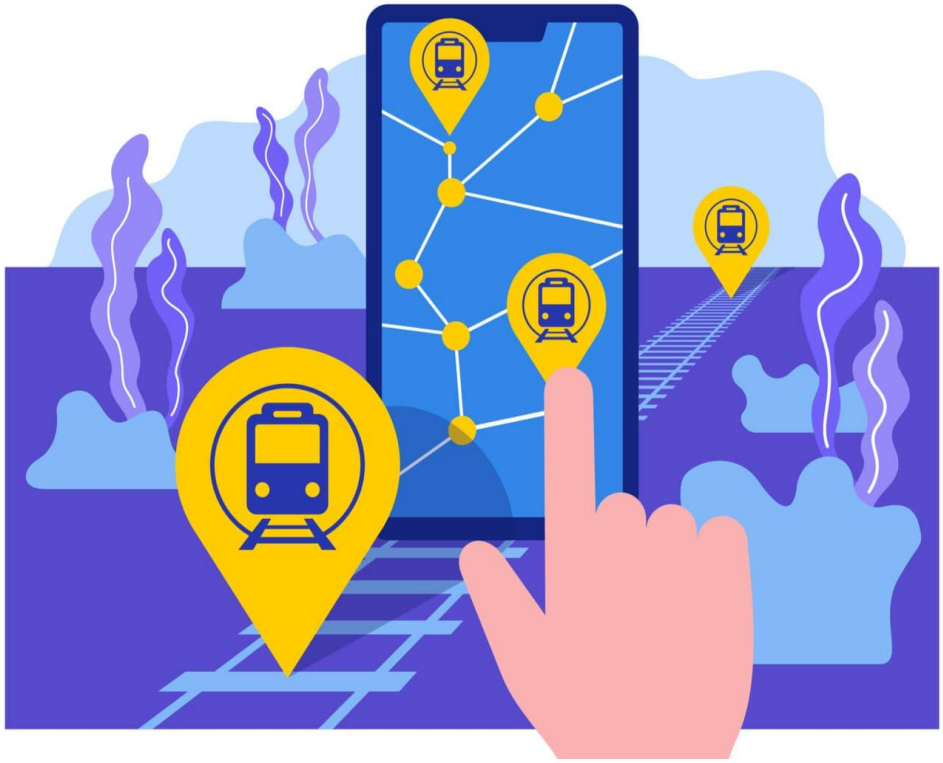
CS301: Database Systems

Railway Reservation Application



- SEARCH FOR TRAINS
- BOOK TICKETS
- SEARCH ALTERNATIVE ROUTES
- ADMIN FUNCTIONALITY
- BOOKING HISTORY

Course Project - Report

Submitted By : TEAM

Akshat Goel -- 2018CSB1067 -- CSE'18 Batch

Aman Bilaiya -- 2018CSB1069 -- CSE'18 Batch

Prateek Saini -- 2018CSB1114 -- CSE'18 Batch

Submitted To : Dr. Viswanath Gunturi Sir

Schema Diagram and Implementation details of the project

::TECHNOLOGY STACK USED::

