www.ijonest.net

Course and Faculty Management System

Christopher Grime
University of La Verne, USA

Jozef Goetz
University of La Verne, USA

To cite this article:

Grime, C. & Goetz, J. (2023). Course and Faculty Management System. *International Journal on Engineering, Science, and Technology (IJonEST), 5*(2), 138-160. https://doi.org/10.46328/ijonest.163

International Journal on Engineering, Science, and Technology
2023, Vol. 5, No. 2, 138-160 https://doi.org/10.46328/ijonest.163

Course and Faculty Management System

Christopher Grime, Jozef Goetz

Article Info

Article History

Received:
01 January 2023
Accepted:
14 April 2023

Keywords

Engineering
University
Education
Management system
Software development

Abstract

The purpose of the paper is to show how to expand the low code interactive framework in order to develop a web app for the broad needs of different fields. The goal of this work is to give a chance to computer science senior project students to work on a broad spectrum of projects using Apache, HTML, CSS, JavaScript, PHP, and MySQL. In this paper the web app is Course and Faculty Management System allowing educational organizations efficiently organize and manage their courses and faculty. The web application is interactive, responsive, secured, password and database driven app. The website is accessible on all devices. Public users can view the courses the university offers as well as which instructors teach them. Courses can be filtered by term and subject. Users that are logged in and have the correct permissions can edit/add/delete courses, departments, and instructors in the database. Admins can import files containing information about multiple courses as well as export the courses in the database to a csv file. The admin can also edit the other user's permissions. In conclusion, the project has been successfully designed and implemented according to best practices and finally tested on a web hosting server provider.

Introduction

Keeping track of different entities is an important task and can be very useful for monitoring events. The project [Miranda-Hill, W., Goetz, J. (2019, June 30 – July 4)] aims to prototype the functionality of a user-generated geospatial meteorology map (for keeping track of temperature and pressure) based on low code interactive framework [Butler, T., & Yank, K. (2017)]. This includes the design and implementation of a database driven website with a public and a password protected admin component, in addition to a database, web server and hardware component. Another project Patient Care Reporting App [Guarrera, A., Goetz, J. (2022, *May* 10)] is based on low code interactive framework [Butler, T., & Yank, K. (2017)] as well. The purpose of the app was to provide a platform to simulate an electronic patient care reporting system for the students to interact with. Additional project [Flores Marquez, A., Goetz, J. (2023)] based on low code interactive framework [Butler, T., & Yank, K. (2017)] is Certificate Management Application that the goal was to develop a web application that keeps track of recipients of computer science certificates by managing (viewing, adding, editing, deleting) certificates, recipients, courses, categories and setting the users permissions.

The basic goal of Course and Faculty Management System is to keep track of courses and faculty by managing (viewing, adding, editing, deleting) courses, instructors, subjects, departments, timeslots, and course attributes

with given constrains for each entity. Course and Faculty Management System is a custom data management system designed to meet the needs of universities. Data management systems help companies and organizations organize and manage large data sets. Course management systems, in particular are ways for universities to organize their course schedules. There are various systems available to universities to assist in creation of schedules, though many do not meet the requirements of the organization. These limitations require the scheduler to manually check for conflicts and errors in their schedule.

Smaller organizations often rely on the use of spreadsheets to create their schedules. Spreadsheets introduce additional opportunities for errors to arise. Some of the considerations that a scheduler must take into consideration include:

- Time constraints in the instructors' schedules
- Major courses cannot be offered at overlapping times that do not let students meet their requirements.
- Instructor's schedules cannot be overloaded
- Enough courses must be offered to meet the demand of interested students

Some examples of employee management systems that are popular in the industry today are Kronos, Banner and PeopleSoft. These particular programs expand upon the capabilities of traditional Enterprise Resource Planning systems in order to adapt to the unique nature of university management. The most advantageous outcomes of using such a system are allowing users to work from the same data at the same time, allowing the latest technology to be leveraged by the users, ease of data access, fewer points of security breaches, and improved working efficiency of the users [Kumar, M., Garg, A., & Kumar, A. 2021]. Additional outcomes of the improved efficiency provided by university management systems are better access to information to assist in planning and management, better services provided to the faculty and students, fewer business risks, increased income and decreased expenses [Soliman, Karia 2015]. These outcomes provide huge incentives to move to a university management system from older, outdated methods of resource and schedule management.

Course and Faculty Management System combines tasks that currently require multiple applications and data sources. The application improves the efficiency and accuracy of the university schedulers. These optimizations save the university hours of time and allows the university to make better informed decisions regarding their course offerings. The data stored in the Course and Faculty Management System could also be provided for other purposes. For example, the data could be used to assist in generating contracts of instructors. Storing the university's data in a management system such as this would allow analytics to be used to create reports that provide insights into many aspects of the university.

Problem Statement

The purpose of the paper is to show how to expand the low code interactive framework [Kevin Yank, Tom Butler 2017] in order to develop a web app for the broad needs of different fields. The low code framework (LCF) is to expand a framework based on PHP and MySQL for creating low cost, customized, and integrated Web based Course and Faculty Management System. Moreover, the framework should host many users which can have

access from different clients the same time. The basic goal of Course and Faculty Management System is to keep track of courses and faculty by managing (viewing, adding, editing, deleting) courses, instructors, subjects, departments, timeslot, course attributes with given constrains for each entity.

The University of La Verne currently uses multiple systems to manage their course offerings and faculty members. These systems are unable to communicate with each other. The university often has the need to resort to using spreadsheets to consolidate the data and create schedules. This process requires manually gathering and cleaning data from multiple sources. This is a time consuming and error prone task. Errors can range from simple mistyping of data to miscalculating the number of courses an instructor has.

The purpose of the project is to build Course and Faculty Management System from scratch and solve those problems as well by allowing mass import of course data. When course data is imported, Course and Faculty Management System automatically checks for errors and conflicts with existing data.

The custom-built Course and Faculty Management System solves these problems by allowing mass import of course data. When course data is imported, Course and Faculty Management System automatically checks for errors and conflicts with existing data. This will ensure that there are no conflicts in an instructor's schedule. Mass importing and exporting eliminates human error when entering data into the Course and Faculty Management System.

Proposed expansions to the Course and Faculty Management System include further automation of schedule generation and analytical report writing. The inclusion of report writing to the Course and Faculty Management System would eliminate a time-consuming process of manually creating reports while being precise and accurate. The reports generated would allow the university to make informed decisions about faculty, courses, and students that benefit the university, faculty and students. An additional future goal of the Course and Faculty Management System is to provide an efficient way for the University to create instructor contracts.

System Design

Design Overview

The Course and Faculty Management System is a full-stack web based application consisting of three main components. The three components are the user interface, the server, and the database. The technology stack currently used by Course and Faculty Management System is HTML, CSS, JavaScript, PHP, and MySQL.

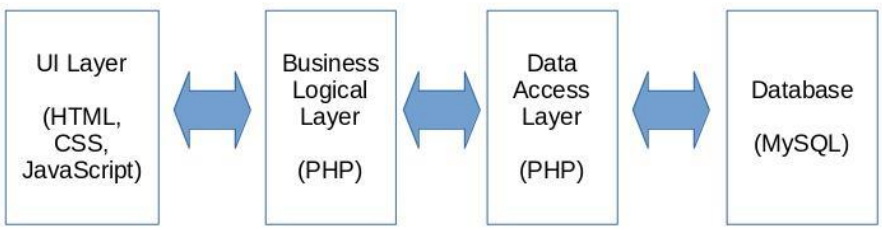


Figure 1. Framework Logical Layers (FLL)

The framework logical layer (FLL) architecture consists of three functional layers (see Figure 1):

- *User Interface, Presentation Layer* (*UI)*
- *Business Logical Layer (BLL)*
- *Data Access Layer (DAL)*

The front-end, user interface (UI) layer of this application is to present web pages on the client side using HTML, CSS, and JavaScript generated on the server side. The user interface allows users to view, add, update, and delete data relating to courses and instructors. The components of the web application interface are dynamically created using the template engine provided by the PHP framework. The application also includes a permissions-based access system. The permissions system is used to generate and display only the relevant user interface elements required by the active user.

The back-end, server components (BLL and DAL) of the application is coded in PHP. The server application is built on a PHP framework. This framework provides methods to connect to a MySQL database and perform queries on the connected database. The server application performs calculations on data and also adds, deletes or modifies data in the database. The system prevents malicious users from performing SQL injection to run undesired code in the web application. The PHP framework provides a PHP routing system that allows the server to handle GET, POST, and PUT requests. The framework is robust and flexible allowing a large variety of web applications to be built with it.

The last component is a MySQL Server as its Relational Database Management System (RDBMS) but it's considered a third party application and not a logical layer [Azma, H., Goetz, J. 2007]. *RDBMS is used* to store data. The web application's back-end connects to the database and interacts with it using SQL queries. The foundation for this functionality is provided by the framework and expanded in the Course and Faculty Management System.

The functional diagram in Figure 2 shows the admin user interaction with any admin web page.

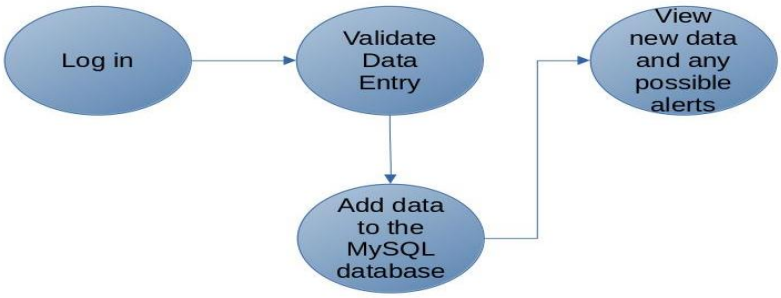


Figure 2. The Functional Diagram shows the Admin User Interaction with the System.

User Interface Layer

The user interface uses HTML for the structure of the application's pages, CSS for the styling and placement of

elements on the application, and JavaScript for dynamic elements within the application. The website conforms to the [Web Design Best Practices Checklist (2023)].

