Stanford University

BMI211/CS271: Introduction to BioMedical Informatics: System Design

HIVdb+: Extending HIVdb functionality while securing data privacy

Final Project

Prepared for: Dr. Amar K. Das, M.D.,Ph.D.
Prepared by: Rashmi Raj, Zahid Khan, Genaro Hernandez Jr

March 17, 2006

Table of Contents

ABSTRACT **2**

BACKGROUND **2**

PROBLEM STATEMENT **3**

SYSTEM DESIGN **4**

DESIGN METHODOLOGY 4
RUP: A Brief Overview *4*
Why we selected RUP *5*
REQUIREMENTS ANALYSIS 5
Identifying the Users *5*
Functional Requirements *6*
Nonfunctional Requirements *6*
User Interface Requirements *7*
Workflow Analysis *7*
Use Case Analysis *7*
A Cognitive Perspective *8*
SYSTEM ARCHITECTURE 10
Remote Server Architecture *11*
SYSTEM EVALUATION 13
SYSTEM IMPLEMENTATION 14
Cost Analysis *14*

SUMMARY **15**

REFERENCES **16**

APPENDIX A **17**

ALTERNATIVE SYSTEM ARCHITECTURE 17

APPENDIX B **18**

DATA MODEL [10] 18

APPENDIX C **19**

USER INTERFACE DESIGN CONCEPT 19

APPENDIX D **20**

USER FLOW (FOR 'LAB TECHNICIAN' ROLE) 20
Screen 2.2.2: Enter Tagged Sequence Data *21*
Typical Result of Genotype Interpretation query *22*

HIVdb+: Extending HIVdb functionality while securing data privacy

Abstract

The tremendous impact that HIV has had on the world can not only be seen in the striking current HIV statistics, but also in the enormous amount of information related to HIV that is generated on almost a daily basis. Much of this new information is associated with viral drug resistance, HIV treatment and sequencing of the viral genome that is relevant to drug action. Consequently, it has become difficult to conserve an updated knowledge of HIV resistance to antiretroviral therapies, to combine this knowledge with sequencing results from individuals infected with HIV, and to store all of the aforementioned information. The Stanford HIV Drug Resistance Database (HIVdb) addresses these issues to some degree. However, HIVdb does not allow users to personalize queries and to store that private information in some fashion. We propose HIVdb+ in order to overcome these limitations. HIVdb+ will allow users to query HIVdb while simultaneously including personal information such as a unique identifier or name for an individual whose sequence is under examination. HIVdb+ will also provide additional functionality such as detecting anomalies in lab results. The application will provide the user a searchable repository of past genotype queries. It will provide this functionality while maintaining protecting data privacy.

Background

Since the end of 2005, it has been estimated that 40.3 million adults and children are living with HIV/AIDS. Although AIDS has spread worldwide, its presence is greater in certain parts of the world such as Africa. It is estimated that in Sub-Saharan Africa 25.8 million adults and children are living with HIV/AIDS. In North America it is estimated that 1.2 million adults and children are living with HIV/AIDS. By December 2005 women accounted for 46 percent of all adults living with HIV worldwide, and for 57 percent in Sub-Saharan Africa. Consequently, perinatal infection has resulted in a significant number of children born with HIV as well as other consequences [1].

AIDS has led to serious consequences for individuals, healthcare systems of countries unable to cope with many AIDS victims, and the national economies of those countries due to the loss of young to middle aged who are economically most productive. For example, children are orphaned all over the world by the loss of one or both of their parents to AIDS. Africa has 12 million AIDS orphans. In addition, young people 15 to 24 years old account for half of all new HIV infections worldwide—more than 6000 become infected with HIV every day. More than 25 million people have died of AIDS since 1981, about the time when HIV was discovered.

The human inmmuno deficiency virus (HIV) was discovered in the early 1980's. There are two types of HIV, HIV-1 and HIV-2. Both types are known to cause AIDS, although infection with HIV-1 is more common worldwide. Structurally, the mature virion of HIV-1 is an icosahedral sphere. The outer envelope is formed from the host cell membrane and is a lipid bilayer. This bilayer contains host cell proteins and spikes of the viral envelope glycoproteins gp120 (spike of envelope glycoprotein) and gp41 (trans-membranous glycoprotein). Inside the lipid bilayer are the internal structural capsid and core proteins, p17 (matrix protein), p24 (ribonucleic protein), p7 (nucleocapsid, RNA with protein surround), and p6. These proteins enclose two copies of the single-stranded RNA genome and multiple reverse transcriptase molecules in the center of the virus particle [2]. Since its discovery, the virus has infected millions of individuals in a worldwide epidemic.

HIV-1 infects cells possessing the CD4 cell receptor in order to replicate. The virus recognizes CD4 by binding to it via the gp120 viral envelope glycoprotein. The viral and cell membranes fuse. The HIV contents are released into the cell cytoplasm. The capsid dissolves inside the host cell. Thus the virus exposes its contents including viral RNA and reverse transcriptase (RT) to the cytoplasm of the infected cell. Reverse transcriptase uses the RNA as template for a DNA transcription of the viral genome. This results in an RNA-DNA hybrid. The RNA strand is removed by the RT ribonuclease H activity (RNase H), leaving the single strand of DNA. A second strand of viral DNA is made, complementary to the first DNA strand. The newly transcribed DNA viral genome moves into the nucleus where it is integrated into the host chromosomal DNA by HIV-1 integrase. The proviral DNA becomes a template for transcription into RNA from which new viral particles are made to complete the replication cycle. Once a cell is infected with HIV, several destruction paths are possible, including budding and syncitium formation [3].

Several drugs are available to combat HIV at specific times and targets during its replication cycle. These drugs inhibit RT and protease function. Some classes of drugs that have been approved for HIV-1 infection include:

- Nucleoside reverse transcriptase inhibitors (NRTIs)
- Nucleotide reverse transcriptase inhibitors (NtRTIs)
- Non-nucleoside reverse transcriptase inhibitors (NNRTIs)
- Protease inhibitors (PIs)

The NRTIs, NtRTIs, and NNRTIs are reverse transcriptase inhibitors (RTIs). In addition, other drugs are currently being researched for their potential to inhibit fusion of the viral and cellular membranes at the commencement of HIV infection and to inhibit integrase function.

Problem Statement

The tremendous impact that HIV has had on the world can not only be seen in the striking current HIV statistics, but also in the enormous amount of information related to HIV that is generated on almost a daily basis. Much of this new information is associated with viral drug resistance, HIV treatment and sequencing of the viral genome that is relevant to drug action. Consequently, it has become difficult to conserve an updated knowledge of HIV resistance to antiretroviral therapies, to combine this knowledge with sequencing results from individuals infected with HIV, and to store all of the aforementioned information. The Stanford HIV Drug Resistance Database (HIVdb) addresses these issues to some degree. Users can query HIVdb to determine HIV drug resistance by using the Genotype Resistance Interpretation program found on the website (http://hivdb.stanford.edu/) [4]. The program interprets user-entered mutations to infer the level of resistance to NRTIs, NNRTIs, and PIs. The user has the option of obtaining a mutation list analysis by entering protease and RT mutations or a sequence analysis by entering complete sequence information. In either case, the HIVdb program provides the level of HIV drug resistance. However, there are limitations to this process.

The main limitation to this process is that users cannot include personal identification information with their requests and then store those results at HIVdb. One reason why this added functionality is not presently included is that storing personal information on HIVdb is a sensitive issue for users. However, if some storage capacity were combined with HIVdb, it could allow users to do more with their data. Users would be able to enter HIV genomic information for a person to organize and determine certain information such:

- Personal information (e.g., name, clinic, physician, sample collection date, sample-received date, date of data entry, file name).
- Drug resistance interpretation data.
- Quality control data

The drug resistance interpretation data refers to the results obtained from the HIVdb genotype resistance interpretation data. The quality control feature would allow laboratories to assess the quality of their sequencing efforts and to monitor the mutations in a person over time. For example, if two sequences from the same person were significantly different, this would be a sign of caution. Perhaps samples were mixed up or the dominant viral strain in the person is now different? To solve this problem, we propose HIVdb Plus or HIVdb+. HIVdb+ will allow users to query HIVdb while simultaneously including personal information such as a unique identifier or name for an individual whose sequence is under examination. HIVdb+ will store all information and query results locally onto the user's computer to resolve the issue of data privacy and storing the data in a location far away from the user.

