Last Updated: Tue Jul 27 2021 19:56:13 GMT+0000 (UTC)

Notion to Ebook Open Publication

Table of Contents

Table of Contents 2
FormAssist 3
- Objective- 3
- Workflow And Description of the App 4
- OCR 5
- Generating Links 5

GitHub Project Bot 6
- Inputs 6
- Organization-scope project 6
- Used Owned project 6
- Example - 6

RISCV Heritage 8
- Valid Instructions 8
- Team Members 8
- Technology Stack 8
- File Structure 8
- Features 9
- General Instructions To Run Locally 9
- Functionalities Our Project Offers 9

Train Ticketing 12
- Project Design Overview 12
- ER Diagram 12
- Key Features 13
- Technology Stack 14
- Functionalities Of Our Project 14
    - Functionalites for Admin 14
- Project Walkthrough 14
- API Endpoints Overview 17

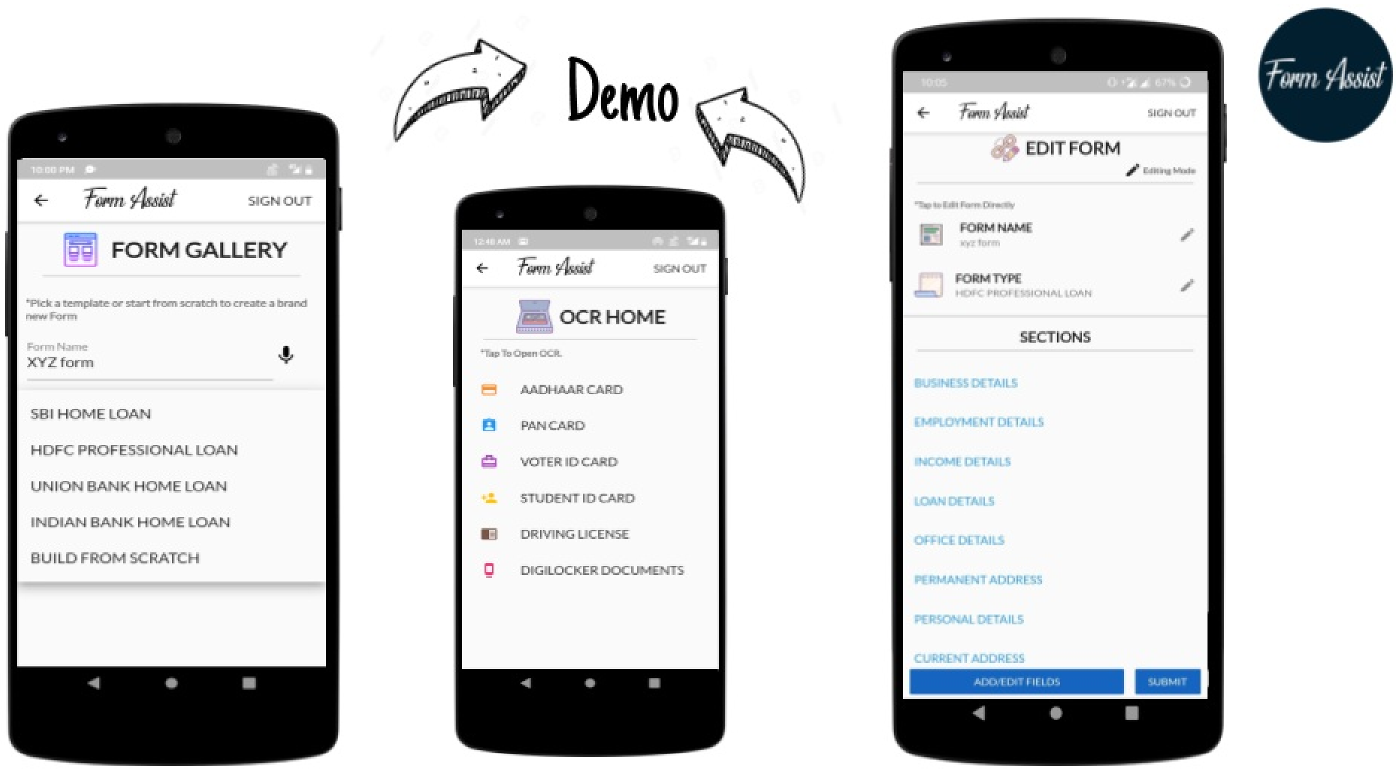
FormAssist

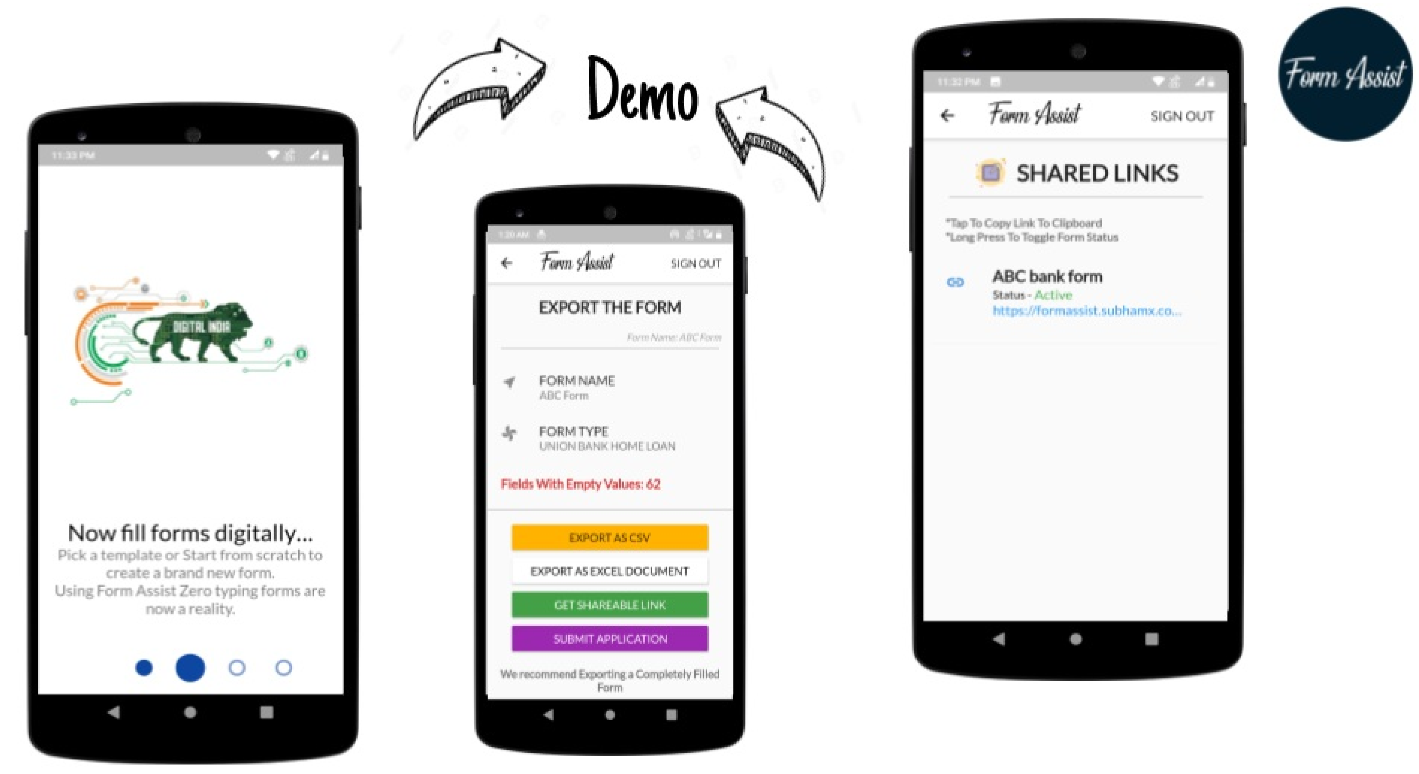
About	FormAssist allows users to add data both manually and through third-party integrations which will be stored in the database and can be used in future whenever needed.
Link	https://github.com/subhamX/FormAssist

FormAssist allows users to add data both manually and through third-party integrations which will be stored in the database and can be used in future whenever needed.

Objective-

Creating an app to automate the filling of bank forms and to reduce the redundancy in the job on both the sides i.e. on the user as well as bankside to a large extent. The app also tries to support the concept of zero typing to a large extent.





Now fill forms digitally...
Pick a template or Start from scratch to create a brand new form.
Using Form Assist Zero typing forms are now a reality.

EXPORT THE FORM
Form Name: ABC Form
FORM NAME
ABC Form
FORM TYPE
UNION BANK HOME LOAN
Fields With Empty Values: 62
We recommend Exporting a Completely Filled Form

SHARED LINKS
*Tap To Copy Link To Clipboard
*Long Press To Toggle Form Status
ABC bank form
Status - Active
https://formassist.subhamx.co...

Workflow And Description of the App

1. First, we begin by downloading and installing the app from the play store or any other valid available source.
2. Then when the installation process is completed we open the app and begin exploring it. The first screen(windows) which appears after successfully installing the app is the Launch screen where the user has been provided with three options:
   - Log-In (if he already has an account with our app.)
   - Sign -Up(if he is first time user of our app.)
   - Google Sign-In and Google Sign-Up (provided Sign-in and Sign-up facilities via Google Accounts as well.)
3. To make the process faster and typing free we have integrated the page with Speech to Text facilities and in case of Google Sign in only a single tap is required to select the Google Account.
4. After successfully signing in or signing up the user is landed to the home page of the app. The home page provides the user with three facilities which are also the main function of the app:
   - New Form (to create a new form)
   - Form List (to view, edit, delete and export already created forms)
   - Active Links (generating Form Links)
5. Now if the user wants to make a new form he selects the New Form option and a new screen pops up where the user provides with the name of the form and also selects the kind of template which the user wants to use. In the template selection field the user can select any one of the readymade templates provided smartly by us or he/she can also generate his/her own form by selecting Choose Custom Template. If the user selects Choose Custom Template then a new screen is launched which provides users with the various options of different sections namely Business, Employment, Loan, Permanent Address, etc which the user may want to include in his/her form (although none of the sections has been made mandatory by us since this form is completely user-generated.) and each section has related fields within it.
6. Now the user selects the sections which he deems necessary for his form and after selecting a particular section he is provided with underlying fields of the section from which he may select any number of fields and confirm the selection of fields by clicking on submit button and if the user wants to unselect some field

then a reset button has also been provided. If none of the sections or fields within different sections satisfies users requirements then he can make a completely new field by selecting Others option in the Sections page and give the name of the field (s) and our speech to text converter shall after confirmation from the user there itself will create a new field for the user's current form which can also be used in other forms as well as it is not deleted but remains save in others section.

