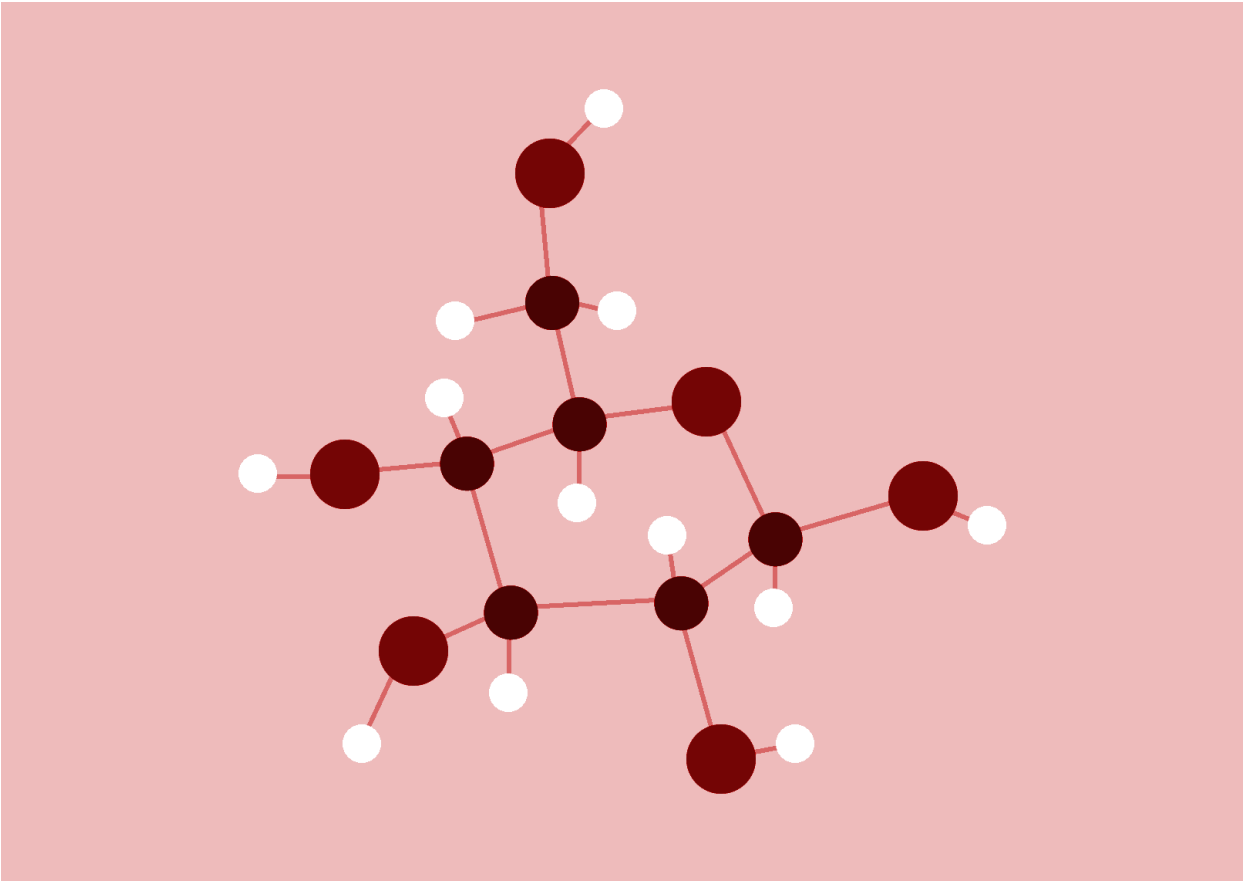
Railway Management System

Database Management System



Group Members

Dhyey Patel AU2040257

Raj Patel AU2040115

Ruchit Ghodasara AU2040033

2nd Year Computer Science and Engineering

INTRODUCTION

This project is about the railway management system. It covers trains, routes on which these trains run, various stations upon these routes, train timings and current status and available seats. The overall flow of the program is inspired by the Indian Railway Catering and Tourism Corporation (IRCTC) website.

