Environmental Monitoring System

Project Report

Group Members

Arachchi R W M 100039D
Bandara K A M N 100056C
Bashani J P S 100059M
Pemasiri H T A S 100374A

This is submitted as a partial fulfillment for module CS 3042 – Database System.
Department of Computer Science and Engineering.
University of Moratuwa.
March 23, 2013

DECLARATION

We hereby declare that the project report entitled “Environmental Management System” submitted for the course module CS3042 is a record of an original work done by us under the guidance of Dr.Shehan Perera. We have done our project work adhering to the Honor code of the Department of Computer Science and Engineering, Faculty of Engineering, University of Moratuwa.

ACKNOWLEDGEMENTS

We sincerely acknowledge with so much contentment and gratitude the assistance and the supervision given every person right through this project to make it a success. Specially to start with Dr.Shehan Perera, Senior Lecturer, Department of Computer Science, University of Moratuwa, for his continuous support , guidance and administration throughout the project.

We also like to express our sincere gratitude to Mr. Charith Chithranangan, lecturer Department of Computer Science and Engineering, University of Moratuwa for the assistance provided.

At last we would like to be grateful to the all the people who helped us in various capacities to make this project a success. .

TABLE OF CONTENT

1. INTRODUCTION

This is on an application which has been developed to support the Colombo city mayor to analyze the environmental condition in the city according to the data collected by the sensors located in several locations in the city.

1.1 The project Client

The main client of the project is the Colombo city mayor. The responsible of the city Mayor is to serve the citizens of his municipal area, uplift the living condition of them and to work towards the betterment of the whole city.

More over being the Mayor of Colombo, our client has to contribute to the national occasions and he should be able to create a positive impression on the visitors of the city as Colombo is the commercial capital of Sri Lanka.

The mayor expects to get a better understanding about the city through the newly implemented system.

1.2 The Business Case

1.2.1 Objective

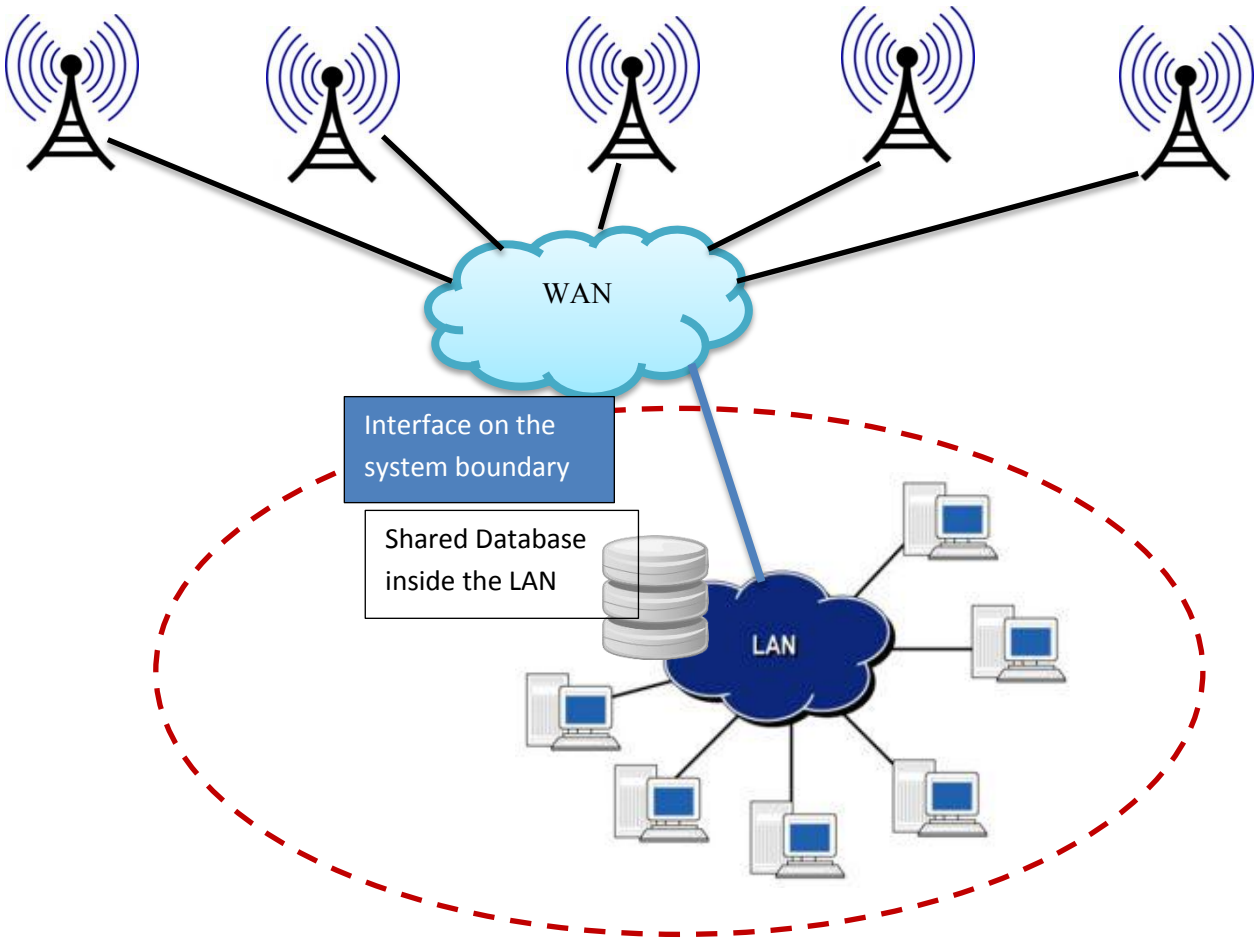
The objective of the new Environment Monitoring System (EMS) is to provide efficient, complete, comprehensive, integrated, "frond-end" information management system to the Mayor and other authorized parties so that they can get an understanding of the environmental condition of the area both in the long run and short run.

