

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO DBMS

A database is simply an organized collection of related data, typically stored on disk, and accessible by possibly many concurrent users. Databases are generally separated into application areas. For example, one database may contain Human Resource (employee and payroll) data; another may contain sales data; another may contain accounting data; and so on. Databases are managed by a DBMS. Many Database Systems are being used which are in turn managed by many other Database Management Systems. A Database Management System (DBMS) is a set of programs that manages any number of databases. Basically DBMS is a software tool to organize (create, retrieve, update and manage) data in a database. The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Database systems are meant to handle large collection of information. Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

1.2 BRIEF OUTLINE OF PROJECT

The project “WEATHER REPORTING SYSTEM” is a system for reporting the various details in different cities. It manages the weather data, user details, city details, hourly details and daily details of a user.

The system also allows user from any city to update the weather status whenever they want to do so.

The purpose of this project is to design, build and implement an weather reporting system with anytime and anywhere access availability. All user information will be stored in a SQL database. The main purpose of this project is to provide an easy-to-use interface for users of a city to interact with each other.

Any person can enter the data of his place. There is no separate admin rights for entering the data, the admin can verify the data in the back end and has rights to alter them. SO for people interested in updating data about their place, climate and such things it is easy for them to do so. Other users can also view the details of others as it's directly shown in the respective pages.

1.3 PROJECT GOAL

This software product the weather reporting system is to improve the services for all the users of this database. This also reduce the manual work of the persons in admin panel and the bundle of registers that to find how many users are in the particular, because through this system you can store the data of number of users living in different cities tirelessly.

1.4 SCOPE

The application program developed can be used in various fields as follows:

- It is user-friendly.
- It is easy to use and modify.
- We can add and update the information whenever we needed
- It can be used user of different cities easily.

CHAPTER 2

SYSTEM REQUIREMENTS

A software requirement definition is an abstract description of the services which the system should provide, and the constraints under which the system must operate. It should only specify the external behavior of the system. The requirements are specified as below:

2.1 HARDWARE REQUIREMENTS

- **Operating system:** Windows 7 or later
- **Processor:** Intel Pentium 4 or later
- **Memory:** 2GB Minimum ,4GB Recommended
- **Screen Resolution:**1280*1024 or larger
- **Application Window Size:**1024*680 or larger
- **Internet Connection:** Not Required

2.2 SOFTWARE REQUIREMENTS:

- Windows 7 or higher version OS.
- Google chrome v70.0.3538 or greater.
- XAMPP web server.
- NetBeans IDE.

CHAPTER 3

PROBLEM DESCRIPTION

WEATHER REPORTING SYSTEM

Weather reporting system contains, User login details which consist of Sno, username and password. **City details** which consists of City ID, City name, Longitude, Latitude, Zip code and Country. **User details** consists of User ID, Name, City ID, Country, and Added on column to know when the data is added. **Hourly Status** consist of Hourly ID, Weather ID, Temperature, Humidity, Wind Speed, Direction and Pressure. **Daily Details** consist of Daily ID, Weather ID, Date, Min Temp, Max Temp, Avg Humidity, Sun rise and Sun set. **Weather Status** consists of Weather ID, Weather status, City ID.

The system uses the following tables for maintaining this details.

- LOGIN DETAILS
- USER DETAILS
- WEATHER STATUS
- HOURLY STATUS
- DAILY STATUS
- CITY DETAILS

The table details are as follows:

Table 3.1: Login

COLUMN NAME	DATATYPE & SIZE	CONSTRAINTS	DESCRIPTION
sno	INT	PRIMARY KEY	Unique Id of user
username	VARCHAR(50)	NOT NULL	Full name of the user
password	VARCHAR(50)	NOT NULL	Password for login

Table 3.2: City

COLUMN NAME	DATATYPE & SIZE	CONSTRAINTS	DESCRIPTION
City_ID	VARCHAR(50)	PRIMARY KEY	Unique ID of a City
Name	VARCHAR(50)	NOT NULL	Full name of city
Longitude	VARCHAR(50)	NOT NULL	Longitude of place
Latitude	VARCHAR(50)	NOT NULL	Latitude of place
Zip	VARCHAR(50)	NOT NULL	Zip code of the city
Country	VARCHAR(50)	NOT NULL	Name of country
no_of_users	VARCHAR(50)	NOT NULL	Number of users in a particular city

Table 3.3: User

COLUMN NAME	DATATYPE & SIZE	CONSTRAINTS	DESCRIPTION
uid	VARCHAR(50)	PRIMARY KEY	Unique ID of user
name	VARCHAR(50)	NOT NULL	Name of the user
City_ID	VARCHAR(50)	FOREIGN KEY	City ID
country	VARCHAR(50)	NOT NULL	Country of user
addon	VARCHAR(50)	NOT NULL	Date of adding details

Table 3.4: Daily

COLUMN NAME	DATATYPE & SIZE	CONSTRAINTS	DESCRIPTION
did	VARCHAR(50)	PRIMARY KEY	Unique Daily ID
wid	VARCHAR(50)	FOREIGN KEY	Weaather ID
dates	VARCHAR(50)	NOT NULL	Date of use
mint	VARCHAR(50)	NOT NULL	Min Temperature
maxt	VARCHAR(50)	NOT NULL	Max Temperature
avghum	VARCHAR(50)	NOT NULL	Average humidity
sunr	VARCHAR(50)	NOT NULL	Sun Rise
suns	VARCHAR(50)	NOT NULL	Sun Set

Table 3.5: Hourly

COLUMN NAME	DATATYPE & SIZE	CONSTRAINTS	DESCRIPTION
hid	VARCHAR(50)	PRIMARY KEY	Hourly ID
wid	VARCHAR(50)	FOREIGN KEY	Weather ID
temp	VARCHAR(50)	NOT NULL	Temperature
humidity	VARCHAR(50)	NOT NULL	Humidity at that hour
wind	VARCHAR(50)	NOT NULL	Speed of Wind
direction	VARCHAR(50)	NOT NULL	Direction of wind
pressure	VARCHAR(50)	NOT NULL	Atmospheric pressure

Table 3.6: Status

COLUMN NAME	DATATYPE & SIZE	CONSTRAINTS	DESCRIPTION
wid	VARCHAR(50)	PRIMARY KEY	Weather ID
wstatus	VARCHAR(50)	NOT NULL	Climate of that city
City_ID	VARCHAR(50)	FOREIGN KEY	City ID

