

Junjie Wan. A Web-Based Database Management System. A Master's Paper for the M.S. in I.S degree. April, 2017. 51 pages. Advisor: Robert Capra.

This project develops a web database management system for Gfeller Center at the University of North Carolina at Chapel Hill. This system allows users to upload and download structured data in terms of specific requirements.

The Gfeller Center works on research that studies the health condition of athletes. Researchers have collected a large amount of data from previous studies and they keep collecting data as the research goes on. The researchers need tools to help manage and organize the data and need advanced functionality to support their research. Therefore, this project helps to develop a web database system to manage data according to the requirements of the researchers in Gfeller Center. The system development includes requirements analysis, functionality analysis, and interface design. Finally, this project proposes a plan for usability testing to evaluate the system.

Headings:

Information Systems

Requirements Analysis

Interface Design

System Implementation

Usability Testing

A WEB-BASED DATABASE MANAGEMENT SYSTEM

by
Junjie Wan

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2017

Approved by

Robert Capra

Table of Contents

1. Introduction.....	2
2. Literature Review.....	4
2.1 Database System.....	4
2.2 Requirements Gathering.....	4
2.3 Usability Testing.....	6
3. System Design and Implementation	8
3.1 Requirements Analysis	8
3.2 Architecture	10
3.2.1 <i>Uploading Subsystem</i>	12
3.2.2 <i>Variable Subsystem</i>	21
3.2.3 <i>Downloading subsystem</i>	27
3.2.4 <i>File subsystem</i>	31
3.3 Interface	34
4. Usability Testing.....	39
4.1 Tasks.....	40
4.2 Questionnaire.....	42
5. Conclusion	45
References.....	47

1. Introduction

The Sports Medicine Center, Gfeller Center of UNC, are working on a research project which studies the health condition of athletes. Researchers have collected a large amount of data from previous studies and they keep collecting data as the research goes on. The data comes from various sources based on different studies, sports, age levels, funding resources, and investigators. So the size of data keeps growing. However, the researchers use spreadsheet management software like Excel to manage the data. Furthermore, they have frequent interaction with the data and sometimes require advanced functions like merging tables and exporting part of a table. Those operations are time-consuming and have impact on the efficiency of research. Therefore, they need a database management system with the ability of merging all the data into an integrated database and providing advanced functions to serve for the research.

I developed a database management system to help the Gfeller Center researchers manage the collected data. Briefly, the DBMS is constructed as a web system running in a LAMP (Linux, Apache, MySQL, PHP) system. The interface is created using HTML, CSS, and JavaScript. Also I have taken advantage of Bootstrap¹ to design the layout and decorate the components. Some JavaScript plug-ins are used for specific implementation. I will introduce them later in the architecture section. The interaction between website

and database is realized by PHP, and the database used is MySQL.

This paper describes the process of developing the database management system, including requirement analysis, system design, and system implementation. For system design and implementation, I describe them based on subsystems and their use cases, since use cases are basic components of a system. Moreover, the development of a DBMS which simply realizes required functions is not enough since the system serves for users who frequently deal with large data sets. One of the most important aspects of building a web system is testing for usability. Users are accustomed to being able to figure out how to use a web system quickly. Most of them will not take the time to figure out a site that is not usable. Usable web systems increase user satisfaction whereas web systems that violate usability conventions can confuse users and result in the waste of time.. Therefore, this paper will also describe a proposed plan for usability testing of the system including the goals and the tasks.

2. Literature Review

2.1 Database System

“A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data.” (Rouse, 2015, paragraph 1). Nowadays, there are millions of DBMSs being used by organizations around the world. They are running on various platform and environments. Also, they provide various services to users.

In 1997, McHugh proposed a DBMS named Lore (McHugh, 1997). This system was used to manage semistructured data. The data could be unsystematic but Lore still tried to keep the original structure and did some processes to build a structured database. Lore would parse and pre-process the data then store them in the database. Lore processes data by query language and uses a “Query Plan Generator” to optimize queries according to the structure of data.

2.2 Requirements Gathering

Mifsud presented that “A requirement is a statement about an intended product that specifies what it should do or how to do it. For requirements to be effectively

implemented and measured, they must be specific, unambiguous and clear” (Mifsud, 2013, section 1). He also indicated that “requirement gathering is an iterative process” (Mifsud, 2013, section 2). For example, in the process of requirements analysis a developer may find out that what he thinks the system should do is not exactly what the user wants. The developer could arrange another talk with the user and clarify the ambiguity. There are several ways of gathering requirements including interviews, questionnaire, and direct observation. According to different conditions, requirements gathering processes need to be selected properly and the practices are also important during gathering requirements. Young has listed several suggested practices in his article. For instance, “Document the rationale for each requirement”, “Establish a mechanism to control changes to requirements and new requirements”, and “Use requirements gathering techniques that are known, familiar, and proven in the organization such as requirements workshops, prototyping, and storyboards” (Young, 2002, p10).

Basically, the methods used for requirements gathering in this project were iterative interviews with the users. Then I documented each requirement and abstracted out corresponding module or components to serve for next interview. If the users were satisfied with my draft implementation for a requirement, I would finalize the iterative process of that requirement.

2.3 Usability Testing

“Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users.” (Usability testing, 2017, paragraph 1).

Holzinger (2005) said that “Testing with end users is the most fundamental usability method and is in some sense indispensable. It provides direct information about how people use our systems and their exact problems with a specific interface” (Holzinger, 2005, p73). He also described three methods of usability testing -- thinking aloud (THA), field observation, and questionnaires. Holzinger thought that thinking aloud could be the most useful method. Thinking aloud is a method of suggesting users to do brain storm during using a system and share their thoughts. When users share their experience immediately, developers are able to find out misleading parts of the system. Field observation is viewed as the simplest method by Holzinger. It needs developers to observe users when they are using a system and take notes to record their actions. Questionnaire is also important in some cases. A well designed questionnaire can help to find out users’ preferred features.

Nowadays, organizations pay more attention to the use and costs of systems, while little attention has been paid to the system effectiveness. “Evaluation of system effectiveness is difficult due to its multidimensionality, its quantitative and qualitative aspects, and the multiple. and often conflicting, evaluator viewpoints.” (Hamilton, 1981, p55). Furthermore, the main purpose of information system functionality performances

evaluation is improving quality of maintenance.

Learnability is one of the five dimensions of usability testing. Vatrapu (2008) indicated that learnability can be evaluated by analyzing qualitative data and quantitative information collected from experiments and case studies. Efficiency is another dimension of usability testing. Hamilton and Chervany (1981) proposed that system can be evaluated by efficiency. This evaluation perspective focus on the resources used by the system including human resource, financial cost, and materials.

3. System Design and Implementation

3.1 Requirements Analysis

“Requirement analysis is the process of analyzing a new or modified product and determining the user expectations for the product” (Rouse, 2007, paragraph 1). In detail, it includes specifying input data, obtained result, and output data.

For the system presented in this paper, the requirement analysis has gone through an iterative process in which the developer and user keep revising and optimizing the system in terms of user interfaces, functions, and data flow. Before the users are satisfied with the system design, we have arranged several meetings to talk about their requirements and my proposed design.

