Class scheduling Web App

Anubhav Rawal
Western Michigan University, rawal.anubhav@outlook.com

Follow this and additional works at: https://scholarworks.wmich.edu/honors_theses

Part of the Software Engineering Commons

Recommended Citation
https://scholarworks.wmich.edu/honors_theses/3476

This Honors Thesis-Open Access is brought to you for free and open access by the Lee Honors College at ScholarWorks at WMU. It has been accepted for inclusion in Honors Theses by an authorized administrator of ScholarWorks at WMU. For more information, please contact wmu-scholarworks@wmich.edu.
Class Scheduling Application in Django
Senior Design Final Project

Altan Rawal, Brenden Rasmussen, Michael Coffey
12-2-2021
Table of Contents

Abstract .................................................................................................................................................. 4
Problem Statement ................................................................................................................................ 4  
   Need ................................................................................................................................................ 4
   Objective(s)....................................................................................................................................... 4
   Terms, Acronyms, Glossary ................................................................................................................. 5
Problem Analysis and Research .............................................................................................................. 6  
   Django Web Framework ..................................................................................................................... 6
   File Parser ........................................................................................................................................... 6
   Conflict Checking ................................................................................................................................ 6
Requirements ......................................................................................................................................... 7
   Non-functional requirements .............................................................................................................. 7
      Platform: ......................................................................................................................................... 7
      Performance: .................................................................................................................................. 7
      Usability: ......................................................................................................................................... 7
      Security: .......................................................................................................................................... 7
   Functional requirements ..................................................................................................................... 8
      User: ................................................................................................................................................ 8
      Importing an excel: .......................................................................................................................... 8
      Dynamic editing: ............................................................................................................................... 8
      API Interface and Save Changes: ..................................................................................................... 8
      Recycle Bin: ..................................................................................................................................... 9
Standards and Constraints ...................................................................................................................... 9
   Applicable Standards .......................................................................................................................... 9
   Constraints.......................................................................................................................................... 9
   Cost.................................................................................................................................................... 9
Project Management Plan.........................................................................................................................................17
Development Costs...............................................................................................................................................17
Institution Costs .................................................................................................................................................17
Sponsor Costs ....................................................................................................................................................17
Team Costs .......................................................................................................................................................17
Abstract

This Scheduling Web Application in Django was built to allow a user to create a schedule for upcoming semesters. Users with appropriate privileges could upload an excel schedule to which instructors could be assigned. Additionally features included dynamic editing for admin users and viewing the main schedule for general users. Our goal was to create an application from the ground up using the Django framework to accomplish these tasks.

Problem Statement

Need

Every semester, the registrar's office at Western Michigan University sends a list of classes being offered that semester. Dr. Carr and the Computer Science Department administrators need to be able to schedule the faculty, locations, and times for each class in the upcoming semester. This scheduling was done using spreadsheets but was given an upgrade 7 years ago. Dr. Carr created a web application in Ruby on Rails to edit the schedule and display it for himself and other users. He had the ability to export this newly updated schedule and send it back to the University to make it an official schedule. This application was an improvement in previous methods but was not as user-friendly as it could be. Dr. Carr asked for a team to rebuild the application using the web framework, Django.

Objective(s)

The current application was developed in the web framework, Ruby on Rails. This application was able to import a schedule, assign instructors, meeting times, and rooms to every single class. The application is used by the head of the department to create an efficient schedule for the semester. The new application, built in the web framework, Django, will allow the administrator to accomplish the tasks of the current application. However, the administrator should be able to edit the information all in one area and be able to edit multiple classes in one editing session. Once the schedule is saved, the application should notify the user of any conflicts within the newly edited schedule.

From another user's perspective, they should have the ability to sign in and view the department's potential schedule. They will not have the ability to make any edits unless given permission by the administrator.
With time constraints and lack of knowledge in technologies, the project was assigned to us knowing that all objectives would not fully be met. However, it was the team’s priority to develop the application with regards to future development.

