NORTHWEST NAZARENE UNIVERSITY

Counseling Department Scheduling and Accreditation Tool

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Zachary Garner
2019

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Zachary Garner
2019

Counseling Department Scheduling and Accreditation Tool

Author: 
Zachary Garner

Approved: Barry Myers
Dr. Barry Myers, Department of Mathematics and Computer Science,
Faculty Advisor

Approved: 
Dr. Curtis Garner, Department of Counseling Education, Second Reader

Approved: Barry Myers
Barry L. Myers, Ph.D., Chair, Department of Mathematics & Computer Science

ABSTRACT

Creating a Simple and Easy to Use Scheduling and Accreditation Application Based on Existing Scheduling Documents Used by NNU's Counseling Department.

GARNER, ZACHARY (Department of Mathematics and Computer Science), MYERS, DR. BARRY (Department of Mathematics and Computer Science).

For department chairs and their support staff, scheduling courses can be some of the most tedious and time-consuming work they are involved in. Currently, courses are hand scheduled and checked to ensure they do not create conflicts for students in any semester of the program. A need exists for an automated scheduling application that can check for schedule conflicts and craft a complete schedule accordingly. Calculating information for the Council for Accreditation of Counseling & Related Educational Programs or CACREP Accreditation is a similar situation for NNU's Counseling Department, as many data points must be considered in the calculation, which leads to a lengthy process that ultimately creates little value. The purpose of this project was to create a web-based application that could both create a completed schedule and calculate the CACREP ratio using the same spreadsheets and documents that the department uses in their current approach. It was also important that the project was easy to use and understand so simplicity of design was a key component. The project consists of a website designed using HTML, CSS, JavaScript, and PHP which communicates with a python application to parse data from the uploaded files and formulate output. It is hosted at nnu-counseling-calculator.tk.

Acknowledgments

I would like to thank my friends and family, including Curtis Garner, Mallory Garner, Christina Grubaugh, and Ryan Pacheco for helping me through the entirety of college, as well as their valuable input on this project. In addition, I would like to thank Dr. Curtis Garner for envisioning this project and encouraging me to make it a reality. Finally, I would like to thank Dr. Barry Myers, and Dr. Dale Hamilton for their guidance and advice throughout my time in the computer science program.

Table of Contents

Table of Figures

Overview

The primary goal of this project was initially to create a program that would completely take over all class scheduling duties for any university department. It would have then been marketable as a product to universities to increase productivity. The revised purpose of this product was to develop a program specifically for NNU's counseling department that could efficiently schedule times for courses using existing documents. The secondary purpose was to use the same documents in order to calculate Council for Accreditation of Counseling & Related Educational Programs or CACREP Accreditation ratios for the department. It was important for the program to be accessible as a website and easy to use.

Background

On the surface scheduling courses appears to be trivial; however, there are many factors to consider when choosing a time for a course. Some of the most important aspects are the availability of instructors at any given time, and potential conflicts between courses that are required in the same semester. Complicating the issue further is the fact that there are three different tracks in the counselor education program: marriage and family counseling, school counseling, and clinical counseling. Each of these tracks has their own scheduling needs, so they all must be considered with constructing a cohesive class schedule.

Before the development of this project, NNU's counseling department had no set procedure for scheduling courses. Class schedules have usually been pieced together on a chalkboard using a combination of Microsoft Excel files called "loadsheets" and Microsoft Word files called "plans of study". This outdated method of scheduling has proven to be extremely time consuming for the department and is consuming resources that could be more productive in another area.

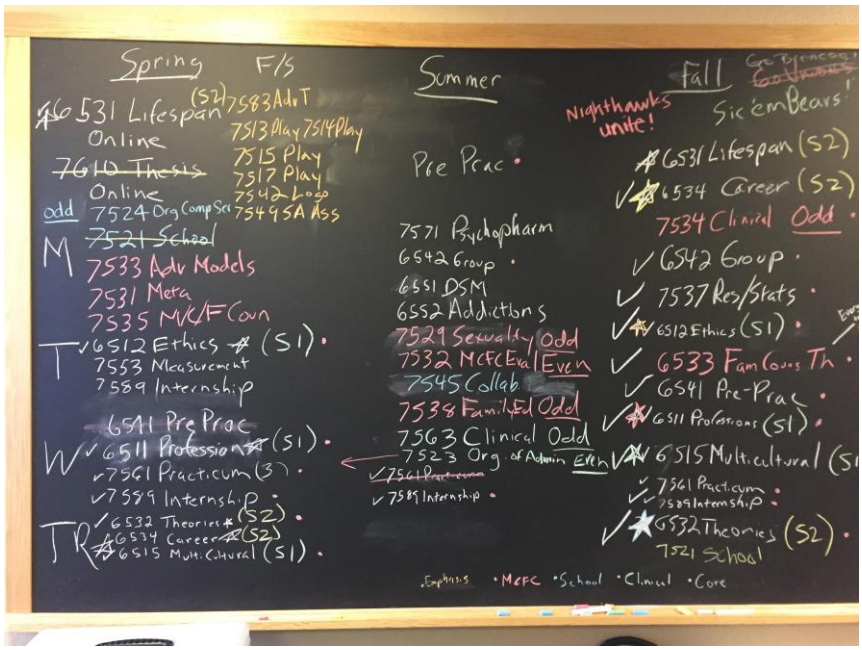

Figure 1 - Current Scheduling Method

In addition to scheduling courses, obtaining CACREP Accreditation is incredibly important for NNU's counselor education program. CACREP Accreditation provides recognition that the content and quality of the program has been evaluated and meets standards set by the profession. By graduating from an accredited program, students increase their chances of getting licensed as a counselor and ultimately landing a job. NNU's counselor education program has to meet certain criteria to maintain their CACREP Accreditation, one of which is the student to faculty ratio. This ratio is based on the number of course hours taken by students and taught by faculty as opposed to a strict headcount of students and professors, which complicates the calculation. Before the creation of this project, NNU's counselor education program was calculating the CACREP student to faculty ratio by hand with a paper form.

Project Planning

As this was a stand-alone project started from scratch, a significant amount of project planning had to take place to ensure a satisfactory product was produced. Firstly, alternative

software packages were researched in order to ensure that the class scheduling problem was either unique enough or valuable enough to pursue. While there is a large variety of professionally developed software that can assist in the scheduling of courses, there are not many completely automated options and the software is generally quite expensive. Analyzing the viability of existing software further proved the need for custom software to be produced for NNU's Counselor Education department.

Next, specific requirements for the project had to be identified. This was accomplished with a series of semi-formal interviews with the stakeholders of the project. These interviews proved to be challenging as they highlighted the difficulties of having meaningful dialogue between technical producers and nontechnical consumers. Still, progress was made and eventually resulted in clear user requirements and a reasonable scope for a one-person team. The project had to: schedule courses partly or completely automatically, calculate the CACREP student to faculty ratio automatically, use existing documents, and have a graphical user interface. Non-required user desires included: ease of use, ability to perform calculations quickly, and availability on any platform or computer. The scope was limited to scheduling courses and calculating the CACREP ratio with all other operations considered to be future work.

After the requirements had been identified, an appropriate language could be chosen for the project. Python was the obvious choice for the backend of the application as it is regarded as easy to write and has many packages for data collection which was an important aspect of the project. Then, a solution for a frontend and graphical user interface or GUI had to be considered. Two possible options were to make the application web based and have an html/JavaScript/CSS frontend or to create a desktop application with a Python frontend. A web-based application was

ultimately chosen as it satisfied more user desires. By hosting the project on a server there would be no need for users to download and install the project or worry about what version they had. The latest version would always be available on any computer as long as the user had access to internet.

