

Project Report

Database Management Systems Lab (CS 3710)

Online Train Information and Reservation System

Team Members

Kaushal Kishore (111601008)
Sativik Choudhary (111601021)
Himanshu Rai (111601032)

Contents

Contents	2
Acknowledgement	4
Introduction	5
Requirement Specifications	6
ER Diagram	7
Relations	8
station	8
train	8
neighbours	8
path	9
user	9
ticket	10
reservation	10
Schema Diagram	11
Functions	12
station_code_name	12
train_no_name	12
calculate_fare	12
station_code_index	12
check_seat_available	12
Procedures	13
city_stations	13
train_details	13
train_between_stations	13
seat_info	13
waitlisted_status	14
pnr_info	14
booking_history	14
user_info	14
book_ticket	14
cancel_ticket	14

confirm_ticket	14
available_seat_list	14
Triggers	15
check_path_insert	15
check_reservation_insert	15
check_ticket_insert	15
check_ticket_update	15
Roles	16
rl_admin	16
rl_user	16
rl_public	16
Implementation Details	17
Explaining GUI	18
Signup/Login	18
Profile	19
Booking	20
Train Info	21
User Booked Tickets	22
Contributions	23
Kaushal Kishore	23
Satvik Choudhary	23
Himanshu Rai	23
Appendix	24

Acknowledgement

In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this project gives us much pleasure. We would like to show our gratitude to Dr. Sahely Bhadra and Dr. Mrinal Kanti Das, our course instructors, for giving us a good guideline for the project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in completing this project.

Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve the project. We thank all the people for their help directly and indirectly to complete our project.

Kaushal Kishore (111601008)
Satvik Choudhary (111601021)
Himanshu Rai (111601032)

Introduction

The Indian Railways (IR) carries about 5.5 lakhs passengers in reserved accommodation every day. The Computerised Passenger Reservation System (PRS) facilitates the booking and cancellation of tickets from any of the 4000 terminals (i.e. PRS booking window all over the countries). These tickets can be booked or cancelled for journeys commencing in any part of India and ending in any other part, with travel time as long as 72 hours and distance up to several thousand kilometers.

Even though the booking system for Indian Railways is centrally computerised, managing such a huge network can be challenging and hence the Indian Railways decided to introduce online railway reservation to make booking train tickets easier and more convenient. To take advantage of the Indian Railways online railway reservation users need to go to the IRCTC website, in addition to making bookings one can check the status of departing and arriving trains, train itineraries, ticket accessibility, postponing and cancelling railway bookings, make tatkal reservations, Indian Railways time table among others.

The special benefits of using the online railway reservation facility is that passengers can reserve tickets with comfort at their home without having to stand in long queues at railway stations or having to take the trouble to go to a travel agent.

In this project, we will be simulating the railway reservation system and understand the various challenges faced by such a large system and try to come up with efficient solutions for such problems.

