# CS 301 - Introduction to Database Systems
# Course Project Specification
# Railway Reservation System
# Weightage: 20% 2022 -- 2023 Semester I

**General Instructions:**

- **Allowed team size: 1,** 2 or 3 members. Projects from teams of size 3 would be more complicated for the same score and same deadline. Teams of size 1 and size 2 would be doing the same amount of work for the same deadline.
- Students should use the given idea and develop it into a full-scale project proposal. We are just giving a just high level idea of the project and by no means a full specification.
- You are allowed C/ C++, JAVA, PHP and Python for these projects. For any other language, you should first get an approval.
- Periodic discussions are mandatory to give a proper direction (and guidance as needed) to the projects.
- Do not disappear after taking a topic and show up only in the week before the final deadline. In such cases, instructors would simply proceed with their discretion on what should have been done!

**Important Notes:**

- SQL tutorial: https://www.w3schools.com/sql/
- Refer material on database triggers, database authorization, functions and procedures.
- Web applications development in Java, Python
- Transaction support in PostgreSQL https://www.postgresql.org/docs/current/mvcc.html
- Serializable isolation level in PostgreSQL
  https://www.postgresql.org/docs/current/transaction-iso.html#XACT-SERIALIZABLE

**Project Description**

The goal is to develop a railway reservation system. For this project, you are expected to develop code for the back-end (Database part + code in high level language e.g., Java for accessing the database) of the system. Given that this course focuses on topics related to database systems, we would expect a more thorough effort on the database part (schema, stored procedures and triggers) of the project. Following concepts need to be incorporated into the system. For the purpose of this course project, you build this application as a multi-user system. In other words, simultaneous access to the booking function by two or more agents is very much possible. Also assume that each ticket is booked for the entire journey of the train. In other words, for this part of the course project, there is no need to store the station information of trains. Furthermore, assume that all trains run on all days.

**Regarding Trains:**

1. A train is identified by a unique train-number.
2. A train has few AC coaches and several Sleeper class coaches.
3. Different trains have different numbers of AC and Sleeper coaches.
4. Even for the same train, the number of AC Coaches and Sleeper coaches can be different across different dates.
5. The Admin decides the number of AC coaches and Sleeper coaches in a train for a specific date. After these are decided, the train is released (for that date) into the "booking system" for ticket booking.
6. The number of berths and type of berths (lower, middle, upper, etc) are fixed in an AC coach and a Sleeper coach. Refer Appendix A for details on number berth numbers (and their types) in AC and Sleeper coaches.

**Railway Ticket Reservation:**

1. A ticket makes a booking for its passengers on a particular train-number for a particular date. Railway ticket booking is allowed only if the particular train of interest is released into the "booking system" for the date of interest by the Admin.
2. A ticket has a unique number called its PNR number.
3. Assume that each ticket books berth(s) for the entire journey of the train.
4. A single ticket can have multiple passengers. Each passenger is associated with Name, Age, and Gender attributes. In a ticket, each of the passengers is assigned a berth. Also, the ticket would mention if it is a lower-berth, upper-berth or middle berth.
5. The booking function takes the following input: (a) number of passengers; (b) name of all passengers; (c) train number; (d) date of journey; (e) choice of AC coach or Sleeper (this choice is the same for all passengers). The ticket is booked only if all passengers are able to get a reservation on their choice of date, otherwise an error is thrown.
6. The journey ticket consists of the following: (a) PNR (auto generated by application) (b) name all of the passengers, (c) Coach number, berth number and type (Lower, Middle, Upper, etc) of each passenger; (d) train number and (e) date of journey.

**Key Evaluation Aspects:**

- Your schema should have **least possible redundancy of data and attributes must be stored in appropriate entities.** You must clearly state any assumptions made while designing the schema. Reasonable ones would be accepted and graded.

- Evaluation would be based on throughput. During the evaluation, we would give you a series of ticket booking requests (around 1000--2000). You are required to process these requests in a concurrent fashion (using multi-threading libraries available in JAVA, Python etc.) on a system containing 50 cores (or more). The TAs would note the total time spent in processing these requests.

- Your application should have relevant stored procedures and triggers to maintain/implement the required business logic constraints. For example, there should be appropriate stored procedures for releasing a train into the booking system (for a particular date), booking a ticket etc. Checks should be made such that booking is not done before the train is released into the "booking system." Checks should be made to ensure that different tickets do not book the ***same berth*** in the same train on the same journey date. Similarly, your system should not book more tickets than the number of seats available in a coach (and train) for the particular choice of journey date.

**Additional Work for Teams of Size 3 (**PLEASE CONTACT INSTRUCTOR BEFORE STARTING THIS PART)**

Implement a search procedure for searching trains between any two stations. Note that this part is completely independent (from the perspective of evaluation) of the specifications mentioned in the previous section. In other words, teams of size-3 need not worry about station names in the ticket booking part. Following things need to be considered for implementing the search procedure.

1. You would have to store the stations (and their corresponding arrival and departure times) as served by a train in your database. Also you would have to store the order of stations seen by the train in its route.
2. You can assume that at-most one break is needed to reach from anywhere to anywhere. If no path is found, then your application should return “journey plan is not possible.”
3. While recommending connections, you need to check the compatibility of the arrival and departure times of the trains involved in the connection.

## APPENDIX A

**AC Coach Composition**

| Berth Number | Type |
|---|---|
| 1 | LB |
| 2 | LB |
| 3 | UB |
| 4 | UB |
| 5 | SL |
| 6 | SU |
| 7 | LB |
| 8 | LB |
| 9 | UB |
| 10 | UB |
| 11 | SL |
| 12 | SU |
| 13 | LB |
| 14 | LB |
| 15 | UB |
| 16 | UB |
| 17 | SL |
| 18 | SU |

**Sleeper Coach Composition**

| Berth Number | Type |
|---|---|
| 1 | LB |
| 2 | MB |
| 3 | UB |
| 4 | LB |
| 5 | MB |
| 6 | UB |
| 7 | SL |
| 8 | SU |
| 9 | LB |
| 10 | MB |
| 11 | UB |
| 12 | LB |
| 13 | MB |
| 14 | UB |
| 15 | SL |
| 16 | SU |
| 17 | LB |
| 18 | MB |
| 19 | UB |
| 20 | LB |
| 21 | MB |
| 22 | UB |
| 23 | SL |
| 24 | SU |