DATABASE MANAGEMENT SYSTEM

Redcan Trains

Giani Zail Singh Campus College Of Engineering And Techlogoly, Bathinda

Submitted to: Er. Manpreet Ma'am

Group Members:

Prahlad Kumar	160280071
Arpit Kumar Singh	170280537

ACKNOWLEDGEMENT

We have made this report file on the project "**Redcan Trains**". We have tried our best to elucidate all the relevant detail to the topic to be included in the report, while in the beginning we have tried to give a general view about this topic. Our efforts have ended on a successful note. I express my sincere gratitude to **Er. Manpreet kaur sidhu**, for giving us this opportunity to develop a project on Web Technology and Application. Without this, it wouldn't have been possible to develop a project in this domain.

TABLE OF CONTENTS

ABSTRACT

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 kilometres.

In the given project we will be developing a SQLite Database which will help users to find train details, enquire about trains running between given two stations, book tickets and know the exact rates of their tickets to the desired destination.

With the help of online booking people can book their tickets online through internet, sitting in their home by a single click of mouse.

The main objective of the project is management of the database of Railway System. This is done by creating database of the trains between various stations, user database, booking database and many more. The database is then connected to main program using interconnection of the program with the database using NodeJS.

To access this Railway Ticket Booking System Project , users have to register by giving their entire details such as their name, full address details, sex, age, date of birth, nationality, mobile number, email id. After successful registration, users will be provided with their login id and password. The Ticket Management System has applicants and administrators.

Ticket Booking Offices are located in various parts of the state and each office is looked after by administrators. Each administrator has a unique identity, name, address, start date of work at an office in particular location.

CHAPTER 1
INTRODUCTION

PROBLEM STATEMENT

Indian Railways (IR) is India's national railway system operated by the Ministry of Railways. It manages the fourth-largest railway network in the world by size, with 121,407 kilometres (75,439 mi) of total track over a 67,368-kilometre (41,861 mi) route. IR runs more than 20,000 passenger trains daily, on both long-distance and suburban routes, from 7,349 stations across India. The trains have a five-digit numbering system. In the freight segment, IR runs more than 9,200 trains daily.

Pseudo Indian Railway Catering and Tourism Corporation is a subsidiary of the Indian Railways that handles the catering, tourism and online ticketing operations of the Indian railways, with around 5,50,000 to 6,00,000 bookings everyday is the world's second busiest. It's tagline is "Lifeline of the nation".

It is known for changing the face of railway ticketing in India. It pioneered internet-based rail ticket booking through its website, as well as from the mobile phones via WiFi, GPRS.In addition to e-tickets, Indian Railways Catering and Tourism Corporation also offers I-tickets that are basically like regular tickets.

In the given project we will be developing a SQL Database which will help users to find train details, enquire about trains running between given two stations, book tickets and know the exact rates of their tickets to the desired destination.

With the help of online booking people can book their tickets online through internet, sitting in their home by a single click of mouse.

The main objective of the project is management of the database of Railway System. This is done by creating database of the trains between various stations,user database,booking database and many more. The database is then connected to main program using interconnection of the program with the database using Django

1.1 Actors

People who interact with database:

