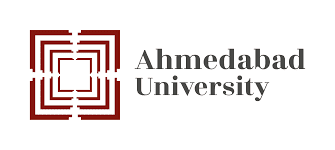
**P**

**CSE250 | DATABASE MANAGEMENT SYSTEM**

**Section 2**

Final Project

**Railway Ticket Booking and Management System**

Submitted to

**Prof. Shefali Naik**

By

Group Members

|  |  |
| --- | --- |
| Name | Enrollment Number |
| Anvi Sheth | AU22L20001 |
| Muskan Chauhan | AU22L20003 |

**Introduction**

Railway Booking and Management System is an advanced web-based program, designed to supervise the complexities of railway operations and resource management. As a central hub, RBMS enables the smooth execution of vital functions like keeping an eye on train timetables, organizing reservations for tickets, keeping track of passenger data, and supervising related operations. RBMS facilitates the ability of government agencies and travellers to improve customer happiness, increase productivity, and streamline railway operations through an intuitive interface.

Accessible from any web-enabled device at any time, RBMS functions as a full solution. The user-friendly design facilitates efficient online planning and booking by providing customers with easy access to essential information such as train schedules, seat availability, and pricing data.

RBMS is a revolutionary technology that benefits both railway firms and passengers by focusing efficiency and ease in the railway management process .As a solid framework that streamlines the intricacies of railway operations, RBMS essentially signifies a paradigm leap in railway management systems.

**Requirement Specification**

1. Reporting and Analytics: Train timetables, passenger volume, and other pertinent indicators ought to be able to be reported on by the system. Real-time data monitoring, processing, and visualization should be possible with the system.

2. Compatibility: The system needs to work with the railway system's current hardware and software infrastructure. The system ought to be built with support for many browsers and devices, be platform-neutral, and offer an API for system integration.

3. Security: To safeguard user data and stop illegal access, the system needs to have strong security mechanisms in place. Secure communication methods, user authentication, and access control should all be included in the system

4. Usability: Everyone who uses it, including those with little technological experience, should be able to easily operate and navigate the system. The system ought to support numerous languages, have an easy-to-use user interface, and provide clear instructions.

5. User Management: A variety of users, such as passengers and railroad employees, should be able to be managed by the system. The user name, password, and user type should all be able to be added to and modified in user records within the system.

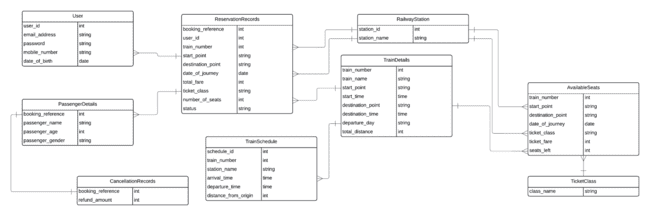
6. Train Management: The system must be able to control the travel routes, timetables, and seating configurations of various train types. The train's number, name, type, departure and arrival stations, and distance should all be able to be added to and modified via the system.

7. Schedule Management: The railway network's different train schedules should be able to be managed by the system. The creation and editing of schedule entries, which include the train number, departure and arrival times, and fare, should be supported by the system.

8. Ticket Management: Bookings and cancellations of tickets for various trains and timetables should be handled by the system. The generation and editing of ticket records, which include the fare, seat number, schedule number, train number, and passenger name, should be possible through the system.

9. Station Management: The railway network's many stations, together with their names, codes, and positions, should be managed by the system. Station records, such as the name, code, and location of the station, should be able to be created and modified via the system.

**ER-DIAGRAM**

****

**Database Design**

**1.) Table- Cancellation**

**CODE:**

|  |
| --- |
| **CREATE** **TABLE** **IF** **NOT** **EXISTS** `cancellation\_records` (  `booking\_reference` int(11) **NOT** **NULL**,  `refund\_amount` int(11) **DEFAULT** '0',  PRIMARY **KEY** (`booking\_reference`) ) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=latin1; |

|  |
| --- |
| +-------------------+--------------+------+-----+---------+-------+ | Field | Type | **Null** | Key | Default | Extra | +-------------------+--------------+------+-----+---------+-------+ | booking\_reference | int(11) | **NO** | PRI | **NULL** | | | refund\_amount | int(11) | **YES** | | 0 | | +-------------------+--------------+------+-----+---------+-------+ |

**2.) Table- Ticket Class**

**CODE:**

|  |
| --- |
| CREATE TABLE **IF** **NOT** EXISTS `ticket\_class` (  `class\_name` varchar(10) **NOT** **NULL**,  PRIMARY KEY (`class\_name`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1; |

|  |
| --- |
| +------------+-------------+------+-----+---------+-------+ | Field | Type | **Null** | Key | Default | Extra | +------------+-------------+------+-----+---------+-------+ | class\_name | varchar(10) | **NO** | PRI | **NULL** | | +------------+-------------+------+-----+---------+-------+ |

**3.) Table- Available Seats**

**CODE:**

|  |
| --- |
| **CREATE** TABLE **IF** **NOT** **EXISTS** `available\_seats` (  `train\_number` int(11) **NOT** **NULL**,  `start\_point` varchar(50) **NOT** **NULL** COMMENT 'Starting\_Point',  `destination\_point` varchar(50) **NOT** **NULL** COMMENT 'Destination\_Point',  `date\_of\_journey` date **NOT** **NULL**,  `ticket\_class` varchar(10) **NOT** **NULL**,  `ticket\_fare` int(11) **NOT** **NULL**,  `seats\_left` int(11) **NOT** **NULL**,  **PRIMARY** **KEY** (`train\_number`,`start\_point`,`destination\_point`,`date\_of\_journey`,`ticket\_class`),  **KEY** `ticket\_class` (`ticket\_class`),  **KEY** `start\_point` (`start\_point`,`destination\_point`),  **KEY** `destination\_point` (`destination\_point`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1; |

|  |
| --- |
| +-------------------+--------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------------------+--------------+------+-----+---------+-------+ | train\_number | int(11) | NO | MUL | NULL | | | start\_point | varchar(50) | NO | | NULL | | | destination\_point | varchar(50) | NO | | NULL | | | date\_of\_journey | date | NO | | NULL | | | ticket\_class | varchar(10) | NO | MUL | NULL | | | ticket\_fare | int(11) | NO | | NULL | | | seats\_left | int(11) | NO | | NULL | | +-------------------+--------------+------+-----+---------+-------+ |

**4.) Table- Passenger Details**

**CODE:**

|  |
| --- |
| **CREATE** TABLE **IF** **NOT** **EXISTS** `passenger\_details` (  `booking\_reference` int(11) **NOT** **NULL**,  `passenger\_name` varchar(50) **NOT** **NULL**,  `passenger\_age` int(11) **NOT** **NULL**,  `passenger\_gender` varchar(10) **NOT** **NULL**,  **PRIMARY** **KEY** (`booking\_reference`,`passenger\_name`,`passenger\_age`,`passenger\_gender`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1; |

|  |
| --- |
| +-------------------+--------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------------------+--------------+------+-----+---------+-------+ | booking\_reference | int(11) | NO | MUL | NULL | | | passenger\_name | varchar(50) | NO | | NULL | | | passenger\_age | int(11) | NO | | NULL | | | passenger\_gender | varchar(10) | NO | | NULL | | +-------------------+--------------+------+-----+---------+-------+ |

**5.) Table- Reservation**

**CODE:**

|  |
| --- |
| **CREATE** TABLE **IF** **NOT** **EXISTS** `reservation\_records` (  `booking\_reference` int(11) **NOT** **NULL** AUTO\_INCREMENT,  `user\_id` int(11) **NOT** **NULL**,  `train\_number` int(11) **NOT** **NULL**,  `start\_point` varchar(50) **NOT** **NULL**,  `destination\_point` varchar(50) **NOT** **NULL**,  `date\_of\_journey` date **NOT** **NULL**,  `total\_fare` int(11) **NOT** **NULL**,  `ticket\_class` varchar(50) **NOT** **NULL**,  `number\_of\_seats` int(11) **NOT** **NULL**,  `status` varchar(50) **NOT** **NULL**,  **PRIMARY** **KEY** (`booking\_reference`),  **UNIQUE** **KEY** `unique\_booking` (`user\_id`,`train\_number`,`date\_of\_journey`,`status`),  **UNIQUE** **KEY** `unique\_booking\_details` (`booking\_reference`,`user\_id`,`train\_number`,`date\_of\_journey`,`ticket\_class`,`status`),  **UNIQUE** **KEY** `unique\_booking\_details\_extended` (`booking\_reference`,`user\_id`,`train\_number`,`start\_point`,`destination\_point`,`date\_of\_journey`,`total\_fare`,`ticket\_class`,`number\_of\_seats`,`status`),  **KEY** `foreign\_key\_user\_id` (`user\_id`),  **KEY** `foreign\_key\_train\_number\_date\_class` (`train\_number`,`date\_of\_journey`,`ticket\_class`),  **KEY** `ticket\_class` (`ticket\_class`),  **KEY** `start\_point` (`start\_point`,`destination\_point`),  **KEY** `destination\_point` (`destination\_point`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=61; |

|  |
| --- |
| +-------------------+--------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------------------+--------------+------+-----+---------+-------+ | booking\_reference | int(11) | NO | PRI | NULL | | | user\_id | int(11) | NO | MUL | NULL | | | train\_number | int(11) | NO | MUL | NULL | | | start\_point | varchar(50) | NO | MUL | NULL | | | destination\_point | varchar(50) | NO | MUL | NULL | | | date\_of\_journey | date | NO | MUL | NULL | | | total\_fare | int(11) | NO | | NULL | | | ticket\_class | varchar(50) | NO | | NULL | | | number\_of\_seats | int(11) | NO | | NULL | | | status | varchar(50) | NO | | NULL | | +-------------------+--------------+------+-----+---------+-------+ |

