University of Waterloo

Department of Management Sciences

MSCI 342 Homework 5

Instructor: Dr. Mark Smucker

Team 8

Upload Conflict File (Toni Li)
One of the main components of this system is to distribute students into teams with constraints on their existing interpersonal conflicts. With this in mind, it was a must have to have functionality in the system to have users upload the conflict file that was generated by Team 7. The focus of this story was to enable users to upload the conflict file from Team 7 and pass the information in the file to the Python optimization model.

> REQ-1. If the user clicks the upload file button, a file uploader pops up to allow the user to select a file from their computer
>
> REQ-2. If the user clicks the generate teams button, the file will be passed to the model
>
> REQ-3. If the user uploads a file, the name of their file should appear on the page so that they can ensure they uploaded the file that they wanted

Download Sample Files (Toni Li)
This story was implemented to allow users access to examples of acceptable versions of the student data file and the conflicts file. This was important to enable users to see the type of teams that the system would output based on our sample files and to allow users to see the format of an acceptable student data file and conflict file if they were having issues uploading a file of their own. It was determined that the best way to have these sample files were to embed them on each of the pages where the files were being uploaded. For simplicity, they were made to just be links to files that were held in the same directory as our system files.

> REQ-1. If the user clicks on the ‘Download Sample File’ link on the Upload Student Data page, a sample student data file should download directly to the users computer
>
> REQ-2. If the user clicks on the ‘Download Sample File’ link on the Upload Student Conflicts page, a sample conflict data file should download directly to the users

Upload file format checking (Toni Li)
For our first sprint, there was no error checking for when the user uploaded files. This needed to be improved upon to better guide the user to upload the correct files so that the model could run properly. In this story, both pages where a file was expected to be uploaded had to have error checking built in. The requirements were to check for the correct file type (CSV) and empty files. With each of these test cases, the appropriate error should show and prompt the user to make the necessary changes. To further prevent users from uploading incorrect files, the buttons Next/Generate Teams button is hidden until a file that meets the type and size criteria is uploaded. When the user uploads a correct file, text should appear to notify the user of a successful upload and the Next/Generate Teams button are unhidden from the bottom right corner of the page.

> REQ-1. If the user uploads a file that is not of type CSV, an error should appear that says “Error: Incorrect file type. Ensure file type is CSV and click the Upload

Student/Conflict File button to try again" and the Next/Generate Teams button should be hidden.

REQ-2. If the user uploads a file that is empty, an error should appear that says "Error: File is empty. Please click the Upload Student/Conflict File button to try again." and the Next/Generate Teams button should be hidden.

REQ-3 If the user uploads a non-empty file of type CSV, text should appear that says "Successful upload!" and the Next/Generate Teams button should appear.

Errors on Upload Student Data Files Content (Toni Li)

This story was created in order to make the error checking of uploading the student data files more robust. To add onto the checking of file type and file size, this story was focused on the error checking of the contents of the file, ensuring that at a minimum, the headers for the columns in the file were as expected. This is important to do to ensure that the model takes in the input data that it is expecting and can run the optimization model.

REQ-1. If the user uploads a file that does not have the same column headers as the example file, an error should appear that says "Error: Incorrect file contents. The file uploaded contained headers that were not expected. Please click the Upload Student File button and try again." and the Next button should be hidden.

REQ-2. If the user uploads a non-empty file of type CSV, text should appear that says "Successful upload!" and the Next button should appear.

Populating Team Table (Buse Çelik)

This story was created to ensure the teams the optimization model generated were outputted on the web page correctly in a readable format. After having clicked the generate teams button, the user is then directed to the team output page, which is where all the students are placed on their respective teams with their respective information (ID, name, GPA, Gender, Conflicts).

REQ-1. Students are displayed on the correct teams

REQ-2. Each student has their respective ID, name, GPA, Gender, and Conflicts displayed

Adjusting Student Conflict Column (Buse Çelik)

Initially the conflict column on the team output page had the students' user id, but after discussions with the client, it was decided to have the column display the students first and last name, rather their user id.

REQ-1. If a student is in conflict with another student, then their first and last name is displayed in the conflict column.

Errors on Conflict File Upload (Buse Çelik)
After uploading the conflict file, the system is to check that the file is in the correct format that teammate requires it to be in or else the model will not be able to run, and so the teams will not be generated and outputted. The user will be informed if the headers in the conflict file are incorrect.

> REQ-1. If the conflict file uploaded has the correct headers, user will be informed of successful upload.
>
> REQ-2. If the conflict file uploaded has the incorrect headers, user will be informed of the unsuccessful upload stating the headers are incorrect.