System Design

Design Methodology

After reviewing the software methodologies illustrated in class (Waterfall, Spiral etc.) we chose to implement this project using the Rational Unified Process (RUP) approach.

RUP: A Brief Overview

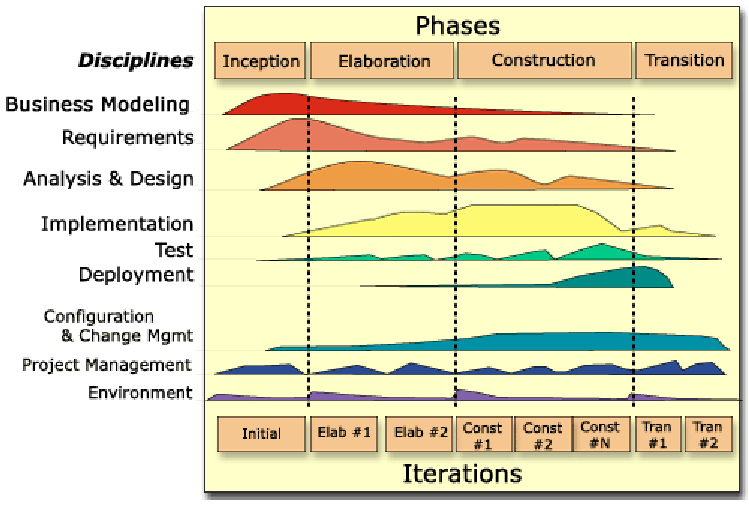


Figure 1: Phases of Rational Unified Process

The RUP process is a milestone-driven iterative software development framework where the software development lifecycle spans four phases [5]:

1. **Inception**: In this phase, the business case for developing a solution is evaluated. Factors such as financial viability, availability of resources and project plan are considered;

2. **Elaboration**: Once management has given the go-ahead, the system architect, system analysts and business analysts get together to design the system. Key artifacts generated in this phase include:
    a. A use-case model in which the use-cases and the actors have been identified and most of the use-case descriptions are developed. The use-case model should be 80% complete.
    b. A description of the software architecture in a software system development process.
    c. An executable architecture that realizes architecturally significant use cases.
    d. Business case and risk list which are revised.
    e. A development plan for the overall project.

3. **Construction**: The bulk of the software programming gets done in the construction phase. The key milestone of this phase is the product prototype;

4. **Transition**: In this phase, the product is transitioned from the development environment to the end-user environment. Tasks include user training, quality assurance and bug fixing.

Why we selected RUP

The primary reason for choosing RUP for this project is that it not only provides a framework for System Analysis and Design but specifies key milestones that need to be delivered at each phase. In keeping with RUP guidelines, we used UML to produce all the artifacts. UML allowed us to 'speak the same language' when we discussed the system requirements or the system architecture in a group setting [6].

Moreover, RUP best practices embody the key design goals that should be kept in mind when building bioinformatics solutions. These best practices include:

1. Develop software iteratively: Iterative development allows the project to be successively refined and addresses the highest risk use cases in the earliest iterations, thereby reducing risk;

2. Manage requirements: Requirements Management is well-defined in RUP and the proper tools are provided to capture requirements at the earliest stage of the project;

3. Use component based architecture: Component-based architecture creates a system that is easily extensible, intuitively understandable and promotes software reuse. RUP encourages Object-Oriented modeling. Object Oriented analysis and design are most suited to our system since it allows us to build a software model that matches the domain model very closely;

Requirements Analysis

This section describes the requirements for HIVdb+ project along with its actors and interaction between them. It is based on our analysis of the existing HIVDB, scenarios provided by the HIVDB group at Stanford and current requirements of the lab technicians.

The main goal of the requirements analysis phase is deciding precisely what to build and documenting the results of the requirements gathering. Requirements Analysis is critical for the success of any bioinformatics project since faults in the requirement phase cause a ripple effect in a project lifecycle where one fault might cause cascading errors in implementation and testing. The cost associated with a fault follows the "snowball paradigm": the cost increases exponentially as it moves from one phase to the next. Our team evaluated several approaches to requirement analysis and used a hybrid model for gathering and understanding requirements. In addition to conducting user interviews, we also relied on ethnography to study the existing workflow of users in a lab setting.

The HIVdb+ solution will cater to two sets of requirements:

1. Existing HIVdb functions: The HIV Drug Resistance Database at Stanford accepts user-submitted protease and RT sequences and returns the inferred level of resistance to 17 FDA-approved protease and RT inhibitors [3]. Several labs, patients and researchers use this service for gene sequence analysis and it is an extremely useful tool. HIVdb+ should provide this functionality as well. The queries that are submitted to HIVdb+ can have additional patient identification information or they can be anonymous queries;

2. Value Added features: In addition to existing functionality, HIVdb+ will offer several value-added services such as querying historical data, drawing inferences on patient history and prescribing course of treatment based on patient genotype/drug resistance information.

Identifying the Users

After defining the problem statement, we identified the following three users of the system:

1. **Lab Technician**: The Lab Technician is the primary user of the HIVdb+ application. He obtains the sample and follows a specific protocol for sequencing the relevant part of the HIV genome. This region of the genome

includes the RT gene and part of the protease gene and is associated with drug activity. Once the sample has been sequenced, the technician queries the HIVdb+ database to retrieve the list of drugs that the patient may potentially be resistant to (based on the genotype information). The technician can also query previously submitted queries and results by patient name and date.

2. **Researcher**: A researcher can use HIVdb+ to query historical data and draw inferences on the information. For instance, if the researcher finds that two sequences submitted on the same day are too similar in structure, this might indicate sample contamination. HIVdb+ can also provide tools to draw additional inferences from the dataset such as correlation between virus-load, virus mutations and drug resistance.

3. **Physician**: While the researcher uses HIVdb+ to draw inferences about HIV-related data from an academic perspective, the physician uses HIVdb+ to retrieve data that helps with providing better clinical care to her patients. For instance, the physician can retrieve data about the patient's genotype and treatment history and based on that information, suggest further course of treatment.

Functional Requirements

We derived the functional requirements of the application based on our study of the users and existing workflows. The following list describes the main functional requirements:

1. **Genotype interpretation information**: The user should be allowed to enter sequence data (or gene mutation data) and retrieve an interpretation of that data from a drug resistance perspective. This functionality is currently provided by HIVdb already but a key difference is that the user should have the choice of tagging each query with patient identification information;

2. **Retrieve additional inferences from tagged queries**: As a consequence of tagging HIVdb queries with patient identification information, the results generated from these queries will include additional inferred information. For instance, if the Lab Technician submits sequence data that is too similar to another sample that had been submitted earlier in the day, HIVdb+ will provide a low 'Quality score' that indicates that particular sample might be contaminated. Another example of additional inference includes the case where there is a large variation in genotype samples from the same patient. This might indicate that the patient has been re-infected or that a specific sample was contaminated;

3. **Querying historical patient data**: A search screen should be provided to query previously submitted genotype queries. These searches can be performed based on parameters like patient name, date range and/or date sample was submitted. Search results should be displayed in a concise tabular format with the option of 'drilling-down' into individual records for more detail;

4. **Trend Analysis**: HIVdb+ will mine the data repository and draw trends that correlate gene mutations, viral load and drug efficacy. The physician can use these trends to prescribe further course of treatment for the patient;

Nonfunctional Requirements

In addition to meeting the functional requirements above, HIVdb+ should meet the following non-functional specifications:

- **Security**: The primary goal of HIVdb+ is to provide services without allowing unauthorized eavesdropping on the data. To address security, data stored in the local database should deploy strict access control. Authorized users list (name and password) should be encrypted;

- **Easy to use**: HIVdb+ is designed for users from different backgrounds and with little technical knowledge. UI is simple and self-explanatory that does not require much user training;

- **Fast response time and retrieval rate**: HIVdb+ tries to minimize the traffic between the central HIVdb repository and virus labs. This results in better response time and retrieval rate;

- **Internationalization**: HIVdb+ is designed for global users so internationalization is a key non-functional requirement. The final application should support different languages;

- **Maintainability and reliability**: Once a user is registered, HIVdb+ should provide full automatic maintenance where user does not need to worry about low level data details;

- **Extensibility and Reusability**: HIVdb+ uses a layered architecture so that it can be extended and applied to other applications with minimum effort. For instance, with some customization, the same system can be used to track Tuberculosis and Cancer patients as well.

