Chapter 2

Design of an integrated comparative genomics environment

2.1 Introduction

There are many types of studies and tools that researchers may want to use when performing comparative genomic studies. As seen in chapter 1, some methods intimately linked with comparative genomics include sequence alignment, BLAST, synteny studies, SNP analyses etc. Often, analysis servers are dedicated to hosting specific analysis tools. This requires a user to navigate to the specific server, upload data, submit data for processing and then finally download the results. This process is often time consuming. If a user, on the other hand, chooses to download and use a specific tool on their local computer, there is still the requirement of downloading and installation of the actual software. The skills and technical know-how required for these tasks are often outside the scope of the regular lab-based biological researcher. Furthermore, after attempting to use software, the results produced by the software may not be in a form compatible for input into other software, thus requiring further processing of the results which is not a trivial task. This justifies the need for an integrated system that not only offers easy access to all the main comparative genomics tools, but also a degree of compatibility and communication between the software, thus the need for an integrated comparative genomics environment. In this study, such a system was developed as a sub-module of the larger FunGIMS (Functional Genomics Information Management System) project. The comparative genomics environment was built as part of FunGIMS for several reasons. Firstly, it offered a pre-built and stable database schema with which to expand on. FunGIMS also already featured a high performance server that was suitable for the inclusion of a new module. Also, importantly, is that the main function of FunGIMS was to offer a highly integrated software environment catering for the main biological data-types which should ideally include comparative genomics. This chapter firstly deals with a brief overview of FunGIMS, then with comparative genomics environments in general and what they have to offer, followed by the design of the new comparative genomics environment which was developed in this study.

2.2 FunGIMS

2.2.1 Overview of FunGIMS

As already mentioned, FunGIMS or the Functional Genomics Information Management System is a system aimed at providing a web-based environment where biological researchers are able to manage a variety of functional genomics data types (private and public) including micro-array data, protein and DNA sequences as well as small-molecule data. Sub-modules within the project, each specifically geared toward the different data types, allow for users to also perform general, commonly performed analyses tasks on the relevant data types. These sub-modules are built into FunGIMS in such a manner so as to facilitate 'cross-talk' between the various data-types and establish biologically relevant linkages between the various data-types. Core modules comprising FunGIMS are responsible for the security, database access, data storage and user management of the system. FunGIMS design was based on the Model-View-Controller design paradigm (discussed later) and henceforth, a brief description of FunGIMS will be given within this context.

2.2.2 Model

One of the main purposes for the development of FunGIMS was the need for integration of several biological data types. For this reason, a versatile data model had to be employed. Although, no one data model at the time of development, catered fully for the needs of this project, the FuGE (Functional Genomics Experiment) data model however was most suited to our needs as it was developed to facilitate convergence of data standards for high-throughput, comprehensive analyses in biology (Jones *et al.*, 2007). The FuGE data model, in essence provided a framework upon which custom designed sub-models could be built, specifically geared toward the various data types handled by the system. The FuGE object model, in its base classes Describable and Identifiable handle all aspects of the security and access to data. (Pizarro *et al.*, 2006)

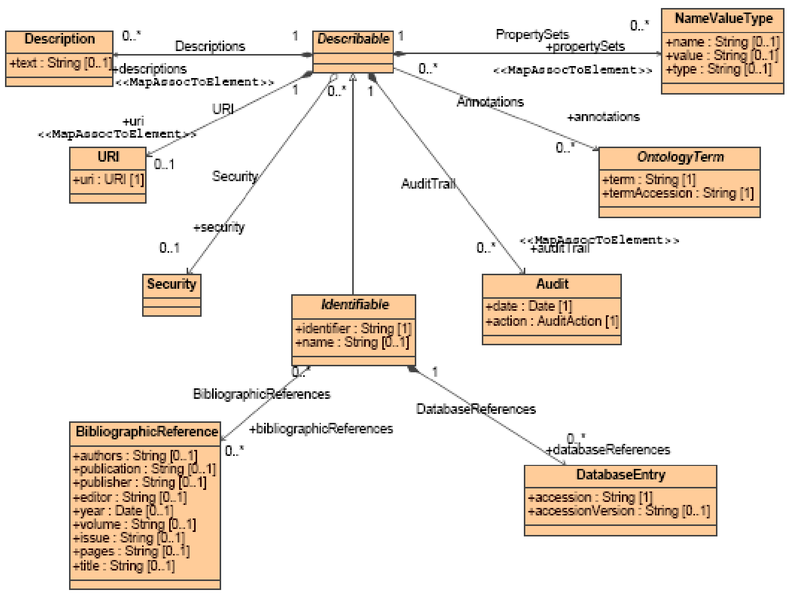


Figure 2.1: Main base classes within FuGE. Newly developed classes developed within FunGIMS inherit from these classes (Pizarro *et al.*, 2006).

By the process of polymorphic inheritance, all subsequent classes developed, inherited from the Identifiable and Describable base classes and thus inherited all their functionality. More of FuGE will be discussed later.

2.2.3 View

All views within FunGIMS were generated with the KID templating language and converted to HTML before being delivered to the client's browser. Due to FunGIMS being a collection of software modules, common themes and cascading style sheets (CSS) were employed by the various developers working on the various modules. This sharing of styles helped to maintain a common look and feel when within FunGIMS or any of the sub-modules. Style sharing was facilitated by each webpage inheriting from a single 'master.kid' file. Subsequent to inheritance, develops then added and customized their web pages to suit the data-type and context.

2.2.4 Controller

The FunGIMS controller is where all the functionality for the system is held. The root controller class contains all functions necessary to run the system. Functions declared within the controller are registered by the CherryPy server and request URLs (as well as parameters) from a clients browser are mapped to these functions. Turbogears makes extensive use of CherryPy which forms the controller layer of the FunGIMS system.

2.3 Examples of comparative genomics environments and what they have to offer

Several comparative genomics environments are available through the web and the variety of functionalities they offer are quite broad. One example of a contemporary comparative genomics environment is VISTA found at (http://genome.lbl.gov/vista/index.shtml). VISTA is a web-based comparative genomics environment offering a range of tools such as whole genome alignment wgVISTA), normal sequence comparison (mVISTA), and basic phylogenetic analyses (Phylo-VISTA). VISTA provides many great features but lacks the ability to store user data and user results generated by the software. Furthermore, the various tools runs from different servers therefore, there is no integration of the results with the different software components. On the whole, VISTA is a great comparative genomics environment offering most of the major tools that are required for comparative genomics studies. Sybil, another example of a comparative genomics environment is a web-based system allowing users to perform tasks such as genome browsing, genomic-region comparison, syntenic viewing (Figure 2.2) and a few others.

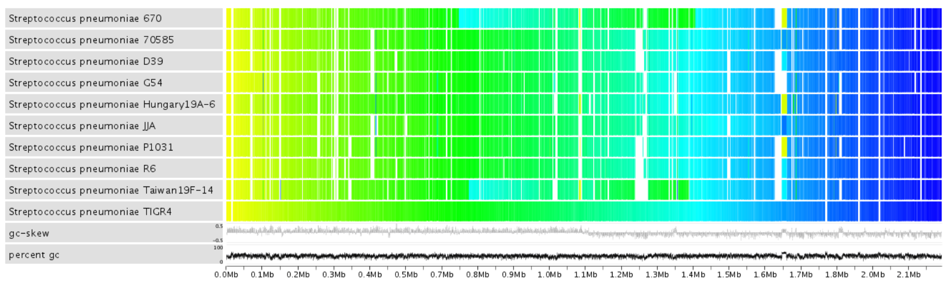

Streptococcus pneumoniae 670
Streptococcus pneumoniae 70585
Streptococcus pneumoniae D39
Streptococcus pneumoniae G54
Streptococcus pneumoniae Hungary19A-6
Streptococcus pneumoniae JJA
Streptococcus pneumoniae P1031
Streptococcus pneumoniae R6
Streptococcus pneumoniae Taiwan19F-14
Streptococcus pneumoniae TIGR4
gc-skew
0.5
-0.5
percent gc
100
0
0.0Mb 0.1Mb 0.2Mb 0.3Mb 0.4Mb 0.5Mb 0.7Mb 0.8Mb 0.9Mb 1.0Mb 1.1Mb 1.2Mb 1.3Mb 1.4Mb 1.5Mb 1.6Mb 1.7Mb 1.8Mb 2.0Mb 2.1Mb

