Functional Specifications for Team TerraUser

The Web-based User Management Project

Michelle Harr
Naoko Tsunekawa
Daniel Wallace

19 February 2002

LIST OF TABLES AND FIGURES

1. Introduction

This document is an official functional specification of the TerraUser Web-based User Management software. This project is part of Northern Arizona University (NAU) College of Engineering and Technology's (CET) Senior Capstone Design 2001-2002. Sponsorship is provided by Deborah Lee Soltesz from the U.S. Geological Survey and advisement provided by Dr. Eck Doerry, Professor of Computer Science at NAU.

This document describes the behavior of the product and contains the technical information and data needed for the design. It translates the software requirements document into a technical description, which ensures the product feature requirements are correctly understood before moving into the design process.

2. Software overview

2.1 Background / History

Congress created the U.S. Geological Survey (USGS) in 1879 as a science agency for the Department of the Interior. The USGS serves as a national science provider and fact-finding agency that provides a scientific understanding about natural resource conditions, issues, and problems. USGS scientists collect, monitor and analyze large amounts of data about the Earth and solar system. The government and citizens in all walks of life use the information the USGS produces for various reasons, including addressing pressing social issues. The USGS uses the vast scientific information for a wide range of products, such as maps, and scientific solutions, such as wetlands restoration and hazardous waste disposal.

The USGS Terrestrial Remote Sensing Team at the Flagstaff Field Center consists of a four-member group: Pat Chavez *(Remote Sensing Scientist and Group Leader),* Stuart Sides *(Computer Scientist),* Deborah Lee Soltesz *(Web Mistress), and* Miguel Velasco *(Image Processing Specialist)*. They work with satellite, multispectral, airborne, shipborne sidescan sonar, and DEM digital images. This team does such things as digital mosaicing, extraction and mapping of earth science information, geometric and radiometric calibration and corrections, and multitemporal change detection. The team has set up TerraWeb as a way for people to access this information, along with a way to organize and manage some of their data.

Currently USGS TerraWeb applications have minimal security. Users are not required to log on to access these web applications. No current user management system is in place. Since data management and data analysis/manipulation is the main function of many of these applications, it is imperative that there be some sort of security standards when using these systems. These TerraWeb applications are fairly new, therefore application uses and functions are evolving for the group's needs.

The objective of the project that we have been given by USGS is to design and implement an efficient and secure interface to other USGS TerraWeb applications, along with a stand-alone application used to administer the user management system. The software will allow users to securely and easily access other interactive TerraWeb applications.

2.2 Product Description

The goal of this product is to provide an efficient, secure interface to other USGS TerraWeb applications, along with a stand-alone application used to administer the user management system. The system will be built on a SuSE Linux server running an Apache web server.

There are two parts to the TerraUser interactive web application. First, the interactive web application allows users to securely and easily access other interactive USGS TerraWeb applications. A variety of user information is stored in a MySQL database: user name, team name, personal preferences, priority level, level of access, etc. This allows users to customize their interface with preferences. Also, users are allowed to retrieve and update certain information from the database through the interface.

Second, administrators are able to manage user accounts and permissions through the stand-alone management system. The product centralizes the user management system and provides to administrators a way to manage the user's different access levels. Users information can be easily manage through administration interface.

Figure 2.1 below shows a rough overview of the TerraUser application. Users will use a secure socket layer (SSL) to login to the TerraUser application. The TerraUser application will communicate to a MySQL database using JDBC. The TerraUser application will be able to send requested information to the TerraWeb applications.

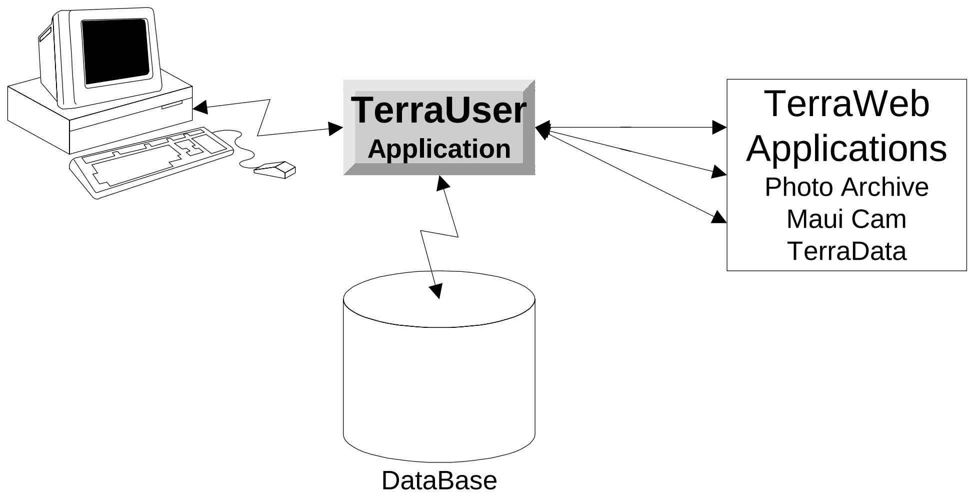


Figure 2.1: High-level Application Overview

2.3 Product Functions

The overall functionality must:
Provide a secure interface for users to login to TerraWeb applications

- Provide a centralized way to manage users and their access/priority level
- Use a web-based interface
- Allow the user to:
    - Supply a user name and password to be granted access to applications
    - Supply personal preferences
- Allow the administrator to specify users, access levels, priority levels, and available applications
- Store all the data in its database

These functional requirements are elaborated and discussed in more detail in Section 3.

2.4 Users

There are three kinds of users:

- *Administrators:* Users who use the administration application to edit user information in the database. They enter data such as who the user is, what privileges the user has and the priority level of the user.
- *Editors:* Users belonging to a specific work group or multiple work groups who have access to all information belonging to their group. These users also have read access to information that is marked public by other groups.
- *Guests:* Users who are allowed very limited access and information to applications. These users can only search and read information that is marked as public.

2.5 General Constraints

Listed below are the constraints that have been proposed by the client as well as those which reflect project domain specifications.

- The system will have to secure user information sent through the Internet. This will be achieved by using the secure HTTPS protocol.
- The system must adhere to accessibility and government guidelines (System must not use cookies, etc).
- The system must not require specific browser to be run.
- The system development/integration/testing must be completed by April 26, 2002, for the Capstone Project Conference.

2.6 Technical Constraints

The TerraUser software must meet the following minimal requirements:

- The system will be designed to be scalable to meet future needs of the client.
- The implementation must utilize specific technologies provided on the server.

The following table is a brief summary of the technology required by the project and available on the server:

Category	Technology Used
Operating System	SuSE Linux
Web Server	Apache
Java Server	Apache Tomcat
Server Side Interfacing	Java, JDBC, JSP, JavaScript
Database	MySQL
Security	SSL

Table 2.1: Technical Requirements

The design must provide a completely web-based interface. All interfaces must meet with HTML 4.0 minimum standards and be in compliance with the Rehabilitation Act of 1973, Amendments of 1998, section 508.

2.7 Assumptions and Dependencies

Listed below are assumptions that have been considered:

- All TerraWeb applications will use the same technologies to provide easy interfacing.
- A server exists.
- Users have access to a standard browser.
- Direct access to the server is available.
- The server has direct access to the TerraWeb server (i.e. not going through firewalls).

3.1 High-Level Architecture

Figure 3.1 below shows a rough diagram of the TerraUser interface. Users will have to log in through a browser to gain access to TerraWeb applications, a user preference page, etc.