7. When the user has added all the required fields in his form he can generate the blueprint of the form by clicking on the Create Form button which would lead the user to his form with the name of the form and all the section names been displayed. The user can then fill-up the form by selecting the different fields from different sections. The user can also edit the name of the form, the details of once filled fields and can also add new fields into the form by selecting Change Form Structure which would redirect the user to the Sections page.

The user need not type anywhere to fill details since speech to text converter has been provided which would make the app interactive and fast as well. Once all the details are filled the user can click on the Submit button which would finish the process of form creation and addition of details in the form.
The app redirects the user to the Form List screen where all the previously created forms along with the newly created form are present and the user can his form by just clicking on it which would show the various sections of the form along with the filled values if any. (The user can at any moment edit, add, delete any fields, sections from his form by clicking on edit form button which would redirect the user to the Sections Page where he could do the needful with all the options available in the bottom section of the screen in the form of buttons.)

OCR

Our app also provides the utility of prepopulating the user fields in the bank using OCR(Object Character Recognition) which would surely help the user save time as he would just have to upload the photo of his documents(Aadhaar Card, Pan Card, Credit Card) and our indigenous OCR model shall extract features from the image and prepopulate the fields in the form.

Generating Links

Once a user feels that his form to be forwarded to the bank is ready then he can go to the form list screen from the home page and select his form and click on the Export option which would generate a link of the form pdf in the Active Links screen which is ready to be shared on any of the mediums and the user will also be notified through a mail generated by the Form Assist Database providing the user with a ready to use link of the form.

- The user can at any time view his profile by clicking on My Data widget in the home screen.
- The user at no point has to type anything as other modes of taking information have been provided.
- The user at any point of time sign out of the app by clicking on the Sign-Out option provided at the navigation bar and can return to the home page by clicking on the Form Assist icon provided in the same.

GitHub Project Bot

About	� GitHub Project Bot fetches recently created Pull Requests and updates the Project Column
Link	https://github.com/subhamX/github-project-bot

Description: � `GitHub Project Bot` fetches recently created Pull Requests and updates the Project Column

Functionality: This GitHub action allows you to use any webhook events to automate the process of updation of project cards with recently created `Pull Requests`

Inputs

`REPO_URL:`
Complete URL of Repository whose Pull Request you want to automate

`PROJECT_URL:`
Complete URL of GitHub Project you want to use

`COLUMN_NAME:`
Name of an existing column of the project specifed above into which you want to place pull-requests

`ACCESS_TOKEN:`
An Access Token to create new Project Card

Organization-scope project

1. Set the URL of Complete URL of Repository whose Pull Request you want to automate to `REPO_URL`
2. Set the URL of Organization-scope project to `PROJECT_URL`
3. Set the name of an existing column of the project specifed above into which you want to place pull-requests to `COLUMN_NAME`
4. Use ${{ secrets.ACCESS_TOKEN }} to set the access token to `ACCESS_TOKEN`

Used Owned project

1. Set the URL of Complete URL of Repository whose Pull Request you want to automate to `REPO_URL`
2. Set the URL of User owned project to `PROJECT_URL`
3. Set the name of an existing column of the project into which you want to place pull-requests to `COLUMN_NAME`
4. Use ${{ secrets.ACCESS_TOKEN }} to set the access token to `ACCESS_TOKEN`

Example -

Scheduling the workflow to run at specific UTC time using `cron`.
The following cron schedule expression will run `At every 5th minute from 0 through 59` and will run the following `TWO STEPS`

```
name: Add Recently Created Pull Requests to Project Column
on:
  schedule:
    - cron: "0/5 * * * *"
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: Handle Repo1
      uses: subhamX/github-project-bot@v1.0.0
      with:
        ACCESS_TOKEN: ${{ secrets.ACCESS_TOKEN }}
        COLUMN_NAME: In Progress
        PROJECT_URL: <https://github.com/orgs/ORG_NAME/projects/1>
        # For User Owned Project -> PROJECT_URL: <https://github.com/users/UNAME/projects/1>
        REPO_URL: <https://github.com/ORG_NAME/repo1>
        # For User Owned Repo -> REPO_URL: <https://github.com/UNAME/repo1>
    - name: Handle Repo2
      uses: subhamX/github-project-bot@v1.0.0
      with:
        ACCESS_TOKEN: ${{ secrets.ACCESS_TOKEN }}
        COLUMN_NAME: In Progress
        PROJECT_URL: <https://github.com/orgs/ORG_NAME/projects/1>
        # For User Owned Project -> PROJECT_URL: <https://github.com/users/subhamX/projects/1>
        REPO_URL: <https://github.com/ORG_NAME/repo2>
        # For User Owned Repo -> REPO_URL: <https://github.com/UNAME/repo2>
```

RISCV Heritage

About	💻 A web simulator which converts the Assembly code written in RISCV syntax to Machine code.
Link	https://github.com/subhamX/riscv

The following website is a course project under the guidance of Dr T.V Kalyan. It is a web simulator which converts the `Assembly code` written `RISCV syntax` to `Machine code` and provides a user-friendly environment for its execution. The simulator implements pipelining, branch prediction and data forwarding, which can be enabled or disabled by the user.

Valid Instructions

R format:

add, and, or, sll, slt, sra, srl, sub, xor, mul, div, rem

I format:

addi, andi, ori, lb, ld, lh, lw, jalr

S format:

sb, sw, sd, sh

SB format:

beq, bne, bge, blt

U format:

auipc, lui

UJ format:

jal

Assembler Directives:

.text, .data, .byte, .half, .word, .dword, .asciiz

Team Members

Bharat Ladrecha	2018CSB1080
Subham Sahu	2018EEB1183

Technology Stack

1. Typescript
2. HTML/CSS
3. NodeJS (For Development Only)

File Structure

```
src
│ README.md
│ tsconfig.json
│
└───assets
│ index.html
│
└───css
│ style.css
```

```
│
└───js
│ main.ts
│
└───encode
│
└───execute
```

encode contains files required to convert `Assembly Code` written in `RISC V` syntax into `Machine Code`

execute contains files required to execute the generated `Machine Code`

main.ts is the entrypoint of the application

Features

1. On pressing `Ctrl+S` on **Editor Pane** current file will be downloaded.
2. On pressing `Ctrl+S` on **Simulator Pane** Output file will be downloaded if it's assembled.
3. **Editor Pane** supports `Ctrl+/` as comment shortcut
4. On pressing `Ctrl+D` on **Simulator Pane** stats file will be downloaded provided that execution is complete.

General Instructions To Run Locally

1. If you do not have a copy of `RISCV Heritage` clone the repo and checkout `GUI` branch. If you do have the project then go to `step 2`

```
git clone URL

cd riscv
```

1. Install all dependencies

```
npm install
```

1. Now to run the development server run the following command in terminal

```
npm run start:dev
```

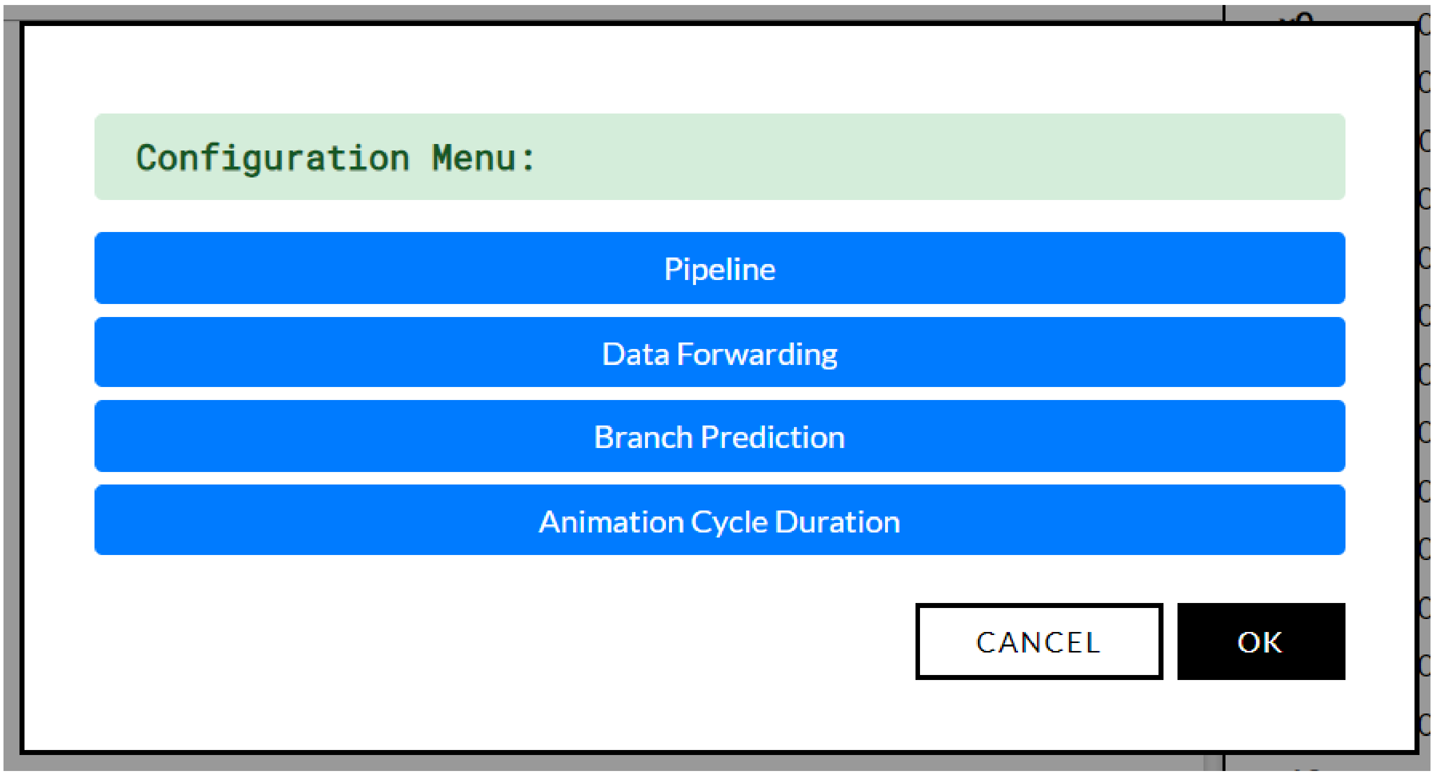
1. Now access the `RISCV Heritage` by visiting http://localhost:1234/
2. To build the project run the following command in terminal

```
npm run build
```