Figure 2.2: Screenshot of Sybil's synteny gradient

Sybil offers 2D graphical displays of back-end pre-calculated comparative datasets however, the scope of its analytical software offering is very limited and users must rely on data that is pre-calculated on their system. The BioMAX knowledge management environment (www.biomax.com) is yet another example of a comparative environment. BioMAX system serves to integrate data from various sources (i.e. databases such as KEGG, BIND, DIP etc) by dynamically building semantic networks between the various data-types. BioMAX then allows the user to perform advanced queries on the underlying data in a very simple way. Users of BioMAX may create, manage and visualize scientific models as an extendible network of interrelated concepts. Through APIs, other tools such as R and Bioconductor scripts may be integrated into the system. BioMAX is quite an advanced distributed software platform allowing users to manage data from various sources and subsequently analyse the data based on the semantic networks built by BioMAX. This is very useful when dealing with matured data sources containing well annotated data. However, when it comes to new sequence data, and the fact many more organisms are being sequenced, often unknown, means that this system is not well suited due to the inadequacy of publicly available data.

2.4 Requirements

2.4.1 User interface requirements

At the heart of design of any software project is the requirements and skill level of the end user. The software in question must therefore, cater for all the requirements of the user as closely as possible and be suited to the users level of computing knowledge. In terms of requirements, this software needed to offer a unique range of tools dealing with comparative genomics. All tools needed to be integrated into one interface which would provide access to the various functions by the click of a button. The software had to allow for the tools to be used with uploaded data as well as data stored on the central system. Needless to say, the software had to therefore make provisions for the storage of user data on a central database. Storage of personal data on an external server also needs a security component to protect data privacy, thus the software needed a security component. Lastly, access to the software needed to be versatile and easy to maintain, therefore the entire software was created as a web based system affording users access from any computer with an internet connection. In terms of user skill level, the end users of this software include researchers and students alike. This group is mostly comprised of experimental based biologists who often do not have very advanced skills in terms of software and databse installation, software usage and data handling. This system had to therefore, abstract these components from their view and merely offer a very simplified, intuitive web-based interface to the full functionality of the back-end system. A simple login and password, set up by the administrator will allow any user to access the full functionality of the system, and access to their allowed sub-set of data. In summary, design of this software was closely guided by the abovementioned requirements of the users and as far as possible, usage of the system was intended to be as simple and intuitive.

2.4.2 Analysis Requirements

Prior to the actual addition of software into the system, careful consideration had to be given to to the types of analyses and scientific investigation typically undertaken by biological researchers in comparative genomics studies. Typically, researchers want to upload nucleotide and protein sequences and BLAST these against non-redundant databases in order to gauge taxonomy or identify their sequences. Alignment of two or more sequences (nucleotide or protein) in order to assess sequence similarity and differences is also a popular task. Downstream of this, users often want to examine single nucleotide polymorphisms between their sequences in order to make informed decisions regarding primer design, protein conformational changes and various other mutations. The construction of phylogenetic trees has become common place in assessing phylogenetic relationships amongst various sequences in question. Based on the researchers' need to perform these and other tasks, tools such as BLAST, MAFFT, ClustalW, Phyllip and BlastZ were integrated into the system. Integration, a key theme of this work has implications on several layers. In terms of software integration, this merely describes the centralized accessibility to the various tools and inter-compatibility of output and inputs from and to the various software. On the database level, integration points to the fact that various sub-models (sub-schemas) form part of the larger database model functioning in a compatible and complimentary manner, although each sub-model is capable of functioning exclusively. On the project level, the comparative genomics module is also integrated into the FunGIMS project as it subscribes to the technologies, database structure and coding standards employed by FunGIMS. Strong emphasis was placed on integration on all these levels throughout development of the system.

2.4.3 Data structure requirements

In the design of this data model, an extension of the Functional Genomics (FuGE) object model (Pizarro *et al.*, 2006) was employed with major adaptations (see later). A complete stand-alone solution, incorporating all biological data-types is an impossible task due to the vast array of experimental techniques and data-types that are currently and prospectively available. In light of this, the FuGE development team instead attempted to model only those aspects shared amongst the various functional genomics experiments such as researcher contact information, sample preparation and protocols. This was achieved in two ways. Firstly FuGE makes available a general database structure which enables for the referencing of external data formats which captures the meta-data which in turn provides context to a specific scientific investigation. Secondly, and most importantly, these abovementioned reference points acts as a start point for extensions to the FuGE model which translates to the defining of sub-models specific to any functional genomics technology (Pizarro *et al.*, 2006). In this way, the FuGE object model allows for limitless extension while maintaining a high level of integration and thus provided the perfect base to build on. Ease of data integration, with the help of FuGE thus became a reality, however, the data model developed for this system had to also take into account several key factors. Volume is one of the major concerns especially in the context of biological data. Biological experiments produce data at enormous quantities. Furthermore, with the latest DNA sequencing technologies being implemented globally, a concomitant rise in database growth is also expected. Biological sequence databases can range in volume from thousands to billions and even trillions of entries, thus the design of the database should be appropriately scalable. With databases of such large proportions comes the question of speed. The design of the database should take into account that every entry in the database may need to be searched upon at some stage, therefore this is another crucial aspect to bear in mind during the design. Data integration is also quite an important feature that needs to feature in this data structure. Due to the great heterogeneity in sequence data, such as annotations, various sequence types, genomic sequences, protein sequencing, referencing material and so forth, great care needs to be taken to make sure that the overall data model can appropriately handle the various types of sequence and meta-data. Furthermore, new types of technologies and sequence classes are being produced all the time, therefore the design of the system should be built with the prospect of extensibility to other future data types. Data security is extremely important in this day and age when academic publications depend on raw data being kept private. This system needs to be designed with great care being taken to protect visibility of individual as well as group data. The security classes controlling data visibility should make sure that only system users with the proper privileges may see certain data and the system data should also be totally protected from the individuals external to the system. Details regarding how these criteria were fulfilled in the design measure taken as well as the choice of technologies used at every stage will be covered in the design principles section next.

2.5 Design Principles

In the design of this software, several key principles were taken into account:

- The system must be web-based and intuitive to use
- The back-end data structures must be able to handle large amounts of data while not compromising on speed and agility in the integration of the various types.
- The software should take care of data security precluding wrongful visibility of data.
- Lastly, the software must offer the range of comparative genomics software tools providing a range of useful functionalities from a single interface.

2.5.1 User interface requirements

Principles used in the design of the interface for this software were largely based on the user requirements set out. Therefore, first and foremost, the web-based interface was designed so that usage would be very simple and efficient. This meant that there would not be the addition of too many icons and menus on the screen to inundate the user. Rather, fewer screen items were added and navigation to tools and user data was made possible in a maximum of two mouse clicks. Soft colors were chosen in the design of the interface so as to not be irritable to the eyes after many hours of usage. The inclusion of the software tools to this system was based on what were the major and most common tools used by researchers. Furthermore our system was to offer two unique tools in addition to the commonly available ones which were, the Seqword Genome Browser as well as the Mycobacterial comparison analyses suite. Details of the design and implementation follows.

2.5.2 Data structure requirements

Design of the data structures to handle the back-end data was largely influenced by the data requirements. Three main design aspects needed to be taken into account. Firstly, the data structure needed to be highly scalable and should be able to perform optimally even after being populated with millions of entries. Secondly, the data structure needed to cater for several different data types while maintaining a high degree of integrity. Thirdly, database security was a crucial aspect and data privacy was therefore crucial in the design of the database schema. Lastly, provision needed to be made for the addition of new data types if the need did arise, thus, the database model was designed in a manner that was conducive to extension to other data types.

2.5.3 Software components and technologies employed

The software and technologies employed for this project are outlined below.

