Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/40/20230622

Grader system built on ruby on rails

Haojie Hu

Art and Science College, The Ohio State University, 5001 Olentangy River R.D., Taylor House STE#133, Columbus, OHIO, America

Hu.2339@buckeyemail.osu.edu

Abstract. This paper presents a Ruby on Rails-based grader application system designed to streamline the process of matching students with grading positions within the Computer Science and Engineering (CSE) department. Motivated by the need for efficiency and consistency, the system offers role-based user access, enabling students, instructors, and administrators to engage seamlessly. Leveraging Model-View-Controller (MVC) architecture, the system integrates external tools for enhanced development, while the dynamic database schema efficiently manages data. Key functionalities encompass application submission, administrator interface, and real-time course management. This innovative system fosters collaboration, improves administrative oversight, and adapts to changing academic demands. In conclusion, the presented grader application system has achieved powerful functions which enables users to login, view available courses, and most importantly, accept or decline the applications from students whom want to be a grader for a specific course.

Keywords: Model view controller, rails, ruby, web application.

1. Introduction

In today's dynamic educational landscape, managing the allocation of qualified undergraduate students to various course sections for grading purposes is a complex endeavor. The Computer Science and Engineering (CSE) department at university is no exception, as it annually recruits a substantial number of student graders to assist with its diverse array of courses [1,2,3]. However, the existing approach to matching students with course sections lacks a systematic and efficient workflow. This deficiency has prompted the need for a more structured solution that not only streamlines the allocation process but also enhances the quality of information gathering and consistency.

The aim of this paper is to present a comprehensive overview of a Ruby on Rails web application designed to address the challenges [3,4,5]. By leveraging the capabilities of modern web technologies, this application seeks to revolutionize the way student graders are assigned to course sections within the CSE department. We will delve into the design, functionality, and critical features of the application, outlining how it facilitates the seamless interaction between students, instructors, and administrators, while harnessing the power of an external API for data retrieval and manipulation.

Through this paper, we intend to provide a comprehensive understanding of the innovative web application, its technical underpinnings, and its potential to transform the way course grading assignments are managed within the CSE department. By adopting cutting-edge technologies and adhering to best practices in software design, this application promises to revolutionize the efficiency and effectiveness of the grading allocation process, benefiting both students and faculty alike.

In the subsequent sections, we will delve into the specifics of the project requirements, the implementation details, and the anticipated outcomes of deploying this web application within the CSE department.

2. Tools utilized

In the pursuit of developing a robust and innovative Ruby on Rails application to streamline the course grading assignment process, a strategic selection of auxiliary tools was employed to enhance various aspects of the project. Beyond the fundamental Rails framework and its encompassing GEM ecosystem, a quartet of specialized tools played a pivotal role in shaping the project's success [5].

- *JSONvue*

Firstly, the integration of JSONvue, a Chrome extension tailored for API interactions, empowered seamless communication with the external OSU course catalog API. This tool facilitated the real-time visualization and interpretation of API responses, thereby expediting data extraction and validation processes.

- *DBDiagram*

To ensure a comprehensive understanding of the underlying database structure, DBDiagram, a web-based platform, was harnessed to visually depict the database schema. This intuitive visualization facilitated collaborative design decisions and promoted a lucid comprehension of the data architecture.

- *GitHub Copilot*

Furthermore, the efficiency of development was significantly augmented by leveraging GitHub Copilot, an extension for Visual Studio Code. This AI-powered tool extended beyond the conventional Rails GEMs, offering intelligent code suggestions that expedited coding and improved code quality, ultimately contributing to an optimized development workflow.

- *ERB-VSCode*

Additionally, the ERB-VSCode-Snippets extension for Visual Studio Code furnished a repository of pre-configured code snippets tailored for Embedded Ruby (ERB) templates [6]. These snippets expedited the creation of dynamic web content within the Rails views, streamlining the process of crafting user interfaces and enhancing the consistency of code implementation.

