**NEWSPAPER AGENCY**

**APPLICATION DEVELOPMENT – III**

**IMPLEMENTATION PHASE**

*Submitted in partial fulfilment for the award of the degree*

*of*

**Master of Science**

***in***

**Information Technology**

*by*

**SANKET SURESH PETHKAR**

**REG NO: 14MIN2879**



**School of Information Technology and Engineering**

August, 2016

Table of Contents

[1. INTRODUCTION 7](#_Toc458785225)

[2. SOFTWARE REQUIREMENT SPECIFICATION 8](#_Toc458785226)

[2.1.1. Login to the Application 8](#_Toc458785227)

[2.1.1.1. Introduction - 8](#_Toc458785228)

[2.1.1.2. Processing– 8](#_Toc458785229)

[2.1.1.3. Outputs– 8](#_Toc458785230)

[2.1.1.4. Error Handling- 8](#_Toc458785231)

[2.1.2. User Request for Newspaper 8](#_Toc458785232)

[2.1.2.1. Introduction – 8](#_Toc458785233)

[2.1.2.2. Inputs – 9](#_Toc458785234)

[2.1.2.3. Processing– 9](#_Toc458785235)

[2.1.2.4. Outputs– 9](#_Toc458785236)

[2.1.2.5. Error Handling- 9](#_Toc458785237)

[2.1.3. User track request 9](#_Toc458785238)

[2.1.3.1. Introduction – 9](#_Toc458785239)

[2.1.3.2. Inputs – 9](#_Toc458785240)

[2.1.3.3. Processing– 9](#_Toc458785241)

[2.1.3.4. Outputs– 9](#_Toc458785242)

[2.1.3.5. Handling- 10](#_Toc458785243)

[2.1.4. Admin view stock 10](#_Toc458785244)

[2.1.4.1. Introduction – 10](#_Toc458785245)

[2.1.4.2. Inputs – 10](#_Toc458785246)

[2.1.4.3. Processing– 10](#_Toc458785247)

[2.1.4.4. Outputs– 10](#_Toc458785248)

[2.1.4.5. Error Handling- 10](#_Toc458785249)

[2.1.5. Admin add/update stock 10](#_Toc458785250)

[2.1.5.1. Introduction – 10](#_Toc458785251)

[2.1.5.2. Inputs – 11](#_Toc458785252)

[2.1.5.3. Processing– 11](#_Toc458785253)

[2.1.5.4. Outputs– 11](#_Toc458785254)

[2.1.5.5. Error Handling- 11](#_Toc458785255)

[2.1.6. Change Password 11](#_Toc458785256)

[2.1.6.1. Introduction – 11](#_Toc458785257)

[2.1.6.2. Inputs – 11](#_Toc458785258)

[2.1.6.3. Processing– 11](#_Toc458785259)

[2.1.6.4. Outputs– 11](#_Toc458785260)

[2.1.6.5. Error Handling- 12](#_Toc458785261)

[2.2. Non-Functional Requirements 13](#_Toc458785262)

[2.2.1. Performance 13](#_Toc458785263)

[2.2.2. Reliability 13](#_Toc458785264)

[2.2.3. Availability 13](#_Toc458785265)

[2.2.4. Security 14](#_Toc458785266)

[2.2.5. Maintainability 14](#_Toc458785267)

[2.2.6. Portability 14](#_Toc458785268)

[3. LITERATURE SURVEY 14](#_Toc458785269)

[3.1 PURPOSE OF LITERATURE SURVEY 14](#_Toc458785270)

[3.2 METHODOLOGY 15](#_Toc458785271)

[4. Detailed Design of The Project 17](#_Toc458785272)

[4.1 PRODUCT PERSPECTIVE 17](#_Toc458785273)

[4.2 SOFTWARE INTERFACES 17](#_Toc458785274)

[4.3 PRODUCT FUNCTIONS 17](#_Toc458785275)

[4.4 ARCHITECTURAL DESIGN 18](#_Toc458785276)

[4.5 USECASE DIAGRAM 18](#_Toc458785277)

[4.6 CLASS DIAGRAM 19](#_Toc458785278)

[4.7 SEQUENCE DIAGRAM 20](#_Toc458785279)

[5. Implementation of System 23](#_Toc458785280)

[5.1 DETAILED MODULE DESCRIPTION 23](#_Toc458785281)

[5.1.1 Introduction 23](#_Toc458785282)

[Process 23](#_Toc458785283)

[5.1.2 Existing System 24](#_Toc458785284)

[5.1.3 Proposed System 24](#_Toc458785285)

[5.1.4 Modules 25](#_Toc458785286)