1.2.2 Business Benefits

- Make efficient data collection and manipulation
- Identify the possible hazards as soon as possible
- Observe unusual temperature variations and Co2 % variations.
- Respond to emergencies.

1.3 Project Scope

The scope of the system spans though the back end data base to the application level. The control and the transmission of data between the proposed system and the sensors is out from the scope of this project. But it is a responsibility of the system to create an adopter which acts as an interface in the system boundary.



1.4 Functionality Requirements of the system

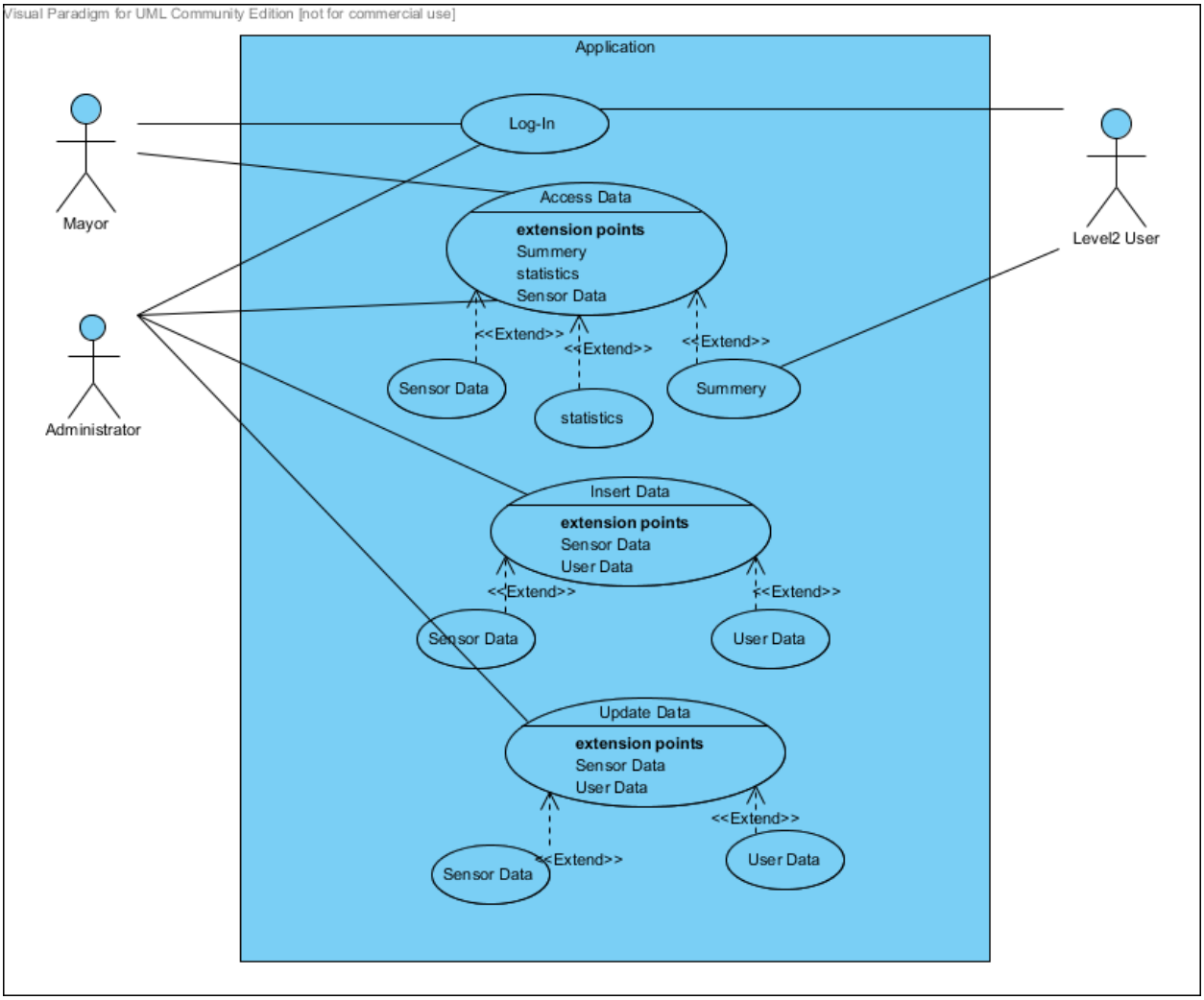
- Capability of registering a new user for the system while assigning him a privilege level according to the designation.
- View the current environment condition for the users according to the demand
- View the average values of environmental parameters according on demand of the users
- Generating graphs and tables for the user
- Capability of identifying the failed sensors.
- Provide details of the sensors along with the details of the particular service providers, agents and contract details

1.5 Non- functional requirements

- The users of the system should authenticate themselves before they login to the system.
- Capability of editing sensitive data should be there only for highly privileged parties
- The system should be able process 10 queries within a second
- A user should be able to adopt to the system after 5 hour training sessions
- The system should be capable of providing service all 24 hours of the day and the average downtime shall not exceed 5 minutes in any one day.
- In case if the system goes down, the restart within two minutes should be possible.

