVM-M user is in CVM-M Interface

Description:

1. Use case begins when CVM-M user clicks on **Search** button (Refer to Use Case ID CVM-M-HL-12)
2. System shall display compile information to the CVM-M user by following an XML schema determine by the system.
3. Use case ends when CVM-M user has seen the information on the user interface (see Appendix F: Figure 29)

Relevant Requirements:

XML Markup Language

Post-conditions:

1. The information is been look at, by the CVM-M user.
2. The information is been display to CVM-M user

Alternative Courses of Action:

None

Extensions:

None

Exceptions:

1. The information to be display was not properly match to the XML schema

Concurrent Uses:

None

Related Use Cases:

CVM-M-SL-12

Decision Support:

Frequency: On average 100 request are made daily by CVM-M user.

Criticality: High, the task of the system cannot be finalized

Risk: Medium, implementing this use case requires the use of XML

Constraints:

1. Usability:
 - a) Training is not require to use this feature
 - b) A help file should be available to describe the information been displayed.
2. Reliability:
 - a) Meantime to failure: 10% failure for every 24 Hours of operation is acceptable.
 - b) System shall be available 24 hours a day except when is being maintained.
3. Performance:
 - a) System shall take no longer than 30 seconds to layout information being display to the CVM-M user.
4. Supportability:
 - a) The display of the information needs a control that can read and layout XML data

Modification History:

Owner: Ivan Olmos.

Initiation date: 09/29/2010

Date last modified: 10/04/2010

Session Time Out Use Case (CVM-M-SL-17)

Session Time Out is way to keep unauthorized user s out of the system when the user session in idle for a extensive amount of time

- Use Case Path – Unauthorized user takes control of user session.
- Use Case Path – Authorized user is no longer able to logon to system

Use Case: Session Time Out			
Use Case Path: Unauthorized user takes control of user session.			
Security Threat: The system continues to perform task after user leaves the station, meaning that an unauthorized user took over the session.			
Preconditions: 1) The user forgot to logoff system. 2) No mechanism exist to close session while is not been used.			
User Interactions	Misuser Interactions	System Interactions	System Action
		System shall have a system that checks for session inactivity	
User forgets to log off the system	The misuser, which is an unauthorized user takes over the session.		System shall prevent misuser from using a session that is not authorized
Post conditions: 1) The System shall have reacted to misuse interaction immediately, so that no unauthorized user could get access to the open session. 2) The System shall not have let unauthorized user take control of the session.			

Use Case: Session Time Out			
Use Case Path: Authorized user is no longer able to logon to system			
Security Threat: Authorized user credentials are changed, so that the user cannot get back into the system. System will be compromised given it a status of insecure			
Preconditions: 1) Authorized user credentials changed. 2) System is unaware of vulnerability.			
User Interactions	Misuser Interactions	System Interactions	System Action
		System shall have a system that checks for session inactivity	
User cannot logon with the proper credentials			System shall deactivated user account after a number of unsuccessful logins.
	Misuser, which is an unauthorized user takes over the session and changes the logon credentials.		System shall ask for old password before changing logon credentials
Post conditions: 1) The System shall have reacted to misuse interaction immediately, so that no unauthorized user could get access to the open session. 2) The System shall have notified user of change of password.			

10.3 Appendix C – User Interface designs

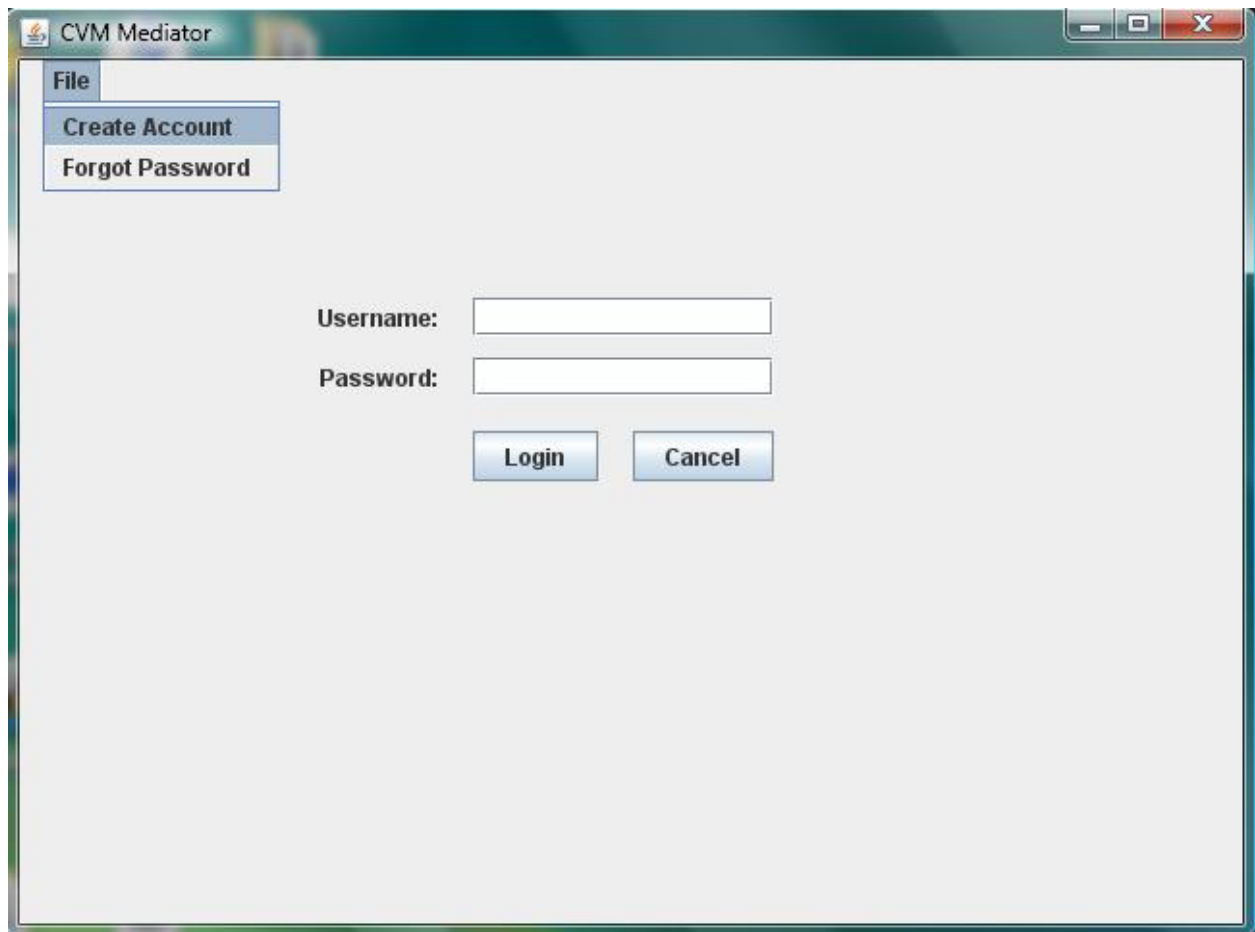
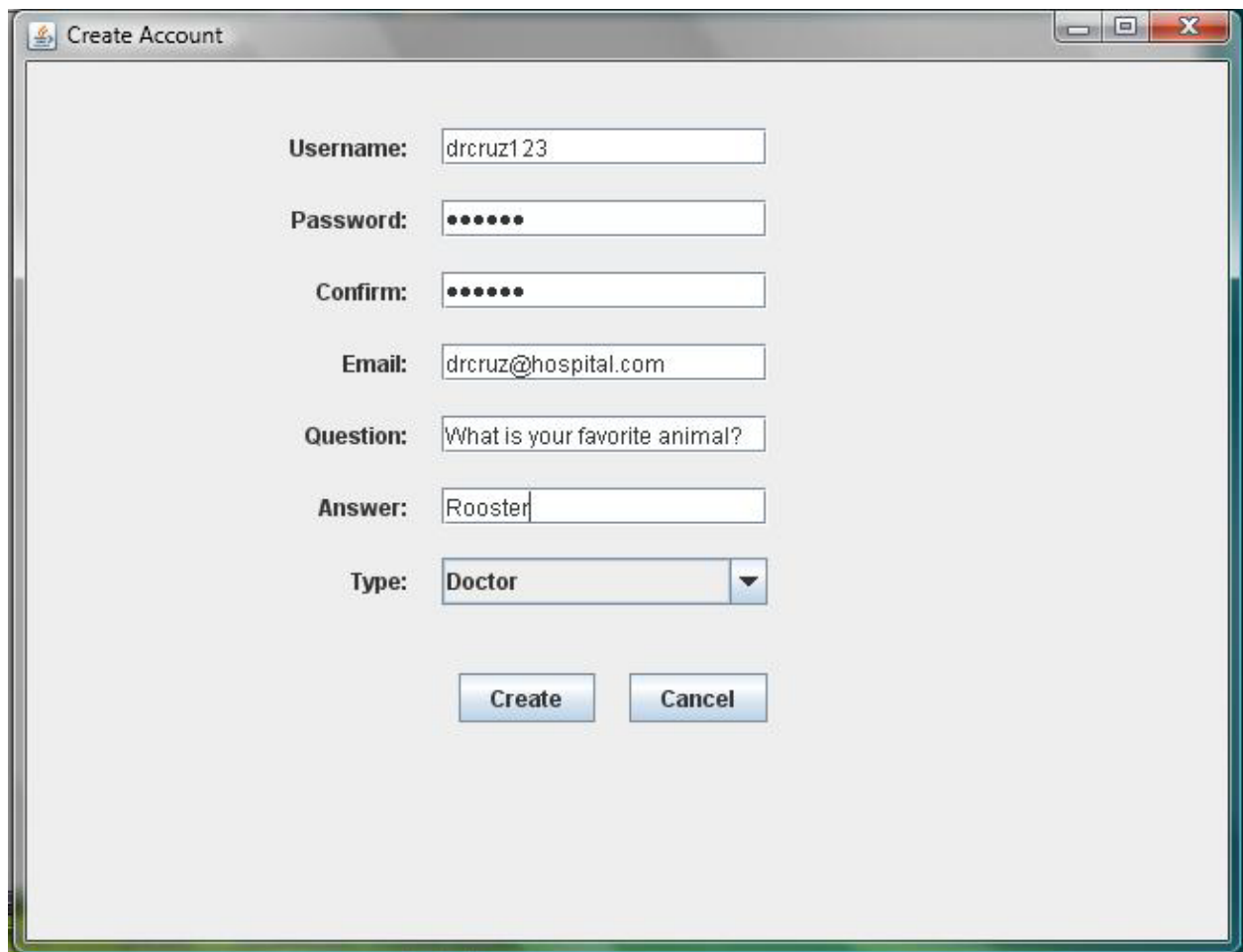


Figure 12: Login Screen



A screenshot of a 'Create Account' dialog box. The dialog has a title bar with a small icon and the text 'Create Account'. It contains several input fields for user registration: Username (drcruz123), Password (masked with dots), Confirm (masked with dots), Email (drcruz@hospital.com), Question (What is your favorite animal?), Answer (Rooster), and Type (a dropdown menu currently showing 'Doctor'). At the bottom are 'Create' and 'Cancel' buttons.