Languages Used - PHP, HTML, CSS, Javascript and Bootstrap

Database Used - MySQL

(a) TABLE DETAILS

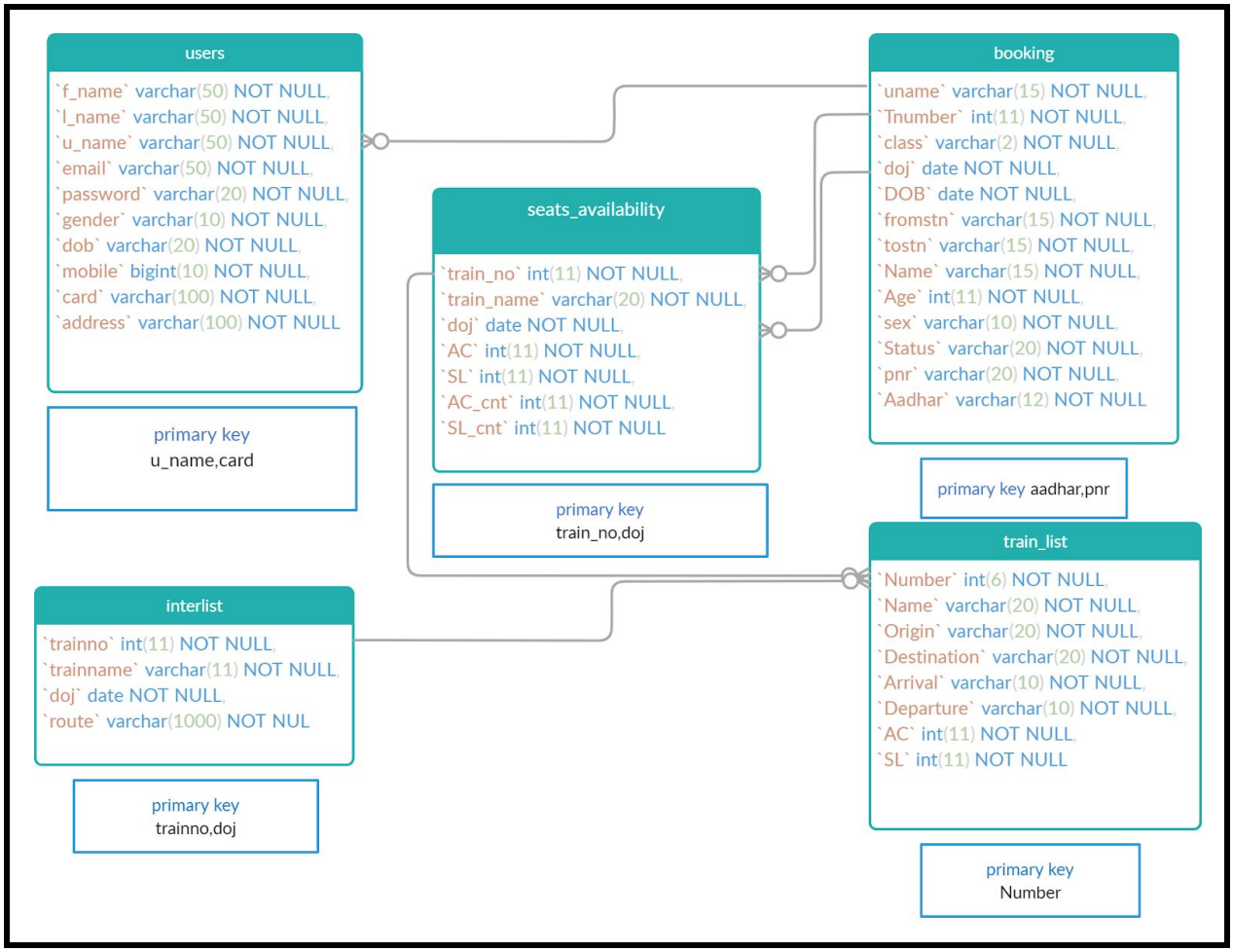


FIG 1 :- ER DIAGRAM

:: TABLE ATTRIBUTES ::

TABLE users: [f_name, l_name, u_name, email, password, gender, dob, mobile, card,address**]**

- primary key(u_name, card)
- Card - user credit card number
- dob - date of birth

TABLE train_list: [number ,name ,origin ,destination, arrival, departure,AC(fare),SL(fare) **]**

- primary key(number)
- AC - Train AC seat fare per passenger
- SL - Train SL seat fare per passenger

TABLE seat_availability: [train_no, train_name, doj, AC(coach), SL(coach), AC_cnt, SL_cnt **]**

- primary key(train_no,doj)
- foreign key(train_no) references train_list(number)
- AC - Total AC coaches in train
- SL - Total SL coaches in train
- AC_cnt - Total AC seats remaining ----- Initially equal to AC*18
- SL_cnt - Total SL seats remaining ----- Initially equal to SL*24

TABLE booking: [aadhar, uname, Tnumber, class, doj, DOB, from, to, Name, Age, sex, status, pnr **]**

