**Software installation**

1. Download and install Python 3.12.0
2. Download install vs code
3. Type `pip install streamlit`

**Software run**

1. Open project on vs code
2. Open terminal
3. Type `streamlit run app.py`

**Project Explanation**

This project uses Python and the Streamlit library to create a timetable scheduling system. The system leverages a Genetic Algorithm (GA) to efficiently allocate teachers, subjects, and venues while minimizing conflicts like teacher or venue clashes.

**1. Library Imports**

The code begins by importing essential libraries:

* **Streamlit** for building the web-based user interface.
* **Pandas** for handling data frames.
* **NumPy** for numerical operations.
* **DEAP** (Distributed Evolutionary Algorithms in Python) for implementing the Genetic Algorithm.
* **Random** for random selection operations.
* **Functools.partial** for partial function application.

**2. Data Structure Definitions**

Three custom classes are defined:

* **Teacher Class**:
  + Represents a teacher and contains a list of subjects that the teacher can teach.
  + Methods:
    - assign\_subject(subject): Adds a subject to the teacher's list.
* **Subject Class**:
  + Represents a subject.
* **Venue Class**:
  + Represents a venue where classes can be held.

**3. Genetic Algorithm Setup**

The Genetic Algorithm (GA) is used to generate optimal or near-optimal timetables. The DEAP library is used to implement the GA.

* **Fitness Function**:
  + The code creates a FitnessMin class to minimize conflicts.
  + It also defines an Individual class that inherits from a Python list and uses FitnessMin.
* **Generate Individual**:
  + The generate\_individual function creates a timetable by randomly assigning teachers, subjects, and venues for each section and hour.
  + Each individual (solution) in the population represents a complete timetable.

**4. Fitness Evaluation Function**

* The evaluate function checks for teacher and venue clashes in the timetable.
  + The function iterates over each slot in the timetable to ensure no teacher or venue is double-booked.
  + A penalty score is incremented whenever a clash is detected. The GA aims to minimize this penalty score.

**5. Genetic Operators**

* **Mutation**:
  + The mutate function swaps a randomly selected teacher or venue in the timetable to introduce variety and help the algorithm escape local minima.
* **Crossover**:
  + The crossover function performs a single-point crossover between two individuals. This helps combine features of two solutions, potentially leading to better offspring.

**6. Timetable Generation Function**

The generate\_timetable\_ga function orchestrates the entire GA process:

* It sets up a Toolbox object that contains the genetic operators:
  + individual to generate individuals.
  + population to generate an initial population of size 100.
  + evaluate for fitness evaluation.
  + mate for crossover.
  + mutate for mutation.
  + select for tournament-based selection.
* **Genetic Algorithm Execution**:
  + The eaSimple function from DEAP runs the GA with the following parameters:
    - Crossover probability (cxpb) of 0.7.
    - Mutation probability (mutpb) of 0.2.
    - 50 generations (ngen).
  + The algorithm returns the best solution (individual) with the lowest penalty score.
* **Output**:
  + The best timetable is extracted and formatted into a Pandas DataFrame for easy visualization.

**7. Streamlit User Interface**

The user interface is built using Streamlit, which provides an interactive web-based form to collect user inputs:

* **Number Inputs**:
  + Number of teachers, sections, subjects, venues, and hours per day.
* **Input Validation**:
  + Ensures that the number of hours per day does not exceed the number of teachers.
  + Ensures the number of venues is greater than or equal to the number of sections.
* **Subjects Information**:
  + Users can expand sections to input details about each subject.
* **Teachers Information**:
  + Users can assign subjects that each teacher can teach.
* **Venues Information**:
  + Users can specify venue names.
* **Timetable Generation**:
  + Once all inputs are provided, users can click the "Generate Timetable" button.
  + If all necessary data is entered, the system uses the GA to generate and display the timetable.
  + If any data is missing, appropriate error messages are displayed.

**8. Conclusion**

This solution uses a Genetic Algorithm to optimize timetable scheduling by minimizing conflicts. The system allows flexibility in terms of teacher assignments, subjects, and venue availability, making it suitable for educational institutions looking to automate their scheduling process.