RAILWAY RESERVATION SYSTEM

Team – 9

PROBLEM STATEMENT

DEVELOP A RESERVATİON PLATFORM, (HOTEL ROOMS/BUS SEATS/RAİLWAY BERTHS BOOKİNG SYSTEM), WHERE ROOMS/SEATS ARE OFFERED BASED ON USER REQUESTS

PROBABLE FEATURES OF THE SYSTEM:

- DEFİNE RESERVATİON POLİCY
- DİSPLAY VARİOUS DEALS AND PACKAGES
- CREATE A DYNAMİC PRİCİNG RESERVATİON SYSTEM
- SEATS/ROOMS ALLOCATİON POLİCİES (EG. AGED PERSONS WİLL BE ALLOCATED FRONT ROWS/LOWER COACHES)
- TEXT-BASED USER İNTERFACE & AVAİLABİLİTY MATRİX VİSUALİZATİON, REPORT & TİCKET GENERATİON

CONCEPTS REQUİRED

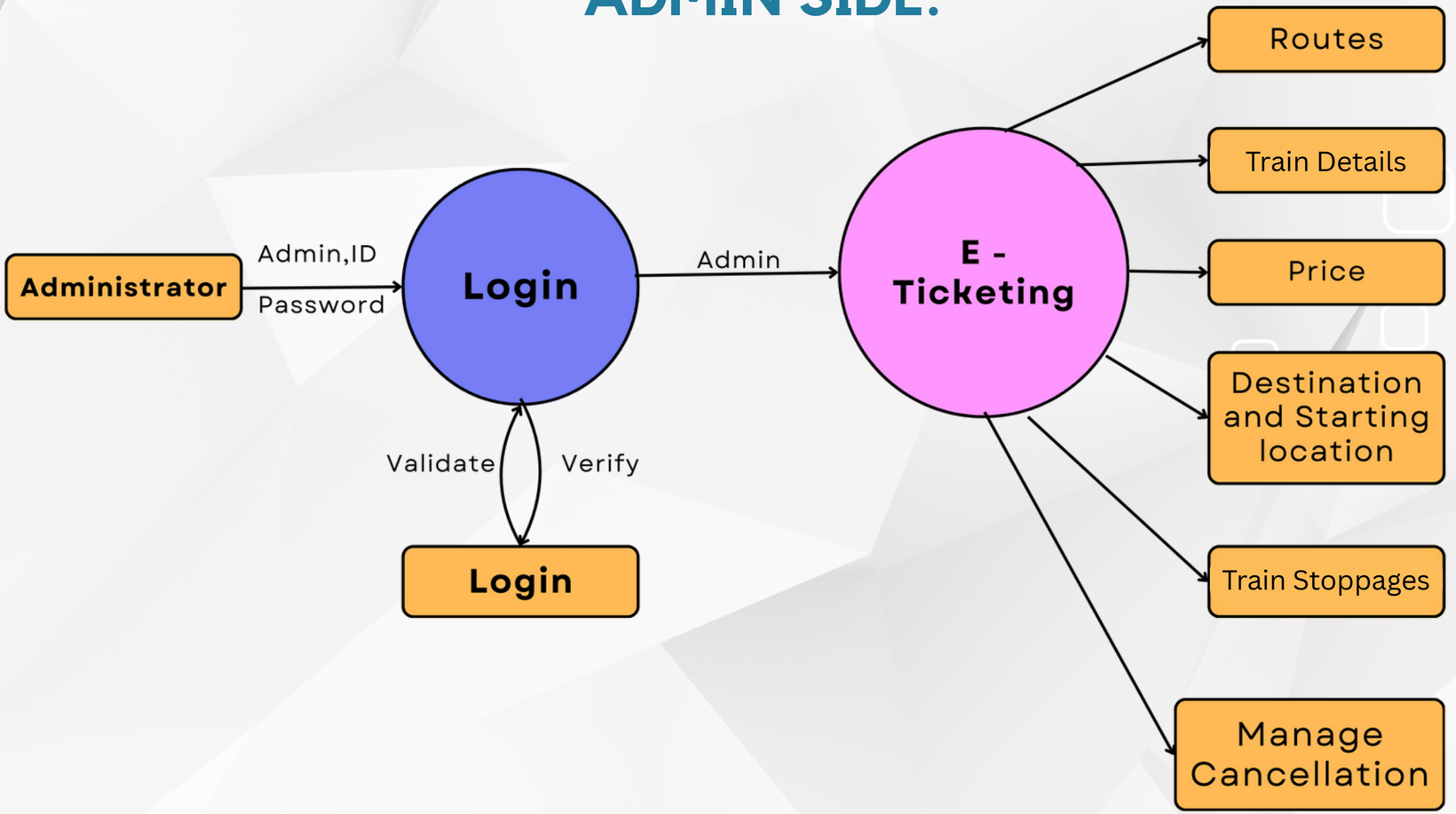
- **Multithreading:** Allows multiple user requests to be handled concurrently for improved performance and responsiveness.
- **Sockets and Networking**: Facilitates communication between client applications and the server for booking requests and data retrieval.
- **Database:** Stores and manages all data related to trains, seats, users, and reservations, ensuring transactional consistency.
- **Transactions:** Ensures that booking operations are atomic, completing fully or rolling back in case of failure.