**Terms, Acronyms, Glossary**

**AJAX**: A JavaScript plugin that communicates frontend to Django

**Database**: a structured set of data held in a computer, especially one that is accessible in various ways.

**Django**: a web framework built in python

**DoS**: Denial of Service

**jQuery**: jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and Ajax.

**Model-View-Controller (MVC)**: A software design pattern that is mostly used for interfaces. This a central feature of Web Frameworks

**MySQL**: A database language that is embedded into the system.

**NGiNX**: is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache

**Python**: A high-level programming language. Developed in 1991.

**Ruby**: A high-level programming language developed in the mid-1990s.

**Ruby on Rails**: An older web application framework that was written using Ruby.

**Spreadsheet**: an electronic document in which data is arranged in the rows and columns of a grid and can be manipulated and used in calculations.

**UI (User Interface)**: The specific parts of the application with which a user interacts.

**URL**: Uniform Resource Locators

**User Authentication**: Checking user’s credentials and permissions to make sure that users only have access to features.

**User Permissions**: Out of all the functions of the program, it is what the user can do. Not every user can use every function.

**uWSGI**: aims at developing a full stack for building hosting services, typically used to run Python web applications.

**UX (User Experience)**: All aspects of the application that impact the user, e.g., aesthetics and function.
Web Framework: a software framework designed to support the development of web applications; automates some of the overhead associated with common activities performed in web development. (Wikipedia, 2021)
.xlsx: A file type used in Excel spreadsheets.

Problem Analysis and Research

Django Web Framework

The entire application would be developed using this framework. The problem the group had was the lack of knowledge of web frameworks. To overcome this problem, after initially accepting this project, the group decided that before designing and working on the final project, we would each take time to access materials and learn the web framework first. This was done through Django’s tutorial program that walks the user through many features that the framework offers. After this tutorial project was completed, each member went on their own path to learning more about technologies. After all the research was finished, all members got together to design the structure of the application.

File Parser

To start with scheduling a semester in our application, the user will need to upload a file that the program can parse and import the data into the database. With the current format of the files from the University’s registrar office, a custom parser needed to be built. There are many lines that are left blank and needed to be skipped. To handle this potential problem, the package python package pandas was implemented. An algorithm was implemented to grab all information for a single class and import it as one to the database. This algorithm finds a new course registration number, then grabs the information from that row and any information existing in the next rows if the next row is not a new course registration number.

Conflict Checking

Conflict checking would give the user an appropriate message informing them of where there could be a potential conflict. The issue with conflict checking was determining how and when the check would be conducted. We decided after the user saves to the database that the checker would occur. The best method of going about this we found was to use MySQL queries to put into python. Then an algorithm
would be running on this information to check for any potential conflicts. To import this information into python, we used the python package sqlalchamy.

Requirements

Non-functional requirements

Platform:
The user-facing portion of this system was done using HTML, Bootstrap, and jQuery to provide users with a seamless responsive design that they can interact with. The backend API is designed in Django API framework that connects to a MySQL database. The interfacing between database and backend server uses Django's API CRUD libraries which are requested from the Ajax from client side.

Performance:
The application was designed to handle multiple entries in one request, and hence if the server remains up it can handle all the requests made to it. Django itself has a bit slower performance, however, as the data being handled is so small that it’s hard to notice the difference between using Django any other available frameworks.

Usability:
The front-end design has been kept simple enough for anyone to use it with little to no training. All the information being displayed currently on the GUI is meant to be self-explanatory. For running the application locally and deployment, there is a Readme.md file provided for reference.

Security:
Currently, we lack in the department of security we haven’t established the user accounts and privileges yet. The site is vulnerable to DoS attacks for the site will process any amount of request for any user within the current scope. The master API key is also exposed to the client side, which should be customized to each user instead. However, all these concerns can we easily addressed with a secure user design implementation. Most of other vulnerabilities are sealed off by Django itself.
Functional requirements

The new page followed in the footpath of the legacy software to avoid any drastic changes and need of training for the new software. The URL’s and user interface locations and design were inspired by the legacy software too.

User

Our client requested us for three level of user for accessing the application.