CHAPTER 4

SYSTEM DESIGN

4.1 ER Diagram

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

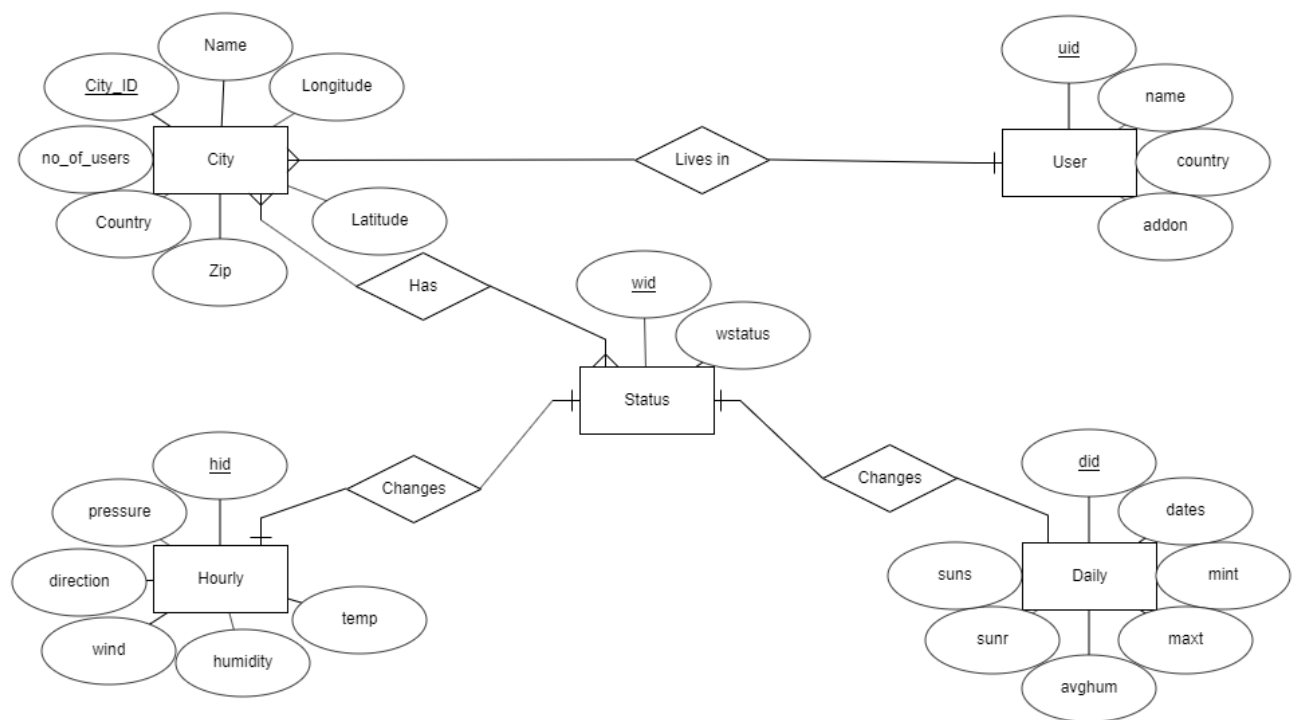


Figure 4.1- ER Diagram for Weather Reporting System

4.2 Schema Diagram:

A database schema is the skeleton structure that represents the logical view of the entire database. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams.

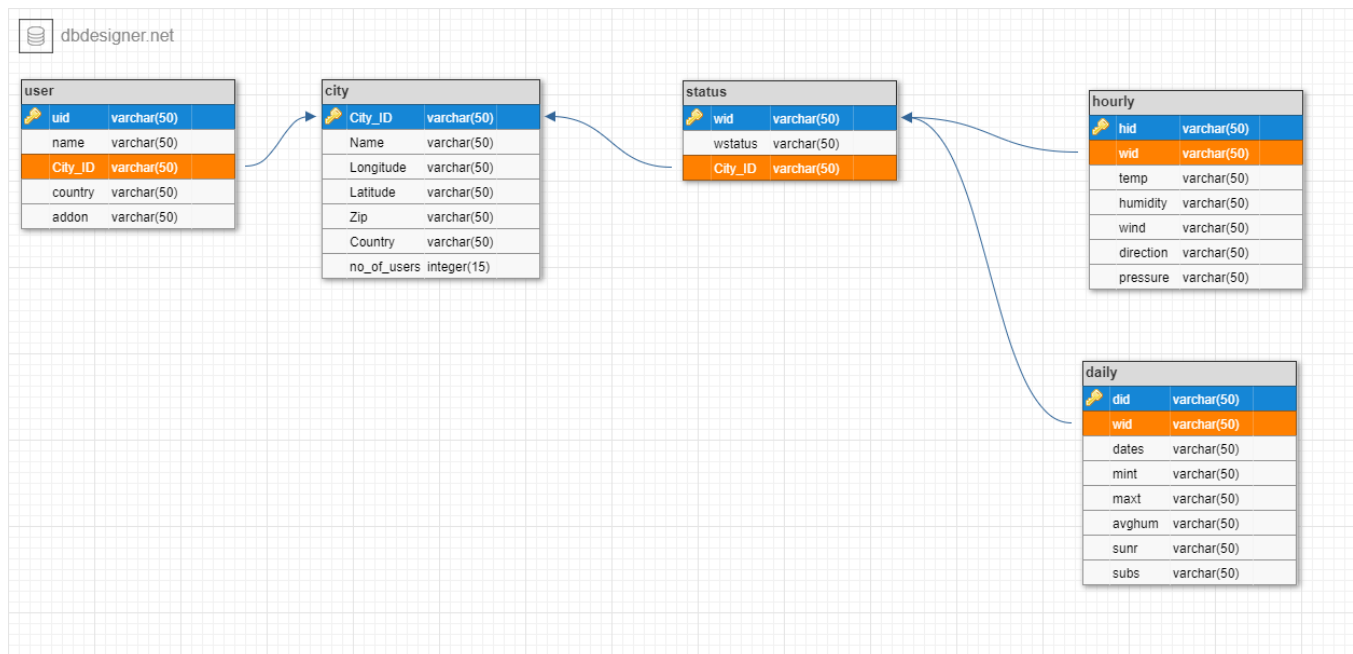


Figure 4.2-Schema diagram for Weather Reporting System

4.3 Normal Form

4.3.1 First Normal Form(1NF)

A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.