- Admin
- User

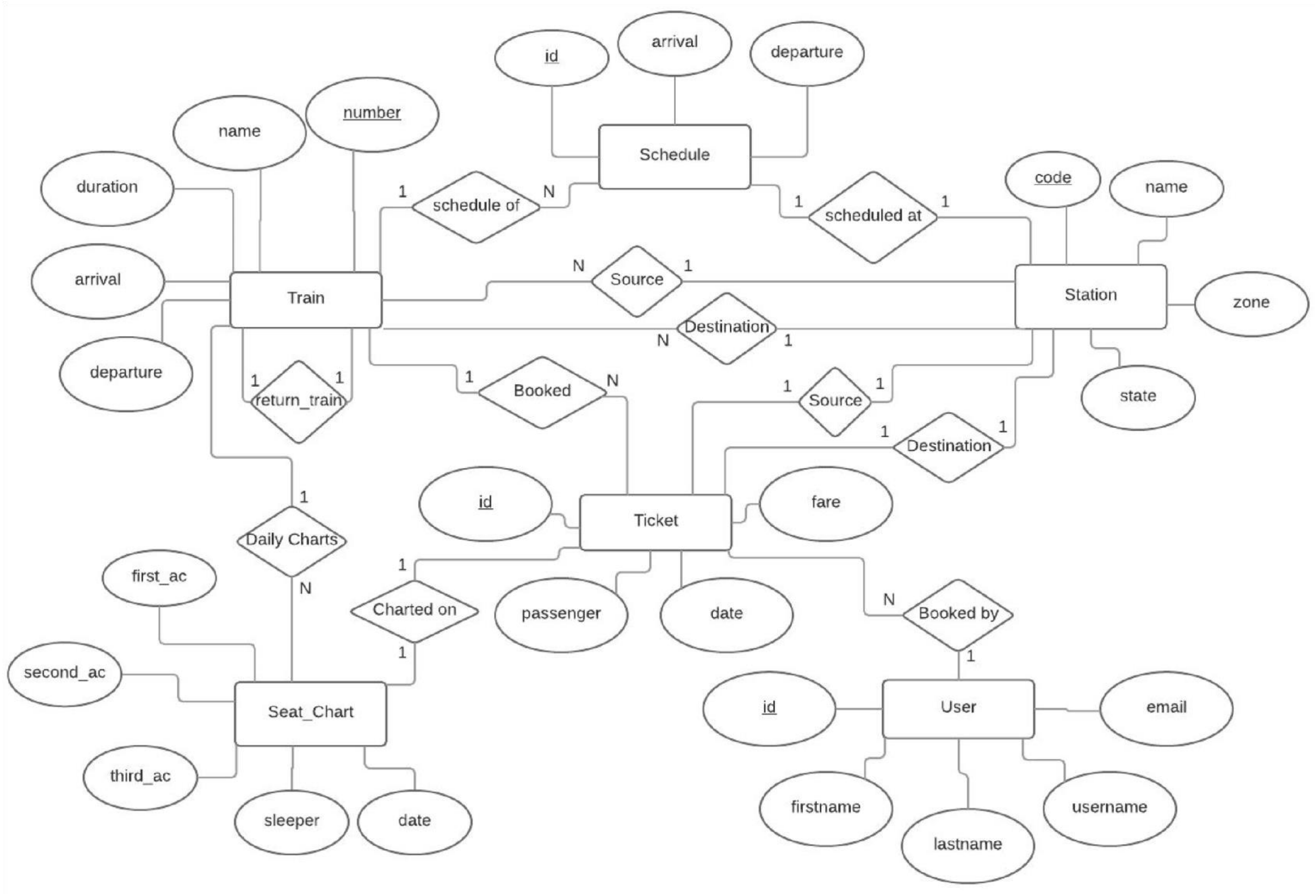
1.2 SOME SAMPLE QUERIES

Different actors have different access to the database:

- A Admin can
    - Check and edit the list of stations in a country ○ Obtain and edit/add trains between stations.
    - View and also edit the tickets of any user.
    - Change/view the schedule for a train
    - Generate chart for a specific train for a given journey
- A User can
    - Get list of ticket that he/she has booked ○ Book multiple tickets.
    - Cancel the ticket he/she has booked

CHAPTER 2
SYSTEM DESIGN

2.1 LOGICAL DESIGN (CONCEPTUAL DESIGN)



2.2 SCHEMA DIAGRAM

Users:

username	email	first_name	last_name	password

Station:

code	state	name	zone	address

Train:

number	source	name	zone	arrival	departure
return_train	dest	duration_h	duration_m	type	distance

Schedule:

train	station	arrival	day	departure

Seat_Chart:

train	date	second_ac	third_ac	sleeper	first_ac

Ticket:

passenger	train	type	chart	user	source	dest	source_schedule	dest_schedule	date

2.3 ADVANCED LOGICAL DESIGN

2.3.1 NORMALIZATION TECHNIQUES

Every candidate key in a table determines all other attributes and no non-key attributes determine any attributes in the tables. Hence all tables are already in 3rd NF.

4th NF is not required since there are no repeated values in any of the tables.

2.3.1 GLOBAL SCHEMA

Users	
Attribute name	**Attribut (type in**
username	char(20)
email	char(20)
first_name	char(10)
last_name	char(10)
password	char(20)

Train	
Attribute name	**Attribute size (type in bytes)**
arrival	char(8)
source	char(10)
name	char(30)
zone	char(10)
number	char(15)

Stations	
Attribute name	**Attribute size (type in bytes)**
state	char(20)
code	char(10)
name	char(30)
zone	char(10)
address	char(50)

Schedule	
Attribute name	**Attribute siz bytes)**
arrival	char(8)
day	int
train	char(15)
station	char(10)
id	int
departure	char(8)

eat_Chart	
Attribute name	**Attribute siz bytes)**
train	char(15)
first_ac	int
second_ac	int
third_ac	int
sleeper	int
date	date

Ticket	
Attribute name	**Attribute size (type in bytes)**
passenger	char(20)
train	char(15)
type	char(2)
chart	int
user	char(20)
source	char(10)
dest	char(10)
source_schedule	int
dest_schedule	int
date	date
fare	int

2.4. QUERIES

2.4.1 <u>Queries to be executed:</u>

1. **User login.**
   SELECT * FROM Users WHERE username = input_name AND password = password;

2. **Search for trains.**

```
SELECT * FROM trainDetails
```

3. **Book Tickets.**

```
INSERT INTO Ticket
(input_name,input_train,input_seat_type,input_chart,user.username,input_source,input_dest,input_sourceSchedule,input_destSchedule, input_date,fare)
```

4. **View previous bookings**

```
SELECT * FROM Ticket  WHERE user = cookie_user;
```

5. Cancel Bookings

```
DELETE FROM Ticket
WHERE user = cookie_user AND id = input_id;
```

2.4.2 VIEWS

```
CREATE view trainDetails AS
SELECT t.name, t.number, s1.name, s2.name, sc1.departure, sc2.arrival, sch.seats
FROM Train t
JOIN  Station s1 on s1.code=input_source
JOIN Station s2 on s2.code=input_dest
JOIN Schedule sc1 on sc1.station=s1.code AND sc1.train=t.number
JOIN Schedule sc2 on sc2.station=s2.code AND sc2.train=t.number
JOIN Seat_Chart sch on t.number=sch.train AND sch.date=input_date
WHERE sc1.departure < sc2.arrival
```

2.4.3 TRIGGERS

1. **For booking tickets:**

```
DROP TRIGGER IF EXISTS book_ticket;
DELIMITER //
CREATE TRIGGER book_ticket AFTER INSERT ON Ticket
FOR EACH ROW
BEGIN
Update Seat_Chart
Set seats=seats-1
WHERE
```

```
train=NEW.train AND date=NEW.date
```

2. **For cancelling tickets:**

```
DROP TRIGGER IF EXISTS cancel_ticket;
DELIMITER //
CREATE TRIGGER cancel_ticket AFTER DELETE ON Ticket
FOR EACH ROW
BEGIN
Update Seat_Chart
Set seats=seats+1
WHERE
train=OLD.train AND date=OLD.date
```

CHAPTER 3 PHYSICAL DESIGN

3.1 ASSUMPTIONS

Number of tuples in each relation

User	1000
Train	2000
Station	10000
Schedule	20000
Seat_Chart	60000
Ticket	80000

3.2 STORAGE REQUIREMENTS: DISK PARAMETERS

Avg Seek Time
Rotational Delay(Latency time)
Block Transfer Time

Block pointer size
Block Size

Following are the assumptions which are considered for storage requirements: ●
Fixed length records are considered for all relations.