Course and Faculty Management System has two main components, the public and admin components. The public functionality and the main menu are shown in Figure 3. The user interface of Course and Faculty Management System is designed as a dashboard to function as a menu.

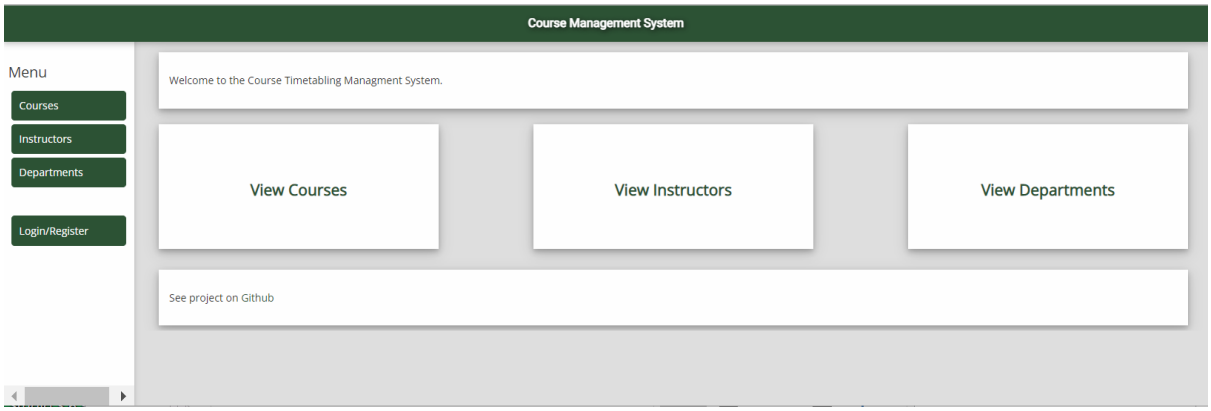


Figure 3. The Public Component of Course and Faculty Management System Menu

All public entities such as courses, instructors and departments are only available to view and don't have interactions buttons such as add, edit or delete (see Figure 4 - 6).

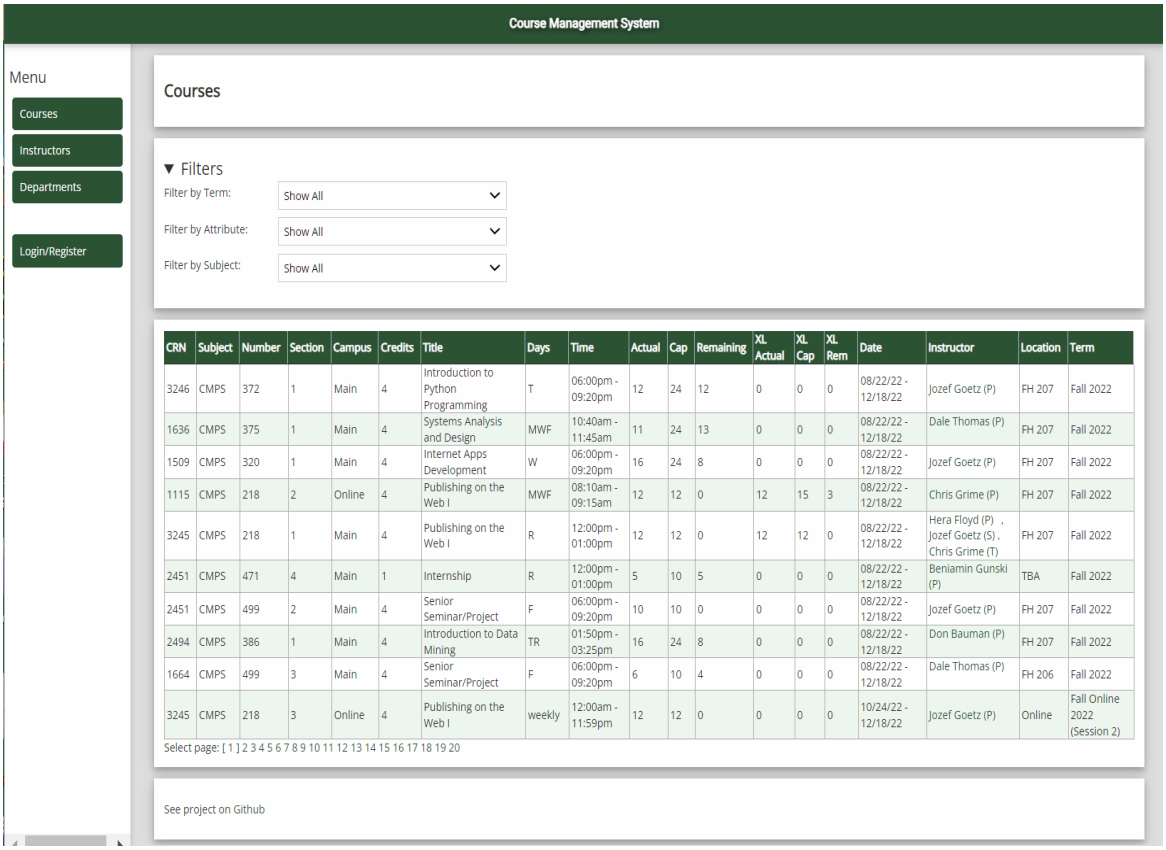


Figure 4. The Public View of the Courses Page

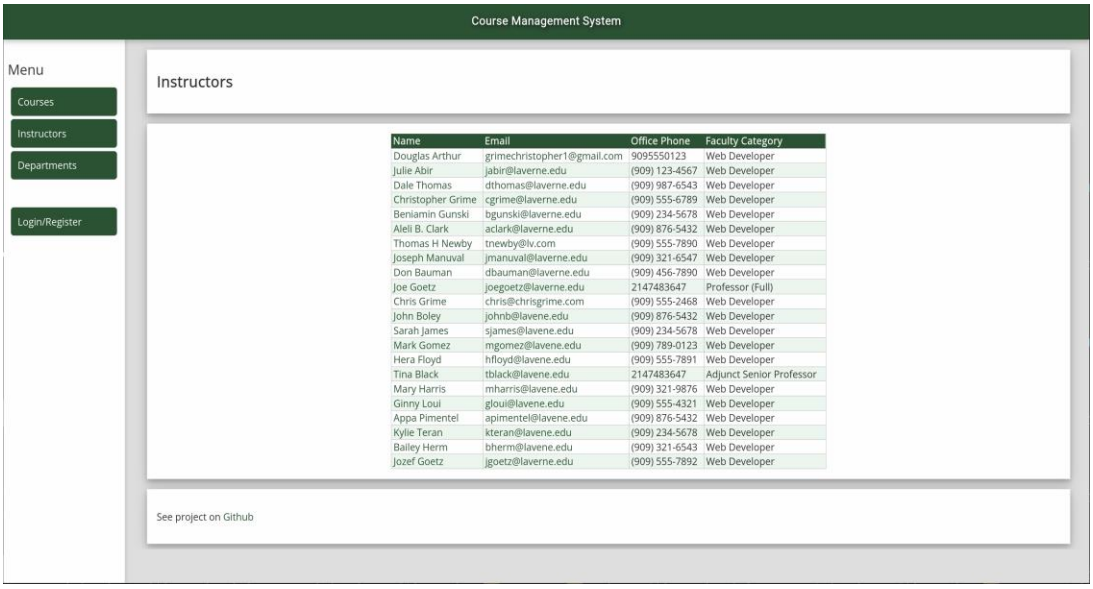


Figure 5. The Public View of the Instructors Page

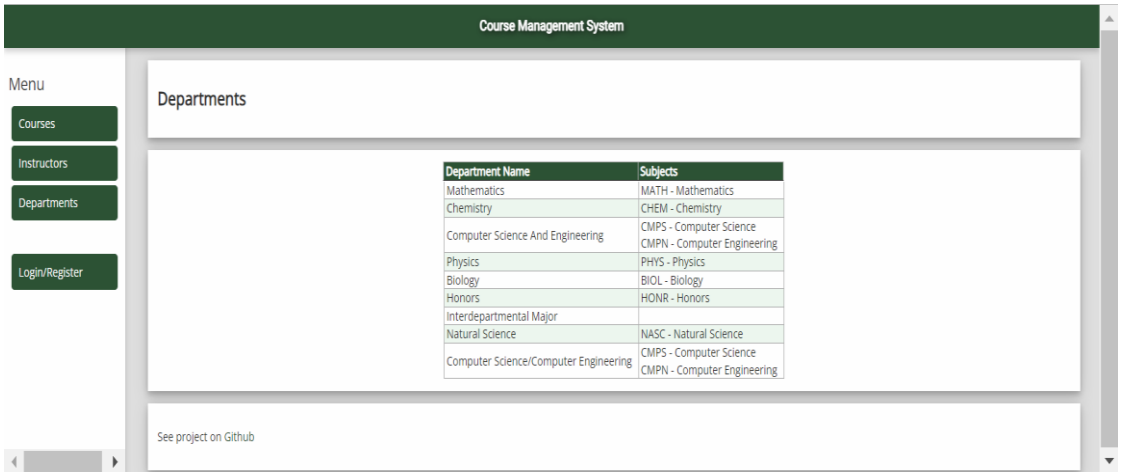


Figure 6. The Public View of the Departments Page

In order to have admin privileges the user needs to be registered (see Figure 7) and then proceed to login (see Figure 8).