User Interface Requirements

In keeping with the project vision statement, the application should be easy to use. We do not expect the user to be technically savvy. However, a key UI decision had to be made when designing the installation screens. Client installation involves installing a local database and an application server. There are two approaches to installation:

a. Install the servers without asking the user any questions: This approach has the benefit that the novice user does not see any technical jargon ('Application Server', 'Database server') but it has the downside that an advanced user who wants to configure an existing database to work with the application cannot do so;
b. Ask the user several questions before installing the servers: This approach is great for the advanced user but will most likely confuse the non-technical user.

The UI that we finally selected has been carefully crafted to provide default options to the user at each stage with an 'Advanced' option on each screen. Clicking the 'Advanced' button will take the user to another screen where they can configure existing servers. This approach has the simplicity of the former approach and the flexibility of the latter.

Workflow Analysis

1. **Genotype Interpretation**: The current workflow for querying genotype interpretation information from Stanford HIVdb involves the lab technicians submitting the sequence data online via hivdb.stanford.edu and obtaining the results instantly. The proposed workflow is very similar with the following variation:
   a. The user has to login to the system providing a login/password;
   b. On the query screen, the user can optionally provide patient identification data;

2. **Quality Control**: Currently, there is no standardized process for labs to determine when samples are contaminated. At best, each lab has its own customized application that tracks sample variation. In the worse case, there is no systematic approach to detect contamination.

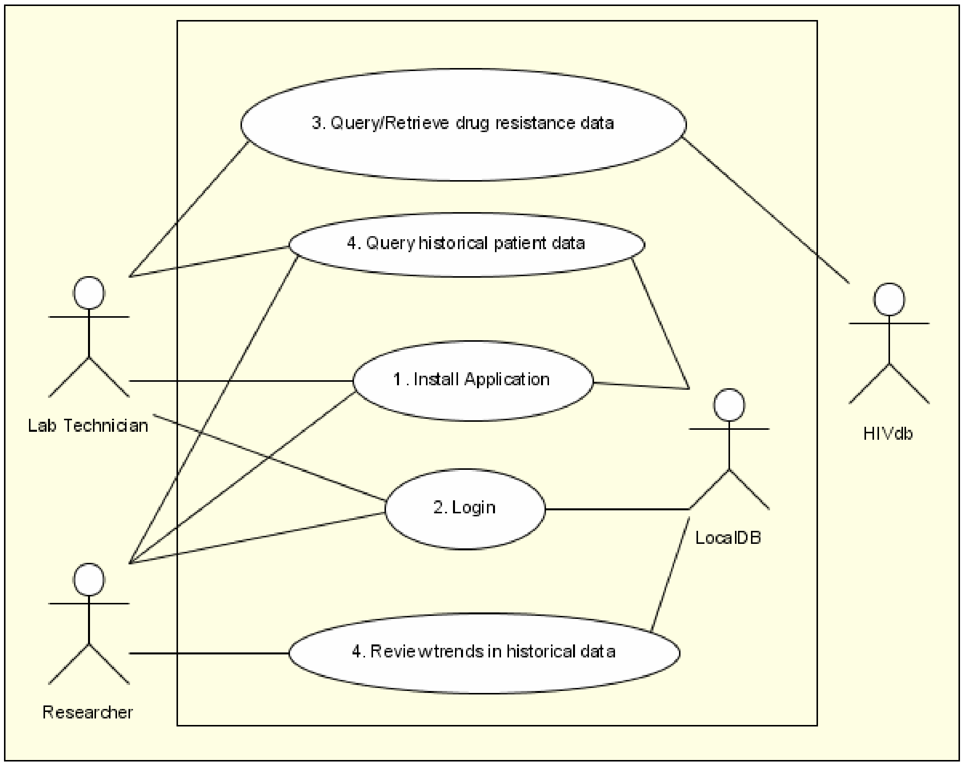
   With HIVdb+, a quality score will be displayed along with the results of the genotype interpretation query. This score will be based on a comparison of the current sample to other samples in the database or other samples of the same patient;

3. **Search Patient History**: As with quality control, different labs may provide the search functionality in different ways, or not at all. In the worst case, all results are stored in print and a manual search is required to build patient history.

   HIVdb+ will provide a standardized easy-to-understand search page where users can enter the required search parameters (such as Patient Name, Date Range, Physician Name etc.). The results of the search are displayed in a tabular form and the user can drill down to the individual reports.

Use Case Analysis

We used Use Case Analysis as the primary tool during the Requirements Analysis phase. The following use case diagram identifies the actors and the main use cases:


3. Query/Retrieve drug resistance data
4. Query historical patient data
1 . Install Application
2. Login
4. Reviewtrends in historical data
Lab Technician
Researcher
LocalDB
HIVdb

Use Case Descriptions

1. **Install Application**: This is a one-time installation process that the user (Lab Technician, Researcher) will initiate from the browser. The installer will create a local application server, a local database (of the user's choice) and install the client-side code necessary for addressing the requirements of the lab technicians;

2. **Login**: Users of the system are required to Login to use the system. We will the 'Role-based UI' methodology to design the user interface. This ensures that the user interface is geared to helping the specific user that is logged in. For instance, if the Lab Technician logs in, the UI will not display the option to perform trend analysis on historical data whereas if the Researcher logs in, this feature is displayed.

3. **Query/Retrieve drug resistance data**: The user will enter the sequence data (either Protease and RT mutations or the complete sequence) along with additional parameters (Patient name, Date/Time). The query and the results are stored in the local database.

4. **Query Historical Patient Data**: The user is presented with an easy-to-use Search box where she can enter any/all search parameters and submit a query. The query is run against the local database and the resulting data is displayed in a tabular format.

A Cognitive Perspective

To gather reasonably complete and reliable requirements for HIVdb+, we used different requirements gathering approaches including self-report and ethnographic study. We met with our primary user, Tommy Liu, several times and asked questions for understanding user needs. While self-report provides a direct and efficient way of information gathering, it may not be accurate due to inherent cognitive biases in the user's description of the requirements. The distributed nature of some tasks makes it difficult for us to infer the requirements by speaking to just a subset of people involved in understanding that task. To avoid information deficit due to cognitive biases we performed ethnographic studies [8]. We visited the Stanford Virology Lab for observing and understanding the existing work flow and user needs. This hybrid approach allowed us to capture requirements that we would otherwise not have been able to get if we used just an individual approach.

For instance, during the self-report the user told us that a drop-down menu is used to make a selection for a particular value. However, on visiting the virology lab in the ethnographic study, we saw the lab technician struggle to make a selection in the drop down menu because the menu contained over a thousand options! It also took a long time for the drop-down to render.

We have taken particular care to address these usability concerns in addition to meeting the functional requirements of the project. For instance, we resolved the particular problem mentioned above by introducing the concept of a 'List of Values'. (Refer to Appendix C for details).