Create Account

Username: drcruz123

Password: •••••

Confirm: •••••

Email: drcruz@hospital.com

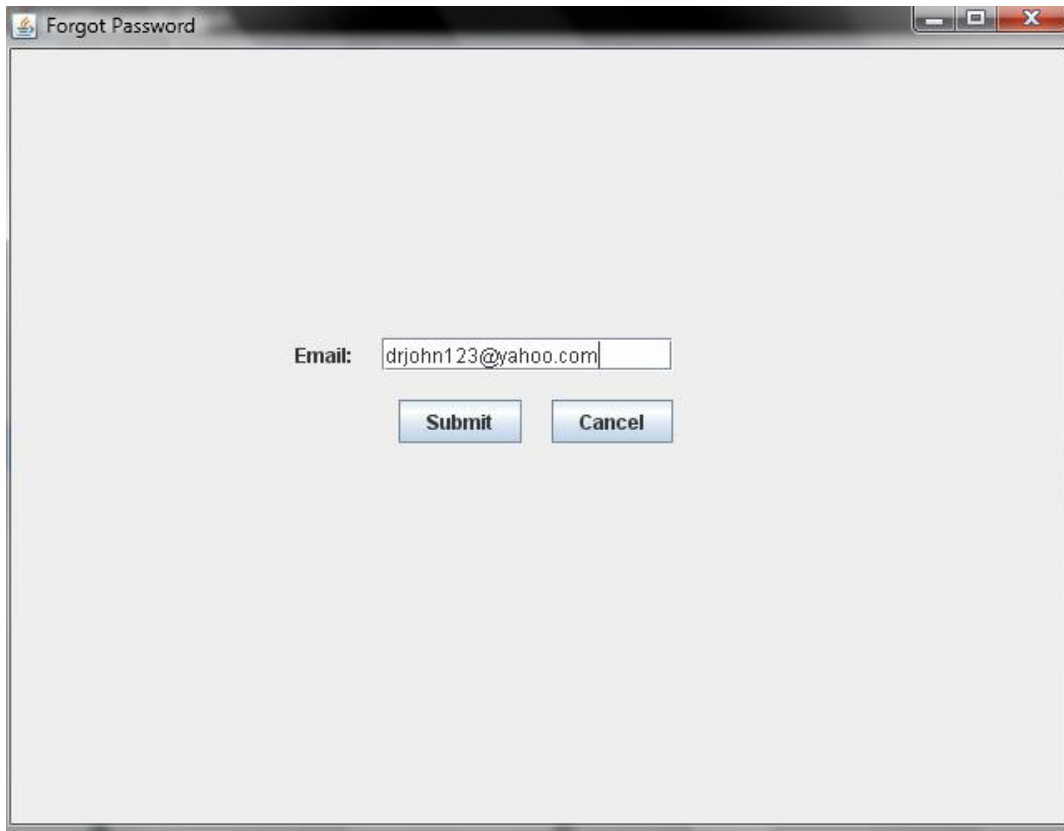
Question: What is your favorite animal?

Answer: Rooster

Type: Doctor ▼

Create **Cancel**

Figure 13: Create Account

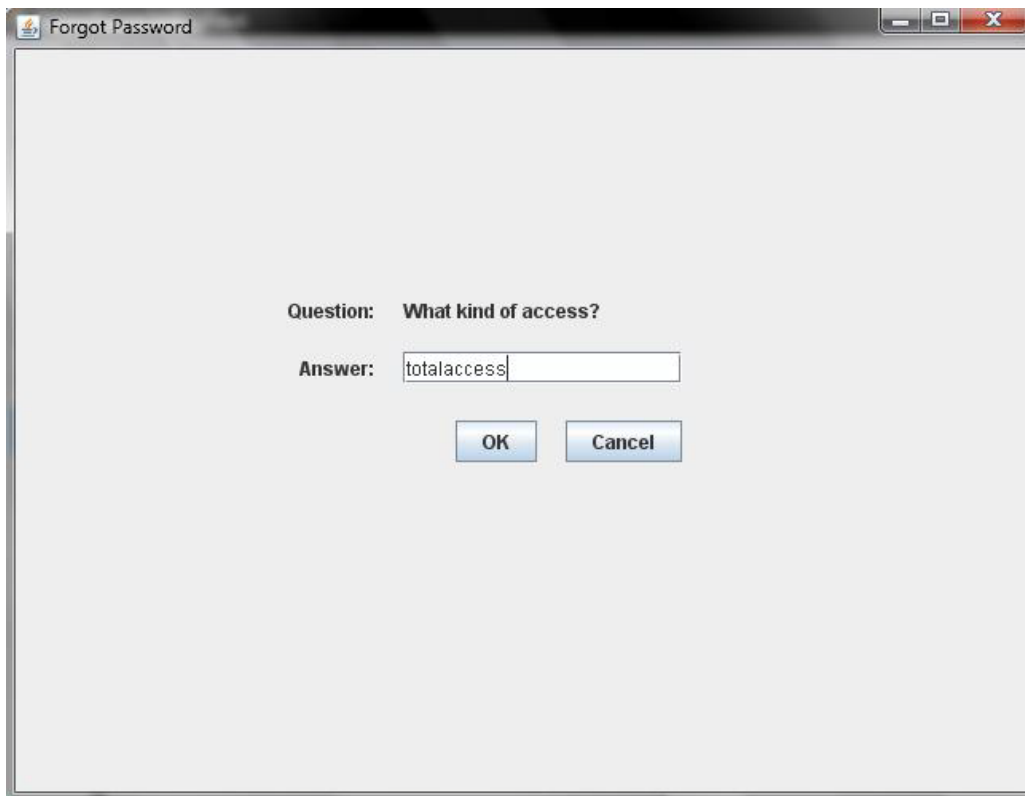


A screenshot of a Windows-style dialog box titled "Forgot Password". The dialog has a light gray background and a standard Windows window frame with minimize, maximize, and close buttons in the top right corner. In the center, there is a label "Email:" followed by a text input field containing the email address "drjohn123@yahoo.com". Below the input field are two buttons: "Submit" and "Cancel", both with a blue gradient and white text.

Forgot Password

Email:

Figure 14: Forgot Password Step 1



A screenshot of a Windows-style dialog box titled "Forgot Password". The dialog has a light gray background and a standard Windows window frame with minimize, maximize, and close buttons in the top right corner. In the center, there is a label "Question:" followed by the text "What kind of access?". Below this, there is a label "Answer:" followed by a text input field containing the text "totalaccess". At the bottom are two buttons: "OK" and "Cancel", both with a blue gradient and white text.

Forgot Password

Question: What kind of access?

Answer:

Figure 15: Forgot Password Step 2

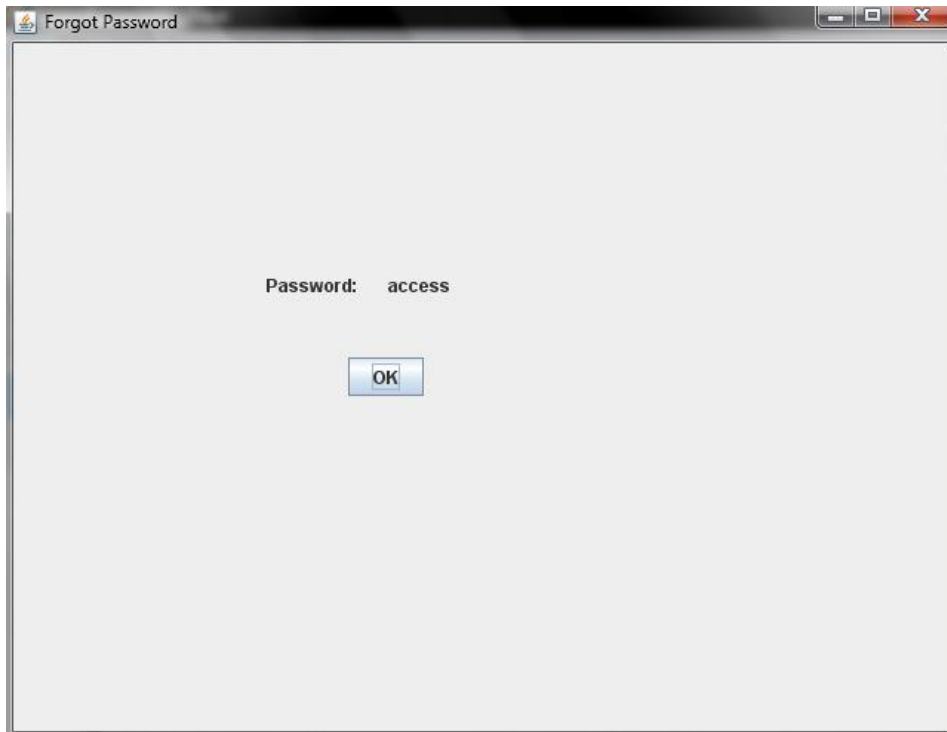


Figure 16: Forgot Password Step 3

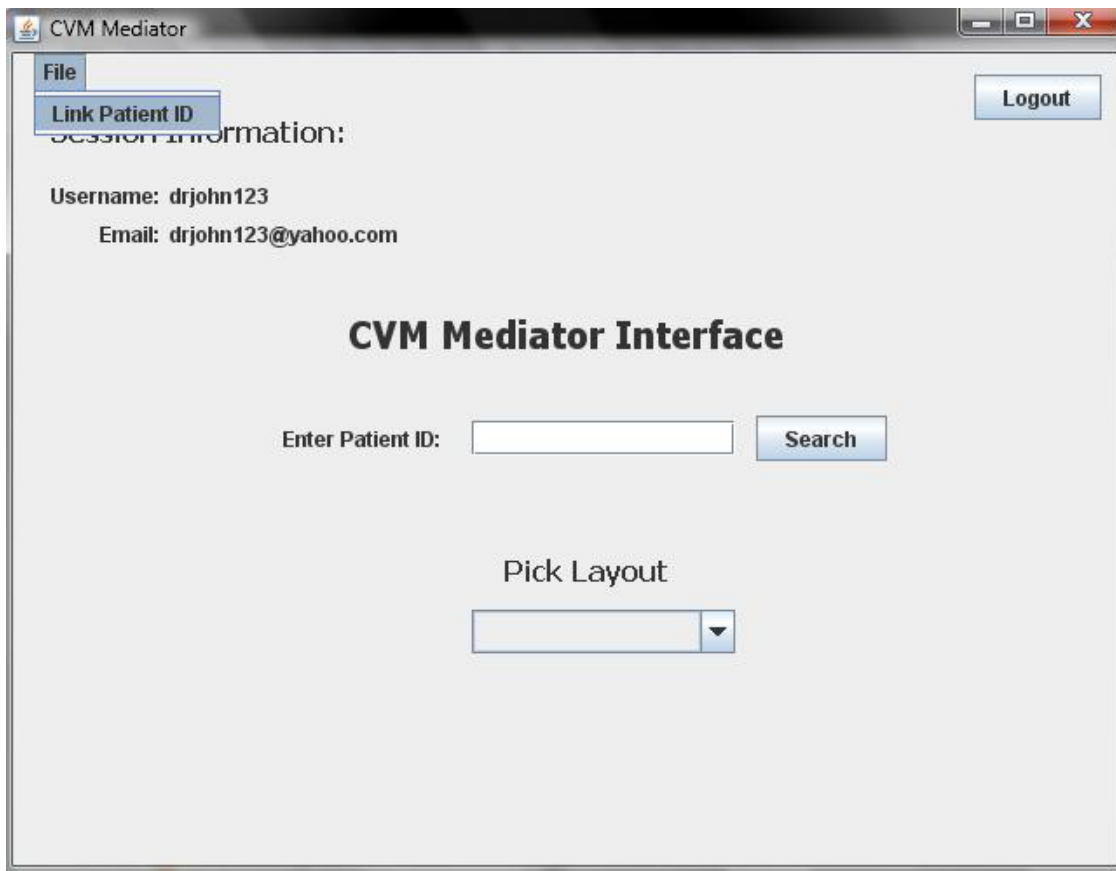


Figure 17: CVM-M Interface

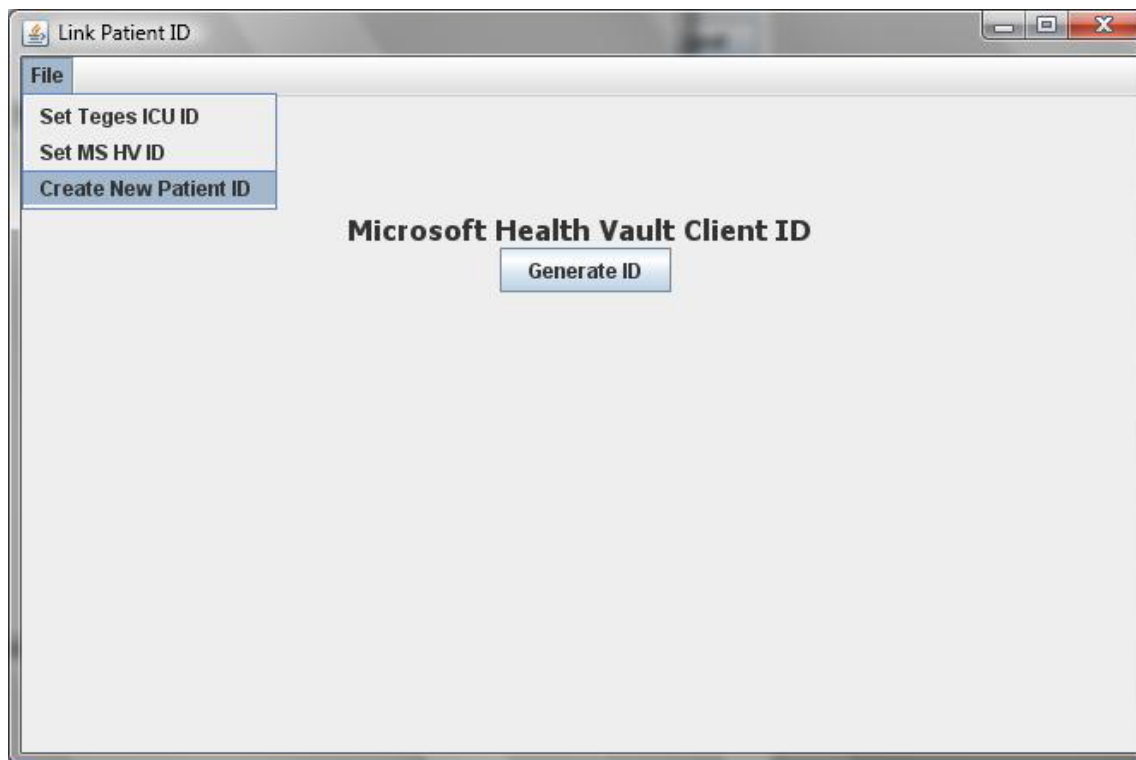


Figure 18: Create Patient ID



Figure 19: Create Patient ID Interface

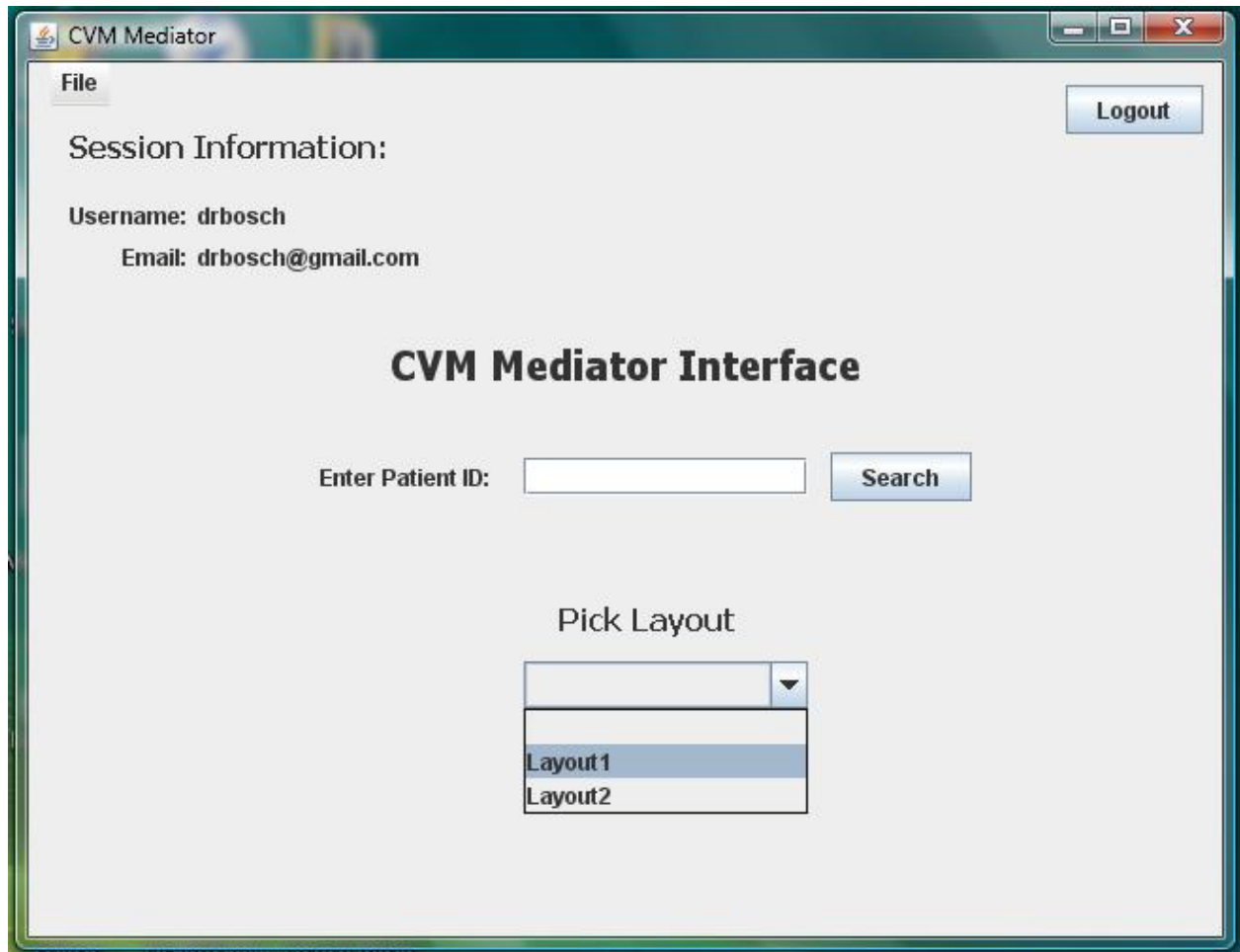


Figure 20: Layouts Selection

The screenshot shows a window titled "CVM Mediator" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a menu bar with a "File" option. In the top right corner, there is a "Logout" button. Below the menu bar, the text "Session Information:" is displayed. Underneath, the "Username: drjohn123" and "Email: drjohn123@yahoo.com" are listed. The main heading in the center is "CVM Mediator Interface". Below this, there is a section for "Enter Patient ID:" with a text input field containing "789" and a "Search" button to its right. Further down, there is a "Pick Layout" section with a dropdown menu currently showing "Layout1".

CVM Mediator

File

Logout

Session Information:

Username: drjohn123
Email: drjohn123@yahoo.com

CVM Mediator Interface

Enter Patient ID:

Pick Layout

▼

Figure 21: Patient Search

CVM Mediator

Health Summary



PATIENT INFORMATION			
NAME	ADDRESS	PHONE	EMAIL
Ivan tend	6789 NW 123 St	tel:3056789894	mailto:ivanox@mbhcc.org
Miami, FL			

ALLERGIES		
NAME	REACTIONS	STATUS
Penicillin	Hives	completed
Aspirin	Wheezing	completed
Codeine	Nausea	completed

