**MSCI 436 Executive Summary**

Team Members:

Connor O’Brien

Maisie Brown

Ravkirat Kang

Time is a precious resource that is limited for everyone, especially university students. Creating an effective study schedule is hard to do and even harder to maintain when there are so many distractions in a young person’s life. Our business problem deals with people who have a difficult time scheduling their time effectively to excel academically. With a properly optimized study schedule, a student can maintain an active lifestyle and still excel academically. To solve this problem, we have created a DSS in Excel that takes a user’s ‘.ics’ calendar file and optimizes it to fill their free time with study times according to several variables that the user inputs according to their preferences.

The decision maker is relevant to the course material as the system will be using inputs from the user to create an optimal schedule for studying. Users will create weighted criteria for their courses and tasks to complete, and the system will use all these decision variables in Solver to create the final schedule. We also implemented an artificial neural network (ANN) and data gathering to accurately predict the difficulty rating of each course when taught by a specific professor. This project will help us further enhance our optimization modeling skills as the system will require us to create optimization problems with multiple constraints.

The system first uses a form to prompt the user to import their personal calendar in ‘.ics’ format (easily exported from Google/Apple calendar apps) as well as the daily hour range in which they are willing to study (eg. 11 AM to 11 PM) and the date to which they want the calendar optimized. Using the hour range, the program creates an integer calendar on the Optimized tab with each column representing an hour in the range specified by the user and each row representing a date from today to the specified end date.

The first module then opens the calendar in Excel and copies it to the Data sheet. It iterates through the calendar line by line and retrieves information (start/end times, starting date, recurrence, description, etc.) about each event and places the event in the Optimized calendar. Events are classified by their hour, so if there is an event from 1:30 to 2:30 for example, the program would place that event in the columns representing hours 13 and 14 (1 and 2 in 24h time-format).

After the calendar has been imported, parsed and the events are placed in the Optimized sheet, the user then adds up to 6 courses, along with their difficulty rating (1-10) and the professor for each course. When the form is submitted, the program exports the course names into Access as tables with fields ‘Rating’ and ‘Professor’. If the course table exists, Excel adds one row of score and professor into the existing Access table, otherwise it creates a new table and adds the one rating/professor 3 times into the table so the user can later run an ANN to predict the overall course difficulty score.

The user then clicks on the third button, which allows them to add study events to the collection on the Main sheet. The user selects the course, the date, the start time of the event as well as the number of hours they wish to spend studying/working on the event as well as the type of event (lab, assignment, test, quiz). Then they click the Add to Calendar button to either clear the calendar of user input events, leaving the original imported calendar, or to add the events collection into the calendar. The program finds the first n free hours before each event starts and assigns them to studying where n is the number of studying hours.

Next the user clicks the ANN button which pulls all the ratings in the database for each course and professor, sets up an ANN for each course and solves to predict the actual rating for each course, which is then put on the Main sheet. After this, they click the Optimize Calendar button which sets up a optimization problem for Solver. The program first iterates week by week through the calendar and courses section to get the variables needed for the problem, which are: Hi (total free hours in week i), Dj  (Difficulty of Course j from the ANN) and the minimum and maximum study hours per course per week. It then initializes the Solver Matrix with the minimum hours and optimizes the following problem where Tij is the actual study hours per course per week.

s.t.

minhours Tij maxhours

Hi for each i

s.t.