- primary key(aadhar)
- foreign key(uname) references users(u_name)
- foreign key(Tnumber,doj) references seat_availability(train_no,doj)
- Status - Complete Seat details [Coach No. Seat No. Berth Type] **Eg:-** 1 6 SL
- DOB - date of booking
- doj - date of journey
- class - AC/SL

TABLE interlist: [trainno, trainname, doj, route **] -- THIS IS TO FIND ALTERNATIVE ROUTE**

- primary key(aadhar)
- foreign key(uname) references users(u_name)
- foreign key(Tnumber,doj) references seat_availability(train_no,doj)
- doj - date of journey
- route - Route followed by the train [**A** (Arr Time # Dept Time) → **B**() → **C**() → **D**() .....]

* REST ATTRIBUTE NAMES ARE SELF EXPLANATORY

(b) TRIGGER & STORED PROCEDURES USED

IMPLEMENTED TRIGGERS & CHECKS AS SERVER CODE.

- ➢ For signup and login check
- ➢ For edit profile details
- ➢ For password update
- ➢ To search direct and alternative trains for a given date
- ➢ For seat availability check
- ➢ Some input checks in UI

PROCEDURES

- ➢ ticket search MySQL procedure
- ➢ insert ticket details MySQL procedure [after booking]
- ➢ update remaining seats MySQL procedure
- ➢ find alternative route MySQL procedure
- ➢ fetch ticket details and display booking history

(c) TICKET BOOKING PROCEDURE

Basic Logic →

- ➔ Seat allocation is done in a **linear order** i.e., if the seats are available in train for the required journey data then the seats will be allocated first coach wise followed by seat wise.
- ➔ User inputs journey details -- Date of journey , Source, Destination via UI form
- ➔ Then at the backend we have implemented a **ticket search MySQL procedure** which runs a query (by theta join of train_list and seat_availability tables) to display train list that satisfies the user journey details
- ➔ Followed by this the user selects the AC/SL class and asks to book a ticket.
- ➔ Now the user will be directed to the booking portal : Here, user input details for each passenger -- Name, Age, Aadhar, Gender. **Note:** At most 5 passengers are allowed per ticket.

UI VIEW OF BOOKING ENGINE

Seat Allocation Algorithm →

➔ We have a count of seats remaining from the seat_availability table in a particular class AC/SL selected by the user (say in variable **value**).

➔ As per the given layout we have 18 seats per AC coach and 24 seats per SL coach.

➔ We have used a **sitting array** to map seat numbers to their respective berth types. Using the basic modulo operations we are determining the current coach and seat number from the **value. [**code snippet below**]**

➔ After the seat is allotted and pnr is generated, booking details are updated into the **booking** table via **insert ticket MySQL procedure** and also the seat remaining is updated into the **seat_availability** table via **update seats MySQL procedure.**

```php
$AC_sitting = array("LB","LB","UB","UB","SL","SU");
$SL_sitting = array("LB","MB","UB","LB","MB","UB","SL","SU");

if($seat=="AC")
{
    $seat_no = $coach_cnt*18 - $value + 1;
    $coach_no = ceil($seat_no/18);

    if($seat_no%18==0)  $seat_no = 18;
    else  $seat_no=$seat_no%18;

    $seat_type = $AC_sitting[($seat_no-1)%6];
}

else if($seat=="SL")
{
    $seat_no = $coach_cnt*24 - $value + 1;
    $coach_no = ceil($seat_no/24);

    if($seat_no%24==0) $seat_no = 24;
    else $seat_no=$seat_no%24;

    $seat_type = $SL_sitting[($seat_no-1)%8];
}
```

CODE SNIPPET FOR SEAT ALLOCATION

```php
$PNR = '';
$curr_date = date("Y/m/d");
$split = (explode("/",$datee));

$curr_time = time();

for($i=0;$i<count($split);$i++){
    $PNR .= "$split[$i]";
}
$PNR = $PNR.$curr_time;
```

We have implemented a unique **PNR generation method** by concatenating current time and date. Since at a given time only a single user can book so the PNR will be unique every time [Even if a new database is loaded].

