Project Team 16 - Railway Reservation Management Database

Team Members:
Jinal Butani,
Harshitha Keshavaraju Vijayalakshmi,
Sai Phani Bhargavi,
Sumedh Joglekar

PROJECT PROPOSAL

Objective :

To create a railway reservation management database system for all registered users.

Scope :

Railway database that will allow registered users to search for trains from database, book/reserve/cancel railway tickets, also to navigate through past booked tickets. Database will also have data related to trains, costs and available seats.

Content :

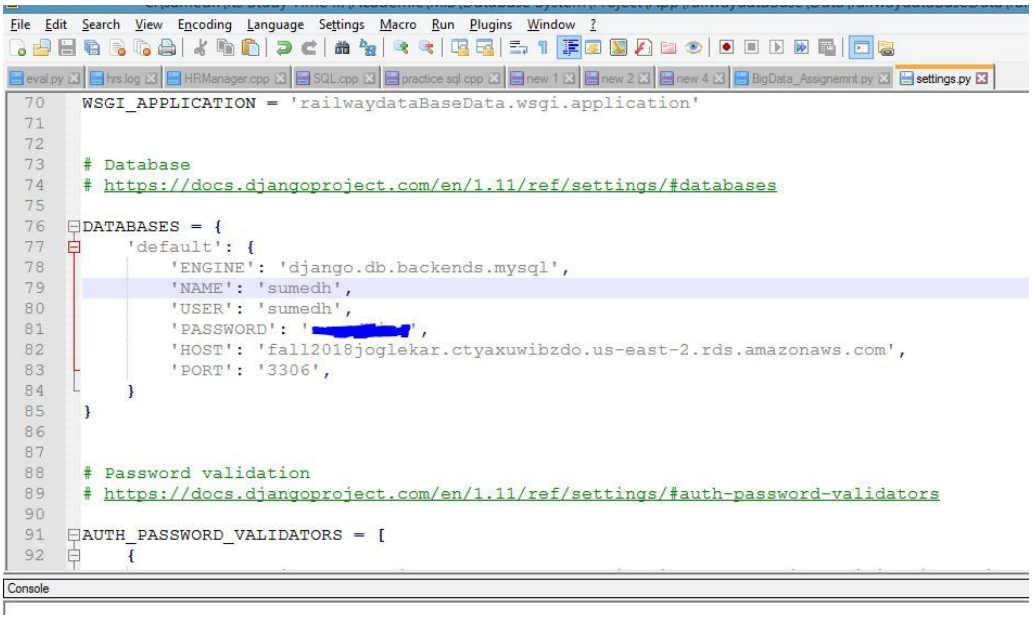
1. Railway database will store information of passengers(users) who will be accessing the database
2. The users will be able to view the train details as train number, route, schedule and available seats
3. The users will be able to search and list/filter trains based on date,time route etc
4. The users will be able to book/reserve tickets , select seats and class, make payment.
5. Administrator who will update train schedule details regularly.
6. The user travel history will be maintained in the database.
7. User can contact (message) support for any queries.

PROJECT ENVIRONMENT

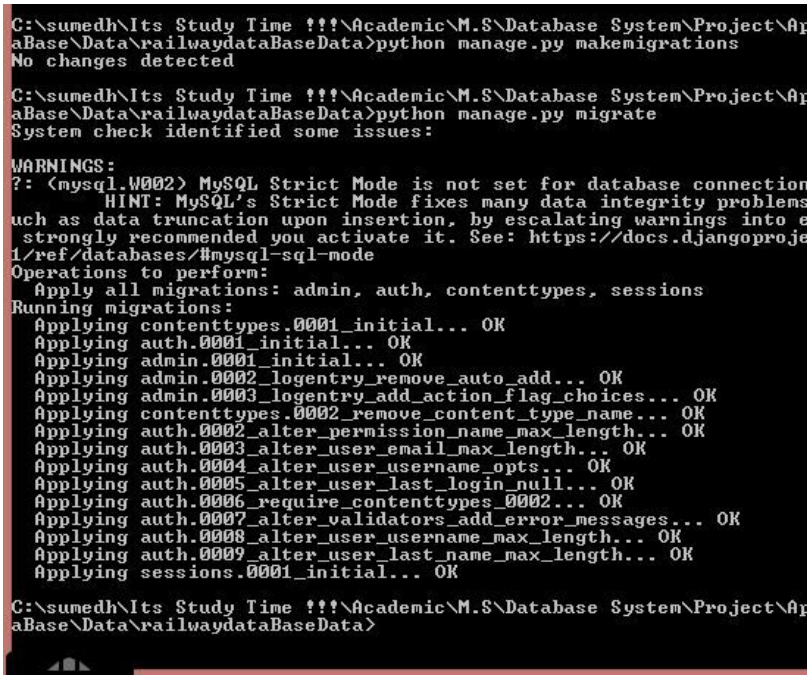
1. For the implementation of database we are using MySQL hosted on AWS.
2. For the UI implementation we are going to use Django Framework.
3. We have created the new schema for our DB on AWS with the use of Django.

Steps Performed in order to connect to the database :-

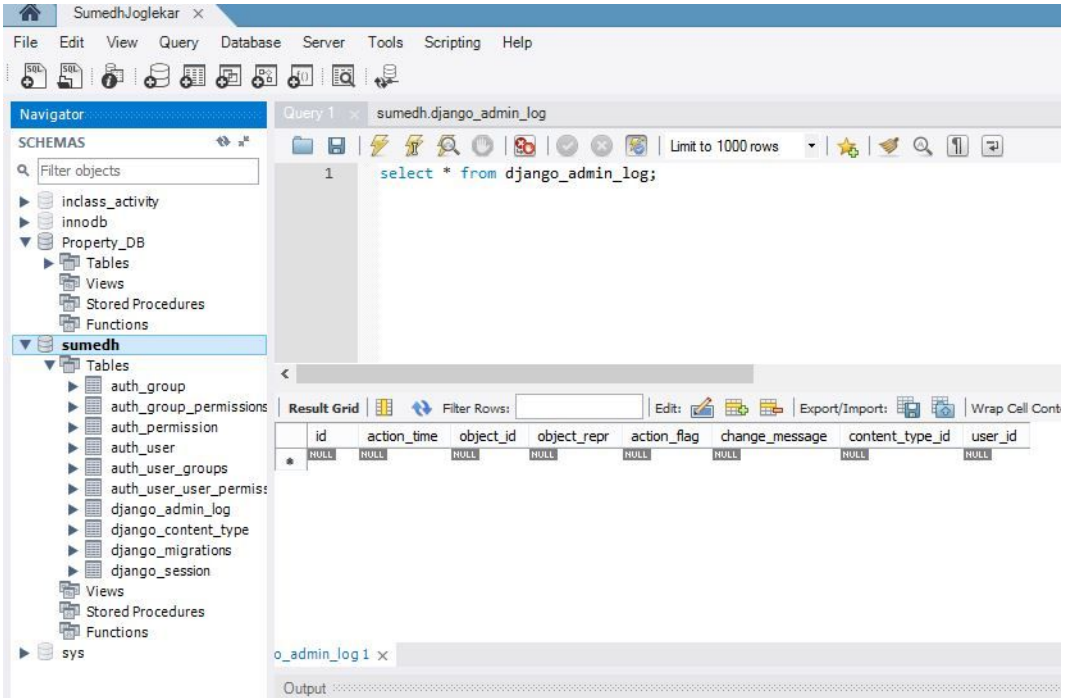
1. We have mentioned the Database name and the connection details in the settings.y of Django in order to connect to specific schema.