Here are some of the important requirements:

- The system should be able to process Excel spreadsheets. Because users feel like doing pre-process before uploading and post-process after downloading, in this case Excel is their most frequently used software. Therefore, Excel spreadsheets will be the most appropriate carrier of data, and the system is required to allow users to upload and download Excel spreadsheet.

- Because data is collected from different studies, the users want to upload data based on studies. In other words, they like to combine those data from the same study into the same table of the database. Besides, there are four attributes to categorize data: age level, sports, primary investigator, and funding source. Every set of data which is going to be uploaded should be marked by those categories at first. Different sets of data of the same study could have different value in any of those categories.
- For one study, there will be many spreadsheets to be uploaded to the database management system, and each one could have its own distinct variables and also several common variables with other spreadsheets. Considering one of the requirements is to combine those data from the same study into the same table of database, the system should be able to merge those spreadsheets into a table by adding new attributes and processing duplicated attributes.
- When downloading, based on different research purposes the users may wish to download only a subset of attributes and relative data from a complete table. Also they may need to repeatedly download these subset of attributes.
- Usually, the users would like to compare between several studies in terms of some common attributes. Therefore, they like to have a function to present common attributes based on the studies they select. Also, these common attributes are prepared to be downloaded repeatedly like the previous

requirement.

- As for downloading, the users would like to preview the data to be downloaded and to set the target directory.
- The users need a file manager to manage files which are uploaded to the server. They need basic functions including renaming, moving and deleting.

3.2 Architecture

According to the requirements collected from the users, the system needs to provide four main functions which are uploading, downloading, attribute manager and file manager. Therefore, the system is designed to be combined by four subsystems – uploading subsystem, variable subsystem, downloading subsystem, and file subsystem. Figure1 is the use-case model diagram for the data management system. It was created using Popkin Software's system architect and represents the relationships between the actors and the use cases.

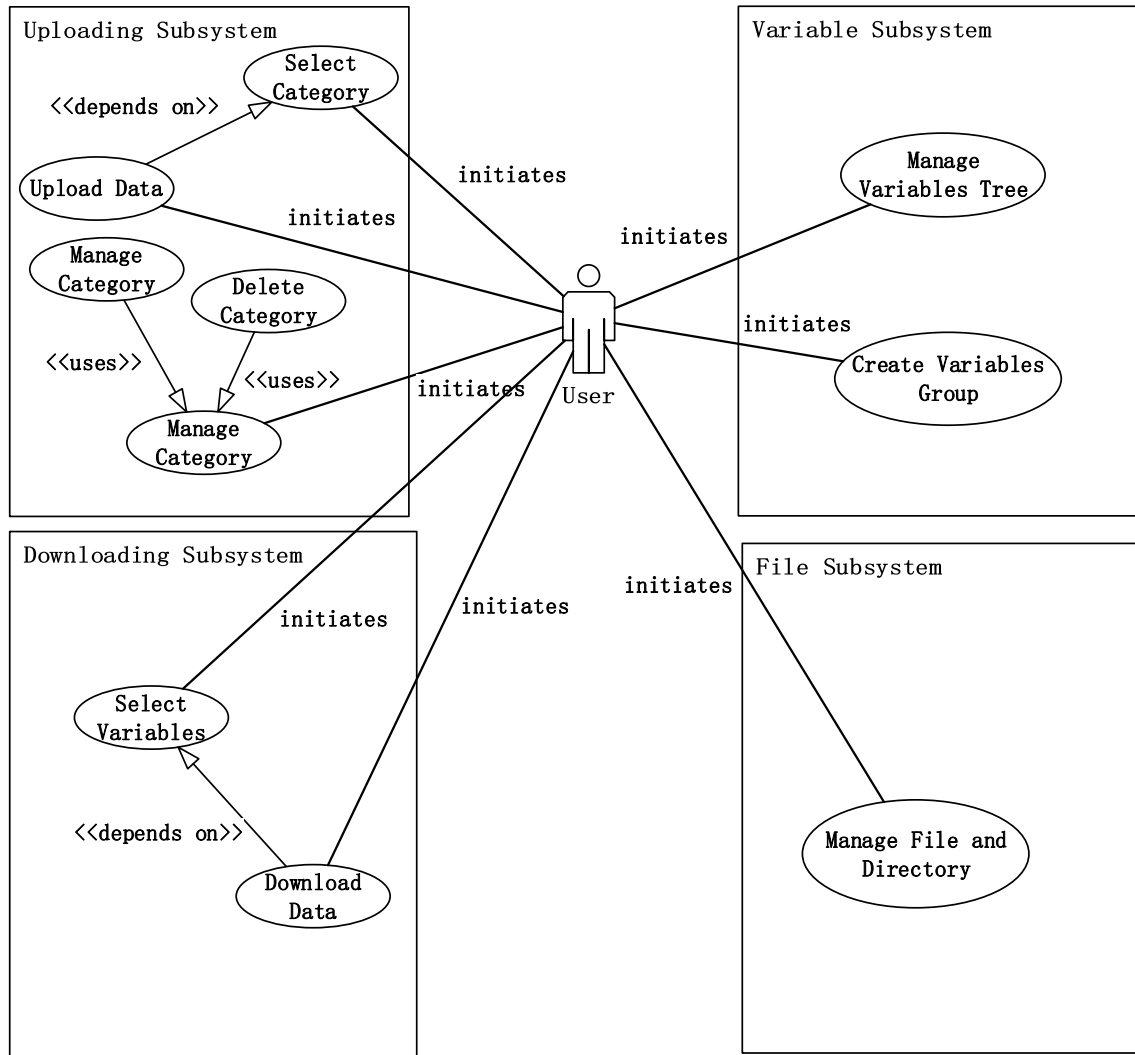


Figure 1

In the following parts, the project will describe important use cases for each subsystem. For each use case, I will present its implementation including interface and critical code segment.

3.2.1 Uploading Subsystem

Imagine that a researcher named Jay has a dataset to be uploaded. The dataset is contained in an Excel spreadsheet named “DSET26”. It comes from a previous study called “Athletes Brain Health” and the data of the dataset is collected from high school basketball athletes. The primary investigator who collects the data is Joshua Wayne and the funding source is supported by Company K.

Use Case 1 – Manage category

This use case describes the event of a researcher managing the value of category. It is an abstract use case which is extracted from use case - add category and use case - delete category.

Before Jay uploads the dataset, he needs to set categories to provide basic information. However, Jay finds that “Athletes Brain Health” is a new study to this system and *Primary Investigator* doesn’t contain the value he needs (Figure 2a and Figure 2b). So he decides to add corresponding values to *Study* and *Primary Investigator*. In ‘Add Category’ module, Jay checks the ‘Study’ radio button and types the new study name “Athletes Brain Health” into the ‘Name’ bar then clicks ‘Add’ button (Figure 3a). Also, to add a new value to *Primary Investigator* Jay does the similar action (Figure 3b). Finally, both *Study* and *Primary Investigator* category have necessary values now (Figure

4a and Figure 4b). Specially, when Jay adds a new study, the system will create a new directory with the same name under the directory “/data/upload”, which is for uploading other datasets of this study to the same place in the server.

