**Documentation Technique – Projet de création de groupes d’étudiants**

1. **Présentation Générale**

**1.1. Objectif du projet**

Le projet consiste à développer une application web permettant à un professeur de créer automatiquement des groupes d’élèves en tenant compte des préférences exprimées par ceux-ci.

**1.2. Fonctionnalités principales**

* Authentification (étudiant / professeur)
* Suppression des élèves par l’enseignant
* Saisie des préférences d’équipe par les étudiants (pondération 1-100)
* Modifié ces préférences tant que les choix sont autorisés
* Calcul automatique des groupes via un algorithme de satisfaction (CP-SAT)
* Affichage des groupes et du score global de satisfaction
* Interface de gestion pour le professeur (publication des résultats, ouverture/fermeture des choix, nombre d’étudiant)
* Possibilité de supprimer **tous les groupes** via bouton enseignant.

1. **Architecture Système**

**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. Choix Technologiques**

| **Composant** | **Technologie** | **Justification** |
| --- | --- | --- |
| Backend | **Flask/Python** | Léger, Simple et Maîtrisé |
| Templates | **Jinja2** | Intégration simple avec Flask, déjà expérimenté |
| Authentification | **MySQL** | Système de gestion de base de données plus robuste |
| Stockage préf. | **JSON** | Simple à lire/modifier dynamiquement |
| Algorithme | **Google OR-Tools (CP-SAT)** | Résolution de contraintes efficace |
| Frontend | **HTML + CSS + JS** | Interface simple et dynamique |

**4. Interprétation du sujet**

* Le **professeur** est l’administrateur : il se connecte, il gère les utilisateurs, la taille des groupes, le lancement de l’algorithme, la fermeture et l’ouverture des choix aux étudiants et la publication des résultats.
* Les **étudiants** accèdent à l’application via login, saisissent leurs préférences (somme => 100), peuvent les modifier tant que la saisie est ouverte, et voient les résultats une fois publiés et publication fermé

**5. Fonctionnement de l’algorithme de création de groupes**

**1. Objectif global**

L’objectif est de former automatiquement des groupes d’étudiants tout en maximisant la satisfaction des préférences mutuelles exprimées entre eux. Cela se fait via un solveur de contraintes (CP-SAT de Google OR-Tools) qui trouve la répartition optimale.

**2. Variables et contraintes**

**Variables**

x[i, g] : variable booléenne qui vaut 1 si l’étudiant i est assigné au groupe g, sinon 0.

**Contraintes**

* Affectation unique : chaque étudiant est affecté à exactement un groupe

1. **Calcul des affinités**

Chaque étudiant peut exprimer des préférences pondérées pour ses camarades (par exemple : "Je veux être avec Alice, poids = 50"). Ces préférences sont transformées en matrice de score :

* **score[i][j] = poids donné par l'étudiant i à l’étudiant j**

On calcule ensuite le score mutuel entre deux étudiants :

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

1. **Fonction objectif**

L’algorithme cherche à maximiser la somme des affinités mutuelles à l’intérieur des groupes. Pour cela :

* Pour chaque groupe g, et chaque paire d’étudiants (i, j) :
* On crée une variable same\_group qui vaut 1 si i et j sont tous les deux dans g , 0 sinon.
* On ajoute mutual\_score \* same\_group à l’objectif

**5. Comment le solveur décide des groupes**

Le solveur CP-SAT explore toutes les affectations possibles des étudiants aux groupes sous les contraintes suivantes :

* Chaque étudiant est dans un seul groupe
* Les groupes sont équilibrés
* L’objectif est de maximiser la somme des scores mutuels
* Cela donne une répartition optimale selon les affinités exprimées.

**6. Équilibre des groupes**

* **Taille équilibrée :**L’algorithme garantit une taille quasi égale pour chaque groupe (±1 si nécessaire).
* **Équilibre des scores :**Ce n’est pas garanti. L’algorithme maximise la somme totale des affinités, pas l’égalité entre groupes. Donc, certains groupes peuvent avoir un score très élevé, d’autres moins, si cela maximise le total global.

**6. Instructions d'installation et de mise en route**

**6.1. Prérequis**

Python 3.8+

MySQL Server (ex: via Wamp, Laragon ou 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

- pour la bdd, un script sql MySQL se trouve dans le git, il faut le télécharger et l’importer dans un logiciel MySQL (Wamp, Laragon, etc..).

Il faut modifier le mot de passe dans app.py afin d’établir les condition.

**6.3. Lancement de l’application**

cd src

python app.py

L'application sera disponible à l'adresse : [http://localhost:5000](http://localhost:5000/)

**7. Crédits**

Candice : Développeuse Front-End

Joe : Développeur Back-End

Maxence : Ingénieur Data