# CST-126 Project Guide

**Directions:** Throughout this course, students will create a database-driven web application using PHP and MySQL.

## Contents

## Project Overview

In this course, students will create a blogging site implementing the design, requirements, expectations, components, and functionality described below. Each programming project will have two purposes:

1. Expand and reinforce previously covered concepts.
2. Challenge students to incorporate new concepts and techniques into their PHP and MySQL programming skill set.

## Design

1. Use the "Project Report Template," located within the Course Materials.
    a. Topic, Date, Revision, and Team fields properly populated
    b. Approach clearly documented and kept up to date
    c. Decisions clearly documented and kept up to date
    d. Clear demonstration and intent showing that this is a "living" design specification
2. Completed Sitemap to support all functionality delivered.
3. Completed ER diagram to support all functionality delivered.
4. Mock diagrams, white board screenshots, and other meeting notes captured during team meetings.
5. Tests for every single aspect of code as well as assessment of what works, what does not work, what can be fixed, and what cannot be fixed.
6. A list of open issues and known bugs.
7. A list of suggested implementations for the next or future version.

## Technical Requirements

1. Complete and comprehensive Install Instructions that could be executed to install application, install database, and configuration application at the Hostable Service Provider.
2. Complete DDL script that could be used to recreate schema (table structure and appropriate test data) "anywhere" including external hosting environment.
3. Complete DDL script that includes all proper constraints (NOT NULL, PK, FK, etc.) to support a scalable and properly designed database.
4. Common User Interface design (via use of common CSS and/or Page Fragments).
5. All PHP code is cleanly formatted, class headers documented with block comments, class method headers documented with block comments, class method implementation documented with pseudo code single line comments.
6. Application hosted and accessible externally outside of localhost development environment.

## Functional Requirements

1. Clean appealing user interface design with an Application Title and easy to use Menu System.
2. Display of clear and concise error messages (for all data entry forms or system errors).
3. Inability for the same user (username and password) to register multiple times.
4. Ability to register new Users.
5. Ability to log in to the application using the database for authentication.
6. Ability to display a list of blogs.
7. Ability to create a new blog.
8. Ability to update a blog (only by the author of the blog).
9. Ability to delete a blog (only by the author of the blog).
10. Ability to search for blogs based on a keyword search in the blog title or blog content.
11. Ability to comment on a blog (only by blogs not created by the author).
12. Ability to log out of the application.

| Component | Functions | Relevant table |
|---|---|---|
| Self-Registration | Process user information<br>Verify that user does not exist<br>Store information in users table | Users personal information |
| User Login | Process user information<br>Authenticate user<br>Password retrieval / (reset not retrieve). How this is implemented is a complicated question. | Login & Passwords |
| Main App Administration screen | Create user<br>Manage user's roles and permissions<br>Delete users<br>Email users (individual and group)<br>Edit any post or comment<br>Edit categories (if implemented) | Roles and permissions |
| Content Administration screen | Manage posts and categories<br>Remove post<br>Edit Post<br>Tag post | Categories<br>Posts |
| New Post | Create post<br>Reply to post (you can't reply to a post when creating one).<br>Store post in post table<br>Categorize post<br>Tag post<br>Filter language | Posts |
| List of posts | Lists posts<br>Filter list (by date, topic, keyword)<br>Sort list<br>View post<br>Pagination | Posts |
| Search | Search post (by user, by category, by date, by | Posts |

| | | |
|---|---|---|
| | title)<br>Search user (by last name, role) | Users |
| Read post | Comment on a post<br>Rate post (this we never discussed) | Posts<br>Users<br>Ratings |

**Additional Functionality Requirements**

Within the Main Screen, there should be a list of xx posts plus links to perform the following functions

1. Simple search function
2. User self-registration
3. Log in to users account
4. Comment on a post (do we do anonymous comments or not?)

Once a user is logged in additional links should go to these functions:

1. Create a new post
2. Edit an existing post that they created
3. View/edit user profile

If there is an admin role then the list of available functions includes a link to the App Admin screen.

If there is a content admin role then the list of available functions include a link to the Content Admin screen.

