FINAL REPORT
INSTRUCTOR-COURSE ASSIGNMENT APPLICATION

PROJECT MANAGER: KIANA MCDANIEL
DAVID JOHNSON, PHILLIP TRAN

TEXAS STATE UNIVERSITY
INGRAM SCHOOL OF ENGINEERING

TEXAS STATE UNIVERSITY
601 UNIVERSITY DR
SAN MARCOS, TEXAS, 78666

05/01/2019

coursebrew

TEXAS STATE UNIVERSITY
The rising STAR *of Texas*

1 OVERVIEW

1.1 Executive Summary

Our Senior Design project was sponsored by Dr. C. Rich Compeau and Texas State University. Our team consisted of three computer engineering majors. Our project's purpose was to provide Program Coordinators at Texas State University's Ingram School of Engineering with a tool that would allow them to more easily perform instructor-course assignments. The solution decided on was a server-based program that could be accessed through a webpage to allow for multiple users, remote access, and the ability to store and retrieve data as needed. The project was overall successful, the final product does everything it was intended to, albeit not perfectly and it can be improved upon. The RESTful API design worked very well to provide a system with low latency and low memory usage, SQLite worked well to provide a simple database solution although it may need to be replaced as the data size and number of tables increases. Using three different systems, and a total of five different languages made the project more complex than it needed to be and is probably the biggest detriment to the system.

1.2 Abstract

The nature of instructor-course assignment is tedious and challenging due to the amount of information needed to make each assignment. With our Senior Design project, we aimed to develop a web-based application (coursebrew) for Program Coordinators that simplified the process of instructor-course assignment. In addition to providing an intuitive user interface, coursebrew makes instructor-course recommendations and warns the user when different assignment criteria are not met or over met. The software architecture used to build coursebrew is modular, which allows future developers to easily manage and scale the system.

Contents

2 LIST OF FIGURES

3 LIST OF TABLES

4 PROBLEM DESCRIPTION

Every semester at Texas State University, Program Coordinators have the task of assigning instructors to courses. This task is simple in concept but becomes much harder to carry out when you add in all of the relevant information such as instructor's preferences for specific courses, instructor's availability and preferred workload, and other variables. Currently, Program Coordinators use an outdated, hard-to-navigate Excel Spreadsheet to assign instructors to courses. At the time, the Excel Spreadsheet, though inefficient, was the only way to manage and assign instructors to courses. Our project deliverable is an intuitive, user-friendly web-based application that provides a clear process of instructor-course assignment.

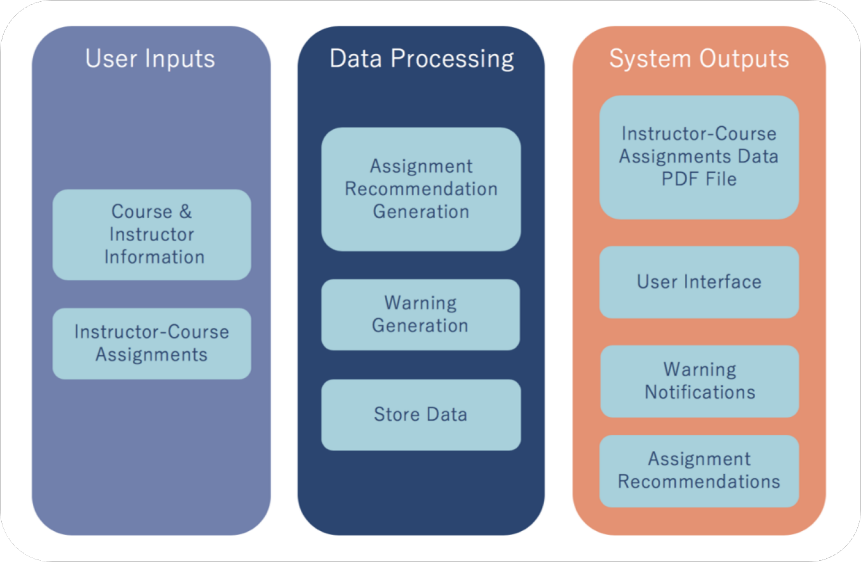

Figure 1 Input-Output Chart – displays the inputs and outputs of our system.

It should be noted, the entirety of our system was designed and implemented by our team.