2. From Django we are able to connect to Database using ‘python manage.py migrate’ command.



3. Default Django tables are created successfully in the mentioned database.



HIGH LEVEL REQUIREMENTS

Initial user roles

User Roles	Description
User	Can be registered users / administrator
Train	Holds train details
Traveler	Will have the travel details of traveler
Route	Will hold connecting station details
Train_status	The seats availability on particular date
Message	Contains messages of Administrative queries
Reservation	Travel history of reservations
Station	Holds station details

Initial user story descriptions

Story ID	Story description
US1	As a guest I want to register so that I can travel.
US2	As a administrator I want to update train schedule (train status, route details, station details,class fare).
US3	As a user I want to search and list all the trains which are available for me so that I can book tickets.
US4	As a user I want to check the train status, route details, station details and travel cost.
US5	As a user I want to book tickets so that traveler can reserve the seat.
US6	As a user I want to cancel reservation so that traveler can release the seat.
US7	As a user I want to view my travel history.
US8	As a user I want to send message to administrator for any queries.
US9	As a administrator I want to reply to the queries.

HIGH LEVEL CONCEPTUAL DESIGN

Entities

User
Train
Traveler
Route
Train_status
Message
Reservation
Station

Relationships

User search Train
User lists Train
User books Traveler
User make Reservation
User cancel Reservation.
User checks Train_status.
Train has Train_status
Train has Route
Route contains Station
User can message User
User(Administrator) can update Train(Train schedule, Train Details).

Project: Sprint 1 - Database design and implementation

REFINED HIGH LEVEL REQUIREMENTS

Initial user roles

User Roles	Description
User	Can be registered users
Administrator	Administrator can update the Train Schedule and reply to the queries
Traveler	Will have the travel details of traveler

Part 1: Refine requirements

Subset of User Stories chosen for Sprint1 : { US1, US3, US4 from sprint 0}

Story ID	Story description
US1	As a guest, I want to register so that I can travel.
US2	As a user, I want to search all the trains which are available with cost.
US3	As a user, I want to search and list all the trains which are available for all source and destination.
US4	As a user, I want to check the train seat availability so that I can book the ticket.
US5	As a user, I want to check the station details so that i can view options.

Part 2: Perform conceptual design

CONCEPTUAL DESIGN

Entity : User

Attributes:

- user_id (Simple,Primary Key)
- ssn (Simple)
- name (Composite)
  - first_name
  - last_name
- email_id (Simple)
- phone_no (Multi-valued,Composite)
  - country_code
  - area_code
  - prefix
  - line_number
- gender (Simple)
- date_of_birth (Simple)
- address (Composite)
  - address_line1
  - city
  - State
  - zip_code

Note :-

1. User_Id will be the primary key of table User and User_id will get auto generated when the user gets registered.
2. Email_id is simple assuming each user will use 1 email id
3. Phone_no is multi valued as each user can have multiple phone number and it will be composite as can be divided further divided into country_code, area_code, prefix, line_number.

Entity : Train

Attributes :

- train_number (simple)
- train_name (simple)
- source (simple)
- destination (simple)
- no_available_seats (simple)
- travel_fare (simple)

Note :

1.Train_number will be the primary key of the table.

2. no_available_seats will give how many seats are available.

Entity: UserAccount
Attributes:
- username (simple)
- password

Note :
In the UserAccount table username will be the primary key.

Entity : Station
Attributes :
- station_no (simple)
- station_name (simple)
- location (composite)
  - address_line1
  - city
  - state
- zip_code

Note :
station_no will be the primary key of Station.

Entity: Ticket
Attributes:
- ticket_no,
- user_id,
- train_number ,
- travel_date,
- no_of_passengers,
- ticket_status
- total_cost
- source
- destination

Entity: passenger
Attributes:

Note :
1. ticket_no will be the primary key of Ticket.

Relationship: **User** has **UserAccount**
Cardinality: one to one

Participation:
User entity has total participation with the UserAccount entity
UserAccount entity has total participation with the User entity

Relationship: **Train** stopsAt **Station**

Cardinality: many to many
Participation:
Train entity has total participation with the Station entity
Station entity has total participation with the Train entity

Relationship: **Train** startsAt **Station**

Cardinality: many to many
Participation:
Train entity has total participation with the Station entity
Station entity has total participation with the Train entity

Part 3: Perform logical design

LOGICAL DESIGN

Table : User
Column :

- user_id
- ssn
- first_name
- last_name
- email_id
- phone_no1
- phone_no2
- gender
- date_of_birth
- address_line1
- city
- state
- zip_code

Note :

1. phone_no is restricted to have only 2 values per user.

Table : Train
Column :

train_number
train_name
source
destination
no_available_seats
travel_fare

Table : UserAccount
Column :

user_id (foreign key referencing to user_id from User)
username
password

Note : .

1. In order to maintain the security measures, password details of the user will get stored in different table called UserAccount.
2. user_id will be used to link user information of particular user.

Table: Station
Column :

station_no
station_name
address_line1
city
state
zip_code

Note :

1.Station_no will be the primary key to uniquely identify the the table.

Table: StartsAt
Column:

station_no (foreign key referencing to station_no from Station)
train_number (foreign key referencing to train_number from Train)
arrival_time

departure_time

Note:
For many-to-many relationship “Train startsAt Station” create table startsAt and add station_no, train_number, departure_time as columns to it.

For many-to-many relationship “Train stopsAt Station” create table stopsAt and add station_no, train_number, arrival_time as columns to it. But this has much of the same attributes as that of startsAt table and to avoid redundancy, combine arrival_time with startsAt table.

Part 4: Implement and deploy database

DML SQL Queries:

Table : User

```
CREATE TABLE User(
user_id int(10)  AUTO_INCREMENT primary key,
ssn varchar(12) not null,
first_name varchar(80) not null,
last_name varchar(80) not null,
email_id varchar(80),
phone_number1 double not null,
phone_number2 double,
gender varchar(10) not null,
date_of_birth date not null,
address_line1 varchar(100) not null,
city varchar(50) not null,
state varchar(50) not null,
zip_code varchar(20) not null
);
```

Table : UserAccount

```
create table UserAccount(
user_id int(10) not null,
username varchar(20) primary key,
```

```
password varchar(20) not null,
FOREIGN KEY fk_user_id(user_id) REFERENCES User(user_id)
);
```

Table : Train

```
CREATE TABLE Train(
  train_number int(11),
  train_name varchar(25) not null,
  tsource varchar(25) not null,
  destination varchar(25) not null,
  no_available_seats int(10),
  travel_fare int(10) not null,
PRIMARY KEY(train_number)
);
```

Table : Station

```
create table station(
   station_no  int(10) primary key,
   station_name varchar(50) not null,
   address_line1 varchar(100) not null,
   city  text(25) not null,
   state text(35) not null,
   zip_code int(10) not null
);
```

Table: StartsAt

```
create table startsAt(
   station_no  int(10),
   train_number int(10),
   arrival_time time,
   departure_time time,
primary key(station_no, train_number),
FOREIGN KEY (station_no) REFERENCES  station(station_no),
FOREIGN KEY (train_number) REFERENCES Train(train_number)
);
```

DDL SQL Queries :