User query by just specifying boarding station and destination station along with journey date. To save data as an all possible combination of these three parameters is not feasible, as there may be thousands of stations and the number of pairs will be in order of this factorial! So, station matches according to each input have to be computed by some procedure at run time. We have implemented this logic.

To book a ticket one has to either have login credentials or can create a new profile. In this implementation each ticket holder must have their unique user credential. They can further use this information to book tickets in future. And to record individual journeys we refer to the user as a passenger. A passenger must have user credentials, and on one user id they may have more than one journey.

Any train has certain specifications, number of compartments, AC/Non-AC coaches and current availability of these seats. We have limited complexity and size of the project by only considering seats availability as a final attribute. But we design our tables in a way that these attributes can be inserted as per the requirements. All these constraints have their individual tables and can add in future.

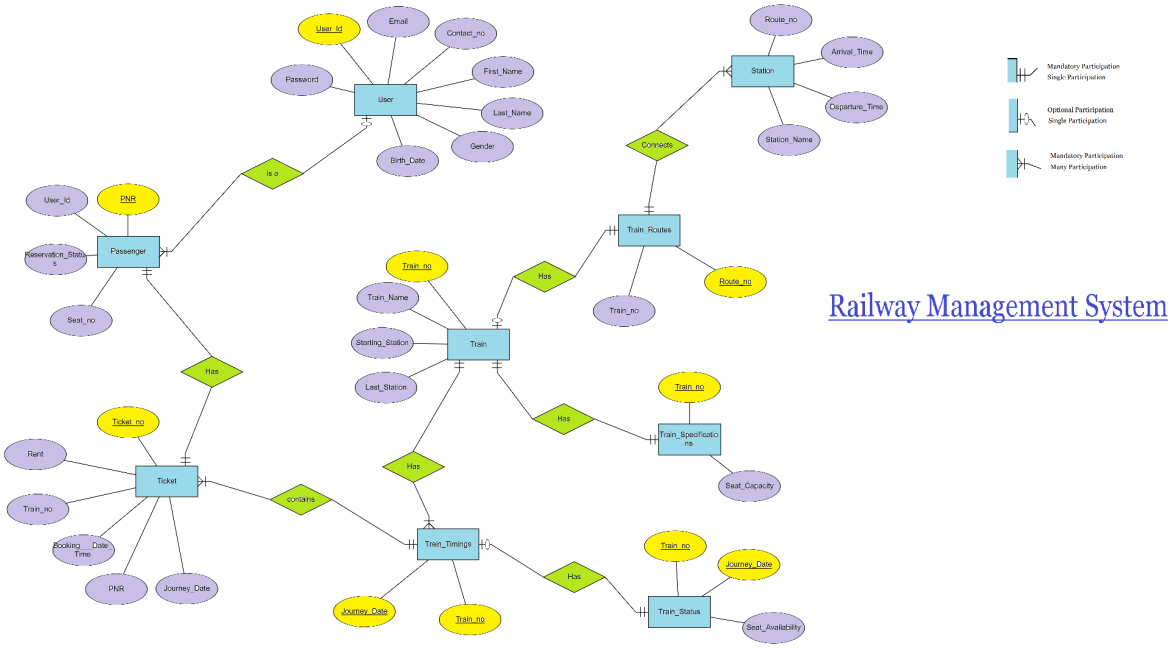
The station data and timings have been referred directly from IRCTC to show output which seems meaningful. A hidden table to maintain user information activities, such as any updates, has been available on the backend.

Some of the important and crucial points we have to omit to make the project simple. And one of them is payment handling. We simply generate the ticket and update reservation status in passenger credentials by considering completion of payment.