- The system should encrypt the sensitive data when storing.
- System should provide a comprehensive error messages to users describing errors occurred during a process to take necessary actions according to the level of control of the user.

1.6 High level use case diagram of the required system



2. DESIGN OF THE SYSTEM

The requirement specification and the requirement analysis revealed that the required system is operational application which interacts with the users. Moreover this is a system which operates within an organization that is connected by a Local Area Network.
So the system is developed in such a way that it shares a central database through a LAN.

At designing perfect technology assumption was made.

The system will be implemented using the client server architecture. All the users are allowed to use their desktop computer to login to the system. It will connect to the server over the intra-net. The application server will connect to the database server in order to access the data.

3. DATABASE DESIGN

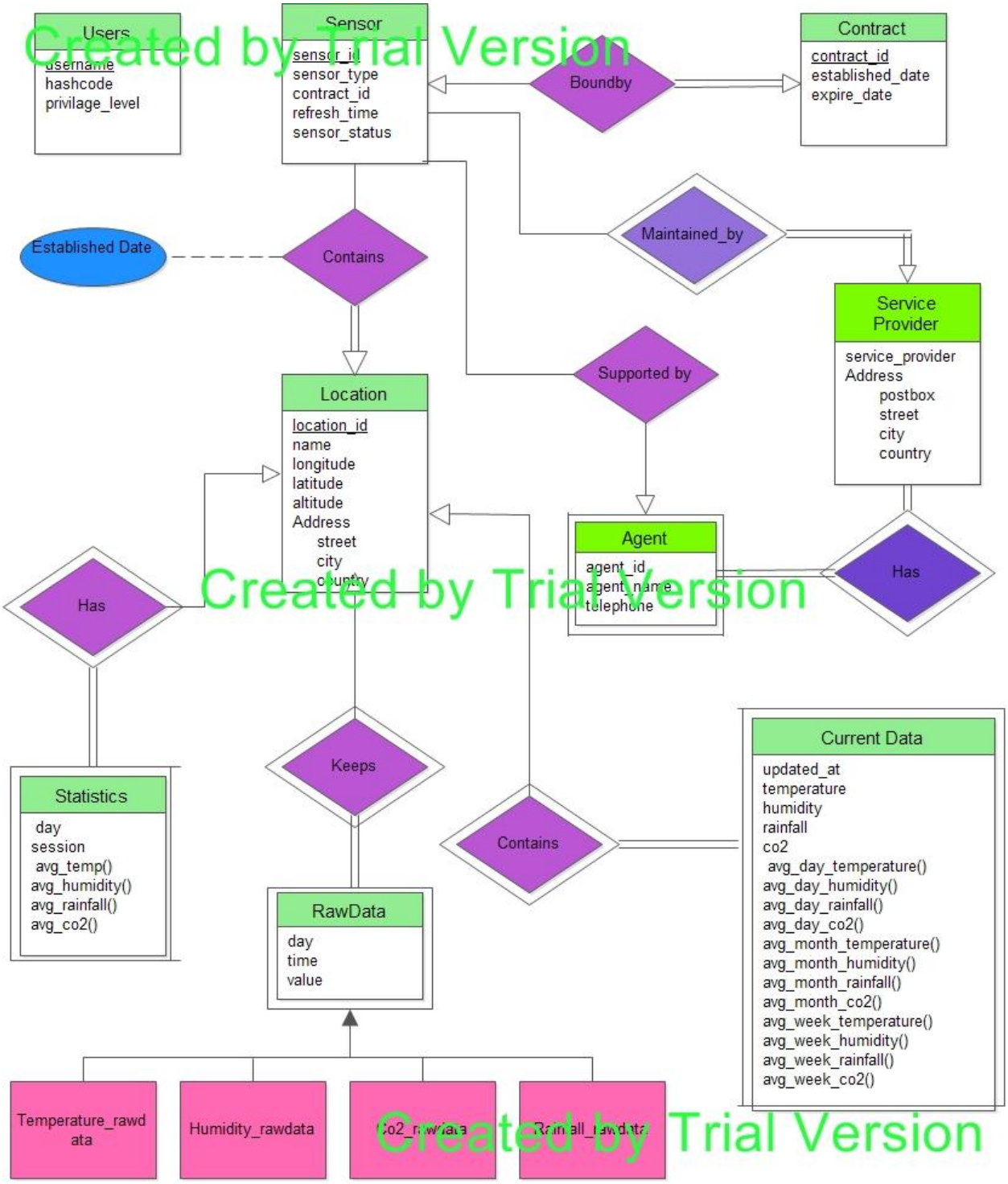
3.1 ER/ EER Diagram

To get an abstract view of the domain and the structure of the environment the ER/EER diagram was drawn.

3.1.1 Design decisions of the ER diagram

- ”User” was identified as a separate entity which is not related with any other entity. The purpose of having this entity is to provide a proper authentication for the system.
- The “Sensor” entity is related with the “Contract” entity. Participation of “Contract” is total on “Bounded by”. This is because in the given domain though a sensor can exist without a contract; there is no point of having a contract without having a sensor. A contract is related to A sensor.
- Each “Sensor” is “Maintained by” a “Service Provider”. For the proposed system there is no use of having a “Service provider” who is not related to any sensor. So participation of “Service provider” in the relationship “Maintained_by” is total.
- “Agent” represents each individual who takes care of the sensors through a “Service Provider”. In the given context an “Agent” can be a member of one or more “Service Provider”s. So the primary key of “Agent” consist the primary key of “Service Provider” as well as “agent_id”, “Agent” acts as a weak entity in the domain.
- The system collects data from several “Location”s. A “Location” may contain one or more “Sensors”. For the domain, storing “Location”s without a “Sensor” would be a waste of effort. So on the relation “Contain”, participation of “Location” is total.
- “Location”, keeps track of “Statistics” , “Raw Data” , data which is retrieved by the sensors and “Current Data”.
- “Location” keeps “temperature_rawdata” , “c02_rawdata”,”humidity_rawdata” and “Rainfall_rawdata” entities . Though they are different entities they share the same features(day,time.value). So we combined these entities through specialization into a higher-level entity called “raw_data” .