Project Vision Statements

At times, it was hard to meet the functional requirements of our system while addressing cognitive concerns as well. For instance, early on in the project while deciding the system architecture, we had to decide whether we should ask the user to install a local database. While this was a critical requirement for addressing the functional requirement of storing private data at the client side, such a requirement would not recognize the concern that the user might be a non-technical person who would have difficulty installing the database.[1]

In order to address these cognitive concerns while designing the application, we devised three vision statements that the designers and developers should keep in mind while implementing the functional requirements:

1. Maintain data privacy at all costs;
2. Solution should be easy to use;
3. Build application quickly;

The vision statements above are listed in order of priority. For instance, imagine two approaches to a particular design concern: one satisfying the first project vision while the second meets the second project vision. We would choose the first approach over the second since the concern for data privacy overrides the requirement for ease-of-use.

[1] We eventually designed an installation UI that allows the novice user to complete the installation by just clicking 'Next' without changing the default options but at the same time allows an 'advanced' option on each screen to allow the technically-savvy user to customize the installation.

System Architecture

The unique selling proposition of our solution is the novel system architecture on which it is built. Our application had to address the concern of client data privacy while still allowing access to advanced algorithms that allowed remote labs to make inferences on private data.

We model Knowledge as a combination of Data and Algorithms that act on that data. For client labs to access Knowledge, they need to store historical patient data **and** have access to algorithms that will process that data.

In our architecture, Knowledge is partitioned over two geographical locations:

1. A Local Database at the remote location contains historical patient data;
2. The algorithms that process the historical data are located at the Stanford HIVdb.

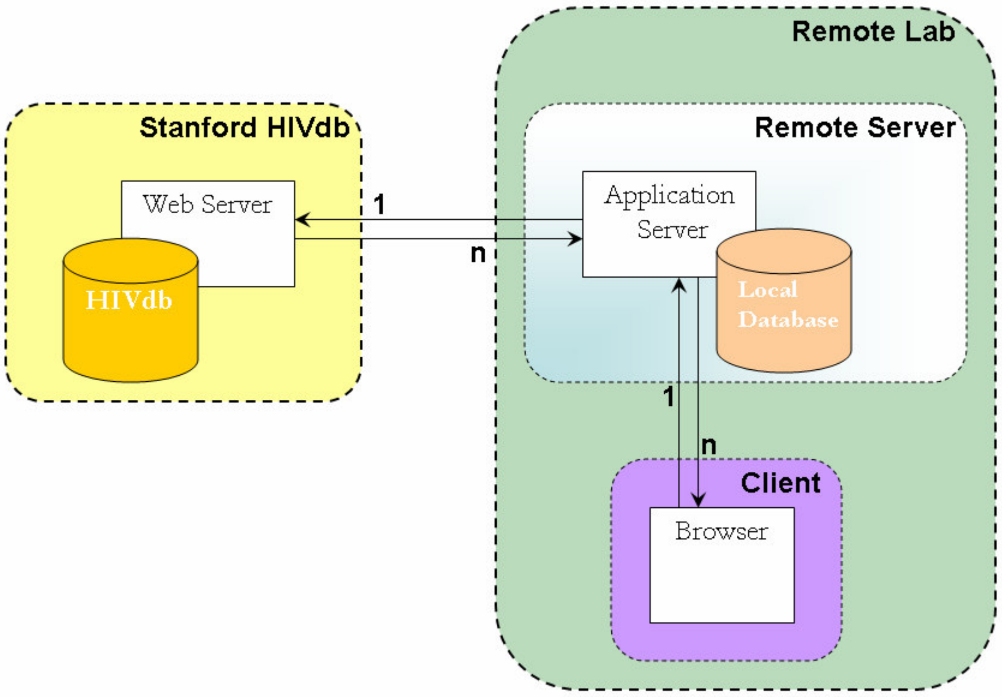


Figure 2: System Architecture Overview

We transfer the processing algorithms to the remote site and process the data locally. Transferring an algorithm to the processing site rather than transferring the data to be processed to the location of the central processor, is a key feature of our solution. This transfer of algorithm is achieved in the following ways:

1. Upon first installation, the client automatically gets a copy of the processing algorithms;
2. The RemoteServer regularly checks the Stanford HIVdb server for any updates on the processing algorithm. If a new upgrade is available, the algorithm is downloaded using standard Object Serialization/Deserialization techniques. The algorithm is packaged in a JAR file for easy transfer over the network.

For an alternative architecture where data is transferred to Stanford HIVdb for processing, please refer to Appendix A.

The HIVdb+ solution can be split into three primary components:

1. **Stanford HIVdb**: This is the existing application (database and web server) maintained at Stanford;

2. **Remote Server**: This is the installation of our software at the client site. It consists of a local database and a light-weight application server. If local databases and web servers already exist at the remote site, we use these servers instead of new installations;

3. **Client**: This is the user of our web-based application. The user may have one of three roles: administrator, lab technician and researcher.

Data flows in the following directions:

1. **From Remote Server to Client**: Clients always access functionality through the Remote Server. This ensures that all patient data is safely captured in the Local Database

2. **From Stanford HIVdb to Remote Server**: When the client issues a Genotype Interpretation query to the Remote Server, the Remote Server delegates this operation to Stanford HIVdb (via a web service). When the Stanford HIVdb responds with the results, these results are first stored in the Local Database and then relayed to the Client browser. In effect, the Remote Server acts as a proxy for the Stanford HIVdb for the Genotype Interpretation service.

Remote Server Architecture

The Remote Server in the system architecture above is the primary component in the HIVdb+ solution. We have decided to implement this solution using Java 2 Enterprise Edition (J2EE).

The proposed application will be based on the standard 3-tier architecture and the Model-View-Controller design pattern. In order to reduce 're-inventing the wheel' and minimize time-to-market we will use existing J2EE frameworks, where applicable. The following figure illustrates the three tier architecture and the choice of J2EE development frameworks:

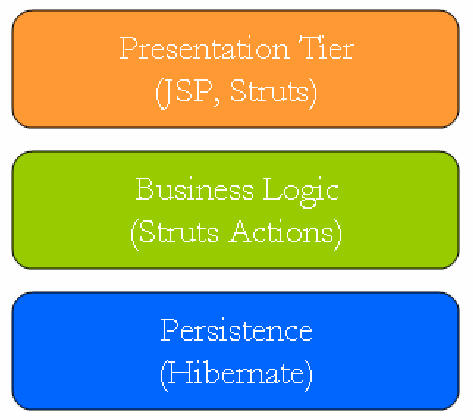


Figure 3: Remote Server 3-tier architecture

Software Components

1. **Knowledge Base:** Knowledge is modeled in our application in an object-oriented model. As mentioned earlier, we model knowledge as the data and the algorithms that process that data. Our object-oriented model captures this idea very well as data is stored in class attributes and the operations on data are stored in class methods. As mentioned previously, the latest version of the algorithms that act on the data are obtained from the Stanford HIVdb server. Our model mirrors the ontology-driven object model in PharmGKB;

2. **Database**: Physical Data is partitioned into two databases with genotype interpretation information being stored in the centralized Stanford HIVdb and local patient history being stored at the client site local database. It is important to note that at no point is confidential patient history data transferred to the Stanford HIVdb. Moreover, we allow the user to use an existing database instead of installing the default database that comes with our application. The Metawrapper/Wrapper framework ensures queries are translated into the query language understood by the local database.

   The local database ER model mirrors the Stanford HIVdb. In addition to utilizing the entities that store existing HIVdb information, we will also make use of temporal components (to stripe database queries by time) and patient identification entities. Since HIVdb+ uses a 'Role-based user interface' and requires users to login to the application, the extended ER model also includes entities to capture the requirements of user authorization. Refer to Appendix B for the extended ER model.

3. **Middleware:** To translate data from the object oriented model to the relational model we use the MetaWrapper/Wrapper pattern. More specifically, we use the well-known Hibernate Object/Relational Mapping framework to do the bi-directional translation.

