[GROUP 32]

[Phase 3]

Student1 Han Li, hli494, Section A
Student2 Mingzong Liu, mliu348, Section A
Student3 Jiaying Wang, jwang777, Section A
Student4 Yongzheng Zhang, yzhang3076, Section A

For creating table:

1.User

```
CREATE TABLE user (
   EmailAddress      VARCHAR (100)   NOT NULL,
   UserName          VARCHAR (100)   NOT NULL,
   Password          VARCHAR(100)    NOT NULL,
   JobType ENUM ('Admin','CityScientist','CityOfficial') NOT NULL,
   PRIMARY KEY (UserName),
   UNIQUE (EmailAddress)
);
```

2.City Offical

```
CREATE TABLE CityOfficial (
   CUserName          VARCHAR(100)   NOT NULL,
   Title              VARCHAR(100),
   Approved           BOOLEAN          DEFAULT NULL,
   CCity                 VARCHAR(30),
   CState             VARCHAR(20),
   PRIMARY KEY (CUserName),
   FOREIGN KEY (CUserName)  REFERENCES User(UserName)
   ON DELETE CASCADE ON UPDATE CASCADE,
   FOREIGN KEY (CCity, CState) REFERENCES CityState(City, State)
   ON DELETE SET NULL ON UPDATE CASCADE
);
```

3.City State

```
CREATE TABLE CityState(
   City    VARCHAR(30) NOT NULL,
   State  VARCHAR(20) NOT NULL,
   PRIMARY KEY (City,State)
);
```

4.POI

```
CREATE TABLE POI(
   DateFlaged     DATE,
   Flag        BOOLEAN DEFAULT FALSE,
   LocationName VARCHAR(100) NOT NULL,
   Zipcode     INTEGER(5),
   PCity        VARCHAR(30),
   PState     VARCHAR(20),
   PRIMARY KEY (LocationName),
```

```
   FOREIGN KEY (PCity,PState)  REFERENCES CityState(City,State) ON DELETE SET NULL
ON  UPDATE CASCADE
);
```

5. Data Point

```
CREATE TABLE DataPoint (
   Accepted     BOOLEAN DEFAULT NULL,
   DateTime     DATETIME NOT NULL,
   DataValue     INTEGER,
   DLocationName VARCHAR(100) NOT NULL,
   DType        VARCHAR(30),
        PRIMARY KEY(DateTime, DLocationName),
        FOREIGN KEY(DLocationName) REFERENCES POI (LocationName) ON DELETE
CASCADE ON UPDATE CASCADE,
         FOREIGN KEY (DType) REFERENCES DataType (Type) ON DELETE CASCADE ON
UPDATE CASCADE
);
```

6. Data Type

```
CREATE TABLE  DataType (
   Type        VARCHAR(30) NOT NULL,
   PRIMARY KEY (Type)
);
```

Completing the tasks:

1. Login Page
   a. SELECT * FROM user WHERE UserName='"+un+"'&& Password='"+pw+"' //Judge whether password fits username, un: username get from GUI, pw: password get from GUI.
   b. SELECT JobType FROM user WHERE UserName='"+un+"'
//After logging in, jump to specific GUI form according to user type, un: username get from GUI.
   c. SELECT * FROM CityOfficial WHERE CUserName = '"+un+"'
//Judge whether the user is denied or pending, if so, he/she cannot log in.

2. Register Page
   a. SELECT * FROM user WHERE UserName = '"+un+"'
//Judge whether username has been existed, un: username get from GUI.
   b. INSERT INTO CityOfficial " + "VALUES
   ('"+un+"','"+jt+"',FALSE ,'"+cy+"','"+st+"','"+tt+"')
//If new user is City Official, insert the new user into City Official table and user table, un is username, jt is jobtype, FALSE means not been accepted, cy is city, st is state, and tt is titile.
   c. INSERT INTO user " + "VALUES ('"+em+"','"+un+"','"+pw+"','"+jt+"')
//If new user is City Scientist, insert the new user into user table.
   d. SELECT City FROM CityState WHERE State = '"+state+"';
//Filter cities added into drop down box, according to the state that already been chosen.
   e. SELECT DISTINCT(State) FROM CityState
//Add states into drop down box according to the states in database.