Functionalities Our Project Offers

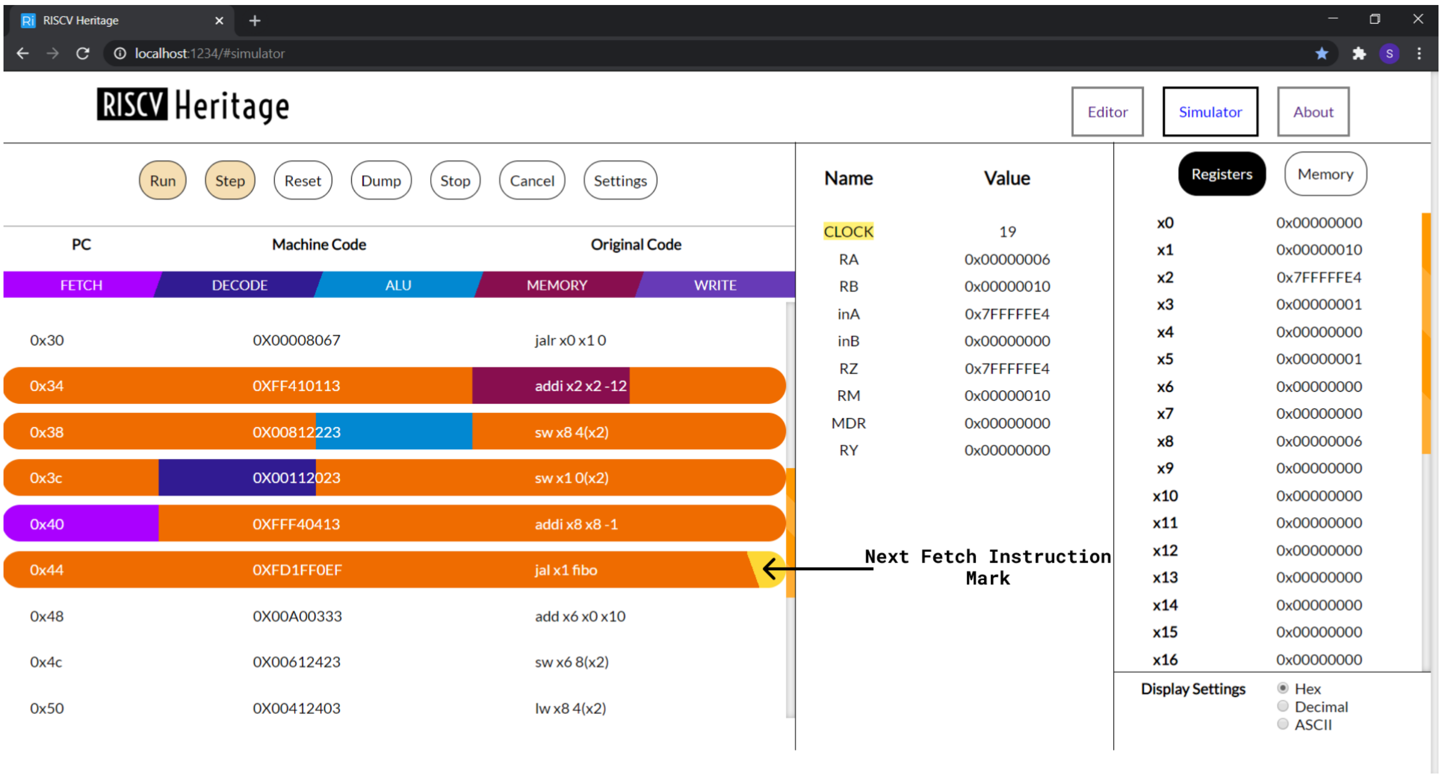
1. **BreakPoint:-** Users can use a breakpoint for debugging. Users can select as many breakpoints as s(he) wants. During the execution of instructions, the program will stop after the execution of selected instructions in the order of program flow.
2. **Run:-** To run the complete program in a single go, users can use it. For pipelined version the *animation cycle duration* can be tuned by the user from the settings menu.
3. **Step:-** Run the program instruction wise(one instruction at a time).
4. **Stop:-** Stop the execution process using the stop button.
5. **Reset:-** Reset the current execution process. It will also reset the memory and registers state.
6. **Dump:-** Users can copy the machine code to clipboard using it.
7. **Cancel:-** Destroys the current execution and moves one step back.

8. **Settings:-** Execution configuration can be changed by clicking this button before assembling the code. If assembled then it shows the configuration in READ ONLY mode.



- We provide our user with some options to enable/disable

1. Pipelined execution
2. Data Forwarding
3. Branch prediction
4. Animation Cycle Duration, to increase or decrease program execution speed for pipelined execution
5. Each pipelined instruction execution passes through five stages, Fetch, Decode, ALU, Memory, and Write Back. All these stages are colour-coded. So when an instruction has completely executed in any stage, the instruction gets colour-coded with a colour corresponding to the pipeline stage.



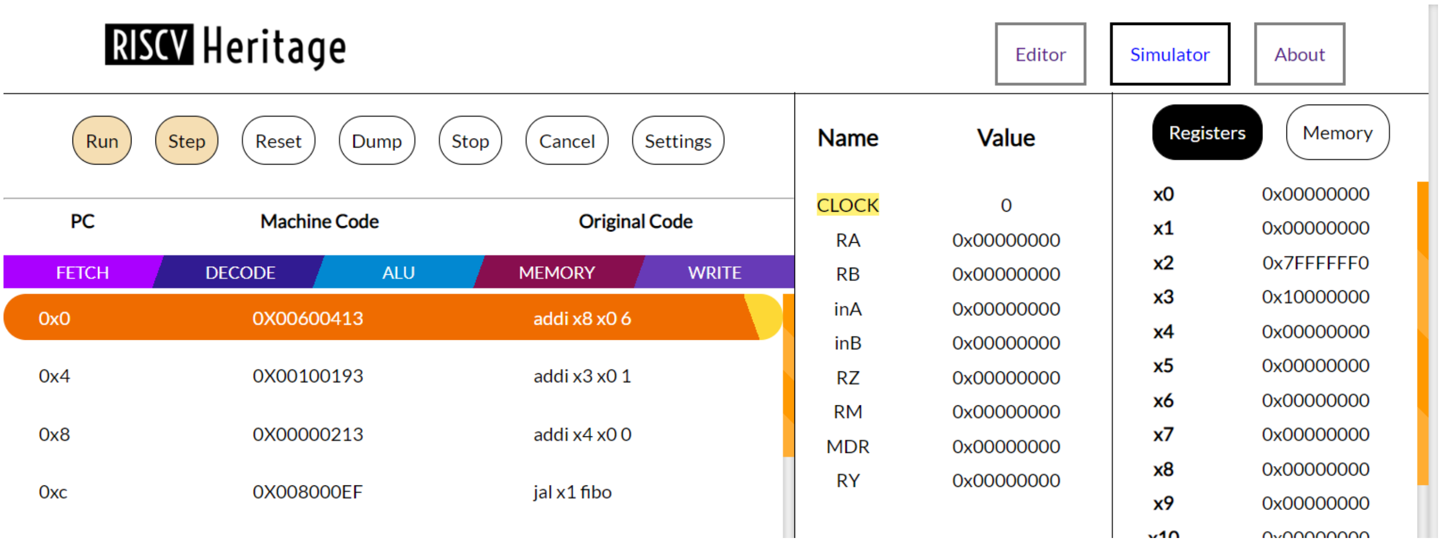
PC	Machine Code	Original Code
0x30	0X00008067	jalr x0 x1 0
0x34	0XFF410113	addi x2 x2 -12
0x38	0X00812223	sw x8 4(x2)
0x3c	0X00112023	sw x1 0(x2)
0x40	0XFFF40413	addi x8 x8 -1
0x44	0XFD1FF0EF	jal x1 fibo
0x48	0X00A00333	add x6 x0 x10
0x4c	0X00612423	sw x6 8(x2)
0x50	0X00412403	lw x8 4(x2)

Name	Value
CLOCK	19
RA	0x00000006
RB	0x00000010
inA	0x7FFFFFE4
inB	0x00000000
RZ	0x7FFFFFE4
RM	0x00000010
MDR	0x00000000
RY	0x00000000

x0	0x00000000
x1	0x00000010
x2	0x7FFFFFE4
x3	0x00000001
x4	0x00000000
x5	0x00000001
x6	0x00000000
x7	0x00000000
x8	0x00000006
x9	0x00000000
x10	0x00000000
x11	0x00000000
x12	0x00000000
x13	0x00000000
x14	0x00000000
x15	0x00000000
x16	0x00000000

Display Settings [✓ Hex] Decimal ASCII

1. We have added a separate pane for pipelined execution through which the user can see the current status of pipelined ISA.

