Enhancing Collaboration by Providing a Shared Environment in wEMBOSS

A Thesis
Presented to
The Faculty of Computer Science Department
University of Missouri-Columbia

In Partial Fulfillment
of the Requirements for the Degree

Master of Science

By
Prathyusha Muddasani

Dr. Gordon K. Springer, Thesis Supervisor

July 2011

The undersigned, appointed by the Dean of the Graduate School, have examined the project entitled

Enhancing Collaboration by Providing a Shared Environment in wEMBOSS

Presented by

Prathyusha Muddasani

A candidate for the degree of

Master of Science

And hereby certify that in their opinion it is worthy of acceptance.

Dr. Gordon K. Springer

Dr. William Harrison

Dr. Jianlin Cheng

Acknowledgements

All praise to the Almighty God who has given me knowledge, patience, and perseverance to finish my Master's Thesis.

Apart from the efforts of me, the success of this project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I would like to convey my greatest appreciation to my advisor, Dr. Gordon Springer. I can't say thank you enough for his tremendous support and help. I feel motivated and encouraged every time I attend his meeting. Without his encouragement and guidance this project would not have materialized.

Special thanks go to Dr. William Harrison and Dr. Jianlin Cheng for the invaluable help and suggestions. I appreciate their time and efforts in serving my thesis committee.

Most of all, I am greatly indebted to my parents, Srinath Reddy and Padmaja Reddy, for their blessings, encouragement and support at every step of my life. If not for the inspiration given by them I would not have gained success. Words are inadequate in offering thanks to my husband, Bharathwajan Narayanan, for his tolerance levels in reviewing my drafts though not his subject.

Finally, yet importantly, I would like to express my heartfelt thanks to my sister, my cousins and my friends for their help and wishes for the successful completion of this project.

Enhancing Collaboration by Providing a Shared Environment in wEMBOSS

Prathyusha Muddasani
Dr. Gordon K. Springer, Thesis Advisor

Abstract

wEMBOSS, a web interface for EMBOSS suite of programs, is a powerful tool developed to serve the needs of molecular biology user community. It started as a coordinated effort from *Martin Sarachu* of the Argentinean EMBnet Node and *Marc Colet* from the Belgian EMBnet node.

Exchange of thoughts brings new ideas into life. Collaboration is a process where two or more people work together to achieve a common goal. Collaboration is especially important to researchers in obtaining better results. wEMBOSS being a platform for research, the idea of introducing a collaborative environment into wEMBOSS seems worth considering. The current project deals with implementing a shared environment in wEMBOSS by introducing the concept of groups and sharing. Users can create groups, add members to these groups and assign groups to projects on which the group members collectively work and share results. This kind of a shared environment not only helps with better but also faster results. This project also has an interface for wEMBOSS administrator, which helps manage data efficiently. Another important point to be discussed about wEMBOSS is the authentication system. wEMBOSS uses the basic HTTP authentication using *.htaccess* and *.htpasswd* files, where in it is mandatory for the users of wEMBOSS to be known by the server where wEMBOSS resides. Any outside user cannot use wEMBOSS. It does not sound sensible and secure to provide access to any random user to the server where wEMBOSS resides. It becomes a hassle especially when users from other organizations would like to use wEMBOSS at a particular organization and contribute to the research. In order to solve this problem, the current project uses a different authentication system. The Shibboleth system is integrated with wEMBOSS which provides access to users from federated organizations formed based upon trust. Users are provided access to wEMBOSS after authenticating and authorizing themselves.

Table of Contents

Table of Figures

Chapter 1. Introduction

The complexity of molecular data and its enormous volume has led to an absolute requirement for computerized databases and sequence analysis tools. Efficient algorithms and approaches are now being derived to deal with the volume and complexity of molecular data, which help researchers advance their understanding of our genetic legacy and its role in health and disease. The latest advances in the field of software have led researchers to better access the analysis and computational tools. Today with the rapid growth in the fields of Bioinformatics and Biocomputing, more good programmers are developing software; many making it freely available. EMBOSS [1], the European Molecular Biology Open Software Suite, is one such software analysis package which is freely available on the Internet and is developed for the needs of the molecular biology (e.g. EMBnet) user community.

Using EMBOSS scientists are able to analyze the DNA sequence data with the help of about 150 sequence analysis programs present in the EMBOSS package. Using EMBOSS one can perform sequence alignment, rapid database searching with sequence patterns, protein motif identification, including domain analysis and many more.

Emboss programs are written in the computer programming language “C” and the software is mainly designed for the Unix operating system, although attempts are being made to run the software on PCs with Windows and Macintoshes. All EMBOSS programs are generally accessible through command line or a console interface. As a result it lacks the ease of use for most of the EMBOSS users, as many of them are biologists and are not familiar with the UNIX command line style of program execution. To address this problem many graphical interfaces to EMBOSS have been developed.

wEMBOSS [2], a joint effort of Argentinean EMBnet node and Belgian EMBnet node, is a web interface for the EMBOSS package where all the EMBOSS programs are accessible through interactive web pages in a user friendly way. wEMBOSS supplies each user with space and tools to organize and review his or her work. Each wEMBOSS user is provided with separate and private workspace where his/her work can be stored and reviewed securely.

wEMBOSS logically organizes work into projects facilitating the review of all project related data for the users. Many functions are provided to manage files in the project. Data can be incorporated into the project by creating a new file, by uploading a file from a local computer or by retrieving data from the databases available in the local EMBOSS installation. Similarly files can also be downloaded to the local computer as shown in Figure 1.1.

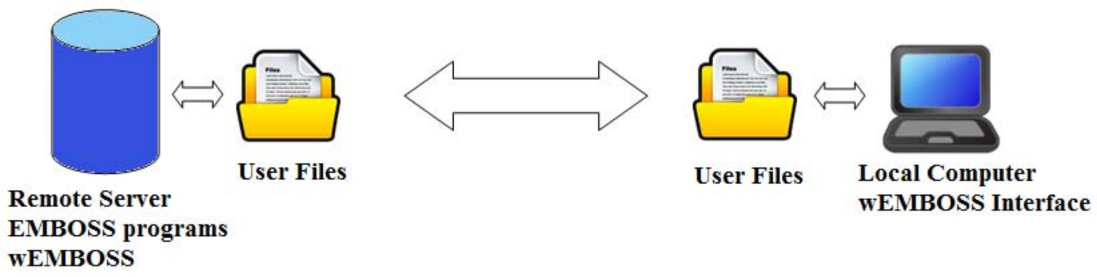


Figure 1.1: wEMBOSS - A Web Interface for EMBOSS

In order for the users to access wEMBOSS, they must be a valid user in the server where wEMBOSS resides. Users are provided with user accounts on the server.

wEMBOSS implements the basic HTTP authentication using *.htaccess*[1] and *.htpasswd*[2] files, where in it is mandatory for the users of wEMBOSS to be known by the server where wEMBOSS resides. It checks whether the user exists on the system by calling standard system functions. What happens when the number of users who would like to use wEMBOSS is very large? Certainly this causes maintenance and performance issues on a heavily used system with a large number of users. What happens when some researcher outside your organization is interested in participating in the research by becoming a member of wEMBOSS at your organization? This kind of participation is not allowed because of two main issues. Firstly, wEMBOSS requires user profiles to be created on the server for its users. But it is not a good idea to create user profiles in your server to users from different organizations as there might exist some security breaches. Secondly, wEMBOSS does not provide the opportunity to work in collaboration with different users of the same organization or different organizations. Allowing users to work in groups by sharing data among themselves would be of great help to users, especially to those users who are in the research field.

To address the questions mentioned above, the current project integrates Shibboleth [5] for authenticating and authorizing users, by which users of different organizations are provided access to wEMBOSS with the Shibboleth authentication scheme and helping them to work in collaboration. This also mitigates the complexity involved in providing and managing individual accounts for each user, with the high level of support involved in the current end user authentication of wEMBOSS. Users are

[1] *This file is the key to providing who has access to the files in a directory and what types of operations they are allowed to perform within this directory*
[2] *The .htpasswd file contains the usernames and encrypted passwords of those individuals who has access to your directory*

neither provided with accounts in the server nor a separate Username-Password database is needed with this type of authentication system. A detailed description of Shibboleth is given in the next chapter. Data corresponding to each user is maintained under the root repository where every wEMBOSS user is provided with a limited amount of space.

As mentioned earlier, this project addresses the problem of sharing work by introducing group management and file sharing in wEMBOSS. This provides the ability to share the research work among different users in wEMBOSS. There are several ways in which a user can share his/her work. They can share it with a single user, or group/groups of users or to all the users in the system. This project advances the usability of wEMBOSS, thereby helping biologists carry on their work in a more adaptable, secure and robust environment. The rest of the report explains in detail how the proposed approach is implemented.

Chapter 2. Background

This chapter gives an overview of the EMBOSS software package and its web interface tool, wEMBOSS. It is important to know about EMBOSS and wEMBOSS before discussing the current project in order to get a clear understanding of the project. This chapter also discusses the technologies used in developing the current project.

2.1 EMBOSS

The "The European Molecular Biology Open Software Suite", popularly known as EMBOSS, is a comprehensive package of sequence analysis and display programs. EMBOSS is a free Open Source software analysis package specially developed for the needs of the molecular biology (e.g. EMBnet) user community**.**

EMBOSS being a free software, is extensively used by many not only for educational purposes but also in production environments. EMBOSS can be easily installed on most UNIX/Linux, MS Windows and Mac OS systems. EMBOSS programs are executed on the command line. EMBOSS is integrated with GUIs and web interfaces like wEMBOSS and JEMBOSS[3] in order to provide users with a better user interface. EMBOSS also provides extensive libraries thereby providing a platform for other scientists to develop new software. Users can easily work with EMBOSS as every program interface is designed the same way. Mastering one is enough to work with all others. Users need not worry about the amount of data that can be processed using EMBOSS, as there are no size limits while processing data.

2.1.1 EMBOSS Programs

EMBOSS consists of hundreds of programs for analyzing Nucleic/Protein sequences that include display of protein statistics, pattern searching, merging sequences to

[3] *JEMBOSS is an open source software that provides graphical user interface for EMBOSS.*

make a consensus, data management, feature predictions, multiple sequence alignment and more. All EMBOSS programs are logically organized into groups depending on their functions.

Sequences can be read and written in many different formats. Almost every sequence analysis package like EMBOSS and every sequence database with a collection of sequences has its sequence format. Sequence formats define the way in which sequence data and its related information can be read/written. Every sequence is recognized by a unique ID number and/or Accession number. Every sequence format has its own way of arranging the sequence identification number, any comments related to the sequence and the sequence itself. From sequence databases an entry can be picked by its ID or accession-number. Sequences can be created in any sequence editor, such as mse[4], or any plain text editor. Examples of some well-known sequence formats are gcg, embl, fasta etc.

For every EMBOSS program, inputs can be given by specifying the database or file name that contains the input sequence(s) to work on and the sequence(s) itself. With a wide variety of sequence formats users might get confused but EMBOSS automatically recognizes the input sequence format, reads sequences from databases or files and executes the program. However, if a sequence is not specified in any recognized format analyzing the sequence data can be difficult.

All EMBOSS programs use a common style of specifying input sequences and outputs generated. The Uniform Sequence Address or USA is the sequence specifying scheme used by all EMBOSS programs. A Sequence can be specified as “file”, “format::file:entry”, “dbname:entry” or “@listfile”, where *file* is the name of the file that

[4] *Mse is a multiple sequence editor that is compatible with EMBOSS. This editor does not come with the EMBOSS package*

contains the sequence, *format* is any recognized sequence format, *entry* is the sequence identification number, *dbname* is the installed database of format dbname and *@listfile* is the name a file that acts like an index of other files. Similar to input sequence format, output formats can also be specified. Output sequences are created in the format specified to the program whereas a default format, *fasta*, is used if no format for an output sequence is specified.

Given below is an example of an EMBOSS program with input and output files mentioned. **Seqret** is an EMBOSS program that reads a sequence file and writes it out in any format we wish to convert to.

```
>seqret file1.seq embl::file2.seq
```

In this example, the program converts sequences in the file ‘file1.seq’ into the format 'embl' and stores them in the file 'file2.seq'.

Apart from the input sequences and output files, there are several other parameters that can be specified to an EMBOSS program for concise and better results. In the absence of any mandatory parameters, EMBOSS programs prompt for any required parameter values.

2.2 WEMBOSS

wEMBOSS is one of the user interfaces for the EMBOSS software. It provides a user friendly web based environment for the users to access EMBOSS. All the Emboss programs can be accessed and users’ work can be organized and reviewed with the help of various functions available in wEMBOSS. Since EMBOSS programs are accessed through a command line it is a lot more convenient for the users to work with such user interface tools and can better contribute to the research.

wEMBOSS provides each user with a private workspace where his/her work is stored in a systematic manner and is separated from the rest of the users' data. All the programs included in the EMBOSS installation can be accessed by using wEMBOSS and can be executed with the help of several tools available in wEMBOSS that are explained in the following sections.

2.2.1 Working with wEMBOSS

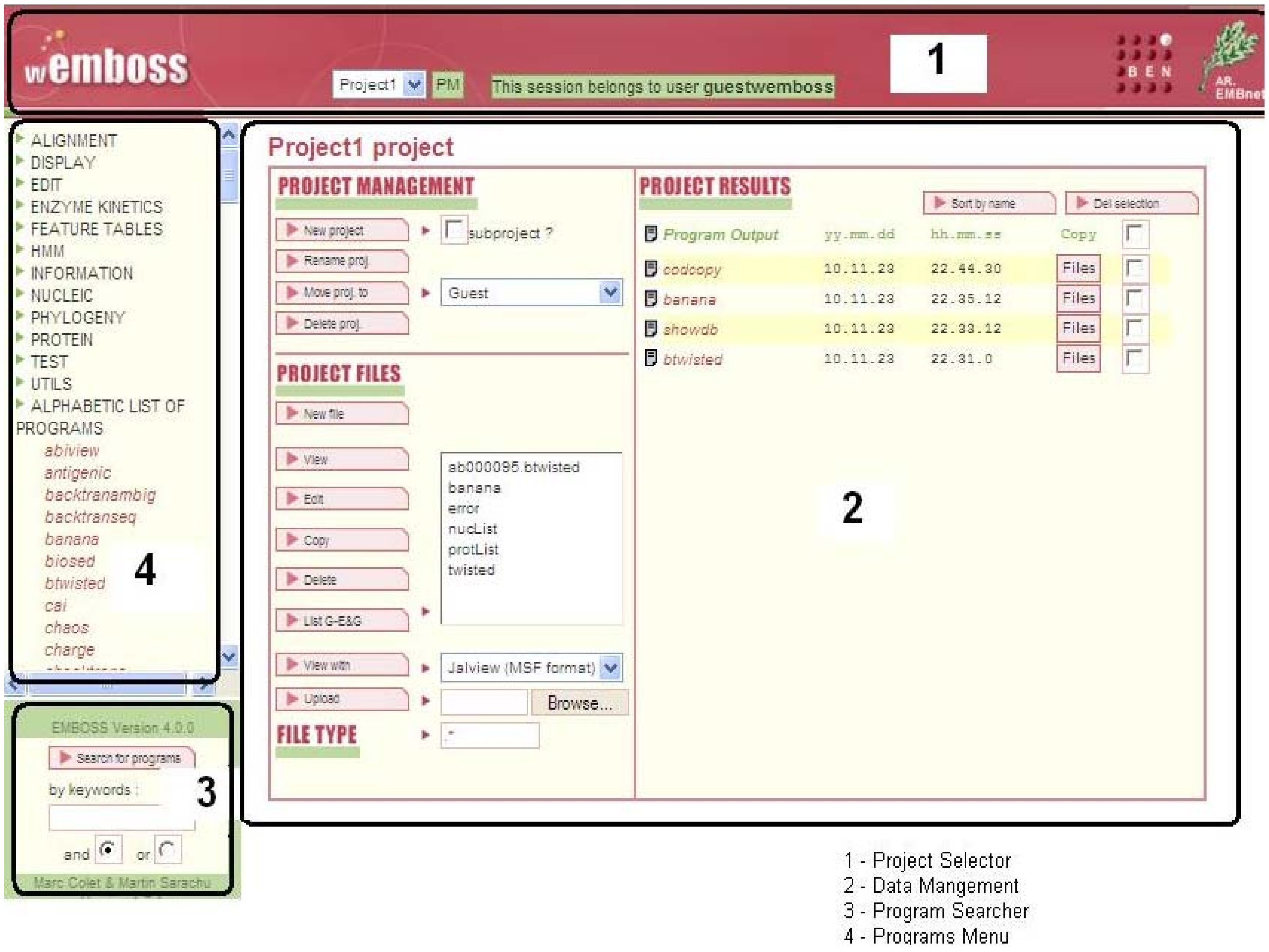
wEMBOSS provides a web-based access to the system. The interface of wEMBOSS is organized into four primary sections. They are project selector, programs menu, program searcher and data management. Each of these sections are described briefly in the following paragraphs.

Project selector is designed to help users select a project and work on the selected project. Project selector comprises of the top most frame of the wEMBOSS interface. Once a project is created it appears in the projects menu. All the files related to the project selected appear in the data management section. Once established, the user can perform several operations like create new files, view/edit/copy/delete existing files and many other operations available on the data management section.

Programs menu, located at the left side of the interface screen, displays all the programs that come with the EMBOSS installation. These programs are presented in a drop down menu arranged in an alphabetical order by groups or by program name. Users can select any program and run the program by providing the required inputs. Inputs to the EMBOSS programs can be provided from available databases, current project directory or uploaded from a user's local computer. Output of the programs can be saved either as a file in the project directory or downloaded to a user's local computer or used as input to other programs. wEMBOSS also allows the administrator to prevent the display of certain programs, like programs that are not suitable for web access, to users.

Program searcher helps users search for programs that are present in the programs menu. All the matched programs are displayed in a new window where the users can either run the program or view the help manual that is provided with all the programs. The program searcher section is located at the bottom of the wEMBOSS interface.

The data management section comprises most of the wEMBOSS interface display area. Data management routines help users organize data into different projects and sub projects. This section is further divided into three sub sections; project management, project files and project results. Project management, as the name suggests, has several tools that manage projects. Operations like create, rename and delete a project are performed with the help of project management routines. All tools related to file management can be seen in the project files section. The last sub section, project results, displays all the results of any program executed. Figure 2.1 shows the alignment of all the sections of the wEMBOSS web interface display.



Program Output	yy.mm.dd	hh.mm.ss	Copy	
codcopy	10.11.23	22.44.30	Files	
banana	10.11.23	22.35.12	Files	
showdb	10.11.23	22.33.12	Files	
btwisted	10.11.23	22.31.0	Files	

Figure 2.1: Sections of wEMBOSS Web Interface

The data in wEMBOSS revolve around the concept of a project and its management. wEMBOSS organizes work into projects. As mentioned earlier, every user is provided with a private workspace under the user's home directory, where all work is organized into projects. A project is simply a directory in the user's workspace where the user can create files and sub projects within a project. A user can give any name to the project at the time of its creation. A directory with the same name is created in the user's home directory. A user can create as many projects, sub projects and files as permitted by the system.

By organizing work into projects, users can easily place files related to a particular research activity together. This project data can be accessed during the current session and saved for subsequent use at a later time. Users can also rename projects, delete projects and move projects to a different location. Under the user's home directory is the directory named *wProjects,* which is the root directory for wEMBOSS projects. All the projects created by the user are placed under this directory. All files and sub projects related to a project are placed under the project's directory. The file structure used by wEMBOSS is shown in Figure 2.2 below.