Modeling is important both pre and post implementation to help the programmer understand how to implement the project and to inform the user how the project functions. The data flow diagram models the journey the data takes through the application and includes the expected inputs and outputs. It is extremely useful as it helps the programmer to visualize how the data should be transformed at each point of the application. The data flow diagram for the class scheduler and the CACREP ratio calculator follows (Online Diagram Software & Visual Solution).

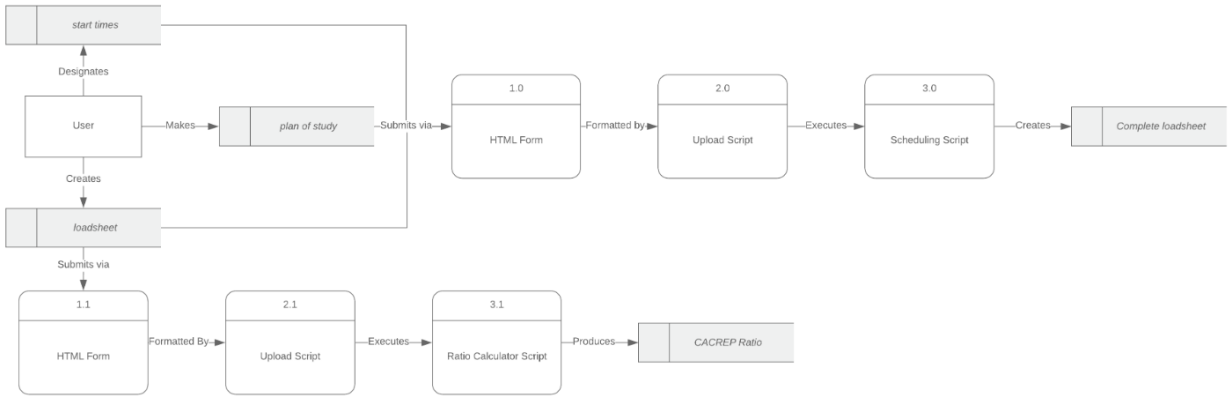

Figure 2 - Data Flow Diagram

A class diagram is another useful modeling tool that displays the data and functions of each class in a project. It also displays the relationships the classes have with one another whether that relationship is association, inheritance, aggregation, or composition. An up to date class diagram makes it easy to understand the inner working of a project and is extremely useful for enabling future work on the project. The class diagram for the class scheduler and the CACREP ratio calculator follows (Online Diagram Software & Visual Solution).

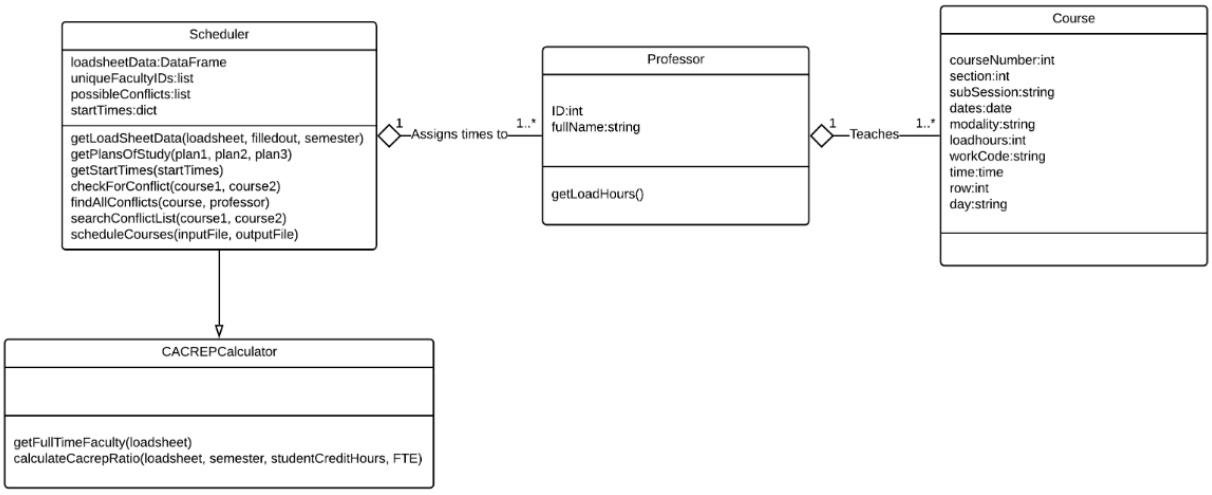

Figure 3 - Class Diagram

By carefully planning, organizing, and documenting before, during, and after the development of the project it was ensured that the project satisfied the user's needs, stayed within a reasonable scope, and was ready for more work in the future.

User Data

Data was pivotal for the success of this project, and a great deal of work went into choosing the correct data source and reading it into the program. As previously noted, the counseling program was scheduling courses using a Microsoft Excel file called the "loadsheet" which tracked the course load assigned to each professor, as well as multiple Microsoft Word files called "plans of study" which detailed the required courses for each specialty track by semester. The decision was made to use these files as input in the program because the counseling department was already familiar with them, and the main goal was to reduce their workload. In addition, a new .csv file had to be created to store the earliest and latest time that courses could be scheduled for each day of the week. This was important to prevent courses from being scheduled at 5:00 AM, 11:30 PM, or other unrealistic hours.

Options for reading in data from an Excel (.xlsx) file in Python are relatively limited. The

Pandas package is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language (Python Data Analysis Library). Not only does pandas have the capability to read in .xlsx data into an easy to use and understand dataframe, but it also is available as part of the easy to install Anaconda environment which made it perfectly suited for this project (Anaconda Python/R Distribution). Packages for reading in data from a table in a Word (.docx) file were also obscure. The Python-docx library was a library mainly for constructing and modifying Word files, but ultimately it did have the capability to pull data from a table in a Word document (Python-Docx). These libraries enabled the use of the legacy documents which will make the change to the new system smoother than if entirely new documents had to be created.

Class Scheduler

Originally the sole purpose for the project, the class scheduling portion of the project remained the focus throughout the project. As work began on the project the most important question became, "How much leeway should the program be given?" Should the program be able to choose the day of the week the class is offered? The dates it is offered? The semester it is offered? Ultimately, giving the class scheduling program more control added more complexity to the program, and the counselor education department was reluctant to cede too much of their autonomy to a computer program, so the decision was made to allow the program to only choose the time of day each course was offered. This allowed professors to choose the day of the week they wanted to teach each course, while the class scheduling application chose a time that ensured there would not be any conflicts.

The completed class scheduling application consists of three parts: an html front end, a PHP layer to transport uploaded files, and a Python back end to perform calculations and update

the files. When files are uploaded via the front end, they are temporarily stored in a folder on the server where they can be accessed. After upload is completed, the PHP script operating on the server calls the Python script using PHP's exec() command. This is equivalent to running the Python script via the command line, and allows command line arguments (the location of the temporarily stored files) to be passed in. Three to five input files can be passed in, one partially completed loadsheet, one to three plans of study, and one file detailing the possible start and end times for classes on each day of the week. The output of the Python scrip is a fully completed loadsheet. In essence, the script assigns a start and end time to each course based on the number of credit hours and professor availability while avoiding the creation of conflicts between courses that must be taken concurrently. The completed loadsheet is then made available to the user as a download. Upon completion of that download, all uploaded and created files on the server are erased by the PHP script. This ensures that the server's file system will never be inundated with unnecessary files, and that the server will not have memory issues even after thousands of load sheets are created.