Collectively, the amalgamation of these supplementary tools fortified the development journey, enhancing efficiency, code quality, and collaboration, and ultimately leading to the creation of a sophisticated web application poised to revolutionize the course grading assignment process within the CSE department

3. Architecture and Design

In alignment with the fundamental principles of the Model-View-Controller (MVC) architecture, our Ruby on Rails application has been meticulously structured to ensure modularity, separation of concerns, and efficient data flow [4,7]. This architectural approach has enabled us to create a robust and maintainable solution for the course grading assignment process within the CSE department (Figure 1).

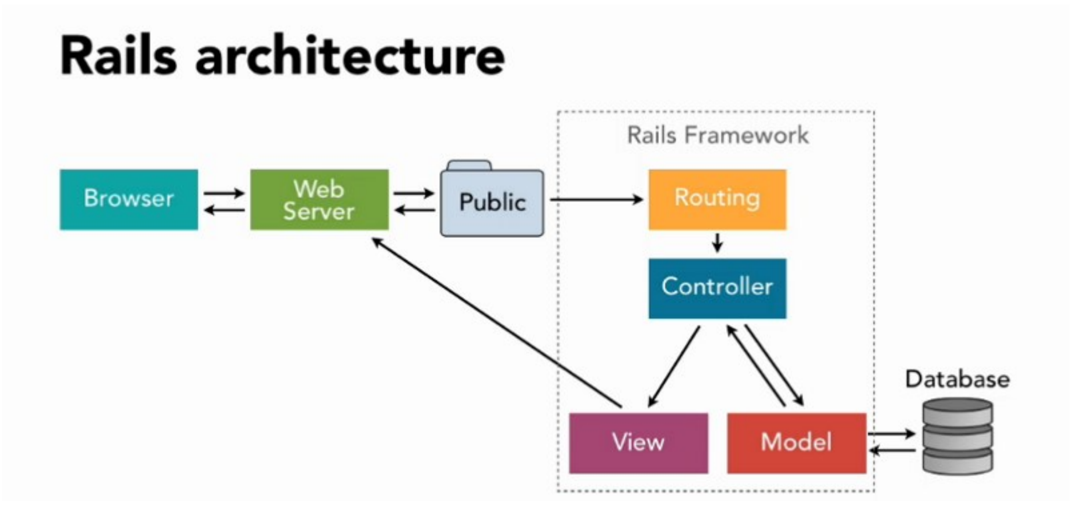


Figure 1. Rails architecture (Original)

- *Models*

At the core of our application's data management are the models, which encapsulate the essential entities and their relationships. Notably, the `Course`, `Section`, `Term`, and `User` models play pivotal roles in representing courses, course sections, academic terms, and users respectively. These models are complemented by associations such as `belongs_to`, `has_many`, and `has_one`, which seamlessly establish relationships and enable data integrity. For instance, the `User` model associates with an `Application` model to facilitate the submission and management of grading applications. Furthermore, the `Week` model, linked to the `Section` model, enables structured handling of lab-based course schedules.

- *Controllers*

The controller layer of our application encompasses three distinct categories. The `Admin` directory hosts controllers that manage administrative functionalities, including course catalog reloading, course editing, and user permissions. Meanwhile, the `User` directory houses controllers responsible for user authentication, registration, and profile management. Finally, directly under the controller directory, a suite of controllers orchestrates the primary functionalities of the application, such as courses, sections, and user profiles [8]. For example, in the application controller file, we wrote functions such as update_allowed_parameters, check_admin, check_sign_in, and check_teacher functions, allowing users make changes on application. These controllers not only handle user input and interactions but also mediate the communication between models and views.

- *Views*

Utilizing the Embedded Ruby (ERB) template system, our application's views seamlessly combine dynamic data with HTML markup to create interactive and user-centric interfaces [5]. While the views are grounded in simplicity, they effectively render information from the models to the end users. Although no special templates were employed, our views are distinguished by their effective communication of data and the seamless integration of CSS files for styling purposes.