- A MySQL database
- The Turbogears web-development toolkit
- Several open-source biological analyses and graphical software tools (example BlastZ, Phylip, Laj and Clustal)
- Several programming languages (Python, BioPython, Javascript, HTML, KID)
- Integrated development environment (IDE)
- SQLAlchemy and SQLObject
- XML-RPC
- AJAX

Details regarding how each of these components was integrated will be discussed in the next section.

2.6 Model-View-Controller Architecture and integration

2.6.1 Model-View-Controller Pattern

The model-view-controller (MVC) pattern is a concept in software development that can be implemented as both a design pattern as well as an architectural pattern. The MVC concept basically aims to separate an application into tiers or layers viz the model, view and controller layers such each of the layers can be independently modified without adversely affecting the other layers. The model layer represents the actual data or content as well as the rules governing how the data is managed. The view is essentially the component that is involved with presentation of the data in the model such as the text, checkboxes, buttons etc. The controller is the business layer of the system and handles all the logic of the system as well as communication between the view and model. This is comprised of the actual programming code. Successful implementation of the MVC pattern means that the user interface or the business logic layer can be modified independently while still maintaining system integrity. Using the MVC design pattern also allows several views of same underlying model. One of the simplest examples demonstrating MVC design pattern is found in a browser. Where the model is represented by the content (HTML), the view is represented by the CSS as it dictates how the data will be presented and finally the controller is represented by the controller as this controls communication between the content (model) and the css (view) and controls which data items will be displayed. This same MVC design pattern was employed for this project and details of its implementation will herewith be discussed.

2.6.2 Integration of the various components under the M-V-C design pattern

For this project, Turbogears was the python based web-development framework used. Turbogears is designed around the MVC paradigm and thus allows for separation of the various MVC components. For each of the various MVC components, the strategies, software and technologies used in the context of this project will now be discussed.

2.6.2.1 The Model Layer

The model layer represents the actual database, the data and the classes defining the various object types. In this layer the following technologies and software were used

- MySQL (Database server)
- SQLAlchemy (Object relational mapper)
- SQLObject (Object relational mapper)

MySQL, a robust, fast, reliable and highly scalable database server with a proven track record was used for data storage (6). MySQL comes in open-source database versions and also satisfied all the design criteria set-out above and was thus the database of choice. MySQL uses SQL (Structured Query Language) to communicate with other programs. MySQL also has its own set of augmented SQL commands which offer users more advanced and specific functionality. Communication with the database and its data was accomplished by using the object relational mappers (ORM) SQLObject and SQLAlchemy. SQLObject is an extremely popular and well established object relational manager that allows a user to interface with databases via objects. SQLObject treats tables as classes, rows as instances and table columns as class attributes. SQLObject also includes a python-object based query language that allows higher level usage of SQL thus providing users with more versatile database interfacing and a greater independence from actual SQL code (7). SQLAlchemy is another SQL toolkit and ORM for the python programming language. SQLAlchemy is quite a mature ORM offering very efficient and high-powered database access. This high-powered access is partly due to the fact that SQLAlchemy does not view database tables as mere tables but rather as 'relational algebra engines'. SQLAlchemy allows mapping of classes against databases in several ways thus allowing many useful and powerful features such as cascading, complex selects, complex joins, sub-queries unions and many others (8). Both SQLObject and SQLAlchemy were used in this project as they both provided abstract access to the database in a python friendly manner while still offering all the power and functionality of SQL from within the Turbogears framework.

2.6.2.2 The View Layer

The view layer is essentially the user interface and it is what the user interacts with. This includes the web browser, buttons, etc. In this layer, the following technologies were employed

- Kid templating
- Javascript & Mochikit

All the above-mentioned items were used to create graphical user interfaces for the underlying data model. Users can point their web browser to the correct url and thus interact with the underlying data model via the controller. Kid is essentially a template generation engine that is used for XML compatible vocabularies written in the python programming language (9). Within the Turbogears framework, Kid is extremely useful for several reasons. It allows one to incorporate python into the actual templates. It allows inheritance, thus many pages can display from one template by merely including the appropriate XML tags. Kid also forces users to adhere to valid XML standards thus precluding syntax errors such as mismatched tags within the template. The kid templates produced within Turbogears then gets converted to XHTML and along with the data, is delivered to the client (i.e a browser) and displayed. Also involved with the view layer is Javascript and Mochikit. Javascript is a popular scripting language that is used in client-side web development. Javascript is useful in that it makes the interface more interactive. Also, due to the fact that it runs on the client side (i.e in the browser) it reacts very quickly to users inputs. It is used in this project for quick validation of user input on the web-page as well as for inter-communication with DOM components within the web-page. Mochikit, is essentially a lightweight javascript library that adds python-like features to javascript. Both javascript and mochikit are highly compatible with the Turbogears framework and both contributed substantially to the creation of a user friendly view.

2.6.2.3 The Controller Layer

The controller layer is the most computationally intensive and complex layer. This layer is typically responsible for receiving and responding to requests made by the user and also for communication and modification of the underlying data model. The main technologies used here are the CherryPy server and the python programming language. CherryPy is an object oriented web-application framework that uses the python programming language (10). CherryPy was designed with the aim of facilitating rapid development of web based applications by wrapping the HTTP protocol. Due to CherryPy being very pythonic in nature, CherryPy may be used as a regular python module. Turbogears makes extensive use of CherryPy to map user request URLs and parameters to python functions. Python functions are written within Turbogears and handles data and user requests in specific ways. The results are then passed back to the user/client via CherryPy as a server response. All the analyses functionalities offered by the system are essentially python functions contained within the Turbogears framework and handled by the CherryPy module. Python is a dynamic, object-oriented programming language (11) and it can be used as a simple scripting language or for the development of high powered web-development frameworks such as Turbogears. Due to its ease of use and versatility, it was chosen for the development of this project. The various aspects of the MVC design pattern has been explained, the following figure aims to summarize the MVC design through interaction of the various technologies within Turbogears.

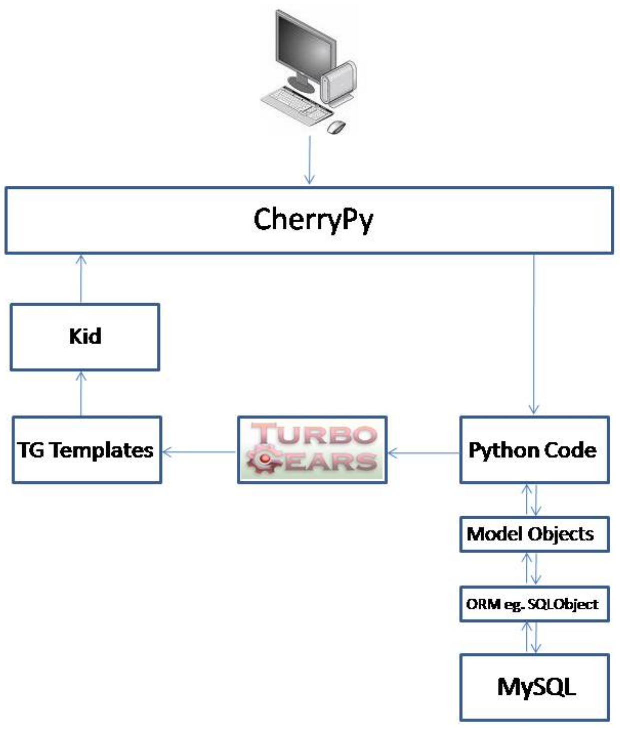


Figure 2.3: Figure illustrating the MVC design pattern in the context of Turbogears. Numbers represent the order of events subsequent to a user making a server request from the browser.

Using Turbogears and its MVC architecture, an integrated web-based comparative genomics analysis suite was developed. The analysis functions offered by the system are very varied but are all integrated into one web-based environment. The general system implementation will now be discussed.

2.7 Technical implementation details

This section will deal with the technical details of the project implementation in the context of the MVC design pattern that was previously explained. For every MVC layer, an explanation will be given regarding the software and technologies and how exactly they were used and integrated.

2.7.1 Database implementation

This essentially represents the model layer. MySQL was the database server used and all data in the system was stored in specific tables defined, created and populated using the SQLAlchemy ORM. The actual database structure was based on the FuGE model however, the structure was substantially adapted in order to deal with all the unique data types within the system. The basic class of the model was the Identifiable class and all datatypes inherit from this class (Figure 2.4).