Set Category

Study:

Age Level:

Sports:

Primary Investigator:

Funding Source:

Figure 2a

Set Category

Study:

Age Level:

Sports:

Primary Investigator:

Funding Source:

Figure 2b

Add Category

Name:

Study Age Level Sport Primary Investigator Funding Source

Figure 3a

Add Category

Name:

Study Age Level Sport Primary Investigator Funding Source

Figure 3b

The screenshot shows a form titled "Set Category" with five input fields. The "Study:" field is a dropdown menu. The "Age Level:" field is a dropdown menu that is currently open, showing two options: "Athlete Brain Health" and "College Students Health". The "Sports:" field is a dropdown menu. The "Primary Investigator:" field is a dropdown menu. The "Funding Source:" field is a dropdown menu.

Figure 4a

The screenshot shows the same "Set Category" form. In this view, the "Funding Source:" dropdown menu is open, showing two options: "Hospital X" and "Joshua Wayne". The other fields are the same as in Figure 4a.

Figure 4b

Implementation: After Jay types a new category name and selects a radio button, once he clicks the ‘Add’ button, the values of input controls will be submitted to the currently same page by HTTP POST request. After the page reloads, the conditional will check if the value is posted and is valid, if yes, the new value of the category will be inserted to the database. Code Segment 1 shows the query language of inserting, also it shows the directory creating statements triggered by adding new study.

```
$insertCategory = "insert into ".$radio_group." (name) values('".$category_name."')";
mysqli_query($db, $insertCategory);

if($radio_group == "studies"){
    mkdir ($filefolder.$category_name);
}
```

Code Segment 1

Now, Jay realizes that he has added a wrong value like “baseball” to *Sports* and he wants to remove the wrong value. He would select “baseball” from the list provided by *Sports* and click ‘Delete’ button (Figure 5).

Figure 5

Implementation: Like adding category, the values that he selects will be submitted to the currently same page by HTTP POST request. After being checked by conditionals the value will be removed from the database. Code segment 2 shows the implementation of the system removing a value from *Sports*.

```
$deleteCategory = "delete from sports where name='".$d_sport.'";  
mysqli_query($db, $deleteCategory);
```

Code Segment 2

Use Case 2 – Select categories

This use case describes the event of a researcher setting values for each category. It is precondition of the use case – Upload Excel spreadsheet.

Now, Jay has all required values of categories. All he needs to do is to select a value for each category (Figure 6).

The image shows a form titled "Set Category" with five rows of dropdown menus. Each row has an icon to the left of the label: a book for "Study", a bar chart for "Age Level", a trophy for "Sports", a person for "Primary Investigator", and a briefcase for "Funding Source". The selected values are "Athlete Brain Health", "High School", "Basketball", "Joshua Wayne", and "Company K" respectively.

Category	Selected Value
Study	Athlete Brain Health
Age Level	High School
Sports	Basketball
Primary Investigator	Joshua Wayne
Funding Source	Company K

Figure 6

Use Case 3 – Upload data

This use case describes the event of a researcher uploading an Excel spreadsheet. It depends on the use case -- Set categories, which means the researcher must set all categories before successfully uploading.

After setting all the categories, Jay is ready to upload the dataset “DSET26”. He clicks ‘Choose File’ button then finds and select the Excel file named “DSET26” in the dialog opened by the browser. Next, he selects the target directory in terms of current study from the directory list. Then he clicks ‘Upload’ button to finish uploading (Figure 6). The prompt shown below the ‘Upload’ button will provide prompts about uploading status. If the file uploaded successfully, the green bar will show (Figure 7a), otherwise, one of the following red bars will indicate the missing information (Figure 7b - Figure 7f).

Upload File

Choose File DSET26.xlsx

Directory: /upload/Athlete Brain Health/ ▾

Upload

Waiting for uploading. Please set category first.

Figure 6

Well done! You successfully upload this file.

Figure 7a

Failed. Please choose file and target directory.

Figure 7b

Failed. Please choose file to upload.

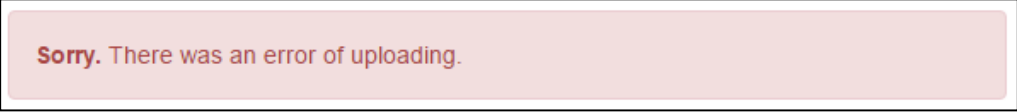
Figure 7c

Failed. Please choose target directory.

Figure 7d

Failed. Please set categories.

Figure 7e



Sorry. There was an error of uploading.

Figure 7f

Implementation: As long as Jay clicks 'Upload' button, the information he has provided including categories, spreadsheet, and target directory will be submitted to the current page by HTTP POST request. At the beginning of this page, it will check if the information is complete and valid. If there is some information missing, the bar will return one of the error messages, otherwise, the page will start to process uploaded spreadsheet. Code segment 3 shows a small part of conditionals for checking the completeness and validness of submitted information and deciding corresponding prompt and bar theme.

```

if(!isset($_FILES["fileToUpload"]) && !isset($_POST["target_dir"])){
?>
    <script>
        $(document).ready(function(){
            $("#upload_info").attr("class", "alert alert-info");
            $("#upload_info").html("<strong>Waiting for uploading.</strong> Please
                set category first.");
        });
    </script>
<?php
}
else if($_FILES["fileToUpload"]["name"] == "" && isset($_POST["target_dir"]) && $_
    _POST["target_dir"] != ""){
?>

    <script>
        $(document).ready(function(){
            $("#upload_info").attr("class", "alert alert-danger");
            $("#upload_info").html("<strong>Failed.</strong> Please choose file
                to upload.");
        });
    </script>
<?php
}

```

Code Segment 3

If all provided information is complete and valid, the system will start to upload the spreadsheet. Firstly, the system checks if the study is a brand new study, if it is, the system will create a new table for this study and add four basic category attribute to the table in the database (Code segment 4). A query statement “SHOW TABLES LIKE [TABLE NAME]” is used in the code segment.

```
$query = "show tables like '" . $study . "'";
if(mysqli_num_rows(mysqli_query($db, $query)) != 1){
    //Create new table with four basic variables.
    $createTable = "create table " . $study . "(id int primary key, agelevel
        varchar(20), sport varchar(20), investigator varchar(30), fundingsource
        varchar(30))";
    mysqli_query($db, $createTable);
}
```

Code Segment 4

To process Excel spreadsheets, I take advantage of a library called “phpExcel” to extract data of spreadsheet and push them to PHP arrays. Then there could be three cases to handle with:

1. There would be new attributes from the uploaded dataset, which don't exist in the table.
2. There would be some records identified by “id” already existing in the table.
3. There would be some new records which don't exist in the table..