   > **"**Hibernate is a powerful, high performance object/relational persistence and query service. Hibernate lets you develop persistent classes following object-oriented idiom - including association, inheritance, polymorphism, composition, and collections. Hibernate allows you to express queries in its own portable SQL extension (HQL), as well as in native SQL**" (http://www.hibernate.org/)**

   The MetaWrapper will accept queries in the domain language (a 'semantic' query such as '*Retrieve all sequence/drug resistance data for Patient A in a specific date range*') and Hibernate will translate the query into the implementation-specific database query (a 'syntactic' query such as 'SELECT * FROM sequence_data WHERE patient_id = :patientID and date_range between :from and :to').

   Providing support for an existing database type is simply a matter of defining an XML file showing the mapping from our Object Model to the local database schema. During installation, we will provide the user a visual mapping tool to map their existing schema to our object model.

4. **User Interface:** Java Struts (http://struts.apache.org) will be used to implement the presentation layer. Struts has been used to implement the user interface for several leading web applications. It provides several key benefits that are attractive for this application:
   - **a.** Clearly separates Presentation from the Model: This allows us to change the presentation layer at a later stage. For instance, the presentation layer could be re-written to display on PDAs without changing any of the model or business logic;

   - **b.** Built-in support for internationalization: Adding support for a new language is simply a matter of defining a new XML file with messages in a new language. Since our application is accessed from labs around the world, this feature is especially useful;

   - **c.** Allows parallel development: Since Struts separates UI from business logic, both of these can be developed in parallel thereby reducing development time.

   Based on the functional requirements and interactions with the future users of the system, we devised a set of user flows and screen mockups. These can be seen in Appendix D.

The following UML package diagram illustrates the M-V-C pattern and the MetaWrapper/Wrapper patterns discussed above:

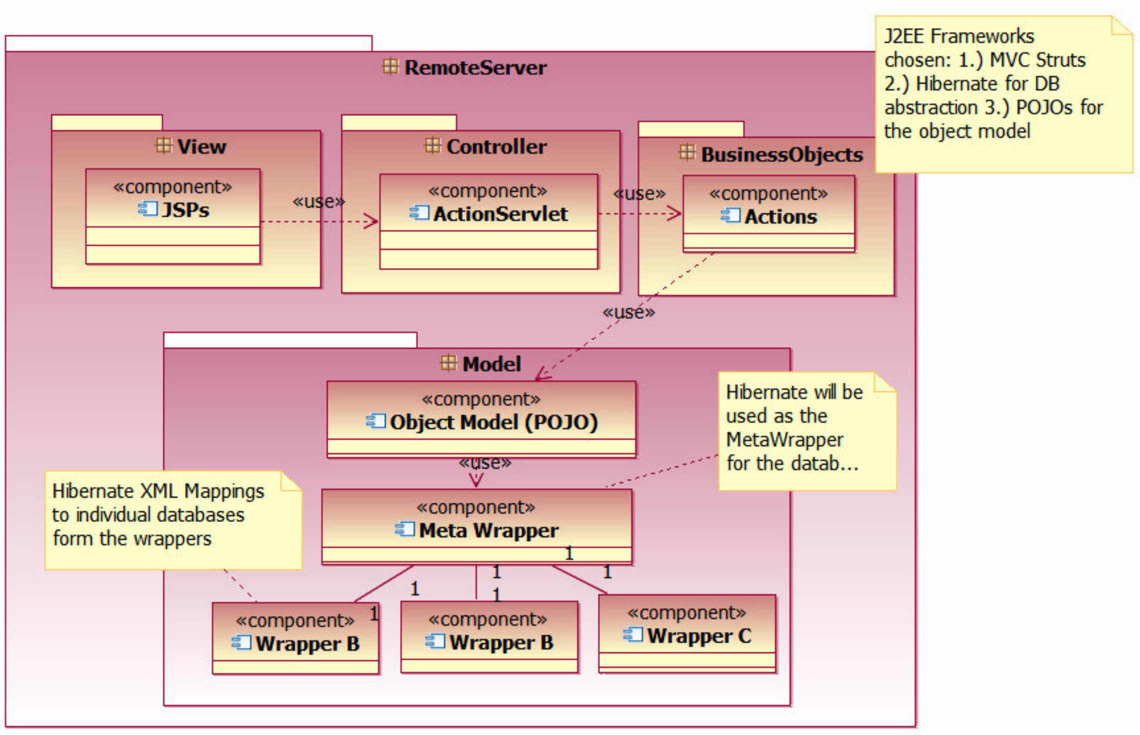


Figure 4: Remote Server package diagram

System Evaluation

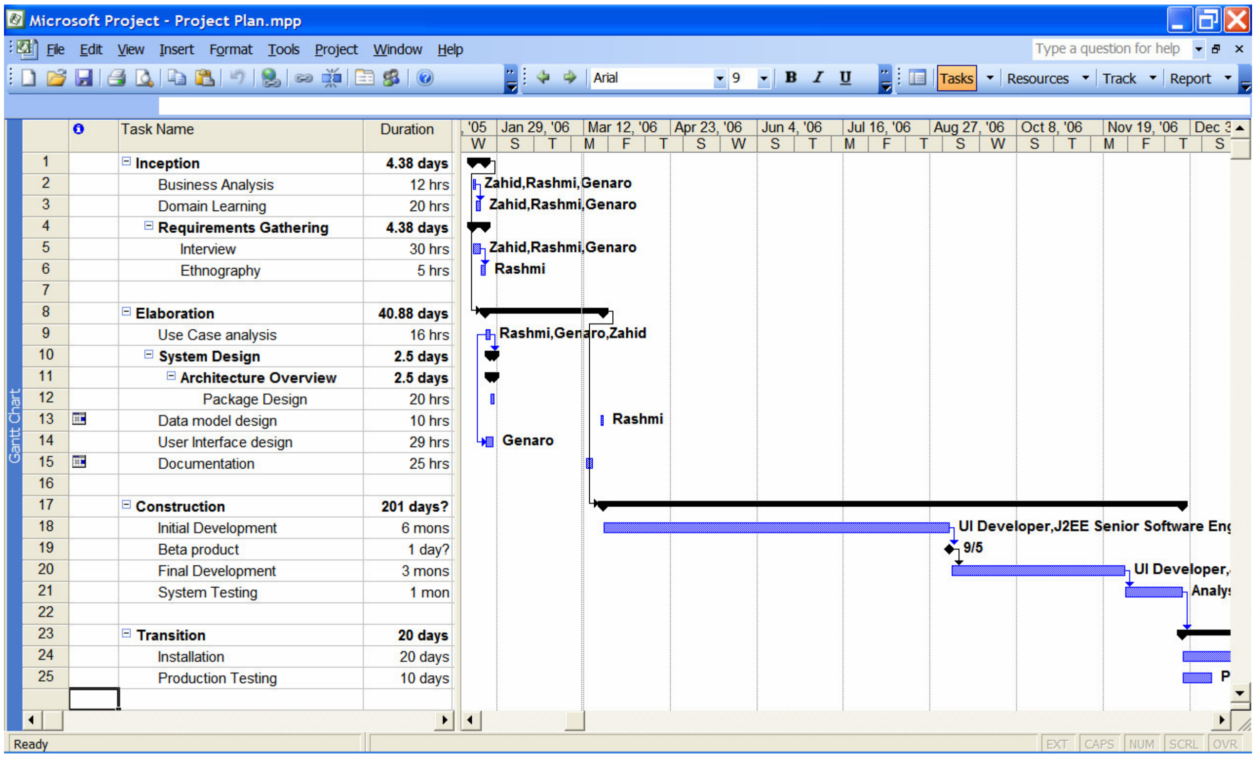
System evaluation is a critical step in the software engineering process that requires a wide range of data-collection activities designed to answer questions ranging from casual questions (what is the purpose of HIVdb+?) to the more focused questions (What benefits will HIVdb+ offer over the existing set of features of HIVdb?)[1] There are several reasons for performing studies including promotional, scholarly, pragmatic, ethical and medicolegal reasons. [1] It has been seen that the primary reason of IT projects failure in healthcare is the poor understanding of practical utility and user needs. In the evaluation of our system, we will use subjective and objective approaches to evaluate user needs and practical utility of the system.