- The superuser:
  - Has access to all information and can edit any page on demand
  - Is to be accessed by the client himself

- The admin:
  - Has access to all information and can perform edits to their specific department
  - Is to be accessed/assigned to the department heads

- The basic user:
  - Is to be assigned to the instructors
  - Has only view access to their department’s information.

Importing an excel

The web app takes in a excel file to initialize a term in the database. The excel file parse is customized to the WMU’s special formatting on departmental class schedule report. After the file is successfully read into the database, user is prompted about the status of the file upload.

Dynamic editing

One of the main reasons for the upgrade was to gain the ability to edit multiple entries at the same time within the same page. Hence, we used borrowed HTML’s content editable attribute to allow us to make multiple changes within the table and then pack everything into a jQuery array when needed to communicate the information to the server.

API Interface and Save Changes

Here we are using an API interface to process and store any information from the user’s side. This includes either from the GUI frontend or pre-programmed web request. The design is to assign a unique API key to each user with varying privileges so that they can make request to the API endpoint for updates.
Recycle Bin

The entries within the semester page are never meant to be deleted permanently, hence, we made a separate page where user can access all the deleted entries and if they so wish can also recover them as needed. The only true way to delete the entries for good will be to delete the whole term from the home page.

Standards and Constraints

Applicable Standards

This project will run on servers used by the WMU Computer Science department. This is a Linux environment, but the application should be accessible from any browser on any system within the WMU network (including with VPN access). Some applications will need to be present in the server to run the application: python (with additional python packages) & MySQL for the application, and NGiNX & uWSGI for hosting.

Potential Improvements to the backend will need to be done with Python/Django and MySQL. Frontend improvements will need to be done in the Django templates using HTML/Django with bootstrap styling and JQuery functionality.

Constraints

Cost

There are no associated costs with the development of the project. The materials used were the developers’ machines, and free cloud storage with Github.

The cost to run the deployed application are the cost of running it on the server. This application accounts for a trivial portion of the traffic on the server.

The final cost to consider is maintenance. This application is not yet complete, but when it is there may be a need to have a developer make upgrades. If this is done as a student project, there won’t be a financial cost. However, if the maintenance work is done by a professional developer, then this could have a substantial cost.

Scope

The scope of this project is the import, database storage, display, and management of schedule data.

Time

The time constraints for this project were to complete our work during Spring and Fall semesters of 2021.
Ethical Analysis

Identify the Moral Issue

In our application, we hold the entire schedule of the semester. This contains information about the location of every single instructor at specified times. Although information about each instructor’s schedule is available elsewhere, this application contains every single instructor in the computer science department in a single place. Anyone who has access to this site will be able to gain information about the locations of any instructor they wish. Is it morally sound to hold all the information about each instructor’s locations in a single place?

Identify Additional Facts Helpful to Making the Decision

1. How many other resources are available for finding information on schedules?
2. What is the level of sensitivity of the information provided?
3. Who has the ability to view this site?

Identify the Alternatives Available

An alternative could be a serverless design that would allow the user to upload a schedule and work on it in that working session, then export the new copy. This would not save the information in location available to many people. Once the user is done working on the schedule, the information would be removed. This alternative would give the user all of the functionality of the program and remove the ability for instructor’s locations to be accessed in a single place.

Identify the Personal Impacts to the Decision-Maker

The Computer Science Department at Western Michigan University has an extensive list of instructors available. In the general sense, if an individual were to take information from this site to negatively affect the life of any of the instructors, the department would lose credibility with the current instructors and any future candidates.

Ethics Assessment According to Contemporary Theories of Business Ethics

Shareholder Theory

The Shareholder theory is a view that the intended reason for making decisions is to maximize the profits generated to benefit its shareholders. In this specific instance, a breach would cause the
computer science department to lose credibility with the instructors that they hire. The loss of credibility would mean a smaller pool of potential instructors in the future making the entire computer science degree less credible and less competitive. This would affect the reputation of the department and make it harder to get students to enroll with this degree.

On the other hand, not using this application would make it more difficult to create a smooth schedule for each instructor. The result of having tough schedules would make it so the instructor would not be as interested in working for the department. Having a reputation for bad scheduling would weaken the pool of potential instructors and make enrollment significantly less with having a less competitive degree.