**6.) Table- Train Schedule**

**CODE:**

|  |
| --- |
| **CREATE** TABLE **IF** **NOT** **EXISTS** `train\_schedule` (  `schedule\_id` int(11) **NOT** **NULL** AUTO\_INCREMENT,  `train\_number` int(11) **NOT** **NULL**,  `station\_name` varchar(50) **NOT** **NULL**,  `arrival\_time` time **NOT** **NULL**,  `departure\_time` time **NOT** **NULL** DEFAULT '00:00:00',  `distance\_from\_origin` int(11) **NOT** **NULL**,  **PRIMARY** **KEY** (`schedule\_id`),  **KEY** `train\_number` (`train\_number`),  **KEY** `station\_name` (`station\_name`),  **KEY** `distance\_from\_origin` (`distance\_from\_origin`),  **KEY** `schedule\_id\_extended` (`schedule\_id`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=42; |

|  |
| --- |
| +-----------------------+--------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-----------------------+--------------+------+-----+---------+-------+ | schedule\_id | int(11) | NO | PRI | NULL | | | train\_number | int(11) | NO | MUL | NULL | | | station\_name | varchar(50) | NO | MUL | NULL | | | arrival\_time | time | NO | | NULL | | | departure\_time | time | NO | | 00:00:00| | | distance\_from\_origin | int(11) | NO | | NULL | | +-----------------------+--------------+------+-----+---------+-------+ |

**7.) Table- Railway Station**

**CODE:**

|  |
| --- |
| **CREATE** TABLE **IF** **NOT** **EXISTS** `railway\_station` (  `station\_id` int(11) **NOT** **NULL** AUTO\_INCREMENT,  `station\_name` varchar(50) **NOT** **NULL**,  **PRIMARY** **KEY** (`station\_name`),  **KEY** `station\_id` (`station\_id`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=14; |

|  |
| --- |
| +-------------+-------------+------+-----+---------+-------+ | Field | Type | **Null** | Key | Default | Extra | +-------------+-------------+------+-----+---------+-------+ | station\_id | int(11) | **NO** | PRI | **NULL** | | | station\_name| varchar(50) | **NO** | | **NULL** | | +-------------+-------------+------+-----+---------+-------+ |

**8.) Table- Train Details**

**CODE:**

|  |
| --- |
| **CREATE** **TABLE** **IF** **NOT** **EXISTS** `train\_details` (  `train\_number` int(11) **NOT** **NULL** AUTO\_INCREMENT **COMMENT** 'Train\_no',  `train\_name` varchar(50) **NOT** **NULL** **COMMENT** 'Train\_name',  `start\_point` varchar(50) **NOT** **NULL** **COMMENT** 'Starting\_Point',  `start\_time` **time** **NOT** **NULL** **COMMENT** 'Arrival\_Time',  `destination\_point` varchar(50) **NOT** **NULL** **COMMENT** 'Destination\_Point',  `destination\_time` **time** **NOT** **NULL**,  `departure\_day` varchar(10) **DEFAULT** **NULL** **COMMENT** 'Day',  `total\_distance` int(11) **NOT** **NULL** **COMMENT** 'Distance',  PRIMARY **KEY** (`train\_number`),  **KEY** `start\_point` (`start\_point`),  **KEY** `destination\_point` (`destination\_point`) ) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=latin1 AUTO\_INCREMENT=24; |

|  |
| --- |
| +-------------------+--------------+------+-----+---------+-------+ | Field | Type | Null | Key | Default | Extra | +-------------------+--------------+------+-----+---------+-------+ | train\_number | int(11) | NO | PRI | NULL | | | train\_name | varchar(50) | NO | | NULL | | | start\_point | varchar(50) | NO | MUL | NULL | | | start\_time | time | NO | | NULL | | | destination\_point | varchar(50) | NO | MUL | NULL | | | destination\_time | time | NO | | NULL | | | departure\_day | varchar(10) | YES | | NULL | | | total\_distance | int(11) | NO | | NULL | | +-------------------+--------------+------+-----+---------+-------+ |

**9.) Table- user**

**CODE:**

|  |
| --- |
| **CREATE** TABLE **IF** **NOT** **EXISTS** `users` (  `user\_id` int(11) **NOT** **NULL** AUTO\_INCREMENT,  `email\_address` varchar(50) **NOT** **NULL**,  `password` varchar(50) **NOT** **NULL**,  `mobile\_number` varchar(10) **NOT** **NULL**,  `date\_of\_birth` date **NOT** **NULL**,  **PRIMARY** **KEY** (`user\_id`),  **UNIQUE** **KEY** `unique\_mobile\_number` (`mobile\_number`),  **UNIQUE** **KEY** `unique\_email\_address` (`email\_address`) ) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=21; |

|  |
| --- |
| +-----------+--------------+------+-----+---------+----------------+ | Field | Type | Null | Key | Default | Extra | +-----------+--------------+------+-----+---------+----------------+ | id | int(11) | NO | PRI | NULL | auto\_increment | | emailid | varchar(50) | NO | UNI | NULL | | | password | varchar(50) | NO | | NULL | | | mobileno | varchar(10) | NO | UNI | NULL | | | dob | date | NO | | NULL | | +-----------+--------------+------+-----+---------+----------------+ |

Constraints

1. **Constraints for table `classseats`**

|  |
| --- |
| **ALTER** **TABLE** `classseats`  **ADD** **CONSTRAINT** `fk\_classseats\_trainno` FOREIGN **KEY** (`trainno`) **REFERENCES** `train` (`trainno`),  **ADD** **CONSTRAINT** `fk\_classseats\_start\_point` FOREIGN **KEY** (`sp`) **REFERENCES** `station` (`sname`),  **ADD** **CONSTRAINT** `fk\_classseats\_destination\_point` FOREIGN **KEY** (`dp`) **REFERENCES** `station` (`sname`),  **ADD** **CONSTRAINT** `fk\_classseats\_ticket\_class` FOREIGN **KEY** (`class`) **REFERENCES** `ticket\_class` (`class\_name`); |

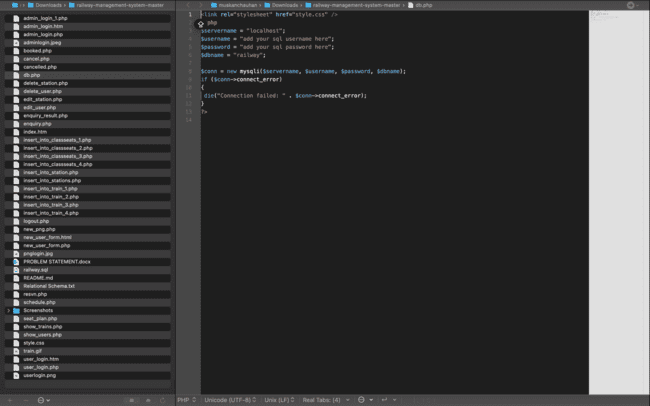
1. **Constraints for table `resv`**

|  |
| --- |
| **ALTER** **TABLE** `resv`  **ADD** **CONSTRAINT** `fk\_resv\_trainno` FOREIGN **KEY** (`trainno`) **REFERENCES** `train` (`trainno`),  **ADD** **CONSTRAINT** `fk\_resv\_start\_point` FOREIGN **KEY** (`sp`) **REFERENCES** `station` (`sname`),  **ADD** **CONSTRAINT** `fk\_resv\_destination\_point` FOREIGN **KEY** (`dp`) **REFERENCES** `station` (`sname`); |

**Code of connection of frontend and backend**

**Connection between frontend and backend**

**CODE:**

****

**Codes of procedures, functions and triggers**

**PROCEDURES:**

1. **To add a new train to the database.**

|  |
| --- |
| DELIMITER //  **CREATE** **PROCEDURE** **sp\_add\_train** (  **IN** p\_trainno INT,   **IN** p\_name VARCHAR(50),   **IN** **p\_source** **VARCHAR**(50),   **IN** **p\_dest** **VARCHAR**(50),   **IN** **p\_date** **DATE** ) **BEGIN**  **INSERT** **INTO** **train** (train\_number, train\_name, start\_point, destination\_point, start\_time)  **VALUES** (p\_trainno, p\_name, p\_source, p\_dest, p\_date); **END** //  DELIMITER ; |