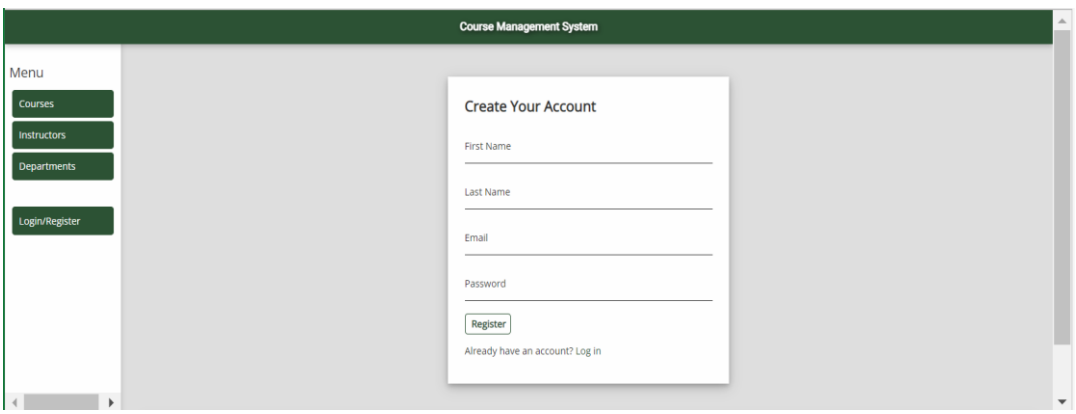


Figure 7. The User Registration Page

The main page, after the user is authenticated, has buttons for each of the main entities: courses, instructors, and departments. The user interface of Course and Faculty Management System consists of the following sections: header, navigation side bar menu, main, and footer. The buttons on the main page navigate to a certain entity. The functionality of each of the page changes based on the permissions of the logged in user. The just logged in user, for the first time, doesn't have ability to add, edit or delete any record for any entity (see Figure 4 - 6). The super admin user who has all permissions can change selectively all eight web page combinations of edit permissions (see Edit Permission in Figure 22). Figures 9, 11 – 12, 14 – 16, 18, 20 - 21, 23 show all adding, editing and deleting permissions. All pages are responsive to the display size of a smart phone, tablet and desktop.

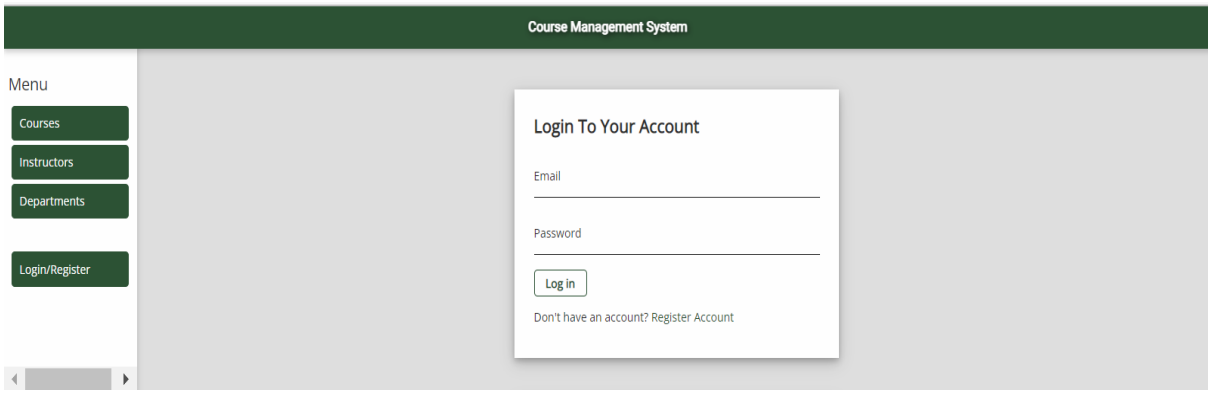


Figure 8. The Login Page

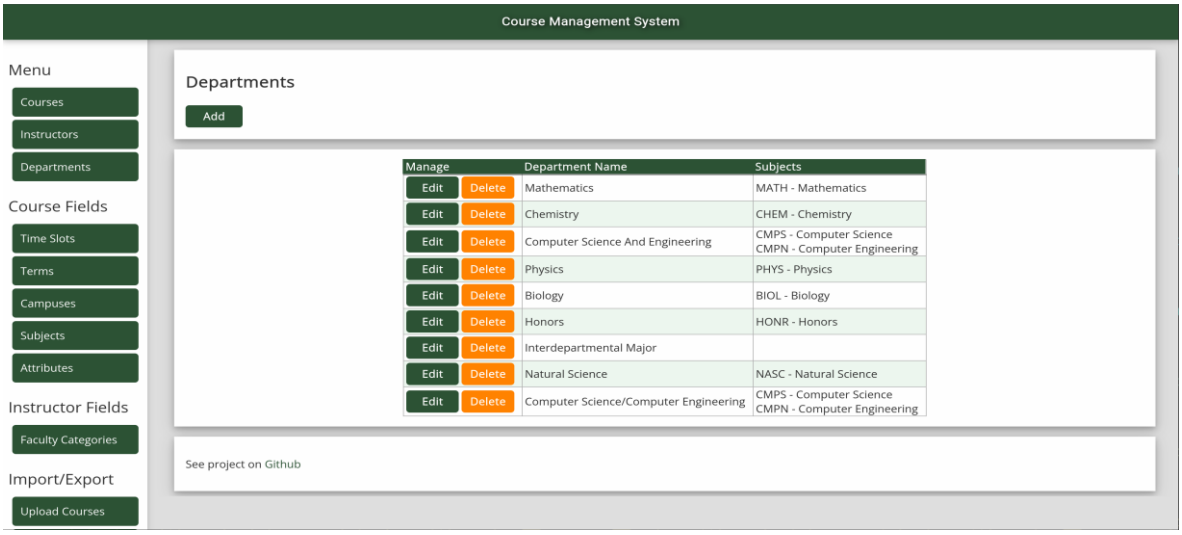


Figure 9. The Departments Page as a User with Editing Permissions

The courses page has the ability to mass import and export course data in a specific csv format (see Figure 10).

The courses page also includes filters to allow users to quickly navigate the large dataset of courses. The main portion of the courses page is dedicated to a table displaying courses and the courses' attributes concisely. The table also has pagination at the bottom to improve performance by reducing the amount of data that needs to be loaded at a single point in time (see Figure 11). Under the courses table any errors or warnings (e.g. a time conflict between the first row and the third row in Figure 12) that have been found by the system are displayed.

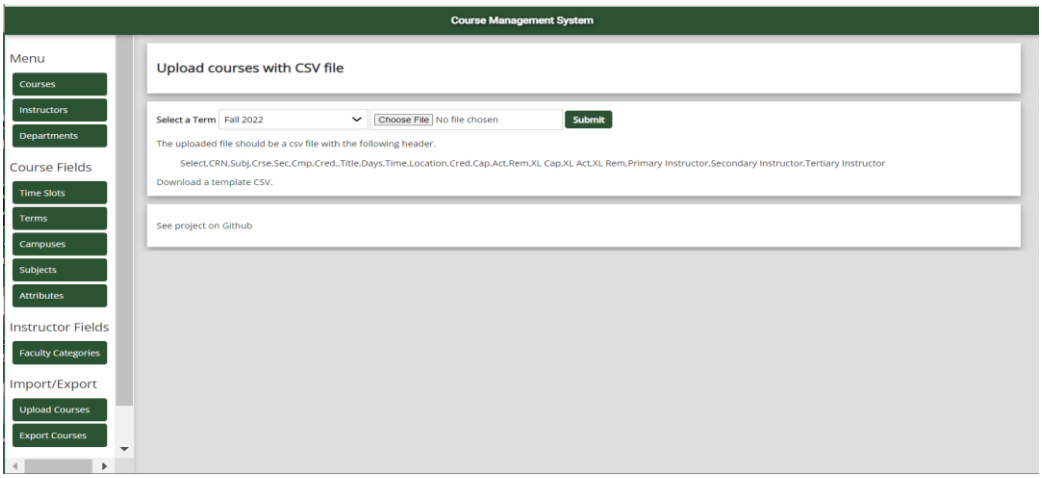


Figure 10. The Upload Courses Page

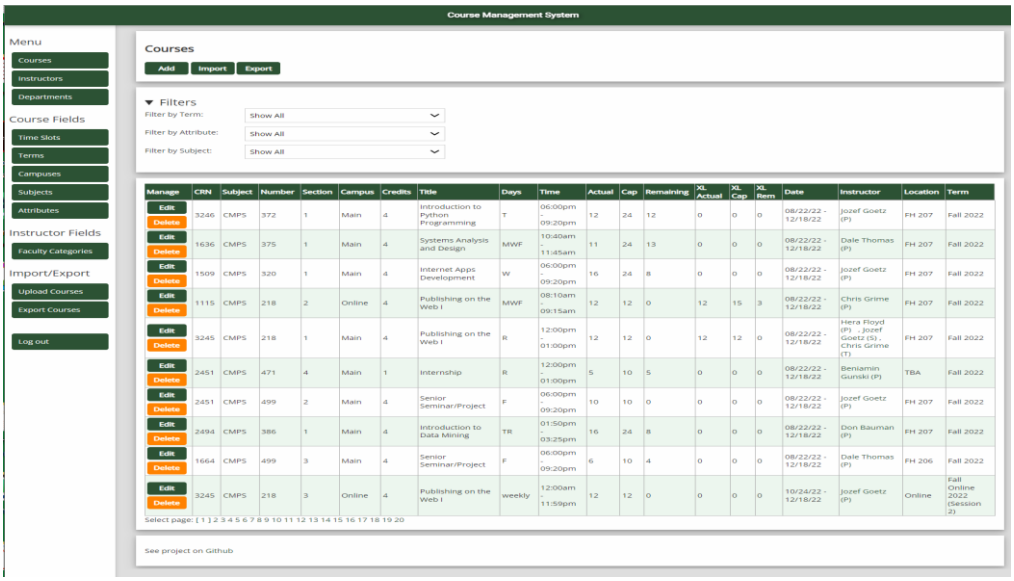


Figure 11. The Courses Page Displaying the Courses Data in a Table Format

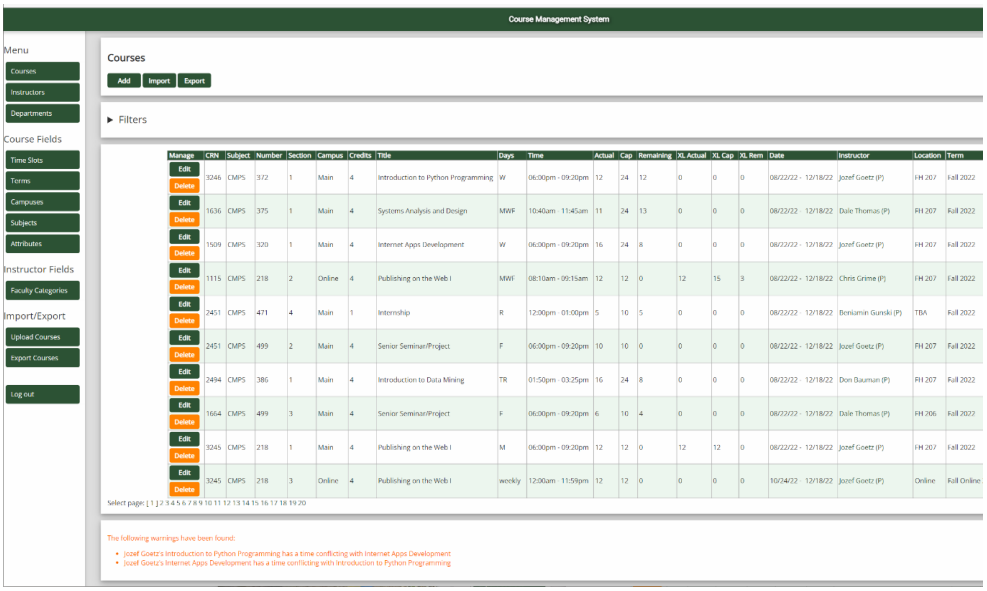


Figure 12. The Courses Page Displaying the Problems found in the Course Schedule List

