**CST-256 CLC Project Guide**

**Directions:** Throughout this course, students will incrementally design and build a Professional Networking Application. **Note:** This project will be developed within Collaborative Learning Communities (CLC) as directed by the instructor.

Contents

[Project Requirements 2](#_Toc528659709)

[Milestone 1: Registration and Login Modules 3](#_Toc528659710)

[Milestone 2: Member Profiles and Administration Modules 5](#_Toc528659711)

[Milestone 3: Member E-Portfolio Module 7](#_Toc528659712)

[Milestone 4: Affinity Groups Module 9](#_Toc528659713)

[Milestone 5: Job Match Module 11](#_Toc528659714)

[Milestone 6: Web API Module 13](#_Toc528659715)

[Benchmark – Milestone 7: Professional Networking Application 15](#_Toc528659716)

# Project Requirements

The following is a summary of the Professional Networking application requirements and expectations that should be supported by the completion of this project. The deliverables should reflect the design and application that supports the following technical requirements and functional requirements.

**Design**

1. Use of updated standard "Design Report Template," located within the Course Materials, to include:
   1. Topic, Date, Revision, and Team fields properly populated
   2. Approach clearly documented and kept up to date
   3. Decisions clearly documented and kept up to date
2. Clear demonstration and intent showing that this is a "living" design specification.
3. Completed Sitemap to support all functionality delivered as part of Topics 1-6.
4. Completed ER diagram to support all functionality delivered as part of Topics 1-6.
5. Completed security design.
6. Job search and job listing REST API design.
7. Mock diagrams, white board screenshots, and other meeting notes captured during team meetings.
8. Test of every single aspect of the code and assessment of what works, what does not work, what can be fixed, and what cannot be fixed.
9. A list of open issues and known bugs.
10. A list of suggested implementations for the next or future version.

**Technical Requirements**

1. Complete and comprehensive installation instructions that could be executed by a teammate or professor to install the application and database, as well as configure the 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 and Layouts).
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 is designed as a scalable and maintainable N-Layer Application Architecture leveraging MVC, DAO, Façade, and DTO design patterns.
7. Application is designed as a scalable and maintainable Application Architecture leveraging appropriate components (Controllers, Views, Models, Data Validation, and Layouts) of the Laravel Framework using OO patterns and techniques.
8. Application is secure such that a user must be authenticated to access non-Login and Registration pages.
9. Application can be supported in hosted environment outside of localhost development environment leveraging proper exception handling, error page display, and application logging.
10. Application can be 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. Ability to Register new Users.
4. Ability to Log in to the application using database for authentication (custom security design).
5. Ability to create a User Profile.
6. Ability to attach Affinity Groups to a User Profile.
7. Ability for an Administrator to Suspend or Delete a User Profile.
8. Ability for an Administrator to create new Job Postings.
9. Ability to search for Job Postings based on keyword search in Job Title and/or Job Listing.
10. Ability to search for Job Postings and retrieve Job Listing via a REST API.
11. Ability to Log out of the application.

# Milestone 1: Registration and Login Modules

**Overview**

This is a collaborative learning community (CLC) assignment. In this assignment, students will create a User Registration module and User Login module for a professional networking site. The initial application framework with a menu navigation system should also be developed such that a user could navigate between the User Registration module and User Login module.

*Team discussion will be needed concerning:*

1. The components of the professional networking site
2. Possible user interface and menu system designs
3. User attributes to be captured during registration