3. AddNewDataPoint Page
    a. SELECT * FROM DataPoint WHERE DLocationName='"+ln+"' AND DateTime = '"+dt+"'

//If user add a datapoint with existed location name and date time, reject it and show error message, dt is the date and time input by user, ln is the location name selected by user.

    b. INSERT INTO DataPoint " + "VALUES (FALSE ,'"+dt+"','"+dv+"','"+ln+"','"+dy+"')

//If no duplicated happens, insert into DataPoint table a new DataPoint with dt(date and time), dv(data value), ln(location name) and dy(data type) input by user.

    c. SELECT DISTINCT(Type) FROM DataType

//Add data type into drop down box according to values database

    d. SELECT DISTINCT(LocationName) FROM POI

//Add location name into drop down box according to values in database

4. AddNewLocation Page
    a. SELECT * FROM POI WHERE LocationName='"+ln+"'

//Judge whether the new location has been already existed in database, ln is the location name input by users.

    b. INSERT INTO POI " + "VALUES (null,false,'"+ln+"','"+zip+"','"+ct+"','"+st+"')

//Insert new location into POT table with values of DateFlaged(null), Flag(null, which means pending), location name (ln), zipcode(zip), city(ct), and state(st)

    c. SELECT City FROM CityState WHERE State = '"+state+"';

//Filter cities added into drop down box, according to the state that already been chosen.

    d. SELECT DISTINCT(State) FROM CityState

//Add states into drop down box according to the states in database.

5. PendingAccounts Page
    a. SELECT CityOfficial.Approved, CityOfficial.CUserName, user.EmailAddress, CityOfficial.CCity, CityOfficial.CState, CityOfficial.Title FROM CityOfficial, user WHERE CityOfficial.Approved IS NULL AND CityOfficial.CUserName = user.UserName;

//Select approved status, username, email address, city, state, title of city officials whose approved status is null (which means pending) when viewing all pending city official accounts.

    b. UPDATE CityOfficial SET Approved = FALSE WHERE CUserName = '" + user_name + "';

//When a row is selected, change that city official's approved status to false if it is rejected.

    c. UPDATE CityOfficial SET Approved = TRUE WHERE CUserName = '" + user_name + "';

//When a row is selected, change that city official's approved status to true if it is approved.

    d. SELECT Approved, CUserName, EmailAddress, CCity, CState, Title FROM CityOfficial, user WHERE Approved IS NULL AND CUserName = UserName;

//Select approved status, username, email address, city, state, title of city officials whose approved status is null (which means pending) when the content of the table is updated.

6. PendingDataPoint Page
    a. SELECT * FROM DataPoint WHERE Accepted IS NULL;

//Select accepted status, datetime, datavalue, DlocationName, DType of data point whose accepted status is null (which means pending) when viewing all pending data points.

    b. UPDATE DataPoint SET Accepted = FALSE WHERE DLocationName = '" + location_name + "' and DateTime = '" + date_time + "';

//When a row is selected, change its accepted status to false if it is rejected, which corresponds to the data point with same combination of location name and date time.

c. UPDATE DataPoint SET Accepted = TRUE WHERE DLocationName = '" + location_name + '" and DateTime = '" + date_time + '";
//When a row is selected, change its accepted status to true if it is accepted, which corresponds to the data point with same combination of location name and date time.

d. SELECT * FROM DataPoint WHERE Accepted IS NULL;
//Select accepted status, datetime, datavalue, DlocationName, DType of data point whose accepted status is null (which means pending) when the content of the table is updated.

