WEB APPLICATION FOR DETECTION OF ROUTES BETWEEN TWO STATIONS WITH NO DIRECT TRAIN CONNECTIVITY

A Thesis Submitted in

Partial Fulfilment of the Requirements for the Degree of

Bachelor of Technology

Submitted by:

Vivek Kumar (18-1-3-030)

Sushant Kumar (18-1-3-063)

Under the Supervision of:

Dr. Risha Mal

Assistant Professor,

Department of Electrical Engineering

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR, 2022

WEB APPLICATION FOR DETECTION OF ROUTES BETWEEN TWO STATIONS WITH NO DIRECT TRAIN CONNECTIVITY

A Thesis Submitted in

Partial Fulfilment of the Requirements for the Degree of

Bachelor of Technology

Submitted by:

Vivek Kumar (18-1-3-030)

Sushant Kumar (18-1-3-063)

Under the Supervision of:

Dr. Risha Mal

Assistant Professor,

Department of Electrical Engineering

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR, 2022

©NATIONAL INSTITUTE OF TECHNOLOGY
SILCHAR, 2022

ALL RIGHTS RESERVED

DECLARATION

We Vivek Kumar and Sushant Kumar, declare that this thesis titled, “WEB APPLICATION FOR DETECTION OF ROUTES BETWEEN TWO STATIONS WITH NO DIRECT TRAIN CONNECTIVITY” and the work presented in it are our own.

We confirm that:

- This work was done wholly or mainly while in candidature for a B.Tech degree at this University.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.
- Where the thesis is based on work done by our self jointly with others. We have made clear exactly what was done by others and what we have contributed our self.

Signed:

....

Date:

....

Department of Electrical Engineering

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

SILCHAR-788010, ASSAM

Certificate

It is certified that the work contained in this thesis entitled “WEB APPLICATION FOR DETECTION OF ROUTES BETWEEN TWO STATIONS WITH NO DIRECT TRAIN CONNECTIVITY” submitted by Vivek Kumar and Sushant Kumar bearing scholar numbers 18-1-3-030 and 18-1-3-063 respectively for the award of Bachelor of Technology in Electrical Engineering is absolutely based on their own work carried out under the supervision of Dr. Risha Mal and this work has not been submitted anywhere else for any degree.

Dr. Risha Mal

Project Supervisor

Department of Electrical Engineering

National Institute of Technology Silchar

Department of Electrical Engineering

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

SILCHAR-788010, ASSAM

Certificate

It is certified that the work contained in this thesis entitled "WEB APPLICATION FOR DETECTION OF ROUTES BETWEEN TWO STATIONS WITH NO DIRECT TRAIN CONNECTIVITY" submitted by Vivek Kumar and Sushant Kumar bearing scholar numbers 18-1-3-030 and 18-1-3-063 respectively for the award of Bachelor of Technology in Electrical Engineering is absolutely based on their own work carried out under the supervision of Dr. Risha Mal and this work has not been submitted anywhere else for any degree.

Dr. Jyoti Prakash Mishra

Head of Department

Department of Electrical Engineering

National Institute of Technology Silchar

“To Succeed in your mission, you must have a single-minded devotion to your goal.”

Dr. APJ Abdul Kalam

Abstract

To provide proper route and to avoid complicacy in journey through trains, we have come up with some ideas. We have tried to implement these ideas. For this first of all we have tested whole scenario on the small database, which can be exceeded further according to our needs. We are not only interested in suggesting name of trains but along with it we are also reflecting total distance travelled in each and every lap and how may laps you have to cover to complete your journey. This Project is also hosted on firebase, a type of cloud server. Now with the generated URL anyone can be able to access the webpage. We have also tried to implement whole functionalities in Application.

Acknowledgements

Any accomplishment requires the effort of many people and this work is no different. First of all, we would like to acknowledge our families who supported us with love and understanding that led us to this level of success.

Secondly, we would like to express our sincere gratitude to our project guide and mentor, Dr. Risha Mal, Assistant Professor, Department of Electrical Engineering, National Institute of Technology Silchar, for her kind and continuous support for this work of study. This thesis would not have been possible without her encouragement throughout.

Finally, we would like to state our thanks to our friends and batch mates who have always been there for us in times of need. Every effort has been made to give credit where it is due for the material contained and we express our gratitude to everyone who may contributed to this work, even though anonymously.

TABLE OF CONTENTS

Declaration…………………………………………………………………………………...3

Certificate……………………………………………………………………………….4-5

Abstract……………………………………………………………………………………7

Acknowledgement………………………………………………………………………...8

Table of Contents………………………………………………………………………9-10

List of Figures……………………………………………………………………………11

1. Introduction ……………………………………………………………………….12-14

    1.1. Motivation of the work…………………………………………………………...12

    1.2. The Dire Need…………………………………………………………………….13

    1.3. Rescue……………………………………..……………………………………..13

    1.4. Tools Used……………………………………………………………………….14

2. Literature…………...…………………………………………………………...15-17

3. Methodology……….………………………………………………………………..18-25

    3.1. Implemented Approach…………………………………………………………...18

4. Complexity Analysis………………………………………………………………26-27

    4.1. Time Complexity…………………………………………………………………26

    4.2. Space Complexity………………………………………………………………...27

5. The Challenges…………………………………………………………………….28-32

    5.1. Fixing Issue with Upper case and Lower case……………………………………28

    5.2. Suggesting a name……………………………………………………………….29

    5.3. Scaling the database………………………………………………………………30

5.4. Optimizing the Time Complexity…………………………………………………...31

5.5. Cost analysis…………………………………………………………………...31

5.6. Adding Distances in each lap…………………………………………………….32

5.7. Adding Train Number and Fare………………………………………………….33

6. Results…………………………………….……………………………………….34-35

6.1. Technology Required for an App………………………………………………...34

6.2. App User Interface……………………….…………………………..………..35

7. Conclusions & Future Scope….……………………………………………………..36-37

7.1. Conclusion……………………………………………………………………….36

7.2. Future Scope……………………………………………………………………..36

8. Prototype……………………………………………………………………………….. 38

9. References……………………………………………………………………………….39

LIST OF FIGURES

2.1. Representation of vertices and Edge	15
3.1. Front-end design of web-page	20
3.2. Pop-up tab of Search Trains	21
3.3. Find Route UI	22
3.4. Pop-up tab of Fetch Stations	22
3.5: Fetch Status of PNR status UI	23
3.6: Track train status UI	23
3.7: Results of Direct Trains	24
3.8: Results of In-Direct Trains	25
5.1: Suggesting name suggestions	30
6.1: User-Interface of App	35
6.2: Pop-tab of Search Trains in App	35

Chapter 1

INTRODUCTION

In this Project we are trying to implement an idea which will deduce route when there is no direct train found between source and destination. IRCTC database contains a large number of records of Train schedule with there arrival and departure time from every station with every possible detail. This data has to be configured to produce the shortest and multiple alternate paths through a given set of operations.

From the given data base, where every particular trains have a several stoppages at several stations, our task is to find the any simple path from a given source station to a given destination station. Simple Path is the path from one station to another such that no station is visited more than once. If there is no simple path possible then just return "Not possible" otherwise it will show every possible way to reach the destination.