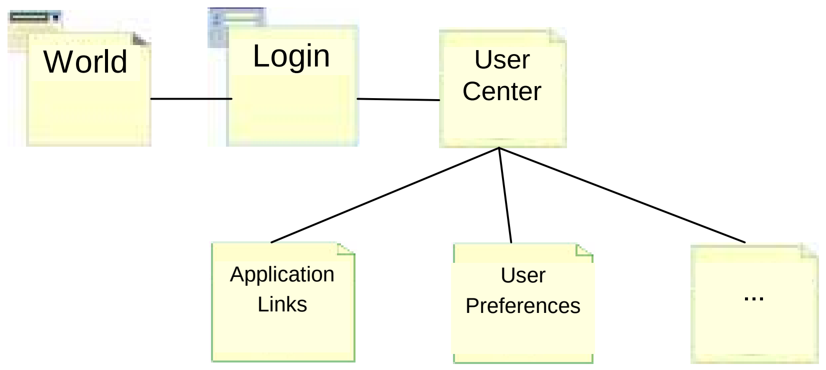


Figure 3.1: High-level Interface Overview

The architecture of the software will be fairly simple. Modules will be created at each level of technology: WWW (HTML, JavaScript), server (JSP, JavaServlets), and database (MySQL). The modules will act as an internal interface between each technology. The web-page modules use the database modules as an interface between the web pages and the database. This will be an effective and efficient way to provide easy interaction between the web-interface and the database.

A three-tier architecture can provide flexibility, reusability, and scalability when used for a distributed client/server design. It is a common architecture used for many Internet applications. An architecture overview is shown below in Figure 3.2.

Using a three-tier architecture will allow the information transfer between the web server and the database server to be optimized. The architecture will allow the interface to be easily scalable. The architecture also provides a framework for sub-system control and communication.

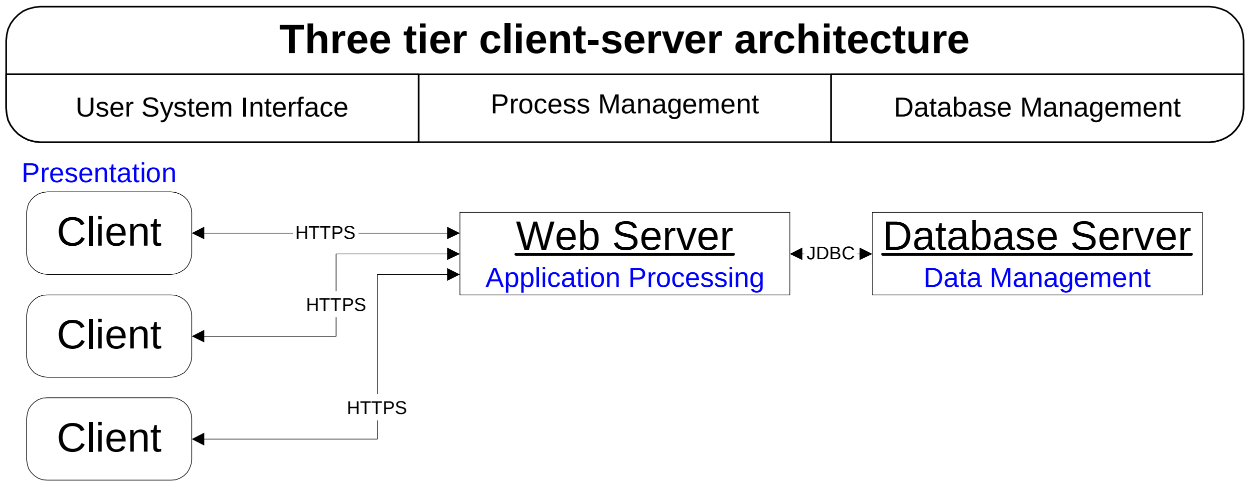


Figure 3.2: Architecture

3.2 Users

Administrators

Administrators manage users, teams and their authority to access information. The following table is a list of the identified requirements for the administrators.

Requirements prioritized with 1 = Crucial, 2 = Very high, 3 = High, 4 = Average, 5 = Low, 6 = Very low. Difficulty is rated from 1 to 10 where 10 = very difficult and 1 = very easy.

Requirement	Priority	Difficulty
Add a new user	1	2
Delete existing user	1	2
Update user information	1	2
Add/Update/Delete team information	1	2
Log off all users	3	8
Post Message of the Day (MOTD)	5	1
View active users or logs	2	2
Add information fields	1	4
Set password reset/expire	1	5

Table 3.1: Administrator Requirements

Editors

Editors are the main users of the interface and applications. These users perform the actual work. The following table is a list of the identified requirements for the editor.

Requirements prioritized with 1 = Crucial, 2 = Very high, 3 = High, 4 = Average, 5 = Low, 6 = Very low. Difficulty is rated from 1 to 10 where 10 = very difficult and 1 = very easy.

Requirement	Priority	Difficulty
Change password	1	2
Change preference	1	4
Access to applications	1	8
Search option	5	8
E-mail to administrator	2	1

Table 3.2: Editor Requirements

Guests

Guests have limited access to applications and information. The following table is a list of the identified requirements for the guest.

Requirements prioritized with 1 = Crucial, 2 = Very high, 3 = High, 4 = Average, 5 = Low, 6 = Very low. Difficulty is rated from 1 to 10 where 10 = very difficult and 1 = very easy.

Requirement	Priority	Difficulty
View application data that is marked public	1	8
Search for public information	1	5
Post messages to administrators or teams	5	1

Table 3.3: Guest Requirements

3.3 Functionality Overview

The following specific functions must be provided:

1. **User Accounts:**
   - provides storage of user information including but not limited to: user login name, password, priority level, access rights, group membership, and interface preferences
   - managed through an administrator interface

2. **Centralized User Login:**
   - requires use of user login name and password to access the software
   - provides users the ability to change the password at any time
   - uses encryption to send data
   - has expiration times for passwords set by administrators forcing users to change their passwords after the expiration time

3. **Interactive Web Application for Administrators:**
    - provides an independent application available exclusively to administrators
    - provides the ability to add or delete information that can be stored for users
    - provides the ability to alter information stored for any given user
    - provides the ability to add or delete users
    - provides options to set the password expiration time
    - provides monitoring of user activities and the option to enable outputting of activities to a log file

4. **Interactive Web Application for All Users:**
    - uses secure login
    - provides access to the Maui Cam, TerraData, and Photo Archive applications
    - provides user interface customization stored with their user data
    - allows applications to retrieve the user's customization
    - allows users to view and manage their system accounts in one place
    - allows users to use TerraWeb applications they have been granted access to
    - allows users the option to change their password

3.4 User Interfaces

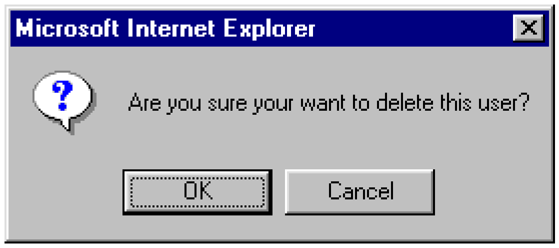
The interface to the software is entirely web-based; all input will come from forms embedded in the web pages. This means that:

- All input data will be in a text format.
- The data will be sent to our application as text.
- The data will be transformed to various data types.

The interface will use four basic types of inputs: text fields, drop-down selection menus, buttons, and links. The following constraints will apply:

- The login and password text fields will be checked for illegal characters.
- The login field will be limited to alphanumeric characters, the dash "-" and the underscore "_".
- The other text fields will allow the user to enter text freely and will be processed for compatibility with the database while preserving the text. The processing will primarily involve using doubled double quotes and encoding returns and backslashes.
- Any buttons or links that appear on any given page will have an obvious functional effect and may bring up a dialog box asking a yes or no question.
- Many of the inputs available to the user will be a drop-down selection menu.
- Inputs will have predefined values associated with them and thus require no processing before being used.

The figures below show the different types of inputs available to the user.



Are you sure your want to delete this user?