1. **To delete a train from the database.**

|  |
| --- |
| DELIMITER //  **CREATE** **PROCEDURE** **sp\_delete\_train** (**IN** p\_trainno INT) **BEGIN**  **DELETE** **FROM** **train**  **WHERE** **train\_number** = **p\_trainno**; **END** //  DELIMITER ; |

1. **To search for trains based on the source and destination stations.**

|  |
| --- |
| DELIMITER //  CREATE PROCEDURE sp\_search\_train (IN p\_source VARCHAR(50), IN p\_dest VARCHAR(50)) BEGIN  SELECT \* FROM train  WHERE start\_point = p\_source AND destination\_point = p\_dest; END //  DELIMITER ; |

1. **To book a ticket for a passenger.**

|  |
| --- |
| DELIMITER //  **CREATE** **PROCEDURE** sp\_book\_ticket (  **IN** p\_name VARCHAR(50),   **IN** p\_age INT,   **IN** p\_gender CHAR(1),   **IN** p\_trainno INT,   **IN** p\_class VARCHAR(20),   **IN** p\_journeydate DATE ) **BEGIN**  **DECLARE** classseats INT;  DECLARE bookedseats INT;  DECLARE availableseats INT;    **SELECT** seats\_left **INTO** classseats **FROM** classseats  **WHERE** trainno = p\_trainno **AND** class = p\_class;    **SELECT** COUNT(\*) **INTO** bookedseats **FROM** reservation\_records  **WHERE** trainno = p\_trainno **AND** ticket\_class = p\_class **AND** date\_of\_journey = p\_journeydate;    SET availableseats = classseats - bookedseats;    IF availableseats > 0 THEN  **INSERT** **INTO** reservation\_records (passenger\_name, passenger\_age, passenger\_gender, train\_number, ticket\_class, date\_of\_journey)  **VALUES** (p\_name, p\_age, p\_gender, p\_trainno, p\_class, p\_journeydate);    **SELECT** CONCAT('Ticket booked for passenger ', p\_name) **AS** message;  ELSE  **SELECT** 'No seats available in the specified class.' **AS** message;  END IF; END //  DELIMITER ; |

1. **To cancel a booked ticket.**

|  |
| --- |
| DELIMITER //  **CREATE** **PROCEDURE** **sp\_cancel\_ticket** (**IN** p\_name VARCHAR(50), **IN** **p\_journeydate** **DATE**) **BEGIN**  **DELETE** **FROM** **reservation\_records**  **WHERE** **passenger\_name** = **p\_name** **AND** **date\_of\_journey** = **p\_journeydate**; **END** //  DELIMITER ; |

**TRIGGERS:**

1. **Triggers `classseats`**
2. BEFORE INSERT ON `classseats`

|  |
| --- |
| DELIMITER //  **CREATE** TRIGGER `before\_insert\_on\_classseats` BEFORE INSERT **ON** `classseats` **FOR** **EACH** ROW  **BEGIN**  **IF** DATEDIFF(CURDATE(), **NEW**.doj) > 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Check date!!!';  **END** **IF**;   **IF** **NEW**.fare <= 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Check fare!!!';  **END** **IF**;   **IF** **NEW**.seatsleft <= 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Check seats!!!';  **END** **IF**; **END** //  DELIMITER ; |

1. BEFORE UPDATE ON `classseats`

|  |
| --- |
| DELIMITER //  **CREATE** TRIGGER `before\_update\_on\_classseats` BEFORE UPDATE **ON** `classseats` **FOR** **EACH** ROW  **BEGIN**  **IF** DATEDIFF(CURDATE(), **NEW**.doj) > 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Check date!!!';  **END** **IF**;   **IF** **NEW**.fare <= 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Check fare!!!';  **END** **IF**;   **IF** **NEW**.seatsleft <= 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Check seats!!!';  **END** **IF**; **END** //  DELIMITER ; |

**2. Triggers `pd`**

1. BEFORE INSERT ON `pd`

|  |
| --- |
| DELIMITER //  **CREATE** **TRIGGER** `before\_insert\_on\_pd` **BEFORE** **INSERT** **ON** `pd` **FOR** **EACH** **ROW**  **BEGIN**  **IF** NEW.pgender **NOT** **IN** ('M', 'F') **THEN**  SIGNAL **SQLSTATE** '45000' **SET** MESSAGE\_TEXT = 'Enter M for Male or F for Female.';  **END** **IF**; **END** //  DELIMITER ; |

1. BEFORE UPDATE ON `pd`

|  |
| --- |
| DELIMITER //  **CREATE** **TRIGGER** `before\_update\_on\_pd` **BEFORE** **UPDATE** **ON** `pd` **FOR** **EACH** **ROW**  **BEGIN**  **IF** NEW.pgender **NOT** **IN** ('M', 'F') **THEN**  SIGNAL **SQLSTATE** '45000' **SET** MESSAGE\_TEXT = 'Enter M for Male or F for Female.';  **END** **IF**; **END** //  DELIMITER ; |

**3. Triggers `resv`**

1. AFTER INSERT ON `resv`

|  |
| --- |
| DELIMITER //  **CREATE** **TRIGGER** `after\_insert\_on\_resv` **AFTER** **INSERT** **ON** `resv` **FOR** **EACH** **ROW**  **BEGIN**  **UPDATE** classseats   **SET** seats\_left = seats\_left - NEW.nos   **WHERE** trainno = NEW.trainno   **AND** **class** = NEW.ticket\_class   **AND** doj = NEW.date\_of\_journey   **AND** sp = NEW.start\_point   **AND** dp = NEW.destination\_point; **END** //  DELIMITER ; |

1. AFTER UPDATE ON `resv`

|  |
| --- |
| DELIMITER //  **CREATE** TRIGGER `after\_update\_on\_resv` AFTER UPDATE **ON** `resv` **FOR** **EACH** ROW  **BEGIN**  **IF** (**NEW**.status = 'CANCELLED' **AND** DATEDIFF(**NEW**.date\_of\_journey, CURDATE()) < 0) **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Cancellation Not Possible!!!!';  **END** **IF**;   **IF** (**NEW**.status = 'CANCELLED' **AND** DATEDIFF(**NEW**.date\_of\_journey, CURDATE()) > 0) **THEN**  UPDATE classseats   **SET** seats\_left = seats\_left + **NEW**.nos   **WHERE** trainno = **NEW**.trainno   **AND** ticket\_class = **NEW**.ticket\_class   **AND** doj = **NEW**.date\_of\_journey   **AND** sp = **NEW**.start\_point   **AND** dp = **NEW**.destination\_point;   **IF** DATEDIFF(**NEW**.date\_of\_journey, CURDATE()) >= 30 **THEN**  INSERT **INTO** canc VALUES (**NEW**.pnr, **NEW**.tfare);  **END** **IF**;   **IF** DATEDIFF(**NEW**.date\_of\_journey, CURDATE()) < 30 **THEN**  INSERT **INTO** canc VALUES (**NEW**.pnr, 0.5 \* **NEW**.tfare);  **END** **IF**;  **END** **IF**; **END** //  DELIMITER ; |

1. BEFORE INSERT ON `resv`

|  |
| --- |
| DELIMITER //  **CREATE** TRIGGER `before\_insert\_on\_resv` BEFORE INSERT **ON** `resv` **FOR** **EACH** ROW  **BEGIN**  **IF** **NEW**.tfare < 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Negative balance NOT possible';  **END** **IF**;   **IF** **NEW**.nos <= 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Negative OR 0 seats NOT possible';  **END** **IF**;   DECLARE available\_seats INT;  **SELECT** seats\_left **INTO** available\_seats   **FROM** classseats   **WHERE** trainno = **NEW**.trainno   **AND** ticket\_class = **NEW**.ticket\_class   **AND** doj = **NEW**.date\_of\_journey   **AND** sp = **NEW**.start\_point   **AND** dp = **NEW**.destination\_point;   **IF** available\_seats < **NEW**.nos **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Not enough seats available!!!';  **END** **IF**;   **IF** DATEDIFF(**NEW**.date\_of\_journey, CURDATE()) < 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Booking Not Possible!!!!';  **END** **IF**;   **SET** **NEW**.status = 'BOOKED'; **END** //  DELIMITER ; |

1. BEFORE UPDATE ON `resv`

|  |
| --- |
| DELIMITER //  **CREATE** TRIGGER `before\_update\_on\_resv` BEFORE UPDATE **ON** `resv` **FOR** **EACH** ROW  **BEGIN**  **IF** **NEW**.tfare < 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Negative balance NOT possible';  **END** **IF**;   **IF** **NEW**.nos <= 0 **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Negative OR 0 seats NOT possible';  **END** **IF**;   DECLARE available\_seats INT;  **SELECT** seats\_left **INTO** available\_seats   **FROM** classseats   **WHERE** trainno = **NEW**.trainno   **AND** ticket\_class = **NEW**.ticket\_class   **AND** doj = **NEW**.date\_of\_journey   **AND** sp = **NEW**.start\_point   **AND** dp = **NEW**.destination\_point;   **IF** available\_seats < **NEW**.nos **THEN**  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Not enough seats available!!!';  **END** **IF**; **END** //  DELIMITER ; |