CODE SNIPPET FOR UNIQUE PNR GENERATION

❑ $ticket_status = $coach_no." ".$seat_no." ".$seat_type;

(d) TRAIN RELEASE BY ADMIN

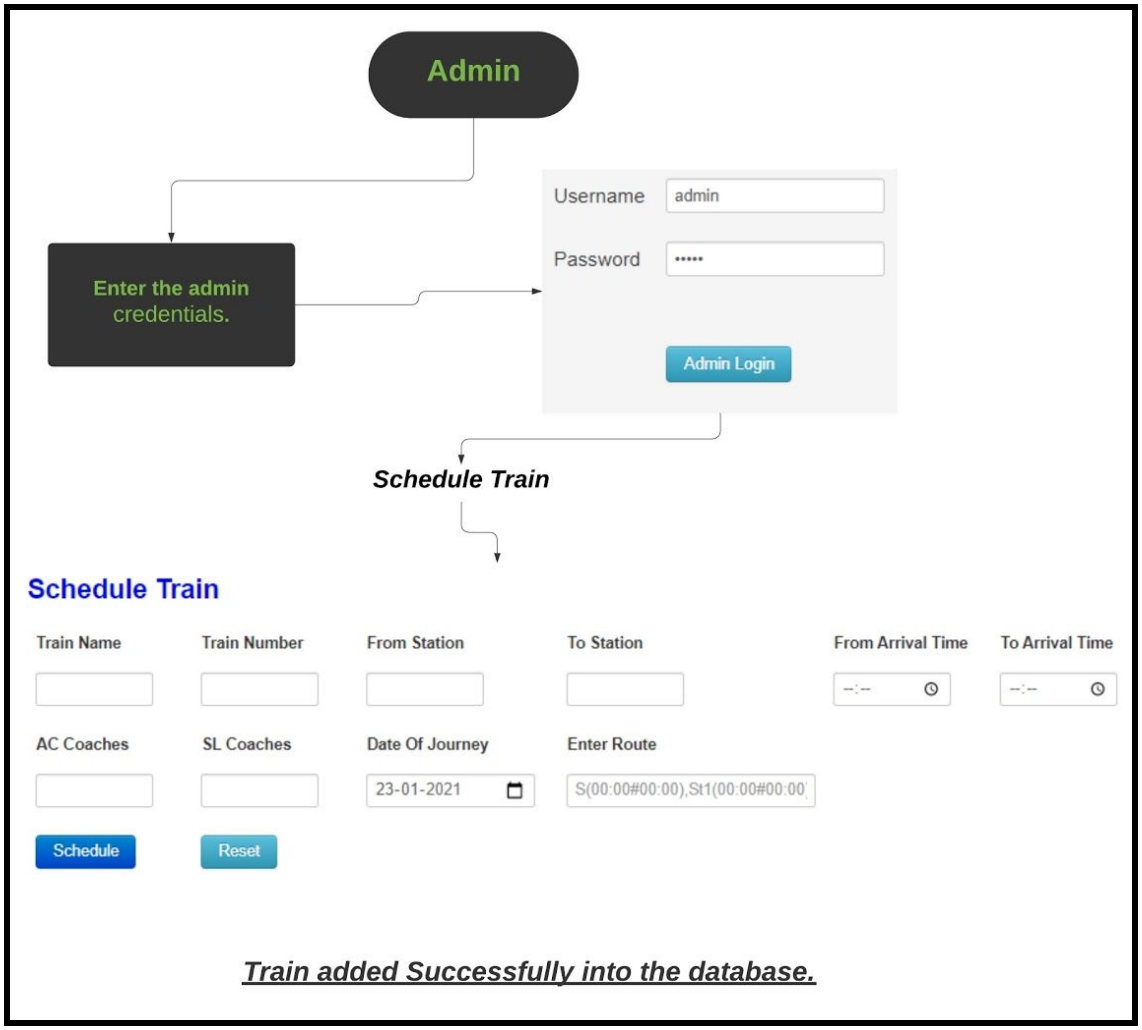
Basic Logic →

First the admin login into the admin portal using “admin” username and password “admin”.

UI inputs the complete train details from the admin -- Train Number, Train Name, Train Route, AC Coaches, SL Coaches , Date to release train, Arrival and Departure time.