RESERVATION POLICY

- **Priority Allocation:** Elderly (Age ≥ 60) get lower coaches.
- **Cancellation Rules:** Allowed before departure; no cancellation post-departure.
- **Dynamic Pricing:** +10% price increase when 80% seats are full. Tickets booked on boarding point are 30% more expensive.
- **Group Discounts:**10% off for 3 seats, 20% off for 5 or more.
- **Waitlisting:** Waitlisted seats get updated/promoted as seats become available.

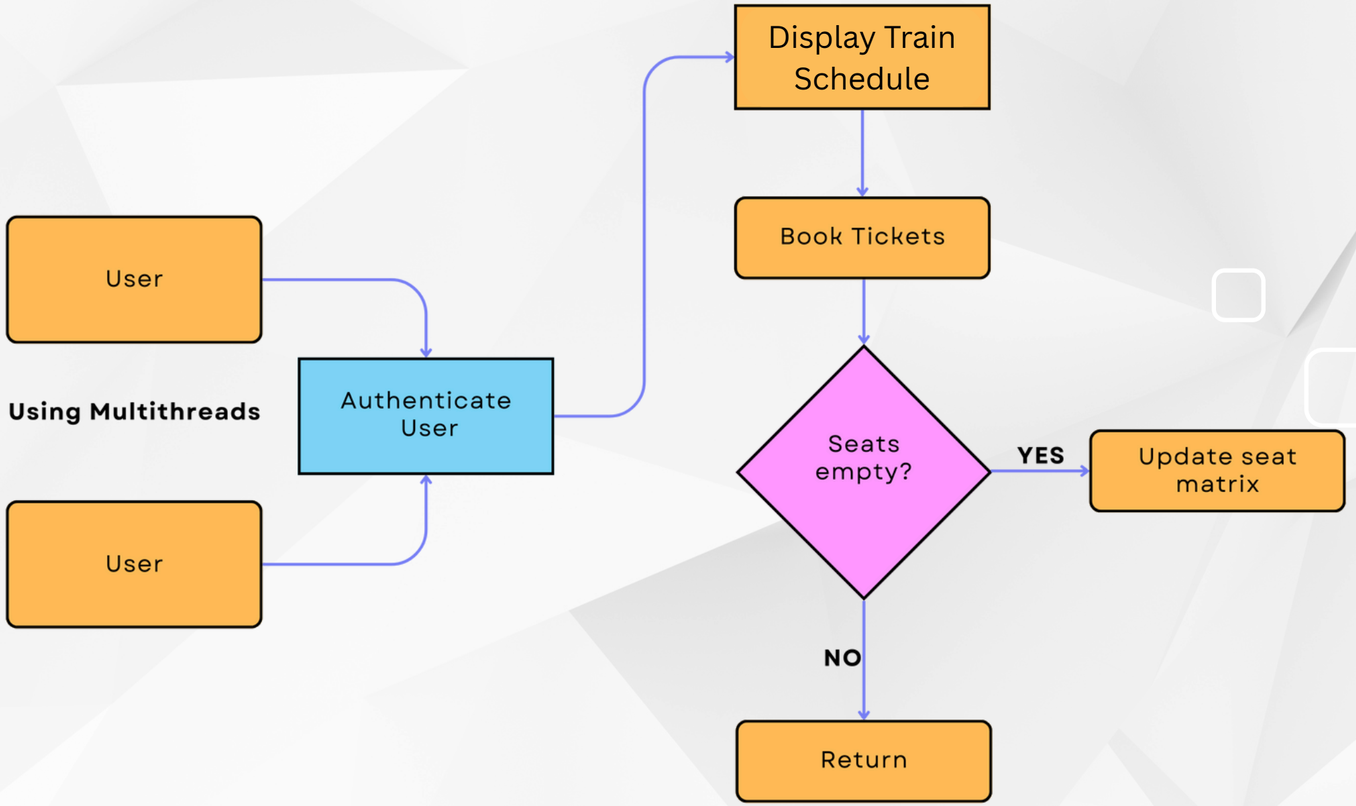
SCHEMATIC OVERVIEW

ADMİN SIDE:

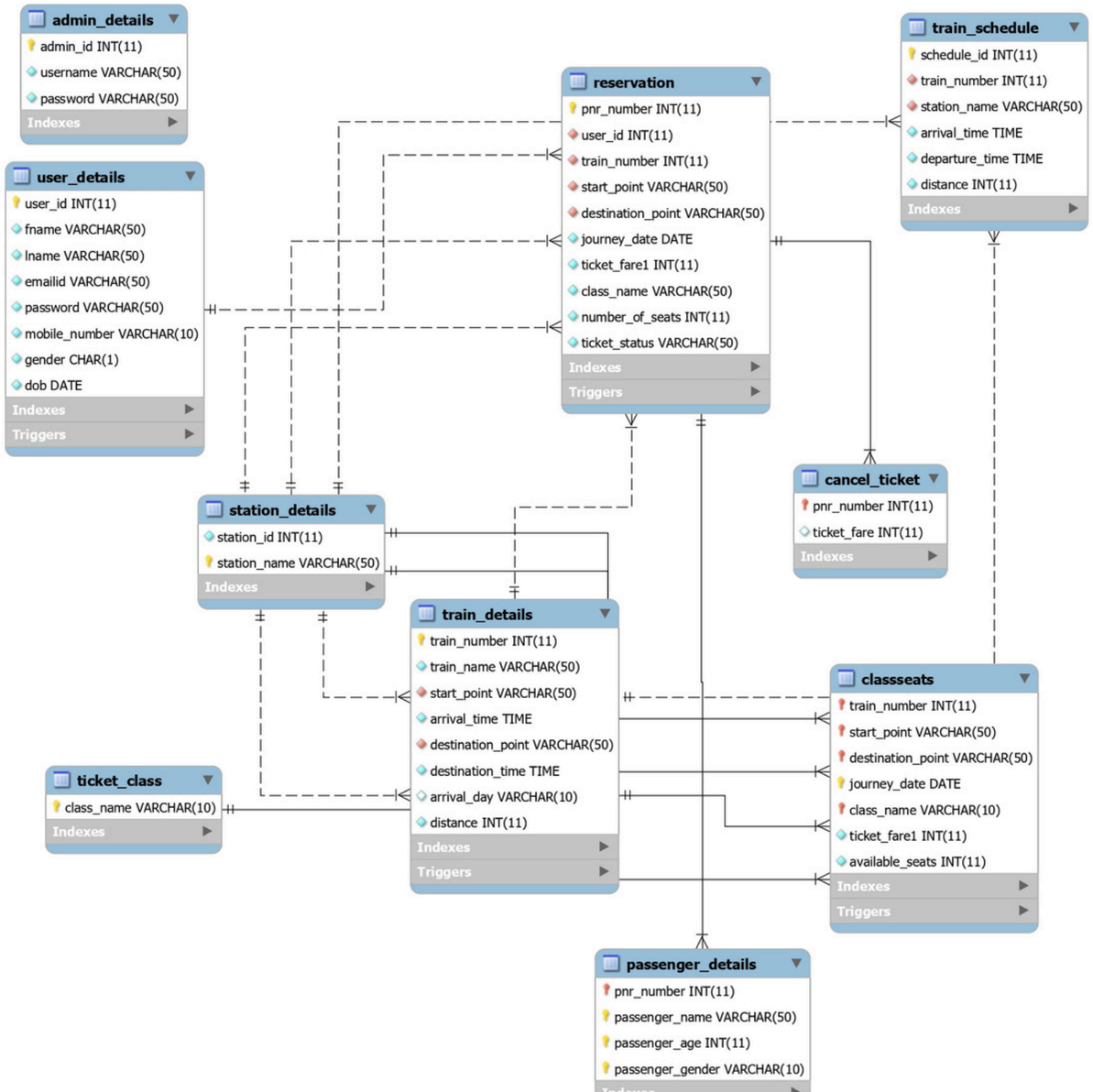


SCHEMATIC OVERVIEW

USER SIDE:



ER Diagrammatic representation



User Interface

Users easily view within the terminal's interactive seat matrix, streamlining the booking process. Once the payment is completed user receive a generated Ticket, all within the intuitive terminal interface for an efficient train booking experience."

```
TRAIN DETAILS
-------------------------------------------------
Train Number      : 56789
Train Name        : Express
Source            : AAA
Destination       : EEE
Price             : $50
Available Seats   : 72

Seat Matrix :
-------------------------------------------------
[ B1 ][  1 ][  2 ]   [ B2 ][  3 ][  4 ]   [ B3 ][  5 ][  6 ]
[    ][  7 ][  8 ]   [    ][  9 ][ 10 ]   [    ][ 11 ][ 12 ]
-------------------------------------------------
[ C1 ][ 13 ][ 14 ]   [ C2 ][ 15 ][ 16 ]   [ C3 ][ 17 ][ 18 ]
[    ][ 19 ][ 20 ]   [    ][ 21 ][ 22 ]   [    ][ 23 ][ 24 ]
-------------------------------------------------
[ D1 ][ 25 ][ 26 ]   [ D2 ][ 27 ][ 28 ]   [ D3 ][ 29 ][ 30 ]
[    ][ 31 ][ 32 ]   [    ][ 33 ][ 34 ]   [    ][ 35 ][ 36 ]
-------------------------------------------------
[ E1 ][ 37 ][ 38 ]   [ E2 ][ 39 ][ 40 ]   [ E3 ][ 41 ][ 42 ]
[    ][ 43 ][ 44 ]   [    ][ 45 ][ 46 ]   [    ][ 47 ][ 48 ]
-------------------------------------------------
[ F1 ][ 49 ][ 50 ]   [ F2 ][ 51 ][ 52 ]   [ F3 ][ 53 ][ 54 ]
[    ][ 55 ][ 56 ]   [    ][ 57 ][ 58 ]   [    ][ 59 ][ 60 ]
-------------------------------------------------
[ G1 ][ 61 ][ 62 ]   [ G2 ][ 63 ][ 64 ]   [ G3 ][ 65 ][ 66 ]
[    ][ 67 ][ 68 ]   [    ][ 69 ][ 70 ]   [    ][ 71 ][ 72 ]
-------------------------------------------------

Note: 'B', 'C', 'D', 'E', 'F', 'G' denote coach/bogey numbers.
      Numbers within brackets denote seat numbers.
```

```
Train Ticket Details
=====================================================
Train Number                    : 56789
Train Name                      : Express
Boarding Station                : AAA
Destination Station               : EEE
Departure from Boarding         : 09:15 AM
Arrival Time to Destination     : 07:30 PM
Date of Journey                 : 2025-05-10
=====================================================

Passenger Details:
=====================================================
Passenger Name(s)               : Priya Sharma, Rahul Verma
Age                             : 28, 42
Seat Number                     : C2/15, C2/16
=====================================================

Ticket Details:
=====================================================
Ticket Price                    : $85.00
Transaction ID                  : TRN987654321
Booking ID                      : TRB987654321
=====================================================
```