- The delimiter for each field is length of the field
- Total number of records in respective relations (provided in below table).
- Block size is 1024 bytes.
- Record doesn’t span over multiple blocks (this can be achieved by taking floor function during calculating number of records per block to restrict single record doesn’t span over blocks).
- Block pointer(Bp) size is 4 bytes
- Average Seek Time(S) is 20 ms irrespective of any site.
- Average Disk rotation time (Latency) Time (L) is 10 ms irrespective of any site.
- Block transfer rate (Tr) is 0.5 ms irrespective of any site.
- **Blocking factor= ceil(Block size / Record size in bytes)** ● **# no of blocks = ceil(# of records/ Blocking factor)**

Relation	# of records	Record size in bytes	Blocking factor	# no. of blocks
User	1000	80	12	84
Train	2000	123	8	250
Station	10000	120	8	1250
Schedule	20000	49	20	1000
Seat_Chart	60000	40	25	2400
Ticket	80000	100	10	8000

Total number of blocks used = 84+250+1250+1000+2400+8000
= 12,984 blocks

3.3 ACCESS METHODS:

Considering the assumption we can calculate easily the size of single record (tuple) of every relation with the help of Schema. The above table gives the number of records in each relation, size of each record, blocking factor for a particular block of that relation and number of blocks required to store entire relation.

Having records on secondary storage, if you want to access them faster, then you need indexing. If a database is frequently queried and it is too large then it is supposed to have index to increase performance. There are various indexes used in databases. Here, we consider the following indexing scheme: Primary Index, Clustered Index and Secondary index. Based on the query, we decide what type of indexing file.

Relation	Indexing type	Indexing attribute(s)	Is a key?
User	Primary	Username	Yes
Train	Primary	Number	Yes
Stations	Primary	Code	Yes
Schedule	Primary	Id	Yes
Seat_Chart	Primary	{ Train, Date }	Yes
Ticket	Primary	{ User, Train, Date }	Yes

The following table explains what is the disk block access time to extract particular record for all the relations.

Relation	# of records	# no of data blocks	Index size per record	# of index records per block	# no of index blocks	# no of block access with indexing	# no of block access with indexing
User	1000	84	24	42	24	84	6
Train	2000	250	19	53	38	250	6
Stations	10000	1250	14	73	137	1250	8
Schedule	20000	1000	8	128	157	1000	8
Seat_Chart	60000	2400	27	37	1622	2400	12
Ticket	80000	8000	47	21	3810	8000	13

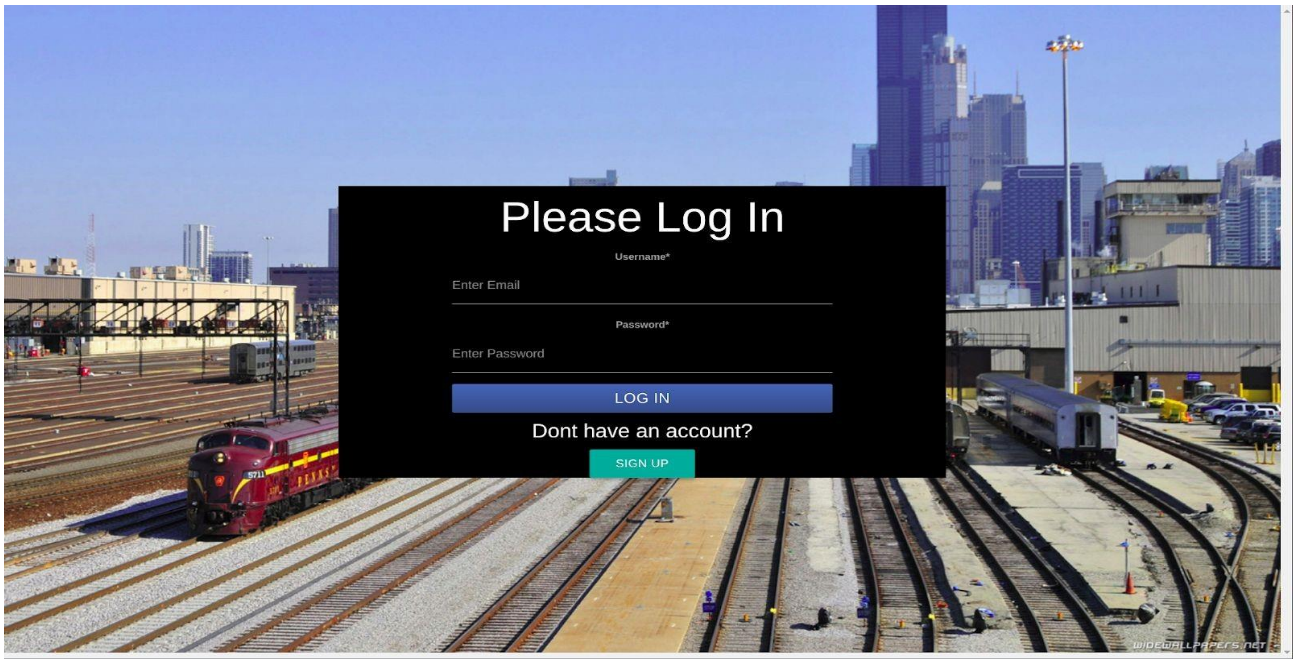
# of index records per block= Block size / Index size per record

# no of index blocks =ceil(# of records / # of index records per block)
# no of block access without indexing = # no of data blocks
number of block accesses with indexing = ceil [log(# no of index blocks)] + 1
Indexing the data file definitely reduces the number of block accesses needed to find particular record from the data file. The complete statistics is shown in above table.