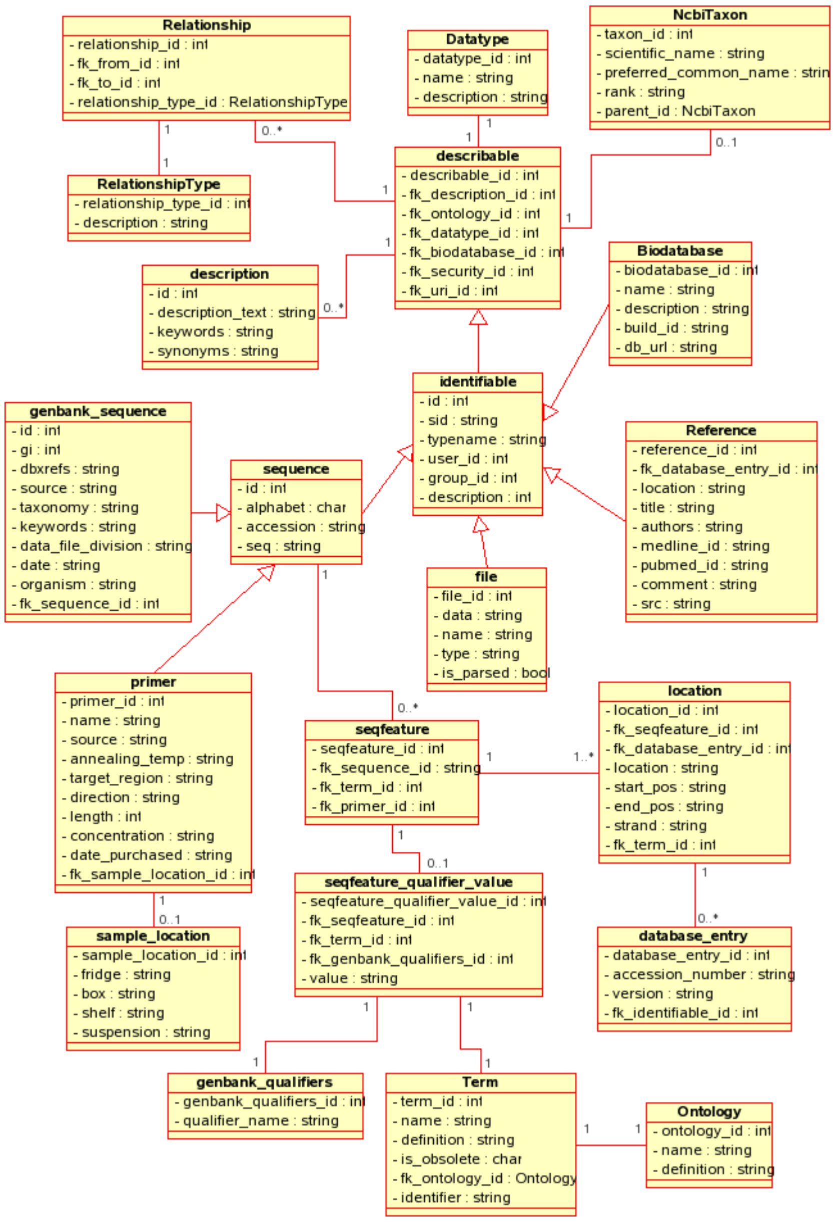
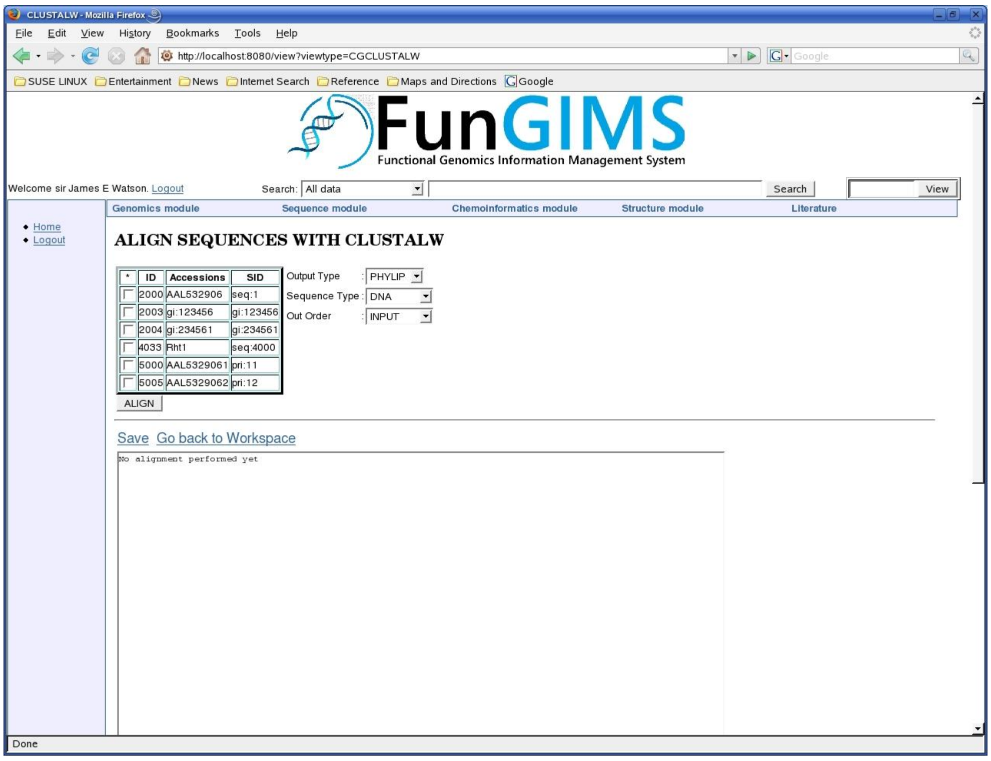


Figure 2.4: UML class diagram showing some of the major classes used in the database and the relationships between them.

Using eric3 as the Integrated Development Environment (IDE), all SQLAlchemy classes, tables and mappers were defined in the 'model.py' file under a Turbogears project. Turbogears then offers a command-line utility 'tg-admin sql create' which creates all the tables in MySQL and maps them together according to the definitions laid out in the model.py file. SQLAlchemy was used extensively because it also allows one to perform basic and complex MySQL commands from within the controller (more about this to follow). SQLAlchemy connects to the database via drivers configured in the 'dev.cfg' file also located under the Turbogears project folder.

2.7.2 Graphical User Interface (GUI)

The GUI is essentially all the user sees and interacts with and thus represents the view layer. In Turbogears, views are generated using the KID templating engine which generates HTML, recognized by all browsers. Using Eric3 again, the KID templating language was used to script HTML and Javascript to define how the data and the HTML page elements would be displayed. KID eventually generates HTML which is delivered to the browser by the server (i.e CherryPy). Users may through the use of this HTML rendered by web browser such as Firefox, Konqueror or Mozilla interact with the underlying system or in other words, communicate with the controller (which in turn communicates with the model). This interaction includes simple visualization of their data but also manipulation of their data and other requests. Also, the view is responsible for displaying of data. A simple scenario is as follows. A user navigates to his data storage page and then wants to delete a specific entry. This is made possible by a form on the page. The user then clicks on the "delete entry" button thus invoking a request to the controller which then interprets the request and performs the action on the database with the help of the ORM (i.e SQLAlchemy). All requests are handled by the CherryPy server (see later). Also involved in the view is Javascript. Javascript in this project was used for various purposes such as for notification to the users, user-input validation and for inter-communication of the data within the page (Figure 2.5).