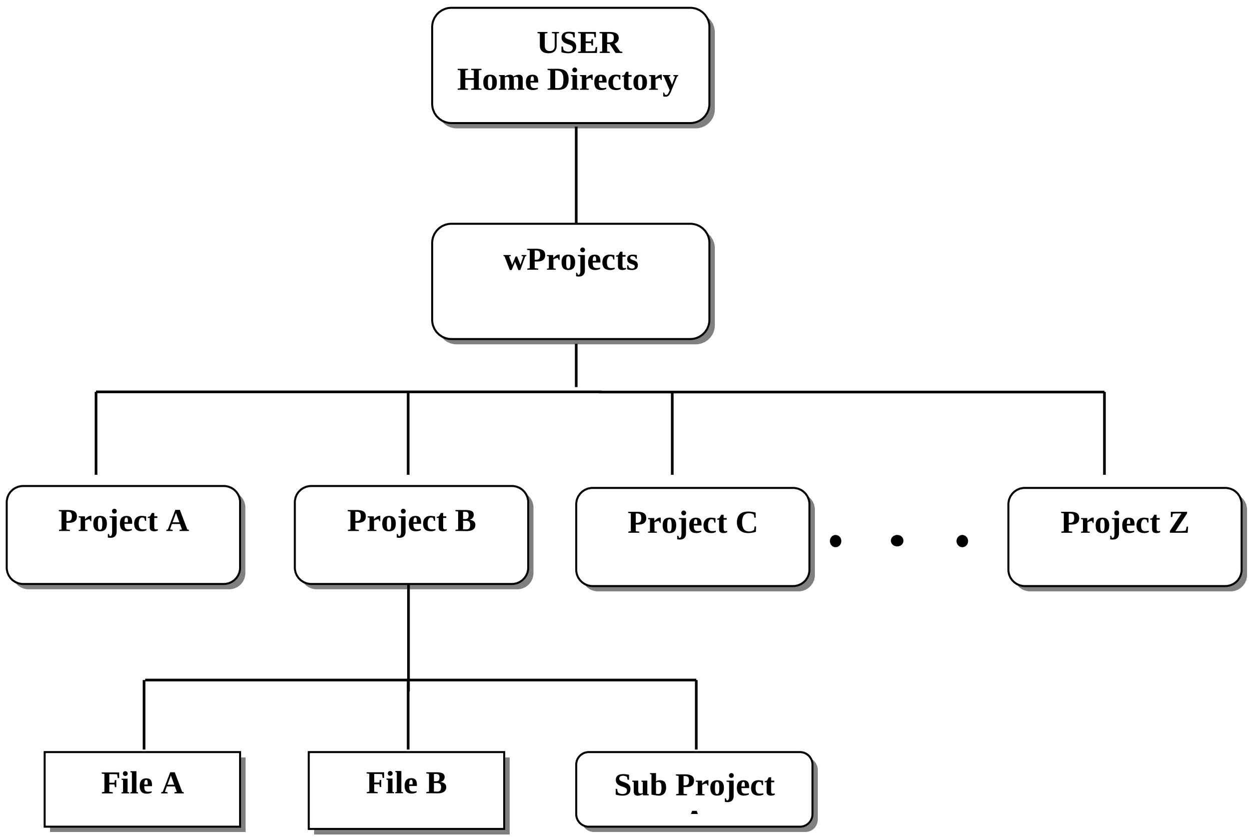


Figure 2.2. File Structure of wEMBOSS

Several services are provided in wEMBOSS to effectively manage files in a project. Once a project is created, users have the option to rename the project to another name or move the project and its contents into some other existing project. The project moved becomes a sub project of the project it is moved into. Users can delete an existing project, which also deletes all the project's contents. Project data can be viewed using the view function, edited to modify their contents, deleted or copied to another project. Two special sequence lists, *protList* and *nucList*, are created in each project to include user or database sequence names to be used as an input allowing quick access to these sequences from any program. Names of files containing sequences, names of sequences in the USA (as defined in Section 2.1.2) format are accepted as *nucList* and *protList* file entries. These files are only shown for programs that require protein or nucleotide sequences as input, respectively. Instead of entering the database or file names repeatedly, including them in the *nucList* and *protList* files is easier to reference while executing a program. Users can also filter the data by some matching criteria after selecting an entry from one of the *nucList* and *protList* files, while executing a program. The file *NucList* only has entries corresponding to nucleotide sequences and *protList* contains entries corresponding to only protein sequences.

While running a program, results are saved under the current project with the indication of the name of the executed program and the exact date and time it was run. The interface also allows users to copy a given file from the results to the current or another project capable of using these data as input. An output file appears in the project results section and is not part of project files. An output file therefore cannot be directly used as an input to any EMBOSS program. It has to be first copied as a file to the project, either with the same name or a different name and it can then be used as an input to a program.

2.2.2 Internal Working of wEMBOSS

wEMBOSS is composed of four main components, authentication, client side validation, parsing and presentation.

In order to use wEMBOSS, users have to be valid members of the server in which wEMBOSS is present. Since wEMBOSS uses *basic http authentication*[5], whenever a user attempts to access wEMBOSS, a prompt is displayed asking for the user's authentication details. The authentication program checks if the user is a valid user in the system. Validation is performed every time the user interacts with the server side of wEMBOSS, although the user password is required only once by the web browser. After the validation is performed, the user can work with wEMBOSS. The first time a user logs in, he is prompted to create a new project. This is required since every program requires at least one project initially to store data.

Validations are of two types; client side validation and server side validation. Server side validation involves performing checks on the server. Client side validation involves validating user input before a query is posted to the server. Whenever a user wishes to execute a program, basic validations such as check for the input entries, and creation of a project before creating files are checked using the client side validation module. The system does not have to contact the server for such small validations.

Every EMBOSS program is associated with an ACD (Ajax[6] Command Definitions) file. This file has information about the input files, output files generated, and describes the necessary parameters required by the respective programs. It also validates and writes error

[5] *Basic http authentication is the simplest form of authentication where a web browser or client program is required to provide username and password to access a resource.*
[6] *Asynchronous JavaScript and XML, also known as Ajax is a technique by which web applications exchange data with a server and update parts of a web page without reloading the whole page.*

messages if any necessary parameters are missing; any parameters are outside a specified range or any dependencies among the parameters of the program exist. The ACD file contains a special purpose language called Ajax Command Definitions or ACD, specially designed for EMBOSS. These ACD files are translated into Perl language files called SACD files that contain a nested hash structure holding information of the associated programs. These SACD files are in turn evaluated during the execution of the CGI (Common Gateway Interface) script that generates an HTML page that is displayed to the users. The wEMBOSS CGI script along with a few Perl modules are responsible for the presentation of the wEMBOSS interface and working of various functions that are discussed in further chapters. Every section of wEMBOSS is associated with a Perl module and this module is responsible for the presentation of its section. The *input.pm*[7] module parses the SACD structure for the requested EMBOSS program and prepares dynamically a form to gather input choices of the user. The HTML page of the program is then generated and displayed in the user's browser window.

2.3 Technologies Used

The current project uses Perl and C programming languages. Java Script is also used for client side validation. Majority of the project is developed in Perl. CGI technology, SACD files and HTML along with Perl modules are responsible for the display and analysis of user data, as well as execution of EMBOSS programs. Users' sequences are easily analyzed by using the regular expressions of Perl. Also Perl has dynamic loaders that help extend Perl with programs written in C as well as create compiled libraries that can be interpreted by the Perl interpreter. Current project has programs written in C, developed for interaction with the

[7] *One of the perl modules responsible for the interface of wEMBOSS. Input.pm displays the HTML page of all EMBOSS programs.*

GDBM database and manipulation of information stored in the database. Data stored in the database corresponds to the groups and sharing information that are discussed in the later chapters

wEMBOSS also uses JavaScript. JavaScript does minor user validations and adds some functionality to wEMBOSS. JavaScript makes the application run faster since a request does not have to be sent to the server to make small validations that can be taken care of at the client's side. Validations such as, check for proper project or group names, checks for the minimum number of users in a group and so on are validated with the help of a JavaScript file in wEMBOSS. A CSS[8] (cascading style sheet) file is used for enhancing the look of wEMBOSS application that makes it appealing to the user.

2.4 Dynamic Web Content Technology

As already mentioned wEMBOSS is a web based application that uses CGI **[4]** technology for dynamic interaction with the users. The Common Gateway Interface, most commonly known as CGI, is a standard way of transferring data between a World Wide Web server and an application program. A CGI program is a filter for requests sent to a server to transform requests to dynamically produced output based on that request. When a client sends a request to the web server, the web server locates the CGI program on the server. It is the CGI program that processes the request and sends the appropriate output back to the web server and from there to the user. The process is clearly shown in Figure 2.3 below.

[8] *Cascading Style Sheets are used to control the style and display of pages written in a markup language. They are mostly used for designing web pages.*

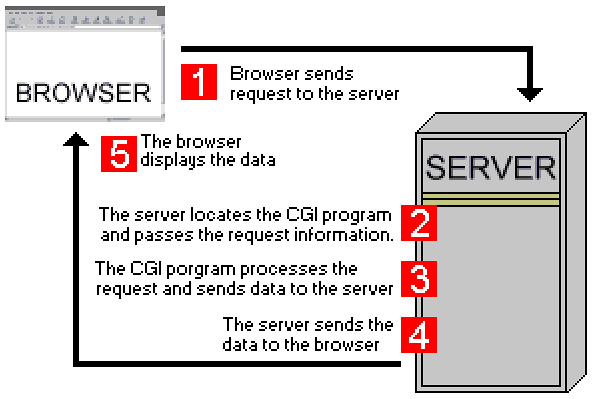

BROWSER
1 Browser sends request to the server
SERVER
5 The browser displays the data
The server locates the CGI program and passes the request information. 2
The CGI pogram processes the request and sends data to the server 3
The server sends the data to the browser 4

Figure 2.3: Client Server Interaction in CGI Applications **[4]**

CGI programs are one of the common ways for Web servers to serve users dynamically. CGI standards are platform independent and can run on most operating systems, although they are popularly used on UNIX systems. As wEMBOSS is web based and requires execution of Perl scripts dynamically, CGI technology plays an important role in the working of wEMBOSS.

Chapter 3. System Design

This chapter focuses on the design of the system that includes the application architecture, directory structure of the system and different types of projects involved. The database design and authentication mechanisms are also discussed in this chapter. The word *wEMBOSS* in this chapter refers to the new enhanced version of wEMBOSS developed by this project.

3.1 Application Architecture

One of the key elements of any system design is the application architecture. The application architecture defines what functions each component of the application does and how components interact with one another. To reduce the complexity, applications are broken into several layers or tiers where each layer is dedicated to perform a specific type of service. Any web based application like wEMBOSS incorporates the Client/Server Architecture that involves at least two layers. The Client/Server architecture involves service providers called servers and service requesters called clients, who communicate over a network. Every user that uses a web browser is a client and data accessed by these clients through web pages is stored on one or more servers. Separation of components improves the ability of the application to change easily according to different user and different system requirements. wEMBOSS uses a Client/Server architecture involving three layers. The three layers are client, application and data layers.

The Client layer or the user interface layer provides an interface between the user and the system and gives the user access to the application. This layer, usually a web browser, processes and displays HTML data and formatting instructions, issues HTML requests and processes the responses generated by the application layer. In wEMBOSS, the client layer is responsible for the display of the user's project contents, program results and allows for data manipulations.

The application layer contains a centralized processing logic that facilitates management and administration. This layer receives requests from clients and generates HTML responses after applying application logic and manipulating data present in the data layer. This layer mediates between the client and data layers. wEMBOSS has eight Perl modules, each one corresponding to one of the eight possible areas of action. The areas are project management, program execution, program display, start of application, project display, program search, view files and parsing of SACD files. All these modules revolve around one CGI script that is called every time the user sends a request. The CGI script in turn calls any of the above mentioned Perl modules depending on the request sent by the user. For instance, when a user selects a project from the projects list, the *titleAction*[9] module identifies the project selected and sends a query to the CGI script to call the *PMAction*[10] module to generate the selected project's contents. The workflow process from data acquisition to data processing and delivery of data is managed by the application layer.

The third layer is called the data layer that manages the data stored and is responsible for handling queries submitted by the application layer. It consists of the data management system and the data itself. In wEMBOSS, this layer consists of the GDBM database and data repository. Figure 3.1 shows the three layers of wEMBOSS.

[9] *This is one of the eight Perl modules that is responsible for displaying the projects list and lets users select any project from the list.*
[10] *This Perl module helps the user to manage data corresponding to each project.*

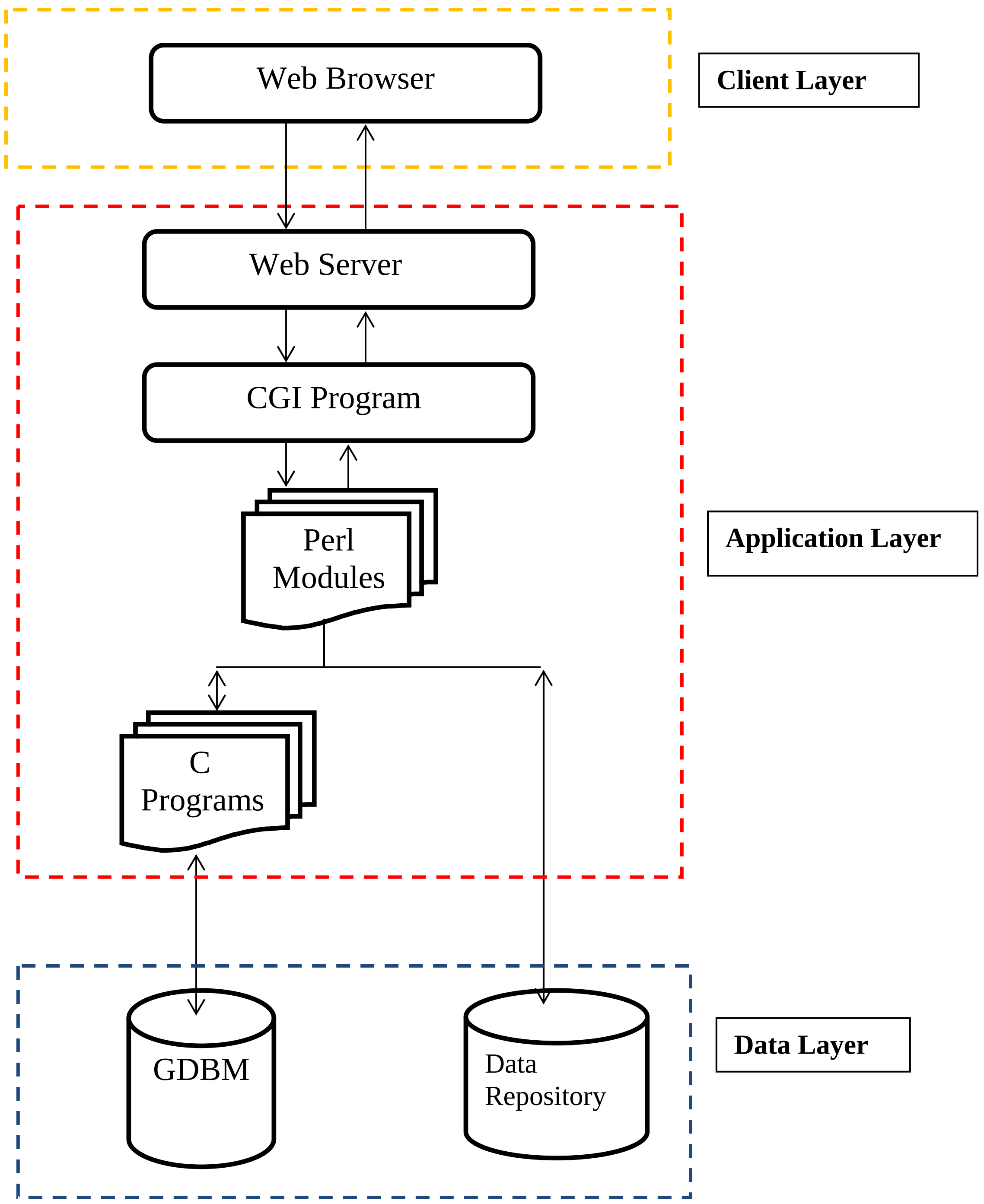


Figure 3.1. Three Layered Architecture of wEMBOSS

3.2 System Overview

This section describes different types of projects in wEMBOSS and how the facility of sharing data introduces a collaborative environment to wEMBOSS. Also, the file structure of the enhanced wEMBOSS is described in detail.

3.2.1 Project Types

In wEMBOSS, projects[11] are broadly classified into four types depending on how data is shared by the project owners with other users in the wEMBOSS system. The sharing can be private, group, shared and public projects.

Private projects are the basic type of projects that are created by simply providing a name to the project. As the name suggests, the contents of these projects are only accessible to the creator or owner of the project. Only the owner of the project has complete authority to perform any operations on the project such as file creation, renaming the project, move the project into another project, or delete the project.

In wEMBOSS a project that involves one or more groups is called a group project. A group consists of two or more users, formed for the purpose of working collectively on projects and sharing data among the members of the group. Groups can be created by any user who is authorized to use wEMBOSS. Projects that are associated with at least one group are called group projects. The concept of groups is introduced to promote the ability of users to work on research projects in a collaborative way. Any user willing to allow other users to work on his projects can do so by simply creating a group of users he wishes to collaborate with and associating this group with his projects. A group can also be associated with one or more projects. Similarly a project can be associated with one or more groups. However there

[11] *A project in wEMBOSS is a directory that contains files and sub directories (sub projects) that belong to the project.*

is a limit to the number of groups a project can be assigned to and the number of projects a group can be associated with. The creation of groups, sharing data and user limitations are dealt in detail in the next chapter. In the case of a group project, the project contents are accessible to the project owner and the members of the group to which the project is assigned. Members of a group can view, edit project contents, create new files and access project results. However they are not eligible to delete files of the project created by other members for obvious reasons. Groups are assigned a group leader. The group creator by default becomes the group leader. The group leader has all access rights on the projects assigned to the group. Shared projects are similar to the group projects except that in shared projects there are no groups involved.

In some cases users might just be interested in sharing certain data with individual users only and not a group as a whole. In such situations wEMBOSS users can work with shared projects. Project contents are shared with individual users with choice of permissions by the owner of the project. Read and write permissions can be granted on files to users. Permissions can also be changed or revoked from a user at any point of time by the owner of the project. Even after granting permissions on a file the shared user can never be able to create new files in the shared project or perform actions like delete, copy, rename on the shared file. This limitation is to protect the owner's project contents against any inappropriate use by the shared users. Read permission is the default permission. Users can view a file with a read permission on the file. Write/edit permission gives the user authority to make changes to the file. However there are also limitations on the number of files that can be shared with a user and the number of users a file can be shared with. Setting up this limitation saves storage space in the database.

Public projects, as the name suggests, are projects whose contents are shared with all the users in the wEMBOSS system. Owner of the project can select the option of sharing files of a project to the entire set of users of wEMBOSS. Similar to shared projects, owners are given the option of choosing permissions on files to be shared with other users. Also, permissions can be revoked at any point of time by the owner. A detailed description of how groups are created, files are shared, and permissions are defined is given in the Chapters 4 and 5.

3.2.2 Directory Structure

The directory structure is the way in which directories and files are organized in a system and displayed to the users. Files are grouped according to their purpose and are arranged into a hierarchy of folders for quick retrieval. Similar to operating systems, wEMBOSS has its own directory structure to group data of different users separately. Every wEMBOSS user is provided with a personal workspace where he can create project folders to manage his data and results. All projects, sub projects, files and project results of the user are stored in his personal workspace. The introduction of different types of projects into wEMBOSS and the concept of sharing has changed its file system structure significantly. The way projects, files of different users are arranged is described in subsequent paragraphs. The file system structure of wEMBOSS is shown in Figure 3.2.