Our application's views epitomize a harmonious blend of functionality and aesthetics, with the homepage standing as a testament to this design philosophy. At the apex of the homepage, a commanding black banner unfurls, adorned with the OSU emblem on the left and the succinct title "Grader System" beneath. The banner serves as a beacon, ushering users into the application's world. Directly beneath, a discreet white row extends, accommodating the user's name and a "Log Out" button, positioned for convenience and ease of access. This row encapsulates essential user-related information within a clean and unobtrusive layout. Continuing the visual journey, five distinct grey plates unfold beneath the banner and user row. Each plate represents a gateway to a distinct functional realm: "Admin

Page," "Course Catalog," "Teacher," "Application Page," and "Profile." This intuitive arrangement ensures effortless navigation and encourages users to explore facets most relevant to their roles. Incorporating CSS seamlessly, our views merge dynamic content and design elements, culminating in a user-centric and visually engaging homepage. Through thoughtful color choices, structured layouts, and strategic placement of essential information, the homepage sets a compelling tone for the user experience within the application (Figure 2).

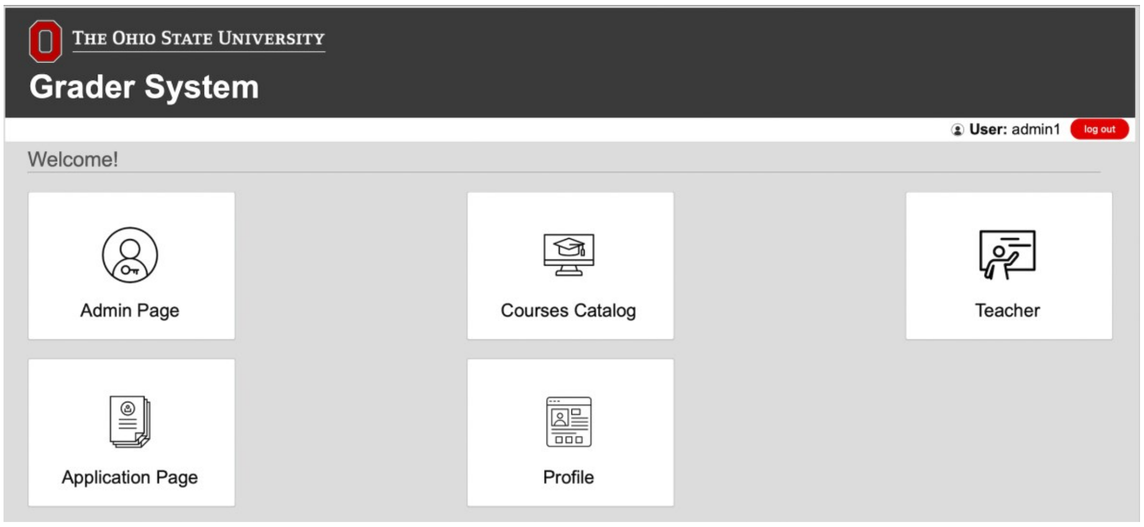


Figure 2. Homepage of Views (Original)

4. Database Design and Schema

The foundation of our application's functionality and data management lies within a meticulously designed database schema (Figure 3). The schema encapsulates the intricate interplay of entities, attributes, and relationships, creating a coherent structure that underpins the entire system [9,10].