## Blog Project Suggested Schema

**Users**

- User_id PK, auto generated by mysql or app
- User_name unique, not null, required
- User_role required, not null, FK to role.role_id
- User_nickname unique, not null, if not provided use first/last name
- First_name required, not null
- Middle_Name optional
- Last_name required, not null
- Email1 required, unique, not null
- Email2
- Address1
- Address2
- City
- State
- Zipcode
- Country
- User_banned y/n, default to 'n'
- User_deleted y/n, default to 'n'

**Posts**

- Post_id PK, auto generated by mysql or app
- Post_title unique, required, not null
- Category_id FK to category.category_id
- Post_content not null, required
- Posted_date auto, required
- Posted_by FK to user.user_id, required
- Updated_date auto
- Updated_by FK to user.user_id
- Deleted_Flag y/n, default to 'n'

**Comments**

- Comment_id PK, auto generated by mysql or app
- Post_Id FK to post.post_id, required
- Comment_text required, not null
- Posted_date auto
- Posted_by FK to user.user_id, required
- Updated_date auto
- Updated_by FK to user.user_id
- Deleted_flag y/n, default to 'n'

**Categories**

- Category_id PK
- Category_name unique, required, not null
- Created_Date auto
- Created_by FK to user.user_id, required
- Active_flag y/n, default to 'y'

**Roles**

- Role_id unique, PK, auto generated by mysql or app
- Role_name unique, not null
- Role_description required, not null
- Created_date auto
- Created_by FK to user.user_id
- Updated_date auto
- Updated_by FK to user.user_id
- Active_flag y/n, default to 'y'

**Optional**

**AuditLogRoles**

Use a trigger to insert before source table changes (update, delete) along with date and user_id of made the change

**AuditLogUsers**

Use a trigger to insert before source table changes (update, delete) along with date and user_id of made the change

## Test Plan

1. Preparation
    a. Delete all the data in your database tables.
2. Registration
    a. Bring up Registration Form.
    b. Try to register user without filling out Form.
    c. Ensure proper data validation errors displayed.
    d. Register a new user.
    e. Ensure proper positive success message is displayed.
    f. Try to register an existing user.
    g. Ensure proper error message displayed.
3. Log In
    a. Bring up the Login Form.
    b. Try to log in without filling out Form.
    c. Ensure proper data validation errors displayed.
    d. Try to log in with incorrect Username.
    e. Ensure proper data validation error is displayed.
    f. Try to log in with incorrect Password.
    g. Ensure proper data validation error is displayed.
    h. Try to log in with correct Username and Password.
    i. Ensure main application page, proper menu, and logo are displayed.
4. Log Off
    a. Log off the application.
    b. Ensure start application page, proper menu, and logo are displayed.
5. Add a Blog
    a. Log in to the application.
    b. Bring up the Blog Entry Form.
    c. Try to create a new Blog without filling out Form.
    d. Ensure proper data validation errors displayed.
    e. Try to create a new Blog without filling out Blog Title.
    f. Ensure proper data validation errors displayed.
    g. Try to create a new Blog with without filling out Blog Content.

h. Ensure proper data validation errors displayed.
i. Create a new Blog.
j. Ensure proper positive success message is displayed.

6. Display Blog List
    a. Bring up the Blog List.
    b. Ensure only the Blog Title is displayed in the List.
    c. Ensure that View Blog option is displayed in the List.
    d. Ensure that Update and Delete Blog options are displayed in the List only for Blog Owners. Note, depending on the user interface design this functionality may be available as a "Display My Blogs" menu option.
7. View a Blog
    a. Bring up the Blog List.
    b. View a Blog from the List.
    c. Ensure that the Blog Title, Blog Content, and (optional) Blog Author is displayed.
    d. Ensure navigation back to the Blog List is available.
8. Update a Blog
    a. Bring up the Blog List (or My Blog List).
    b. Bring up the Blog Update Form.
    c. Ensure that the Blog Title and Blog Content is displayed in the Form.
    d. Try to update the Blog by removing the Title and Content from the Form.
    e. Ensure proper data validation errors displayed.
    f. Try to update the Blog Title.
    g. Ensure proper positive success message is displayed.
    h. Bring up the Blog List (or My Blog List).
    i. Ensure Blog Title was updated.
    j. Try to update the Blog Content.
    k. Ensure proper positive success message is displayed.
    l. Bring up the Blog List (or My Blog List).
    m. Ensure Blog Content was updated.
9. Delete a Blog
    a. Bring up the Blog List (or My Blog List).
    b. Bring up the Blog Delete page.
    c. Try to delete the Blog Title.

d. Ensure prompt is displayed with ability to cancel or accept the operation.
e. Ensure if operation is canceled that proper navigation is available.
f. Ensure if operation is accepting that a proper positive success message is displayed.
g. Bring up the Blog List (or My Blog List).
h. Ensure Blog Content was deleted.