For the first case, I store the existing attribute names in an array and the attribute names of the spreadsheet in another, then I compare these two arrays to find out new attributes and add new attributes to related table in the database (Code segment 5).

```

//1. Get the first line of the spreadsheet.
$cellIterator->setIterateOnlyExistingCells(true);
foreach($cellIterator as $cell){
    array_push($variables_array, $cell->getValue());
    $totalColumn++;
}

//2. Get variables already existing in the table, and store them in an
array.
$query = "show columns from " . $study;
$result = mysqli_query($db, $query);
$existVariables = array();
while($row = mysqli_fetch_assoc($result)){
    array_push($existVariables, $row['Field']);
}

//3. Check each variable of the spreadsheet to see if it already exists.
//If not add the var into the table.
$addColumn = "alter table " . $study . " add(";
for ($r = 1; $r < count($variables_array); $r++){
    if(!in_array($variables_array[$r], $existVariables)){
        $hasUnexistVar = true;
        $addColumn .= $variables_array[$r] . " varchar(30),";
    }
}
if($hasUnexistVar){
    $addColumn = substr($addColumn, 0, -1);
    $addColumn .= ")";
    mysqli_query($db, $addColumn);
}

```

Code Segment 5

For the second case, I update all the values for the existing record using query statement “INSERT INTO [TABLE NAME] ON DUPLICATE KEY UPDATE [COLUMN NAME_1] = [VALUE_1], [COLUMN NAME_2] = [VALUE_2] ...” (Code Segment 6). Such query statement can significantly improve efficiency, since it updates all the values by calling the time-consuming function “mysqli_query()” once. In other words, it significantly reduces the times of communicating with database than updating the value for each attribute.

```

$updateRow = "insert into " . $study . "(id) values(" . $fields[0].") on duplicate
key update ";
$updateRow .= "agelevel='".$age_level."',sport='".$sport."',investigator='".$
investigator."',fundingsource='".$fundingsource.'";
for($k = 1; $k < count($fields)-1; $k++){
    $updateRow .= "," . $variables_array[$k] . "='" . $fields[$k] . "'";
}
mysqli_query($db, $updateRow);|

```

Code Segment 6

For the third case, I use basic query statement “INSERT INTO [TABLE NAME] ([COLUMN NAME_1], [COLUMN_NAME_2], ...) VALUES ([VALUE_1], [VALUE_2], ...) (Code Segment 7).

```

$createNewRow = "insert into " . $study . "(
    id,agelevel,sport,investigator,fundingsource";
for($k = 1; $k < count($fields); $k++){
    $createNewRow .= "," . $variables_array[$k];
}
$createNewRow .= ") values(" . $fields[0] . "," . $age_level . "," . $sport . "," . $
    . $investigator . "," . $fundingsource . """;
for($k = 1; $k < count($fields); $k++){
    $createNewRow .= "," . $fields[$k] . """;
}
$createNewRow .= """;
mysqli_query($db, $createNewRow);

```

Code Segment 7

3.2.2 Variable Subsystem

Jay and his colleagues have populated the database by uploading several datasets of different studies. Now, in order to start further research, Jay wants to download part of the data from the study “BrainHealth”. But before that, Jay needs to create a variable group combined with variables which he would like to download from the study.

Use Case 1 – Create variable(s) group

This use case describes the event of a researcher creating variable(s) group by selecting part or all of (common) attributes from one or more tables of database.

Jay clicks the ‘Create Group’ button, then all studies will be shown as checkboxes. Jay checks ‘BrainHealth’ to be the related study, then all attributes of the study will be listed as checkboxes below the ‘Common Variables’ label (Figure 8a). Jay checks several attributes and click ‘Create’ button, then a new tree node is created below ‘Variable Tree’ label and it is ready to be renamed (Figure 8b). He names it “group1” and the group has been created successfully. When he clicks any node in the group, the ‘Related Study’ label will present the name of related study of this variables group (Figure 9).

The screenshot shows a web interface titled "Operations". At the top left is a green "Create Group" button, and at the top right is a "Search" input field. Below this is the "Related Study" section, which contains four checkboxes: "Athlete Brain Health", "BrainHealth" (which is checked), "College Students Health", and "HeartHealth". Underneath is the "Common Variables" section, which lists 14 items, each with an unchecked checkbox: "Select All", "agelevel", "sport", "investigator", "fundingsource", "test_date", "test_time", "subject_id", "birth_date", "signature", "PICSS_completed", "PICSS_lastname", "PICSS_firstname", "PICSS_middleinitial", "PICSS_question_1", "PICSS_question_2", "PICSS_question_3", "PICSS_question_4", "PICSS_question_5", "PICSS_question_6", "PICSS_question_7", "PICSS_question_8", "PICSS_question_9", "PICSS_question_10", "PICSS_question_11", "PICSS_question_12", and "PICSS_question_13".

Figure 8a

pcss_question_13
 pcss_question_14
 pcss_question_15
 pcss_question_16
 pcss_question_17
 pcss_question_18

Variables Tree

Related Study:

Figure 8b

Variables Tree

Related Study: BrainHealth

- group1
 - agelevel
 - PICSS_question_7
 - PICSS_question_13
 - complex_attention_domain_pr
 - verm_delayed_target_rt
 - vism_selected

Figure 9

Implementation: When the ‘Create Group’ button is clicked, a JavaScript function named “create_group()” is invoked. Inside the function, AJAX sends a HTTP GET request to “get_studies.php” file, then “get_studies.php” retrieve all study names from database and return them as html data to AJAX. Next, that html data is shown in a pair of ‘<div>’ tags whose id is “group_setting” (Code Segment 8).

```
function create_group(){  
  
    $.ajax({  
        method: "GET",  
        url : "get_studies.php",  
        dataType: "html",  
        success: function(data){  
            $("#heading").attr("class", "panel-heading");  
            $("#heading").text("Related Study");  
            $("#group_setting").html(data);  
            document.getElementById('buttons').style.display = 'block';  
        }  
    });  
}
```

Code Segment 8

Similarly, when checking one or more study checkboxes, a function called “check_funtion()” is invoked. Inside this function, those study names who have been checked are pushed to an array called “study_array” and AJAX uses HTTP POST to send the array to “get_variables.php” file, then “get_variables.php” retrieves common attributes among those checked studies and wraps them in html elements and returns to AJAX. Finally, those attributes decorated in checkboxes are presented in the page. Code segment 9 shows the code of “check_function()”.

```

function check_function(){
    $("#var_heading").attr("class", "panel-heading");
    $("#var_heading").text("Common Variables");

    var studyRef = document.getElementsByName("study");
    var study_array = [];

    for(var i=0; i<studyRef.length; i++)
        if(studyRef[i].checked)
            study_array.push(studyRef[i].value);

    $.ajax({
        method: "POST",
        url : "get_variables.php",
        data : {checkedStudies : study_array},
        dataType: "html",
        success: function(data){
            $("#var_setting").attr("class", "panel-body");
            $("#var_setting").html(data);
        }
    });
}

```

Code Segment 9

After selecting all or part of the variables, once the ‘Create’ button is clicked, a function called “set_study()” is invoked (Code Segment 10). Specially, the variable tree is realized by taking advantage of a jquery plug-in called “jsTree”. It provides interactive trees structure, and it is easily extendable, themable and configurable. Therefore, inside function “set_study()”, an object named “tree” is created and a function of the object is invoked. When invoking the object’s function, a parameter “create_group” is sent as a message to the tree in order to create a new group which is a parent node with several child nodes.

```
function set_study(){
    var tree = $('#jstree').jstree(true);
    tree.refresh(false, false, 'create_group');
}
```

Code Segment 10

Use Case 2 - Manage variable(s) tree

This use case describes the event of a researcher managing nodes of variables tree by performing some simple actions.

Jay finds that he includes a useless variable named “PICSS_question_13” in “group1”. So he right-clicks the mouse on the variable and a list of operations is shown (Figure 10). Then he clicks ‘delete’ to remove the variable from the group. Similarly, to do other operations Jay simply needs to click other buttons. All the actions provided to managing variables tree are creating, renaming, deleting, copying, cutting, and pasting.

