# groupre Design Document

Architecture

Abstract Overview

The following diagram outlines groupre's basic flow. Inputs are received in the form of arguments, which include two input csv files – chairs and students – which are passed in through the web-app user interface alongside various other settings that will control the functionality of groupre. These arguments are taken in by the groupre input module and accordingly sorted through the matching algorithms that the input files call for. Once all matching is completed, groupre outputs a csv file containing the results of the sorting algorithms. Eventually, this output will instead be sent to the web-app user interface for user editing before a final output file is generated.
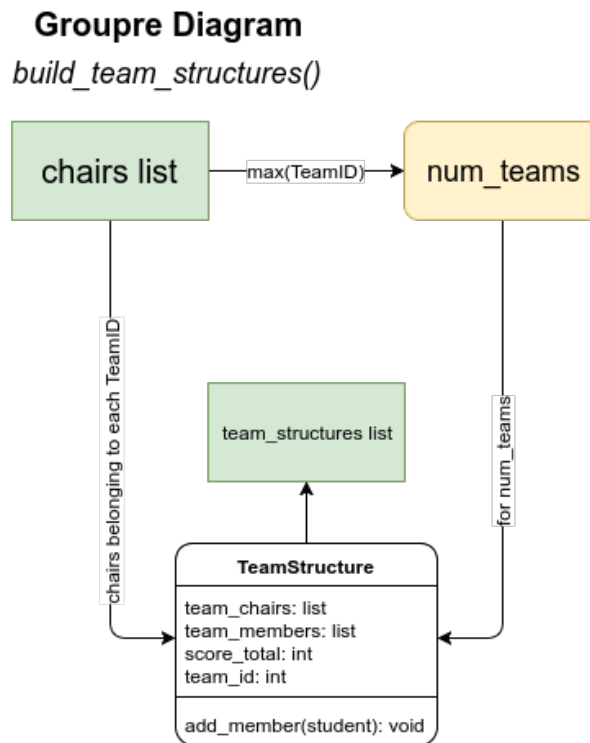

**Abstract Overview**
Groupre Web App (alt.)
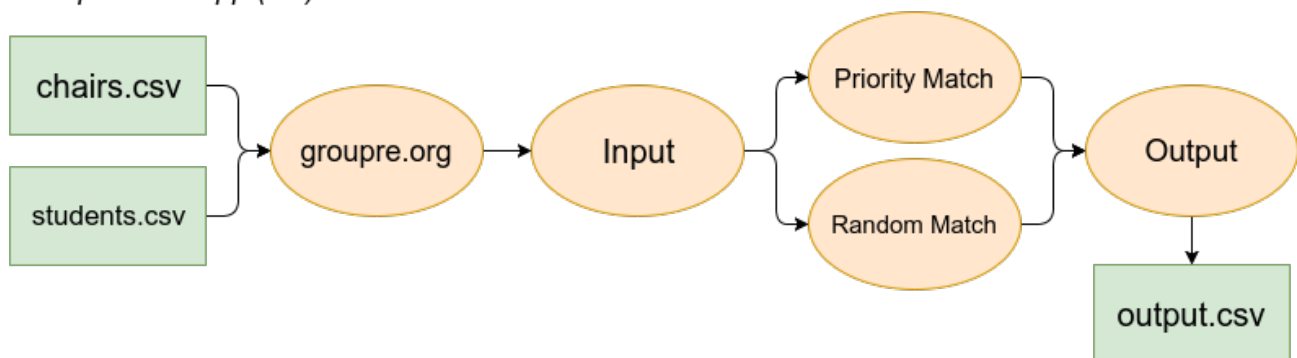
Team Structures

The following diagram depicts a high-level overview of the Team Structures that will be used for student-to-student attribute matching. Team Structures are created through the input chairs csv file, which determines the amount of teams to keep track of as well as what chairs belong to what team at run-time.

