Requirements
Final Version

PHP Group Ware Project

ICS 125
Summer 2002 – 10-Week Session
Professor Hadar Ziv

Team 10 – Project X
http://tina.alinaghian.com/ics125/index.html
Tina Alinaghian (57291129)
Patrick Walsh (91105649)
Fang Ming Lo (52013421)
Austin Lee (50792270)
Siu Leung (51390400)

TABLE OF CONTENTS

SECTION 1: INTRODUCTION 5

PROJECT OVERVIEW 5

SECTION 2: PROJECT PLAN 7

SCHEDULE 7

WEEK 4, JULY 15TH - 21ST 7

WEEK 5, JULY 22ND – 28TH 7

WEEK 6, JULY 29TH – AUGUST 4TH 7

WEEK 7, AUGUST 5TH – 11TH 7

WEEKS 8,9, AUGUST 12TH – 25TH 7

WEEK 10, AUGUST 26TH – SEPTEMBER 1ST 7

PROJECT RISKS AND CHALLENGES 8

A. KNOWLEDGE 8

B. CUSTOMER CONSTRAINTS 8

C. PERSONNEL CONSTRAINTS 8

D. BETA PLATFORM 9

E. LACK OF DOCUMENTATION 9

F. TIME CONSTRAINTS 9

PROJECT RESOURCES 10

TECHNICAL AND PERSONAL TEAM RESOURCES: 10

SOFTWARE AND HARDWARE RESOURCES: 11

STAFF ORGANIZATION 12

TEAM MEMBERS: 12

SCHEDULE OF LEADERSHIP RESOURCES: 12

MEMBER RESPONSIBILITIES: 12

TRACKING AND CONTROL MECHANISMS 13

LIFECYCLE CONSIDERATIONS 14

SECTION 3: REQUIREMENTS 15
ADMINISTRATIVE FUNCTIONS 15
ADMINISTRATIVE PAGES: 15
CATEGORY MANAGEMENT: 15
MANAGEMENT OF SITE LOOK AND FEEL (TEMPLATES): 15
MANAGEMENT OF SITE-WIDE CONTENT: 16
CONTRIBUTOR FUNCTIONS 17
CONTRIBUTOR AUTHENTICATION: 17
ADDITION OF PAGES IN EXISTING CATEGORIES: 17
MODIFICATION OF PAGES IN EXISTING CATEGORIES: 17
DELETION OF PAGES FROM EXISTING CATEGORIES: 17
PAGE GENERATION ENGINE FUNCTIONS 18
GENERATE DYNAMIC PAGES 18
PERMISSIONS CHECKING: 18
PHPNUKE BLOCK COMPATIBILITY: 18
ACTORS 19
SYSTEM REQUIREMENTS AND ENVIRONMENT CHARACTERISTICS 21
THE SERVER 21
THE CLIENTS 21
SECTION 4: USE CASE SCENARIOS 22
ADMINISTRATIVE USE CASES 22
ADMINISTRATIVE USE CASE DIAGRAM: 22
ADD CATEGORY: 23
EDIT CATEGORY: 25
EDIT SITE CONTENT: 27
EDIT BLOCKS: 29
CONTRIBUTOR USE CASES: 32

Contributor Use Case Diagram: 32
Manage Page: 33
Edit Page: 35
Page Generation Use Cases: 38
Page Generation Use Case Diagram: 38
Generate Page: 38
SECTION 5: TEST CASE PLAN 41
Unit Testing 41
Integration Testing 45
System Testing 49
SECTION 6: USER INTERFACE DIAGRAMS 52
Administrative UI Diagrams 52
Figure 1: Administrative Menu UI Diagram: 52
Figure 2: Category Manager UI Diagram 1: 52
Figure 3: Add/Edit Category UI Diagram 2: 53
Figure 4: Edit Header / Footer Content UI Diagram: 54
Contributor UI Diagrams 55
Figure 5: Page Manager UI Diagram: 55
Figure 6: Add/Edit Page UI Diagram: 56
Page Generation UI Diagrams 57
Figure 7: Generated Page: Site Contents UI Diagram: 57
Figure 8: Generated Page: Site Index UI Diagram: 58
Figure 9: Generated Page: Category Table of Contents UI Diagram: 59
Figure 10: Generated Page UI Diagram: 59

SECTION 1: INTRODUCTION

Project Overview

Team 10 will build a "Collaborative Web Content Management" application based on the phpGroupWare platform. When complete, the application will be installed on the phpGroupWare web site to demonstrate its capabilities and test it in a real-world environment.

The system will allow a community of individuals to contribute to a web site by granting individual members or groups within the community permission to view and edit specific sections of the web site. In general, the viewers of the web site function in one of three roles: Web Site Administrators, Web Site Contributors, and Anonymous Web Site Viewers. For short, we will refer to these roles as Administrators, Contributors, and Viewers. Administrators are the all-powerful beings that manage the entire site. Contributors are authenticated users with permissions to manage specific sections of the web site. Viewers are unauthenticated viewers of the site. Sections of the web site, also called Categories, may be viewable by anonymous viewers or by specified contributors. Administrators may always view and edit all categories on the web site.

The Administrator's functions are:

1. Manage categories and the view/edit permissions.
2. Manage the overall look and feel of the site.

The Contributor's functions are:

1. Addition/ deletion of pages in an existing category.
2. Modification of the content in existing pages.

The Viewer's functions are:

1. Browse the web site.

The entire web site will be dynamically generated from a database. Hence, there will be a Page Generation Engine that generates web pages based on the Administrators' specified look and feel and the Contributors' content. The Page Generation Engine will evaluate whether the viewer has permissions to view a particular page before generating it.

The entire application will be written using the phpGroupWare backend. This means that the list of users and groups and administrators will be taken from an existing phpGroupWare installation. phpGroupWare will handle the management of these users and groups. phpGroupWare will also be responsible for storing and retrieving all data that needs to be saved. This includes all of the content, the list of categories and permissions, etc.

Finally, the completed application will have some compatibility with an already established web application of similar functionality. PHPNuke is a program used to generate dynamic web sites. It was built to allow third parties to create add-ons called “blocks” that can be added to a web page. These PHPNuke blocks are small pockets of information that usually sit on the side of a web page. For example, a block may contain a list of stocks and their current prices in a box on the left side of a page. There is a wide variety of these blocks already available in which this application will support.

SECTION 2: PROJECT PLAN

Schedule

Week 4, July 15th - 21st

7/18: Requirements Iteration #2 and Test Plan Iteration #1 due.

Finish requirements document and begin work on the program design. Investigate technical challenges and determine independent modules.

Week 5, July 22nd – 28th

7/25: Test Plan Iteration #2 and Design Iteration #1 due.

Submit the design document draft and divide up the programming assignments. Each team member should become comfortable with what they're going to have to do, including understanding phpGroupWare and how it works and how we'll be using it.

Week 6, July 29th – August 4th

Nothing due.

Begin coding skeleton structure and determining how, if at all, the design will have to be modified.

Week 7, August 5th – 11th

8/8: Design Iteration #2 and Code Iteration #1 due.

Complete coding the core functions – those functions and classes that will be needed by all of the modules and that, unfinished, would hold up further development, must be finished this week.

Weeks 8,9, August 12th – 25th

Nothing due.

Complete all of the core functionality and begin testing.

Week 10, August 26th – September 1st

8/27: Final code and all deliverables due.

Do full testing, install and test on phpGroupWare.org server, and, if time permits, add gravy features.

SECTION 2: PROJECT PLAN (CONT.)

Project Risks and Challenges

A. Knowledge

i. 3 of the 5 team members have either limited experience or no experience at all developing web applications and particularly programming in PHP. This poses a large risk as the learning curve associated in getting up-to-speed enough on PHP and web application development is hard to predict. As such, the time required for certain team members to complete certain coding tasks will be highly variable. With the short time frame of the project, this presents a monstrous risk.

ii. In addition to the basic programming language, 4 of the 5-team members have not had experience using collaborative development tools such as the Concurrent Versioning System (CVS) or Microsoft Visual Source Safe. This will potentially slow development down and create a difficulty in finishing the project in the specified time frame.

iii. 4 of the 5 team members have limited or no experience using Linux machines. Since the web server and development will be taking place on a Linux machine, team members will also have to pick up at least a rudimentary understanding of Linux in order to develop for the project. This again may take up more time than predicted.

iv. 3 of the 5 team members have limited or no experience with SQL and relational databases. Again, this may have a negative effect on the team's ability to get the application finished on time.

B. Customer Constraints

i. The customer, Dan Kuykendall, the leader of the phpGroupWare project and our technical adviser, is currently starting a new job, raising two children including a three-week-old baby, and squeezing us in when he can. This means that he is not as available to us as we would like and, as the project moves forward and we need help and advice on technical matters, this may become a problem.

C. Personnel Constraints

i. All of the team members are either working part-time or full-time jobs or attending other summer school classes in addition to ICS 125. This means that

the time that each team member has to dedicate to the project is finite and potentially more limited than is required to adequately finish the project in the allotted time.

D. Beta Platform

i. phpGroupWare is the platform of choice for this project. It serves almost as an operating system, handling the low level details such as data storage and retrieval and user authentication and permissions. However, phpGroupWare is not a released product and is currently in Beta testing. This project is being built on a Beta Release Candidate, meaning that the code is not entirely stable and far from bug-free. This may not be an issue at all, or we may run into problems with phpGroupWare that will sidetrack us from the project at hand and prevent or delay an adequate completion of the project.