minhours Tij maxhours

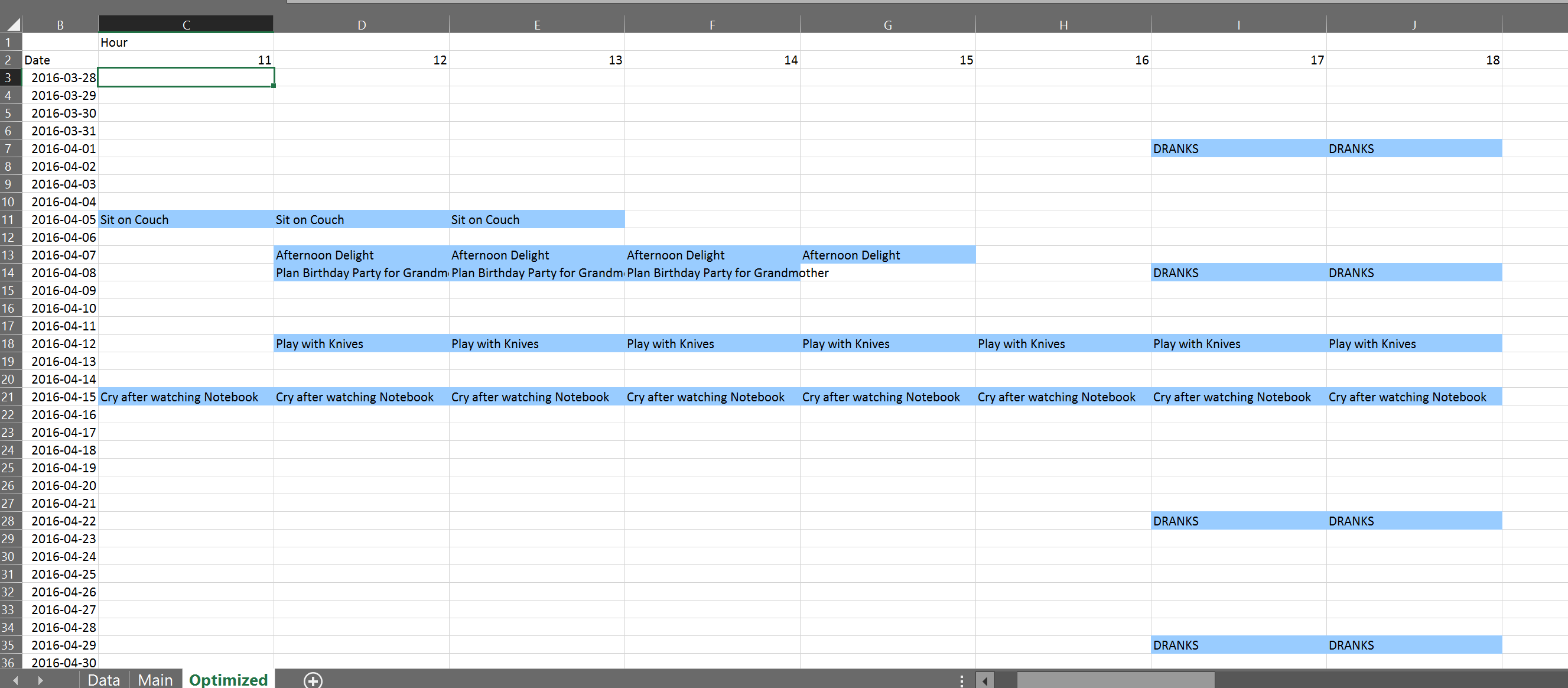
Hi for each i

After solver finishes (about 5 to 10 minutes), the program adds the optimal study times into the free slots in the calendar. Since most of the optimal values aren’t whole numbers, the program first adds as many 1 hour slots as it can for each course, then takes the remainder, converts it into a minute value then adds that value to a string which, at the end, fills up the remaining free hours in the week. The string is in the following format: ***“Study MSCI 444 for 36 minutes, Study MSCI 311 for 36 minutes, Study MSCI 334 for 48 minutes”.*** This is repeated for all weeks in the calendar.

