



ISP CLIENT MANAGEMENT SYSTEM

Advanced Database Management System Project

Abstract

A database system for local ISP to maintain client information, billing and support.

Md. Sazid Hossain Banna

12-20245-1

Contents

Business & System Summary	2
Business Environment	2
Technical Summary.....	2
Project Objectives.....	3
Justification of database	3
Implementation	4
Code	8
Future Tasks:.....	9
Conclusion	9

Business & System Summary

This is an ISP client management database in which entities related to ISP is saved. The core entity of the system is a client. A client has address & connection details. Connection details contains IP configuration & package information. A client can submit ticket about a certain problem. A ticket can have issue type and issue description. For every month, a bill is auto-generated with bill ID against every clients in the client table. There is also a table which keeps track of any previous dues.

Local ISPs generally login to the router panel to find details about a certain user. Using this system will reduce the time to handle customer complaints and billings.

Business Environment

An ISP (Internet Service Provider) is a commercial company which provides internet connection and service to users. A small local ISP can have as low as thousand clients which can top to as high as five thousands. ISP will enter new client data in the system. Bills will be generated automatically at the start of each month. Authorized users can modify, view and edit client data. Support will handle clients' tickets.

Technical Summary

The main entity of the database is **CLIENTDETAILS**. It contains client addresses. A client has a **PACKAGE_DETAILS**. **PACKAGE_DETAILS** has one to many relationship with **CLIENTDETAILS**. A client can submit **TICKETS**. **COMPLAINDETAILS** holds the client complaints. A **COMPLAINLOG** keeps the record of resolve date. A **BILL_TRACK** tracks dues of every **CLIENT_ID** in the **CLIENTDETAILS** table.

Project Objectives

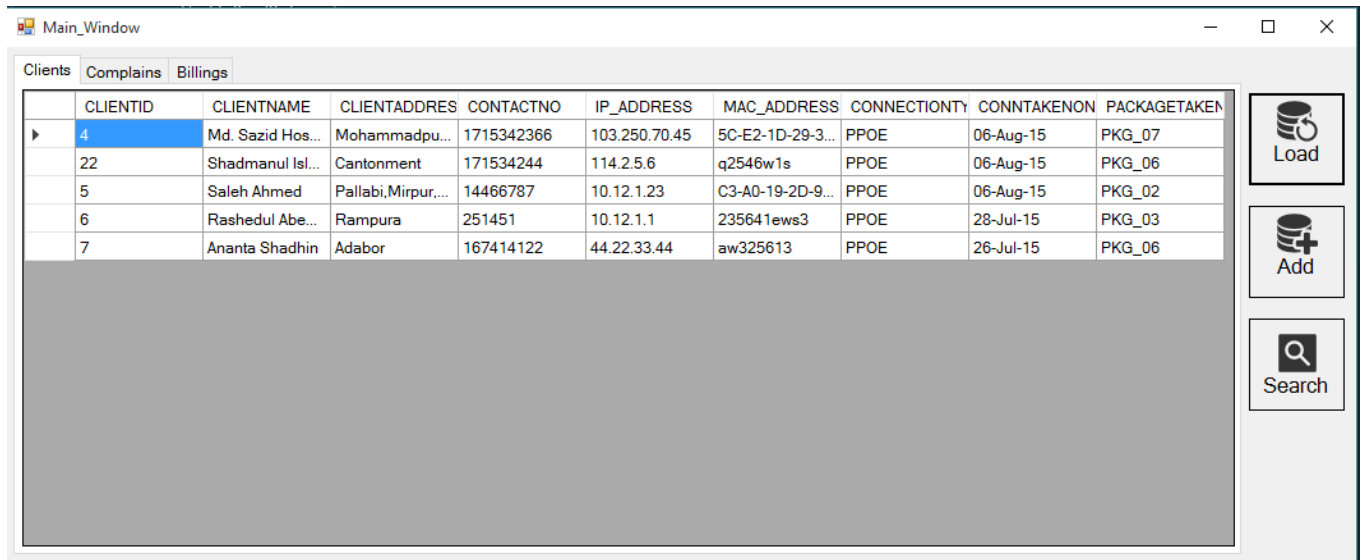
- Manage all client information easily.
- Manage billing system automatically.
- Keep track of system stability and condition via user support tickets.
- Make customer support more effective and organized.

Justification of database

In Bangladesh most of the local ISP companies doesn't use any form of application or database to manage their clients. They manually maintain their clients' information, support and billing. This is quite time consuming and inefficient. Lack of organized information handling results in poor support and management.

The proposed system will help to easily manage client details. The billing system will reduce human made errors which will reduce overall system loss. The support ticket subsystem will help ISP officials to solve client issues faster. It will also identify system structure issues from client feedback.