3.1.2. ER/EER Diagram of the system



3.2. Database Schema

A reduction of ER/EER diagram to the database schema was the next step.

3.2.1. Database Schema for the system

- Users(username,hashcode,privilege_level)
- Location(location_id,name,longitude,latitude,altitude,street,city,country)
- Serviceprovider(id,{address},e-mail)
- Agent(agent_id,agent_name,service_provider,tp_no,address)
- Contract(contract_id,survice_provider,agent_id,established_date,expire_date)
- Sensor(sensor_id,sensor_type,contract_id,refresh_time)
- Contains(location_id,sensor_id,established_data)
- Temperature_rawdata(location_id,day,time,value)
- humidity_rawdata(location_id,day,time,value)
- co2_rawdata(location_id,day,time,value)
- rainfall_rawdata(location_id,day,time,value)
- Statistics(location_id,day,session,avg_temp,avg_humidity, avg_rainfall,avg_co2)
- Current_data(location_id,updated_at, teprature, humidity, rainfall,co2_percentage,

  avg_day_temp, avg_ day_humidity, avg_ day_rainfall, ,avg_ day_co2_percentage,

  avg_month_temp, avg_month_humidity, avg_month_rainfall,

  avg_month_co2_percentage, avg_year_temp, avg_year_humidity, avg_year_rainfall,

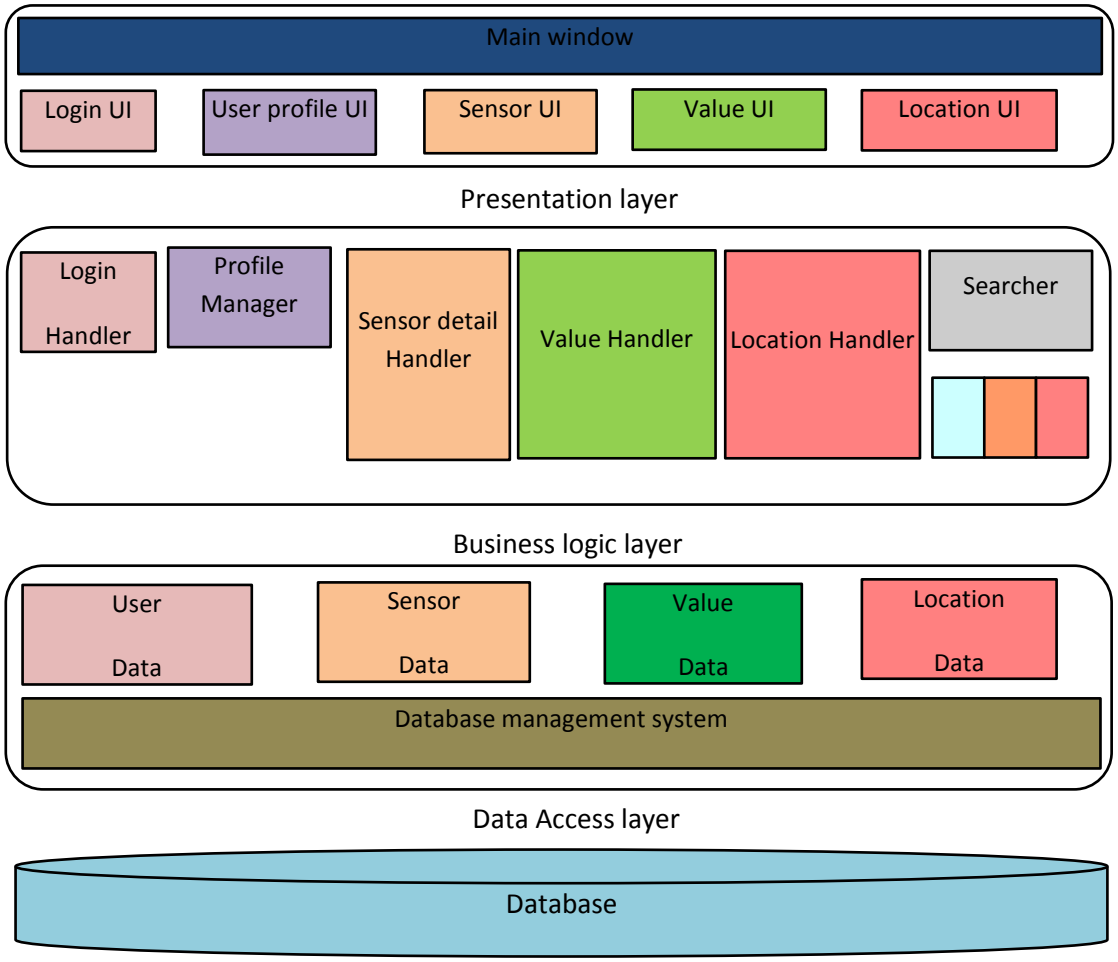
  ,avg_year_co2_percentage)