User :

```
insert into
User(ssn,first_name,last_name,email_id,phone_number1,phone_number2,gender,date_of_birth,addre
ss_line1,city,state,zip_code)
```

values('100000000','Jinal','Butani','jbutani@uncc.edu','9802262049','9506065760','female','1996/10/20','10001 c, graduate ln','Charlotte','NC','28262');
insert into
User(ssn,first_name,last_name,email_id,phone_number1,phone_number2,gender,date_of_birth,address_line1,city,state,zip_code)
values('459000000','Sumedh','Joglekar','sjogleka@uncc.edu','9028251242','7049573530','male','1994/05/04','516 Barton Creek Dr, Apt C','Charlotte','NC','28262');
insert into
User(ssn,first_name,last_name,email_id,phone_number1,phone_number2,gender,date_of_birth,address_line1,city,state,zip_code)
values('56000000','Gaurav','Mahadik','gmahadik@uncc.edu','9969449896','7049572230','male','1996/08/20','200 Barton Creek Dr, Apt D','Charlotte','NC','28262');
insert into
User(ssn,first_name,last_name,email_id,phone_number1,phone_number2,gender,date_of_birth,address_line1,city,state,zip_code)
values('9786875980','Sakshat','Surve','ssurve@uncc.edu','9969112128','7049578888','male','1996/01/12','9402 University Terrace Dr, Apt F','Charlotte','NC','28262');
insert into
User(ssn,first_name,last_name,email_id,phone_number1,phone_number2,gender,date_of_birth,address_line1,city,state,zip_code)
values('5198567441','Praik','Parekh','pparekh@uncc.edu','8097470356','7049572886','male','1996/12/25','9421,University Bldv','Charlotte','NC','28262');

UserAccount :

insert into UserAccount (user_id,username,password)
values ('1','jinal01','jinal01');
insert into UserAccount (user_id,username,password)
values ('2','sumedh','sumjog');
insert into UserAccount (user_id,username,password)
values ('3','gaurav','gm007');
insert into UserAccount (user_id,username,password)
values ('4','sakshat','ssurve');
insert into UserAccount (user_id,username,password)
values ('5','Pratik','pparekh2')

Train :

INSERT INTO Train (train_number,train_name,tsource,destination,no_available_seats,travel_fare)
VALUES
(12345, 'charlotteexp', 'charlotte', 'tampa', 3, 25);
INSERT INTO Train (train_number,train_name,tsource,destination,no_available_seats,travel_fare)
VALUES

(23456, 'chicagoexp', 'chicago','newyork',2,40);
INSERT INTO Train (train_number,train_name,tsource,destination,no_available_seats,travel_fare)
VALUES (34567,'bostonexp','boston','lafayette',65,17);
INSERT INTO Train
(train_number ,train_name,tsource,destination,no_available_seats,travel_fare)VALUES
(67890,'atlantaexp','atlanta','raleigh',11,70);
INSERT INTO Train
(train_number,train_name,tsource,destination,no_available_seats,travel_fare)VALUES
(45678,'texasexp','dallas','chicago',45,56);

Station :

INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(234,'Charlotte Station','1914 N Tryon St','Charlotte','North Carolina',28262);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(250,'Union Station','601 N Nebraska Ave','Tampa','Florida',32003);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(125,'Cary Station','211 N Academy St','Cary','North Carolina',28262);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(129,'Union Station','225 S Canal St','Chicago','Illinois',60001);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(325,'Washington Union Station','50 Massachusetts Ave NE','Washington','DC',20002);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(225,'Penn Station','IRT Broadway,Seventh Avenue Line,34 St','New York','NY',10119);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(285,'Back Bay Station','145 Dartmouth St','Boston','Massachusetts',02116);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(415,'lafayette station','200 N Second St','lafayette','Louisiana',47901);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(365,'Peachtree Station','1688 Peachtree St NW','Atlanta','Georgia',31119);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(248,'Raleigh Union Station','510 W Martin St','Raleigh','North Carolina',27601);
INSERT INTO station(station_no,station_name,address_line1,city,state,zip_code) VALUES
(437,'Union Station','400 S Houston St','Dallas','Texas',75202);

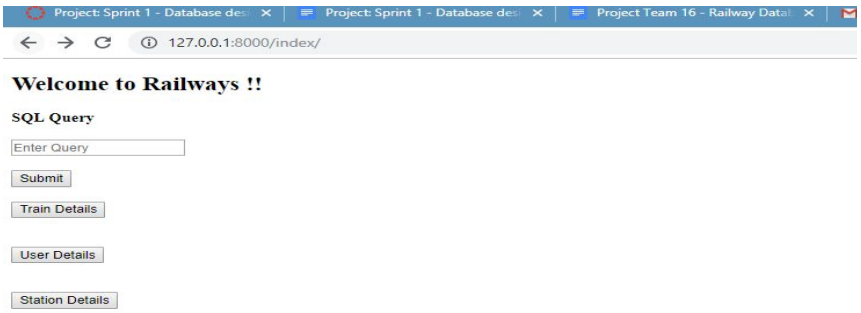
StartsAt:

INSERT INTO startsAt(station_no,train_number,arrival_time,departure_time) VALUES
(234,12345,null,'01:46:00'),
(125,12345,'09:53:00','11:05:00'),
(250, 12345,'09:23:00',null),
(129,23456,null,'21:30:00'),
(225,23456,'18:45:00',null);

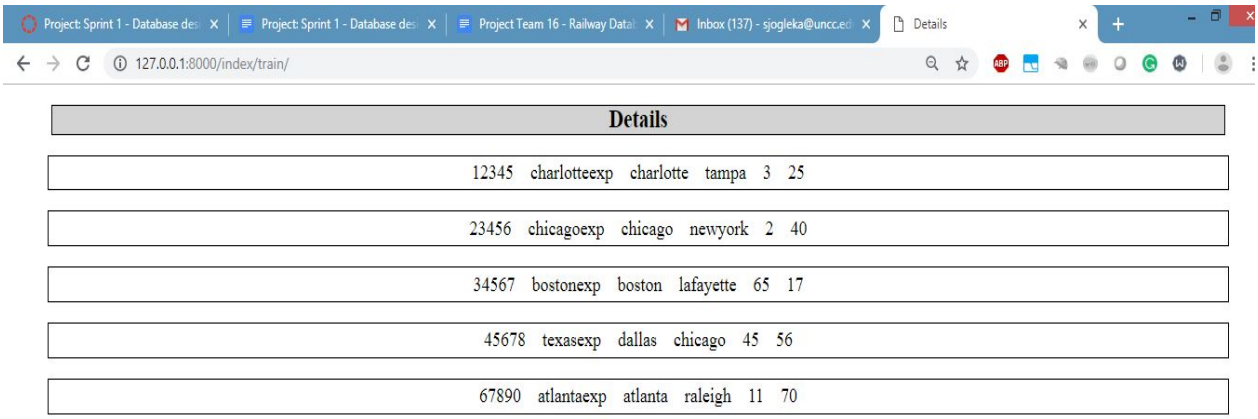
Part 5: Demonstrate key SQL queries

Screenshots:

1. Home Page

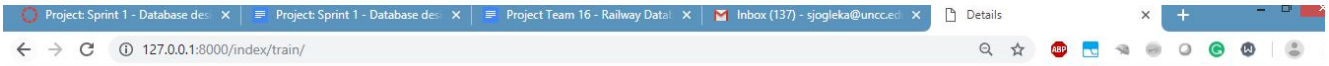


2. Train Details



Details
12345 charlotteexp charlotte tampa 3 25
23456 chicagoexp chicago newyork 2 40
34567 bostonexp boston lafayette 65 17
45678 texasexp dallas chicago 45 56
67890 atlantaexp atlanta raleigh 11 70