Figure 22: Display Information

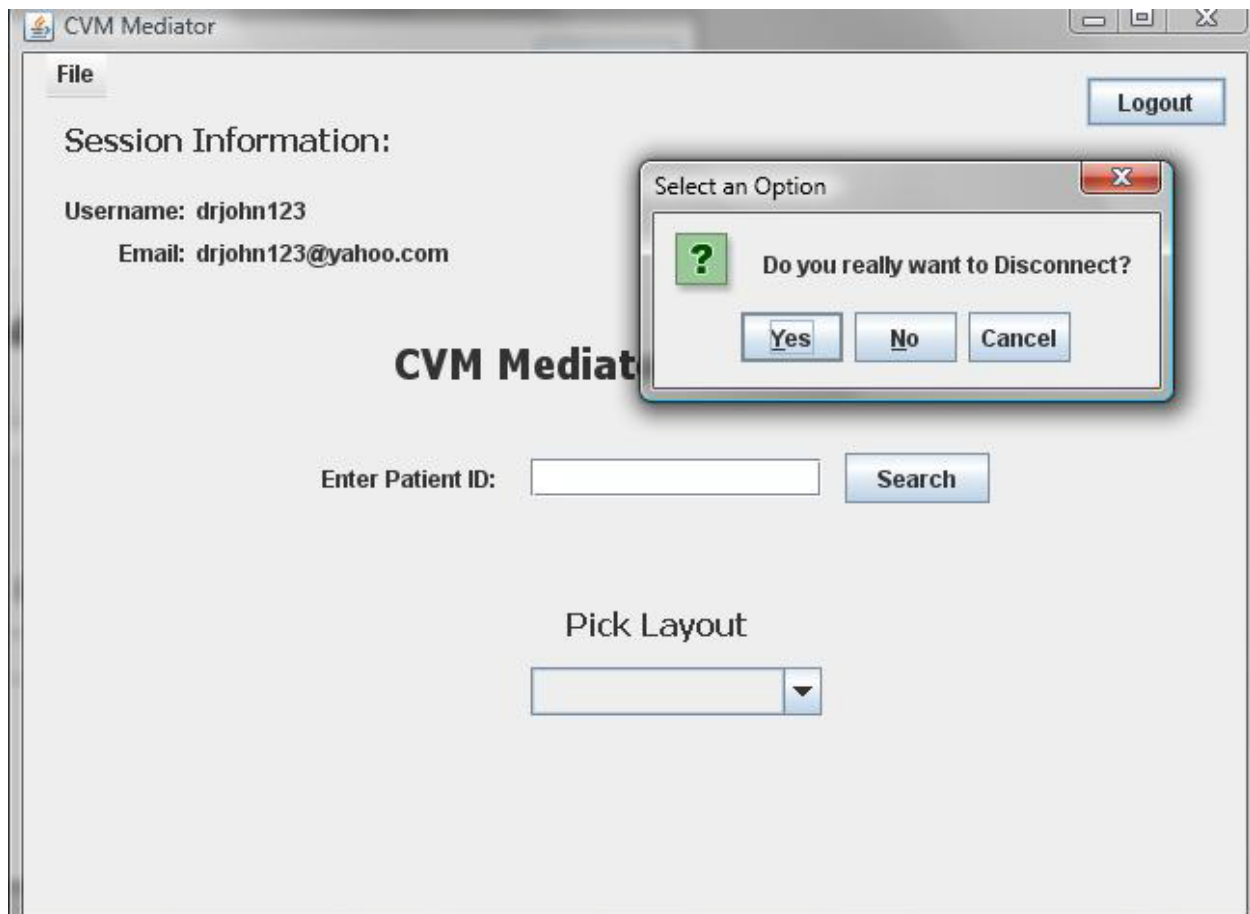


Figure 23: Logout

10.4 Appendix D – Analysis models (static and dynamic)

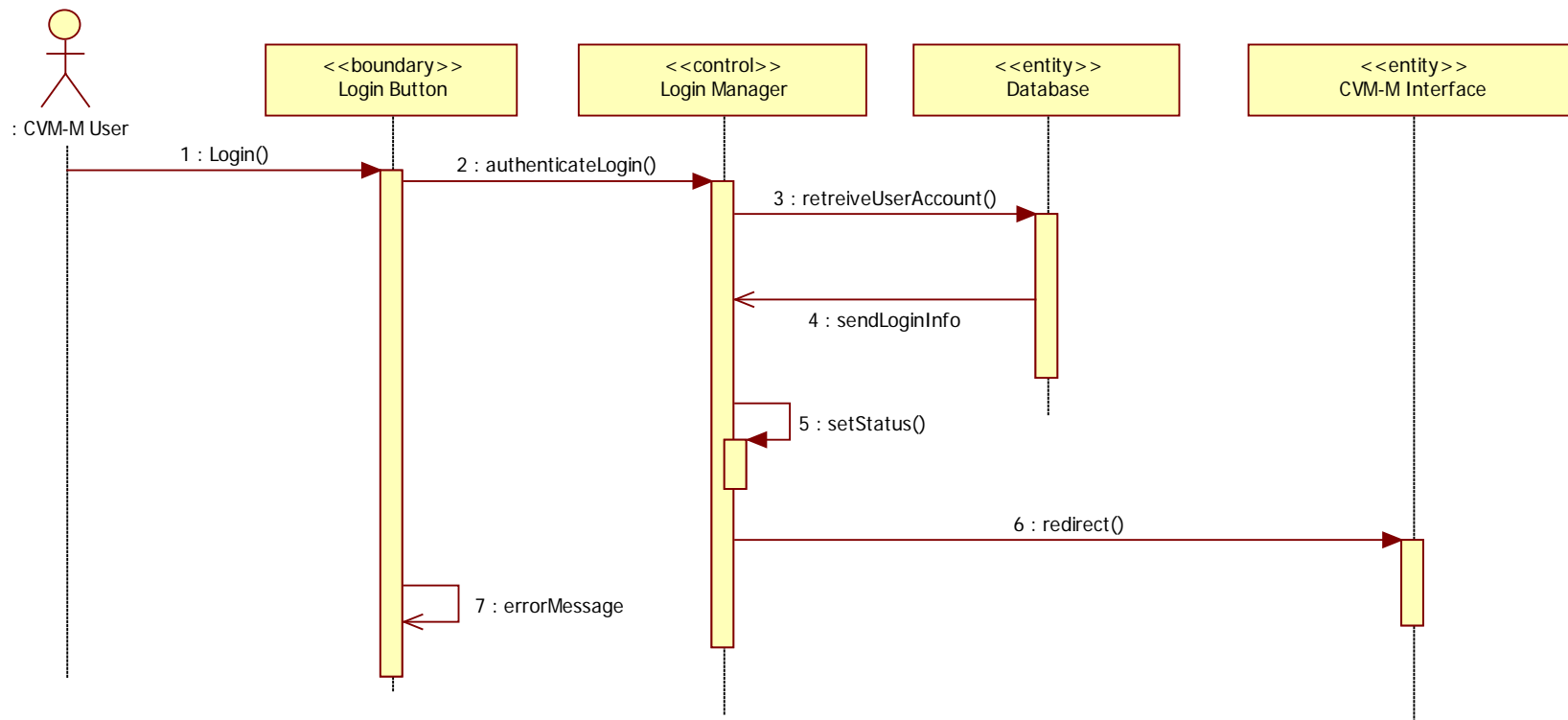


Figure 24 - Login Sequence Diagram

Description:

The Login Sequence Diagram shows the process that the system goes through when a user attempts to login onto the CVM-M Interface.

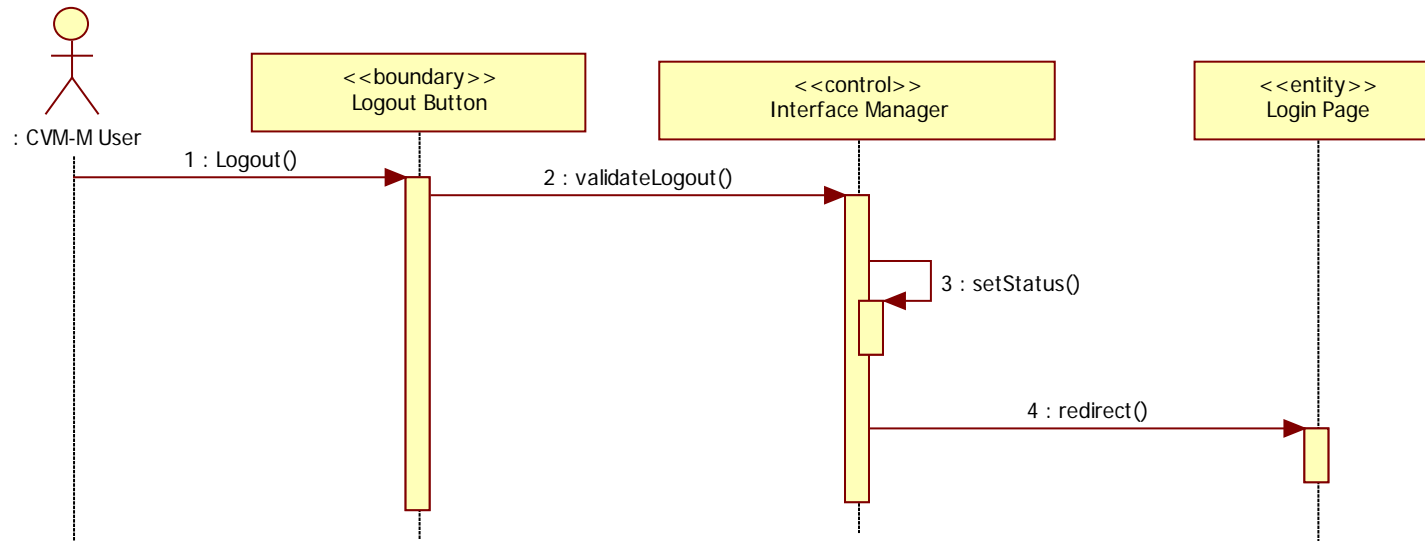


Figure 25 - Logout Sequence Diagram

Description:

The Logout Sequence Diagram shows the process that the system goes through when a user attempts to logout to the CVM-M Interface.

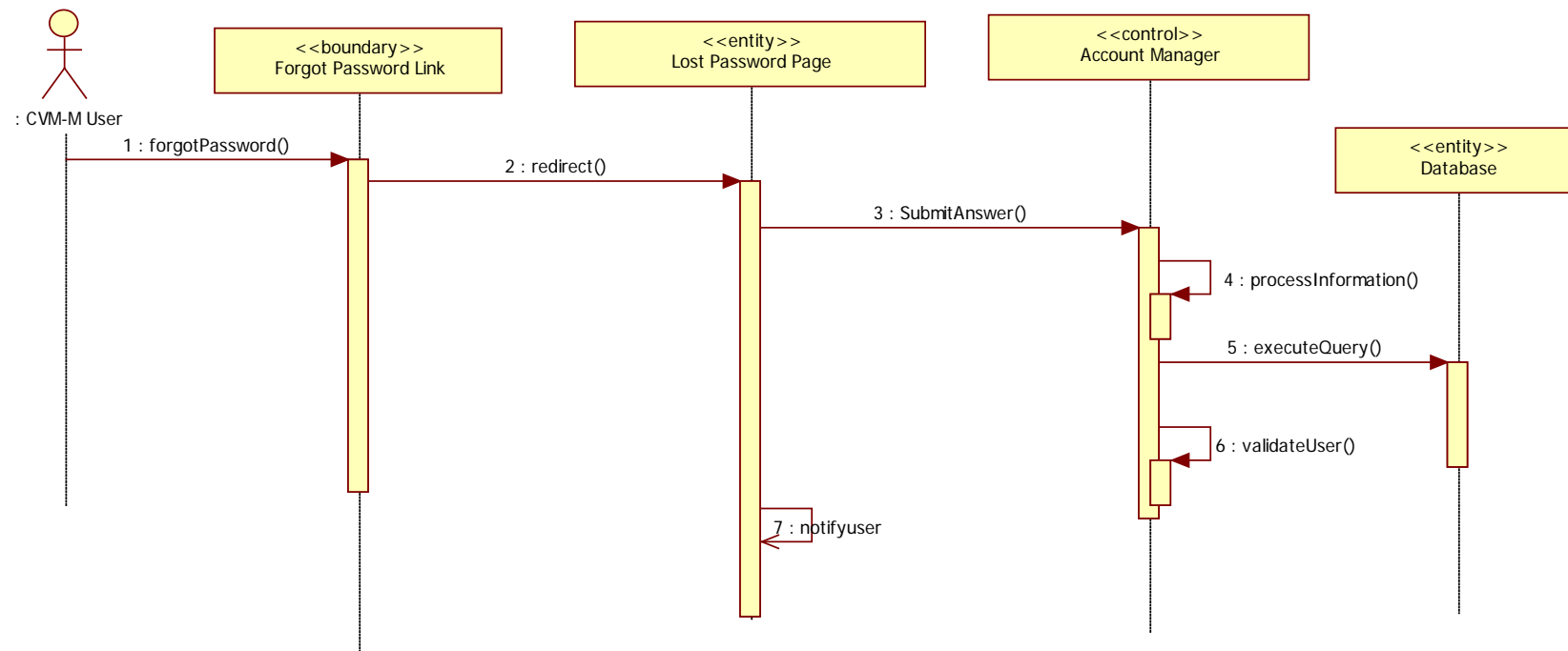


Figure 26 - Forgot Password Sequence Diagram

Description:

The Forgot Password Sequence Diagram shows the process that the system goes through when a user forgets his password and attempts to recover it.

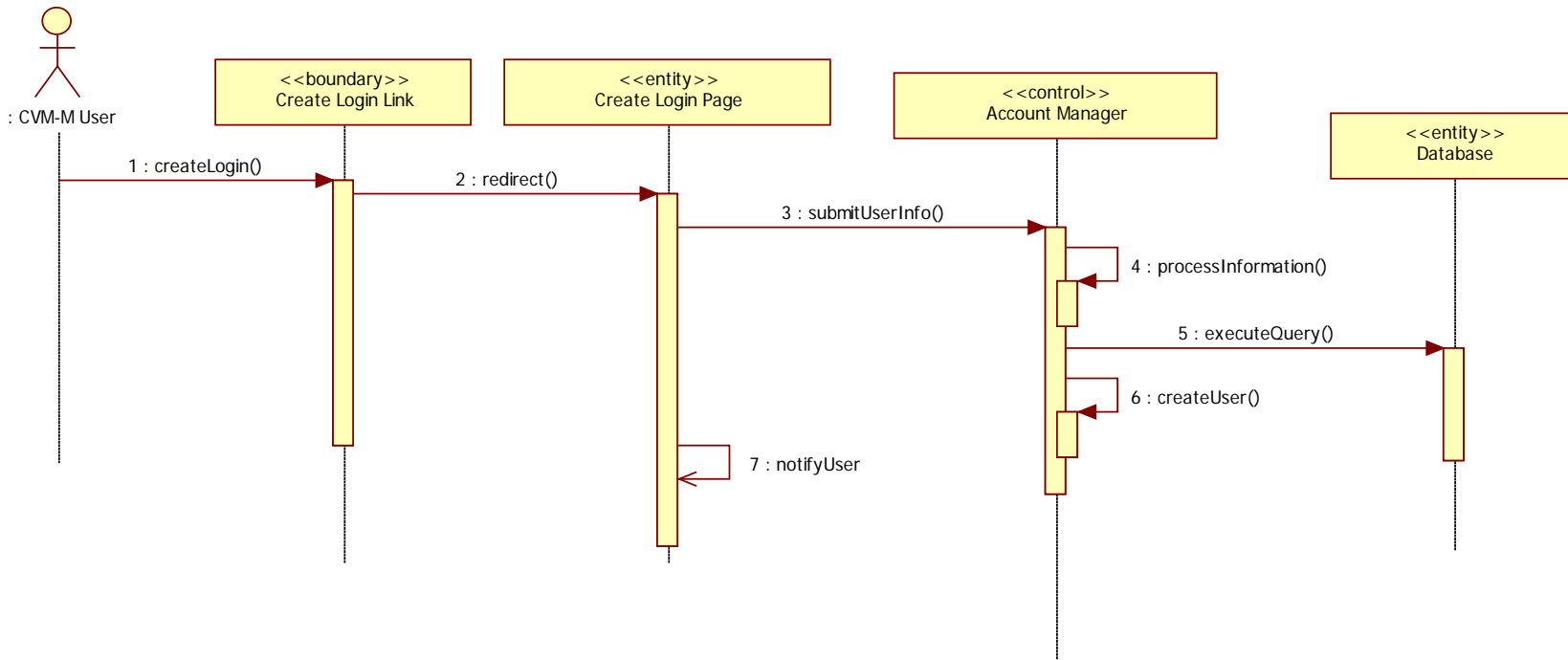


Figure 27 - Create Login Sequence Diagram

Description:

The Create Login Sequence Diagram shows the process that the system goes through when a new user attempts to create a login account.