4.3.2 Second Normal Form(2NF)

A relation is in second normal form if it fulfills the following two requirements:

- It is in first normal form.
- It does not have any non-prime attribute that is functionally dependent on any proper subset of any candidate key of the relation. A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation.

4.4.3 Third Normal Form (3NF)

3NF is a normal form that is used in normalizing a database design to reduce the duplication of data and ensure referential integrity by ensuring that:

- The entity is in second normal form.
- No non-prime (non-key) attribute is transitively dependent on any key i.e. no non-prime attribute depends on other non-prime attributes. All the non-prime attribute must depend only on the candidate keys.

All the tables in WEATHER REPORTING SYSTEM satisfy all the three normal forms.

CHAPTER 5

IMPLEMENTATION

List of codes used for implementation:

- * LOGIN CODE
- * REGISTRATION CODE
- * CODE FOR CITY(insert,update,delete)
- * CODE FOR USER(insert,update,delete)
- * CODE FOR STATUS(insert,update,delete)
- * CODE FOR DAILY(insert,update,delete)
- * CODE FOR HOURLY(insert,update,delete)

5.1 List of codes

Code for login page:

```
try{
    Class.forName("com.mysql.jdbc.Driver");
    Connection con =(Connection)
    DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb", "root","");
    String sql ="SELECT * FROM `login` WHERE username=? and password =?";
    PreparedStatement pst =(PreparedStatement) con.prepareStatement(sql);
    pst.setString(1,username.getText());
    pst.setString(2, password.getText());
    ResultSet rs = pst.executeQuery();
    if(rs.next()){

        HOME menu = new HOME();
        menu.setVisible(true);
        setVisible(false);
    }
}
```

```
    }
    else{
        JOptionPane.showMessageDialog(null,"Username and Password don't Match");
    }
    con.close();
}
catch(Exception e){
    JOptionPane.showMessageDialog(null,e);
}
}
```

Code for Register page:

```
try{
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
    String C_ID= username.getText();
    String VARIABLE_1 =password.getText();
    PreparedStatement ps;
    String query = "INSERT INTO `login`(`username`, `password`) VALUES(?,?)";
    ps = con.prepareStatement(query);
    ps.setString(1,C_ID);
    ps.setString(2,VARIABLE_1);
    if(ps.executeUpdate() > 0)
    {
        JOptionPane.showMessageDialog(null,"User Registered");
        Login lg = new Login();
        lg.setVisible(true);
        setVisible(false);
    }
}
```

```
}  
    catch(Exception e)  
    {
```

Code for City page

INSERT:

```
try{  
    Class.forName("com.mysql.jdbc.Driver");  
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");  
    String C_ID= City_ID.getText();  
    String VARIABLE_1 = Name.getText();  
    String VARIABLE_2= Longitude.getText();  
    String VARIABLE_3 = Latitude.getText();  
    String VARIABLE_4 = Zip.getText();  
    String CH_ID = Country.getText();  
    String USER = no_of_users.getText();  
  
    PreparedStatement ps;  
    String query = "INSERT INTO `city`(`City_ID`, `Name`, `Longitude`, `Latitude`, `Zip`,  
    `Country`,no_of_users) VALUES(?,?,?,?,?,?,?)";  
    ps = con.prepareStatement(query);  
    ps.setString(1,C_ID);  
    ps.setString(2,VARIABLE_1);  
    ps.setString(3,GENDER);  
    ps.setString(4,VARIABLE_3);  
    ps.setString(5,VARIABLE_4);  
    ps.setString(6,CH_ID);  
    ps.setString(7,"0");  
  
    if(ps.executeUpdate() > 0)
```

```
{
OptionPane.showMessageDialog(null,"Details Added");
City lg = new City();
lg.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{

}
}

public void showtable()
{
try
{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb", "root","");
String sql="SELECT * FROM city";
PreparedStatement ps = con.prepareStatement(sql);
ResultSet rs = ps.executeQuery();
City.setModel(DbUtils.resultSetToTableModel(rs));
}
catch(Exception e)
{
OptionPane.showMessageDialog(null,e);
}
}
```

UPDATE:

```
try{
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
    String sub_name = City_ID.getText();
    String sub_id = Name.getText();
    String fac_id = Longitude.getText();
    String dep_name = Latitude.getText();
    String zipc = Zip.getText();
    String countr = Country.getText();

    PreparedStatement ps;
    String query = "UPDATE `city` SET
    `Name`='"+sub_id+"',`Longitude`='"+fac_id+"',`Zip`='"+zipc+"',`Country`='"+countr+"',`Latitud
    e`='"+dep_name+"' WHERE `City_ID`='"+sub_name+"'";
    ps = con.prepareStatement(query);

    if(ps.executeUpdate() > 0)
    {
        JOptionPane.showMessageDialog(null,"Updated Successfully");
        City s = new City();
        s.setVisible(true);
        setVisible(false);
    }
    }
    catch(Exception e)
    {
    }
}
```

DELETE:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= City_ID.getText();
PreparedStatement ps;
String query = "DELETE FROM `city` WHERE `City_ID`='"+C_ID+"' ";
ps = con.prepareStatement(query);
if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"Deleted Successfully");
City s = new City();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
}
}
```

Code for User page**INSERT:**

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= uid.getText();
String VARIABLE_1 = name.getText();
String VARIABLE_2= City_ID.getText();
String VARIABLE_3 = country.getText();
```



```
String VARIABLE_4 = addon.getText();
PreparedStatement ps;
String query = "INSERT INTO `user`(`uid`, `name`, `City_ID`, `country`, `addon`)
VALUES(?,?,?,?,?)";
ps = con.prepareStatement(query);
ps.setString(1,C_ID);
ps.setString(2,VARIABLE_1);
ps.setString(3,GENDER);
ps.setString(4,VARIABLE_3);
ps.setString(5,VARIABLE_4);
if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"User Details Added");
User lg = new User();
lg.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
}
}
```

UPDATE:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String sub_name = uid.getText();
String sub_id = name.getText();
String fac_id = City_ID.getText();
```