Requirement Specifications

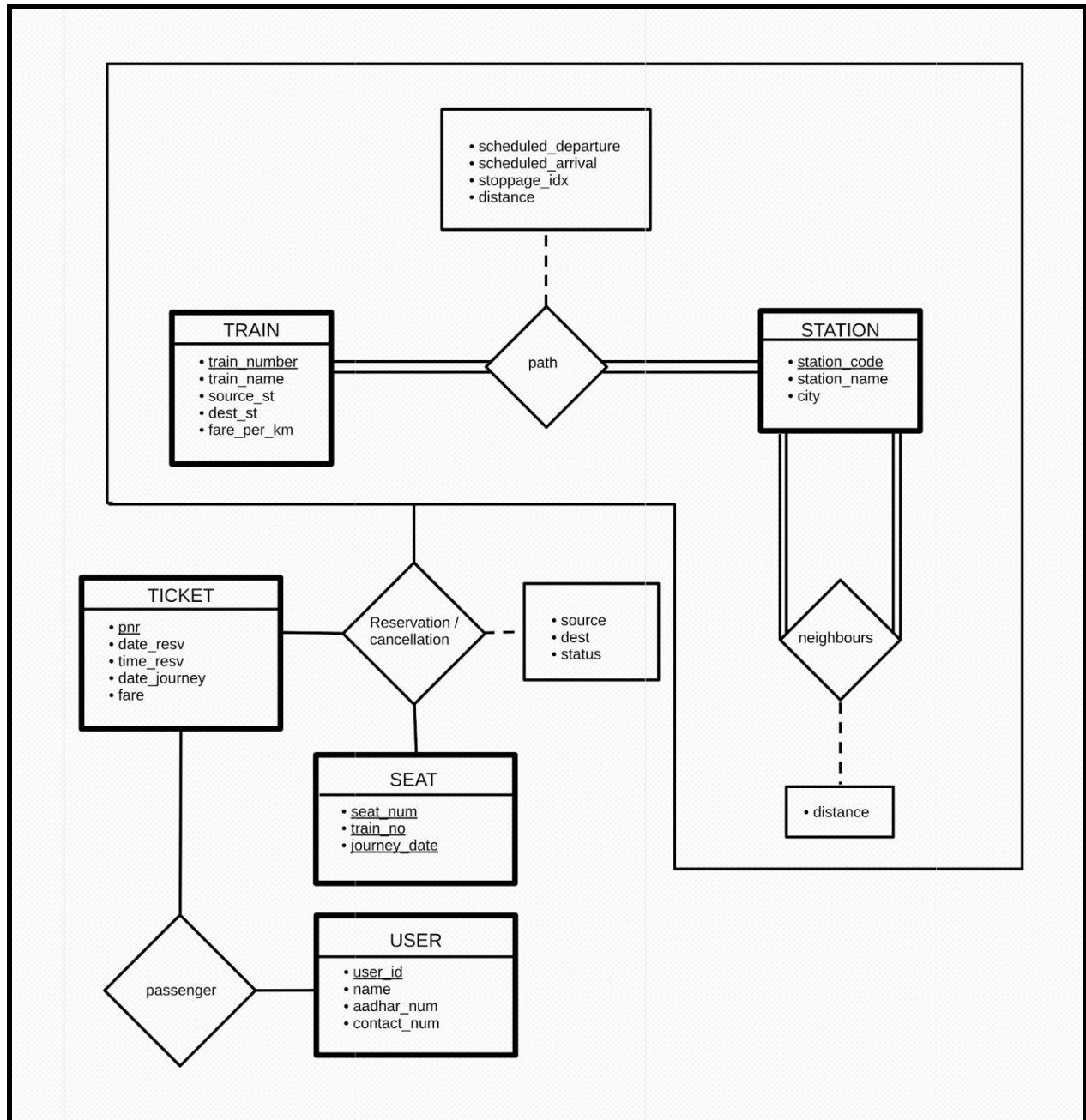
This project aims at development of an Online Railway Reservation Utility which facilitates the Railway customers to get information about currently running trains, manage their reservations online, and the administrators to modify the backend databases in a user-friendly manner.

Anyone can access the system and query about information about trains from one station to the other, timings of a specific train, fare, distance travelled by a train, and also the currently available seats for booking. However, a user cannot book a ticket as such.

The Customers are required to register on the server for getting the privilege to book their ticket. Upon registration, each user gets an account which is essentially the 'view level' for the customer. The account contains comprehensive information of the user entered during registration and permits the customer to book ticket, get access to his past reservations, update his account details and also see the status of currently running trains.

The Railway Administrator is the second party in the transactions. The administrator is required to login using a master password, once authenticated as an administrator, one has access and right of modification to the critical information stored in the database at the server. This includes the account information of the customers, attributes and statistics of stations, addition of new trains, description of the train stoppages, all the reservations that have been made, etc.

ER Diagram



Relations

The relations in the database were chosen to minimise the redundancy and at the same time ensuring that the required data is stored and retrieved in an optimal manner. There are a total of seven tables in the database.

- **station**

This table contains information about a particular station. The attributes include station name, station code and city in which the station is located.

- ❖ **Primary key**

- *station_code*



Diagram of the station table structure. The table is titled 'station' and contains three attributes: station_code (varchar(5)), station_name (varchar(50)), and city (varchar(50)). The table is shown in a yellow box with a grid icon in the top left corner. Below the table, the text 'Powered by yFiles' is visible.

station	
station_code	varchar(5)
station_name	varchar(50)
city	varchar(50)

- **train**

This table contains information of a particular train - its name, number, source, destination and fare.

- ❖ **Primary key**

- *train_no*

- ❖ **Foreign keys**

- *source_st* (station(*station_code*))

- *dest_st* (station(*station_code*))

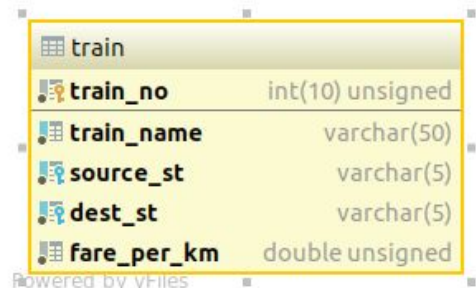


Diagram of the train table structure. The table is titled 'train' and contains five attributes: train_no (int(10) unsigned), train_name (varchar(50)), source_st (varchar(5)), dest_st (varchar(5)), and fare_per_km (double unsigned). The table is shown in a yellow box with a grid icon in the top left corner. Below the table, the text 'Powered by yFiles' is visible.

train	
train_no	int(10) unsigned
train_name	varchar(50)
source_st	varchar(5)
dest_st	varchar(5)
fare_per_km	double unsigned

- **neighbours**

A tuple, referencing two stations, exist in this relation only if they are neighbour of each other. This table also stores the distance between the neighbouring stations.