E. Lack of Documentation

i. phpGroupWare and phpNuke both have limited up-to-date documentation for developers. This means that much of the time we will have to reverse engineer these programs and/ or attempt to talk with various developers, who may live half way around the world, to resolve problems or issues.

F. Time Constraints

i. All of the above risks are risks that the project might not be done in time, rather than that it might not be done at all. Given sufficient developer time and attention, this project will certainly be completed. However, this project has a very strict 10-week time line. In fact, the coding portion of the project will last at most 4 weeks – a short amount of time to code, integrate, and test five semi-independently developed portions of code.

These risks are all real and any one of them could be fatal to the project. Only disciplined management, clear and uninterrupted organization, a well-defined design document, and careful attention to the set schedule can overcome these risks. Also, if any one-team member gets stuck somewhere, other team members will have to help in order to keep the project moving forward.

SECTION 2: PROJECT PLAN (CONT.)

Project Resources

Technical and Personal Team Resources:

Team 10 consists of five talented individuals all with their own unique personal and technical strengths. These strengths will be utilized to accomplish our goals in this project. Bellow are the personal and technical strengths of each member on our team:

Patrick Walsh

Technical Strengths: Patrick can be characterized as having a strong collaborative development experience as both a developer and a manager. Patrick also has a strong php experience, strong program design experience, good phpGroupWare experience, and strong Linux experience.

Personal Strengths: Patrick has very strong communication and leadership skills. He is a very helpful individual who is always willing to assist others in their responsibilities and tasks.

Tina Alinaghian

Technical Strengths: Tina has proficient knowledge of web page design and development. She has experience with many database platforms including mySQL and msSQL. She also has experience with HTML, PHP, ASP, Java and C++.

Personal Strengths: Tina is a hard worker and a very quick learner.

Siu Leung

Technical Strengths: Siu is an adequate programmer with knowledge in Java and C++. Although he is not familiar with php, he is excited to pick up a new programming language.

Personal Strengths: Siu is able to socialize and work well with others both inside and outside of a group project-type environment. He is an organized and responsible individual who finishes assigned tasks on time.

Austin Lee

Technical Strengths: Austin is a programmer with knowledge in Java, Visual Basic, C++, and other languages. Although he is not familiar with php, he can quickly pick up new programming languages.

Personal Strengths: Austin had been a researching intern in a research center in Korea (Samsung Advanced Institute of Technology). His solid mathematical background could help others with logical problems.

Fang Ming Lo

Technical Strengths: Ming is knowledgeable in Java and C++. He has also used HTML in the past, and although he may be a bit rusty with it, he is fairly confident he can pick it up fast. He is currently trying to learn .NET and is interested in learning the new and useful languages currently out.

Personal Strengths: Ming is very organized and can work well in a group. He takes his assigned jobs very seriously. He is also a very easygoing person and very easy to get along with.

Software and Hardware Resources:

We will also be utilizing the following software on our web and development server to complete the project:

1. Mandrake Linux 8.2
2. Apache-AdvancedExtranetServer 1.3.23
3. PHP 4.1.2
4. phpGroupWare 0.9.14RC3
5. OpenSSH 3.4p1
6. CVS 1.11.1p1
7. MySQL 3.23.47

SECTION 2: PROJECT PLAN (CONT.)

Staff Organization

Team Members:

Patrick Walsh	91105649
Siu Leung	51390400
Tina Alinaghian	57291129
Fang Ming Lo	52013421
Austin Lee	50792270

Schedule of Leadership Resources:

The leadership rotation schedule of Team 10 is based on the schedule of the class. Since there are five deliverables, each team member is responsible for a deliverable and takes leadership of the group for that deliverable. The following is the leadership rotation schedule for each deliverable:

Deliverable	Due date	Leader
Requirements Iteration 1	July 11	Patrick Walsh
Requirements Iteration 2 / Test Plan iteration 1	July 19	Siu Leung
Test Plan Iteration2 / Design Iteration1	July 26	Tina Alinaghian
Design Iteration2 / Code –Iteration1	August 8	Fang Ming Lo
Code-Iteration2 (Final) + / All final deliverables	August 27	Austin Lee

Member Responsibilities:

Each member of our development group is responsible for the development of individual component of the project. The individual task includes the requirement specification, testing plan, design, and implementation of their component:

Name	Responsibility
Patrick Walsh	PHPNuke blocks generation
Siu Leung	Page (including Template) generation
Tina Alinaghian	Category Management
Fang Ming Lo	Site look & feel management
Austin Lee	Contributor functions

SECTION 2: PROJECT PLAN (CONT.)

Tracking and Control Mechanisms

We have laid out a specific schedule that may become more detailed when the design document is created. Our progress will be closely measured against the schedule to be sure that the project is not falling behind. If the project begins to fall behind schedule, for whatever reason, steps will have to be taken to jump-start the project. Those steps will be determined by the current project manager, based on the specific problems causing the delay.

The code will be kept in a CVS repository. CVS, the Concurrent Versioning System, tracks revisions in documents. Because most of the team is unfamiliar with the program, their files will be automatically "checked in" to the versioning system on a regular basis. In this way, a central depository will have a fairly up-to-date picture of the state of development. Also, if something stops working suddenly, developers can see what changes have been made since a certain date and undo or fix those changes.

For the Word documents being created, including this document and the design document, Word's own built in versioning will be used. Each person's individual edits and the history of edits in the document are being saved. Each deliverable version is being individually saved as well. Pieces of the document are being worked on and the current project manager will add those pieces to the master document by cutting and pasting.

SECTION 2: PROJECT PLAN (CONT.)

Lifecycle Considerations

After much careful consideration and group feedback from each member, the lifecycle best preferred for such a project at hand is the spiral model. Although we do not have sufficient time to build a rapid prototype in this class, the spiral model is still the perfect model for such a task. Before each phase, a risk analysis is prepared. A verification phase proceeds each subsequent phase. The most important risk factor to take into consideration deals with the programming language being used to complete the project. Since only 2 members of our group are familiar with php, they will need to do a bit of reading on their own time to be prepared for the implementation phase of the project.

SECTION 3: REQUIREMENTS

Administrative Functions

Administrative Pages:

The administrative pages will all be housed in the phpGroupWare environment. An administrator must successfully log in to phpGroupWare before being able to access the administrative functions listed below.

Category Management:

Every category represents a section of the site. The administrator may define as many categories as he/she wishes. Each category will have an ACL (Access Control List) associated with it that defines the users and groups that have read and edit permissions to the category. The administrator can manage the categories of the site by

- Adding categories and category descriptions to the site
- Removing categories from the site
- Modifying existing category descriptions
- Adding a user or group to the list of people with read permissions for a particular category
- Adding a user or group to the list of people with write permissions for a particular category
- Removing a user or group from the list of people with read permissions
- Removing a user or group from the list of people with write permissions

When setting "read and write" permissions on existing categories, administrators have the option of choosing whether the category can be read and/or written to by all groups (including anonymous viewers who have not been authenticated) or whether the category can only be read and/or written to by selective groups. These groups are already created and established in the phpGroupWare API.

Management of site look and feel (templates):

The look and feel of the administrative areas will be controlled by phpGroupWare, but the Page Generation Engine will control the look and feel of the generated pages. This involves two parts. First, the administrator will choose a template

from a list of installed PHPNuke templates. This will determine the color scheme, fonts used, spacing, and so forth. Incorporation of PHPNuke Templates will not be available till version 2.

Management of site-wide content:

The administrator will create site wide content, such as navigation links at the top of the site and copyright notices at the bottom. Every web page generated by the "Page Generation Engine" (see the section below for an explanation of the Page Generation Engine) can be broken into five parts. These are the header, footer, left menu bar, right menu bar, and main content area. The administrator controls all of these areas except for the main content area. (The Contributors control the main content area, though the template that the administrator chooses will be applied to this area to keep a uniform look and feel.) If he/she chooses, the administrator may choose not to use one of the page parts so that, for instance, a site may have a header and footer, but no side menu bars.

In addition to putting static content into each of these page parts, the administrator may choose to display selected "PHPNuke blocks" (explained below) in either of the sidebars in an order specified by the administrator. These blocks typically contain dynamic content, such as the number of users currently logged in, breaking news headlines, or user polls. PHPNuke Blocks will not be incorporated until verion 2.

SECTION 3: REQUIREMENTS (CONT.)

Contributor Functions

Contributor Authentication:

Contributors of the site can add, modify, and delete pages in existing categories that they have access to. However, before they can contribute to a category, they must first log in and be authenticated. Once they are authenticated, they will be told which categories they are able to contribute to along with which categories they are able to read material from.

Addition of Pages in Existing Categories:

Contributors of the site can add pages to categories that they have write access to. When adding a page to an existing category on the site, they will be asked to specify a title, sub-title, short description of the page, and the main content of the page. They must however first be logged in and be given authorization to be able to write to a category.

Modification of Pages in Existing Categories:

Site contributors can modify existing pages in categories that they have access to. They must however first be logged in and be given authorization to be able to write to a category.

Deletion of Pages from Existing Categories:

Site contributors can delete pages from categories that they have access to. They must however first be logged in and be given authorization to be able to write to a category.