5 PROGRESS TOWARDS A SOLUTION

5.1 Design Decisions

Figure 2 illustrates our system design and the primary design decisions made throughout it. All of our design decisions are guided by the desire to create an ergonomic, efficient, modular system. In order to make the system modular we decided to separate the data processing from the frontend display. Angular was used to generate the webpage since it allows us to create a live application and handles the variable data from the server well. (sentence about other options). Python was chosen to implement the bulk of our data processing. This decision was made since it is a relatively simple language to learn and includes support for several interfaces for potential future work. Flask was used to implement the Python code since it provides a RESTful API, meaning that the program is stateless and pushes the bulk of the application data to the client-side application, allowing for less refreshes and a significantly reduced amount of data transfer. The database solution we selected was SQLite, originally, we were going to use PostgreSQL, however the scope was small enough that SQLite was more efficient while still providing the versatility and low time-complexity of SQL. The usage of a SQL database as opposed to a text file or other data solution allowed for concurrent accesses which allows simultaneous users.

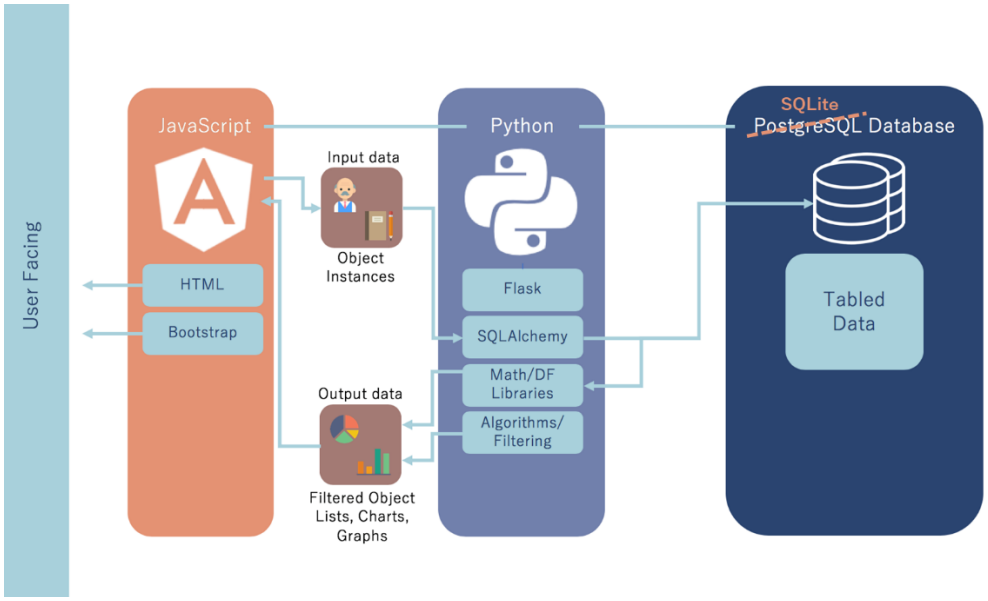

Figure 2 Software Architecture Design – the software design of our system, technologies used, and relationships between them.

5.2 Design Approach

A top-down approach was used to design our system. We first determined the overall system architecture we would need for basic deliverables, then the specific software interfaces and systems we would have to implement, and finally programmed the individual modules to implement those interfaces and systems. Figure 2 shows the software tools used in the project itself, but in the design process we had to use additional tools to set up our developing environment. We used the Ubuntu distribution of Linux to set up a server to host our software, we also used npm and pip (package manager software for Linux) to acquire the different software tools we would need for the project.

5.3 Project Approach

We settled on the specific software used for the project early on. Once we decided to segment the project into discrete parts, we chose to have each engineer focus on a separate part of the project, specifically the frontend design, the data-handling and logical scripts, and the software framework and interconnection. The progress in each part went on apart from the other two, although in general each engineer was working on the same functionality. For example, the webpage to handle creating a new instructor was created when the Python script to handle generating a new instructor in the database based on JSON data was being written.
Once the initial webpage was created, we would establish a set format of JSON data to be sent back and forth between Angular (HTML/JavaScript) and Flask (Python). This allowed us to work on each primary function apart from the others.