Diagram of the neighbours table structure. The table is titled 'neighbours' and contains three attributes: st_a (varchar(5)), st_b (varchar(5)), and distance (double unsigned). The table is shown in a yellow box with a grid icon in the top left corner. Below the table, the text 'Powered by yFiles' is visible.

neighbours	
st_a	varchar(5)
st_b	varchar(5)
distance	double unsigned

❖ **Primary key**

➤ *st_a, st_b*

❖ **Foreign keys**

➤ *st_a* (station(*station_code*))

➤ *st_b* (station(*station_code*))

● **path**

This table stores the route information for all the trains - their scheduled arrival and departure times, the distance they have travelled from the their source stations for all the stoppages are stored in this table.

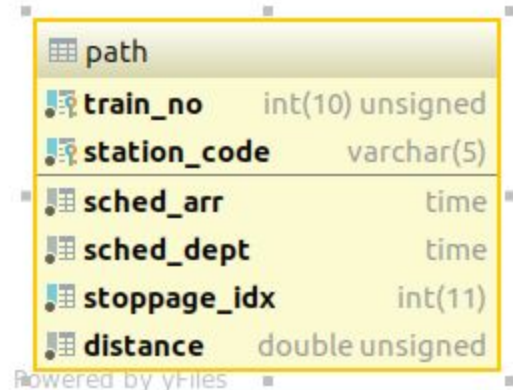
❖ **Primary key**

➤ *train_no, station_code*

❖ **Foreign keys**

➤ *train_no* (train(*train_no*))

➤ *station_code* (station(*station_code*))



path	
train_no	int(10) unsigned
station_code	varchar(5)
sched_arr	time
sched_dept	time
stoppage_idx	int(11)
distance	double unsigned

Powered by yFiles

● **user**

A user cannot make any reservations unless he has signed up. This table stores the information about users who have signed up for making reservations. It stores the user's name, userid, password (used to login for making reservations), his contact and aadhaar number.

❖ **Primary key**

➤ *User_id*

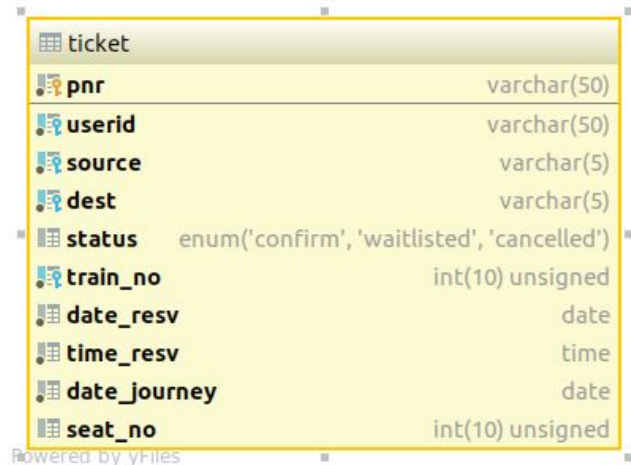


user	
userid	varchar(50)
password_hash	varchar(255)
name	varchar(100)
aadhar_no	bigint(20) unsigned
contact_no	varchar(20)

Powered by yFiles

- **ticket**

This table stores data of all the booked tickets - userid of the user who booked the ticket, train for which the reservation was made, pnr number of the ticket, date and time of booking, date of journey, the current status of (confirmed, waiting or cancelled) and the seat number (if the seat is confirmed).



Powered by yFiles

ticket	
pnr	varchar(50)
userid	varchar(50)
source	varchar(5)
dest	varchar(5)
status	enum('confirm', 'waitlisted', 'cancelled')
train_no	int(10) unsigned
date_resv	date
time_resv	time
date_journey	date
seat_no	int(10) unsigned

- ❖ **Primary key**

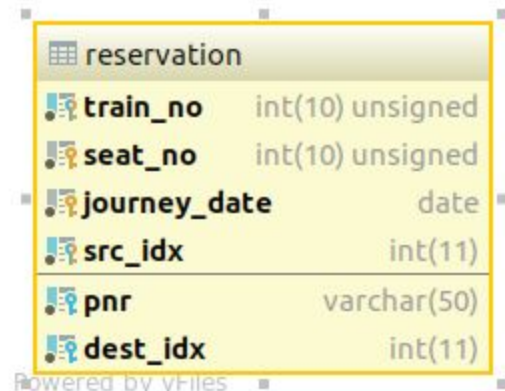
- *pnr*

- ❖ **Foreign keys**

- *userid* (*user(userid)*)
 - *source* (*station(station_code)*)
 - *dest* (*station(station_code)*)
 - *train_no* (*train*)

- **reservation**

This is the last table in our database. This table stores the reservation info of each train - which seats are booked, on which day and from which station to which station.