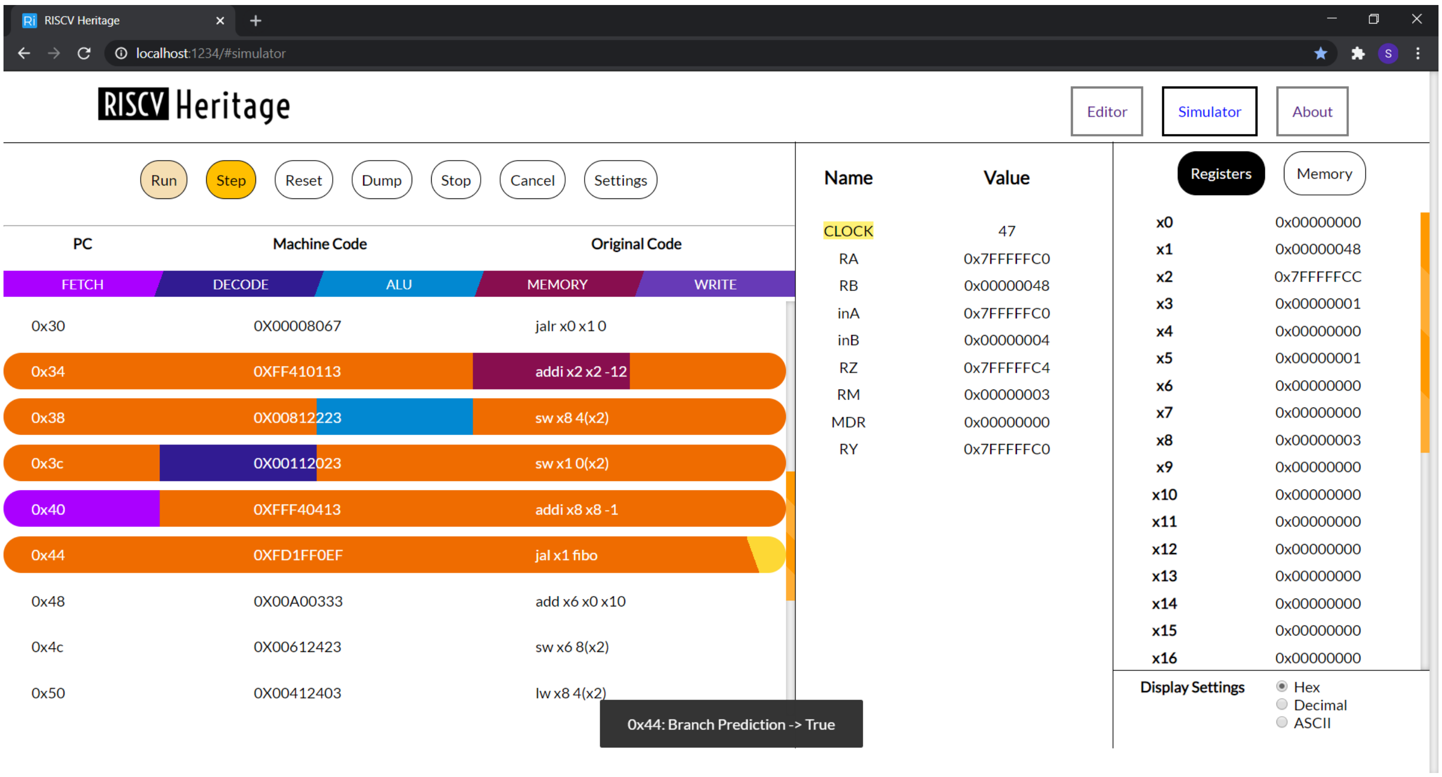


PC	Machine Code	Original Code
0x0	0X00600413	addi x8 x0 6
0x4	0X00100193	addi x3 x0 1
0x8	0X00000213	addi x4 x0 0
0xc	0X008000EF	jal x1 fibo

Name	Value
CLOCK	0
RA	0x00000000
RB	0x00000000
inA	0x00000000
inB	0x00000000
RZ	0x00000000
RM	0x00000000
MDR	0x00000000
RY	0x00000000

x0	0x00000000
x1	0x00000000
x2	0x7FFFFFF0
x3	0x10000000
x4	0x00000000
x5	0x00000000
x6	0x00000000
x7	0x00000000
x8	0x00000000
x9	0x00000000

1. To make the app more informative, whenever there is a stall, data forwarding, pipeline flushing or branch prediction etc. Our application shows a toast to notify such event.



PC	Machine Code	Original Code
0x30	0X00008067	jalr x0 x1 0
0x34	0XFF410113	addi x2 x2 -12
0x38	0X00812223	sw x8 4(x2)
0x3c	0X00112023	sw x1 0(x2)
0x40	0XFFF40413	addi x8 x8 -1
0x44	0XFD1FF0EF	jal x1 fibo
0x48	0X00A00333	add x6 x0 x10
0x4c	0X00612423	sw x6 8(x2)
0x50	0X00412403	lw x8 4(x2)

0x44: Branch Prediction -> True

Name	Value
CLOCK	47
RA	0x7FFFFFC0
RB	0x00000048
inA	0x7FFFFFC0
inB	0x00000004
RZ	0x7FFFFFC4
RM	0x00000003
MDR	0x00000000
RY	0x7FFFFFC0

x0	0x00000000
x1	0x00000048
x2	0x7FFFFFCC
x3	0x00000001
x4	0x00000000
x5	0x00000001
x6	0x00000000
x7	0x00000000
x8	0x00000003
x9	0x00000000
x10	0x00000000
x11	0x00000000
x12	0x00000000
x13	0x00000000
x14	0x00000000
x15	0x00000000
x16	0x00000000

Display Settings [✓ Hex] Decimal ASCII

1. Memory Segment displays only those values into which something is explicitly written during program execution or in the data segment and all other values which are not shown are `0x00`
2. Although code segment data is shown in memory it cannot be retrieved by the user, by accessing that location. This is to ensure that there is no structural hazard.
3. On any error, the program will alert the user and will not proceed further for execution
4. Any instruction which uses **Double** like `sd, ld` are invalid since it is `32-bit system`
5. We have assumed that hazards are those dependencies which causes our pipeline to stall. So, any data dependency which are successfully handled by data forwarding won't be considered as a data hazard.

Latest Deployed version: RISCV Heritage

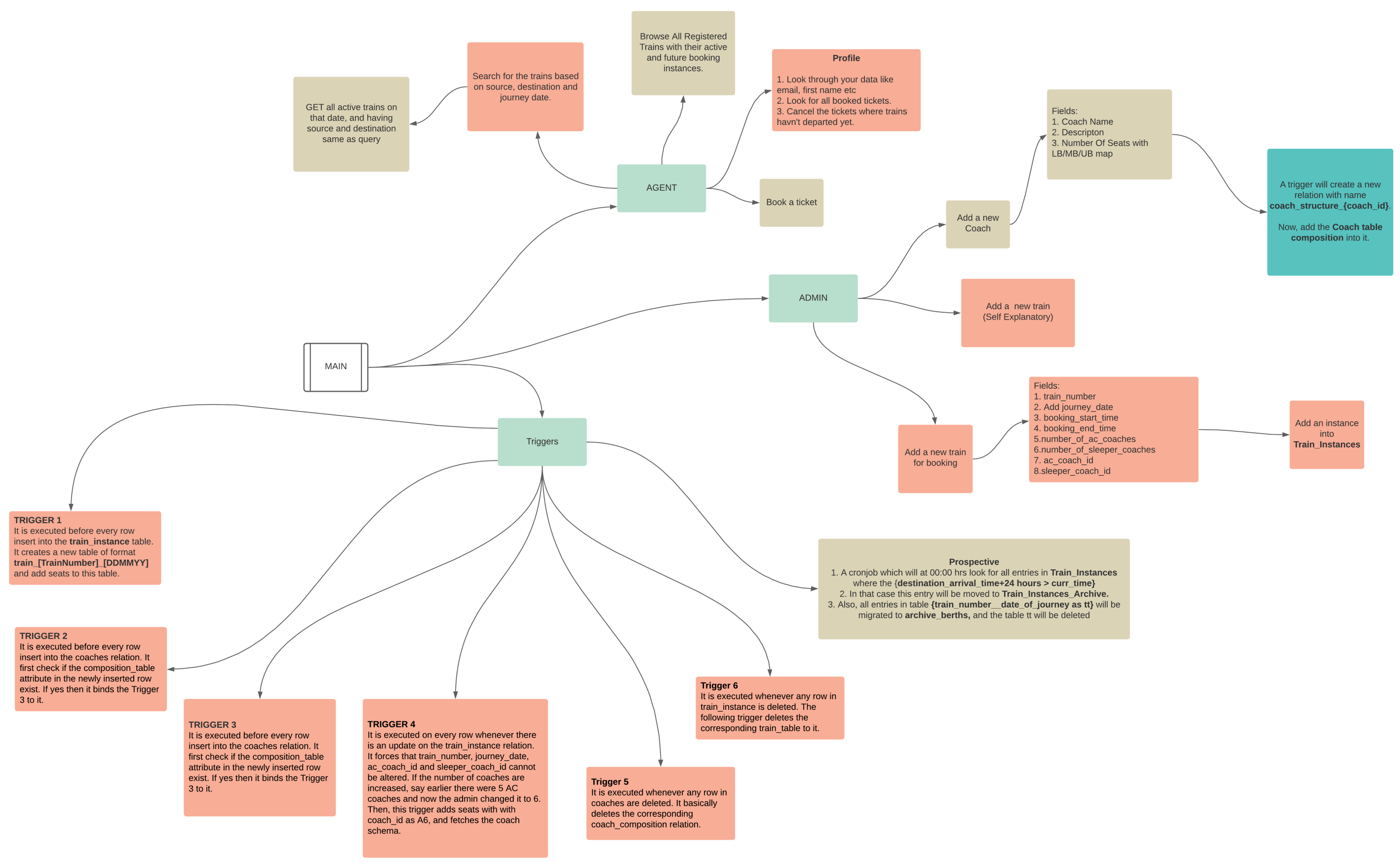
Train Ticketing

About	▯ A powerful Railway Ticket Booking Portal built using React, Node, PostgreSQL. Using dynamic SQL techniques, stored procedures, triggers for consistency, and faster query execution.
Link	https://github.com/subhamX/train-ticketing