Figure 3.3: Dialog Box

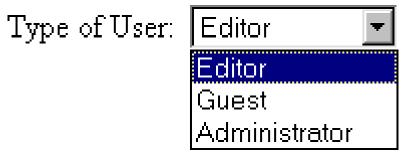
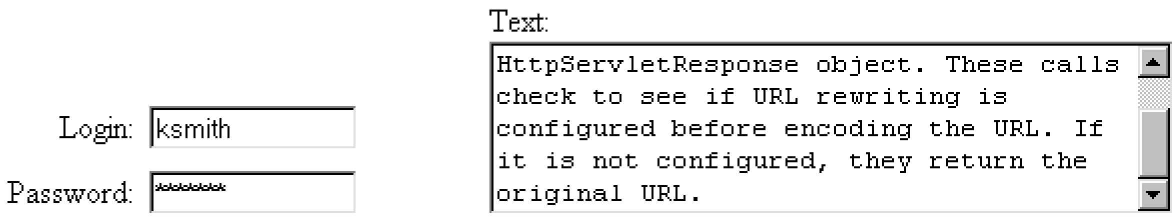


Figure 3.4: Drop-down Selection Menu



Login: ksmith

Password: ********

```
HttpServletResponse object. These calls
check to see if URL rewriting is
configured before encoding the URL. If
it is not configured, they return the
original URL.
```

Figure 3.5: Various Text Fields

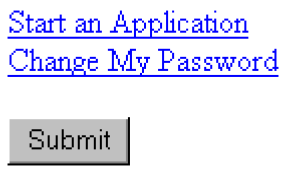


Figure 3.6: Two Links and a Button

Outputs will be the TerraData applications (after login), confirmation of any changes users/administrators make, search results, or various error messages when inputs are not valid.

3.5 Processing

User Access – Editor/Guest

All the users must login through the TerraUser interface and become verified before any process is selected. The editor/guest has the following selections:

- change password (editor only)
- start TerraData applications (editor only)
- add/modify user's preference
- search option
- send e-mails

Process flows and data flows are shown below in Figures 3.7 and 3.8.

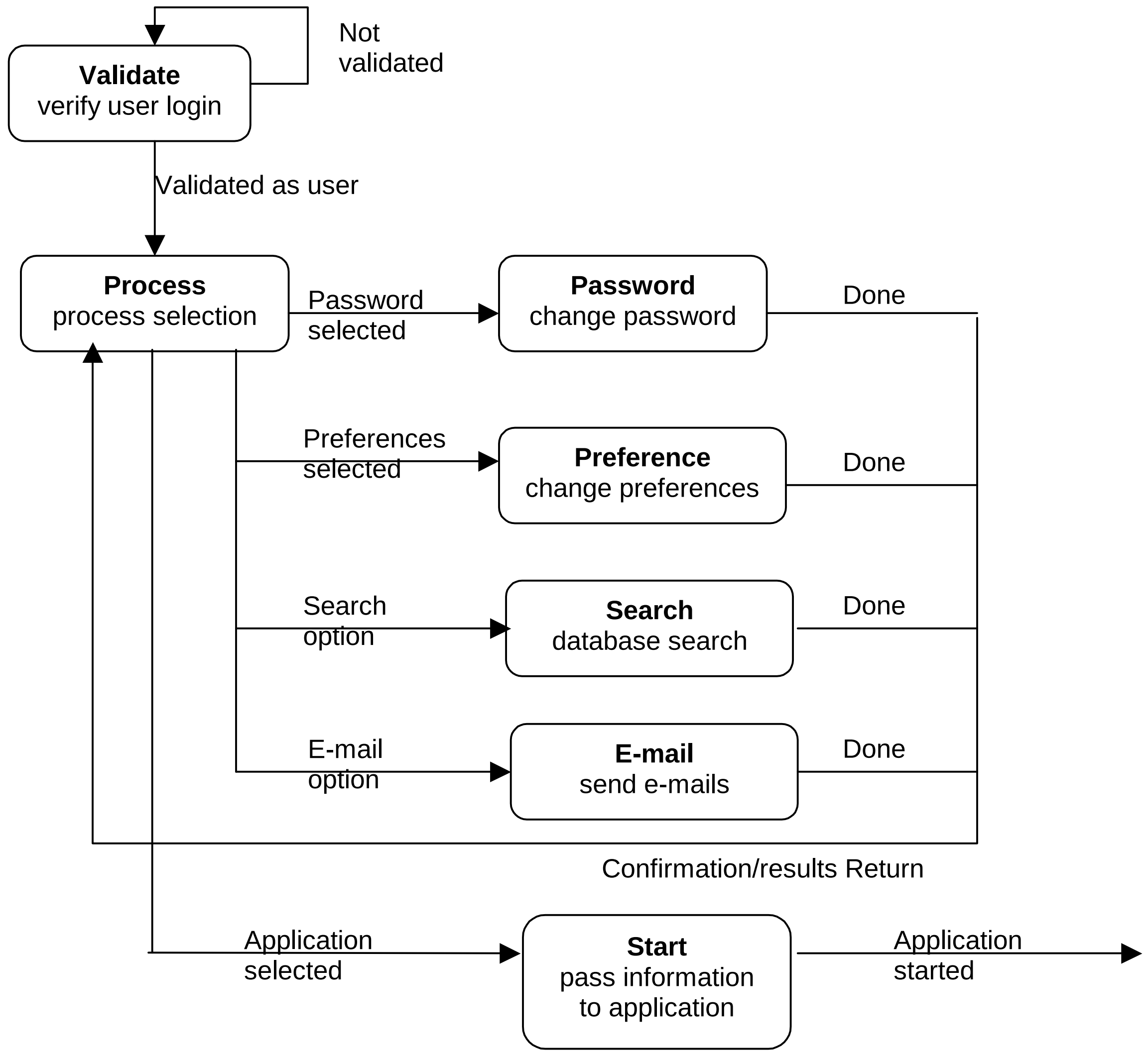


Figure 3.7: State Diagram User Access

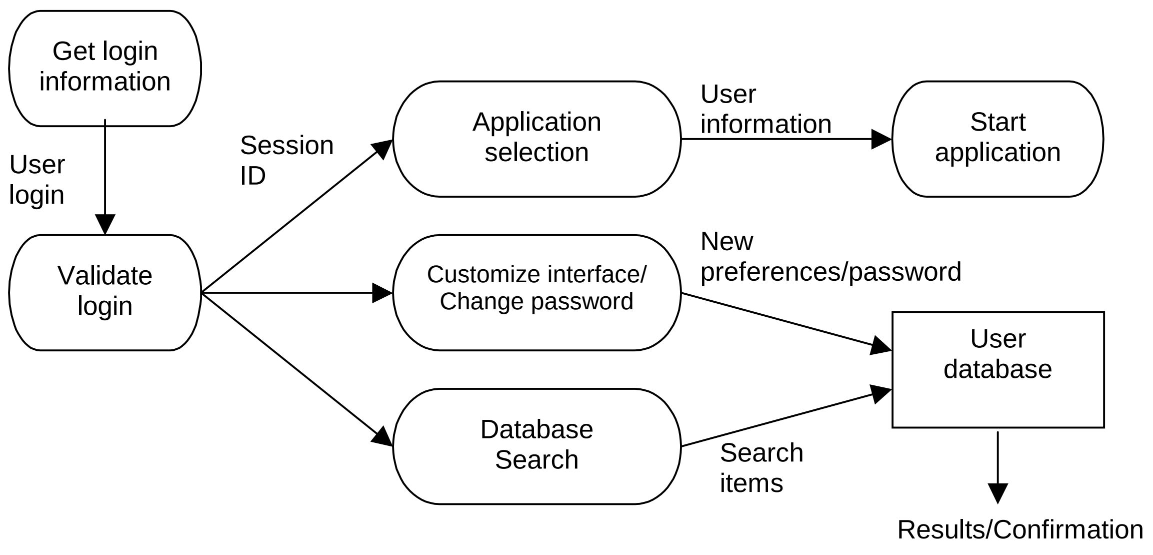


Figure 3.8: User Access Data Flow Diagram

User Access – Administrator

All administrators must login through the TerraUser interface and be verified before any process is selected. The administrator has the following selections:

- update user information
- add/delete users
- reset/expire user's password
- add/delete/update teams
- view user log files
- add/delete/modify user database fields
- log user off
- post message of the day (MOTD)

Process flows and data flows are shown in Figure 3.9 and 3.10.