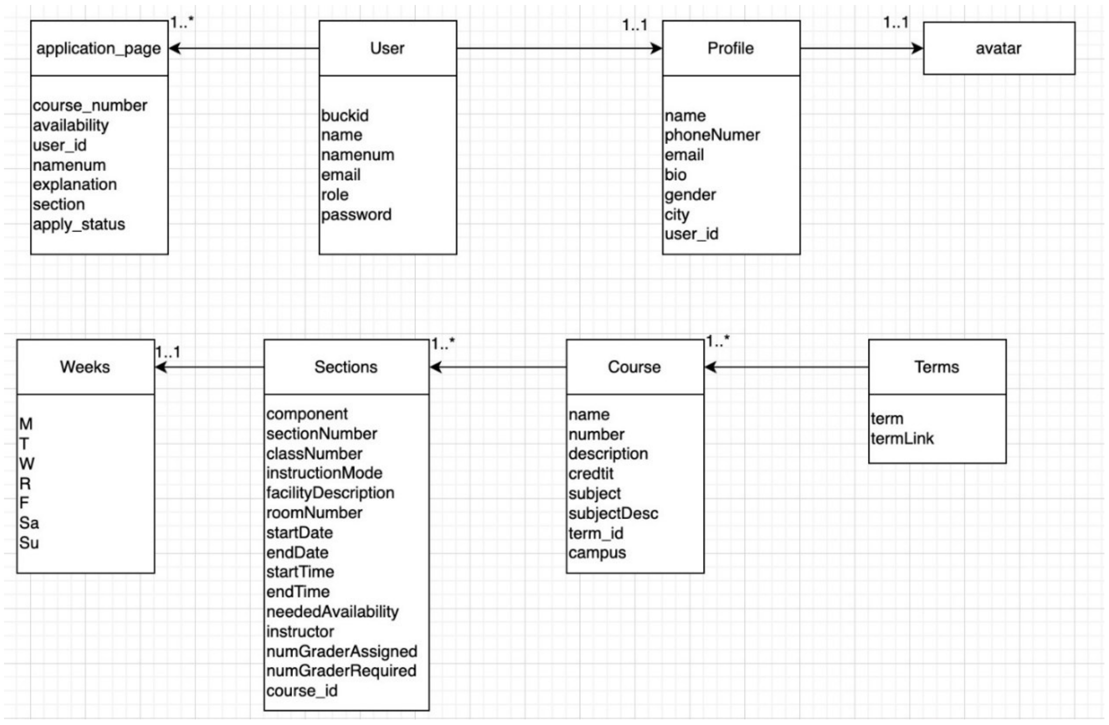


Figure 3. Entity-Relational Diagram/Schema Diagram (Original)

- *User-Centric Entities*

Central to our schema is the `User` entity, representing the users of the system. This entity harmoniously captures essential user-related information, including names, email addresses, roles, and authentication credentials. The `Profile` entity complements the `User`, offering a platform for users to provide additional details such as contact information, biography, and an optional avatar. This user-centric approach aligns seamlessly with the application's user management and authentication.

- *Courses, Sections, and Terms*

The schema artfully accommodates academic information through the `Course`, `Section`, and `Term` entities. A `Course` encompasses attributes like name, number, description, and subject, enabling the cataloging of distinct courses. Each `Course` establishes a one-to-many relationship with `Sections`, which encapsulate class details such as section numbers, instructors, and schedules. A `Term` serves as a bridge between `Courses` and provides the temporal context for academic offerings.

- *Application and Recommendations*

The schema supports the core workflow of the grader assignment process through the `ApplicationPage` entity. Applicants' submissions, including course preferences and availability, are stored here. Furthermore, the `Recommendations` entity encapsulates recommendations provided by instructors, comprising the teacher-student relationship and recommendation content.

- *Dynamic Data and Availability*

For dynamic data management, the schema integrates the `Week` entity, intricately linked with `Sections`, to represent weekly availability. This dynamic schema element empowers the application to allocate graders based on their schedule preferences and section demands.

- *Efficient Data Storage with Active Storage*

Our database design is further enriched by the seamless integration of Active Storage, an essential component for managing file attachments. Tables such as `ActiveStorageBlobs` and `ActiveStorageAttachments` efficiently store and associate attachments, ensuring streamlined access to files like avatars and other user-generated content.

In summation, our meticulously crafted database schema harmoniously embodies the application's core functionalities and user interactions. The schema's structured design, thoughtful entity attributes, and judiciously defined relationships provide a robust and intuitive foundation for data management, reinforcing the seamless user experience and dynamic workflow of our Ruby on Rails application.

5. Functionality and Features

Our Ruby on Rails application boasts a comprehensive suite of functionalities and features carefully designed to streamline and enhance the intricate processes within the grading system. These capabilities encompass a wide spectrum, catering to the needs of students, instructors, and administrators alike.