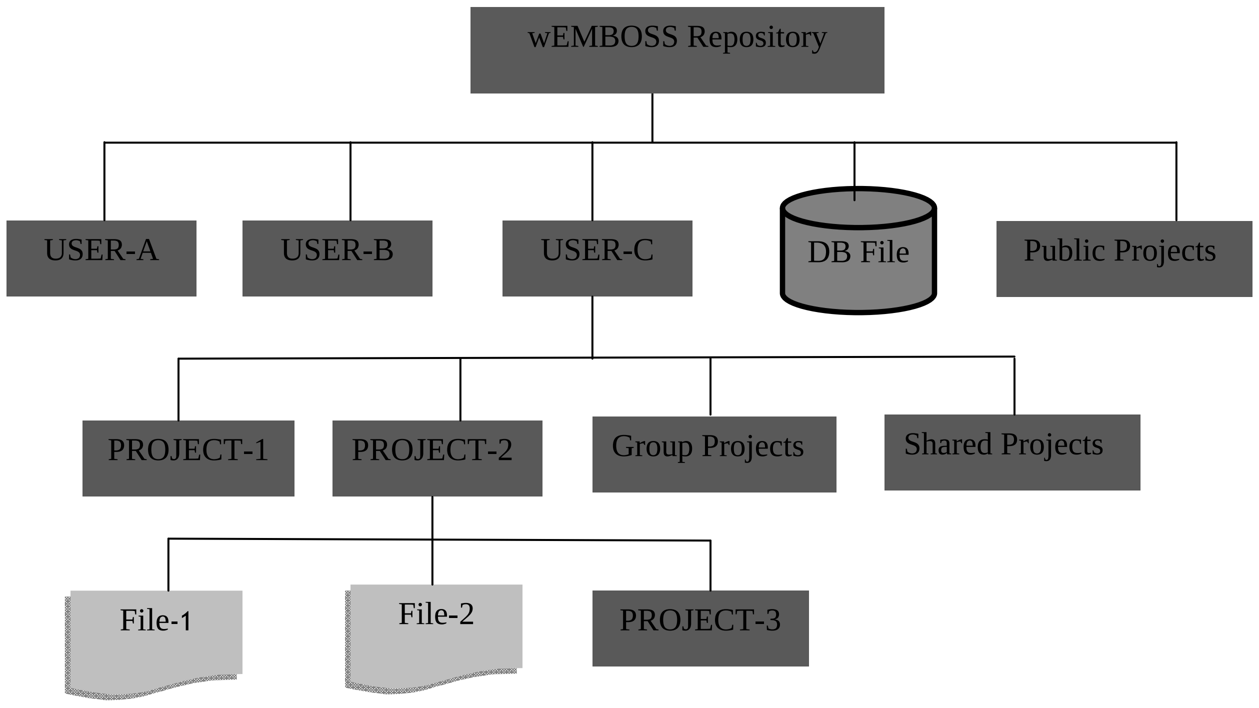


Figure 3.2. Directory Structure of Enhanced wEMBOSS

From Figure 3.2 it can be clearly seen that every user has a separate working directory under the wEMBOSS root directory. USER-A, USER-B AND USER-C are example users shown in the figure that depicts three different users' working directories. These working directories are created when a user logs into wEMBOSS system for the very first time. A check is made if the user is already allotted a working space. As already discussed, a project is also a directory and all projects belonging to each user are located under the user's directory that in turn is located under the wEMBOSS root directory. PROJECT-1 and PROJECT-2 are examples of projects that belong to USER-C shown in the figure. Every time USER-C logs into wEMBOSS these directories are listed as projects on the wEMBOSS homepage. As can be seen from the figure, any files or sub projects created in a project are placed under the project's directory.

As mentioned previously, there also exist group, shared and public projects other than private projects. When a new group is created, it is placed under the *Group Projects* directory. This new group is a directory by itself and is given the name of the group at the time of its creation. If at any time, a project is assigned to a group by its owner, a symbolic link is created in the group directory which points to the project assigned to the group. By this arrangement, all the projects assigned to a particular group can be easily known just by tracking the links in the group directory. Such projects are called group projects. Now, how do group members get access to these group projects? As you can see from Figure 3.3, a symbolic link is created in the *Group Projects* directory of each of the group members. The figure shows User A is a member of Group 1. The symbolic link created in User A's *Group Projects* directory points to Group 1 created under the *Group Projects* directory of User B. This acts like an indirect link from User A's *Group Projects* directory to projects assigned to Group 1, as Group1 directory itself has symbolic links pointing to its assigned projects.

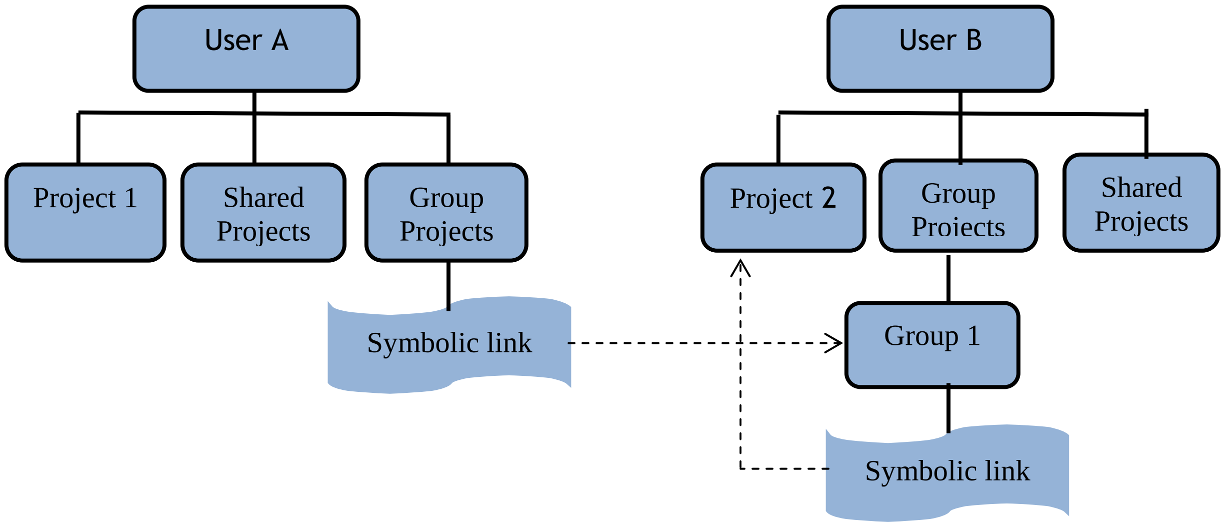


Figure 3.3 Sharing of Group Projects

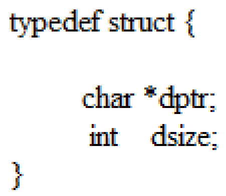
As mentioned previously, *Group Projects* directory has links to group projects. Similarly, *Shared Projects* directory has links to shared projects and *Public Projects* directory has links to public projects. Every time a user shares his project's contents with other users, a symbolic link is created in the *Shared Projects* directory of the shared user. This link points to the project shared. Shared projects appear in the projects list of the sharer user. As a result the shared user has access to the shared projects. When access to a shared project is revoked by its owner, the symbolic link present in the shared user's Shared Projects directory is unlinked and as a result the shared user will no longer be able to view the contents of the shared project.

Public projects are also shared in a similar way. The only difference is that the directory that contains all symbolic links to projects is placed under the wEMBOSS root directory and not under each user's working directory as in the case of group and shared projects. In other words the *Public Projects* directory is accessible to all users of wEMBOSS and is not specific to a single user. Each user has a *Group Projects* and *Shared Projects* directory that are created along with the working directory of the user. When a user makes his project public, a symbolic link is created in the *Public Projects* directory and all other users in the system have access to the public projects.

The database file, *DB_file*, stores information corresponding to groups, projects and user limitations. Information like members of a group; groups associated with a project; files of a project and permissions with which they are shared are stored in this file. Other information like: users with whom projects are shared; groups owned by a user; and projects assigned to a group are fetched from the file system of wEMBOSS. The process of sharing project contents is discussed in detail in Chapter 4.

3.3 Database Design

The GNU database manager [9], a well-known hash based database, is used for the storage of groups, sharing and administrative information in wEMBOSS. This database manager, popularly known as GDBM is a set of database routines that uses extensible hashing and is commonly used on the UNIX platforms. wEMBOSS does not need a relational database that uses an extensible set of SQL statements, as no complex queries are involved and data can be accessed by simply providing a key. A simple hash based key-data pair database is perfect for this kind of application. GDBM keys as well as data are of type *datum* that is a structure as defined in the Figure 3.4.



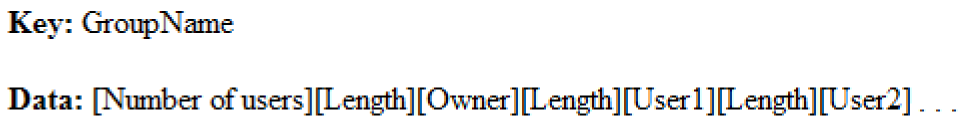
```
typedef struct {

        char *dptr;
         int    dsize;
}
```

Figure 3.4 Structure that Defines the Basic Unit of Data Storage in GDBM

The pointer *dptr* points to the key or data value and *dsize* corresponds to the length of key or data value. The above structure allows for arbitrary sized key and data values. Keeping this design issue in mind GDBM is chosen for data storage. GDBM stores data in a data file on disk, Data corresponding to groups, shared projects and administrative information are retrieved by simply providing the key value.

3.3.1 Data Storage

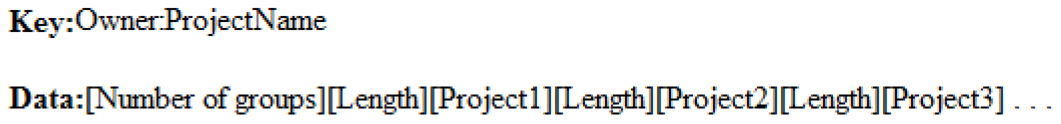
Data corresponding to groups, projects and administrative information is stored in GDBM. When a group is created, the name of the group, the owner of the group, members of the group and the total number of members in the group are stored in the database. The key value for such records is the group name and the rest of the group information is stored as the data value. Data value is stored in the format [Length][username]. Key-data pair of a record that stores group information is shown in the Figure 3.5 below.


Key: GroupName

Data: [Number of users][Length][Owner][Length][User1][Length][User2] . . .

Figure 3.5 Key-Data Pair of a Record that Holds Group Information

In the format given in Figure 3.5, *Number of users* is the total number of members in the group. Here *length* field corresponds to the length of respective usernames that appear immediately after the *length* field. For example, let the username be 'Martin', so the length of the name 'Martin' is 6. This value is stored as [6][Martin] in the database. Search for a username becomes a lot easier with this format. From the above example we know that the username 'Martin' is of length six. So, while searching for a username, the value to be compared is compared with the first six characters of the data. The pointer keeps moving ahead until it finds a match. Also, integer values like *Number of users, Length* are stored as binary. Groups associated with a project are stored in a similar fashion. In this case key and data pair is shown in the Figure 3.6.

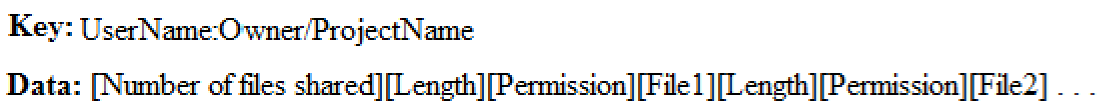

Key: Owner:ProjectName

Data:[Number of groups][Length][Project1][Length][Project2][Length][Project3] . . .

Figure 3.6 Key-Data Pair of a Record that Holds Groups of a Project

Since project names can be duplicated in wEMBOSS, specifying the owner of the project in the key value avoids the duplication of keys in the database.

Information about files in a project, along with granted permissions, shared with users by the owner of the project is also stored in the database. Key-data pair for such records is as shown in the Figure 3.7.



Key: UserName:Owner/ProjectName

Data: [Number of files shared][Length][Permission][File1][Length][Permission][File2] . . .

Figure 3.7 Key-Data Pair of a Record that Holds Shared Filenames of a User

In the key value *UserName* is the user with whom files are shared by *Owner* of the project with name *ProjectName.*

As mentioned previously, wEMBOSS has several user limitations that are set by the wEMBOSS administrator. Each of these limitations are discussed in detail in Chapter 5.

3.4 Integration with Shibboleth

The wEMBOSS system uses an authentication scheme that is based, in part, on Shibboleth [5]. Shibboleth is standards based, open source software package for web single sign-on within an organization or among organizations that have an established trust relationship in order to form a Virtual Organization. Most applications follow the traditional method of allocating usernames and passwords to users for authentication. Shibboleth avoids administrative as well as storage overhead involved in maintaining usernames and passwords. Shibboleth's single sign on methodology gives access to multiple applications by authenticating just once. Shibboleth's main objective is to facilitate authentication followed by authorization of a user requesting access to a restricted resource.

The Shibboleth system has two main components, identity provider (IDP) and Service Provider (SP). The identity provider is the software run by the organization whose users request access to a secured resource with in the same organization or at a different organization. The service provider software is run by the provider of a restricted service or resource. To gain access to a restricted resource a user first authenticates with his home institutions credentials. On successful authentication, the user's necessary identity information is passed by the identity provider to the service provider upon request. Depending on the attributes of the user provided by the identity provider, the Service Provider makes authorization decisions based on the requirements of the requested service. Virtual organizations can maintain an entitlement server that defines and manages virtual organization entitlements that are independent of any organization. Authorization decisions are also made by the entitlements of a user pertaining to the virtual organization, if required by the requested service.

3.4.1 Shibboleth and Entitlement Server Protocol

A brief Step by step procedure of how Shibboleth integrated with an Entitlement Server works is explained in this section.

1. A user sends a request to a Service Provider to access a restricted resource, which is wEMBOSS in our case, through the web.
2. The browser is redirected to WAYF (Where Are You From) service by the Shibboleth Indexical Reference Establisher [SHIRE] component of the Service Provider.
3. A user identifies his home institution from a list of institutions in the network, displayed by the WAYF service.
4. The request is then passed to the appropriate identity provider's handle service by the WAYF service.
5. The Identity Provider contacts the user for credentials for authentication purpose
6. The user provides credentials to the home institution.
7. Upon successful authentication, a unique handle is generated by the Identity Provider's handle service and is sent to Service Provider's SHIRE indicating the user is who he claims to be.
8. The handle is now passed to the Shibboleth Attribute Requester[SHAR] by SHIRE

9. Service Provider now needs to make authorization decisions. It needs to know if the user has the necessary attributes to access the resource. SHAR requests the Identity Provider's attribute authority for attribute assertions. Requests can also be made to the Entitlement Server for any attributes pertaining to the Virtual Organization, as discussed from Step 11 through Step 14.
10. The attribute authority now looks for the user's attributes according to the user's release policy and sends the requested attributes to the SHAR.
11. The attribute authority can also contact the Entitlement Server for user entitlements.
12. Entitlement Server's client application, which resides at the Service Provider, makes requests to Entitlement Server.
13. Entitlement Server looks up in the database for user's entitlements and replies with assertions.
14. This information is sent back to the SHAR and access is either granted or denied to the user.

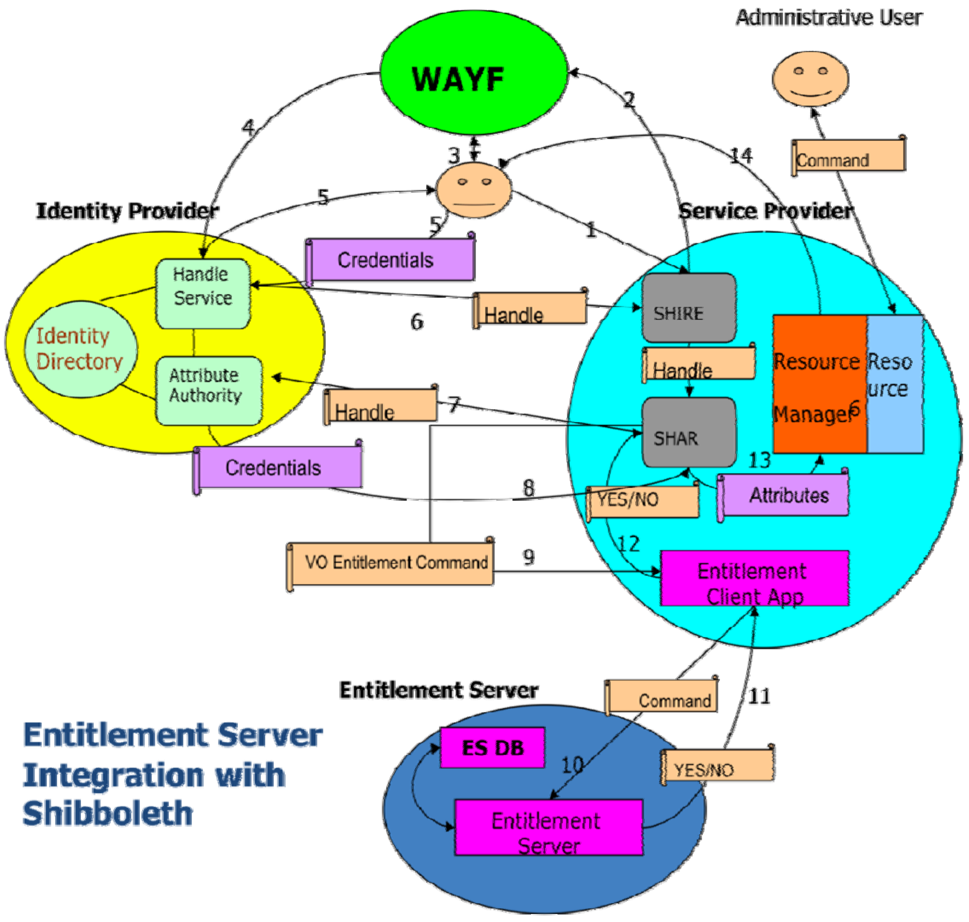


Figure 3.8 Shibboleth Authentication Protocol [6]

Any user wanting to work with wEMBOSS can do so by directing the browser to the wEMBOSS web address, which automatically redirects the browser to the (where are you from)WAYF service. A user can gain access to wEMBOSS only if his home institution is present in the list of organizations provided by the WAYF service. After the user selects his home institution and authenticates himself successfully, authorization decisions are made by requesting user attributes. Entitlement checks are made by contacting the Entitlement Server. Any user with the wEMBOSS entitlement is authorized to work with wEMBOSS.

Chapter 4. System Implementation

Collaborative work refers to processes, methodologies and environments in which professionals engage in a common task and individuals depend on and are accountable to each other. Groups with a common mission or goal are normally involved in such collaborations. Similarly, enhanced wEMBOSS also provides a collaborative environment for users with a common mission. A clear understanding of different types of projects; sharing of data; group description and the role played by a group in wEMBOSS are given in Chapter 3. The current chapter provides an insight on various functions in wEMBOSS which facilitate collaborative work such as creating groups, assigning projects to groups, sharing data with other users, and using other users' project results. Each of the above mentioned functions are explained in detail in this chapter.

All group management and file sharing functions are handled by the *Project Management* module. Any function initiated by the user requires wEMBOSS CGI script to process the input and pass the query to the corresponding Perl script. The concerned Perl script then handles the query and performs the necessary action.

4.1 Group Management

The *Group Management* section in wEMBOSS provides users with a set of functions that help them create a group, share data with the group members and various other activities involving a group. The following paragraphs explain each of the functions that are present in the *Group Management* section.

The *New Group* command, as the name implies, allows users to create a new group. Clicking on the *New Group* button displays a pop up window, where the name for new group is entered. The Project Management module handles all the functions related to groups and sharing besides managing projects and files. This module accepts the user’s command and displays a pop-up window with the help of a JavaScript file. After the user enters the name of the group to be created, a list of users in the wEMBOSS system is displayed allowing him to add users to the newly created group. The group creator/owner can either select users from the list displayed or can search for a user by entering the user’s name in the search box. The list dynamically populates with users whose names match with the value entered in the search box. Dynamic population of users is achieved with the help of Java Script regular expressions, which perform pattern-matching functions every time a character is entered in the search box. A group is comprised of at least two members, as it does not make any sense for a group to have just one member. Therefore, a minimum of two users should be chosen from the users list, which otherwise displays an error message asking the group creator to select more users. Also, there is a limitation on the maximum number of users in a group. Addition of a large number of users to a group not only requires enormous storage space for storing group information in the database, but also requires numerous links to be created in each of the group member’s working directories. The worst scenario occurs when a user adds all other users in the system to his group. The wEMBOSS administrator sets the ‘number of group users’ limitation. However, default values are set initially to all the limitation values, before the administrator changes them. Error messages are also displayed in scenarios where a group with the same name already exists in the system or when a user tries to exceed the maximum number of groups that can be created by him. Once a group is successfully created, an entry is made in the database with the group name, number of users in the group and all the members of the group including the user who created it. In addition, a directory with the group name is created under the *Group Projects* directory of the user who created the group. Links to this group directory are created in the *Group Projects* directory of those users who are members of the newly created group. As a result, the group members will now have access to the contents of the group. Every user can create groups although there is a limit to the number of groups each user can create.

Creating a group is not enough to provide the group members access to group projects. The next step is to associate the group with a project to be worked on. *Assign Group* command helps users to accomplish this task. A list of groups owned/created by a user is displayed in the *Group Management* section. In order to associate a group with the current project, user needs to select a group first and then click on *Assign Group* button. If, either a project or group is not selected then a pop-up window appears asking the user to select the required entity. Similar to user selection, group name, if known, can also be typed in the search box. If the operation is successful an entry is made in the database where the numbers of groups, names of all groups associated with a project are stored. A link to the associated project is created in the group directory present in the *Group Projects* directory of the creator/owner. The group directory now contains links to the projects, also known as group projects, assigned to the group. As already discussed, group members have access to the group directory contents. As a result, indirect links are formed from the group members working directories to the group projects, allowing them to access group projects and work collectively. Formation of links to the group directory and to the group projects is shown in the Figure 4.1.