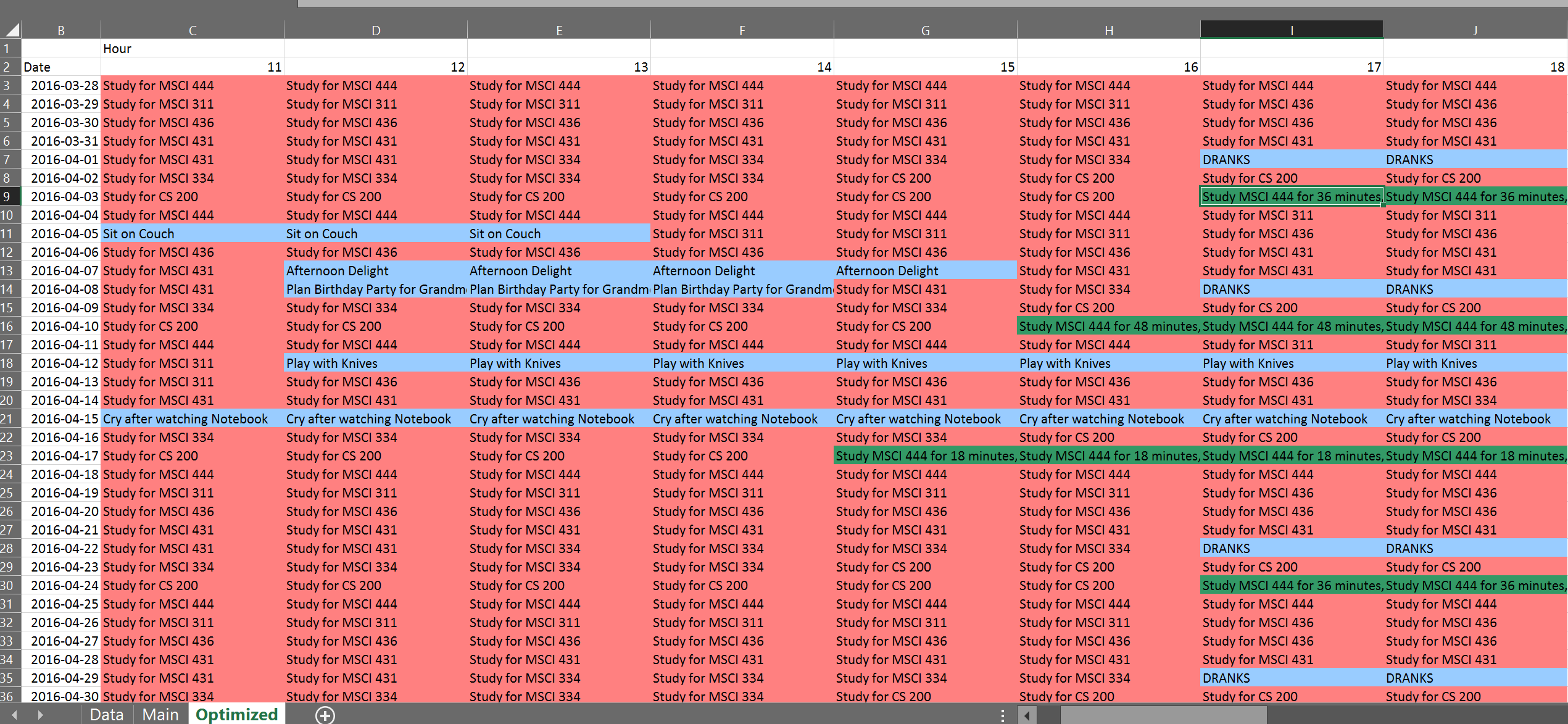
Finally, the user clicksthe Export ICS button, which iterates through the optimized calendar adding each non-original event to a collection, which is then fed through a conversion method to create strings and adds them to the original .ics file which was copied into the Data tab. Then the program opens a stream writer, writes the optimized events to a text file, then saves it as an .ics in the local directory. The user uses this file as a suggested study schedule and can import it into the calendar app of their choice.

The program places the majority of its information on the Data tab which is hidden for the majority of the time, but can be accessed using the show/hide Data button. This allows the user to view both the ANN and the Solver Matrix if they wish. The program uses several of the concepts covered in this course, including an ANN, Solver Optimization Problem, Text Mining from the imported calendar and exporting/importing to and from a database in order to give the user a suggestion as to how they should approach their unique business problem. Future functions of this application include adding non-academic events to the optimized calendar (Gym, Groceries, etc.), improved error handling and half hour time slots to further optimize the schedule.