- **Subjective Evaluation**: Subjective evaluation techniques make use of expert observers and users of the system. As mentioned earlier we spent a significant amount of time for requirement gathering and requirement analysis so that we could understand user needs accurately. We have already displayed our system (UI) to one of the users (Tommy) for getting his opinion before moving forward in the design process. We plan to keep users in the loop throughout the design and implementation process;

- **Objective Evaluation**: Objective evaluation techniques for HIVdb+ can be based either on explicit knowledge or on a comparison model with existing systems. We will do a comparative study based on a local evaluation of the HIVdb+ application. The same technique could be extended to various different levels of granularity.

  Local Evaluation: We will compare HIVdb+ with HIVdb for performance and number of new and returning users metrics. Furthermore, we plan to provide a feedback system for evaluating user experience and for improving the system. HIVdb+ is designed to minimize the traffic between central server and local lab users. This will definitely increase the performance for data rate and throughput.

System Implementation



Microsoft Project - Project Plan.mpp

	Task Name	Duration
1	Inception	4.38 days
2	Business Analysis	12 hrs
3	Domain Learning	20 hrs
4	Requirements Gathering	4.38 days
5	Interview	30 hrs
6	Ethnography	5 hrs
7		
8	Elaboration	40.88 days
9	Use Case analysis	16 hrs
10	System Design	2.5 days
11	Architecture Overview	2.5 days
12	Package Design	20 hrs
13	Data model design	10 hrs
14	User Interface design	29 hrs
15	Documentation	25 hrs
16		
17	Construction	201 days?
18	Initial Development	6 mons
19	Beta product	1 day?
20	Final Development	3 mons
21	System Testing	1 mon
22		
23	Transition	20 days
24	Installation	20 days
25	Production Testing	10 days

The above is a proposed project plan that follows the four-phase RUP development cycle. Note the following points in the project plan:

1. In this class, we completed the Inception and Elaboration phases of the project. This took a total of 45 days (or 45*8 hours);
2. The Construction and Transition phases will span approximately 9 months;
3. The following resources are required to complete the rest of the project:
    a. Business Analyst;
    b. UI developer;
    c. J2EE Software Engineer;
    d. Windows developer;
    e. Software Test Engineer.

Project Scoping

In order to mitigate project risk, we recommend that the following features be left for a future release:

1. Trends Analysis: Since there won't be much data in the local data repository, this feature will not be of much use initially. Since it will take a long time to implement and the cost-benefits ratio seems to be too high, we recommend that this feature be postponed for a future release;
2. Customized Installation: As mentioned earlier, HIVdb+ provides the user the flexibility to use an existing database and/or application server. However, this flexibility comes at the cost of more complex installation. Implementing customization in the installation application might prove to be a time-sink since custom tools need to be developed to provide the data-mapping required to generate the Hibernate XML mappings. This feature is best left for a future release as well.

Cost Analysis

The total estimated cost for completing the project using the five resources above is USD155,000.

Task	Resources	Average Hourly Pay	Total Cost (USD)
Initial Development	b/c/d	$35/hour	110,000
Final Development	c/d	$35/hour	33,600
Testing/Installation	a/e	$30/hour	9,600
		TOTAL	**153,200**

We split development into two phases: the bulk of development efforts will be concentrated in the first phase. Initial Development will result in a Beta product that will be released to a small audience for feedback and product refinement.

Based on the user feedback, there will be a final round of development to fine-tune the application before a larger public release.

Hardware Cost: USD1500 (for server and RAID hard disk setup)

Summary

With the explosion in the number of HIV patients and their gene-sequencing data, it has become increasingly important to store the data efficiently and mine it for relevant and important information. We proposed a novel approach to extend the existing HIVdb functionality while addressing the critical concern for data privacy. We outlined a feasible plan to build the proposed solution in a timely and cost-effective manner. The generic nature of the system architecture used to implement our solution ensures that a similar design pattern can be used to implement other solutions that require access to processing power without divulging personal information.

References

[1] *AIDS Epidemic Update*, December 2005

[2] *HIV Resistance and Implications for Therapy*; Brendan Larder, Douglas Richman and Stefano Vella

[3] *HIV Drug Resistance Database*
http://hivdb.stanford.edu/

[4] *Molecular Biology of an HIV infection: an interactive animation*, Kristin Henry
http://www.galaxygoo.org/hiv/hiv_lifecycle.html

[5] *Business modeling practices: Using IBM Rational Unified Process*
Maria Ericsson, Principal Consultant, IBM

[6] *Rational Unified Process*
http://en.wikipedia.org/wiki/Rational_Unified_Process

[7] *Software Requirement: A Tutorial*, Stuart R. Faulk

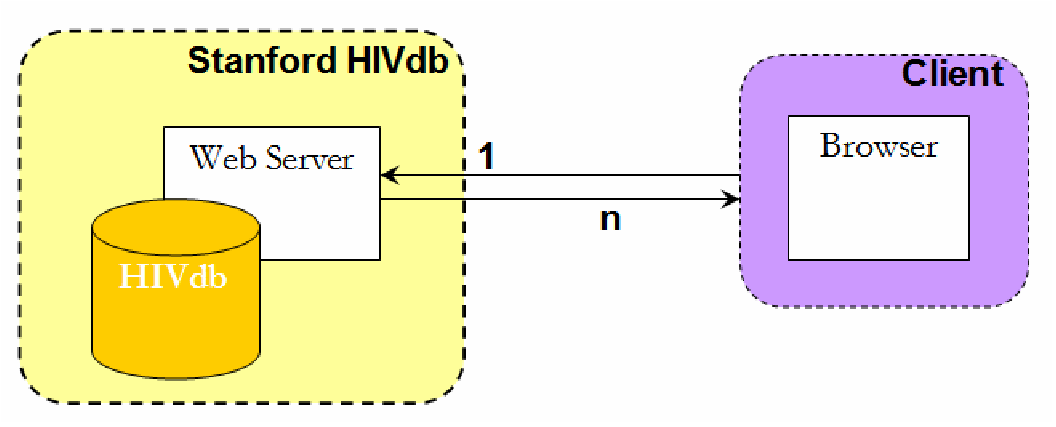
[8] *Using ethnography to investigate life scientists' information need,* Diana E. Forsyth

[9] *Evaluation and Technology Assessment*, Charles P. Friedman, Douglas K. Owens, and Jeremy C. Wyatt

[10] *HIV Drug Resistance Database: Database Schema* http://hivdb.stanford.edu/pages/documentPage/schema.html