Though removing the ability to view instructor’s schedules would help the instructors be at less risk of a potential attack, having tough schedules for every instructor is very undesirable for employees. Information is available elsewhere, but a decrease in the instructors would significantly hurt the shareholders. Therefore, it is not morally incorrect to hold the information about instructor’s schedules.

**Stakeholder Theory**

The stakeholder theory states that everyone who is affected by this application should receive some sort of benefit. By not retaining the information about all instructors’ locations, there would be less risk that an instructor leaves the University. With this being true, the students, faculty, and employers would benefit since the student would have a more consistent schedule.

However, by not retaining the information, some classes could be missed due to not being able to handle conflicts properly. The scheduler application allows for a smoother schedule that would not create stress for the instructors. This would allow for efficiently scheduling each instructor and creating a schedule with the most classes available. Having more classes available will create a better chance for the students to succeed. Students learning also benefits the school by creating a reputation for higher education and it also benefits the employers since they will have more qualified candidates.

Though losing any instructor would significantly hurt the integrity of the schedule, the scheduler application very minimally increases this risk. It is more important to have an efficient schedule that offers the most to the students attending. Therefore, the stakeholder theory states that it is not morally incorrect to hold the information about instructors’ schedules.

**Virtue Theory**

The virtue theory states that any type of business decision should not undermine the rights of any person. As said above, it is not a great practice to retain potentially sensitive information. On the
contrary, this information can be accessed elsewhere and the consequences of not doing it may result in loss of staff or an inefficient schedule.

**Conclusion**

The question we are concerned with is if it is morally sound to retain information about the location of every instructor in the computer science department. The shareholder theory argues for retaining the information. The stakeholder theory also states that it is morally sound to retain this information to create better schedules. The virtue theory agrees with the prior theories stating that it is morally sound to retain this information. After assessing the theories with our project, we have found that 3 out of 3 theories agree that it is morally sound to retain the information upon the instructor's locations. Therefore, we recommend retaining this information to create efficient schedules for the computer science department.

**Health and Safety**

Our application does require the users to use a computer to complete the intended task. Extended use of the application could cause computer vision syndrome. This may result in eyestrain, headaches, blurred vision, dry eyes, and/or neck and shoulder pain. This could be a result of bad posture, poor lighting, glare on the screen, or incorrect viewing distances. To combat this, we recommend using the 20-20-20 rule. This means a 20-second break every 20 minutes to view something 20 feet away. (American Optometric Association, n.d.)

Aside from the risks of using this application for extended periods of time, we have found that there are no other health or safety risks for the users.

**System Design**

Our application consists of two major branches, the frontend, and the backend. The frontend in our application is a client-side server that allows the user to interact with the website and make changes to any schedule they import. The backend in our application is a server-side and holds the MySQL database containing the schedule.

The frontend contains all the functional requirements that require user interfacing. The user interface allows the user to edit any information on any page, add a new row, sort each column, search each record, delete rows, and save to the database. The frontend pushes these changes to Django where data is handled and sent to the backend.

The backend contains the storage of the data. This backend consists of a database in MySQL that has two major functions. The first function is receiving data from the frontend passed by Django. This
allows the user to interface and save their information. The second function of the backend is to allow Django to extract information from the database and give it to the frontend to display to the user. This design allows for easy development of new features that require interfacing and saving information. The communication between the frontend and backend is made quite simple and clear by using Django as a medium.

Database design

The database uses the entity-relationship model. This model uses a table to represent each entity that is used and needed to complete the functionality of the application. The entities in the database are instructors, meeting times, rooms, and semester. Every instructor, room, and meeting time have a one-to-many relationship with each of the classes meaning any of these three entities could be assigned to multiple classes in that semester. Building the database with this design makes it clearer in assigning each class its required information. The figure below depicts the relationship between our tables.
Testing

Our approach to testing was to use test-cases. We met with our client every two weeks to demonstrate functionality and solicit user stories. These stories created test-cases that were used during development to assure components worked and the UX was smooth.
Results

Realization of Requirements

The team was not able to reach all the requirements provided by the client. The first requirement that was completed was the ability to import an excel file provided by the University. The import stores all the information into the database like the previous version of the application. The user is also able to dynamically edit the information on the semester and assign instructors, rooms, and times to every single class. After editing the user will have the ability to save these edits to the database. Finally, the last requirement that was met was the recycle bin feature. Any deleted class can be recovered on a different page.

