

# **Intelligent Train Reservation and Prediction System**

Project Report submitted in partial fulfillment of the requirement  
for the degree of

Bachelor of Technology  
in

**Information Technology**

Under the Supervision of

**Dr. Pooja Jain**

By

**Shagun Garg (111425)**

To



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

## Certificate

This is to certify that project report entitled **Intelligent Train Reservation and Prediction System**, submitted by **Shagun Garg** in partial fulfillment for the award of degree of Bachelor of Technology in Information Technology to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Dr. Pooja Jain

Assistant Professor (Senior Grade)

JUIT, Waknaghat

## Acknowledgement

On the very outset of this report, I would like to extend my sincere & heartfelt obligation towards all the personages who have helped me in this endeavor. Without their active guidance, help, cooperation & encouragement, I would not have made headway in the project.

I would like to show my greatest appreciation to **Dr. Pooja Jain**. I feel motivated every time I get her encouragement. For her coherent guidance throughout the tenure of the project, I feel fortunate to be taught by **Dr. Pooja Jain**, who gave me her unwavering support. Besides being my mentor, she taught me that there is no substitute for hard work.

I express my gratitude and sincere thanks to **Prof. R.M.K. Sinha**, Dean, Department of Computer Science and Engineering, for allowing me to undertake this project.

I deeply express my sincere thanks to **Brig. (Retd.) S. P. Ghrera**, Head of Department, Department of Computer Science and Engineering, for encouraging and allowing me to present this project

I would also thank my parents **Mr. Sanjay Garg** and **Mrs. Alpana Garg** for teaching me how to be successful and how to achieve my goal which helped in completing this project. They have supported me in every step of my life.

Date:

Shagun Garg

# **Table of Content**

Chapter 1. An Overview	1
1.1 Introduction	1
1.2 Motivation	2
1.3 Objective	2
1.4 Approach Used	2
1.5 Scope	3
Chapter 2. Literature survey	4
2.1 Geographic Coordinate System	4
2.1.1 Latitudes and Longitudes	4
2.2 Distance Calculation	5
2.2.1 Flat Earth Model	6
2.2.2 Spherical Earth Model	8
2.3 Positioning Representation	9
2.3.1 Standard Position Representation	10
2.3.1.1 Cartesian 3D Vector	10
2.3.1.2 Spherical Horizontal and Vertical Component	11
2.4 Prediction	12
2.4.1 Case based Reasoning	12
2.4.2 Association Rule	14
2.5 Probability	15
2.5.1 Conditional Probability	16
2.5.2 Bayes Theorem	17
2.5.3 Bayesian Probability	18
Chapter 3. Application Designing	19
3.1 Feasibility Study	19
3.1.1 Operational Feasibility	19
3.1.2 Technical Feasibility	19
3.1.3 Social Feasibility	20
3.2 Basic Design	20
3.2.1 Railway System	20
3.2.2 Prediction System	23
3.3 UML Diagrams	25

3.3.1 Use Case	25
3.3.2 Sequence Diagram	25
3.3.3 Data Flow Diagram	27
3.4 Code Snippets	29
3.5 Database Design	48
3.6 Snapshots	56
Chapter 4. Conclusion and Future Work	61
References	62

## **List of Tables**

Table 1: Symbols used to describe basic relations between two coordinate frames.	10
Table 2: user_detail	48
Table 3: train_detail	49
Table 4: lat_long	50
Table 5: station_code	51
Table 6: train_code	52
Table 7: user	53
Table 8: probabilities	54
Table 9: trip_details	55

## List of Figures

Figure 1: Latitude- Longitude	5
Figure 2: Flat Earth Model	6
Figure 3: Latitude Length	8
Figure 4: Longitude Length	8
Figure 5: Flat and Spherical Earth Model	9
Figure 6: Use case	25
Figure 7: Sequence Diagram for registration	26
Figure 8: Sequence Diagram for Booking Ticket	26
Figure 9: DFD level 0	27
Figure 10: DFD level 1	28
Figure 11: Database User_detail	48
Figure 12: Database train_detail	49
Figure 13: Database lat_long	50
Figure 14: Database station_code	51
Figure 15: Database train_route	52
Figure 16: Database user_details	53
Figure 17: Database probabilities	54
Figure 18: Database trip_details	55
Figure 19: Sign Up	56
Figure 20: Successfully Registered	56
Figure 21: Login	57
Figure 22: Forgot Password	57
Figure 23: User home (Direct Route Example)	58
Figure 24: Trains Output (Direct Route Example)	58
Figure 25: User home (No Direct Route Example)	59
Figure 26: Trains Output (Direct Route Example)	59
Figure 27: Prediction output	60

## Abstract

Time and ease of work are the most important things that one wants while doing some work. Everyone wishes to minimize the effort in doing his/her work and also wants to reduce the time that work is taking.

Talking about Indian Railways, the largest railway network in the world. There are lakhs of people who book their tickets electronically via irctc.co.in, indianrailways.gov.in or some other website. The existing websites only show the trains between the two directly connected stations only that means if both the stations have a direct train between them. Now suppose if we have a person who wants to go from a source to destination which are not directly the existing systems shows “No Train Exists”.

This project is designed to overcome the above drawback of the existing system. Removing this drawback of existing system of showing the message No Train Exists this system will tell the route between the two stations, like an intermediate station where you can switch train for the destination.

This was one part of saving time. Secondly for a person with a tight schedule, he/she sometimes forget to get his/her tickets done on time and stucked at last time with no other option of paying higher amount then required. Here comes the second feature of this system, Prediction System. It will keep a past record of all journeys made by the user In last three years and based on this past record, it will predict the future journey of user and send an alert mail to the respective user reminding him/her about the journey and asking if he/she is willing to make the trip or not and if yes, then to get the tickets done on time. Thus saving the headache of remembering the dates before as users will have a remainder system and can save their money and time.



# **Chapter 1: An Overview**

## **1.1 Introduction**

Indian Railways, one of the largest railway network of the world. There are around 23 million people that travel daily out of which half are suburban passengers (Delhi, Mumbai, Kolkata metro etc.). There are 7,172 railway stations in India out of which only 1500(approx.) stations provide reservation facility. We have 12,617 passenger trains running across the network and only 2,578 trains have reservation coaches like the trains Rajdhani Express, Shatabdi Express, Mail/Superfast Express etc.

For getting a reservation in trains initially we used stand in long queues on railway station to get a reserved ticket. But with advancement of time and technology, railways started online reservation facility using various sites like irctc.co.in, indianrail.gov.in etc. and making it easy for passengers to get their tickets done by paying a little bit higher prices for saving their time and for ease of work.

All the websites for online reservation first ask for the stations to travel between and then show all the trains between those two stations and user can then select any train that suits him/her. And in case if there is no train between those two, a message is prompted that No Train Exists. That means that there is no direct train between those two stations. In such cases we first need to look for all the intermediate stations between those two using google search. After this we need to look for the trains that we can have for those intermediate stations and that suits our timings also like we need to wait for the minimum time during changing trains and also the shortest route which can follow. This work consumes a lot of time and a headache for the user. And this a loop hole in present online reservation system which needs to be overcome and cleared off.

For a person with a busy schedule and does not have time to remember about his tickets to be done for a journey that he makes every year or every month on same dates or nearby dates. At the time of journey or a day or two before the journey, they realize that their tickets are not done or are still pending or are in waiting state faces a problem. For such people their needs to be remainder sort of thing that could remind them to getting their

tickets done right on time so that they can make their journey easily. For such thing I have a Prediction system. This system will have a past record, a record of past three years of all the users and based on this record their future journeys will be predicted and an email will send to them as a reminder to get their tickets done on right time.

## **1.2 Motivation**

The motivation of this project comes from the above explained loop hole in present system. This loop hole is a big problem for people who have less knowledge about railways and are not frequent travelers and totally relay on the website they are using. In the case of no direct train between the stations they have to travel, they get confused and for them a question arrives, What to do next? To solve this dilemma I thought of making such a system that could solve this loop hole and removes the above question and helps the people to ease out their work.

The motivation for the second part comes from live experiences we face when we forget to get our tickets on time, so a prediction system can help us in reminding about the journey's that are likely to be made.

## **1.3 Objective**

The Project aims in building an Intelligent Train Reservation Prediction System. The system will focus on two things:

- To show the route and trains between the stations that are not directly connected to each other.
- To predict the future journeys of the user's based on their past journeys.

## **1.4 Approach Used**

In this system we need to find the cities that lie between the source and destination. For this purpose we could different ways but the one that I used is the Latitude and Longitude of the cities. I took the latitudes and longitudes of the source and destination and the look for all the cities whose lats and longs lie between the lats and longs of the source and

destination cities. Now I looked that which all cities have reservation facility. Then finally finding the train connectivity between all the cities that we calculated on basics on lats and longs. And the result is shown to the user.

For the second the approach followed is as follows

For each user, the dates of all the journey's made in previous three years are stored. Each time a user is picked whose future journey's has to be predicted. The system fetches the stored data of particular user i.e. the previous year dates, compares them to one another that the dates are same or nearby dates (nearby dates are defined as  $\pm 3$  days). Accordingly each user is given the probability about his/her prediction. Then the user with a minimum threshold probability will be send an alert mail reminding the user to get the tickets done if he/she is interested for the journey.

## **1.5 Scope**

The scope of this application can be very wide. If the system is designed accurately and exactly of what is thought to be design then it can solve a big problem of existing system. This system could give a big assistance or could be a huge add on to the present system. This will help all those who possess less knowledge about railway network and wants to do their works on click.

## **Chapter 2: Literature survey**

### **2.1 Geographic Coordinate system**

A geographic coordinate system (GCS) uses a three-dimensional spherical surface to define locations on the earth. A GCS is often incorrectly called a datum, but a datum is only one part of a GCS. A GCS includes an angular unit of measure, a prime meridian, and a datum (based on a spheroid).

A point is referenced by its longitude and latitude values. Longitude and latitude are angles measured from the earth's center to a point on the earth's surface. The angles often are measured in degrees (or in grads). The following illustration shows the world as a globe with longitude and latitude values.

In the spherical system, horizontal lines, or east–west lines, are lines of equal latitude, or parallels. Vertical lines, or north–south lines, are lines of equal longitude, or meridians. These lines encompass the globe and form a gridded network called a graticule

#### **2.1.1 Latitude and Longitude**

The line of latitude midway between the poles is called the equator. It defines the line of zero latitude. The line of zero longitude is called the prime meridian. For most geographic coordinate systems, the prime meridian is the longitude that passes through Greenwich, England. Other countries use longitude lines that pass through Bern, Bogota, and Paris as prime meridians. The origin of the graticule (0,0) is defined by where the equator and prime meridian intersect. The globe is then divided into four geographical quadrants that are based on compass bearings from the origin. North and south are above and below the equator, and west and east are to the left and right of the prime meridian.

Latitude and longitude values are traditionally measured either in decimal degrees or in degrees, minutes, and seconds (DMS). Latitude values are measured relative to the equator and range from  $-90^\circ$  at the South Pole to  $+90^\circ$  at the North Pole. Longitude values are measured relative to the prime meridian. They range from  $-180^\circ$  when traveling west to  $180^\circ$  when traveling east. If the prime meridian is at Greenwich, then Australia, which is south of the equator and east of Greenwich, has positive longitude values and negative

latitude values. It may be helpful to equate longitude values with X and latitude values with Y. Data defined on a geographic coordinate system is displayed as if a degree is a linear unit of measure.

Although longitude and latitude can locate exact positions on the surface of the globe, they are not uniform units of measure. Only along the equator does the distance represented by one degree of longitude approximate the distance represented by one degree of latitude. This is because the equator is the only parallel as large as a meridian. (Circles with the same radius as the spherical earth are called great circles. The equator and all meridians are great circles.)

Above and below the equator, the circles defining the parallels of latitude get gradually smaller until they become a single point at the North and South Poles where the meridians converge. As the meridians converge toward the poles, the distance represented by one degree of longitude decreases to zero. On the Clarke 1866 spheroid, one degree of longitude at the equator equals 111.321 km, while at 60° latitude it is only 55.802 km. Because degrees of latitude and longitude don't have a standard length, you can't measure distances or areas accurately or display the data easily on a flat map or computer screen.