- *General Functionality*

At the core of our application's user experience is a user-friendly homepage that seamlessly integrates options for login, registration, and course catalog browsing. For authenticated users, a password reset feature is available, enhancing security and accessibility (Figure 4). During signup, users can specify their role as a Student, Instructor, or Admin, with at least one default Admin created with a predefined password. Our system ensures that user identities adhere to the OSU name.#@osu.edu format, facilitating efficient linkage to instructor recommendations, grader assignments, and future expansions.

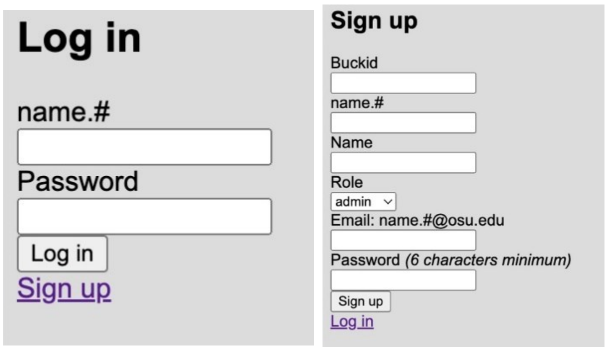
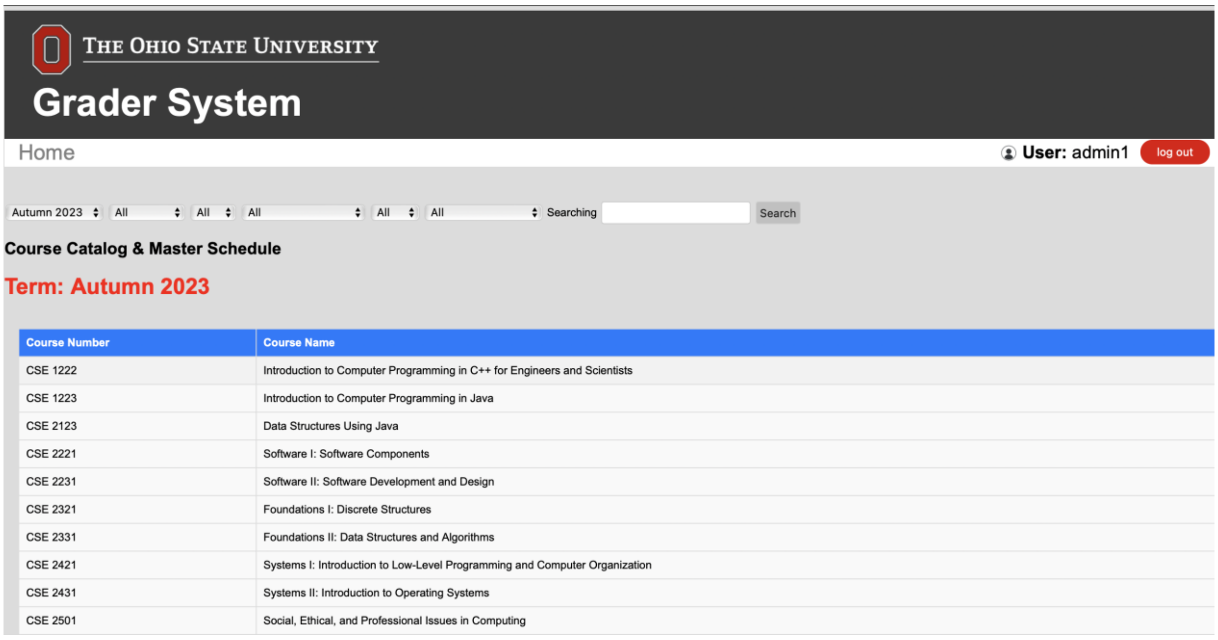


Figure 4. Functionalities of Log In and Sign Up (Original)

- *Student Functionality*

Students are empowered with an intuitive login mechanism, granting them access to a wealth of academic offerings through the course catalog. This functionality facilitates efficient exploration of course details, enabling students to make informed academic decisions (Figure 5).