Powered by yFiles

reservation	
train_no	int(10) unsigned
seat_no	int(10) unsigned
journey_date	date
src_idx	int(11)
pnr	varchar(50)
dest_idx	int(11)

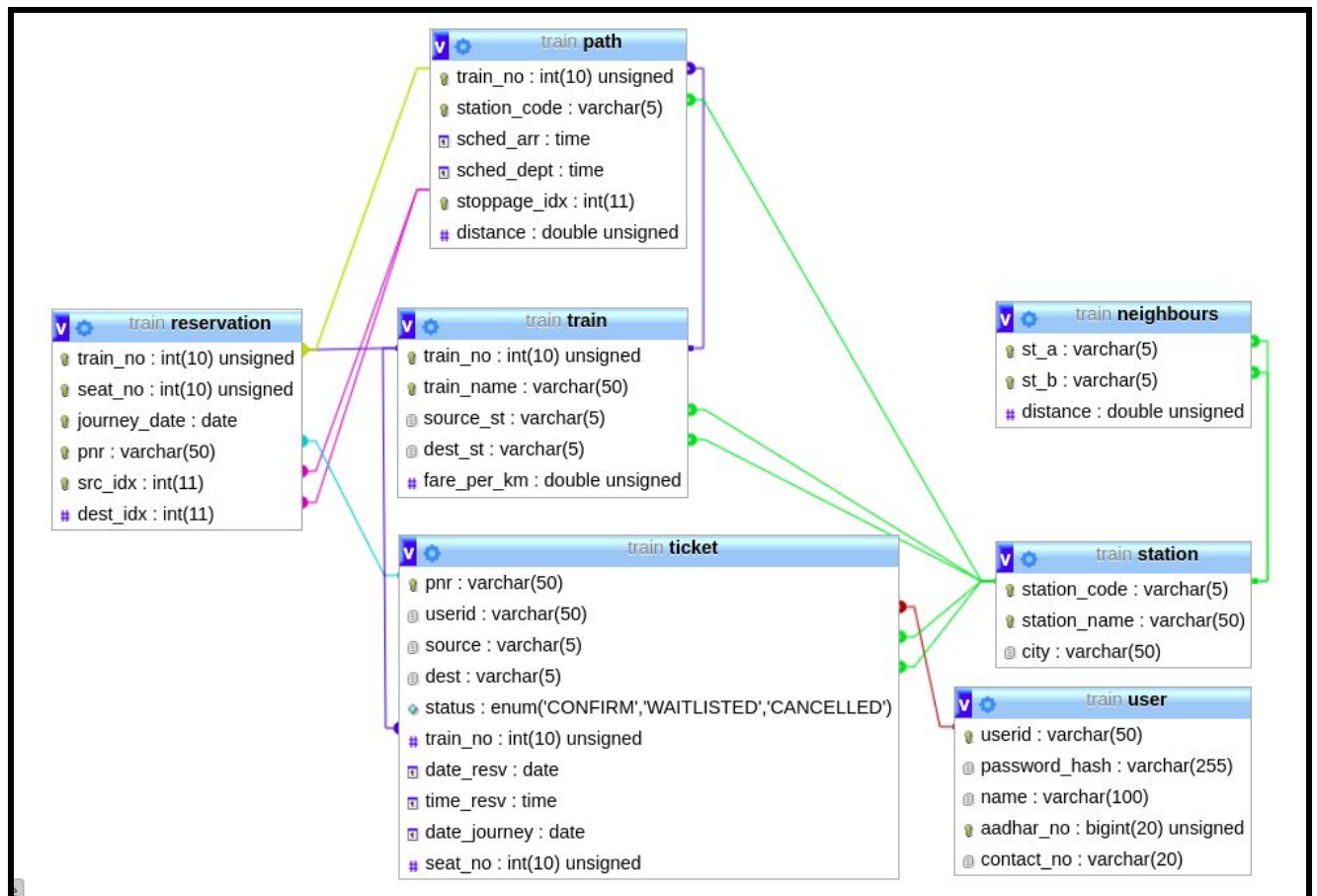
- ❖ **Primary key**

- *train_no, seat_no, src_idx, journey_date*

- ❖ **Foreign keys**

- *train_no, src_idx* (*path(train_no, stoppage_idx)*)
 - *train_no, dest_idx* (*path(train_no, stoppage_idx)*)

Schema Diagram



Functions

The following functions are defined, most of them are for the ease of implementation of graphical interface or are used somewhere in other functions and procedures. All the functions are written with various checks and signals errors whenever some illegal action is being performed through them.

- **station_code_name**
This function returns the station name from a given station code by querying *station* table.
- **train_no_name**
This function returns the train name from its number by querying *train* table.
- **calculate_fare**
This function calculates the total fare for a given combination of train, source and destination stations. For this it queries the *path* table.
- **station_code_index**
This function gives the stoppage index of a given station for a given train. For this it queries the *path* table.
- **check_seat_available**
This function checks whether a given seat is available (not yet booked) for a given train on a given date for given pair of source and destination stations. For this it accesses the reservation table.

Procedures

The following procedures are defined, most of them are for the ease of implementation of graphical interface or are used somewhere in other functions and procedures. All the functions are written with various checks and signals errors whenever some illegal action is being performed through them.