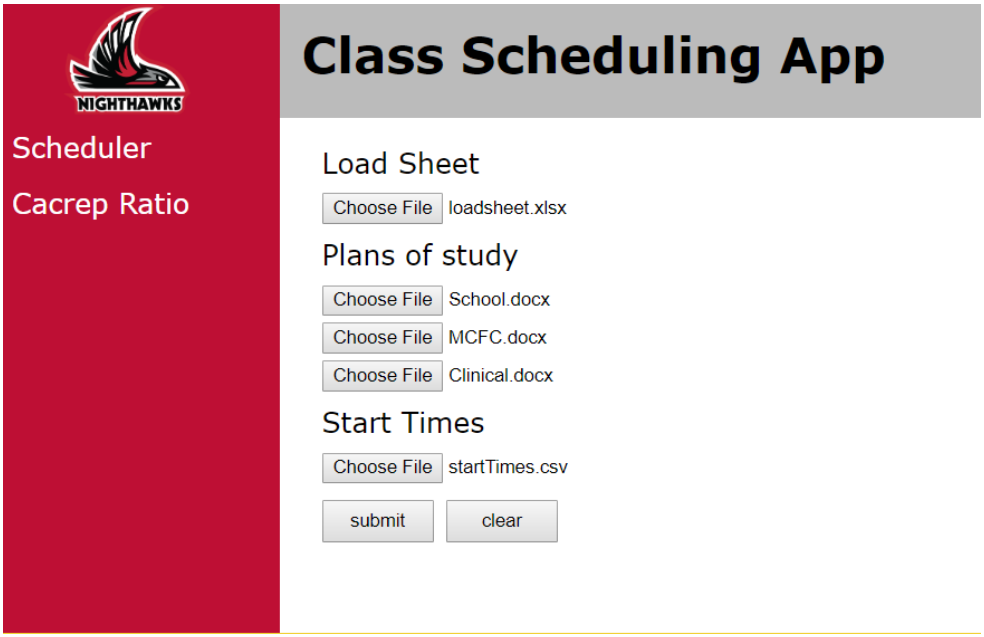

Figure 4 - Class Scheduling Front End

CACREP Ratio Calculator

The secondary purpose of this project was to calculate the student to faculty ratio using the CACREP standards. Originally, this was simply a side project, but after the efficiency of the class scheduler was realized, it was determined that a calculator for the CACREP accreditation student to teacher ratio was a must have for the program. There are several steps to calculating this ratio for a semester, the first of which is calculating the number of full-time equivalent or FTE students are attending the program. This can be calculated by dividing the total number of student credit hours by the number of hours needed to be a full-time student (9 for NNU's counseling department). Next, the number of FTE professors must be determined. The information for this can all be found on the loadsheet. Firstly, all full-time faculty count as 1 full-time faculty unless they are teaching an overload in which case they count as $1 + \frac{overloadHours}{FTEHours}$. Adjunct professors are also part of the calculation in which they count as $\frac{loadHours}{FTEHours}$. The total FTE professors is then the number of FTE full time professors + the number of FTE adjunct professors. Finally, the number of FTE students is divided by the number of FTE professors which gives the CACREP approved student to teacher ratio. While this student to faculty ratio is calculated on a per semester basis, it can then be averaged with other semesters to find the yearly ratio. The goal for each year is to have a CACREP student to teacher ratio of under 12, as that is the ratio required to maintain accreditation (Urofsky).

The CACREP ratio calculator was built in a similar manner to the class scheduling portion of the project. Once again, it consists of an html frontend, a PHP middle layer, and a Python backend that does the calculations. The input for the CACREP ratio calculator consists of an html form containing: one file upload for the loadsheet, a dropdown selector for the semester to calculate, a textbox for the number of student credit hours, and an optional textbox for the

FTE Hours (defaulted to 9). This form is then sent to the PHP server-side script using the HTTP post method which is necessary to send file data, and also keeps the information private. The PHP script then checks the form data sent to confirm it contains the correct file type and data types for each field. Finally, the form data is formatted into a string and passed to the Python script as arguments. After the Python script calculates the CACREP ratio using the formula previously mentioned, the data is passed back to the PHP script and displayed to the user using the PHP echo() function. This setup allows the user to conveniently calculate the student to faculty ratio on demand without having to download files. After completion of the CACREP Ratio Calculator, it was used to determine the CACREP Ratio for the fall 2018 and spring 2019 semesters. The same calculations were also performed by hand as they have been in the past. The results from the automated calculator and the hand done calculations were compared in order to confirm the accuracy of the calculator. The automated CACREP Ratio Calculator produced the correct results which proved it could now be used in place of calculations done by hand.

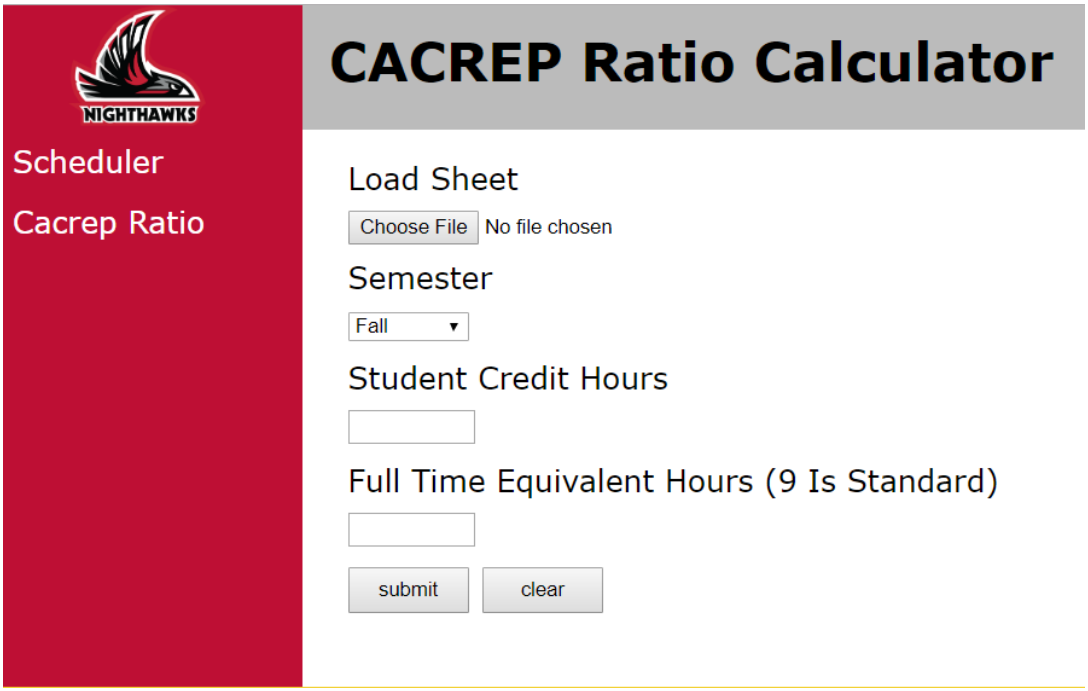

Figure 5 - CACREP Ratio Calculator Front End

Hosting the Project

Although the project was developed locally, once it was completed it was important to host the project, so it was available on the world wide web. The domain nnu-counseling-calculator.tk was chosen because .tk domains are available for free from FreeNom.com. Having a domain ensured that users would not have to access the website via IP address which is much more cumbersome. In the interest of providing a long-term server without having to make monthly or yearly payments, a home server was chosen to host the website instead of a cloud server. This home server was configured as a Linux, Apache, MySQL, and PHP or LAMP stack and was assigned a static IP address on its local network. Anaconda, the right Python version, and the appropriate packages were also installed. The router was then configured to forward all activity on 184.155.121.122:80 (web requests to the router's public IP address) to the web server. Finally, the domain nnu-counseling-calculator.tk was made to forward all traffic to 184.155.121.122:80. This created a fully functioning website that was accessible anywhere in the world and that had no monthly or yearly fees attached.