7. PendingFunction Page
No SQL queries!!

8. DataTypenew Page

a. state_findpoi = "SELECT datavalue,dtype,DateTime FROM DataPoint WHERE DLocationName = '"+location+"' AND Accepted = " True ";
//state_findpoi is a string in Java, and we add more queries on it based on many judgements, after finishing these judgements, we get the final query to be executed.

b. state_findpoi = state_findpoi;
//We don't need to add more queries on the state_findpoi since DataType is empty.

c. state_findpoi = state_findpoi + "AND dtype = '"+ s_type + "' ";
//We need to add more queries on the state_findpoi since DataType has value.

d. state_findpoi = state_findpoi;
//We don't need to add more queries on the state_findpoi since both DataValue are empty.

e. state_findpoi = state_findpoi + " AND datavalue >= '" +l_value+ "' ";
//We need to add more queries on the state_findpoi since left DataValue has value .

f. state_findpoi = state_findpoi + " AND datavalue <= '" +r_value+ "' ";
//We need to add more queries on the state_findpoi since right DataValue has value .

g. state_findpoi = state_findpoi + "AND datavalue <='" + r_value + "' AND datavalue >= '"+l_value+"'";
//We need to add more queries on the state_findpoi since both DataValue have value and the left value is smaller than right value .

h. state_findpoi = state_findpoi;
//We don't need to add more queries on the state_findpoi since start DateValue is empty.

i. state_findpoi = state_findpoi + " AND DateTime > '"+ startdate3 +"'";
//We need to add more queries on the state_findpoi since start DateValue has value.

j. state_findpoi =state_findpoi;
//We don't need to add more queries on the state_findpoi since end DateValue is empty.

k. state_findpoi = state_findpoi + "AND DateTime <'"+ enddate3+"'";
//We need to add more queries on the state_findpoi since end DateValue has value.

l. state_findpoi = state_findpoi + "ORDER BY DateTime;";
//We need to add more queries on the state_findpoi since we need to order the result by DateTime.

m. state_findpoi = "SELECT datavalue,dtype,DateTime FROM DataPoint WHERE DLocationName = '"+location+"' AND Accepted = " True ";
//We repete the queries since we need to refresh the jPanel.

n. state_findpoi = state_findpoi;

o. state_findpoi = state_findpoi + "AND dtype = '"+ s_type + "' ";

p. state_findpoi = state_findpoi;

q. state_findpoi = state_findpoi + " AND datavalue >= '" +l_value+ "' ";

r. state_findpoi = state_findpoi + " AND datavalue <= '" +r_value+ "' ";

s. state_findpoi = state_findpoi + "AND datavalue <='" + r_value + "' AND datavalue >= '"+l_value+"'";

t. state_findpoi = state_findpoi;

u. state_findpoi = state_findpoi + " AND DateTime > '"+ startdate3 +"'";

v. state_findpoi =state_findpoi;

w. state_findpoi = state_findpoi + "AND DateTime <'"+ enddate3+"'";

x. state_findpoi = state_findpoi + "ORDER BY DateTime;";

y. "UPDATE POI SET Flag = 1,DateFlaged = '"+dateflag+"' WHERE LocationName = '"+location+"';"//Update a location with flag

z. "UPDATE POI SET Flag = 0,DateFlaged = NULL where LocationName = '"+location+"';"//Update a location with unflag

9. ChooseFuntionallynew Page
No SQL queries!!

10. ViewPoinew Page

a. SELECT DISTINCT(LocationName) FROM POI
//We need to fill the LocationName Box with the data.

b. SELECT DISTINCT(PCity) FROM POI
//We need to fill the City Box with the data.

c. SELECT DISTINCT(PState) FROM POI
//We need to fill the State Box with the data.

d. sql = SELECT LocationName, PCity, PState, Zipcode, Flag, DateFlaged FROM POI WHERE Flag = " + flag + ";
//sql is a string in Java, and we add more queries on it based on many judgements, after finishing these judgements, we get the final query to be executed.