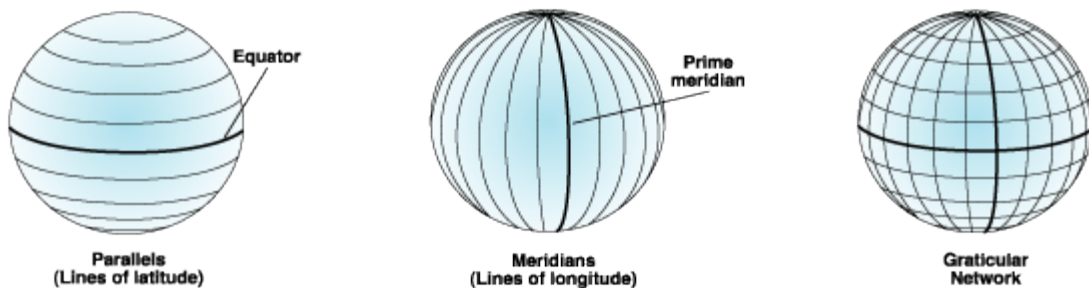


Figure 1: Latitude- Longitude

## **2.2 Distance Calculation**

The current FCC Rules for distance calculations use two methods: flat-earth and spherical-earth. The flat earth method assumes the distance between two points to be the hypotenuse of a right triangle whose sides are determined by the difference in latitude and longitude of the starting and ending points, multiplied by the length per degree of latitude and longitude

at the mid-latitude of the two points, as shown in Figure below. The *flat-earth* term is not meant to be disparaging, and merely refers to the use of a right triangle to calculate the distance. Because the lengths of a degree of longitude and a degree of latitude used in the flat earth method are derived from an ellipsoid rather than a spheroid model of the earth, the flat-earth method is actually more accurate than the spherical-earth method for short to moderate distances.

The spherical-earth method uses conventional spherical trigonometry to determine the distance. Section 73.208 of the FCC Rules now requires that the flat-earth method be used for distances up to and including 475 km. Distances greater than 475 km must be calculated using the spherical-earth method, which becomes more accurate than the flat-earth method for large distances.

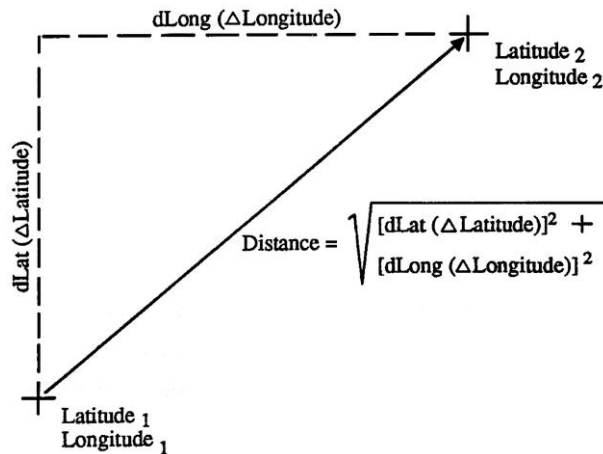


Figure 2: Flat Earth Method

### 2.2.1 Flat Earth Method

In FCC Docket 80-90, formulas were substituted for the tables previously used for determining the length of a degree of latitude or of longitude as a function of latitude. However, the coefficients adopted in Docket 80-90 truncated to only two terms the trigonometric series used to generate the tables and “adjusted” the coefficients by a factor of (1.609/1.609347) because of the Docket 80-90 decision to define the conversion factor

from U.S. statute miles to kilometers as 1.609, rather than the value of (5,280 ft/mile) \* (1200/3937 m/ft) \* (1/1000 km/m), or 1.609347219 km/mile (approximately).

In the Second Report and Order to Docket 86-144, the FCC corrected these problems by adopting the full precision, non-truncated trigonometric series for the arc length formulas given in the 1966 edition of U.S. Naval Hydrographic Office Publication Number 9 (H.O. 9) or the American Practical Navigator or simply Bowditch after Nathaniel Bowditch (1773–1838), its original author. These trigonometric series are based upon a binomial theorem expansion of an ellipsoid model of the earth corresponding to the Clarke spheroid of 1866, upon which topographic maps in the United States are currently based.<sup>6</sup>

The trigonometric series defining the length of one degree of latitude and one degree of longitude for the Clarke spheroid of 1866 are:

$$dLat = 111.13209 - 0.56605 \cos(2L) + 0.00120 \cos(4L)$$

$$dLong = 111.41513 \cos(L) - 0.09455 \cos(3L) + 0.00012 \cos(5L)$$

where, dLat is the length in kilometers of one degree of latitude at latitude L and dLong is the length in kilometers of one degree of longitude, again at latitude L.

The latitude, L, is taken as the mid-latitude of the two points between which the distance is to be calculated, as follows:

$$L = (\text{Latitude}_1 + \text{Latitude}_2)/2$$

Where, Latitude<sub>1</sub> and Latitude<sub>2</sub> are the latitudes of the starting and ending points. Similarly, Longitude<sub>1</sub> and Longitude<sub>2</sub> are the longitudes of the starting and ending points. In all cases, north latitudes are treated as positive and south latitudes as negative, and west longitudes are treated as positive and east longitudes as negative.

The distance between two points is then given by the Pythagorean Theorem:

$$D = ([dLat(Lat_1 - Lat_2)]^2 + [dLong(Long_1 - Long_2)]^2)^{1/2}$$

Plots showing how the lengths of one degree of latitude and longitude vary with latitude are given in Figures 3 and 4.

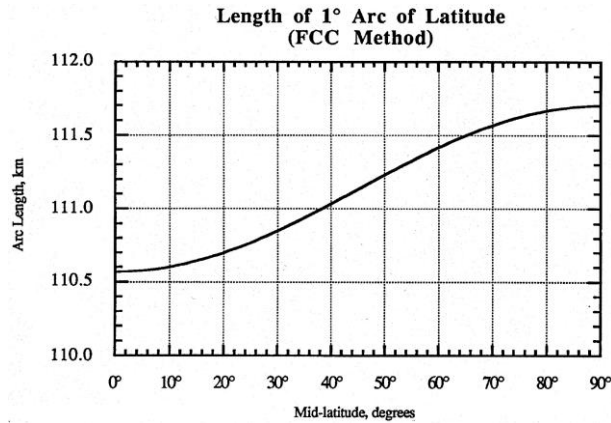


Figure 3: Latitude Length

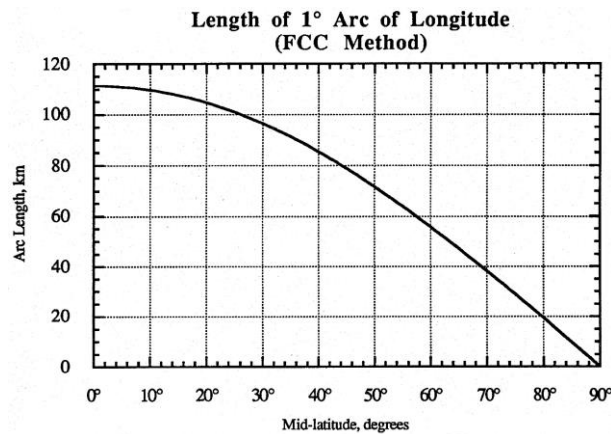


Figure 4: Longitude Length

## 2.2.2 Spherical Earth Method

The formula for the spherical-earth distance, or great-circle distance, is:

$$D = K \cos^{-1}[(\sin \text{Lat}_1)(\sin \text{Lat}_2) + (\cos \text{Lat}_1)(\cos \text{Lat}_2) \cos(\text{Long}_2 - \text{Long}_1)]$$

The constant K is in km/degree and is determined by the radius of the sphere being modeled. The FCC has never defined the earth radius to be used for spherical earth calculations. The example given in Section 73.185(d) of the FCC Rules suggests an earth radius of 6,365 km (K = 111.090 km/degree). A 6,373 km radius (K = 111.230 km/degree) is implied by the 5,280-mile <sup>4</sup>/<sub>3</sub>-earth radius given in Section 73.684(c)(1) of the FCC Rules. This <sup>4</sup>/<sub>3</sub>-earth radius was also used in FCC Report No. R-6410, “Elevation and Depression Angle Tables,” September 15, 1964. An earth radius of 6,367 km (K = 111.125 km/degree) can be deduced from the 1,852-meter definition of a nautical mile.



Finally, an earth radius of 6,371 km ( $K = 111.195$  km/degree) corresponds to the mean radius of the Clarke spheroid of 1866.

Until such time as the FCC so specifies, the author suggests using the mean Clarke spheroid value of 6,371 km ( $K = 111.195$  km/degree).

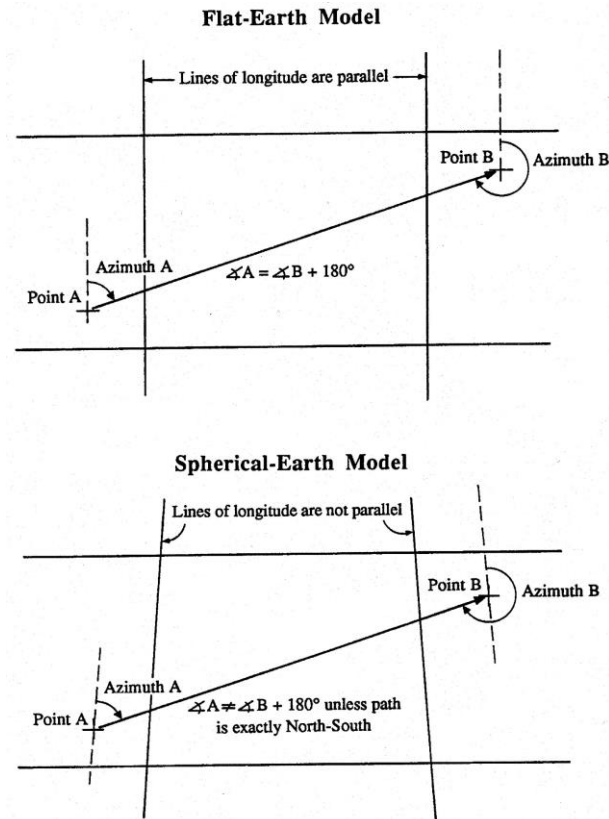


Figure 5: Flat and Spherical Earth Model

## 2.3 Position Representation

Position calculations, e.g. adding, subtracting, interpolating, and averaging positions, depend on the representation used, both with respect to simplicity of the written code and accuracy of the result. The latitude/longitude representation is widely used, but near the pole singularities, this representation has several complex properties, such as error in latitude leading to error in longitude. Longitude also has a discontinuity at  $\pm 180^\circ$ . These properties may lead to large errors in many standard algorithms. Using an ellipsoidal Earth model also makes latitude/longitude calculations complex or approximate. Other common representations of horizontal position include UTM and local Cartesian ‘flat Earth’

approximations, but these usually only give approximate answers, and are complex to use over larger distances. The normal vector to the Earth ellipsoid (called n-vector) is a non-singular position representation that turns out to be very convenient for practical position calculations. This paper presents this representation, and compares it with other alternatives, showing that n-vector is simpler to use and gives exact answers for all global positions, and all distances, for both ellipsoidal and spherical Earth models. In addition, two functions based on n-vector are presented, that further simplify most practical position calculations, while ensuring full accuracy.

### 2.3.1 Standard Position Representation

Quantity	Symbol	Description
Position vector	$\sim p_{AB}$	A vector whose length and direction is such that it goes from the origin of frame A to the origin of frame B, i.e. the position of B relative to A.
Velocity vector	$\sim \underline{v}_A B$	The velocity of the origin of frame B, relative to frame A. The underline indicates that both the position and orientation of A is relevant (whereas only the position of B matters).
Rotation matrix	$R_{AB}$	A 3x3 direction cosine matrix (DCM) describing the orientation of frame B relative to frame A.
Angular velocity	$\underline{v} \sim_{AB}$	The angular velocity of frame B relative to frame A.

Table 1: Symbols used to describe basic relations between two coordinate frames.

#### 2.3.1.1 Cartesian 3D vector

When representing the position of a general coordinate frame B relative to a reference coordinate frame A, the most intuitive quantity to use is the position vector from A to B, decomposed in A,  $p^A_{AB}$ . We can represent the position of a body frame (B) relative to the

Earth (E), by using  $p_{EB}^E$ . This (Cartesian) vector is often referred to as Earth Centered Earth Fixed (ECEF) vector. While this representation is non-singular and intuitive, there are many situations where other representations are more practical when positioning an object relative to the Earth reference ellipsoid.

