SAN JOSÉ STATE UNIVERSITY

CMPE 275 - Term Project

Advisor : Professor Charles Zhang

California Ultra Speed Rail - Ticket Management System

Team:

Shiva Kumar Padma	- 011545313
Sai Ravi Tejabhishek Sreepada	- 011169002
Huzaifa Aejaz	- 011490453

Contents

Abstract

California Ultra-Speed Rail(CUSR) is an organization that runs trains in a Californian city, San Jose. California Ultra-Speed Rail(CUSR) Transport management system that manages and tracks ticketing. CUSR is a system with 26 stations from A to Z in alphabetical order with A being the northernmost station and Z being the southernmost station. There will be trains running in both the directions on their own tracks without sharing. Each train is either Northbound or Southbound. Ticketing has to be done for all kinds of trains with different conditions

The goal of this project is to build a ticket booking and management system that efficiently helps CUSR with several tasks such as ticket search, purchase, cancel for the users and train cancel, train capacity and system reset for the admin.

Motivation

The motivation is to develop a system where users would be able to search tickets, book tickets, cancel bookings. The system provides the admin the features to cancel trains and automatically rebook the passengers to the next available train.

The system is built using Spring boot which is Springs frameworks offering which promotes ‘Convention over Configuration’. The standalone application is wired up with a React based frontend application.

The other main reason for the development of this application was to learn a variety of technologies that go into the development of a distributed system. From complex search to registration, cancellation , authentication are all handled in the application.
The end product is a distributed system using varied technologies which communicate using the HTTP protocol without any processing latencies and halting.

Introduction

California Ultra-Speed Rail(CUSR) application lets users to search, book, manage the booking , cancel the bookings for trains.The data for this system consists of entries for trains which travel between 26 stations from A - Z. The trains are can be classified into two types based on the direction of their travel 1. Northbound and Southbound. In addition to this the trains are of two types 1. Regular and 2. Express.

The Regular trains run for every 15 minutes and Express trains run at whole hours and stop at only F, K, P, U, and Z. The trains can be booked searched in two different manners, 1. One-Way and 2. Round Trip.

As the name suggests these trip types can be used to have search results for a one way trip or a round trip. Apart from these we can have many other search criteria such as Exact Time and Number of Connections which can be used to filter results to meet specific criteria.

The project also provides additional features for the user to cancel and rebook the train. Apart from this the admin can cancel trains and also reset the entire system. An automatic ticket emailing system is in place which will send emails to the user on train cancellation , booking and rebooking etc. The system also has Google OAuth and Facebook OAuth for providing authentication. The users email id is used to process all the requests and the tickets are also mailed to the email id of the user.

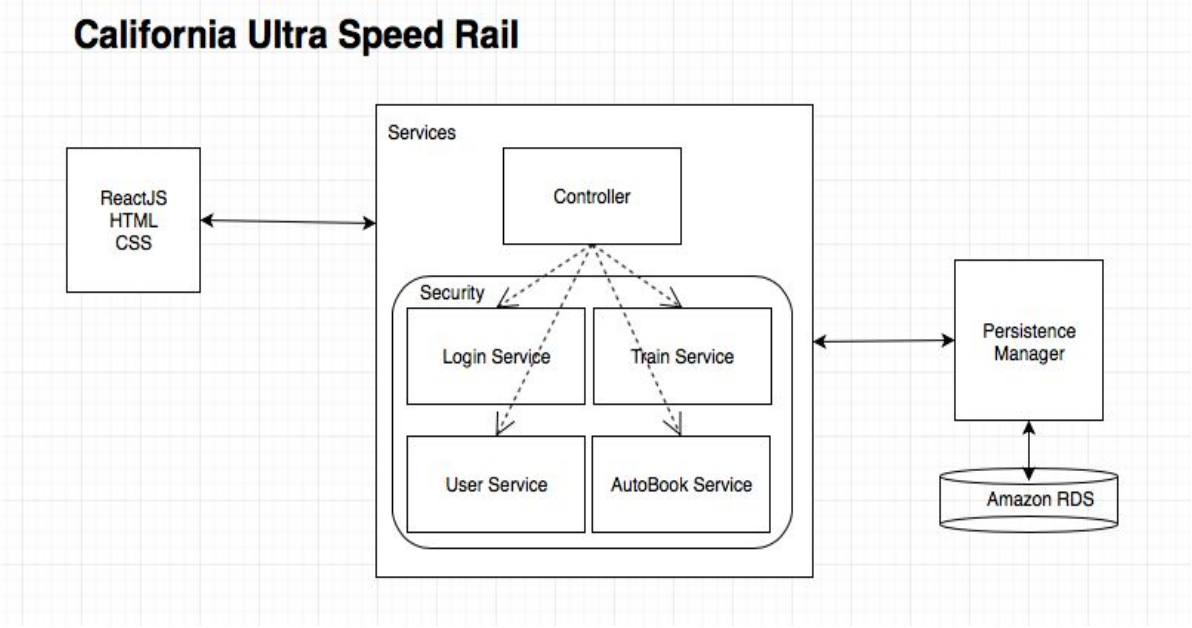
Technical Design

Backend **Restful API** Design:
Main URL: http://ec2-54-193-54-75.us-west-1.compute.amazonaws.com:9090

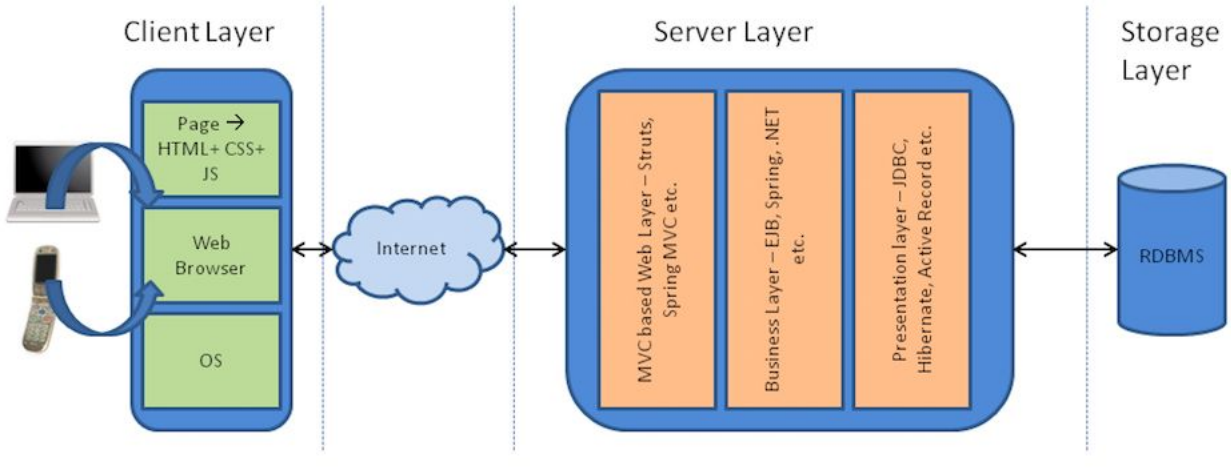
URL	Request Type	Params	Response	Description
/train	GET	depStation arrStation type depTime exactTimeFalg NumofConnections tripDate	200 : OK 404: Not Found 400: Bad Request 500: Internal Server Error	This URL is used to fetch search results
/train/cancel	POST	trainName tripDate	200 : OK 404: Not Found 400: Bad Request 500: Internal Server Error	This URL is cancel a train and automate rebooking for existing passengers.
/ticket	POST	userName noOfPassengers passengerNames oneWayTripTime returnTripTime totalTripPrice depDate arrDate oneWayTrips returnTrips	200 : OK 404: Not Found 400: Bad Request 500: Internal Server Error	This URL is used for booking the tickets
/ticket/reset	POST	capacity	200 : OK 404: Not Found 400: Bad Request 500: Internal Server Error	This URL is used to reset the application and changing the available

				number of seats and also deleting all booking data
/ticket/{ticket_id}	GET		200 : OK 404: Not Found 500: Internal Server Error	This URL is used to search ticket based on id.
/ticket	GET	userName	200 : OK 404: Not Found 500: Internal Server Error	This URL returns all bookings for user.
ticket/{ticket_id}/ cancel	POST		200 : OK 404: Not Found 400: Bad Request 500: Internal Server Error	This URL is used cancel a ticket booking by a user
/ticket/{ticket_id}/ rebook	POST		200 : OK 404: Not Found 400: Bad Request 500: Internal Server Error	This URL is to rebook an already cancelled ticket automatically.

