Design Document

for

ETM Advising Database

**Version 1.3 approved**

**Prepared by Holly Fox, Jake Mulligan, Jordan Zenisek, Kevin Godenswager**

**Ohio University Senior Design**

**April 21st, 2019**

**Table of Contents**

[**Introduction**](#_3znysh7) **4**

[Purpose](#_2et92p0) 4

[Product Scope](#_1t3h5sf) 4

[**Design Overview**](#_2s8eyo1) **4**

[Current Problem](#_17dp8vu) 4

[Current Solution](#_jp457nnisfrh) 4

[Proposed Solution](#_z337ya) 4

[Technologies Used](#_3j2qqm3) 5

[**System Architecture**](#_2xcytpi) **5**

[**System Components**](#_2bn6wsx) **6**

[Network and Internet Services](#_wfcp2ir2i85j) 6

[Database](#_l044mzbgvb6t) 6

[User Interface](#_nlu92sg60ten) 6

[**Class Diagram**](#_147n2zr) **7**

[**Database**](#_e4ojvdnhj6s1) **7**

[Source of Data](#_gubv4tsttgwl) 7

[Collected Data](#_ndnzpu2gqryz) 8

[**Software Features**](#_j73j6ku2cm8) **9**

[User Interface Diagrams](#_t9t6jrpd35w2) 9

[Use Cases](#_nl75b9j0twmm) 10

[**Security and Privacy**](#_akwf5490xy9j) **13**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Holly Fox  Jordan Zenisek  Kevin Godenswager  Jacob Mulligan | 10-10-18 | Initial Creation | 1.0 |
| Holly Fox  Jordan Zenisek  Kevin Godenswager  Jacob Mulligan | 1-17-19 | Updates based on changes. | 1.1 |
| Holly Fox  Jordan Zenisek  Kevin Godenswager  Jacob Mulligan | 1-31-19 | Updates | 1.2 |
| Jordan Zenisek | 4-18-19 | Final Updates reflecting end product. | 1.3 |

# Introduction

## Purpose

The purpose of this project was to design a web application for Engineering Technology and Management (ETM) advisors. The procedure was fully manual, and the new application automates the process. This is done by providing views for a student’s Program of Study, Program Planner, and Course Loading which provides professors an estimate of the number of students who will be registering for each ETM course by semester.

## Product Scope

The scope of this project was to automate the advising process for ETM at Ohio University. The DARS are parsed with a Python script which inputs data into the database. The database was implemented using Amazon Web Services with MySQL. The web-application features a Program Planner view which will plan students future courses and a Course Loading view which will predict how many students will take a course during a semester.

The product was handed over on 4/25/19 and all final questions were answered, now it is fully up to the client to handle any upkeep.

# Design Overview

## Problem Given

ETM advising lacks an automated system to aid the advising process. The current method is to fill out paper copies of the courses students have taken and plan the rest of the courses by manually checking to see if they meet all of the requisites in order to move forward in their program of study. This system is slow and laborious.

## Current Solution

There is no current solution to the laborious ETM Advising process beyond the manual paper process.

On 4/25/19, the client was given the below proposed solution in full.

## Proposed Solution

The proposed solution to decrease workload and streamline the ETM Advising process by creating a web application to replace the paper advising forms, allowing advisors to quickly and easily access each student’s Program of Study, as well as use projected enrollment statistics to plan course offerings accordingly.

## Technologies Used

We used CSS, Javascript, Node.js, Express Framework, and Pug Framework to implement the RESTful web-application. These technologies create a user friendly experience as well as make the application visually pleasing. We used Python for parsing the University DARS reports. After extracting data from PDFs the data is inputted into the web-application through the database. This greatly reduces time compared to manual input. The database is implemented in MySQL and currently connects to AWS (the client is required to host from their own server upon receiving the product).

Node.js, is a popular web development application, making it a perfect fit for the Engineering Technology Management System. It was essential that our backend could handle multiple connections in a single instance. Node.js is based on Google Chrome’s V8 JavaScript Engine, which is very fast in executing JavaScript code. On top of that the lightweight framework is able to host and serve our web pages. An added bonus is widespread community support providing many useful resources and examples.