### **2.3.1.2 Separating horizontal and vertical components**

For many position calculations, it is desirable and most intuitive to treat horizontal and vertical positions independently. This is for instance useful in a navigation system, where horizontal and vertical position are usually measured by different sensors at different points in time, or in a vehicle autopilot, where horizontal and vertical position are often controlled independently. In such applications, we usually compare two horizontal positions, and thus we need a quantity for representing horizontal position independently of the vertical height/depth. It should thus be possible to represent horizontal position without considering the vertical position, and vice versa. If the vector  $p_{EB}^E$  is used, the horizontal and vertical positions are not separated as desired.

### **Latitude and longitude**

A common solution for obtaining separate horizontal and vertical positions is the use of latitude, longitude and height/depth. However, this representation has a severe limitation; the two singularities at latitudes  $\pm 90^\circ$ , where longitude is undefined. In addition, when getting close to the singularities, the representation exhibits considerable non-linearity and extreme latitude dependency, leading to reduced accuracy in many algorithms. Thus, these coordinates are not suitable for algorithms that should be able to calculate positions far north or far south. In addition, calculations near  $\pm 180^\circ$  longitude become complicated due to the discontinuity.

### **UTM and UPS**

Horizontal position can also be represented by defining an Earth fixed coordinate system based on a map projection (i.e., a mapping of points on a curved surface to a plane) valid in a limited geographical area. One such system is Universal Transverse Mercator (UTM), specifying 60 longitude zones, covering the globe except for the Polar Regions (Snyder,

1987). For the Polar Regions, a similar system, Universal Polar Stereographic (UPS), defines horizontal positions (Hager et al., 1989). While these systems are well-defined and the coordinate values approximately correspond to meters, they have an inherent distortion due to the projection and thus a corresponding error in many calculations (e.g. a difference vector between two UTM coordinates will give a length (in meters) and direction (relative to north) that both have errors compared to the true values). In addition, general calculations get very complex when crossing zones (Hager et al., 1989).

### **Rotation matrix**

In a set of navigation equations, integrating measurements from an inertial measurement unit, horizontal position is often stored together with an azimuth angle in a rotation matrix (Savage, 2000). Although it has nice properties with respect to the pole singularities (similar to n-vector), this matrix representation is not suited for pure horizontal position representation.

## **2.4 Prediction**

A prediction or forecast is a statement about the way things will happen in the future, often but not always based on experience or knowledge. While there is much overlap between prediction and forecast, a prediction may be a statement that some outcome is expected, while a forecast is more specific, and may cover a range of possible outcomes. A "prediction" may be contrasted with a "projection", which is explicitly dependent on stated assumptions.

Although guaranteed accurate information about the future is in many cases impossible, prediction can be useful to assist in making plans about possible developments

### **2.4.1 Case based reasoning**

Case-based reasoning (CBR), broadly construed, is the process of solving new problems based on the solutions of similar past problems. An auto mechanic who fixes an engine by recalling another car that exhibited similar symptoms is using case-based reasoning.

A lawyer who advocates a particular outcome in a trial based on legal precedents or a judge who creates case law is using case-based reasoning. So, too, an engineer copying working elements of nature (practicing biomimicry), is treating nature as a database of solutions to problems. Case-based reasoning is a prominent kind of analogy making.

It has been argued that case-based reasoning is not only a powerful method for computer reasoning, but also a pervasive behavior in everyday human problem solving; or, more radically, that all reasoning is based on past cases personally experienced. This view is related to prototype theory, which is most deeply explored in cognitive science.

## **Process**

Case-based reasoning has been formalized for purposes of computer reasoning as a four-step process.

1. **Retrieve**: Given a target problem, retrieve from memory cases relevant to solving it. A case consists of a problem, its solution, and, typically, annotations about how the solution was derived. For example, suppose Fred wants to prepare blueberry pancakes. Being a novice cook, the most relevant experience he can recall is one in which he successfully made plain pancakes. The procedure he followed for making the plain pancakes, together with justifications for decisions made along the way, constitutes Fred's retrieved case.
2. **Reuse**: Map the solution from the previous case to the target problem. This may involve adapting the solution as needed to fit the new situation. In the pancake example, Fred must adapt his retrieved solution to include the addition of blueberries.
3. **Revise**: Having mapped the previous solution to the target situation, test the new solution in the real world (or a simulation) and, if necessary, revise. Suppose Fred adapted his pancake solution by adding blueberries to the batter. After mixing, he discovers that the batter has turned blue – an undesired effect. This suggests the following revision: delay the addition of blueberries until after the batter has been ladled into the pan.

4. **Retain:** After the solution has been successfully adapted to the target problem, store the resulting experience as a new case in memory. Fred, accordingly, records his new-found procedure for making blueberry pancakes, thereby enriching his set of stored experiences, and better preparing him for future pancake-making demands.

### 2.4.2 Association rule

Association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. It is intended to identify strong rules discovered in databases using different measures of interestingness. Based on the concept of strong rules, association rules were introduced for discovering regularities between products in large-scale transaction data recorded by point-of-sale (POS) systems in supermarkets. For example, the rule {onions,potato} => {burger} found in the sales data of a supermarket would indicate that if a customer buys onions and potatoes together, they are likely to also buy hamburger meat. Such information can be used as the basis for decisions about marketing activities such as, e.g., promotional pricing or product placements. In addition to the above example from market basket analysis association rules are employed today in many application areas including Web usage mining, intrusion detection, Continuous production, and bioinformatics. In contrast with sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

Association rules are created by analyzing data for frequent if/then patterns and using the criteria support and confidence to identify the most important relationships. Support is an indication of how frequently the items appear in the database. Confidence indicates the number of times the if/then statements have been found to be true. In data mining, association rules are useful for analyzing and predicting customer behavior. They play an

important part in shopping basket data analysis, product clustering, and catalog design and store layout. Programmers use association rules to build programs capable of machine learning. Machine learning is a type of artificial intelligence (AI) that seeks to build programs with the ability to become more efficient without being explicitly programmed.

## **Process**

Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Association rule generation is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent itemsets in a database.
2. Second, these frequent itemsets and the minimum confidence constraint are used to form rules.

While the second step is straightforward, the first step needs more attention.

Finding all frequent itemsets in a database is difficult since it involves searching all possible itemsets (item combinations). The set of possible item sets is the power set over  $I$  and has size  $2^n - 1$  (excluding the empty set which is not a valid itemset). Although the size of the powerset grows exponentially in the number of items  $n$  in  $I$ , efficient search is possible using the downward-closure property of support which guarantees that for a frequent itemset, all its subsets are also frequent and thus for an infrequent itemset, all its supersets must also be infrequent. Exploiting this property, efficient algorithms can find all frequent itemsets.

## **2.5 Probability**

Probability is the measure of the likeliness that an event will occur. Probability is quantified as a number between 0 and 1 (where 0 indicates impossibility and 1 indicates certainty). The higher the probability of an event, the more certain we are that the event will occur. These concepts have been given an axiomatic mathematical formalization

in probability theory (see probability axioms), which is used widely in such areas of study as mathematics, statistics, finance, gambling, science (in particular physics), artificial intelligence/machine learning, computer science, and philosophy to, for example, draw inferences about the expected frequency of events. Probability theory is also used to describe the underlying mechanics and regularities of complex systems.

### **2.5.1 Conditional Probability**

In probability theory, a conditional probability measures the probability of an event given that (by assumption, presumption, assertion or evidence) another event has occurred. For example, the probability that any given person has a cough on any given day may be only 5%. But if we know or assume that the person has a cold, then they are much more likely to be coughing. The conditional probability of coughing given that you have a cold might be a much higher 75%. The concept of conditional probability is one of the most fundamental and one of the most important concepts in probability theory. But conditional probabilities can be quite slippery and require careful interpretation. For example, there need not be a causal or temporal relationship between  $A$  and  $B$ .

Given two events  $A$  and  $B$  from the sigma-field of a probability space with  $P(B) > 0$ , the conditional probability of  $A$  given  $B$  is defined as the quotient of the probability of the joint of events  $A$  and  $B$ , and the probability of  $B$ :

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

If the event of interest is  $A$  and the event  $B$  is known or assumed to have occurred, "the conditional probability of  $A$  given  $B$ ", or "the probability of  $A$  under the condition  $B$ ", is usually written as  $P(A|B)$ , or sometimes  $P_B(A)$ .

In general, it cannot be assumed that  $P(A|B) = P(B|A)$ . This can be an insidious error, even for those who are highly conversant with statistics. The relationship between  $P(A|B)$  and  $P(B|A)$  is given by Bayes' theorem.



## 2.5.2 Bayes' Theorem

In probability theory and statistics, Bayes' theorem (alternatively Bayes' law or Bayes' rule) relates current probability to prior probability. It is important in the mathematical manipulation of conditional probabilities. When applied, the probabilities involved in Bayes' theorem may have different interpretations. In one of these interpretations, the theorem is used directly as part of a particular approach to statistical inference. In particular, with the Bayesian interpretation of probability, the theorem expresses how a subjective degree of belief should rationally change to account for evidence: this is Bayesian inference, which is fundamental to Bayesian statistics. However, Bayes' theorem has applications in a wide range of calculations involving probabilities, not just in Bayesian inference.

Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)},$$

Where A and B are events.

- $P(A)$  and  $P(B)$  are the probabilities of A and B independent of each other.
- $P(A | B)$ , a conditional probability, is the probability of A given that B is true.
- $P(B | A)$ , is the probability of B given that A is true.

For example: Let  $A_i$  denote the event that a randomly chosen item was made by the  $i$ th machine (for  $i = 1,2,3$ ). Let B denote the event that a randomly chosen item is defective. Then, we are given the following information:

$$P(A_1) = 0.2, \quad P(A_2) = 0.3, \quad P(A_3) = 0.5.$$

If the item was made by machine  $A_1$ , then the probability that it is defective is 0.05; that is,  $P(B | A_1) = 0.05$ . Overall, we have

$$P(B | A_1) = 0.05, \quad P(B | A_2) = 0.03, \quad P(B | A_3) = 0.01.$$

To answer the original question, we first find  $P(B)$ . That can be done in the following way:

$$P(B) = \sum_i P(B | A_i) P(A_i) = (0.05)(0.2) + (0.03)(0.3) + (0.01)(0.5) = 0.024.$$

Hence 2.4% of the total output of the factory is defective.

### **2.5.3 Bayesian Probability**

Bayesian probability is one interpretation of the concept of probability. The Bayesian interpretation of probability can be seen as an extension of propositional logic that enables reasoning with hypotheses, i.e., the propositions whose truth or falsity is uncertain. Bayesian probability belongs to the category of evidential probabilities; to evaluate the probability of a hypothesis, the Bayesian probabilistic specifies some prior probability, which is then updated in the light of new, relevant data (evidence). The Bayesian interpretation provides a standard set of procedures and formulae to perform this calculation. In contrast to interpreting probability as the "frequency" or "propensity" of some phenomenon, Bayesian probability is a quantity that we assign for the purpose of representing a state of knowledge, or a state of belief. In the Bayesian view, a probability is assigned to a hypothesis, whereas under the frequentist view, a hypothesis is typically tested without being assigned a probability.

# **Chapter 3: Application Designing**

## **3.1 Feasibility Study**

“FEASIBILITY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. The feasibility of the project is analyzed in this phase. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed environment must be feasible. For feasibility analysis, some understanding of the major requirements for system is essential.

The key objective of the feasibility study is to weigh up two types of feasibility. They are:

- Operational feasibility
- Technical feasibility
- Social feasibility

### **3.1.1 Operational feasibility**

Operational feasibility is necessary as it ensures that the project developed is a successful one. As the execution process of the proposed work is very much user friendly, the operational feasibility of the project is high.

### **3.1.2 Technical feasibility**

Technical feasibility analysis makes a comparison between the level of technology available and that is needed for the project development of the project. The level of technology consists of the factors like software tools, machine environment, and platform developed and so on.

### **3.1.3 Social feasibility**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

## **3.2 Basic Design**

### **3.2.1 Railway system**

The basic design of this system focuses on finding the route between two stations. For the route we need to have the list of cities/stations that lie between them. We have many ways to get this list. Few of them are:

- Using Google Maps
- Creating a network of cities
- Using Latitudes and Longitudes

#### **Using Google Maps**

In this method I need to integrate Google Maps into my system. The source and destination would be the input to the Google Map API and the list of cities that lie between them would be my output. In this my system would totally relay on the Maps. Any error or any disturbance or any kind of change in format of API working or any other changes to the working of Google Maps would affect my system and a result the system would fail to work. Moreover for using this API I would need a continuous Internet connection connected to my system and if I don't have any connection this system will not work. And depending on the speed of Internet the processing speed of my system will be affected making it slow in giving the output.