The edit button allows editing of the selected course (see Figure 13).

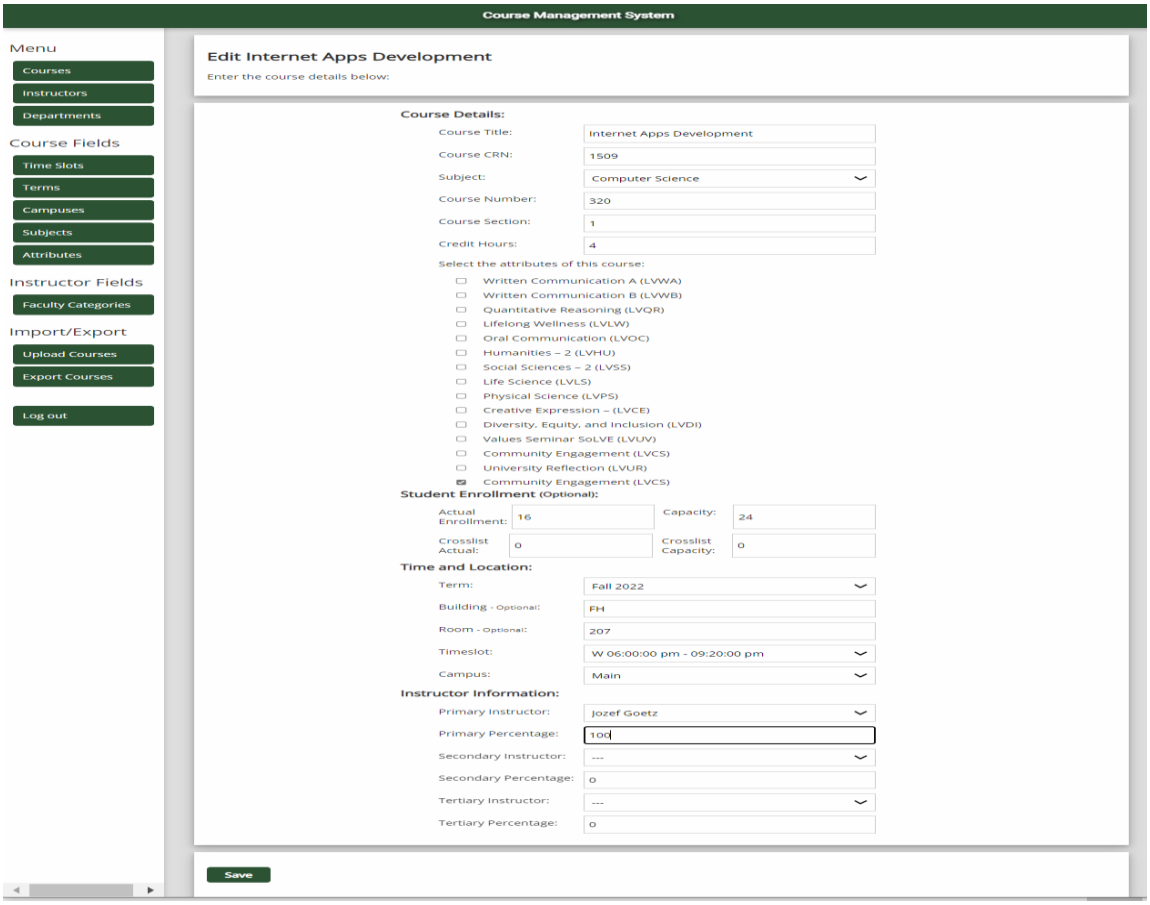


Figure 13. The Edit Courses Page

On the edit courses page (Figure 13), the course subjects, attributes, terms, timeslots, campuses in the drop-downs are editable on the side-bar menu after clicking the corresponding menu items and pressing the edit or delete buttons for the selected entities (see Figures 9, 11 – 12, 14 – 16, 18, 20 - 21, 23). The instructors in the drop-downs of the Instructor Information of Figure 13 are editable on the side-bar menu after clicking the instructor button (see Figure 21 – 23).