5.4 Engineering Standards

Standard	Title	Application	Relevance
ISO/IEC 9075	Information technology – Database languages – SQL	SQLite uses the SQL standard, as do all of our queries to our SQLite system	Data storage
RFC 8259	The JavaScript Object Notation (JSON) Data Interchange Format	We use JSON to transfer data between the Angular frontend and Flask backend	Data transfer

Table 1 Engineering Standards

5.5 Progress Towards Goals

Once we settled on the design for our project, the process of getting from start to finish for our deliverables was pretty cut and dry, although there were a few bumps. We settled on using Angular, Flask, and PostgreSQL shortly after starting the project, and were able to make decent headway in developing the frontend interface for the project. Once the frontend was developed, we started working on the different functions of the project, adding and editing instructors and courses, as well as displaying the lists of them. By the time these were finished, we realized that it would be easier to transition to SQLite and store our database in a single file rather than trying to interface with a third system. The next steps were in developing the rest of the functions such as assignments, assignment recommendations, and warning algorithms.

At the beginning of the project, our sponsor wanted us to interface with a software called UniTime which would be able to handle the class scheduling once our software was used to make instructor-course assignments. As we worked through the instructor-course assignments, we realized that our software would not be able to interface with UniTime, this was an oversight on our part, as we did not realize that UniTime changed its formatting with every update. We looked at UniTime when our sponsor initially requested it early in the project, and at the time the format that UniTime used would work with our system, but sometime during the project, UniTime was updated, and we realized that if UniTime updates its file format and how it takes in

data frequently, then our software would have to be constantly updated to work with it. Under the advice of our sponsor, UniTime was then removed from the project since the goal of this project was to be stable without a need for updates.

Another issue we ran into in the second semester was the login system. Ideally, we would have our software interface with Texas State University's login system, since our software would be deployed on Texas State University Server's, to be used by Texas State University faculty. When we contacted ITAC (the University's IT department), we were informed that the entire project would have to be finished before they could begin to evaluate it and offer login security for it. Since the evaluation process could take several months after our project was finished, we developed a temporary username+key login system to be used in its place.

5.6 Verification

Because of the nature of our project, it was difficult to verify the functionality of the program, the vast majority of our functions are either pass or fail, and don't have a quantitative measure of success. The response time and memory usage were both verified throughout the development process with the assistance of Google's Chrome developer tools. Tests 1-7 shown in Figure 3 were all tested and verified visually throughout the design process, the entire database being populated multiple times by hand with the add courses, and add instructors functions, then verified by the view courses and instructors functions. The assignment recommendation was tested by setting instructor-course assignments to match a partial set of assignments from a previous semester, then verified by observing that the instructor who was actually assigned to that course was one of the recommended instructors. The security would be verified by using a middleman application that would try to decrypt packets, however since we haven't been able to implement security measures, the tests were not performed.

5.7 Characterization Results

	TEST CASE	SPECIFICATIONS		RESULT/COMPLIANCE
		input	expected output	
1	view courses	user mouse clicks to navigate web-page database's stored course data	list of all courses currently in the system for current school year shown	http://198.58.116.113:4200/courses
2	add course	user mouse clicks to navigate web-page user keyboard inputs to fill out course information	a form will appear that shows data fields to be entered by the user added course shown in "View Courses" page	http://198.58.116.113:4200/add-course
3	edit course	user mouse clicks to navigate web-page user keyboard inputs to fill out course information	a form will appear that shows data fields filled with current course data edited course shown in "View Courses" page	http://198.58.116.113:4200/edit-course/EE2400
4	view instructors	user mouse clicks to navigate web-page database's stored instructor data	list of all instructors currently in the system for current school year shown	http://198.58.116.113:4200/instructors
5	add instructor	user mouse clicks to navigate web-page user keyboard inputs to fill out instructor information	a form will appear that shows data fields to be entered by the user added instructor shown in "View Instructors" page	http://198.58.116.113:4200/add-instructor
6	edit instructor	user mouse clicks to navigate web-page user keyboard inputs to fill out instructor information	a form will appear that shows data fields filled with current instructor data edited instructor shown in "View Instructors" page	http://198.58.116.113:4200/instructors/SAA
7	make an instructor-course assignment	user mouse clicks to navigate web-page database's stored course and instructor data	a dropdown list of instructors will be shown with top 5 "best-fit" instructors at the top of the list any appropriate warnings generated and displayed assignment reflected on the user interface	http://198.58.116.113:4200/courses/EE2400
8	assignment recommendation	database's stored instructor data	top 5 "best-fit" instructors listed in order of what the system has calculated to be most ideal to least ideal	visually confirmed to generate list containing professors commonly assigned to that course
9	PDF export	user mouse clicks to navigate web-page	PDF containing all instructor-course assignments for the current year	Export successfully represents assignments
10	response time	standard operation of the program (simulated by first 9 tests)	response time below 50ms on average	verified to be on average 38ms, at most 48ms
11	memory usage	standard operation of the program (simulated by first 9 tests)	memory usage below 500MB	verified to be on average 43MB, 400MB with browser bloating
12	security	standard operation of the program (simulated by first 9 tests)	encrypted data streams transmitted data not recovered by entities outside of the system	failed
13	system stability	standard operation of the program (simulated by first 9 tests)	webpage not "crashed" by a series of statistically random inputs	passing by monkeytest, must be reverified whenever program is changed