3.2.2 Design decisions of the database schema

- The EMS user details are stored in USER table.
- Location table keeps the details of locations where sensors are established with data constrain added to variables (longitude, latitude, altitude) so that the location is ensured to be within city boundary.
- Service provider details are stored in separate table. In all the schemas the composite attribute "address" is stored in the same table as the addresses are unique and so that it would not add redundancy.
- Separate table "Agent" is created to store multiple agents of service providers. On the other hand an agent can work under one or many service providers. So both agent_id and service_provider_id makes the primary key for this schema. Moreover having multivalued attributes in a separate table "service providers" and "agent" completes the 4th normal form.
- The contract schema stores details of the contracts established with the service providers through agents.
- The sensor details are stored in "sensor" table and the relationship between sensors and their maintaining agents are linked by the foreign key "contract_id".
- Then separate schemas are designed for each type of data such as "temperature_rawdata", "humidity_rawdata".
- Our initial decision was to store all data collected from one sensor station to be stored in one table, creating raw data tables for each location in a schema of location_id_rawdata(teprature, humidity, rainfall, co2_percentage).
- But as the refresh time of each sensor is different (for example though temperature differs so often, even once a second, other factors such as CO2 percentage or humidity is updated less often) in that design if a raw is added at each update the data updated at less frequent may contain null values or repeated same data values which consumes unnecessary memory.
- The functional dependencies existing in the system are decomposed in the form BCNF.
- This decision reduces the cost of calculating average values of each attribute in the Colombo city as well because instead of having expensive joins we can retrieve values of each data type from one place.
- The current_data schema is different and it does not adhere to the normal forms. But we identified the crucial need of this schema when analyzing functional and non-functional needs of the user. In order to reduce operation cost, computational overhead and increase the response time for user this schema is important. This contains the data accessed by majority of users most often. Though this schema is redundant to the database (the data in this schema can be retrieved or calculated from data contain in other tables), as this table is of exact size, and does not grow unless new sensor types or locations added that redundant is negligible in comparison to the increased efficiency

4. IMPLEMENTATION

4.1. Layered Architecture for the system



Layered architecture for the system.

4.2. Implementation Plan

After the completion of design phase we divided the project implementation into phases and specify a schedule for each phase including the following steps for each phase.

- Coding
- Unit testing
- Integration
- Integration testing

Here we basically divided implementation into three parts.

1) Presentation Layer
2) Business Logic Layer
3) Data Access Layer

4.3. Implementing Presentation Layer and Business Logic Layer

Both the presentation layer and the Business logic layer were implemented by using C#.net frame work. For the presentation layer implementation the package Windows.Form was used.

Here we implemented the user interfaces

1) Based on user
    - Administrator
    - Mayor
    - Normal user
2) Based on type of interface
    - Login interfaces
    - Data accessing interfaces
    - Data editing/updating interfaces

Business Logic layer which connected to the presentation layer through a façade was also implemented using .net frame work.
Pre-developed components of this frame work made the development easier and efficient.

4.4. Implementing application backend

Under this part we do the database implementation. Here we did coding and testing for sql queries for

- inserting data,
- editing/updating data
- retrieving data

This the most important part of the implementation since whole system going to deal with the data which are store in database.

For implementation of the application back end we use following resources.

❖ Tools

- MySQL on Windows Apache, MySQL ,php/perl/python server

Behind the many reasons of using MySQL over other data base management systems its ability of cross functional operability through providing APIs , the embedded capabilities of security handling , concurrency controlling and other features like triggering, data encrypting were taken in to the consideration.

4.5. Testing

It is important to ensure the system does what we expect before we use it in a 'live' environment. So we have to do testing in implementation. The testing is done in two stages.

1) Unit testing
2) Integration testing
3) Usability testing

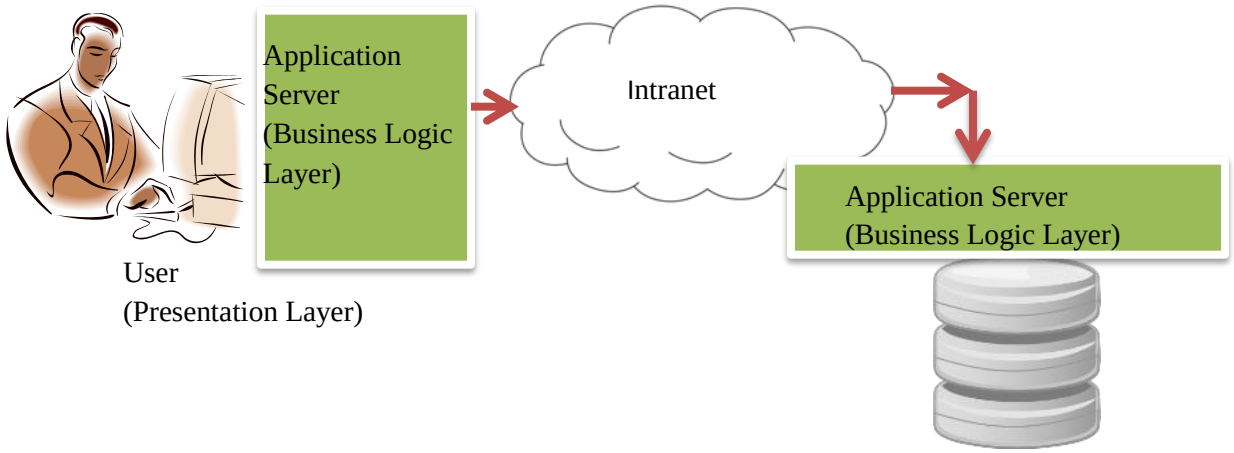
Product testing is done in two steps

1) Unit testing
   Unit testing was done for individual components. Here we did testing for application front end, application back end as well as the business logic layer.
   Application Backend
   Did testing for the sql queries written for the database for Inserting data, deleting data editing data, retrieving data.
   Application Front end
   Did testing for the user interfaces.
   - Based on user (admin, user who access data, etc)
   - Based on type (login interfaces, data accessing interfaces, data editing/updating interfaces)

2) Integration testing
   After integrating all the unit tests above separately, integration tests were carried for whole system to check whether it is able to fulfill the expected functional requirements.