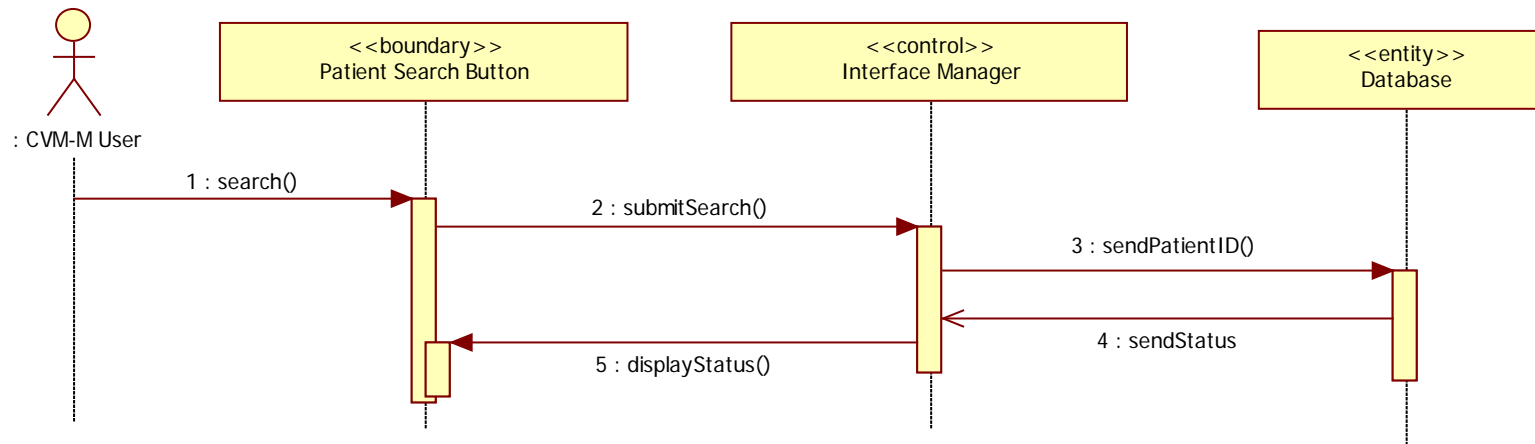


Figure 28 - Query Sequence Diagram

Description:
The Query Information Sequence Diagram shows the process that the system goes through when a new query is submitted.

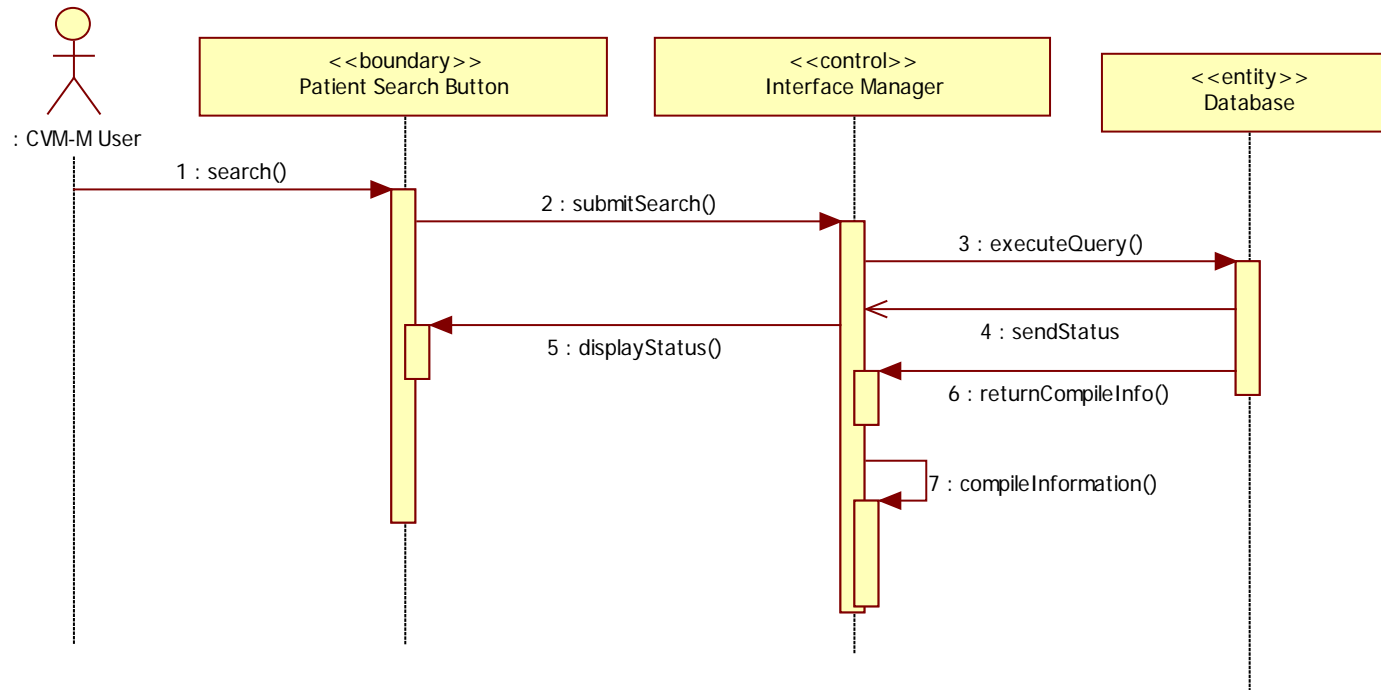


Figure 29 - Compile Information Sequence Diagram

Description:

The Compile Information Sequence Diagram shows the process that the system goes through when the information been query is compiled.

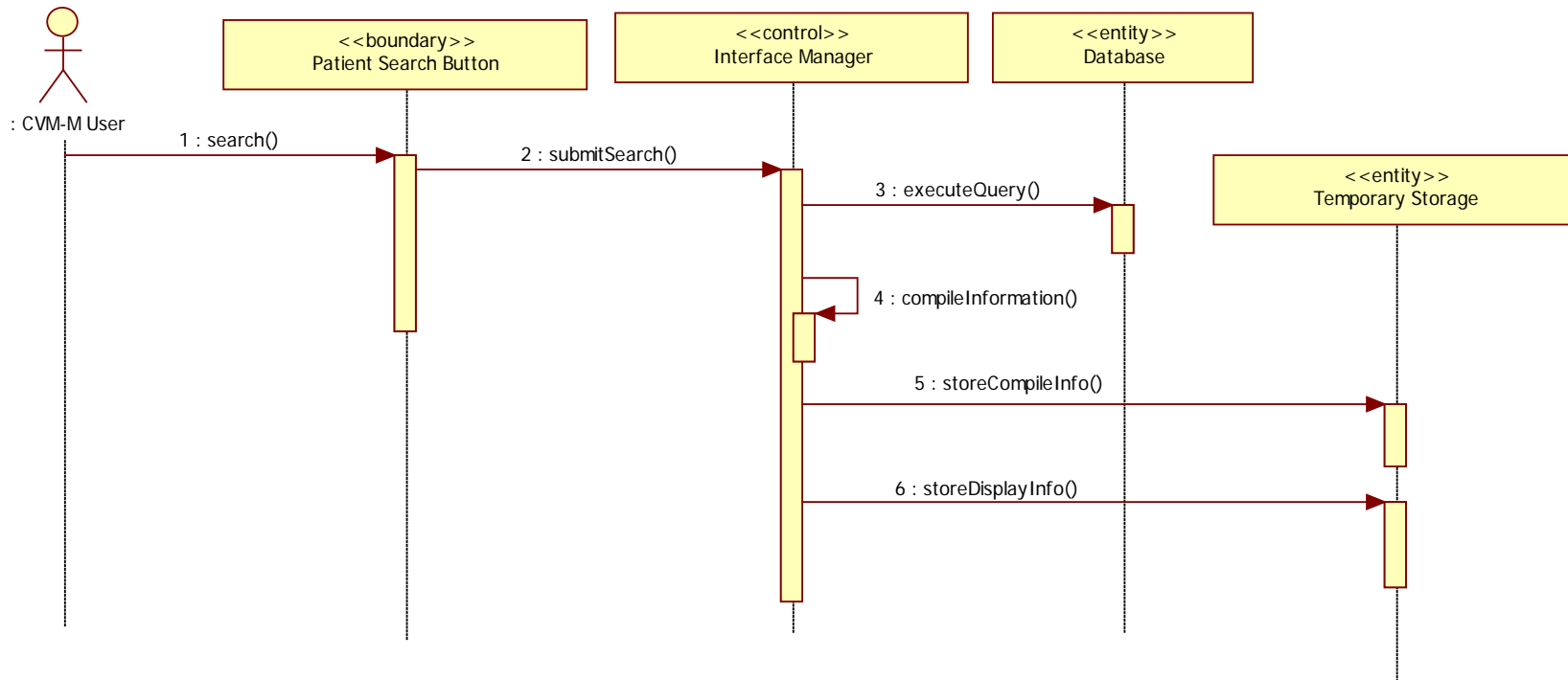


Figure 30 - Temporary Storage Sequence Diagram

Description:

The Temporary Storage Sequence Diagram shows the process that the system goes through when the information compiled is stored in a temporary folder for the user.

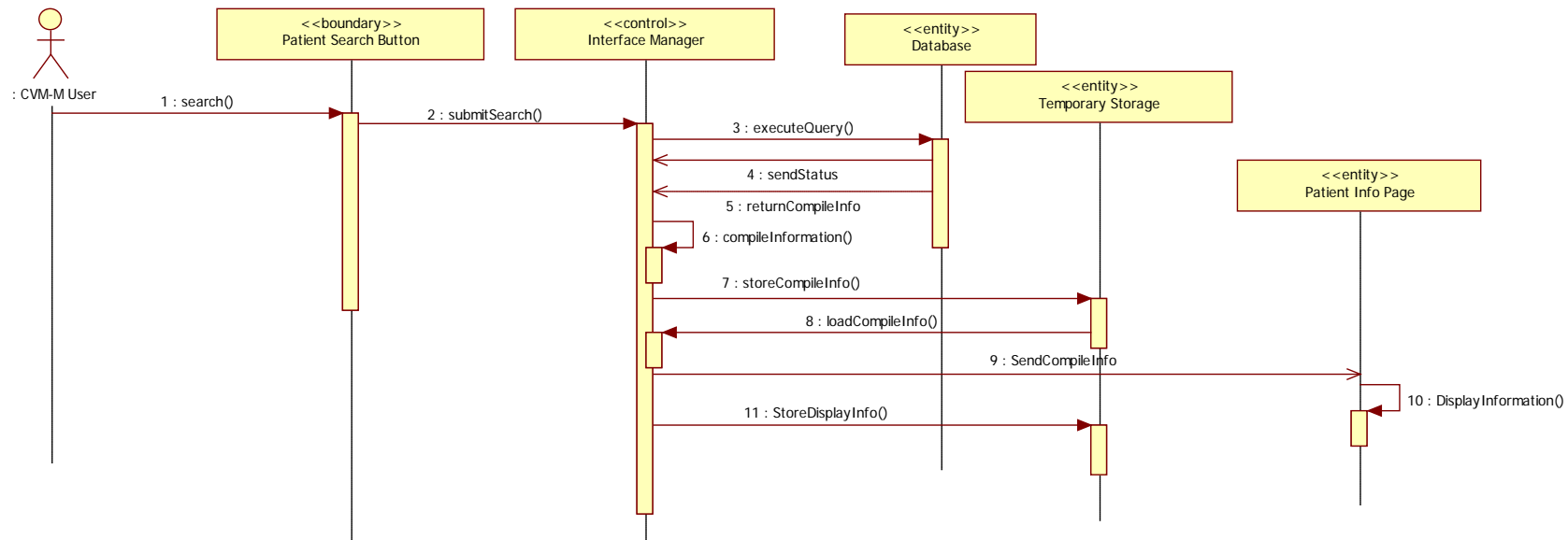


Figure 31 - Display Compile Information Sequence Diagram

Description:
 The Display Compile Information Sequence Diagram shows the process that the system goes through when the information compiled is displayed for the user in the CVM-M Interface.