“Configure Team” Page - Front End (Hashim Vekar)
The purpose of the “Configure Teams” page is display information from the uploaded students.csv file and to take in input that helps with the development of teams. The page shows the number of students found in the students information file, a text box asking for the number of members the user wants per team, and the number of teams created. The requirements of this page is to ensure that the page accepts the input of the user on members per team and performs the right calculation of number of teams that will be created.

> REQ 1 - For the team input size, the user has to put an integer greater than 0, but less than the total number of students
>
> REQ 2 - When team input size is entered, the text explaining how many teams of certain members will be shown e.g. 2 teams of 3, 1 team of 2

Export File as a .txt file (Hashim Vekar)
The only output of our application is the export of the generated teams. Exporting a .txt file that contains 3 columns of student information: the nexusID, full name, and team number. This story is to ensure the file is of the correct format and output when exporting teams.

> REQ 1 - Every time the user clicks the “Export Teams” button on the Teams page, a .txt file gets downloaded to the local downloads folder with the correct format and student information.

Export File as a .csv (Hashim Vekar)
One of the main functionalities of our application is to export the generated teams for the user. Exporting a .csv file that contains all of the student’s information (as seen in the Student upload file: nexusID, first name, last name, email, GPA, and gender) with an additional column of the team number they are on, completes this functionality. The user can manipulate and analyze the contents of the students’ information before they are disclosed. This story is to ensure the file is of correct format and output that is desired by the client when exporting teams.

REQ 1 - Every time the user clicks the black “Export Teams” button on the Teams page, a .csv file gets downloaded to the local downloads folder with the correct format and student information (including the team number).

Model Failure Notification due to Unexpected Interruption (Ian Kemp)

While the model is designed to always output a feasible solution, there is a possibility that the model either fails to execute or cannot produce a feasible solution (for reasons unexpected or unknown). Without this story, it is possible that the model may appear like it is executing when it has actually stopped. The user is then forced to wait and finally exit the page when they have decided to give up. To prevent this from happening, the application must react to an error and update the interface so the user can proceed to either fix the problem or try again. The requirements are as follows:

REQ-1. If the execution of the model fails, the script stops executing and the interface updates to show that the model is no longer running.

REQ-2. If the execution of the model fails, the interface displays what type of error occurred (either due to invalid input, infeasible solution, or other unexpected issues).

REQ-3. If the execution of the model fails, the interface displays the exact error message produced by the Python script failing.

REQ-4. All notifications should specify at least one suggested action for the user to proceed in order to successfully generate teams.

Conflict Distribution Model Constraint Update (Ian Kemp)

An important constraint of the model is preventing the occurrence of student conflicts. If a student has indicated that they are in conflict with another student in the class, they should not be placed on the same team. Each pair of students of a conflict counts as one conflict, and it is that students may have indicated that they both have conflict with each other. The model has the conflict constraint as the absolute highest priority, ensuring that student conflicts must always be satisfied before attempting to satisfy other model constraints.

REQ-1. If student A has a conflict with student B, student A is not placed on a team with student B by the model.

REQ-2. If student A has a conflict with student B, and student B has a conflict with student A, student A is not placed on a team with student B by the model.

REQ-3. If a student has indicated more than one conflict, all conflicts will be satisfied according to REQ-1 and REQ-2.

REQ-4. If there is no feasible configuration of students that prevents conflicts, the model will return a solution that minimizes the number of active conflicts. When a suboptimal

solution is returned, a visual indicator notifies the user that the output teams contain unsatisfied conflicts.

Team Size Distribution Model Constraint Update (Ian Kemp)
Within the application, the user enters the desired team size for all teams. This is a maximum value, so teams will always be the specified size or one member smaller. Within the model, a team size constraint minimizes the deviation of actual team size from the desired team size. A slack variable is added that records the amount that a team fails to reach the desired team size.

REQ-1. Every student must be placed on a team by the model.

REQ-2. Every team must have either the number of students indicated by the user, or one less student such that all teams are as balanced as possible.

Gender Distribution Model Constraint Update (David Herrington)
In our application, the user has the ability to upload a file that consists of student's genders. As of now, the options are 'w' for women, 'm' for man, and 'x' for others. There is a possibility for adding another variable as prefer not to disclose, and will be finalized once a meeting with the equity office has taken place. This specific constraint in the model is required to construct teams such that they consist of greater or equal to 50% females. In the case that the class is less than 50% females, 100% male teams are formed with the remaining students. In other words, this constraint is to ensure that those who identify as women are to never be a minority on a team. This story has already been worked on previously, and is needing to be further updated in the model. The requirements are as follows:

REQ-1. If the file has a column for genders, the three options should be included in the model and distributed accordingly.