```
String dep_name = country.getText();
String zipc = addon.getText();
PreparedStatement ps;
String query = "UPDATE `user` SET
`name`='"+sub_id+"',`City_ID`='"+fac_id+"',`country`='"+dep_name+"',`addon`='"+zipc+"
WHERE `uid`='"+sub_name+"'";
ps = con.prepareStatement(query);
if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"Updated Successfully");
User s = new User();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
} }
```

DELETE:

```
try{
CallableStatement cs;
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= uid.getText();
PreparedStatement ps;
String query = "DELETE FROM `user` WHERE `uid`='"+C_ID+" ";
ps = con.prepareStatement(query);
cs = con.prepareCall("{ call store('"+C_ID+"')}");
```

```
if(cs.executeUpdate() > 0)
{
OptionPane.showMessageDialog(null,"Deleted Successfully");
User s = new User();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
}
}
```

Code for Status page

INSERT:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= wid.getText();
String VARIABLE_1 =wstatus.getText();
String CITY= City_ID.getText();
PreparedStatement ps;
String query = "INSERT INTO `status`(`wid`,`wstatus`,`City_ID`) VALUES(?,?,?)";
ps = con.prepareStatement(query);
ps.setString(1,C_ID);
ps.setString(2,VARIABLE_1);
ps.setString(3,CITY);
if(ps.executeUpdate() > 0)
{
}
```

```
JOptionPane.showMessageDialog(null,"Details Added");
Status lg = new Status();
lg.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
}
}
```

UPDATE:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String sub_name = wid.getText();
String sub_id = wstatus.getText();
String CITY = City_ID.getText();
PreparedStatement ps;
String query = "UPDATE `status` SET `wstatus`='"+sub_id+"`,`City_ID`='"+CITY+"` WHERE `wid`='"+sub_name+"'";
ps = con.prepareStatement(query);
if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"Updated Successfully");
Status s = new Status();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
```

```
{  
}  
}
```

DELETE:

```
try{  
Class.forName("com.mysql.jdbc.Driver");  
Connection con =  
DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");  
String C_ID= wid.getText();  
PreparedStatement ps;  
String query = "DELETE FROM `status` WHERE `wid`='"+C_ID+"' ";  
ps = con.prepareStatement(query);  
if(ps.executeUpdate() > 0)  
{  
JOptionPane.showMessageDialog(null,"Deleted Successfully");  
Status s = new Status();  
s.setVisible(true);  
setVisible(false);  
}  
}  
catch(Exception e)  
{  
}
```

Code for Daily page**INSERT:**

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= did.getText();
String VARIABLE_1 = wid.getText();
String VARIABLE_3 = dates.getText();
String VARIABLE_4 = mint.getText();
String CH_ID = maxt.getText();
String DIR = avghum.getText();
String PRESS = sunr.getText();
String SUN = suns.getText();
PreparedStatement ps;
String query = "INSERT INTO `daily`(`did`, `wid`, `dates`, `mint`, `maxt`, `avghum`, `sunr`,
`suns`) VALUES(?,?,?,?,?,?,?)";
ps = con.prepareStatement(query);
ps.setString(1,C_ID);
ps.setString(2,VARIABLE_1);
ps.setString(3,VARIABLE_3);
ps.setString(4,VARIABLE_4);
ps.setString(5,CH_ID);
ps.setString(6,DIR);
ps.setString(7,PRESS);
    ps.setString(8,SUN);

if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"Data Added");
```

```
Daily lg = new Daily();
lg.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{

}
}
```

UPDATE:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String sub_name = did.getText();
String sub_id = wid.getText();

String dep_name = dates.getText();
String zipc = mint.getText();
String countr = maxt.getText();
String dir = avghum.getText();
String pressr = sunr.getText();
String sun = suns.getText();
PreparedStatement ps;
String query = "UPDATE `daily` SET
`wid`='"+sub_id+"',`dates`='"+dep_name+"',`mint`='"+zipc+"',`maxt`='"+countr+"',`avghum`='"+
dir+"',`sunr`='"+pressr+"',`suns`='"+sun+"' WHERE `did`='"+sub_name+"'";
ps = con.prepareStatement(query);
if(ps.executeUpdate() > 0)
{
```

```
JOptionPane.showMessageDialog(null,"Updated Successfully");
Daily s = new Daily();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{ } }
```

DELETE:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= did.getText();
PreparedStatement ps;
String query = "DELETE FROM `daily` WHERE `did`='"+C_ID+"' ";
ps = con.prepareStatement(query);

if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"Data Deleted");
Daily s = new Daily();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
}
}
```


Code for Hourly page**INSERT:**

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= hid.getText();
String VARIABLE_2= wid.getText();
String VARIABLE_3 = temp.getText();
String VARIABLE_4 = humidity.getText();
String CH_ID = wind.getText();
String DIR = direction.getText();
String PRESS = pressure.getText();
PreparedStatement ps;
String query = "INSERT INTO `hourly`(`hid`,`wid`,`temp`,`humidity`,`wind`,`direction`,`pressure`) VALUES(?,?,?,?,?,?,?)";
ps = con.prepareStatement(query);
ps.setString(1,C_ID);
ps.setString(2,GENDER);
ps.setString(3,VARIABLE_3);
ps.setString(4,VARIABLE_4);
ps.setString(5,CH_ID);
ps.setString(6,DIR);
ps.setString(7,PRESS);
if(ps.executeUpdate() > 0)
{JOptionPane.showMessageDialog(null,"Data Added");
Hourly lg = new Hourly();
lg.setVisible(true);
setVisible(false);} } catch(Exception e)
{ } }
```

UPDATE:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String sub_name = hid.getText();
String fac_id = wid.getText();
String dep_name = temp.getText();
String zipc = humidity.getText();
String countr = wind.getText();
String dir = direction.getText();
String pressr = pressure.getText();
PreparedStatement ps;
String query = "UPDATE `hourly` SET
`wid`='"+fac_id+"',`humidity`='"+zipc+"',`wind`='"+countr+"',`direction`='"+dir+"',`pressure`='"+
pressr+"',`temp`='"+dep_name+"' WHERE `hid`='"+sub_name+"'";
ps = con.prepareStatement(query);
if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"Updated Successfully");
Hourly s = new Hourly();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
}
}
```