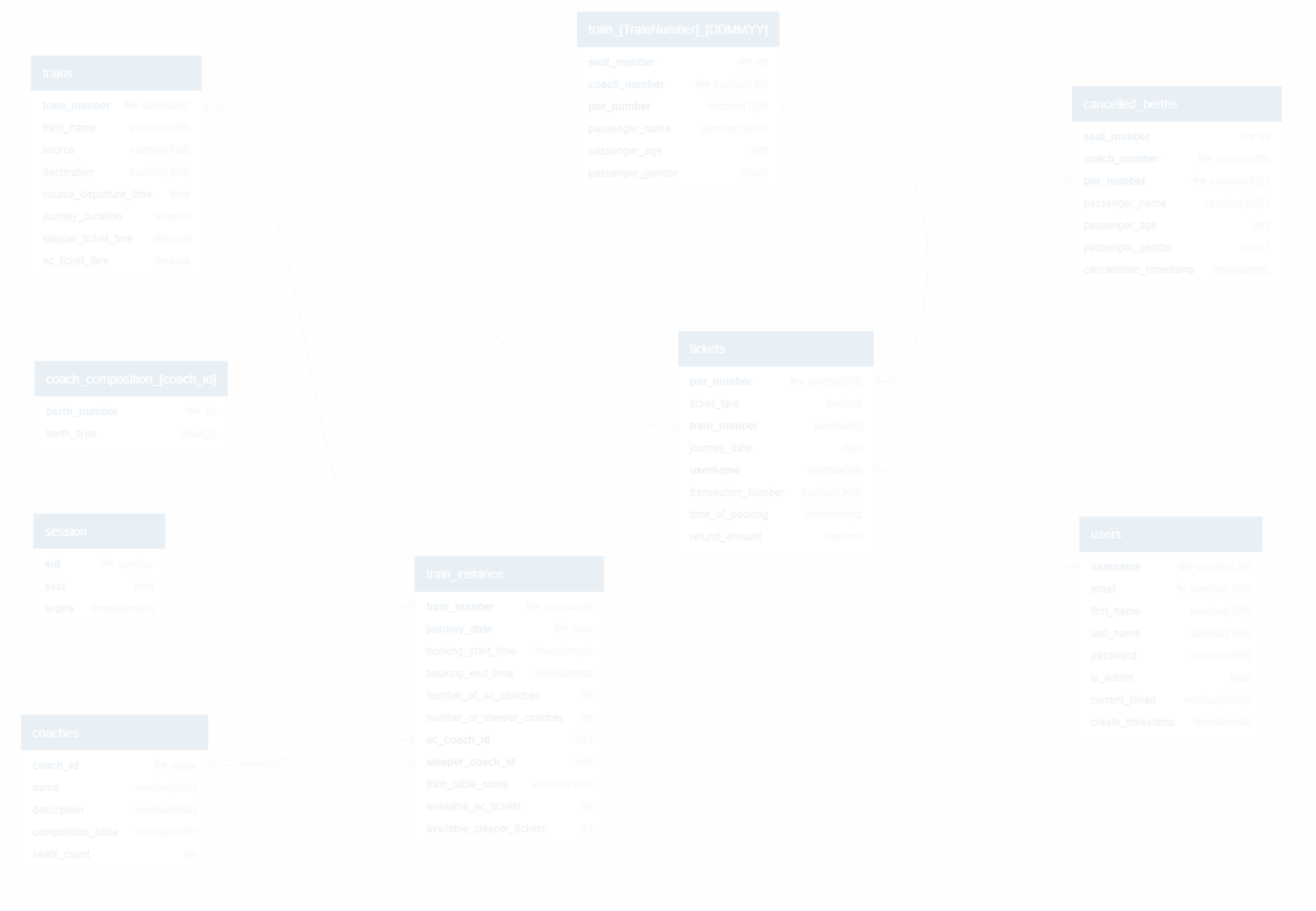
The following website is a Railway Ticket Booking Portal. This project embraces the Client-Server architecture, where the server and database are hosted separately and the client communicates via REST APIs.

The latest version of the **Railway Reservation System** is deployed at **traintkt.herokuapp.com**.

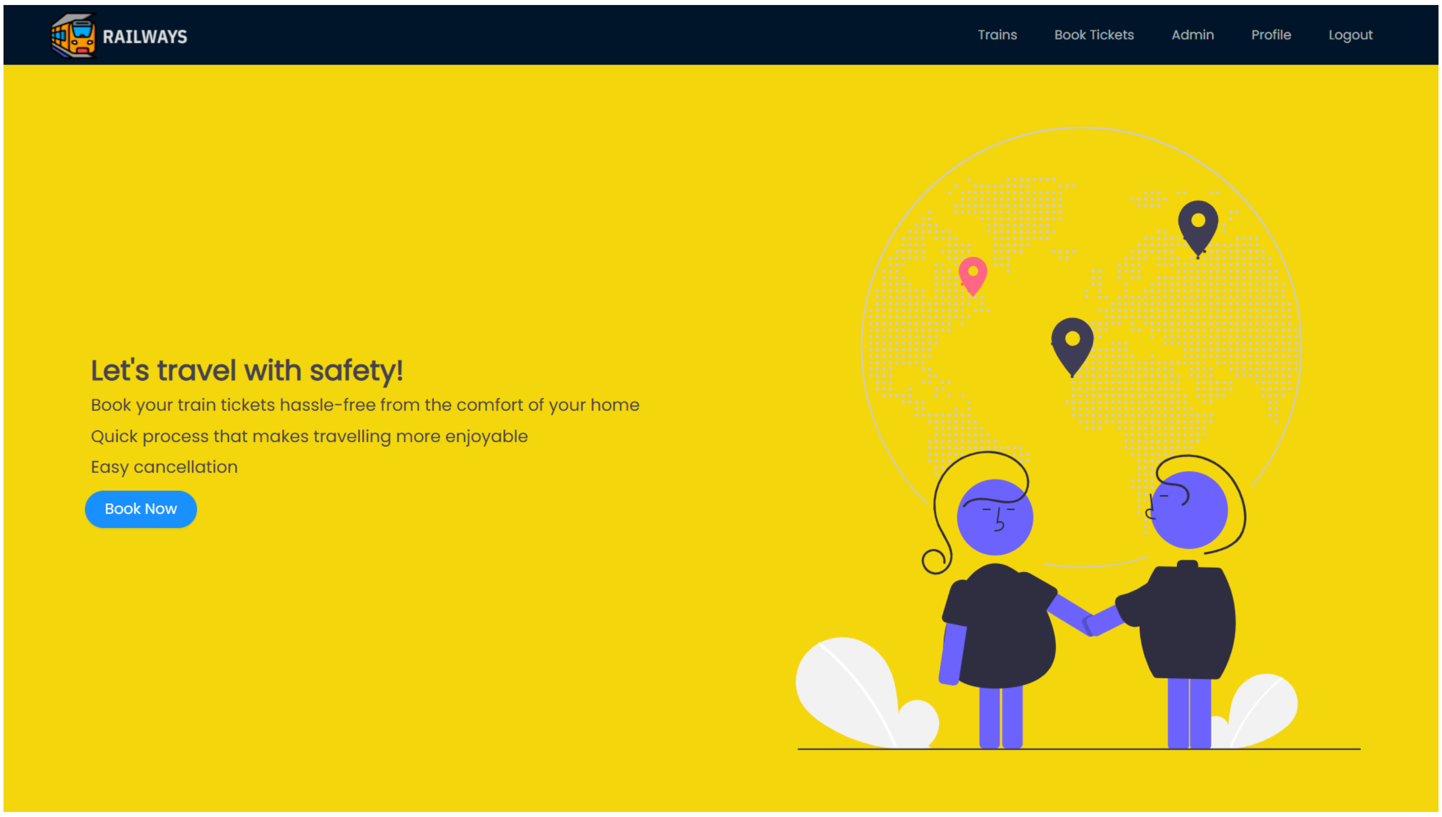
Project Design Overview


Browse All Registered Trains with their active and future booking instances.
Search for the trains based on source, destination and journey date.
GET all active trains on that date, and having source and destination same as query
Profile
1. Look through your data like email, first name etc 2. Look for all booked tickets. 3. Cancel the tickets where trains havn't departed yet.
AGENT
Book a ticket
Fields: 1. Coach Name 2. Descripton 3. Number Of Seats with LB/MB/UB map
A trigger will create a new relation with name **coach_structure_{coach_id}**. Now, add the **Coach table composition** into it.
Add a new Coach
ADMIN
Add a new train (Self Explanatory)
MAIN
Triggers
Add a new train for booking
Fields: 1. train_number 2. Add journey_date 3. booking_start_time 4. booking_end_time 5.number_of_ac_coaches 6.number_of_sleeper_coaches 7. ac_coach_id 8.sleeper_coach_id
Add an instance into **Train_Instances**
TRIGGER 1 It is executed before every row insert into the **train_instance** table. It creates a new table of format **train_[TrainNumber]_[DDMMYY]** and add seats to this table.
Prospective 1. A cronjob which will at 00:00 hrs look for all entries in **Train_Instances** where the **{destination_arrival_time+24 hours > curr_time}** 2. In that case this entry will be moved to **Train_Instances_Archive.** 3. Also, all entries in table **{train_number__date_of_journey as tt}** will be migrated to **archive_berths,** and the table tt will be deleted
TRIGGER 2 It is executed before every row insert into the coaches relation. It first check if the composition_table attribute in the newly inserted row exist. If yes then it binds the Trigger 3 to it.
Trigger 6 It is executed whenever any row in train_instance is deleted. The following trigger deletes the corresponding train_table to it.
TRIGGER 3 It is executed before every row insert into the coaches relation. It first check if the composition_table attribute in the newly inserted row exist. If yes then it binds the Trigger 3 to it.
TRIGGER 4 It is executed on every row whenever there is an update on the train_instance relation. It forces that train_number, journey_date, ac_coach_id and sleeper_coach_id cannot be altered. If the number of coaches are increased, say earlier there were 5 AC coaches and now the admin changed it to 6. Then, this trigger adds seats with with coach_id as A6, and fetches the coach schema.
Trigger 5 It is executed whenever any row in coaches are deleted. It basically deletes the corresponding coach_composition relation.

ER Diagram



Note: All blue and bolded attributes are primary key. All attributes with ? is indicating that the field can be NULL. Relations having [X] are dynamically created where X is some variable.



Let's travel with safety!

Book your train tickets hassle-free from the comfort of your home

Quick process that makes travelling more enjoyable

Easy cancellation

Key Features

1. The project uses dynamic SQL techniques, procedures, triggers, for consistency, and faster query execution.
2. Client can browse though the trains catalogue, their details and all active booking instances of it.
3. The project implements an easy to use trains search view with autosuggestions for train stations. Client can enter source, destination and date of journey and find if there are any trains available for following parameters.
4. Both booking tickets and cancel tickets operation can be performed on multiple passengers at once.
5. Admin can add trains, coaches, and booking instances from the

user interface and need not write any SQL query.

1. Our portal supports multiple coaches, and the Admin can define which coach schema to choose at the time of adding a booking instance.