REQ-2. If there is women on a team, the team will consist of at least 50% women.

GPA Average Distribution Model Constraint (David Herrington)
The model uses inputted students' GPAs to balance the strength of teams. This story's purpose is to make sure the team averages are close to that of the overall average. It finds the individual team's average and compares it to the class average. This constraint is minimizing that difference between the two in order to properly balance teams.

REQ-1 The difference between any teams average and the overall class average will be minimized

REQ-2 The difference between any teams average and another teams average is also minimized

GPA Deviation Distribution Model Constraint (David Herrington)
As mentioned above, the model uses inputted students' GPAs to balance the strength of teams. This story's purpose is to ensure that student's averages are distributed evenly instead of the total team average just being close to the class average. For example, if the class average is 80, a team could have 60, 60, 100, 100 and a team average of 80. This satisfies the first GPA constraint, but it is not an even distribution. The deviation constraint ensures that students are distributed optimally among teams. By minimizing the deviation from the average, every team will have almost the exact same distribution of averages, such that they are mathematically balanced.

REQ-1. Teams must be distributed such that there is not an uneven balance of GPAs

REQ-1. Teams must be distributed such that their total deviation is minimized, resulting in having an optimal range of group members.

Make front-end user friendly (Nikhar Dhingra)

Story: As a user, I want to be able to interact with the user-friendly frontend for the system

Draft Requirement: Ensure the user is able to:
- See the page title at the top of every page
- Read the instructions on the right side of every page
- Transition from the first page to the next, only after he/she has uploaded a .csv file

Concise Requirement:
Given: I am a professor on any web-page
When: I view the page
Then: I should be able to see the title and instructions of the page
Acceptance Criteria: The user can interpret and follow the instructions on each page. The user can easily recognise the sequence of buttons he/she has to interact with.

REQ-1**.** The user can not move to the next page without uploading a file.

Enter number of teams (Nikhar Dhingra)

Story: As a user, I should be able to enter in the number of teams, that I want to create using the system.
Draft Requirement: Ensure the user is able to:
- Enter a number between 1 and the total number of students he/she has in the class roster.

Concise Requirement:
Given: I am a professor on page 2
When: I enter in the number of teams

Then: I should be able to see the number of students in each team
Acceptance Criteria: The user should be able to enter and view the number of students in each team.

Constraint selection for the website (Nikhar Dhingra)

Story: As a user, I should be able to select whether I want GPA constraint factored in for generating teams.

Draft requirement: Ensure the user is able to:

- Select or Un-select the GPA constraint.

Concise Requirement:
Given: I am a professor on page 2
When: I select the constraint GPA
Then: The model should use that for generating teams
Acceptance Criteria: The user should be able to generate teams with GPA constraint if GPA checkbox is “selected”. If GPA checkbox isn’t selected then the team should be generated with only conflicts in mind.

Real-time issues box verification (Yining Huang)
Issues are created when suboptimal solutions are created by the goal optimization model. These are inter-team conflicts that have not been met by the constraints. Within the scope of this project at the stage of sprint 4, there are two types of possible issues: GPA issues and personal conflicts issues. A GPA issue happens when a team is missing a student from the top tier or bottom tier or both (Top tier students have the top n GPAs in the class; Bottom tier students have the lowest n GPAs in the class, where n is the number of teams). A conflict issue happens when a student is placed on a team with whom they have a personal conflict with.

REQ1: shows all inter-team issues on loading the teams output page

REQ2: updates all inter-team issues when a member is being moved to another team

Configure team page - dynamic information (Yining Huang)
The configure-team page dynamically shows the number of students that are included in the students file uploaded in the previous page. The user can then enter the maximum number of people per team, and the page will dynamically tell the user the number of teams that will be generated, along with the number of people on each team, even when the teams have different sizes.

REQ1: dynamically show how many students have been found in the uploaded file

REQ2: user cannot enter any non-negative integers for team size

REQ3: the page dynamically shows how many teams will be created, along with how many people are on each team. This gets updated every time the user changes the team size in Req 2. The result will only contain non-negative integer values, both for the number of teams and the number of students on each team

Manually move members to other teams (Yining Huang)

User can move a student to another team after viewing the output page if they wish, despite the fact that the output will be the most optimal solution.

REQ1: the user interface reflects the move upon clicking “submit”

REQ2: the exported file reflects the move upon clicking “export teams”

REQ3: user can move a student to any other team, except for the team he is current on

REQ4: it is clear to the user which student is being moved, from and to which team