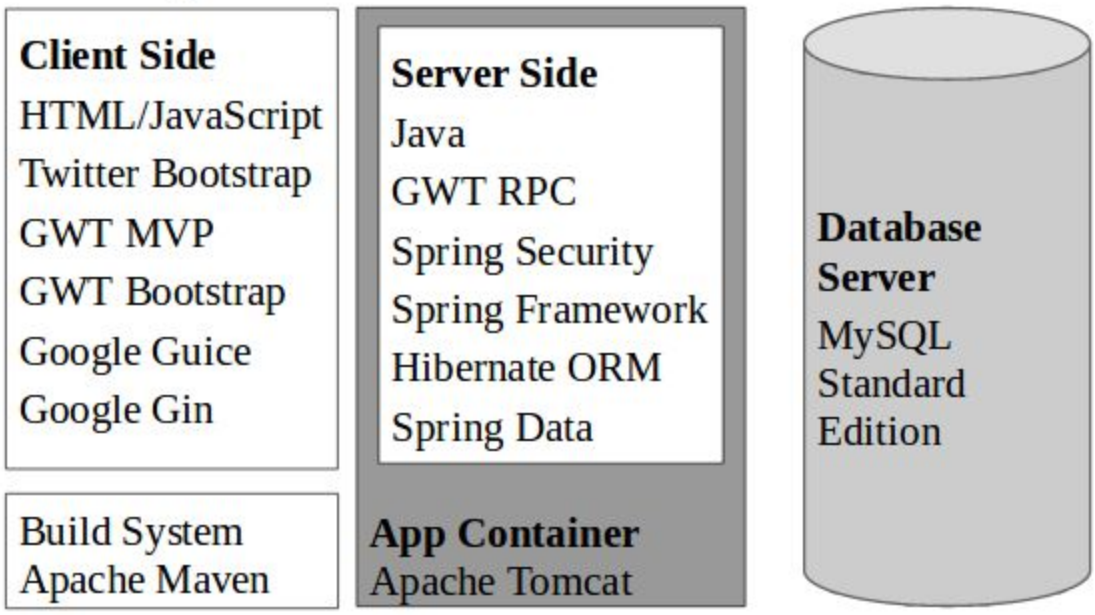
This Application has a three tier architecture

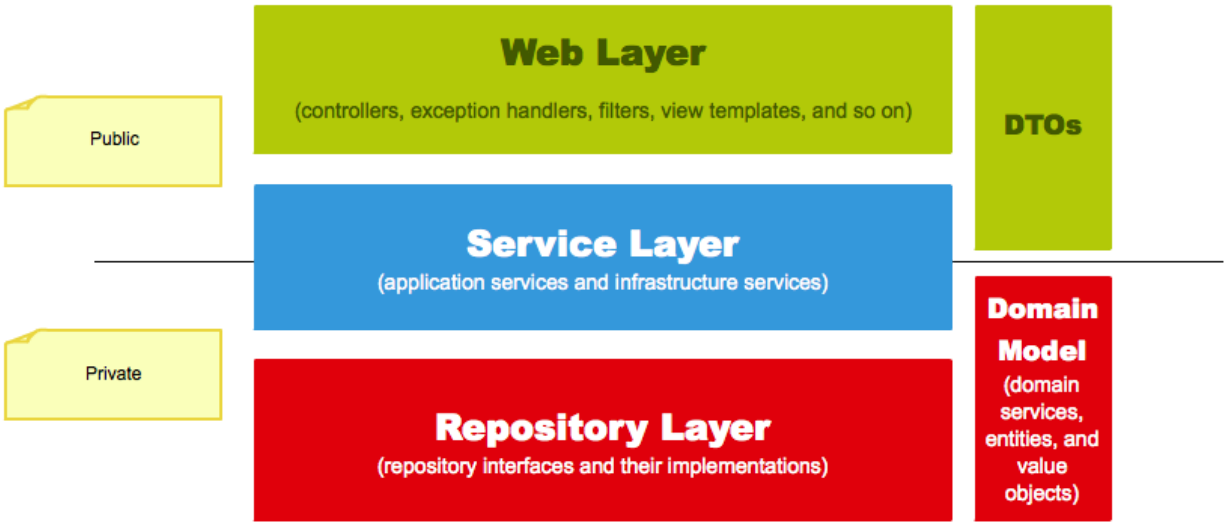


The controller handles all the requests. Based on the type of the request, controller redirects the request to its specific service module.



As explained in the figure above, Server layer has been implemented with Spring Boot and and for the storage, Amazon's RDS(MySQL) has been used that contains all the data.





Technology choices

Why Spring ?

Spring is the most popular framework when building Java Enterprise Edition (EE) applications. Its core feature is dependency injection which results in inversion of control.
Best thing about Spring is you can have numerous tools and APIs in your Java project and there would be one thing keeping them together, Spring.

Dependency Injection / Inversion of Control

The core features of Spring are:

- Access the database from Java application - Spring's JDBC or ORM modules.
- Logging feature in your Java application - Spring's AOP module.
- Creation of a web based Java application - Spring's Web Module.
- Messaging-based application - Spring's Messaging module.
- Implementing security features in Java application - Spring's Security module.

Spring is the most comprehensive framework you could use to build your Enterprise level Java application.

And this is the reason we chose Spring over others.

Why React?