SECTION 3: REQUIREMENTS (CONT.)

Page Generation Engine Functions

Generate Dynamic Pages

The main function of the Page Generation Engine is to deliver web pages to web site viewers. To do this, the Page Generation Engine will take the site-wide content specified by the administrator and combine it with the page specific content specified by the contributors, format it, apply the PHPNuke template to it, and deliver it to the viewer.

Permissions Checking:

Before generating and delivering a page, the Page Generation Engine will ask the phpGroupWare backend whether the current user (or, if no user is logged in, the anonymous user) has permissions to view the requested category. If the permission is granted, the page is generated and delivered. If the permission is not granted, an error is shown.

PHPNuke Block Compatibility:

PHPNuke is a widely used dynamic website generator. It manages news articles and user comments on each article, among other things. The community of PHPNuke users has embraced a feature of PHPNuke called blocks. Blocks are a mechanism for third party developers to add added functionality to a site in the form of a box of information or a form on one of the sidebars of a page. PHPNuke has its own set of functions and interfaces to make these blocks work. This application will duplicate the necessary functions and routines to allow those blocks to work (including, if necessary, ported routines for accessing session and login information). So when a page is being generated, in addition to reading in the static content for the sidebars, each block that the administrator has specified will be executed and displayed within a box.

SECTION 3: REQUIREMENTS (CONT.)

Actors

The actors are as follows:

1. **Site Administrator**

   The Site Administrator is a logged-in user with an existing phpGroupWare administrator account on a particular system. As such, that person may install applications, change user permissions and group memberships and so forth for phpGroupWare's many applications. Anyone who is an administrator of the given phpGroupWare system, is also, by extension, an administrator of the Web Content Management application.

   The Site Administrator's function is to dictate which users or groups have permission to edit portions of the web site. Also, the Site Administrator determines the site-wide look and feel of the web site, including navigational bars, menus, copyright notices, and other elements that do not change from page to page.

2. **Site Contributor**

   A Site Contributor is a logged-in user who has been granted the permission to edit one or more categories on a site. The Site Administrator granted this permission. This person can go to the category on the web site that they have permission to maintain and then add pages or edit or delete existing pages within that category.

3. **Site Viewer**

   The Site Viewer is either a logged-in user who does not have permission to edit any categories, or a user who has not logged in, but is an anonymous viewer. This person may view all of the web pages in those sections that are marked as publicly readable.

4. **phpGroupWare ACL**

   The phpGroupWare ACL (Access Control List) is an external system to the Web Content Manager. It has a number of functions and capabilities intended to manage users, groups, and permissions. An application may define a

number of objects and then grant users or groups various types of permissions for those objects. Then, at a later point, the application can query the phpGroupWare ACL to see if the current user has a particular permission for a given object.

5. phpGroupWare DB

The phpGroupWare DB (Database) is an external system that applications use to manage the storage, retrieval, cataloging and management of data. PhpGroupWare provides a set of routines that allow the details of the backend database to remain hidden from the application. In this way, errors are handled smoothly and many different database systems and configurations may be used.

SECTION 3: REQUIREMENTS (CONT.)

System Requirements and Environment Characteristics

The Server

First and foremost the Web Content Manager will require phpGroupWare in order to run. phpGroupWare is the equivalent of the Operating System. It will be used to manage the storage and retrieval of data, the display and delivery of content, the authentication and identification of users, and all other back-end activities. phpGroupWare was designed to run on web servers that can interpret PHP files. It works with both PHP3 and PHP4. It can use a number of databases for data storage, including MySQL, Microsoft SQL Server, Oracle, and PostgreSQL. It will also run under Apache on Microsoft NT, Internet Information Server on Microsoft NT, and Apache on several flavors of Unix. It may even run under other Operating Systems. It can draw its list of users from a database, from an LDAP directory, or from a variety of other places including NIS.

Despite phpGroupWare's tremendous cross-platform support, we are developing our Web Content Manager specifically for use with PHP4 running under Apache on a Red Hat Linux system and using a MySQL database for storing both regular data and the user list. When the Web Content Manager is complete, it will be installed under just such a system on the phpGroupWare.org web server. It will also be tested under Mandrake Linux.

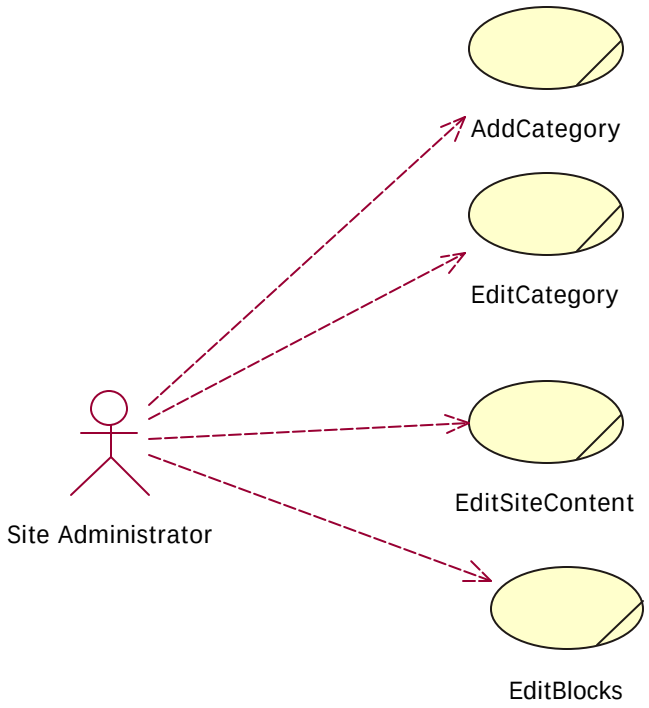
The Clients

While we are developing for one particular server configuration, we are developing for multiple client configurations. If our product is successful, it will be usable by Macintosh users, Windows users, and Linux users. It will be viewable with Opera, Netscape, Mozilla, Konqueror, and Internet Explorer. However, we do not have the resources to test all of these configurations. So we will be sure that the site is viewable and usable under Konqueror on Linux, and Opera and Internet Explorer under Windows. For this cross-platform compatibility, client-side scripting will be kept to an absolute minimum, if it is used at all.

SECTION 4: USE CASE SCENARIOS

Administrative Use Cases

Administrative Use Case Diagram:



Add Category:

Use Case Item	Description
Name	Add new category.
Status	Approved - Completed
Author	Tina Alinaghian
Purpose	Allows an administrator to add a new category to the site along with the categories permissions.
Overview	The administrator can add new categories to the site to allow contributors of the site to add pages to the site. When the administrator adds a new category he will also set permissions for the category. These permissions state which users and/or groups have read and/or write access to the categories. When adding a new category, the administrator will be asked to give a category name, category description, group access permissions, individual access permissions (if any), and whether or not anonymous viewers have read/write access to the category. (See User Interface Figure 3) The newly added category should be updated to the phpGroupWare data backend. The title of the added category should be included in the list of the categories on the Category Manager Page (See User Interface Figure 2) and on the Generated Site Index and Site Contents Pages (See User Interface Figure 7 & 8). If the administrator gives incorrect inputs, the page will be reloaded and an error message should be shown to the administrator as a notice.
Priority	Since this functionality allows the administrators to create categories in which contributors can contribute to, this functionality has high priority.
Actors: **Primary** **Secondary**	 Administrators PhpGroupWare ACL and PhpGroupWare data backend
Pre-conditions	1. Administrative Accessibility ▪ Before adding a new category, the administrator

Use Case Item	Description
	must be authenticated by the phpGroupWare ACL.
Post-Conditions:	
Success	The newly added category should be added to the phpGroupWare data backend. The title of the added category should be included in the list of the categories on the Category Manager Page (See User Interface Figure 2) and on the Generated Site Index and Site Contents Pages (See User Interface Figure 7 & 8).
Failure	1. Duplicate Category Name ▪ A new category must have a unique name; a category by the same name can not already exists. 2. Missing Fields ▪ All basic setting fields (the category name and the category description must be filled out).
Related Use Cases	1. Edit Category
Cross-References and Notes	Sample user interfaces 1. Category Manager UI (UI Interface Diagram 2). 2. Add/Edit Category UI (UI Interface Diagram 3). 3. Generated Page Site Contents UI (UI Diagram 7). 4. Generated Page Site Index UI (UI Diagram 8).
Basic Course	1. Go to Manage Categories from the Administrative Menu (UI Diagram 1). 2. Click on the “Add Catagory” Button bellow the “Manage Catagories” section. 3. Fill out the “Basic Settings”: Category Name and Category Description. 4. Set group permissions (if any) by checking read and/or write boxes next to existing group names. 5. Set user permissions (if any) by checking read and/or write boxes next to existing user names. 6. Click “Save” button to add the new category to the site.
Successful Alternative Courses:	3a. If no category name or description is given, will come back and prompt the administrator that he needs to fill out those fields.