Now, an **add train MySQL procedure** is called to insert train details into the database.

Tables Affected in order→ train_list followed by seats_availabilty and interlist.

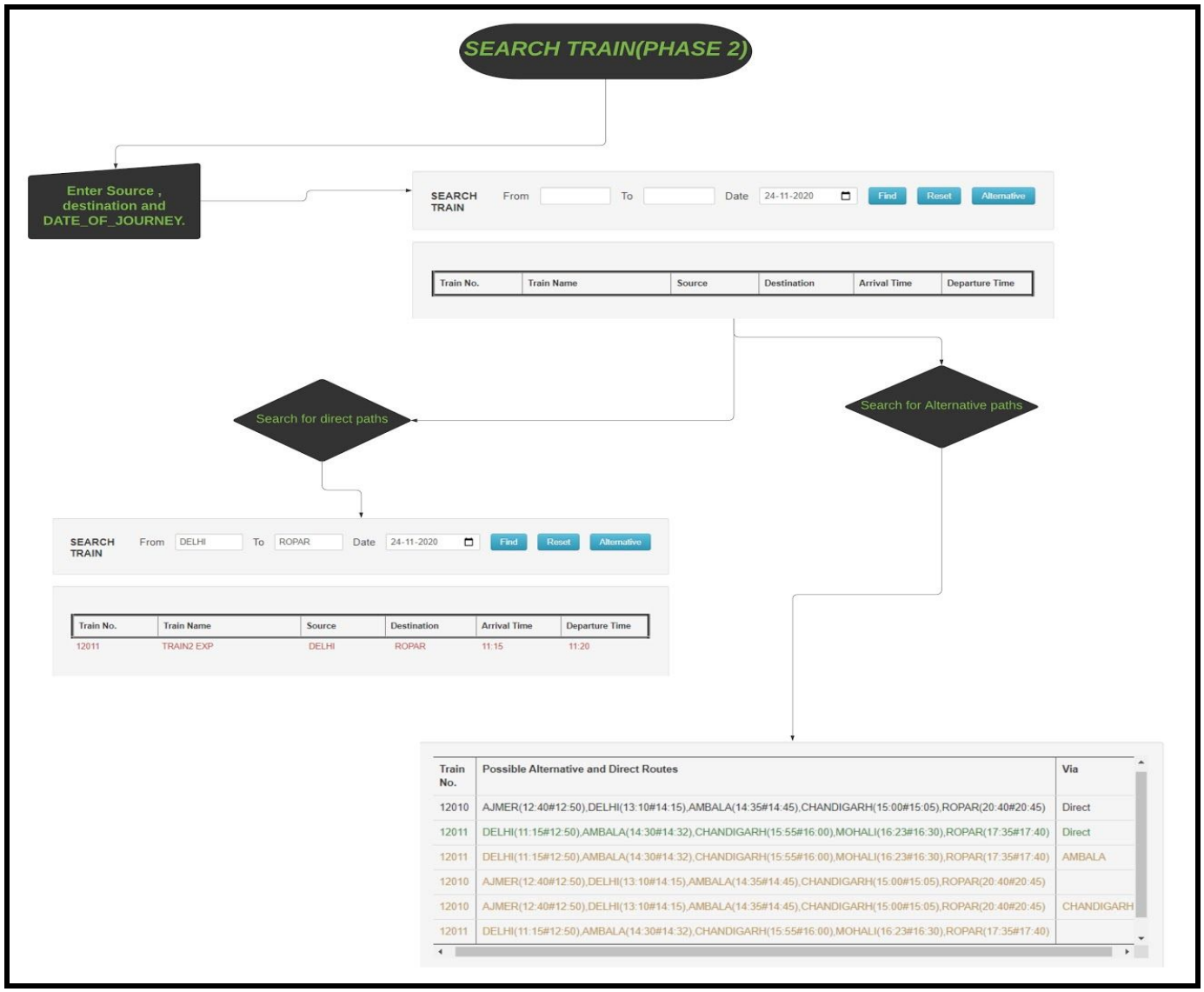


UI VIEW OF ADMIN PAGE

(e) ALTERNATIVE ROUTE SEARCH

Search procedure for searching trains between any two stations.