DELETE:

```
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/logdb","root","");
String C_ID= hid.getText();
PreparedStatement ps;
String query = "DELETE FROM `hourly` WHERE `hid`='"+C_ID+"' ";
ps = con.prepareStatement(query);
if(ps.executeUpdate() > 0)
{
JOptionPane.showMessageDialog(null,"Data Deleted");
Hourly s = new Hourly();
s.setVisible(true);
setVisible(false);
}
}
catch(Exception e)
{
}
}
```

5.2 SQL Stored Procedure

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `store` (IN `del` VARCHAR(50)) NO  
SQL  
DELETE FROM user WHERE uid = del$$
```

There is only one stored procedure which is present in the user table. This is called whenever a user wants to delete details.

5.3 Trigger Details

```
CREATE TRIGGER `ct` AFTER INSERT ON `user` FOR EACH ROW UPDATE city c  
SET no_of_users = (SELECT COUNT(*) FROM user u WHERE u.City_ID=c.City_ID)
```

There are 2 triggers used namely ct and ut. ct trigger is used to calculate number of users present in particular city and ut trigger updates the no of users present in city table after deletion in user table.

CHAPTER 6

SCREEN SHOTS

List of screen shots:

- LOGIN PAGE
- REGISTRATION PAGE
- CITY PAGE
- USER PAGE
- STATUS PAGE
- HOURLY PAGE
- DAILY PAGE

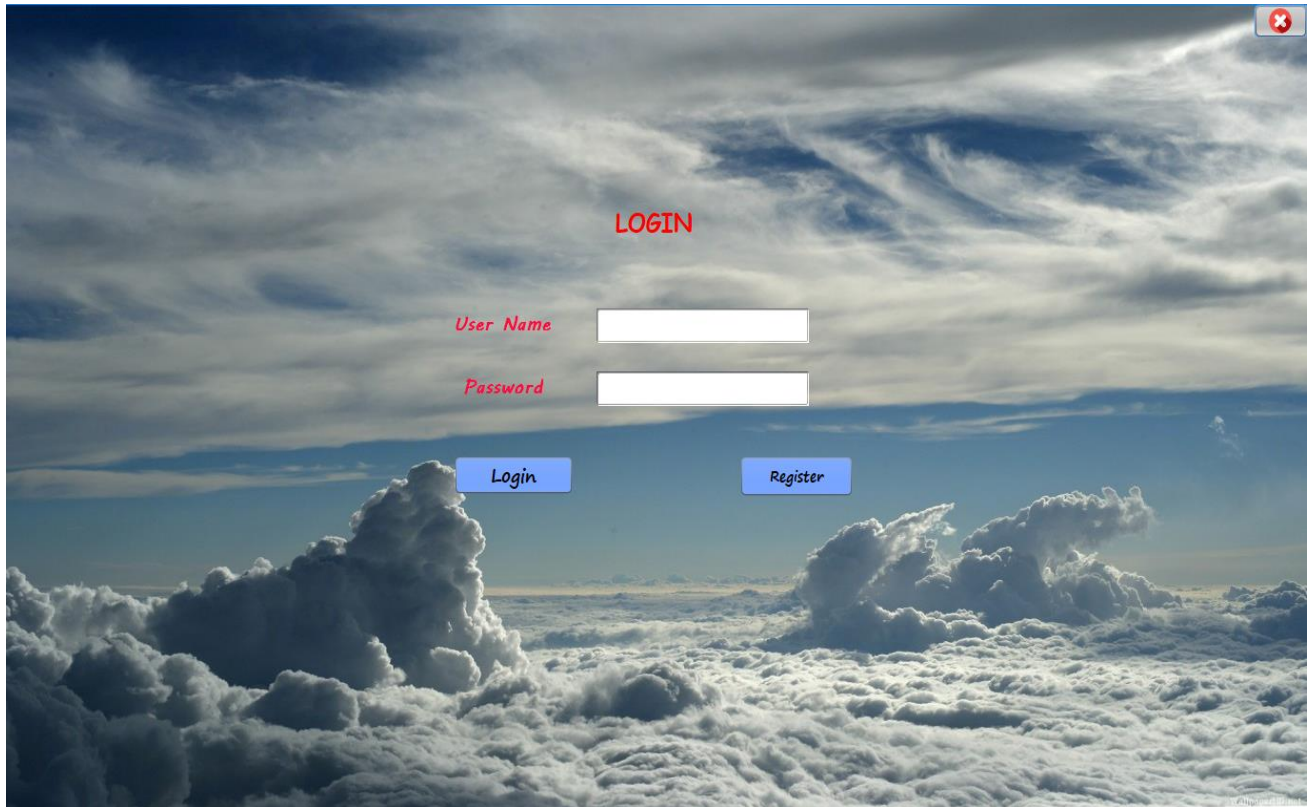
LOGIN PAGE:

Fig 6.1 Login page