Groupre Diagram
build_team_structures()

```
chairs list ──max(TeamID)──► num_teams
     │                            │
chairs belonging to each TeamID   for num_teams
     │                            │
     │      team_structures list  │
     │              ▲             │
     │              │             │
     │     ┌──────────────────┐   │
     └────►│  TeamStructure   │◄──┘
           ├──────────────────┤
           │ team_chairs: list│
           │ team_members: list│
           │ score_total: int │
           │ team_id: int     │
           ├──────────────────┤
           │ add_member(student): void │
           └──────────────────┘
```

Creation of Teams

The following diagram depicts the workflow of the groupre module's allocation of its inputs into the matching algorithms it uses. The input csv files are stored as lists, and Team Structures are generated. These Students are then allocated into two primary matching algorithms, Priority Match and Random Match, depending on the amount of preferences a given Student has. After running through the sorting algorithms, a list of Teams is generated and sent back to the input/output module of groupre, which produces the output file.

Groupre Diagram
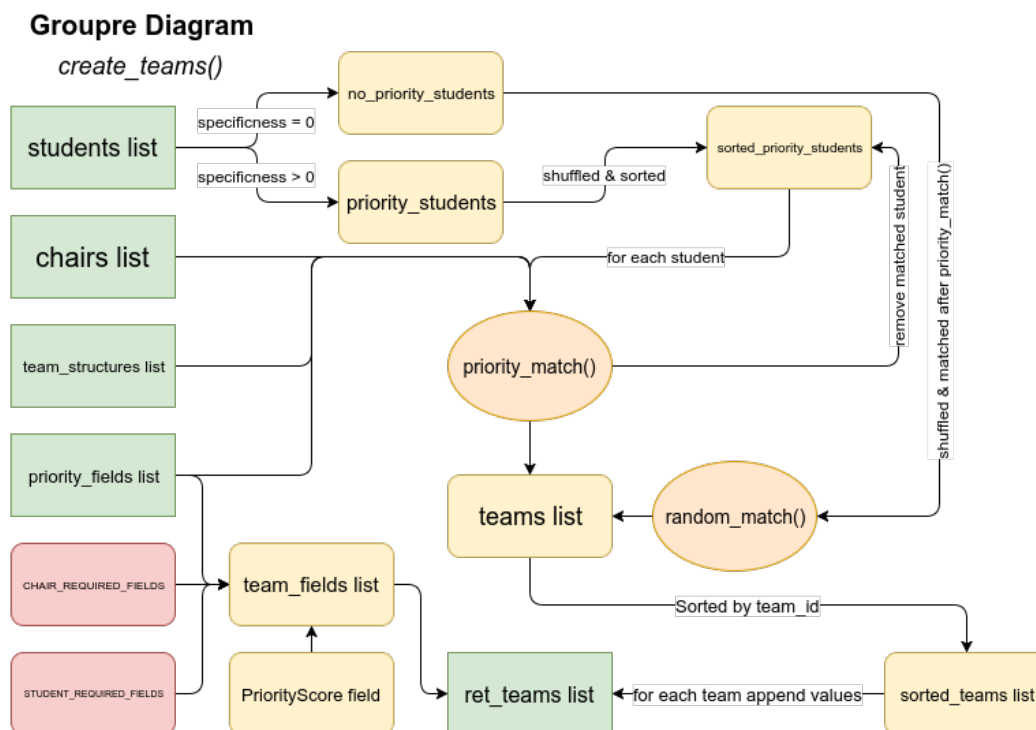
groupre Overview

This is a more detailed diagrammatic overview of each step that the groupre module takes between receiving input and producing an output file.
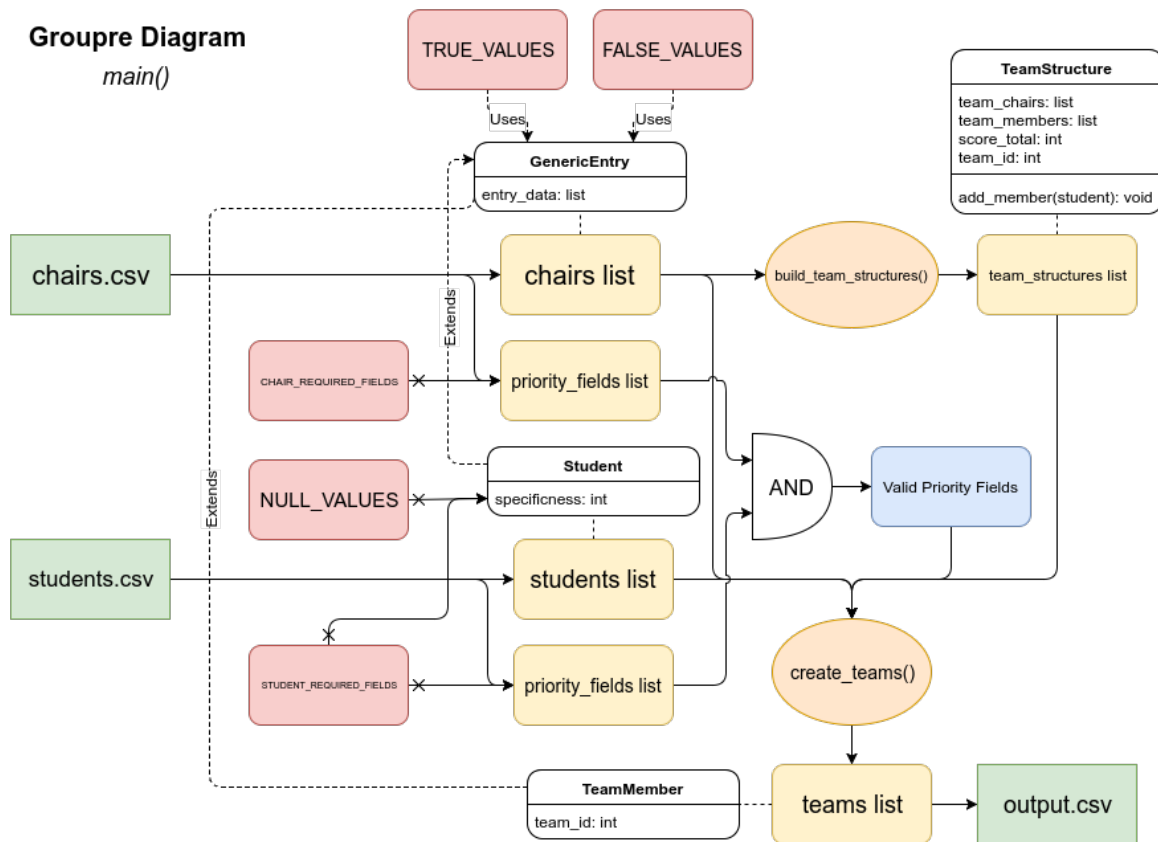
Decomposition

Modules

- Input

  - In this module, data is taken into the program as arguments, which denote options that the user has chosen as well as their specified input csv files that will be processed. This data is parsed and stored into data structures, which are maintained in lists. These internal data structures are then passed into groupre's sorting algorithms based on their attributes.

- Matching Algorithms