Course Catalog & Master Schedule

Term: Autumn 2023

Course Number	Course Name
CSE 1222	Introduction to Computer Programming in C++ for Engineers and Scientists
CSE 1223	Introduction to Computer Programming in Java
CSE 2123	Data Structures Using Java
CSE 2221	Software I: Software Components
CSE 2231	Software II: Software Development and Design
CSE 2321	Foundations I: Discrete Structures
CSE 2331	Foundations II: Data Structures and Algorithms
CSE 2421	Systems I: Introduction to Low-Level Programming and Computer Organization
CSE 2431	Systems II: Introduction to Operating Systems
CSE 2501	Social, Ethical, and Professional Issues in Computing

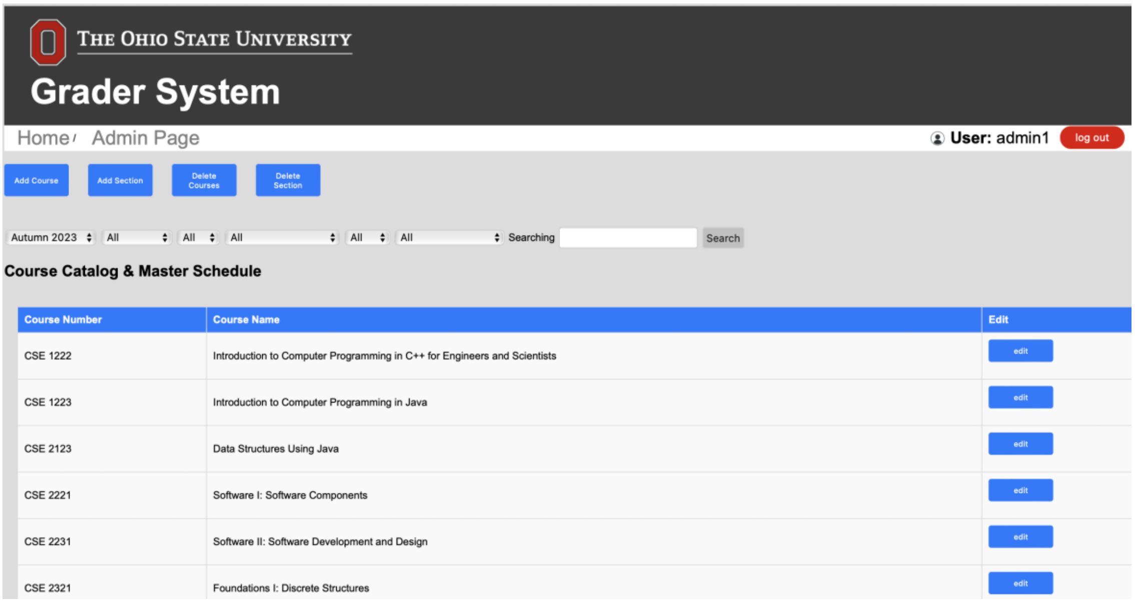
Figure 5. Functionality of Browsing Courses (Original)

- *Instructor Functionality*

Instructors benefit from a dedicated login portal, granting them entry into the course catalog. This tailored access equips instructors with the information they need to effectively plan and manage their course offerings.

- *Admin Functionality*

Administrators wield a comprehensive toolkit encompassing course catalog browsing, editing, and reloading functionalities. Administrators are empowered to oversee the hiring process, review applications, and approve Instructor or Admin requests. This functionality places administrative control at the heart of the system, ensuring seamless and effective management (Figure 6).



