Backend Documentation: Seat Allocation & PDF Generation System

Purpose

The backend system is designed to allocate exam seats for students based on department, year, roll numbers, room, and seating separation rules. It automatically:

1. Parses user-uploaded input (from a `.txt` file created by the frontend).
2. Fetches room details from a MySQL database.
3. Expands roll ranges into individual student roll numbers.
4. Allocates students to seats while maintaining required separation rules.
5. Exports the final seating plan as a PDF with clear layouts.

System Workflow

1. Input Handling (Flask `index` route)

- User uploads a `.txt` file (generated from frontend form).
- Flask saves the file in the `uploads/` directory.

Each line in the file represents one instruction with format:

```
<Dept>#<Year>$<Rolls>!<Date>@<Room>%<Separation>
```
Example:

```
ENGA#UG-1$1-50,55,60-65!25/08/2025@15%2
```

-
    - Dept = `ENGA`
    - Year = `UG-1`

    - Roll range = `1-50,55,60-65`
    - Date = `25/08/2025`
    - Room = `15`
    - Separation = `2`

---

2. Parsing Functions

`parse_line(line: str)`

- Splits an input line into meaningful fields.
- Returns: `(subject, year, roll_range, date, room, separation)`

`expand_rolls(roll_text: str)`

- Converts roll ranges into a list of individual roll numbers.
- Example: `"1-3,5"` → `[1, 2, 3, 5]`

---

3. Database Interaction

`get_room_info(room_id)`

- Connects to the MySQL database `ExamSeatAllowtment`.
- Reads table `RoomInfo` and fetches:
    - `RoomId`
    - `TotalCapacity`
    - `BenchPerCol` (comma-separated format, e.g., `"4,5,6"`)
- Constructs a seat matrix:

    - Two benches per column (left + right side).
    - Filled with `"e"` representing an empty seat.

Returns: `seat_matrix` (list of seat columns).

---

4. Seat Allocation

`can_place(seat_matrix, c, r, dept, separation)`

- Checks if a student can sit at `(col=c, row=r)`:
    - Ensures no nearby student from the same department is within the given `separation` distance.

`allocate_seats(seat_matrix, rolls, dept, year, separation)`

- Iterates through the seat matrix column by column.
- Places students (roll numbers) while respecting separation rules.
- Replaces `"e"` with `(roll, dept, year)` tuple.

---

5. PDF Export

`rotate_for_pdf(seat_matrix)`

- Converts column-based structure into row-based (for ReportLab Table).

`export_pdf(pdf_path, totalRooms)`

- Builds a PDF file using ReportLab:
    - Adds header (College Name, Date, Room).
    - Creates a seating Table with:
        - Fixed seat width.

        - Fixed gutter width after every 2 columns.
        - Row height.
    - Borders around occupied or empty seats.
    - Empty cells are shown with consistent size for alignment.
- Saves final output in `output/All_Seating_Allotments.pdf`.

Control Flow Summary

1. User uploads `input.txt` (via frontend form).
2. Flask `index()` saves and reads the file.
3. For each line:
    - `parse_line()` extracts details.
    - `expand_rolls()` expands roll ranges.
    - `get_room_info()` fetches seating matrix.
    - `allocate_seats()` places students.
    - Stores results in `totalRooms`.
4. After processing all inputs:
    - `export_pdf()` generates one consolidated PDF.
5. Returns `pdf-viewer.html` to show available PDF downloads.

Output

- Generates PDF seating charts in `/output/All_Seating_Allotments.pdf`.

- Students are properly arranged in rooms with separation rules enforced.
- Admin can download the PDF directly via `/download/<filename>`.

In summary:
The backend takes input instructions, calculates student seating, and produces a professional seating plan PDF — automating a manual and error-prone process.