10. Search a Blog
    a. Bring up the Search Form.
    b. Try to search for a Blog without filling out Form.
    c. Ensure proper data validation errors displayed.
    d. Try to search for a Blog by Title and/or Content.
    e. Ensure Blog List is displayed.
    f. Ensure View (and Update and Delete Blog available for Blog owner) is available.
11. Comment a Blog
    a. Bring up a View Blog page (via Blog List or Blog Search).
    b. Bring up the Comment Form.
    c. Try to comment on a Blog without filling out Form.
    d. Ensure proper data validation errors displayed.
    e. Comment on a Blog.
    f. Bring up a View Blog page (via Blog List or Blog Search).
    g. Ensure Blog Comments are visible.
12. General UI
    a. Menu visible at all times with the correct menu items for the application context.
    b. Application Title and Logo visible on all pages.
    c. Error pages integrated within application pages and menu system.
    d. Navigation does not require use of browser back buttons.
    e. All forms have min and length data validation rules.
    f. Proper Error pages are displayed with clear error message.
    g. Pages and Forms are laid out cleanly using layout controls such as a HTML Table.

## Milestone 1: User Registration Module

**Objective:** Create a user registration page for your blog.

**Activity**: Browse the Internet and review several registration pages for sites similar to the one being built in this course. Create a list of several elements you think should make up a

registration page. Make sure to obtain instructor approval for the elements of your registration page.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during registration.
2. In PHP, write the functions that will enable the capturing of user input and store in the database.
3. In HTML, build the minimally functional form to capture user registration.

**Deliverables**

A fully functional Registration Page, including:

1. The schema
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML
5. Project Report

**Submission**

Submit the following:

1. A Project Report (using the "Project Report Template," located within the Course Materials) containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.
2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections

## Milestone 2: Login Page

**Objective:** Create a login page for your blog.

**Activity**: Browse the Internet and review several login pages for sites similar to the one being built in this course. Create a list of several elements you think should make up a login page. Make sure to obtain instructor approval for the elements of your login page. Consider the execution flow to capture the user interaction.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the log in process.
2. In PHP, write the functions that will enable the capturing of user input and store in the database.
3. In HTML, build the functional login form to capture user login.
4. In PHP, perform the necessary user authentication. Before you code, document every detail of the user experience such as: number of login trials allowed, constraints on the password, etc.
5. Based on the desired interaction and data captured, build the MySQL tables to store this information and enable the desired functionality.
6. Connect the Registration and Login pages.
7. Document any revisions that might be necessary to code, schema, tables, or user interface created in the previous module.

**Deliverables**

A fully functional Login Page, including:

1. The schema diagram/wireframe
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.
2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections

**Implementation Notes**

Ensure that concepts and functionality addressed in the course are reflected in your project.

Consult external resources and references to improve and expand your project as needed.

## Milestone 3: Blog Post

**Objective:** Create a new blog post.

**Activity**: Browse the Internet and review several blogs to develop a feel for what posting a blog entails. In preparation, begin to document the following:

1. The user experience and make a list of all the elements that would make up the blog post, the interface, and necessary backend support.
2. How the new post interface will fit with the rest of the blog project as well as any limitations you might impose on the writer.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the blog posting process.
2. In PHP, write the functions that will enable the capturing of the new post and store in the database.
3. In HTML, build the minimally functional form to capture the new post.
4. In PHP, perform the necessary verification of compliance with constraints you decided upon. Before you code, decide and diagram every detail of the user experience such as: size of post, language filters, the inclusion of add-on HTML editors, etc.
5. Based on the desired interaction and data captured, build the MySQL tables to store this information and enable the desired functionality.
6. Connect this module to previous modules of your database and ensure cohesiveness. Document any revisions that might be necessary to code, tables, or user interface created in previous module(s).
7. In PHP, implement a simple language filter that will look for certain forbidden words in the text and approve or disapprove the post. You may opt to research on your own how to accomplish the same in a high-level programming language. Implement this as a separate application, which will be used as a (simulated) external service to the blog.