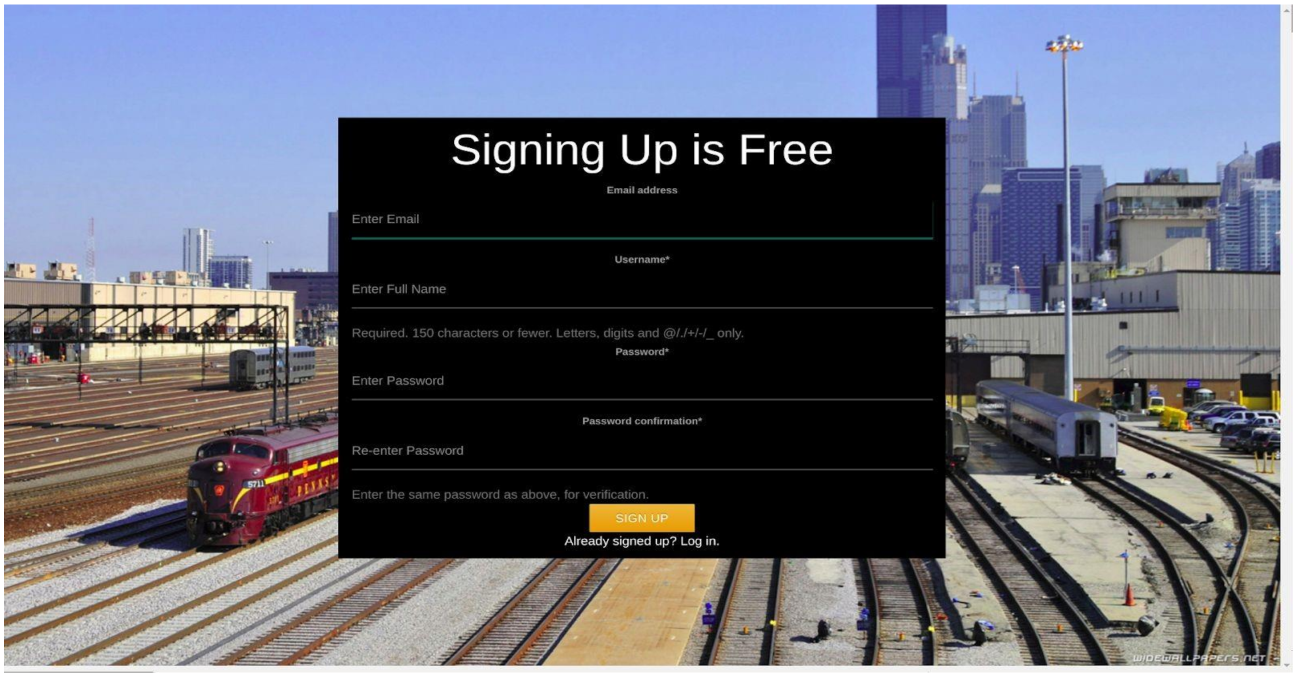
3.4 TIMINGS

Disk access time = Average seek time + latency time + block transfer time =
20 + 10 + 0.5
= 30.5 ms
Therefore, to access one random block and transfer it, the time is 30.5ms.
If the blocks are consecutive seek time and latency time are not included. Also, there can be overhead delay and queuing delay.