Overall, this covers a lot. Also, can be extended further. So, we feel happy to put our project before you.

INDEX

ER Diagram



Railway Management System

This is our Entity Relationship (ER) Diagram. Containing 9 tables, one additional hidden table to record changes into the User Table. All the data is divided according to this schema and stored in individual tables.

User and Passenger

These tables contain user information. Passenger, more specifically, covers journey related information as well.

Ticket

This table contains journey information; like train_no, journey date, fare of traveling - as well as PNR no to identify passengers and by which user.

Train and Train Timing

Train contains the train name and route on which it runs along with the first and last station. These two stations are only for referring routes at glance. Train timing relates to the train table by train number and contains the journey date. This manages journey dates. Any new schedule has to be added in this table.

Train routes and Station

Each train runs on a specific route. And each route covers many stations. Most of the users travel between the intermediate stations and not from where the train starts and the last stop. So, the Station table contains all the intermediate stations as well along with its expected time of journey, both forward and backward journeys.

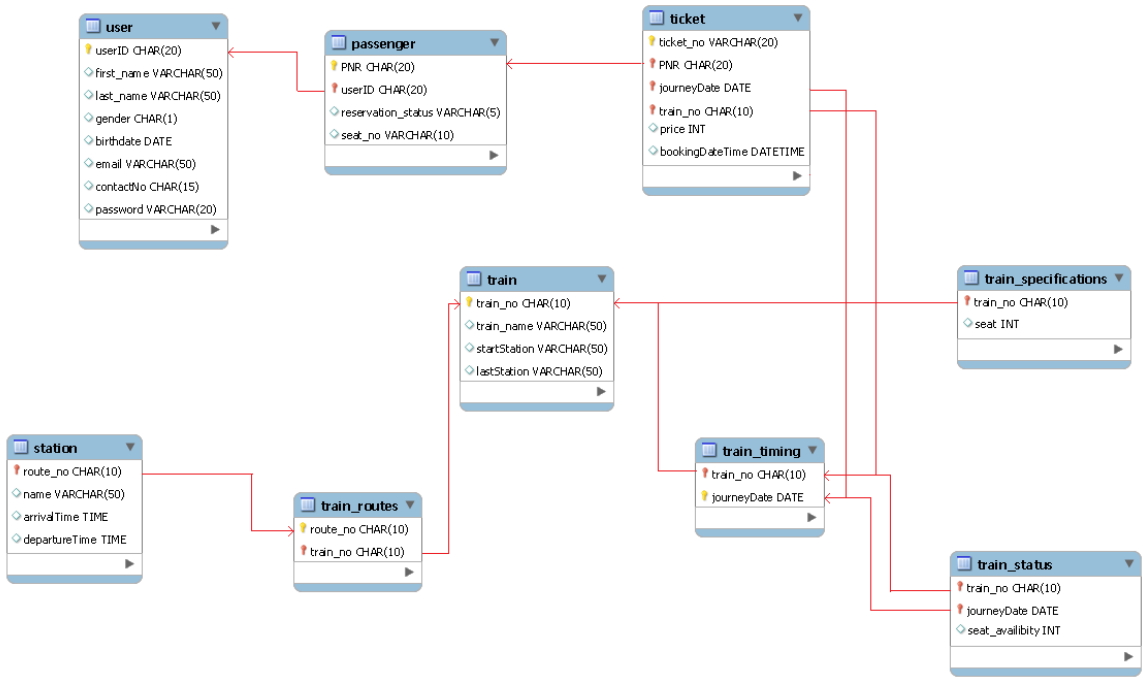
Train Specification and Train Station

Here, this table shows the capacity of each train, more specifically train along with journey date, and current available seats. It can be extended to different types of seat, sleeper or seater, or coach type, AC/Non-AC.

Track User

This hidden table records all the changes made on the User table. Changes may include new user creation, deletion of user account or any update on that table.

Relational Schema



The following schema represents logical relations along with the allotted attribute size and type.