CODE SNIPPETS - Client side

```cpp
// Updated: Removed coach index selection from client side; server decides allocation logic
#include <iostream>
#include <string>
#include <arpa/inet.h>
#include <unistd.h>
#include <cstring>
#include <vector>
#include <sstream>
#include <utility>

using namespace std;

class User {
private: ···
public:
    User(int id, string fn, string ln, string em, string mod, char g, string dt) ···
    int getId() const { return user_id; }
    string getFullName() const { return fname + " " + lname; }
};

struct TicketDetails { ···
};

vector<string> split(const string &s, char delimiter) { ···
}

// CoachInfo and TrainInfo definitions remain unchanged
struct CoachInfo { ···
};
```

```cpp
User* handleLoginResponse(string& response) { ···
}

void registrationFlow() { ···
}

void loginFlow() { ···
}

int main() {
    while (true) {
        cout << "\n===== Railway Reservation System =====" << endl
             << "1. Register" << endl
             << "2. Login" << endl
             << "3. Exit" << endl
             << "Enter choice: ";
        int choice; cin >> choice;
        switch (choice) {
            case 1: registrationFlow(); break;
            case 2: loginFlow();        break;
            case 3: return 0;
            default: cout << "Invalid choice!" << endl;
        }
    }
}
```

CODE SNIPPETS - Server side

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <pthread.h>
#include <semaphore.h>
#include <fcntl.h>
#include <mysql/mysql.h>

#define MAX_BUFFER 8192
#define PORT 5566
#define MAX_PASSENGERS 10
#define MAX_NAME_LEN 64

MYSQL *conn;

// Initialize MySQL connection
void initializeDatabase() {
    conn = mysql_init(NULL);
    if (!mysql_real_connect(conn, "localhost", "mysqldb", "Shivam.1257",
                            "Traintest", 3306, NULL, 0)) {···
        sendResponse(sock, buf);···
        sendResponse(sock, buf);
    } else {
        sendResponse(sock, "ERROR|Invalid credentials\n");
    }
    mysql_free_result(res);
}
```