**4. Triggers `train`**

1. BEFORE INSERT ON `train`

|  |
| --- |
| DELIMITER //  **CREATE** **TRIGGER** `before\_insert\_on\_train` **BEFORE** **INSERT** **ON** `train` **FOR** **EACH** **ROW**  **BEGIN**  **IF** (NEW.dt < NEW.st **AND** NEW.dd = 'Day 1') **THEN**  SIGNAL **SQLSTATE** '45000' **SET** MESSAGE\_TEXT = 'Improper Timings';  **END** **IF**;   IF (NEW.dp = NEW.sp) THEN  SIGNAL SQLSTATE '45000' **SET** MESSAGE\_TEXT = 'Same Starting & Destination Points not allowed';  **END** **IF**; **END** //  DELIMITER ; |

1. BEFORE UPDATE ON `train`

|  |
| --- |
| DELIMITER //  **CREATE** **TRIGGER** `before\_update\_on\_train` **BEFORE** **UPDATE** **ON** `train` **FOR** **EACH** **ROW**  **BEGIN**  **IF** (NEW.dt < NEW.st **AND** NEW.dd = 'Day 1') **THEN**  SIGNAL **SQLSTATE** '45000' **SET** MESSAGE\_TEXT = 'Improper Timings';  **END** **IF**; **END** //  DELIMITER ; |

**5. Triggers `user`**

1. BEFORE INSERT ON `user`

|  |
| --- |
| DELIMITER //  **CREATE** **TRIGGER** `before\_insert\_on\_user` **BEFORE** **INSERT** **ON** `user` **FOR** **EACH** **ROW**  **BEGIN**  **IF** (**YEAR**(**CURDATE**()) - **YEAR**(NEW.dob)) < 18 **THEN**  SIGNAL **SQLSTATE** '45000' **SET** MESSAGE\_TEXT = 'Minimum age bar of 18 years.';  **END** **IF**; **END** //  DELIMITER ; |

1. BEFORE UPDATE ON `user`

|  |
| --- |
| DELIMITER //  **CREATE** **TRIGGER** `before\_update\_on\_user` **BEFORE** **UPDATE** **ON** `user` **FOR** **EACH** **ROW**  **BEGIN**  **IF** (**YEAR**(**CURDATE**()) - **YEAR**(NEW.dob)) < 18 **THEN**  SIGNAL **SQLSTATE** '45000' **SET** MESSAGE\_TEXT = 'Minimum age bar of 18 years.';  **END** **IF**; **END** //  DELIMITER ; |

**Functions:**

1. **To check availability of seats.**

|  |
| --- |
| DELIMITER //   **CREATE** **FUNCTION** check\_available\_seats(train\_no INT, class\_name VARCHAR(20), journey\_date DATE) **RETURNS** INT **BEGIN**  **DECLARE** total\_seats INT;  **DECLARE** booked\_seats INT;  **DECLARE** available\_seats INT;     **SELECT** seats\_left **INTO** total\_seats **FROM** classseats **WHERE** trainno = train\_no **AND** ticket\_class = class\_name;  **SELECT** **COUNT**(\*) **INTO** booked\_seats **FROM** resv **WHERE** trainno = train\_no **AND** ticket\_class = class\_name **AND** date\_of\_journey = journey\_date;    **SET** available\_seats = total\_seats - booked\_seats;    RETURN available\_seats; **END** //  DELIMITER ; |

1. **To get the list of trains running between two stations on a given date.**

|  |
| --- |
| DELIMITER //  **CREATE** **FUNCTION** get\_trains\_running(sp\_name VARCHAR(50), dp\_name VARCHAR(50), journey\_date DATE) **RETURNS** VARCHAR(500) **BEGIN**  **DECLARE** train\_list VARCHAR(500);    **SELECT** **GROUP\_CONCAT**(**DISTINCT** train.tname) **INTO** train\_list  **FROM** classseats  **INNER** **JOIN** train **ON** classseats.trainno = train.trainno  **INNER** **JOIN** schedule **ON** train.trainno = schedule.trainno  **WHERE** classseats.sp = sp\_name **AND** classseats.dp = dp\_name **AND** schedule.arrival\_time >= journey\_date **AND** schedule.arrival\_time < journey\_date + INTERVAL 1 **DAY**;    RETURN train\_list; **END** //  DELIMITER ; |

1. **To get the list of stations between two stations for a given train.**

|  |
| --- |
| DELIMITER //  **CREATE** **FUNCTION** get\_stations\_between(train\_no INT, sp\_name VARCHAR(50), dp\_name VARCHAR(50)) **RETURNS** VARCHAR(500) **BEGIN**  **DECLARE** station\_list VARCHAR(500);    **SELECT** **GROUP\_CONCAT**(**DISTINCT** station.sname **ORDER** **BY** route.distance **ASC** SEPARATOR ', ') **INTO** station\_list  **FROM** route  **INNER** **JOIN** station **ON** route.sname = station.sname  **WHERE** route.trainno = train\_no **AND** (route.sname >= sp\_name **AND** route.sname <= dp\_name);    RETURN station\_list; **END** //  DELIMITER ; |

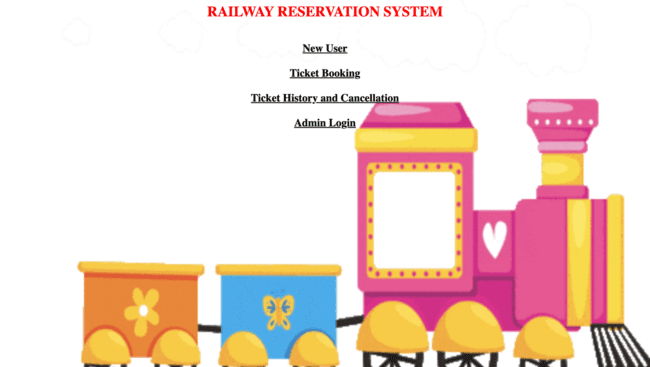
1. **To get the list of passengers with their seat numbers for a given PNR number.**

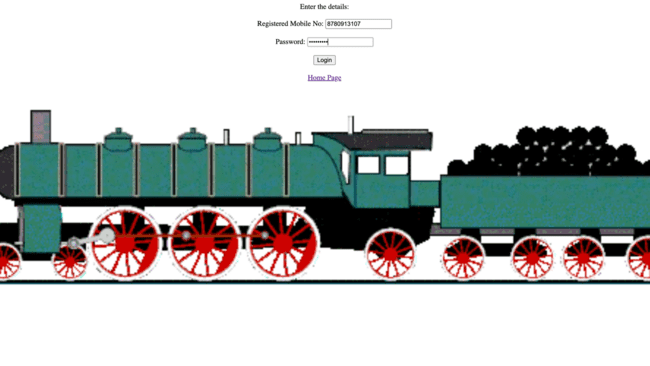
|  |
| --- |
| DELIMITER //  **CREATE** **FUNCTION** get\_passenger\_list(pnr\_num INT) **RETURNS** VARCHAR(500) **BEGIN**  **DECLARE** passenger\_list VARCHAR(500);    **SELECT** **GROUP\_CONCAT**(**CONCAT**(pname, ' - Seat No. ', seatno) SEPARATOR ', ') **INTO** passenger\_list  **FROM** pd  **WHERE** pnr = pnr\_num;    RETURN passenger\_list; **END** //  DELIMITER ; |

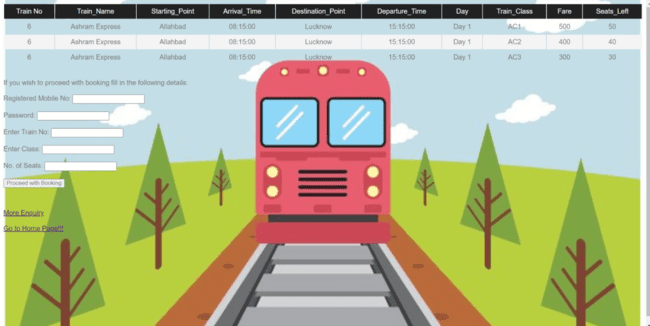
1. **To check if a user is eligible for a senior citizen discount.**

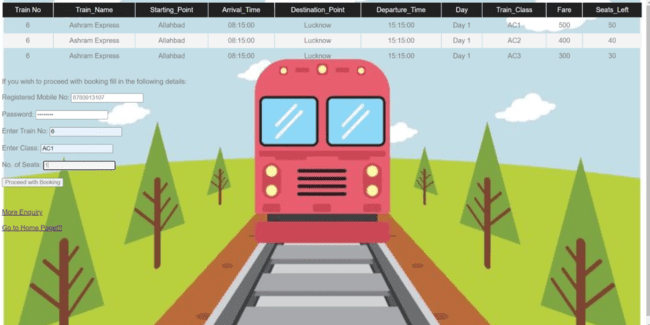
|  |
| --- |
| DELIMITER //  **CREATE** **FUNCTION** check\_senior\_citizen(age INT) **RETURNS** VARCHAR(20) **BEGIN**  **IF** age >= 60 **THEN**  **RETURN** 'Eligible';  ELSE  RETURN 'Not eligible';  **END** **IF**; **END** //  DELIMITER ; |
|  |