3. User Details



Details
1 100000000 Jinal Butani jbutani@uncc.edu 9802262049.0 9506065760.0 female 1996-10-20 10001 c, graduate ln Charlotte NC 28262
2 459000000 Sumedh Joglekar sjogleka@uncc.edu 9028251242.0 7049573530.0 male 1994-05-04 516 Barton Creek Dr, Apt C Charlotte NC 28262
3 56000000 Gaurav Mahadik gmahadik@uncc.edu 9969449896.0 7049572230.0 male 1996-08-20 200 Barton Creek Dr, Apt D Charlotte NC 28262
4 9786875980 Sakshat Surve ssurve@uncc.edu 9969112128.0 7049578888.0 male 1996-01-12 9402 University Terrace Dr, Apt F Charlotte NC 28262
5 5198567441 Praik Parekh pparekh@uncc.edu 8097470356.0 7049572886.0 male 1996-12-25 9421,University Bldv Charlotte NC 28262

4. Station Details

Details
125 Cary Station 211 N Academy St Cary North Carolina 28262
129 Union Station 225 S Canal St Chicago Illinois 60001
225 Penn Station IRT Broadway,Seventh Avenue Line,34 St New York NY 10119
234 Charlotte Station 1914 N Tryon St Charlotte North Carolina 28262
248 Raleigh Union Station 510 W Martin St Raleigh North Carolina 27601
250 Union Station 601 N Nebraska Ave Tampa Florida 32003
285 Back Bay Station 145 Dartmouth St Boston Massachusetts 2116
325 Washington Union Station 50 Massachusetts Ave NE Washington DC 20002
365 Peachtree Station 1688 Peachtree St NW Atlanta Georgia 31119
415 lafayette station 200 N Second St lafayette Louisiana 47901
437 Union Station 400 S Houston St Dallas Texas 75202

Project: Sprint 2 - Database design and implementation

User roles

User Roles	Description
User	Can be registered users
Administrator	Administrator can update the Train Schedule and reply to the queries
Passenger	Will have the travel details of Passenger

Part 1: Refine requirements

Subset of User Stories for Sprint 2:

Story ID	Story description
US1	As a guest, I want to register so that I can travel.
US2	As a user, I want to search and list all the trains which are available for the desired source and destination.
US3	As a user, I want to check the train seat availability so that I can book the ticket.
US4	As a user, I want to check the station details so that I can view the options.
US5	**As a passenger I need to have ticket so that I can travel.**
US6	**As a user I want to book tickets so that passenger can reserve the seat.**
US7	**As a user I want to cancel tickets.**
US8	**As a user I want to view my travel history.**

Part 2: Perform conceptual design

CONCEPTUAL DESIGN

Entity : User
Attributes:

- ssn (Simple)
- name (Composite)
  - first_name
  - last_name
- email_id (Simple)
- phone_no (Multi-valued,Composite)
  - country_code
  - area_code
  - prefix
  - line_number
- gender (Simple)
- date_of_birth (Simple)
- address (Composite)
  - address_line1
  - city
  - State
  - zip_code

Note :-

1. Email_id will be the natural primary key User Entity
2. Phone_no is multi valued as each user can have multiple phone number and it will be composite as can be divided further divided into country_code, area_code, prefix, line_number.

Entity: UserAccount
Attributes:

- username (simple)
- password

Note :
In the UserAccount table username will be the natural primary key.

Entity : Train
Attributes :

- train_number (simple)
- train_name (simple)
- source (simple)
- destination (simple)
- total_capacity(simple)
- base_fare(simple)

Note :

1.train_number will be the natural primary key of the Entity.
2. total_capacity will give total capacity of train.
3. base_fare will be per hault.

Entity: Availability
Attributes:

- train_number
- train_running_day
- no_available_seats(derived)

Note :

1.train_number and train_running_date will be composite primary key of the Entity .
2. The Train_running_day will be monday to Sunday value, limiting to 7 days.

Entity : Station
Attributes :

- station_no (simple)
- station_name (simple)
- location (composite)
  - address_line1
  - city
  - state
  - zip_code

Note :
station_no will be the natural primary key of Station.

Entity : Ticket
Attributes:

- ticket_no (simple)
- train_number (simple)
- travel_day (simple)
- no_of_passengers (simple)
- ticket_status (simple)
- source (simple)
- destination (simple)

Note:

1. ticket_no will be the natural primary key to the Ticket entity.
2. No_of_passengers holds the number of tickets booked by the user.
3. Ticket_status will have the booked/cancelled information.

Entity : Passenger
Attributes:

- name (Composite)
  - first_name
  - last_name
- email_id (Simple)
- phone_no (Multi-valued,Composite)
  - country_code
  - area_code
  - prefix
  - line_number
- gender (Simple)

Note:

1. Passengers are the actual travelers in the train.
2. Email_id is identified as a natural primary key.

Relationship: **User** has **UserAccount**
Cardinality: one to one
Participation:
User entity has total participation with the UserAccount entity
UserAccount entity has total participation with the User entity

Relationship: **Train** stopsAt **Station**
Cardinality: many to many
Participation:

Train entity has total participation with the Station entity
Station entity has total participation with the Train entity
Relationship: **Train** startsAt **Station**
Cardinality: many to many
Participation:
Train entity has total participation with the Station entity
Station entity has total participation with the Train entity

Relationship: **User** books **Ticket**
Cardinality: one to many
Participation:
User entity has partial participation with the Ticket entity
Ticket entity has total participation with the User entity

Relationship: **User** cancels **Ticket**
Cardinality: one to many
Participation:
User entity has partial participation with the Ticket entity
Ticket entity has total participation with the User entity

Relationship: **User** checks **Availability**
Cardinality: one to many
Participation:
User entity has partial participation with the Availability entity
Availability entity has partial participation with the User entity

Relationship: **Passengers** have **Ticket**
Cardinality: many to many
Participation:
Passengers entity has total participation with the Ticket entity
Ticket entity has total participation with the Passengers entity

Part 3: Perform logical design

LOGICAL DESIGN

Table : User
Column :

- username
- first_name
- last_name
- email_id
- phone_no1
- phone_no2
- gender
- date_of_birth
- address_line1
- city
- state
- zip_code

Note :

1. phone_no is restricted to have only 2 values per user.
2. Introducing username as synthetic primary key to the table

Table : UserAccount
Column :

- username(foreign key referencing to username from User)
- password

Note :

1. In order to maintain the security measures, password details of the user will get stored in different table called UserAccount.
2. username will be used to link user information of particular user.

Table: Station
Column :

- station_no
- station_name
- address_line1
- city
- state
- zip_code

Note :

1. Station_no will be the primary key to uniquely identify the the table.

Table: Train
Column :

train_number
train_name
source
destination
total_capacity
base_fare

Table: StartsAt
Column:

station_no (foreign key referencing to station_no from Station)
train_number(foreign key referencing to train_number from Train)
arrival_time
departure_time
hault

Note:
1. For many-to-many relationship “Train startsAt Station” we need to create table startsAt and add station_no, train_number, departure_time as columns to it.
2. For many-to-many relationship “Train stopsAt Station” we need to create table stopsAt and add station_no, train_number, arrival_time as columns to it. But these are the same attributes as that of startsAt table and to avoid redundancy, combine arrival_time with startsAt table.