The user can login with unique username and password. If username and password does not exist then they have to go for new registration using register page.

REGISTER PAGE:

Registration

Name

Gender

Mail

Password

Register

Fig 6.2 Register page

Register page includes the information of the user who wants to register. Users can register the account by clicking on register button.

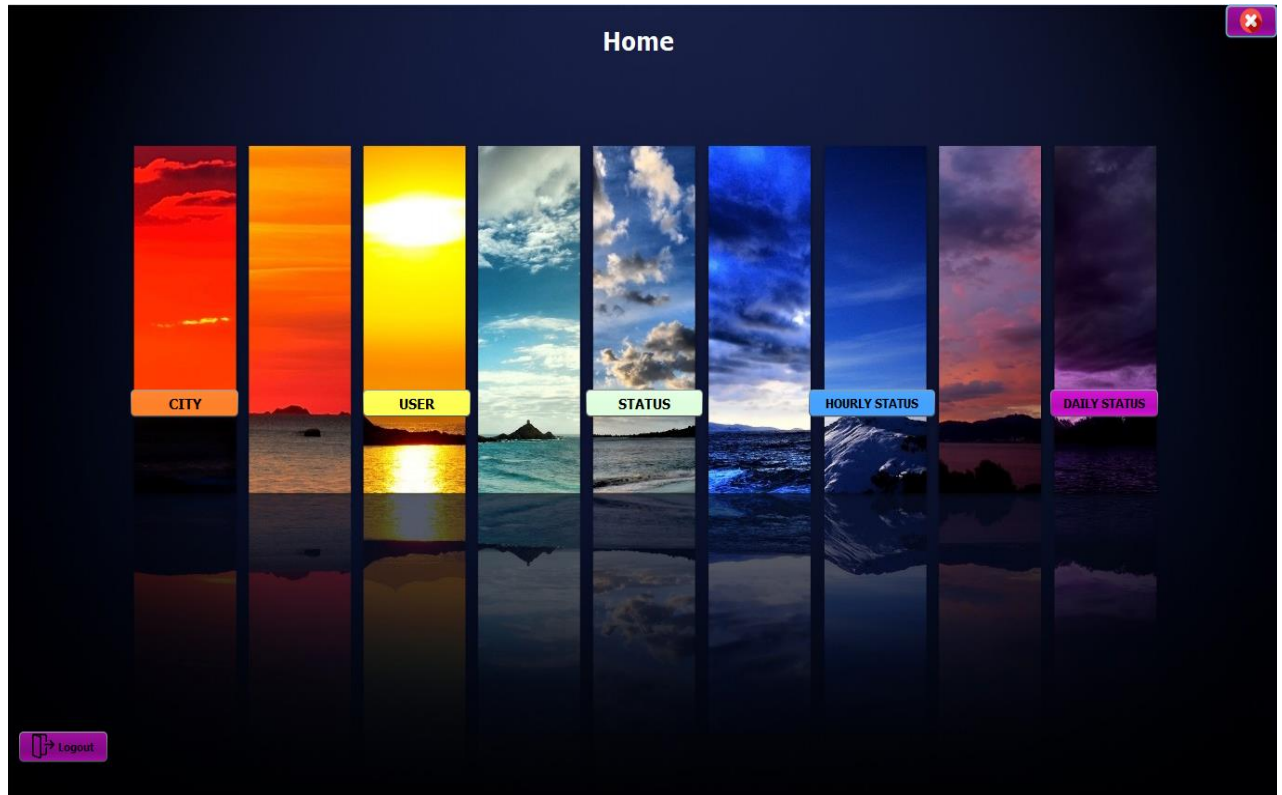
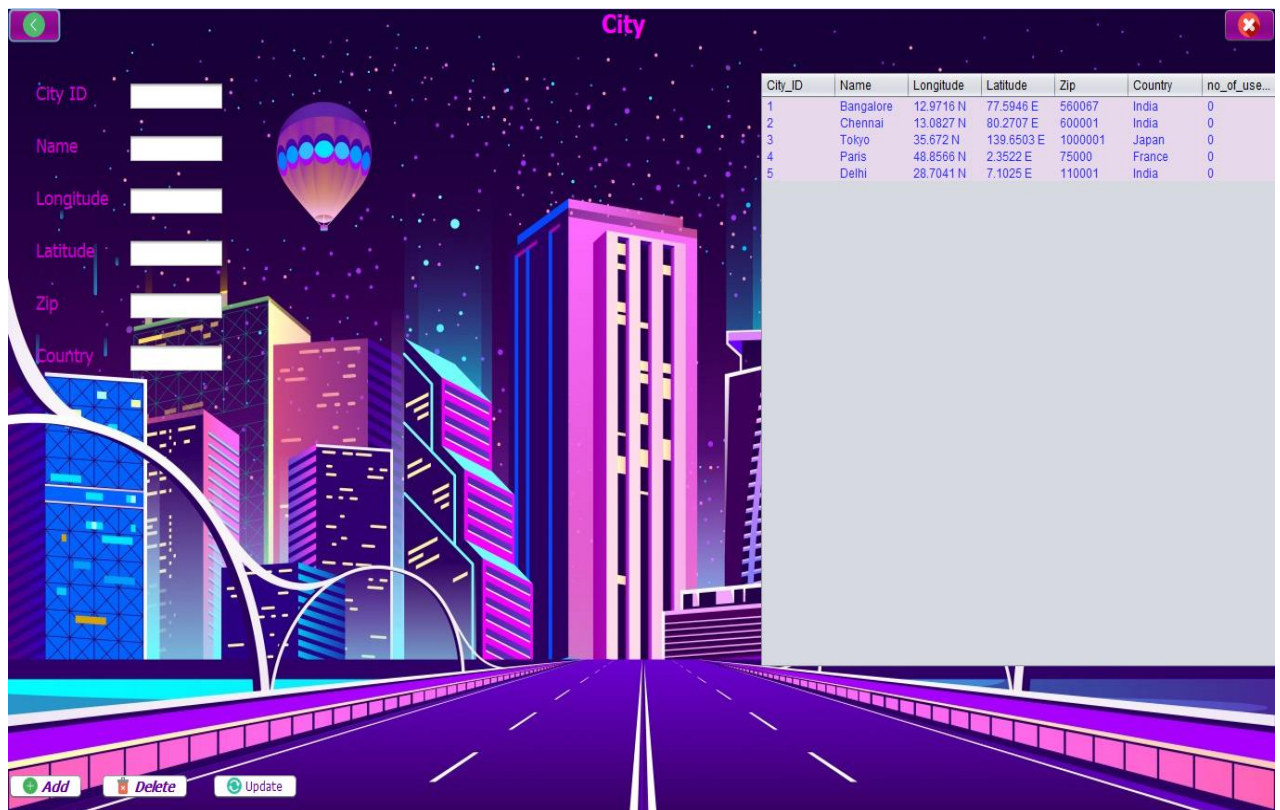
HOME PAGE:

Fig 6.3 Home page

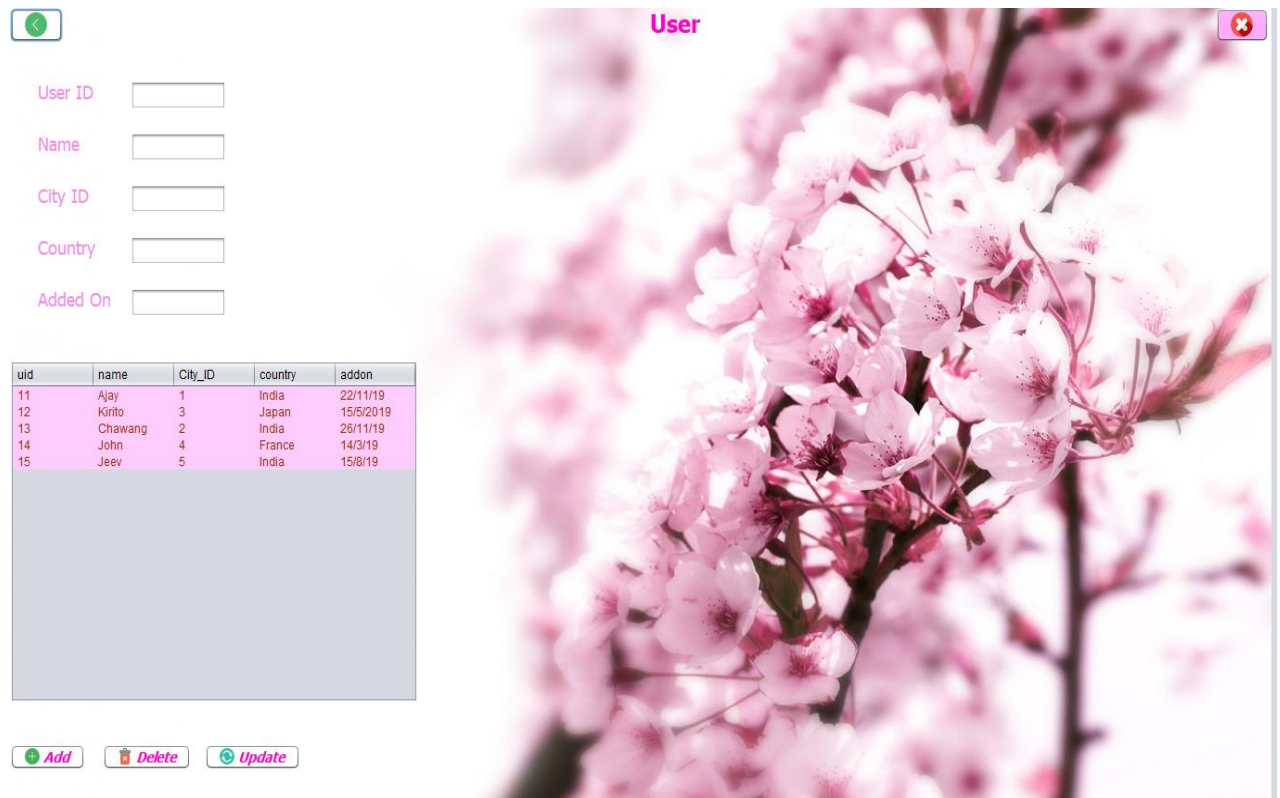
This is the home page or main page of a weather reporting system. This page defines all the table information included in it. this page includes the following tables such as city, user, status, hourly, daily.

CITY PAGE:

City_ID	Name	Longitude	Latitude	Zip	Country	no_of_use...
1	Bangalore	12.9716 N	77.5946 E	560067	India	0
2	Chennai	13.0827 N	80.2707 E	600001	India	0
3	Tokyo	35.672 N	139.6503 E	1000001	Japan	0
4	Paris	48.8566 N	2.3522 E	75000	France	0
5	Delhi	28.7041 N	7.1025 E	110001	India	0

Fig 6.4 City Page

City table is used to add details of the city like name, longitude, latitude of the city and its zip code and the country name. Cities are provided with unique id which is used for further information. The entire city details are listed in this page.

USER PAGE:

User ID

Name

City ID

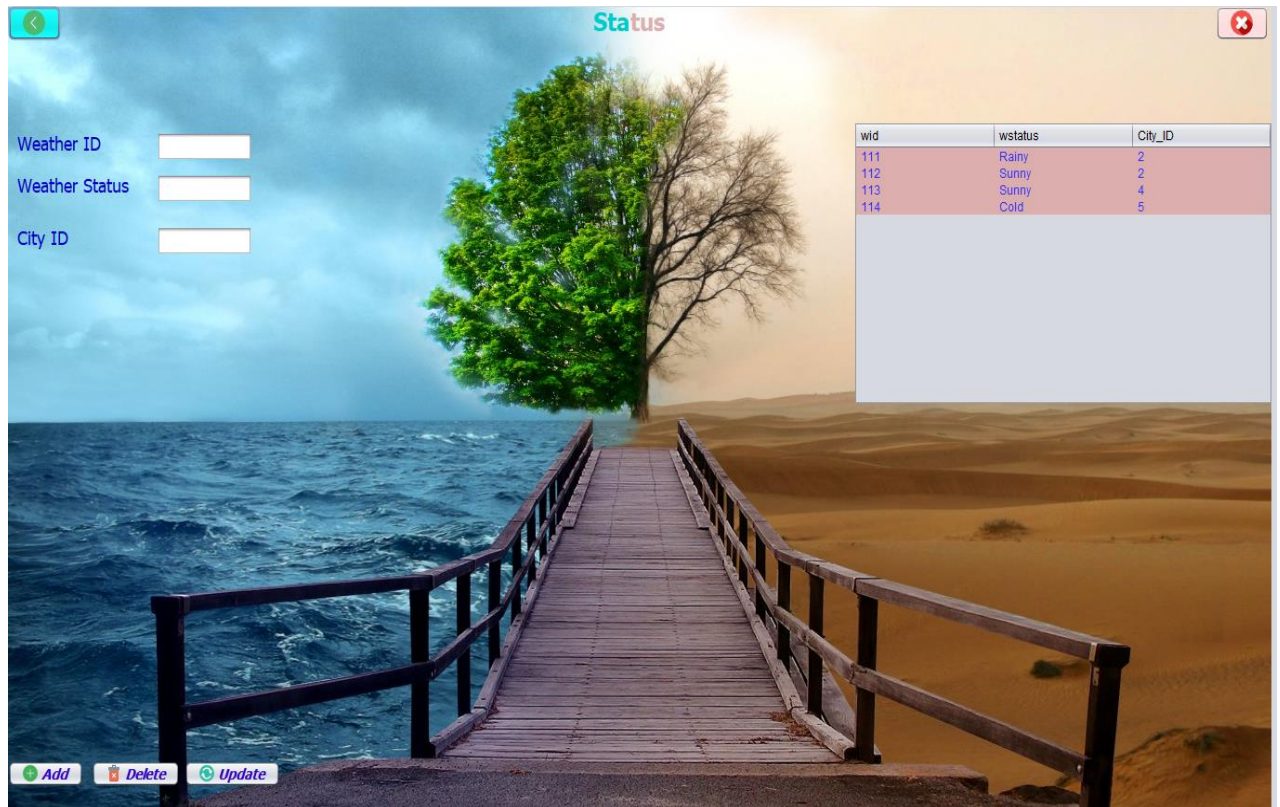