5. DEPLOYMENT

5.1. Deployment diagram



5.2. Database tuning

At the deployment considering the real world status triggers and schedules were made for the automation of updating.

5.2.1. Triggers

Triggers were made to update the corresponding values of the table “current_data”. This will eliminate the problems that occur due to the various response times of various types of sensors.

Example of such an update is that updating humidity of the “current_data” table when a new raw is entered to the “humidity_rawdata” table. The trigger can update all the related values for the attribute “humidity”

Eg: “update_humidity” trigger

```
CREATE TRIGGER update_humidity AFTER INSERT ON humidity_rawdata FOR
EACH ROW
UPDATE current_data SET humidity=NEW.value,updated_at=NEW.time,
avg_day_humidity=(SELECT avg(value) from humidity_rawdata
where humidity_rawdata.day>DATE_SUB(new.day,INTERVAL 24 HOUR) or
(humidity_rawdata.day=DATE_SUB(new.day,INTERVAL 1 DAY) and
humidity_rawdata.time<CURTIME)),
avg_week_humidity=(SELECT avg(value) from humidity_rawdata
where humidity_rawdata.day>DATE_SUB(new.day,INTERVAL 7 DAY)),
avg_month_humidity=(SELECT avg(value) from humidity_rawdata
where humidity_rawdata.day>DATE_SUB(new.day,INTERVAL 30 DAY))
where location_id=NEW.location_id;
```

In the same way triggers are used for other attributes as well.

5.2.2. Schedulers

For the updates that should be carried out over time period repeatedly the event schedulers are used which are awoken according to the time. There are many such jobs in the EMS.

For an example to update the in the “Statistics” table a schedulers have been used which calculate average values over two times a day.

Eg:

```
DELIMITER $$
CREATE EVENT update_statistics1
ON SCHEDULE EVERY 24 HOUR STARTS '2013-03-22 12:00:00'
ON COMPLETION PRESERVE
DO BEGIN
INSERT INTO statictics VALUES (CU RDATE(),1,null,'l1',null,null,null);
UPDATE statictics as s SET
statictics.avg_temp=(select avg(value) from temperature_rawdata where
temperature_rawdata.day=CURDATE() and temperature_rawdata.time<CAST('12:00:00' AS
time) ),
statictics.avg_humidity=(select avg(value) from humidity_rawdata where
humidity_rawdata.day=CURDATE() and humidity_rawdata.time<CAST('12:00:00' AS time) ),
statictics.avg_co2=(select avg(value) from co2_rawdata where co2_rawdata.day=CURDATE()
and co2_rawdata.time<CAST('12:00:00' AS time) ),
statictics.avg_rainfall=(select avg(value) from rainfall_rawdata where
rainfall_rawdata.day=CURDATE() and rainfall_rawdata.time<CAST('12:00:00' AS time) )
WHERE  s.day=CURDATE() AND s.session=1 AND s.location_id='l1';

#same insert and update statements for all the locations changing the value of location_id
END$$
DELIMITER ;
```

5.2.3 Batch Updates

The sensors provide continues output of raw data and refresh time differ from sensor to sensor. But reading data from sensors at each change arouse overhead as there are many sensors and one or the other may change at any time. So we tuned the database as such that all sensor data is updated at once every half an hour. This is done as a “batch update”. So number of inserts are performed at a single communication with the database, which is much efficient than list of single updates.

Eg:

```
PreparedStatement pStmt = (PreparedStatement) conn.prepareStatement("insert into temperature_rawdata values(?,?,?,?)");
pStmt.setString(1, "L4");
pStmt.setDate(2, watch.getDate());
pStmt.setTime(3, watch.getTime());
pStmt.setFloat(4, (float) buffer.getData(4));
System.out.println(pStmt);
pStmt.executeUpdate();
pStmt.addBatch();
:
:

pStmt.executeBatch();
```

If latest data is needed, accurate less than 30 minutes, then functionalities are provided such that user can force update.

5.2.4 We modified the queries used to calculate average values, which containing “where” clause to have “group by” clause. So multiple queries are reduced in to one, saving the execution cost.

5.2.5 Nested sub queries were identified to be very expensive, and cannot be optimized by the automatic query optimizers are available. So when tuning, nested sub queries are reduced and avoided as much as possible.

5.2.6 Indexes are created for the date and time columns in all data tables, as well as for the location_id on the statistics table, because we access data grouped by date, time or location very often.

5.2.7 Windowing is used to calculate moving averages of data. Some of these calculations were triggered at insert statements, so latest data is used in calculating averages and data is precise at the best level possible.

5.2.8 Though the authorization of the EMS system is established through application level, it is implemented at the database level as well, because the database can be shared, or accessed by another application interfaces over the time. There are 3 privilege levels enforced for the Administrator, Mayor and the Normal user.

6 FUNCTIONALITY OF THE SYSTEM

The parties that are allowed to access the system can be mainly divided in to three categories. The functionality provided to each category varies according to the privileges given to them.