**Deliverables**

A fully functional Blog Post Page, including:

1. The schema diagram/wireframe
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML
5. All necessary, JavaScript, or other language

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.

2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections
3. Revised project and code files (as needed). Make sure any revisions are noted within the project report.

**Implementation Notes**

Ensure that concepts and functionality addressed in the course are reflected in your project.

Consult external resources and references to improve and expand your project as needed.

## Milestone 4: Deploying and Hosting Your Website

**Objective:** Create an Azure account and upload all files to your account. Interface for the blog administrator to categorize (tag) and remove posts.

**Activity**: Browse the Internet and review several blogs to develop a feel for what post management entails and looks like. In preparation, begin to document the following:

1. The user experience, making a list of all the elements and functions your content administrator panel would do.
2. The interface, specific functions, and necessary backend support.
3. How the content administrator interface will visually fit with the rest of the blog project.
4. How the new functionality might work with existing functionality (do not hesitate to modify prior modules).
5. Any limitations you might impose on the system. How will you verify the integrity of the data in your database and what integrity means in the context of this project.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the blog posting process to reflect new information required by the content administrator. Modify the schema, the tables, the keys, and the relationships accordingly.
2. In PHP, write the functions that will enable the management of blog post data (CRUD), with a special emphasis on categorization (tagging).
3. In HTML, build the minimally functional form to display lists of posts and GUI elements connected with the PHP backend functionality.
4. In PHP, perform the necessary verification of compliance with constraints you decided upon. Before you code, discuss and decide every detail of the user experience such as: how information will be displayed, what happens after an action is taken, what if a wrong action is taken, recovery from errors, etc.

5. Based on the desired interaction and data captured, build the MySQL tables to store this information and enable the desired functionality.
6. Connect this module to previous modules of your database and ensure cohesiveness. Discuss any revisions that might be necessary to code, tables, schema, or user interface created in previous module(s).
7. In PHP, implement simple data integrity verification mechanism. You may opt to research on your own how to accomplish the same in another high-level language, such as Python. Implement this as a separate application, which will be used as a (simulated) external service to the blog.

**Deliverables**

A fully functional Content Administrator, including:

1. The schema diagram/wireframe
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML
5. All necessary, JavaScript, or other language

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.
2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections
3. Revised project and code files (as needed). Make sure any revisions are noted within the project report.

**Implementation Notes**

Ensure that concepts and functionality addressed in the course are reflected in your project.

Consult external resources and references to improve and expand your project as needed.

## Milestone 5: User Administrator

**Objective:** Create a management interface for the blog administrator to manage user roles and permissions.

**Activity**: Browse the Internet and review several blogs to develop a feel for what user management entails and looks like. In preparation, begin to document the following:

1. The user experience, making a list of all the elements and functions your user administrator panel would do.
2. The interface, specific functions, and necessary backend support.
3. How the user administrator will list and read a blog.
4. The various types of users, their permissions, and the reason you would have the permissions levels you decide upon.
5. How the new functionality might work with existing functionality (do not hesitate to modify prior modules).
6. Any limitations you might impose on the system.
7. How you would verify the integrity of the data in your database and what integrity means in the context of this project.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the blog posting process to reflect new information required by the user administrator. Modify the schema, the tables, the keys, and the relationships accordingly. You might need to add new fields to existing tables to support new functionality. You might need to normalize existing tables as well.
2. In PHP, write the functions that will enable the management of user data and their permissions. Modify the User Registration Page to accommodate the new roles and set a default role.
3. In HTML, build the minimally functional form to display lists of posts and GUI elements connected with the PHP backend functionality.
4. In PHP, perform the necessary verification of compliance with constraints you decided upon. Before you code, discuss and decide every detail of the user experience such as: how information will be displayed, what happens after an action is taken, what if a wrong action is taken, recovery from errors, etc.
5. Based on the desired interaction and data captured, build the MySQL tables to store this information and enable the desired functionality.
6. Connect this module to previous modules of your database and ensure cohesiveness. Document any revisions that might be necessary to code, tables, schema, or user interface created in previous module(s).
7. In PHP, implement simple user verification mechanism. You may opt to research on your own how to accomplish the same in a high-level language. Test whether a user with a given role can only access data and perform functions associated with that role. Implement this as a separate application, which will be used as a (simulated) external service to the blog.