When approaching the need for a storage system MySQL fit the bill. Historically known for its role in supporting websites was just the first attraction. MySQL also uses relational algorithms to store data which worked wonderfully with the system’s needs. Oracle also provides a free community edition that comes with all the necessary tools. The best part about MySQL is that if Oracle ever decided to stop support the inventor and other developers are dedicated to keeping it alive. They have already created forks of the programs code base.

We choose to add Express and PUG to our Node.js because they provided all the necessary tools needed to develop the backend without having to start completely from scratch. The Express framework provides an essential tool for handling HTTP requests and provides many needed functions to serve the application overall. The PUG framework provided us with a much needed tool for injecting database data into the HTML code. The choice to use CSS was simple because of its popularity to format the HTML to look presentable.

Lastly, Python was chosen uniquely for its vast array of frameworks. The popular programing language already provided a tool to extract data from PDF files. Pythons scripting language was a match made in heaven for our needs in extracting and placing data into MySQL. Pythons powerful library and widespread support added to the desire to us the programing language to achieve our needs.

# System Architecture

The architecture consists of a few parts. The user makes a request from the web server for a particular student. Then they will be able to interact with the information and make changes accordingly. Once completed they will submit the updated information and send it back to the server. The server will collect the information and place it into the database.

# System Components

## Network and Internet Services

The user contacts the web server by sending an HTTP request to the server. The web server then authenticates the user and provides the requested information. The web server is contacted when the user requests information and when it receives new information from the user. All information will pass through the internet so that advisors have control of the information in a central location.

## Database

All data for the program is stored in the database. Data relations are further explained in the Class Diagram located in section 5. Queries from the web server fill in the required data for the views shown in the section 7.2. The user can then update data on the web application. Older versions can also archived in the database and each students archives can be referenced on the web page.

## User Interface

This application is based off of the documents advisors currently use to record student data. The documents include three different dashboards or views -- Program of Study, Program Planner, and Course Loading view. The dashboards are editable with autofill dropdown menus connected with the database making it easy to use and limiting user error. When a user is finished making changes, they may submit them and it will be archived in the database. A diagram of the user interface can be found in section 7.1.