- Primary keys are given in the yellow flag. Foreign keys are in red.
- Foreign key relations or data flow is given by the red line arrow.

Table Design

user

Column	Type	Null	Default	Links to	Comments	Media type
userID *(Primary)*	char(20)	No			Identify each User unequally	
first_name	varchar(50)	Yes	*NULL*		First Name	
last_name	varchar(50)	Yes	*NULL*		Last Name	
gender	char(1)	Yes	*NULL*		Male [M] Female [F] Other [O]	
birthdate	date	Yes	*NULL*		Record Birthdate	
email	varchar(50)	Yes	*NULL*		Record E-mail	
contactNo	char(15)	Yes	*NULL*		Record Contact number	
password	varchar(20)	Yes	*NULL*		User Password	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	userID	10	A	No	

passenger

Column	Type	Null	Default	Links to	Comments	Media type
PNR *(Primary)*	char(20)	No			Unique id for each Passenger for every journey	
userID *(Primary)*	char(20)	No		user -> userID	Identify USER	
reservation_status	varchar(5)	Yes	*NULL*		Status of Ticket Reservation	
seat_no	varchar(10)	Yes	*NULL*		Seat Number	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	PNR	0	A	No	
				userID	0	A	No	
fk_passenger_user	BTREE	No	No	userID	0	A	No	

ticket

Column	Type	Null	Default	Links to	Comments	Media type
ticket_no *(Primary)*	varchar(20)	No			Ticket Number	
PNR	char(20)	No		passenger -> PNR	Refer Passenger and Journey	
journeyDate	date	No		train_timing -> journeyDate	Refer Journey Date	
train_no	char(10)	No		train_timing -> train_no	Refer respective train	
price	int(11)	Yes	*NULL*		Total fair for traveling (depends upon Journey time)	
bookingDateTime	datetime	Yes	*NULL*		Record when the ticket has been confirmed	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	ticket_no	0	A	No	
fk_ticket_passenger1_idx	BTREE	No	No	PNR	0	A	No	

fk_ticket_train_timing1_idx	BTREE	No	No	journeyDate	0	A	No	
				train_no	0	A	No	

train

Column	Type	Null	Default	Links to	Comments	Media type
train_no *(Primary)*	char(10)	No			Uniquely Identify trains	
train_name	varchar(50)	Yes	*NULL*		Name for each train	
startStation	varchar(50)	Yes	*NULL*		Starting of Journey route	
lastStation	varchar(50)	Yes	*NULL*		Last stop of Journey route	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	train_no	6	A	No	

train_timing

Column	Type	Null	Default	Links to	Comments	Media type
train_no *(Primary)*	char(10)	No		train -> train_no	Refer Train	
journeyDate *(Primary)*	date	No			Refer Journey Date	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	journeyDate	16	A	No	
				train_no	33	A	No	
fk_train_taming_train1	BTREE	No	No	train_no	16	A	No	

train_routes

Column	Type	Null	Default	Links to	Comments	Media type
route_no *(Primary)*	char(10)	No			Uniquely Identifies each route on which a train moves	
train_no *(Primary)*	char(10)	No		train -> train_no	Refers train	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	route_no	6	A	No	
				train_no	6	A	No	
fk_train_routes_train1_idx	BTREE	No	No	train_no	6	A	No	

station

Column	Type	Null	Default	Links to	Comments	Media type
route_no	char(10)	No		train_routes -> route_no	Route Identifier	
station_name	varchar(50)	Yes	*NULL*		All stations under each routes	
arrivalTime	time	Yes	*NULL*		Maintains Journey timeline for forward Journey	
departureTime	time	Yes	*NULL*		Maintains timeline over return path	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
fk_station_train_routes1_idx	BTREE	No	No	route_no	13	A	No	

train_specifications

						type
train_no *(Primary)*	char(10)	No		train -> train_no	Refers train	
seat	int(11)	Yes	*NULL*		Seat capacity [can be though of for each; sleeper, AC, non-AC]	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	train_no	6	A	No	

train_status

Column	Type	Null	Default	Links to	Comments	Media type
train_no *(Primary)*	char(10)	No		train_timing -> train_no	Refer train	
journeyDate *(Primary)*	date	No		train_timing -> journeyDate	Refer Journey Date	
seat_availibility	int(11)	Yes	*NULL*		Current status of seats available	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	train_no	16	A	No	
				journeyDate	33	A	No	