#### **By creating a network of cities**

In this method a large network of all the cities that have reservation facility around 1500 cities needs to be created, That roughly means a database graph with 1500 cities that would be stored in a matrix form of size 1500\*1500. This task needs a big database and a huge processing time in comparing all the values of the matrix. We first need to look for the cities that are closer to the source and then for all those cities we need to check whether they have a direct route to destination or not. If not, then we need to go back to database again look for some cities connected to source and follow the same process for all those until we find the a city that we a direct connectivity with destination also. This whole task takes a huge memory space, rather memory is not a constraint these days, But it was taking a long time in accessing the database many times and hence was slower.

The above two methods have their own drawbacks and hence are not implemented in this system. What is implemented here is the third option described below.

### Using Latitudes and Longitudes

In this method I have used a database of all major cities with their latitudes and longitudes. First the latitudes and longitudes of the source and destination were fetched and stored. Then one by one all the cities with their latitudes and longitudes are fetched and compared with the stored latitudes and longitudes. If the new fetched coordinates lie between the stored one then this city have a possibility of lying between the route between source and destination and is stored. After checking for all the cities, all the cities which satisfy this condition are stored in an array. Each city from the array list is then checked for having a direct connectivity and as soon as we find such a city the system shows the result with all the trains from source to the intermediate city found above and from that city to destination.

#### For example:

- 1) We have two stations Chandigarh (cdg) and Deoband (dbd), these two stations don't have any direct train between them so in such cases this method works.

Latitude and Longitude of cdg are: 30°43'N and 76°47'E

Latitude and Longitude of dbd are: 29°42'N and 77°43'E

Now we will look for the lats and longs that lie between these two coordinates.

AMBALA CANT JN 30°23'N 76°56'E

Jagadhri 30°10'N 77°20'E

Karnal 29°42'N 77°02'E

Saharanpur 29°58'N 77°33'E

So all these cities have lats and longs in relation with given stations. So now the system will check for the trains between the source and the cities in the list one by one. Once we have train between source and any city the system start checking the train between that city and destination, if we get some trains between the city and destination, the system will show the output making that city as intermedicator city. As in this the intermedicator city will be Ambala Cant Jn and the output will all the trains between source and Ambala cant and the trains between Ambala cant and destination will be shown as final output.

- 2) We have two stations Chandigarh (cdg) and Hapur (hpu), these two stations don't have any direct train between them so in such cases this method works.

Latitude and Longitude of cdg are: 30°43'N and 76°47'E

Latitude and Longitude of hpu are: 28°45'N and 77°45'E

Now we will look for the lats and longs that lie between these two coordinates.

AMBALA CANT JN 30°23'N 76°56'E

Chandigarh 30°43'N 76°47'E

Deoband 29°42'N 77°43'E

Hapur 28°45'N 77°45'E

Jagadhri 30°10'N 77°20'E

Kandhla 29°18'N 77°19'E

Karnal 29°42'N 77°02'E

Khatauli 29°17'N 77°43'E

Khekra 28°52'N 77°20'E

Meerut City 29°01'N 77°42'E

Muzaffarnagar 29°26'N 77°40'W

Panipat JN 29°25'N 77°02'E

Saharanpur 29°58'N 77°33'E

Sonipat 29°0'N 77°5'E

So all these cities have lats and longs in relation with given stations. So now the system will check for the trains between the source and the cities in the list one by one. Once we have train between source and any city the system start checking the train between that city and destination, if we get some trains between the city and destination, the system will show the output making that city as intermedaiator city. As in this the intermedaiator city will be Saharanpur and the output will all the trains between source and Saharanpur and the trains between Saharanpur and destination will be shown as final output.

### **3.2.2 Prediction System**

The basic design of this part focuses on fetching data, comparing data, assigning probabilities and sending alert based on probabilities.

First the stored data of a particular user is fetched i.e. the dates of his/her previous journeys. These dates are compared through various cases. According to the case matched they are assigned a probability of making a journey in next year and if the user has a probability greater than threshold probability, an alert/remainder mail is sent to the user.

There are many cases to be checked upon like

1. If the previous journeys are all on same exact date.
2. If two of previous journeys are on same date and third one on different date but in same month.
3. If all three journeys are on different dates but in same month.
4. If all three journeys are on different dates but two of them in same month and third one in different month.
5. If all three journeys are on different dates and in different months.
6. If there are only two journeys i.e. the user skips any year among the three previous years, then in this case with two journeys all above 5 cases will be checked.
7. If there is only one journey in any of the year, well in this case no other condition has to checked.

If user has same date of all the journeys then the predicted date will also be same.

But if any of the date differs from other two or all three dates are different, then the date of most recent journey will be set as predicted date for next year. Also in this case of different date, if dates are nearby i.e. with a difference of  $\pm 3$  are considered.

For example: if we have Jan 2,2013 and Jan 4, 2014, these two dates will be considered as a valid case since they have a difference of 2 days and the same goes for cases like Jan 31,2013 and Feb 1,2014.

But if, we have something like Jan 4,2013 and Jan 8,2014, this cases will not be considered as it has a difference of 4 days.

### Calculation of probability

There are three previous years 2013, 2013, 2014 and the next year for which journey has to be predicted is 2015. Let the previous years be A1, A2, A3. And predicted year be B.

The user will either go In previous years or not, so the probability of journeys in previous is either 0 or 1 i.e.  $P(A1) = 0$  or 1 and same for other two.

Now considering conditional probability that if user has gone in 2012 and will go in 2015 has a probability of 0.15 and for year 2013 and 2015 has 0.35 and for the year 2014 and 2015 it is 0.45.

Means,  $P(B/A1) = 0.15$

$$P(B/A2) = 0.35$$

$$P(B/A3) = 0.45$$

Now, by law of total probability and Bayes theorem,

$$P(B) = P(B/A1)*P(A1) + P(B/A2)*P(A2) + P(B/A3)*P(A3)$$

Suppose we have  $P(A1)=1$ ,  $P(A2)=0$ ,  $P(A3)=1$  then  $P(B)$ ,

$$P(B) = 0.15*1+0.35*0+0.45*1$$

$$P(B) = 0.6.$$



After calculating the probability of each user, the probability is checked with threshold probability and if calculated probability is greater, an alert email is sent to the user..

### **3.3 UML Diagrams**

#### **3.3.1 Use Case**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.



Figure 6: Use case

#### **3.3.2 Sequence diagram**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

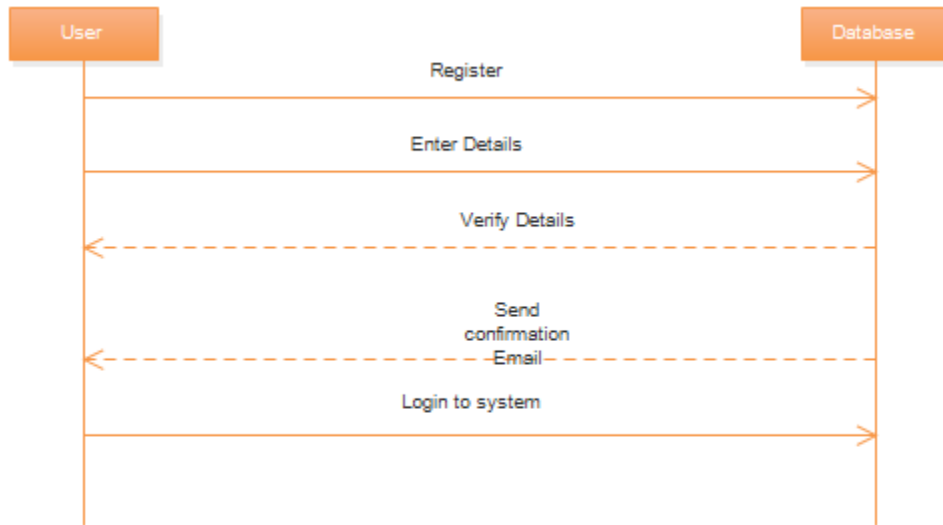


Figure 7: Sequence Diagram for registration



Figure 8: Sequence Diagram for Booking Ticket

### 3.3.3 Data Flow diagram

The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

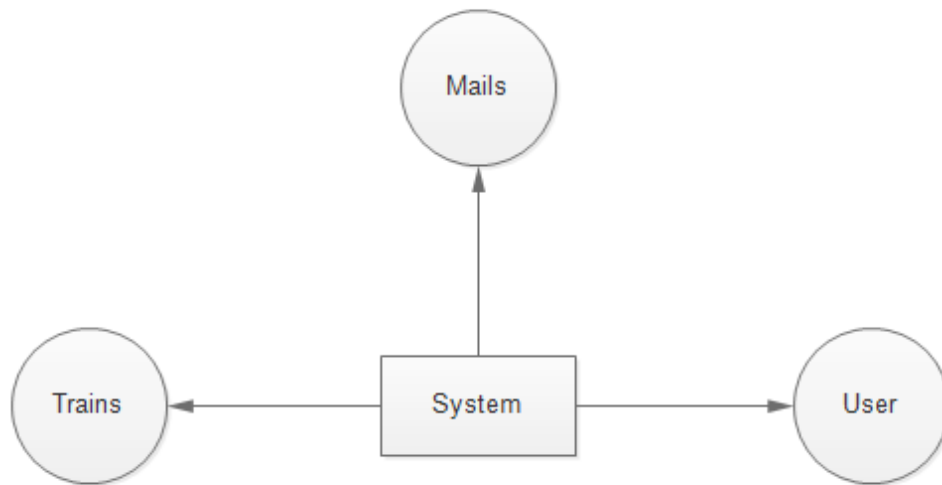


Figure 9: DFD level 0

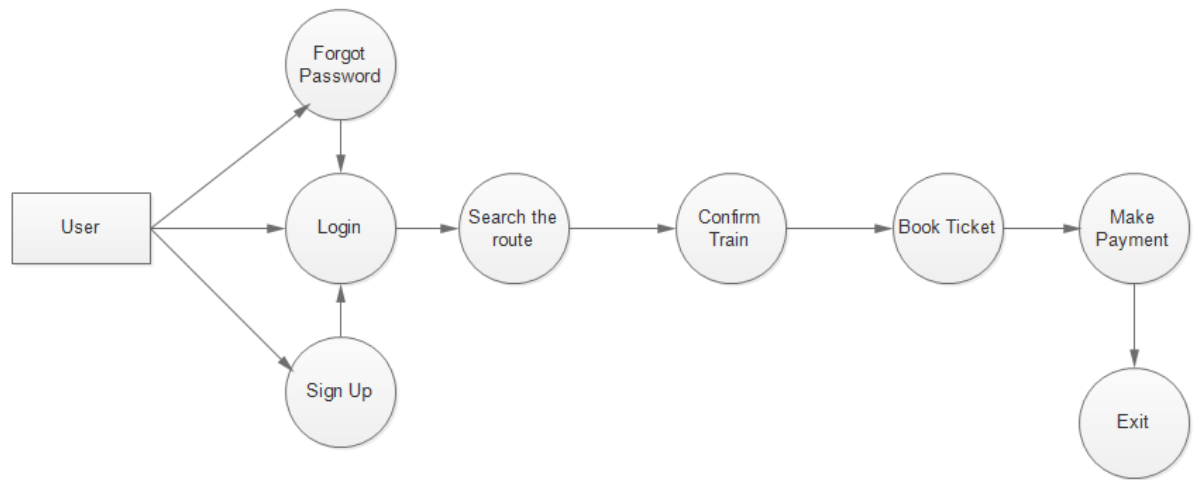


Figure 10: DFD level 1

## 3.4 Code Snippet

### No direct Module

```
try
{
    String source=request.getParameter("source");
    String dest=request.getParameter("dest");
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con=DriverManager.getConnection("jdbc:odbc:log1");
    Statement stmt=con.createStatement();
    ResultSet rs = null;
    //getting station names using station codes
    q="select * from station_code where station_code='"+source+"'";
    q1="select * from station_code where station_code='"+dest+"'";
    rs=stmt.executeQuery(q);
    while(rs.next())
        sname=rs.getString("station_name");
    rs=stmt.executeQuery(q1);
    while(rs.next())
        dname=rs.getString("station_name");
    //Getting lat and long of station name
    q="select * from lat_long where station_name='"+sname+"'";
    q1="select * from lat_long where station_name='"+dname+"'";
    rs=stmt.executeQuery(q);
    while(rs.next()) {
        slat=rs.getString("lat");
        slong=rs.getString("long");
    }
    rs=stmt.executeQuery(q1);
    while(rs.next()) {
```

```

    dlat=rs.getString("lat");
    dlong=rs.getString("long");
}
//converting lat and long to double array
s=calculate(slat,slong,out);
d=calculate(dlat,dlong,out);
//Getting nearby lat and long points
i=1;
while(i<=1635) {
    q="select * from lat_long where id="+i;
    i++;
    rs=stmt.executeQuery(q);
    while(rs.next()) {
        tname=rs.getString("station_name");
        tlat=rs.getString("lat");
        tlong=rs.getString("long");
        t=calculate(tlat,tlong,out);
        if((s[0]<=t[0]&&t[0]<=d[0])&&(s[1]<=t[1]&&t[1]<=d[1]))
            tlist[j++]=tname;
        else if((s[0]<=t[0]&&t[0]<=d[0])&&(s[1]>=t[1]&&t[1]>=d[1]))
            tlist[j++]=tname;
        else if((s[0]>=t[0]&&t[0]>=d[0])&&(s[1]>=t[1]&&t[1]>=d[1]))
            tlist[j++]=tname;
        else if((s[0]>=t[0]&&t[0]>=d[0])&&(s[1]<=t[1]&&t[1]<=d[1]))
            tlist[j++]=tname;
    }
}
//getting train codes from train name
for(i=0;i<j;i++) {
    q="select * from station_code where station_name='"+tlist[i]+'";