Table : Availability
Columns :

train_number (foreign key referencing to train_number from Train)
train_running_day
no_available_seats

Note:

1. train_number and train_running_day will be the composite primary key.
2. no_available_seats will be calculated as total_capacity - count(no_of_passengers) for particular train number on particular day.

Table: Ticket
Column:

- ticket_no
- train_number (foreign key referencing to train_number from Train)
- travel_day
- no_of_passengers
- ticket_status
- source
- destination

Note :

1. ticket_no will be the primary key of Ticket.

Table : BooksCancels
Column:

- username (foreign key referencing to username from User table)
- ticket_no (foreign key referencing to ticket_no from Ticket table, also primary key Travel_History table)

Note:

1. ticket_no, will be the primary key to the BooksCancel table.

Table : Passenger
Column:

- email_id (primary key)
- first_name
- last_name
- phone_no
- gender

Note:

1. email_id will be the primary key of Ticket.
2. Phone_no is changed to single valued attribute.

Table: Passenger_Ticket
Column:

- ticket_no (foreign key referencing to ticket_no from Ticket table
- email_id (foreign key referencing to Passenger)

Note:

1. email_id, ticket_no will be the primary key of Ticket.

Part 4: Normalization

Table : User
Column :

username
first_name
last_name
email_id
phone_no1
phone_no2
gender
date_of_birth
address_line1
city
state
zip_code

Highest Normalization Level: 2NF
Justification:

- Zip code depends on city so there is transitive dependency.
- We can keep property table in 2NF as attribute zip code is not unique and creation of separate table for zip code will not be feasible as it will not reduce space requirement to store the data also it will increase unnecessary joins while creating the tables.

Table : UserAccount
Column :

username
password

1. Highest Normalization level : 4NF

Table: Station
Column :

station_no
station_name
address_line1
city
state
zip_code

Highest Normalization Level: 2NF

Justification:
● Zip code depends on city so there is transitive dependency.
● We can keep property table in 2NF as attribute zip code is not unique and creation of separate table for zip code will not be feasible as it will not reduce space requirement to store the data also it will increase unnecessary joins while creating the tables.

Table: Train
Column :
train_number
train_name
source
destination
total_capacity
Base_fare

1. Highest Normalization level : 4NF

Table: StartsAt
Column:
station_no (foreign key referencing to station_no from Station)
train_number(foreign key referencing to train_number from Train)
arrival_time
departure_time
hault

1. Highest Normalization level : 2NF
2. Table StartsAt is in 2NF as departure_time depends on arrival_time.
3. This can be resolved by creating separate tables for arrival_time and departure_time but this will lead to redundant data and hence keeping it in 2NF.

Table : Availability
Columns :
train_number (foreign key referencing to train_number from Train)
train_running_day
no_available_seats

1. Highest Normalization level : 4NF as no_available_seats depends on both train_running_day and train_number.

Table: Ticket
Column:

ticket_no
train_number (foreign key referencing to train_number from Train)
travel_day
no_of_passengers
ticket_status
source
destination

1. Highest Normalization level : 2NF

Justification: There is a transitive dependency with source and train_number and destination and train_number. We are keeping it in 2NF only as creating another table will result in complexity while doing data retrieval.

Table : BooksCancels
Column:

username (foreign key referencing to username from User table)
ticket_no (foreign key referencing to ticket_no from Ticket table, also primary key user_ticket_info table)

1.Highest Normalization level : 4NF

Table: Passenger_Ticket
Column:

ticket_no (foreign key referencing to ticket_no from Ticket table)
email_id (foreign key referencing to Passenger)

Note:

1. email_id, ticket_no will be the primary key of Ticket.

Table : Passenger
Column:

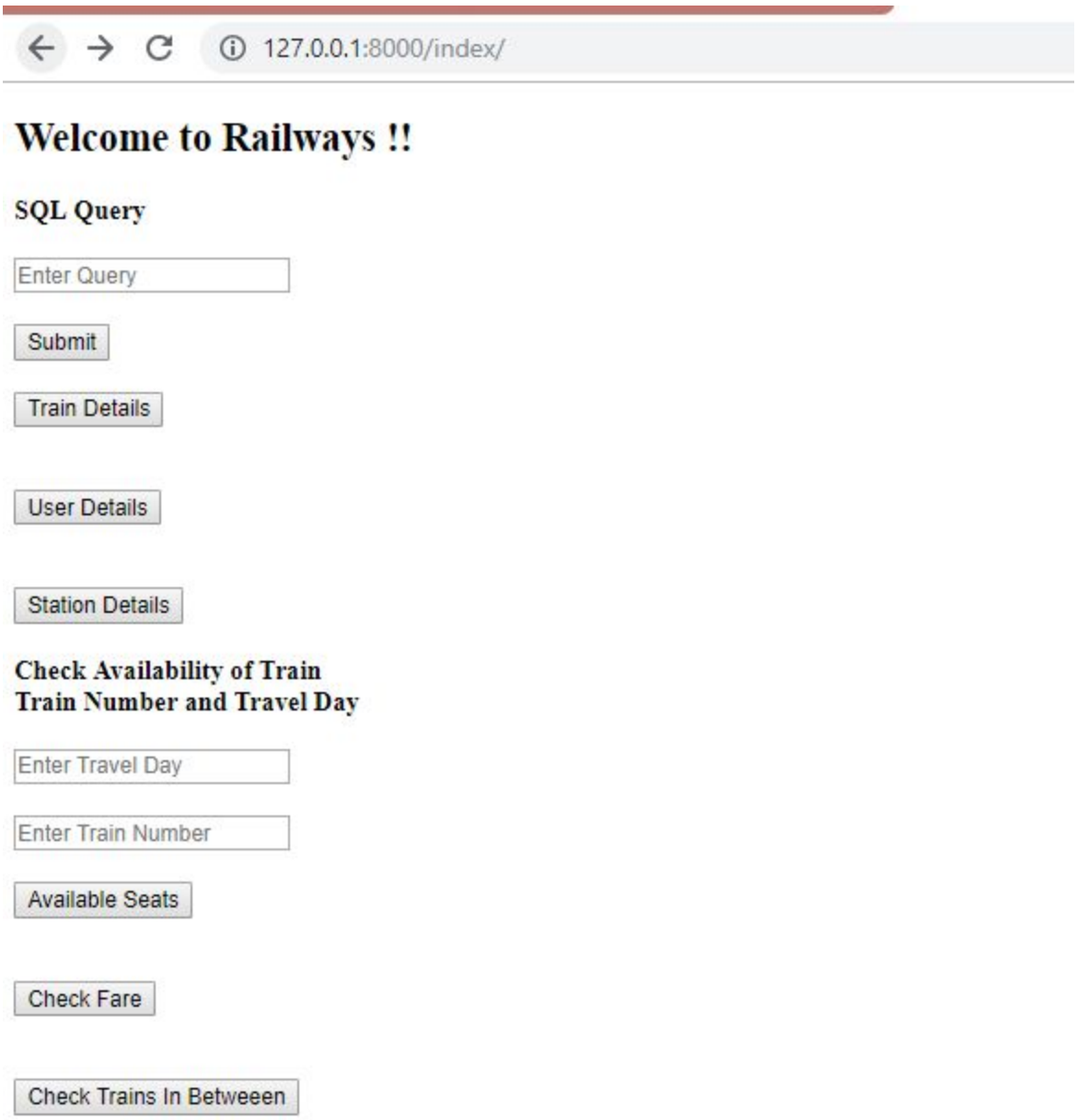
email_id (primary key)
first_name
last_name
phone_no
gender