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the registration and login processes.
2. Use the Laravel framework and MVC design pattern to create the necessary models, routes, and controllers to support User Registration and User Login. Write supporting service class methods that will enable the capturing of user input to store the information in database and support the login authentication. If necessary, use the functions you wrote in CST-236 as a starting point and refactor this functionality into object orientated classes. Standard PHP functions should not be used.
3. Use the Laravel framework and MVC design pattern to create the necessary views using HTML with CSS and JavaScript as needed. Build the minimally functional forms to capture user registration and login information.
4. Create the back-end business and data services to support the registration and login process. Use PHP classes to develop the services.
5. In PHP, perform the necessary user login logic using the database for authentication. Implement authentication as separate business and data service classes so you can reuse later (assess whether the classes you created in CST-236 can be reused or refactored). Before you code, discuss and decide every detail of the user experience such as: number of login trials allowed, constraints on the password, etc. Recall the similar discussion in CST-236 and improve the code and design.
6. Integrate the Registration and Login modules using a minimal application framework and navigation menu system.
7. Discuss any revisions that might be necessary to code, schema, tables, or user interface created in future modules.
8. Perform a peer code review with your team members ensuring that the code is readable, formatted, and maintainable across your team. Unused code (including code that is commented out) should be removed from the codebase.

**Deliverables**

A fully functional Registration Page and a fully functional Login Page, including:

1. Design report
2. All necessary SQL tables (with supporting DDL script)
3. All necessary PHP classes
4. All necessary HTML, CSS, and JavaScript
5. A fully functional User Login Module and User Registration Module
6. Screencast providing a demo of all new functionality and walk through of all new code

**Submission**

Submit the following to the learning management system:

1. The design report (with a URL of your screencast and GIT repository)
2. All application and database code hosted on an approved GCU Cloud Hosting Provider. Approved GCU Cloud Hosting Providers are listed in the Student Success Center. Hosting providers not listed are not approved by GCU and should not be used to host applications.
3. Code Documentation:
   1. In each source file include a commented header with the following information: project name and version, module name and version, programmer(s) name(s), date, a short synopsis of the module, and references.
   2. Class header comments for all non-framework developed code
   3. Method header comments for all new classes
   4. Method comments for all non-obvious code sections
4. All code checked in a GIT repository
5. Upload a zip file of all source code and supporting code artifacts (you may have to exclude large Laravel libraries from the zip file).

# Milestone 2: Member Profiles and Administration Modules

**Overview**

In this assignment, students will:

1. Create a Member Profile module that will allow a user to enter their user profile (demographic data, contact information, etc).
2. Create an Administration module that supports a management interface for the professional networking administrator to manage member roles, permissions, and their profiles.
3. Create initial administrative functionality to suspend (logical delete) a user and permanently (physical delete) remove a user should be implemented.
4. Continue to enhance the application by improving the menu navigation system, adding headers, footers, and a logo.

*Team discussion will be needed concerning:*

1. The attributes of a professional profile
2. How the administrator interface will visually fit with the rest of the professional networking site project
3. The various types of members, their permissions, and the reason those permissions levels were decided upon
4. How a company might post a job ad on this site. In order to support this, discuss what would be the minimal elements of information you would need to store, generate, and display.
5. How the new functionality might work with existing functionality; do not hesitate to modify prior modules.

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required by the member and profiles administrator. Modify the schema, the E-R diagrams, 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 split existing tables as well.
2. Using the Laravel framework and MVC design pattern to create the necessary models, routes, and controllers to support Member Profile and Administration functionality. The Administration module should enable the management of member data, their permissions, and profiles components. Modify the User Registration Page to accommodate the new roles and set a default role (if needed).
3. Use the Laravel framework and MVC design pattern to create the necessary views using HTML with CSS and JavaScript as needed. Build the minimally functional forms to capture user profile and administrative information. Use HTML and Blade Templates to create the views. Using CSS, style and lay out the view appropriately.
4. Create the back-end business and data services. Decide what data should be accessed and displayed based on the project requirements and your design. Use PHP classes to develop the services.
5. Create the back-end business and data services to support the Administrative functionality. The Administrator should be able to suspend (logical delete) a user and also permanently (physical delete) a user. Use PHP classes to develop the services.
6. Integrate the Registration, Login, and Profile modules using an application framework and navigation menu system.
7. Discuss any revisions that might be necessary to code, schema, tables, or user interface created in future modules.
8. Perform a peer code review with your team members ensuring that the code is readable, formatted, and maintainable across your team. Unused code (including code that is commented out) should be removed from the codebase.