Procedures

+

Procedures			
No.	Procedure Name	Description	Parameters
1.	creatingUserProfile	It inserts user data in user table.	first_name VARCHAR(50) last_name VARCHAR(50) gender CHAR(1) birthdate DATE email VARCHAR(50) contactNo CHAR(15) password VARCHAR(20)
2.	bookTicket	It qualifies user as passenger and book ticket for passenger.	userID CHAR(20) train_no VARCHAR(20) journeyDate DATE departureTime TIME arrivalTime TIME
3.	cancelTicket	It cancels ticket	ticketNo VARCHAR(20) PNR CHAR(20)
4.	Train_enquiry	It fetch train data	

		according to customer journey details.	boarding_station VARCHAR(50) destination_station VARCHAR(50) journey_date DATE
5.	validatingContactNO.	It checks contact number is valid or not. (10 digit)	contactNo CHAR(15)
6.	validatingUser	It checks for user is existing in current database or not and also checks password if user already exists.	userID VARCHAR(20) password VARCHAR(20)

Code

1. creatingUserProfile

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `creatingUserProfile`(
	first_name VARCHAR(50),
	last_name VARCHAR(50),
	gender CHAR(1),
	birthdate DATE,
	email VARCHAR(50),
	contactNo CHAR(15),
	password VARCHAR(20)
)
BEGIN
		call validatingContactNo(contactNo);

		INSERT INTO user values(
			userIdGenerator(first_name, last_name),
			first_name,
			last_name,
			gender,
			birthdate,
			email,
			contactNo,
			password
		);

END$$
DELIMITER ;
```

2. bookTicket

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `bookTicket`(IN `userID`
CHAR(20), IN `train_no` VARCHAR(20), IN `journeyDate` DATE, IN
`departureTime` TIME, IN `arrivalTime` TIME)
BEGIN

    DECLARE PNR char(20);
```

```
    Set PNR = PNRGenerator();

    INSERT INTO passenger VALUES(
            PNR,
            userID,
            'NO',
            seat_no()
    );



    INSERT INTO ticket VALUES(
       ticketNoGenerator(),
                                  PNR,
            journeyDate,
            train_no,
            rentCalculator(departureTime, arrivalTime),
            NOW()
    );
END$$
DELIMITER ;
```

3. cancelTicket

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `cancelTicket`(ticketNo
VARCHAR(20), PNR CHAR(20))
BEGIN
    DELETE FROM ticket
    WHERE ticket_no = ticketNo;

    DELETE FROM passenger
    WHERE PNR = PNR;
END$$
DELIMITER ;
```

4. Train_enquiry

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `train_enquiry`(
                boarding_station VARCHAR(50),
```

```
    destination_station VARCHAR(50),
    journey_date DATE
)
BEGIN

SELECT train_no as train_num, (SELECT train_name from train where
train_no=train_num) as train, route_no, startStation_1, arriving_time,
endStation_2, destination_time, rent, journeyDate from train_timing INNER JOIN

(SELECT train_no, route_no, startStation_1,
if(A_time_1<A_time_2,A_time_1,D_time_1) as arriving_time, endStation_2,
if(A_time_1>A_time_2,D_time_2, A_time_2) as destination_time,
rentCalculator(if(A_time_1>A_time_2,D_time_2, A_time_2),
if(A_time_1<A_time_2,A_time_1,D_time_1)) AS 'rent' from train_routes
INNER JOIN
((SELECT route_no, t1.station_name as startStation_1, t1.arrivalTime as A_time_1,
t1.departureTime as D_time_1, t2.station_name as endStation_2, t2.arrivalTime as
A_time_2, t2.departureTime as D_time_2 from
        (SELECT route_no, station_name, arrivalTime, departureTime from station
where station_name = boarding_station) as t1            INNER JOIN
        (SELECT route_no, station_name, arrivalTime, departureTime from station
where station_name = destination_station) as t2
        using(route_no)) as t3)
            using (route_no)) as temp_name using(train_no) WHERE
journeyDate=journey_date;

END$$
DELIMITER ;
```