Note:

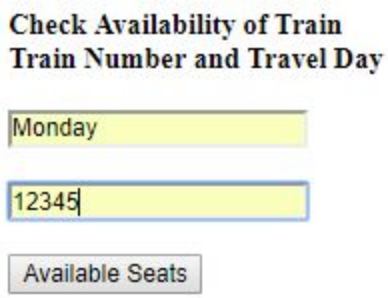
1. Highest Normalization level : 4NF

Part 5: Query Execution

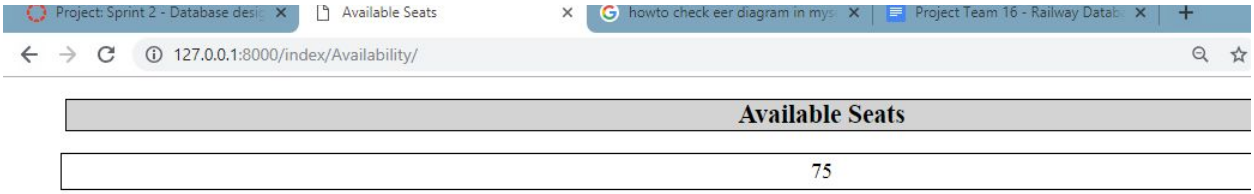
1. Home Page



2. Check Availability of Seats in particular train on particular Day



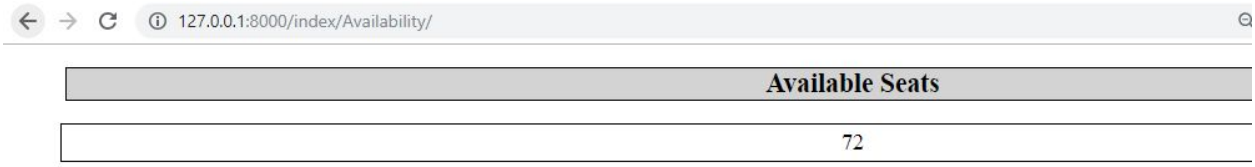
Output:



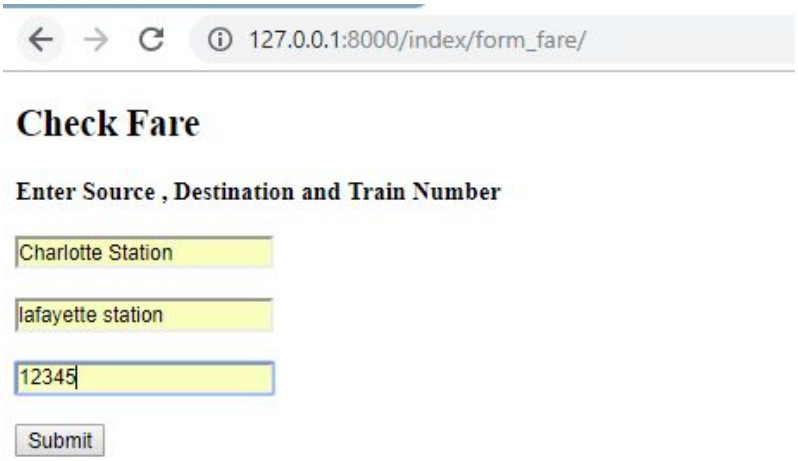
After New Ticket created :

insert into Ticket values('FF090G009',12345,'Monday',**3**,'Confirm','Tempa','New York');

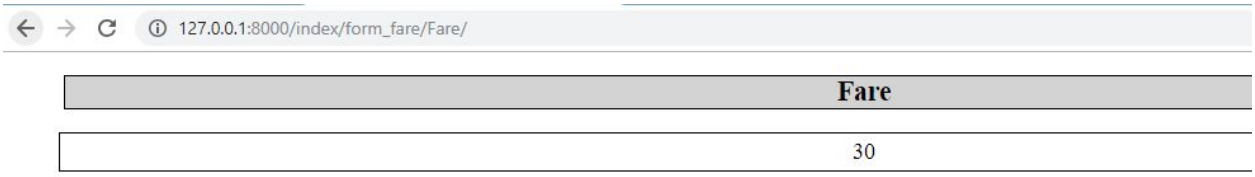
Changed Output:



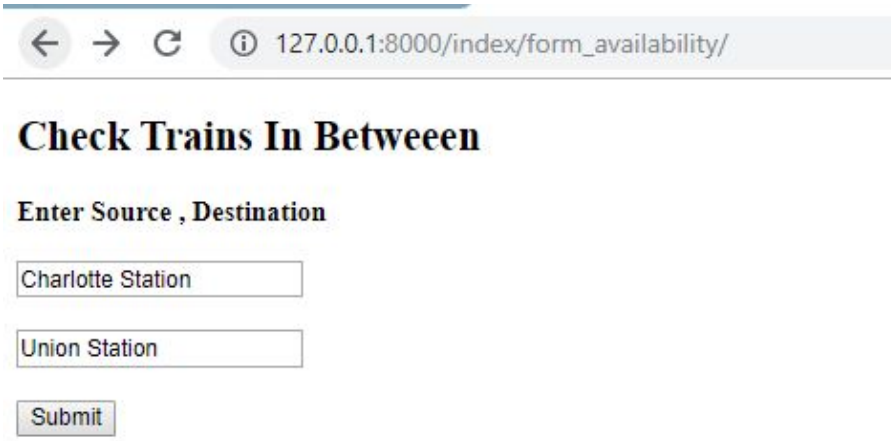
3. Check Cost of travel between desired source and destination and train number



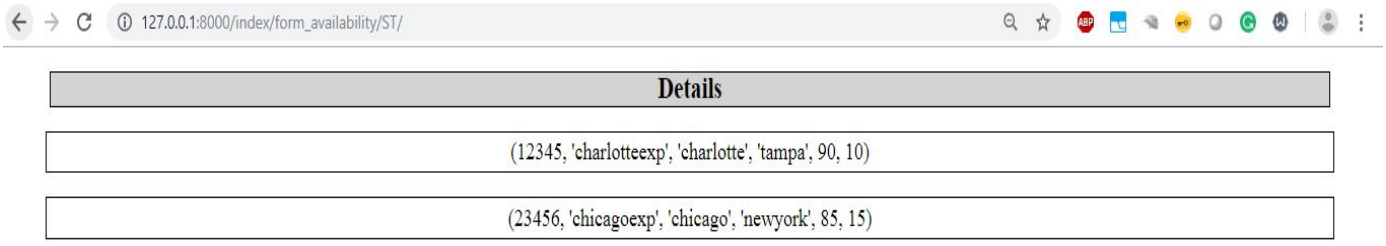
Output:



4. Check In Between Trains:



Output:


127.0.0.1:8000/index/form_availability/ST/
Details
(12345, 'charlotteexp', 'charlotte', 'tampa', 90, 10)
(23456, 'chicagoexp', 'chicago', 'newyork', 85, 15)

Project: Sprint 3 - Database design and implementation

User roles

User Roles	Description
User	Can be registered users
Administrator	Administrator can update the Train Schedule and reply to the queries
Passenger	Will have the travel details of Passenger

Part 1: Refine requirements

Story ID	Story description
US1	As a guest, I want to register so that I can travel.
US2	As a user, I want to search and list all the trains which are available for the desired source and destination.
US3	As a user, I want to check the train seat availability so that I can book the ticket.
US4	As a user, I want to check the station details so that I can view the options.
US5	As a user I want to book tickets so that passenger can reserve the seat.
US6	As a user I want to cancel tickets.
US7	As a user I want to view my travel history.
US8	As a administrator I want to add new train schedule (train details, route details, station details).
US9	As a user I want to send message to administrator for any queries.
US10	As a administrator I want to reply to the queries.