Implementation

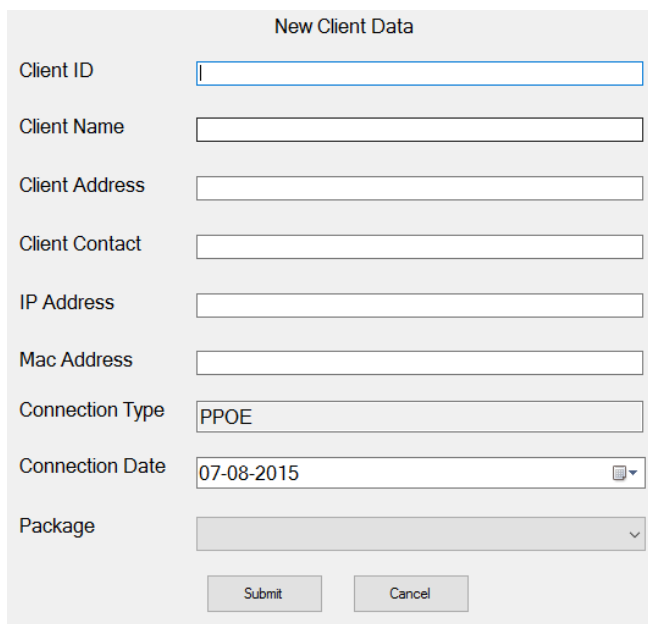


The screenshot shows a window titled "Main_Window" with three tabs: "Clients", "Complains", and "Billings". The "Clients" tab is active, displaying a table with the following data:

	CLIENTID	CLIENTNAME	CLIENTADDRESS	CONTACTNO	IP_ADDRESS	MAC_ADDRESS	CONNECTIONTY	CONNTAKENON	PACKAGETAKEN
▶	4	Md. Sazid Hos...	Mohammadpu...	1715342366	103.250.70.45	5C-E2-1D-29-3...	PPOE	06-Aug-15	PKG_07
	22	Shadmanul Isl...	Cantonment	171534244	114.2.5.6	q2546w1s	PPOE	06-Aug-15	PKG_06
	5	Saleh Ahmed	Pallabi,Mirpur,...	14466787	10.12.1.23	C3-A0-19-2D-9...	PPOE	06-Aug-15	PKG_02
	6	Rashedul Abe...	Rampura	251451	10.12.1.1	235641ews3	PPOE	28-Jul-15	PKG_03
	7	Ananta Shadhin	Adabor	167414122	44.22.33.44	aw325613	PPOE	26-Jul-15	PKG_06

On the right side of the window, there are three buttons: "Load" (with a database icon), "Add" (with a plus icon), and "Search" (with a magnifying glass icon).

1. *Main Window*: The main window consists of a single window with 3 tabs. On the right there are three universal buttons for LOAD, ADD and SEARCH. The buttons works only for the focused tab. Here in the picture, the focused tab is Clients and the load button loaded the client data in the data source. Double clicking on any of the cell of any of the row triggers Search/remove/modify form.



The screenshot shows a form titled "New Client Data" with the following fields:

- Client ID:
- Client Name:
- Client Address:
- Client Contact:
- IP Address:
- Mac Address:
- Connection Type:
- Connection Date: (with a calendar icon)
- Package:

At the bottom of the form are two buttons: "Submit" and "Cancel".

2. *Add Client*: Client ID is generated from sequence in the database. Rest of the form is self-explanatory.

Search, Remove or Modify

Search via Client ID

4

Submit Clear

CLIENTID	CLIENTNAME	CLIENTADDRESS	CONTACTNO	IP_ADDR
4	Md. Sazid Hossain	Mohammadpur, ...	1715342366	103.250.70.45

Client Data

Client ID: 4

Client Name: Md. Sazid Hossain Banna

Client Address: Mohammadpur, Dhaka

Client Contact: 1715342366

IP Address: 103.250.70.45

Mac Address: 5C-E2-1D-29-3B-AC

Connection Type: PPOE

Connection Date: 06-Aug-15

Package: PKG_07

Modify Delete

3. *Client Search, Remove or Modify*: The form loads the data from either from the main_window cell_click or the search_window cell_click. At initial stage modify button is disabled. Clicking on any button enables to modify the field and enables modify button while disabling Delete button.

Main_Window

Clients Complain Billings

COMPLAIN_ID	CLIENTID	PROBLEMDETA	PROBLEMSTATI	PROBDATE
6-070815-0838	6	Random loss	Resolved	07-Aug-15 8:3...
4-070815-0838	4	Random loss	Initiated	20-Aug-15 8:3...

Load

Add

Search

4. *Complain Main_window*: Works the same way as clients. Complain have three states. Initiated, Unresolved & Resolved. A complain either can be resolved or unresolved.