5. validatingContactNO

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `validatingContactNo`(
        contactNo CHAR(15)
)
BEGIN
        IF  length(contactNo) != 10 THEN
                SIGNAL SQLSTATE '22003'
                        SET MESSAGE_TEXT = 'Invalid contact number. Only Enter 10
digit number.';
```

```
        END IF;
END$$
DELIMITER ;
```

6. validatingUser

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` PROCEDURE `validatingUser`(IN `userID`
VARCHAR(20), IN `password` VARCHAR(20))
BEGIN
        IF (userID in (SELECT userID from user) ) then
                SELECT 'User is valid.';
        else
                SELECT 'User is not valid.';
  END IF;

END$$
DELIMITER ;
```

Functions

Functions				
No.	Function Name	Description	Parameters	Return
1.	rentCalculator	It calculate rent from total journey time. (Keeping in mind scope of the project, we calculated rent based on journey time instead of distance)	departureTime TIME arrivalTime TIME	rent INT
2.	userIDgenerator	It generate unique user ID from first and last name of user	first_name VARCHAR(50) last_name VARCHAR(50)	userID char(20)
3.	PNRGenerator	It generate unique PNR NO	No parameters	PNR char(20)
4.	ageCalculator	It calculutes age from birthdate	birthdate DATE	age INT
5.	seat_no	It allotes seat to passenger at the time of ticket booking from available seats in train.	trainNo CHAR(10) journey_Date Date	Seat_no INT
6.	ticketNoGenerator	It generate unique ticket no	No parameters	ticket_no char(20)

Triggers					
No.	Trigger Name	Description	Operation Table	Target Table	Operation constraint
1.	track_user_on_sinUp	It keeps a record in user_activity table when user sign up.	user	user_activities	after insert
2.	track_user_updation	It keeps a record in user_activity table when user updates profile.	user	user_activities	after update
3.	track_user_on_delete	It keeps a record in user_activity table when user deletes profile.	user	user_activities	after delete
4.	update_seat_availibility	It updates ticket availability in train status table after passenger successfully books ticket.	ticket	train_status	after insert
5.	updating_reservation_status	It updates reservation status to YES in passenger table after passenger successfully books ticket.	ticket	passenger	after insert

Code

1. rentCalculator

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION
`rentCalculator`(`departureTime` TIME, `arrivalTime` TIME) RETURNS int(11)
   READS SQL DATA
BEGIN
        DECLARE rent INT DEFAULT 0;
   DECLARE timeInMinutes INT;

   SET timeInMinutes = TIMESTAMPDIFF(minute, arrivalTime,  departureTime) ;
   SET rent  = timeInMinutes * 2;
        RETURN ABS(rent);
END$$
DELIMITER ;
```

2. userIDgenerator

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `userIdGenerator`(first_name
VARCHAR(50),
      last_name VARCHAR(50)
) RETURNS char(20) CHARSET utf8
   READS SQL DATA
BEGIN
        DECLARE counter INT DEFAULT 0;
   DECLARE userID CHAR(20);
   SET counter = (SELECT COUNT(*) FROM user) + 1;
   SET userID =
CONCAT('UID',TRIM(LEFT(first_name,1)),TRIM(LEFT(last_name,1)),CONVERT(coun
ter, CHAR) );
        RETURN userID;
END$$
DELIMITER ;
```

