**Technical Documentation – Student Group Creation Project**

1. **General Overview**

**1.1. Project Objective**

The project involves developing a web application that allows a teacher to automatically create student groups while taking into account the preferences expressed by the students.

**1.2. Main Feature**

* Authentication (student / teacher)

• Deletion of students by the teacher

• Entry of team preferences by students (weighting from 1 to 100)

• Ability to modify preferences as long as choices are allowed

• Automatic group generation using a satisfaction algorithm (CP-SAT)

• Display of groups and overall satisfaction score

• Management interface for the teacher (publishing results, opening/closing choices, student count)

• Option for the teacher to delete all groups via a button

1. **System Architecture**

**box-cert-project/**

**│**

**│── data/**

**││─ choice.json**

**││─ group.json**

**││─ users.sql**

**││─ auth/**

**│── docs/**

**│── src/**

**│ │── auth/**

**│ │ └── app.py**

**│ │── static/**

**│ │ │── images/**

**│ │ │── auth.css**

**│ │ │── styles.css**

**│ │── templates/**

**│ │ │── auth/**

**│ │ │ │── signin.html**

**│ │ │ │── signup.html**

**│ │ │── index.html**

**│ │ │── student.html**

**│ │ │── teacher.html**

**│ │── algo.py**

**│── tests/**

**│ └── test\_algo.py**

**├── README.md**

**├── requirements.txt**

**└── to\_do\_list.xlsx**

**3. Technological Choices**

| **Composant** | **Technology** | **Justification** |
| --- | --- | --- |
| Backend | **Flask/Python** | Lightweight, simple, and well-understood |
| Templates | **Jinja2** | Easy integration with Flask, already familiar |
| Authentication | **MySQL** | More robust database management system |
| Preference Storage | **JSON** | Easy to read and modify dynamically |
| Algorithm | **Google OR-Tools (CP-SAT)** | Efficient constraint solving |
| Frontend | **HTML + CSS + JS** | Simple and dynamic interface |

**4. Interpretation of the Topic**

* The teacher is the administrator: they log in, manage users, set the group size, launch the algorithm, open and close the preference submission period for students, and publish the results.
* Students access the application via login, enter their preferences (sum = 100), can modify them as long as submission is open, and view the results once they are published and the submission is closed.

**5. Group Creation Algorithm Functioning**

**1. Overall Objective**

The goal is to automatically form student groups while maximizing mutual preference satisfaction expressed between them. This is achieved using a constraint solver (Google OR-Tools CP-SAT), which finds the optimal distribution.

1. **Variables and Constraints**

**Variables**

x[i, g]: Boolean variable that is 1 if student *i* is assigned to group *g*, otherwise 0.

**Constraints**

* Unique Assignment: Each student must be assigned to exactly one group.

1. **Affinity Calculation**

Each student can express weighted preferences for their peers (e.g., "I want to be with Alice, weight = 50"). These preferences are transformed into a score matrix:

* **score[i][j] = weight given by student i to student j**

The mutual score between two students is then calculated as:

* **mutual\_score = score[i][j] + score[j][i]**

1. **Objective Function**

The algorithm aims to maximize the sum of mutual affinities within groups. For this:

* For each group *g* and each pair of students (i, j):
* Create a variable same\_group which is 1 if both i and j are in group g, 0 otherwise.
* Add mutual\_score \* same\_group to the objective function.

**5. How the Solver Decides Groups**

The CP-SAT solver explores all possible assignments of students to groups under the following constraints:

* Each student is in exactly one group
* Groups are balanced in size
* The objective is to maximize the sum of mutual scores
* This results in an optimal distribution based on the expressed affinities.

**6. Group Balance**

* **Balanced Size:**The algorithm ensures nearly equal group sizes (±1 if necessary).
* **Score Balance**Not guaranteed. The algorithm maximizes the total sum of affinities, not equality between groups. Therefore, some groups may have very high scores while others less so, if this maximizes the overall total.

**6. Installation and Getting Started Instructions**

**6.1. Prerequisites**

Python 3.8+

MySQL Server (e.g., via Wamp, Laragon or MySQL Workbench)

Flask==2.3.3

mysql-connector-python==8.2.0

Werkzeug==2.3.7

ortools==9.6.2534

Jinja2==3.1.2

**6.2. Installation**

- git clone https://github.com/Fjsksksk/box-cert-project

- cd box-cert-project

- pip install -r requirements.txt

- For the database, a MySQL SQL script is included in the repo; download it and import it using a MySQL tool (Wamp, Laragon, etc.). Modify the password in app.py to set the correct connection parameters.

**6.3. Running the Application**

cd src

python app.py

The app will be accessible at: [http://localhost:5000](http://localhost:5000/)

**7. Credits**

Candice : Front-End Developer

Joe : Back-End Developer

Maxence : Data Engineer