Last Place Champions

Progress Report

2/08/18

Contents

Introduction

Team Members

Students

- Brian Hooper – Team Lead / Front-End Developer
- Nick Rohde – Server and Database Developer
- Rico Adrian – Back-End Developer

Advisor

- Dr. Donald Davendra

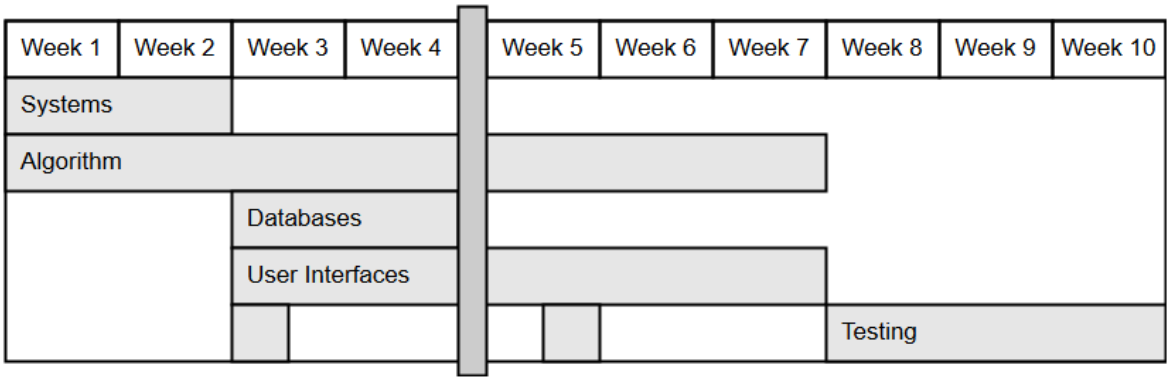
Client

- Dr. Szilárd Vajda
- Dr. Donald Davendra

Description

This product is being designed to replace the old advising tool which is currently in use by the CS department. It will be a standalone website that does not tie into any larger or separate set of software, other than the databases and web server software that will host the system. The data on which the application acts - Catalog, schedule, and student records - may be hosted either on the systems internal database or accessed from an external database.

Progress Report



Major Milestones

The majority of the code for the database handler and classes has been completed, and the virtual Linux server is ready to receive the deployed project. The front-end design, layout, user interface, and static code is complete for the first iteration, and the second iteration is currently in progress. Handling of dynamic in-browser events, including manipulation of course and schedule data should be done in a few weeks. The basic framework for the scheduling algorithm is in place.

Currently in Progress

Most of the work right now is being done to get a working implementation of the scheduling algorithm. The design of the base case for the algorithm is nearly complete with a brute-force method, and a preliminary design for a branch-and-bound optimization is ready to implement when the brute-force method is working. The code for the major pieces of the algorithm are in place, including checking prerequisites and quarterly offerings, and generating a list of possible courses for each specific quarter and iteration of the algorithm. Adding courses to an in-progress schedule is mostly implemented but currently non-functional.

To Do

The scheduling algorithm, database, and user interface are currently somewhat disconnected, so some work will need to be done to get these modules working together.

We will need to make some minor updates to the server, including opening a port for web traffic, before the project can be published. Authentication and multi-threading will need to be implemented to safely handle student data and allow multiple users of the system at a time. The first round of usability tests needs to be completed to ensure the user interface meets the requirements of the client.

Individual Requirement Progress

Front-End

Login Page

Started, passwords properly hashed in the database but no front-end functionality yet implemented.

Advising Page

Nearing completion. Currently able to load a test case student plan and manipulate the plan in the browser. Adding and removing courses, modifying schedule and quarter information, and displaying met/unmet requirements is complete. Next step will be wrapping the modified plan into a container that can be sent to the scheduling algorithm.

Drop down menu for students

Mostly complete. Allows user to select a student, starting quarter and catalog year, and degree, and passes that information to the advising page.

Menu for plan constraints

Not implemented, will be added to student menu.