Course Catalog & Master Schedule

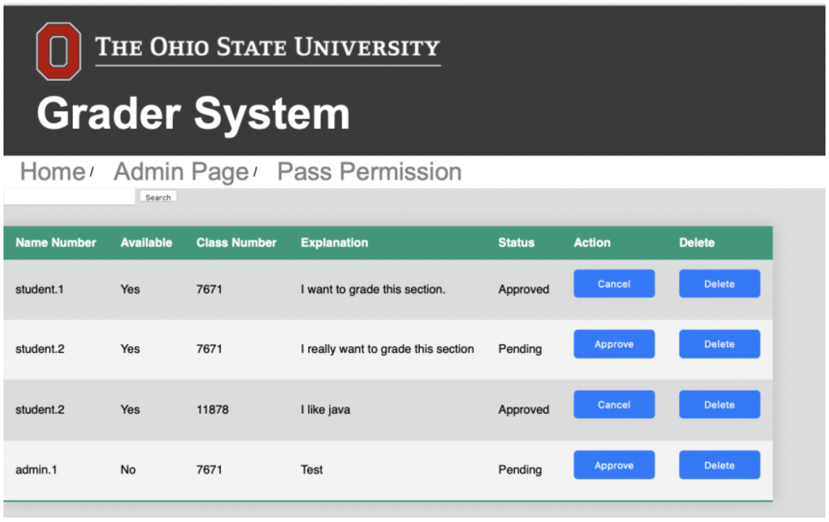
Course Number	Course Name	Edit
CSE 1222	Introduction to Computer Programming in C++ for Engineers and Scientists	edit
CSE 1223	Introduction to Computer Programming in Java	edit
CSE 2123	Data Structures Using Java	edit
CSE 2221	Software I: Software Components	edit
CSE 2231	Software II: Software Development and Design	edit
CSE 2321	Foundations I: Discrete Structures	edit

Figure 6. Functionality of Editing Course Information (Original)

- *Critical Features*

Our application's critical features address the fundamental aspects of the grading system, ensuring its smooth operation and comprehensive coverage. A standardized application submission form empowers students to express their interest in grading positions, providing essential contact information and course preferences. Dynamic schedule information allows graders to be matched with sections requiring specific availability, adapting to changing schedules and qualifications over different semesters.

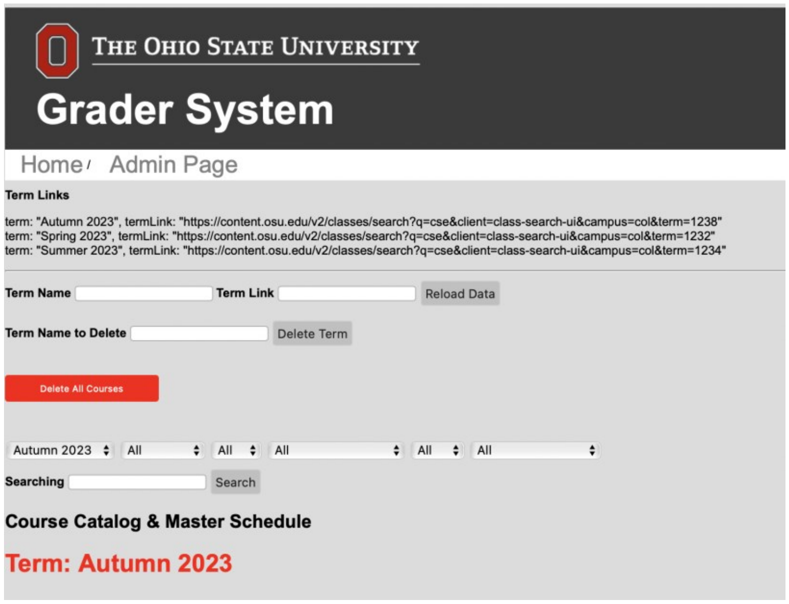
The administrator interface is a linchpin of the system, furnishing office staff with vital insights to make informed hiring decisions. This interface illuminates sections with assigned graders and highlights those in need, facilitating efficient allocation. Administrators can also adjust the number of required graders, optimizing staffing resources (Figure 7).