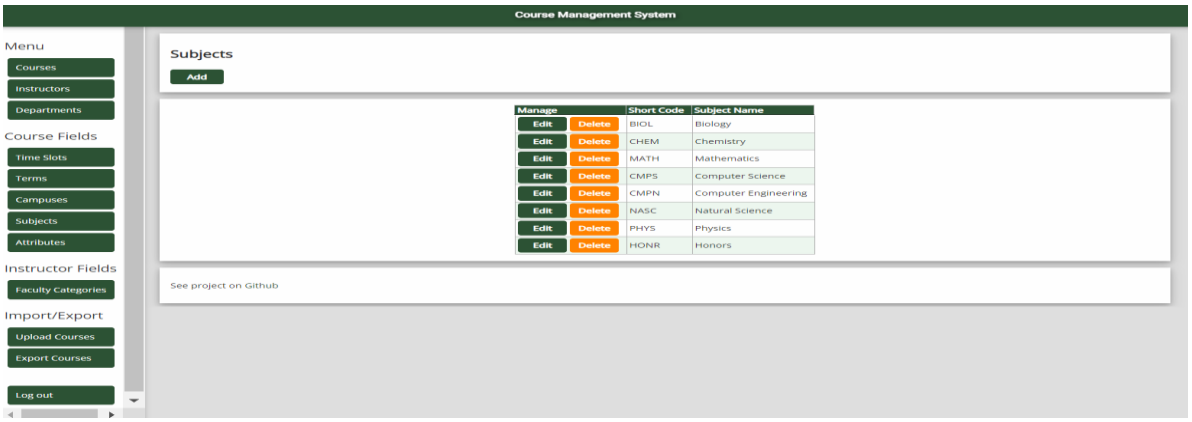


Figure 14. The Course Subjects Page

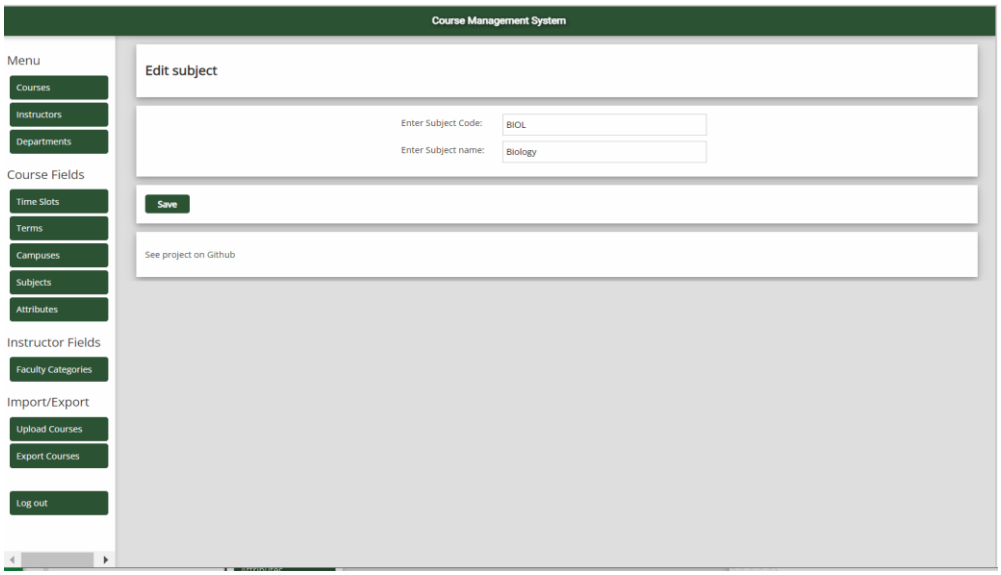


Figure 15. The Edit Course Subjects Page

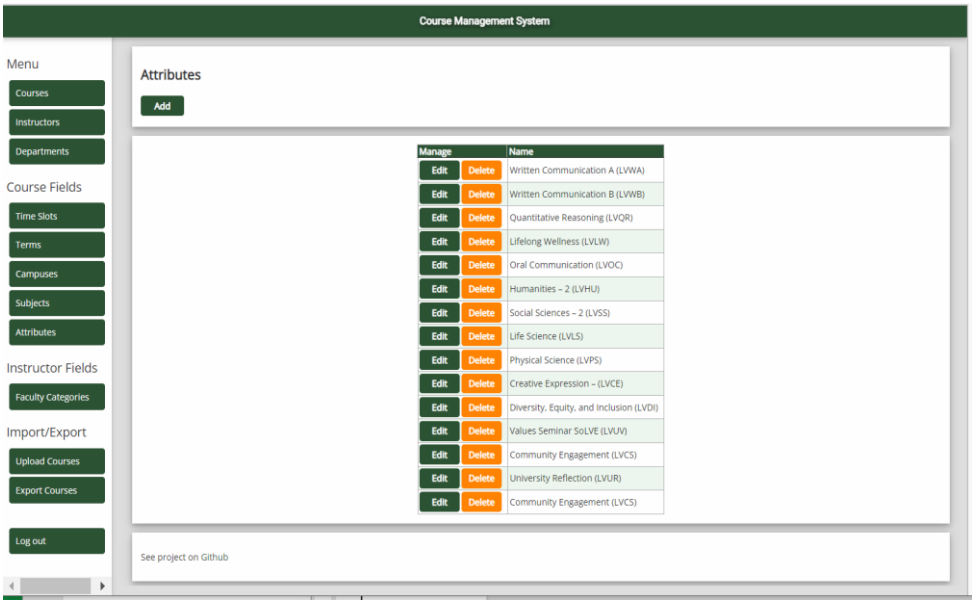


Figure 16. The Course Attributes Page

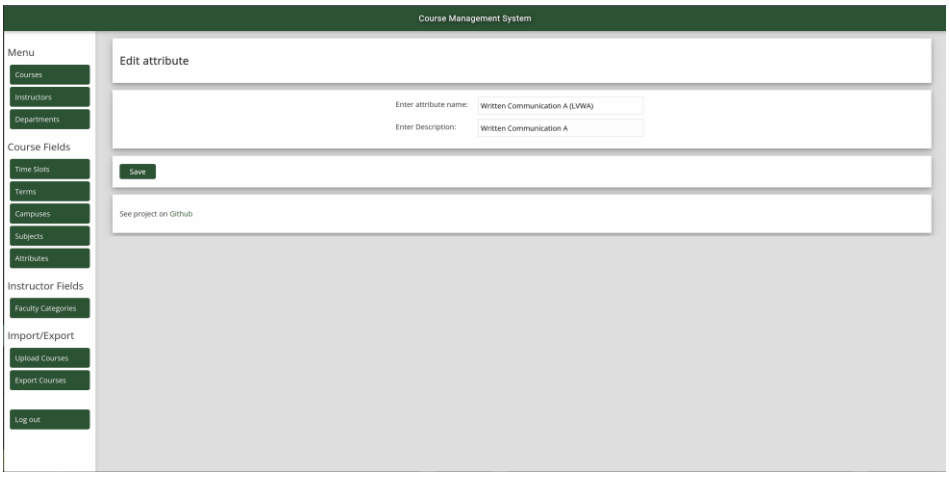


Figure 17. The Edit Course Attributes Page

The other entity pages follow the same pattern as courses for timeslots, terms and campuses. Each entity is organized into tables for a clean and organized view (see Figure 18 - 20).

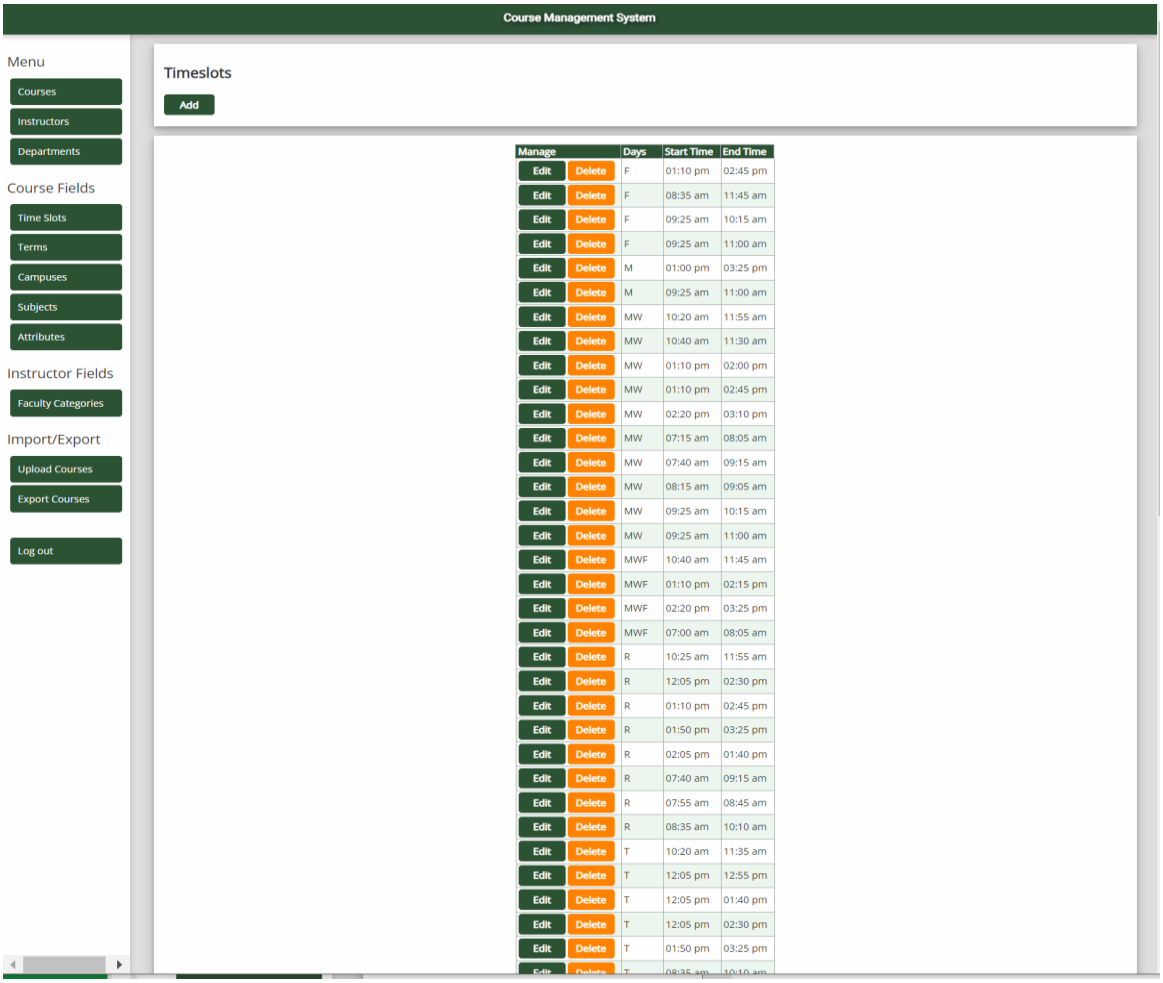


Figure 18. The Timeslots Page

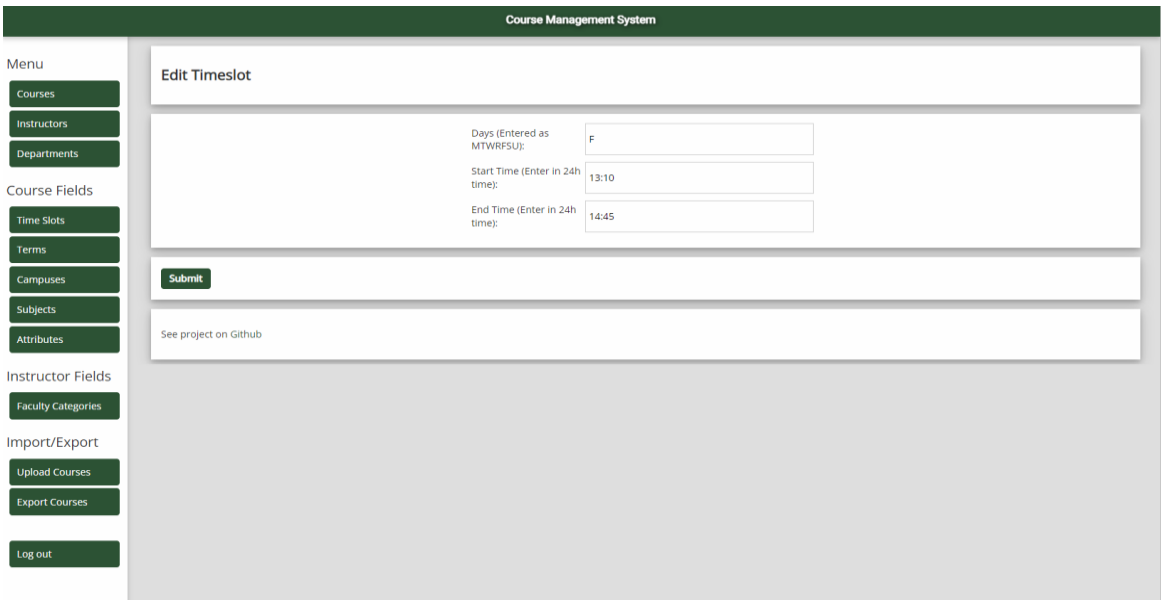


Figure 19. The Edit Timeslot Page

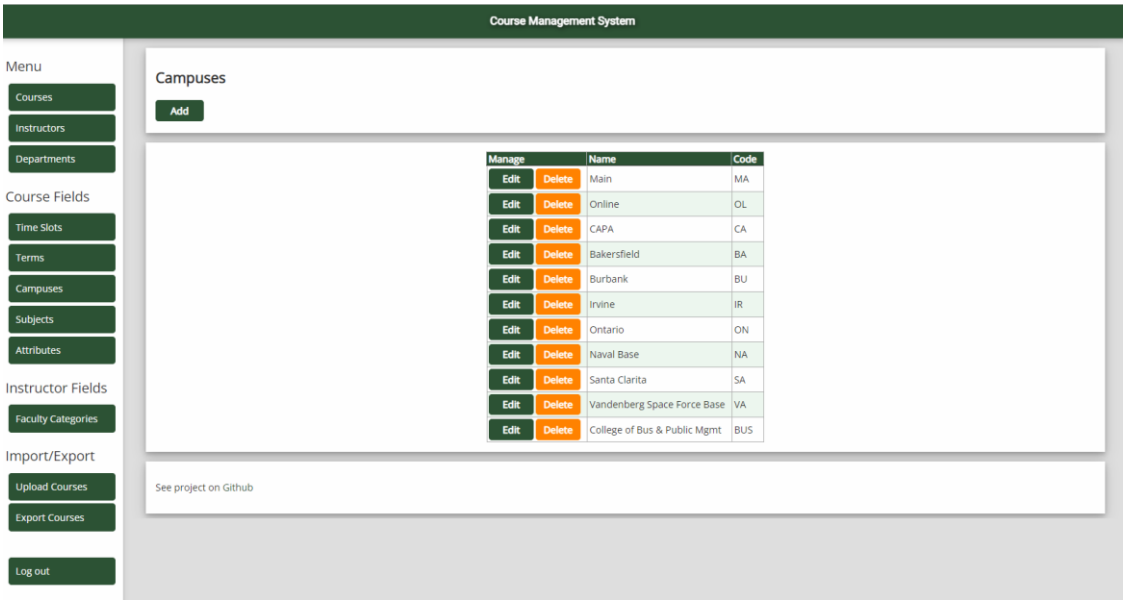


Figure 20. The Campuses Page

Similarly, the instructors are organized in a table with direct email links and phone numbers of the instructor (see Figure 21). Users with the permissions to edit instructors will be able to edit the instructor info by clicking the corresponding edit button.