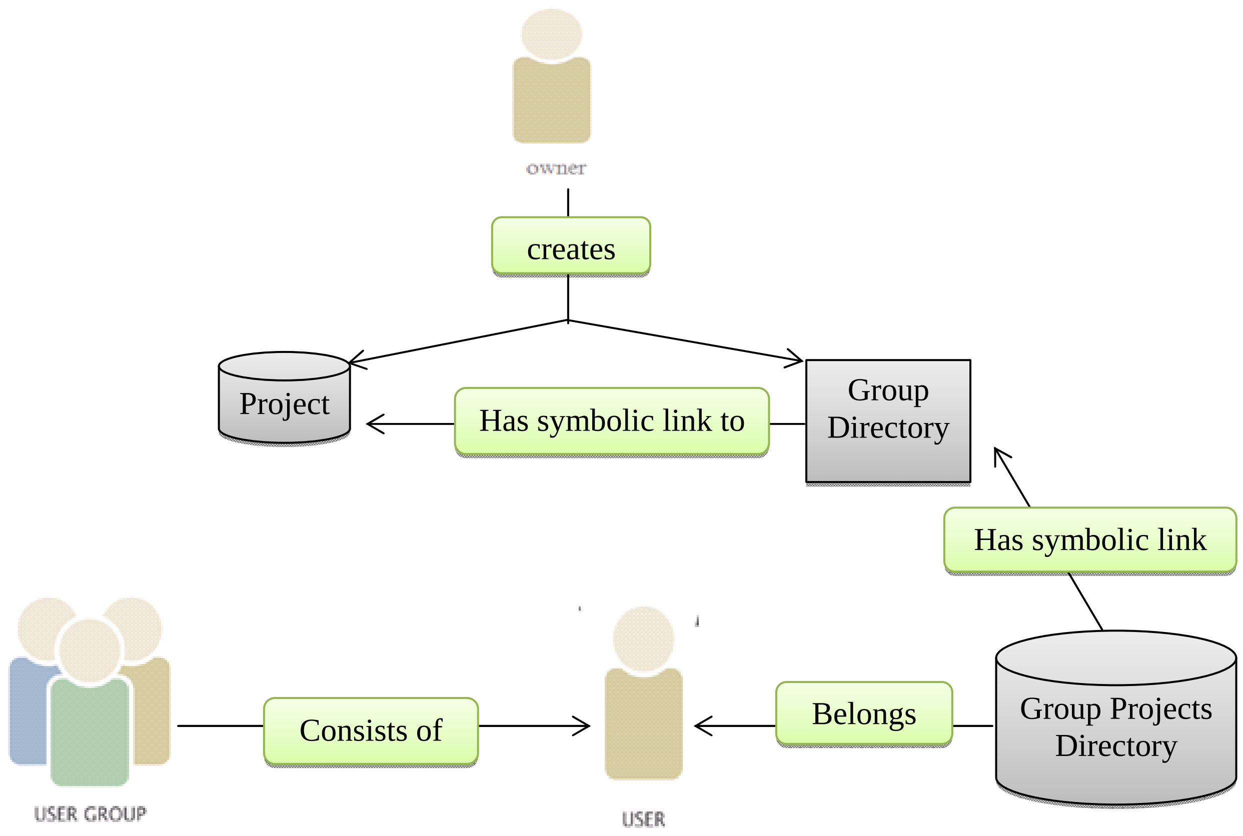


Figure 4.1: Pictorial Representation of Sharing Group Projects.

Default user limitation value of ten is set on the number of groups that can be associated with a project or number of projects that can be assigned to a group, for the same reasons mentioned previously. These limitations can be changed by the administrator. A user is issued an error message about the limitation, if he tries to exceed the count. Trying to associate a group that is already associated with a particular project displays an error message to the user. A user can filter the groups list by making a choice of either viewing all groups owned by him or groups associated with the current project. When a user selects to view all groups owned by him, data is acquired from his *Group Projects* directory, that contains all the groups created by him. When the choice to view all groups associated with the current project is selected, the database is searched using current project name as the key and the data retrieved is displayed in a list.

Group members can not only be added at the time of group creation, but can be either added or removed at any point of time. The number of group members is always within the limitation value set by the administrator. *Add Members* function lets users add new members to the group. However, only owner of the group is allowed to perform this operation. By selecting a group from the groups list and clicking on *Add Members* button displays a list of users not already present in the group. Users belonging to the group selected, are retrieved from the database and are filtered from all the users in the system. This function avoids duplicating members in the list. Selected users become the new group members and are able to access the group projects. The *Remove Members* button displays all the members of the selected group. It works in a similar fashion to *Add Members*. Once users are removed from a group, they no longer have access to the group projects.

A user can unshare his project from a group at any time. Group members no longer are able to access the project contents. This act deletes the symbolic link to the owner's project, previously created in the *Group Directory*. This group is also removed from the list of groups assigned to the project, from the database. A project can be unshared from a group with the help of the *Unassign Group* button.

Unassigning a group from a project and deleting a group are two different actions. Unassigning precedes the deleting actions. When a group is deleted all the data in the database pertaining to the group is deleted from the database. This group is also removed from all the projects previously associated with the group. The group directory is also deleted. To delete a group, user selects a group from the list and clicks on the *Delete Group* button.

Group members are allowed to create files in a group project. However, a user can delete or rename only those files created by him in the group project. Group members can edit all files of a group project irrespective of the owner of the file being edited. In order to keep track of all files created in a group project, a log is created in the database when a file is created by a group member. The log keeps track of the creator of the file, user who last edited the file and the time stamp at which the file is edited.

All the above mentioned functions let users access other users' project contents. All the group members except the project owner are prohibited from performing certain actions like deleting, renaming or moving projects to a different location. Only the owner of the project has all permissions on the project’s data.

4.2 Data Sharing

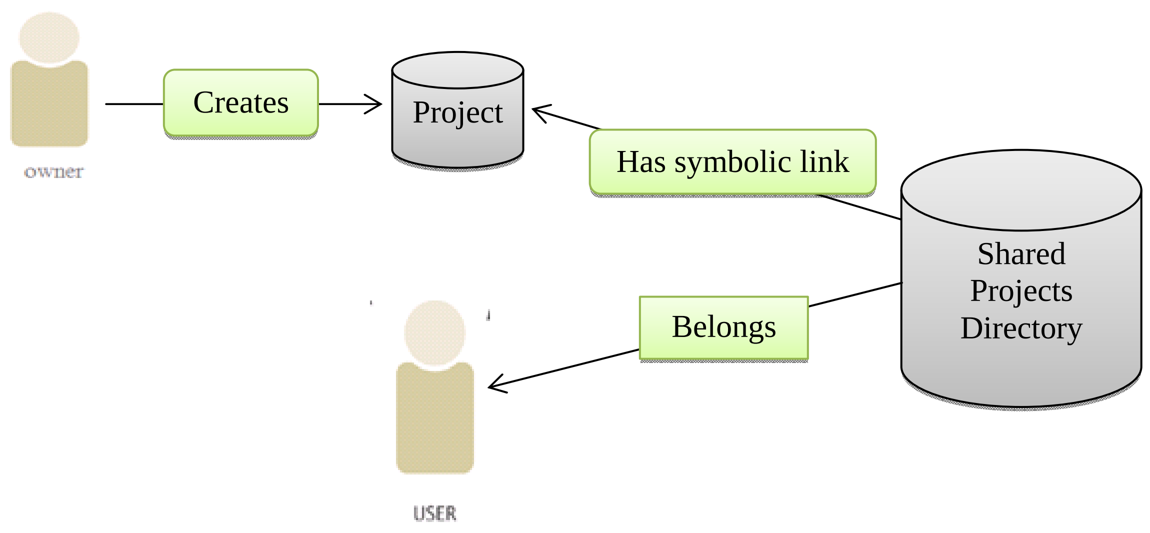
Similar to group projects, shared projects also contribute towards collaborative research. In some cases users prefer to share different files to individual users rather than creating a group and sharing data to the group. File sharing in wEMBOSS is the concept of giving access rights on individual files to specific users by the owner of the project. When a file is shared, the user with whom the file is shared can view the file. Edit permission on a shared file gives the sharer[12] the ability to make changes in the shared file. Sharers can also use a shared file as input to a wEMBOSS program. Functions available for sharing data are discussed in the subsequent paragraphs.

Users can choose any other user(s) in the system and share selected files with them using the *Share Files* command. When using the Share Files command, a list of all the users present in the wEMBOSS system is displayed. Users can either be selected by scrolling down the list or by typing their names in the search box present. As already explained usernames are populated dynamically when typed in the search box. Usernames of all the users in the system are retrieved

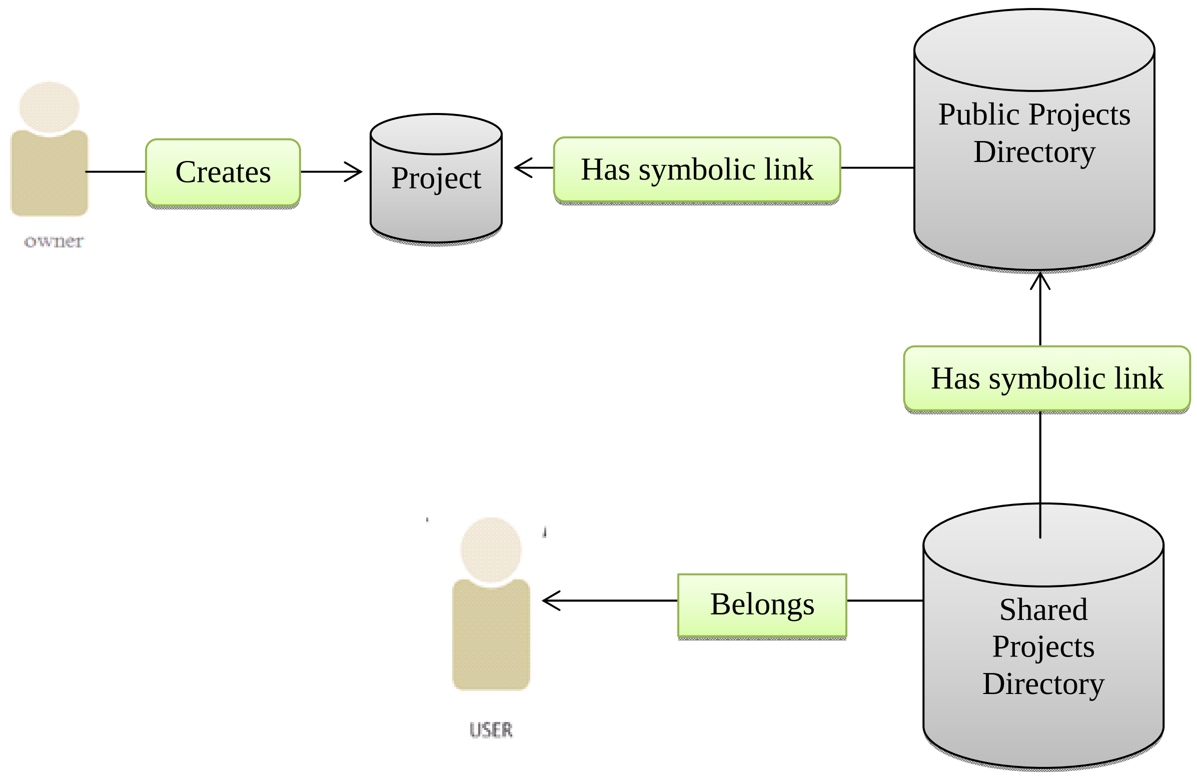
[12] *Sharer is a user with whom files are shared by the owner of a project*

into an array and the list is dynamically populated from this array. An option of sharing files with everyone in the system is also available. Projects with files shared to everyone in the system are called *Public Projects*. After selecting a particular user with whom files are to be shared, a list of all files belonging to the current project is displayed. The files list is populated by running a search on the current project directory. This listing uses a Depth First Search, commonly known as DFS algorithm. If a symbolic link to the shared project already exists in the *Shared Projects* directory of the sharer, then the files selected are simply stored in the database. If not, a link is first created and then an entry is made in the database. Sharer can now see the shared project in his list of projects. Before displaying contents of a shared project to the sharer, the database is queried to get the list of files of the shared project. Similar process is followed while sharing a project result with a sharer. The only difference is that a project result is shared only for viewing purpose and not editing. Owner of a project is also provided with an option to view all the sharers of the project. Figure 4.2 shows sharing of projects to individual users in wEMBOSS.

.



(a) Individual Users



(b) All Users in wEMBOSS

Figure 4.2: Pictorial Representation of Sharing a Project

The owner of a project can edit permissions on files shared with sharers at any point of time. To edit permissions on a file, the owner simply selects another user from the users list and clicks on *Edit Permissions* button. A new list is populated with a set of files shared with the user selected from the users list. Shared files along with the *write* permissions given on respective files are displayed. In this case, the database is used to get the list of files of a project along with their corresponding permissions shared with the sharer. Check boxes are provided so that the project owner can change permission on the shared files. Any changes made are updated in the database. These updates are atomic. Either all updates are made or none. If an error occurs, appropriate messages are displayed to the user.

Files shared with a user can be withdrawn in a similar fashion to editing permissions. A user has to first select another user from the list and click on *Unshared Files* button to unshare a file from the user selected. The list is then populated with files shared with the selected user. After the user selects files to unshare, changes are reflected in the database. If all the files of a particular shared project are unshared, the link to this shared project is broken. There might also be some exceptional situations wherein a user unknowingly deletes files or projects shared with other users. In such situations sharers are displayed empty shared projects. It might also leave symbolic links dangling and corresponding data in the database untouched and obsolete. wEMBOSS has a smart way of handling such situations. Every time a user tries to access a shared project, the link to the project is checked for its validity. Any dangling links are removed and empty shared projects are not displayed in the projects list of a sharer. wEMBOSS also checks if files of a shared project exist. If a shared file is deleted by the owner, then its corresponding entry is deleted from the database.

Chapter 5. System Administration

System administration is necessary for wEMBOSS to effectively manage users' data both in the data repository as well as in the database. Duties of a wEMBOSS administrator include, but are not limited to setting limitations on various user activities, discarding undesirable data, especially data that belongs to users deleted from the system. As mentioned previously, users might misuse the system if no limitations are set on various activities. Obsolete data uses storage space which otherwise can be used for storing useful information. This chapter discusses each of the administrative functions.

5.1 User Limitations

Setting up user limitations is one of the major tasks in wEMBOSS. These limitations prevent users from exploiting the system. User limitations are stored in the database as a single entry. A structure called limitations is defined in the wEMBOSS header file. An array of this structure that contains information of each of the user limitations, acts as a lookup table. Information of a user limitation includes number of bytes used to store the limitation value in the database, offset[13] of the limitation value in the database entry, and description of the limitation value.

There are six user limitations defined, each of which are assigned a default value. As already mentioned, administrator can change the limitation values at any time. The six user limitations are namely maxProjsPerUser, maxGrpsPerProj, maxUsersPerGrp, maxProjsPerGrp, maxShrFilesPerPerson, and maxSharersPerProj.

maxProjsPerUser limitation holds the value of maximum projects a user can create. This limitation controls the user from creating numerous projects/subprojects. The

[13] *Offset variable holds the position of the limitation value considering the start position to have the value zero.*

maxGrpsPerProj limitation holds the maximum number of groups that can be associated with a single project. Associating a large number of groups to a project creates a multiple number of links to this project and also requires a lot of storage space for storing related data in the database. The *maxUsersPerGrp* is a limitation on the maximum users that a group can have. The *maxProjsPerGrp* is a limitation for maximum projects that can be assigned to a group. This is different from *maxGrpsPerProj* limitation where in the *maxProjsPerGrp* limitation, more than one project can be assigned to a single group and in *maxGrpsPerProj* limitation, more than one group can be associated with a single project. *maxShrFilesPerPerson* is set to the maximum number of files of a shared project that can be shared with a single user. This controls the user from sharing a large number of files to a single user. *maxSharersPerProj* limitation restricts the owner of a shared project from sharing files with more users than the limitation value assigned.

Corresponding limitations are checked each time a user performs an operation. Exceeding any of the limitation values stops the user from performing the operation and displays an appropriate error message.

5.2 Data Management

The Administrator has several authorities other than setting up limitations, which help in managing data in the database effectively. *Delete Group*, *Unassign Group* and *Unshare Project* are a few among several operations of an administrator. All the administrator tools are developed in the C programming language and are executed on the command line interface. The administrator interface provides a set of options, each one corresponding to a specific function. These functions are especially important in instances where a user is to be removed from the system. Clearing wEMBOSS of obsolete data saves space in the database that can otherwise be used to store necessary data. Consider a situation where a user who previously shared a project with another user is no longer using the system. Removal of data related to shared files/projects from the database and unlinking of symbolic links are done automatically as discussed in Chapter 4. There might also be situations where both the owner of a shared project and the sharer are no longer using the system. In such cases, removal of these users' working directories gets rid of all the links between these directories, but the data in the database remains untouched. Such data may get accumulated using up space for storage and potentially slows the system. Explicit removal of data directly from the database is therefore required. Operations of an administrator mentioned above are explained in the subsequent paragraphs.

A group can be unassigned from a project by an administrator by the function *Unassign Group*. This operation precedes deletion of a group. All the groups assigned to a project are stored in the database with project name as the key. The *Unassign Group* operation removes a group from the list of assigned groups to a project. Project name and the group name are provided as inputs in this case. An option for removal of all groups associated with a project is also provided.

The *Delete Groups* function is used to delete a specific group. The delete group function deletes all data associated with a particular group in the database. The group name is provided as input to this program. A group has to be unassigned from a project(s), if previously assigned to any, before deleting it.

Unshare Project is another helpful operation which gets rid of unnecessary data. Projects can be unshared from an individual user or from the public. This operation erases all the information related to projects/files shared with a specific user or public. Project names and user names (in case of an individual user) are provided as inputs. In any situation where the operation is not successful appropriate messages are displayed.

Another important option that is provided in the administrator interface is to set user limitations. The six user limitations, previously discussed are displayed as options. The administrator can select any of these options to change the value of the corresponding user limitation.

Chapter 6. Working with Enhanced wEMBOSS

Working of the enhanced wEMBOSS with usage examples are explained in this section. This application is hosted by University of Missouri and can be accessed by anyone who belongs to an organization within the list of federated organizations of the Great Plains Network virtual organization and has the proper entitlements needed for this application.

Since the wEMBOSS service is a web based application, it can be accessed on the web with the URL https://web.rnet.missouri.edu/wEMBOSS. This page is now redirected to the WAYF service where the user selects his home institution. Let us assume the user selects University of Missouri System as the home intitution. The page is redicted to University of Missouri authentication page. After successful authentication and authorization checks the project access page is displayed as shown in the Figure 6.1. Clicking on 'Project Access' button redirects the page to the wEMBOSS application where the user can access EMBOSS programs and share projects with other users. Figure 6.1 shows the home page of the application.