Table 2 Test Cases

5.8 Deficiencies

Deficiency	Effect	Solution	Time need to solve
encryption	Data sent from server to client is not secured and could be intercepted if someone were so inclined	The easiest solution would be to program encryption and decryption on the endpoints of the system, at the point where Python and JavaScript create and receive JSON packets	It would likely take about 2 weeks to design and implement, then an additional 3 weeks to debug and fully integrate
login system	Login system can be easily broken into, allowing someone to corrupt the database if they had access to the webpage	The best solution would be for the program to have Texas State University's login system embedded into it	It would take anywhere from 2 weeks to 3 months for ITAC to evaluate the program
class scheduler (stretch goal)	Overall project is incomplete (although our scope is finished), since Program Coordinators still need to perform additional work outside our program to complete their job	The system is designed to be modular, so the overall design can be expanded to include class-rooms, times, and student count	Based on the time it took to develop the instructor-course assignments, and the increased complexity of class scheduling, it would likely take about 3-4 months to fully develop the class scheduling functionality

Table 3 Deficiencies

5.9 Iterations and Redefinitions

Our original project definition as outlined in our Statement of Work is:

> "Our product is an intuitive, web-based application that allows University Program Coordinators to assign professors to courses and export this data as an XML file. Currently, Program Coordinators use an outdated, hard-to-navigate Excel Spreadsheet to assign professors to courses. This method is inefficient, as it causes users to spend more time figuring out where features and data are located, rather than simply completing the task. Our product will benefit the sponsor by having a user-friendly, easy-to-navigate interface that provides a clear guidance on the process of professor-course assignment. By integrating the XML export feature into our product, Program Coordinators will be able to easily import this data into UniTime, a Class-Scheduling open-source software. Our product development team consists of three Undergraduate EE/CE students who are advised by a Sponsoring Faculty Member and two Student Mentors. We will be designing and developing the front end and back end of this web-based application using various programming languages and technologies. The project work will take place on and around Texas State University, San Marcos Campus. Our product will be ready in May 2019."

One of the biggest changes in our design was the removal of UniTime as a requirement. As previously discussed, UniTime was determined to be unsuitable for our project, so its removal made sense. This change occurred halfway through the second semester, and didn't negatively impact our project, since it was not integral to our design. The main lesson from this change was

that in any design it is necessary to fully research any component of the design, and to avoid relying on external resources that you cannot ensure are consistent. A big change between our original and current project definition is that we are much more specific about how we implemented the design, mostly due to refinement and completion of our design.

6 CONSTRAINTS

6.1 Budgetary

Our project consisted solely of software. There are immense amounts of open-source technologies and resources that exist on the world wide web. All of the technologies and resources used to implement coursebrew were open-source. The server used to run coursebrew was already owned by one of our team members, Phillip Tran. No university funds were used to develop coursebrew.

6.2 Design Feasibility

Although each of us on this team are Computer Engineering concentrated, we did not have much prior experience building a full-fledged application. Building applications and using the technologies we choose was not something that was taught to us in our Computer Science courses. This slightly limited us and caused us to go through a lot of trial and error when developing coursebrew.

6.3 Manufacturability

In the real world, software applications are designed to be manageable by a System Administrator. In our case, this would be a student or future Senior Design team building on to our project. When we were designing our system, we took this into consideration from the start by developing software that is modular, manageable and scalable. Each one of these characteristics allows a future System Administrator or Senior Design team to manage coursebrew or build on to its functionality. In order to permit the ability of coursebrew to be used and worked on in the future, we provide Developer documentation for the entirety of our system.

6.4 Maintainability

The software architecture of coursebrew is designed to be modular. The database API is integrated into the frontend of our system via Angular Injectables. All of the frontend components are separated (home page, instructor, course, etc.) into their own modules and classes, which makes maintenance more accessible. The backend has maintenance functions so system administrators can manage the database outside of the frontend application. Developer documentation is provided for the frontend and backend of our system, along with directions to create an instance of coursebrew on a server.