    - The matching algorithms used by groupre process the data collected by the input module, prioritize them based on their attributes, and sort them based on their attributes in comparison to possible target chairs as well as other students that are already within teams. The output of these matching algorithms is then sent back to the main module of groupre, which handles both input and output for final output file generation.

- Output

    - The results of the matching algorithms are taken and allocated into a text csv format, which is then returned either to the user in the case of command-line operation or to the web-app for download or displaying back to the user.

Design Decisions

The back-end of our project is to be coded in Python, which will support basic offline functionality as a standalone module. This module is to then be integrated into a web-app deployed on Carolina CloudApps as a Python 3 application. As the back-end code is built upon, the web interface will be completed alongside it, offering a more user-friendly visual tool for utilizing the groupre module. The web-app is to be designed and coded using a mixture of Python code in the Flask framework and JavaScript alongside basic HTML.

The GUI will be designed such that the user is able to dynamically construct the classroom in a grid-like fashion using standard keyboard-shortcuts for productivity. Their classroom design will be able to be stored for future use either by the user alone or by other users who wish to start with a template in the same room "Blueprint". The user will be able to select and assign various rows and columns of Chairs to which they can assign attributes, which will be represented by colored icons. After the classroom is built, it will be used by the groupre module as the Chairs input file. The output of the groupre module will eventually be interpreted such that it is displayed within the same blueprint that the user designed, so that they can make manual edits if necessary before downloading a finalized output file.