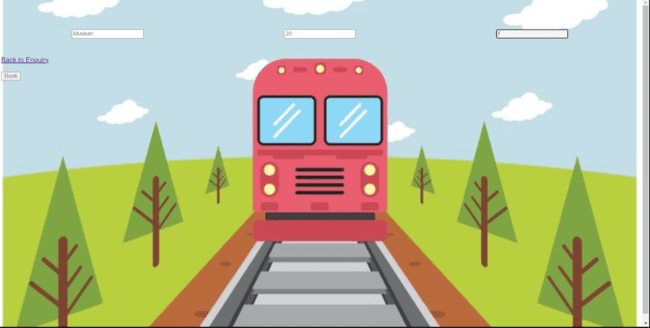
**Screenshots of results generated after procedures and functions are called on the front-end**

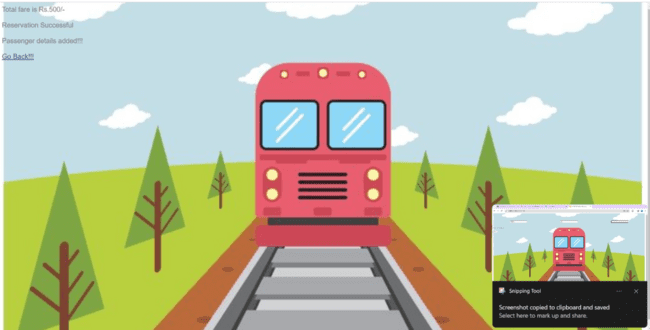
****

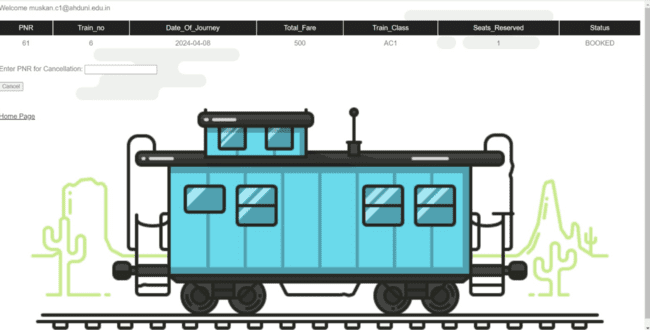
****

****

****

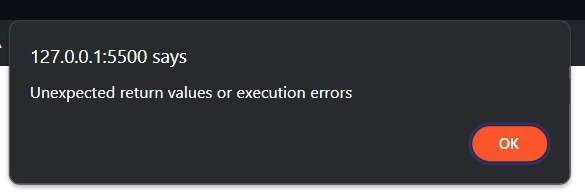
****

****

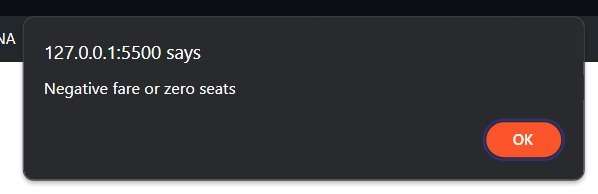
****

**Screen shots of errors generated on the front-end when trigger is fired**

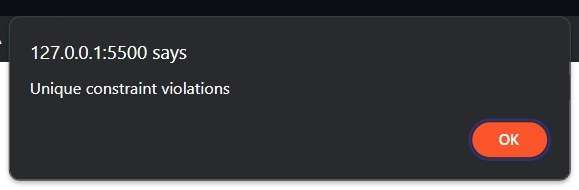
**1.**

****

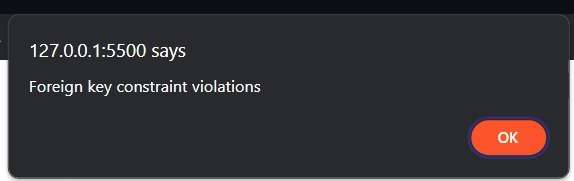
**2.**

****

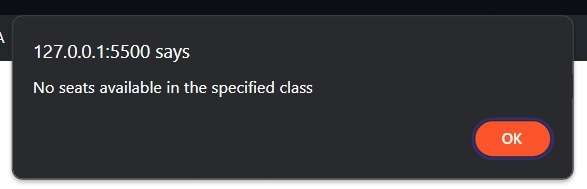