10.5 Appendix E – Design models (static and dynamic)

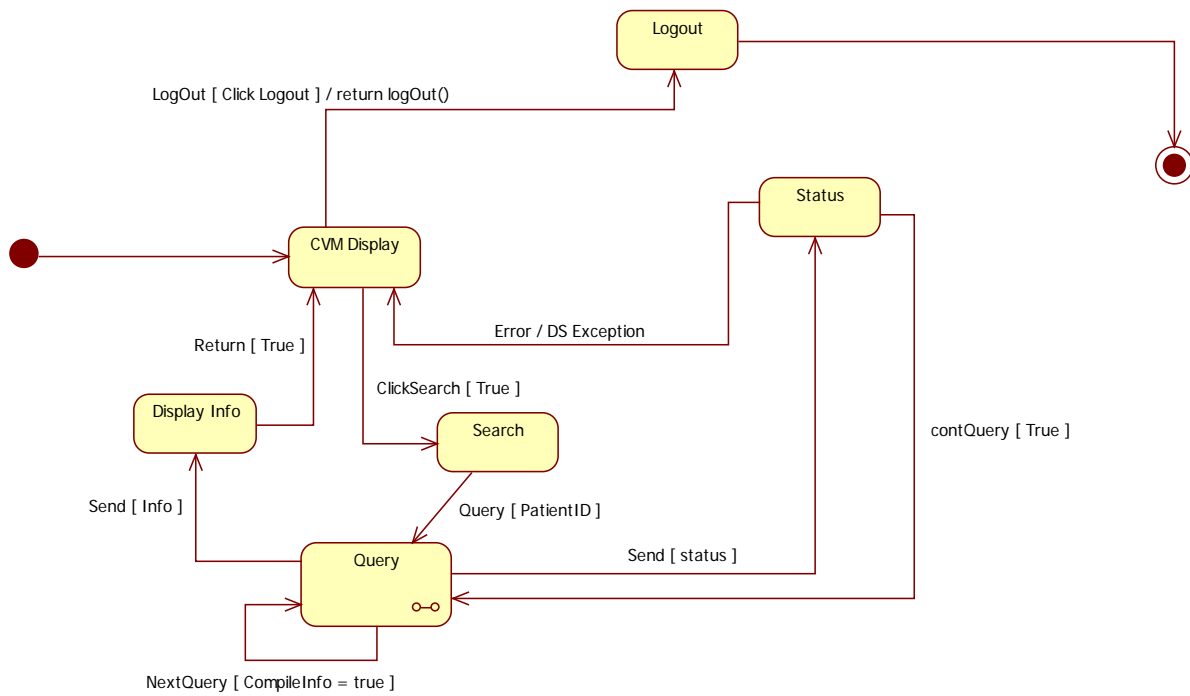


Figure 31: State Machine

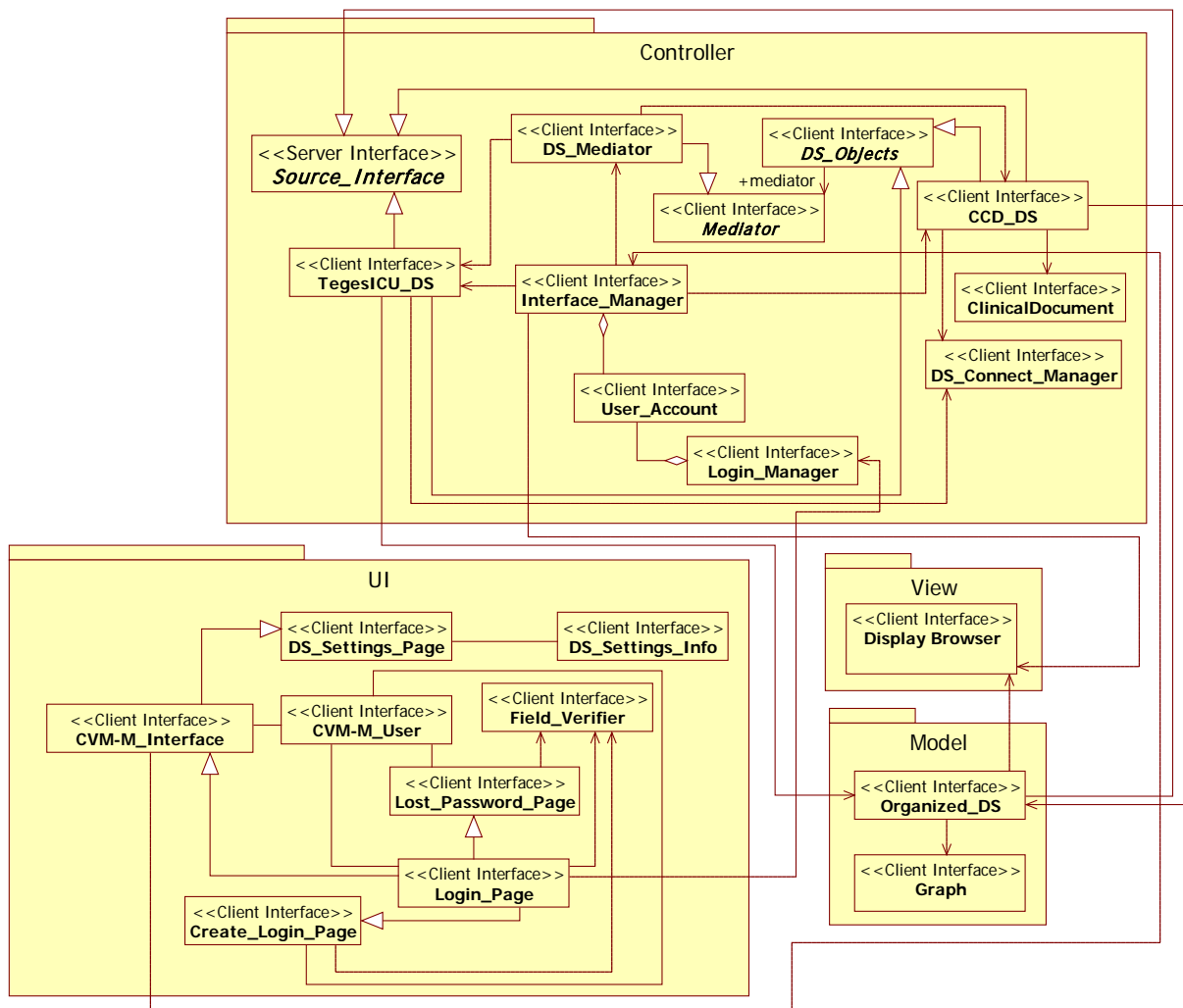


Figure 32: Minimal class diagram

10.6 Appendix F – Documented Class interfaces (code) and constraints.

```
package cvm.mediator.controller;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.JComboBox;
import javax.xml.bind.JAXBException;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

import cvm.mediator.model.Organizer_DS;
import cvm.mediator.view.HTMLDisplayer;

public class InterfaceManager
{
    private CVM_M_user cvmUser;
    private Organizer_DS organizer;
    private TegesICU_DS tegesICU_System;
    private ArrayList<CCD_DS> list_CCD;
    private ArrayList<CVM_M_patient> patients;
    private String model;

    public InterfaceManager() throws Exception
    {
        this.list_CCD = new ArrayList<CCD_DS>();
        this.patients = new ArrayList<CVM_M_patient>();
        retrievePatients();

        this.organizer = new Organizer_DS();
    }
}
```

Pre-Condition: System has started running and the patients have been taking from the administration database.

Invariants: No patient may have been created at the time of retrieval, probably database issues could have occur also that were ignored.

Post-Conditions: The list is populated on the dropdown list the window that will link the data source to the patient.

private void retrievePatients() throws Exception

```

{
    DS_Connect_CVM_Admin.retrievePatients(this.patients);
}

```

```

public void setCVMUser(CVM_M_user user)
{
    this.cvmUser = user;
}

```

Pre-Condition: User has logged into the system and patient being used must have being link to all data sources; were the information, should be pulled from.

Invariants: While this function is running the system might be place on hold for a couple of seconds, so no process could be executed at this time.

Post-Conditions: The health information for the patient has been compile and displayed

```

public boolean retrieveDS(String patientID, String layout)
    throws Exception
{
    CVM_M_patient patient;

    patient = this.findPatient(patientID);

    this.model = layout;

    if (patient == null)
    {
        return false;
    }

    MSHealth_DS ccd_DS = new
        MSHealth_DS(patient.getMSHealthID(), "MSHealth");

    if (patient.getMSHealthClientID() != null)
    {
        if (!MSHealth_DS.setAppID(patient.getMSHealthClientID()))
        {
            throw new Exception(
                "Microsoft HealthVault App Client ID could
                not be assign.");
        }
    }
    else
    {
        throw new Exception(
            "Patient must be tied to a Microsoft HealthVault
            App Client ID");
    }

    if (patient.getMSHealthID() == null)
    {
        throw new Exception(
            "Patient must be tied to a Microsoft HealthVault
            Person ID");
    }
}

```



```

        ccd_DS.connect();
        ccd_DS.transfer_CCD();
        this.list_CCD.add(ccd_DS);
        this.organizer.addCCD_DS(ccd_DS);
        System.out.println(
            this.list_CCD.get(0).CCD_Document.getId().getRoot());

        this.tegesICU_System = new TegesICU_DS(patient.getTegesID(),
            "jdbc:sqlserver://", "exserver.cs.fiu.edu", "1433",
            "TegesICU", "aahmad", "aahmad");
        this.tegesICU_System.connect();
        this.tegesICU_System.transferData();
        this.tegesICU_System.disconnect();
        this.organizer.setTegesICU(tegesICU_System);

        this.organizer.compileData(patient.getPatientID());

        //this.organizer.constructTreeTeges();

        return true;
    }

```

Pre-Condition: System has started running and the patients have been taking from the administration database.

Invariants: There could be database errors retrieving the items or patients could not exist, since they never were added.

Post-Conditions: System can now use this list of patients to perform any kind of search, retrieval or update on information, regarding the patients.

```

public ArrayList<CVM_M_patient> getPatients()
{
    return this.patients;
}

```

Pre-Condition: Patients have been retrieved and store in the system.

Invariants: No patients have been taking into the structure, so nothing can be look for.

Post-Conditions: A patient has been found and can be either use for processing or be a sign to not enter new patients.

```

private CVM_M_patient findPatient(String patientID)
{
    CVM_M_patient tempPatient;

    Iterator it = this.patients.iterator();

    while(it.hasNext())
    {
        tempPatient = (CVM_M_patient)it.next();
        if(tempPatient.getPatientID().compareTo(patientID) == 0)
        {
            return tempPatient;
        }
    }
}

```

```

        return null;
    }

```

Pre-Condition: Information about a patient has already been compile and been expanded into a XML file.

Invariants: Files IO and search errors can come up as the process is going.

Post-Conditions: An HTML file is displayed on a reliable HTML browser.

```

public void openGeneratedXML()
    throws TransformerException, IOException
{
    String filePath_HTML;

    filePath_HTML = convertToHTML(this.organizer.getXMLFile());

    FileOperations.copy("model/logo.jpg",
        System.getProperty("java.io.tmpdir") + "logo.jpg");
    FileOperations.copy("model/style.css",
        System.getProperty("java.io.tmpdir") + "style.css");
    FileOperations.copy("model/style2.css",
        System.getProperty("java.io.tmpdir") + "style2.css");

    HTMLDisplayer.displayURL("file://" + filePath_HTML);
}

```

Pre-Condition: Information about a patient has already been compile and been exported into an XML file.

Invariants: A file could not be open, or found. The conversion came up with errors so no HTML document could generate it. The HTML was not save properly.

Post-Conditions: The generated HTML file is saved in a folder.

```

private String convertToHTML(String filePath)
    throws FileNotFoundException, TransformerException
{
    String gen_HTML = System.getProperty("java.io.tmpdir") +
        "CVM_M_View.html";
    String styleSheet;
    TransformerFactory tFactory;
    Transformer transformer;

    if (this.model == null)
    {
        styleSheet = "model/Default.xsl";
    }
    else
    {
        styleSheet = "model/" + this.model + ".xsl";
    }

    tFactory = TransformerFactory.newInstance();
    transformer = tFactory.newTransformer(
        new StreamSource(styleSheet));
    transformer.transform(new StreamSource(filePath),
        new StreamResult(new FileOutputStream(gen_HTML)));
}

```

```

        System.out.println("*** The output is written in "+
            gen_HTML+" **");

        return gen_HTML;
    }

```

Pre-Condition: The patients have been retrieved from the administrative database and populated into a structure in the system, which are showing a dropdown, user has log in into the system.