Appendix A

Alternative System Architecture

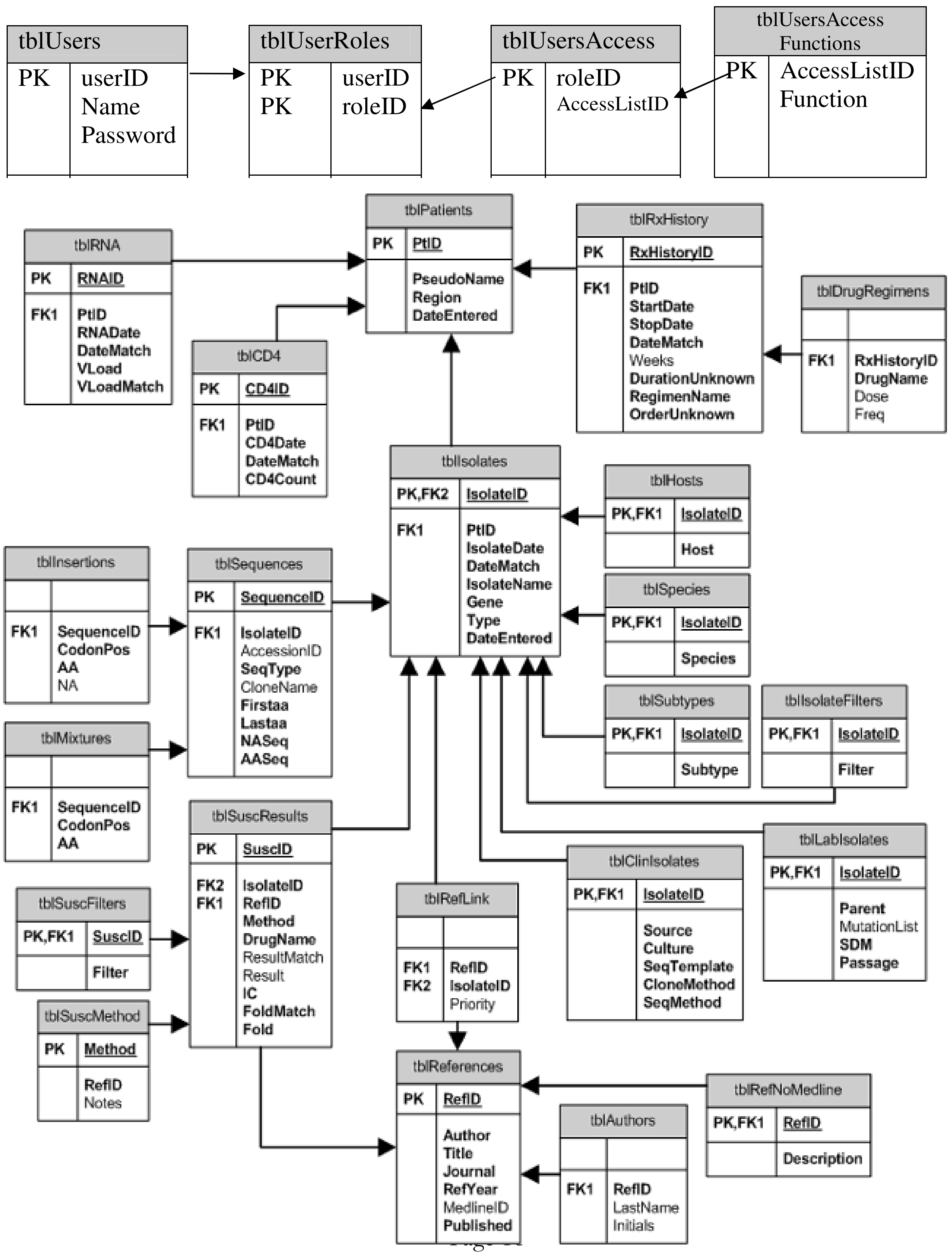


The architecture above was considered as a candidate for the system architecture. The user accesses functionality through a standard web browser. However, even though this architecture meets most of the functional requirements of the system, it stores localized data at the Stanford HIVdb server. This data is encrypted so that, theoretically, only the Client Lab has access to this information. However there are several drawbacks to this approach:

1. Even though localized data stored on the central datastore is secured using a secure encryption protocol, labs overseas are reluctant to save patient-related information at a remote location;
2. Since there is no data stored on the client side, clients cannot build any customized applications around their data;
3. In this model, Stanford will host the databases for clients from all around the world. This will increase the burden on Stanford servers and increase the likelihood of server outages, which will in turn affect labs all around the world.

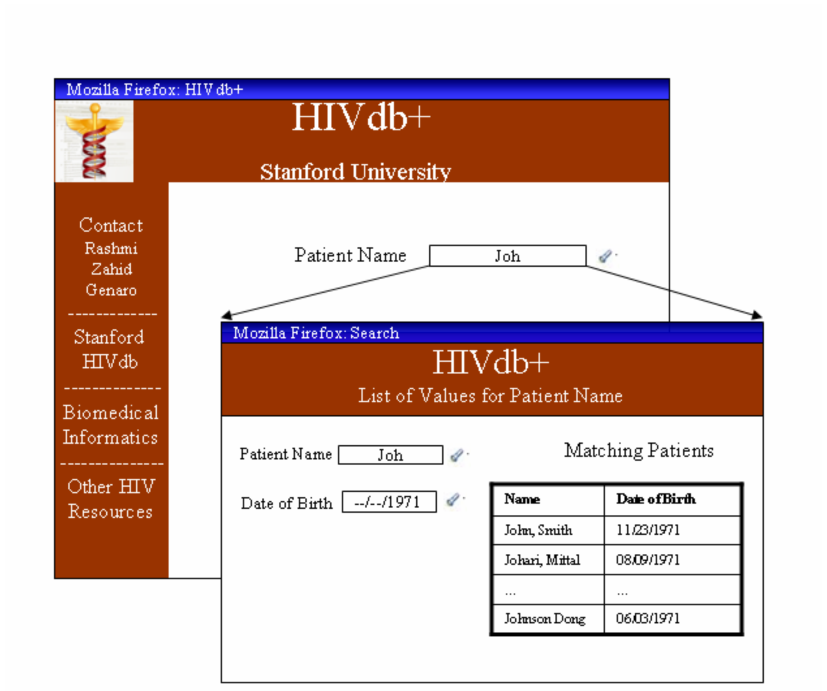
Appendix B

Data Model [10]



Appendix C

User Interface Design concept



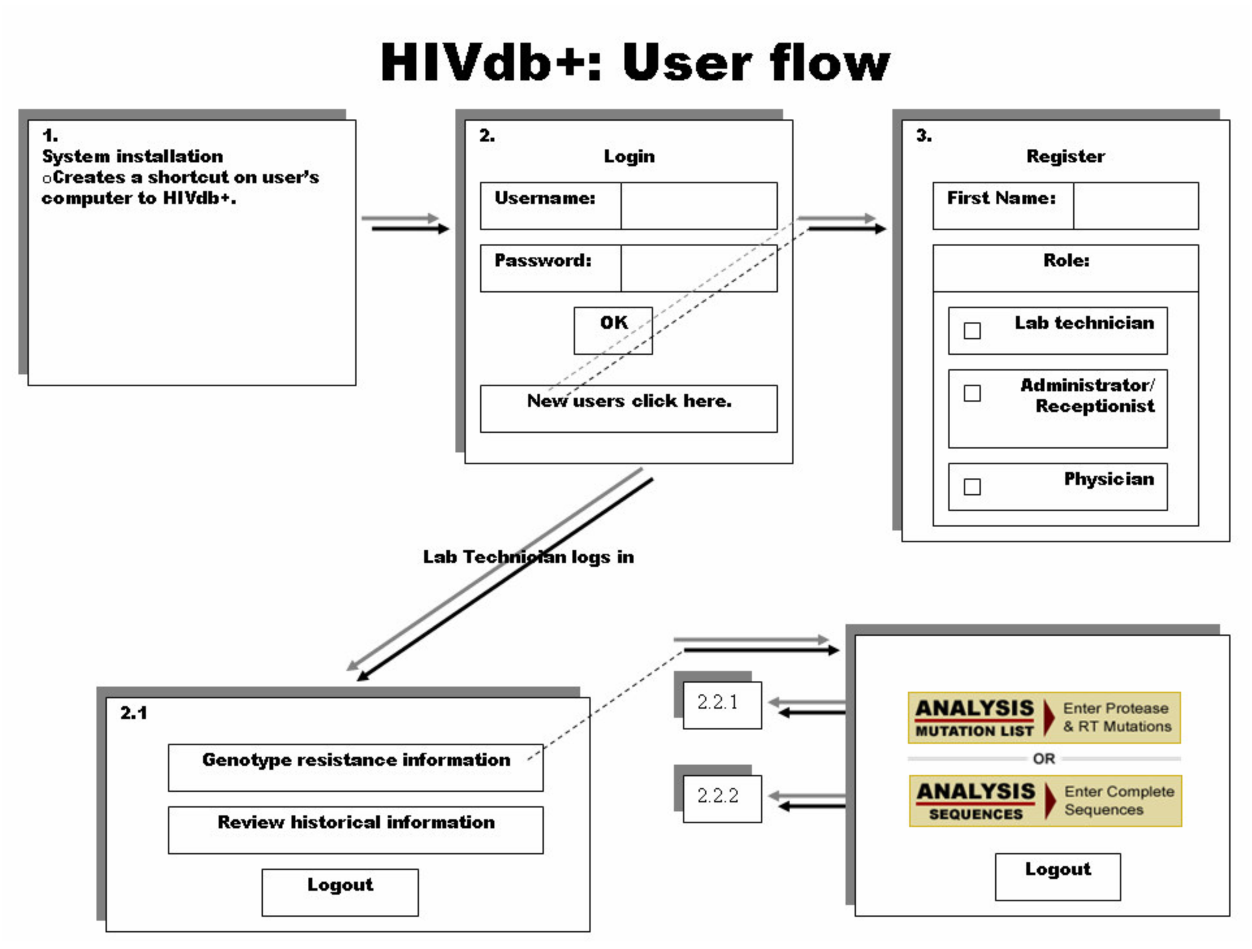
Name	Date of Birth
John, Smith	11/23/1971
Johari, Mittal	08/09/1971
...	...
Johnson Dong	06/03/1971