# Class Diagram

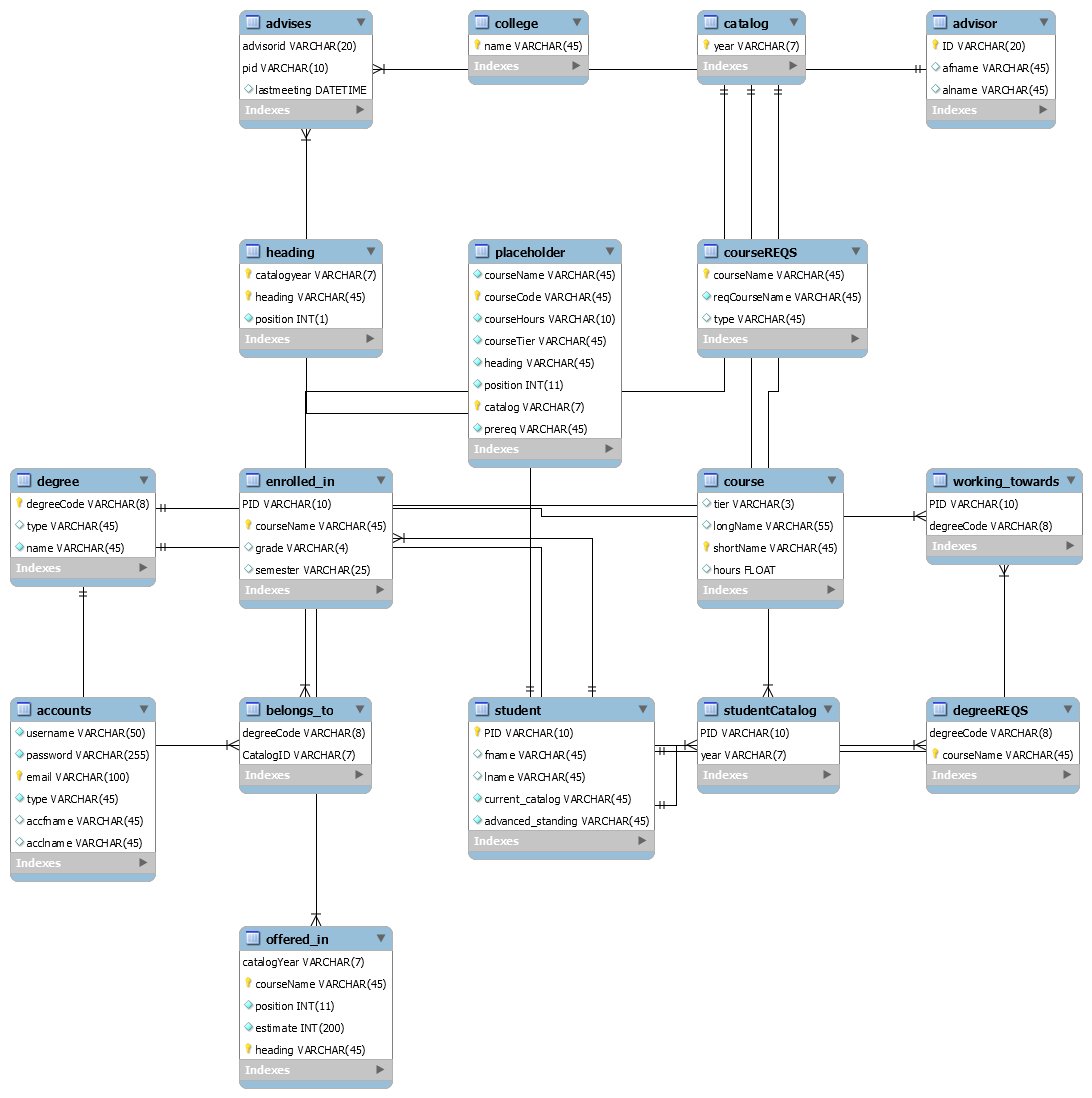


Image 1: This chart describes the database model that the application is based upon. There are five main entities: Students, Advisors, Courses, Degrees and Catalogs. Students are identified by their PID, are enrolled in Courses, assigned an Advisor, working towards a Degree, and belong to a certain Catalog. Courses are identified by their short name, and ‘offered’ in various Catalogs. Degrees are identified by their degree code, and belong to various Catalogs. Advisors are identified by their id (email handle), and will be assigned to multiple students. This diagram will be used to aide in maintenance and use of the database by giving understanding to the underlying structure.

# Database

## Source of Data

Parsed or automatically collected data used for this software is taken from student DARS Reports, and the online OHIO Course Catalogs.

## Collected Data

Collected student data includes:

* Identification
  + Name
  + PID
  + Email
* Academic Career Information
  + Degree
  + Level
  + Advisor ID
  + Catalog
  + College
  + Campus
  + Certificate/Minor information
* Academic Career History
  + Courses taken
    - Credit hours
    - Grade

Collected course related data will include:

* Catalog information
  + i.e. required courses by catalog
* Course information
  + Long name
  + Short name
  + ID
  + Tier
  + Requisites
    - Prerequisites
    - Corequisites
    - Sequential

# Software Features

## User Interface Diagrams

Image 2: The above image shows how users will interact with the software. An advisor first chooses whether they would like to view a students PP, POS, the User Manual, Course Loading View, or Add a Student. From there they continue making decisions until they find the information they would like. An admin will be able to do all the same things as the advisors, with the added ability being able to edit the layout, reset a user’s password, and also remove a user. This diagram is useful for showing the user what steps should be taken to reach a certain view.