1.1. MOTIVATION OF THE WORK

- Railways are the most opted form of transport now a days. If we look over the data, on an average the railway network transported over 22 million passengers every day.
- Many Small villages or towns which contains a local Railways station but if they have to went to any farther location first of all they have to reach to a Station which have higher connectivity.

- Above Problem is solved by traversing through all simple paths connecting between initial station to final station using a modified version of Search and find the path amongst them. A simple solution is to start from source, go to all adjacent vertices, and in recursive manner for further adjacent stations until we reach the destination.

1.2. THE DIRE NEED

- There are 22 million people boarding and arriving at almost 7,325 stations across all over India. Out of these 7,325 stations, not all stations are well connected. So, we can't ignore the fact that many people have to use some connecting trains between two stations.
- For an instance let us take an example of a person who wants to go from Patna (capital of Bihar) to Silchar. If we try to search any train on any day except on Friday, it shows no direct train available. The problem is that if the person has never travelled in this route, he will have to search for every station between Patna and Silchar.
- Even if we consider 1% of people travelling has to face this problem than almost 2.2 million people face this problem every day.
- So, considering above facts there is a dire need to solve this problem.

1.3. RESCUE

- Here comes the rescue, Trains trip, as we call it, gives the possible route for any two given stations even if there is no direct train between given source and destinations.
- If there are multiple possibilities between source and destinations then it gives the list.

1.4. TOOLS USED

- DATA STRUCTURES: It is the way of organizing data so that it can be used effectively. In our project we will have to store different stations between source and destinations, so we have to use different data structures. Since mostly we have to just store all the stations between source and destinations so, mostly we will be using vectors as a storing entity. Apart from vector we have other different data structures also like arrays, linked lists. We have some other abstract data structures also like stack, queue etc. We will be covering vector in later pages.
- ALGORITHMS: Algorithms are set of instructions for solving logical and mathematical problems. There are many algorithms depending on the problem like greedy algorithms, searching sorting, dynamic programming, Depth First Search, Breadth First Search etc. Since stations are like graph structures so in our project only Depth First Search and Breadth First Search are useful

There are several approaches to find out the route between two stations as given below:

1. One of the approaches is storing all the database of trains in graph format.
2. On this graph we will perform searching algorithm.

Chapter 2

LITERATURE

A graph can be defined as group of vertices and edges that are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship[2].

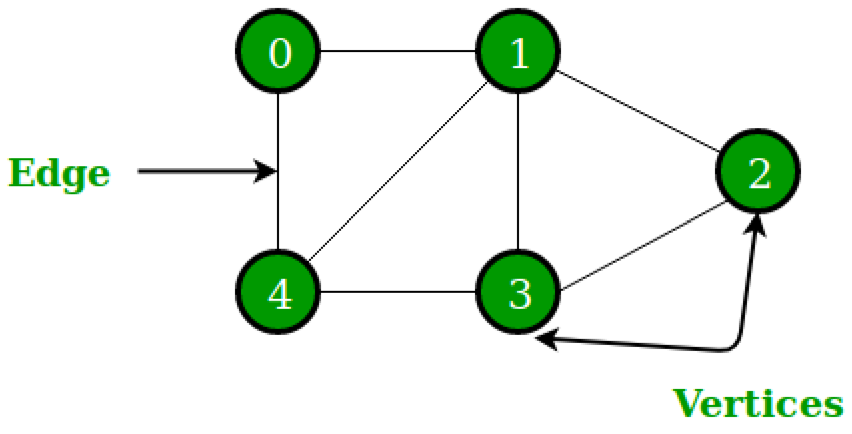


Figure 2.1: Representation of vertices and Edge

In the above Graph,

the set of vertices V = {0,1,2,3,4} and

the set of edges E = {01, 12, 23, 34, 04, 14, 13}.

We can either use Breadth First or Depth First Traversal to find if there is a path between two vertices.

Depth first Search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph data structure. Traversal means visiting all the nodes of a graph.[3]
A standard DFS implementation puts each vertex of the graph into one of two categories: Visited and Not Visited.
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.
The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.
4. Keep repeating steps 2 and 3 until the stack is empty.

DFS stands for Depth First Search is a edge based technique. It uses the Stack data structure, performs two stages, first visited vertices are pushed into stack and second if there is no vertices then visited vertices are popped.

Breadth First Search or Breadth First Traversal is a for searching all the vertices of a graph data structure. A standard BFS implementation puts each vertex of the graph into one of two categories-Visited and Not Visited.[3]
The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.
The algorithm works as follows:

1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.

4. Keep repeating steps 2 and 3 until the queue is empty.

The graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the BFS algorithm on every node BFS stands for Breadth First Search is a vertex-based technique for finding a shortest path in graph. It uses a Queue data structure which follows first in first out. In BFS, one vertex is selected at a time when it is visited and marked then its adjacent are visited and stored in the queue. It is slower than DFS.

Chapter 3

METHODOLOGY

We have discussed a lot of ways which can solve this problem but finally we came up with only these approaches. Because we think we can work very conveniently with these approaches.

3.1. IMPLEMETED APPROACH

In this approach, we have changed the way in which data (database of trains) has been stored. In the above two cases, we have suggested GRAPH in which all the details of train is stored. Just like a web. But here due to several limitations, we are taking vector of vectors. In place of vector of vectors, we can also take 2-D matrix. But to optimise space we have taken vector of vectors.

If we have ‘n’ number of trains, then we are forming ‘n’ numbers of vectors. Each vector representing a particular train. Further these ‘n’ each and all vectors contain another vector, which store all details about the stoppage of that particular train from which that vector belongs. Let’s consider 1st train has ‘m’ stoppage, then 1st vector contains vector of m size, storing details of every ‘m’ stoppage junction.

Let’s say X = Source,

Y = Destination.

For Direct Trains:

Finding direct trains is an easy task. What we will do is that we first try to traverse the whole trains and find if the source is present or not. If the source is present then we start searching just after that. If we find the destination too then there is direct train between given stations. So, we display the required trains.

For Indirect Trains:

Finding indirect train is not that much easy. We have considered brute force solution to achieve this. We first traverse the whole array and try to find the source in the array. If we are able to find the source in the array then we just call on function which gives the direct train between two stations. We do this for every station which is present later in the array. So, in this way we can find the indirect trains for any two source and destination.

But before implementing all the above ideas, first of all we have to develop a platform to display all these details.

Using HTML, CSS, JavaScript, and Bootstrap we have developed a responsive web page which handle all the functionalities.