Country

Added On

uid	name	City_ID	country	addon
11	Ajay	1	India	22/11/19
12	Kirito	3	Japan	15/5/2019
13	Chawang	2	India	26/11/19
14	John	4	France	14/3/19
15	Jeev	5	India	15/8/19

Fig 6.5 User Page

User page has insert that is to add user details like City, name and etc. And delete and update for any changes to be made in table and it has entire details of user.

STATUS PAGE:

wid	wstatus	City_ID
111	Rainy	2
112	Sunny	2
113	Sunny	4
114	Cold	5

Fig 6.6 Status page

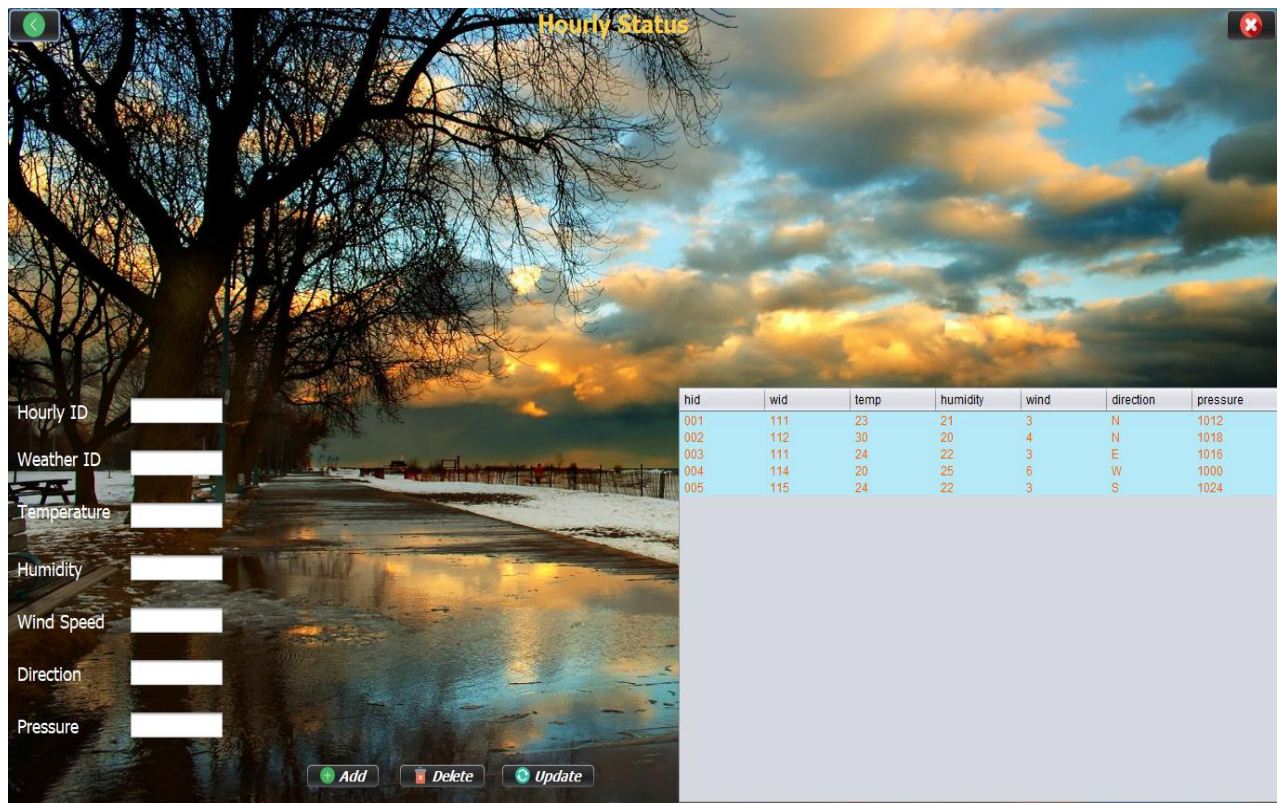
Status page has details of current weather status and city. We can add, delete and update the tables.

DAILY PAGE:

did	wid	dates	mint	maxt	avghum	sunr	suns
001	111	22/10/...	22	30	22	6:00	6:25
002	115	15/08/...	24	40	24	5:45	6:30
003	112	12/06/9	18	26	25	6:00	6:00
004	114	12/12/...	15	20	16	5:57	6:00
005	112	14/09/...	24	27	23	5:50	6:45

Fig 6.7 Daily Page

Daily table has the entire details of the weather status of a particular day. We can check the status by using unique key id.

HOURLY PAGE:

hid	wid	temp	humidity	wind	direction	pressure
001	111	23	21	3	N	1012
002	112	30	20	4	N	1018
003	111	24	22	3	E	1016
004	114	20	25	6	W	1000
005	115	24	22	3	S	1024

Fig 6.8 Hourly Page

Hourly page has the details of all information about the climate of the place at that hour, it has the basic information. We can check the details by using unique key hid.

CHAPTER 7

CONCLUSION & FUTURE SCOPE

With the theoretical inclination of our syllabus it becomes very essential to take the at most advantage of any opportunity of gaining practical experience that comes along. The building blocks of this Major Project "WEATHER REPORTING SYSTEM" was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

- The planning that goes into implementing a project.
- The importance of proper planning and an organized methodology.
- The key element of team spirit and co-ordination in a successful project.

The project also provided us the opportunity of interacting with our teachers and to gain from their best experience.

FUTURE RECOMMENDATION

WEATHER REPORTING SYSTEM is a software application to build such a way that it can be used by lot of users with advanced security and multiple features.

This is a simple way of managing details of climate and cities. There is no need of books, pen, paper or any diary to maintain the report of climate and cities, in this easily users can store the details. And they can easily retrieve the details.

REFERENCES

[1].Database System Model, Languages, Design and Application Programming, Ramez Elmasri and Shamkant B. Navathe, 7th edition, 2017, Pearson.

[2].Database Management System, Ramkrishnan, and Gehrke, 3rd edition, 2014, McGrawHill.