**Deliverables**

A fully functional User Administrator Page, including:

1. The schema diagram/wireframe
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML
5. All necessary, JavaScript, or other language

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.
2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections
3. Revised project and code files (as needed). Make sure any revisions are noted within the project report.

## Milestone 6: List of Blog Posts With Update and Delete Blog Module

**Objective:** Create the page that displays the content of the blog site.

**Activity**: Browse the Internet and review several blogs to develop a feel for what the lists of blogs and the display of a blog post (or several posts) entail and look like. In preparation, begin to document the following:

1. The user experience, making a list of all the elements and functions your list of posts and blog display would do and show.
2. The interface, specific functions, and necessary backend support.
3. How the list of posts and the display of content for one or more posts will visually fit with the rest of the blog project.
4. The various types of posts, their categories, and the reasoning for implementing your selected features.
5. How the new functionality might work with existing functionality (do not hesitate to modify prior modules).
6. Any limitations you might impose on the system.
7. How will you would verify the validity of the content of each post and whether users are authorized to view and add posts.
8. What type of statistical information might be useful to the blog administrator to know.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the blog listing and individual post display process, to reflect new information required by the list of blog and content display. Modify the schema, the tables, the keys, and the relationships accordingly. You might need to add new fields to existing tables to support new functionality.
2. In PHP, write the functions that will enable the extraction of data from content tables and their display. Implement several filters to display only (sorted) posts that meet certain criteria such as date, time, content, authors, etc.
3. In HTML, build the minimally functional form to display lists of posts, the associated content, relevant information about the author, and GUI elements connected with the PHP backend functionality.
4. In PHP, perform the necessary verification of compliance with constraints you decided upon, with a special emphasis on content validity and correctness of information displayed. Before you code, discuss and decide every detail of the user experience such as: how information will be displayed, what happens after an action is taken, what if a wrong action is taken, recovery from errors, etc.
5. Based on the desired interaction and data captured, build the MySQL tables to store this information and enable the desired functionality. Connect this module to previous modules of your database and ensure cohesiveness. Document any revisions that might be necessary to code, tables, schema, or user interface created in previous module(s).
6. In PHP, implement a content statistics tool, to inform the blog manager on different metrics such as number of users, number of posts, size of data, dates, etc. You may opt to research on your own how to accomplish the same in another high-level language. Implement this as a separate application, which will be used as a (simulated) external service to the blog.

**Deliverables**

A fully functional List of Blog Posts and Content Page, including:

1. The schema diagram/wirefram
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML
5. All necessary, JavaScript, or other language

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.
2. All necessary files uploaded to the GCU Cloud Hosting Solution.

a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.

b. Comments within the code explaining non-obvious sections

4. Revised project and code files (as needed). Make sure any revisions are noted within the project report.

## Milestone 7: Search Blog Module

**Objective:** Create the page that allows an authorized user to search for posts using multiple criteria.

**Activity**: Browse the Internet and review several blogs to develop a feel for what the search for blog posts, content, and the display of a search result entails and looks like. In preparation, begin to document the following:

1. The user experience, making a list of all the elements and functions your search will require and what search results would be displayed.
2. The interface, specific functions, and necessary backend support. Build in the basic functionality to allow a user to search for keywords or a title of a blog
3. How the search interface and the display of content for search results will visually fit with the rest of the blog project.
4. The various types of posts, their categories, and how this would affect your choice of search criteria.
5. How the new functionality might work with existing functionality (do not hesitate to modify prior modules). You might need to modify existing tables to accommodate for the information required by the search mechanism and criteria.
6. Any limitations you might impose on the system.
7. How you would verify the validity of the content of each post and whether users are authorized to perform the search actions.
8. What type of feedback you would provide to the user and what reporting might be generated on the backend (e.g., store the search for repeated use).

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the search and the display of search result. Plan well what you want to store and what you want to display only without storage, to reflect the functionality you decided upon. Modify the schema, the tables, the keys, and the relationships accordingly. You might need to add new fields to existing tables to support new functionality.
2. In PHP, write the functions that will enable the extraction of data from content tables and their display. Implement several filters to display only the data requested during the search. If no data is found, display appropriate feedback and suggest an alternative search.

