Railway Berth Booking System

- User Registration And Authentication
- User-Friendly Text Interface
- User Login from Terminals of different machines
- Multithreaded Platform to Enhance Scalability
- Well Documented Program for Easy Modification

Given: [1] ReserveSys (Railway/Hotel/Bus) platform

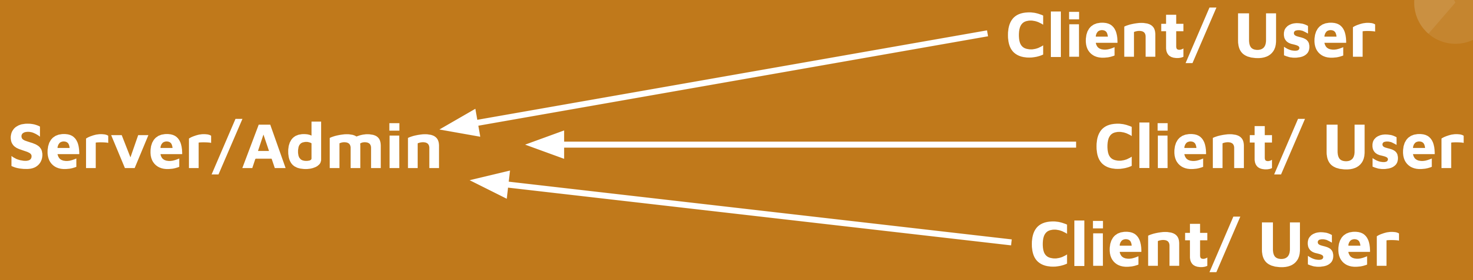
Develop a reservation platform, (hotel rooms/bus seats/railway berths booking system), where rooms/seats are offered based on user requests

Probable Features of the system:

- Define reservation policy
- Display various deals and packages
- Create a dynamic pricing reservation system
- Seats/Rooms allocation policies (e.g. aged persons will be allocated front rows/lower floors)
- Text-based user interface & availability matrix visualization, report & ticket generation

The system satisfy the following requirements:

- Enables user login from different terminals from different physical machines
- Enables User registration and authentication (using some hash based password)
- A user-friendly text interface
- A multithreaded platform to enable scalability
- Provide a well documented header file with set of well defined APIs/function interface so that any other reservation scenario may be implemented with minor modification

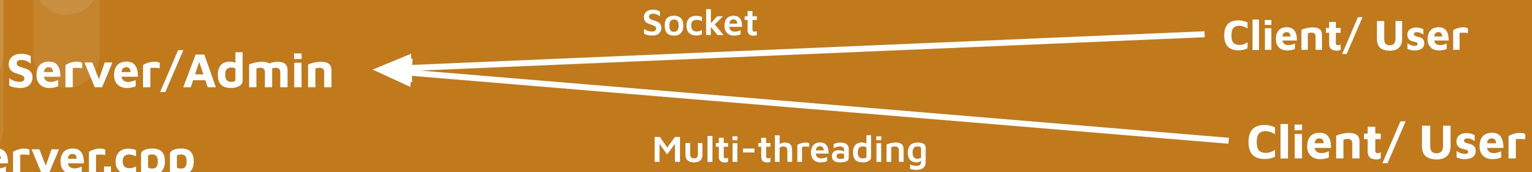


Server Configuration:

- User Management:
  Create, Authenticate, Delete, Update
- Train Management:
  Add, Remove, Display
- Reservation Management:
  Book, Cancel, Delete, Maintain seat matrix, Maintain waiting list,

Client Configuration:

- Account Management:
  Register, Login, Update, Delete
- Ticket Management:
  Reserve, Cancel, Check Schedule, Check ticket status, Check allotted seat



Socket

Server/Admin

Client/ User

Client/ User

Multi-threading

<u>File: server.cpp</u>

Server: (Objects)

✷ User: (Functions)

Create(Mobile, Password, age),
Authenticate(Mobile, Password),
Delete(Mobile, Password)

✷ Train: (Functions)

Add(Train-no, Source, Destination),
Remove(Train-no, Source, Destination),
Display() -Return: trn-no, src,dest, st-time, end-time, price[dynamic per booked seats]

✷ Reservation: (Functions)[++Policy-Later]

(ticket.txt)Book-Ticket(user,train),
(cancel.txt)Cancel-Ticket(user,train), [update empty seats, refund]
Show-Coach(train)

✷ Ticket: mobileTicketID.txt [format]

Date-time, Train-no, src, dest, price, berth, name, age, mobile, TicketID=date-train-time

File: client.cpp

Client: (Objects)`

✷ Account: (Functions)

Register(Mobile, Password,age),
Login(Mobile, Password),
Delete-Account(Mobile, Password)