3. PNRGenerator

```
DELIMITER $$
```

```
CREATE DEFINER=`root`@`localhost` FUNCTION `PNRGenerator`() RETURNS
char(20) CHARSET utf8
   READS SQL DATA
BEGIN
        DECLARE counter INT DEFAULT 0;
   DECLARE PNR CHAR(20);
   SET counter = (SELECT COUNT(*) FROM passenger) + 1;
   SET PNR = CONCAT('PNR00000', CONVERT(counter, CHAR) );
        RETURN PNR;
END$$
DELIMITER ;
```

4. ageCalculator

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `ageCalculator`(birthdate
DATE) RETURNS int(11)
   READS SQL DATA
BEGIN
        DECLARE age INT DEFAULT 0;
   SET age = ROUND((DATEDIFF(CURDATE(), birthdate) / 365), 0);
        RETURN age;
END$$
DELIMITER ;
```

5. seat_no

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `seat_no`(trainNo CHAR(10),
      journey_Date Date
) RETURNS int(11)
   READS SQL DATA
BEGIN
        DECLARE seat_no INT DEFAULT 0;
   SET seat_no = (SELECT seat_availibility FROM train_status as ts WHERE
ts.train_no = trainNo  and ts.journeyDate = journey_Date);
        RETURN seat_no;
END$$
DELIMITER ;
```

6. ticketNoGenerator

```
DELIMITER $$
CREATE DEFINER=`root`@`localhost` FUNCTION `ticketNoGenerator`()
RETURNS varchar(20) CHARSET utf8
    READS SQL DATA
BEGIN
        DECLARE counter INT DEFAULT 00000000;
    DECLARE ticket_no VARCHAR(20);
    SET ticket_no = CONCAT('T0000',CONVERT(counter, CHAR) );
    SET counter = (SELECT COUNT(*) FROM ticket) + 1;
        RETURN ticket_no;
END$$
DELIMITER ;
```

Triggers

We have created 5 triggers…

Code

1. track_user_on_sinUp

```
CREATE TRIGGER `track_user_on_sinUp` AFTER INSERT ON `user`
 FOR EACH ROW BEGIN
        INSERT INTO user_activities VALUES(NEW.userID, NOW(),
        'Created_Profile');
END
```

2. track_user_updation

```
CREATE TRIGGER `track_user_updation` AFTER UPDATE ON `user`
 FOR EACH ROW BEGIN
        INSERT INTO user_activities VALUES(NEW.userID, NOW(),
        'Updated_Profile');
END
```

3. track_user_on_delete

```
CREATE TRIGGER `track_user_on_delete` AFTER DELETE ON `user`
 FOR EACH ROW BEGIN
        INSERT INTO user_activities VALUES(OLD.userID, NOW(),
        'Deleted_Profile');
END
```

4. update_seat_availibility

```
CREATE TRIGGER `update_seat_availibility` AFTER INSERT ON `ticket`
 FOR EACH ROW BEGIN
        UPDATE train_status
   SET seat_availibility = seat_availibility - 1
   WHERE train_no = NEW.train_no;
END
```

5. updating_reservation_status

```
CREATE TRIGGER `updating_reservation_status` AFTER INSERT ON
`ticket`
 FOR EACH ROW BEGIN
        UPDATE passenger as p
   SET reservation_status = 'YES'
   WHERE p.PNR = NEW.PNR;
END
```