Use Case Item	Description
	3b. If category name already exists, will come back with an error, prompting for a new category name. 7b. Instead of pressing save, the administrator can press "reset" to undo changes made to the category.
Unsuccessful Alternative Courses:	N/A
Open Issues	N/A

Edit Category:

Use Case Item	Description
Name	Edit Category
Status	Approved - Completed
Author	Tina Alinaghian
Purpose	Allows an administrator to edit an existing categories basic settings (i.e., its name and description) and permissions (i.e., who has read/write access to the category).
Overview	Many times, the name or description of a category will need to be changed. This use case allows the administrator to change the name and/or description of an existing category. Also, the access permissions to a certain category may need to be changed. New groups may be formed that need read/write access to a category, or new members will join whom will need read/write access to a category. This use case allows the administrator to change the permissions of a category. If the administrator does not fill in all the necessary info, or renames the category to a name that already exists in the phpGroupWare database, or does not set any permissions when editing the category permissions, an error will be returned.
Priority	This functionality has a very high priority as it ultimately

Use Case Item	Description
	deals with site security since the administrator may want to remove someones permissions from a category.
Actors:	
Primary	Administrators
Secondary	PhpGroupWare ACL and PhpGroupWare data backend
Pre-conditions	1. Administrative Accessibility ▪ Before editing the settings of an existing category, the administrator must be authenticated by the phpGroupWare ACL.
Post-Conditions:	
Success	1. The category setting changes should modify the category description in the phpGroupWare data backend. The modifications to the category should be seen in the list of categories on the Category Manager Page (See User Interface Figure 2) and on the Generated Site Index and Site Contents Pages (See User Interface Figure 7 & 8). 2. The category permission changes should modify the category permissions in the phpGroupWare data backend. Groups/individuals who now have access to the category should see the category name in the list of categories on the Generated Site Index and Site Contents Pages (See User Interface Figure 7 & 8), and those groups/individuals who lost permissions to the category should no longer see the category name in the list of categories.
Failure	1. Duplicate Category Name ▪ A new category must have a unique name; a category by the same name can not already exists. 2. Missing Fields ▪ All basic setting fields (the category name and the category description must be filled out).
Related Use Cases	1. Add Category
Cross-References	Sample user interfaces

Use Case Item	Description
and Notes	1. Category Manager UI (UI Interface Diagram 2). 2. Add/Edit Category UI (UI Interface Diagram 3). 3. Generated Page Site Contents UI (UI Diagram 7). 4. Generated Page Site Index UI (UI Diagram 8).
Basic Course	1. Go to Manage Categories from the Administrative Menu (UI Diagram 1). 2. Click on the "Edit" Button next to the Category name you wish to edit. 3. Fill out the "Basic Settings": Category Name and Category Description. 4. Change group permissions (if any) by checking read and/or write boxes next to existing group names. 5. Change user permissions (if any) by checking read and/or write boxes next to existing group names. 6. Click "Save" button to save the changes to the category to the phpGroupWare database.
Successful Alternative Courses:	3a. If no category name or description is given, will come back and warn the administrator that he needs to fill out those fields. 3b. If category name already exists, will come back with an error, asking for a new category name. 7b. Instead of pressing save, the administrator can press "reset" to undo changes made to the category.
Unsuccessful Alternative Courses:	N/A
Open Issues	N/A

Edit Site Content:

Use Case Item	Description
Name	Edit Site Content
Status	Approved – Completion of Header/Footer, Template selection implementation due out in version 2.0

Use Case Item	Description
Author	Fang Ming Lo
Purpose	It is used to manage the overall site look and format the header and/or footer of the site.
Overview	SiteFormatManager, the administrator uses this to control the over all look/theme of the site. When the administrator gets his permission checked, the administrator will be redirected to a page where there he can modify the header/footer and/or change the theme of the site. Information on the current header/footer and theme will be retrieved automatically from the database and displayed on the page for the administrator to view and modify. The administrator doesn't have to modify both; he can choose to modify either one or both. After the changes have been made, the administrator can click on the "save" button on the bottom of the page to save the page into the database and administrator will be redirected back to the Home page. The administrator doesn't have to save the changes at the end, he can just quit the Manager by clicking the "Quit" button to go back to the Home page.
Priority	High
Actors: **Primary** **Secondary**	 Administrators PhpGroupWare ACL and PhpGroupWare data backend
Pre-conditions **Post-Conditions:** **Success**	1. Before being able to modify the header/footer contents, the administrator should log in to be authenticated from phpGroupWare ACL. 2. Previous contents should be displayed in the text fields for both the header and footer. If this is the first time use then the text fields should be blank. 3. Theme list must be stored in the database. The Header/Footer will display exactly what the administrator intends to show. The overall look of the site should be as the same as the

Use Case Item	Description
Failure	administrator chose. The Header/Footer do not display the correct contents or the format is incorrect. The theme of the site isn't the same as the one chose by the administrator.
Related Use Cases	N/A
Cross-References and Notes	Sample user interfaces Edit Header/Footer User Interface(Figure 4 of UI Diagrams)
Basic Course	1. User permission check from PhpGW ACL. 2. Request information from PhpGW database. 3. Administrator changes the header/footer and/or site theme. 4. Administrator saves the changes.
Successful Alternative Courses:	1. User Permission check from PhpGW ACL. 2. Request information from PhpGW database. 3. Requested data are loaded into respected fields. 4. Administrator clicks on the cancel button without save.
Unsuccessful Alternative Courses:	1. User permission check from PhpGW ACL. 2. Request information from PhpGW database. 3. PhpGW database can't be loaded.
Open Issues	The database might fail to operate.

Edit Blocks:

Use Case Item	Description
Name	Edit Blocks
Status	Coming version 2.0
Author	Fang Ming Lo
Purpose	The administrator selects and organizes the block for the sidebars here.

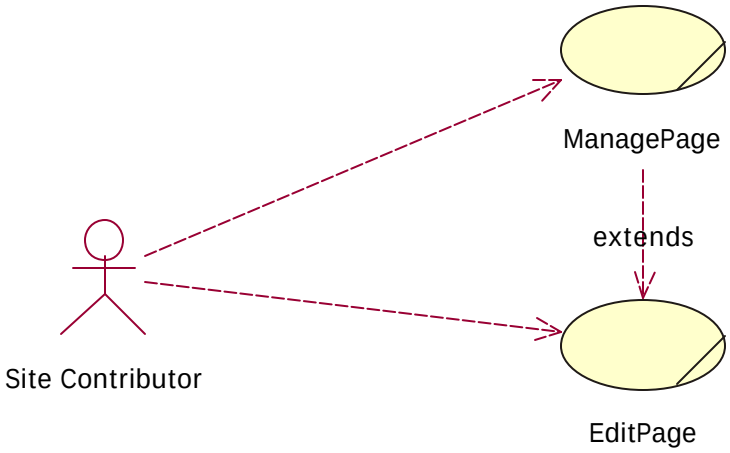
Use Case Item	Description
Overview	Blocks selection, it allows the administrator to modify the sidebars. Changing the placement of the blocks modifies the bars. First the user has to login to verify he/she is an administrator. ACL will do the user verification. When the user is verified as the administrator, the phpGroupWare database will be accessed and the blocks will be retrieved. Administrator then can select which blocks to activate and where to put them. After everything is done, the administrator can save the changes and generate a final page to check for any errors.
Priority	Very Low
Actors: **Primary** **Secondary**	 Administrator PhpGroupWare ACL and PhpGroupWare data backend
Pre-conditions **Post-conditions:** **Success** **Failure**	The contents for the blocks are set, and total number of blocks are determined. The blocks on the side bars are placed in the proper place as the administrator defined. The blocks on the sidebars are not in the order as the administrator defined or the contents of the blocks are incorrect.
Related Use Cases	N/A
Cross-References and Notes	Sample user interfaces Manage Category User Interface (Figure 6 of User Interface Diagrams)
Basic Course	1. Request. 2. Permission check. 3. Request blocks information from phpGroupWare database. 4. Administrator modifies the blocks. 5. Save the modification. 6. Contents change saved confirmation window. 7. Return to the front page.

Use Case Item	Description
Successful Alternative Courses:	1. Request. 2. Permission check. 3. Permission not granted. 4. Return to front page.
Unsuccessful Alternative Courses:	1. Request. 2. Permission check. 3. Request blocks information from phpGroupWare database. 4. Database failure/can't be found
Open Issues	Blocks information can't be retrieved from the database.

SECTION4: USE CASE SCENARIOS(CONT.)

Contributor Use Cases:

Contributor Use Case Diagram:



Manage Page:

Use Case Item	Description
Name	Manage Page
Status	Approved - Completed
Author	Austin Lee
Purpose	To allow the contributors of the site to manage the categories that they have write access to.
Overview	The site allows a community of people to contribute to a web site by granting individual members within the community permission. Contributors are authenticated users with permission to manage certain categories of the site. This "Page Managing" use case gives two functionalities to contributors with authentication from phpGroupWare ACL: Add/Remove pages in the available categories. After logging in to the site, the contributors can see the available categories with the list of the pages in the categories. (See User Interface of Category Managing page.) After selecting adding a new page to an available category, the blank page- managing page will be loaded. (See User Interface of Page Managing page.) The site contributor can provide the information of the page and the page information should be updated by clicking "Save" button. Removing process does not access the page-editing page. Through clicking "Remove" button, the selected page should be removed from the phpGroupWare data backend.
Priority	Since this functionality allows the contributors with authentication to specific categories to manage the content of the categories, this use case has the high priority.
Actors: **Primary** **Secondary**	 Contributors PhpGroupWare ACL and PhpGroupWare data backend
Pre-conditions	Category accessibility

Use Case Item	Description
	▪ Before adding new page to certain category, the contributor should log in to be authenticated. ▪ Before a page in a category, the contributor should log in to be authenticated. Duplicated page name ▪ A new page cannot have duplicated page name, which already exists in the category. Existing page and category. ▪ Authenticated contributor can add a page in existing category.
Post-Conditions: **Success** **Failure**	The add management choice should be bring the page-editing page. Removing management choice should remove the page from the data backend. Error message to the contributors
Related Use Cases	Edit page
Cross-References and Notes	Sample user interfaces 1. Page Manager User Interface (Figure 5 of UI Diagrams) 2. Add/Edit Page User Interface (Figure 6 of UI Diagrams)
Basic Course	Adding new page 1. Choose the specific category to add a page 2. Click “Add new page” button 3. Blank Page- managing page will be loaded. 4. After providing the page information, clicking “Save” button should be update the page to the data backend. Removing a page 1. Choose the page to remove. 2. Click “Remove” button next to the page title. 3. The page will be removed from the PhpGroupWare data backend.
Successful Alternative Courses:	1. Choose the specific category to add a page 2. Click “Add new page” button 3. Blank Page- managing page will be loaded. 4. Provide information of the page.