CHAPTER 4 IMPLEMENTATION AND RESULTS 1. Login Form



2. Signup Form



Signing Up is Free

Email address

Enter Email

Username*

Enter Full Name

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password*

Enter Password

Password confirmation*

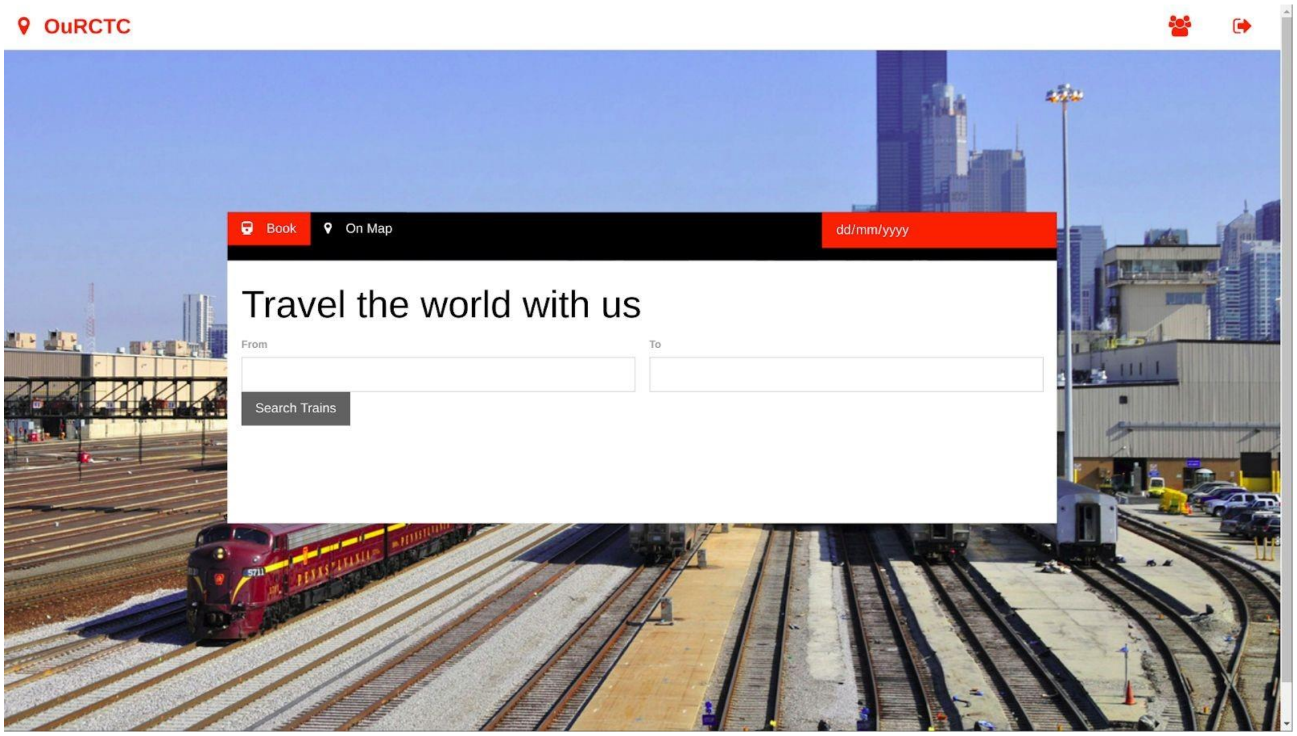
Re-enter Password

Enter the same password as above, for verification.

SIGN UP

Already signed up? Log in.