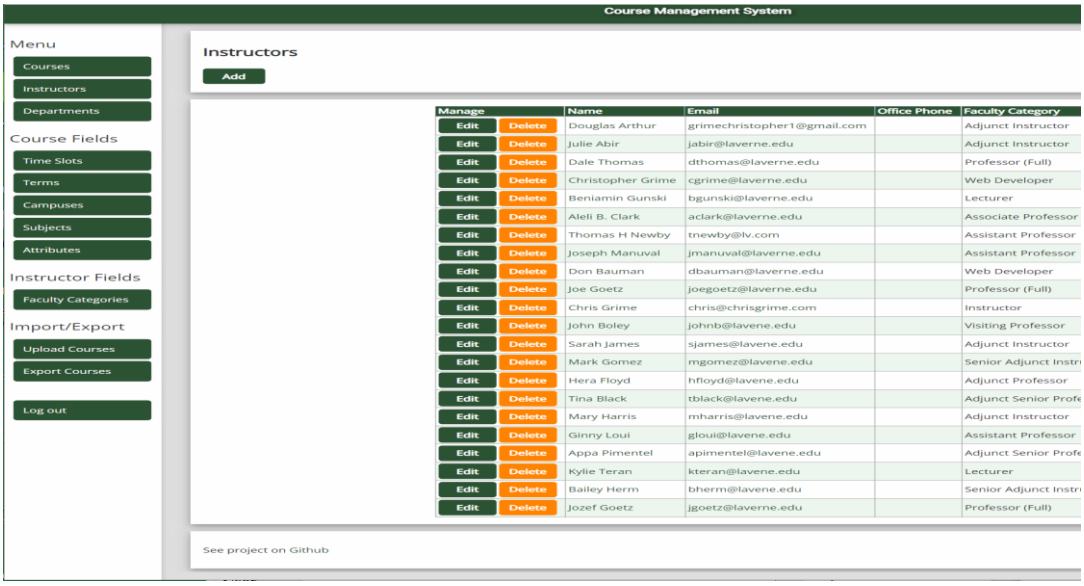


Figure 21. The Instructors List Page

Clicking the edit button of the instructors page the instructor's information can be displayed and can be edited permissions and other information (see Figure 22).

A unique feature to the instructor page is the ability to view instructor details on a details page. The instructor details page is available by clicking the instructor name on the instructor list page. The page shows information about the instructor and a list of the courses that instructor is assigned to as well (see Figure 23). The courses are

displayed in a table similar to the table on the courses page.

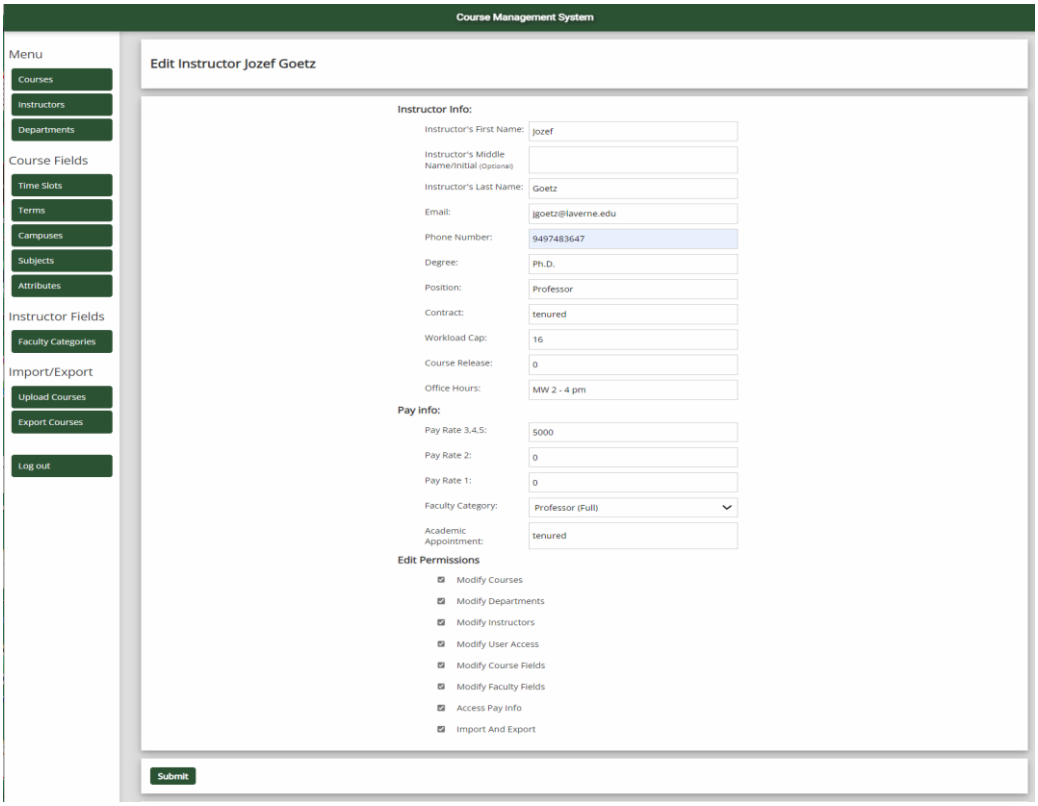


Figure 22. The Edit Instructor Page with Available Permissions

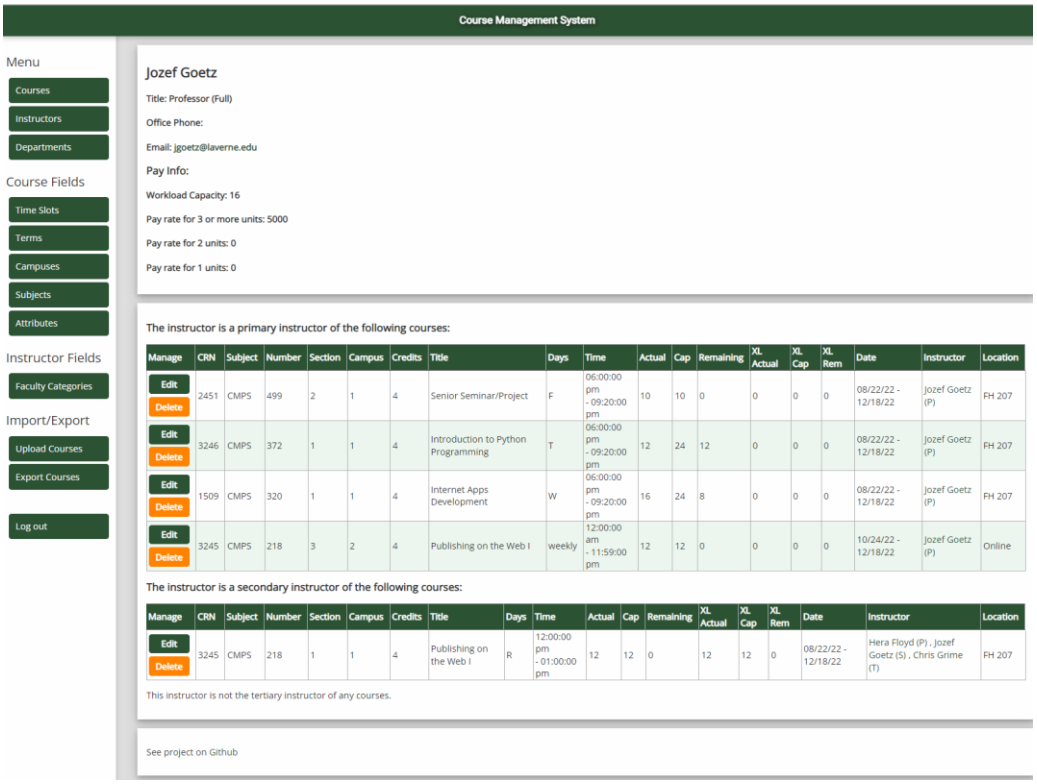


Figure 23. The Instructor Details Page, Displaying Info about the Selected Instructor

Business Logical Layer

The Business Logical Layer (BLL) consists of PHP generic and project specific classes where data are processed. A specific connection PHP class facilitates the communication with the Data Access Layer (DAL). The BLL hides the SQL statements calls from the UI Layer (see Figure 1). The low code framework provides methods in PHP generic classes to get and send SQL commands to the database. The application also uses GET, POST and PUT requests via a PHP routing system for accessing numerous elements.

Any user can have multiple permissions assigned to them. The permissions system allows for the admin of the site to grant only the necessary permissions to the users. The following are permissions that are currently available to be assigned to any user (see Figure 22).

- *Modify Courses* – Allows users to change course related data.
- *Modify Departments* – Allows admin to modify departments and
- *Modify Instructors* – Allows admin to change Fields related to instructors
- *Modify User Access* – Allows changing the permissions of users
- *Modify Course Fields* – Allow access to create update and delete fields related to courses such as: term, timeslot, attribute, and campuses
- *Modify Faculty Fields* – Allows the admin to create delete and edit the faculty categories
- *Access Pay Info* – Allows the admin to modify and view instructors' pay rates
- Import and Export – Allows the admin access to the mass import and export features.