3. In HTML, build the minimally functional form to display the search result, the associated content, relevant statistical data about the search (number of items retrieved, time to retrieve it, data size, etc.), and GUI elements connected with the PHP backend functionality.
4. In PHP, perform the necessary verification of compliance with constraints you decided upon, with a special emphasis on user permission to perform the search. Before you code, discuss and decide every detail of the user experience such as: how information will be displayed, what happens after an action is taken, what if a wrong action is taken, recovery from errors, etc.
5. Based on the desired interaction and data captured, build the MySQL tables to store this information and enable the desired functionality. Connect this module to previous modules of your database and ensure cohesiveness. Document any revisions that might be necessary to code, tables, schema, or user interface created in previous module(s).
6. In PHP, implement a content statistics tool, to inform the blog manager on different metrics regarding the activities of users on the blog site. You may opt to research on your own how to accomplish the same in another high-level language.
7. Generate an automated email to users who are not in compliance with a certain rule. Implement this as a separate application, which will be used as a (simulated) external service to the blog.

**Deliverables**

A fully functional Search Page, including:

1. The schema diagram/wireframe
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML
5. All necessary, JavaScript, or other language

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.
2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections
3. Revised project and code files (as needed). Make sure any revisions are noted within the project report.

## Milestone 8: Comment on and Rate a Blog Post

**Objective:** Create the mechanism that allows an authorized user to reply (comment) on a post and rate it.

**Activity**: Browse the Internet and review several blogs to develop a feel for what posting a comment to a blog and rating it entails. In preparation, begin to document the following:

1. The user experience, making a list of all the elements that would make up the blog post, the interface, and necessary backend support. Focus on the rating mechanism (visually) and associated mathematical calculations.
2. How the new post interface will fit with the rest of the blog project.
3. Any limitations you might impose on the rating system and how the ratings would be stored and displayed to other users.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the blog commenting and rating process.
2. In PHP, write the functions that will enable the capturing of the new comment post and store in the database. Implement the MySQL and PHP functionality that would enables the connection between a post and associated replies.
3. In HTML, build the minimally functional form to capture the new comment and rating.
4. In PHP, perform the necessary verification of compliance with constraints you decided upon. Before you code, discuss and decide every detail of the user experience such as: size of post, language filters, the inclusion of add-on HTML editors, etc.
5. Based on the desired interaction and data captured, build the MySQL tables to store this information and enable the desired functionality. Modify the list of blogs and content to include the (indented) replies. You may (although not required) implement multiple levels of replies. Connect this module to previous modules of your database and ensure cohesiveness. Document any revisions that might be necessary to code, tables, or user interface created in previous module(s). Your project might require major revisions at this point to accommodate the reply and rate features.
6. In PHP, implement a reporting system to the manager and users to display the top rated posts (i.e., top 10). You may opt to research on your own how to accomplish the same in another high-level language. Implement this as a separate application, which will be used as a (simulated) external service to the blog.

**Deliverables**

A fully Comment and Rate Mechanism, including:

1. The schema diagram/wireframe
2. All necessary SQL tables
3. All necessary PHP code

4. All necessary HTML
5. All necessary, JavaScript, or other language

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.
2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections
3. Revised project and code files (as needed). Make sure any revisions are noted within the project report.

## Final Blog Application

**Objective:** Revise, improve, and complete the final project.

**Activity**: Review the project features revisiting all decisions made so far. Critically evaluate any bugs, malfunctions, visuals, and features that should be replaced, tweaked, fixed, or expanded; by making decisions based on the information learned throughout the course. Review the documentation in all previous submissions. Test every single aspect of your code and assess what works, what doesn't, what can be fixed, and cannot. Overall, improve every single aspect of your project.

**Deliverables**

A fully functional Database, including:

1. The updated schema/wireframe up to and including the following deliverables
2. All necessary SQL tables
3. All necessary PHP code
4. All necessary HTML
5. All necessary, JavaScript, or other language
6. A list of known bugs
7. A list of suggested implementations for the next version

**Submission**

Submit the following:

1. An updated Project Report containing a description of the project, a list of all modules and files along with a user guide. Within the report, make sure to explain how the desired functionality, features, and constraints have been implemented in code.

2. All necessary files uploaded to the GCU Cloud Hosting Solution.
    a. In each file, include a commented header with the following information: Project name and version, Module name and version, Programmer(s) name(s), Date, Short synopsis of the module, References.
    b. Comments within the code explaining non-obvious sections
3. Revised project and code files (as needed). Make sure any revisions are noted within the project report.