Invariants: The TegesICU database is down, or there was errors retrieving the necessary database on the SQL syntax.

Post-Conditions: A dropdown list is populated in the window where the TegesICU data source is link to the patient.

```

public void retrievePatientList(JComboBox droplist)
    throws SQLException
{
    this.tegesICU_System = new TegesICU_DS("jdbc:sqlserver://",
        "exserver.cs.fiu.edu", "1433", "TegesICU", "aahmad",
        "aahmad");
    this.tegesICU_System.connect();
    this.tegesICU_System.retrievePatientIDs(droplist);
    this.tegesICU_System.disconnect();
}
}

```

10.7 Appendix G - Documented code for test drivers and stubs.

The code below is all the test drivers and stubs created to test the Subsystem.

```
package TestStubs;

import static org.junit.Assert.*;

import java.io.File;
import java.lang.reflect.Field;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import cvm.mediator.controller.CCD_DS;
import cvm.mediator.controller.TegesICU_DS;
import cvm.mediator.model.Graph;
import cvm.mediator.model.Organizer_DS;

public class OrginizerTest
{
    private String filePath_Gen_XML;
    Graph theGraph;
    CCD_DS CCD;
    JAXBContext jc;
    Unmarshaller u;
    Organizer_DS org;
    Organizer_DS org1;
    Organizer_DS org2;
    TegesICU_DS teges;

    @Before
    public void setUp() throws Exception
    {
        this.filePath_Gen_XML = System.getProperty("java.io.tmpdir")
            + "CCD_File_Created.xml";
        jc = JAXBContext.newInstance("org.hl7.v3");
        u = jc.createUnmarshaller();
        org = new Organizer_DS();
        org1 = new Organizer_DS();
        org2 = new Organizer_DS();

        teges = new TegesICU_DS("1932", "jdbc:sqlserver://",
"exserver.cs.fiu.edu", "1433", "TegesICU", "aahmad", "aahmad");
        teges.connect();
        teges.transferData();
        teges.disconnect();
        org1.setTegesICU(teges);

        CCD = new CCD_DS("123", "doctor",
"C:\\Users\\Owner\\Desktop\\HealthVaultCCD.xml");
        CCD.connect();
    }
}
```

```

        CCD.transfer_CCD();
        org.addCCD_DS(CCD);
        org1.addCCD_DS(CCD);
        File f = new File(filePath_Gen_XML);
        if(f.exists())
            f.delete();
    }

    @After
    public void tearDown() throws Exception
    {
        this.filePath_Gen_XML = null;
        theGraph = null;
        jc = null;
        u = null;
        CCD = null;
        org = null;
        org1 = null;
        org2 = null;
        teges = null;
    }

    @Test
    public void CompileDataSD1() throws Exception
    {
        org.compileData();
        File f = new File(filePath_Gen_XML);
        theGraph = getGraph();
        System.out.println(theGraph.getNumOfVertex());
        assertEquals("compiled", true, f.exists());
        if(f.exists())
            f.delete();
    }

    @Test
    public void compiledataSD2() throws Exception
    {
        org1.compileData();
        File f = new File(filePath_Gen_XML);
        theGraph = getGraph1();
        System.out.println(theGraph.getNumOfVertex());
        assertEquals("compiled", true, f.exists());
        if(f.exists())
            f.delete();
    }

    @Test
    public void compiledataRD1() throws Exception
    {
        org2.compileData();
        File f = new File(filePath_Gen_XML);
        theGraph = getGraph2();
        System.out.println(theGraph.getNumOfVertex());
        boolean check = false;
        if(theGraph.getNumOfVertex() == 15 && f.exists())
            check = true;
        assertEquals("compiled", true, check);
    }

```

```

    }

    public Graph getGraph() throws Exception {

        Class c = org.getClass();

        // get the reflected object
        Field field = c.getDeclaredField("theGraph");
        // set accessible true
        field.setAccessible(true);

        return (Graph) field.get(org);

    }

    public Graph getGraph1() throws Exception {

        Class c = org1.getClass();

        // get the reflected object
        Field field = c.getDeclaredField("theGraph");
        // set accessible true
        field.setAccessible(true);

        return (Graph) field.get(org1);

    }

    public Graph getGraph2() throws Exception {

        Class c = org2.getClass();

        // get the reflected object
        Field field = c.getDeclaredField("theGraph");
        // set accessible true
        field.setAccessible(true);

        return (Graph) field.get(org2);

    }

}

```

10.8 Appendix H – Diary of meeting and tasks for the entire semester.

Phase I

Team 6 Meeting #1		08-28-2010	
		3:00PM-4:00PM	
		ECS 212	
Meeting called by:	All Members	Type of meeting:	General Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Jandry Guerra		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra		
Please read:	Luis Bautista could not meet because he was out of the country.		
Please bring:	N/A		
Agenda item:	Project introduction	Presenter:	Peter Clarke
Discussion:			
Professor Clarke introduced us to the project and described some aspects of the project. We saw videos describing the functionalities and a prototype of the software. We then discussed some of the project requirements such as:			
<div>➤ Be able to handle multiple data sources</div> <div>➤ Be able to formulate XML base on Data sources</div> <div>➤ Display XML for multiple users</div> <div>➤ Base XML Display on predefined user’s layouts.</div>			
Conclusions:			
Jandry Guerra was decided to be the team leader for phase I. Group decided to have a meeting in order to plan and understand the project. General meetings will be every Thursday at 3:25pm on ECS 145/ECS 212.			
Agenda item:	Project planning and understanding	Presenter:	All Members
Tasks:			

- Read documentation found in cis.fiu.edu/cml.
- Formulate a problem statement

Team 6 Meeting #2		08-28-2010	
		4:00PM-5:30PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Jandry Guerra		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra		
Please read:	Luis Bautista could not meet because he was out of the country.		
Please bring:	N/A		
Agenda item:	Project planning and understanding	Presenter:	Ivan Olmos
Discussion:			
We mainly discussed details of the project in order to understand it more in depth. Ivan explained the Architecture for on-demand sharing of EMRs Figure found in the cis.fiu.edu/cml website. We then plan our meeting schedules.			
Conclusions:			
We concluded that meetings will usually be on Saturdays at 3:00pm or Fridays at 4:00pm. Note: subject to change			
Agenda item:	Database details	Presenter:	All Members
Tasks:			
Read about HL7 and DICOM, to get familiarize with what we are going to be working with.			

Team 6 Meeting #3		09-02-2010	
		3:00PM-4:00PM	
		ECS 212	
Meeting called by:	All Members	Type of meeting:	General Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Luis Batista		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Batista		
Please read:	N/A		
Please bring:	N/A		
Agenda item:	Database details	Presenter:	All Members
Discussion:	<p>Discussion was mainly about the Tegos database. Yali clarified some details about the databases and the mediator. The group then discussed about the documentation to be done, and each member was assigned some task.</p>		
Conclusions:	<p>Professor Clark requested the group to present the documentation of what we had done so far on 09/09/2010. Each member was assigned a task (draft by 09/02/2010).</p>		
Agenda item:	Documentation and Problem definition	Presenter:	All Members
Tasks:	<ul style="list-style-type: none"> ➤ Eduardo Flores: Alternative Solutions ➤ Ivan Flores: Constrains and Limitations ➤ Jandry Guerra: Alternative Solutions ➤ Luis Batista: Feasibility matrix 		

Team 6 Meeting #4		09-06-2010	
		2:30PM-6:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Luis Batista		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Batista		
Please read:	N/A		
Please bring:	A draft of the documentation that was assigned on previews meeting.		
Agenda item:	Documentation and Problem definition	Presenter:	All Members
Discussion:			
Each member presented the documentation they had done up-to-date. The group reviews the documentation and fixed any mistakes or added new information.			
Conclusions:			
The group decided to show documentation to Professor Clarke on 09/07/2010.			
Agenda item:	Meeting with Tom Gomez, Director of Program Management	Presenter:	All Members
Tasks:			
➤ Luis Batista: High Level Requirements			
➤ Jandry Guerra: Project Plan			

Team 6 Meeting #5		09-06-2010	
		6:00PM-7:30PM	
		ECS 212	
Meeting called by:	All Members	Type of meeting:	Presentation
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Luis Batista		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Batista		
Please read:	N/A		
Please bring:	N/A		
Agenda item:	Meeting with Tom Gomez, Director of Program Management	Presenter:	Tom Gomez
Discussion:			
The functionality of the CONNECT database was described and how it works behind the scenes. The communication standards that would be use for the CONNECT database were described as HL7 XML, which is CCD.			
Conclusions:			
The group learned about CONNECT, how it works and its functionalities.			
Agenda item:	Review documentation with Professor Clarke	Presenter:	All Members
Tasks:			
➤ Submit all documentation to be reviewed on 09/07/2010 once is done.			

Team 6 Meeting #6		09-07-2010	
		3:30PM-4:40PM	
		ECS 145	
Meeting called by:	All Members	Type of meeting:	Class Meeting
Facilitator:	All Members	Note taker:	All Members
Timekeeper:	Luis Batista		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Batista		
Please read:	N/A		
Please bring:	Jandry Guerra: Documentation draft		
Agenda item:	Review documentation with Professor Clarke	Presenter:	All Members
Discussion:			
The group showed the documentation to Professor Clarke. We discussed and clarify some of the mistakes in the document, and added new information.			
Conclusions:			
Group received feedback from Professor Clarke.			
Agenda item:	Review documentation with Professor Clarke	Presenter:	All Members
Tasks:			
➤ Each Member had to fix the document(due 09/09/2010)			

Team 6 Meeting #7		09-09-2010	
		3:30PM-4:40PM	
		ECS 145	
Meeting called by:	All Members	Type of meeting:	Class Meeting
Facilitator:	All Members	Note taker:	All Members
Timekeeper:	Luis Batista		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Batista		
Please read:	N/A		
Please bring:	Jandry Guerra: Documentation		
Agenda item:	Review documentation with Professor Clarke	Presenter:	All Members
Discussion:			
The group showed the documentation to Professor Clarke. We discussed and clarify some of the mistakes in the document, and added new information.			
Conclusions:			
Group received feedback from Professor Clarke.			
Agenda item:	Document Assembly	Presenter:	All Members
Tasks:			
➤ Submit all documentation to Google group.			