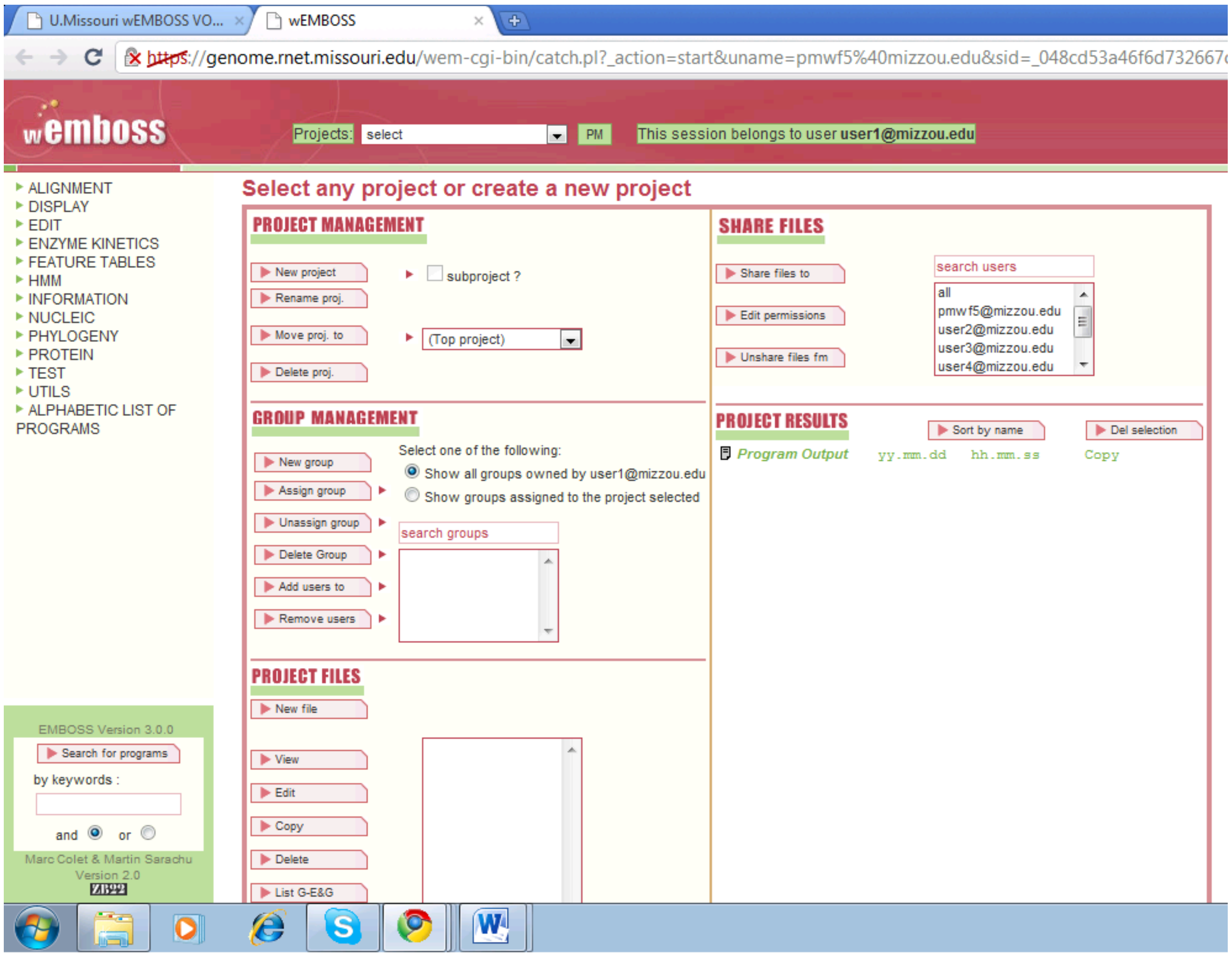
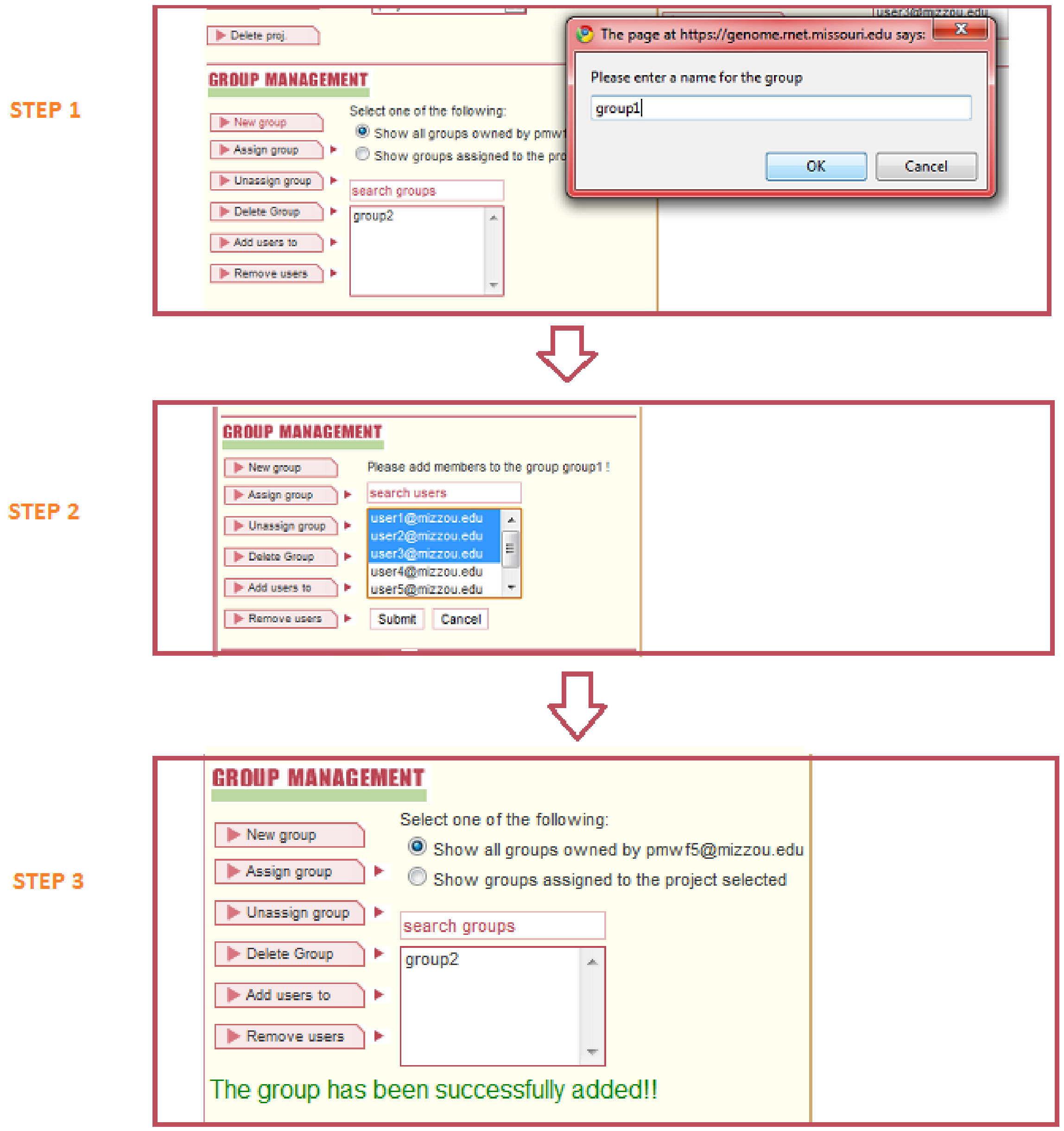


Figure 6.1 wEMBOSS Home Page

Group Management

Functioning of each of the group management functions are demonstrated with examples. For simplicity let us assume the user, pmwf5@mizzou.edu, has already created two projects namely ‘project1’ and ‘project2’.

New Group: Creating a group is the primary step for sharing projects with a group. Clicking on the ‘New Group’ button prompts the user to enter the group name. The user now selects members of the group and clicks on submit to create the group. Limitations apply and the group name is checked against the database to see if a group with same name exists. After the group is created it appears in the list of groups. Figure 6.2 shows the step by step process of group creation.



The page at https://genome.rnet.missouri.edu says:

Please enter a name for the group

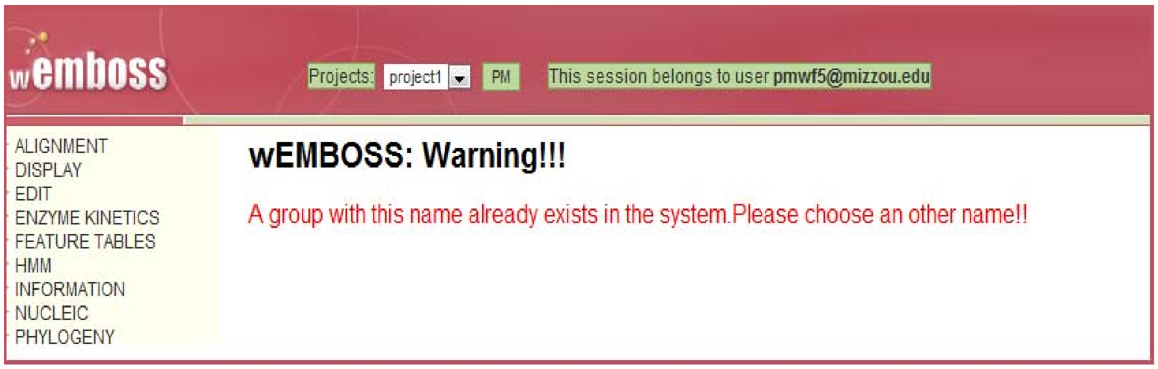
group1

Please add members to the group group1 !

The group has been successfully added!!

Figure 6.2: Creation of New Group

Figure 6.3 shows the warning messages displayed in case of any violations while creating a group.

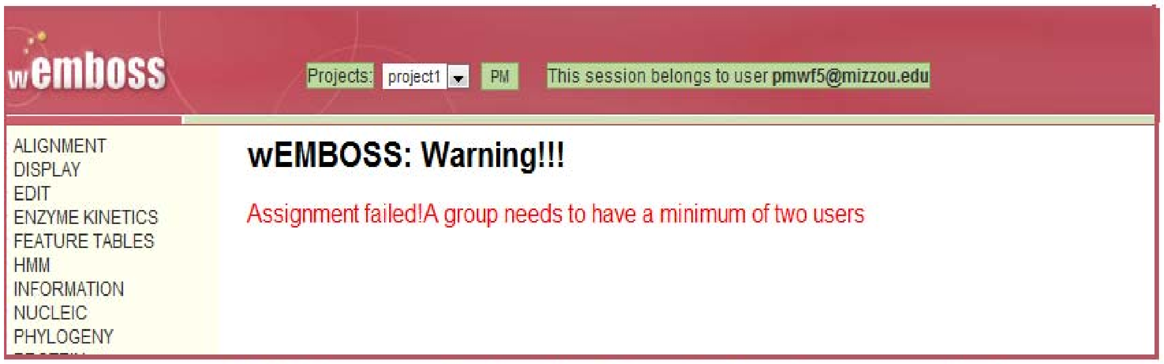


This session belongs to user pmwf5@mizzou.edu

wEMBOSS: Warning!!!

A group with this name already exists in the system.Please choose an other name!!

a) Warning message displayed when an already exiting group name is chosen



This session belongs to user pmwf5@mizzou.edu

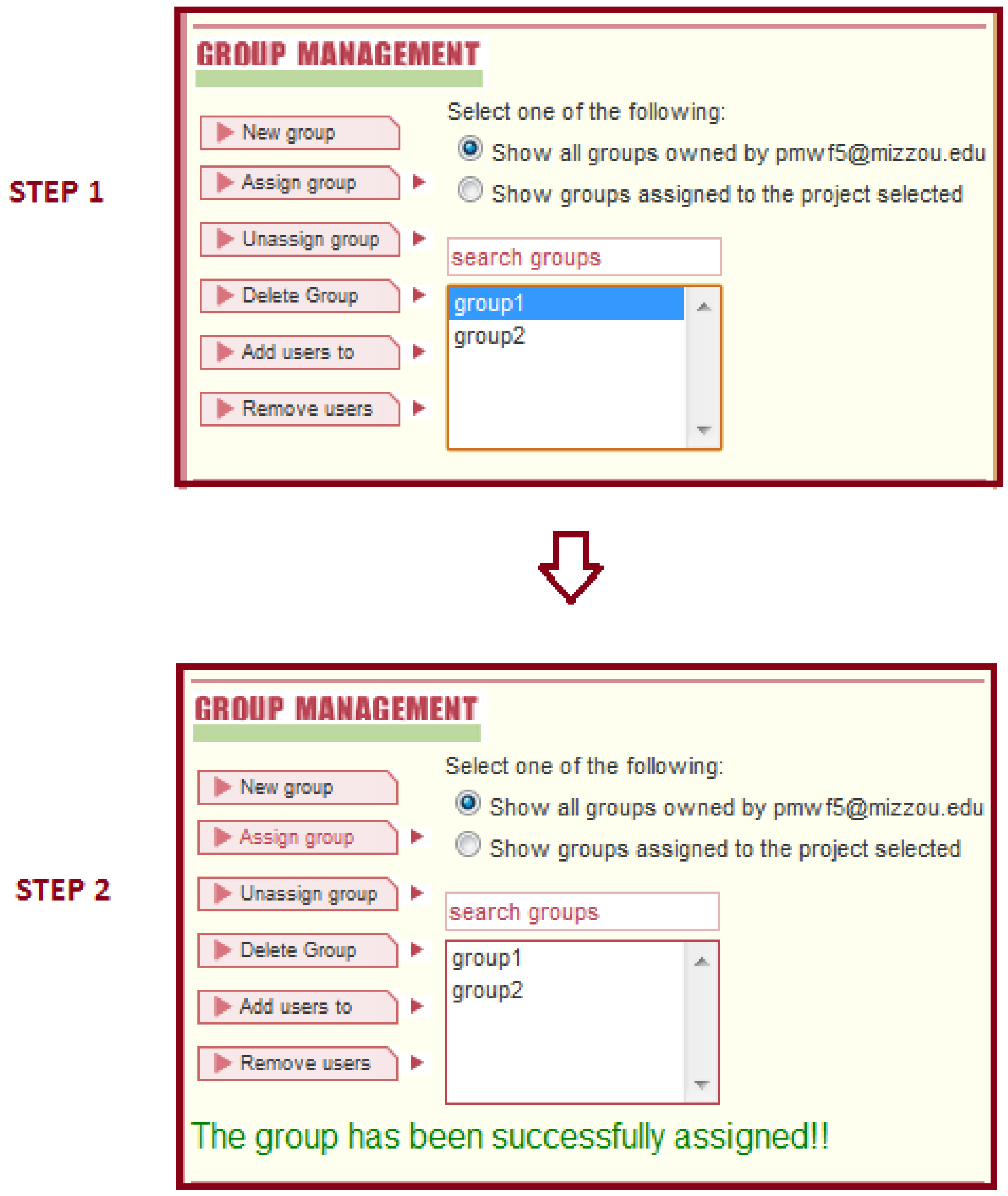
wEMBOSS: Warning!!!

Assignment failed!A group needs to have a minimum of two users

b) Warning message displayed when the number of users in a group are less than minimum number

Figure 6.3: Warnining Messages Displayed in Case of Group Creation

Assign Group: The user needs to associate a group with a project in order to share this project with the group members. The project assigned is called a group project. After selecting the group to be assigned from the group list and the project to be associate with from the project list in the title section, the 'Assign Group' button is clicked. Not selecting any one of the two results an error and is displayed to the user. The group is now assigned to the project and the group members can access the contents. Figures 6.4 and 6.5 show the process of assigning a group and the error messages displayed in case of any violations.


STEP 1

GROUP MANAGEMENT

Select one of the following:

[✓ Show all groups owned by pmwf5@mizzou.edu]

Show groups assigned to the project selected

search groups

group1

group2

STEP 2

GROUP MANAGEMENT

Select one of the following:

[✓ Show all groups owned by pmwf5@mizzou.edu]

Show groups assigned to the project selected

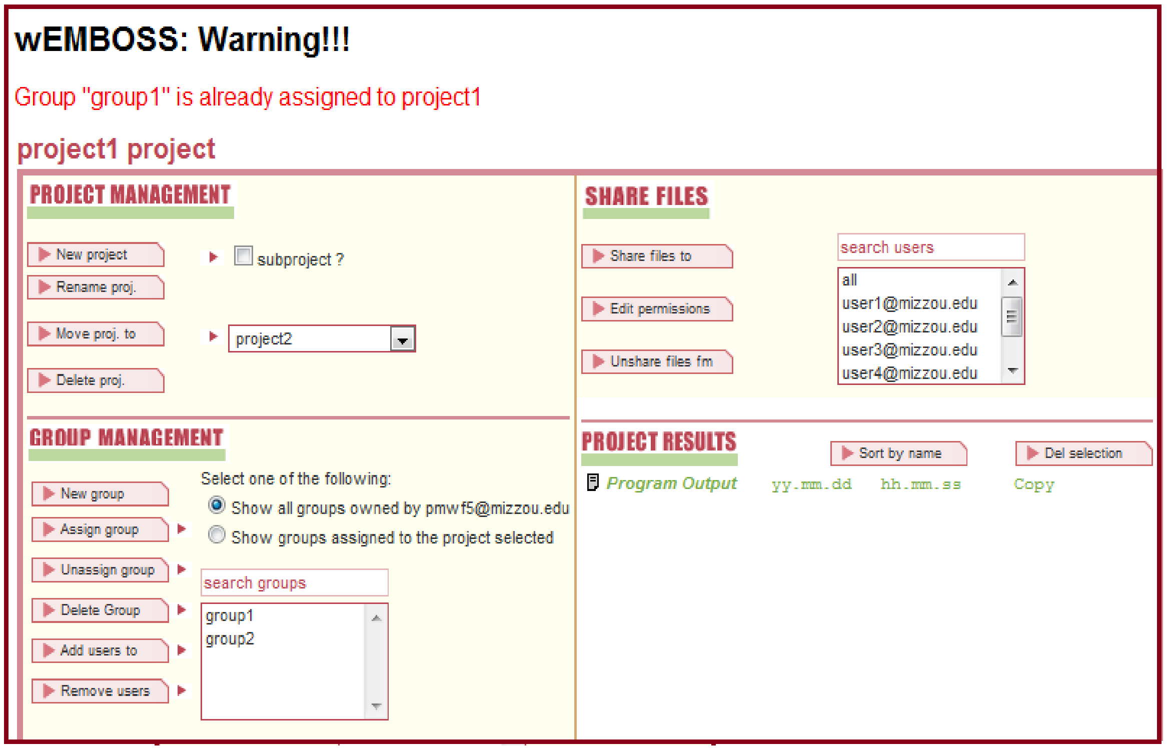
search groups

group1

group2

The group has been successfully assigned!!

Figure 6.4: Assignment of a Group to a Project


wEMBOSS: Warning!!!

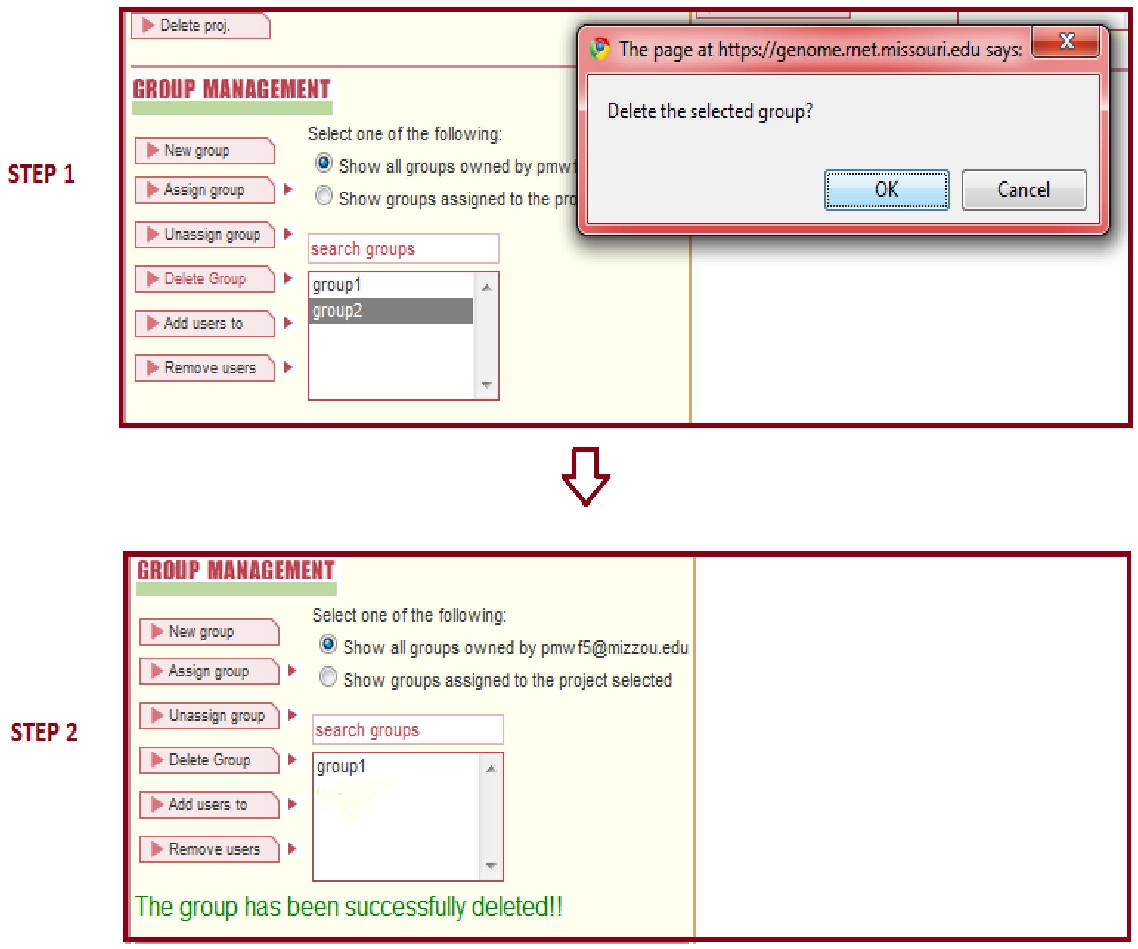
Group "group1" is already assigned to project1

project1 project

Figure 6.5: Message Displayed When a Group is Assigned More Than Once

Unassign Group: Withdrawal of a group from a project can be done in a similar faashion to assigning a group. The user needs to select a project and a group to be unassigned and click on the ‘Unassign Group’ button. Any error messages or success messages are displayed.