- **city_stations**

This procedure returns the station name and code of all the stations from a given city. For this it uses *station* table.

- **train_details**

This procedure displays all the details of a particular train. These details include station name, code through which the train travels, the scheduled arrival and departure at these stations and also the cumulative distance travelled by the train so far in its journey. For this it accesses the *path* table.

- **train_between_stations**

This procedure gives the list of all available trains between two given stations, the distance travelled by the train and also its departure and arrival time at the two stations respectively. For this it accesses the *path* table.

- **seat_info**

This procedure gives all the bookings corresponding to a given train, seat and date combination, that is from which station to which station it is booked and the pnr of the ticket through which it is booked. For this it accesses the *path*, *reservation* tables.

- **waitlisted_status**

This procedure gives the list of pnr, source and destination of all the waitlisted tickets for a given train, date and seat combination. Like the previous procedure it also accesses the *path*, *reservation* tables.

- **pnr_info**

This procedure returns all the info about a given pnr stored in the *ticket* table.

- **booking_history**

This procedure returns all the previous bookings done by a given user, with the help of *ticket* table.

- **user_info**

This procedure gives the details of a user corresponding to a given userid, from the *user* table.

- **book_ticket**

This procedure books a given seat of a train for a user, for a given date of journey. For this it inserts a record in the *ticket* table.

- **cancel_ticket**

This procedure cancels a previously booked ticket for a given pnr by updating the *ticket* table.

- **confirm_ticket**

This procedure confirms a previously waitlisted ticket to a given seat corresponding to a given pnr.

- **available_seat_list**

This procedure returns the list of seats that are available for a certain journey. For this it uses the table *seat list* and the function *check_seat_available*.

Triggers

The following triggers four triggers are used. Some of them check constraint satisfaction upon table update and other updates other tables when a table is updated.

- **check_path_insert**

This trigger is for insert on *path* table. It checks whether a train whose data is being inserted into *path* table already exist in *train* table and also the data of the station (if any) that comes before (in train's route) the station for which this new data is being inserted, exist in the *path* table. If so, it automatically updates the distance attribute for the new record based on the already entered data, with the help *neighbour* table.

- **check_reservation_insert**

This trigger is for insert on *reservation* table. It verifies that the new seat being reserved is not already reserved. If not so, it signals an error.

- **check_ticket_insert**

This trigger is for insert on *ticket* table. It updates the new record being inserted with the time and date of reservation (by querying system date and time), its status to waiting if applicable (default is confirmed) and then also inserts the corresponding seat data in the *reservation* table if a ticket is booked confirmed.

- **check_ticket_update**

This trigger is for update on *ticket* table. Whenever a confirmed ticket is cancelled or a waitlisted ticket is confirmed, it deletes or inserts corresponding data in the *reservation* table.

Roles

There are three roles in the system as specified below.

- **rl_admin**

This is the role given to admin users. The users having this role gets all the valid permissions on the database. They can modify any data, add new data, delete any old data and also has the permission to modify the existing system design.

- **rl_user**

This role is given to a user who has currently logged in with a valid account which was earlier signed up using the GUI. Apart from general permissions to query a table, this role also gives a user authority to insert and update data into *ticket* and *reservation* table. This means that user is able to book a new ticket or cancel his previously booked ticket.

- **rl_public**

This role is given to an unlogged user, who is just using the visual interface. He basically has the permissions to query on the tables. So, he can get information about the trains between the stations, the timings of a particular train, fare for a given journey, seats availability to book tickets etc.

Implementation Details

The train information system part of the project is nothing more than querying on the tables already stored in the database. However, the reservation system involves a bit of complexity.

To implement the reservation system, we have two tables - *ticket* and *reservation*. In the *ticket* table we store information mainly regarding a ticket - the date, time of booking and journey, userid of the user who booked the ticket, ticket status and seat if any allotted. In the *reservation* table, we store information regarding seats - which seats are booked on which date and from which station to which station. So, now if we have a new request to book a ticket on a particular date from a source to destination for a train - we can query on the reservation table and check which seats are already allotted for some or whole part of this journey. Those seats can't be used for the current booking. The remaining seats if any are available for bookings. In case if no seats are available the system books the ticket but keeps its status as waiting. If later some ticket gets cancelled it automatically checks if there are some ticket with waiting status and can be confirmed, if found updates their status to confirmed with correspondingly allotted seat number.