6.2 Login functionality

This will help to log the user to the system while making sure the proper authentication.

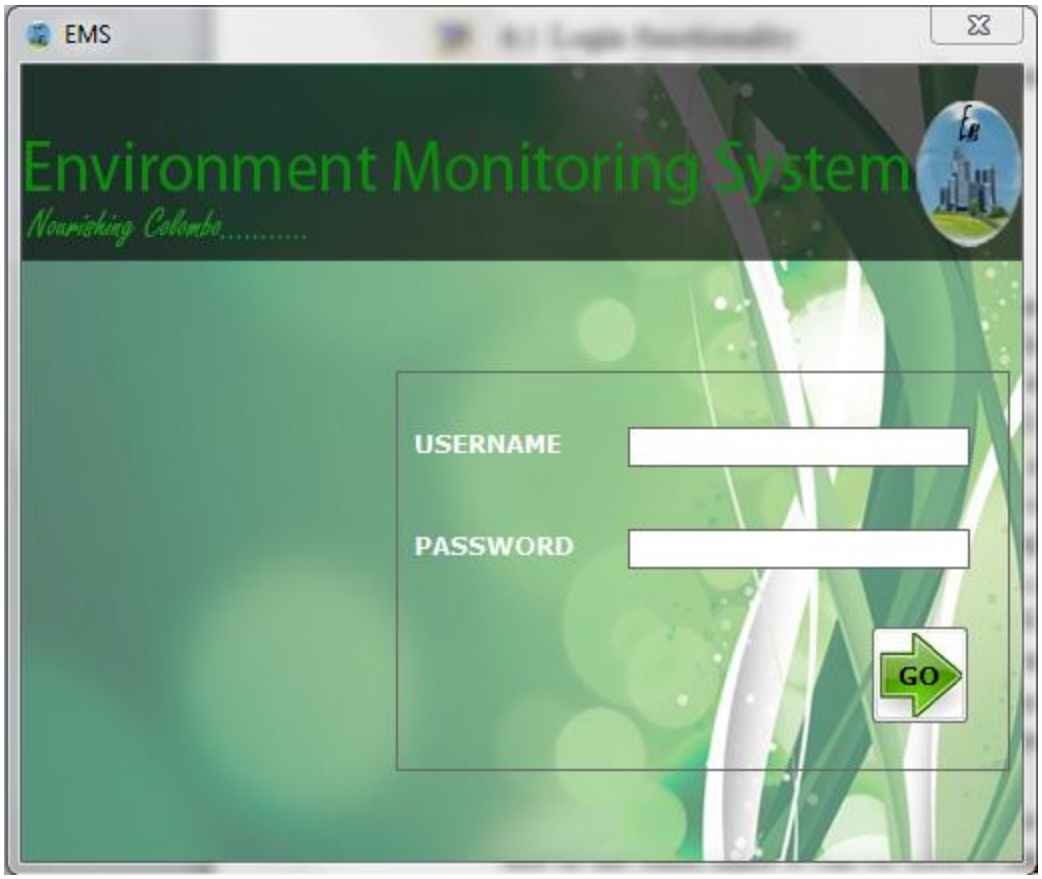


Figure a

The users are capable of login to the system by entering their username and the password (Figure a). If the user name and the password do not match with each other the system will asks to reenter the password (Figure b)