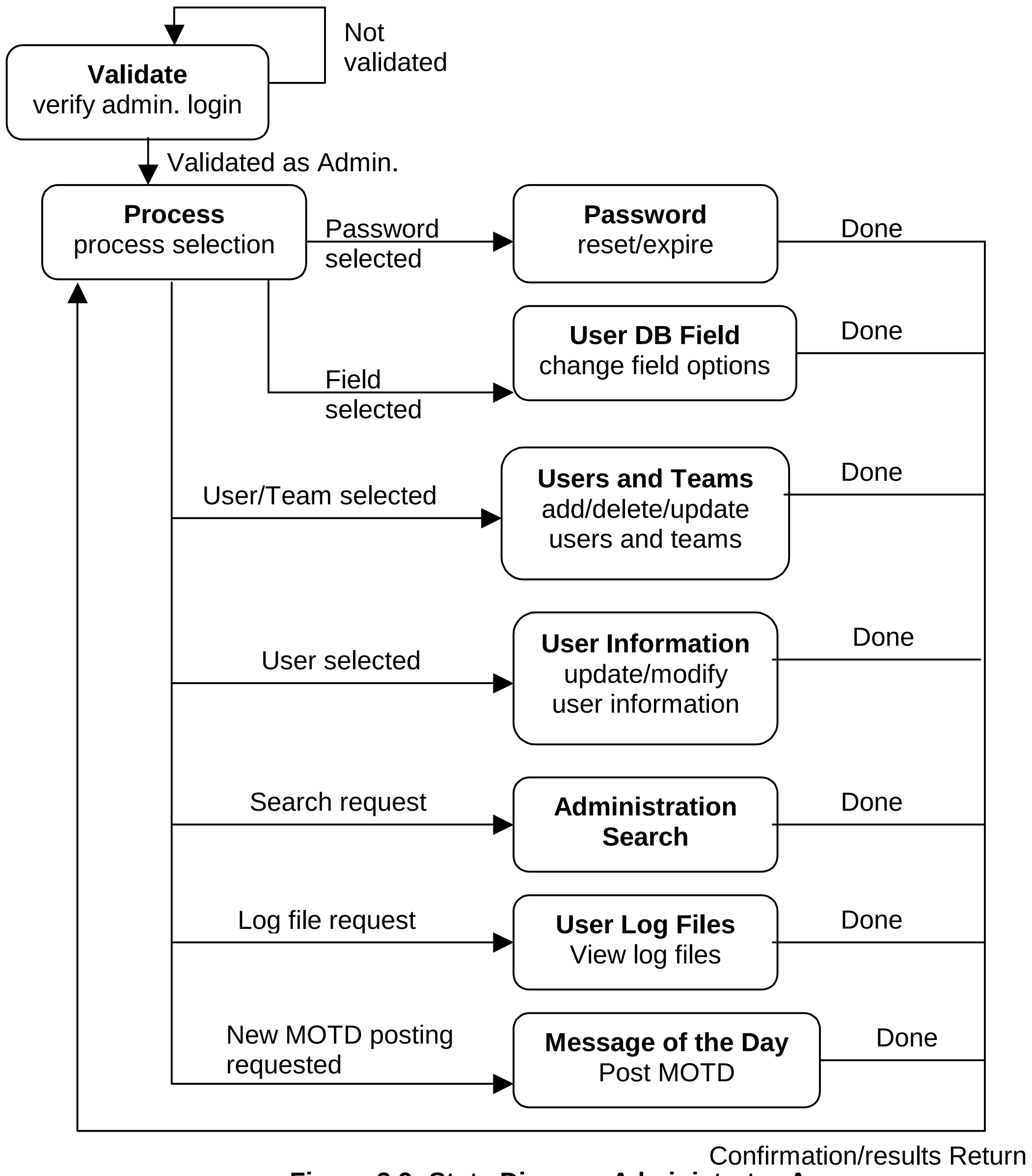


Figure 3.9: State Diagram Administrator Access

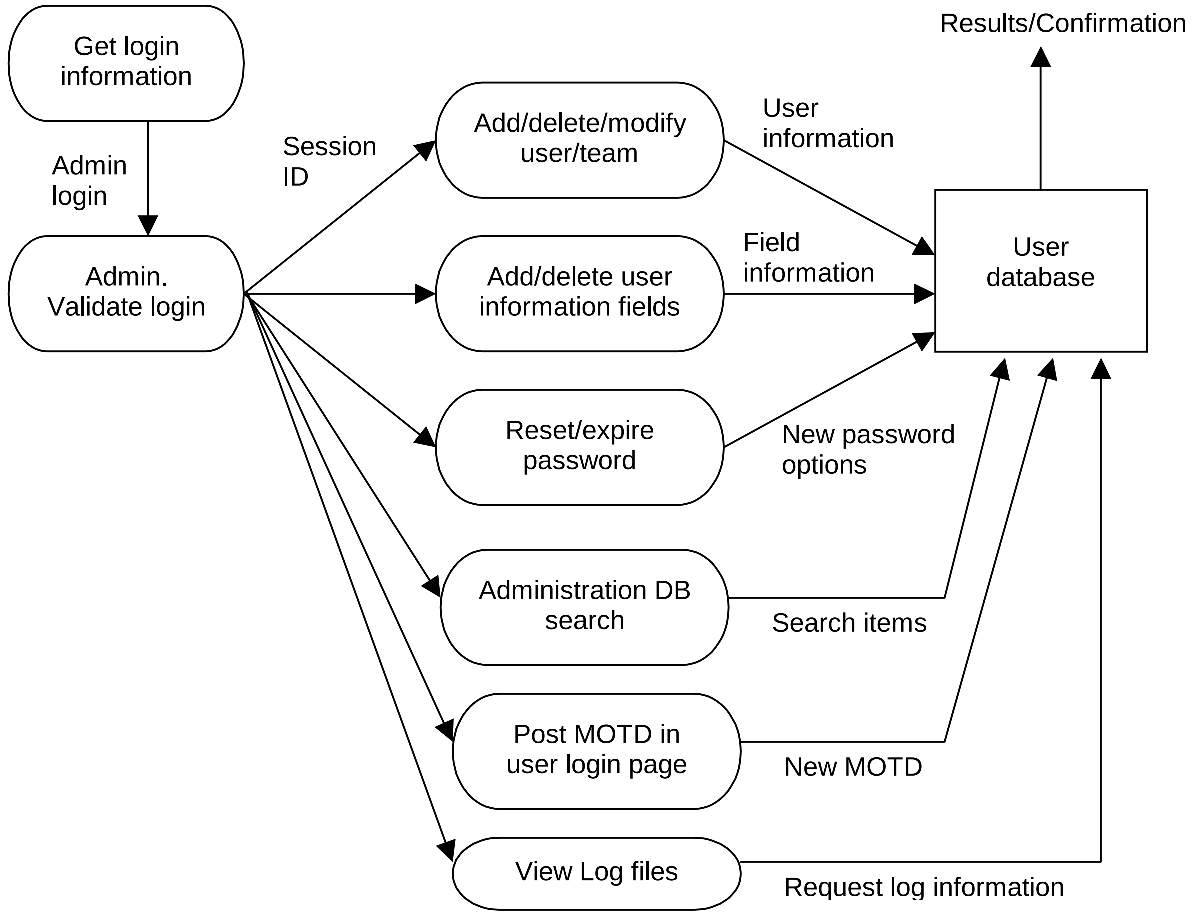


Figure 3.10: Administrator Access Data Flow Diagram

4.1 How the Product will be used

Figure 4.1 shows a conceptual representation of the web pages in the TerraUser application. The Guest is basically the same as an editor, but with a limited functionality. The Administrator interface allows administrators to manage and control user information.