The Ohio State University
Grader System
Home / Admin Page / Pass Permission
Search

Name Number	Available	Class Number	Explanation	Status	Action	Delete
student.1	Yes	7671	I want to grade this section.	Approved	Cancel	Delete
student.2	Yes	7671	I really want to grade this section	Pending	Approve	Delete
student.2	Yes	11878	I like java	Approved	Cancel	Delete
admin.1	No	7671	Test	Pending	Approve	Delete

Figure 7. Functionality of Pass Permission (Original)

The management of available courses and sections, which evolve with each semester, is seamlessly addressed by our application. It dynamically accommodates changes in course offerings, adapting to shifting academic demands and schedule variations. Leveraging external data sources, our system ensures section offering information is always current, facilitating timely updates. Figure 8 is the functionality of updating course



Term Links

term: "Autumn 2023", termLink: "https://content.osu.edu/v2/classes/search?q=cse&client=class-search-ui&campus=col&term=1238"
term: "Spring 2023", termLink: "https://content.osu.edu/v2/classes/search?q=cse&client=class-search-ui&campus=col&term=1232"
term: "Summer 2023", termLink: "https://content.osu.edu/v2/classes/search?q=cse&client=class-search-ui&campus=col&term=1234"

Course Catalog & Master Schedule

Term: Autumn 2023

Figure 8. Functionality of Updating Course (Original)

Recommendation submission, a pivotal feature, bridges the gap between instructors and potential graders. Our standardized form empowers instructors to endorse students for future considerations or explicitly request their assignment as graders for specific sections. This feature anticipates the complexities of recommendation submissions even when the recommended student has not yet logged into the system, ensuring a comprehensive and inclusive process.

6. Conclusion

In this paper, we have presented a comprehensive overview of our Ruby on Rails-based grader assignment system, highlighting its architecture, design, database schema, and an array of functionalities that collectively contribute to its robust and user-centric nature. With a strong foundation rooted in the MVC architecture, our application seamlessly integrates various tools and technologies to create an intuitive and efficient platform. In the future, we will try to improve the scale of the application, more specifically, revise the database so that it's large enough to enable more users to log in to the application.

References

[1] Denys Klochkov and Jan Mulawka, 2021, Improving Ruby on Rails-Based Web Application Performance, 2021 *Mdpi Ag* **12 319**, p 319.

[2] Emoto, M., Yoshida, M., Iwata, C., Inagaki, S., Nagayama, Y., Emoto, M., Yoshida, M., Iwata, C., Inagaki, S., Nagayama, Y., Efficient development of web applications for remote participation using Ruby on Rails, *ISSN: 0920-3796, Publisher: Elsevier B.V.* 2010

[3] Jacquelyn W. BLAZ and Patricia F. Perace, Using Ruby On Rails to Develop A Web Interface: A Research-Based Exemplar with a Computerized Physical Activity Reporter, 2019, pp 281-283

[4] Wang Ning Li Liming Wang Yanzhang Wang Yi-bing Wang Jing, 2008, Research on the Web Information System Development Platform Based on MVC Design Pattern, DOI: 10.1109/WIIAT.2008.64.

[5] UZUN, Erdinç, 01-11-2020, A regular expression generator based on CSS selectors for efficient extraction from HTML pages., *Turk J Elec Eng & Comp Sci,* 2020, 28: pp 3389-3401, doi:10.3906/elk-2004-67

[6] Saputra, Dimas Gilang Azizah, Fazat Nur, 2013, A Metadata Approach for Building Web Application User Interface., Procedia Technology; 2013, **11**, p903-911.

[7] Al-Hawari, F., Software design patterns for data management features in web-based information systems, *Journal of King Saud University* 2022, **34(10)**:10028-10043

[8] Guaman, D. Delgado, S. Perez, J., Classifying Model-View-Controller Software Applications Using Self-Organizing Maps, *IEEE Access*, 2021. 9:45201-45229 2021

[9] Curry, E. Grace, P., 2008, Flexible Self-Management Using the Model-View-Controller Pattern, I*EEE Software,* 2008 **25(3)**.

[10] Sauter, Patrick, Vögler, Gabriel, Specht, Günther, Flor, Thomas, 2005, A Model-View-Controller extension for pervasive multi-client user interfaces, *Personal & Ubiquitous Computing*, 2005, 100-107.