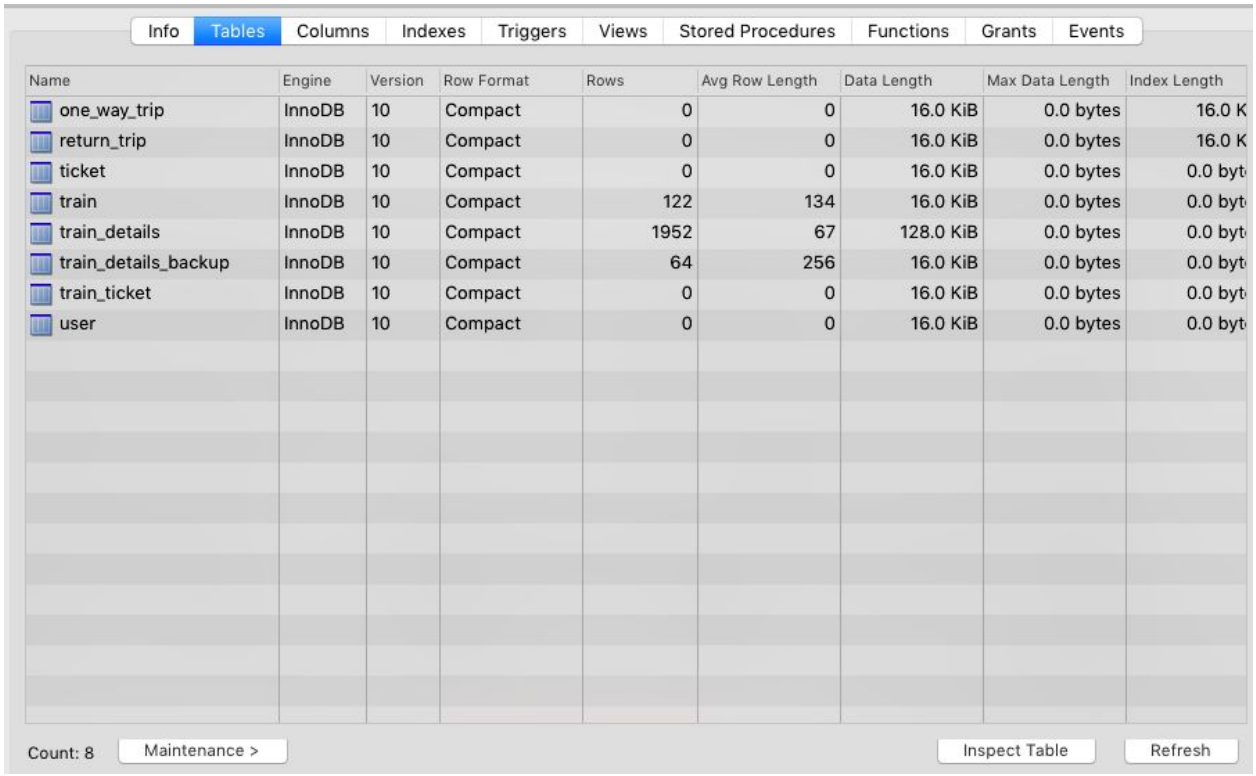
React is a javascript library that can be used to build user interfaces. The user interfaces are highly interactive and efficiently update according to the state changes.
React is also component based and hence we can create a number of components having their own state and management of those states. We can push the state changes through props so that other components may react to the state changes and provide their response. Rich data can also be passed through props through the layers of components.

Database

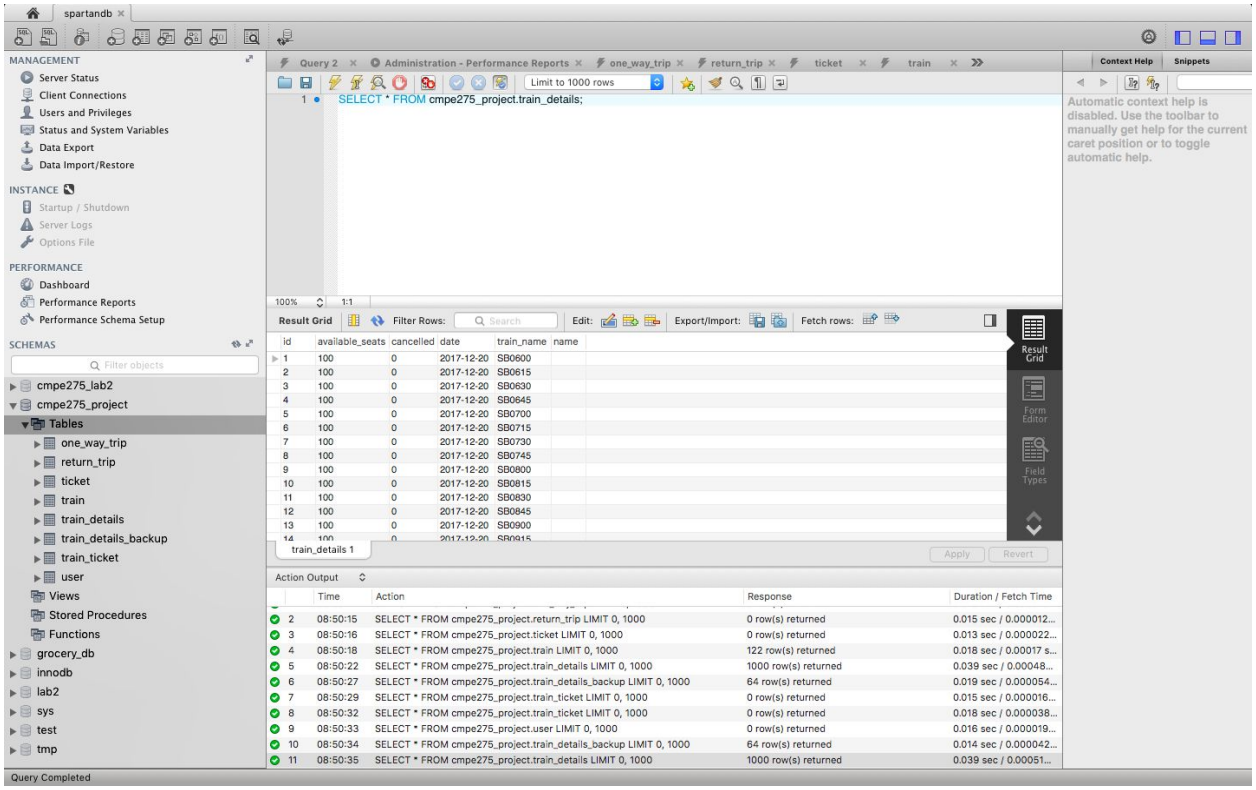
We used MySQL as the database provided by Amazon. RDS (Relational Database Service) is an Amazon Web Services offering. In short, it is a Database as a Service, where Amazon deploys and operates your database. It takes care of tasks like backup and patching the database software, as well as high availability. A few databases are supported by RDS, we are mainly interested in MySQL although - Amazon supports MySQL and MariaDB.

Database Design

Tables:



Data:



Implemented Stored Procedures to seed train's data and Trigger to implement autobook feature.

1.Stored procedure to insert Southbound and Northbound trains -

```
CREATE PROCEDURE insertTrains()
BEGIN
DECLARE n_sb varchar(30);
DECLARE date_to_insert date;
DECLARE time_iterator time;
DECLARE time_string varchar(30);
DECLARE i int default 0;
SET time_iterator := STR_TO_DATE('06:00','%H:%i');
SET i := 0;
WHILE i < 61 DO
        SELECT TIME_FORMAT(time_iterator,'%H:%i') into time_string from dual;
        SELECT CONCAT('SB',time_string) INTO n_sb FROM dual; #NB
        SET n_sb := REPLACE(n_sb, ':', '');
        INSERT INTO cmpe275_project.train(name,origin,start_time,train_type)
values(n_sb,'A',time_iterator,'Regular');
        SELECT DATE_ADD(time_iterator, INTERVAL 15 MINUTE) INTO time_iterator FROM dual;
        SET i := i + 1;
END WHILE;
END
```