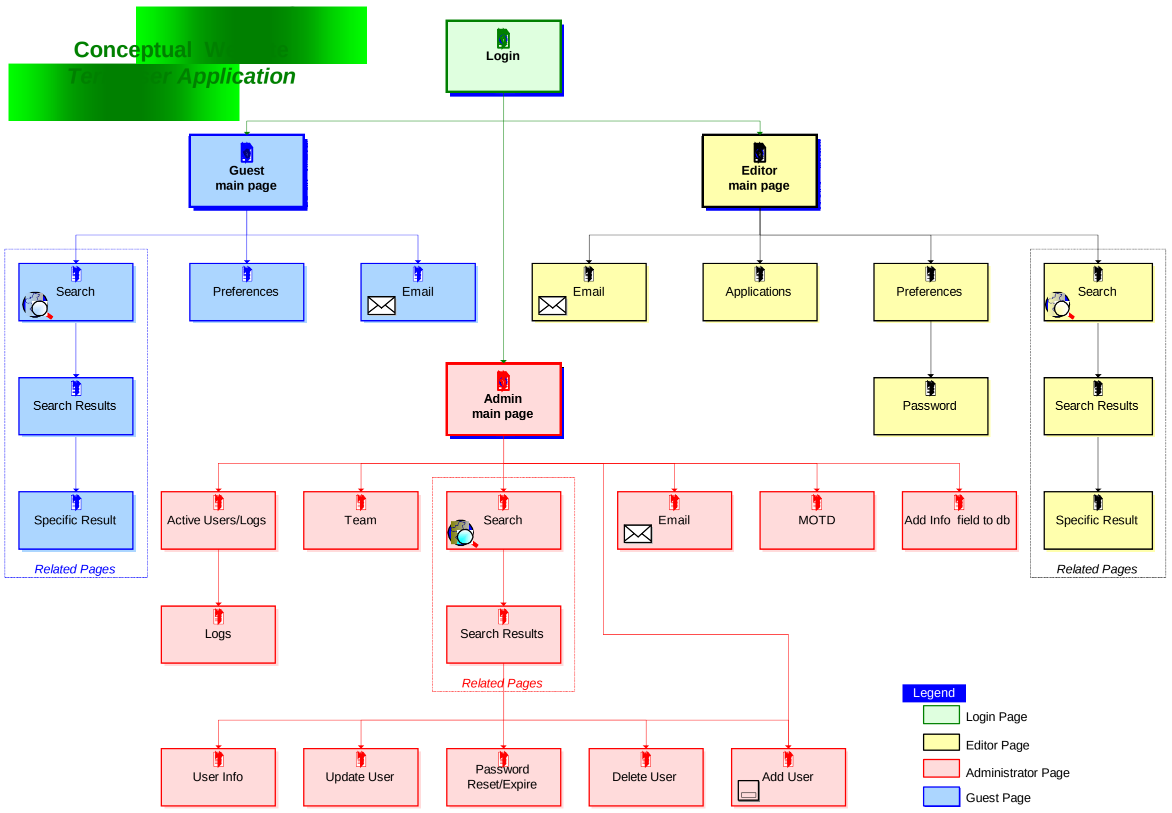


Figure 4.1: Conceptual Website

4.2 Use Cases

By identifying the project's use cases we are able to abstract, technology free, dialogues of user interaction and system responsibilities. These use cases will convey a series of actions that a user must initiate as they use the system to resolve problems. By describing the use cases we hope express the specific functions of the application in a way that pulls all the ideas together.

Figure 4.2 shows the all of possible use cases of TerraUser application. Users login to the system as guests, editors, or administrators. Depends on the users level of access, variety of things can be done through the interface. Specific use cases are shown in Table 4.1, followed by details of each use case.

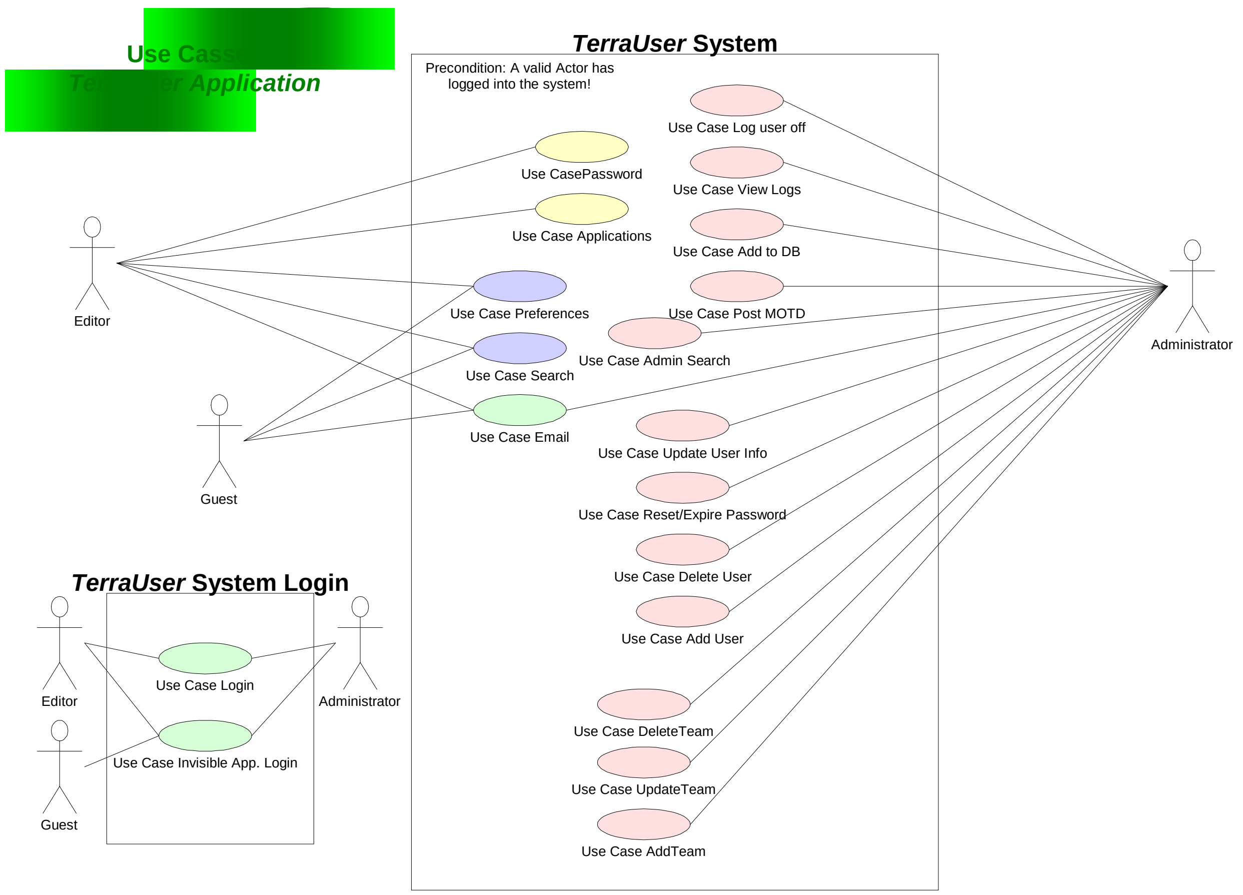


Figure 4.2: Use Cases

Index	Use Case	Users
1	Invisible Application Login	Editor/Guest
2	Change Password	Editor
3	Access to Applications	Editor/Guest
4	TerraUser Login	Administrator/Editor/Guest
5	Change Preference	Editor/Guest
6	Search	Editor/Guest
7	E-mail	Administrator/Editor/Guest
8	Add User	Administrator
9	Delete User	Administrator
10	Update User Information	Administrator
11	Password Reset/Expire	Administrator
12	Add Team	Administrator
13	Update Team	Administrator
14	Delete Team	Administrator
15	Log off Users	Administrator
16	Post MOTD (Message of the Day)	Administrator
17	View Active Users and Logs	Administrator
18	Administrator Search	Administrator
19	Add Information Fields	Administrator

Table 4.1: Table of Use Cases

Use Case 1: Invisible Application Login

The following use case is used to define optimal paths for an Editor, or a Guest to get logged in to a specific application directly.