The web page is, which is developed by us looks like:

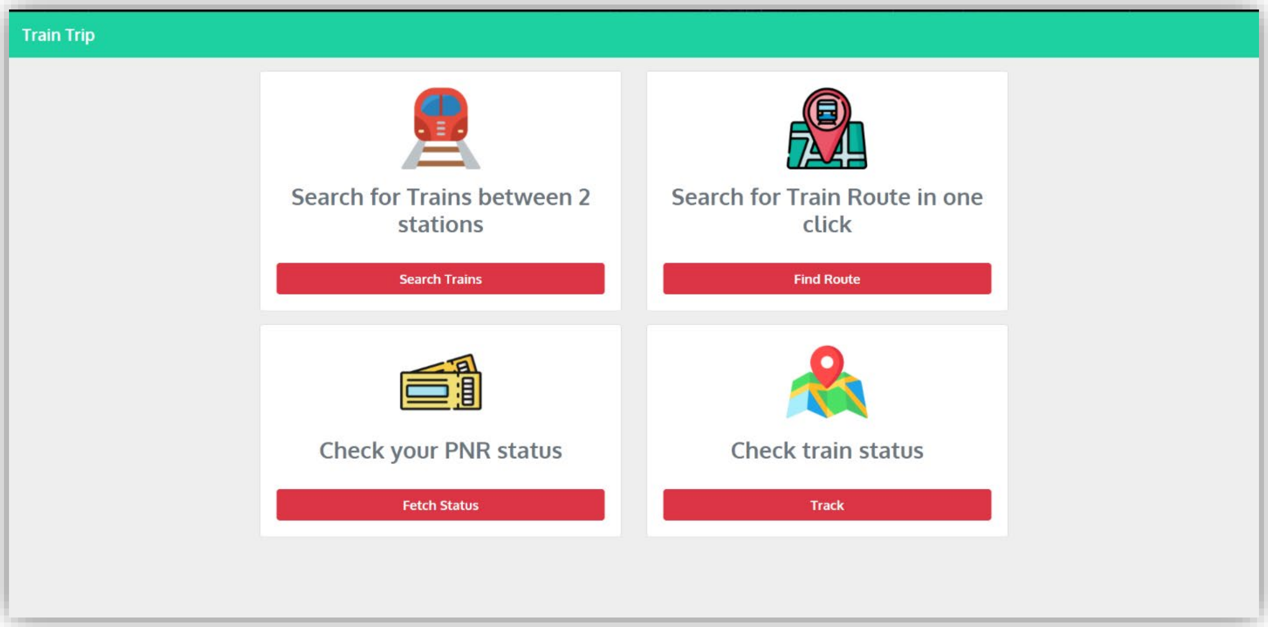


Figure 3.1: Front-end design of web-page

We have divided the web page in four divisions,

Four divisions are:

Search for Trains between 2 stations

Search for Train Route in one click

Check your PNR status

Check train status

Out of these four divisions, we have mainly focused on 1st divisions. In order to complete this 1st divisions, we have used an "ALERT" which further takes source and destination from users.

ALERT pop-up like this:

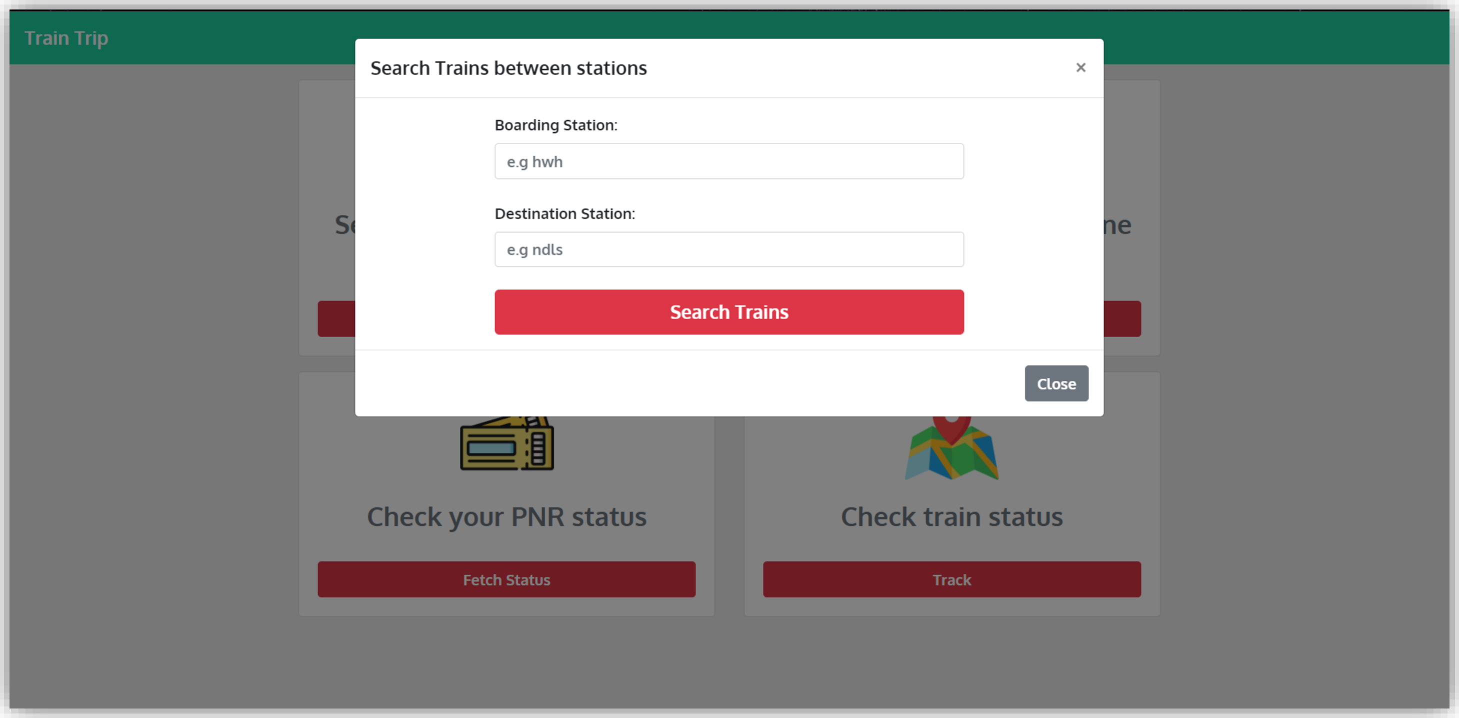


Figure 3.2: Pop-up tab of Search Trains

Similarly, "ALERT" has been designed for all the remaining three sections.