CLUSTALW - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/view?viewtype=CGCLUSTALW
SUSE LINUX Entertainment News Internet Search Reference Maps and Directions Google
FunGIMS
Functional Genomics Information Management System
Welcome sir James E Watson. Logout
Search: All data
Search
View
Genomics module
Sequence module
Chemoinformatics module
Structure module
Literature
Home
Logout
ALIGN
Save Go back to Workspace
Done

ALIGN SEQUENCES WITH CLUSTALW

*	ID	Accessions	SID
☐	2000	AAL532906	seq:1
☐	2003	gi:123456	gi:123456
☐	2004	gi:234561	gi:234561
☐	4033	Rht1	seq:4000
☐	5000	AAL5329061	pri:11
☐	5005	AAL5329062	pri:12

Output Type : PHYLIP
Sequence Type : DNA
Out Order : INPUT

```
No alignment performed yet
```

Figure 2.5: An example of a typical view that a user sees. Everything visualized on the page is essentially HTML generated by KID. The 'ALIGN' button is the users way of communicating with the controller and in-turn, the underlying data. Javascript is responsible for user-input validation.

The above figure (Figure 2.5) is an example of a typical view a user will be faced with. The 'ALIGN' button is provided so that the user may communicate with his data via the server (controller). Javascript, used for input validation will in this case, check that the user first selected several sequences before allowing the request to be sent to the server. Checking request submission in this way is advantageous in that it precludes invalid requests from being submitted and unnecessarily utilizing the server. In the project, the view was designed to be intuitive and user-friendly. These two attributes are important for success of software. The colors chosen were soft and easy on the eyes. Drop-down menus were added to facilitate easy navigation and cluttered screens which often overwhelms users, were avoided. As already mentioned previously, the view is a means for a user to communicate with the controller. The controller, which is by far the 'nerve center' of the project will now be dealt with.

2.7.3 The Controller

The controller layer is the most computationally intensive layer and it controls the model layer beneath it, the view layer above it, communicates with 3rd party software and is responsible for sometimes carrying out analyses functions. Within the Turbogears framework, the analyses functions and all instructions for the system, correspond to methods contained within the main class which is the 'Root Controller'. The Root controller and its methods are found in the 'controllers.py' file. When a user makes a request to the server via the view, CherryPy maps this URL and the parameters to a specific function contained within the root controller. The function within the root class is then executed either by CherryPy itself or instructions are handed to a third party software via an extended-mark-up-language – remote procedure call or XML-RPC. Remote procedure calls are essential in cases when third-party software or data sources are located on separate computers. The controller class lastly, receives results or data from the analyses software or databases respectively and decides on which view to invoke. In order to elucidate the functioning of the controller, a typical example of a request/response model will now be given continuing with the 'delete entry' example used above. From the view, the user will make a request to delete an entry by clicking a button. CherryPy then receives this request (along with the parameters such as the ID of the entry to be deleted) and maps it to the correct 'delete entry' method contained within the root controller. The method, with the aid of the ORM (e.g SQLAlchemy) then communicates directly with the database to delete the user specified entry. SQLAlchemy first compiles the command into a MySQL friendly format and issues this to the database. The ORM then receives communication from MySQL detailing whether the 'delete entry' command was successful or not and passes this back to CherryPy. CherryPy via the same method that was first invoked, then decides which KID template and data to display. CherryPy , now generates the HTML from the specifications within the KID template and passes this via the HTTP protocol back to the browser which is then visible to the user. Needless to say, the controller (represented by CherryPy) shoulders a lot of the communication and computational burden involved with interaction of the system. It is responsible to relay messages to and from the user to the server and for the relay of messages to and from the model as well as deciding which views are to be passed back to the browser (client). The above example outlines much of the technical details involved in this system. In the next section, however, a more high level, user centered view will be used to explain the implementation system in terms of the general CG tools offered by the system.

2.8 Implementation of a general comparative genomics environment

Although there are a multitude of analyses tools already available on the web, it is very useful and efficient to have a centralized server environment where users may have access to various tools.

This was the idea behind the development of this project. Addition of all available CG tools into one analysis suite is unrealistic, therefore a few key analyses tools were chosen for inclusion into the project. The sub-set of analysis functions chosen would be sufficient in meeting the basic needs of CG research, however, novel tools were also added (details of these will be dealt with later). The basic set of analysis tools included within the project are :

- DNA sequence alignment (MAFFT and Clustal)
- Genome Alignment (BlastZ)
- Phylogeny analysis (Phylip)

In order to upload the users data into system so that it conforms to the standards of the model. The data was first passed through the relevant analyses tools. The output from these would then be parsed by various python scripts which then subsequently inserted the formatted data into the relevant tables.

2.8.1 DNA sequence alignment

The alignment of nucleic acid sequences is one of the most basic and popular bioinformatics analyses being performed and was thus included in the system. Once logged into the system, a user may choose to perform sequence alignments by selecting the appropriate drop-down menus.

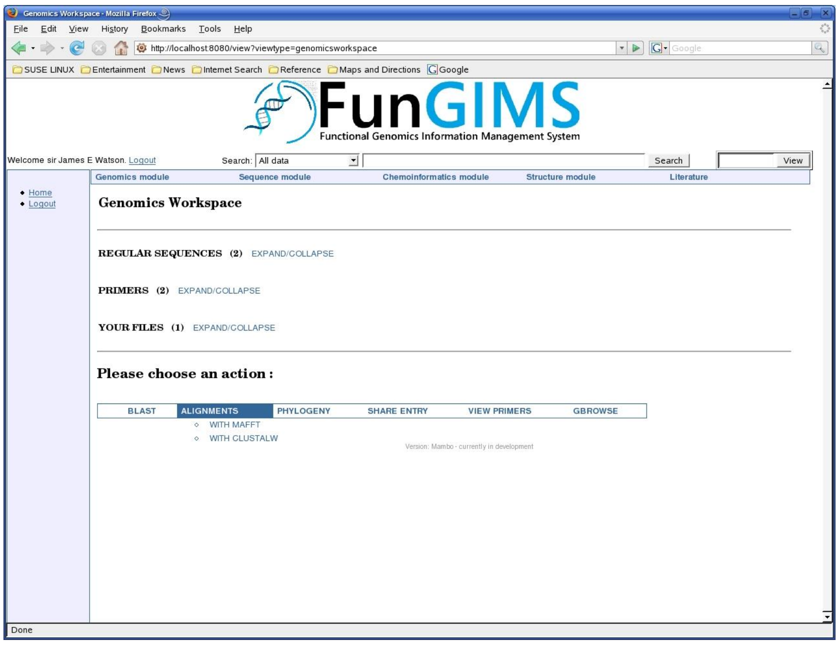
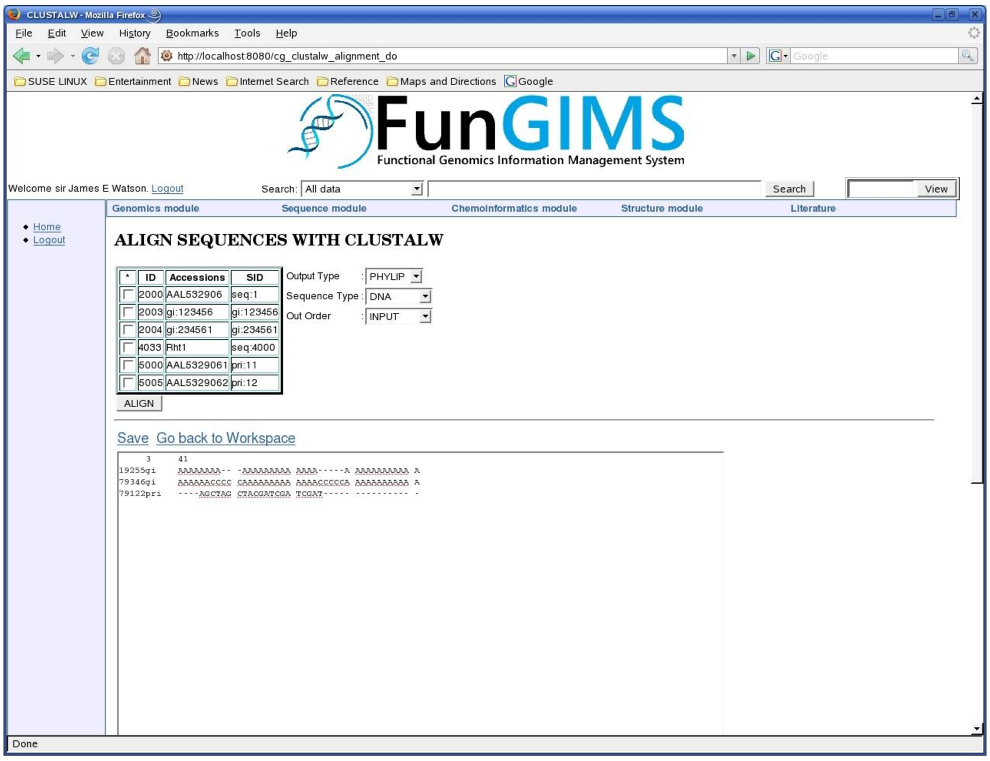