Technology Stack

1. TypeScript
2. ReactJS (Frontend)
3. NodeJS (Backend)
4. PostgreSQL (Database)

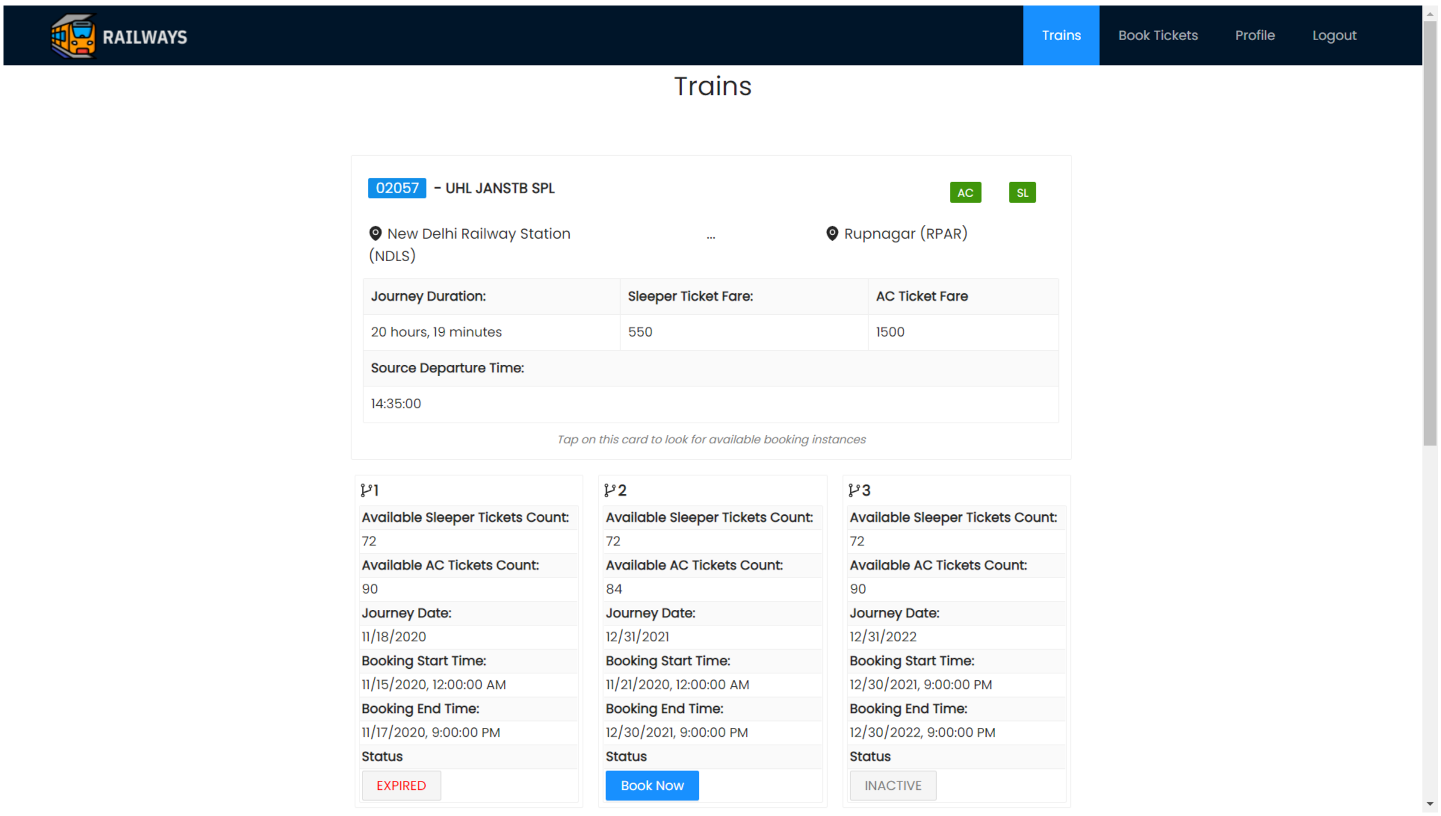
Functionalities Of Our Project

Use cases have been divided based on the end user i.e. Admin and Booking Agents.

Functionalites for Admin

1. Separate Authentication for Admin, and a dedicated admin dashboard.
2. Admin can add new `Trains`, `Coaches` with different seating schemas into the database for the dashboard.
3. Add and update `Reservation status` for trains.
4. Admin can perform the above operations without writing any database query by using our **simple and convenient User Interface**.

Project Walkthrough



Trains

02057 - UHL JANSTB SPL AC SL

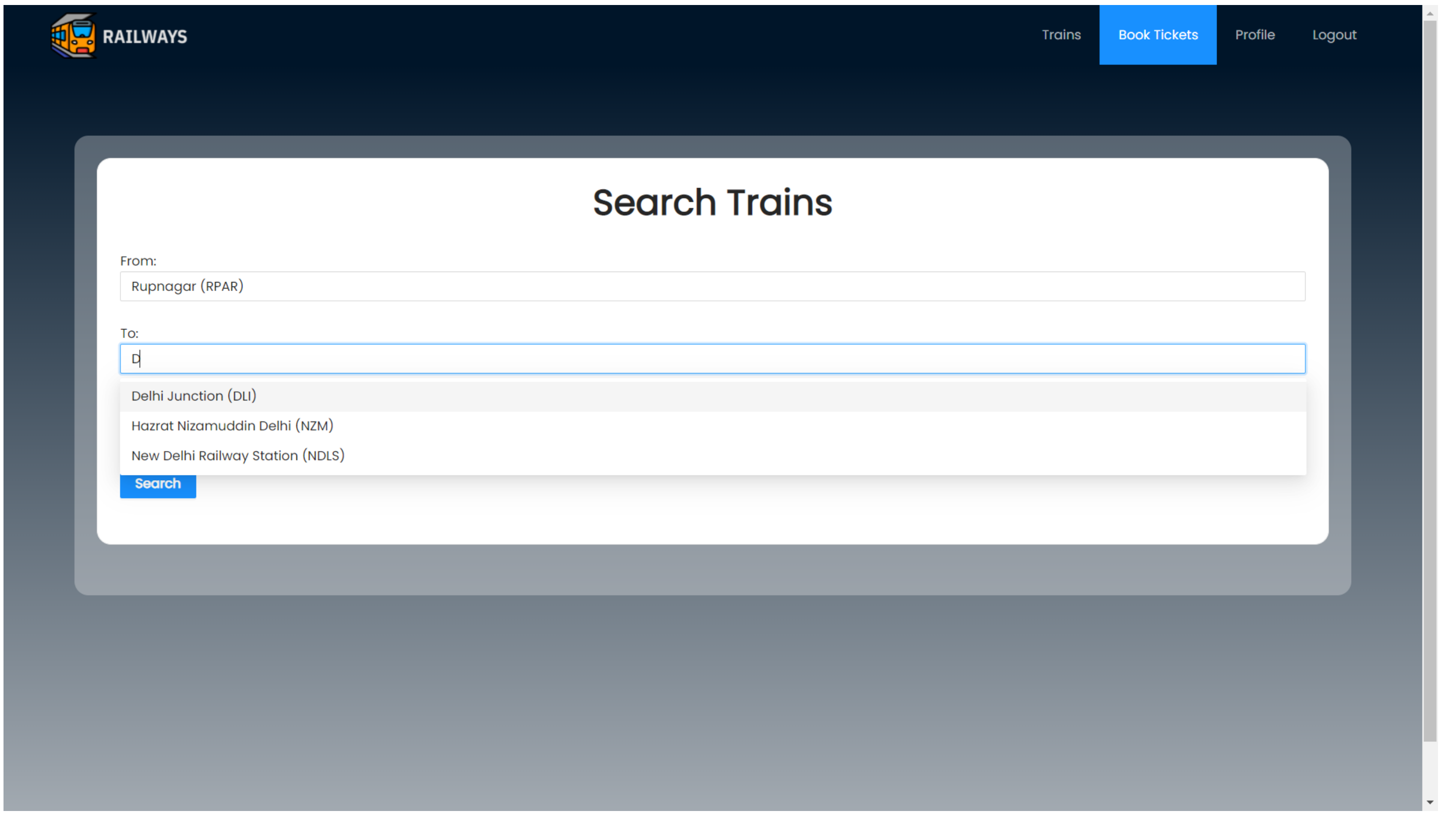
New Delhi Railway Station (NDLS) ... Rupnagar (RPAR)

Journey Duration:	Sleeper Ticket Fare:	AC Ticket Fare
20 hours, 19 minutes	550	1500
Source Departure Time:		
14:35:00		

Tap on this card to look for available booking instances

1	2	3
Available Sleeper Tickets Count:	Available Sleeper Tickets Count:	Available Sleeper Tickets Count:
72	72	72
Available AC Tickets Count:	Available AC Tickets Count:	Available AC Tickets Count:
90	84	90
Journey Date:	Journey Date:	Journey Date:
11/18/2020	12/31/2021	12/31/2022
Booking Start Time:	Booking Start Time:	Booking Start Time:
11/15/2020, 12:00:00 AM	11/21/2020, 12:00:00 AM	12/30/2021, 9:00:00 PM
Booking End Time:	Booking End Time:	Booking End Time:
11/17/2020, 9:00:00 PM	12/30/2021, 9:00:00 PM	12/30/2022, 9:00:00 PM
Status	Status	Status
EXPIRED	Book Now	INACTIVE

Trains Search Route can help users quickly find the available trains based on starting location, ending location, and the journey date. To ease searching for a specific station, **we autocomplete the source and destination fields based on the keywords** entered by users with the possible stations.