## Use Cases

The primary users of the ETM Advising Database are advisors and administrators. There are three primary uses, as given below:

* Administration
  + Adding advisors to database as elements
  + Allowing access to other administrators, advisors, and approved faculty
  + Importing DARS reports
  + Importing/updating course catalogs and relevant information
  + Extraneous database/server/web upkeep
  + Updating arrangement of web pages
* Advising
  + Importing and updating student programs of study
    - Adding, changing grades
    - Adding, changing certificates/minors
  + Creating or proposing projected programs of study
  + Accessing archived programs of study
* Planning
  + View anticipated students to enroll in courses
    - View enrollment by semester, projecting four years into the future

These use cases will be accessible through three separate views:

* Program of Study
* Program Planner
* Course Loading

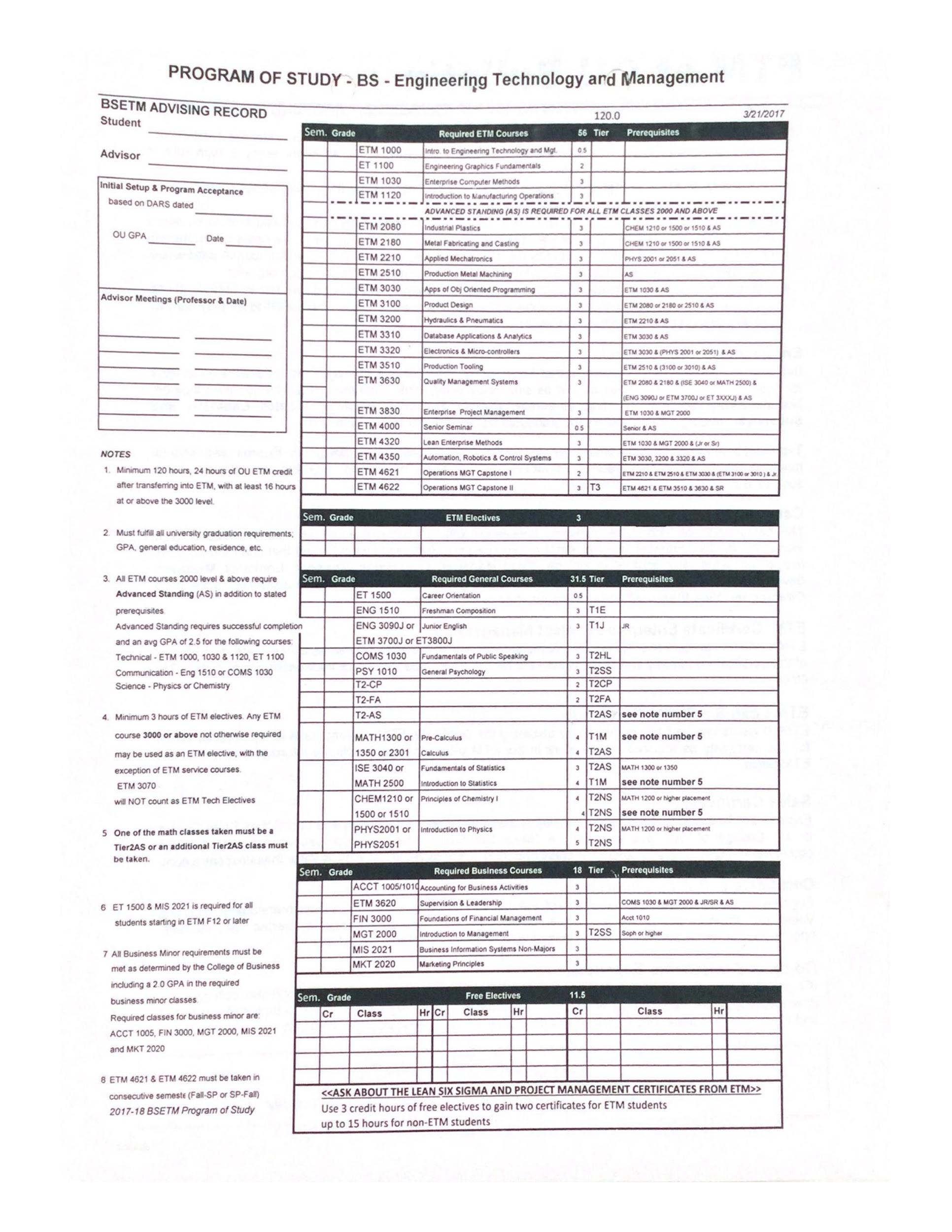


Image 3: The above image is a model for the Program of Study (POS) view. A POS can be loaded for each student and will show the required ETM courses and that students state of completion. Changes can be submitted creating an archived POS.