e. sql = sql + "AND LocationName = '" + locationname + "' ";
//We add more queries on sql since LocationName is one Judge condition.

f. sql = sql + "AND PCity = '" + city +"' ";
//We add more queries on sql since City is one Judge condition.

g. sql = sql + "AND PState = '"+ state + "' ";
//We add more queries on sql since State is one Judge condition.

h. sql = sql + "AND Zipcode ="+ zipcode + "";
//We add more queries on sql since Zipcode is one Judge condition.

i. sql = sql + " AND DateFlaged > '"+ startdate +"'";
//We add more queries on sql since startdate is one Judge condition.

j. sql = sql + "AND DateFlaged <'"+ enddate+"'";
//We add more queries on sql since enddate is one Judge condition.

k. sql = SELECT LocationName, PCity, PState, Zipcode, Flag, DateFlaged FROM POI WHERE Flag = " + flag "

//We repete the queries since we need to refresh the jPanel.

l. sql = sql + "AND LocationName = '" + locationname + "' ";
m. sql = sql + "AND PCity = '" + city +"' ";
n. sql = sql + "AND PState = '"+ state + "' ";
o. sql = sql + "AND Zipcode ="+ zipcode + "";
p. sql = sql + " AND DateFlaged > '"+ startdate +"'";
q. sql = sql + "AND DateFlaged <'"+ enddate+"'";
r. sql = sql + ";";
s. SELECT Distinct DType FROM DataPoint WHERE DLocationName = '"+location+"';

11. PoiReportnew Page

a.

```
CREATE VIEW aq_report AS SELECT LocationName,PCity,PState,Avg(DataValue) as
AQ_avg_value,Max(DataValue) AS AQ_max_value,DType,Flag,COUNT(*) AS
record_num,Min(DataValue) AS AQ_MIN FROM DataPoint AS d1, POI AS d2 where
d1.DLocationName = d2.LocationName AND d1.DType = "Air Quality" AND d1.Accepted
= true GROUP BY d2.LocationName;
```

//Create a view to show location name,city,state, related,datavalue infromation of data type Air Quality.

b.

```
CREATE VIEW mold_report AS
SELECT LocationName,PCity,PState,Avg(DataValue) AS
Mold_avg_value,Max(DataValue) AS Mold_max_value,DType,Flag,COUNT(*) AS
record_num,Min(DataValue) AS Mold_MIN FROM DataPoint AS d1, POI AS d2 WHERE
d1.DLocationName = d2.LocationName AND d1.DType = "Mold" and d1.Accepted = true
GROUP BY d2.LocationName;
```

//Create a view to show location name,city,state, related,datavalue infromation of data type mold.

c.

```
CREATE VIEW numcount AS SELECT DLocationName, count(*) AS
number_of_records FROM  DataPoint GROUP BY DLocationName;
```

//Inorder to calculate the number of records of different locationname

d.

```
SELECT
r1.LocationName,r1.PCity,r1.PState,Mold_avg_value,Mold_max_value,Mold_MIN,AQ_a
vg_value,AQ_max_value,AQ_Min,n.number_of_records AS record_num,r1.Flag FROM
aq_report AS r1 left JOIN mold_report AS r2 ON r1.LocationName = r2.LocationName
Left JOIN numcount AS n ON n.DLocationName = r1.LocationName
UNION
SELECT
r2.LocationName,r2.PCity,r2.PState,Mold_avg_value,Mold_max_value,Mold_MIN,AQ_a
vg_value,AQ_max_value,AQ_Min,n.number_of_records AS record_num,r2.Flag FROM
aq_report AS r1 right join mold_report AS r2 ON r1.LocationName = r2.LocationName
LEFT JOIN numcount AS n ON n.DLocationName = r2.LocationName;
```

//select results according to different datatype. And join the two views to create the expected result