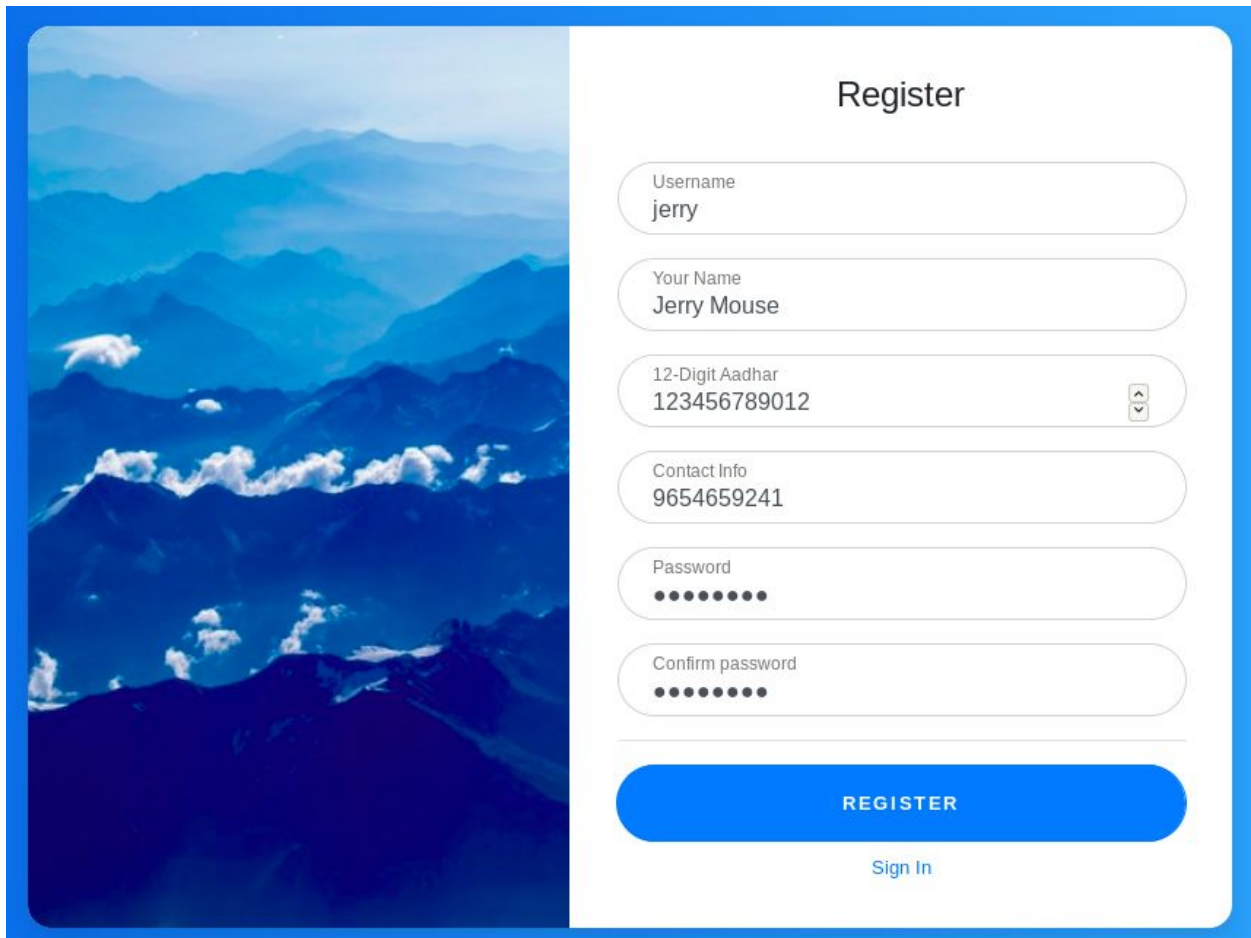
Also when a ticket is booked or gets confirmed its data is also inserted in the *reservation* table along with that in the *ticket* table. Similarly, when a ticket is cancelled, its status is updated in the *ticket* table and corresponding entry being deleted from the reservation table.

Explaining GUI

- **Signup/Login**

To support multiple users at the same time while doing signup we are using transactions with the isolation level as serializable. This is to avoid instances of having duplicate username instances when 2 users try to use it at the same time.

For every user who registers with the site, is also created as a DBMS user. By doing that saves us from implementing our own privilege system and creating/saving passwords or their hashes. This has benefits related to security of the website.