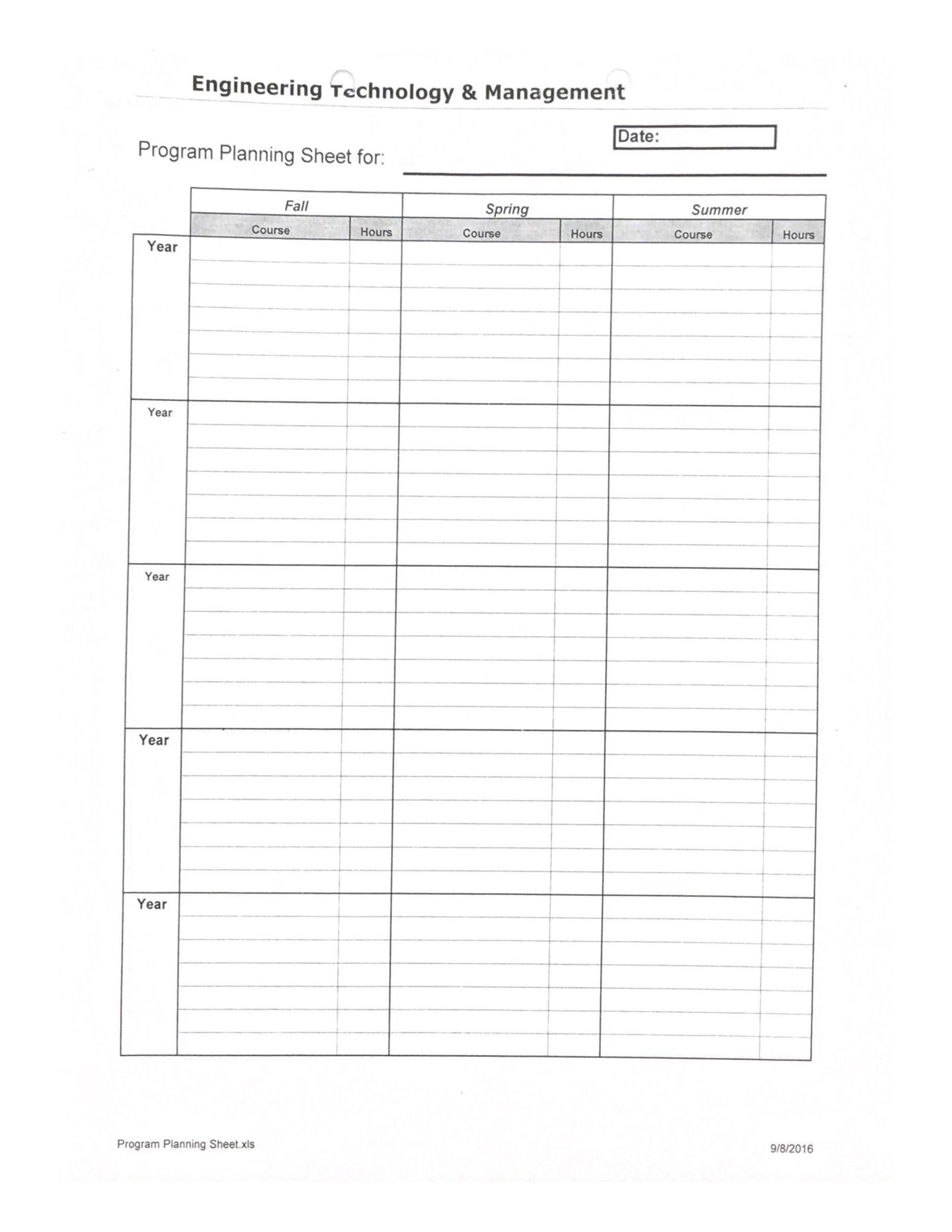


Image 4: The above image is a model we were given for the Program Planner (PP) view. A PP can be loaded for each student and shows a projection of when they will take classes separated by semester and year.