```c
int main() {
    initializeDatabase();
    int server_sock = socket(AF_INET, SOCK_STREAM, 0);
    struct sockaddr_in server_addr = {0};
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = INADDR_ANY;
    bind(server_sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
    listen(server_sock, 10);
    printf("[+] Server listening on port %d\n", PORT);
    while (1) {
        struct sockaddr_in client_addr;
        socklen_t addr_size = sizeof(client_addr);
        int client_sock = accept(server_sock, (struct sockaddr*)&client_addr, &addr_size);
        printf("[+] Client connected\n");
        pthread_t tid;
        pthread_create(&tid, NULL, clientHandler, (void*)(intptr_t)client_sock);
        pthread_detach(tid);
    }
    mysql_close(conn);
    return 0;
}
```

🧩 Client-Side Modular Design

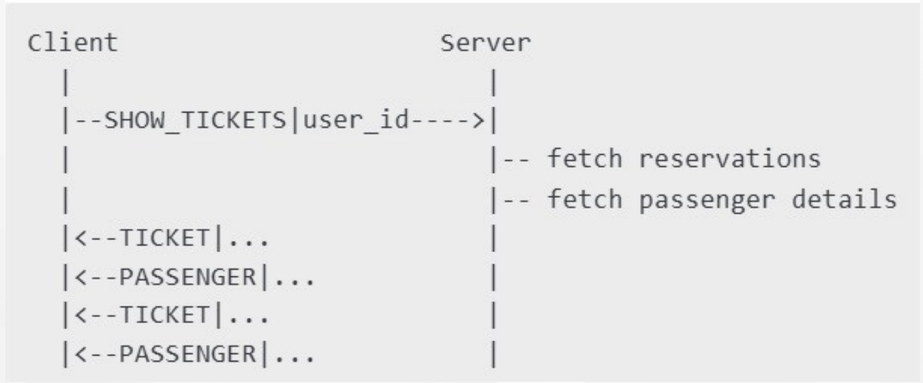
Module Name	Responsibilities	Functions Handling This Module
`User Authentication`	Handles user login and registration. Sends/receives data from server.	`handleLogin()`, `handleRegistration()`
`Booking Management`	Gathers booking details, passenger info, sends booking requests.	`handleBookTicket()`
`Payment Interface`	Displays payment request, handles 2-minute payment timeout and confirmation.	`handleBookTicket()`
`Ticket Management`	Parses and displays booked/cancelled ticket data.	`handleShowTickets()`, `parseTicketDetails()`
`Communication Layer`	TCP socket client, encapsulates client-server communication.	`sendRequestToServer()`
`UI Layer`	Provides interactive CLI for the user (menus, prompts, messages).	`displayMenu()`, `showAvailableTrains()`

🧱 Server-Side Modular Design

Module Name	Responsibilities	Functions Handling This Module