Use Case	***Invisible Application Login: Guest, Editor***
Description	This use case describes the interactions that take place when an Editor, or a Guest wants to get logged in directly to an application, in order to gain access to the system.
Scenario	Eddie (an Editor) is a scientist working on some DEM (Digital Elevation Model) data. He would like to use the TerraData application to access the metadata from the images that he had previously pulled from the system. Eddie types in the URL for the TerraData application. He must then enter his username and password in the provided area and hit the 'Login' button. The system will authenticate Eddie and put him directly into the TerraData application.
Actor(s)	Editor or Guest
Assumptions	We assume that Eddie has an active account.
Steps	1. Eddie enters top-level URL for application. 2. Eddie enters username and password. 3. Eddie presses the 'Login' button. 4. System will verify the information. 5. If any information is incorrect, the system will display correct error message and prompt Eddie to correct information. 6. Eddie should now be logged in to application.
Non-Functional	*Performance:* Should take less than 30 seconds to process after hitting 'Log in' *Reliability*: Users shouldn't be able to gain access to application without logging in. Once system is up and running Logins should be available when system is operational. *Frequency:* Must login each time want to access application, user logged out after thirty minuets of inactivity. *Fault Tolerance:* *Priority:*
Issues	

Table 4.2: Use Case Invisible Application Login

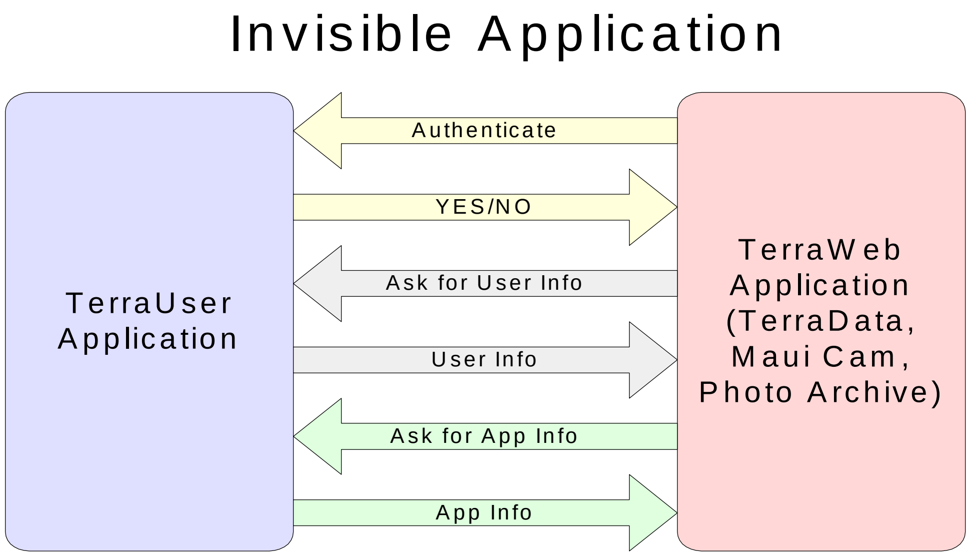


Figure 4.2: Invisible Application Communication

Figure 4.2 above shows the communications between the TerraUser application and a TerraWeb application. The purpose of the invisible login is to have a generic login page for a TerraWeb application, so that when the user logs in they go directly into the TerraWeb application, instead of having to go through the TerraUser interface. The TerraWeb application can also ask for other types of information form the TerraUser system, such as information about the user, or some piece of application information, like the users preferences.

Use Case 2: TerraUser Login

The following use case is used to define optimal paths for an Administrator, an Editor, or a Guest to get logged in and gain access to the system.

Use Case	***TerraUser Login: Administrator, Guest, Editor***
Description	This use case describes the interactions that take place when an Administrator, an Editor, or a Guest wants to get logged in, in order to gain access to the system.
Scenario	Abigail is an Administrator. She wants to get logged into the TerraUser system so that she can add some users. Abigail types in the top-level URL for the TerraUser application. She must then enter her username and password in the provided area, select 'Administrator' as the user type and hit the 'login' button. The system will authenticate Abigail and put her directly into the TerraUser application.
Actor(s)	Administrator, Editor, or Guest
Assumptions	We assume that Abigail has an active account.
Steps	1. Abigail enters URL for login page in browser. 2. Abigail enters username and password. 3. Abigail selects 'appropriate user type 'Administrator' from the drop down menu labeled 'Type of User:' 4. Abigail presses the 'Login' button. 5. System will verify the information. 6. If any information is incorrect, the system will display correct error message and prompt Abigail to correct information. 7. If Abigail needs help they can click on the help link. 8. Abigail should now be logged in to appropriate interface.
Non-Functional	*Performance:* Should take less than 30 seconds to process after hitting 'Log in' *Reliability*: Users shouldn't be able to gain access to application without logging in. Once system is up and running Logins should be available when system is operational. *Frequency:* Must login each time want to access application, user logged out after thirty minuets of inactivity. *Fault Tolerance:* *Priority:*
Issues	

Table 4.3: Use Case TerraUser Login

<u>Use Case 3: Changing Password</u>

The following use case is used to define optimal paths for an Editor to be able to change their password.

Use Case	***Change Password: Editor***
Description	This use case describes the interactions that take place when an Editor wants to be able to change his/her password on the system.
Scenario	Eddie is an Editor. He wants to be able to use the Maui cam application. Eddie asks Abigail for access to this system. Abigail, the Administrator adds Eddie as a user to the system, gives him the URL for the TerraUser application, and asks him to login and change his password. Eddie Loges into the system, navigates to the preferences page. Eddie hits the 'Password' button, and changes his password.
Actor(s)	Editor
Assumptions	Precondition: Eddie has logged into the system.
Steps	1. Eddie clicks on the 'Preferences' page link. 2. Eddie clicks on the 'Password' button. 3. Eddie enters current password, and new password twice. 4. Eddie enters the 'update' button. 5. System verifies information, and displays appropriate message. 6. If there is an error in entering the data, the Eddie is prompted to reenter the data.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.4: Use Case Change Password

Use Case 4: Access to Applications

The following use case is used to define optimal paths for an Editor to be able to access certain web-based applications that have been set by the Administrator. An administrator can give a whole group access to an application or individual users or a combination of both.

Use Case	***Access to Applications: Editor, Guest***
Description	This use case describes the interactions that take place when an Editor wants to access other web-based applications that belong to the system.
Scenario	Eddie a scientist has editor privileges on the TerraData application. Eddie wants to gain access to the TerraData application through the TerraUser interface.
Actor(s)	Editor or Guest
Assumptions	Precondition: a valid Eddie has logged into the system.
Steps	1. Eddie clicks on the 'Application' link on main page 2. System takes editor to page with list of the applications they have access to. 3. Eddie clicks on application they want to use.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	Implementation issues still awaiting resolution.

Table 4.5: Use Case Access to Applications

Use Case 5: Changing Preference

The following use case is used to define optimal paths for an Editor, or a Guest to be able to change their user preferences on the system (i.e. Change the look and feel of the application).

Use Case	***Change Preferences: Editor, Guest***
Description	This use case describes the interactions that take place when an Editor wants to be able to change his/her preferences on the system.
Scenario	Gus the guest is checking out the system to see how the interface works. He has a hard time reading some of the text on the page so he wants to changes his preferences to make the text bigger.
Actor(s)	Editor or Guest
Assumptions	Precondition: Gus has logged into the system. *Guest preferences will be reset to default upon logout. *Settings are saved for the Editor.
Steps	1. Gus clicks on the 'Preferences' link on the main page. 2. Gus Clicks on dropdown menus to change such things as color, font size, etc… 3. Gus clicks the update button at the bottom of the page. 4. Gus also has the choice to hit the 'default' button to restore default settings.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.6 Use Case Change Preferences

Use Case 6: Search

The following use case is used to define optimal paths for an Editor, or a Guest to be able to perform a search on the system that will return matching results that they have ownership of, permission to access, or records that are marked public.

Use Case	***Search: Editor, Guest***
Description	This use case describes the interactions that take place when an Editor wants to search the system, the system will only return matching results that are owned by a group they belong to, or are marked public.
Scenario	Gus the guest is really excited to learn all he can about the geology of the Grand Canyon. He wants to do a search and see if he can find any information, so he runs a search.
Actor(s)	Editor or Guest
Assumptions	Precondition: Gus has logged into the system.
Steps	1. Gus clicks on the ‘Search’ link on the main page. 2. Gus types in search criteria, such as a key word. 3. System verifies the information. 4. System performs the search and returns results that match and are marked public in the database. 5. (For Editor search will also return records that they own or belong to their group). 6. User may click on the links to view data in greater detail.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.7: Use Case Search

<u>Use Case 7: E-mail</u>

The following use case is used to define optimal paths for an Editor to be able to send an email to the Administrators of the system, to members of a group they belong to, or one or more individuals that are members of the same group that the Editor belongs to.