We use the 'List of Values' (LOV) design feature when the user has to select from a large list. For instance, if the user has to enter a Patient Name, she enters the first few letters of the name and clicks on the 'flashlight' icon next to the field. If the letters that she entered uniquely identify the patient, then the name is automatically completed for her. However, if there are several patients whose names start with the letters she entered, then a search-and-select screen is displayed where the user can select from a list of names. Another advantage of using LOVs is that it helps the user to identify two patients who might have the same name. If there are two users called 'John Smith', then the user can do a further search on 'Date of Birth' to select the intended patient.

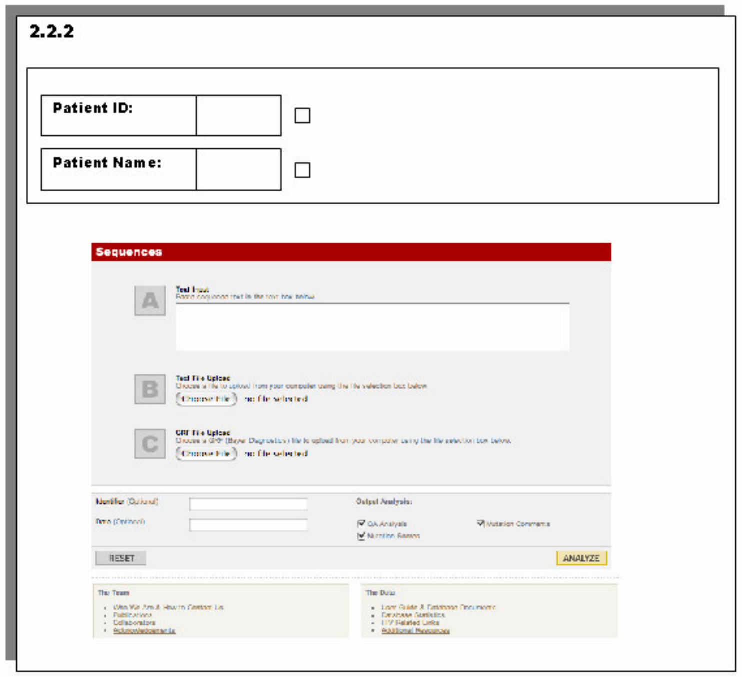
The alternative to this design feature is a standard 'drop-down' menu. But when the set of choices is very large, these menus are difficult to use and take a very long time to render on the screen.

Appendix D

User Flow (for 'Lab Technician' role)



Screen 2.2.2: Enter Tagged Sequence Data



2.2.2

Patient ID:		☐
Patient Name:		☐

Sequences

A **Text Input**
Paste sequence text in the text box below

B **Text File Upload**
Choose a file to upload from your computer using the file selection box below
Choose File no file selected

C **GRF File Upload**
Choose a GRF (Bayer Diagnostics) file to upload from your computer using the file selection box below.
Choose File no file selected

Identifier (Optional)

Date (Optional)

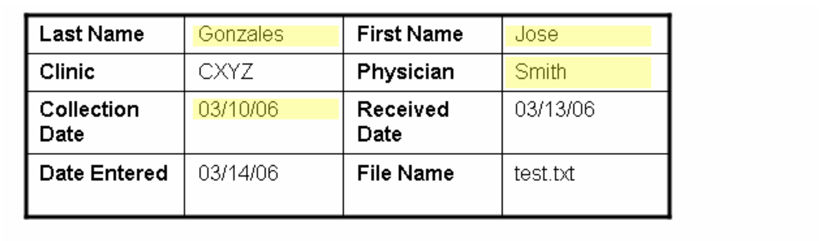
Output Analysis:
✓ QA Analysis
✓ Mutation Comments
✓ Nucleotide Search

RESET ANALYZE

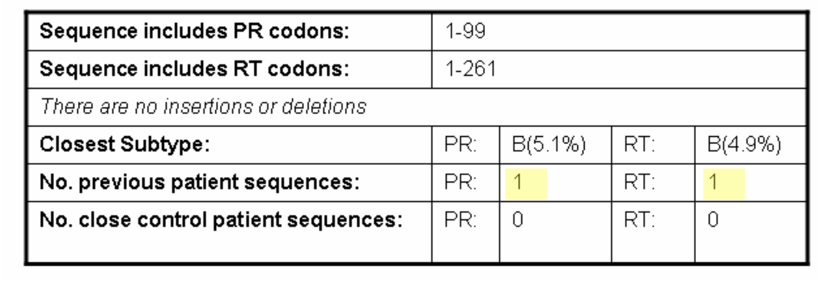
The Team
- Who We Are & How to Contact Us
- Publications
- Collaborators
- Acknowledgements

The Data
- User Guide & Database Documents
- Database Statistics
- HIV Related Links
- Additional Resources

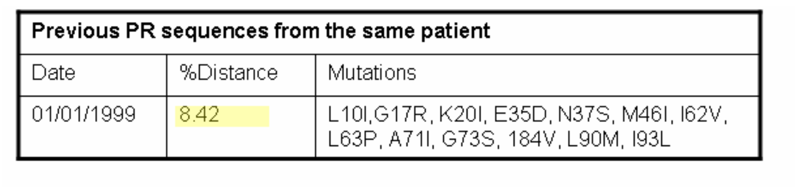
Typical Result of Genotype Interpretation query



Last Name	Gonzales	**First Name**	Jose
Clinic	CXYZ	**Physician**	Smith
Collection Date	03/10/06	**Received Date**	03/13/06
Date Entered	03/14/06	**File Name**	test.txt

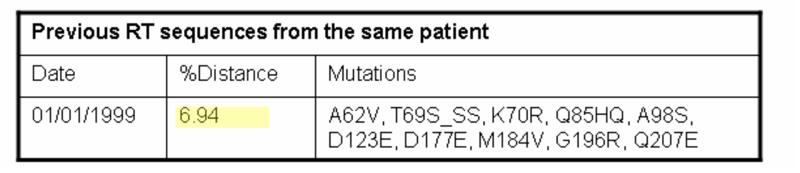


Sequence includes PR codons:	1-99			
Sequence includes RT codons:	1-261			
There are no insertions or deletions				
Closest Subtype:	PR:	B(5.1%)	RT:	B(4.9%)
No. previous patient sequences:	PR:	1	RT:	1
No. close control patient sequences:	PR:	0	RT:	0



Previous PR sequences from the same patient

Date	%Distance	Mutations
01/01/1999	8.42	L10I,G17R, K20I, E35D, N37S, M46I, I62V, L63P, A71I, G73S, 184V, L90M, I93L



Previous RT sequences from the same patient

Date	%Distance	Mutations
01/01/1999	6.94	A62V, T69S_SS, K70R, Q85HQ, A98S, D123E, D177E, M184V, G196R, Q207E