**3.**

****

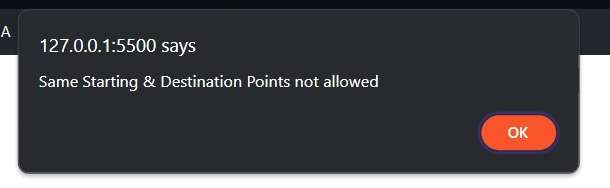
**4.**

****

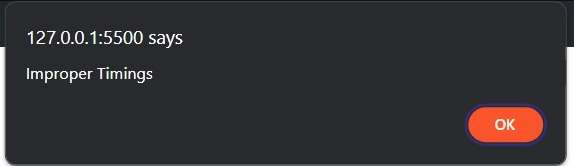
**5.**

****

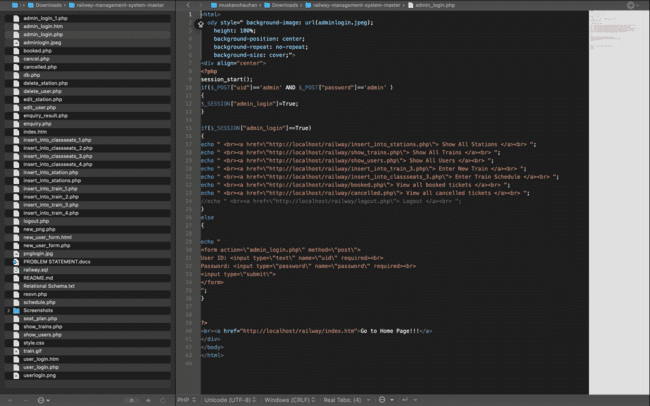
**6.**

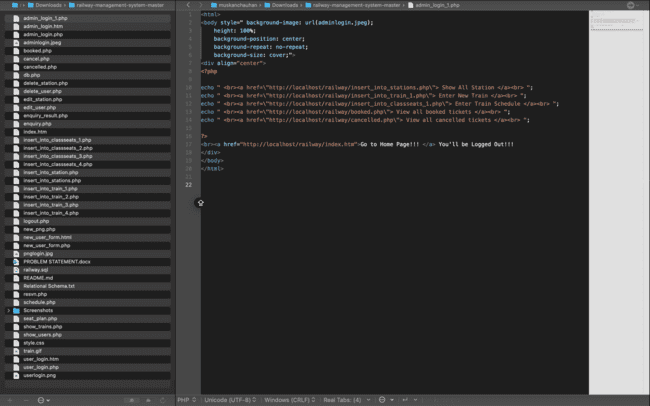
****

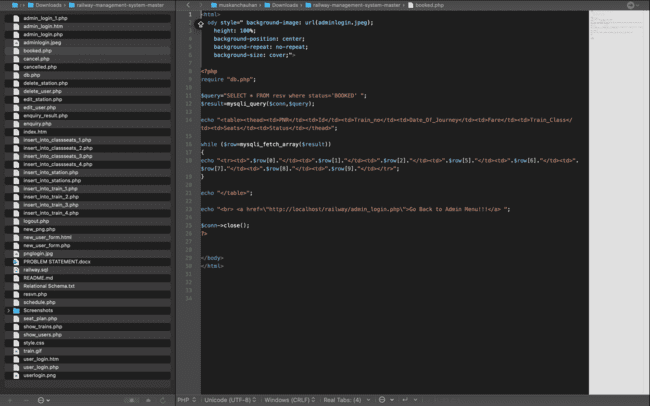
**7.**

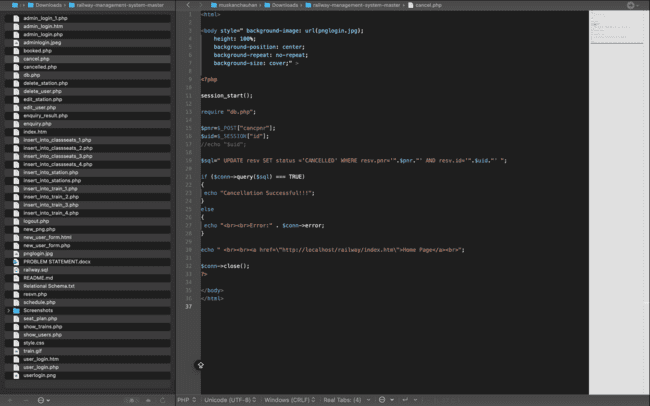
****

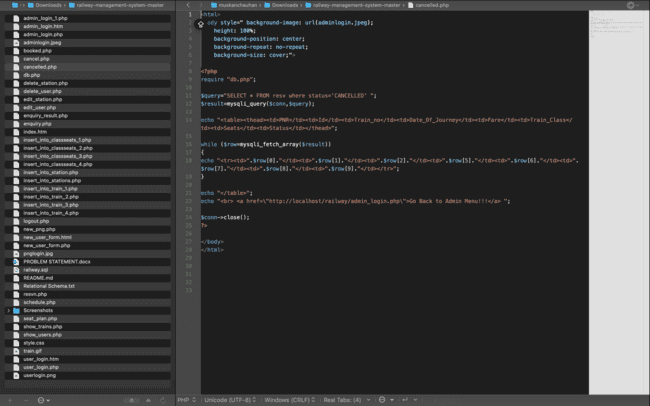
**Screenshots of Backend**

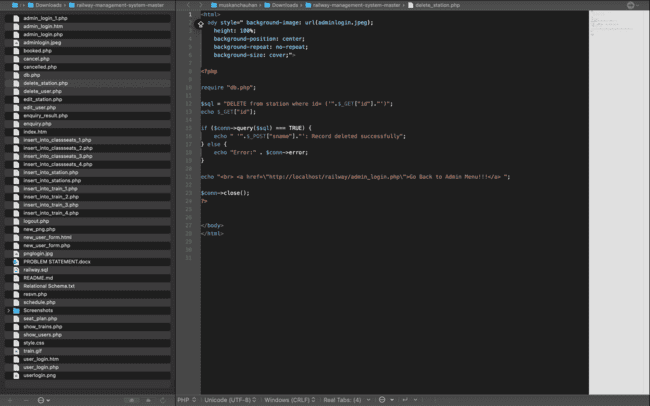
****

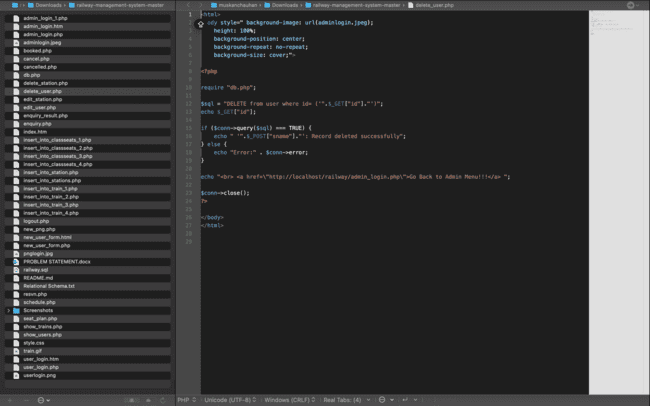
****

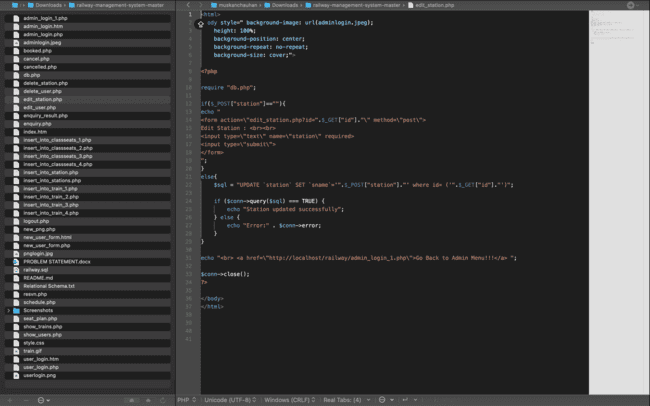
****

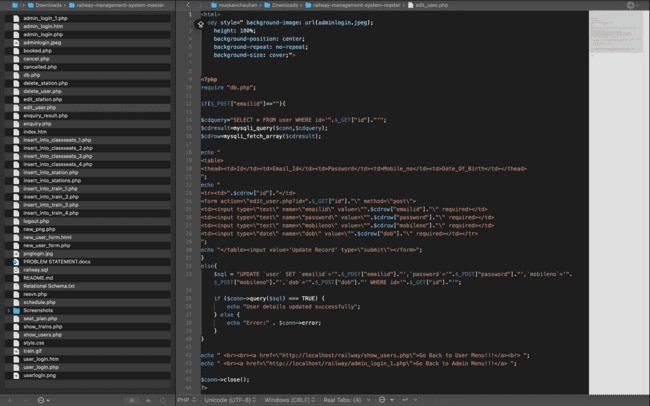
****

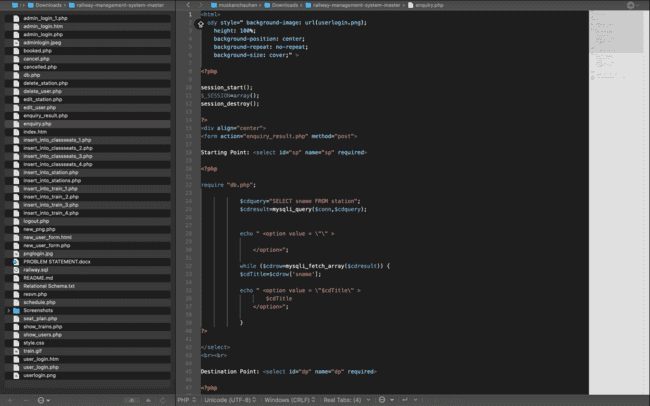