Use Case Item	Description
	5. Click “Reset” button. 6. Blank Page- managing page will be loaded.
Unsuccessful Alternative Courses:	
Open Issues	1. The management of the large list of pages in certain categories. 2. The contribution of the pages with illegal content, such as adult site.

Edit Page:

Use Case Item	Description
Name	Edit a page
Status	Approved - Completed
Author	Austin Lee
Purpose	To allow the contributors of the site to modify existing pages or provide new information of the pages in categories that they have access to.
Overview	“Edit page” use case allows the authenticated contributors modifying existing pages or providing information of the new pages in categories that they have access to. To edit an existing page in specific categories, the contributors should log in to obtain the authentication from phpGroupWare ACL. After logging in to the site, the contributors can see the available categories with the list of the pages in the categories. (See User Interface of Category Managing page.) Through either selecting “Add new page” or “Edit”, the page- managing page should be viewable. Add mode will generate the page- managing page with no information, and edit mode will generate the page-managing page with the information of the page, including its title, subtitle, short description of the page, and the main content of the page, which are obtained from

Use Case Item	Description
	phpGroupWare data backend. (See User Interface of Page Managing page.) After the modification of the information, the modified information should be updated to phpGroupware data backend, and the updated page should be generated. If the modification fails, an error message regarding modification should be shown to the contributor.
Priority	Since this functionality allows the contributors with authentication to specific categories to modify the content of the categories, this functionality has high priority.
Actors: **Primary** **Secondary**	 Site contributors PhpGroupWare ACL & PhpGroupWare data backend
Pre-conditions	Category accessibility ▪ Before editing pages or adding a new page in certain exiting category, the contributor should log in to be authenticated by phpGroupWare ACL. Existence of pages and categories ▪ The contributor can only edit existing pages in existing categories.
Post-conditions: **Success** **Failure**	 The modified information should be updated to phpGroupware data backend, and the updated page should be generated. If the modification fails, an error message regarding modification should be shown to the contributor.
Related Use Cases	N/A
Cross-References and Notes	Sample user interfaces 1. Page Manager User Interface (Figure 5 of UI Diagrams) 2. Add/Edit Page User Interface (Figure 6 of UI Diagrams)
Basic Course	Edit a page 1. Log in the site. 2. Choose the category management from the menu. 3. Choose the specific category to add a page.

Use Case Item	Description
	4. Click “Edit” button next to the page name. 5. Edit the loaded content of the page. 6. Click “Save” button to contribute the page to the category 7. The modified information should be loaded to the data backend..
Successful Alternative Courses:	1. Log in the site. 2. Choose the category management from the menu. 3. Choose the specific category to add a page. 4. Click “Edit” button next to the page name. 5. Edit the loaded content of the page. 6. Click “Reset” button to contribute the page to the category 7. The original information of the page should be loaded again.
Unsuccessful Alternative Courses:	N/A
Open Issues	1. Wrong modification 2. Available backup for modified page

SECTION4: USE CASE SCENARIOS(CONT.)

Page Generation Use Cases:

Page Generation Use Case Diagram:

Generate Page:

Use Case Item	Description
Name	Generate Page
Status	Aproved - Completed
Author	Siu Leung Patrick Walsh
Purpose	If a site viewer requests a page and has permission to view the requested page, the page generator will take the contributor and administrator specified content, apply a theme to it, and send an html page back to the site viewer.
Overview	The main function of the Page Generation Engine is to deliver web pages to web site viewers. To do this, the Page Generation Engine will take the site-wide content specified by the administrator and combine it with the page specific content specified by the contributors, format it, apply the PHPNuke template to it, and deliver it to the viewer.
Priority	Top Priority. Without the generation of pages, the entire system is unusable and pointless.
Actors: Primary Secondary	 Site Viewer phpGroupWare DB phpGroupWare ACL
Pre-conditions	1. Administrator has installed and selected a theme. 2. There are categories with pages in them.

Use Case Item	Description
Post-conditions: **Success** **Failure**	
Related Use Cases	None
Cross-References and Notes	User Interfaces 1. See Fig 9. 2. See Fig 10.
Basic Course	1. Request for a page is made by the Site Viewer. 2. A check to be sure that requested the page exists is made. 3. Assuming the page exists, a permission request is sent to the phpGroupWare ACL actor to see if the current user has access to view the requested page. 4. Assuming permission is granted, the template set (theme) being used is looked up. 5. The header of the site is generated using header content set by the Administrator and formatted using the selected theme. 6. The blocks selected by the Administrator for the left column (if any) are looked up and included in the left column of the page and formatted using the selected theme. 7. The Contributor specified page content is looked up and formatted according to the selected theme. 8. The blocks selected by the Administrator for the right column, if any, are included in the right column of the page and formatted using the selected theme. 9. The footer content specified by the Administrator is looked up and formatted according to the selected theme. 10. The entire page, header, left column, content, right column, and footer is delivered as an html document to the Site Viewer.

Use Case Item	Description
Successful Alternative Courses:	When the check is made to see if the page exists (2), it may be found not to exist. In this case, the page is generated as usual, but the content section (7) will display an error message instead of normal content. When the permissions check is made in step (3), the Site Viewer is not found to have permission to view the page. In this case, the page is generated as normal, but the content section (7) will display an error message instead of the content.
Unsuccessful Alternative Courses:	At any step there may be an error retrieving content from the database (by way of the phpGroupWare DB actor). In this case, an error message will be printed and no other steps in the basic course taken. At any step there may be a problem with the template (theme) that may be unexpected. This will cause the basic course to stop and an error to be printed.
Open Issues	

SECTION 5: TEST CASE PLAN

Unit Testing

Test Case ID	Items being Tested	Input(s)	Expected Outputs	Actual Outputs
1	ManagePages() in Contributor_ManagePage_UI	None	Category Managing Page should be generated.	Correct!
2	EditPage() in Contributor_ManagePage_UI	Category ID and Page ID	Page Editor page should be generated	Correct!
3	GetPermittedCategoryIDList() in Categories.BO	None	An array of category ids	Correct!
4	can_read_page($page_id) in ACL_BO	Page ID	True, if the contributor has permission to read the page. Else, return false	Correct!
5	can_write_page($page_id) in ACL.BO	Page ID	True, if the contributor has the permission to write	Removed from implementation
6	GetFullCategoryIDList() in Categories.SO	None	An array of category ids	Correct!
7	GetPageIDList($cat_id) in Pages.BO	Category ID	Array of page IDs	Correct!
8	AddPage($cat_id)in Pages.BO	Category ID	Return new page id, which is created in DB.	Correct!
9	RemovePage($cat_id) in Pages.BO	Category ID	True, if the page is removed, else False	Correct!
10	GetPage($page_id) in Pages.BO	Page ID	A Page class, which the page id.	Correct!
11	SavePageInfo($page) in Pages.BO	Page class	True, if the page class is saved in the data backend. Else, False	Correct!
12	GetPageIDList($cat_id) in	Category ID	Array of available pages	Correct!