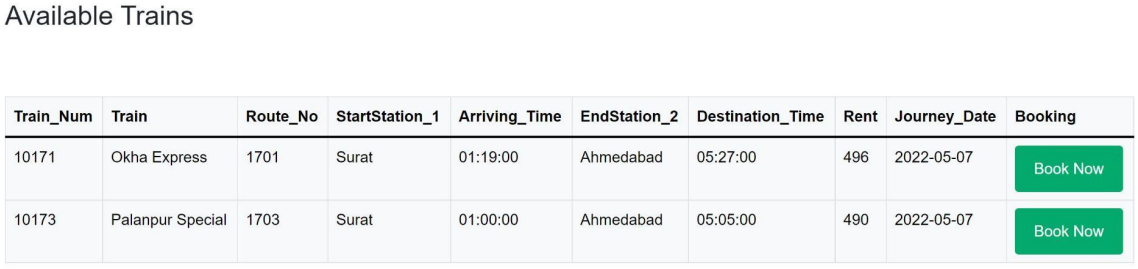
FRONTEND WORK:

Page No:1

Our first page allows the user to select his/her departing station and arrival station along with the date of travel. The page has been created with the help of HTML Forms and CSS for styling along with PHP for establishing connectivity between web pages.

On submitting the form it directs you to a webpage with available trains as shown below

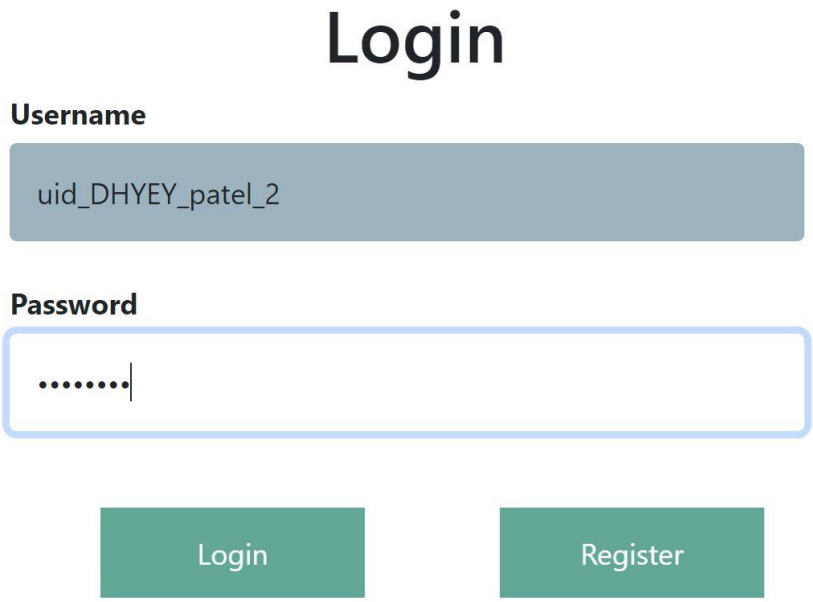
Page No: 2



In this example according to the selected stations, a query is fired that generates the following information about the following trains available from the MySQL database in the backend and a book now to select the particular train.

Page No: 3

I



If you are an existing user of the application then only the login details are required to be filled and then it generates your ticket. Otherwise, you have to click on the register if you are a first-time user.

Passenger Details

First Name

Your name..

Last Name

Your last name..

Gender

M

Birth Date

dd-mm-yyyy

Your Email ID....

Contact No

Password

Submit

After the Login Process is done then your ticket is generated as shown below:

Ticket Details
FirstName: Dhyey
LastName: Patel
Journey Date: 2022-05-07
Train : Okha Express
Start Station: Surat
End Station: Ahmedabad
Rent: 496/-
Have a Happy Journey!

Languages and Technicalities

HTML and CSS:

For WebPage creation

PHP

To connect the backend with the HTML web pages so as to display as well as input the data into the tables.

XAMPP ControlPanel

We used this so as to have an easy interface for our localhost server setup.

For more data visit this drive folder…

https://drive.google.com/drive/folders/1EMgEHLatq-syI5yzI1JaF7T8jaOE6I_Z?usp=sharing

---

END OF THE DOCUMENT