```

```

rs=stmt.executeQuery(q);
while(rs.next())
    tcodelist[i]=rs.getString("station_code");
}
//getting the trains
flag=0;
for(i=0;i<j;i++) {
    m=flag=k=0;
    while(m<50) {
        m++;
        n=0;
        while(n<50) {
            n++;
            if(n>m) {
                q="select      train_no      from      train_route      where
station"+m+"=""+source+"and station"+n+"=""+tcodelist[i]+""";
                rs=stmt.executeQuery(q);
                while(rs.next()) {
                    list[k++]=rs.getString(1);
                    flag=1;          //checks if direct train is available or not.
                    index=i;
                } } } }
            if(flag==1) {
                m=0;
                while(m<50) {
                    m++;
                    n=0;
                    while(n<50) {
                        n++;
                        if(n>m) {

```

```

        q="select      train_no      from      train_route      where
station"+m+"="+"+todelist[i]+"and station"+n+"="+"+dest+""";
        rs=stmt.executeQuery(q);
        while(rs.next()) {
            list1[k1++]=rs.getString(1);
            flag=2;          //checks if direct train is available or not.
        } } } } }
        if(flag==2)
            break;
    }
}
catch(Exception e) {
    out.print(e);
}
}
double[] calculate(String a,String b, PrintWriter out)
{
    double []s=new double[2];
    char[]c = a.toCharArray();
    char[]d = b.toCharArray();
    s[0]=((double)c[0]-48)*10+((double)c[1]-48)+((double)c[3]-48)/10+((double)c[4]-
48)/100;
    s[1]=((double)d[0]-48)*10+((double)d[1]-48)+((double)d[3]-48)/10+((double)d[4]-
48)/100;
    return s;
}
}

```

Explanation:



This module is the heart of this system. This module finds out the transit station or an intermediate station where we can change our train to reach out our destination. In this module first the input from the user are taken and then the latitudes and longitudes are fetched from Db which are in string format and are converted to double format so that they can be easily compared. Next all the latitudes and longitudes of all the cities are fetched one by one and compared to the coordinates of source and destination. If the fetched coordinates lie between the stored one, the fetched one are saved in a list along with the city names. At end of this loop we have a list of cities that could lie on the way to destination from source.

Now all these cities are one by one checked whether they can be served as transit station or not. First the source is checked for having a direct train with these cities, if such a city is found then that city is checked with destination, if trains are between this city and destination then this city will serve as transit station and all the trains from source to transit city and from transit city to destination are stored in different list and the train details are fetched using the method in the servlet Userhome.java. And if there is no train from that city to destination then we look for another city from city list which can have direct from source and the process is repeated until we find a complete route. And the result is shown to the user. The last function calculate() is used to convert the lat and long which are saved as strings in Database to double values which are used for comparison.

### Prediction Module

```
public class prediction_new
{
    public static void main(String[] args)
    {
        float p_2015=0.0f;
        float p_15_12=0.15f,p_15_13=0.30f,p_15_14=0.45f; //baye's probability
        float res[]=new float[100];
        int i=0,j=0,k=0,l=0,diff=0,diff1=0,diff2=0;
```

```

int d[]=new int[3];
int m[]=new int[3];
int y[]=new int[3];
int q[]=new int[3];
String str,str1,str2,str3,e_d=null;
String username[]=new String[100];
String doj[]=new String[5];
String src[]=new String[5];
String dst[]=new String[5];
try
{
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con=DriverManager.getConnection("jdbc:odbc:predict_db");
    Statement stmt=con.createStatement();
    ResultSet rs,rs1=null,rs2=null;
    str="select * from user";
    str1="drop table probabilities";
    str2="create table probabilities (user_name varchar(20), prob float, expected_date
varchar(20))";
    stmt.execute(str1);
    stmt.execute(str2);
    rs=stmt.executeQuery(str);
    while(rs.next())
    {
        username[i++]=rs.getString(1);
    }
    // i has total number of users and username has all the names
    while(j<i)
    {
        k=0;

```

```

str="select * from trips_total where user_name='"+username[j]+'";
rs=stmt.executeQuery(str);
while(rs.next())
{
    doj[k]=rs.getString(3); //in m/d/yyyy
    src[k]=rs.getString(4);
    dst[k]=rs.getString(5);
    System.out.println(username[j]+" "+doj[k]+" "+src[k]+" "+dst[k]);
    q=change(doj[k]);
    m[k]=q[0];
    d[k]=q[1];
    y[k]=q[2];
    //System.out.println(username[j]+" "+d[k]+"/"+m[k]+"/"+y[k]+" "+src[k]+"
"+dst[k]);
    k++;
}
//rs.close();
while(true)
{
    p_2015=0.0f;
    e_d=doj[k-1];
    if(k==1)
    {
        if(y[k-1]==2014)
            p_2015=p_15_14;
        else if(y[k-1]==2013)
            p_2015=p_15_13;
        else if(y[k-1]==2012)
            p_2015=p_15_12;
        break;
    }
}

```

```

}
else if(k==2)
{
if(y[k-1]==2014&&y[k-2]==2013)
{
if((m[k-1]==m[k-2])&&(d[k-1]==d[k-2])) //both same
{
p_2015=p_15_14+p_15_13;
}
else if((m[k-1]==m[k-2])&&(d[k-1]!=d[k-2])) //same month different
dates
{
diff=(int)abs((double)(d[k-1]-d[k-2]));
if(diff<=3)
p_2015=p_15_14+p_15_13;
else
p_2015=0.0f;
}
else if((m[k-1]!=m[k-2])&&(d[k-1]!=d[k-2])) //both different
{
if((int)abs((double)(m[k-1]-m[k-2]))==1)
{
diff=(int)abs((double)(d[k-1]-d[k-2]));
if(m[k-1]==2&&m[k-2]==3)
if(diff>=25&&diff<=27)
p_2015=p_15_14+p_15_13;
else
p_2015=0.0f;
}
else
if(diff>=28&&diff<=30)

```

```

        p_2015=p_15_14+p_15_13;
    else
        p_2015=0.0f;
    }
}
break;
}
else if(y[k-1]==2014&&y[k-2]==2012)
{
    if((m[k-1]==m[k-2])&&(d[k-1]==d[k-2])) //both same
    {
        p_2015=p_15_14+p_15_12;
    }
    else if((m[k-1]==m[k-2])&&(d[k-1]!=d[k-2])) //same month different
    {
        diff=(int)abs((double)(d[k-1]-d[k-2]));
        if(diff<=3)
            p_2015=p_15_14+p_15_12;
        else
            p_2015=0.0f;
    }
    else if((m[k-1]!=m[k-2])&&(d[k-1]!=d[k-2])) //both different
    {
        if((int)abs((double)(m[k-1]-m[k-2]))==1)
        {
            diff=(int)abs((double)(d[k-1]-d[k-2]));
            if((m[k-1]==2&& m[k-2]==3)||(m[k-2]==2&& m[k-1]==3))
                if(diff>=26&& diff<=28)
                    p_2015=p_15_14+p_15_12;
        }
    }
}

```

dates

```

        else
            p_2015=0.0f;
        else
            if(diff>=28&&diff<=30)
                p_2015=p_15_14+p_15_12;
            else
                p_2015=0.0f;
        }
    }
    break;
}
else if(y[k-1]==2013&&y[k-2]==2012)
{
    if((m[k-1]==m[k-2])&&(d[k-1]==d[k-2])) //both same
    {
        p_2015=p_15_13+p_15_12;
    }
    else if((m[k-1]==m[k-2])&&(d[k-1]!=d[k-2])) //same month different
dates
    {
        diff=(int)abs((double)(d[k-1]-d[k-2]));
        if(diff<=3)
            p_2015=p_15_13+p_15_12;
        else
            p_2015=0.0f;
    }
    else if((m[k-1]!=m[k-2])&&(d[k-1]!=d[k-2])) //both different
    {
        if((int)abs((double)(m[k-1]-m[k-2]))==1)
        {

```

```

diff=(int)abs((double)(d[k-1]-d[k-2]));
if((m[k-1]==2&& m[k-2]==3)||(m[k-2]==2&& m[k-1]==3))
    if(diff>=26&& diff<=28)
        p_2015=p_15_13+p_15_12;
    else
        p_2015=0.0f;
else
    if(diff>=28&& diff<=30)
        p_2015=p_15_13+p_15_12;
    else
        p_2015=0.0f;
}
else
{
    p_2015=0.0f;
    break;
}
}
break;
}
break;
}
else if(k==3) //user making trips in all 3 years
{
    if((m[k-1]==m[k-2])&&(m[k-2]==m[k-3])) //all three in same month
    {
        if((d[k-1]==d[k-2])&&(d[k-2]==d[k-3])) // all three on same dates
        {
            p_2015=p_15_14+p_15_13+p_15_12;
            break;
        }
    }
}

```

```

    }
    else if((d[k-1]==d[k-2])||(d[k-2]==d[k-3])||(d[k-3]==d[k-1])) //same
month and 2 on same date, 3rd diff date
    {
        p_2015=p_15_14+p_15_13+p_15_12;
        break;
    }
    else // same months but all different dates
    {
        diff=(int)abs((double)(d[k-1]-d[k-2]));
        diff1=(int)abs((double)(d[k-3]-d[k-2]));
        diff2=(int)abs((double)(d[k-1]-d[k-3]));

if((diff<=3&&diff1<=3)||(diff2<=3&&diff1<=3)||(diff<=3&&diff2<=3))
    {
        p_2015=p_15_14+p_15_13+p_15_12;
        break;
    }
    else
    {
        p_2015=0.0f;
        break;
    }
    }
    }
    else if((m[k-1]==m[k-2])||(m[k-2]==m[k-3])||(m[k-3]==m[k-1])) //two in
same month and third in different
    {
and third diff)
        if((d[k-1]==d[k-2])&&(d[k-2]==d[k-3])) // all same date (two same trips
        {

```



```

        p_2015=p_15_14+p_15_13+p_15_12;
        break;
    }
    else if(((int)abs((double)(m[k-1]-m[k-2]))==1)||((int)abs((double)(m[k-3]-m[k-2]))==1)||((int)abs((double)(m[k-1]-m[k-3]))==1))
    {
        if((d[k-1]==d[k-2])||(d[k-2]==d[k-3])||(d[k-3]==d[k-1])) //2 on same
date, 3rd diff date
        {
            diff=(int)abs((double)(d[k-1]-d[k-2]));
            diff1=(int)abs((double)(d[k-3]-d[k-2]));
            diff2=(int)abs((double)(d[k-1]-d[k-3]));
            if(((m[k-1]==2&& m[k-2]==3)|| (m[k-2]==2&& m[k-1]==3))||((m[k-3]==2&& m[k-2]==3)|| (m[k-2]==2&& m[k-3]==3))||((m[k-1]==2&& m[k-3]==3)|| (m[k-3]==2&& m[k-1]==3)))
            {
                if(((diff==0)&&(diff1==diff2)&&(diff2>=26&&diff2<=28))||((diff1==0)&&(diff==diff2)&&(diff>=26&&diff<=28))||((diff2==0)&&(diff1==diff)&&(diff1>=26&&diff1<=28)))
                {
                    p_2015=p_15_14+p_15_13+p_15_12;
                    break;
                }
            }
        }
    }
    else
    {
        p_2015=0.0f;
        break;
    }
}
else
{