Some requirements were not met due to time constraints on the project. The first of the unsatisfied requirements was conflict detection. Currently, it is not implemented, and the user will not be notified of any conflicts. The other requirement not satisfied was user authentication. The page currently only has one user permission which is complete access.

The initial assessment of this project was that not all requirements would be implemented in the given period. However, for the requirements not met, the structure of the application was built to allow these other ones to be developed in the future. The entire application was built to update the first version and make user interaction more efficient.

Realization of Standards and Constraints

The server can run as required. All necessary dependencies are installed and functional.

Testing results

All testing of the met requirements using the test-cases resulted in full functionality.

Future Work

With our best efforts we couldn’t implement all the features within the given amount of time. Hence, there are few things that the future team can flush out to make the application work flawlessly.

User authentication

Currently anyone who has the URL has the full access to the site. Hence, restricting access to only registered user would be something to be implemented in the future. The registration process can only be done from the Django’s admin site which only a super user can access and assign specific accounts and privileges for a user.
Conflict Checking

This feature was not finished during our limited development window. At the time of this report, implementation was roughly 30% complete. We achieved detection of instructor time conflicts. We did not reach room conflicts or user notification.

Foreign Key relation

Currently the web app is not able to readout the specific foreign key relation and display the appropriate values. For an example, the Instructor column on the semester page is displays a number instead of the name of the instructor. We weren’t able to have appropriate field displayed due to the lack of inner working of Django’s foreign key field and the future team can work on this and have the table fields output more flushed out and friendly to the front-end user.

Expansion to different Departments

The design is made in consideration with the possible expansion to different departments, however, the implementation and testing for the support of different departments was not done. Future team can build in a new support for different departments and user privileges according to the department by limiting the accessibility and visibility of certain pages for a given user.

Exporting a file

Allowing the user to export the information to a spreadsheet would be one of the requirements for handling the data outside of the web app. This can allow sharing of information with the people who only need temporary access to the information and do not need to utilize the service of the web app. This also work as a backup to the information outside of the current database.

Conclusion

Prior to this project, WMU Computer Science administrators scheduled instructors using an application built by a faculty member. This application allowed a user to upload an excel schedule file, edit and add instructors to the schedule, and export the file as an excel. Usage of this application was somewhat cumbersome requiring that editing be done on a separate page. Detection of schedule conflicts was another issue as this was done manually by the user. The creator of this application sought our team to build a new application from the ground up using the Django framework.

This new application would achieve all of the functionality of the existing application and improve the user experience. The new app would allow users to login to the system with three levels of privilege: basic, administrator, and superuser. Basic users can view schedules. Superuser has access to manage
the entire application. Administrator users can upload, edit, and export. Most of the critical features the application needed to fulfil were for the use of the Administrator users.

Our use of test-case testing meant that the application runs smoothly for all operational features. Through frequent client meetings we were able to elicit user stories to cover all of the critical functions.

While the application is not yet complete, it fulfils many of the requirements. From the outset of the project, our client made clear that our goal was to produce an application with as many features as time allowed, and a foundation for further development. We developed an effective application that satisfies the standards and constraints of the project while also providing a foundation for future work.

References


Appendices

Project Management Plan

Our project utilized Agile Methodology and Github’s project planning features. We used 1-2 week sprints (varied depending on team and client schedules). We met with our client every other week for 15-30 minutes. These meetings were to demonstrate new functionality and elicit feedback.

Development Costs

Institution Costs

~$0—This application only uses a negligible portion of all energy and processing power consumed by the servers.

Sponsor Costs

$0—No costs to the sponsor

Team Costs

$0—No costs for the development team.