6.5 Environmental

Coursebrew is an application that is viewed and used completely via a computer screen. The outputs of our system are all viewed digitally. One feature of coursebrew is the ability to export a PDF containing all instructor-course assignments. If desired, the user may print a copy of this

PDF, but it is not necessary and not a requirement of our system since the information can be displayed and shared digitally.

6.6 Health and Safety

When designing coursebrew's user interface, we considered the strain a user might feel on their eyes from looking at a computer screen. With this in mind, we designed coursebrew with soft colors and low flashing to reduce eye strain and avoid potential epileptic reactions.

6.7 Social

The intended users of cousebrew are Program Coordinators, Program Directors, Administrative Assistants, and System Administrators. Access to the program itself will be restricted to these authorized individuals via a username and password combination unique to each user. The program is restricted to these users to avoid potential conflicts of interest or misunderstandings from other individuals involved in the University.

7 BUDGETS

As stated in section 6.1, during the course of this project no university funds were used. Below is an estimated total project cost, which include capital expenses and non-capital expenses.

Estimated Costs			
Cost Category	Materials	Cost	Actual Dollars Spent
Capital Expenses	This project required no pre-existing infrastructure, or other forms of capital expenses to be completed	$0	$0
Non-Capital Expenses	3 laptops and a desktop computer - system implementation/testing	$2,750	$0
	Intel NUC - system deployment/public environment	$350	$0
Total		$3,100	$0

Table 4 Estimated Project Costs

8 WORK SCHEDULE

Major Tasks			
	Task	Team Member	Completion
1	software architecture (research and implementation)	Phillip Tran	yes
2	user interface (design and implementation)	Kiana McDaniel	yes
3	assignment recommendation (design and implementation)	David Johnson	yes
4	warning generation (design and implementation)	David Johnson	yes
5	software interfaces (connection and debugging)	Phillip Tran	yes
6	PDF export (design and implementation)	David Johnson	yes
7	log-in system	Phillip Tran	yes
8	database (design)	David Johnson	yes
9	database (population)	David Johnson & Phillip Tran	yes
8	testing	ALL	yes

Table 5 List of Major Tasks

9 PERSONNEL INTERACTIONS

9.1 Teamwork

Teamwork	
Team Member	Responsibility
Kiana McDaniel	User Experience Architect - user interface design and implementation Project Manager
David Johnson	Backend Developer - program scripting - assignment recommendation - warning generation - data formatting)
Phillip Tran	Software Engineer - program scripting - program interface - server developer

Table 6 Team Members and Responsibilities

9.2 Mentorship

Our sponsor, Dr. C. Rich Compeau, played an important role in defining the requirements of this project, and provided excellent guidance and feedback to us throughout the entirety of this course. During our meetings, which occurred 1 to 2 times a month, we would update him with our progress and ask any questions we had about the requirements. He was very prompt in providing us with feedback and any resources we required. Dr. Compeau also pointed us towards Dr. Stan McClellan, who challenged us to think more about the scope of our project and helped us brainstorm solutions to key challenges related to instructor-course assignment. Our faculty advisor, Mr. Lee Hinkle, continuously motivated thinking within our team and gave us early insight on defining the hardware requirements in our documentation. He also pointed us towards Dr. Jill Seaman, who guided us on defining our software requirements for this project. We really appreciate all of the guidance we received throughout this course from all of the faculty members that advised us.

10 ETHICS

In working with Dr. Compeau on the instructor-course assignment project, the design team has become privy to particular information about the faculty at the University, and as per NSPE code of ethics 1.C we must “not reveal facts, data, or information without the prior consent of the client or employer except as authorized or required by law or this Code.” The majority of this information is in regard to various faculty’s position in the university, their course preferences,

and other information which should not be common knowledge among the student body at the University.

Since this project is entirely software based, there is little need for discussion of health and safety of its users, although the software has been designed with a soft color scheme and a lack of flashing lights to avoid eye fatigue and epileptic reactions.

11 SUMMARY & CONCLUSIONS

The primary goal of this project has been to provide an instructor-course assignment tool to the Program Coordinators of the Ingram School of Engineering at Texas State University. This tool consisted of a webpage frontend that was based in Angular which provided a framework for JavaScript, HTML, and CSS, a Python server built on Flask which provided a clean interface to the Angular frontend through JSON packet transfer, and a SQLite database stored inside of a single file stored on the server. These different software parts served to form a RESTful system, which allows the system to have a very short response time, and low memory usage on the server.