	Pages.SO		under the category id.	
13	AddPage($cat_id) in Pages.SO	Category ID	New page id created in DB	Correct!
14	RemovePage($page_id) in Pages.SO	Page ID	The query call from the DB	Correct!
15	GetPage($page_id) in Pages.SO	Page ID	If the page id exists, a page class with the page ID. Else, return False.	Correct!
16	SavePage($page) in Pages.SO	Page class	True, if the page class is saved.	Correct!
17	ChangeTheme() in Admin.PHPNuke.UI	None	Theme selection page should appear.	Implemented in next version
18	SelectBlocks() in Admin.PHPNuke.UI	None	Blocks managing page should appear.	Implemented in next version
19	is_admin() in ACL.BO	None	Allow the user to go to the request page.	Correct!
20	GetLeftBlocks() in Blocks.Bo	None	The left side blocks should be displayed in the block selection page.	Implemented in next version
21	GetRightBlocks() in Blocks.Bo	None	The right side blocks should be displayed in the block selection page.	Implemented in next version
22	GetAvailableBlocks() in Blocks.BO	None	Blocks information should be retrieved from the database. Either blocks are available or blocks can't be retrieved.	Implemented in next version
23	SetBlock($filename, $side, $position): Boolean in Blocks.BO	Block filename, Side (left/right), Position on side (1-X)	Blocks should be displayed in the way as the administrator assigns them.	Implemented in next version
24	GetSiteHeader() in HeaderFooter.BO	Variable call.	The current header of the site should be displayed in	Correct!, displays "Team

			the text field.	X project."
25	GetSiteFooter() in HeaderFooter.BO	Variable call.	The current footer of the site should be displayed in the text field.	Correct!, displays "Copyrighted 2002"
26	SetSiteHeader() in HeaderFooter.BO	Header (Team X project)	The Administrator's input in the header text field should be displayed in the final generated page.	Correct!, displays "Team X project."
27	SetSiteFooter() in HeaderFooter.BO	Footer (Copyrighted 2002)	The Administrator's input in the footer text field should be displayed in the final generated page.	Correct!, displays "Copyrighted 2002"
28	SetTheme() in Theme.BO	Information of Theme	The site should be looked like the theme the administrator selected.	Implemented in next version
29	GetTheme() in Theme.BO.	Information of Theme	The Theme selected by the administrator should be retrieved.	Implemented in next version
30	GetAvailableTheme() in Theme.BO	None	The themes in the database should be retrieved or a error message should be displayed	Implemented in next version
31	GetLeftBlocks() in Blocks.SO	None	The chosen block should be retrieved from the database.	Implemented in next version
32	GetRightBlocks() in Blocks.SO	None	The chosen block should be retrieved from the database.	Implemented in next version
33	SetBlock() in Blocks.SO	None	The block selection setting set by the administrator should be saved into the database.	Implemented in next version
34	SetPreference($name, $value:	name and	The preference set by the	Correct!

	String) SitePreference.SO	value	administrator should be saved into the database.	
35	GetPreference($name) SitePreference.SO	Name	The requested preference should be retrieved from the database then returned to the admin	Correct!

Integration Testing

Test Case ID	Items being Tested	Input(s)	Expected Output(s)	Actual Output(s)
1	Add new category	1. Go to the "Manage Categories" page in the administrative section of the website. 2. Select "Add new Category" link 3. Fill in all the fields with "valid" inputs. 4. Click "Save" button.	1. The newly added category should be available (either read and/or write) to all the groups/individuals who were given those access permissions. 2. The category should NOT be available (either read and/or write) to groups/individuals who are not given those access permissions. 3. The name of the category should be listed in the Administrative Category Management page so that the administrator can edit the categories settings and/or permissions.	1. Correct! 2. Correct! 3. Correct!
2	Edit Category	1. Go to the "Manage Categories" page in the administrative section of the website. 2. Select "Edit Category" link	1. The edited category should be available (either read and/or write) to all the groups/individuals who were given those	1. Correct! 2. Correct! 3. Correct!

		3. Edit all fields w/ correct inputs 4. Click “Save” button.	access permissions. 2. The category should NOT be available (either read and/or write) to groups/individuals who are not given those access permissions. 3. The new name and description of the category should be listed in the Administrative Category Management page so that the administrator can edit the categories settings and/or permissions.	
3	Delete Category	1. Go to the “Manage Category” page in the administrative section of the website. 2. Select “Delete Category” for a category. 3. An alert box should pop up; hit “ok” to delete the category.	1. The category should be removed from the phpGroupWare database. 2. All pages contributed to the category should also be removed. 3. The category should no longer be listed on the contributor page, nor should it be listed in the Administrative Category Management Page.	1. Correct! 2. Correct! 3. Correct!
4	Header / Footer Change	1. Request for headerfooter page.	Header: Team X project. Footer: Copy righted 2002	Header: Team X

		2. Do a permission checks. 3. After permission is checked, headerfooter editing page appears. 4. Input “Team X project” into header text field and “Copy righted 2002” into footer text field. Then save it.		project. Footer: Copyrighted 2002
5	Select Blocks Page	1. Request for block selection. 2. Do a permission checks. When permission is checked, the selection page shows up, block list has been retrieved from the database. 3. The admin changes the selections.	The blocks in the final generated page should look like the one the admin positioned	Implemented in next version
6	Theme Selection page	1. Request for theme selection page. Permission checks. 2. When the permission is granted, the theme list would be loaded from the database. 3. The Admin can choose the desired theme for the site.	The theme of the site should be exactly the same as the one choose by the admin.	Implemented in next version
7	Add new page	1. Click add new page button in the category-managing page. 2. Provide the content of the page in the page managing page.	1. The new added page should be viewable to the contributor. 2. The name of the page should be included in the list of pages in the	1. Correct! 2. Correct!

		3. Click “Save” button.	category in category-managing page.	
8	remove a page	Clicking “Remove” button in category-managing page.	1. The removed page should be removed from phpGroupWare data backend. 2. The removed page name should be removed from the list of pages in the category.	1. Correct! 2. Correct!
9	Edit a page	1. Click “Edit” button next to the page name in category managing page. 2. Edit the content of the page. 3. Click “Save” button.	1. After the Edit button is clicked, the page editor page should be generated. 2. After clicking “Save” button, the edited page should be appeared in a separate window with confirmation.	1. Correct! 2. Correct!

System Testing

Test Case ID	Items being Tested	Input(s)	Expected Output(s)	Actual Output(s)
1	Contributor_ ManagePage_ UI	Go to the category managing page.	Category managing page should be generated with list of categories and the list of the pages.	Correct!
2	Contributor_ ManagePage_ UI	Go to the category managing page. Click “Add new page” or “Edit” button.	After clicking “Add new page” or “Edit” button, the page managing page should be viewable with proper content.	Correct!
3	add page functions in Pages_BO, Pages_SO & ACL_BO	Add number of pages.	The titles of the pages should be added to the category. The write permission of the contributor should be checked whenever a page is added.	Correct!
4	Editing page functions in Pages_BO, Pages_SO & ACL_BO	Edit a page	The write permission of the contributor should be check before the page managing page is viewable. The edited page should be changed to the edited information.	Correct!
5	Removing a page in Pages_BO, Pages_SO & ACL_BO	Remove a page	The write & read permission of the contributor should be checked whenever a page is removed from the category.	Correct!

			The page title should be removed from the category in the category-managing page. The page should not be viewable.	
6	Admin_ PHPNuke_UI	Select Blocks selection page.	The blocks list should be generated and the selection and position pull down menu should be displayed next to each block	Implemented in next version
7	Setblocks() in both Block_BO and Block_SO	Save blocks setting.	The Blocks setting should be saved into the database. And the displayed blocks should be displayed as the way they are positioned.	Implemented in next version
8	theme list database retrieval in both Theme_BO and Theme_SO	Retrieve the theme list from the database.	The Permission should be checked before the retrieving process. The list will then retrieve and generate for the admin to use;	Implemented in next version
9	blocks data retrieval in both Blocks_BO and Blocks_SO	Retrieve the block list from the database	Before the retrieving process, the permission of the user will be checked. The block list will then retrieve and allow the user to select them.	Implemented in next version
10	save theme function in both Theme_BO and Theme_SO	Select a theme and then save it.	The theme will be saved into the database. The display page should display the same theme as selected.	Implemented in next version

			When the saved theme is called, the saved theme should be retrieved and returned to the user's screen.	

SECTION 6: USER INTERFACE DIAGRAMS

Administrative UI Diagrams

Figure 1: Administrative Menu UI Diagram:

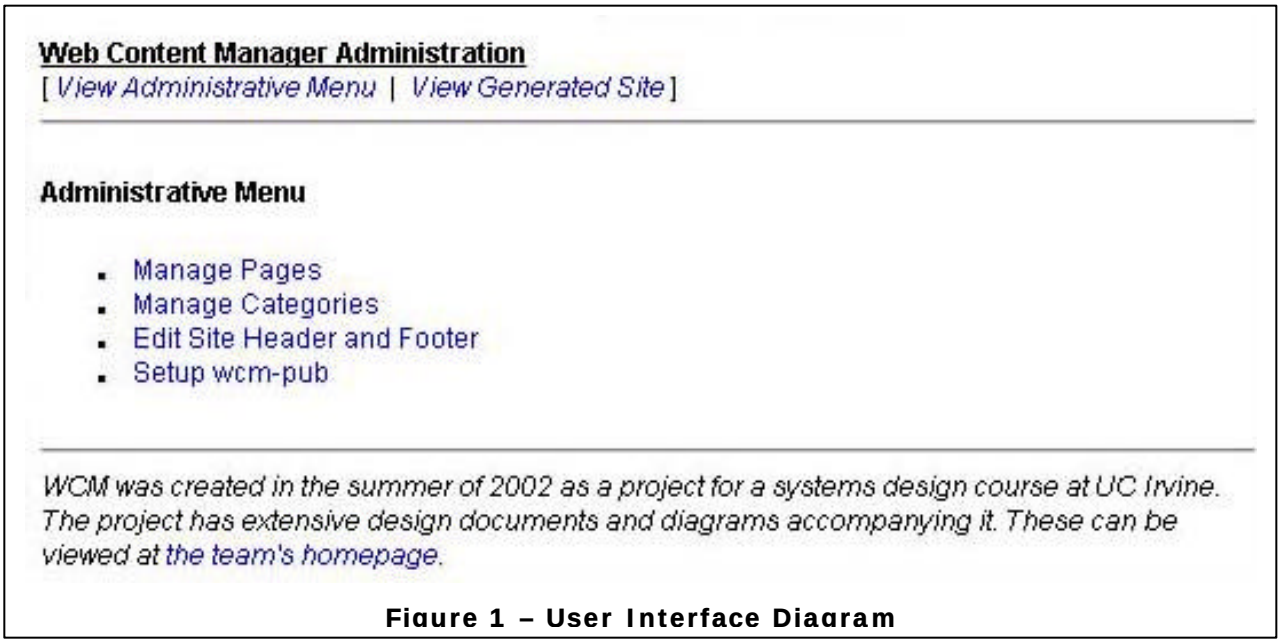


Figure 1 – User Interface Diagram

Figure 2: Category Manager UI Diagram 1:

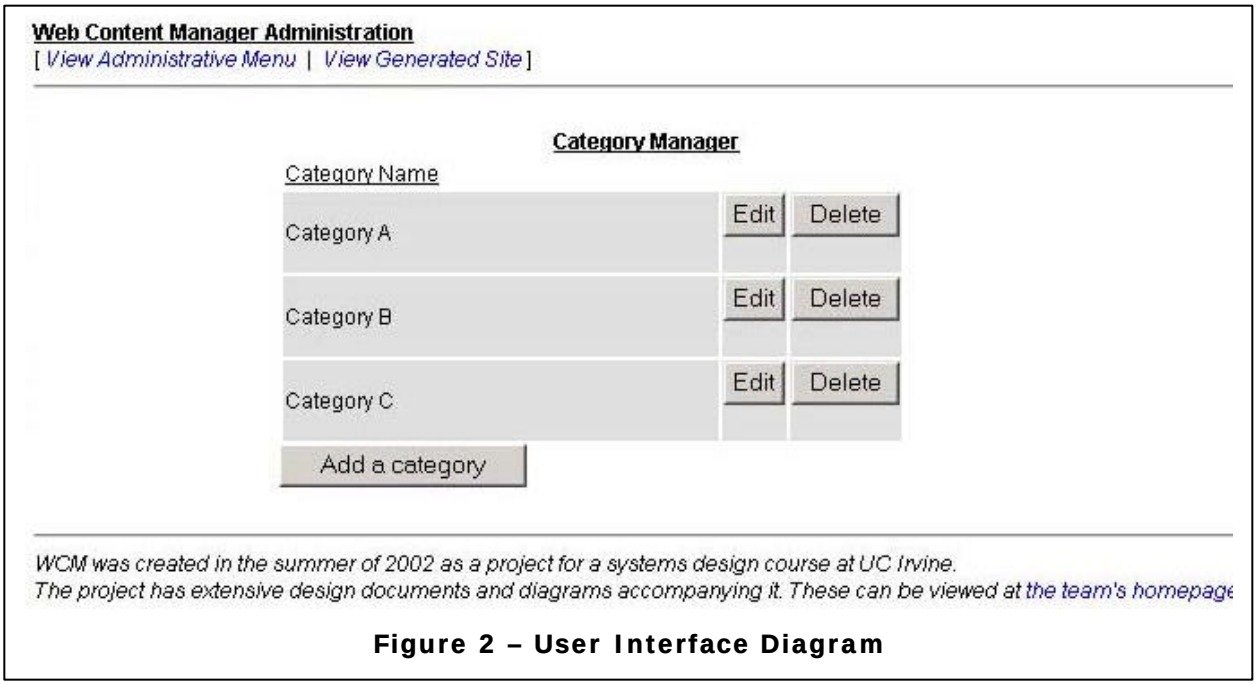


Figure 2 – User Interface Diagram

Figure 3: Add/Edit Category UI Diagram 2:

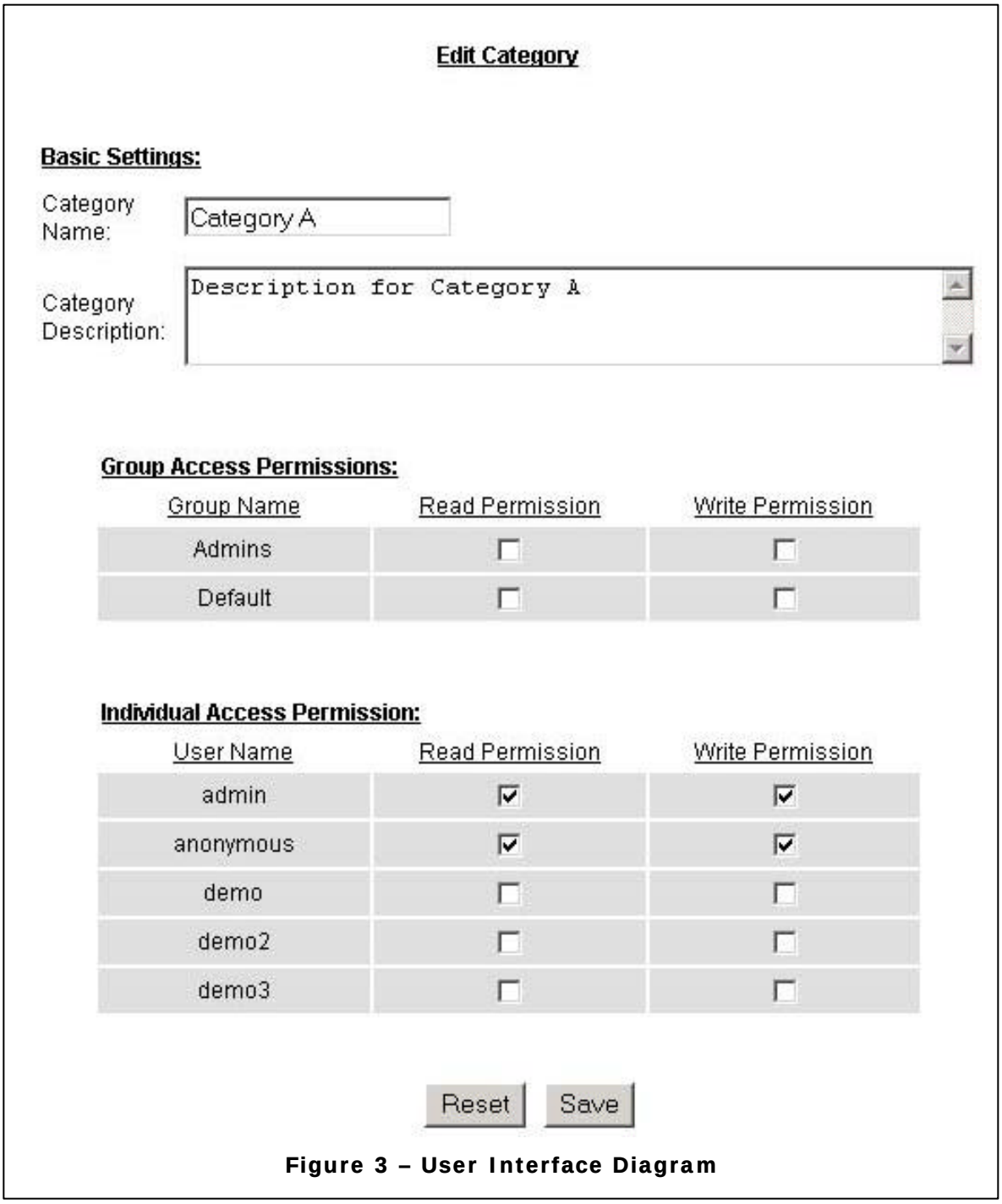


Figure 3 – User Interface Diagram

Figure 4: Edit Header / Footer Content UI Diagram:

Contributor UI Diagrams

Figure 5: Page Manager UI Diagram:

Figure 5 – User Interface Diagram

Figure 6: Add/Edit Page UI Diagram:

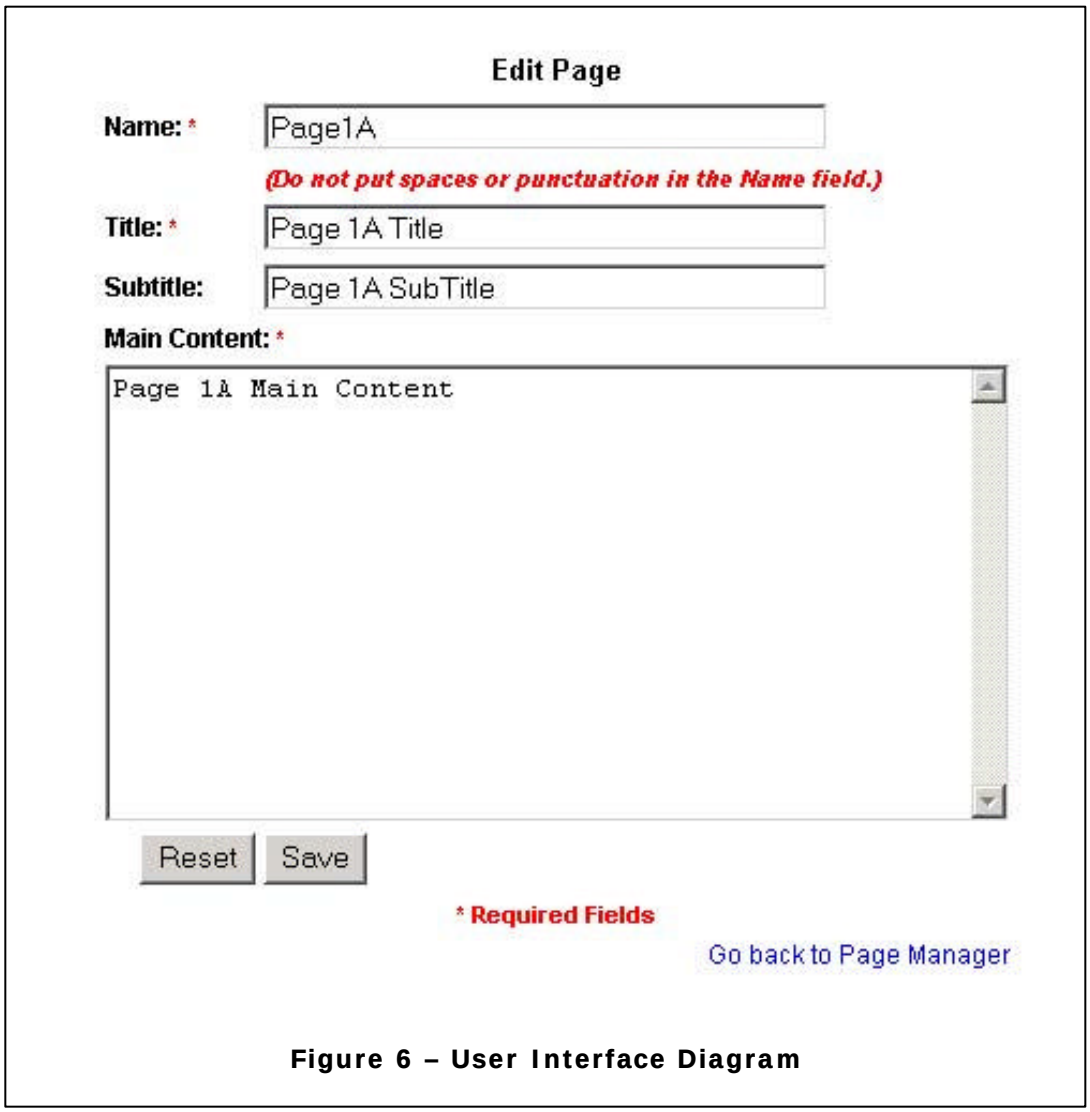


Figure 6 – User Interface Diagram

Page Generation UI Diagrams

Figure 7: Generated Page: Site Contents UI Diagram:

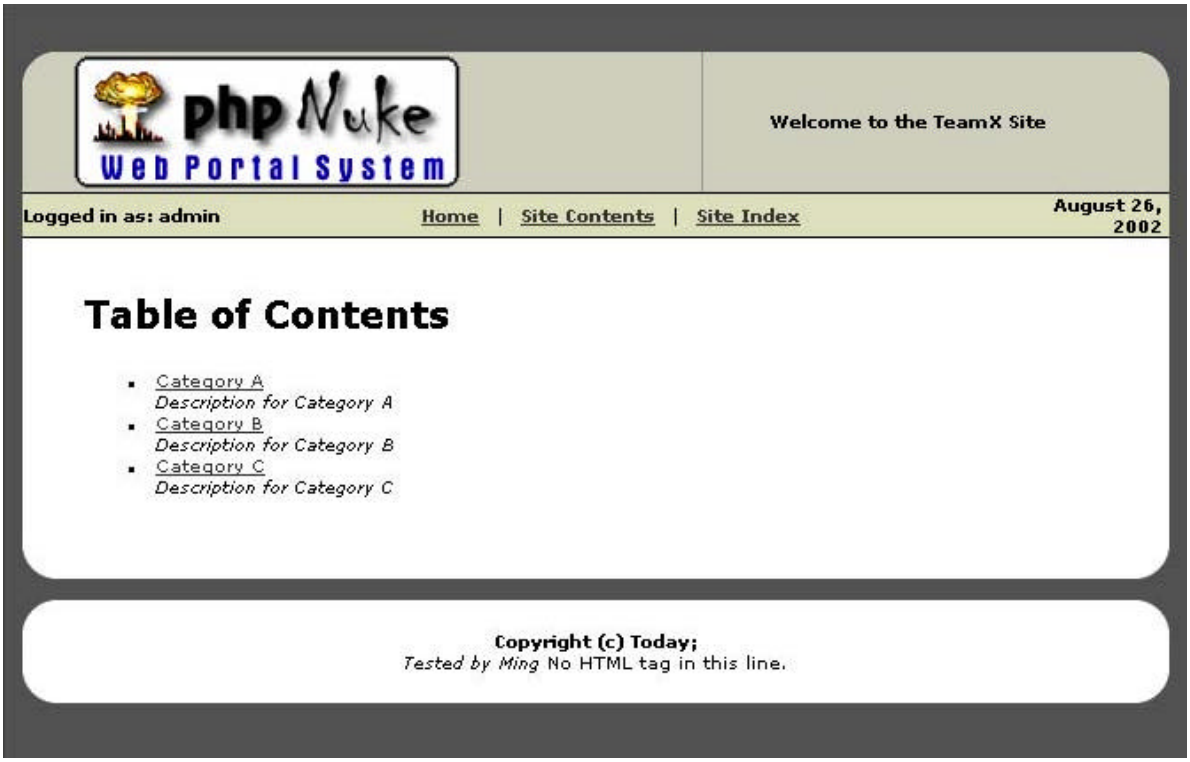


Figure 7 – User Interface Diagram

Figure 8: Generated Page: Site Index UI Diagram:

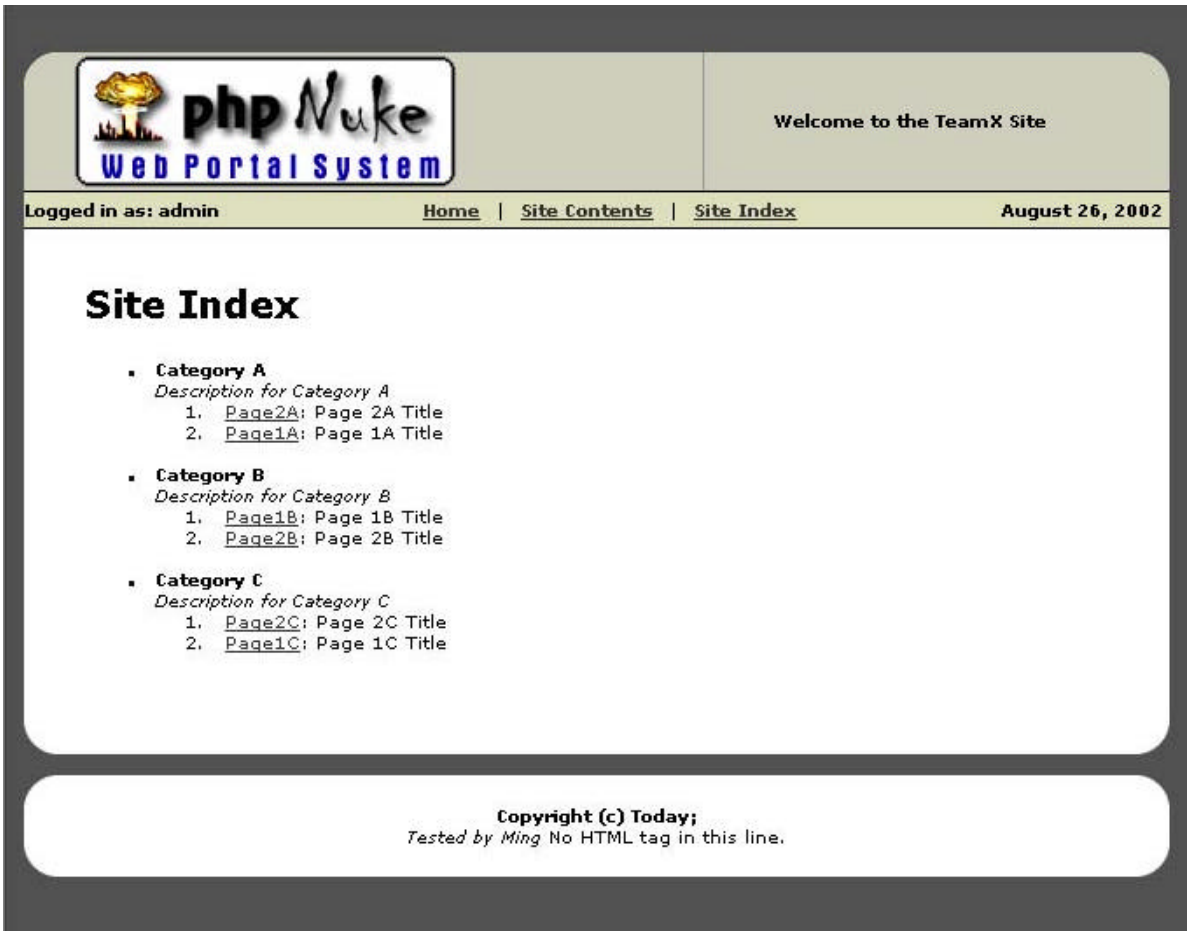


Figure 8 – User Interface Diagram

Figure 9: Generated Page: Category Table of Contents UI Diagram:

Figure 9 – User Interface Diagram

Figure 10: Generated Page UI Diagram:

Figure 10 – User Interface Diagram