2. Stored procedure to insert train details for -

```
CREATE PROCEDURE insertDetails()
BEGIN
DECLARE n_sb varchar(30);
DECLARE date_to_insert date;
DECLARE time_iterator time;
DECLARE time_string varchar(30);
DECLARE i int default 0;
DECLARE j int default 0;

SET date_to_insert :=  STR_TO_DATE('12/20/2017', '%m/%d/%Y');
SET j := 0;
WHILE j<15 DO
   SET i := 0;
         SET time_iterator := STR_TO_DATE('06:00','%H:%i');
   WHILE i < 61 DO
                  SELECT TIME_FORMAT(time_iterator,'%H:%i') into time_string from dual;
                  SELECT CONCAT('NB',time_string) INTO n_sb FROM dual; #SB
                  SET n_sb := REPLACE(n_sb, ':', '');
                  INSERT INTO cmpe275_project.train_details(train_name, date,cancelled) values
(n_sb,date_to_insert,FALSE);
         SELECT DATE_ADD(time_iterator, INTERVAL 15 MINUTE) INTO time_iterator FROM dual;
                  SET i := i + 1;
   END WHILE;
   SELECT DATE_ADD(date_to_insert, INTERVAL 1 DAY) INTO date_to_insert FROM dual;
   SET j := j + 1;
END WHILE;
END//
```

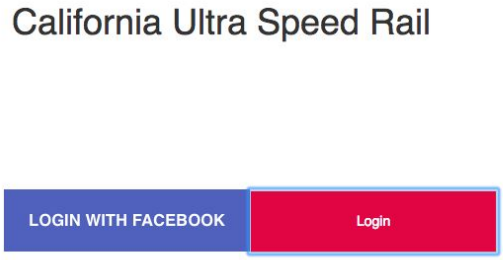
3. Automation Trigger to set auto book flag -

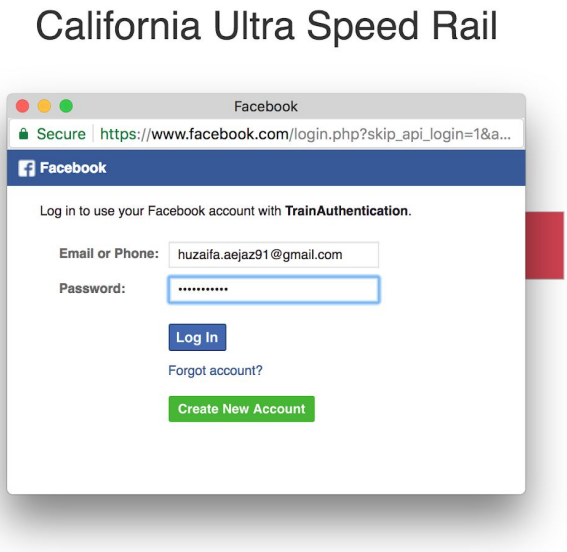
```
CREATE TRIGGER automateTrigger AFTER UPDATE ON cmpe275_project.train_details
FOR EACH ROW
BEGIN
 IF NEW.cancelled <> OLD.cancelled && NEW.cancelled = true
 THEN
   UPDATE cmpe275_project.train_ticket set  automate_rebook = true
   WHERE ticket_id in (select ticket_id
                                          from cmpe275_project.one_way_trip
                                          where name = NEW.train_name
                     )
   AND  dep_date = NEW.date;

END IF;
END;//
DELIMITER ;
```

Description of Features and related screenshots

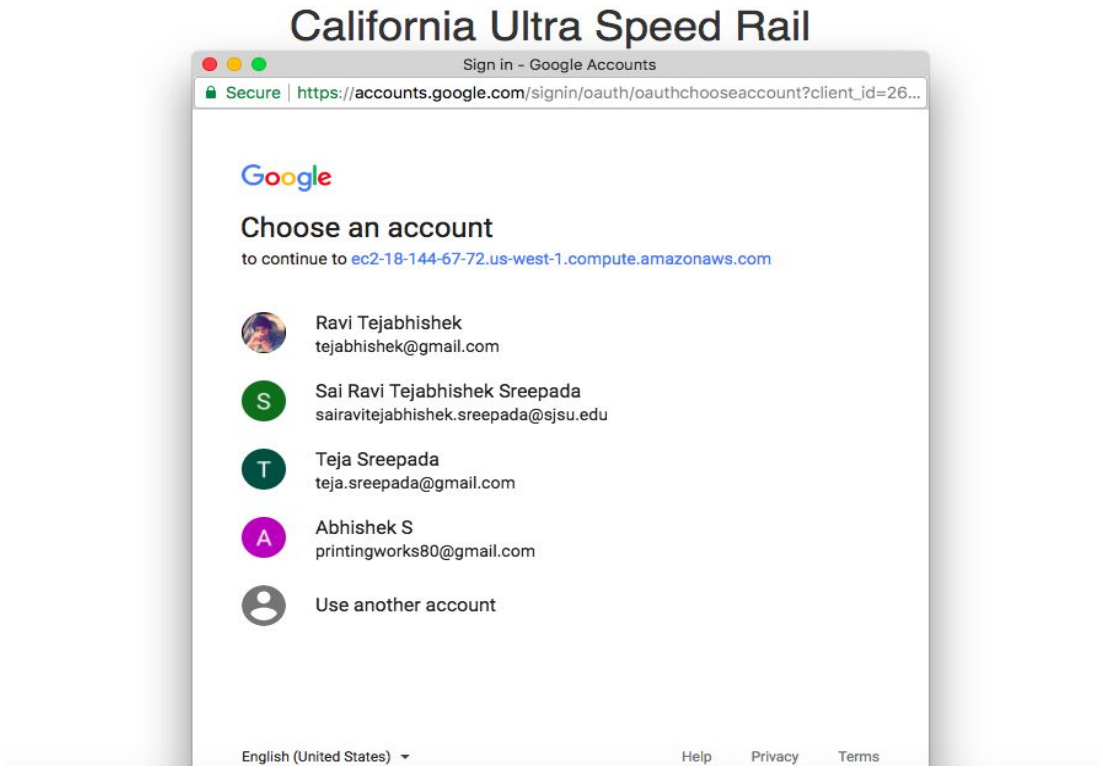
App URL : http://ec2-18-144-67-72.us-west-1.compute.amazonaws.com:3000/main