Team 6 Meeting #8		09-10-2010	
		3:40PM-8:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	All Members
Timekeeper:	Luis Batista		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Batista		
Please read:	N/A		
Please bring:	N/A		
Agenda item:	Document Assembly	Presenter:	All Members
Discussion:			
There was not much discussion during this meeting, it was mainly working together and getting the document together. We did some minor discussion but mainly about the documents syntax.			
Conclusions:			
The group completed the document for the first deliverable as well as the presentation.			
Agenda item:	Presentation	Presenter:	All Members
Tasks:			
➤ Prepare for the Presentation			

Phase II

Team 6 Meeting # 1		9-21-2010	
		7:00PM-9:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Luis Bautista		
Please read:			
Please bring:	N/A		
Agenda item:	Review Software Specification	Presenter:	All Members
Discussion:	<p>The group discussed the software specification and brainstorm on a more detail system requirements. There was also a minimal discussion on some ideas for security use cases.</p>		
Conclusions:	<p>The group wrote 25 system requirements for the CVM-M system.</p>		
Next Agenda item:	Identify Use Cases	Presenter:	All Members
Tasks:	<p>➤ Each team member had to think of 5 use cases including a security one for next meeting.</p>		

Team 6 Meeting # 2		9-23-2010	
		4:30PM-5:30PM	
		ECS 145/ JCCL	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:			
Agenda item:	Identify Use Cases	Presenter:	
Discussion:	<p>The group identifies 20 use cases base on the system specification and selected 10 of those to be implemented. Each member selected which use cases they wanted to write.</p>		
Conclusions:	<p>Use cases were assigned to all members of the team.</p>		
Next Agenda item:	Review Use Cases	Presenter:	All Members
Tasks:	<p>Each member had to write 5 uses cases to be done by next meeting, were they are going to be review. Eduardo: Write scenarios</p>		

Team 6 Meeting # 3		9-28-2010	
		4:30PM-5:30PM	
		ECS 145/ JCCL	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	5 use cases including a security use case		
Agenda item:	Review Use Cases	Presenter:	
Discussion:			
The group reviewed all the use cases. We discussed about some ambiguity and corrected them. Corrected some of the none-functional requirements to make them consistent. We also discussed about StarUML and assigned some task to each member.			
Conclusions:			
Use cases were fixed, and we went over StarUML			
Next Agenda item:	Dynamic and Static Models	Presenter:	All Members
Tasks:			
Ivan: Static Models- Object diagrams Jandry: Use case model/ Static Models Luis: Dynamic models			

Team 6 Meeting # 4		9-29-2010	
		7:30PM-9:40PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Draft of the dynamic and Static Models		
Agenda item:	Dynamic and Static Models	Presenter:	
Discussion:	<p>The discussion was mainly about the relationships between the use cases, and working on the sequence diagrams and object diagram. We helped and revised each others to complete these diagrams. Group was unable to finish all the diagrams; therefore another meeting was requested.</p>		
Conclusions:	<p>Grouped finished some of the diagrams and in the next meeting group will work on the remaining ones.</p>		
Next Agenda item:	Dynamic and Static Models Continuation	Presenter:	All Members
Tasks:	<p>Eduardo: Sequence diagrams Ivan: Object diagram Jandry: Object diagram Luis: Sequence diagrams</p>		

Team 6 Meeting # 5		10-02-2010	
		7:30PM-8:30PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	N/A		
Agenda item:	Dynamic and Static Models Continuation	Presenter:	
Discussion:	<p>Group worked on the remaining diagrams to be done. We then revised all of the diagrams to make sure they were consistent and there were no mistakes. We then talked about the prototype of the system, and scheduled a meeting in order to implement it.</p>		
Conclusions:	<p>A prototype will be implemented on the next meeting.</p>		
Next Agenda item:	Prototype implementation	Presenter:	All Members
Tasks:	<p>Each member had to go over the use cases and specification to make sure we had a clear understanding of the system, in order to create the prototype.</p>		

Team 6 Meeting # 6		10-04-2010	
		7:30PM-11:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	N/A		
Agenda item:	Prototype implementation	Presenter:	
Discussion:	<p>We started working on the prototype and made sure it followed the specifications. Changes were made to the use cases because there were some mistakes and ambiguity.</p>		
Conclusions:	<p>A prototype was implemented, and each member was assigned some part of the document.</p>		
Next Agenda item:	Document Assembly	Presenter:	All Members
Tasks:	<p>Eduardo: Proposed System Requirements Ivan: Review Chapters 1 and 2. Jandry: Project Plan/ Section 4.1 Luis: Cost Estimate</p>		

Team 6 Meeting # 7		10-08-2010	
		7:00PM-12:30PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Ivan Olmos
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Assigned Documentation		
Agenda item:	Document Assembly	Presenter:	
Discussion:	<p>The group put together the document and reviewed each chapter to loop for mistakes. We also corrected some of the mistakes from the previous deliverable. We then created the presentation to be presented on October 12.</p>		
Conclusions:			
	Document and the presentation were completed		
Next Agenda item:	Presentation	Presenter:	All Members
Tasks:			
	Prepare for the presentation.		

Phase III

Team 6 Meeting # 1		10-21-2010	
		3:00PM-4:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Luis Bautista
Timekeeper:	Ivan Olmos		
Attendees:	Eduardo Flores, Ivan Olmos, Luis Bautista, Jandry Guerra		
Please read:			
Please bring:	N/A		
Agenda item:	Review Design Document	Presenter:	All Members
Discussion:	<p>We discuss the different thing we are doing in the document which are going to be different then the last one.</p>		
Conclusions:	<p>We assessed the amount of worked we need it to do.</p>		
Next Agenda item:	Brainstorm on the current document	Presenter:	All Members
Tasks:	<p>➤ Come up with ideas on how to approach it.</p>		

Team 6 Meeting # 2		10-22-2010	
		7:00PM-10:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Luis Bautista
Timekeeper:	Ivan Olmos		
Attendees:	Eduardo Flores, Ivan Olmos, Luis Bautista, Jandry Guerra		
Please read:			
Please bring:	N/A		
Agenda item:	Review Design Document	Presenter:	All Members
Discussion:	<p>The group discussed the Design Document and brainstorm on how to approach this document. We also broke down the work each of us was going to be in charge of.</p>		
Conclusions:	<p>We all add specific things to do in this document before next meeting.</p>		
Next Agenda item:	Status of where we are	Presenter:	All Members
Tasks:	<p>➤ Each member will tell the status of their work.</p>		

Team 6 Meeting # 3		10-28-2010	
		3:00PM-4:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Luis Bautista
Timekeeper:	Ivan Olmos		
Attendees:	Eduardo Flores, Ivan Olmos, Luis Bautista, Jandry Guerra		
Please read:			
Please bring:	N/A		
Agenda item:	Design Patterns	Presenter:	All Members
Discussion:	Discussion on current design pattern being use and if we can implemented.		
Conclusions:	We decided that a new pattern is need it.		
Next Agenda item:	Pick a new pattern	Presenter:	All Members
Tasks:	➤ Talk to Prof. Clark about it.		

Team 6 Meeting # 4		10-28-2010	
		7:00PM-10:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Luis Bautista
Timekeeper:	Ivan Olmos		
Attendees:	Eduardo Flores, Ivan Olmos, Luis Bautista, Jandry Guerra		
Please read:			
Please bring:	N/A		
Agenda item:	Design Patterns	Presenter:	All Members
Discussion:	A new design pattern was selected and started to implement it.		
Conclusions:	We decided to use Proxy design pattern and Mediator design patter		
Next Agenda item:	Continue to talk about progress	Presenter:	All Members
Tasks:	➤ Finish current individual assigned worked.		

Team 6 Meeting # 5		10-29-2010	
		7:00PM-11:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Luis Bautista
Timekeeper:	Ivan Olmos		
Attendees:	Eduardo Flores, Ivan Olmos, Luis Bautista, Jandry Guerra		
Please read:			
Please bring:	N/A		
Agenda item:	Design Document current status	Presenter:	All Members
Discussion:	We talked about the progress on the document and how we could go about doing some of the other work on the document.		
Conclusions:	We manage to finish some of the chapters and only need to polish them.		
Next Agenda item:	Continue to get statues on current work.	Presenter:	All Members
Tasks:	➤ Finish current individual assigned worked.		

Team 6 Meeting # 6		10-30-2010	
		6:00PM-10:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Luis Bautista
Timekeeper:	Ivan Olmos		
Attendees:	Eduardo Flores, Ivan Olmos, Luis Bautista, Jandry Guerra		
Please read:			
Please bring:	N/A		
Agenda item:	Design Document current status	Presenter:	All Members
Discussion:	The finishing of the document		
Conclusions:	Finish the major parts of it. Including chapter 2 and chapter 3.		
Next Agenda item:	Put the document together	Presenter:	All Members
Tasks:	➤ Go over finish document		

Phase IV

Team 6 Meeting # 1		11-05-2010	
		7:00PM-9:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Assigned Documentation		
Agenda item:	Document Assembly	Presenter:	
Discussion:	We discuss what was need it in order to accomplish this phase and how we can do it.		
Conclusions:	Each team member was assigned an specific task to complete before the next meeting.		
Next Agenda item:	Progress	Presenter:	All Members
Tasks:	We need to find where everyone stand on the moment with their current work.		

Team 6 Meeting # 2		11-09-2010	
		7:00PM-11:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Tops		
Agenda item:	Progress on current assigned tasks	Presenter:	
Discussion:	We discuss how the programming part of each task was going.		
Conclusions:	We concluded that each one of us was making progress.		
Next Agenda item:	Progress on the programming part.	Presenter:	All Members
Tasks:	We need to find out the progress on the current task of programming.		

Team 6 Meeting # 3		11-13-2010	
		5:00PM-9:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Tops		
Agenda item:	Programming progress	Presenter:	
Discussion:	<p>We went over everyone current assigned task and found out some of us where falling a bit behind and need it to step it up.</p>		
Conclusions:	<p>We help each other find a solution for things that were slowing us down and impeding our progress.</p>		
Next Agenda item:	Progress	Presenter:	All Members
Tasks:	<p>We need to have the major part of our programming project done.</p>		

Team 6 Meeting # 4		11-19-2010	
		7:00PM-11:30PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Top		
Agenda item:	Be done with major programming parts	Presenter:	
Discussion:	<p>We found out that most of us had finished the major programming parts but there still a lot of work to be done.</p>		
Conclusions:	<p>We assigned more work for those whom had finished their parts</p>		
Next Agenda item:	Progress on current tasks	Presenter:	All Members
Tasks:	<p>Finish the programming portion and start testing our project.</p>		

Team 6 Meeting # 5		11-26-2010	
		7:00PM-12:00AM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Tops		
Agenda item:	Progress on finishing Programming portion.	Presenter:	
Discussion:			
We came to a conclusion that we can start testing the part of the project that was finished.			
Conclusions:			
Part of the project that was finished was tested and passed.			
Next Agenda item:	Finish programming part completely and start fixing bugs, and complete testing.	Presenter:	All Members
Tasks:			
We need to finish the programming portion once and for all and fix the little and complete testing.			

Team 6 Meeting # 6		11-30-2010	
		7:00PM-11:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Tops		
Agenda item:	Finish programming part completely and start fixing bugs, and complete testing.	Presenter:	
Discussion:	We found out that we going to need a few more days to completely finish the programming portion.		
Conclusions:	We decided that we should be done by next meeting with the programming portion.		
Next Agenda item:	Finish testing and start to pull together the documentation.	Presenter:	All Members
Tasks:	We need to finish the testing and start on the documentation.		

Team 6 Meeting # 7		12-02-2010	
		6:00PM-9:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Tops		
Agenda item:	Finish testing and start to pull together the documentation.	Presenter:	
Discussion:			
We are finally done with the programming portion. We talked about the finishing of the testing and how to complete the documentation.			
Conclusions:			
We started to finish the testing part and assigned some team member part of the documentation while others test the software.			
Next Agenda item:	Have the testing completed and also look into finishing the documentation.	Presenter:	All Members
Tasks:			
We need to be done with testing and have the documentation almost finish.			

Team 6 Meeting # 8		12-03-2010	
		7:00PM-11:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Tops		
Agenda item:	Have the testing completed and also look into finishing the documentation.	Presenter:	
Discussion:	We talk about finishing the testing and also about the documentation progress.		
Conclusions:	We are almost done with testing and we gave it an extra day to finish. The documentation is almost done.		
Next Agenda item:	Finish documentation and get ready to record our presentation in case of malfunction during presentation.	Presenter:	All Members
Tasks:	We need to finish documentation and record the programming working under different scenarios.		

Team 6 Meeting # 9		12-05-2010	
		6:00PM-11:00PM	
		JCCL (LABORATORY)	
Meeting called by:	All Members	Type of meeting:	Team Meeting
Facilitator:	All Members	Note taker:	Jandry Guerra
Timekeeper:	Eduardo Flores		
Attendees:	Eduardo Flores, Ivan Olmos, Jandry Guerra, Luis Bautista		
Please read:			
Please bring:	Lap Tops		
Agenda item:	Finish documentation and get ready to record our presentation in case of malfunction during presentation.	Presenter:	
Discussion:	We talk about the finishing of documentation and we started to record our presentation.		
Conclusions:	We finished documentation and started recording or presentation in case of malfunction on Tuesday.		
Next Agenda item:	Start to practice presentation. Get ready for final presentation.	Presenter:	All Members
Tasks:	We need to practice our presentation.		