**Deliverables**

A fully functional Member and Profiles Administrator Page, including

1. Design report
2. All necessary SQL tables (with supporting DDL script)
3. All necessary PHP classes
4. All necessary HTML
5. All necessary additional code and services
6. A fully functional Member Profile Module and Admin Module
7. Screencast providing a demo of all new functionality and walk through of all new code.

**Submission**

Submit the following to the learning management system:

1. The design report (with a URL of your screencast and GIT repository).
2. All application and database code hosted on an approved GCU Cloud Hosting Provider. Approved GCU Cloud Hosting Providers are listed on the Student Success Center. Hosting providers not listed are not approved by GCU and should not be used to host applications.
3. Code documentation:
   1. In each source file, include a commented header with the following information: project name and version, module name and version, programmer(s) name(s), date, a short synopsis of the module, and references.
   2. Class header comments for all non-framework developed code
   3. Method header comments for all new classes
   4. Method comments for all non-obvious code sections
4. All code checked in a GIT repository
5. A zip file of all source code and supporting code artifacts (you may have to exclude large Laravel libraries from the zip file)

# Milestone 3: Member E-Portfolio Module

**Overview**

In this assignment, students will:

1. Create a Professional Member E-Portfolio module that includes a user's job history, skills, and education.
2. Implement new administrative functionality to add new job postings, update existing job postings, and delete existing job postings.
3. Add business data validation to support all form data entry.

*Team discussion will be needed concerning:*

1. The interface, specific functions, and necessary backend support
2. How the e-portfolio functionality and new admin functionality will visually fit with the rest of the professional networking project
3. How the storage and display of multiple job history, skills, and education records will be designed
4. The need to include various elements in an e-portfolio against the technical requirements for their storage and display, being mindful of security and copyright concerns
5. How e-portfolio elements can be leveraged by individual members, discovered by potential employers, and be mined for information
6. How the new functionality might work with existing functionality (do not hesitate to modify prior modules)
7. Any limitations you might impose on the system

**Execution**

Execute this assignment according to the following guidelines:

1. In MySQL, build the necessary tables to store the information required during the e-portfolio creation process. Be mindful of file sizes and copyright.
2. Use the Laravel framework and MVC design pattern to create the necessary models, routes, and controllers to implement all e-portfolio and new admin management functionality, with a special emphasis on categorization (tagging) for job search and discovery.
3. For all forms in the application add the appropriate business data validation rules using the Laravel framework.
4. In MySQL, also build the necessary tables to store the information required during the e-portfolio display process to reflect new information required by the administrator. Modify the schema, the E-R diagrams, the tables, the keys, and the relationships accordingly. Also, write the appropriate database connectivity and interaction classes necessary to manage the e-portfolios and the components within each e-portfolio.
5. Use the Laravel framework and MVC design pattern to create the necessary views using HTML with CSS and JavaScript as needed. Design considerations should be given to how to properly store and display the user's job history, skills, and education, which will likely be multiple records.
6. Create the back-end business and data services to support the new e-portfolio and updated administrative functionality. The administrator should be able to suspend (logical delete) a user and also permanently (physical delete) a user. Use PHP classes to develop the services.
7. Integrate the e-profile and updated administration modules using an application framework and navigation menu system.
8. Discuss any revisions that might be necessary to code, schema, tables, or user interface created in future modules.
9. Perform a peer code review with your team members ensuring that the code is readable, formatted, and maintainable across your team. Unused code (including code that is commented out) should be removed from the codebase.