2nd section - Search for Train Route in one click

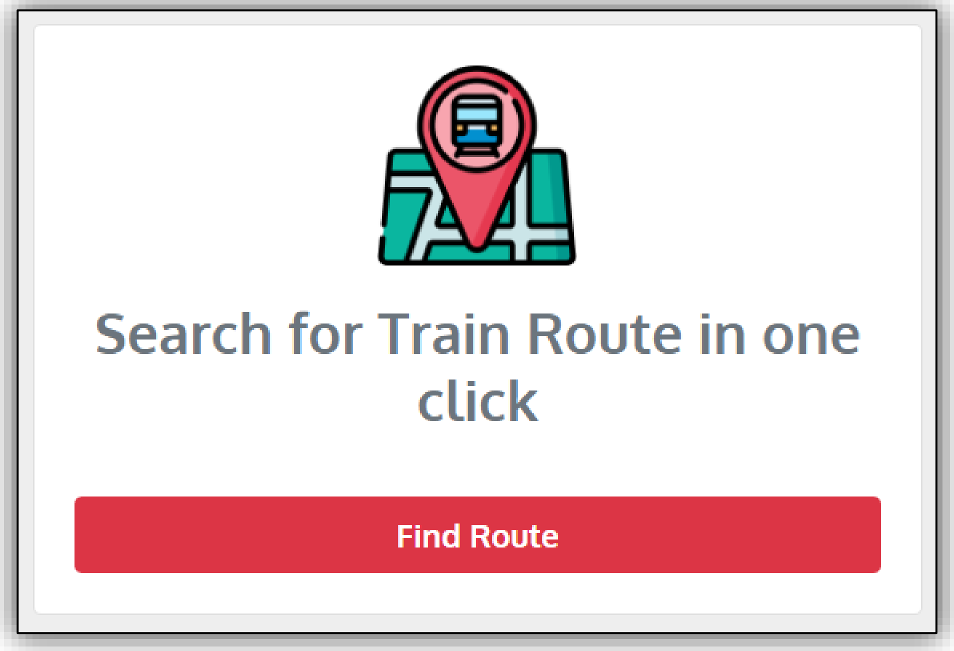


Figure 3.3: Find Route UI

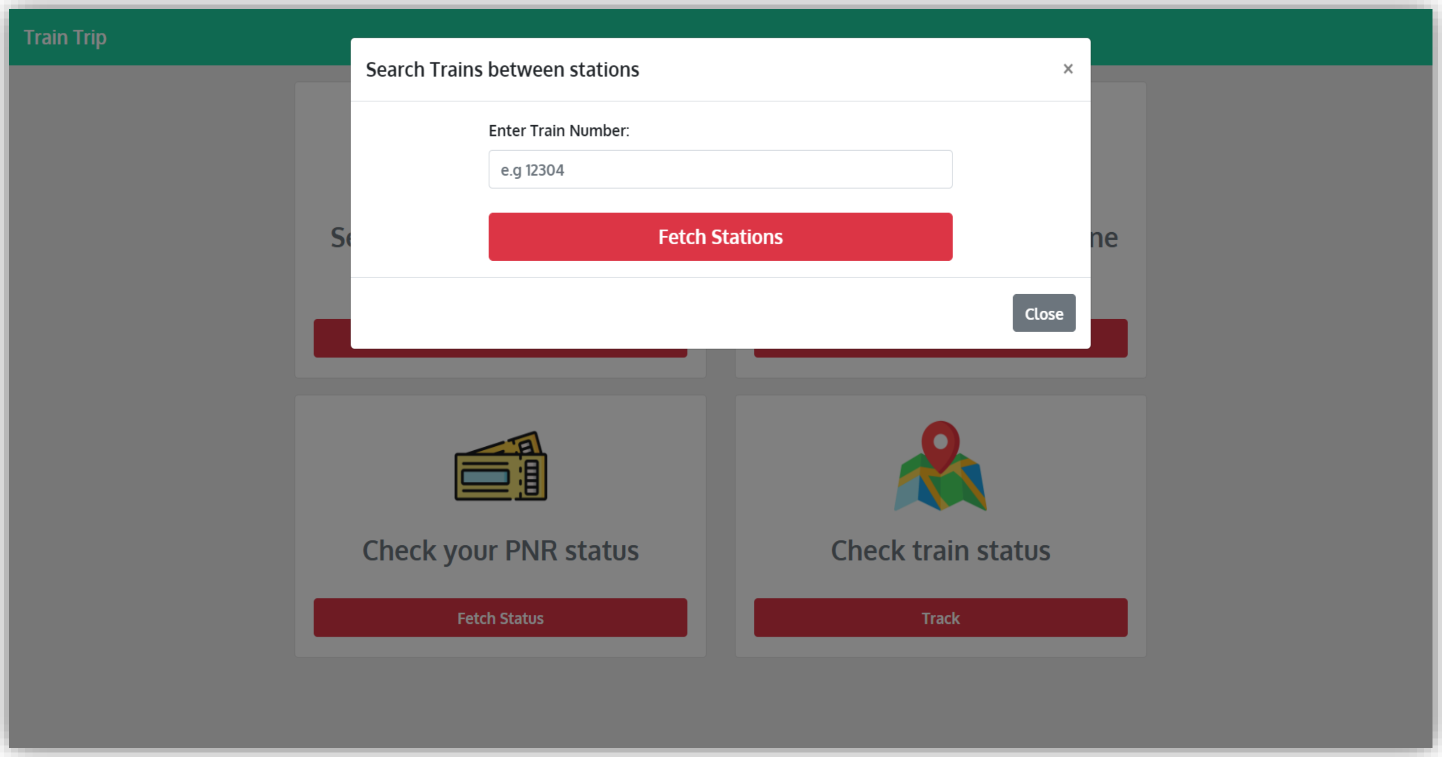


Figure 3.4: Pop-up tab of Fetch Stations

3rd Section - Check your PNR status

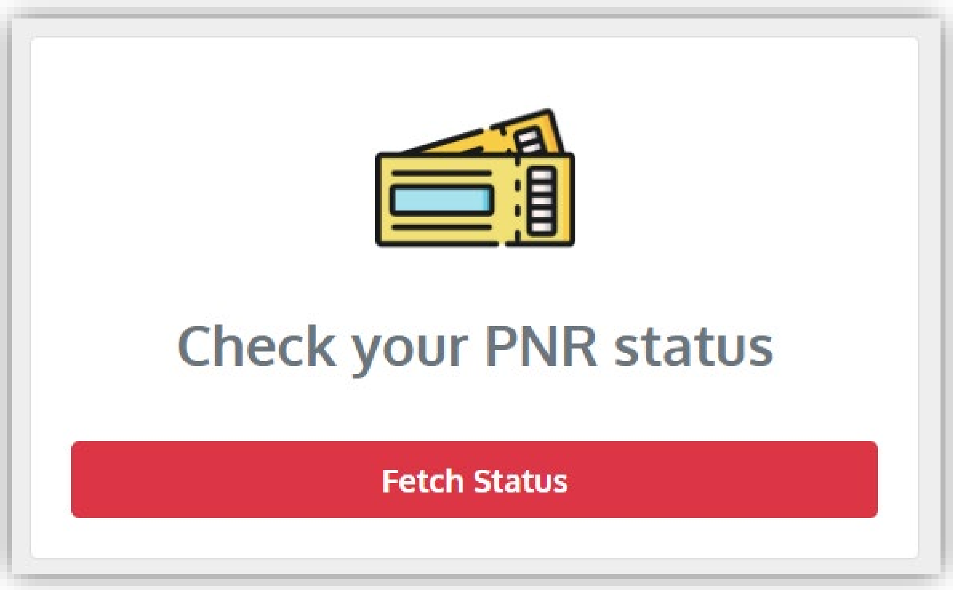


Figure 3.5: Fetch Status of PNR status UI

4th Section - Check train status

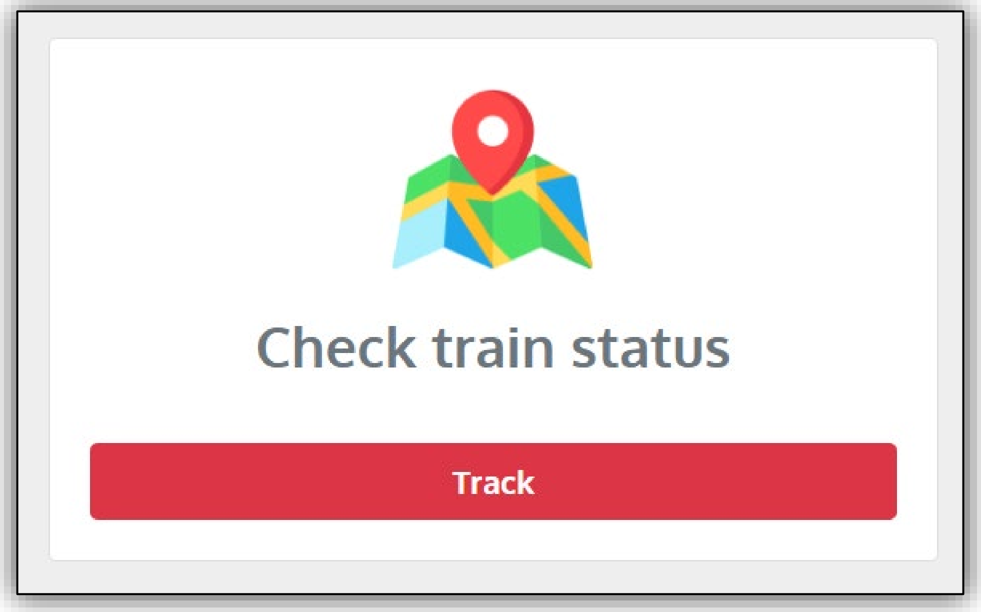
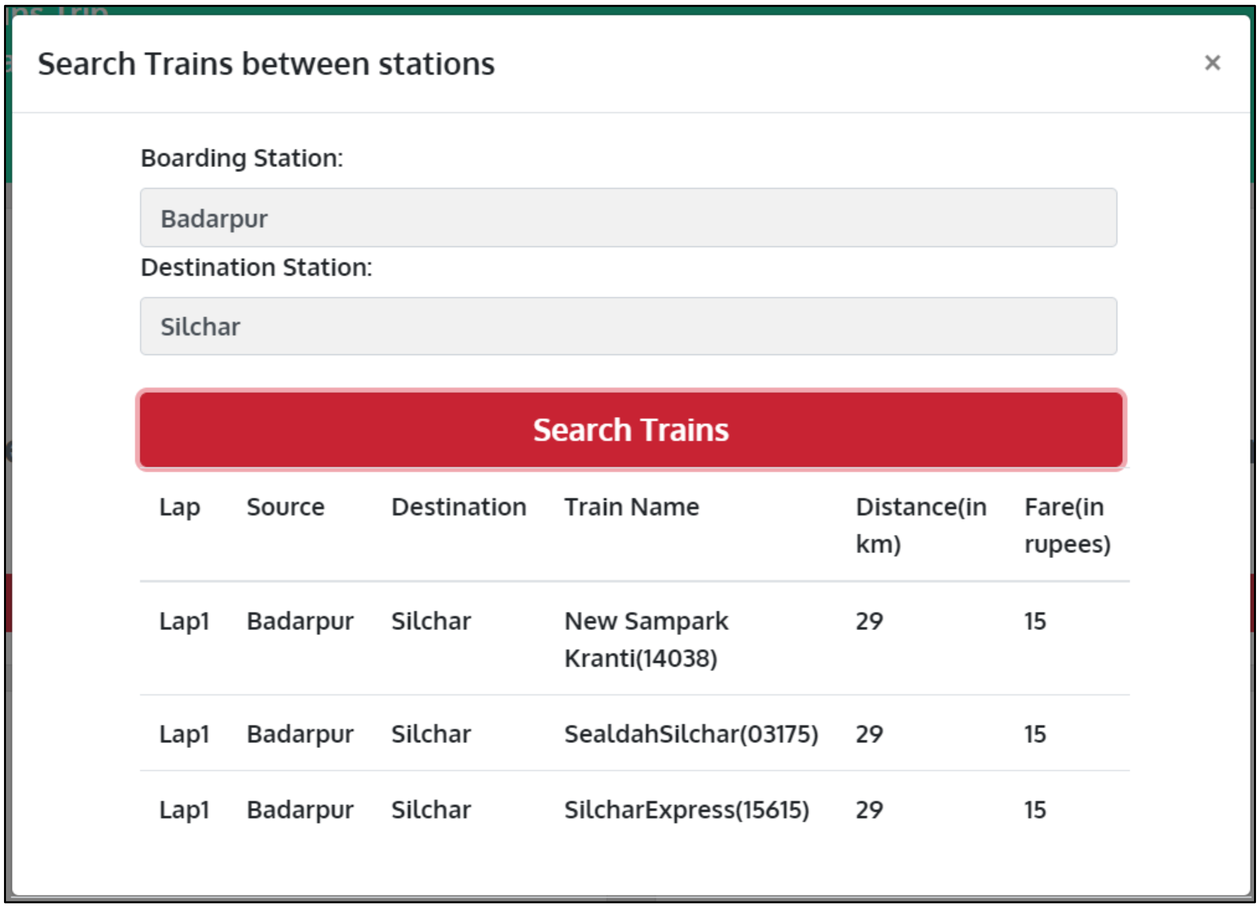


Figure 3.6: Track train status UI

Now Let's come back to the first Section

i.e., <u>**Search for Trains between 2 stations**</u>

Finding direct trains is an easy task. What we will do is that we first try to traverse the whole trains



Search Trains between stations

Boarding Station: Badarpur

Destination Station: Silchar

Lap	Source	Destination	Train Name	Distance(in km)	Fare(in rupees)
Lap1	Badarpur	Silchar	New Sampark Kranti(14038)	29	15
Lap1	Badarpur	Silchar	SealdahSilchar(03175)	29	15
Lap1	Badarpur	Silchar	SilcharExpress(15615)	29	15