Delete Group: The delete group function deletes the selected group. Irrespective if the group being assigned to a project or not, it can be deleted. The user is prompted if he is sure of deleting the group. Any group assigned to a project would be first unasassigned and then deleted. Figure 6.6 shows the deletion of a group.



The page at https://genome.rnet.missouri.edu says:

Delete the selected group?

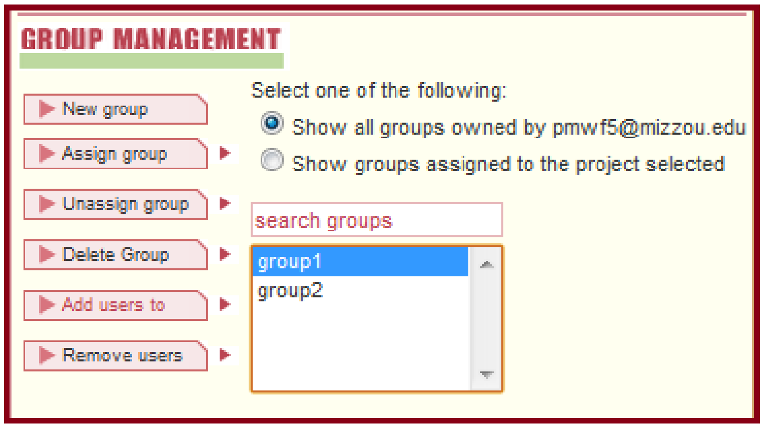
Select one of the following:
- Show all groups owned by pmwf5@mizzou.edu
- Show groups assigned to the project selected

The group has been successfully deleted!!

Figure 6.6: Deletion of a Group

Add Users: Add Users function lets users add members to a group any time.There is again a limitation to the number of users that a group can contain. In order to add new users to a group, the user first selects the group and then clicks on the ‘Add Users’ button. This brings up a list of users not already present in the group. In our case group ‘group1’ has users ‘user1@mizzou.edu’ and ‘user2@mizzou.edu’.The list now contains all users in the system except these two users. Selecting the users and clicking ‘Submit’ adds these users to the group. If the number of users exceeds the limitation of a group an error message is displayed. The process of adding users to a group is shown in the Figure 6.7.

STEP 1



Select one of the following:

[✓ Show all groups owned by pmwf5@mizzou.edu]

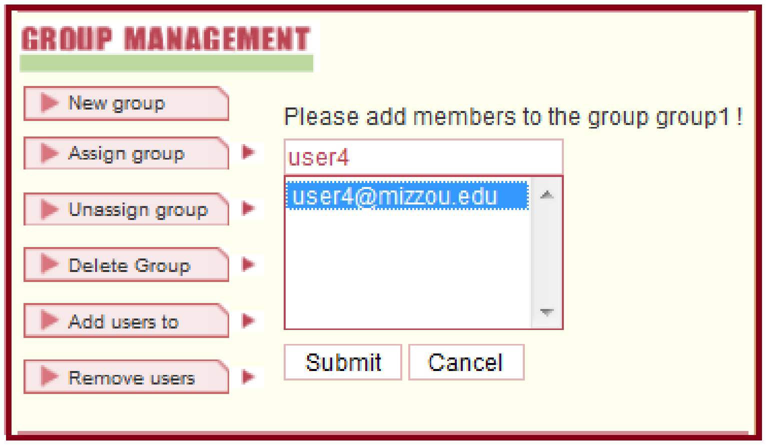
Show groups assigned to the project selected

search groups

group1

group2

STEP 2

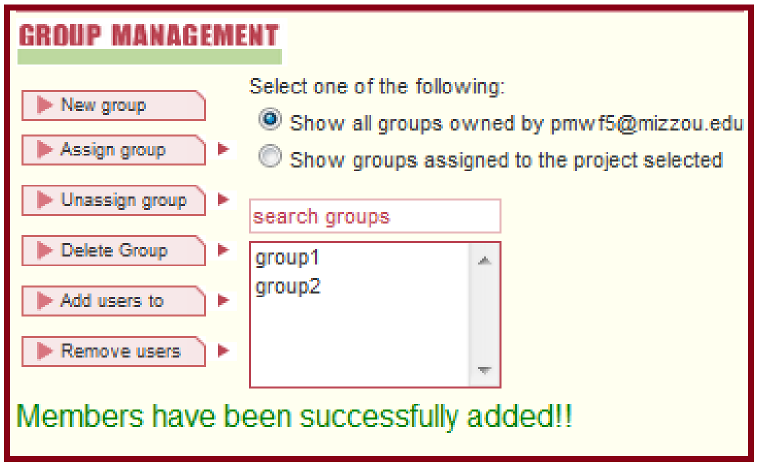


Please add members to the group group1 !

user4

user4@mizzou.edu

STEP 3



Select one of the following:

[✓ Show all groups owned by pmwf5@mizzou.edu]

Show groups assigned to the project selected

search groups

group1

group2

Members have been successfully added!!

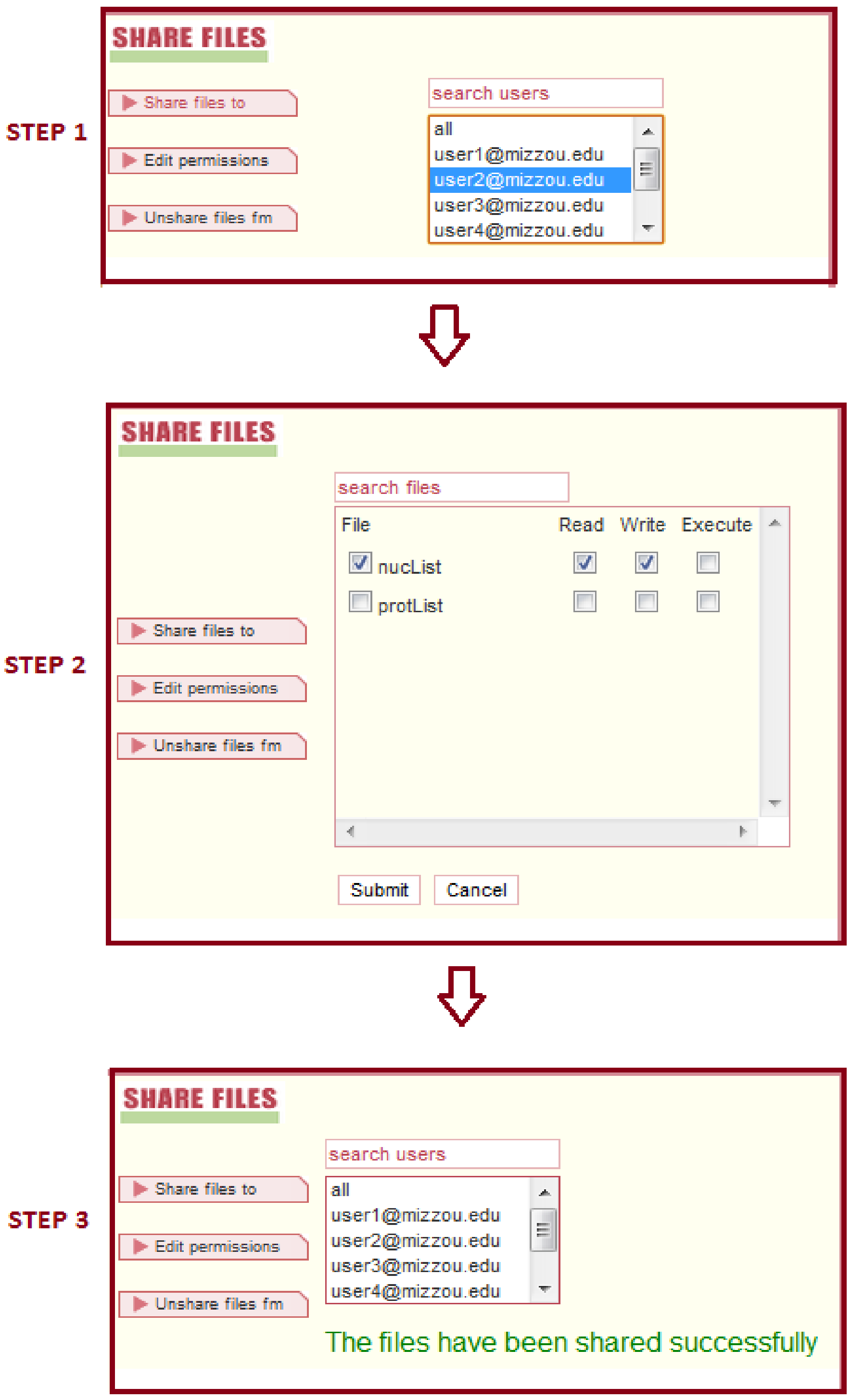
Figure 6.7: Adding Users to a Group

Delete Users: Deleting users from a group can be done in a similar fashion to adding users. The user first selects the group and clicks on the 'Delete Users' button.This shows a list of users present in the group. The user selects the users and clicks submit to delete the users from the group. An error message is displayed if the user tries to delete all or less than minimum numberof users, that is two, from the group.

Share Files

Sharing of files section in wEMBOSS has three funtions, namely 'Share Files', 'Edit Permisssions' and 'Unshare Files'. Working of these functions is demonstrated with examples in furthur paragraphs.

Share Files: This function lets users share files of a project to other users. As already discussed, such projects are called shared projects. In order to share files with anyone the user first selects the project from which files are to be shared. The Share files section displays a list of users in the system, from which a user can be picked by just typing the user name in the search box or selecting the user from the list directly. After selecting a user, clicking on 'Share Files to' button again displays a list of files available in the project. The user then selects the files to be shared and the permissions on these files. Not selecting a user and clicking on the share files button gives a warning message to the user. Like in the case of groups, error messages are displays if the user tries to share a large number of files with other users. Figure 6.8 shows the process of sharing files.



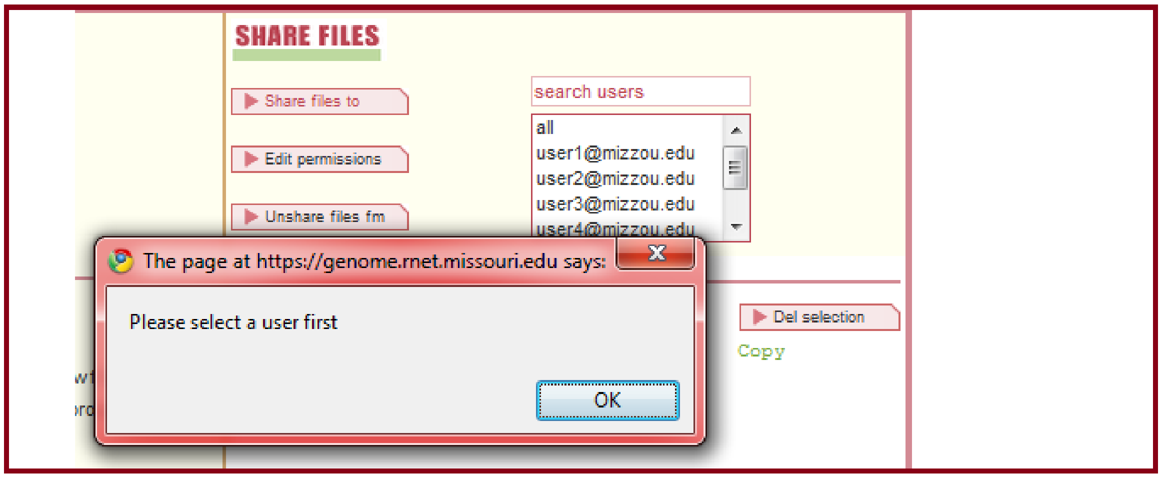
all
user1@mizzou.edu
user2@mizzou.edu
user3@mizzou.edu
user4@mizzou.edu

File	Read	Write	Execute
✓ nucList	✓	✓	☐
☐ protList	☐	☐	☐

all
user1@mizzou.edu
user2@mizzou.edu
user3@mizzou.edu
user4@mizzou.edu

The files have been shared successfully

Figure 6.8: Process of Sharing Files

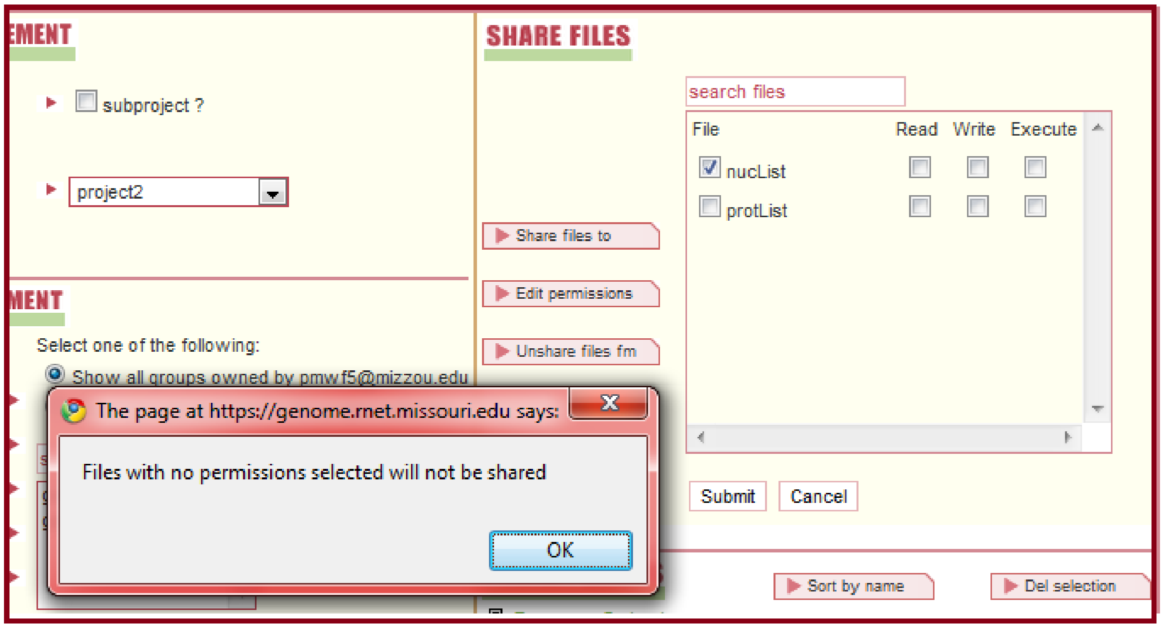


The page at https://genome.rnet.missouri.edu says:

Please select a user first

OK

a) Message displayed while sharing files without selecting a user



The page at https://genome.rnet.missouri.edu says:

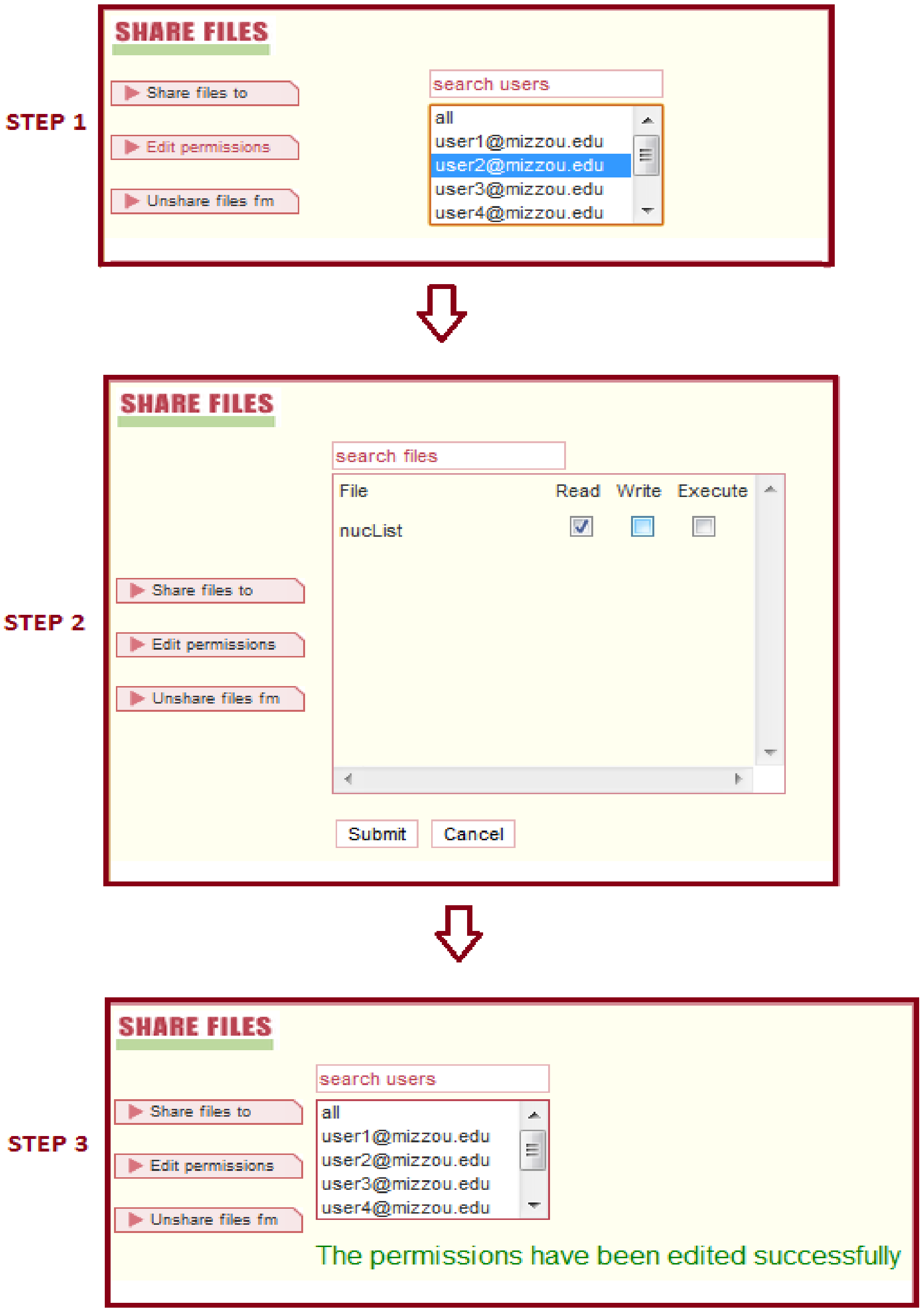
Files with no permissions selected will not be shared

OK

b) Message displayed while sharing files without any permissions set

Figure 6.9: Violations in File Sharing

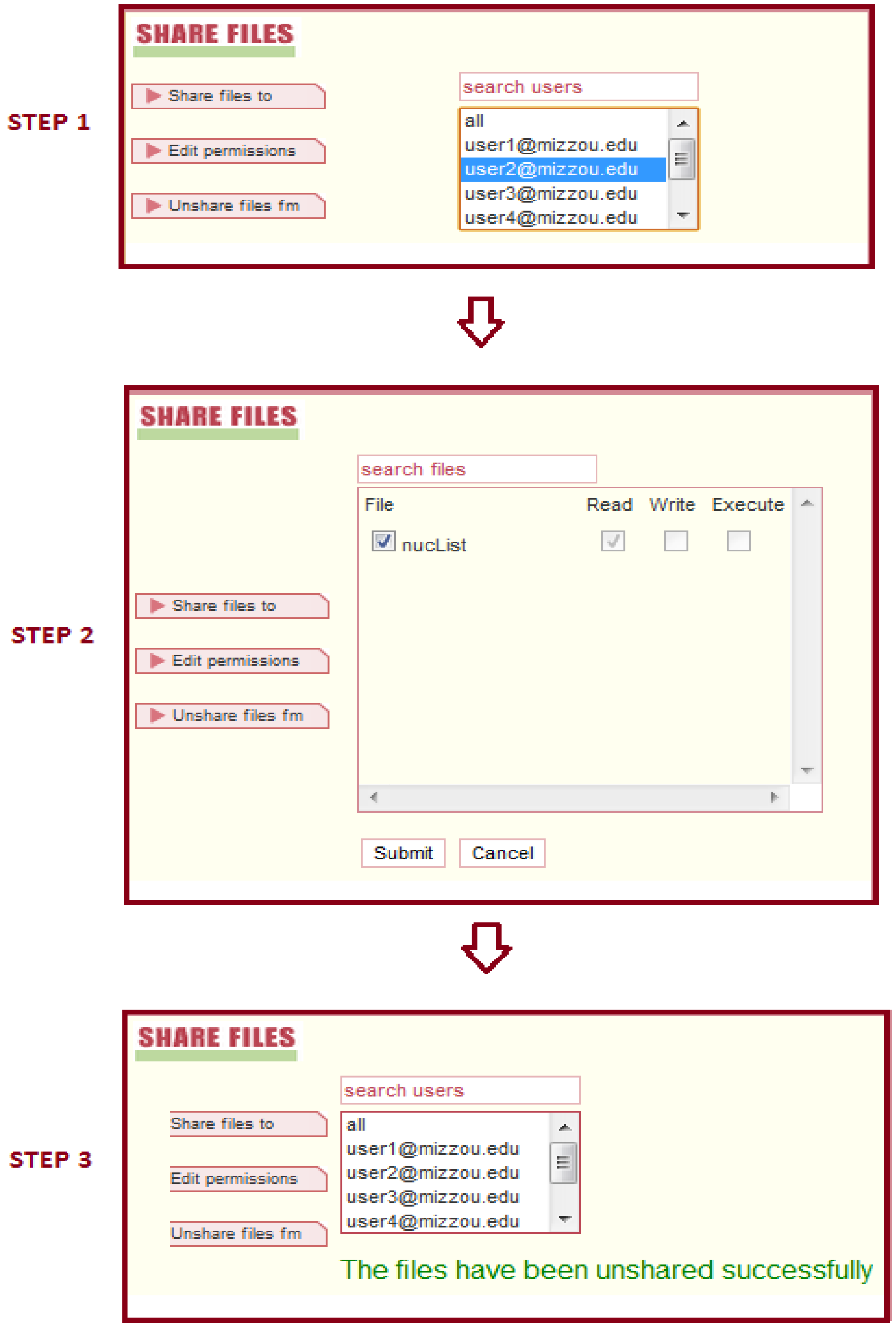
Edit Permissions: Permissions given on a file can be edited using this feature. Like in the case of sharing a file, editing permissions on a file shared to a person can be done by selecting a person first. If any files are previously shared to this person, they appear in a list with permissions given on these files. Permissions can be changed by selecting either read, write or execute check box. Figure 6.10 shows the process of editing permissions on a file already shared to the user shown in the previous example.