****

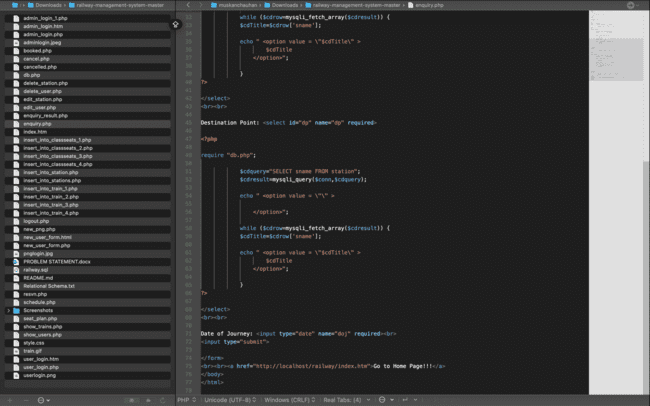
****

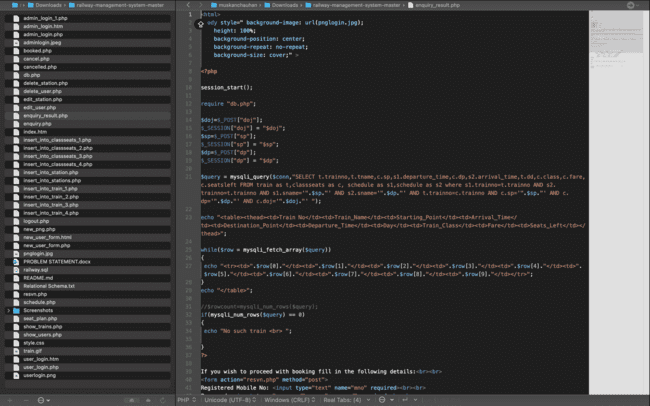
****

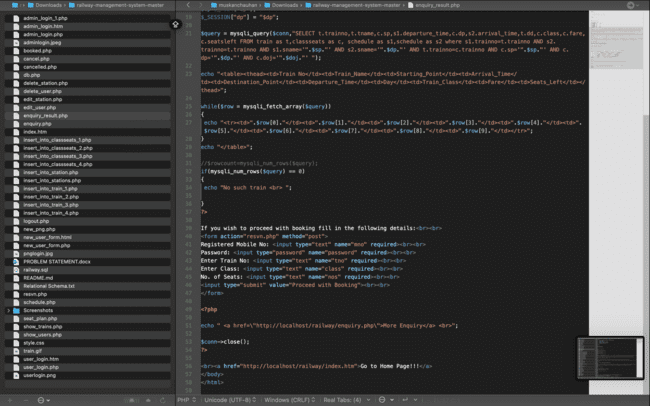
****

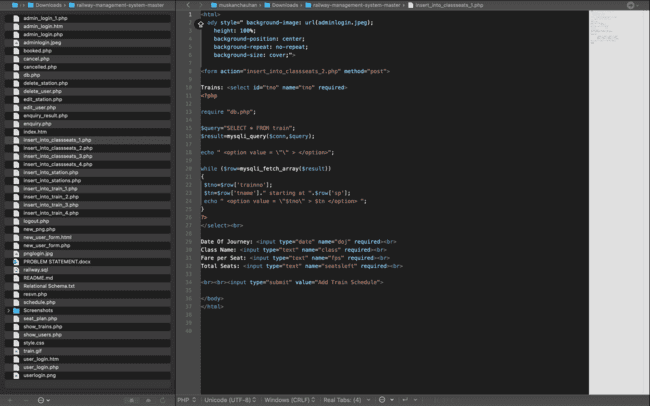
****

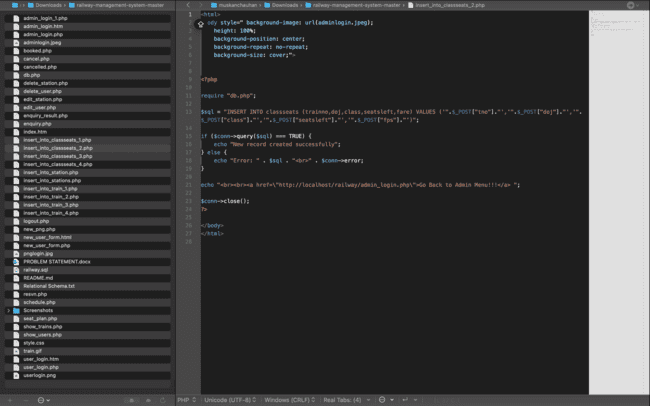
****

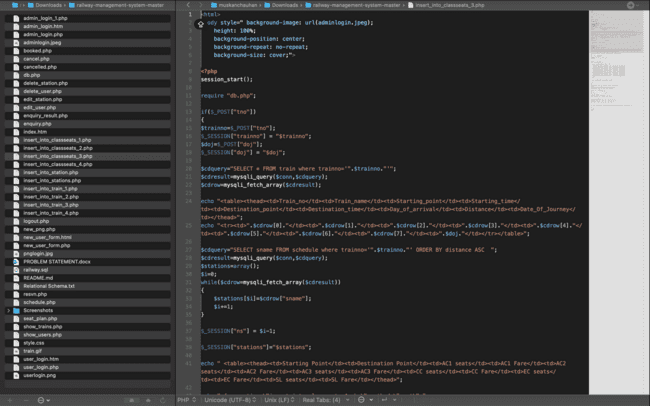
****

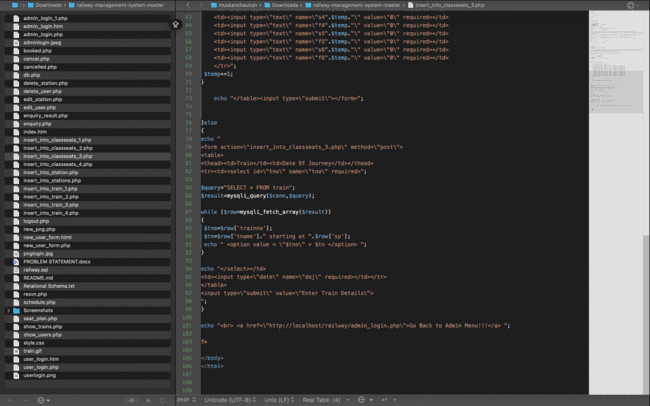
****

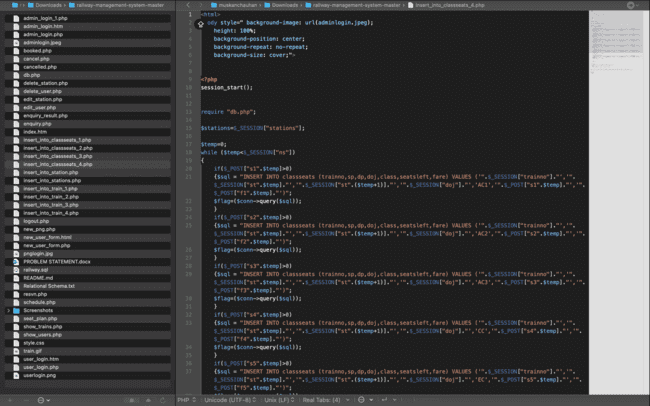
****

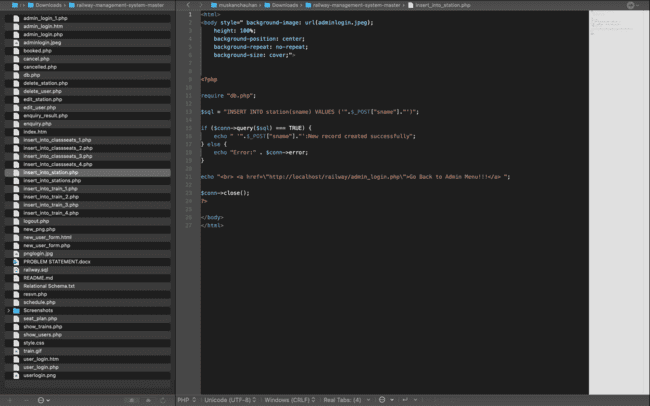
****

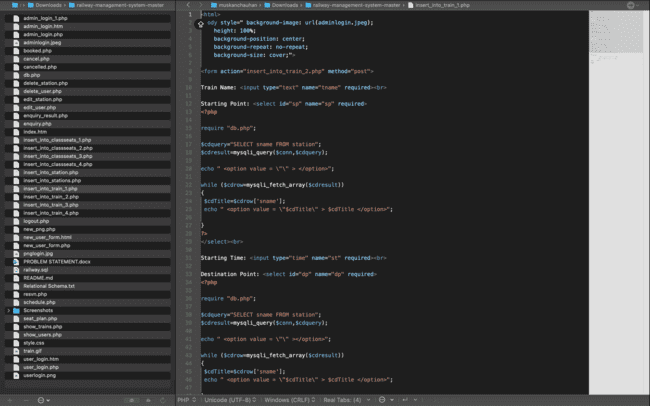
****

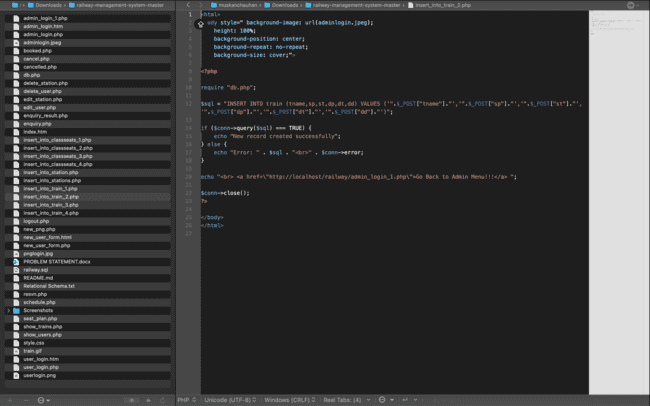
****

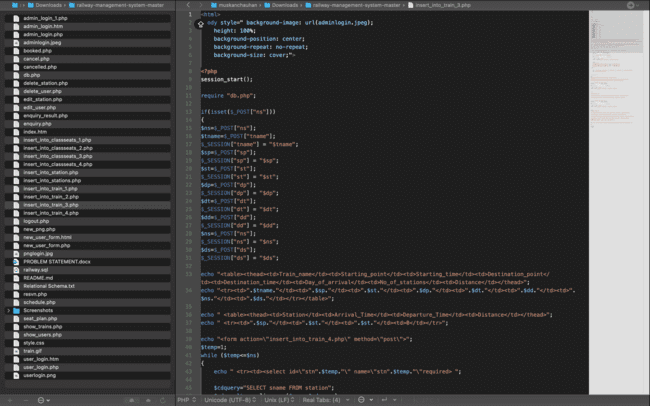