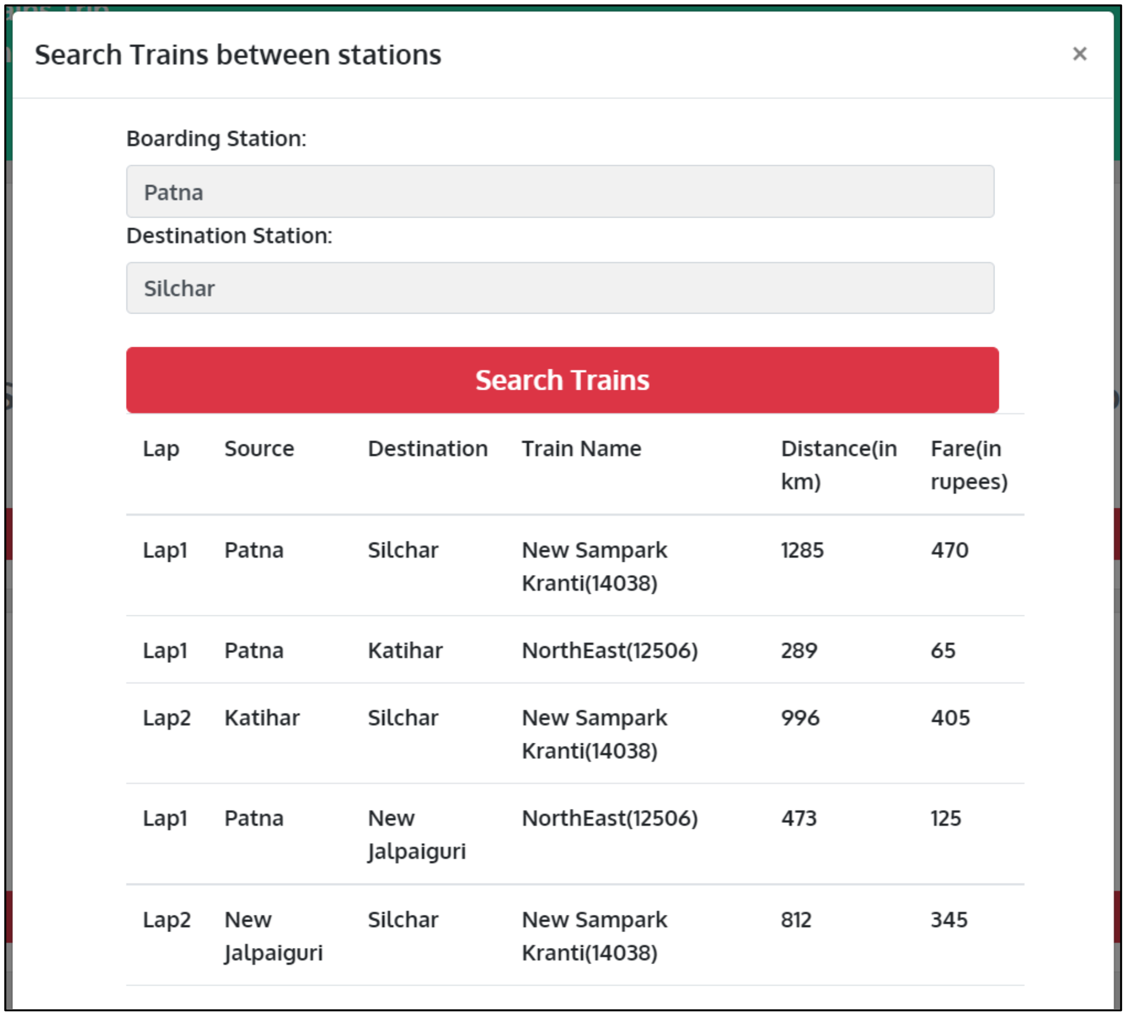
Figure 3.7: Results of Direct Trains

and find if the source is present or not. If the source is present then we start searching just after that. If we find the destination too then there is direct train between given stations. So, we display the required trains.

For Indirect Trains:

Finding indirect train is not that much easy. We have considered brute

force solution to achieve this. We first traverse the whole array and try to find the source in the array. If we are able to find the source in the array then we just call on function which gives the direct train between two stations. We do this for every station which is present later in the array. So, in this way we can find the indirect trains for any two source and destination.



Boarding Station: Patna

Destination Station: Silchar

Lap	Source	Destination	Train Name	Distance(in km)	Fare(in rupees)
Lap1	Patna	Silchar	New Sampark Kranti(14038)	1285	470
Lap1	Patna	Katihar	NorthEast(12506)	289	65
Lap2	Katihar	Silchar	New Sampark Kranti(14038)	996	405
Lap1	Patna	New Jalpaiguri	NorthEast(12506)	473	125
Lap2	New Jalpaiguri	Silchar	New Sampark Kranti(14038)	812	345

Figure 3.8: Results of In-Direct Trains

Chapter 4

COMPLEXITY ANALYSIS

Algorithmic complexity is a measure of how long an algorithm would take to complete given an input of size n. If an algorithm has to scale, it should compute the result within a finite and practical time bound even for large values of n. For this reason, complexity is calculated asymptotically as n approaches infinity

While complexity is usually in terms of time, sometimes complexity is also analyzed in terms of space, which translates to the algorithm's memory requirements. So generally, two types of complexity are kept in mind while writing any algorithm

These two complexities are time complexity and space complexity

4.1. TIME COMPLEXITY:

Time Complexity is the amount of time taken by an algorithm to run, as a function of the length of the input. It measures the time taken to execute each statement of the code in an algorithm.[4]

For any defined problem, there can be N number of solutions. This is true in general. If I have a problem and I discuss about the problem with all of my friends, they will suggest me different solutions. And I am the one who has to decide which solution is the best based on the circumstances.[4]

Time Complexity is most commonly estimated by counting the number of elementary steps performed by any algorithm to finish execution. Algorithm's performance may vary with different types of input data, hence for an algorithm we usually use the worst-case Time complexity of an algorithm because that is the maximum time taken for any input size.[4]

4.2. SPACE COMPLEXITY:

The space complexity of an algorithm or a computer program is the amount of memory space required to solve an instance of the computational problem as a function of characteristics of the input. It is the memory required by an algorithm until it executes completely.[4]

Space complexity is an amount of memory used by the algorithm (including the input values of the algorithm), to execute it completely and produce the result. We know that to execute an algorithm it must be loaded in the main memory. The memory can be used in different forms:

- Variables (This includes the constant values and temporary values)
- Program Instruction
- Execution

Chapter 5

CHALLENGES

These were the main challenges, which we have faced and tried to resolve most of them.

- Although project till now was great and it served its purpose but there were some glitches associated with it.
- Some necessary features like suggesting a name when someone types any letter were missing.
- Some other improvements like optimizing the code, scaling the databases have also been tried.
- Issues related with upper case and lower case have also been fixed as discussed in later pages.

5.1. FIXING ISSUE WITH UPPER CASE AND LOWER CASE

In previous chapters, if we try to search any station in all capital letters or all small letters or mixed capital and small letters then it will not show any result.

So, to solve this problem we have first converted all the letters of source and destination to lower case using the JavaScript function to Lowercase () and then we have used this source and destination to compute all our results.

We could have used some other utilities like loadash to solve this issue. While showing the results we again converted back to standard format i.e., first letter capital and rest small.

5.2. Suggesting a name:

To make web application more user friendly we have added this feature. Whenever someone starts typing any name JavaScript running in the background will search for the possible names and will suggest it to the user.

For example, if someone has to write Silchar then just after typing S in the search bar, all the names starting with S will be shown below. Again when the user types I i.e.,Si then all the names starting with si will be shown in the space

This feature will make user easy to type anything. Moreover, it will also reduce typing errors.