Register

Username
jerry

Your Name
Jerry Mouse

12-Digit Aadhar
123456789012

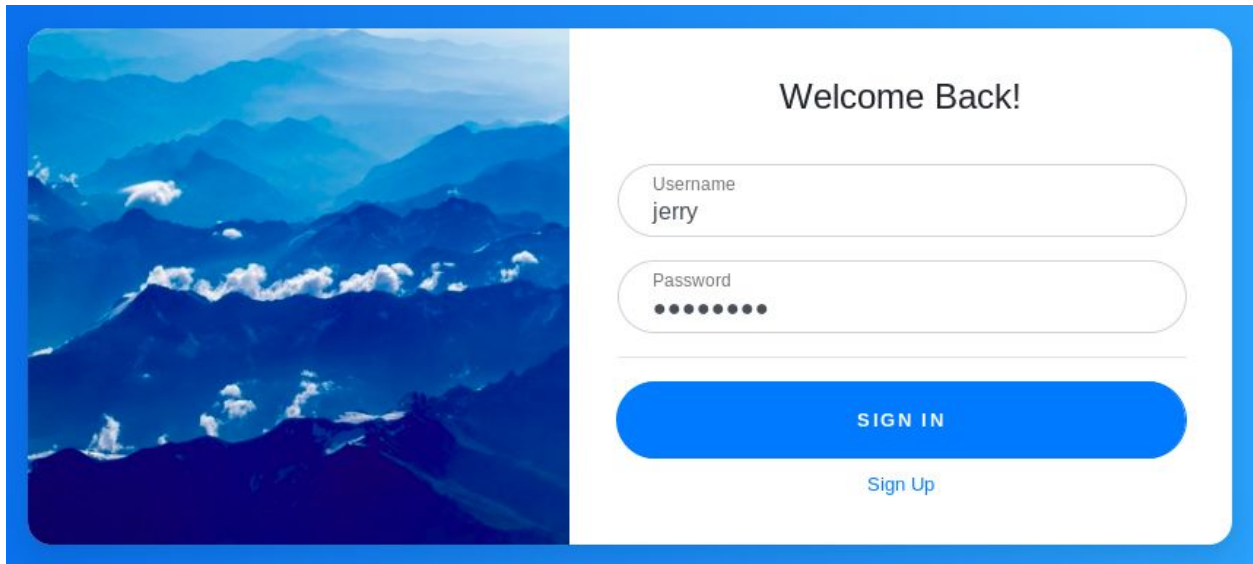
Contact Info
9654659241

Password
●●●●●●●●

Confirm password
●●●●●●●●

REGISTER

[Sign In](#)



Welcome Back!

Username
jerry

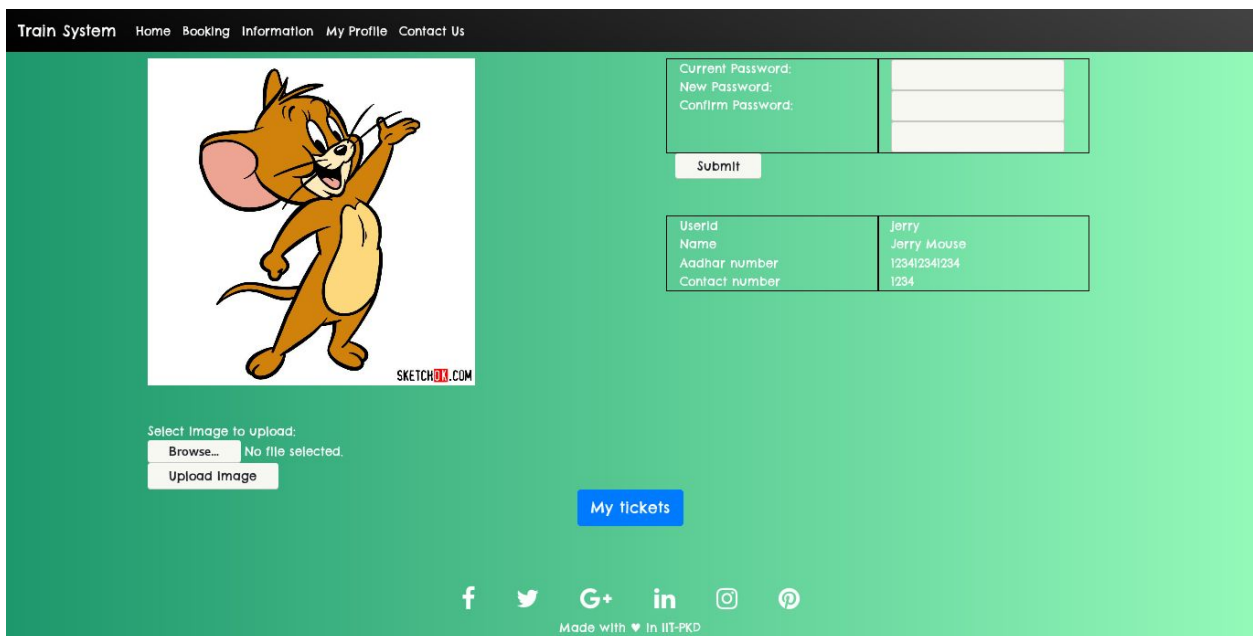
Password
●●●●●●●●

SIGN IN


[Sign Up](#)

- **Profile**

We have two forms that link to two different php scripts. One is for uploading user's profile photo and the other one is for changing current password. In the page we show the users other specific details like Aadhar number, userid, name etc. but the users are not allowed to modify them. Clicking on my tickets opens a list of all the bookings of the user including cancelled tickets.



Train System Home Booking Information My Profile Contact Us



SKETCHOXY.COM

Select Image to upload:

Browse... No file selected.