Genomics Workspace - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/view?viewtype=genomicsworkspace
SUSE LINUX Entertainment News Internet Search Reference Maps and Directions Google
FunGIMS
Functional Genomics Information Management System
Welcome sir James E Watson. Logout
Search: All data
Search
View
Genomics module
Sequence module
Chemoinformatics module
Structure module
Literature
Home
Logout
Genomics Workspace
REGULAR SEQUENCES (2) EXPAND/COLLAPSE
PRIMERS (2) EXPAND/COLLAPSE
YOUR FILES (1) EXPAND/COLLAPSE
Please choose an action :
BLAST
ALIGNMENTS
PHYLOGENY
SHARE ENTRY
VIEW PRIMERS
GBROWSE
WITH MAFFT
WITH CLUSTALW
Version: Mambo - currently in development
Done

Figure 2.6: All functionality within the software suite is accessible via either the main-menu dropdown at the top of the screen or through the sub-menu at the botton of the screen.

A user may either choose to perform MAFFT alignments or ClustalW alignments. Once on the subsequent page, all the sequence data the current user has privileges to is displayed on the top of the screen (Figure 2.7). The user can then select a set of sequences that he/she wishes to align and also select certain available output format options.


CLUSTALW - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/cg_clustalw_alignment_do
SUSE LINUX Entertainment News Internet Search Reference Maps and Directions Google
FunGIMS
Functional Genomics Information Management System
Welcome sir James E Watson. Logout
Search: All data
Search
View
Genomics module
Sequence module
Chemoinformatics module
Structure module
Literature
Home
Logout
ALIGN
Save
Go back to Workspace
Done

ALIGN SEQUENCES WITH CLUSTALW

*	ID	Accessions	SID
☐	2000	AAL532906	seq:1
☐	2003	gi:123456	gi:123456
☐	2004	gi:234561	gi:234561
☐	4033	Rht1	seq:4000
☐	5000	AAL5329061	pri:11
☐	5005	AAL5329062	pri:12

Output Type : PHYLIP
Sequence Type : DNA
Out Order : INPUT

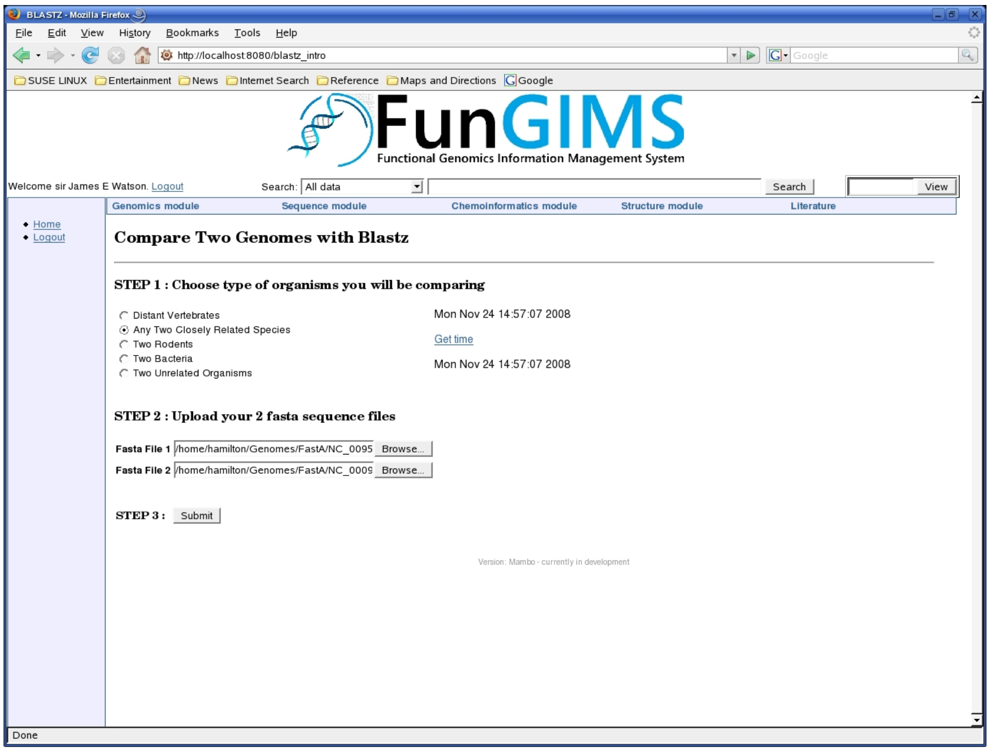
```
     3     41
19255gi    AAAAAAAA-- -AAAAAAAAA AAAA-----A AAAAAAAAAA A
79346gi    AAAAAACCCC CAAAAAAAAA AAAACCCCCA AAAAAAAAAA A
79122pri   ----AGCTAG CTACGATCGA TCGAT----- ---------- -
```

Figure 2.7: Sample page showing the ClustalW alignment results.

Note that, when users submit their requests, in page error checking is handled by Javascript and post-request errors are handled by the controller.

2.8.2 Genome alignment with BlastZ

Genome alignment using BlastZ may be performed by a simple three step procedure. From the main job submission page, users may upload two genome sequences at a time for alignment using BlastZ and then choose their comparison type (Figure 2.8).


BLASTZ - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/blastz_intro
SUSE LINUX Entertainment News Internet Search Reference Maps and Directions Google
FunGIMS
Functional Genomics Information Management System
Search
View
Genomics module Sequence module Chemoinformatics module Structure module Literature
Home
Logout
Get time
Browse...
Submit
Done

Welcome sir James E Watson. Logout

Search: All data

Compare Two Genomes with Blastz

STEP 1 : Choose type of organisms you will be comparing

Distant Vertebrates [✓ Any Two Closely Related Species] Two Rodents Two Bacteria Two Unrelated Organisms

Mon Nov 24 14:57:07 2008

Mon Nov 24 14:57:07 2008

STEP 2 : Upload your 2 fasta sequence files

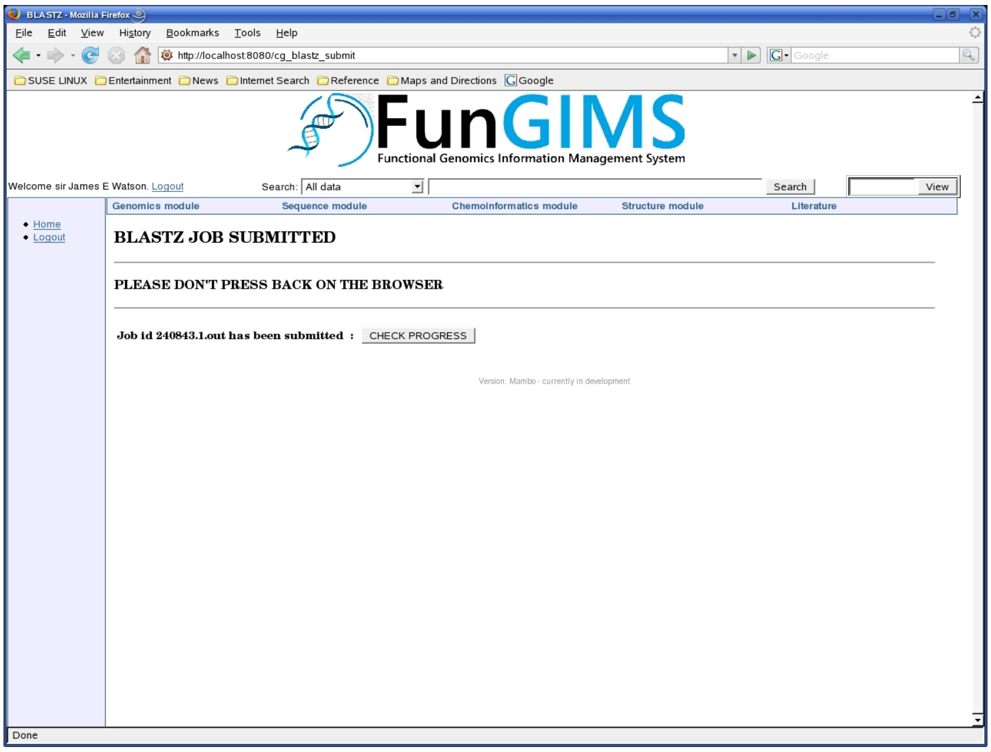
Fasta File 1	/home/hamilton/Genomes/FastA/NC_0095
Fasta File 2	/home/hamilton/Genomes/FastA/NC_0009

STEP 3 :

Version: Mambo - currently in development

Figure 2.8: Main BlastZ submission page for the alignment of whole genomes.