1. Login with facebook.

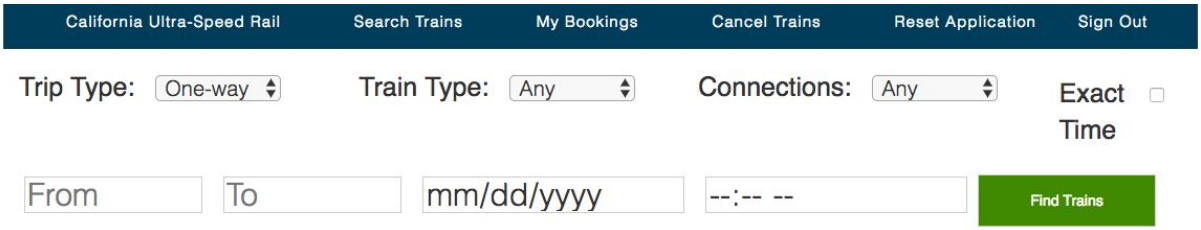




2. Login with Google

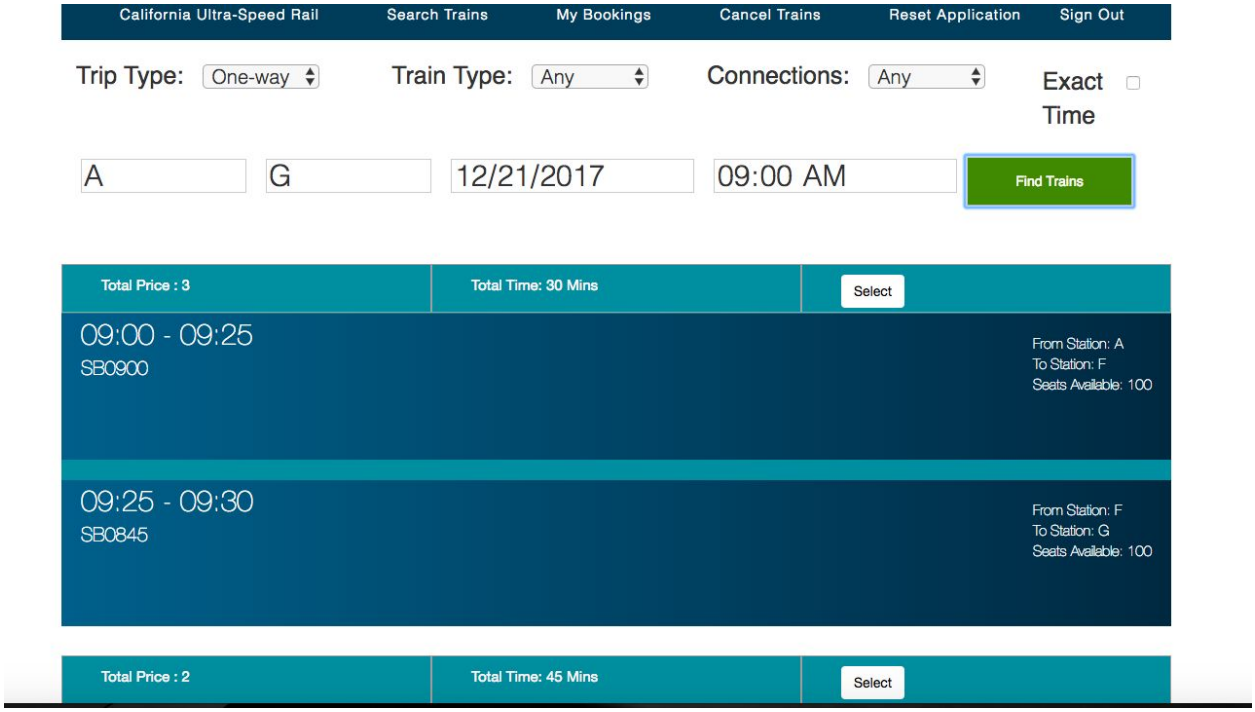


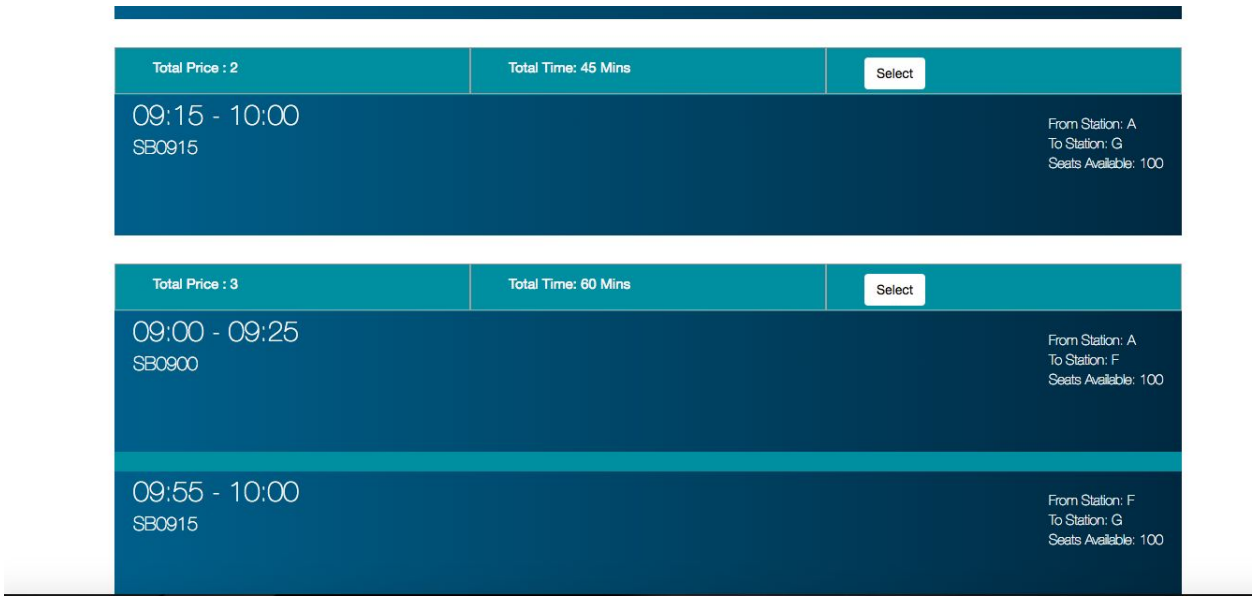
3. Search Ticket



4. Search Results

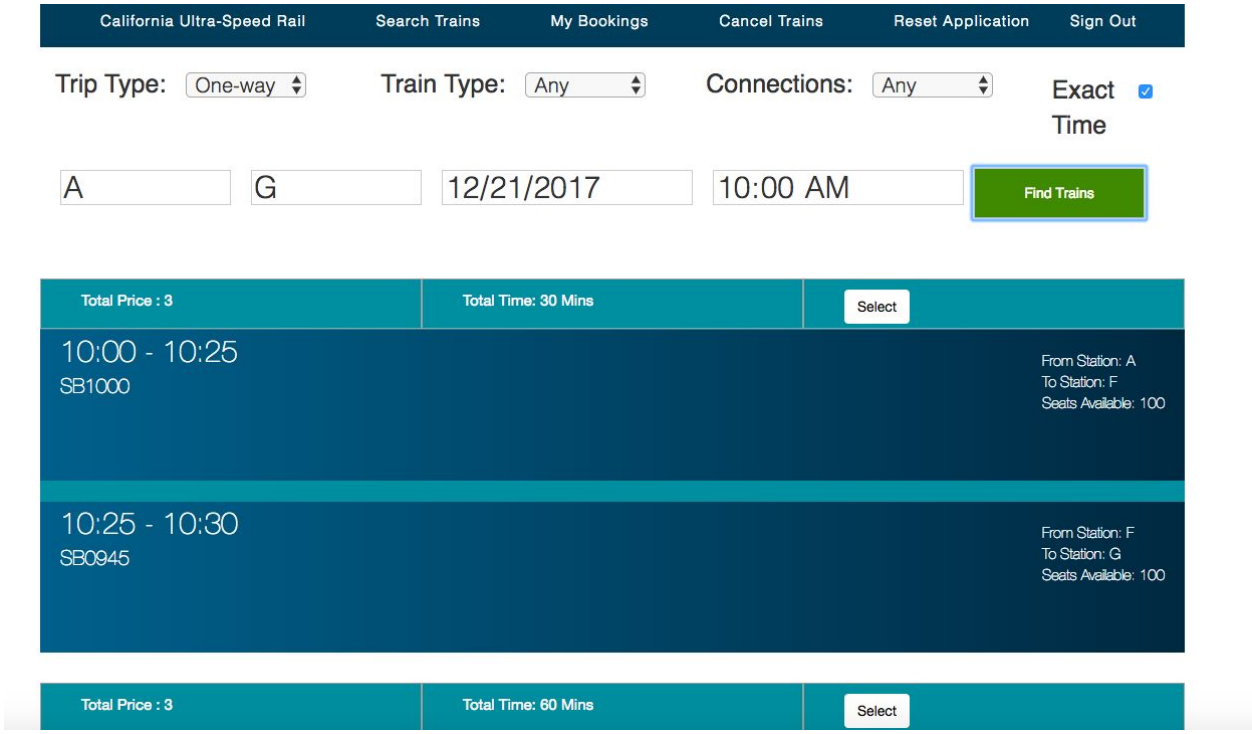
On clicking find search the API gets called which fetches the results sorted according to ascending order and are five in number. These results are then populated on the screen.