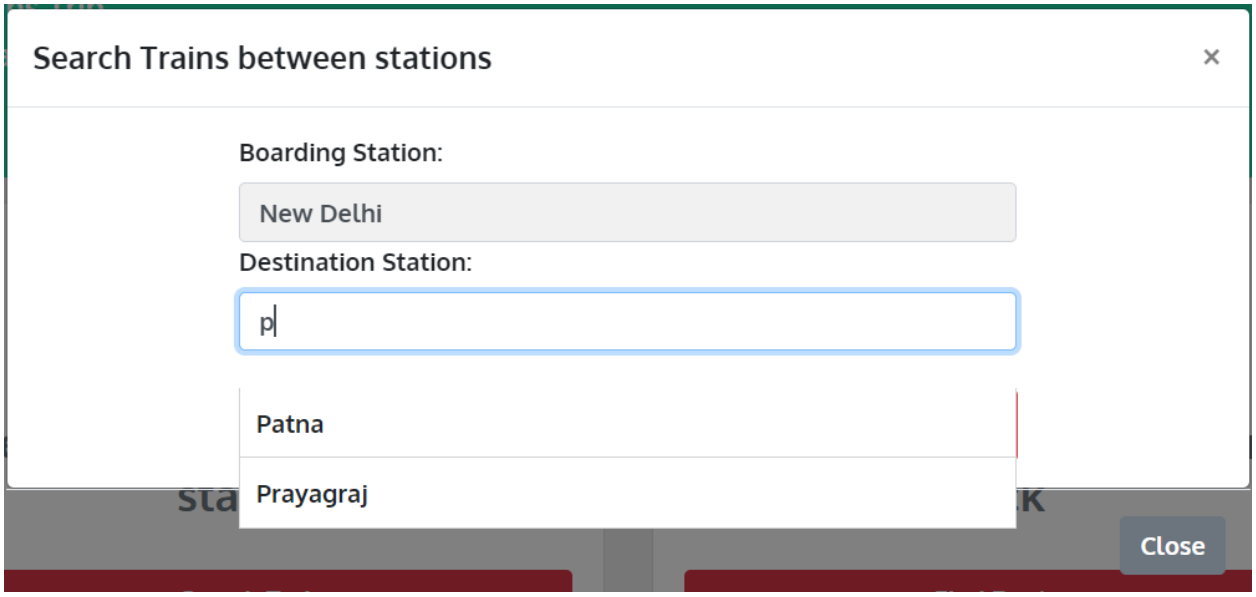


Figure 5.1: Suggesting name suggestions

5.3. Scaling the DATABASE:

Previous database was too small so, we have tried to make database bigger. We added a greater number of trains and the routes. As the database size has significantly increased, the computation time has increased also.

If previously the web app was able to show the result in 10 milliseconds, now it might take 20 millisecond's or more depending on the size of database increased.

To counter this increased time delay we have tried to improve our algorithm so that we can show the result as fast as possible.

In other words, we have tried to reduce the time complexity of our code.

5.4. OPTIMISING THE TIME COMPLEXITY:

The time complexity of Previous code was not that much good. For direct trains it was N*N and for indirect trains it was N raised to power 3.

Instead of making vector of trains we will make adjacency matrix of stations. Every train has its own adjacency matrix.

Let us suppose that there are total of 16 stations i.e., a train crosses total of 16 stations then then we will make a matrix of size 16x16.

If there is a train connects two stations let us suppose station A and station B for instance then adjacency matrix will store 1 at the index matrix[A][B].

Initially the matrix will be filled with 0 denoting that there is no train between any of two stations.

5.5. COST analysis

For optimizing the time complexity, we had to take n adjacency matrix where n is the total number of trains. For each train there was a matrix of size m*m where m was the total number of stations that particular train was visiting. So, we were able to reduce the time complexity from n*n to O (1) for direct train.

For indirect train we were able to reduce time complexity from O(n*n*n) to O(n). But we had to take adjacency matrix which cost us a total of m*n*n space where m is total number of trains and m is the total number of stations where train stops.

5.6. Adding Distances in each lap

Apart from showing the train names for each lap, we will be showing the distance travelled in each lap.

For each lap we will be showing the distance between the two stations.

For this we have taken the help of map. Map has been used to ease the retrieval of data i.e., map can do insertion and retrieval of data in almost constant time.

We have taken the reference of New Delhi and taken the distance of every station from New Delhi and stored in the map.

Let us suppose we have to go from New Delhi to Silchar and the possible option is that we first go from New Delhi to Kanpur and then Kanpur to Silchar then according to our approach the web page will first calculate the distance between Kanpur and New Delhi and will show it as lap1.After that it will calculate distance between Kanpur and Silchar and will show it as lap2.

5.7. Adding Train Number and Fare

To make it easier for the user we have added the train number and fare for each lap.For example let us suppose we have to go from New Delhi to Silchar and the possible option is to go first from New Delhi to Kanpur and then from Kanpur to Silchar then,

In lap1 it will be shown that user can go from New Delhi to Kanpur with the help of given train along with its number and fare will also be shown. Again, in lap2 it will be shown that user can go from Kanpur to Silchar with the help of given train and its number. Along with fare will also be shown.

Chapter 6

RESULTS

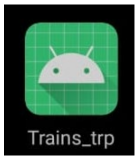
We are also building an App which is performing the same function as the web page. Apps are the applications that are downloaded and installed on our android devices, rather than being rendered within the browser. Apps are much faster and tends to be more advanced n terms of features and functionality.

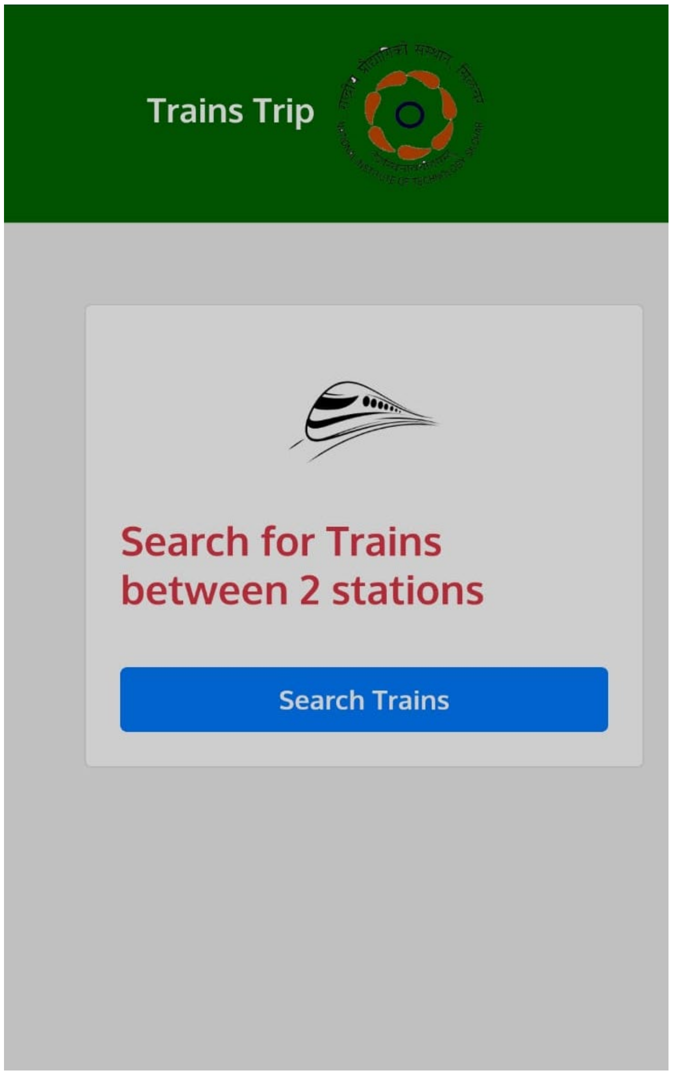
Average time spent on the App is estimated is very less as compared to the time spent in surfing the web pages which are performing the same task. Now Apps are becoming a part of our daily routine and there are very few who live in an isolated world away from Android apps. Apps has flexible interfaces and supports complex functionalities.