Navigation bar

Navigation and layout is complete. Will require minor modifications to properly show users their correct navigation based on their authentication level.

Administrator page

Mostly complete. Allows users to dynamically modify course and degree information from within the browser. Courses can be modified to change name, department, and quarter information, but adding individual prerequisites is not yet complete. Degree requirements will need minor modification after user testing to ensure proper functionality.

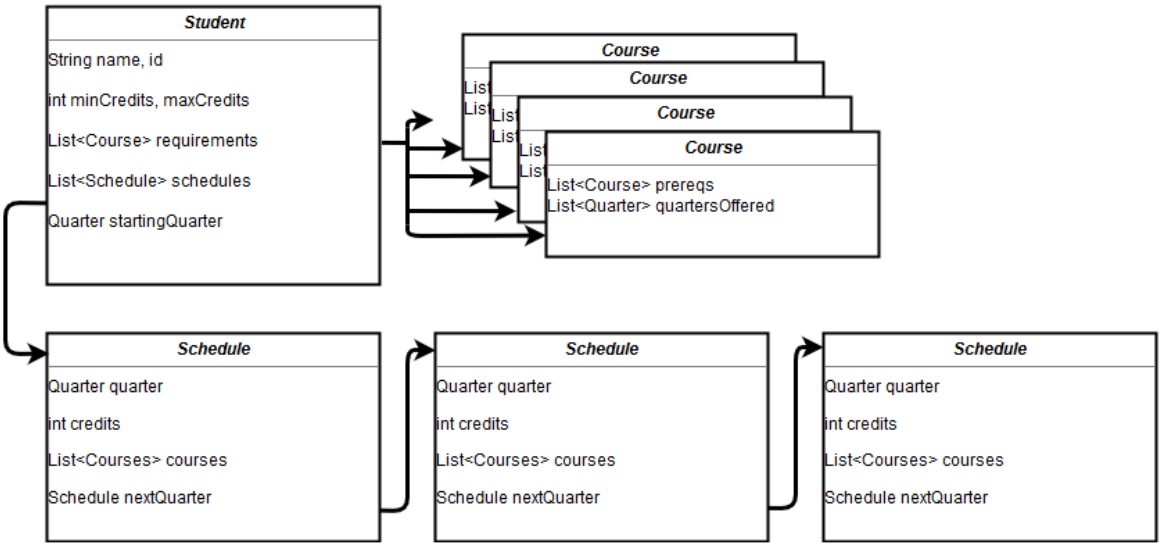
Advising forms

Not yet implemented. Will generate a static, fixed-width html page using same data-transfer method as scheduling algorithm.

Application

Scheduling Algorithm

Partially completed but nonfunctional. The following class diagrams are implemented:



A test driver class has been implemented for testing, and takes both a list of unmet course requirements, and a partially completed, doubly linked list of *Schedule* objects, each containing their quarter, number of credits, and list of courses. The algorithm currently generates permutations of course schedules using a recursive depth-first search, with the completion of all unmet requirements as the base case. The algorithm is able to properly determine at each iteration which courses are possible for the student to take, and automatically moves to the next quarter if a course is not able to fit into the schedule for the current quarter. A round of testing is currently being performed before the algorithm can be developed further.

Database and Server

Individual Databases

All databases are setup and ready to go. There is some test data in them, but we will need to add more test data in the future in order to properly test the function of our algorithm.

Database Handler

The Database Handler is ready for deployment to the server. The main process of the Database Handler is designed to always run on the server, waiting for new clients to connect. Once a client connects, the Handler will spawn a child thread which will handle interactions between the client and the databases.

Web and database server

The virtual linux server has been installed with Debian 9, MariaDB, DB4O, .NET Core, and Mono. Minor configuration changes still need to be made before the site can be deployed.

Multithreading and load management

As mentioned above, our Database handler is running multiple threads to improve performance. This is to maximize the amount of time the Database is being used by a thread. With multiple threads, multiple users' requests can lineup and be ready for database access, thus reducing downtime in between requests.