Owing to the fact that genome sequence alignments are often time consuming, pages subsequent to job submission allow the user to check the status of the job at anytime (Figure 2.9). If the submitted job is unsuccessful due to problems such as inappropriate file formats and such, an error will be thrown and the job discarded. Users are then re-directed back to the main submission page (Figure 2.8).



BLASTZ JOB SUBMITTED

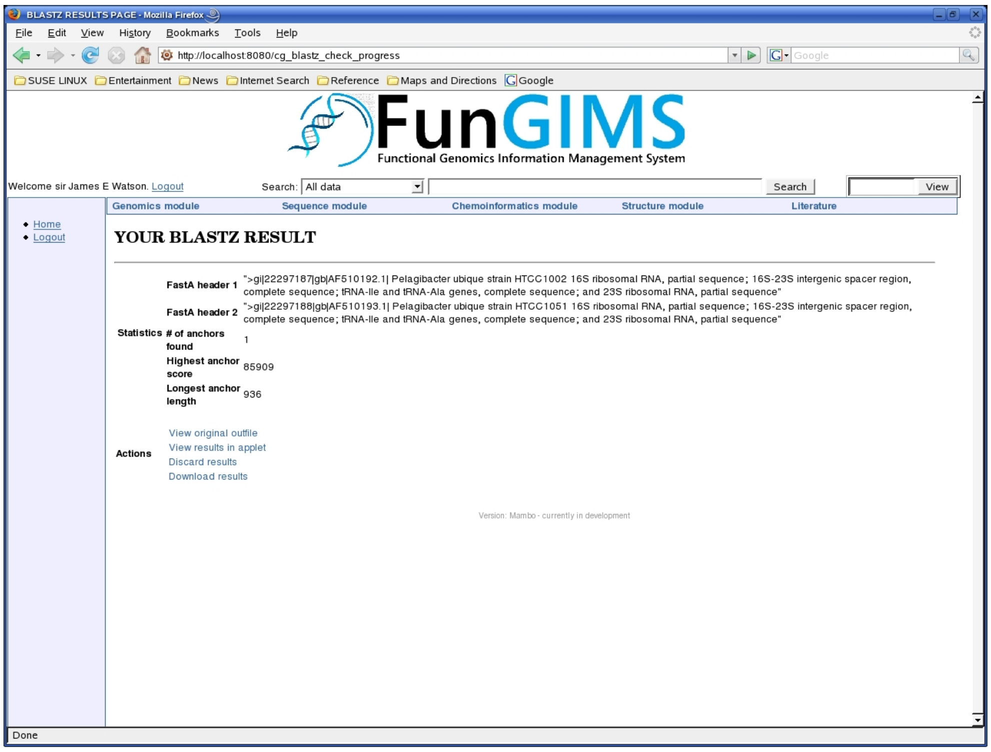
PLEASE DON'T PRESS BACK ON THE BROWSER

Job id 240843.1.out has been submitted :

Version: Mambo - currently in development

Figure 2.9: A successful BlastZ job submission will direct users to this page. Here, users may check the progress of their jobs by clicking the 'CHECK PROGRESS' button.

If the job, however, is successful the user is then presented with the result page where they are presented with a quick summary of the alignment results and the options to view their results in two ways. The main BlastZ result page (Figure 2.10) allows users to view the alignment result file either as plain text or graphically via the Laj applet (Figure 2.11). In terms of plain text results, clicking on the 'View original outfile' item will invoke a text box containing the raw text of the alignment file which a user may peruse. This file the user may download by clicking on the 'Download results' item. In terms of graphic results display, the Laj applet merely allows a user to view an interactive dot plot of their alignment results. The applet is easily viewed by clicking on the 'View results in applet' option. Laj also allows users to zoom into the actual alignment where they can explore sequence similarity on the nucleotide level.


BLASTZ RESULTS PAGE - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/cg_blastz_check_progress
Google
SUSE LINUX
Entertainment
News
Internet Search
Reference
Maps and Directions
Google
FunGIMS
Functional Genomics Information Management System
Welcome sir James E Watson. Logout
Search: All data
Search
View
Genomics module
Sequence module
Chemoinformatics module
Structure module
Literature
Home
Logout
Done

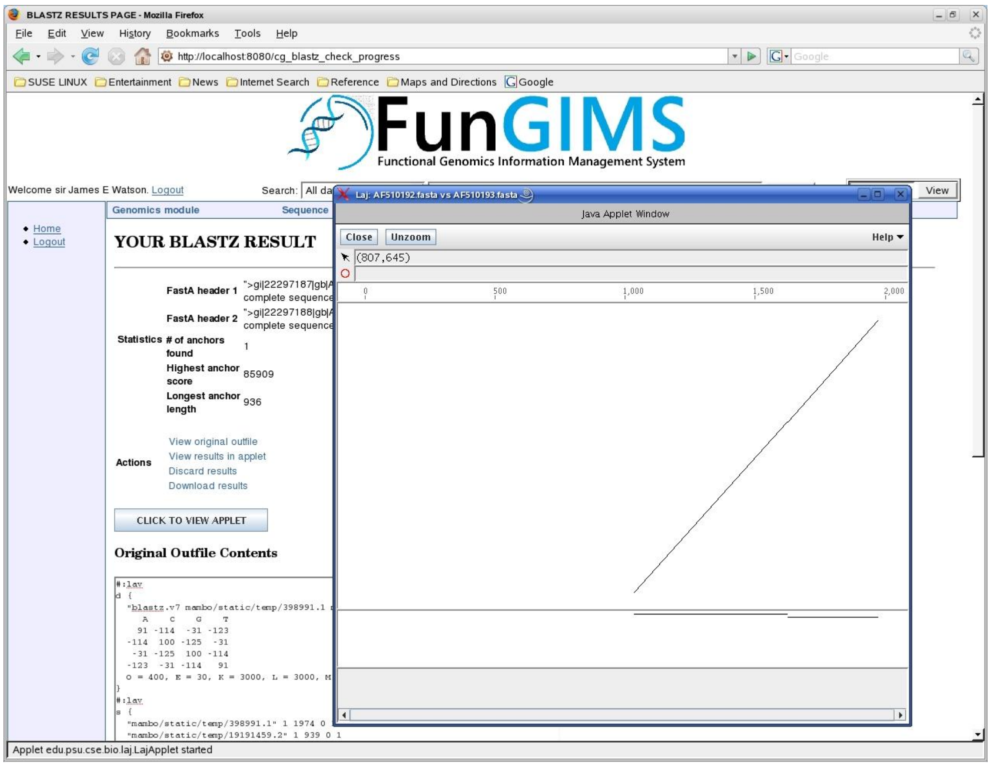
YOUR BLASTZ RESULT

	FastA header 1	">gi|22297187|gb|AF510192.1| Pelagibacter ubique strain HTCC1002 16S ribosomal RNA, partial sequence; 16S-23S intergenic spacer region, complete sequence; tRNA-Ile and tRNA-Ala genes, complete sequence; and 23S ribosomal RNA, partial sequence"
	FastA header 2	">gi|22297188|gb|AF510193.1| Pelagibacter ubique strain HTCC1051 16S ribosomal RNA, partial sequence; 16S-23S intergenic spacer region, complete sequence; tRNA-Ile and tRNA-Ala genes, complete sequence; and 23S ribosomal RNA, partial sequence"
Statistics	# of anchors found	1
	Highest anchor score	85909
	Longest anchor length	936

Actions
- View original outfile
- View results in applet
- Discard results
- Download results

Version: Mambo - currently in development

Figure 2.10: Main result page of a BlastZ submission.


