steps involved in app.py

Initial Setup and Imports

---- Libraries Used:

Flask: For creating the web application.

sqlite3: For interacting with the SQLite database.

pandas: For reading and processing Excel files.

random: For random selection in the genetic algorithm.

prettytable: For displaying data in a formatted table.

enum: For defining conflict types.

---- Flow:

The necessary libraries are imported.

Constants like POPULATION_SIZE, NUMB_OF_ELITE_SCHEDULES, TOURNAMENT_SELECTION_SIZE, and MUTATION_RATE are defined for the genetic algorithm.

Database Manager (DBMgr)

---- Purpose: Manages the database connection and fetches data from the SQLite database.

---- Flow:

Initializes the database connection and fetches data for rooms, timings, faculties, courses, and departments.

Methods like _select_rooms, _select_timings, _select_faculties, etc., query the database and return objects like Room, Timing, Faculty, etc.

The DBMgr class acts as the central point for accessing all data required for timetable generation.

1. Project Overview
Objective: The project aims to generate an optimal timetable using a genetic algorithm. It ensures that constraints like room capacity, faculty availability, and department requirements are satisfied.

Technologies Used:

Flask: For the web interface and API handling.

SQLite: For database management.

Pandas: For processing Excel files.

Genetic Algorithm: For optimizing the timetable.

2. Constraints

The timetable must satisfy the following constraints:

Room Capacity:

The number of students in a class must not exceed the room's seating capacity.

Faculty Availability:

Faculty must be available at the scheduled time.

Room Booking:

Only one class can be scheduled in a room at a given time.

Faculty Booking:

A faculty member cannot teach two classes simultaneously.

Same Department at Same Time:

Two classes from the same department cannot be scheduled at the same time.

3. Initial Population

Population Size: Defined by POPULATION_SIZE (e.g., 10 schedules).

Creation:

Each schedule in the population is created randomly.

For each class in the schedule:

A random room is selected from the available rooms.

A random timing is selected from the available timings.

A random faculty is selected from the available faculties for the course.

This ensures diversity in the initial population.

4. API Functionality

The Flask app provides the following APIs:

Home Page (/):

Renders the index.html template.

About Page (/about):

Renders the about.html template.

Upload Page (/upload):

Accepts an Excel file containing data for rooms, timings, faculties, courses, etc.

Processes the file and inserts data into the SQLite database.

Calls the generate_timetable function to create the timetable.

Download Page (/download):

Allows the user to download the generated timetable (generated_schedule_output.xlsx).

5. Fitness Function

The fitness function evaluates the quality of a schedule based on the number of conflicts.

Steps to Calculate Fitness:
Initialize Conflicts:

An empty list (self._conflicts) is created to store conflicts.

Check Constraints:

Iterate through all classes and check for:

Room capacity conflicts.

Faculty availability conflicts.

Room booking conflicts.

Faculty booking conflicts.

Same department conflicts.

fitness function= 1/(fitness+1)

Calculate Fitness:

Fitness is calculated as:

Fitness
=
1
Number of Conflicts
+
1
Fitness=
Number of Conflicts+1
1

A higher fitness value indicates fewer conflicts.

Update Fitness:

The fitness value is stored in self._fitness and returned.

6. Selection of Population

Tournament Selection:

A subset of schedules (size = TOURNAMENT_SELECTION_SIZE) is randomly selected from the population.

The schedule with the highest fitness in this subset is chosen as a parent.

This process is repeated to select two parents for crossover.

7. Crossover Technique

Purpose: Combines two parent schedules to create a new child schedule.

Technique: Uniform Crossover.

Steps:

Two parent schedules are selected using tournament selection.

For each class in the child schedule:

A random number is generated.

If the number > 0.5, the class is taken from Parent 1.

Otherwise, the class is taken from Parent 2.

The child schedule is added to the new population.

8. Mutation Technique

Purpose: Introduces random changes to maintain diversity in the population.

Technique: Random Mutation.

Steps:

A schedule is selected from the population (excluding elite schedules).

For each class in the schedule:

A random number is generated.

If the number < MUTATION_RATE, the class is replaced with a new randomly generated class.

The mutated schedule is added to the new population.

9. Genetic Algorithm Workflow

Initialize Population:

Create an initial population of random schedules.

Evaluate Fitness:

Calculate the fitness of each schedule.

Sort Population:

Sort the population by fitness in descending order.

Evolve Population:

Perform crossover and mutation to create a new population.

Repeat for a specified number of generations or until a schedule with fitness = 1.0 is found.

Output Best Schedule:

Export the best schedule to an Excel file.

10. Key Points to Remember for Viva

Constraints:

Explain the constraints and why they are important.

Initial Population:

Describe how the initial population is created and why randomness is important.

Fitness Function:

Explain how fitness is calculated and its role in the genetic algorithm.

Selection:

Discuss tournament selection and its advantages.

Crossover:

Explain uniform crossover and how it creates diversity.

Mutation:

Describe random mutation and its role in avoiding local optima.

APIs:

Explain the purpose of each API and how they interact with the database and frontend.

Database:

Discuss how data is stored and retrieved from the SQLite database.

Excel Processing:

Explain how Pandas is used to read and process Excel files.

Output:

Describe how the final timetable is exported to Excel.

11. Sample Questions for Viva

What is the role of the fitness function in the genetic algorithm?

The fitness function evaluates the quality of a schedule by measuring the number of conflicts. It guides the algorithm toward better solutions.

Why is tournament selection used?

Tournament selection is computationally efficient and helps maintain diversity in the population.

How does mutation prevent the algorithm from getting stuck in local optima?

Mutation introduces random changes, allowing the algorithm to explore new solutions and avoid converging to suboptimal solutions.

What are the key constraints in the timetable generation problem?

Room capacity, faculty availability, room booking, faculty booking, and same department at the same time.

How is the initial population created?

The initial population is created by randomly assigning rooms, timings, and faculties to classes in each schedule.