Add_Complaint

Complaints Data

Complain ID: 4-080815-0339

Client ID: 4

Problem Details: Ping spike

Problem Status: Initiated

Date: 08-08-15: 3-39

Submit Clear

5. *Add Complain*: Self-explanatory form fields. Complain ID is auto generated with client ID and datetime of that moment. A trigger in the database checks for valid client id.

Search, Remove or Modify Complaints

Search By: clientid

4

Submit Clear

Complaints Data

Complain ID: 4-070815-0838

Client ID: 4

Problem Details: Random loss

Problem Status: Initiated

Date: 20-Aug-15

Update Delete

	COMPLAIN_ID	CLIENTID	PROBLEMDETAIL	PROBLEMSTA
▶	4-070815-0838	4	Random loss	Initiated

6. *Complain Search, Remove or Modify*: In this form only details and problem status can be updated or the complaint can be deleted.

Main_Window

Clients
Complains
Billings

	BILLID	CLIENTID	PREVIOUS_DUE	CURRENT_BILL	CREATE_DATE	RECIEVED_DATE
	4AUG_15	4	1000	3000	07-Aug-15 8:3...	10-Aug-15
	22AUG_15	22	500	2000	08-Aug-15 8:3...	
	5AUG_15	5	400	1000	08-Aug-15 8:3...	
	6AUG_15	6	600	1400	08-Aug-15 8:3...	
▶	7AUG_15	7	3000	4500	08-Aug-15 8:3...	

Load

Add

Search

7. *Bill Main*: As the bill is auto-generated at the start of every month, Bills cannot be manually added.

Billsmr

Search via

ClientID

4

Submit

Clear

Client ID

4

Current Bill

3000

Previous Due

1000

Invoice Period

07-Aug-15

Recieve Date

10-Aug-15

Modify

Delete

	BILLID	CLIENTID	PREVIOUS_DUE	CURRENT_BILL	CREATE_DATE	RECIEVED_DATE
▶	4AUG_15	4	1000	3000	07-Aug-15 8:38 PM	10-Aug-15

8. *Bill Update*: A bill cannot be deleted. Only receive date can be updated. An update on receive clears the due from bill_track and updates the last_paid date to receive date.

Code

The program is written in c# winform. It uses the .NET Framework 4.5

For the button to work on every window, I used a switch case scenario event to load data and others.

Snippet:

```
private void load_data_Click(object sender, EventArgs e)
{
    DataTable dt_main_window = new DataTable();
    Activity_Layer grid_data_load = new Activity_Layer();
    switch (mainwindow_tabcontrol.SelectedIndex)
    {
        case 0:
        {
            try
            {
                client_dataGridView.DataSource =
grid_data_load.showClientDetails();
            }
            catch(Exception ex)
            {
                MessageBox.Show("Error: " + ex, "OOPS");
            }
        }
        break;
        case 1:
        {
            try
            {
                complain_dataGridView.DataSource =
grid_data_load.showComplainDetails();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex, "OOPS");
            }
        }
        break;
        case 2:
        {
            try
            {
                billings_dataGridView.DataSource =
grid_data_load.showBillingDetails();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex, "OOPS");
            }
        }
        break;
    }
}
```

I used a separate Business logic layer to interact with database.

Database code snippet:

```
public DataTable showClientDetails()
{
    string sql = "select * from ADMS_DB.showclients";
    try
    {
        OracleConnection conn = new OracleConnection(constr);
        conn.Open();
    }
    catch (OracleException e)
    {
        throw e;
    }
    this.a = new OracleDataAdapter(sql, constr);
    this.builder = new OracleCommandBuilder(this.a);
    DataSet dataset = new DataSet();
    try
    {
        this.a.Fill(dataset, "CLIENTDETAILS");
    }
    catch (OracleException e)
    {
        throw e;
    }
    return dataset.Tables["CLIENTDETAILS"];
}
```

To connect with oracle sql server, I had to add library file of Oracle Data Access Components for .NET. The name of namespace is Oracle.DataAccess.Client

Future Tasks:

Due to time and manpower constraints, this project isn't entirely complete. The triggers, functions and views haven't been implemented in the UI. One important feature which is to import client profiles from routers and the ability to modify the router database is incomplete due to time constraint. User privilege level and locking mechanism hasn't been implemented.

I will continue to improve this project over time to make it deployment capable in business scenarios. There are few user-unfriendly errors which let user input wrong data type. It will be fixed in the future.

Conclusion

In this project, I tried to implement an ISP management system based on the feedback of my local ISP which provides subpar customer service due to lack of organized information management. It is meant to be user-friendly and robust. The database should be fast and error proof to safe keep the valuable information of ISP