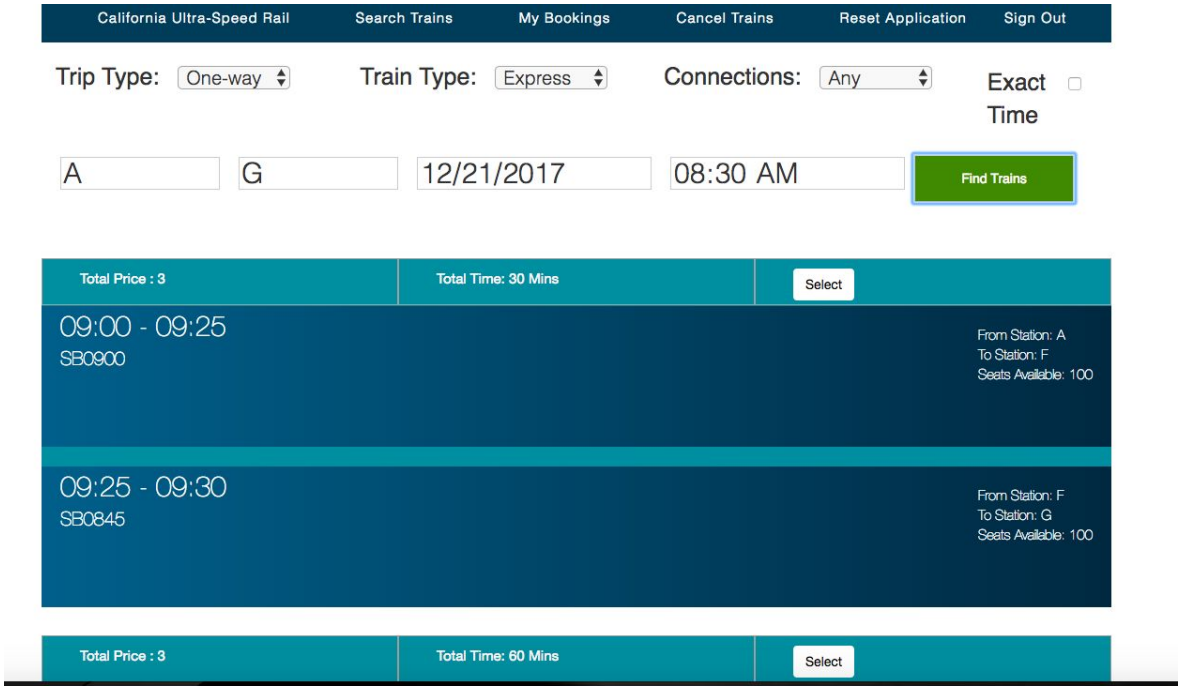
5. Search Results with exact time

The exact time checkbox can be used to obtain results matching only the exact time entered by the user.