```

```

if(((diff==0)&&(diff1==diff2)&&(diff2>=28&&diff2<=30))||((diff1==0)&&(diff==diff2)
)&&(diff>=28&&diff<=30))||((diff2==0)&&(diff1==diff)&&(diff1>=28&&diff1<=30)))
    {
        p_2015=p_15_14+p_15_13+p_15_12;
        break;
    }
else
    {
        p_2015=0.0f;
        break;
    }
}
}
else
{
    diff=(int)abs((double)(d[k-1]-d[k-2]));
    diff1=(int)abs((double)(d[k-3]-d[k-2]));
    diff2=(int)abs((double)(d[k-1]-d[k-3]));

if((diff<=3&&diff1<=3)||((diff2<=3&&diff1<=3)||((diff<=3&&diff2<=3)))
    {
        p_2015=p_15_14+p_15_13+p_15_12;
        break;
    }
    else        if(((m[k-1]==2&&m[k-2]==3)||((m[k-2]==2&&m[k-1]==3))||((m[k-3]==2&&m[k-2]==3)||((m[k-2]==2&&m[k-3]==3))||((m[k-1]==2&&m[k-3]==3)||((m[k-3]==2&&m[k-1]==3))))
    {

if((diff2>=26&&diff2<=28&&diff>=26&&diff<=28)||((diff1>=26&&diff1<=28&&diff2
>=26&&diff2<=28)||((diff1>=26&&diff1<=28&&diff>=26&&diff<=28))

```

```

        {
            p_2015=p_15_14+p_15_13+p_15_12;
            break;
        }
    else
    {
        p_2015=0.0f;
        break;
    }
}
else
if((diff2>=28&&diff2<=30&&diff>=28&&diff<=30)||((diff1>=28&&diff1<=30&&diff2
>=28&&diff2<=30)||((diff1>=28&&diff1<=30&&diff>=28&&diff<=30))
{
    p_2015=p_15_14+p_15_13+p_15_12;
    break;
}
else
if((diff2>=28&&diff2<=30&&diff<=3)||((diff1>=28&&diff1<=30&&diff2<=3)||((diff1>=
28&&diff1<=30&&diff<=3))
{
    p_2015=p_15_14+p_15_13+p_15_12;
    break;
}
else
{
    p_2015=0.0f;
    break;
}
}
}
}

```

```

        }
        else //all three in different months
        {
            p_2015=0.0f;
            break;
        }
    }
}
if(p_2015==0.0)
    e_d=null;
if(e_d!=null)
{
    int len=e_d.length();
    char c[]=e_d.toCharArray();
    c[len-1]=(char)53;
    e_d=new String(c);
}
if(e_d!=null)
    System.out.println("The probability of "+username[j]+" of making a trip in
2015 is "+p_2015+" on "+e_d);
else
    System.out.println("The probability of "+username[j]+" of making a trip in
2015 is "+p_2015);
    str3="Insert into probabilities (user_name,prob,expected_date) values
("+username[j]+"",""+p_2015+"",""+e_d+"");
    stmt.execute(str3);
    j++;
}
str="select probabilities.user_name, probabilities.prob,
probabilities.expected_date, user.emailid from user inner join probabilities on
user.user_name = probabilities.user_name";

```

```

rs2=stmt.executeQuery(str);
//sending emails to the predicted users corresponding to system date
String u,da,ei;
float prob;
while(rs2.next())
{
    u=rs2.getString(1);
    prob=rs2.getFloat(2);
    da=rs2.getString(3);
    ei=rs2.getString(4);
    if(prob>=0.6)
    {
        //send an alert email
        final String uname = "uihand10@gmail.com";
        final String password = "continent";
        String Receiver=ei;
        //System.out.println(Receiver);
        /*Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.ssl.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "465");
        props.put("mail.smtp.user", "username");
        props.put("mail.smtp.password", "password");
        Session session;
        session = Session.getDefaultInstance(props,new javax.mail.Authenticator()
        {
            @Override
            protected PasswordAuthentication getPasswordAuthentication()
            {

```



```
else
{
    d[1]*=10;
    (d[1])+=((int)c[3]-48);
}
if(d[2]==0)
    d[2]=(((int)c[q-4]-48)*1000)+(((int)c[q-3]-48)*100)+(((int)c[q-2]-
48)*10)+((int)c[q-1]-48);
return d;
}
}
```

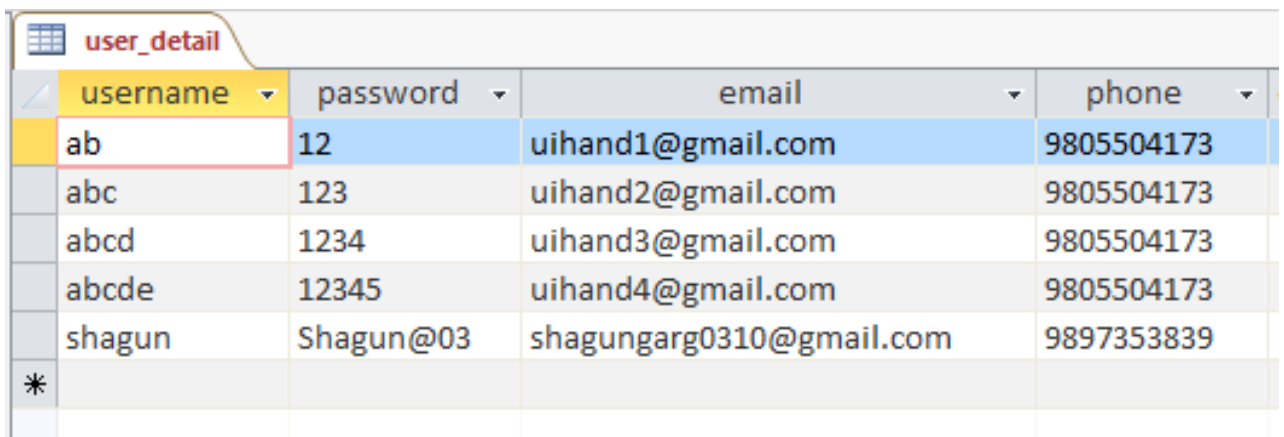
### 3.5 Database Design

There are five tables used in first system and three tables in second system whose specifications are as follows:

**Table 2: User detail**

<u>Field Name</u>	<u>Data type</u>
username	Short text
password	Short text
email	Short text
phone	Short text

Primary key: username



username	password	email	phone
ab	12	uihand1@gmail.com	9805504173
abc	123	uihand2@gmail.com	9805504173
abcd	1234	uihand3@gmail.com	9805504173
abcde	12345	uihand4@gmail.com	9805504173
shagun	Shagun@03	shagungarg0310@gmail.com	9897353839
*			

Figure 11: Database User\_detail



**Table 3: train\_detail**

<u>Field name</u>	<u>Data Type</u>
train_no	Short text
train_name	Short text
src	Short text
src_time	Short text
dst	Short text
dst_time	Short text

Primary key: train\_no

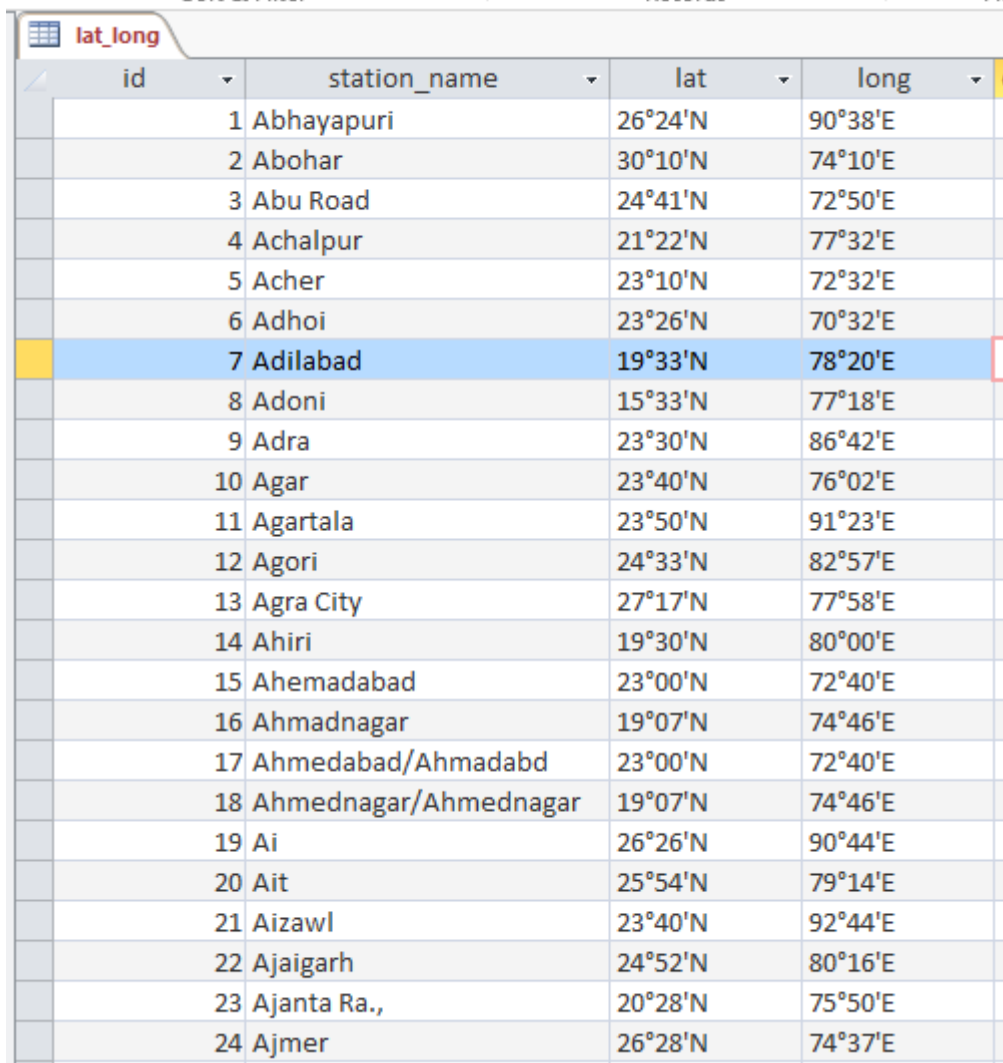
train_no	train_name	src	src_time	dst	dst_time
00651	MAS KCVL PREI	CHENNAI CENT	4:30:00 PM	KOCHUVELI	6:20:00 AM
00652	KCVL MAS PREI	KOCHUVELI	9:30:00 PM	CHENNAI CENTRAL	11:40:00 AM
00653	MAS KCVL PREI	CHENNAI CENT	10:30:00 PM	KOCHUVELI	12:45:00 PM
00655	MAS KCVL PREI	CHENNAI CENT	10:30:00 PM	KOCHUVELI	12:45:00 PM
00851	BNC PREMIUM	BHUBANESWA	10:50:00 PM	BANGALORE CANT	10:40:00 PM
00852	BNC BBS PREM	BANGALORE CA	1:00:00 AM	BHUBANESWAR	1:45:00 AM
01081	LTT GKP SPECIA	LOKMANYATIL	3:50:00 PM	GORAKHPUR JN	2:15:00 AM
01082	GKP LTT SPECIA	GORAKHPUR JN	1:00:00 PM	LOKMANYATILAK T	10:35:00 PM
01451	PUNE LJN SPEC	PUNE JN	10:00:00 PM	LUCKNOW NE	2:30:00 AM
01452	LJN PUNE SPL	LUCKNOW NE	6:30:00 AM	PUNE JN	11:05:00 AM
01651	PUNE HBJ SPL	PUNE JN	2:15:00 PM	HABIBGANJ	5:30:00 AM
01652	HBJ PUNE SPEC	HABIBGANJ	5:25:00 PM	PUNE JN	9:20:00 AM
01655	PUNE JBP SUP S	PUNE JN	11:15:00 AM	JABALPUR	6:00:00 AM
01656	JBP PUNE SUP S	JABALPUR	4:55:00 PM	PUNE JN	11:05:00 AM
02111	LTT LKO AC SPL	LOKMANYATIL	2:20:00 PM	LUCKNOW NR	1:35:00 PM
02112	LKO LTT AC SPL	LUCKNOW NR	4:15:00 PM	LOKMANYATILAK T	5:30:00 PM
02203	SC PREMIUM E	VISHAKAPATN	9:10:00 PM	SECUNDERABAD JN	7:00:00 AM
02204	VSKP PREMIUM	SECUNDERABA	11:00:00 PM	VISHAKAPATNAM	9:05:00 AM
02209	VSKP PREMIUM	SECUNDERABA	11:00:00 PM	VISHAKAPATNAM	9:05:00 AM
02210	SC PREMIUM E	VISHAKAPATN	9:10:00 PM	SECUNDERABAD JN	7:00:00 AM
02409	PUNE UDN AC S	PUNE JN	5:15:00 AM	UDHNA JN	12:15:00 PM
02410	UDN PUNE AC S	UDHNA JN	1:50:00 PM	PUNE JN	10:15:00 PM
02530	UHP GKP PREM	UDHAMPUR	11:45:00 PM	GORAKHPUR JN	11:20:00 PM
02657	BNC TVC PREM	BANGALORE CA	7:15:00 PM	TRIVANDRUM CNTL	8:30:00 AM
02695	YPR JP PREMIU	YESVANTPUR J	11:30:00 AM	JAIPUR	6:35:00 AM
02696	JP YPR PREMIU	JAIPUR	10:15:00 PM	YESVANTPUR JN	5:55:00 PM