6.1. Technology Required for an App

- Android Studio: It is the official integrated development environment (IDE) for Google's Android Operating System, built on JetBrains IntelliJ IDEA software and designed specifically for Android development. Android Studio supports many programming languages like Java, C++, Kotlin etc. [5]
- Java: Java is one of the powerful general-purpose programming languages. Android heavily relies on the Java programming language all the SDKs required to build for android applications use the standard libraires of Java.[7]

6.2. App User Interface





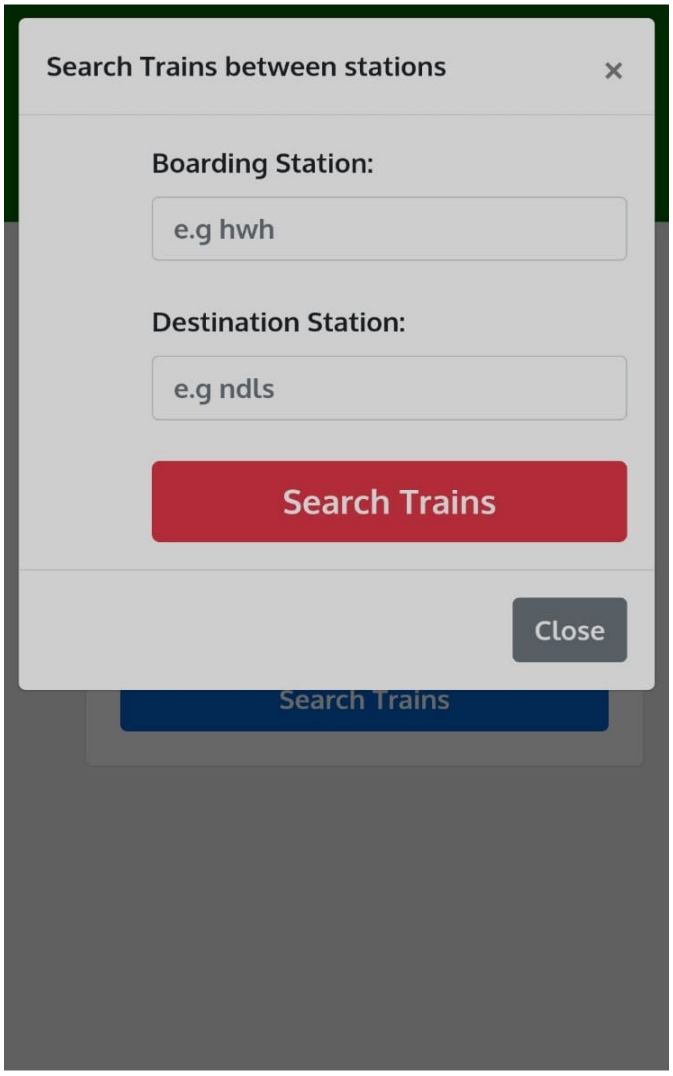


Figure 6.1: User-Interface of App

6.2: Pop-tab of Search Trains in App

Chapter 7

CONCLUSION & FUTURE SCOPE

7.1. Conclusion

We have developed a platform i.e. webpage and an app, which can make journey as easier as possible. Not every person can have knowledge of each immediate stoppage and the train passing through these stoppages, especially if he is travelling to a completely unfamiliar place. Here our platform plays its significant role in deciding a perfect route through which he can reach to his destination very conveniently.

7.2. Future Scope

Just take an example of India. India has the fourth largest railway network with over 22,593 operating trains (9141 freight and 13,452 passenger) with a daily passenger count of 24 million passengers. The whole network of Indian Railways across the country has 7,083 stations. We can store the whole database of these trains and stations, and perform queries on that database. But here we have to optimize the time complexity at the next level so that result is at the finger tip of the user.

Considering a small portion of daily passenger count, even it is also a huge number. Our platform could not able to handle this amount of traffic and providing services to each one of them.

Peak in online traffic can cause websites or apps to slow down and ultimately crash. To avoid it we will have to monitor the web page traffic and designing of everything should be done very wisely. Managing website can be a time-consuming task.

Apart from it there can be a lot of improvement like security issue, understanding user needs and make changes according to it time to time, optimizing content delivery etc.

Prototype

We have developed a webpage consisting of all those functionality as described above. Webpage is hosted on firebase. Firebase is a Google-backed application development software that enables developer to develop Web apps. The generated URL is:

http://trains-trip.web.app/

References

[1] Duckett, J., 2014. JavaScript and JQuery: Interactive Front-End Web Development. 1st ed. USA: Wiley, pp.7-70.

[2] S. Skiena, S., 2010. The Algorithm Design Manual. 2nd ed. Springer, pp.113-190.

[3] Cormen, T., Leiserson, C., Rivest, R. and Stein, C., n.d. Introduction to algorithms. 3rd ed. MIT Press; 3rd edition (September 1, 2009), pp.192-317.

[4] Bae, S., 2019. JavaScript Data Structures and Algorithms: An Introduction to Understanding and Implementing Core Data Structure and Algorithm Fundamentals. 1st ed. Apress, pp.167-189.

[5] Cameron, D., 2013. A Software Engineer Learns HTML5 , Javascript & Jquery. 1st ed. Createspace Independent, pp.143-168.

[6] Niederst Robbins, J., 2017. JavaScript: Programming Basics for Absolute Beginners (Step-By-Step JavaScript Book 1). 1st ed. Nathan Clark, pp.51-99.

[7] Schildt, H., 2004. Java 2. New York: McGraw-Hill/Osborne, pp.697-731.

Rail

ORIGINALITY REPORT

16%	16%	5%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	3%
2	devopedia.org Internet Source	2%
3	www.tutorialspoint.com Internet Source	2%
4	en.wikipedia.org Internet Source	2%
5	www.oniverse.in Internet Source	2%
6	www.slideshare.net Internet Source	1%
7	worldanalysis.net Internet Source	1%
8	www.geeksforgeeks.org Internet Source	1%
9	answeregy.com Internet Source	<1%

10	purehost.bath.ac.uk Internet Source	<1 %
11	algs4.cs.princeton.edu Internet Source	<1 %
12	dspace.chitkara.edu.in Internet Source	<1 %
13	export.arxiv.org Internet Source	<1 %
14	www.javatpoint.com Internet Source	<1 %
15	conferences.ulbsibiu.ro Internet Source	<1 %
16	brot-saputo.fun Internet Source	<1 %

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		