Future Work

While the class scheduling app and CACREP ratio calculator are certainly functional, there are still some improvements that could be made. One area where future work will be necessary is in protecting the website from user errors and providing useful error messages when they do occur. The program needs the loadsheet to be consistently formatted and is unable to handle files that deviate from the norm. While the program contains built in checks for file type and other basic inconsistencies, it currently lacks advanced detection that many users have grown to expect. In the future, modifying the program to automatically correct errors in the formatting of the loadsheet or to inform the user of changes they need to make will be a high priority task.

Another possible area for future work is the generalization of a class scheduling application. This program works as an impressive proof of concept showing how an automated course scheduler can work for one specific department. In order to market this product to a more generalized population, work would need to be done to move away from the use of institution specific data such as the loadsheet and to develop a more generalized method of data entry. There is a significant demand for automated scheduling in college courses, and the creation of a generalized, easy to use scheduler could be extremely marketable.

Conclusion

Overall, this project was challenging, a great learning experience, and a joy to work on. While the coding was difficult in itself, the biggest challenge came from a systems analysis and design perspective. The project started as little more than an idea, so vast amounts of requirements discovery, feasibility considerations, expectation setting, and discussions with nontechnical users had to take place before the design phase could even be started. This was an extremely valuable portion of the project as it gave experience that is not replicable in the classroom. In addition to the learning opportunity that the project provided, the scheduling and accreditation tool is a valuable resource for NNU's counselor education department as it provides increased productivity through automation.

References

"Anaconda Python/R Distribution." *Anaconda*, www.anaconda.com/distribution/.

"Online Diagram Software & Visual Solution." *Lucidchart*, 13 Nov. 2018, www.lucidchart.com/.

"Python Data Analysis Library." *Pandas*, pandas.pydata.org/.

"Python-Docx." *Python-Docx*, python-docx.readthedocs.io/en/latest/

Urofsky, Robert I. "A Reasoned Approach to FTE Faculty." *CACREP*, www.cacrep.org/articles/a-reasoned-approach-to-fte-faculty/..

Appendix A

index.html

```
<!doctype html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
        <title>Class Scheduler</title>
    </head>

    <body>

            <h3 class = "topInfo">
                Class Scheduling App
            </h3>

        <div class="navHolder">
                <img class = "sidebarImg" src="images/Nighthawks1.png">
                <a class ="navCell" href="/index.html">Scheduler</a>
                <a class="navCell" href="/calculator.html">Cacrep Ratio</a>

        </div>

        <div class = "main">
            <form action="submit.php" method = "POST" enctype="multipart/form-
data">
                <div class = "section">
                    <div class="sectionTitle">
                        Load Sheet
                    </div>
                <input type = "file" name = "loadsheet">
                </div>
                <div class = "section">
                    <div class="sectionTitle">
                        Plans of study
                    </div>
                <div>
                <input type = "file" name = "pos1">
                </div>
                <div>
                <input type = "file" name = "pos2">
                </div>
                <div>
```

```
                <input type = "file" name = "pos3">
                </div>
                </div>
                <div class = "section">
                    <div class="sectionTitle">
                        Start Times
                    </div>
                <input type = "file" name = "starttimes">
                </div>
                <div class = "section">
                <input type = "submit" value="submit">
                <input type = "reset" value = "clear">
                </div>
            </form>
        </div>
    </body>
</html>
```

calculator.html

```
<!DOCTYPE html>
<html>
    <head>
            <link rel="stylesheet" href="style.css">
            <title>Cacrep Calculator</title>
    </head>
    <body>
        <h3 class = "topInfo">
            CACREP Ratio Calculator
        </h3>

        <div class="navHolder">
            <img class = "sidebarImg" src="images/Nighthawks1.png">
            <a class ="navCell" href="/index.html">Scheduler</a>
            <a class="navCell" href="/calculator.html">Cacrep Ratio</a>
        </div>
        <div class = "main">
            <form action="calculate.php" method="POST" enctype="multipart/form-
data">
                <div class="section">
                    <div class="sectionTitle">
                        Load Sheet
                    </div>
```

```
                    <input type="file" name="loadSheet">
                </div>
                <div class="section">
                    <div class="sectionTitle">
                        Semester
                    </div>
                    <select name="semester">
                        <option value="0">Fall</option>
                        <option value="1">Spring</option>
                        <option value="2">Summer</option>
                    </select>
                </div>
                <div class="section">
                    <div class="sectionTitle">
                        Student Credit Hours
                    </div>
                    <input type="text" name="hours">
                </div>
                <div class="section">
                    <div class="sectionTitle">
                        Full Time Equivalent Hours (9 Is Standard)
                    </div>
                    <input type="text" name="fte">
                </div>
                <div class = "section">
                    <input type = "submit" value="submit">
                    <input type = "reset" value = "clear">
                </div>
            </form>
        </div>
    </body>
</html>
```

style.css

```
.navHolder {
    height: 100%;
    width: 200px;
    position: fixed;
    z-index: 1;
    top: 0;
    left: 0;
    background-color: #be0f34;
    overflow-x: hidden;
```

```
  }

  /* Side navigation links */
.navCell {
    color: white;
    padding: 8px;
    font-size:20px;
    text-decoration: none;
    display: block;
  }

  /* Change color on hover */
  .navCell:hover {
    background-color: #bbb;
    color: black;
  }
  .sidebarImg{
  max-width: 180px;
  max-height: 80px;
  display: inline-block;
  margin-left: 45px;
  }

  .topInfo{
      padding-top: 15px;
      width:100%;
      height: 67px;
      background-color:#bbb;
      text-align: center;
      font-size: 35px;
      margin-top: 0px;
      margin-bottom: 0px;
  }

  body {
    margin: 0;
    font-family: Verdana, Geneva, Tahoma, sans-serif;
  }

  .main {
    padding-top: 10px;
    margin-left: 200px;
    padding-left: 20px;
    font-size: 20px;
  }
```

```
  .section{
    margin: 10px;
  }

  .section input[type="submit"] {
      width:80px;
      height:30px;
      margin: 2px 2px 2px 0px;
  }

  .section input[type="reset"]{
    width:80px;
    height:30px;
    margin: 2px 2px 2px 0px;
  }

  .section input[type="text"]{
    width: 80px;
    margin: 2px 2px 2px 0px;
  }

  .section select{
    width: 80px;
    margin: 2px 2px 2px 0px;

  }
  .sectionTitle{
    margin-bottom: 5px;

  }
```