3. Home Page



OuRCTC

Book

On Map

dd/mm/yyyy

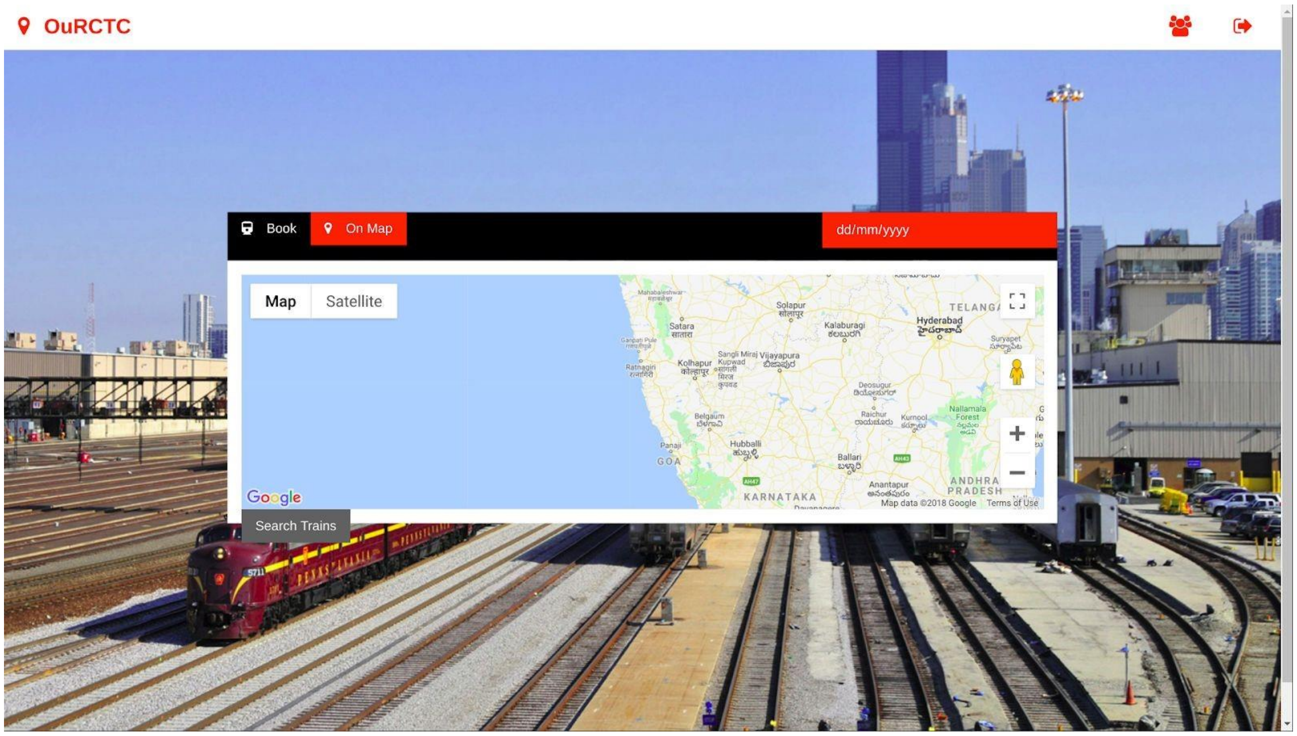
Travel the world with us

From

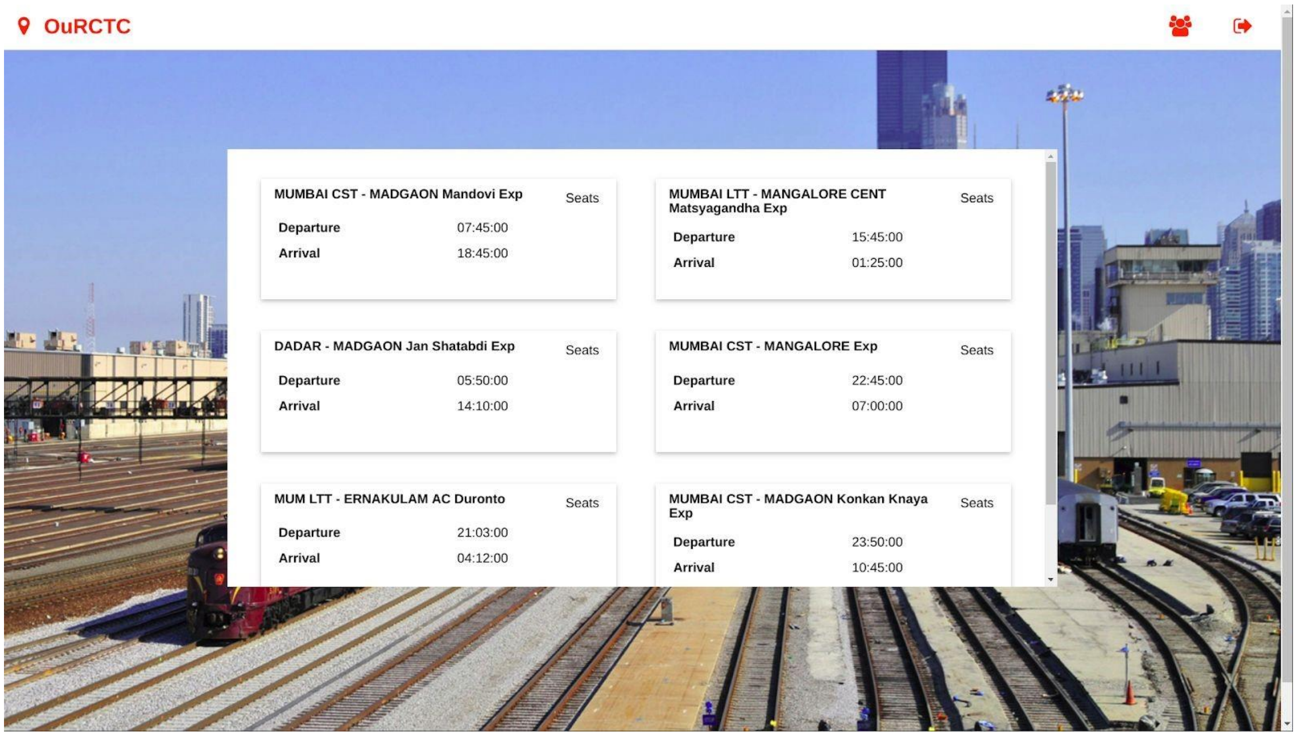
To

Search Trains

4. Select from Map



5. Display Trains


OuRCTC

MUMBAI CST - MADGAON Mandovi Exp — Seats
Departure 07:45:00
Arrival 18:45:00

MUMBAI LTT - MANGALORE CENT Matsyagandha Exp — Seats
Departure 15:45:00
Arrival 01:25:00

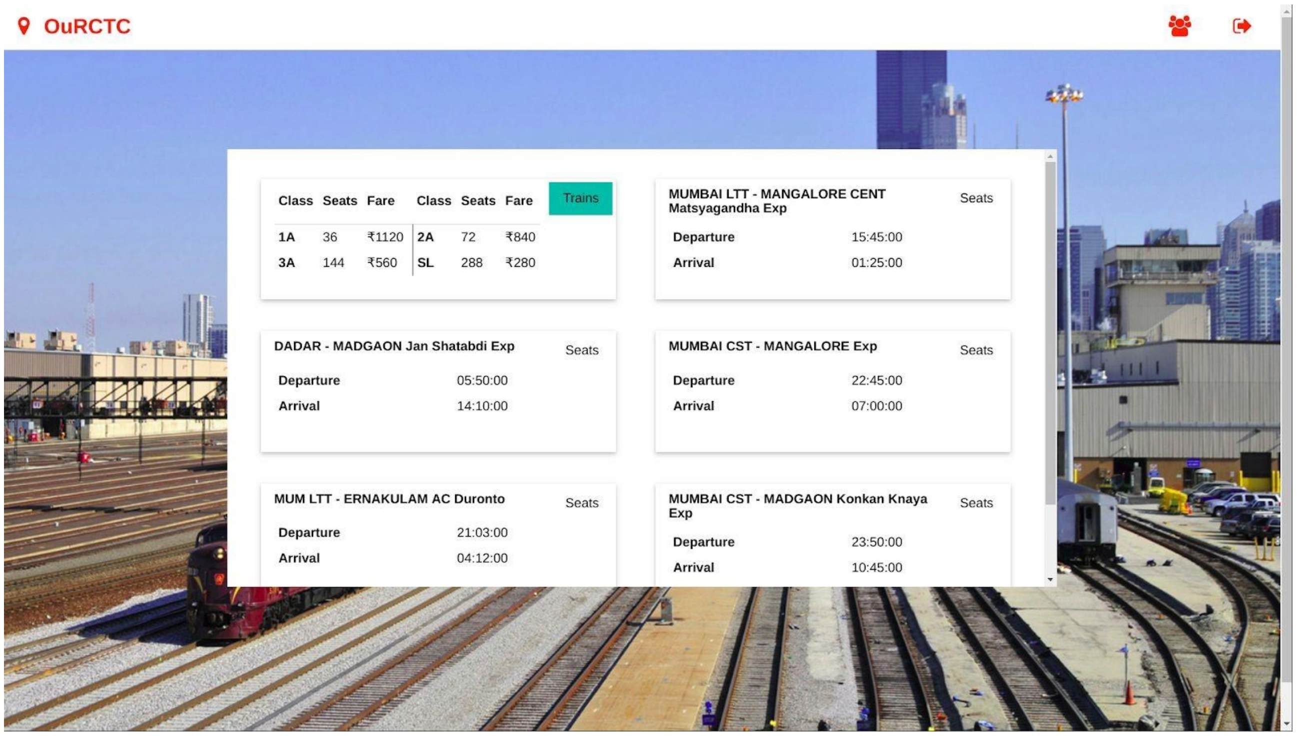
DADAR - MADGAON Jan Shatabdi Exp — Seats
Departure 05:50:00
Arrival 14:10:00