Use Case	***Email: Administrator, Guest, Editor***
Description	This use case describes the interactions that take place when an Administrator, an Editor, or a Guest wants to send an email through the system.
Scenario	Abigail the administrator would like to send out an email to the members of the DataCruncher group, informing them that they all have access to a new application.
Actor(s)	Administrator, Editor, or Guest
Assumptions	Precondition: Abigail has logged into the system. *An Administrator can send an email to an individual, member of a group they belong to, or to all the Users of the system. *An Editor can send email to an individual, members of a group they belong to, or the Administrator of the system. *A Guest can only send an email to the Administrator of the system.
Steps	1. Abigail clicks on the ‘email’ link on the main page. 2. Admin Abigail (also Editors) specifies recipients in ‘to:’ field 3. Abigail fills in the subject field, and their message in the main text box. 4. Abigail clicks on the ‘mail’ button. 5. System sends email message and displays appropriate response to the screen.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.8: Use Case Email

Use Case 8: Add User

The following use case is used to define optimal paths for an Administrator to be able to add a new user to the system.

Use Case	***Add User: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to add a new user to the system.
Scenario	Abigail the administrator has receives a request from Molly that a new student intern has joined her group and needs to be added. Molly specifies the permissions that the student, Carla will require. Abigail adds Carla as a user and sends an email to Carla and Molly.
Actor(s)	Administrat
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on 'Add User' Link on the administrator interface main page. 2. Abigail types in Carla's name, contact info, group, etc., etc… 3. Abigail will select 'Submit' button 4. System will verify the information. 5. If required information is missing, the system will prompt for correction. 6. System will notify if Carla's account has been successfully created. 7. If Abigail already exists, error message will be displayed. 8. If Abigail wants to create another user they may hit the 'reset' button.
Non-Functional	*Performance:* Should take less than 30 seconds to process after hitting 'submit' *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	Not decided if application will first do a check to see if user already exists and display search results??

Table 4.9: Use Case Add User

<u>Use Case 9: Delete User</u>

The following use case is used to define optimal paths for an Administrator to be able to delete an existing user from the system.

Use Case	***Delete User: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to delete an existing user from the system.
Scenario	Abigail the administrator receives and email from manager Molly that her top programmer Zed is leaving the Survey for Greener pastures. Molly sends an email to Abigail requesting that Zed's account be deleted. Abigail deletes Zed's account and sends an email to Molly.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'User Search' link. 2. System displays the 'search' page. 3. Abigail enters search criteria into one of the following search fields (Zed's first name, Last Name, user ID, group, etc, etc…) 4. System displays a table on the 'search Results' page. 5. Abigail clicks on the 'delete' button next to the selected user. 6. Abigail is prompted if they really want to delete Zed 'YES or NO'. 7. System displays appropriate response to the previous question. 8. System should note in the log file which user got deleted.
Non-Functional	*Performance:* Should take less than 30 seconds to process after hitting 'yes' Should not return more than 100 search results, if more system will ask user to refine their search. *Reliability*: Should work every time, and only delete the user that is specified. *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.10: Use Case Delete User

<u>Use Case 10: Update User Information</u>

The following use case is used to define optimal paths for an Administrator to be able to update an existing user's information in the system.

Use Case	***Update User Information: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to update an existing user's information on the system.
Scenario	Abigail the administrator receives an email from Molly that her student Carla has graduated and promoted to a full time position within the group. Carla needs some more permissions changes. Abigail completes the request and sends an email back to Molly and Carla.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'User Search' link. 2. System displays the 'Search' page. 3. Abigail enters search criteria into one of the following search fields (Carla's first name, Last Name, user ID, group, etc, etc…) 4. System displays a table on the 'Search Results'. 5. Abigail clicks on the 'update' button next to the selected user. 6. System displays 'Update User' page with the fields filled in with the known information. 7. Abigail modifies necessary fields. 8. Abigail clicks on the 'update' button located at the bottom of the page. 9. System will verify information. 10. If invalid information is entered the Abigail is prompted to correct this information. 11. System displays appropriate message to screen.
Non-Functional	*Performance:* Should take less than 30 seconds to process after hitting 'update' Should not return more than 100 search results, if more system will ask user to refine their search. *Reliability*: Should work every time, and only update the user's information that was specified. *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.11: Use Case Update User Information

Use Case 11: Password Reset/Expire

The following use case is used to define optimal paths for an Administrator to be able to reset a users password, or set the password to expire at some specified date.

Use Case	***Password Reset/Expire: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to reset a user's password, or have a user's password expire.
Scenario	It has been a while since Eddie (an Editor) the scientist has worked with the Terra applications and he has forgotten his password. Eddie asks Abigail the administrator to please reset his password. Abigail goes into the admin interface, resets Eddie's password, then notifies Eddie that he can logon with this new password and change it to a new one.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'search' link on the main page. 2. Abigail performs search to find desired user. 3. System displays search results page. 4. Abigail Clicks on 'Password' button next to desired user's name. 5. Abigail is promotes to change the password, or have it expire. 6. System verifies information, and displays appropriate error message.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.12: Use Case Password reset/expire

Use Case 12: Add Team

The following use case is used to define optimal paths for an Administrator to be able to add a team to the system.

Use Case	***Add Team: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to add a team to the system.
Scenario	Abigail the administrator has received a request via email that manager Molly needs to create a special group within her team to work an a special project. She includes in the email the new team name, the team members and special team information. Abigail will go into the administrator interface, add the team, then send an email to Molly confirming that the request was completed.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'team' link on the main page. 2. Abigail clicks on 'Add Team' link. 3. Abigail prompted to enter the team information. 4. Abigail click the 'Add' button 5. The system verifies the information 6. If any information is incomplete the system prompts the Abigail to correct it. 7. System verifies information and displays correct message to screen.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.13: Use Case Add Team

Use Case 13: Update Team Information

The following use case is used to define optimal paths for an Administrator to be able to update team information.

Use Case	***Update Team Information: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to update a team's information.
Scenario	Abigail the administrator has received an email from a manager named Frank, noting that frank has recently acquired two new members into his group. Frank would like his new group members added to his group. Abigail must go in and update the team information.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'Team' link on the main page. 2. Abigail clicks on the team name, which is a link to the team info page. 3. Abigail updates information 4. Abigail clicks the 'update' button at the bottom of the page. 5. System verifies information. 6. System prompts Abigail to fill in missing or incorrect and displays correct message to screen.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.14: Use Case Update Team Information

<u>Use Case 14: Delete Team</u>

The following use case is used to define optimal paths for an Administrator to be able to delete a team from the system.

Use Case	***Delete Team: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to delete a team from the system.
Scenario	Abigail the administrator is performing her weekly maintenance tasks on the system. She receives an email from Molly a manager of the wespt group, notifying Abigail that her group has been absorbed into other groups and no longer exists. Group wespt can be deleted.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'team' link on the main page. 2. Abigail clicks on the 'delete' button to the right of the team name. 3. Abigail is prompted with a 'yes' or 'no' question do they really want to do this. 4. System processes request and displays appropriate response on screen.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.15: Use Case Delete Team

Use Case 15: Log Off Users

The following use case is used to define optimal paths for an Administrator to be able to log off all currently connected users and display an appropriate message when the users try to log back in.

Use Case	***Log off Users: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to log off all currently connected users from the system.
Scenario	Abigail the administrator would like to log out all the users that are currently logged in so that she can perform some maintenance on the system.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on 'Active Users/Logs' Link on the administrator interface main page. 2. System shows a dynamic page displaying users currently logged in. 3. User clicks on the 'Log users out' button 4. System kicks all users off except Abigail, and displays appropriate message to screen. Page is dynamic and should update with the number of users logged in.
Non-Functional	*Performance:* All users should be logged out within 30 seconds. *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	We will need to consider blocking users from logging in again, until lock released?? And what about a message to be posted on the login page also.

Table 4.16: Use Case Log off Users

Use Case 16: Post MOTD (Message of the Day)

The following use case is used to define optimal paths for an Administrator to be able to post a Message of the Day that appears on the Login page.