submit.php

```
<?php
//Function to delete all files in directory
//Takes string of file path as argument
function deleteFiles($path)
{
    $files = glob($path . "*");
    foreach($files as $file){
        if(is_file($file))
        {
```

```
            unlink($file);
        }
    }
}
//Function to run class scheduling app based on files sent from html
function runScheduler()
{
//temporary storage location for uploaded files
$uploaddir = 'C:/xampp/htdocs/temp/';
$pycallstring = "";
//Error checking for files to ensure correct type and size uploaded
foreach($_FILES as $myfile)
{
    if($myfile['type'] == "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"
    || $myfile['type'] == "application/vnd.openxmlformats-
officedocument.wordprocessingml.document"
    || $myfile['type'] == "text/csv"
    || $myfile['type'] == ""
    || $myfile['type'] == "application/vnd.ms-excel"
    && $myfile['size'] <= 1000000)
    {
        $uploadfile = $uploaddir . basename($myfile['name']);
        move_uploaded_file($myfile['tmp_name'], $uploadfile);
        $pycallstring .= str_replace('\\', '/', $uploadfile) . " ";
    }
    else
    {
        echo($myfile['type']);
        echo(" not allowed <br>");
        return;
    }
}
//execute python code and return result as well as any error messages produced
exec("D:/Users/Zacht/Anaconda3/envs/classSchedulingApp/python.exe
classSchedulingApp/scheduler.py $pycallstring 2>&1", $output, $returnval);
//if successful
if($returnval == 0)
{
$outfile = $uploaddir . "filledOutSchedule.xlsx";
if(file_exists($outfile))
{
    header('Content-type: application/vnd.ms-excel');
    header('Content-Disposition: attachment; filename="'.basename($outfile).'"');
    header('Pragma: no-cache');
```

```
    header("Expires: 0");
    readfile($outfile);
}
}
else{
    echo "ERROR", "<br>";
    foreach($output as $line)
    {
        echo($line);
        echo("<br>");
    }
}
}
//call functions
runScheduler();
deleteFiles('C:/xampp/htdocs/temp/');

?>
```

calculate.php

```
<!DOCTYPE html>
<?php
//function to delete files in directory
function deleteFiles($path)
{
    $files = glob($path . "*");
    foreach($files as $file){
        if(is_file($file))
        {
            unlink($file);
        }
    }
}
//globals to pass data to html portion
$output = "";
$returnval = "";
//Function to calculate CACREP Ratio
function runCalculator()
{
//temporary location for uploaded files
$uploaddir = 'C:/xampp/htdocs/temp/';
$pycallstring = "";
```

```
//error checking for uploaded files
foreach($_FILES as $myfile)
{
    if($myfile['type'] == "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"
    || $myfile['type'] == "application/vnd.openxmlformats-
officedocument.wordprocessingml.document"
    || $myfile['type'] == "text/csv"
    || $myfile['type'] == ""
    || $myfile['type'] == "application/vnd.ms-excel"
    && $myfile['size'] <= 1000000)
    {
        $uploadfile = $uploaddir . basename($myfile['name']);
        move_uploaded_file($myfile['tmp_name'], $uploadfile);
        $pycallstring .= str_replace('\\', '/', $uploadfile) . " ";
    }
    else
    {
        echo($myfile['type']);
        echo(" not allowed <br>");
        return;
    }
}
foreach($_POST as $postData)
{
    if(is_numeric($postData))
    {
        $pycallstring .= $postData . " ";
    }
    else{
        echo("Data must be integer <br>");
        return;
    }
}
//execute python script with arguments uploaded from html form
    exec("D:/Users/Zacht/Anaconda3/envs/classSchedulingApp/python.exe
classSchedulingApp/calculator.py $pycallstring 2>&1", $GLOBALS["output"],
$GLOBALS["returnval"]);
}
//run functions
runCalculator();
deleteFiles('C:/xampp/htdocs/temp/');
//html to display result of running calculator
?>
<html>
```

```
<head>
        <link rel="stylesheet" href="style.css">
        <title>Cacrep Calculator</title>
</head>
<body>
    <h3 class = "topInfo">
        Cacrep Ratio Calculator
    </h3>

    <div class="navHolder">
        <img class = "sidebarImg" src="images/Nighthawks1.png">
        <a class ="navCell" href="/index.html">Scheduler</a>
        <a class="navCell" href="/calculator.html">Cacrep Ratio</a>
    </div>
    <div class = "main">
   <?php if($returnval == 0)
{
    echo("Ratio = ");
    foreach($output as $line)
    {
        echo round($line, 3);
        echo "<br>";
    }
}
else
{
    echo("ERROR!");
    echo("<br>");
    foreach($output as $line)
    {
        echo $line;
        echo "<br>";
    }
}?>
    <div>
        <button onclick="goBack()">Back</button>
        <script>
        function goBack() {
        window.history.back();
        }
        </script>
    </div>
    </div>
</body>
</html>
```

calculator.py

```
from scheduler import Scheduler
import pandas as pd
from docx.api import Document
import dateTime as dt
from openpyxl import *

#Calculator class is subclass of Scheduler classs
class CacrepCalculator(Scheduler):
    # Uses Scheduler class function to read excel sheet and parses to determine a
list of full time professors
    def getFullTimeFaculty(self, loadsheet):
        fullTimeProfessors = []
        self.df = pd.read_excel(loadsheet, 5)
        for x in range(0, len(self.df["Full time faculty"])):
            fullTimeProfessors.append(self.df["Full time faculty"][x])
        return  fullTimeProfessors

    # Calculates CACREP ratio for one semester
    def calculateCacrepRatio(self, loadsheet, semester, studentCreditHours, FTE =
9):
        studentFTE = studentCreditHours/FTE
        adjunctFTE = 0
        # uses scheduler class functions to obtain data from loadsheet
        self.getLoadSheetData(loadsheet, False, semester)
        fullTimeFaculty = self.getFullTimeFaculty(loadsheet)
        # Calculate full time faculty + overload hours
        for professor in self.professors:
            if professor in fullTimeFaculty:
                loadHours = self.professors[professor].getLoadHours()
                if loadHours > FTE:
                    adjunctFTE += (loadHours - FTE)/FTE
            else:
            # Calculate adjunct faculty hours
                loadHours = self.professors[professor].getLoadHours()
                adjunctFTE += (loadHours)/FTE
        fullTimeFTE = len(fullTimeFaculty)
        facultyFTE = fullTimeFTE + adjunctFTE
        ratio = studentFTE / facultyFTE
        return ratio
```

```
if __name__ == '__main__':
    import sys
    myCalculator = CacrepCalculator()
    ratio = myCalculator.calculateCacrepRatio(sys.argv[1], int(sys.argv[2]),
int(sys.argv[3]), int(sys.argv[4]))
    print(ratio)
```

course.py

```
import dateTime as dt


class Course:

    def __init__(self, courseNumber, section, subSession, dates, modality,
loadHours, workCode, day, row, time=""):
        self.courseNumber = courseNumber
        self.section = section
        self.subSession = subSession
        self.dates = dt.splitDate(dates)
        self.modality = modality
        self.loadHours = loadHours
        self.workCode = workCode
        self.time = dt.splitTime(time)
        self.row = row
        try:
            self.day = list(day)
        except:
            self.day = ""
```