Data Access Layer and Database

The low code framework (LCF) uses MySQL as a relational database management system (RDBMS). The database is normalized to improve performance and disk space utilization. The UI objects use the Business Logical Layer (BLL) to communicate with the Data Access Layer (DAL). The DAL connects to RDBMS and request the content that belongs in the web page. The RDBMS responds by sending the requested content to the DAL. The DAL stores the content into one or more PHP variables of Business Logical Layer (BLL). UI layer outputs the content in the browser as part of the web page using the HTML and CSS code.

The database consists of the database tables with its primary and foreign keys. Primary, foreign keys of the tables and the admin permission determine business logic. Adding, editing and deleting table records are limited by business logic. The database structure is designed around the course and instructor models. The application data table (see Table 1) stores user specific information about courses, instructors, subjects, departments, terms, timeslots, campuses and course attributes. Subjects, terms, timeslots, campuses, and attributes are related to courses. Subjects have a relationship to department. The database table relationships are easily identified when the database structure is represented visually (see Figure 19). Columns in those tables are connected through relationships defined by foreign keys.

Table 1. Application Data Table

(Primary Keys are in bold, Foreign Keys are in italics)

Table Name	**Table Columns**	**Description**
department	**id**, name	Stores information about the organization's departments
subject_department	*subjectid*, *departmentid*	Links subject to department. Many subjects can be part of a single department
subject	**id**, code, name	Stores information about the course subjects
term	**id**, name, termstart, timeend, number	Holds term information. The name of the term, start datetime and end datetime
timeslot	**id**, description, timestart, timeend, days	Holds information for the time a course can be held at. Includes the days, start datetime, and end datetime.
campus	**id**, name, code	Holds information about which campus a course is located
course_attribute	*courseid*, *attrributeid*	Links course to attribute. Many-to-many
attribute	**id**, name, description	Holds attributes set by the organization that a course may have
course	**id**, title, *subjectid*, coursenumber, credithours, *creatorid*, crn, *termid*, *timeslotcode*, capacity, actual, crosslistcapacity, crosslistactual, section, dateadded, courselevel, *primaryinstructor*, *secondaryinstructor*, *tertiaryinstructor*, primarypercentage, secondarypercentage, tertiarypercentage, *campusid*, building, room	Holds information about courses
instructor	**id**, firstname, middlename, lastname, email, password, permissions, phone, degree, position, contract, workloadcap,	Holds information about the faculty

	academicappointment, rate345, rate2, rate1, address, city, state, zip, officephone, *facultycategoryid*	
facultycategory	**id**, name payrate, code	Holds category information about faculty

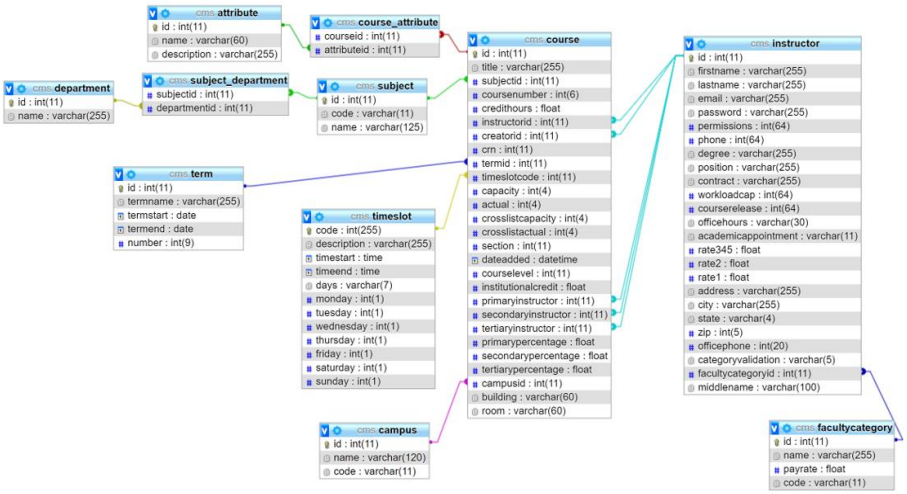


Figure 19. Visual Database Table Relationship

Implementation

The project files in the relationship to the website map is shown in Figure 20. Course and Faculty Management System project with all files is available at github.com Grime C. (2023).

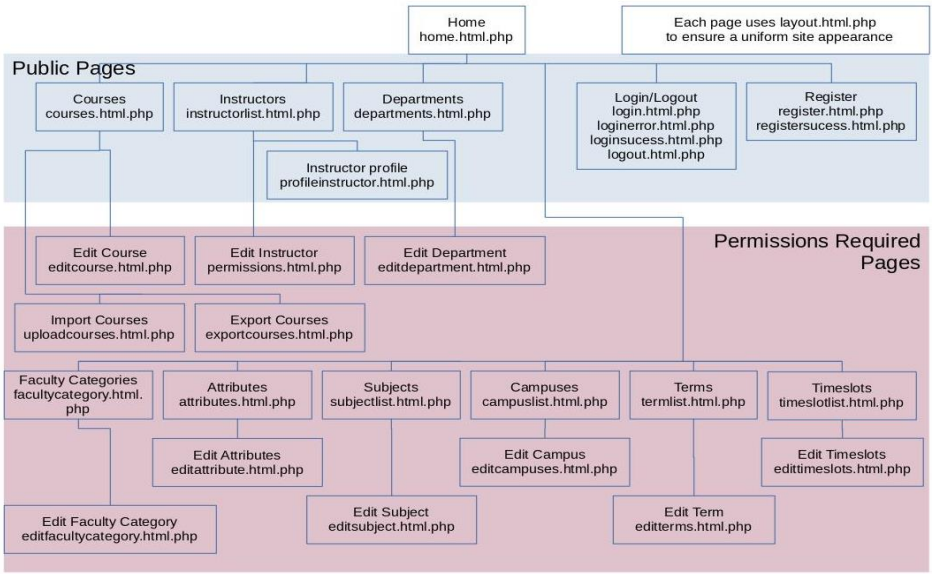


Figure 20. Website map with Corresponding File Names

Test and Results

Testing for the website is a very important step in the project. An example of a test case would be testing the register system (see Figure 21). The system expects four inputs: a first name, a last name, an email address in the correct format, and password. The software has to check if the email is in the correct format and if all the fields in the form are filled. If one of these checks are not true and the software doesn't allow the user to register and doesn't let the incomplete data be sent to the database. An example of a use case for the register system is that the user is able to visit the register page and fill out the form to register in the system. If the entered data is correct (i.e. satisfied the requirements), then the data is sent to the database and the user is able to log in [Flores Marquez, A., Goetz, J. (2023)].

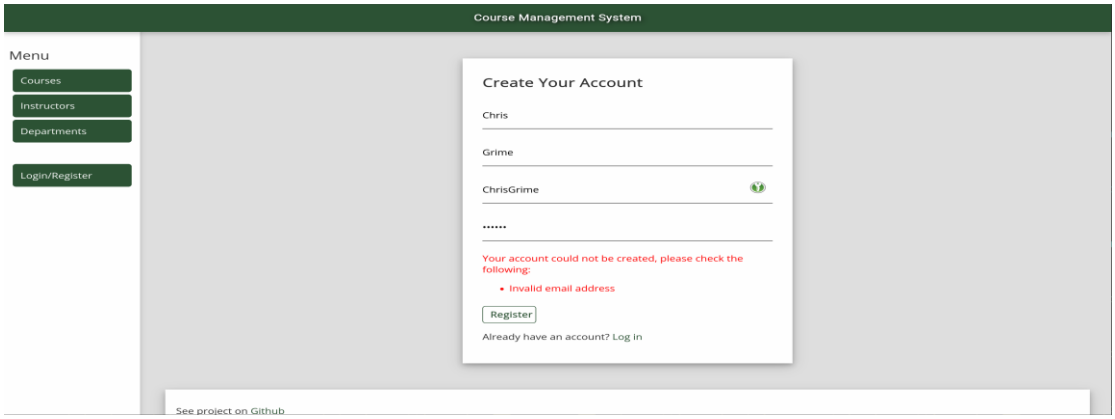
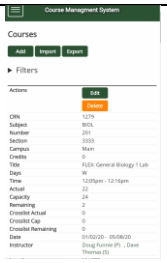
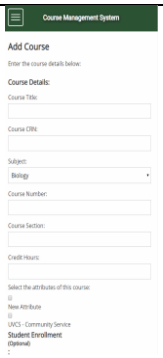


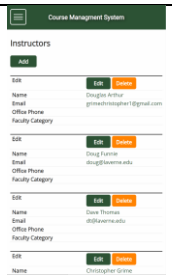
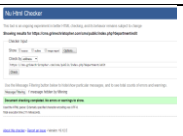
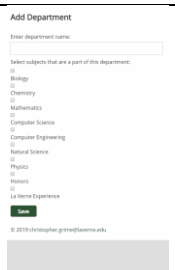
Figure 21. Registration Web Page Showing Error

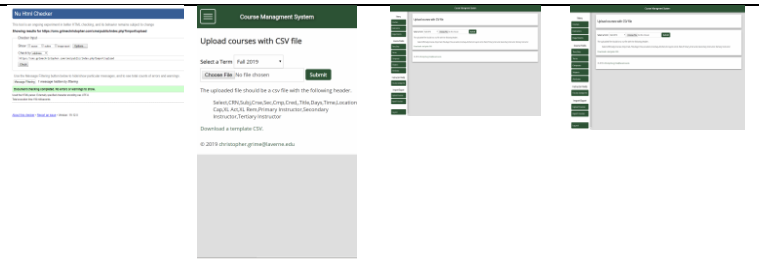
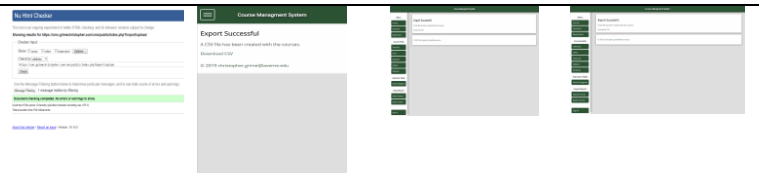
Testing for the website application is performed on all web pages. Responsive design principles are implemented for smart phones, tablets and desktops (see the last three columns Table 2). The example of verification matrix for the web pages courses, instructors, departments, export, import and courses.css is shown in Table 2. All links are tested and they function as intended. Test cases for all possible inputs are performed for each of the input fields on the site. HTML, CSS [Felke-Morris, T. (2021), Deitel, P. J., Deitel, H. M., & Deitel, A. (2012), PHP Butler, T., & Yank, K. (2017)] errors were tested for using validation websites [W3C Markup Validation Service (2023**)**, W3C CSS Validation Service (2023) and **PHP** Code Checker (2023)]. Testing ensures that no improper data is entered into the database.