Part 2: conceptual design

Entity : User

Attributes:

- ssn (Simple)
- name (Composite)
  - first_name
  - last_name
- email_id (Simple)
- phone_no (Multi-valued,Composite)
  - country_code
  - area_code
  - prefix
  - line_number
- gender (Simple)
- date_of_birth (Simple)
- address (Composite)
  - address_line1
  - city
  - State
  - zip_code
- user_type(simple)

Note :

Adding new column to the user table as we have two type of user (customer/admin).

Entity: UserAccount

Attributes:

- username (simple)
- password

Note :

In the UserAccount table username will be the natural primary key.

Entity : Train

Attributes :

- train_number (simple)
- train_name (simple)
- source (simple)
- destination (simple)
- total_capacity(simple)
- base_fare(simple)

Note :

1.train_number will be the natural primary key of the Entity.

2. total_capacity will give total capacity of train.

3. base_fare will be per hault.

Entity: Availability
Attributes:

- train_number
- train_running_day
- no_available_seats(derived)

Note :

1.train_number and train_running_date will be composite primary key of the Entity .
2. The Train_running_day will be monday to Sunday value, limiting to 7 days.

Entity : Station
Attributes :

- station_no (simple)
- station_name (simple)
- location (composite)
  - address_line1
  - city
  - state
  - zip_code

Note :

1. station_no will be the natural primary key of Station.

Entity : Ticket
Attributes:

- train_number (simple)
- travel_day (simple)
- no_of_passengers (simple)
- ticket_status (simple)
- source (simple)
- destination (simple)

Note:.

1. No_of_passengers holds the number of tickets booked by the user.
2. Ticket_status will have the booked/cancelled information

Entity : Passenger
Attributes:

- name (Composite)
  - first_name
  - last_name
- email_id (Simple)
- phone_no (Multi-valued,Composite)
  - country_code
  - area_code
  - prefix
  - line_number
- gender (Simple)

Note:

1. Passengers are the actual travelers in the train.
2. Email_id is identified as a natural primary key.

Entity: Message
Attributes:

- subject_name(simple)
- message_detail(simple)

Relationship: **User** has **UserAccount**
Cardinality: one to one
Participation:
User entity has total participation with the UserAccount entity
UserAccount entity has total participation with the User entity

Relationship: **Train** stopsAt **Station**
Cardinality: many to many
Participation:
Train entity has total participation with the Station entity
Station entity has total participation with the Train entity

Relationship: **Train** startsAt **Station**
Cardinality: many to many
Participation:
Train entity has total participation with the Station entity
Station entity has total participation with the Train entity

Relationship: **User** books **Ticket**
Cardinality: one to many
Participation:
User entity has partial participation with the Ticket entity
Ticket entity has total participation with the User entity

Relationship: **User** cancels **Ticket**
Cardinality: one to many
Participation:
User entity has partial participation with the Ticket entity
Ticket entity has total participation with the User entity

Relationship: **User** checks **Availability**
Cardinality: one to many
Participation:
User entity has partial participation with the Availability entity
Availability entity has partial participation with the User entity

Relationship: **Passengers** have **Ticket**
Cardinality: many to many
Participation:
Passengers entity has total participation with the Ticket entity
Ticket entity has total participation with the Passengers entity

Relationship: **User**(admin) adds new **Train** details
Cardinality: one to many
Participation:
User entity has partial participation with the train entity
Train entity has total participation with the User entity

Relationship: **User**(admin) adds new **Station** details
Cardinality: one to many
Participation:
User entity has partial participation with the Station entity
Train entity has total participation with the User entity

Relationship: **User** send messages **to User(admin)**
Cardinality: many to one
Participation:
User entity has total participation with the User(admin) entity
User(admin) entity has total participation with the User entity

Relationship: **User**(admin) replies to the **User**
Cardinality: one to many
Participation:
User entity has total participation with the User entity
User entity has total participation with the User entity

Part 3: Perform logical design

LOGICAL DESIGN WITH HIGHEST NORMAL FORMS AND INDEXES

Table : User
Column :

- username
- first_name
- last_name
- email_id
- phone_no1
- phone_no2
- gender
- date_of_birth
- address_line1
- city
- state
- Zip_code
- User_type

Highest Normalization Level: 2NF
Justification:

- Zip code depends on city so there is transitive dependency.
- We can keep property table in 2NF as attribute zip code is not unique and creation of separate table for zip code will not be feasible as it will not reduce space requirement to store the data also it will increase unnecessary joins while creating the tables.

Index:username
Columns:username
Classification : Clustered
Justification:username is the primary key in this table and data in the table are ordered in the same way as the primary key. As being primary key it is a natural index to the table.

Index: name_user
Columns:last_name, first_name
Classification : Non-Clustered
Justification : Having first_name and last_name combined as an index helps increase the performance for searching for users with the given first and last name.
Create INDEX name_user on user(last_name,first_name);

Index:email_id
Columns:email_id
Classification : Non-Clustered
Justification : Search for user can be made using email id so that searching can be efficient.
Create INDEX email_id on user(email_id);

Table : UserAccount
Column :

username
password
Highest Normalization level : 4NF

Index:username
Classification : Clustered
Columns:username
Justification:
username is the primary key in this table and data in the table are ordered in the same way as the primary key. So, default index will get created for it.

Table: Station
Column :

station_no
station_name
address_line1
city
state
zip_code

Highest Normalization Level: 2NF
Justification:
● Zip code depends on city so there is transitive dependency.
● We can keep property table in 2NF as attribute zip code is not unique and creation of separate table for zip code will not be feasible as it will not reduce space requirement to store the data also it will increase unnecessary joins while creating the tables.

Index:station_no
Columns:station_no
Classification :Clustered
Justification : station_no is the primary key in this table , hence it is a natural index to the table.

Index:station_name
Columns:station_name
Classification :Non-clustered
Justification : Most of the time data will be searched using station_name and hence creating index for it.
Create INDEX station_name on Station(station_name);

Table: Train
Column :

- train_number
- train_name
- source
- destination
- total_capacity
- Base_fare

Highest Normalization level : 4NF

Index: Train_number
Columns:Train_number
Classification :Clustered
Justification : The train_number is a primary key to the table, hence it is a natural index to the table.

Table: StartsAt
Column:

- station_no (foreign key referencing to station_no from Station)
- train_number(foreign key referencing to train_number from Train)
- arrival_time
- departure_time
- hault

Highest Normalization level : 4NF

Index:station_no and train_number
Columns:station_no and train_number
Classification :Clustered
Justification : The train_number with station_number form the primary key to the table, hence it is a natural index to the table.

Table : Availability
Columns :

- train_number (foreign key referencing to train_number from Train)
- train_running_day
- no_available_seats

Highest Normalization level : 4NF as no_available_seats depends on both train_running_day and train_number.

Index: train_number , train_running_day
Columns:train_running_day, train_number
Classification : Clustered
Justification:train_running_day+ train_number is the primary key in this table and data in the table are ordered in the same way as the primary key. So, it is a natural index to the table.