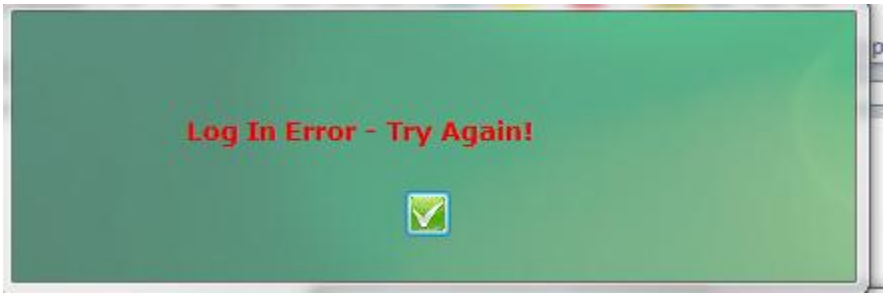


Figure b

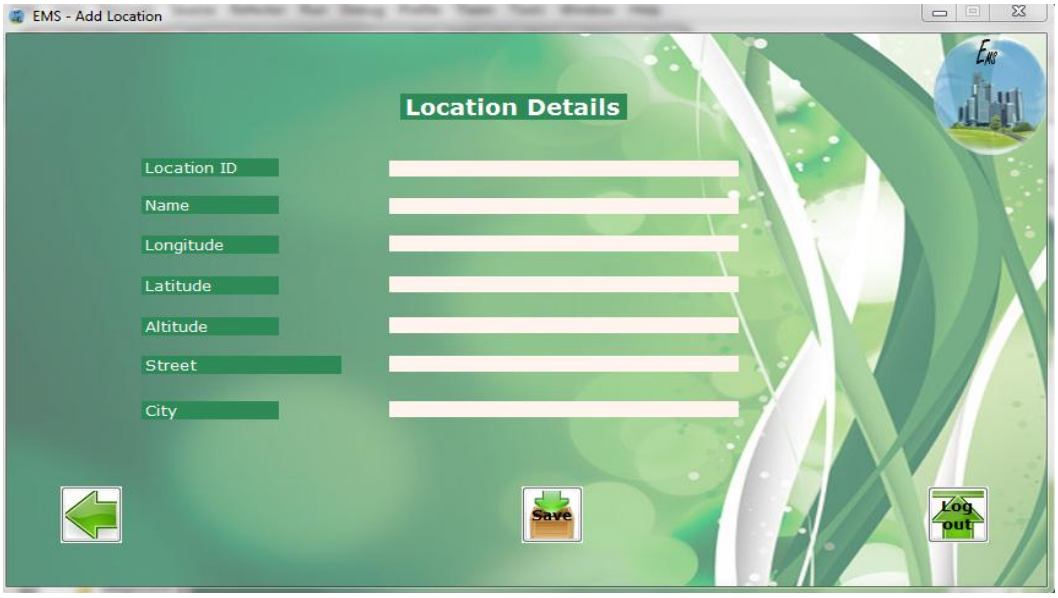
6.3 Functionality for the Admin

After a successful login, the admin is provided by the window as follow, which he can edit, delete, view and add data.

6.2.1. Amending the details by the admin

6.3.4 Once the admin select an option from the control box in the left side and "ADD DATA" from right side, he will be provided with a window appropriate for that purpose.

For an example selecting "LOCATION" along with "ADD DATA" directs the admin to a window as follow. In the same way the admin is able to view data and edit data



6.3.

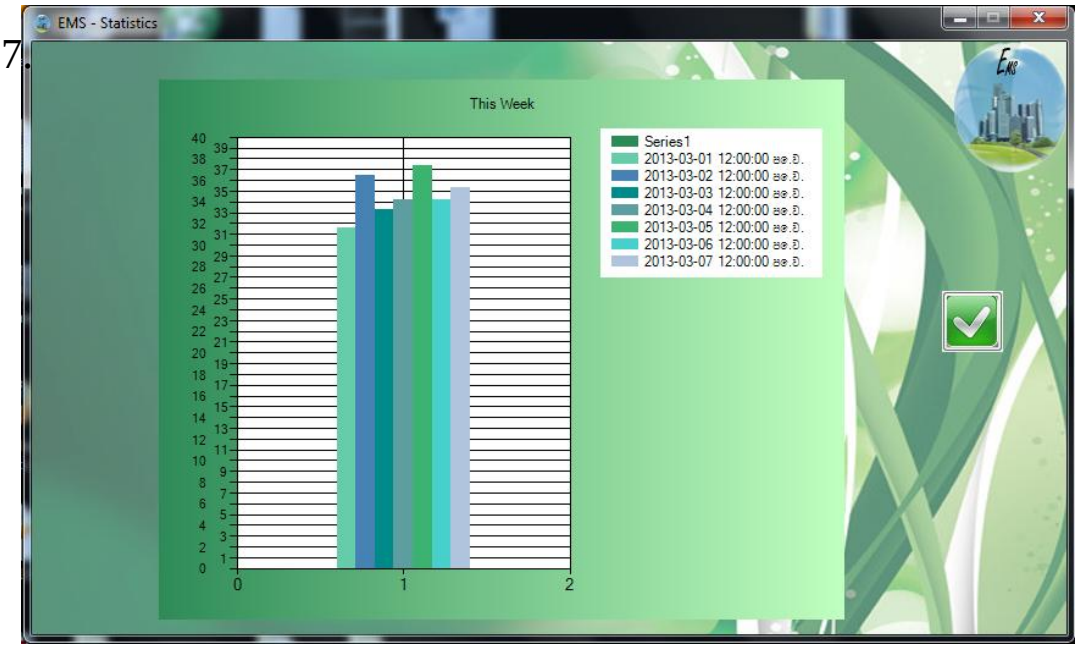
Functionality for the Mayor

The mayor is able to view all the data that can be seen by the admin. The mayor is capable of having all the functionalities as same as the admin except the viewing user details adding, editing and deleting details.

6.4 .Functionality for the normal user

The normal user is able to view all the data which can be seen by the mayor except the sensor details.

7.



7. FUTURE WORKS

- This is developed on limited number of sensors situated in limited number of stations. It can be expanded for more locations and also same location may contain several sensors of same type so that accuracy is reinforced and failure of one sensor would not affect the system data.
- A map can be used to select locations and to show data location vice.
- The data can be used for data mining and predicting of patterns in weather and other environmental factors. This can be used to observe temperature variation over time as well as over the locations. Even maps with geographical values can be generated accordingly.
- This system is identified to have advantages even for the mass community, not only the mayor. So the application can be widespread over World Wide Web and can provide access to public and media and Colombo citizens can retrieve much accurate location based data.
- This system can be expanded in to other cities and provinces of the country and combine system with department of meteorology to have a proper environmental monitoring system which is still lacking.
- EMS mobile applications can be extended so even a mobile user can monitor the environment and in the other hand it can be used to implement an alerting mechanism about undesirable environmental changes to the authorities.

7. DISCUSSION

This is a client based application which connects to a backend database through a server. So database can be situated remotely even though we host it at local host for demonstration purposes. Basically the problems encountered were related to technical issues. Deciding an implementation platform and learning and familiarizing with it were the hardest phase. We were able to continue with the initial database design from the beginning without changing it immensely. ER diagrams and use cases were clarifying enough. From the beginning the design was less redundant adhering to the 3rd normal form at the level best.

Finally the conclusion is that the implemented application is sufficient and success for the defined scope, but this can be extended and scaled further. The database is designed to adapt to such possible scalability.

8. REFERENCES

- Database System Concepts, 6th Edition, S. Sudarshan, Abraham Silberschatz, Henry F. Korth
- MySQL :: MySQL 5.1 Reference Manual
- Msdn.microsoft.com