****

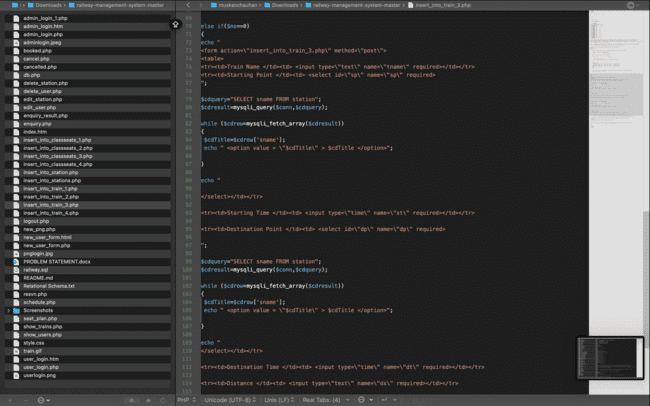
****

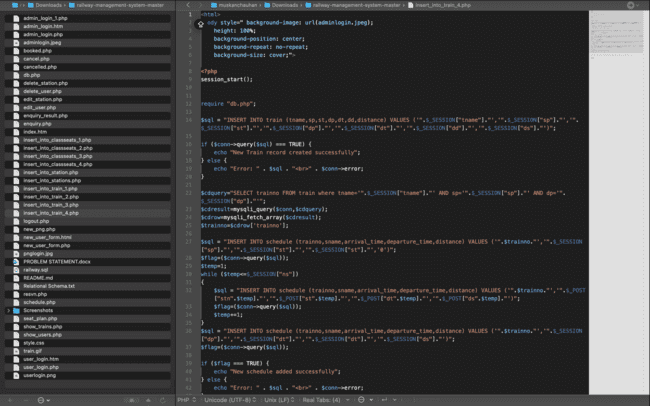
****

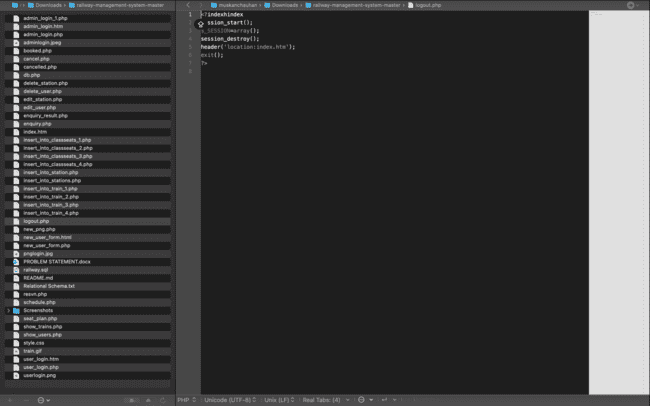
****

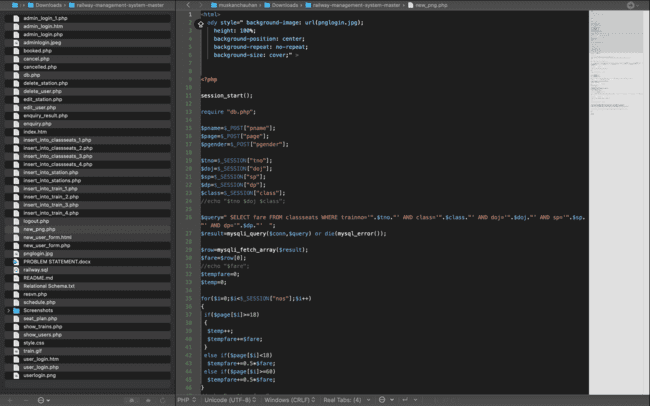
****

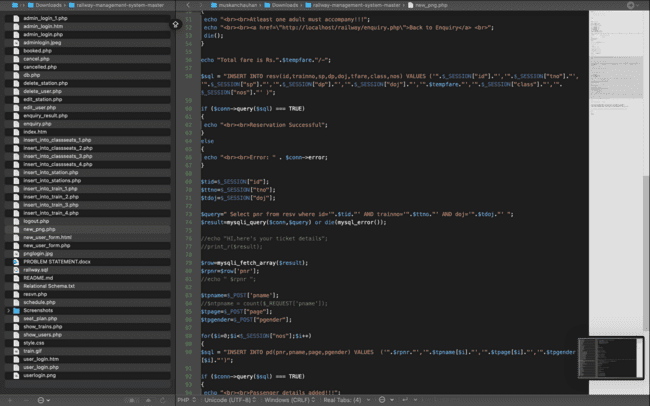
****

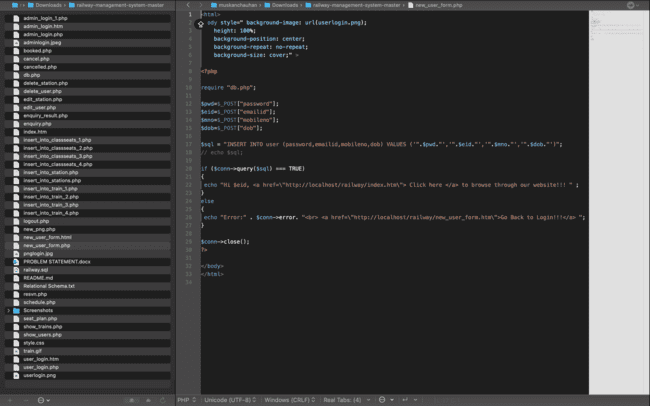
****

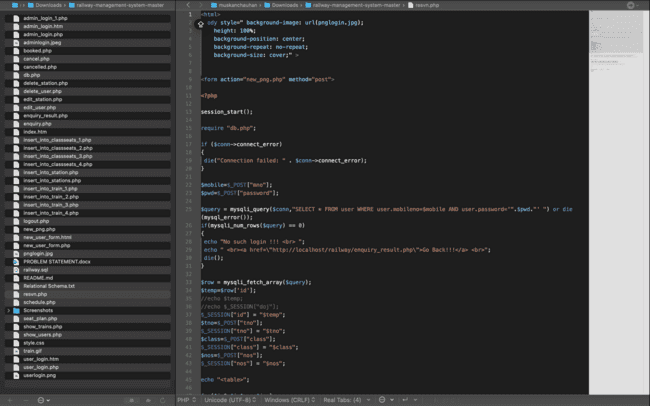
****

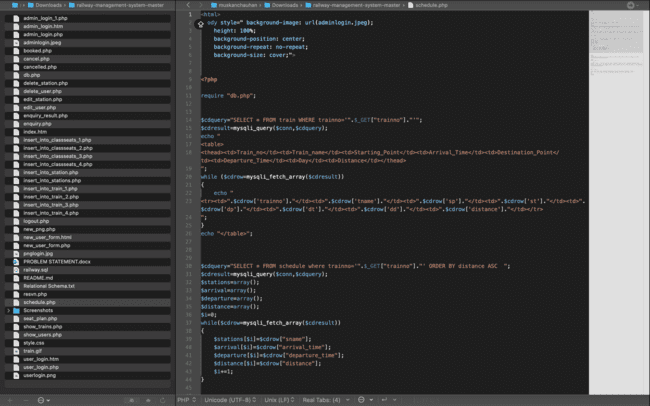
****

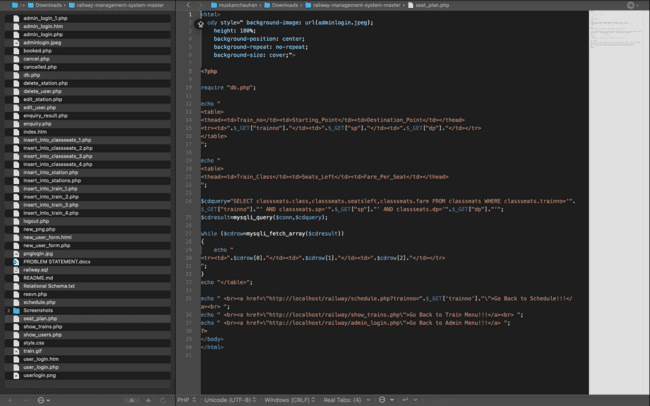
****

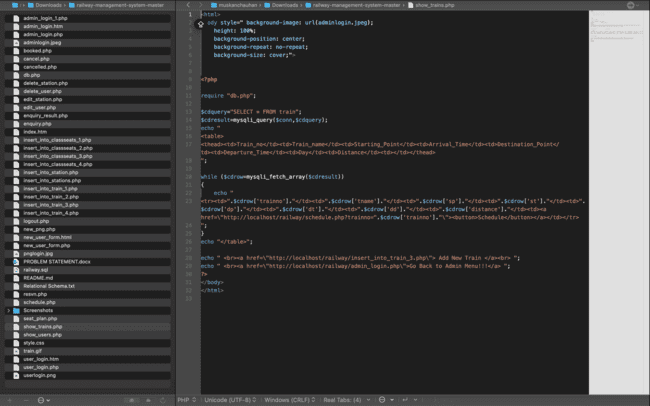
****

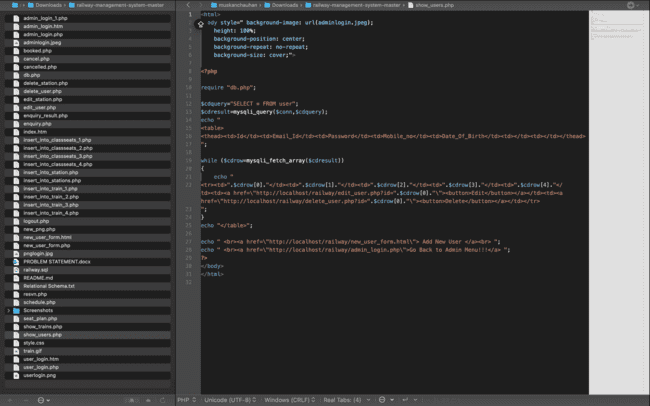
****

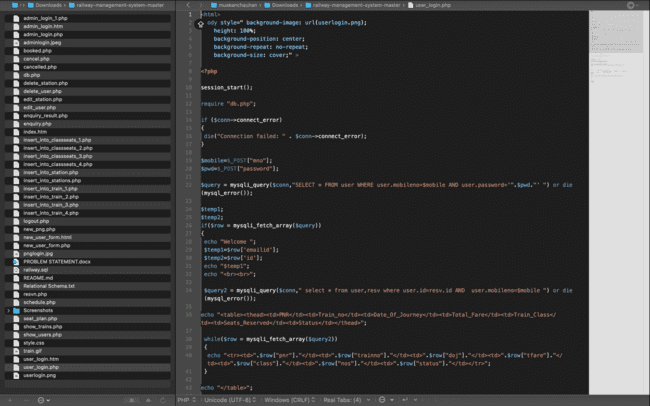
****

****

****

****

****

****