dateTime.py

```
from datetime import datetime
from datetime import timedelta

# converts a string of format 8:20 PM - 10:00 PM to list of date time objects
def splitTime(time):
    try:
        return [datetime.strptime(item, "%I:%M %p") for item in time.split(" -
")]
    except:
        return ""
```

```
# converts a string of format 12/15/18 - 12/30/18 to list of date time objects
def splitDate(date):
    try:
        return [datetime.strptime(item, "%m/%d/%y") for item in date.split(" -
")]
    except:
        return ""

# compares two ranges of dates or times with the td1 and td2 being the 1st range.
td3 and td4 represent the 2nd range
def compareTimesOrDates(td1, td2, td3, td4):
    if td1 == td4 or td2 == td3:
        return False
    elif td1 >= td3 and td1 <= td4:
        return True
    elif td2 >= td3 and td2 <= td4:
        return True
    elif  td3 >= td1 and td3 <= td2:
        return True
    else:
        return False
```

professor.py

```
class Professor:

    def __init__(self, ID, fullName):
        self.ID = ID
        self.fullName = fullName
        self.courses = []
# returns professor course load
    def getLoadHours(self):
        load = 0
        for course in self.courses:
            if course.workCode == "Teaching" or course.workCode == "teaching":
                load += float(course.loadHours)
        return load
```

scheduler.py

```
import pandas as pd
from docx.api import Document
import dateTime as dt
from openpyxl import *

# Class to hold information on each course on the load sheet
from course import Course

#Class to hold information on each professor on the load sheet
from professor import Professor


# Class to do most of work. Uses the Courses and Professor classes
class Scheduler:

    def __init__(self):
        # dictionary of professors indexed by professor ID
        self.professors = {}

        # will hold courses that are not currently assigned to a professor
        # self.unassignedCourses = []

        # pandas data frame to make input of .xlxs file easier
        self.df = ""

        # Holds unique ID's to ensure professors aren't stored more than once
        self.uniqueFacultyIDs = []

        # List of dictionarys using the sequence as a key and a list of course
numbers as the value
        self.possibleConflicts = []

        # Dict of possible start times for courses
        self.startTimes = {}
# input function to get information from loadsheet. Loads professors and then
assigns courses to the professors

    def getLoadSheetData(self, loadSheet, filledOut, semester = 0):
        # Using Pandas read excel function to get input into dataframe
        self.df = pd.read_excel(loadSheet, semester)

# looping through data frame to move data into professor and courses classes
        for x in range(0, len(self.df["Faculty ID"])):
```

```
            if self.df["Faculty ID"][x] not in self.uniqueFacultyIDs:
                self.uniqueFacultyIDs.append(self.df["Faculty ID"][x])
                self.professors[self.df["Faculty ID"][x]] =
Professor(self.df["Faculty ID"][x], self.df["Name"][x])
            if not filledOut:
                self.professors[self.df["Faculty
ID"][x]].courses.append(Course(self.df["Course Number"][x],

self.df["Sec"][x],

self.df["Sub Session"][x],

self.df["Dates"][x],

self.df["Modality"][x],

self.df["Load Hrs"][x],

self.df["Work Code"][x],

self.df["Days"][x],
                                                                x
+ 2))

            elif filledOut:
                self.professors[self.df["Faculty
ID"][x]].courses.append(Course(self.df["Course Number"][x],

self.df["Sec"][x],

self.df["Sub Session"][x],

self.df["Dates"][x],

self.df["Modality"][x],

self.df["Load Hrs"][x],

self.df["Work Code"][x],

self.df["Days"][x]),
                                                                x
+ 2,

self.df["Time"][x])
```

```
# Input function for study plan. This is a word doc containing a table with
information about class sequences
# Uses python-docx to get input and will use this information to determine
possible course scheduling conflicts
    def getPlanOfStudy(self, planOfStudy, planOfStudy1, planofStudy2):
        # python-docx container for document read
        document = Document(planOfStudy)
        table = document.tables[0]
        courseNum = ""
        # sequence holds the semester in which a course should be taken
        sequence = ""

        # looping through table by row
        for i, row in enumerate(table.rows):
            for j, cell in enumerate(row.cells):
                # necessary information in 1st and 4th row
                if i != 0 and j == 0:
                    courseNum = cell.text
                if i != 0 and j == 3:
                    sequence = cell.text
            # First row contains headers so shouldn't look at its information
            if i == 0:
                found = True
                found1 = True
                found2 = True
            else:
                found = False
                found1 = False
                found2 = False
            # search through list of dicts for last read in sequence
            if "or" in sequence:
                sequence = sequence.split("or")
                sequence[0] = sequence[0].rstrip("")
                sequence[1] = sequence[1].lstrip("")
            for section in self.possibleConflicts:
                # it it is there add last read in course number to it
                if not isinstance(sequence, list):
                    if sequence in section:
                        section[sequence].append(courseNum)
                        found = True
                if isinstance(sequence, list):
                    if sequence[0] in section:
                        section[sequence[0]].append(courseNum)
                        found1 = True
```

```
                    if sequence[1] in section:
                        section[sequence[1]].append(courseNum)
                        found2 = True
                # else create new dictionary with sequence and courseNum and
append to possibleConflicts
            if found == False and not isinstance(sequence, list):
                section = {sequence: []}
                section[sequence].append(courseNum)
                self.possibleConflicts.append(section)
            if found1 == False and isinstance(sequence, list):
                section = {sequence[0]: []}
                section[sequence[0]].append(courseNum)
                self.possibleConflicts.append(section)
            if found2 == False and isinstance(sequence, list):
                section = {sequence[1]: []}
                section[sequence[1]].append(courseNum)
                self.possibleConflicts.append(section)
# Function to check for a conflict between two courses
# Using Date Time functions to compare date and time
    def checkForConflict(self, course1, course2):
        try:
            if course1.day == course2.day:
                if dt.compareTimesOrDates(course1.dates[0], course1.dates[1],
course2.dates[0], course2.dates[1]):
                    # All friday saturday courses are taught at same time so they
always conflict
                    if course1.day[0] == "F" and course1.day[1] == "S":
                        return True
                    if dt.compareTimesOrDates(course1.time[0], course1.time[1],
course2.time[0], course2.time[1]):
                        return True
            return False
        except:
            return False
# Function to call check for conflict to compare 1 course to every other course
    def findAllConflicts(self, compareCourse, parentProf):
        for profid, prof in self.professors.items():
            for crs in prof.courses:
                # Conflicts determined by professor and plan of study
                if crs in parentProf.courses or
self.searchConflictList(compareCourse, crs):
                    if self.checkForConflict(compareCourse, crs) and crs !=
compareCourse:
                        return True
        return False
```

```
# Searches the conflict list to determine if a conflict matters
    def searchConflictList(self, course1, course2):
        for sequence in self.possibleConflicts:
            for key in sequence:
                # Slice to get only course num not the name
                if course1.courseNumber[:8] in sequence[key]:
                    if course2.courseNumber[:8] in sequence[key]:
                        return True
        return False

# load csv file with the earliest and latest start times for Monday - Saturday
courses
    def loadStartTimes(self, filename):
        import csv
        with open(filename) as csv_file:
            csv_reader = csv.reader(csv_file)
            for row in csv_reader:
                self.startTimes[row[0]] = dt.splitTime(row[1])

# Schedules courses based on potential start times
# Days are filled from earliest to latest start times
# Course moves back if it conflicts with another course
# Course moved back in 30 minute increments
# Assuming 1 hour for each load hour unless a F-S class
    def scheduleCourses(self, filename, outfile):
        wb = load_workbook(filename)
        ws = wb.active
        # loop through all courses
        for prof in myScheduler.professors:
            for crs in myScheduler.professors[prof].courses:
                # replacing time column
                rowCol = "G" + str(crs.row)
                # only scheduling face to face teaching classes
                if crs.workCode == "Teaching" and crs.modality == "F2F" and
crs.day[0] != "F":
                    crsStart = self.startTimes[crs.day[0]][0]
                    crsEnd = crsStart + dt.timedelta(0,0,0,0,0,crs.loadHours)
                    crs.time = [crsStart, crsEnd]
                    # keep moving start time until there is no conflict
                    while self.findAllConflicts(crs,
myScheduler.professors[prof]) and crsStart <= self.startTimes[crs.day[0]][1]:
                        crsStart += dt.timedelta(0,0,0,0,30)
                        crsEnd += dt.timedelta(0,0,0,0,30)
                        crs.time = [crsStart, crsEnd]
```

```
                    # write to file, using openpyxl
                    ws[rowCol] = dt.datetime.strftime(crsStart, "%I:%M %p") + " -
" + dt.datetime.strftime(crsEnd, "%I:%M %p")
                    # FS courses all taught at same times so hardcoded
                elif crs.workCode == "Teaching" and crs.modality == "F2F" and
crs.day[0] == "F":
                    ws[rowCol] = "(F) 05:00 PM - 10:00 PM (S) 08:00 AM - 05:00
PM"
        wb.save(outfile)
```

```
if __name__ == '__main__':
    import sys
    myScheduler = Scheduler()
    outputpath = "/".join(sys.argv[1].split("/")[0: -1])
    myScheduler.getLoadSheetData(sys.argv[1], False, 0)
    myScheduler.getPlanOfStudy(sys.argv[2], sys.argv[3], sys.argv[4])
    myScheduler.loadStartTimes(sys.argv[5])
    myScheduler.scheduleCourses(sys.argv[1], outputpath +
"/filledOutSchedule.xlsx")
    pass
```