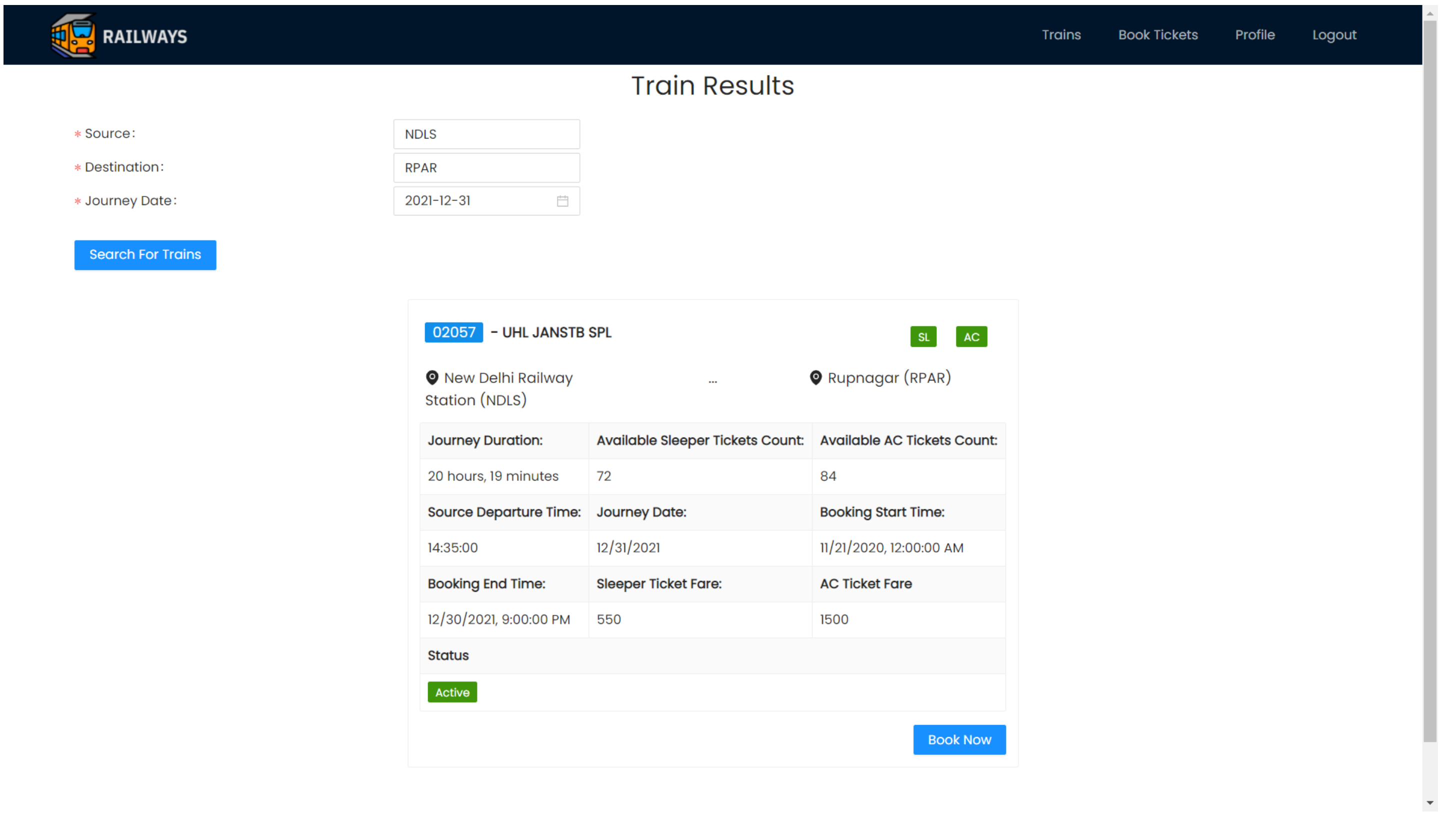


Search Trains

From:
Rupnagar (RPAR)

To:
D

Delhi Junction (DLI)
Hazrat Nizamuddin Delhi (NZM)
New Delhi Railway Station (NDLS)



Train Results

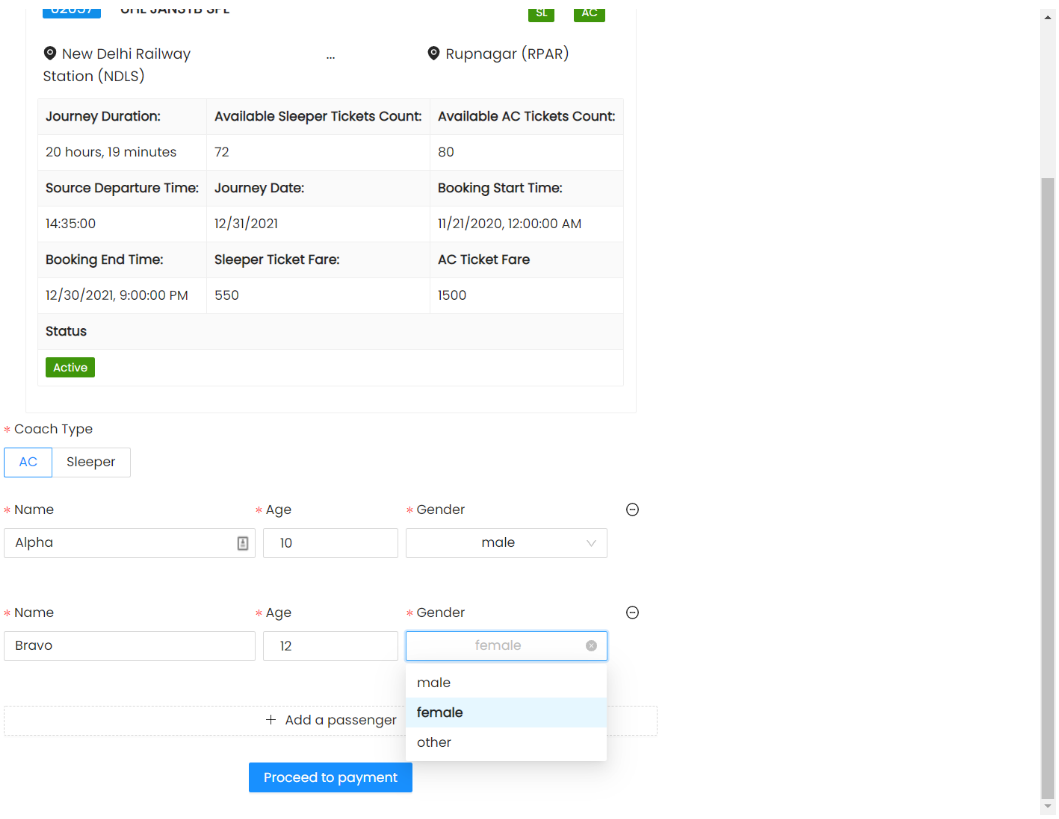
* Source: NDLS
* Destination: RPAR
* Journey Date: 2021-12-31

02057 - UHL JANSTB SPL

New Delhi Railway Station (NDLS) ... Rupnagar (RPAR)

Journey Duration:	Available Sleeper Tickets Count:	Available AC Tickets Count:
20 hours, 19 minutes	72	84
Source Departure Time:	**Journey Date:**	**Booking Start Time:**
14:35:00	12/31/2021	11/21/2020, 12:00:00 AM
Booking End Time:	**Sleeper Ticket Fare:**	**AC Ticket Fare**
12/30/2021, 9:00:00 PM	550	1500
Status		
Active		

Tickets Booking View

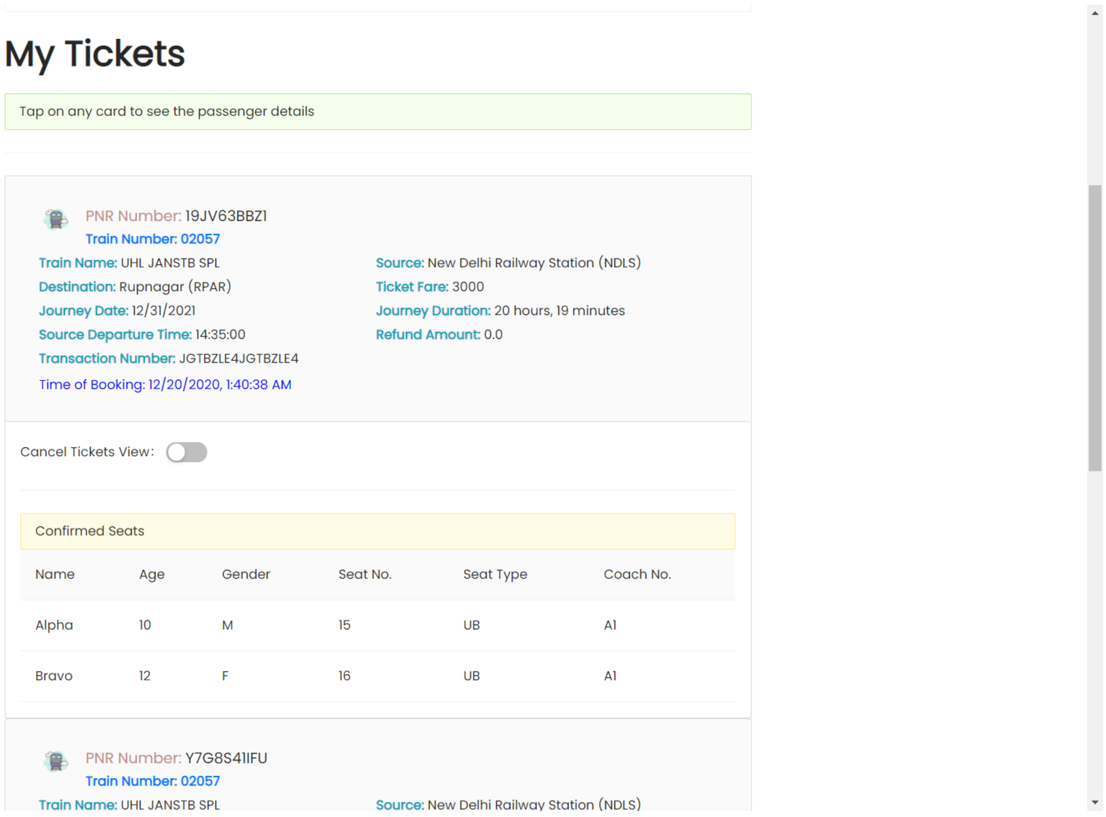


New Delhi Railway Station (NDLS) ... Rupnagar (RPAR)

Journey Duration:	Available Sleeper Tickets Count:	Available AC Tickets Count:
20 hours, 19 minutes	72	80
Source Departure Time:	Journey Date:	Booking Start Time:
14:35:00	12/31/2021	11/21/2020, 12:00:00 AM
Booking End Time:	Sleeper Ticket Fare:	AC Ticket Fare
12/30/2021, 9:00:00 PM	550	1500
Status		
Active		

Name	Age	Gender
Alpha	10	male
Bravo	12	female

User Profile Dashboard showing all past tickets



My Tickets

Tap on any card to see the passenger details

PNR Number: 19JV63BBZ1
Train Number: 02057
Train Name: UHL JANSTB SPL
Source: New Delhi Railway Station (NDLS)
Destination: Rupnagar (RPAR)
Ticket Fare: 3000
Journey Date: 12/31/2021
Journey Duration: 20 hours, 19 minutes
Source Departure Time: 14:35:00
Refund Amount: 0.0
Transaction Number: JGTBZLE4JGTBZLE4
Time of Booking: 12/20/2020, 1:40:38 AM