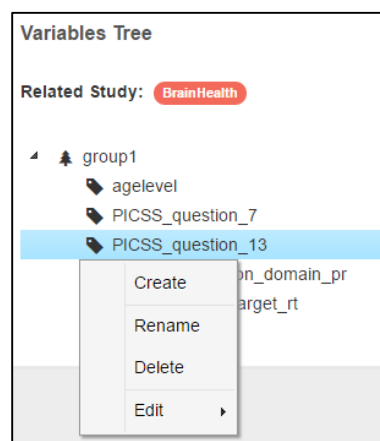


Figure 10

3.2.3 Downloading subsystem

Jay has already created a variables group combined with attributes he wants to download. Now, he decides to download the dataset which contains the variables and values of “group1”.

Use Case 1 – Select variable(s)

This use case describes the event of a researcher selecting variables to be downloaded. It is the precondition of the use case – Download data.

The variables tree is presented in the download page. The difference is that the nodes of the tree can be checked (Figure 11a). Because Jay wants to download all the variables of “group1”, he checks the checkbox of “group1” and all the variables are selected (Figure 11b). Now, the data panel allows Jay to preview the data he wants to download (Figure 12). Jay is able select the number of records to be shown per page and he can also retrieve record by typing any value of the record into the search bar on the top right.

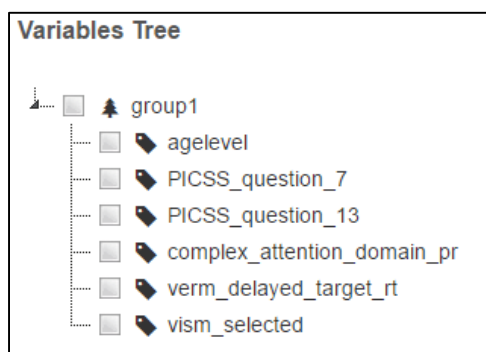


Figure 11a

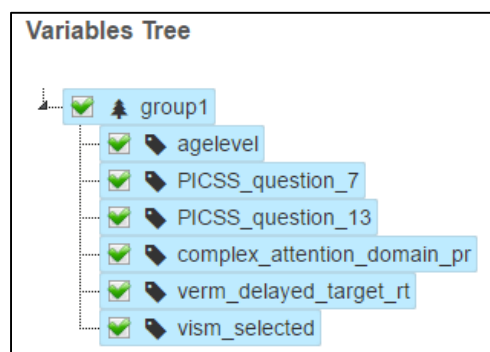


Figure 11b

Data Panel					
10 records per page		Search: <input type="text"/>			
agelevel	complex_attention_domain_pr	verm_delayed_target_rt	vism_selected	PICSS_question_7	PICSS_question_13
College	0	922	1		
College	75	701	1		
College	7	632	1		
College	40	813	1		
College	13	759	1		
College	2	761	1	0	0
College	16	933	1	1	0
College	75	880	1	0	0
College	61	795	1		
College	30	724	1		

Showing 1 to 10 of 1,008 entries

Previous **1** 2 3 4 5 ... 101 Next

Figure 12

Implementation: The variables tree is also implemented by using the JavaScript plugin in “jsTree”. In this case, I grant the “checkbox” function to the variables tree. Code segment 11 shows how the tree is loaded in the page and how to grant functions. The JavaScript code loads the JSON data of the tree and add function like “checkbox” to ‘plugins’, then all parameters and data are encapsulated and decorated by function “jstree()”. Finally, the variables tree is presented in the pair of “<div>” tags whose id is “jstree”.

```

$('#jstree').jstree({
  'core' : {
    'check_callback' : true,
    'data' : {
      'url' : function (node) {
        return 'resources/data/tree.json';
      },
      'data' : function (node) {
        return { 'id' : node.id };
      }
    }
  },
  'plugins':["checkbox", "search", " state", "types"]
})

```

Code Segment 11

After selecting some variables in the tree, the selected variables' names are pushed to an array and sent to a php file "get_data.php" by HTTP POST request through AJAX (Code Segment 12). "get_data.php" retrieves all the data in terms of the variables and related studies, then it decorates the data into a table format and sends it back using AJAX. Now, all the data is stored in a JavaScript variable named "data". I take advantage of a plug-in called "dataTables"¹ to decorate the table by calling "\$('#dataTable').dataTable()". Therefore, the theme and functions of the table are populated (Figure 13).

```

$.ajax({
  method: "POST",
  url : "get_data.php",
  data : {
    study : studyArray,
    variables : variableArray
  },
  dataType: "html",
  success: function(data){
    //console.log(data);
    $("#data-panel").html(data);
    $('#dataTable').dataTable();
    document.getElementById('submit-button').style.display = 'block';
  }
});

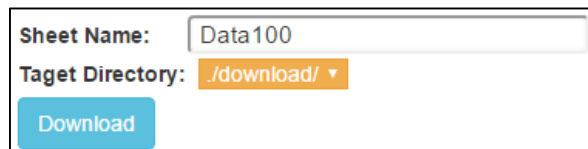
```

Code Segment 12

Use Case 2 – Download data

This use case describes an event of a researcher downloading data which is contained in an Excel spreadsheet from database. Its precondition is the use case – select variable(s).

After Jay previewing the data he plans to download in the data panel, he decides to download that part of data from the database. He types “Data100” into the text area of sheet name and he chooses “./download/” as target directory. Then he clicks the ‘Download’ button. Finally, Jay can find the spreadsheet named “Data100” under the directory “download/” in the server.



Sheet Name:

Target Directory: ./download/ ▾

Figure 13

Implementation: When the ‘Download’ button is clicked, a function named “download_data()” is invoked. Inside the function, JavaScript codes get the values of selected nodes, sheet name, and target directory, then it sends all those information to “ajax_download.php” by HTTP POST request in AJAX (Code Segment 13). In “ajax_download.php”, data will be retrieved from the database by query statements, then I use the library “phpExcel” to create a new Excel spreadsheet and store data in it.

```
$.ajax({
  method: "POST",
  url : "ajax_download.php",
  data : {
    study : studyArray,
    variables : variableArray,
    targetDir : targetDir,
    sheetName : sheetName
  },
  dataType: "html",
  success: function(data){
    //alert(data);
  }
});
```

Code Segment 13

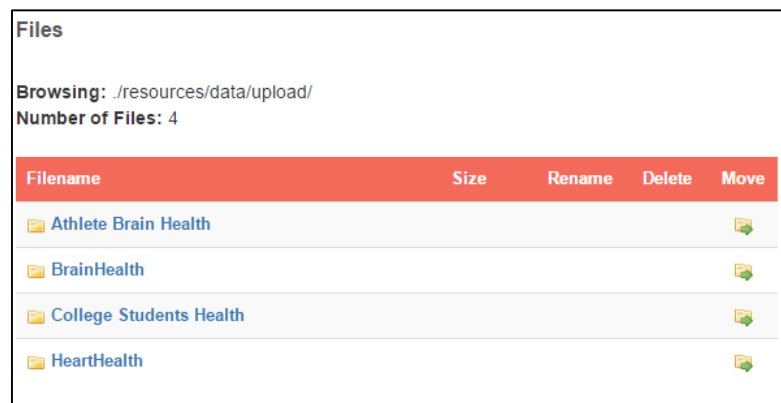
3.2.4 File subsystem

Jay and other researchers have uploaded many files to the system, some of the files need to be removed because they are too old, some of them need to be renamed, or some of them need to be moved to other directories. Now, Jay need to move a file which is named “DSET26” and under the directory “Athlete Brain Health” to the directory “HeartHealth” and delete a file named “PICSS-20160111050501” which is under the

directory “Athlete Brain Health”.