Use Case	***Post MOTD (Message of the day): Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to post a MOTD (Message of the Day) that will appear on the Login page.
Scenario	Abigail the administrator would like to notify users that the system will be off line for maintenance over the weekend. She will do this by posting a message on the login page.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system. Whatever text or characters are in the text box when the post is made will appear as the MOTD on the login page.
Steps	1. Abigail clicks on 'MOTD' link on the Administrator Interface main page. 2. Abigail enters a MOTD in the text box, or modifies message that is in this field. 3. Abigail clicks on the 'update MOTD' button. 4. System displays appropriate message.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.17: Use Case Post MOTD

Use Case 17: View Active Users and Logs

The following use case is used to define optimal paths for an Administrator to be able to view which users are currently logged in and view the log files.

Use Case	***View Active Users and Logs: Administrator***
Description	This use case describes the interactions that take place when an Administrator wants to view how many users are logged in, or view the log files.
Scenario	Abigail the administrator would like to see how many users are currently logged on to the system. Then she would like to view the web logs.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'System Status' link on the main page. 2. System displays dynamic page that shows the number of users that are logged in. 3. To view the log files, Abigail clicks on the 'Log' button. 4. System displays log file on screen, or appropriate message.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	

Table 4.18: Use Case View Active Users and Logs

Use Case 18: Administrator Search

The following use case is used to define optimal paths for an Administrator to be able to search the database for user and team information.

Use Case	***Administrator Search***
Description	This use case describes the interactions that take place when an Administrator wants to search for users, or groups and have the system return matching results.
Scenario	Abigail the administrator would like to search to see if a user currently exists in the database, so she can update their settings.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'Search' link located on main page of Admin interface. 2. Abigail enters search criteria. 3. Abigail clicks the 'Search' button. 4. System verifies information and prompts Abigail for corrections if necessary. 5. System displays results page.
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	Should the number of results that it returns limit the search???

Table 4.19: Use Case Admin Search

Use Case 19: Add Information Fields

The following use case is used to define optimal paths for an Administrator to be able to add new fields or information to the user database.

Use Case	***Add Information Fields***
Description	This use case describes the interactions that take place when an Administrator wants to add user information fields.
Scenario	Abigail the Administrator receives an email request from Dana the application developer for the Photo Archive. Dana would like the TerraUser application to store information on whether the user exited the photo Archive using view A, B, or C. This information will be kept track of in the TerraUser database, and used in the future for other application designs. Abigail navigates to the Add information pave, fills out the form and hits 'submit'. Abigail then sends an email to Dana with the application communication details.
Actor(s)	Administrator
Assumptions	Precondition: Abigail has logged into the system.
Steps	1. Abigail clicks on the 'Add Info' link on the main page. 2. ??? Issues here that have not yet been resolved by design team!!!
Non-Functional	*Performance:* *Reliability*: *Frequency:* *Fault Tolerance:* *Priority:*
Issues	Implementation issues still awaiting resolution.

Table 4.19: Use Case Add Information Fields

5.1 Internal Interfaces

The TerraUser software will be accessed on the Internet through a web-browser. Users must have an Internet connection and a standard browser to access the software. The TerraUser software is a database driven web-application. The software will use the most current standards for implementing JSP pages accessing a MySQL database via JDBC. All HTTP transfers (data over the internet) and SQL access (data from a local database) will be handled by internal functions of the technologies being used.

5.3 Software Interfaces

The TerraUser software will act as an interface between the user and applications they have access to. The applications can be only be accessed through the software. When the user selects an application to run, the software will send the required information. At any time the application is running, a request from the application can be made back to the software to gain any new or different information needed. The request can be made from any of the applications. A single feature in the software will handle all requests. When new applications need to interface to the software, the form of the request will be used in the new application and the software will be ready for the new application. The requests for information can be requested only with valid user login.

5.4 Communication Interfaces

The TerraUser software will use a web-based interface. The software will generate dynamic web pages viewed via HTTP/HTTPS. The software will communicate with the user's web browser using the most current HTTP/HTTPS and HTML standards. All communication from the software to the user will be prompted by user input. All communication from the user will be done through entering data and selecting items on the web pages dynamically created by the software.

6. Performance Requirements

The software must meet certain performance requirements to maximize its usefulness when it is fully implemented and installed.

1. The development of the system must use the technical tools/languages as specified by the USGS sponsor. See section 2.6 for technical constraints.

2. Usability
   - User logins should take no longer than 30 seconds.
   - Access from the interface to the applications should take less than 5 seconds.
   - An error should generate one friendly message that can be easily understood and direct the user toward appropriate help.
   - Specific page-long help should be available at any time by clicking a help icon.
   - Users can only be logged in once.
   - A message will be displayed when server is unavailable for logins.
   - The system will use a web-based interface similar to popular computer interfaces.
   - It shall be accessible by all standard methods of Internet access (modem, DSL, cable, satellite, LAN, broadband, etc.).
   - It should require less than fifteen minutes of training to use.

3. Training
   - Users should require less than fifteen minutes of training to use the interface.
   - Administrators should require at most one hour worth of training to use application.
   - Application developers that want to use the interface should require less than one hour of training to interface (Reading the documentation should be enough).

4. Maintenance
   - The code will be well commented.
   - Documentation will be created about the software design and implementation for future maintenance.
   - The system should be accessible from any machine.
   - The system will reside on a network visible to the Internet.

5. Security
   - The system will enforce user login, which will determine level of accessibility.
   - It shall use encryption to send information across networks.

6. Scalability
   - The system will use a modular design.
   - The system will use a simple and well-documented interface between modules and external software.

7. Design constraints

Throughout the design process there are many design constraints that face this project. Since this project is a web application, there are many more hardware and software constraints that the design team has to work around. The following discusses a set of reasonable measures that will be used later during acceptance testing.

Bandwidth

There is a communications cost associated with sending large amounts of information between the server and the client. We have to keep in mind the speed of the connection between our server and the Internet, as well as the speed of the client's connection to the Internet. We will assume that the minimum Internet connection that a user will have is a 56K modem and that the minimum connection speed between our server and the Internet is 10Mbps.

Hardware

Another constraint to consider in our design is the hardware that is going to be used to run this application. Because this is a web application, we are going to assume that the user's screen is at least 800x600 pixels and use a browser-safe 216-color palette when picking the application's colors. The computer running the application can be any computer meeting the requirements of running a standard HTML 4.0 compatible browser.

Ease of Maintenance

The System must be inexpensive to maintain and repair. Since the U.S. Geological Survey has little money to invest in this venture, we will make sure that costs are negligible and the system is easily maintainable.

Scalability

The system will be scalable both in the size of the system and how many records, users, and data the system can support.

Browser Independence

The application must function properly in all standard browsers. Application will be fully functional in browsers that support HTML 4.0+. The pages will be fully functional in Lynx (pretty rollovers not being considered "functional"). Note: Lynx emulates how many people with disabilities access the web, and is used for testing purposes. Typically, if client-side applications (JavaScript or Java) are only decorative and/or are backed up with NOSCRIPT tags and server-side checks, you're fine. Lynx is used for testing purposes.

Training

There should be little to no training required to use the application. The user will be able to perform each action as described in the use cases (section 3) within a minute each.

8. OTHER REQUIREMENTS

There are other requirements that do not impact the functionality of the product:

Deliverables

- The documents must be in electronic forms: PDF, HTML, or MS Word.
- Documentation and reports should be formatted so they will make usable two-sided hardcopies.
- Documentation should be readable on a computer screen.
- Documentation on setup/implementation of system, database, and security must be provided.
- The code should be well commented in electronic form.
- Manuals for programmers, system/database/web administrators must be provided.
- Online, context-sensitive help must be provided.

The system must follow design guidelines presented in the capstone design course.

9. CONCLUSION

The information provided in this functional specifications documentation serves as a bridge between software requirements and design process for the web-based user management package for the USGS in Flagstaff. The TerraUser team has analyzed the problem, translated the requirements into technical descriptions, and developed a high-level design. This project is economically viable and will be a valuable tool to our clients when managing data and TerraWeb applications.

The TerraUser team is confident that we will design and develop a successful product. We look forward to sharing our product with the USGS to benefit and improve their business.