Appendix B

loadsheet.xlsx

Faculty ID	Name	Course Number	Sec	Sub S	Period	Time	Days	Res	Dates	Location	Modality	Load Hrs	Contract T	Work Code
184570	Garner, Curtis	COUN7537RESEARCH & STATISTICS	1	SM		4:30 PM - 7:20 PM	R	28	8/27/18 - 12/16/18	NNU	F2F	3	C	Teaching
184570	Garner, Curtis	COUN6512ETHICAL & LEGAL ISSUES	1	SM		4:30 PM - 7:20 PM	T	28	8/27/18 - 12/16/18	NNU	F2F	3	C	Teaching
184570	Garner, Curtis	CHAIR ADMINISTRATION										3	C	Administration
184570	Garner, Curtis	INTERNSHIP SITE VISITS										1		
184570	Garner, Curtis	COUN6000COUNSELOR EDUCATION ORIENTATION	1L	SM		Online			8/27/18 - 12/16/18	Online	Online	0	C	Teaching
47643	Pitts, Michael	SCHOLARSHIP										2.33	C	Scholarship
47643	Pitts, Michael	COUN7543MEANING-CENTERED INTERVENTIONS	1	B5		(F) 5:00 PM-10:00 PM	FS	24	10/1/18 - 11/4/18	NNU	F2F	2	C	Teaching
47643	Pitts, Michael	COUN6542GROUP COUNSELING	1	SM		4:30 PM - 7:20 PM	M	24	8/27/18 - 12/16/18	NNU	F2F	3	C	Teaching
47643	Pitts, Michael	COUN7589INTERNSHIP IN COUNSELING	2	SM		4:30 PM - 6:00 PM	T	10	8/27/18 - 12/16/18	Nampa	F2F	2	C	Teaching
47643	Pitts, Michael	CACREP LIASON										2	O	Administration
47643	Pitts, Michael	FACULTY SITE VISITS										1	O	Supervision
394141	Henderson, Jessica	FAMILIES, ETC ADMIN										1.5	C	Administration
394141	Henderson, Jessica	SCHOLARSHIP										2.66	C	Scholarship
394141	Henderson, Jessica	COUN6515MLTCLTRL CNSLING/SOCTL ISS	1	SM		4:30 PM - 7:20 PM	R	22	8/27/18 - 12/16/18	NNU	F2F	3	C	Teaching
394141	Henderson, Jessica	COUN6534CAREER DEVELOPMENT	6L	B8		Online	Online	14	10/22/18 - 12/16/18	Online	Online	3	C	Teaching
394141	Henderson, Jessica	COUN7553MEASUREMENT & ASSESSMENT	6	A8		(F) 5:00 PM-10:00 PM	FS	14	8/27/18 - 10/21/18	Twin Falls	F2F	2	O	Teaching
623	Dick Craig	FACULTY SITE VISITS									F2F	3.5	A	Supervision
26281	Chris McNaught	FACULTY SITE VISITS									F2F	3	A	Supervision
514172	Dennis Baughman	COUN6533FAMILY COUNSELING THEORIES	1	A7		(F) 5:00 PM-10:00 PM	FS		8/27/18 - 10/14/18		F2F	2	A	Teaching
110440	Heather Tustison	COUN6594B ART THERAPY IN PROFESSIONAL PRACTICE	3	C5		(F) 5:00 PM-10:00 PM	FS		11/5/18 - 12/16/18	Nampa	F2F	1	A	Teaching
8706	Linda Arrossa	COUN7589INTERNSHIP IN COUNSELING	6	SM		4:30 PM - 6:30 PM	T	10	8/27/18 - 12/16/18	Twin Falls	F2F	2	A	Teaching
8706	Linda Arrossa	FACULTY SITE VISITS								Twin Falls	F2F	2	A	Supervision
94703	Joy Kaplin	COUN7589INTERNSHIP IN COUNSELING	1	SM		6:30 PM - 8:00 PM	W	10	8/27/18 - 12/16/18	Nampa	F2F	2	A	Teaching
16081	Jeffery Edmiston	COUN6594C ACCEPTANCE AND COMMITMENT THERAPY	1	SM		4:30 PM - 6:20 PM	W	20	8/27/18 - 12/16/18	Nampa	F2F	2	A	Teaching
214662	Amy Curry	COUN7507INTRO TO CHILD-CENTERED PLAY THERAPY	1	C5		(F) 5:00 PM-10:00 PM	FS		11/5/18 - 12/16/18	Nampa	F2F	1	A	Teaching
0	TBD	COUN7508PLAY THERAPY & PRIVATE PRACTICE	1	B6		(F) 5:00 PM-10:00 PM	FS		10/15/18 - 12/2/18	Nampa	F2F	1	A	Teaching
15936	Steve Filer	COUN7546CASE MANAGEMENT IN ADDICTIONS	1	B7		(F) 5:00 PM-10:00 PM	FS		10/22/18 - 12/16/18	Nampa	F2F	2	A	Teaching