**Figure 12: Database train\_detail**

**Table 4: lat long**

<u>Field Name</u>	<u>Data type</u>
id	Auto number
station_name	Short text
lat	Short text
long	Short text

Primary key: id



The screenshot shows a database table named 'lat\_long' with the following data:

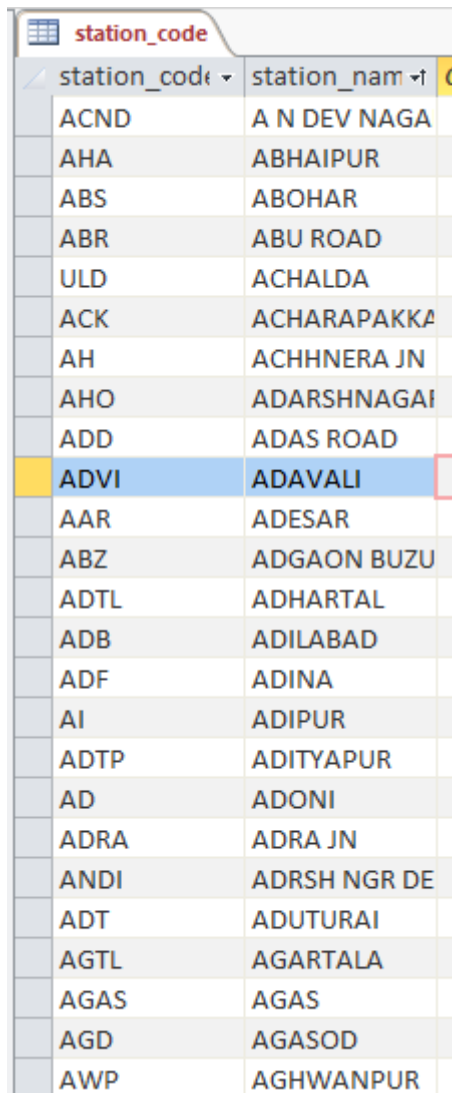
id	station_name	lat	long
1	Abhayapuri	26°24'N	90°38'E
2	Abohar	30°10'N	74°10'E
3	Abu Road	24°41'N	72°50'E
4	Achalpur	21°22'N	77°32'E
5	Acher	23°10'N	72°32'E
6	Adhoi	23°26'N	70°32'E
7	Adilabad	19°33'N	78°20'E
8	Adoni	15°33'N	77°18'E
9	Adra	23°30'N	86°42'E
10	Agar	23°40'N	76°02'E
11	Agartala	23°50'N	91°23'E
12	Agori	24°33'N	82°57'E
13	Agra City	27°17'N	77°58'E
14	Ahiri	19°30'N	80°00'E
15	Ahemadabad	23°00'N	72°40'E
16	Ahmadnagar	19°07'N	74°46'E
17	Ahmedabad/Ahmadabd	23°00'N	72°40'E
18	Ahmednagar/Ahmednagar	19°07'N	74°46'E
19	Ai	26°26'N	90°44'E
20	Ait	25°54'N	79°14'E
21	Aizawl	23°40'N	92°44'E
22	Ajaigarh	24°52'N	80°16'E
23	Ajanta Ra.,	20°28'N	75°50'E
24	Ajmer	26°28'N	74°37'E

Figure 13: Database lat\_long

**Table 5: station\_code**

<u>Field Name</u>	<u>Data type</u>
station_code	Short text
station_name	Short text

Primary key: station\_code



station_code	station_name
ACND	A N DEV NAGA
AHA	ABHAIPUR
ABS	ABOHAR
ABR	ABU ROAD
ULD	ACHALDA
ACK	ACHARAPAKKA
AH	ACHHNERA JN
AHO	ADARSHNAGAI
ADD	ADAS ROAD
ADVI	ADAVALI
AAR	ADESAR
ABZ	ADGAON BUZU
ADTL	ADHARTAL
ADB	ADILABAD
ADF	ADINA
AI	ADIPUR
ADTP	ADITYAPUR
AD	ADONI
ADRA	ADRA JN
ANDI	ADRSH NGR DE
ADT	ADUTURAI
AGTL	AGARTALA
AGAS	AGAS
AGD	AGASOD
AWP	AGHWANPUR

Figure 14: Database station\_code

**Table 6: train route**

<u>Field Name</u>	<u>Data type</u>
train_no	Short text
station1	Short text
station2	Short text
:	:
:	:
Station120	Short text

Primary key: train\_no

train_no	station1	station2	station3	station4	station5	station6	station7	station8	station9	station10	station11	station12	station13
12006	KLK	CDG	UMB	KKDE	NDLS	Source	6:45:00 AM	7:33:00 AM	8:10:00 AM	10:25:00 AM	6:15:00 AM	6:53:00 AM	7:38:00 AM
12012	KLK	CDG	UMB	KKDE	PNP	NDLS	Source	6:15:00 PM	7:03:00 PM	7:38:00 PM	8:30:00 PM	9:55:00 PM	5:45:00 PM
12046	CDG	UMB	NDLS	Source	12:40:00 PM	3:20:00 PM	12:00:00 PM	12:42:00 PM	Destination				
12056	DDN	HW	RK	TPZ	DBD	MOZ	MTC	GZB	TKJ	NDLS	Source	6:18:00 AM	7:00:00 AM
12058	UHL	NLDM	ANSB	KART	RPAR	SASN	CDG	UMB	KKDE	KUN	PNP	SNP	SZM
12232	CDG	UMB	JUDW	JUD	SRE	RK	LRJ	NBD	NGG	DPR	MB	BE	SPN
12288	DDN	HW	RK	DBD	MOZ	MTC	GZB	NZM	KOTA	BRC	ST	BSR	PNVL
12312	KLK	CNDM	CDG	UMB	KKDE	KUN	PNP	SMK	GNU	SNP	ANDI	SZM	DLI
12450	CDG	UMB	PNP	NDLS	NZM	KOTA	BRC	BSR	PNVL	RN	PERN	THVM	KRMI
14096	KLK	CNDM	CDG	DKT	UMB	SHDM	KKDE	NLKR	KUN	GRA	PNP	SMK	GNU
14218	CDG	UMB	SHDM	KKDE	KUN	PNP	SMK	GNU	SNP	NUR	SZM	DLI	SBB
14310	DDN	HW	RK	SRE	DBD	MOZ	MTC	GZB	NZM	FDB	KSV	MTJ	RKM
14318	DDN	HW	RK	SRE	DBD	MOZ	MTC	MDNR	GZB	NZM	FDB	MTJ	RKM
14512	SRE	DBD	MOZ	KAT	MUT	MTC	HPU	GMS	GJL	AMRO	MB	RMU	BE
14646	JAT	SMBX	GHGL	KTHU	PTKC	MEX	DZA	JRC	PGW	LDH	KNN	SIR	RPJ
14682	JUC	PGW	PHR	LDH	DOA	KNN	SIR	RPJ	UBC	UMB	RAA	JUDW	JUD
14887	KLK	CNDM	CDG	DHPR	UMB	UBC	RPJ	PTA	NBA	DUI	BNN	TAPA	PUL
15012	CDG	UMB	JUDW	JUD	SRE	RK	LRJ	MZM	BJO	HLDR	CPS	MNDR	GJL

**Figure 15: Database train route**

**Table 7: user**

<u>Field Name</u>	<u>Data type</u>
User_name	Short text
emailid	Short text
phone	Short text
password	Short text

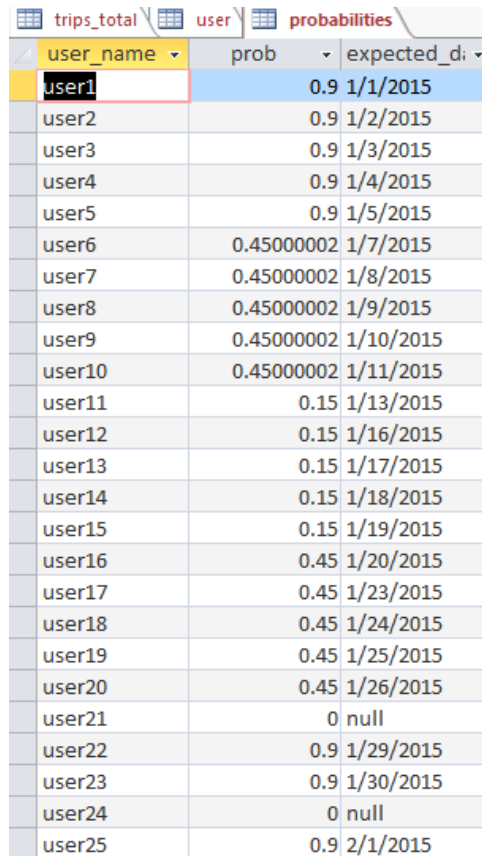
user_name	emailid	phone	password	Click to A
user1	uihand1@gmail.com	9805504173	abcde	
user10	uihand1@gmail.com	9805504173	abcde	
user11	uihand1@gmail.com	9805504173	abcde	
user12	uihand1@gmail.com	9805504173	abcde	
user13	uihand1@gmail.com	9805504173	abcde	
user14	uihand1@gmail.com	9805504173	abcde	
user15	uihand1@gmail.com	9805504173	abcde	
user16	uihand1@gmail.com	9805504173	abcde	
user17	uihand1@gmail.com	9805504173	abcde	
user18	uihand1@gmail.com	9805504173	abcde	
user19	uihand1@gmail.com	9805504173	abcde	
user2	uihand1@gmail.com	9805504173	abcde	
user20	uihand1@gmail.com	9805504173	abcde	
user21	uihand1@gmail.com	9805504173	abcde	
user22	uihand1@gmail.com	9805504173	abcde	
user23	uihand1@gmail.com	9805504173	abcde	
user24	uihand1@gmail.com	9805504173	abcde	
user25	uihand1@gmail.com	9805504173	abcde	
user26	uihand1@gmail.com	9805504173	abcde	
user27	uihand1@gmail.com	9805504173	abcde	
user28	uihand1@gmail.com	9805504173	abcde	
user29	uihand1@gmail.com	9805504173	abcde	
user3	uihand1@gmail.com	9805504173	abcde	
user30	uihand1@gmail.com	9805504173	abcde	
user31	uihand1@gmail.com	9805504173	abcde	
user32	uihand1@gmail.com	9805504173	abcde	

Record: 5 of 50 | No Filter | Search

**Figure 16: Database userdetails**

**Table 8: probabilities**

<u>Field Name</u>	<u>Data type</u>
User_name	Short text
prob	Float
Expected_date	Short text



The screenshot shows a database table with the following data:

user_name	prob	expected_date
user1	0.9	1/1/2015
user2	0.9	1/2/2015
user3	0.9	1/3/2015
user4	0.9	1/4/2015
user5	0.9	1/5/2015
user6	0.45000002	1/7/2015
user7	0.45000002	1/8/2015
user8	0.45000002	1/9/2015
user9	0.45000002	1/10/2015
user10	0.45000002	1/11/2015
user11	0.15	1/13/2015
user12	0.15	1/16/2015
user13	0.15	1/17/2015
user14	0.15	1/18/2015
user15	0.15	1/19/2015
user16	0.45	1/20/2015
user17	0.45	1/23/2015
user18	0.45	1/24/2015
user19	0.45	1/25/2015
user20	0.45	1/26/2015
user21	0	null
user22	0.9	1/29/2015
user23	0.9	1/30/2015
user24	0	null
user25	0.9	2/1/2015