The permissions have been edited successfully

Figure 6.10: Editing Permissions of a Shared File

Unshare Files: Unshare files from anyone is similar to sharing of files.This process is shown in the Figure 6.11.



The files have been unshared successfully

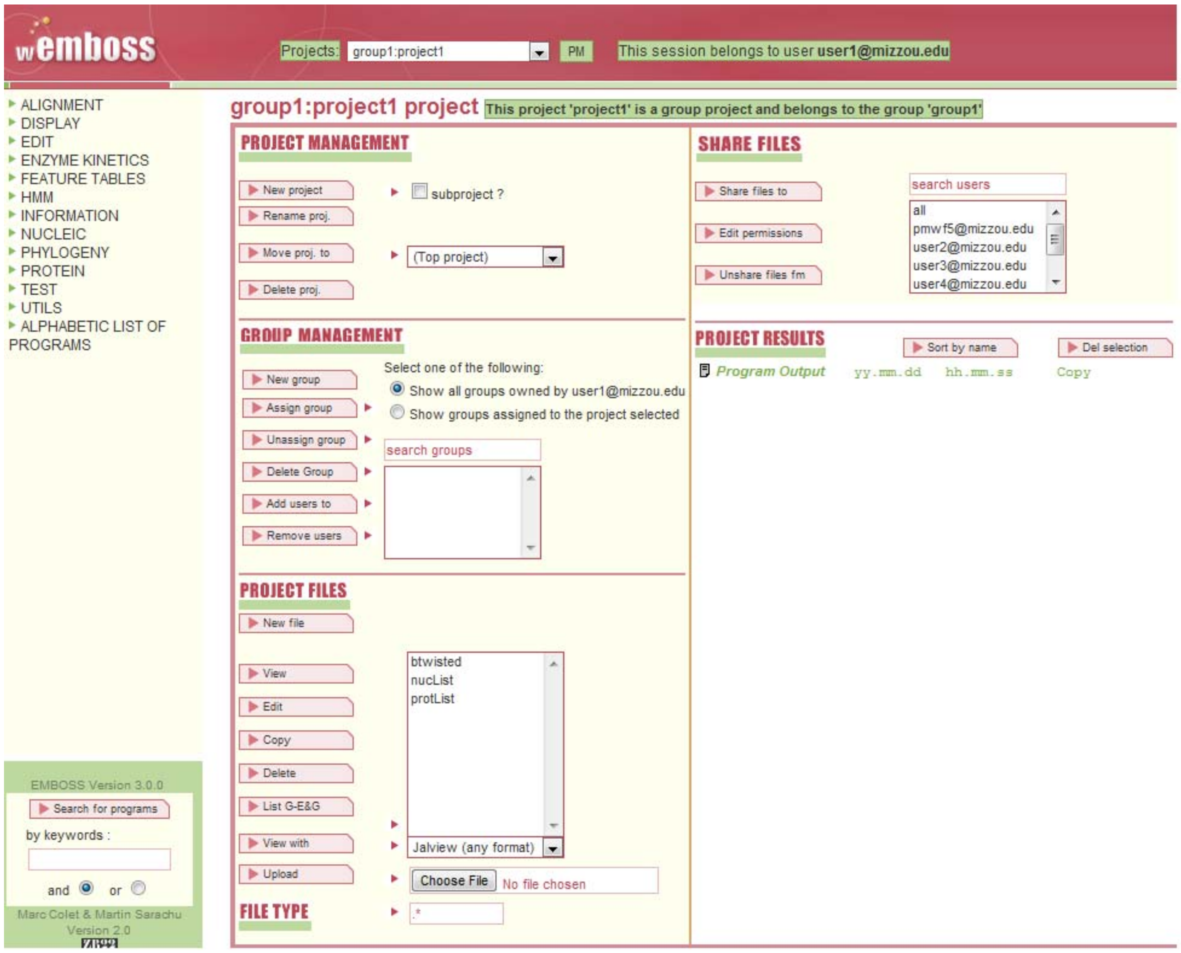
Figure 6.11: Unshare Files From a User

All the above examples are from the user's perspective who owns the projects/files and how he can share projects with other users. Further examples show how a user, with whom projects have been shared, are able to use the shared files.

Group projects

In the previous examples, user 'pmwf5@mizzou.edu' has shared assigned group 'group1' to project 'project1'. Any user who belongs to group1, can view and use the files in this project for research. The following examples are based on the users' perspective who belongs to group1.

When the user selects a group project from the list of projects it appears in the format 'GroupName:ProjectName', which indicates that the project belongs to a group. Also for the convience of the user, a message is displayed to let the user know the project details. Figure 6.12 shows the home page when the user selects a group project.


Projects: group1:project1

This session belongs to user **user1@mizzou.edu**

group1:project1 project **This project 'project1' is a group project and belongs to the group 'group1'**

Select one of the following:
[✓ Show all groups owned by user1@mizzou.edu]
Show groups assigned to the project selected

(Top project)

all
pmwf5@mizzou.edu
user2@mizzou.edu
user3@mizzou.edu
user4@mizzou.edu

Program Output yy.mm.dd hh.mm.ss Copy

btwisted
nucList
protList

Jalview (any format)

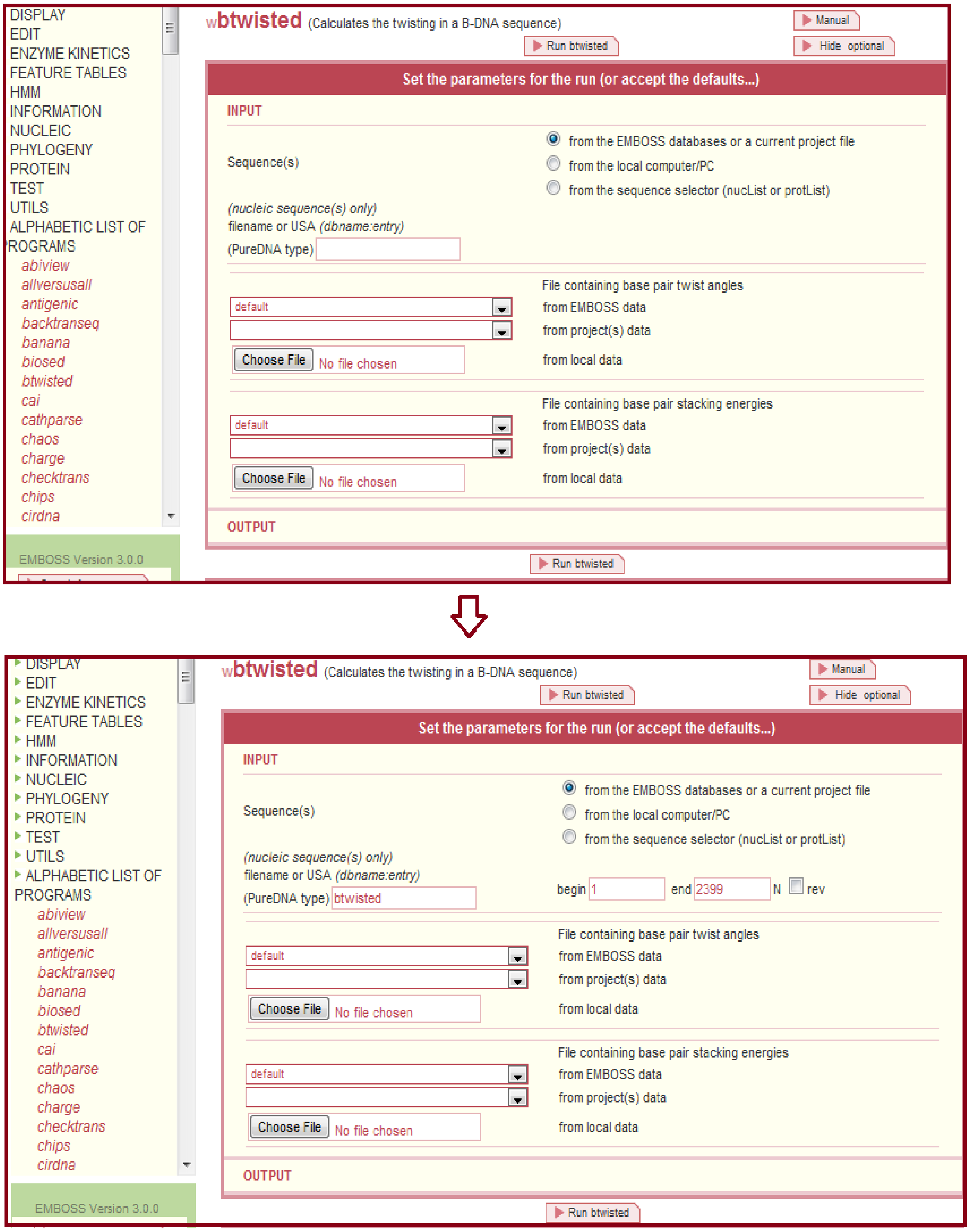
Choose File No file chosen

EMBOSS Version 3.0.0
Marc Colet & Martin Sarachu Version 2.0

Figure 6.12: wEMBOSS Home Page From a Group User's Perspective

From the Figure 6.12, it is evident that all the files of project1 appear in the Project Files section. The user can now work with these files and results obtained appear in the Project Results section, which again can be viewed by all other group members.

As an example, let us run a program named *btwisted* and use the file *btwisted* as an input.This program takes a region of a pure DNA sequence and calculates by simple arithmetic the probable overall twist of the sequence and the stacking energy. The input can be given from an EMBOSS database, a current project file or file from the local computer. In Figure 6.13, the user uses a group project file and runs a program to get an output.



wbtwisted (Calculates the twisting in a B-DNA sequence)

Set the parameters for the run (or accept the defaults...)

INPUT

Sequence(s) [✓ from the EMBOSS databases or a current project file] from the local computer/PC from the sequence selector (nucList or protList)

(nucleic sequence(s) only) filename or USA (dbname:entry) (PureDNA type)

File containing base pair twist angles: default from EMBOSS data; from project(s) data; No file chosen from local data

File containing base pair stacking energies: default from EMBOSS data; from project(s) data; No file chosen from local data

OUTPUT

wbtwisted (Calculates the twisting in a B-DNA sequence)

Set the parameters for the run (or accept the defaults...)

INPUT

Sequence(s) [✓ from the EMBOSS databases or a current project file] from the local computer/PC from the sequence selector (nucList or protList)

(nucleic sequence(s) only) filename or USA (dbname:entry) (PureDNA type) btwisted

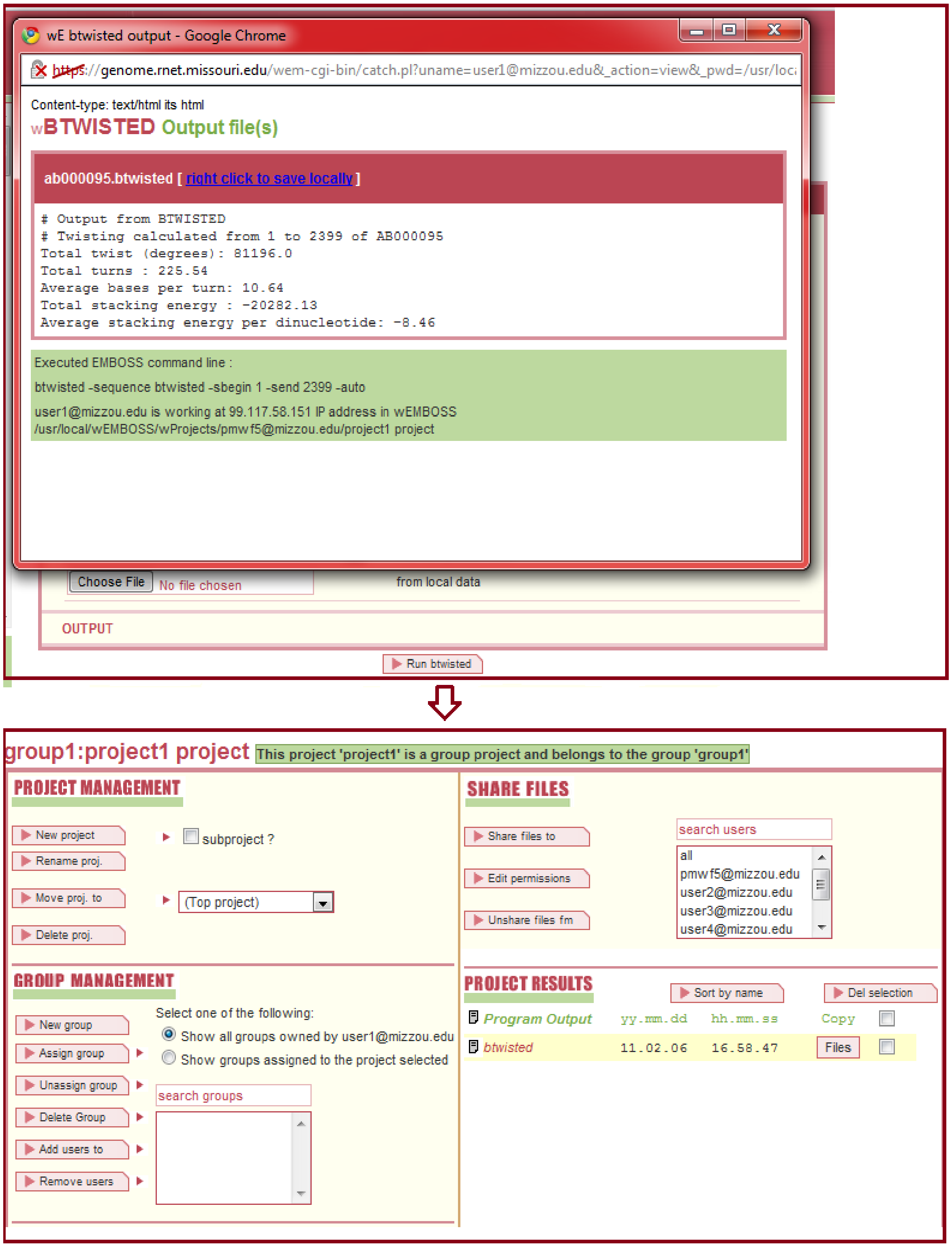
begin 1 end 2399 N rev

File containing base pair twist angles: default from EMBOSS data; from project(s) data; No file chosen from local data

File containing base pair stacking energies: default from EMBOSS data; from project(s) data; No file chosen from local data

OUTPUT

Figure 6.13: Executing EMBOSS Program Using Group Project Files



Content-type: text/html its html

wBTWISTED Output file(s)

ab000095.btwisted [right click to save locally]

```
# Output from BTWISTED
# Twisting calculated from 1 to 2399 of AB000095
Total twist (degrees): 81196.0
Total turns : 225.54
Average bases per turn: 10.64
Total stacking energy : -20282.13
Average stacking energy per dinucleotide: -8.46
```

Executed EMBOSS command line :

btwisted -sequence btwisted -sbegin 1 -send 2399 -auto

user1@mizzou.edu is working at 99.117.58.151 IP address in wEMBOSS
/usr/local/wEMBOSS/wProjects/pmwf5@mizzou.edu/project1 project

group1:project1 project This project 'project1' is a group project and belongs to the group 'group1'

PROJECT MANAGEMENT

SHARE FILES

all
pmwf5@mizzou.edu
user2@mizzou.edu
user3@mizzou.edu
user4@mizzou.edu

GROUP MANAGEMENT

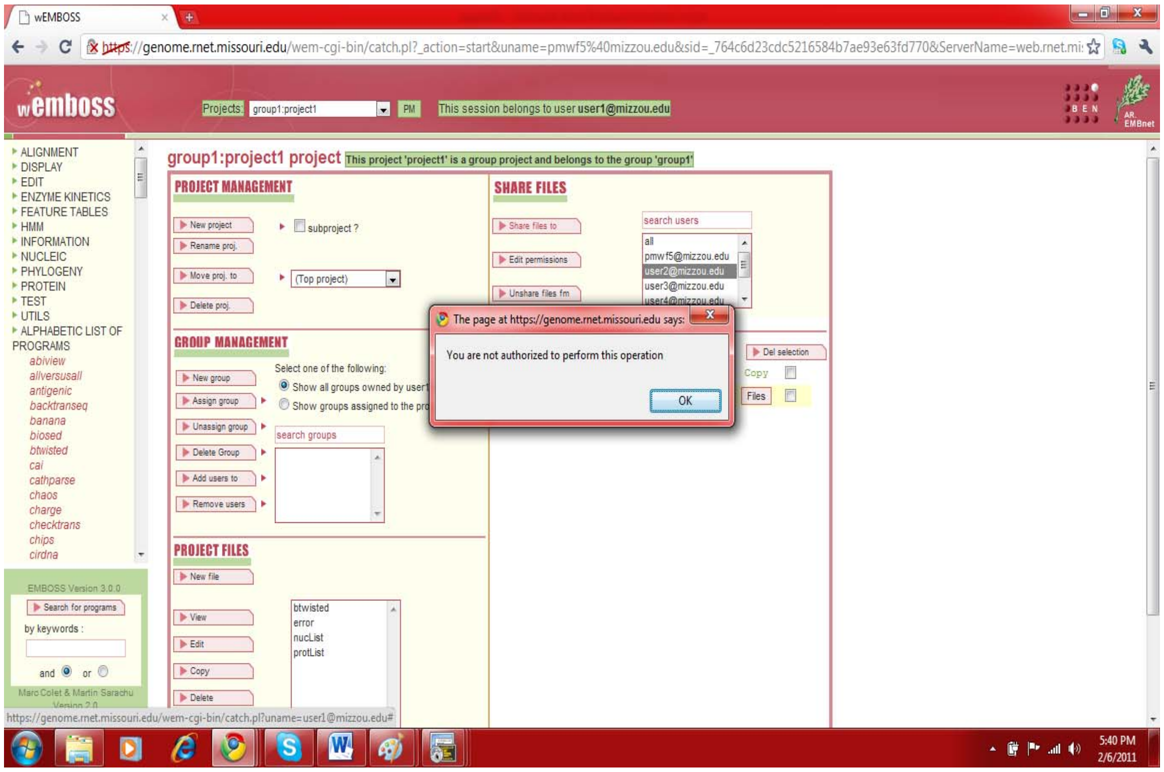
Select one of the following:
[✓ Show all groups owned by user1@mizzou.edu]
Show groups assigned to the project selected