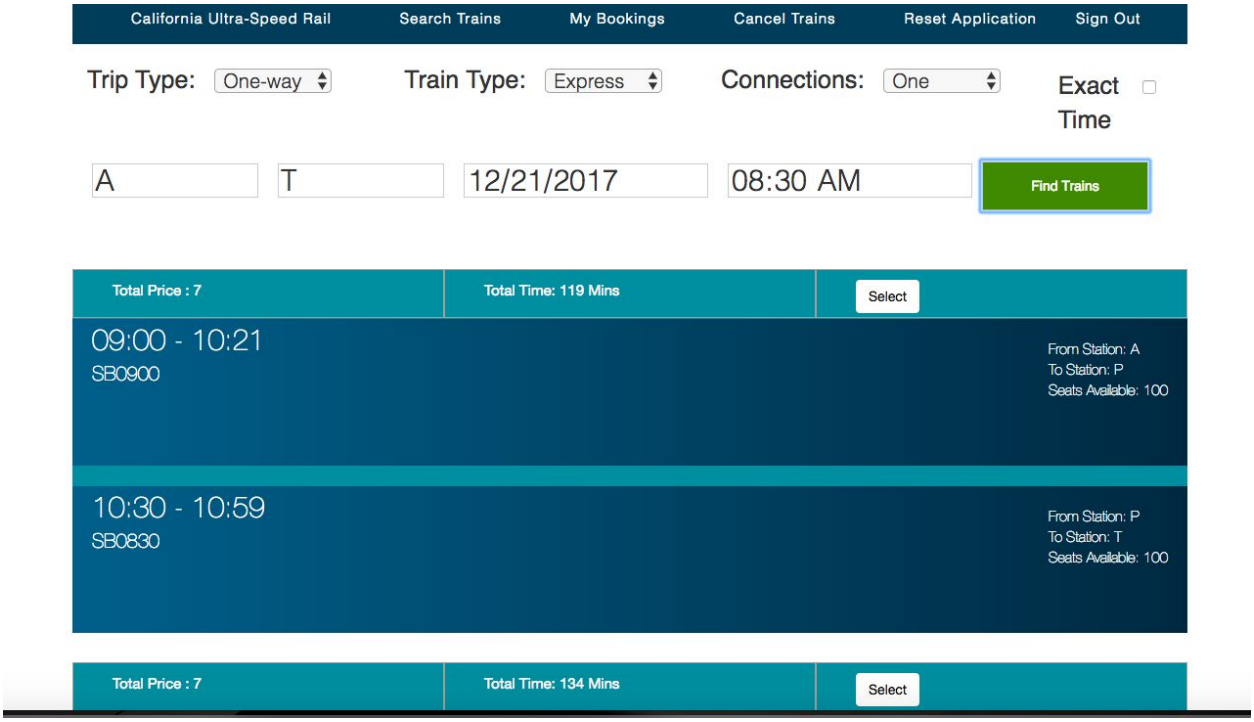


6. Search Results with Express Trains

The

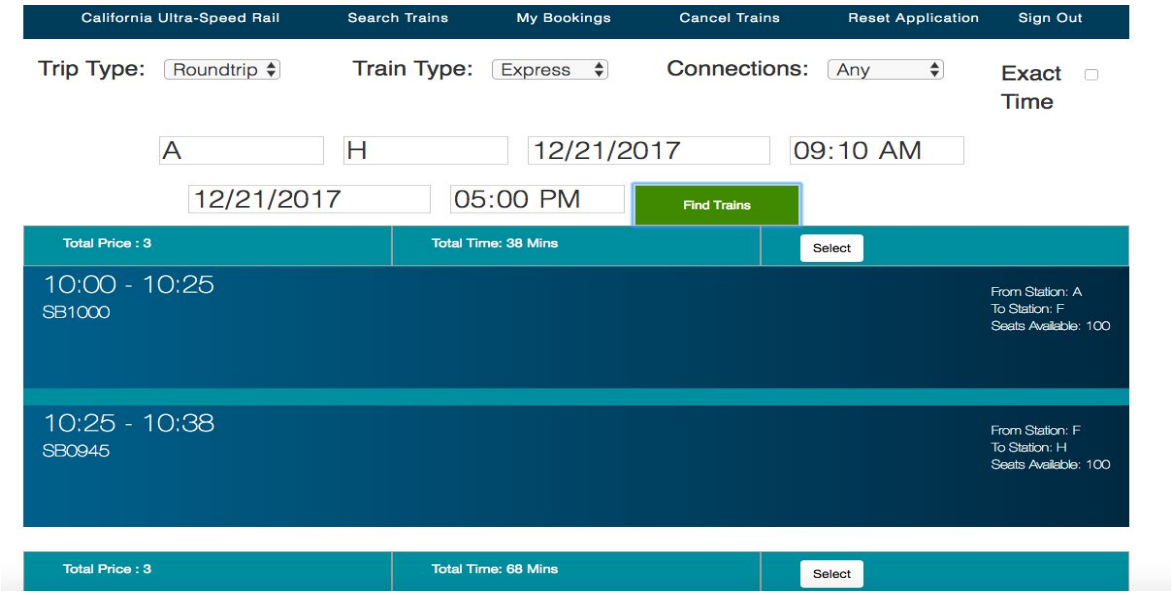


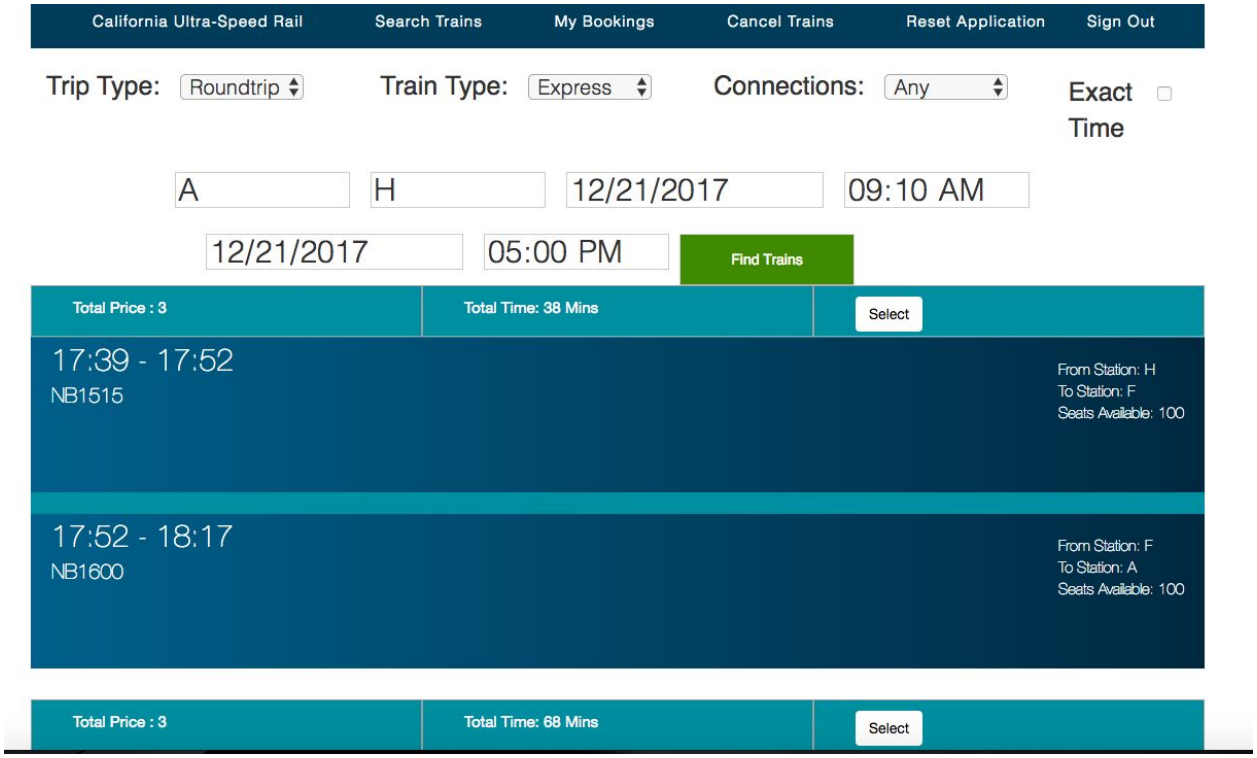
7. Search results with connections specified



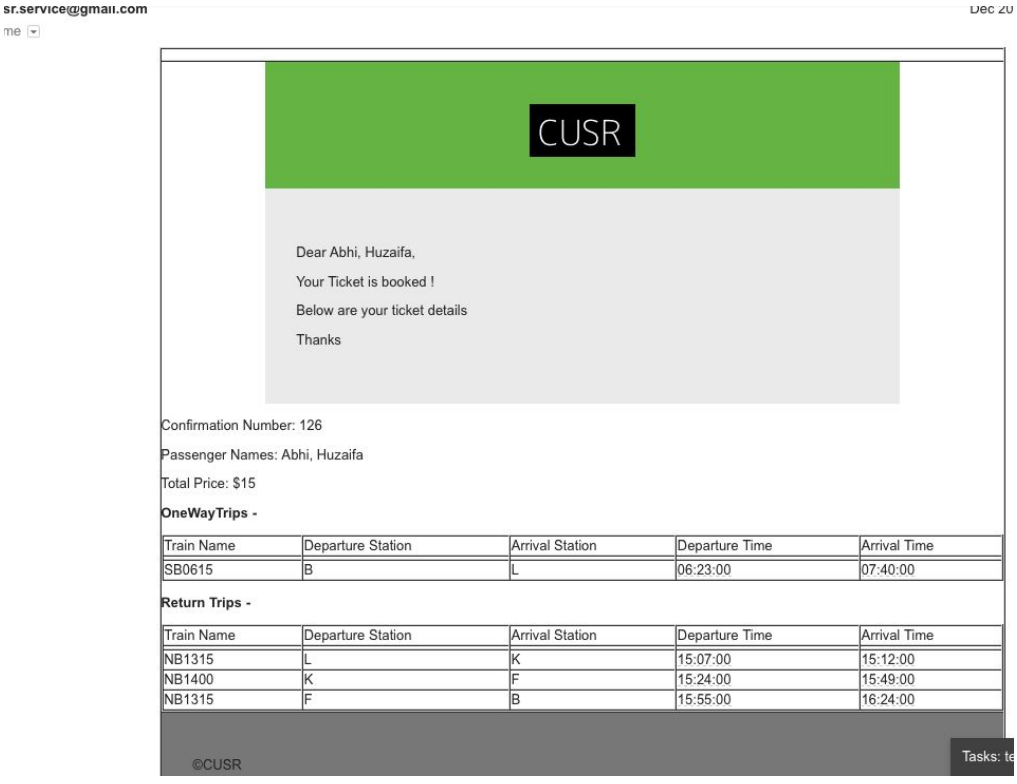
8. Search Results for round trip:

The search results for the round trip are displayed with the first trip results showing up and on clicking ‘Add to cart’ the page for displaying the results for the return trip will be displayed. Then the user can choose the number of passengers in the add to cart page and complete the booking.