The system allows for multiple users to have access to a set of functions including adding, viewing, editing, and removing instructors and courses, assigning instructors to courses, viewing a list of warnings generated regarding those assignments, and export a list of all the assignments made in PDF form. The system also provides a framework to keep backups of the database to undo potential mistakes. The biggest deficiency of the system is its current security, both in data transfer and in access. There is at present no form of encryption provided for the data being sent between client and server, and the current method of securing access to the server is through a simple username and password combination that is more a placeholder than any actual security.

Overall, the deliverables we have made satisfy almost all of our original objectives. Our statement of work lists a homepage, instructor and course management, warnings and assignments management, UniTime export, system documentation, database, and Texas State University login system as the main deliverables. All of these have been completed apart from UniTime and the Texas State University login system, both objectives were removed from our scope by our sponsor. In light of this information, our deliverables satisfy the main project goals.

12 DISCUSSION

12.1 Academic Preparation

Overall, our Texas State Electrical Engineering courses provided us with the critical thinking and problem-solving skills used to design and develop this project. Since this project is purely software, the hardware knowledge gained from our EE courses were not used. From our EE courses, we gained knowledge in designing and breaking down large systems into smaller, more discrete, manageable pieces. This aided us in designing and developing coursebrew to be modular, scalable and manageable.

All three team members have a concentration in Computer Engineering. This concentration includes completing multiple Computer Science courses. The knowledge gained in these courses helped us when defining data structures, test cases, and the object-oriented principles behind the software architecture of coursebrew.

Many of the skills needed to implement coursebrew were not provided through formal education. Our greatest resources throughout this project were Google and API documentation.

12.2 Lessons Learned

As a team, we learned that the engineering process takes strong research, planning, and critical thinking skills. The ability to clearly organize and explain your thoughts is important in the design process. The ability to research and comprehend new skills is important in the implementation process. We learned that having the ability to work cohesively as a team is one of the strongest skills to possess. We learned to give and take criticism with a light-heart.

12.3 Soft Skills

Senior Design gave us the opportunity to improve on our teamwork skills, presentation skills, technical writing skills and problem-solving skills. These skills were improved upon through constantly working together as a team, presenting the design of our project numerous times, and completing the documents required by this course. With each presentation, document, or problem we faced, our skills grew stronger.

12.4 Schedule Deviations

During the first semester of this course each one of us had the most rigorous schedule of our college lives. This was something that was out of our control, since the courses we took along with Senior Design were all required in order to graduate on time. All of us also worked (and still work) jobs during the week and weekends. These hectic schedules caused set backs on the amount of work we could complete on the actual project itself. If we could go back, we would try to work even harder to stay on track during the first semester.

The second semester of this course ran a lot smoother for us. Each week we made huge strides in the progress of our project.

12.5 Staffing

This project could not have been more perfectly staffed. Each one of us are EE/CE concentrated, which means we had enough background knowledge to have an idea of what we were doing with a fully software project. The combination of each of our skills and knowledge is what has led coursebrew to become what it is today. Any more engineers and there would have been too much work done just coordinating our efforts, any less and we would have been overwhelmed.

12.6 Final Observations

If we could have done this project over again, we would have tried even harder to get more of the actual project complete during the first semester of Senior Design. Another thing we would have done differently is made sure that we had a complete database schema to start with. During the middle of this second semester we had to reinitialize the database a few times because there were

attributes that were missing from our tables. These attributes were key to part of the functionality of our system. Another lesson learned (in regard to UniTime) was to fully look into any outside software interfaces we were required to use.

13 ACKNOWLEDGMENTS

David, Phil and I would like to thank Dr. C. Rich Compeau, Mr. Lee Hinkle, Dr. Stan McClellan and Dr. Jill Seaman who all helped us throughout this course.

14 REFERENCES

- Angular Documentation https://angular.io/docs
- Flask Documentation http://flask.pocoo.org/docs/1.0/
- SQLite Documentation https://sqlite.org/docs.html
- Bootstrap Documentation https://getbootstrap.com/docs/4.3/
- W3Schools https://www.w3schools.com/css/default.asp
- Python Guidelines https://docs.python.org/3/
- JavaScript Style Guidelines https://www.w3schools.com/js/default.asp
- UniTime Documentation http://help.unitime.org
- Stack Overflow https://stackoverflow.com