**Deliverables**

A fully functional e-portfolio page, including:

1. Design report
2. All necessary SQL tables (with supporting DDL script)
3. All necessary PHP classes
4. All necessary HTML
5. All other necessary code
6. A fully functional e-profile and administrative module
7. Screencast providing a demo of all new functionality and walk thru of all new code

**Submission**

Submit the following to the learning management system:

1. The design report (with a URL of your screencast and GIT repository)
2. All application and database code hosted on an approved GCU Cloud Hosting Provider. Approved GCU Cloud Hosting Providers are listed on the Student Success Center. Hosting providers not listed are not approved by GCU and should not be used to host applications.
3. Code documentation:
   1. In each source file include a commented header with the following information: project name and version, module name and version, programmer(s) name(s), date, a short synopsis of the module, and references.
   2. Class header comments for all non-framework developed code
   3. Method header comments for all new classes
   4. Method comments for all non-obvious code sections
4. All code checked in a GIT repository
5. A zip file of all source code and supporting code artifacts (you may have to exclude large Laravel libraries from the zip file)

# Milestone 4: Affinity Groups Module

**Overview**

In this assignment, students will:

1. Create an Affinity Groups module based on common interests with the ability to add, update, and delete an Affinity Group. Members should be able to add themselves and remove themselves from an existing Affinity Groups as well as see what other Members are in an Affinity Group.
2. Enhance the security of the application by adding page level security, such that a user cannot access functionality outside of Registration and Login modules without first being logged into the application.
3. Improve the usability of the application by removing the default 'public' URI that is configured within Laravel.

*Team discussion will be needed concerning:*

1. Research on existing networking sites, such as LinkedIn, to determine how Affinity Groups are supported.
2. How the Affinity Group functionality will visually fit with the rest of the professional networking project.
3. How the storage and display of multiple Affinity Group records and associated members will be designed.
4. Any limitations you might impose on the system.
5. How to 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 automatic creation process of affinity groups. Minimize duplication of information stored. Support the ability of a member to be part of multiple affinity groups.
2. Using the Laravel framework and MVC design pattern to create the necessary models, routes, and controllers to support the Affinity Group functionality, with methods necessary to implement all Affinity Groups functionality.
3. Use the Laravel framework and MVC design pattern to create the necessary views using HTML with CSS and JavaScript as needed. Design considerations should be given to how to properly store and display Affinity Groups, which will likely be multiple records.
4. Create the back-end business and data services to support the Affinity Group functionality.
5. For all forms in the application add the appropriate business data validation rules using the Laravel framework.
6. Integrate the Affinity Groups modules using an application framework and navigation menu system.
7. Configure the Laravel framework to remove the default 'public' URI from the application.
8. Discuss any revisions that might be necessary to code, schema, tables, or user interface created in future modules.
9. Perform a peer code review with your team members ensuring that the code is readable, formatted, and maintainable across your team. Unused code (including code that is commented out) should be removed from the codebase.

**Deliverables**

A fully functional Affinity Groups module, including:

1. Design report
2. All necessary SQL tables (with supporting DDL script)
3. All necessary PHP classes
4. All necessary HTML
5. All necessary additional code
6. Screencast providing a demo of all new functionality and walk thru of all new code

**Submission**

Submit the following to the learning management system:

1. The design report (with a URL of your screencast and GIT repository)
2. All application and database code hosted on an approved GCU Cloud Hosting Provider. Approved GCU Cloud Hosting Providers are listed on the Student Success Center. Hosting Providers not listed are not approved by GCU and should not be used to host applications.
3. Code documentation:
   1. In each source file include a commented header with the following information: project name and version, module name and version, programmer(s) name(s), date, a short synopsis of the module, and references.
   2. Class header comments for all non-framework developed code
   3. Method header comments for all new classes
   4. Method comments for all non-obvious code sections
4. All code checked in a GIT repository.
5. A zip file of all source code and supporting code artifacts (you may have to exclude large Laravel libraries from the zip file).

# Milestone 5: Job Match Module

**Overview**

In this assignment, students will:

1. Create the ability to search for a job posting. A Job search should support a keyword search by Job Title and Job Description with the Job search results displayed in a HTML formatted table. The Job search results page should display a hyperlink on one of the fields (like the Job Title) that when clicked will navigate the user to a Job details page.
2. Emulate the flow to apply for a Job Posting.