PROJECT RESULTS

Program Output	yy.mm.dd	hh.mm.ss	Copy	☐
btwisted	11.02.06	16.58.47	Files	☐

Figure 6.14: Project Results Display

Figure 6.15 shows the results obtained after running the program btwisted.This result appears in the Project Results section. The output file can also be copied to another project. Though the group users have all access rights on the project contents, they do not have permissions to share or delete the project contents. They are also not authorized to rename, delete or move the project. Performing these operations alerts the users that they are not authorized.


https://genome.rnet.missouri.edu/wem-cgi-bin/catch.pl?_action=start&uname=pmwf5%40mizzou.edu&sid=_764c6d23cdc5216584b7ae93e63fd770&ServerName=web.rnet.mi

Projects: group1:project1

This session belongs to user **user1@mizzou.edu**

group1:project1 project This project 'project1' is a group project and belongs to the group 'group1'

The page at https://genome.rnet.missouri.edu says:

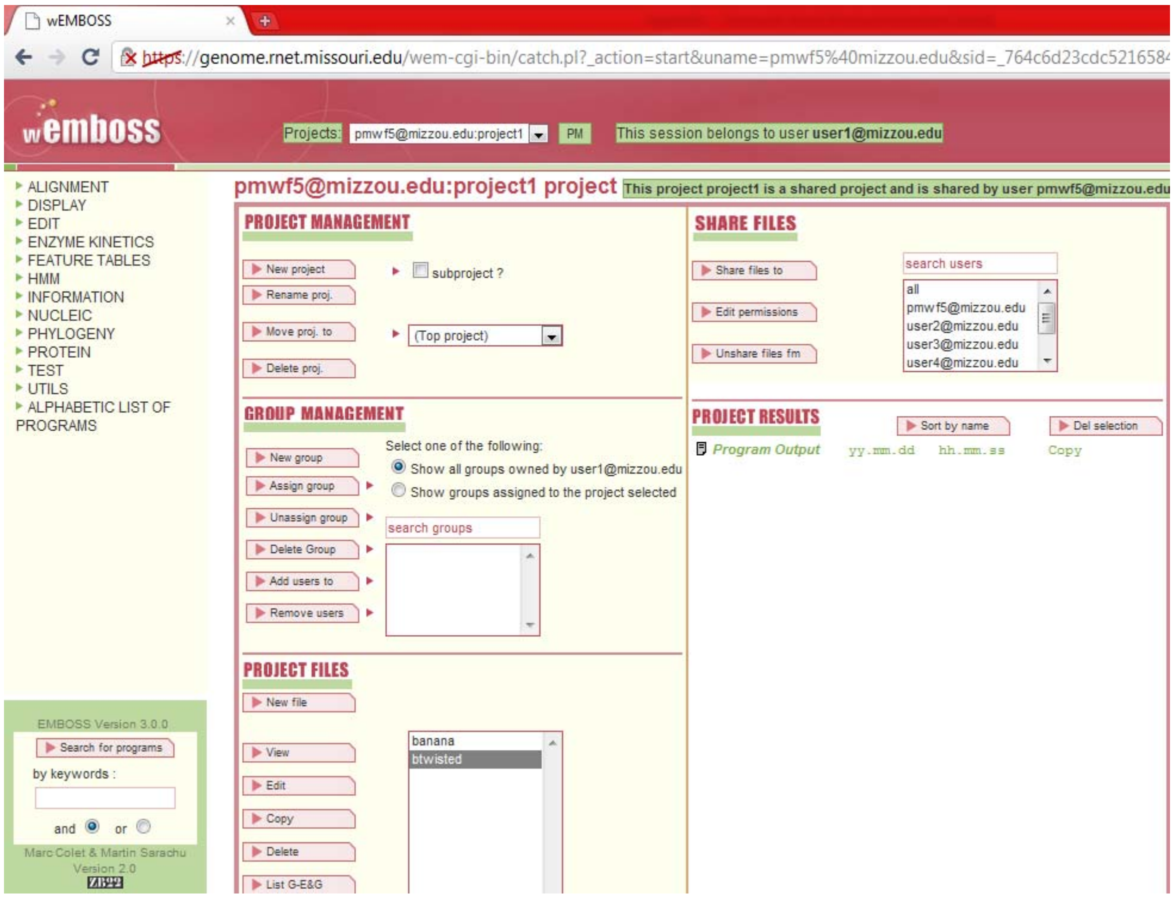
You are not authorized to perform this operation

Figure 6.15: Unauthorized Operation

Shared Projects

Users accessing a shared project can only contents of the projects that are shared to them. The format of the project, as it appears to the shared user, is OwnerName:ProjectName. Like the case of group projects, a message is displayed stating the project name and the user who shared it.

Figure 6.16 shows the home page when a shared project is selected.



~~https~~://genome.rnet.missouri.edu/wem-cgi-bin/catch.pl?_action=start&uname=pmwf5%40mizzou.edu&sid=_764c6d23cdc5216584

Projects: pmwf5@mizzou.edu:project1

This session belongs to user **user1@mizzou.edu**

pmwf5@mizzou.edu:project1 project

This project project1 is a shared project and is shared by user pmwf5@mizzou.edu

Move proj. to: (Top project)

Share files to: all, pmwf5@mizzou.edu, user2@mizzou.edu, user3@mizzou.edu, user4@mizzou.edu

Select one of the following:

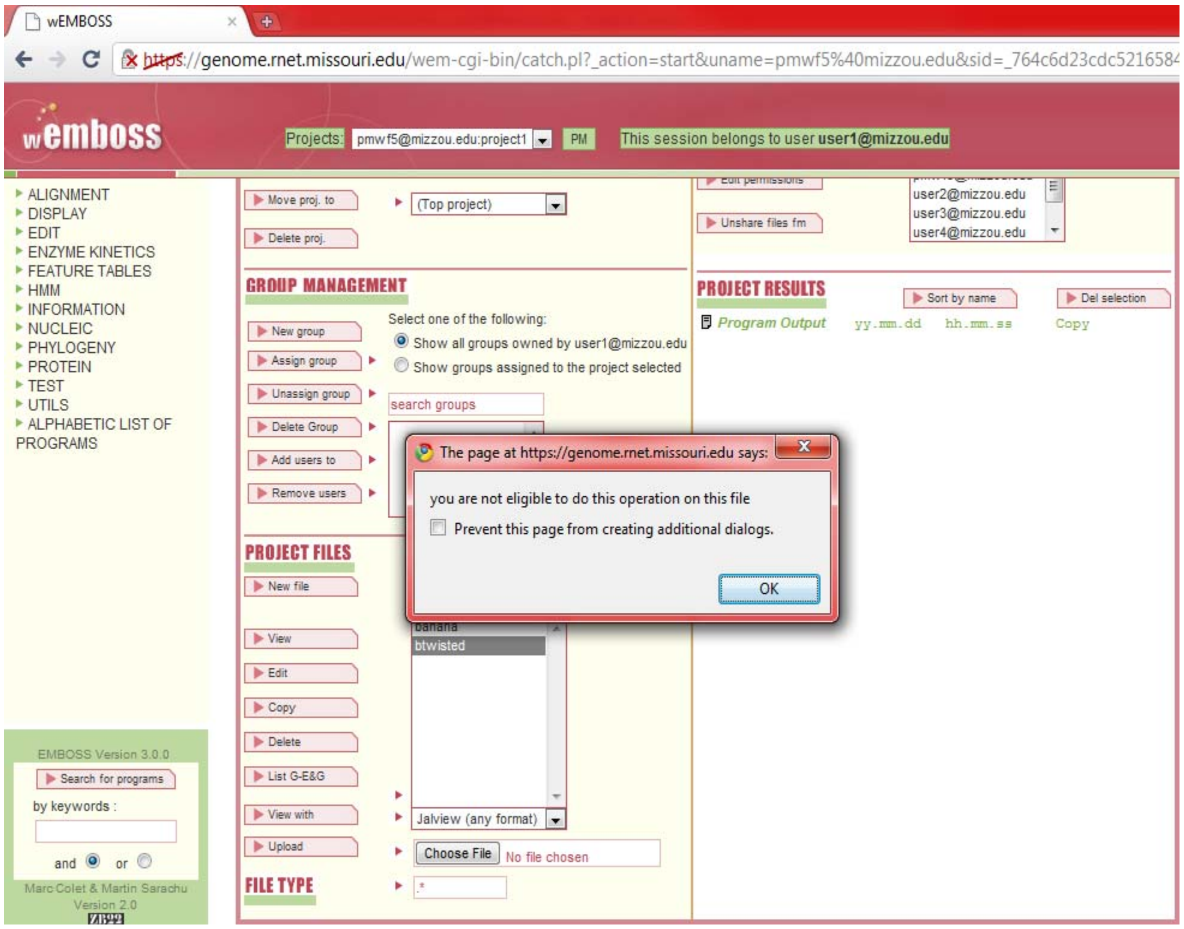
[✓ Show all groups owned by user1@mizzou.edu]

Show groups assigned to the project selected

Project files: banana, btwisted

Figure 6.16: wEMBOSS Home Page From a Shared User's Perspective

As mentioned earlier not all project contents are visible to the shared user. Only files shared to them are visible. As evident from the Figure 6.16, only two files are seen in the files list. Whereas the original project has more files. The user also has access restrictions on files. For example if the user does not have read permission on a file and tries to view it he is informed that he is not authorized to do so as shown in Figure 6.17.



https://genome.rnet.missouri.edu/wem-cgi-bin/catch.pl?_action=start&uname=pmwf5%40mizzou.edu&sid=_764c6d23cdc5216584

Projects: pmwf5@mizzou.edu:project1 PM This session belongs to user **user1@mizzou.edu**

Select one of the following:
● Show all groups owned by user1@mizzou.edu
○ Show groups assigned to the project selected

user2@mizzou.edu
user3@mizzou.edu
user4@mizzou.edu

The page at https://genome.rnet.missouri.edu says:

you are not eligible to do this operation on this file

☐ Prevent this page from creating additional dialogs.

EMBOSS Version 3.0.0
by keywords :
and ● or ○
Marc Colet & Martin Sarachu
Version 2.0

Figure 6.17 : Unauthorized Access

Also only files with execute permissions appear in the list of input files while running a program. Though two files are shared with the user in the above examples only one can be seen while executing programs.

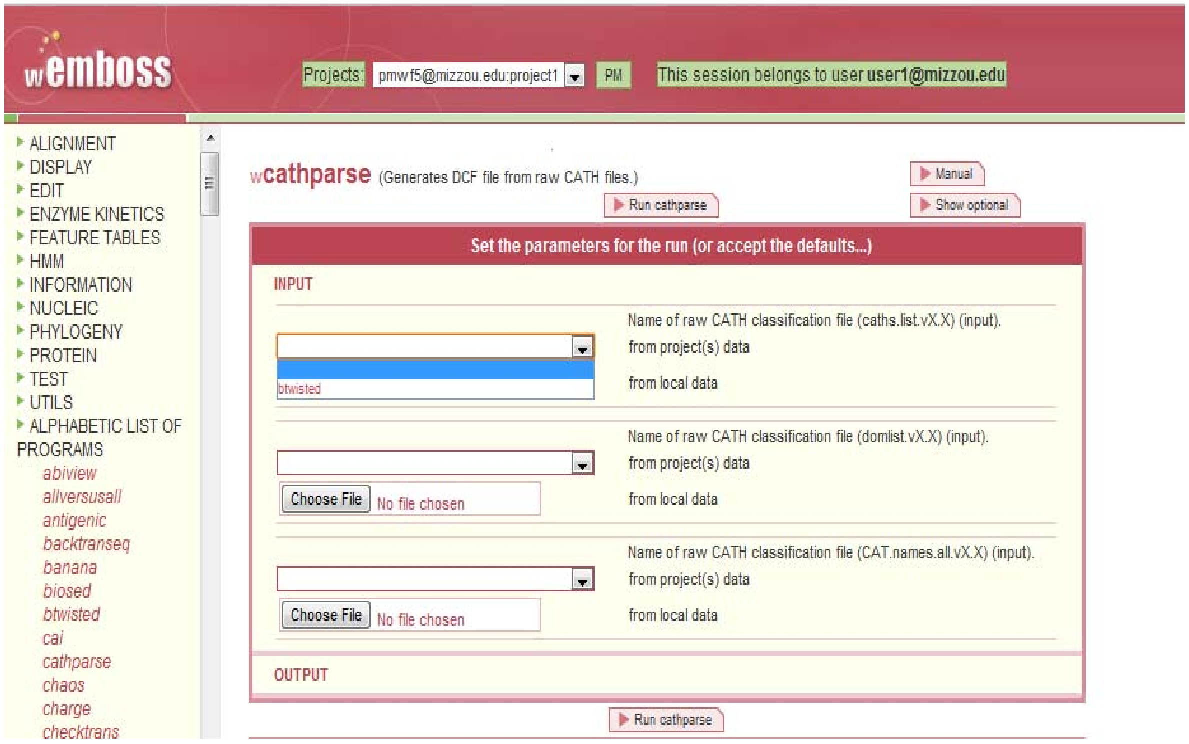

wemboss
Projects: pmwf5@mizzou.edu:project1
PM
This session belongs to user user1@mizzou.edu
ALIGNMENT
DISPLAY
EDIT
ENZYME KINETICS
FEATURE TABLES
HMM
INFORMATION
NUCLEIC
PHYLOGENY
PROTEIN
TEST
UTILS
ALPHABETIC LIST OF PROGRAMS
abiview
allversusall
antigenic
backtranseq
banana
biosed
btwisted
cai
cathparse
chaos
charge
checktrans
wcathparse (Generates DCF file from raw CATH files.)
Manual
Run cathparse
Show optional
Set the parameters for the run (or accept the defaults...)
INPUT
Name of raw CATH classification file (caths.list.vX.X) (input).
from project(s) data
btwisted
from local data
Name of raw CATH classification file (domlist.vX.X) (input).
from project(s) data
Choose File No file chosen
from local data
Name of raw CATH classification file (CAT.names.all.vX.X) (input).
from project(s) data
Choose File No file chosen
from local data
OUTPUT
Run cathparse

Figure 6.18: Files with Execute Permission

Chapter 7. Conclusion

This project report provides a detailed description of wEMBOSS[2], and the enhancements made to it. The report started with an introduction describing the need for a change in wEMBOSS and uses of a collaborative work environment. In the later chapters, the model and working of the enhanced system is discussed. wEMBOSS is a web interface for the EMBOSS**[1]** package where all the EMBOSS programs are accessible through interactive web pages in a user friendly way. Though wEMBOSS is very useful tool to users while working individually, it does not provide users the opportunity to work in collaboration. In order to provide wEMBOSS users the ability to work collectively towards better results, the concept of group management and file sharing has been introduced. The new version of wEMBOSS integrated with sharing capabilities has been discussed in this project report.

wEMBOSS is a very handy tool as it provides a web environment from which users can access EMBOSS in a user friendly way and also manage data efficiently. This application is especially useful to users who are not familiar with the UNIX environment, as EMBOSS can be executed only from the command line. There are of course other tools affiliated with EMBOSS that make interaction with EMBOSS easier. wEMBOSS being a very popular and widely used application is chosen for the current project. This project tries to enhance wEMBOSS by introducing a shared environment where users can share their projects with others and work in collaboration. The new features integrated into the old system work in a similar fashion to the already existing functions in order to make the users comfortable while using the system. Incorporating the Shibboleth authentication and authorization mechanism provides a secured environment to the new system. Technology plays a major role in determining the working of the system. wEMBOSS uses Perl because of its convenient functions for processing strings that are commonly encountered in biological data analysis. Enhancements made to wEMBOSS are also developed in Perl for the same reason mentioned above and also to make the integration smoother. The database management system used in wEMBOSS is GDBM. GDBM is easier to use and maintain when compared to the relational databases like Oracle, and SQL server. Data transactions from the database and data management programs are coded in the C programming language. The new system is highly portable. It works on any UNIX system with a working EMBOSS installation, a web server, a C compiler, Perl and Shibboleth.

Despite the enhancements made, there is still room for improvement in the system. In the current system, the administrator executes programs from UNIX command line to make changes to the data in the database. This can be transformed to a web based interface as not everyone is familiar with the command line interface. A log file can be introduced to keep track of activities of shared project users or group project users. This way the owner of a shared/group project can know when and what files a user used for executing wEMBOSS programs.

In conclusion this project is successful in providing wEMBOSS users with a better way of interacting with fellow collaborators by working in a collaboration environment. This project required a careful study of the existing wEMBOSS system and provided an opportunity to improve my knowledge of operating systems and programming skills required in enhancing the existing system. Completing this project gave me intense satisfaction and confidence from my experience in developing this application.

Glossary

1. **ACD:** Every EMBOSS program is associated with a so called Ajax Command Definitions (ACD) file. This file contains information about the input, output files and other parameters needed by the corresponding program. Also, any mandatory parameters, parameter limits or parameter dependencies are indicated by these files.
2. **Depth First Search (DFS):** Depth-first search (DFS) is an algorithm for traversing or searching a tree, tree structure, or graph. It is implemented in many applications like directory traversing, maze puzzles etc.
3. **EMBnet (European Molecular Biology network):** EMBnet is an international collaboration network that aims to enhance bioinformatics services by bringing together bioinformatics service providers.
4. **GPN:** Great Plains Network, commonly known as GPN is a consortium of universities in the Midwestern states, dedicated to supporting research and education through the use of advanced networking technology.
5. **Identity Provider (IDP):** Identity Provider is the software run by the home institution of a user that can authenticate users from that institution. It provides authentication services on behalf of the institution following the institution's authentication mechanism.
6. **Service Provider (SP):** Service Provider is the software run by the provider sharing a restricted resource. It makes authorization decisions based upon user required attributes/entitlements needed to access the resource.

7. **Single Sign on:** Single sign-on (SSO) is a mechanism whereby authentication of a user is required only once to access all systems or applications that the user has access permissions to.

8. **Shibboleth Indexical Reference Establisher (SHIRE):** SHIRE is a component of a Service Provider that determines if user needs to be authenticated, and, if so redirects the user to a WAYF service in order to identify the home institution and authenticate.

9. **Symbolic Link:** A symbolic link, also known as soft link, is a special kind of file that points to another file or directory. A symbolic link does not contain the data in the target file. It simply points to another entry somewhere in the file system. Programs that read or write to files named by a symbolic link behave as if operating directly on the target file.

10. **Virtual Organization:** It is a set of organizations defined around a set of rules and conditions. Though these organizations are physically separate they function as one unit through the use of common practices and resources for the purpose of one or more identified goals.

11. **WAYF:** The Where Are You From service is used by Shibboleth software to determine what a user's home organization is. This Service lets the user choose his home organization from a list and then redirects the user to this home organization's login page for authentication.

References

[1] Rice,P. Longden,I. and Bleasby,A. "EMBOSS: The European Molecular Biology Open Software Suite" Trends in Genetics June 2000 Vol 16, No 6. Pp.276-277

[2] Sarachu M and Colet M. 2005. “wEMBOSS: a web interface for EMBOSS”. Bioinformatics Vol 21, pp.540-541.

[3] Swiss node of EMBnet. EMBOSS. *Swiss Institute of Bioinformatics* http://www.ch.embnet.org/EMBOSS/index.html (accessed November, 2010)

[4] The Client Server Architecture. *Web Developers Notes.* http://www.webdevelopersnotes.com/basics/client_server_architecture.php3 (Accessed November, 2010)

[5] Internet2. About Shibboleth. (n.d). Retrieved from http://shibboleth.internet2.edu/about.html (Accessed December, 2010).

[6] Ciordas, I. O. FINE-GRAINED AUTHORIZATION IN THE GREAT PLAINS NETWORK VIRTUAL ORGANIZATION. Computer Science, Master’s Thesis, University of Missouri, Columbia. 2007. Retrieved from https://mospace.umsystem.edu/xmlui/handle/10355/4967

[**7]** Anjali, Gupta. 2008. Web-Based Data Repository in the Great Plains Network Virtual Organization. Computer Science, Master’s Project, University of Missouri, Columbia. 2008.