Course WebSite Analysis Document

Objective:

The purpose of our project is to build a simplified way to make a college course website. The problem is that many professors don't get around to making websites because it's too much work for them. The solution is to build a simpler website design interface. With a simpler design interface, the work involved in making a course website becomes trivial, so professors no longer have a reason to not create one. It also makes student's lives easier by keeping track of all their courses.

Domain Analysis:

This program will be used at a university so that professors have an easy way to create a website and so the students have a place to go to locate information about their classes. The users of our program are the professors creating the course and subpages, and the students who are viewing their professor's pages. The customers of our program will be the university hosting the web application. The environment containing our program is the website hosting the web application.

Our system will replace the services currently provided by Moodle, the current course webpage management system. Moodle has a reputation for being complicated and error-prone, and it is not universally used. If professors do not want to use Moodle, the only alternative is for them to create their own site, which is time consuming and prevents the adoption of a single standard for course pages.

Requirements Statement:

I. Functional Requirements
   a. Input
      i. The system will accept username and password from each registered user.
      ii. Users will be marked as students or teachers.
      iii. Students will view webpages.
      iv. Teachers will make or edit webpages.
   b. Output
      i. If a teacher is logged in, the system will output a list of courses they have created.
      ii. If a student is logged in, the system will display list of courses.
      iii. The system outputs the text data and a list of subpages within the course page.
   c. Data

i. The system will store each of the professor's created pages and the text content on those pages.
ii. The system will store a username and password for each user.
iii. The system will store a list containing every page that teacher has created.
iv. The system will store whether the user is a student or professor.
v. The system will save the text data of courses and subpages.

d. Computations
i. The system will take form data input by professors, and convert it into a clear, read-only webpage.
ii. The system needs to determine which pages should exist on a users homepage.

II. Platform Requirements
a. The system will run on a Java server, and a Derby database.
b. The system should be capable of hosting the web application.
c. The system will be accessible on all modern web browsers.
d. The system will be developed within the JSF web application framework.

III. Process
a. Development will be done in incremental steps, using a modular approach.
b. During design, the core features of the system will be divided into several modules, which will be individually implemented, tested, and revised as necessary. As time permits, additional functionality may be added through new modules towards the end of the development process.
c. The final version of the system will be delivered by the end of the semester.
d. The cost of development will be minimal, as no new resources need to be purchased in order to build the system.

IV. Quality
a. The system has two primary values: simplicity and ease of use.
b. Systems for hosting class pages are already in use, so we will focus on making ours as easy to use as possible, to address the weaknesses of other, similar systems.
c. The system will remove as much time and difficulty as possible from the process of creating a course page.

d. It will also provide a simple, clear, and functional way for viewing course pages.

User cases:

User Case: Professor modifies an existing course page

1. Log into existing account
    a. If failure, give error message, no page navigation and allow log in again
2. Homepage shows a list of courses they have made.
3. The professor selects a desired course to view and hits go to course page.
4. The professor clicks the edit button below the course text.
5. The professor modifies the text.
6. When done, the professor clicks the **Submit** Button.

User Case: Professor modifies an existing sub page

1. Log into existing account
    a. If failure, give error message, no page navigation and allow log in again
2. Homepage shows a list of courses they have made.
3. The professor selects a desired course to view and clicks on go to course page.
4. The professor selects a desired subpage to view and clicks go to subpage.
5. The professor clicks the edit button below the subpage text.
6. The professor modifies the text.
7. When done, the professor clicks the **Submit** Button.

Use Case: Student accesses a professor’s course page

1. Logs in to their existing account
    a. If failure, give error message, no page navigation and allow log in again
2. Homepage shows a list of registered courses.
3. Student clicks on the desired course to view and hits go to course page.
4. The application accesses the course’s information and displays the information into an organized, easy to read webpage.

Course WebSite Analysis Document

Use Case: Student accesses a professor's course subpage

1. Logs in to their existing account
    a. If failure, give error message, no page navigation and allow log in again
2. Homepage shows a list of registered courses.
3. Student clicks on the desired course to view and clicks go to course page.
4. Student clicks on the desired subpage to view and clicks go to sub page.
5. The application accesses the subpage's information and displays the information into an organized, easy to read webpage.

Use Case: Professor views their course page

1. Log into existing account
    a. If failure, give error message, no page navigation and allow log in again
2. Homepage shows a list of registered courses.
3. Professor clicks on a desired course to view and clicks go to course page.
4. Application accesses the course's information and displays the information into an organized, easy to read webpage.

Use Case: Professor views their subpage for a course

1. Log into existing account
    a. If failure, give error message, no page navigation and allow log in again
2. Homepage shows a list of courses they have made.
3. The professor selects a desired course that has the subpage they would like to view and clicks on go to course page.
4. The professor selects a desired subpage to view and clicks go to subpage.
5. Application accesses the subpage's information and displays the information into an organized, easy to read webpage.

Use Case: Professor creates a new course page

1. Log into existing account

   a. If failure, give error message, no page navigation and allow log in again
2. Professor clicks on a Add a course button at the bottom of the page.
3. Professor types the desired course name and hits submit.
4. The course is added to the database and the professor is brought back to the homepage.

Use Case: Professor creates a new course subpage

1. Log into existing account
   a. If failure, give error message, no page navigation and allow log in again.
2. Professor clicks on a desired course they would like to add a subpage to and clicks go to course page.
3. Professor clicks on Add a sub page button at the bottom of the page.
4. Professor types the desired course name and clicks submit.
5. The course is added to the database and the professor is brought back to the course page.

Use Case: Professor deletes a course page

1. Log into existing account
   a. If failure, give error message, no page navigation and allow log in again.
2. Professor clicks on a Delete a course button at the bottom of the page.
3. Professor clicks on the desired course name they would like to delete and clicks submit.
4. The course is removed from the database, removed from all the enrolled students' course lists and the professor is brought back to the home page.

Use Case: Professor deletes a course subpage

1. Log into existing account
   a. If failure, give error message, no page navigation and allow log in again.
2. Professor clicks on a desired course that has the subpage they would like to remove in it and clicks go to course page.

3. Professor clicks on a Delete a sub page button at the bottom of the page.
4. Professor clicks on the desired subpage they would like to delete and clicks submit.
5. The subpage is removed from the database and the professor is brought back to the course page.