Use Case 1 – Manage file or directory

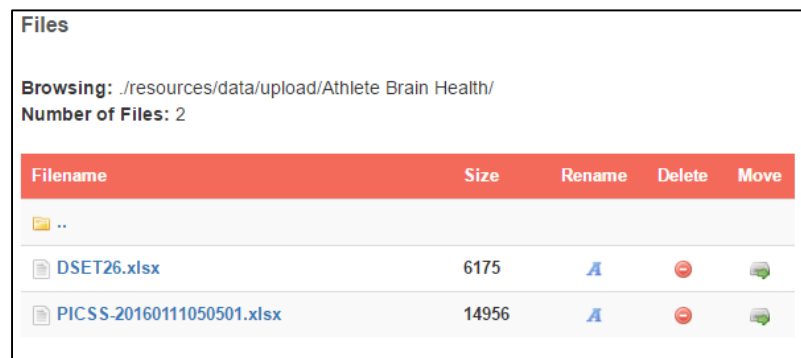
In the page of file manager, Jay clicks and opens the directory “Athlete Brain Health” (Figure 14a). Then he clicks the ‘move’ icon of “DSET26” and a new panel named “Move” is shown above the panel “Files” (Figure 14b). He opens the directory list and selects “/upload/BrainHealth/” (Figure 15). At last, he clicks the button ‘Move’ and a blue bar tells that he moved the file successfully (Figure 16).



The screenshot shows a file manager window titled "Files". The breadcrumb path is ".resources/data/upload/" and it indicates there are 4 files. A table lists the files with columns for Filename, Size, Rename, Delete, and Move. The files listed are Athlete Brain Health, BrainHealth, College Students Health, and HeartHealth, each with a move icon.

Filename	Size	Rename	Delete	Move
Athlete Brain Health				
BrainHealth				
College Students Health				
HeartHealth				

Figure 14a



The screenshot shows the file manager window titled "Files" with the breadcrumb path ".resources/data/upload/Athlete Brain Health/". It indicates there are 2 files. A table lists the files with columns for Filename, Size, Rename, Delete, and Move. The files listed are DSET26.xlsx (6175 bytes) and PICS S-20160111050501.xlsx (14956 bytes), each with a move icon.

Filename	Size	Rename	Delete	Move
..				
DSET26.xlsx	6175			
PICS S-20160111050501.xlsx	14956			

Figure 14b

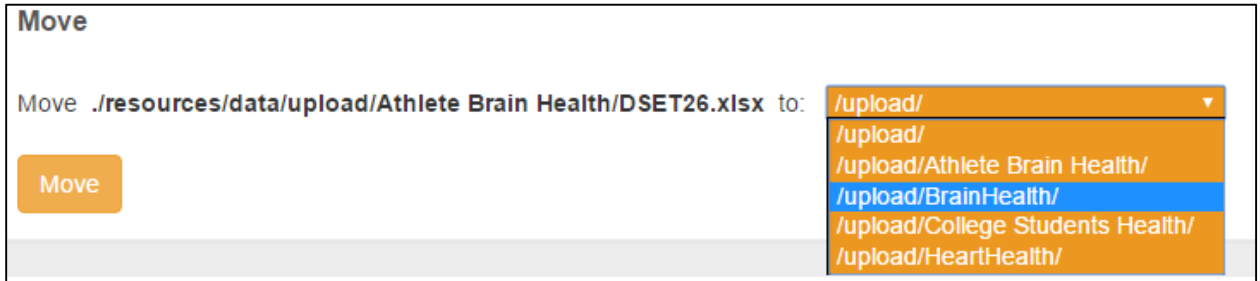


Figure 15



Figure 16

Now, Jay wants to remove the file “PICSS-20160111050501”. He clicks the ‘Delete’ icon and a new panel named “Delete” shows. Jay ensures to delete the file and clicks the ‘Yes’ button (Figure 17). Then a blue bar tells him the file is removed (Figure 18).



Figure 17

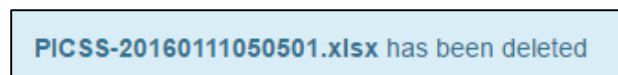


Figure 18

3.3 Interface

As I mentioned above, the database management system is combined by four subsystem - uploading subsystem, variable subsystem, downloading subsystem, and file subsystem. So I developed one page per subsystem. The four pages are:

- Upload Data page
- Download Data page
- Variable Manager page
- File manager page

Briefly, I used HTML, CSS, and JavaScript designing the user interface. And I took advantage of Bootstrap for decorating those components and arranging layout. Some JavaScript plug-ins were used for specific design like the variable tree and data table. For each page, I will describe the components and the reason for design.

Upload Data page:

This page has four panels – ‘set category’ panel, ‘upload file’ panel, ‘add category’ panel, and ‘delete category’ panel (Figure 19). The top two panels serve for uploading data, and the bottom two are combined to be category manager.

The reason why I put uploading module and category manager together in one page is that when a user has changed some values of those categories they can see the change

by clicking once in the first panel, rather than doing another click on the main menu. Moreover, if a user is unsatisfied with the change after checking the lists in ‘Upload Data’ page, they need extra clicks back to category manager of another page. Values of categories are presented in a drop-down list, which is according to user requirements. In ‘Upload File’ panel there is a bar of prompt. The system will tell users about the status of uploading, which is described in the “use case 3” of “uploading subsystem”. Figure 19 is a screenshot of ‘Upload Data’ page.

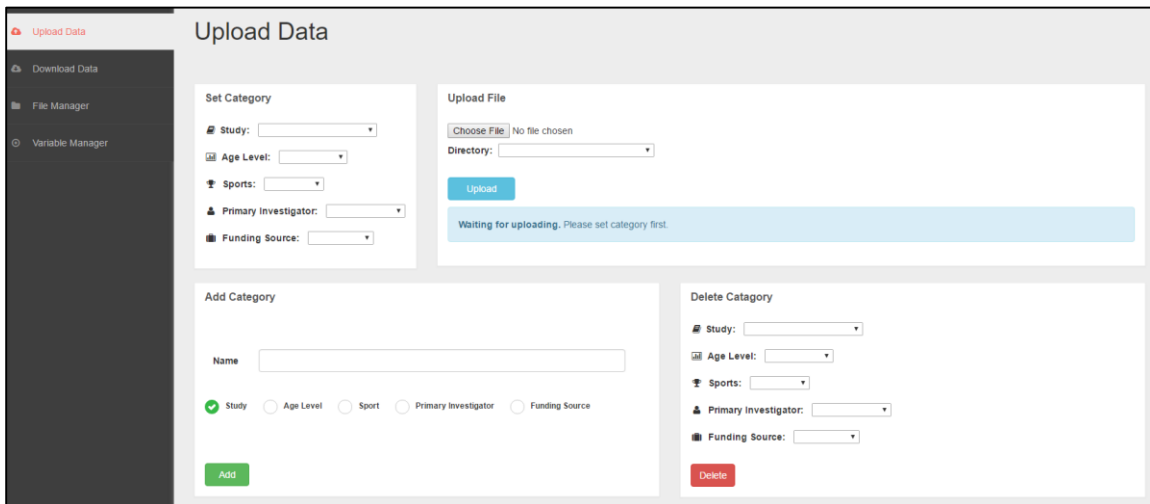


Figure 19

Download Data page:

This page has two panels – ‘Variables Tree’ panel and ‘Data Panel’ panel (Figure 20). The ‘Variable Tree’ presents all the attributes group which are created by users. Users are able to select a whole group by checking the checkbox in the front of the group

name, or they can check any attribute in a group. The data in terms of the attributes will be presented in ‘Data Panel’, since users would feel more confident if they are able to preview the data before downloading. Furthermore, the ‘Data Panel’ has provided search function to help users do simple retrieval. Figure 20 is the screenshot of ‘Download Data’ page.

The screenshot shows the 'Download Data' interface. On the left, there is a sidebar with 'File Manager' and 'Variable Manager'. The 'Variables Tree' in the center shows a hierarchy with 'group1' containing 'agelevel', 'PICSS_question_7', 'PICSS_question_13', 'complex_attention_domain_pr', 'verm_delayed_target_rt', and 'vism_selected', and 'group2'. The 'Data Panel' on the right shows a table with 10 records per page. The table has three columns: 'agelevel', 'complex_attention_domain_pr', and 'verm_delayed_target_rt'. The data rows are as follows:

agelevel	complex_attention_domain_pr	verm_delayed_target_rt
College	0	922
College	75	701
College	7	632
College	40	813
College	13	759
College	2	761
College	16	933
College	75	880
College	61	795
College	30	724

Below the table, it says 'Showing 1 to 10 of 1,008 entries' and includes a pagination control with 'Previous', '1', '2', '3', '4', '5', '101', and 'Next'. At the bottom, there is a 'Sheet Name' field, a 'Target Directory' dropdown set to '/download/', and a 'Download' button.

Figure 20

File Manager page:

This page serves for basic file management of a directory. There is only one panel in the page. At the top of the panel, the system shows the current directory where users are browsing and the number of files in the directory. Then a form is used to list all files under the directory. The header contains filename, size of file, and three functions. The first record of the form is a link back to the parent directory. The filename of files are clickable for downloading. Also, the icons of those three functions are clickable for each

relevant function. Figure 21 is a screenshot of the ‘File Manager’ page

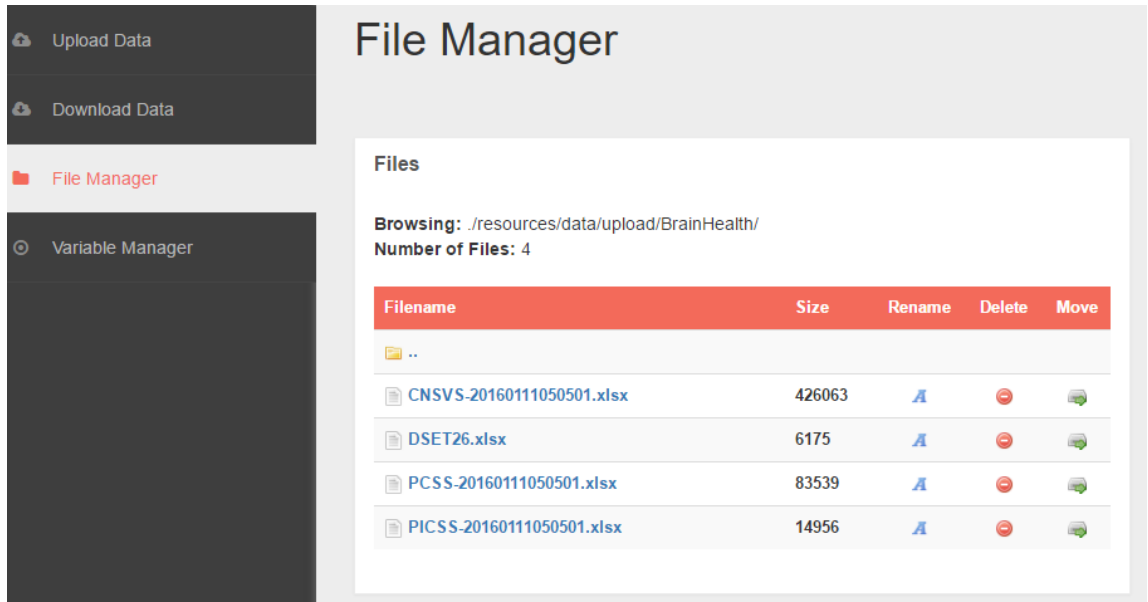


Figure 21

Variables Manager page:

This page serves for users to manage attribute groups. So there is one panel containing the variables tree. The reason why I designed a tree structure for variables management is because of the requirement of users. They like to choose a subset of a whole dataset and download this subset repeatedly in the future, so it would be better for them to create groups of attributes. The group and nodes are like a two level simple tree. Furthermore, it is convenient to manage nodes directly on the tree.

The main operation is creating a new group. When users click the ‘Create Group’ button, some other components like selecting studies and attributes will show under the button, which is described in the use case 1 of variable subsystem. Figure 22 is a screenshot of ‘Variables Manage’ page.

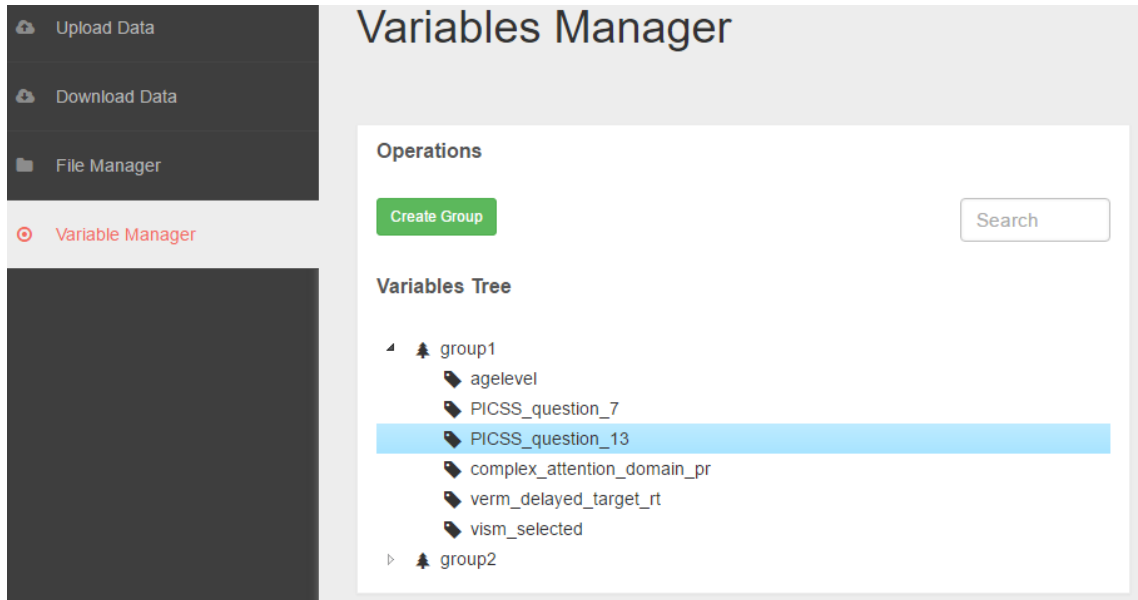


Figure 22

4. Usability Testing

“Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. Usability testing focuses on measuring a human-made product's capacity to meet its intended purpose.” (Usability testing, 2017, section 1)

Usability testing and requirement analysis are two important processes of interacting with users in system lifecycle. Requirement analysis makes sure that a system is able to satisfy users' requirements. In other words, it makes system design stay in the right direction.