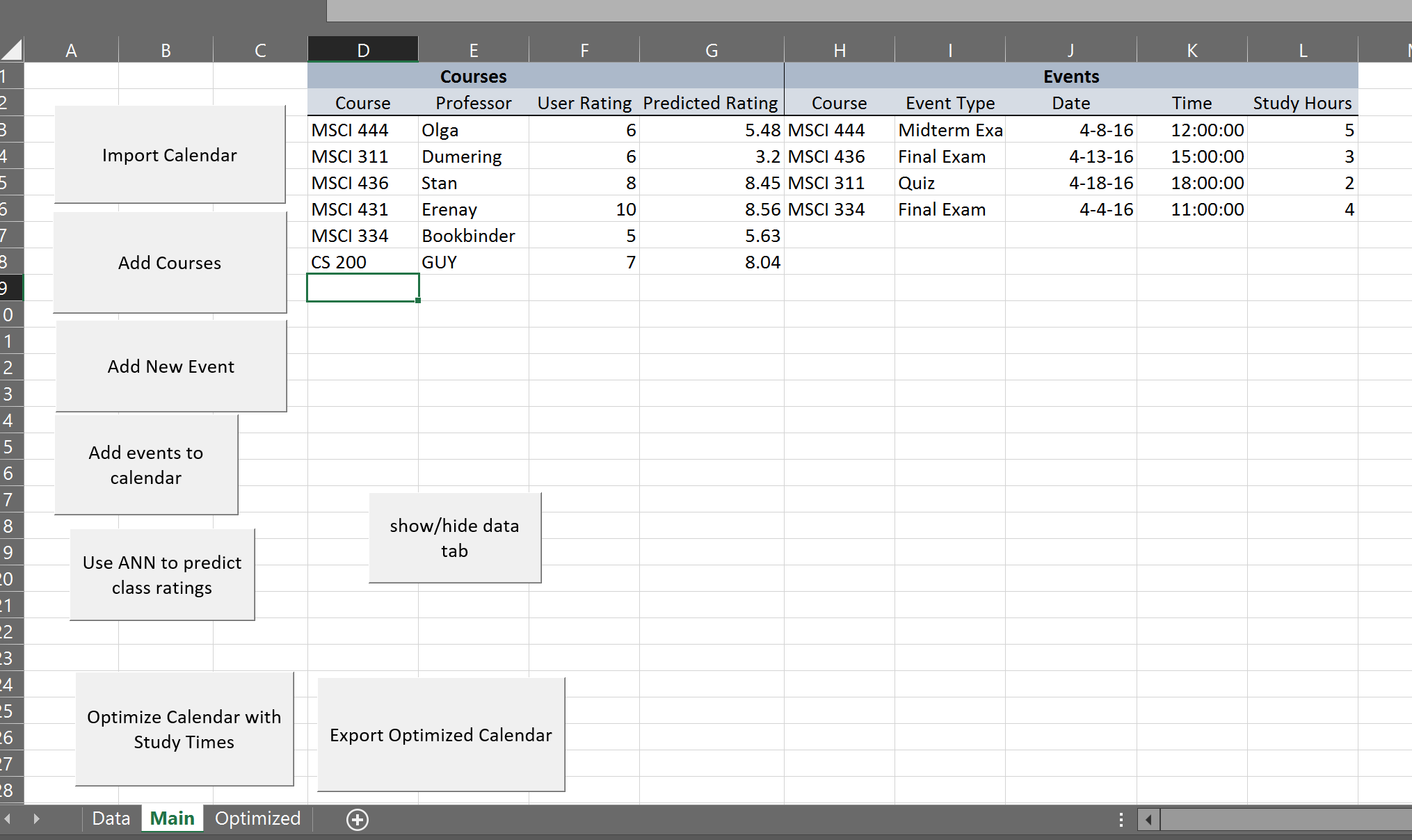
Appendix of Screenshots



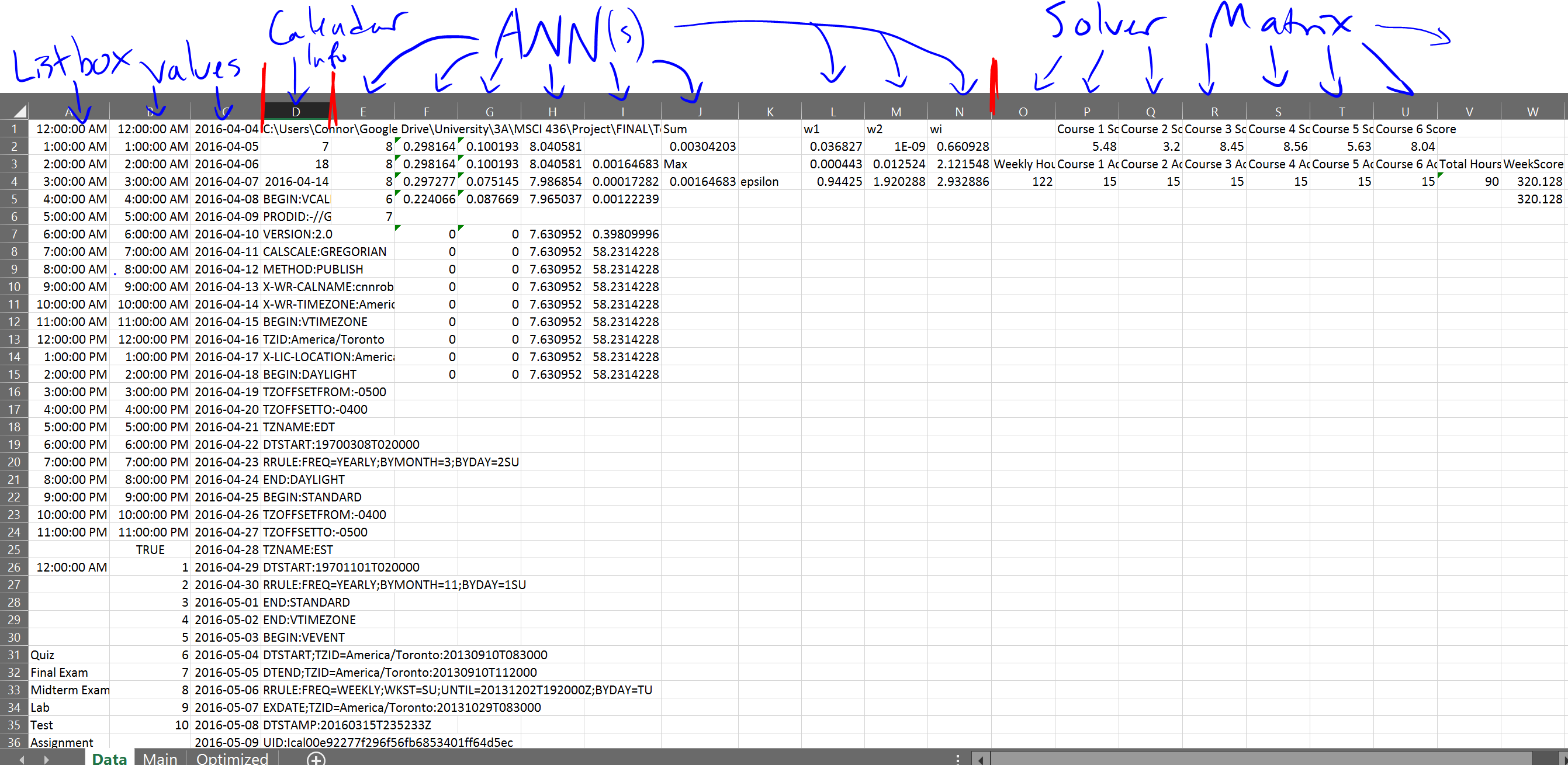
Calendar before optimization