BLASTZ RESULTS PAGE - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/cg_blastz_check_progress
SUSE LINUX Entertainment News Internet Search Reference Maps and Directions Google
FunGIMS
Functional Genomics Information Management System
Laj: AF510192.fasta vs AF510193.fasta
Java Applet Window
Close
Unzoom
Help
(807,645)
0
500
1,000
1,500
2,000
View

Welcome sir James E Watson. Logout

Search: All da

Genomics module Sequence

- Home
- Logout

YOUR BLASTZ RESULT

	FastA header 1	">gi|22297187|gb|A complete sequence
	FastA header 2	">gi|22297188|gb|A complete sequence
Statistics	# of anchors found	1
	Highest anchor score	85909
	Longest anchor length	936

Actions
View original outfile
View results in applet
Discard results
Download results

CLICK TO VIEW APPLET

Original Outfile Contents

```
#:lav
d {
  "blastz.v7 mambo/static/temp/398991.1 
     A    C    G    T
    91 -114  -31 -123
  -114  100 -125  -31
   -31 -125  100 -114
  -123  -31 -114   91
  O = 400, E = 30, K = 3000, L = 3000, M
}
#:lav
s {
  "mambo/static/temp/398991.1" 1 1974 0 
  "mambo/static/temp/19191459.2" 1 939 0 1
```

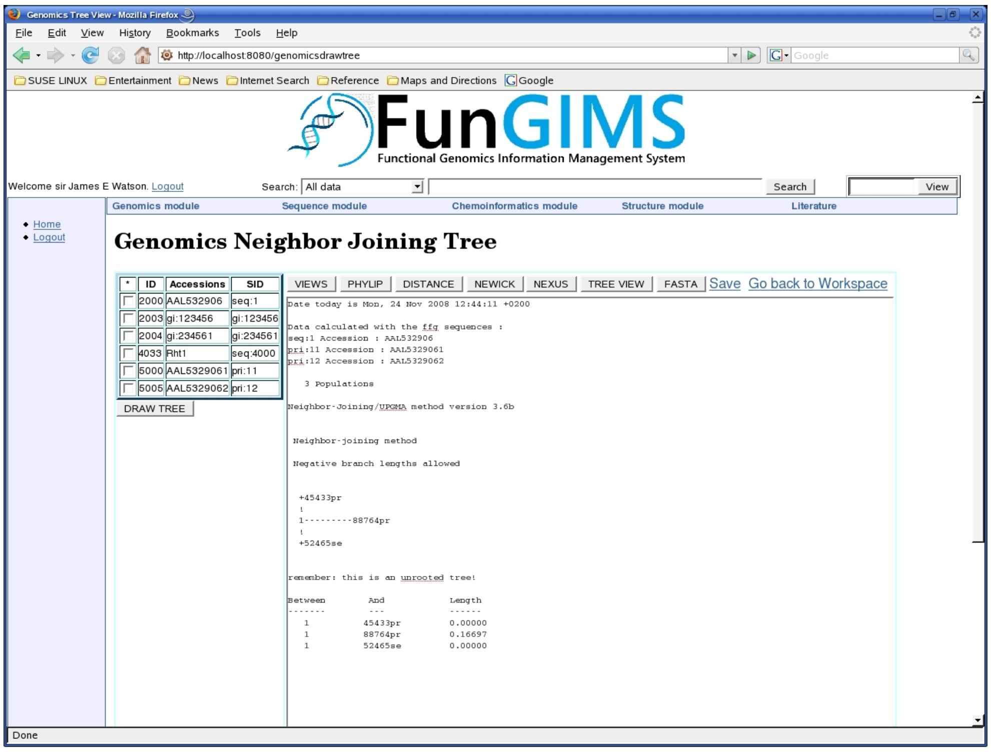
Applet edu.psu.cse.bio.laj.LajApplet started

Figure 2.11: Graphical display of alignment results using the Laj applet.

Once a user is satisfied with his results, he or she may choose to download the results onto their local machine. If a user then exits the main result page, the result files are automatically deleted to save space on the server. Next, a brief overview of the phylogeny module will be presented.

2.8.3 Phylogeny analyses

The comparative genomics module also enables quick and easy neighbor-joining tree generation based on user uploaded sequences. On entering the main 'Genomics neighbor-joining tree' page, the user is presented with a list of genomic sequences that he or she has access to (Figure 2.12).



*	ID	Accessions	SID
☐	2000	AAL532906	seq:1
☐	2003	gi:123456	gi:123456
☐	2004	gi:234561	gi:234561
☐	4033	Rht1	seq:4000
☐	5000	AAL5329061	pri:11
☐	5005	AAL5329062	pri:12

```
Date today is Mon, 24 Nov 2008 12:44:11 +0200

Data calculated with the ffg sequences :
seq:1 Accession : AAL532906
pri:11 Accession : AAL5329061
pri:12 Accession : AAL5329062

   3 Populations

Neighbor-Joining/UPGMA method version 3.6b


 Neighbor-joining method

 Negative branch lengths allowed


  +45433pr
  !
  1---------88764pr
  !
  +52465se


remember: this is an unrooted tree!

Between        And            Length
-------        ---            ------
   1          45433pr         0.00000
   1          88764pr         0.16697
   1          52465se         0.00000
```

Figure 2.12: Neighbor-joining tree result page.

The user selects two or more sequences which they wish to align and then clicks ‘DRAW TREE’. The server then responds by retrieving the actual sequences, performs clustalw alignments, calculates the distance matrix and lastly plots the tree. The text form on the tree is then displayed in a text box. When performing the above step, all default values are assumed as the output is meant to give users an overview of tree topology. Bootstrapping, though not a function in the current version of the software, may be added in subsequent releases. A user also has the option of viewing the tree file in one of several formats including nexus, newick or phylip format for example by simply clicking on one of the buttons above the text box (Figure 2.12). The program used to generate the matrix as well as produce the tree is phylip. The user also has the option of the saving their results onto their local P.C for the purposes of carrying out further analyses or viewing with other programs. This sub-module of the CG module serves to demonstrate the ease at which data and software can be centralized to perform basic analyses tasks.

2.9 Conclusion

Usability, convenience and relevance are crucial aspects to consider in software design. For this project, specifics needs by biologists which is namely, easy access to web tools and data security were identified and guided the design of this piece of software which harmoniously incorporated all the crucial design aspects. FunGIMS, the general management system caters for a wide audience, offering everything from protein sequence analyses functions to chemi-informatics functions. FunGIMS, is a good example of how several types of software and data types can be integrated into one system while still maintaining user friendliness, convenience and relevance. Based on the successes of FunGIMS and using its development framework the comparative genomics sub-module was developed which offers users access to many useful tools pertinent to comparative genomics such as genome alignment and visualization, phylogenetic functions and others. All software and their results produced are seamlessly integrated into this single environment thus allowing researchers to more rapidly assimilate their data. This system serves as an example of how software and data of various types can be integrated 'under one roof' allowing users access to their data under various contexts (sub-modules) while still in a secure environment. The web-access to the system adds a further dimension of convenience permitting users to not only perform their research tasks from any location but also at any time. Some of the shortcomings of the system include performance, coding standards and documentation. Due to the large of amount of underlying relationships that exist between the datatypes, performance, in terms of speed, is sometimes compromised. During development of this system, time was crucial hence, coding standards did not always conform to best practices. This may hamper extensions to the system. Also, not not much emphasis was placed on proper documentation throughout the project and again, this could become problematic when extensions to the system need to be designed. It is hoped that these, and other shortcomings will be addressed as further development of the project continues. Further development of the system will see the addition of new functionalities. In terms of new functionalities, perhaps a high-throughput analyses pipeline may prove a useful addition as well as a full micro-array analyses module.