**Figure 17: Database probabilities**

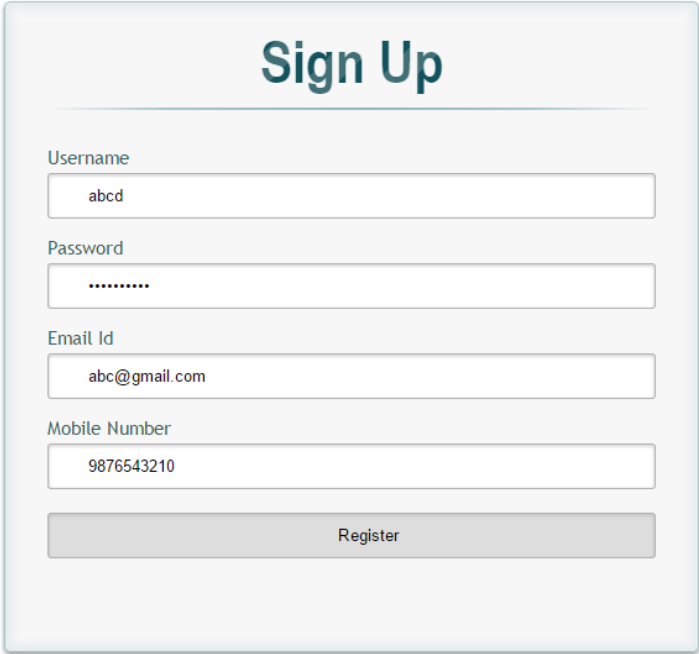
**Table 9: trip details**

<u>Field name</u>	<u>Data Type</u>
id	integer
user_name	Short text
doj	Short text
src	Short text
dst	Short text

ID	user_name	doj	src	dst	Client
1	user1	1/1/2012	Delhi	Mumbai	
2	user2	1/2/2012	Delhi	Mumbai	
3	user3	1/3/2012	Delhi	Mumbai	
4	user4	1/4/2012	Delhi	Mumbai	
5	user5	1/5/2012	Delhi	Mumbai	
6	user6	1/6/2012	Delhi	Mumbai	
7	user7	1/9/2012	Delhi	Mumbai	
8	user8	1/10/2012	Delhi	Mumbai	
9	user9	1/11/2012	Delhi	Mumbai	
10	user10	1/12/2012	Delhi	Mumbai	
11	user11	1/13/2012	Delhi	Mumbai	
12	user12	1/16/2012	Delhi	Mumbai	
13	user13	1/17/2012	Delhi	Mumbai	
14	user14	1/18/2012	Delhi	Mumbai	
15	user15	1/19/2012	Delhi	Mumbai	
16	user21	1/27/2012	Lucknow	Chandigarh	
17	user22	1/30/2012	Lucknow	Chandigarh	
18	user23	1/31/2012	Lucknow	Chandigarh	
19	user24	2/1/2012	Lucknow	Chandigarh	
20	user25	2/2/2012	Lucknow	Chandigarh	
21	user26	2/3/2012	Lucknow	Chandigarh	
22	user27	2/6/2012	Lucknow	Chandigarh	
23	user28	2/7/2012	Lucknow	Chandigarh	
24	user29	2/8/2012	Lucknow	Chandigarh	
25	user30	2/9/2012	Lucknow	Chandigarh	
26	user31	2/10/2012	Lucknow	Chandigarh	

**Figure 18: Database trips\_total**

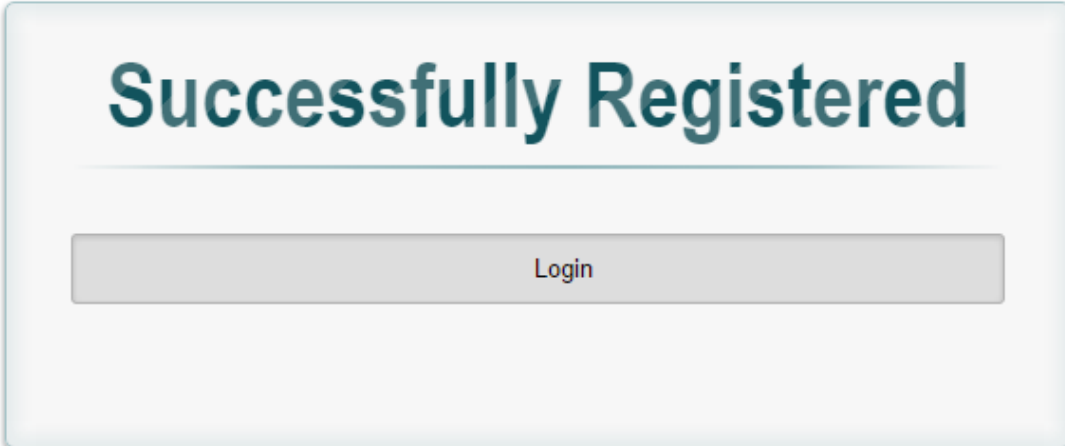
### 3.6 Snapshots



A screenshot of a 'Sign Up' form. The form is titled 'Sign Up' in a large, bold, dark blue font. Below the title, there are four input fields: 'Username' with the value 'abcd', 'Password' with a masked value '.....', 'Email Id' with the value 'abc@gmail.com', and 'Mobile Number' with the value '9876543210'. At the bottom of the form is a grey button labeled 'Register'.

---

Figure 19: Sign Up



A screenshot of a 'Successfully Registered' message. The message is displayed in a large, bold, dark blue font. Below the message is a grey button labeled 'Login'.

---

Figure 20: Successfully Registered



## Log in

---

Username

Your password

[Forgot Password](#)

---

Figure 21: Login

## Forgot Password

---

Enter your registered Email Id

Your new password will sent to your email id

---

Figure 22: Forgot Password

# Welcome

Enter your source station

Enter your destination station

**Search**

Figure 23: User Home (Direct Route Example)

# Trains

CHANDIGARH To SAHARANPUR

Train no	Train name	Departure Time	Arrival Time
15012	CDG LJN EXPRESS	5:05:00 PM	7:35:00 PM
12232	CDG LKO EXPRESS	9:10:00 PM	11:30:00 PM

Figure 24: Trains Output (Direct Route Example)

# Welcome

Enter your source station

Enter your destination station

Figure 25: User Home (No Direct Route Example)

## Trains

CHANDIGARH To AMBALA CANT JN

Train no	Train name	Departure Time	Arrival Time
12450	GOA SMPRK K EXP	1:30:00 AM	2:50:00 AM
12046	CDG NDLS SHTBDI	12:00:00 PM	12:40:00 PM
19718	CDG JP INTERCIT	12:45:00 PM	1:35:00 PM
14218	UNCHAHAH EXP	4:30:00 PM	5:25:00 PM
15012	CDG LJN EXPRESS	5:05:00 PM	6:03:00 PM
12232	CDG LKO EXPRESS	9:10:00 PM	9:50:00 PM
12006	KALKA SHTBDI	6:53:00 AM	7:33:00 AM
22926	PASCHIM EXPRESS	11:20:00 AM	12:05:00 PM
12012	KALKA SHTBDI	6:23:00 PM	7:03:00 PM
12312	KALKA MAIL	1:10:00 AM	2:00:00 AM
14096	HIMALAYAN QUEEN	5:30:00 PM	6:20:00 PM
14887	KLK BME EXPRESS	10:08:00 PM	11:15:00 PM
12058	NDLS JANSHTBDI	7:38:00 AM	8:30:00 AM

AMBALA CANT JN To DEOBAND

Train no	Train name	Departure Time	Arrival Time
19326	ASR INDB EXPRESS	4:10:00 AM	6:10:00 AM
14682	JUC NDLS EXPRES	7:45:00 AM	9:47:00 AM
14646	SHALIMAR EXP	5:00:00 AM	7:06:00 AM

Figure 26: Trains Output (Direct Route Example)

```
Output - prediction (run) ×
The probability of user33 of making a trip in 2015 is 0.15 on 2/14/2015
user34 2/15/2012 Lucknow Chandigarh
The probability of user34 of making a trip in 2015 is 0.15 on 2/15/2015
user35 2/16/2012 Lucknow Chandigarh
The probability of user35 of making a trip in 2015 is 0.15 on 2/16/2015
user36 2/17/2012 Lucknow Chandigarh
user36 2/19/2014 Lucknow Chandigarh
The probability of user36 of making a trip in 2015 is 0.6 on 2/19/2015
user37 2/20/2012 Lucknow Chandigarh
user37 2/20/2014 Lucknow Chandigarh
The probability of user37 of making a trip in 2015 is 0.6 on 2/20/2015
user38 2/21/2012 Lucknow Chandigarh
user38 2/21/2014 Lucknow Chandigarh
The probability of user38 of making a trip in 2015 is 0.6 on 2/21/2015
user39 2/22/2012 Lucknow Chandigarh
user39 2/22/2014 Lucknow Chandigarh
The probability of user39 of making a trip in 2015 is 0.6 on 2/22/2015
user40 2/23/2012 Lucknow Chandigarh
user40 2/23/2014 Lucknow Chandigarh
The probability of user40 of making a trip in 2015 is 0.6 on 2/23/2015
user41 2/24/2012 Ambala Saharanpur
user41 2/23/2013 Ambala Saharanpur
user41 2/24/2014 Ambala Saharanpur
The probability of user41 of making a trip in 2015 is 0.9 on 2/24/2015
user42 2/27/2012 Ambala Saharanpur
user42 2/25/2013 Ambala Saharanpur
user42 2/25/2014 Ambala Saharanpur
The probability of user42 of making a trip in 2015 is 0.9 on 2/25/2015
user43 2/28/2012 Ambala Saharanpur
user43 2/26/2013 Ambala Saharanpur
user43 2/26/2014 Ambala Saharanpur
The probability of user43 of making a trip in 2015 is 0.9 on 2/26/2015
user44 2/29/2012 Ambala Saharanpur
user44 2/27/2013 Ambala Saharanpur
user44 2/27/2014 Ambala Saharanpur
The probability of user44 of making a trip in 2015 is 0.9 on 2/27/2015
user45 3/1/2012 Ambala Saharanpur
user45 2/28/2013 Ambala Saharanpur
```

Figure 27: Prediction Output

## **Chapter 4: Conclusion and Future Work**

The first objective of this project has nearly come to an End. The system now shows the trains between the directly connected stations and also between the stations that are not directly connected by finding an intermediate station where passenger can change the train. This shows all the trains between the source to intermediate station and all trains from that intermediate station to destination.

The second objective is almost completed that is the journeys are predicted based on the previous three years journeys and an alert mail is being sent to the selected users.

In future I would like add some graphical representation of the route in first objective. In second objective, I will like to add some more refinements to the cases used for prediction like weekends situation.

## References

1. Kenneth Gade, A Non-singular Horizontal Position Representation, THE JOURNAL OF NAVIGATION (2010), 63, 395–417.
2. Dane e. Ericksen, p.e., cstre, distance and bearing calculations, hammett & edison, inc., consulting engineers, san francisco, california.
3. ANSI/IEEE Standard 268-1982, Metric Practice, Page 31, Note 14.
4. FCC Public Notice “FCC Interim Procedure for the Specification of Geographic Coordinates,” March 14, 1988.
5. Britting, K.R, Inertial Navigation Systems Analysis. Wiley Interscience. 1971.
6. A. Aamodt, E. Plaza (1994); Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, AI Communications, IOS Press Vol 7: 1 pp 39-59..
7. Piero P. Bonissone, Ramon Lopez de Mantaras, Fuzzy Case-Based Reasoning System.
8. Tomas Olsson, Daniel Gillblad, Peter Funk, and Ning Xiong: Case-Based Reasoning for Explaining Probabilistic Machine Learning, School of Innovation, Design, and Engineering, Mälardalen University, Västerås, Sweden SICS Swedish ICT.
9. Inza, Iñaki, et al. "Feature subset selection by Bayesian network-based optimization." Artificial intelligence 123.1 (2000): 157-184.
10. Zhang, Nevin L., and David Poole. "A simple approach to Bayesian network computations." Proc. of the Tenth Canadian Conference on Artificial Intelligence. 1994.
11. Schiaffino, Silvia N., and Analía Amandi. "User profiling with Case-Based Reasoning and Bayesian Networks." IBERAMIA-SBIA 2000 open discussion track. 2000.
12. Langley, Pat, and Stephanie Sage. "Induction of selective Bayesian classifiers." Proceedings of the Tenth international conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1994.