MUMBAI CST - MANGALORE Exp — Seats
Departure 22:45:00
Arrival 07:00:00

MUM LTT - ERNAKULAM AC Duronto — Seats
Departure 21:03:00
Arrival 04:12:00

MUMBAI CST - MADGAON Konkan Knaya Exp — Seats
Departure 23:50:00
Arrival 10:45:00

6. Display Available Seats



OuRCTC

Class	Seats	Fare	Class	Seats	Fare
1A	36	₹1120	2A	72	₹840
3A	144	₹560	SL	288	₹280

Trains

MUMBAI LTT - MANGALORE CENT Matsyagandha Exp — Seats
Departure 15:45:00
Arrival 01:25:00

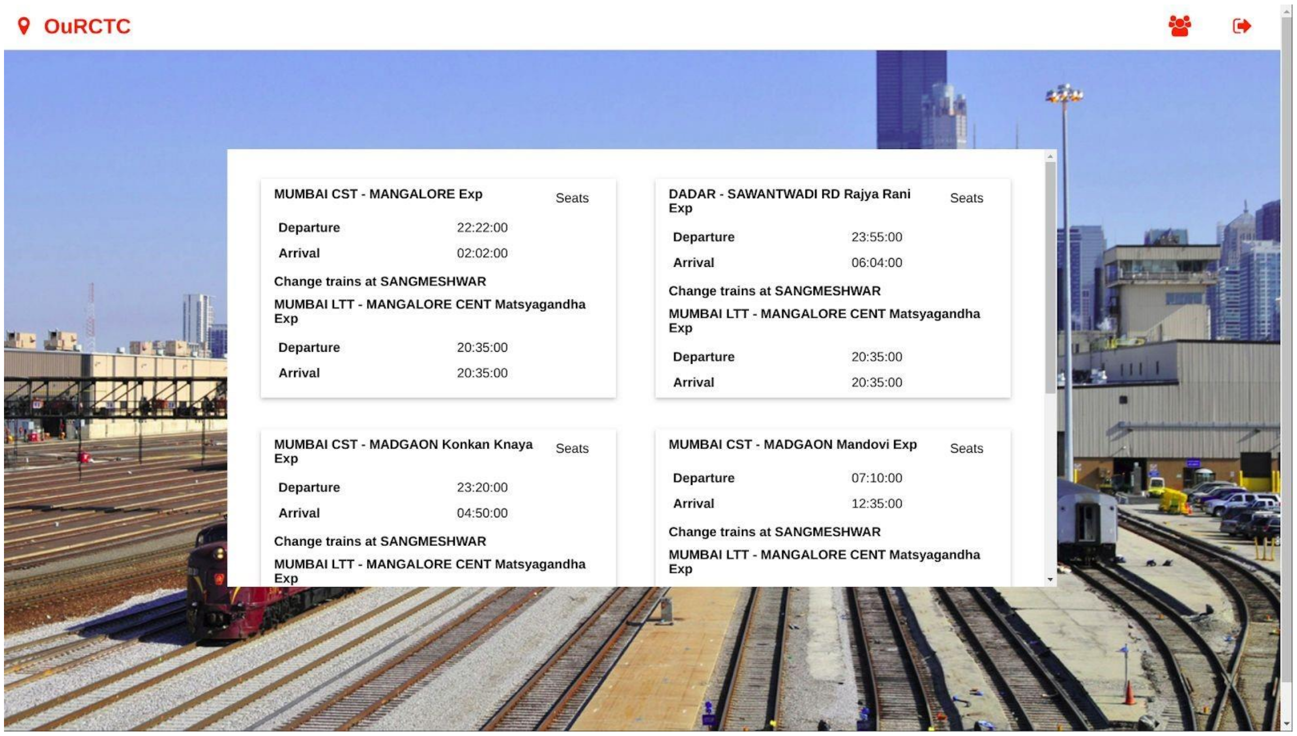
DADAR - MADGAON Jan Shatabdi Exp — Seats
Departure 05:50:00
Arrival 14:10:00

MUMBAI CST - MANGALORE Exp — Seats
Departure 22:45:00
Arrival 07:00:00

MUM LTT - ERNAKULAM AC Duronto — Seats
Departure 21:03:00
Arrival 04:12:00

MUMBAI CST - MADGAON Konkan Knaya Exp — Seats
Departure 23:50:00
Arrival 10:45:00