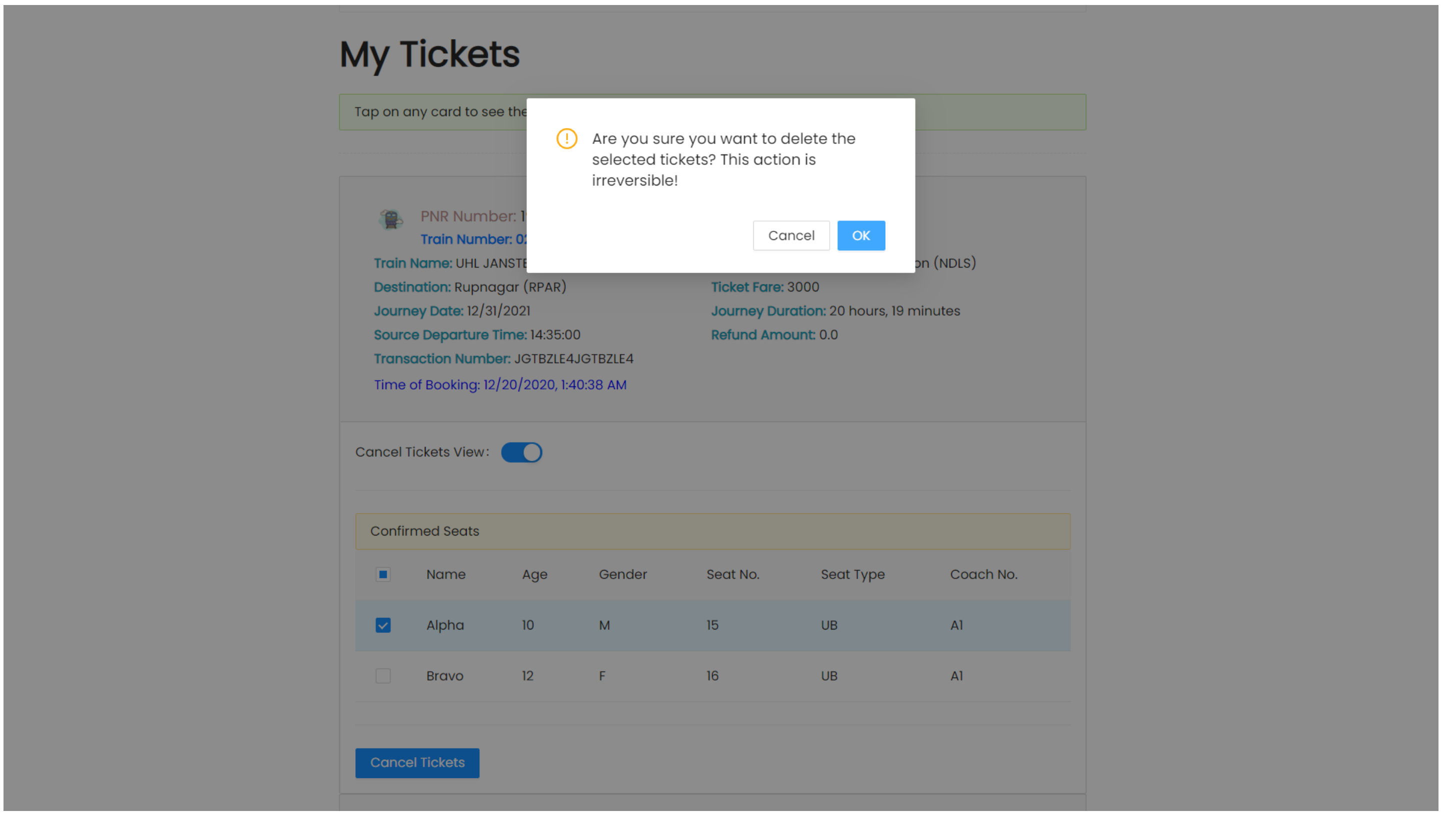
Cancel Tickets View:

Confirmed Seats

Name	Age	Gender	Seat No.	Seat Type	Coach No.
Alpha	10	M	15	UB	A1
Bravo	12	F	16	UB	A1

PNR Number: Y7G8S41IFU
Train Number: 02057
Train Name: UHL JANSTB SPL
Source: New Delhi Railway Station (NDLS)

Users can cancel the confirmed tickets before the train departure. This operation releases the seat for other passengers for booking while the train's booking status is "Active". The refund of the cancellation of tickets will be resolved every day during maintenance hours, this regular update reduces the storage overhead, by releasing the details of refunds.


My Tickets

Tap on any card to see the

Are you sure you want to delete the selected tickets? This action is irreversible!

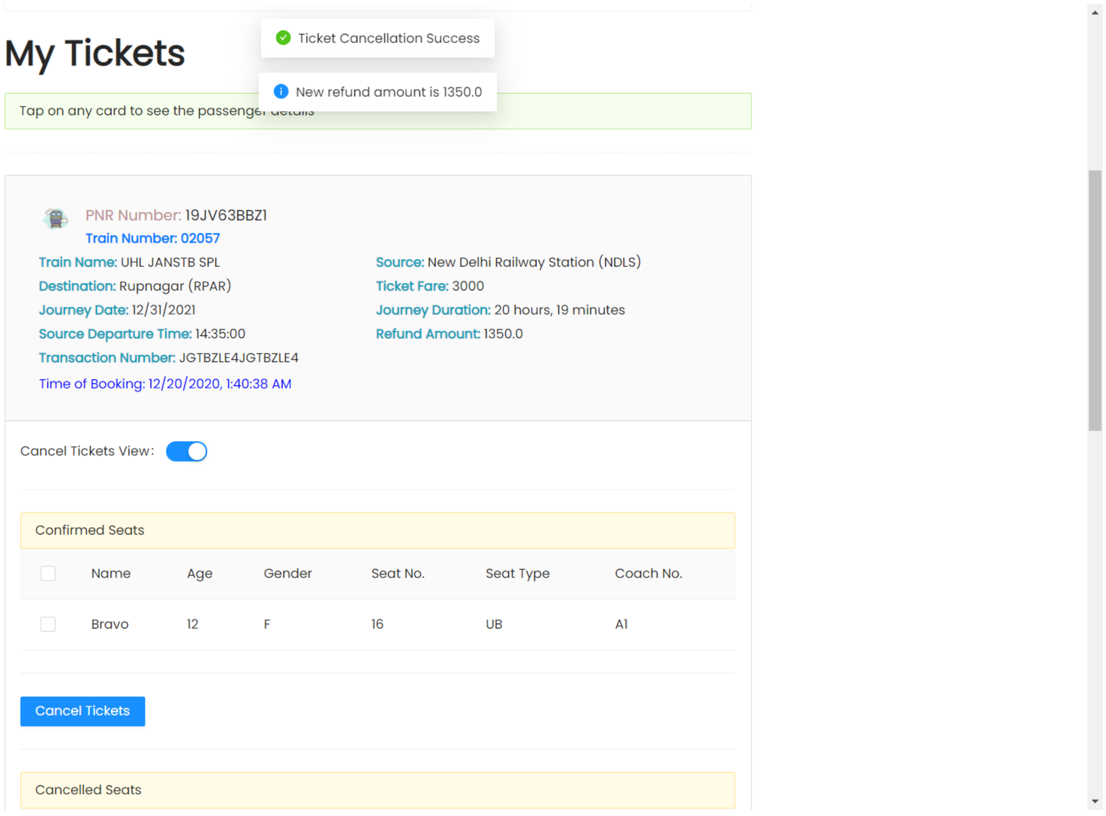
PNR Number: 1
Train Number: 0
Train Name: UHL JANSTE
Destination: Rupnagar (RPAR)
Journey Date: 12/31/2021
Source Departure Time: 14:35:00
Transaction Number: JGTBZLE4JGTBZLE4
Time of Booking: 12/20/2020, 1:40:38 AM

on (NDLS)
Ticket Fare: 3000
Journey Duration: 20 hours, 19 minutes
Refund Amount: 0.0

Cancel Tickets View:

Confirmed Seats

	Name	Age	Gender	Seat No.	Seat Type	Coach No.
✓	Alpha	10	M	15	UB	A1
	Bravo	12	F	16	UB	A1


My Tickets

Ticket Cancellation Success

New refund amount is 1350.0

Tap on any card to see the passenger details

PNR Number: 19JV63BBZ1
Train Number: 02057
Train Name: UHL JANSTB SPL
Destination: Rupnagar (RPAR)
Journey Date: 12/31/2021
Source Departure Time: 14:35:00
Transaction Number: JGTBZLE4JGTBZLE4
Time of Booking: 12/20/2020, 1:40:38 AM

Source: New Delhi Railway Station (NDLS)
Ticket Fare: 3000
Journey Duration: 20 hours, 19 minutes
Refund Amount: 1350.0

Cancel Tickets View:

Confirmed Seats

	Name	Age	Gender	Seat No.	Seat Type	Coach No.
	Bravo	12	F	16	UB	A1

Cancelled Seats

API Endpoints Overview

`POST user/profile` → Gives back information of the user; Only the authenticated user can see his data

`GET trains/list` → shows a list of all available trains; Public View

`GET trains/info/:id` → shows train information, and all bookings available; Public View

`GET /trains/current/active` → Shows all trains which are active booking phase; Public View

`GET /cities/all` → Returns a list of all cities which are either source or destination; Public View.

`POST tickets/book` → Allows user to book a ticket; Must be authenticated

`POST tickets/cancel` → Route to cancel berths from tickets; Must be authenticated; Can cancel only his/her ticket

`GET tickets/info/:pnr` → Gives back passenger list, and other info; Must be authenticated

`GET tickets/all/` → Gives back tickets for a user; Must be authenticated

`GET /coaches/list` → See all coaches; Public View;

`GET /coaches/:id` → Coach Information; Public View

`POST admin/addbookinginstance` → Add a train for booking; Must be an admin

`POST admin/trains/add` → Add a new train; Must be an admin

`POST admin/coaches/add` → Add a coach; Must be an admin;

`GET chart/:train_number/:date` → See the reservation chart; Public View