*Team discussion will be needed concerning:*

1. Browsing the Internet to review several professional networking pages that focus on those that have job ads and job recommendation features to develop a feel for what matching jobs to profiles entails and looks like.
2. The user experience and the elements and functions required to implement a matching algorithm. In addition:
   1. What would be matched and constitutes a good match.
   2. The interface, specific functions, and necessary backend support for the automated matching process (i.e., mining all relevant data).
   3. How the matching between jobs and profiles and the resulting display interfaces will visually fit with the rest of the affinity groups project.
3. How full word and partial word searches would be implemented in a SQL query.
4. How to handle the possibility that a large number of records could be returned from a database query.
5. How to avoid possible SQL injection vulnerabilities.
6. Any limitations you might impose on the system. 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 automatic creation process of matches between profiles and jobs. Support the ability of a member to be matched with multiple jobs and sort the displayed results using multiple sort criteria (user chooses).
2. Using the Laravel framework and MVC design pattern to create the necessary models, routes, and controllers to support a Job Search Criteria form.
3. Use the Laravel framework and MVC design pattern to create the necessary views using HTML with CSS and JavaScript as needed. Design considerations should be given to how to properly display a potentially large number of records that could be returned from a database search.
4. Create the back-end business and data services to support the Job Search functionality. If the number of search results is greater than N (let the students pick N) then N number of Jobs should be displayed along with a warning message that the user should refine their search criteria. The design should also handle the scenario if no search criteria is specified by the user. Either a proper data validation message must be displayed and not a query that returns every Job Posting be returned. Students should also ensure their design is not susceptible to SQL Injection vulnerabilities.
5. For all forms in the application add the appropriate business data validation rules using the Laravel framework.
6. Integrate the Job Search module using an application framework and navigation menu system.
7. Discuss any revisions that might be necessary to code, schema, tables, or user interface created in future modules.
8. Perform a peer code review with your team members ensuring that the code is readable, formatted, and maintainable across your team. Unused code (including code that is commented out) should be removed from the codebase.

**Deliverables**

A fully functional Job Match Module, including:

1. Design report
2. All necessary SQL tables (with supporting DDL script)
3. All necessary PHP classes
4. All necessary HTML
5. All necessary additional code
6. Screencast providing a demo of all new functionality and walk thru of all new code

**Submission**

Submit the following to the learning management system:

1. The design report (with a URL of your screencast and GIT repository)
2. All application and database code hosted on an approved GCU Cloud Hosting Provider. Approved GCU Cloud Hosting Providers are listed on the Student Success Center. Hosting providers not listed are not approved by GCU and should not be used to host applications.
3. Code Documentation:
   1. In each source file include a commented header with the following information: project name and version, module name and version, programmer(s) name(s), date, a short synopsis of the module, and references.
   2. Class header comments for all non-framework developed code
   3. Method header comments for all new classes
   4. Method comments for all non-obvious code sections
4. All code checked in a GIT repository.
5. A zip file of all source code and supporting code artifacts (you may have to exclude large Laravel libraries from the zip file.)

# Milestone 6: Web API Module

**Overview**

In this assignment, students will design and implement public REST APIs that will return a desired Member Profile, return all Job Postings, and return a desired Job Posting. Authenticated access to the REST APIs does not have to be considered and these can be anonymous APIs.

*Team discussion will be needed concerning:*

1. Research on existing sites that have published REST APIs. Discuss the format, layout, and content for existing REST APIs have been documented.
2. The design of each API: your design should include the API description, what URL is used to invoke the API, what HTTP parameters are required, what data validation messages that could be returned, what error messages that could be returned, descriptions for each JSON data attribute that will be returned, and the HTTP status codes that will be returned.
3. Limitations you might impose on the system.