`Request Dispatcher`	Parses incoming client requests and routes to appropriate handlers.	`clientHandler()`
`Booking Handler`	Locks available seats, computes fare, assigns seats, manages semaphores.	`processBooking()`
`Payment Handler`	Records payment status in DB, matches after timeout for ticket confirmation.	`handlePaymentStatus()`
`Ticket Generator`	Builds and sends final ticket response upon successful booking and payment.	`processBooking()`
`Seat Map Manager`	Fetches and formats seat availability per class and coach.	`handleSeatMap()`
`Database Layer`	Centralized SQL logic using MySQL C API (`libmysqlclient`).	`initializeDatabase()` , direct SQL execution
`Thread Manager`	Spawns detached threads to handle individual booking and async client actions.	`pthread_create()` in `clientHandler()`

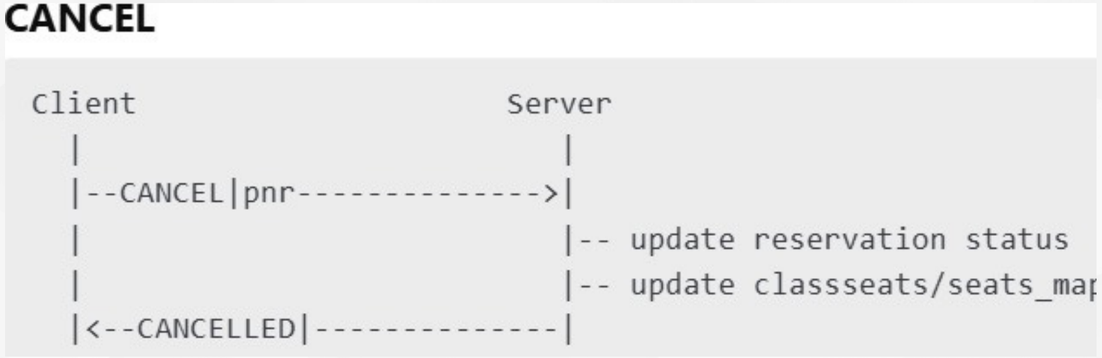
Request Process Flow

SHOW_TICKETS

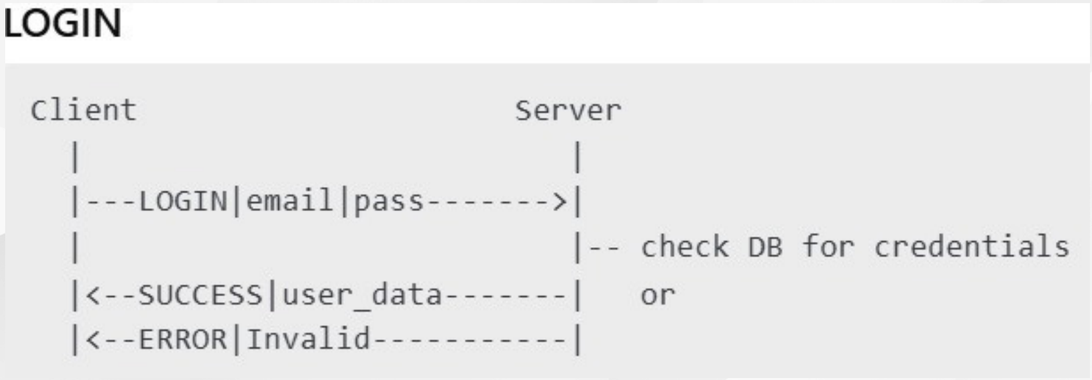


GET_SEATMAP

CANCEL



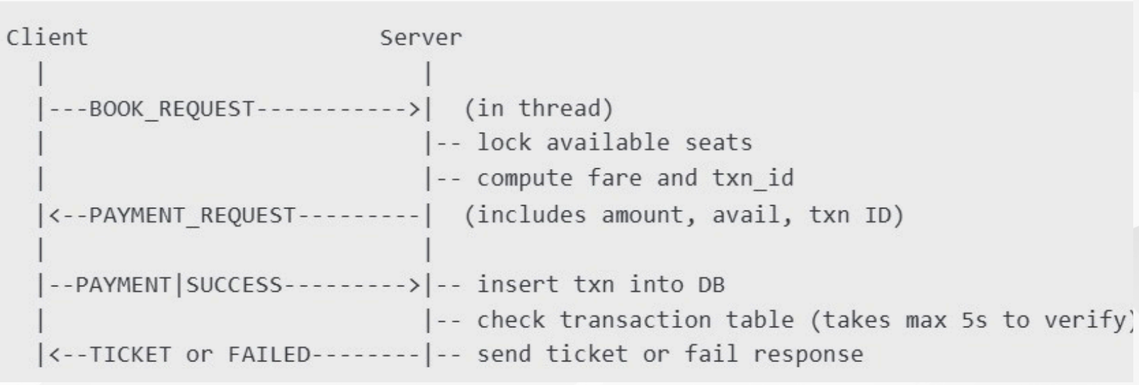
LOGIN



Request Process Flow

LIST_TRAINS

BOOK_REQUEST + PAYMENT + TICKET



REGISTER

TEAM MEMBERS

23CS8181	Arpit Indresh Yadav
23CS8182	Tatsat Kumar
23CS8183	Appakondu ChandraSekhar Redy
23CS8184	Prity Kumari
23CS8185	Pediredla HarshaVardhan Naidu
23CS8186	Ritam Koley
23CS8187	Krish Mahaseth
23CS8188	Shivam Vishwakarma
23CS8189	Nidhi Kumari
23CS8190	Mandala Bharadwaj
23CS8191	Isha Priya Singh

Contribution

System Design : 23CS8182 , 23CS8187

Project Planning : All the team members

Resource management : 23CS8189 , 23CS8184, 23CS8191

Coding : 23CS8186 , 23CS8188

Ppt Design : , 23CS8190, 23CS8183

UI Interfacing : 23CS8185, 23CS8181

THANK YOU