✷ Ticket: (Functions)

Reserve(Mobile, Password,trn-no, src, dest), Cancel(Mobile, Password,ticket-id),
Status(Mobile, Password,ticket-id),
Enquire-Trains(trn-no)

Socket:

Create socket, bind and listen server, user sending request, accept requests(divide across threads), close client, local ip address and routers

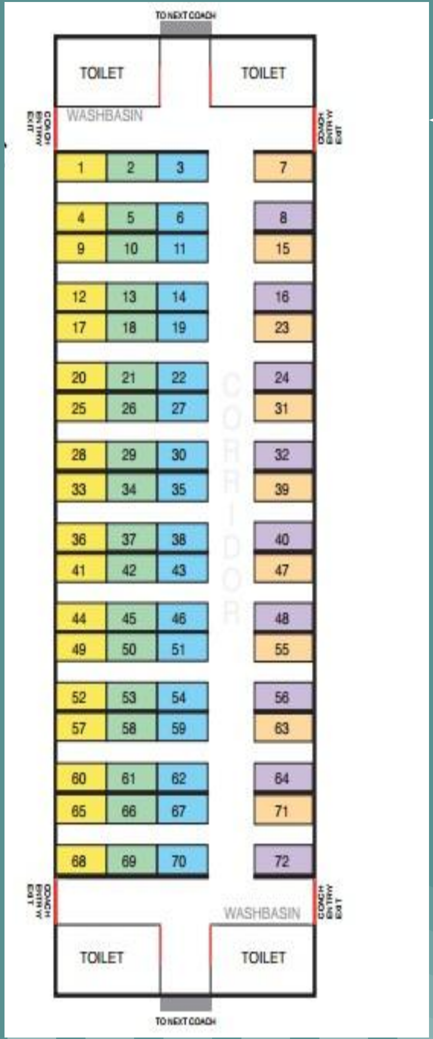
Server Side Objects:

1. User
   a. Data Members: Username, Mobile No. Password, Bookings, Age
   b. Member functions: Register, Authenticate, Display profile, Update, Reserve Ticket
2. Train
   a. Data Members: Train No., Coach List, Seat Matrix, Stations, Schedule, Waiting queue
   b. Member Functions: Add train, Change schedule, Display Information, Update Price, Remove Train
3. Reservation
   a. Data Members: TransactionID, Date, Train No., Username, Seat Allotted, Reservation Status,
   b. Member Functions: Book Ticket, Cancel Ticket, Update reservation status

Client side objects

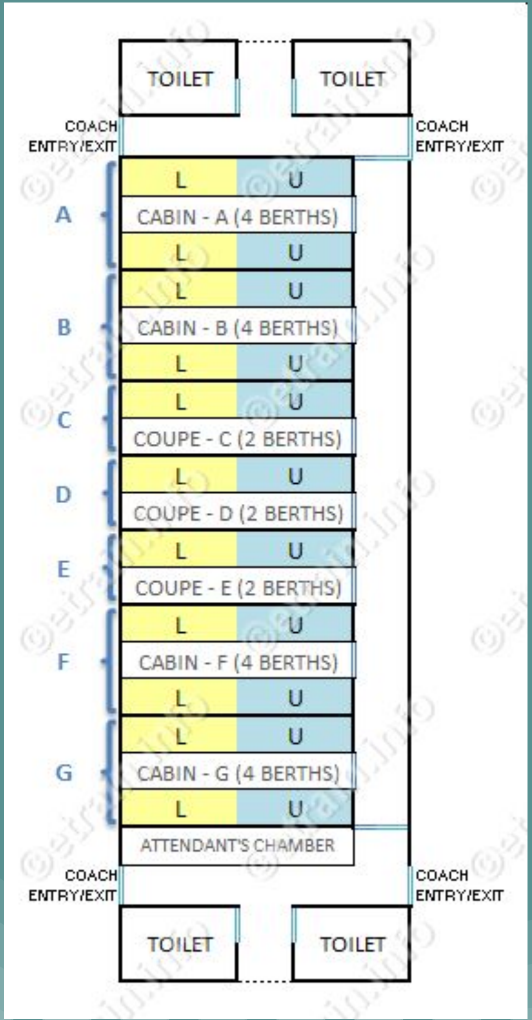
1. User
    a. Data Members-

Berth Design



Sleeper, Chair, 2-AC

1-AC



Reservation Policy:

1. Ticket can be booked before running. Tricket while running will be 30% over priced.
2. Cancellation will not be entertained after train departure.
3. Simultaneous booking of 3 ticket will be 10% off and of 5 ticket will be 20% off.
4. 10% increase in price if 80% seats are full.
5. User Seat preference will be considered first. Age >= 60 will be allotted seat <=23 and >=56 and =Lower if available.