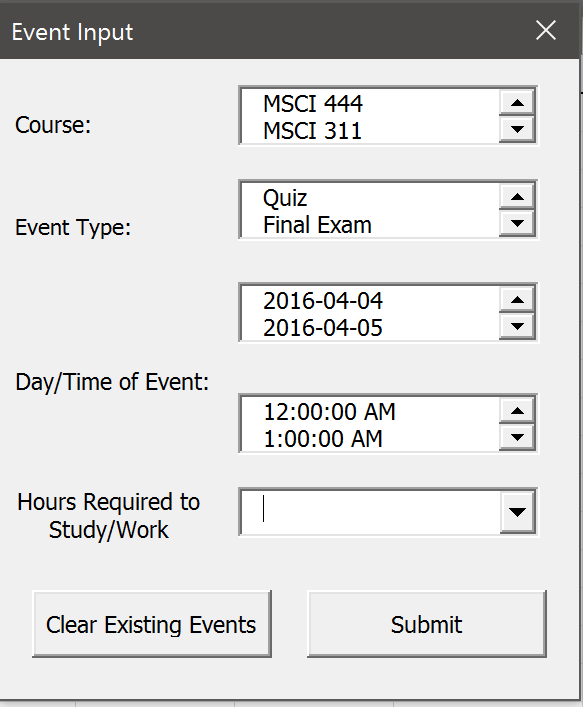
Calendar after optimization

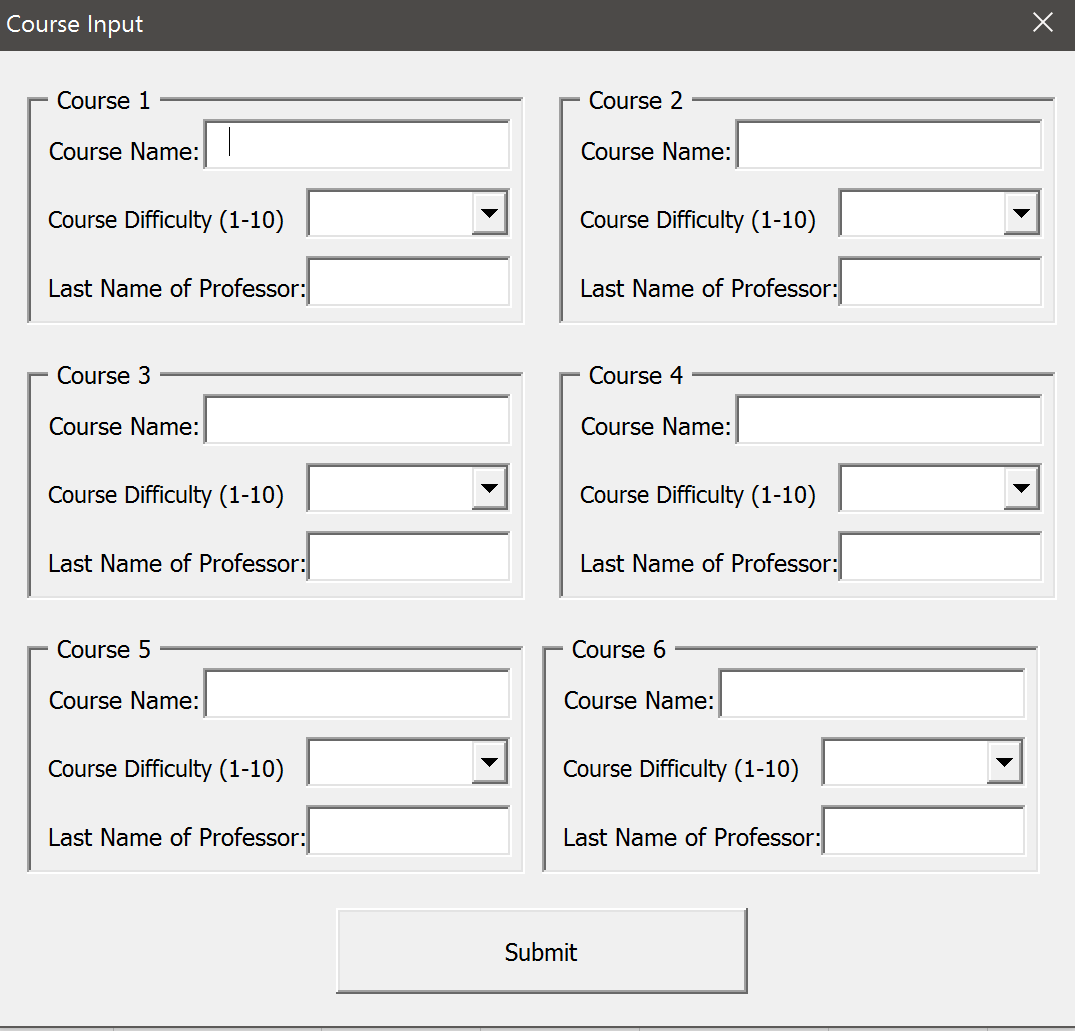


Main worksheet

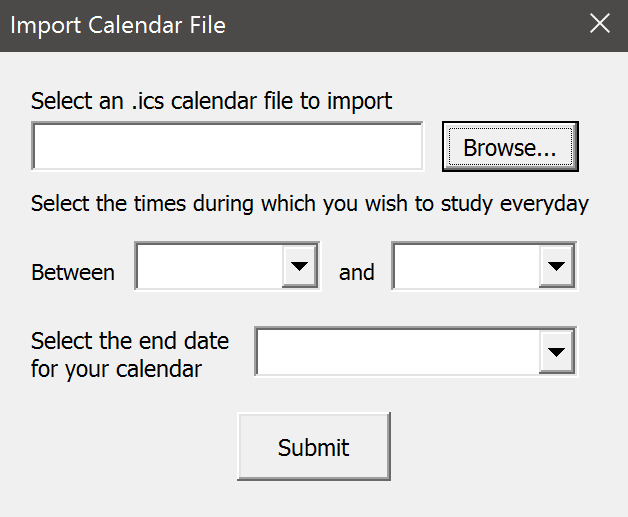


Data worksheet

 Event Input Form



Course Input Form



Import Calendar Form