Clinical.docx

Number	Title	Credits	Recommended Sequence	When Offered	Pre-Requisites
COUN6511	The Profession of Counseling	2	1st Semester, Year 1	Every Fall & Spring	
COUN6512	Ethical and Legal Issues	3	1st Semester, Year 1	Every Fall & Spring	
COUN6515	Multicultural Counseling and Societal Issues	3	1st Semester, Year 1	Every Fall & Spring	
COUN6531	Learning Process & Lifespan Development	2	2nd Semester, Year 1	Every Fall & Spring	
COUN6532	Theories of Counseling	3	2nd Semester, Year 1	Every Spring & Fall	
COUN7553	Measurement & Assessment	2	2nd Semester, Year 1	Every Spring	
COUN6542	Group Counseling	3	3rd Semester, Year 1	Every Summer & Fall	
COUN6551	Diagnosis & Treatment of Psychopathology	3	3rd Semester, Year 1	Every Summer	
COUN6552	Addictions Counseling	2	3rd Semester, Year 1	Every Summer	
COUN6533	Family Counseling Theories	2	4th Semester, Year 2	Every Fall	COUN6532
COUN6541	Pre-Practicum: Basic Counseling Skills	3	4th Semester, Year 2	Every Fall & Spring	COUN6512 & COUN6532 may be concurrent
COUN7537	Research and Statistics	3	4th Semester, Year 2	Every Fall	
COUN7561	Practicum	3	4th Semester, Year 2	Every Fall & Spring	COUN6511,COUN6512, COUN6541, COUN6542 & COUN6551 & approved group
COUN6534	Career Development	3	5th Semester, Year 2	Every Spring & Summer	
COUN7582	Intro to Trauma and Crisis	1	5th Semester, Year 2	Every Fall & Spring - online	
COUN7589	Internship in Counseling	1-2	6th Semester, Year 2	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling
COUN7572	Psychopharmacology	1	6th Semester, Year 2	Every Summer	
COUN7589	Internship in Counseling	4-5	7th Semester, Year 3	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling
COUN7523	Organization and Admin. of Clinical Services	2	Year 2 or 3	Spring Odd Years	
COUN7563	Clinical Intervention & Prevention	3	Year 2 or 3	Summer Odd Years	
COUN7589	Internship in Counseling	4-5	8th Semester, Year 3	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling

MCFC.docx

Number	Title	Credits	Recommended Sequence	When Offered	Pre-Requisites
COUN6511	The Profession of Counseling	2	1st Semester, Year 1	Every Fall & Spring	
COUN6512	Ethical and Legal Issues	3	1st Semester, Year 1	Every Fall & Spring	
COUN6515	Multicultural Counseling and Societal Issues	3	1st Semester, Year 1	Every Fall & Spring	
COUN6531	Learning Process & Lifespan Development	2	2nd Semester, Year 1	Every Fall & Spring	
COUN6532	Theories of Counseling	3	2nd Semester, Year 1	Every Spring & Fall	
COUN7553	Measurement & Assessment	2	2nd Semester, Year 1	Every Spring	
COUN6542	Group Counseling	3	3rd Semester, Year 1	Every Summer & Fall	
COUN6551	Diagnosis & Treatment of Psychopathology (DSM)	3	3rd Semester, Year 1	Every Summer	
COUN6552	Addictions Counseling	2	3rd Semester, Year 1	Every Summer	
COUN6533	Family Counseling Theories	2	4th Semester, Year 2	Every Fall	COUN6532
COUN6541	Pre-Practicum: Basic Counseling Skills	3	4th Semester, Year 2	Every Fall & Spring	COUN6512 & COUN6532 may be concurrent
COUN7537	Research and Statistics	3	4th Semester, Year 2	Every Fall	
COUN7561	Practicum	3	4th Semester, Year 2	Every Fall & Spring	COUN6511,COUN6512, COUN6541, COUN6542 & COUN6551 & approved group
COUN7531	Meta-Theoretical Practices in MCFC	2	4th Semester, Year 2	Every Fall	COUN6533
COUN6534	Career Development	3	5th Semester, Year 2	Every Spring & Summer	
COUN7582	Intro to Trauma and Crisis	1	5th Semester, Year 2	Every Fall & Spring - online	
COUN7589	Internship in Counseling	1-2	6th Semester, Year 2	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling
COUN7572	Psychopharmacology	1	6th Semester, Year 2	Every Summer	
COUN7533	Advanced Models & Interventions in MCFC	2	Spring, Year 2 or 3	Spring Odd Years	COUN6533 & COUN7531
COUN7535	Contemporary Directions in MCFC	1	Spring, Year 2 or 3	*Spring Odd Years*	COUN6533 & COUN7531
COUN7529	Human Sexuality	1	Summer, Year 2 or 3	Summer Odd Years	
COUN7532	MCFC Assessment & Evaluation	2	Summer, Year 2 or 3	Summer Even Years	COUN6533, COUN7531
COUN7538	Family Education Experience	1	Summer, Year 2 or 3	Summer Odd Years	

COUN7534	Clinical Issues in MCFC	2	Fall, Year 2 or 3	Fall Even Years	COUN6533 & COUN7531 may be concurrent
COUN7589	Internship in Counseling	4-5	7th Semester, Year 3	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling
COUN7589	Internship in Counseling	4-5	8th Semester, Year 3	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling

School.docx

Number	Title	Credits	Recommended Sequence	When Offered	Pre-Requisites
COUN6511	The Profession of Counseling	2	1st Semester, Year 1	Every Fall & Spring	
COUN6512	Ethical and Legal Issues	3	1st Semester, Year 1	Every Fall & Spring	
COUN6515	Multicultural Counseling and Societal Issues	3	1st Semester, Year 1	Every Fall & Spring	
COUN6531	Learning Process & Lifespan Development	2	2nd Semester, Year 1	Every Fall & Spring	
COUN6532	Theories of Counseling	3	2nd Semester, Year 1	Every Spring & Fall	
COUN7553	Measurement & Assessment	2	2nd Semester, Year 1	Every Spring	
COUN6542	Group Counseling	3	3rd Semester, Year 1	Every Summer & Fall	
COUN6551	Diagnosis & Treatment of Psychopathology	3	3rd Semester, Year 1	Every Summer	
COUN6552	Addictions Counseling	2	3rd Semester, Year 1	Every Summer	
COUN6533	Family Counseling Theories	2	4th Semester, Year 2	Every Fall	COUN6532
COUN6541	Pre-Practicum: Basic Counseling Skills	3	4th Semester, Year 2	Every Fall & Spring	COUN6512 & COUN6532 may be concurrent
COUN7537	Research and Statistics	3	4th Semester, Year 2	Every Fall	
COUN7521	School Counseling	2	4th Semester, Year 2	Every Fall	
COUN7561	Practicum	3	4th Semester, Year 2	Every Fall & Spring	COUN6511,COUN6512, COUN6541, COUN6542 & COUN6551 & approved group
COUN6534	Career Development	3	5th Semester, Year 2	Every Spring & Summer	
COUN7582	Intro to Trauma and Crisis	1	5th Semester, Year 2	Every Fall & Spring - online	
COUN7589	Internship in Counseling	1-2	6th Semester, Year 2	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling
COUN7524	Organization of Comprehensive School Counseling Services	2	5th Semester, Year 2 or 8th Semester, Year 3	Spring Odd Years	COUN7521

COUN7545	Collaboration & Consultation	2	6th Semester, Year 2 or 9th Semester, Year 3	Summer Even Years	COUN7521
COUN7589	Internship in Counseling	4-5	7th Semester, Year 3	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling
COUN7589	Internship in Counseling	4-5	8th Semester, Year 3	Every Fall, Spring & Summer	COUN7561; 4 sessions individual counseling

startTimes.csv

M	4:30 PM - 8:00 PM
T	4:30 PM - 8:00 PM
W	4:30 PM - 8:00 PM
R	4:30 PM - 8:00 PM
F	4:30 PM - 8:00 PM
S	8:00 AM - 9:00 AM