Upload Image

Current Password:

New Password:

Confirm Password:

Submit

UserId	Jerry
Name	Jerry Mouse
Aadhar number	123412341234
Contact number	1234

My tickets

f t G+ in o p

Made with ♥ In IIT-PKD

- **Booking**

We first take 3 inputs from the user source, destination and date. Then we show a list of trains with all the relevant information required while booking. Then the users can book any number of tickets from 1 to 20. If a seat is available for a given journey then we insert the data in the ticket and reservation table with status 'CONFIRMED' otherwise we enter the data in only ticket table with status 'WAITLISTED' and this can get confirmed when some other ticket is cancelled and allows this ticket to get confirmed.

Train System

Home

Booking

Information

My Profile

Contact Us

New train started by Railway minis

Train Ticket Booking

Find trains

Train Info

Source: Ahmedabad

Destination: Ahmedabad

Dept. Date: 29 / 03 / 2019

Submit Query

Find Trains

Trains Running on: 2019-04-07

TrainNo.	TrainName	SchedDept	SchedArr	Dist.	Fare	Avail. Seats	Qty.
12457	TVC-CAPE-MAS Superfast Express	11:00:00	15:10:00	405	931	20	<div>4</div> <div>✓ GetNow</div>

f

🐦

G+

in

📷

📌

Made with ♥ in IIT-PKD

- **Train Info**

This is just for information purpose, basically the user selects a train in the drop down menu and we show all the details regarding the train as shown in the image.

[Train System](#) [Home](#) [Booking](#) [Information](#) [My Profile](#) [Contact Us](#)

New train stop

Train Ticket Booking

[Find train](#) **Train Info**

Train Name: 12698 - TVC-MAS Express

Submit Query

Get Train Info

StopIdx	StnCode	StnName	SchedArr	SchedDept	Dist
1	TVC	Thiruvananthapuram Central	03:30:00	03:50:00	0
2	PGT	Palakkad Junction	07:15:00	07:20:00	356
3	CBE	Coimbatore Junction	07:45:00	07:50:00	411
4	ED	Erode Junction	08:45:00	08:50:00	493
5	MAS	Chennai Central	11:55:00	12:25:00	888

[f](#) [t](#) [G+](#) [in](#) [@](#) [p](#)

Made with ♥ in IIT-PKD

- **User Booked Tickets**

We show all the tickets booked by the user and for the tickets that are confirmed or are in waiting list can be cancelled by the user.

When we cancel a ticket we also give opportunity to waitlisted tickets on that given day to get confirmed. So we check in a loop and try to confirm as much waitlisted tickets as we can. We give priority on the basis of booking time, so the earlier booked tickets get confirmed first.

Train System Home Booking Information My Profile Contact Us										
pnr	train_no	source	dest	date_resv	time_resv	date_journey	status	seat_no	cancel	
201904061245701200339	12457	CAPE	TPJ	2019-03-28	20:55:39	2019-04-06	CONFIRM	1	OK	
201903311245714000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	14	OK	
201903311245713000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	13	OK	
201903311245712000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	12	OK	
201903311245711000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	11	OK	
201903311245710000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	10	OK	
201903311245709000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	9	OK	
201903311245708000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	8	OK	
201903311245707000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	7	OK	
201903311245706000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	6	OK	
201903311245705000325	12457	CAPE	TPJ	2019-03-28	00:23:25	2019-03-31	CONFIRM	5	OK	
201903311245704230317	12457	CAPE	TPJ	2019-03-28	00:19:13	2019-03-31	CONFIRM	4	OK	
201903311245703230317	12457	CAPE	TPJ	2019-03-28	00:19:13	2019-03-31	CONFIRM	3	OK	
201903311245702230317	12457	CAPE	TPJ	2019-03-28	00:19:13	2019-03-31	CONFIRM	2	OK	
201903311245701230317	12457	CAPE	TPJ	2019-03-28	00:19:04	2019-03-31	CONFIRM	1	OK	

Contributions

- **Kaushal Kishore**

Decided the specifications of the project, built ER-diagram, verified the working of the train information system, helped in the implementation of the reservation system, created sample data for the tables, implemented procedures and helped in implementing triggers, helped in deciding the basic user interface - fully implemented signup/login frontend and backend, booking frontend, and cancelling frontend, generated figures for the report and verified the report. Implemented all ajax queries to communicate with the database asynchronously (it's not very difficult).

- **Satvik Choudhary**

Built ER-diagram, created train information system part of the project, verified the working of the reservation system, designed test cases to check critical parts of the database, implemented functions and helped in implementing procedures, verified the working of triggers, help in deciding the basic user interface (GUI part) - created frontend for profile and backend of profile, booking and cancelling pages, helped in making the report.

- **Himanshu Rai**

Decided the specifications of the project, helped in the implementation of information and reservation system, created sample data for the project, implemented triggers and helped in implementing functions, verified the working of functions and procedures, decided upon the necessary roles in the database, helped in backend of GUI - booking and cancelling, made most parts of the report.

Appendix

- The source code of this project can be found [here](#).
- The map of sample train network included with the project