Usability testing evaluates how well a system satisfies users' requirements and how easily users can accomplish their goals with the system. With feedback from usability testing, a developer could fix some bugs in the system and improve its design or implementation.

Therefore, I designed a usability test plan to evaluate the database management system. The evaluation plan is mainly based on observing participants doing tasks and gathering feedback from them after using the system. In this section, I will present tasks which could be used in observation and questionnaires which could be used to get feedback after each task.

For the tasks, I set up a scenario in which a user called Tracy would like to do several operations when using the system. And I split her series of operations into five tasks. So

participants are suggested to act as Tracy and complete each task. After each task they would be provided a questionnaire which asks about their experience about using the system.

In conclusion, the main purpose of this usability testing is to evaluate four quality components:

- **Learnability:** How easy is it for users to accomplish basic tasks the first time they encounter the design?
- **Efficiency:** Once users have learned the design, how quickly can they perform tasks?
- **Errors:** How many errors do users make, how severe are these errors, and how easily can they recover from the errors?
- **Satisfaction:** How satisfied are users after using the design?

4.1 Tasks

Background: Tracy has a new research study and has collected two sets of data. The data is stored in two Excel spreadsheets. Both of them are in the same and valid format ready to be uploaded. All the records are uniquely identified by SID. And these two datasets have some common and some different attributes.

Now, Tracy needs to upload the two datasets to the system and do some managing jobs. Imagine you are Tracy, please complete a few tasks.

The two spreadsheets are: dataA.xlsx and dataB.xlsx

Task1:

Please add this new study and its attributes to categories. The information of the study is:

- Study name: sport injury
- Age level: 18 – 25
- Sport: football
- Primary investigator: Tracy
- Funding source: Max

Task2:

Given the two Excel spreadsheets, please upload them to the directory of the target study.

Task3:

For a special purpose, Tracy needs to download a subset of the data that should include only part of the variables of this study. So that this set of variables can be used

again in the future, please create a variable group of these combined svariables.

The information for the group is:

- Group name: test group
- Variables: SID, sport, protocol, administrator, psychomotor_speed_domain_ss, complex_attention_domain_raw

Task4:

Now, please download the data based on the variable group. The Target directory is the directory for current study.

The information for the downloaded Excel spreadsheet is:

- Name: test part

Task5:

For some reason, you may want to delete dataA.xlsx from the server and rename dataB.xlsx to uniqueFile.xlsx. Please complete this task in the File Manager.

4.2 Questionnaire

A questionnaire would be given after each task. The questionnaire uses 5-point Likert-type rating scale ranging from strongly disagree to strongly agree. There are

fourteen questions in total, including twelve rating questions and two open questions.

(Lund, 2001)

Rating questions:

1. I think it was easy to complete this task.
2. I think most people would learn to complete this task quickly.
3. I felt very confident in completing this task.
4. I needed to learn a lot before I could complete this task.
5. I think the error message was useful.
6. I think the organization of components was clear.
7. I think input prompts were clear.
8. I think I can use it successfully every time.
9. It makes the task I want to complete easier.
10. It works the way I want it to work.
11. I think the task was completed efficiently.
12. The support information (instruction, messages) helped me a lot for completing the task.

Open questions:

1. Did you encounter any problems or difficulties with the system? If so, please describe.
2. Do you have any suggestions about how to improve the system? Please describe.

5. Conclusion

This project developed a database management system for Gfeller data center of UNC to manage research data. The system allows users to merge data according to studies and download part of the data for further research. There are four main functions which are uploading Excel spreadsheets, downloading data stored in Excel spreadsheets, managing attributes group for data downloading, and managing uploaded files in server.

The project has been through requirements analysis, use case analysis, and system implementation. The requirements gathering phase uses iterative interviews. The use case analysis splits the system into four subsystems and analyzes their use cases. Then based on each use case, the project develops related functions and combines functions into several modules. Finally, those modules are can be viewed as components of pages and allocated to several web pages.

The system is a web database system. The user interfaces are developed using HTML and decorated by CSS. The logical implementation is realized by JavaScript and takes advantage of open source libraries and plug-ins. PHP is used as the server-side back-end scripting language. Finally, the system uses MySQL database and runs on a Linux operating system.

The main future work for this system could be functionality extension. As the research of the users goes on, they may need other advanced functionalities. For instance, the current system allow users to download data based on attributes group, and the attributes group combined by common attributes of several studies. In this dimension, users can download all the records of those attributes. However, in the future, they may like to download part of the records based on some constraints. In other words, the system could provide another dimension for downloading. Furthermore, future work could also include adding security features like granting different authorizations to different user groups.

References

Delone, W. H., & McLean, E. R. (2003). The DeLone and McLean model of information systems success: a ten-year update. *Journal of management information systems*, 19(4), 9-30.

Holzinger, A. (2005). Usability engineering methods for software developers. *Communications of the ACM*, 48(1), 71-74.

Hamilton, S., & Chervany, N. L. (1981). Evaluating information system effectiveness- Part I: Comparing evaluation approaches. *MIS quarterly*, 5(3), 55-69.

Lund, A.M. (2001) *Measuring Usability with the USE Questionnaire*. STC Usability SIG Newsletter, 8:2.

McHugh, J., Abiteboul, S., Goldman, R., Quass, D., & Widom, J. (1997). Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3), 54-66.

Mifsud, J. (2013). *Requirements Gathering: A Step By Step Approach For A Better User Experience (Part 1)*. Retrieved February 10, 2017, from <http://usabilitygeek.com/requirements-gathering-user-experience-pt1/>

Usability testing. (n.d.). In *Wikipedia*. Retrieved February 10, 2017, from https://en.wikipedia.org/wiki/Usability_testing

Rouse, M. (2015). *database management system (DBMS)*. Retrieved from <http://searchsqlserver.techtarget.com/definition/database-management-system>

Rouse, M. (2007). *requirements analysis (requirements engineering)*. Retrieved from <http://searchsoftwarequality.techtarget.com/definition/requirements-analysis>

Vatrapu, R., Suthers, D., & Medina, R. (2008). Usability, sociability, and learnability: A CSCLE design evaluation framework. *International Conference on Computers in Education (ICCE 2008)*, (CD-ROM).

Young, R. R. (2002). Recommended requirements gathering practices. *CROSSTALK The*

Journal of Defense Software Engineering, 15(4), 9-12.