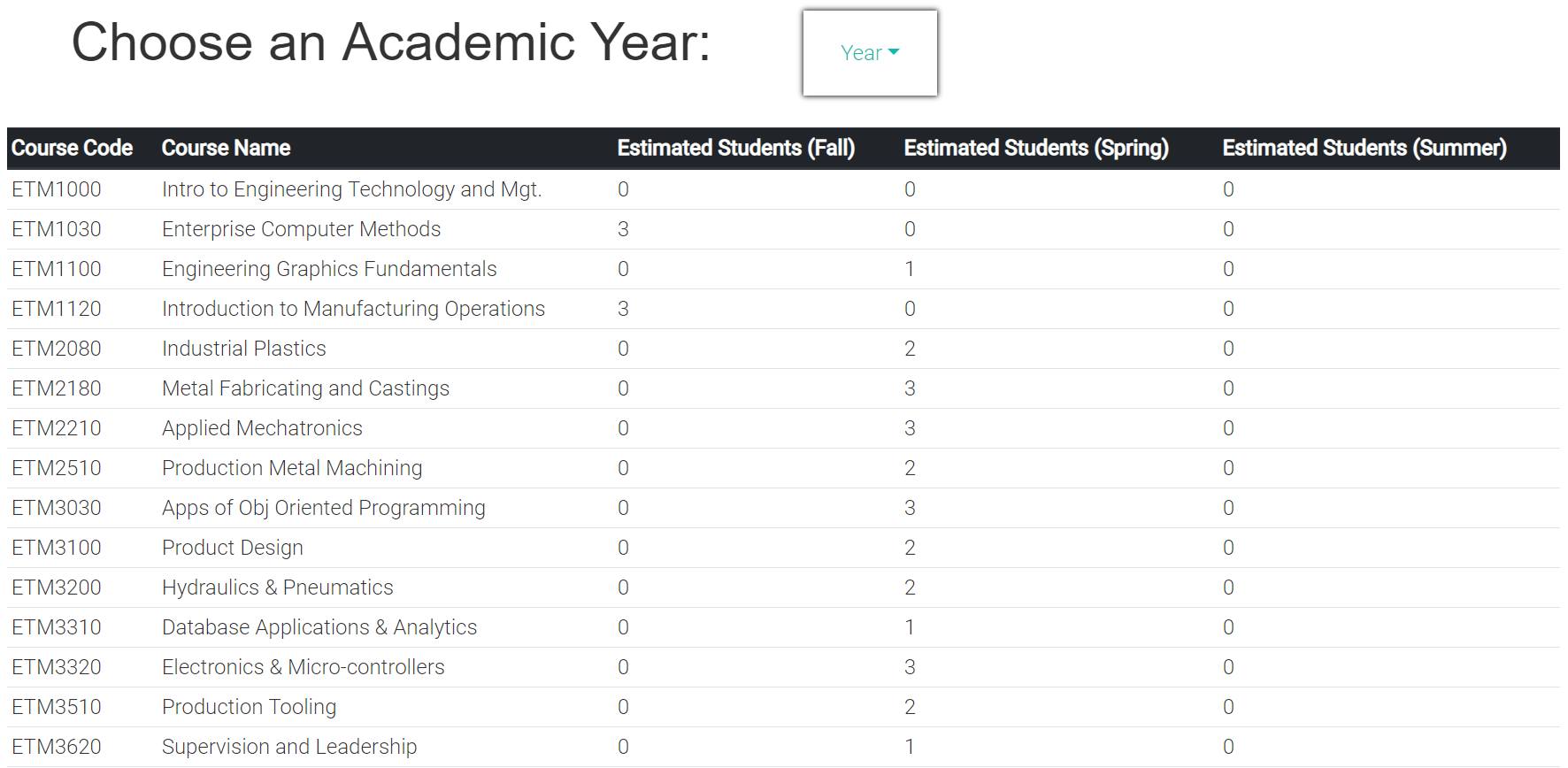


Image 6: A working model of the Course Loading (CL) view, which will per course, semester, and year show projected number of students to enroll.

# Security and Privacy

This web application and associated software are accessible to advisors and administrators only. It will be secured to the same extent as other OHIO software, as well as compliant with all FERPA privacy standards.