Table 2. Test Plan and Results - Verification Matrix for the Web Application

1. Page #/name – list all pages	2. Test item - list hyperlinks, fields, buttons, videos, images, GUI controls, web pages	3. Validation each web page using http://validator.w3.org	4. Outcome in Mobile Phone - android chrome browser, brave browser and Firefox	5. Outcome in - tablet. Resolution 1920x1080 Outcome in Chrome, Firefox,	6. Outcome in - desktop. Resolution 1440*900. Good Outcome in Chrome and Firefox

	Schema: click action => reaction Legend: => means land in			Brave and Edge are all good	
courses.html.php	Edit Button => Edit page Add Button => Add page Delete Button => Deleting is working, pop up to confirm				
editcourses.html.php	Save Button => Working Required text fields display a message and do not allow form submission. Title, subject, crn, section left blank will give display a message to prompt the user to fill the fields in. => Working				 -Errors for blank required fields are shown when field is left blank -Errors for number only field are shown if non-numbers are entered into the field -error handling is working 

instructors.html.php	Edit Button => Edit page Add Button => Add page Delete Button => Deleting is working, pop up to confirm				
permissions.html.php	Submit Button => Working Text Inputs => Working Checkboxes => Working				 The required fields: Name and email cannot be left blank. Validation is working 
departments.html.php	Edit Button => Edit page Add Button => Add page Delete Button => Deleting is working, pop up to confirm				
editdepartments.html.php	Save Button => Working Checkboxes => Working				 Department name gives error is it is left blank

import.html.php	Import Button => Working	
exportcourses.html.php	Download link=> Working	
courses.css		

Discussion

This project has been a great exercise to show how expandable the low code interactive framework is. Many new additions to the various PHP, CSS and HTML files were needed in order to achieve a full functionality of Course and Faculty Management System. One addition was adding management pages for courses, instructors, subjects, departments, campus, timeslots, and course attributes with given constrains for each entity. Software like Notepad++ [Notepad++ (2023)], Visual Studio Code [Visual Studio Code (2023)] and phpMyAdmin, which is bundled with XAMPP, [XAMPP (2023)] were used as the core software when constructing the web application.

The basic goal of Course and Faculty Management System is to keeping track of courses and faculty by managing (viewing, adding, editing, deleting) courses, instructors, subjects, departments, timeslots, course attributes with given constrains for each entity. The project was successfully designed and implemented.

Course and Faculty Management System is an emerging application therefore it is limited in features. The current version allows privileged users to perform create, read, write, and delete operations on the connected MySQL database. Other functionality in the application is more limited. Potential expansions can fit into two categories: data, and data analytics.

Each column for the course page can be expanded by adding ascending and descending ordering features. Another extension is to build a notification system around the administrators, specifically if another administrator permission is changed or its record is modified or deleted. In a data management system that has the potential to hold large amounts of data, such as Course and Faculty Management System, navigating the stored data is an important function. Filters and search functionality for entities such as the courses and instructions should be implemented for any attributes that entity may have. Filtering and search will improve the efficiency of the users of Course and Faculty Management System. Data imported using the mass import feature currently requires data

to be in an exact format. A better solution would be to use application programming interfaces (API) provided by other applications where available. If an API is not available the Course and Faculty Management System should provide a dynamic method of importing data.

Another expansion is to develop a system to allow users the ability to upload and retrieve binary files (images and documents) for instructors and classrooms and have them stored on database server for the user display. Additional development is to introduce a two-factor authentication security subsystem.

Data analytics and report generation are the main future goal of Course and Faculty Management System. Google analytics can be incorporated in the system and pivot table services can be incorporated into the framework as well. The application is excellent at receiving and storing data, the next step is for the application to use the data to create exports. Another extension of the project is to generate instructor contracts and reports to help the university to make informed decisions backed by data.

Conclusions

The goal of this work is achieved within the computer science senior project by studying, learning and expanding coding of the low code, interactive, secure framework for the purpose of designing, implementing and testing Course and Faculty Management System. The main goal of the Senior Project course (CMPS 499) is to provide a capstone and culminating experience in which the student combines knowledge, skills and topics that the student previously learned with new topics learned during the course. Moreover, students need to have a senior project presentation and write a final project report. All computer science undergraduates are required to take a capstone course senior project at the La Verne University and many other universities [Computer Science Curricula (2013)].

Moreover, the senior project course (CMPS 499) satisfies the following Program Criteria for Computer Science student outcomes [Accreditation Board for Engineering and Technology – ABET (2022-2023)] as follows:

- Ability to analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
- Ability to design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
- Ability to communicate effectively in a variety of professional contexts.
- Ability to apply computer science theory and software development fundamentals to produce computing-based solutions.

Before starting work on a senior project, in another class (CMPS 320 - Internet Apps Development), students learn how to construct dynamic, data-driven web applications, and secure, customized content management systems using PHP and MySQL. Students develop skills in many aspects of the software development and deployment process. Students develop basic and intermediate skills in structured Query Language (SQL), MySQL, development using PHP, form processing, and regular expressions. Therefore, many new additions to the various PHP and HTML framework files were successfully added or modified to achieve the well-structured,

full functionality of Course and Faculty Management System. The design, implementation and testing of Course and Faculty Management System with a wide spectrum of functionality is an excellent proof of a senior project achievement beyond regular CMPS 499 requirements.

The current version of Course and Faculty Management System provides an excellent base for expansion. The web application provides the ability to perform create, read, update, and delete operations on university course and faculty data. Expansion of this data management solution would provide increased efficiency and allow any organization that implements it to make optimizations based on their own data. Course and Faculty Management System has a clear purpose, to manage and assist in creation of course scheduling. Course and Faculty Management System also has the potential to assist in any task that requires course and faculty data as well as generate analytical reports.

Acknowledgments

We would like to express our special thanks to our executive secretary of Natural Science Division office for her feedback and suggestions during the project's completion.

References

Accreditation Board for Engineering and Technology – ABET (2022-2023). *Program Criteria for Computer Science*. https://www.abet.org/wp-content/uploads/2022/03/2022-23-CAC-Criteria.pdf.

Azma, H., *Goetz, J. (*2007*). WEBCRM Application Generator, Proceedings of the 2007 International Conference on Internet Computing*, World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'07), Las Vegas, USA, ISBN: 1-60132-044-2, pp 29-35.

Butler, T., & Yank, K. (2017). *Php & MySQL: Novice to Ninja* (6th ed.). SitePoint Pty, Ltd.

Computer Science Curricula (2013), https://ieeecs-media.computer.org/assets/pdf/CS2013-final-report.pdf, Curriculum Guidelines for Undergraduate Degree Programs in Computer Science December 20, 2013, *The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society.*

Grime C. (2023). Course and Faculty Management System, https://github.com/grimechristopher/cms**.**

Deitel, P. J., Deitel, H. M., & Deitel, A. (2012). *Internet & World Wide Web: How to program* (5th ed.). Pearson.

Felke-Morris, T. (2021). *Web Development & Design Foundations with HTML 5* (10th ed.). Pearson.

Flores Marquez, A., Goetz, J. (2023). Certificate Management Application. In M. Shelley & V. Akerson (Eds.), *Proceedings of ILSET 2023-- International Conference on Life Sciences, Engineering and Technology*, Denver, CO, USA. ISTES Organization.

Guarrera, A., *Goetz, J. (2022,* May 10*). Patient Care Reporting App*, https://libapps.s3.amazonaws.com/customers/897/images/Case_Day_2022_v2.png, La Verne's Research Case Day, May 10, 2022.

Kumar, M., Garg, A., & Kumar, A. (2021). Critical factors of post implementation of ERP in Higher Education Systems Survey Review. *IOP Conference Series: Materials Science and Engineering, 1149*(1), 012017.

https://doi.org/10.1088/1757-899x/1149/1/012017.

Miranda-Hill, W., Goetz, J. (2019, *June 30 – July 4*). User-generated Geospatial Meteorology Map Prototype, *Proceedings of the Informing Science + Information Technology Education Conference (InSITE)*, Jerusalem, Israel, pp 257 – 269.

Soliman, M. &, Noorliza, K. (2015). Higher Education Competitive Advantage: Enterprise Resource Planning Systems. *International Journal of Research in Management & Technology*, *5*, 380-384.

Notepad++ (2023). *https://notepad-plus-plus.org/.*

Visual Studio Code (2023). https://code.visualstudio.com/.

W3C Markup Validation Service (2023). https://validator.w3.org/ .

W3C CSS Validation Service (2023). https://jigsaw.w3.org/css-validator/.

Web Design Best Practices Checklist (2023). http://terrymorris.net/bestpractices.

PHP Code Checker (2023). https://phpcodechecker.com/.

XAMPP (2023). https://www.apachefriends.org/download.html.

Author Information

Christopher Grime

https://orcid.org/0009-0007-8713-7147

University of La Verne

1950 Third Street, La Verne, CA 91750

USA

Jozef Goetz

https://orcid.org/0009-0004-7751-3653

University of La Verne

1950 Third Street, La Verne, CA 91750

USA

Contact e-mail: *jgoetz@laverne.edu*