Table: Ticket
Column:

- ticket_no
- train_number (foreign key referencing to train_number from Train)
- travel_day
- no_of_passengers
- ticket_status
- source
- destination

Highest Normalization level : 4NF

Index:ticket_no
Column : ticket_no
Classification :Clustered
Justification :ticket_no is the primary key in this table and data in the table are ordered in the same way as the primary key. So, it is a natural index to the table.

Index:train_number
Column : train_number
Classification :Non-Clustered
Justification :train_number is the foreign key in this table. So default index will get created.

Index:to_check_availability
Columns:train_number,travel_day,no_of_passengers
Justification: by combining train_number,travel_day,no_of_passengers columns as in index user will be able to query the availability details.

Table : BooksCancels
Column:

- username (foreign key referencing to username from User table)
- ticket_no (foreign key referencing to ticket_no from Ticket table, also primary key user_ticket_info table)

Highest Normalization level : 4NF

Index:ticket_no
Columns:ticket_no
Classification : Clustered
Justification : The ticket_no forms the primary key to the table, hence it is a natural index to the table.

Index:username
Column : username
Classification : Non-Clustered
Justification :username is the foreign key in this table. So default index will get created.

Table: Passenger_Ticket
Column:

ticket_no (foreign key referencing to ticket_no from Ticket table)
email_id (foreign key referencing to Passenger)

Note:

email_id, ticket_no will be the primary key of Ticket.

Index:ticket_email
Columns:ticket_no, email_id
Classification :Clustered
Justification : The ticket_no, email_id forms the primary key to the table, hence it is a natural index to the table.

Table : Passenger
Column:

email_id (primary key)
first_name
last_name
phone_no
gender

Note:

Highest Normalization level : 4NF

Index:email_id
Columns:email_id
Classification :Clustered
Justification :email_id is the primary key in this table and data in the table are ordered in the same way as the primary key. So, default index will get created for it .

Index:name
Columns:last_name, first_name
Classification : Non-Clustered
Justification : Often search for passenger name is made in passenger table using last_name and first_name so creating indexes for this reduces the search time.

Table: Message
Attributes:

Message_id
subject_name
sender_username(foreign key referencing to username from User table)
receiver_username(foreign key referencing to username from User table)
message_body

Note:

1. Highest Normalization level : 4NF

Index: Message_id
Columns:Message_id
Classification : Clustered
Justification : Message_id is the primary key in this table and data in the table are ordered in the same way as the primary key. So, default index will get created for it .

Index:sender_username
Columns:sender_username
Classification: Non-Clustered
Justification: Frequently message information will be searched with the use of senders username and hence creating index on it will improve the performance.
Create INDEX name on Message (sender_username);

Procedures:

1. **Availability :-**

Parameters : train_number(IN), travel_day(IN), train_run_day(IN)
Goal : Below steps are involved in this store procedures:-

1. Calculate sum of the number of passengers travelling in particular train on particular day and with the ticket status as 'confirmed'
2. Get the total capacity of train from the Train table.
3. Calculate the available seats in the Train on particular day.
4. After calculating, update the number of available seats in Availability table.

2. **Fare :-**

Parameters : train_number(IN), station_name(as source)(IN), station_name(as destination)(IN),tfare(OUT)
Goal :

1. Get the hault number of the station name entered as source for selected Train
2. Similarly get the hault number for destination.
3. Find out the number of in between stops.
4. Calculate the final fare per person using base fare and number of in between stops.
5. Return the ticket fare.

3. **Availabilitycancel :-**

Parameters : ticket_no(IN)
Goal:

1. This procedure will update the number of available seats when any user cancels the ticket.
2. Get the number of passenger travelling on particular ticket that user wants to cancel.
3. Get the train number and the travel day associated with the ticket so that resective available seats can be modified.
4. Add the number of passenger to the current availability once the ticket is cancelled so that same seats can be booked by other users.

4. **Inbetween:-**

Parameters : statio_number(IN)
Goal:

1. Select the station number of source and destination given by the user.
2. Select the train numbers visiting that particular source and destination.
3. By taking intersection of set of value provided by procedure for source and destination, trains travelling from source and destination can be found out.

Triggers:

Name: after_cacnel
Type : After UPDATE
Goal:

1. Once the user enters the ticket number that needs to be cancelled, update query will get executed.
2. As soon as the update query gets fired, Ticket with that ticket number gets updated and changes the ticket status to cancel.
3. If the given ticket exists and is updated then the call to procedure Availabilitycancel will be made and which will update the availability of the Train on particular day.

View :

Name : bookinghistory
Goal :

1. This particular view will give the information of the tickets booked(confirm as well as cancel) by the user.
2. View will also have the information about the passengers for which user has booked the ticket.

Query :
create view bookinghistory as

```
select
t.ticket_no,tn.train_name,t.train_number,t.travel_day,t.ticket_status,t.source,t.destination
,concat_ws(' ',u.first_name,u.last_name)as 'Booked By',group_concat(concat_ws('
',p.first_name,p.last_name)) as Passengers
from Passenger_Ticket pt
inner join Passenger p on p.email_id = pt.email_id
inner join Ticket t on t.ticket_no = pt.ticket_no
inner join Train tn on tn.train_number = t.train_number
inner join BooksCancels b on b.ticket_no = t.ticket_no
inner join User u on b.username = u.username
group by ticket_no;
```

Output:

```
3
4 • select * from bookinghistory;
5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ticket_no	train_name	train_number	travel_day	ticket_status	source	destination	username	Booked By	Passengers
AA090G3S744	charlotteexp	12345	Monday	Cacnel	Tempa	New York	sumedh	Sumedh Joglekar	Gaurav Mahadik
GG99090G3SP	charlotteexp	12345	Wednesday	Confirm	Charlotte	Cary	sumedh	Sumedh Joglekar	Bhargavi Nadella,Likitha kunapareddy
OAB990989	chicagoexp	23456	Monday	Cancel	Chicago	New York	gaurav	Gaurav Mahadik	Gaurav Mahadik,Samyuktha Nayakanti

Name: PassengerTicket

Goal:

1. By creating the view for the passenger ticket, ticket information about the individual passenger can be retrieved.
2. we can use this view to notify(by email) individual passenger about ticket information.

Query:

```
create view PassengerTicket as
select
t.ticket_no,t.train_number,p.first_name,p.last_name,p.email_id,t.travel_day,t.ticket_status,t.so
urce,t.destination from Ticket t
inner join Passenger_Ticket pt on pt.ticket_no = t.ticket_no
inner join Passenger p on p.email_id = pt.email_id;
```

Output:

```
4 • select * from PassengerTicket;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ticket_no	train_number	first_name	last_name	email_id	travel_day	ticket_status	source	destination
GG99090G3SP	12345	Likitha	kunapareddy	likitha@gmail.com	Wednesday	Confirm	Charlotte	Cary
AA090G3S744	12345	Gaurav	Mahadik	gmahadik@uncc.edu	Monday	Cacnel	Tempa	New York
OAB990989	23456	Gaurav	Mahadik	gmahadik@uncc.edu	Monday	Cancel	Chicago	New York
GG99090G3SP	12345	Bhargavi	Nadella	bhargavin@gmail.com	Wednesday	Confirm	Charlotte	Cary
OAB990989	23456	Samyuktha	Nayakanti	samyuktha@gmail.com	Monday	Cancel	Chicago	New York