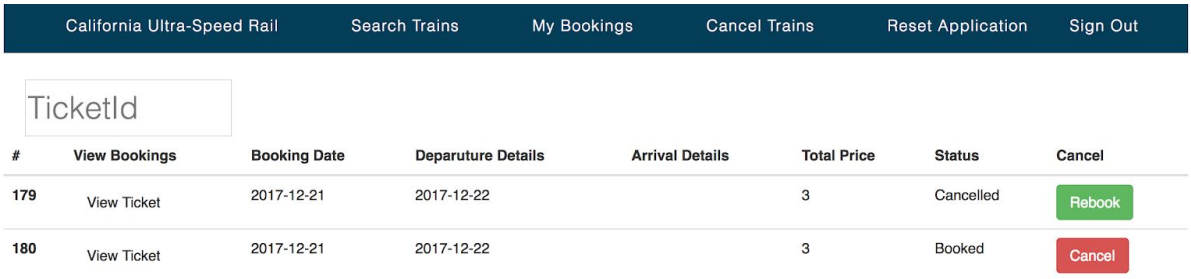




9. Email Confirmation

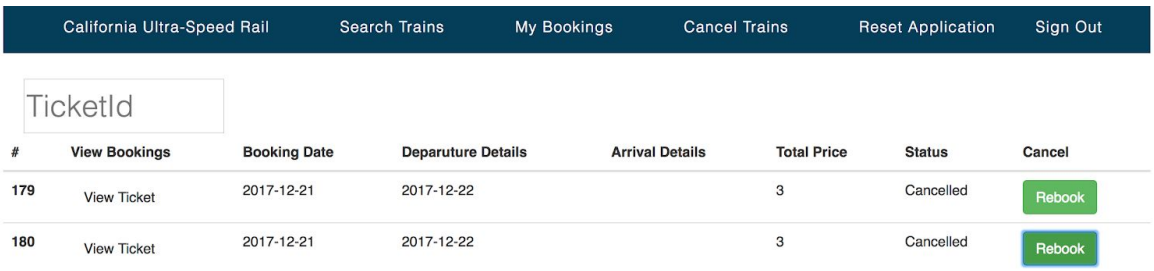


10. My Bookings

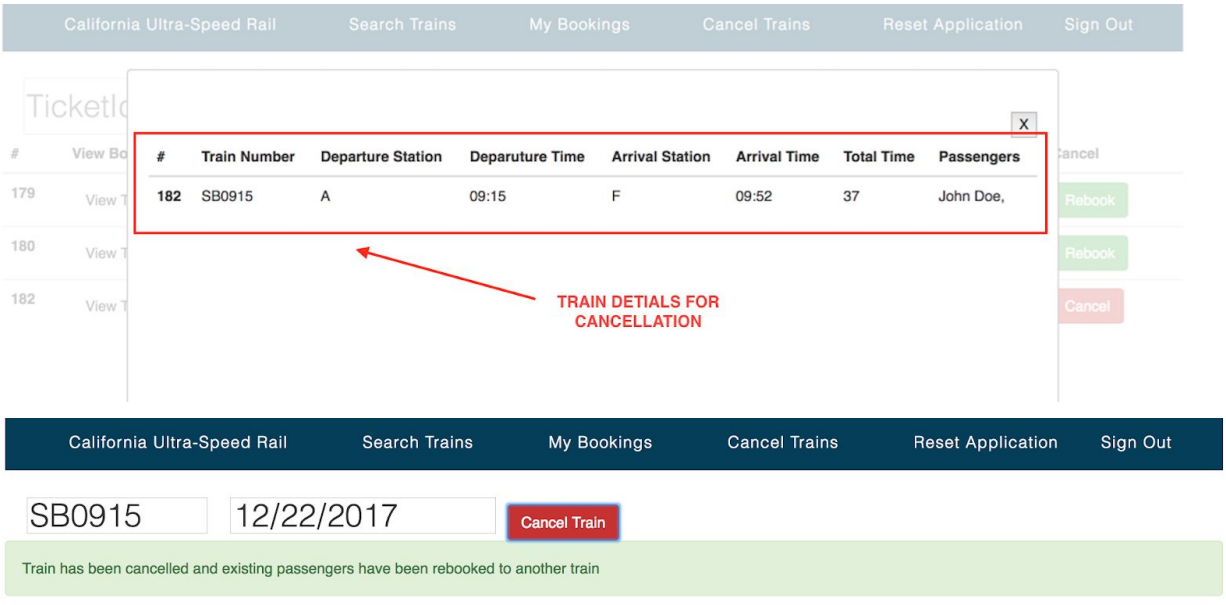


11. Cancel Bookings

On clicking cancel the train gets cancelled and a rebook button replaces the cancel button

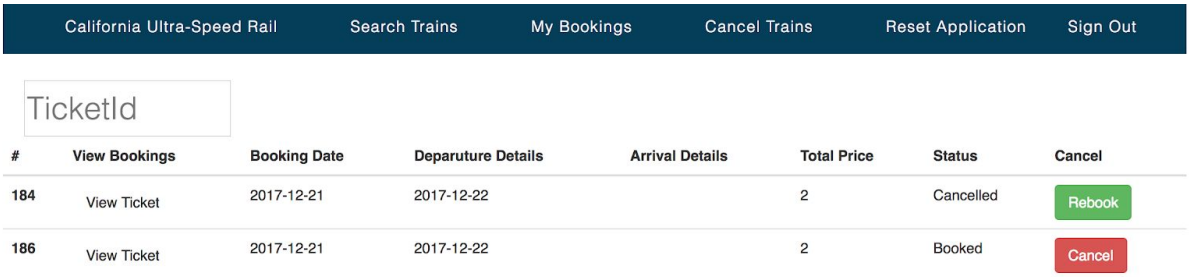


12. Cancel Train



13. Special Feature (AutoBooking)

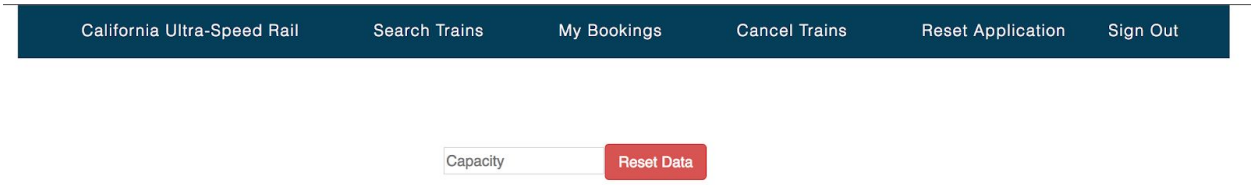
The train gets automatically rebooked for passengers if it gets cancelled as can be seen below



The passengers of the train that got cancelled get automatically rebooked to another train.

14. Reset Application

The train gets reset to the capacity specified and the bookings get cancelled using this.



Testing Plan and Results

Test case	Expected result	Pass/Fail
Exact time flag test	Only the tickets which meet the exact time should be returned	Pass
Express Train test	Only the express trains that run on whole hours should be returned	Pass
Regular Train test	Only the regular trains that run every 15 minutes should be returned	Pass
Number of connections test	The test results should be filtered based on the criteria chosen in the dropdown for ‘None’,’Any’ and ‘One’	Pass
One-way test	A single train should be booked in one direction	Pass
Round trip test	Two trains must be booked for the front and back trip	Pass
Email test	Emails should be sent on booking cancellation and automatic rebooking in case of train cancellation	Pass
Reset App test	The capacity of the trains should be reset and the bookings data must be cleared	Pass
Train Cancellation test	The train for the specified name and date should be cancelled and all the passengers must be rebooked to the next available train automatically and receive emails	Pass

Tested the application well and fixed all the bugs as a part of development of this project. Unit tests are done as soon as an API is developed and is integrated with other APIs and front end.

There were several challenges in integrating some components. But we have overcome them and built it successfully.

Lessons learned and possible future work

Lessons

- Learnt how to design an efficient database
- Learnt how to integrate components of different technologies
- Learnt several new technologies like Spring Boot, Hibernate and React while developing the application
- Learnt how to plan, design and document the code

Possible Future Work

- Payment APIs can be integrated to make it a complete application.
- Design can be modified to make it scalable supporting large number of users.
- Implement Redux data flow architecture