7. Display Connecting Trains



OuRCTC

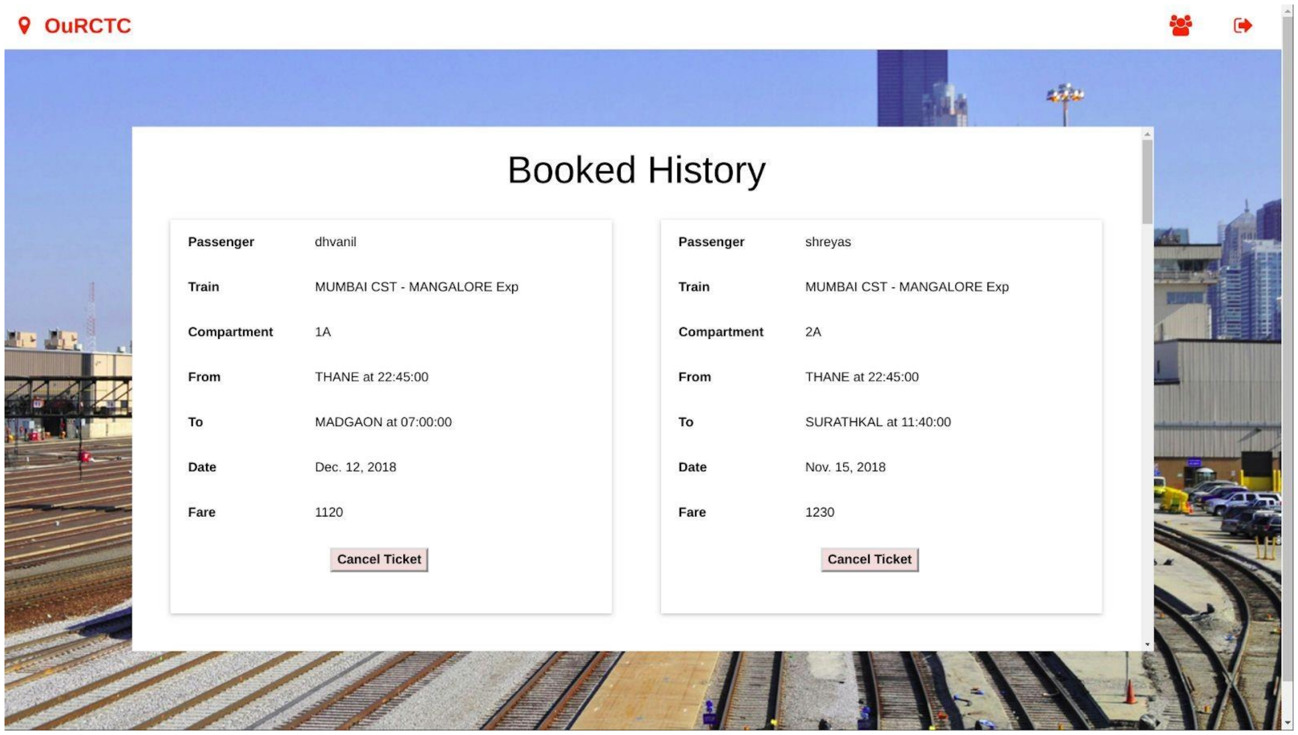
MUMBAI CST - MANGALORE Exp — Seats
Departure 22:22:00
Arrival 02:02:00
Change trains at SANGMESHWAR
MUMBAI LTT - MANGALORE CENT Matsyagandha Exp
Departure 20:35:00
Arrival 20:35:00

DADAR - SAWANTWADI RD Rajya Rani Exp — Seats
Departure 23:55:00
Arrival 06:04:00
Change trains at SANGMESHWAR
MUMBAI LTT - MANGALORE CENT Matsyagandha Exp
Departure 20:35:00
Arrival 20:35:00

MUMBAI CST - MADGAON Konkan Knaya Exp — Seats
Departure 23:20:00
Arrival 04:50:00
Change trains at SANGMESHWAR
MUMBAI LTT - MANGALORE CENT Matsyagandha Exp

MUMBAI CST - MADGAON Mandovi Exp — Seats
Departure 07:10:00
Arrival 12:35:00
Change trains at SANGMESHWAR
MUMBAI LTT - MANGALORE CENT Matsyagandha Exp

8. Booked Ticket History


OuRCTC

Booked History

Passenger	dhvanil
Train	MUMBAI CST - MANGALORE Exp
Compartment	1A
From	THANE at 22:45:00
To	MADGAON at 07:00:00
Date	Dec. 12, 2018
Fare	1120

Passenger	shreyas
Train	MUMBAI CST - MANGALORE Exp
Compartment	2A
From	THANE at 22:45:00
To	SURATHKAL at 11:40:00
Date	Nov. 15, 2018
Fare	1230

CONCLUSION

Indian Railway Catering and Tourism Corporation (IRCTC) is a subsidiary of the Indian Railways that handles the catering, tourism and online ticketing operations of the Indian railways, with around 5,50,000 to 6,00,000 bookings everyday is the world's second busiest. It's tagline is "Lifeline of the nation". It is known for changing the face of railway ticketing in India. Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers. Databases are used to hold administrative information and more specialized data, such as engineering data or economic models. Examples include computerized library systems, flight reservation systems, computerized parts inventory systems, and many content management systems that store websites as collections of webpages in a database. We have tried to implement a part of IRCTC and it has helped us to understand how Database is managed in the website.

REFERENCES

1. https://en.wikipedia.org/wiki/Indian_Railway_Catering_and_Tourism_Corporation
2. https://www.irctc.co.in/nget/train-search
3. https://data.gov.in/keywords/indian-railways
4. http://api.erail.in/
5. https://railwayapi.com/
6. https://indianrailapi.com/
7. https://www.programmableweb.com/api/indian-railways
8. https://data.gov.in/resources/indian-railways-time-table-trains-available-reservation-03082015/api

APPENDIX Description of tool

Application Tools

- Database: MySQL
    - MySQL has proved to be the database for web based applications, because of its performance and scalability, reliability, availability. From the perspective of a database administrator, it’s perfectly reliable and easily maintainable.
- Backend Framework : Django
    - Django encourages clean, practical way of designing highly customizable applications.
    - It is a very reliable, efficient, architecturally sound and secure when building web apps.
- Frontend Tools : HTML, CSS, JavaScript
    - HTML stays the markup language for creating web pages and web applications.
    - CSS is the stylesheet language for styling the documents.
    - JavaScript is the front end scripting language.

Development Tools

- PyCharm Professional
    - PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes with automated code refactorings and rich navigation capabilities.
- Github
    - Helps developers to collaborate over the code easily and for version controlling the source code.