- We have stored the stations (and their corresponding arrival and departure times) as served by a train in the interlist table in the database. Also stored the order of stations seen by the train in its route.
- Search algorithm is implemented assuming that at-most one break is needed to reach from anywhere to anywhere. If no path is found, then our algorithm shows “journey plan is not possible.”
- While recommending alternative routes, we are checking the compatibility of the arrival and departure times of the trains involved in the connection.



UI VIEW OF ALTERNATIVE SEARCH PAGE

UI inputs source, destination stations and journey date. Then via search algorithm, we show the direct and one-hop journey routes from source to destination considering that arrival time of the intermediate station via train1 is strictly less than the departure time of the intermediate station of the train2.

Algorithm:

1) We will be using the **interlist** table. A search query is used to find all trains running on that particular date from the interlist table.
2) We iterate over each train route and store the intermediate stations in an array by performing string operations on the route and then for every pair of the stations in the array we store the train number. We do this thing for all the trains (finally we have a map of all possible pairs of the stations and train number corresponding to it).
3) We also create an array (say **arr**) of all the distinct stations.
4) Then we iterate through the array **arr** and consider each station as the intermediate station and then in the map we select the all possible alternative paths ($map[$src.” “.$intermediate]*$map[$intermediate.” “.$dest]).

```
DIRECT-->
if(map(<SRC,DEST>)!=NULL) "output route"

ALTERNARIVE-->
else if(map(<SRC,intermediate>)!=NULL && map(<intermediate,SRC>)!=NULL)
    "output route"

else "journey plan is not possible"
```

PSEUDO CODE

```
::MAP for train1 :: ["Route 12001 -> a -> x -> y -> c -> b"]
map(<a,x>)=12001 map(<a,y>)=12001 map(<a,c>)=12001 map(<a,b>)=12001
map(<x,y>)=12001 map(<x,c>)=12001 map(<x,b>)=12001
map(<y,c>)=12001 map(<y,b>)=12001
map(<c,b>)=12001

::MAP for train2 :: ["Route 21009 -> k -> x -> c -> y"]
map(<k,x>)=21009 map(<k,c>)=21009 map(<k,y>)=21009
map(<x,c>)=21009 map(<x,y>)=21009
map(<c,y>)=21009

::MAP for train3 :: ["Route 12341 -> a -> x -> b"]
map(<a,x>)=12341 map(<a,b>)=12341
map(<x,b>)=12341

distint_staions() = {a, x, y, c, b, k} // POSSIBLE INTERMEDIATES

EXAMPLE:- SRC = "a"  DEST="y"

DIRECT ROUTE-->
In map we search for map(<a,y>) we get "12001" as direct route

ALTERNARIVE ROUTE-->
LOOP intermediate in distint_staions():
    In map we search for map(<a,intermediate>)=V1 and map(<intermediate,y>)=V2
    We will print the all possible alternative paths iff V1!=NULL && V2!=NULL
    As in out example we get output:
        Alternative1 :"12001" and "21009" via "x"
        Alternative2 :"12341" and "21009" via "x"
    ** We also check the feasibilty of the routes by comparing the arrival of
       train1 and departire time of train2 at intermediate station.
```

ILLUSTRATION OF ALGORITHM USING AN EXAMPLE

---------- DATABASE AUTHORIZATION AND CONNECTION ----------

- First we create the database on the PHP admin and then import railres.sql file
- Then using PHP database management tools we connect our database to cater over the localhost and return query results

```php
<?php

$host="localhost"; // Host name
$username="root"; // Mysql username
$password=""; // Mysql password
$db_name="railres"; // Database name

$conn=mysqli_connect("$host", "$username", "$password")or die("cannot connect");
mysqli_select_db($conn,"$db_name")or die("cannot select DB");

?>
```

Connecting database with the localhost

```php
$sql="SELECT * FROM $tbl_name WHERE u_name='$uname' and password='$pass'";

$result=mysqli_query($conn,$sql) or trigger_error(mysql_error.$sql);
```

Trigger Error on unsuccessful query connect