**Execution**

Execute this assignment according to the following guidelines:

1. Using the Laravel framework and RESTful Controller, implement each of the REST APIs per your design. You should leverage existing business and data services in your implementation.
2. Using PHP implement a DTO using the DTO design pattern that will act as a container for returning all data, status messages, and error message.
3. Ensure data validation is considered in the implementation.
4. Ensure error handling is considered in the implementation.
5. Ensure the scenario of handling large return data sets is considered. A warning message along with the "clipped" return pay loud should be implemented, such that a user cannot retrieve greater than N Job Postings.
6. Consider using a Test Tool, such as Postman, that can be used to regression and functionally test the APIs. **Note:** Refer to the "CST-256 How to Guides," for installation instructions.
7. Integrate the Web API module into the application codebase.
8. Discuss any revisions that might be necessary to code, schema, tables, or user interface created in future modules.
9. Perform a peer code review with your team members ensuring that the code is readable, formatted, and maintainable across your team. Unused code (including code that is commented out) should be removed from the codebase.

**Deliverables**

A fully functional set of REST APIs, including:

1. Web API Design Specification. Web API Design Specification using Word (or Swagger). Consider using design Tools, such as Swagger, versus documenting the API designs using a Word document.
2. The design report
3. Screencast providing a demo of all new functionality and walk thru of all new code

**Submission**

Submit the following to the learning management system:

1. The design report (with a URL of your screencast and GIT repository)
2. The Web API Specification.
3. All application and database code hosted on an approved GCU Cloud Hosting Provider. Approved GCU Cloud Hosting Providers are listed on the Student Success Center. Hosting Providers not listed are not approved by GCU and should not be used to host applications.
4. Code Documentation:
   1. In each source file include a commented header with the following information: project name and version, module name and version, programmer(s) name(s), date, a short synopsis of the module, and references.
   2. Class header comments for all non-framework developed code.
   3. Method header comments for all new classes
   4. Method comments for all non-obvious code sections
5. All code checked in a GIT repository.
6. A zip file of all source code and supporting code artifacts (you may have to exclude large Laravel libraries from the zip file).

# Benchmark – Milestone 7: Professional Networking Application

**Overview**

*This assignment serves to benchmark competency 2.2: Manage database systems that reflect institutional/project needs.*

In this assignment, students will revise, improve, and complete their final project. Enhancements to the application include the integration of Page Security using Laravel Middelware, demonstration of the use of IoC using a Service Provider (example, on a a Login Controller), and cleanup of the application to include proper Exception Handling and common Error Pages.

*Team discussion will be needed concerning:*

1. The test conditions and test data required to full functionally test the application and Web APIs
2. What needs to be cleaned up in the Design Report and Code.

**Execution**

Execute this assignment according to the following guidelines:

1. Using the "Test Case Template," located within the Course Materials, create a test plan and test cases to fully functionally test the Application and APIs.
2. Cleanup the design report and code as necessary.
3. Execute the test plan and test cases, updating the "Test Case Template" with the execution status for each test case. Note: The test plan and test cases should be executed prior to submitting the final project.

**Deliverables**

1. Final design report to ensure the design matches the implementation
2. Final API Design Specification to ensure the design matches the implementation
3. Test plan and test case executed (with test results)
4. All necessary SQL tables (with supporting DDL script)
5. All necessary PHP code
6. All necessary HTML
7. All necessary additional code
8. A fully functional database
9. A list of known bugs
10. A list of suggested implementations for the next version

**Submission**

Submit the following to the learning management system:

1. The final design report (with a URL of your screencast and GIT repository)
2. The final WEB API Specification
3. The test plan and test cases with final test results
4. All application and database code hosted on an approved GCU Cloud Hosting Provider. Approved GCU Cloud Hosting Providers are listed on the Student Success Center. Hosting providers not listed are not approved by GCU and should not be used to host applications.
5. Code Documentation:
   1. In each source file include a commented header with the following information: project name and version, module name and version, programmer(s) name(s), date, a short synopsis of the module, and references.
   2. Class header comments for all non-framework developed code
   3. Method header comments for all new classes
   4. Method comments for all non-obvious code sections
6. All code checked in a GIT repository
7. A zip file of all source code and supporting code artifacts (you may have to exclude large Laravel libraries from the zip file.)