[5.1.4.1 New User Registration Screen 25](#_Toc458785287)

[5.1.4.2 Login to the Application 25](#_Toc458785288)

[5.1.4.3 Cylinder Request by User Screen. 25](#_Toc458785289)

[5.1.4.4 Check Status of Request Screen 25](#_Toc458785290)

[5.1.4.5 Inventory Screen 25](#_Toc458785291)

[5.1.4.6 Change Password 26](#_Toc458785292)

[5.1.5 System Architecture 26](#_Toc458785293)

[5.2 CODE 27](#_Toc458785294)

[5.3 SCREEN SHOTS 61](#_Toc458785295)

[6 TESTING AND DOCUMENTATION 75](#_Toc458785296)

[6.1 TEST PLAN 75](#_Toc458785297)

[6.1.1 INTRODUCTION 75](#_Toc458785298)

[6.1.2 OBJECTIVE 75](#_Toc458785299)

[6.1.3 DEFINITIONS 75](#_Toc458785300)

[6.1.4 TEST ITEMS 75](#_Toc458785301)

[6.1.5 APPROACH 76](#_Toc458785302)

[6.1.5.1 Unit Testing 76](#_Toc458785303)

[6.1.5.2 Integration Testing: 76](#_Toc458785304)

[6.1.5.3 Regression Testing: 76](#_Toc458785305)

[6.1.5.4 Acceptance Testing: 77](#_Toc458785306)

[6.1.5.5 System Testing: 77](#_Toc458785307)

[6.2 TEST DESIGN/TEST CASE SPECIFICATIONS 77](#_Toc458785308)

[6.2.1 Registration Successful 77](#_Toc458785309)

[6.2.2 Registration Failed 78](#_Toc458785310)

[6.2.3 Login Successful 78](#_Toc458785311)

[6.2.4 Login Failed 79](#_Toc458785312)

[6.2.5 Change Password Successful 79](#_Toc458785313)

[6.2.6 Change Password Failed 80](#_Toc458785314)

[6.2.7 User Request Cylinder Successful 80](#_Toc458785315)

[6.2.8 User Request Cylinder Failed 81](#_Toc458785316)

[6.2.9 User check status Successful 81](#_Toc458785317)

[6.2.10 Maintain Stock 82](#_Toc458785318)

[6.2.11 User Interface Testing: 82](#_Toc458785319)

[6.2.12 Pass or Fail Criteria 83](#_Toc458785320)

[6.3 TEST REPORTS 83](#_Toc458785321)

[6.3.1 Introduction 83](#_Toc458785322)

[6.3.2 Test Case Result Summary 83](#_Toc458785323)

[6.3.3 Detailed Test Results 84](#_Toc458785324)

[6.3.3.1 Manual Testing 84](#_Toc458785325)

[6.3.3.2 TC # 1 – Registration 84](#_Toc458785326)

[6.3.3.3 TC # 2 – Login 85](#_Toc458785327)

[6.3.3.4 TC # 3 – User Cylinder Request. 85](#_Toc458785328)

[6.3.3.5 TC # 4 – User check status. 85](#_Toc458785329)

[6.3.3.6 TC # 5 – Admin Add/View Stock. 86](#_Toc458785330)

# **INTRODUCTION**

The modern industrial world is very much advanced in technology and competition in the world is intense. If we want healthy growth of business, one must keep business techniques up-to-date. Today computer is the mandatory part of every business in the world. That is why computer is used for major task like storing records, complex calculations, worldwide communication etc. since it is not possible to complete all these tasks by human itself in efficient way.

The Newspaper agency business is closely related to management of things like customer, stock, supplier, newspaper, and so on. So, if we select manpower to perform management of all these things, following problems may arise:

1. Paper based transaction has become old nowadays because it is time consuming and less secured/reliable.
2. The process of storing all these records also raises the issue of security and space.

Computerization solves all these problems and provides reliability and correctness in the process of management.

Performing operation with the help of computer will save the manual work of agent and so the billing work becomes easy no need for manual calculation. Security of data is maintained. We can retrieve or manage records instantly because of user friendly environment of computer.

The process which is carried out currently has certain drawbacks:-

* All work is carried on manually, hence increases the manpower.
* Also, the manual entries, increases the probability of human errors.
* There are lots of ways, through which the system might crack.
* Also there are separate formats required for different types of customers.

So we developed this Newspaper Agency.

* This project is to manage all the newspapers, their suppliers, customers, distributors, etc.
* It makes easy to generate and maintain bills, payments and auditing.

The prime goals of proposed system are:-

* To provide security of the software with the help of password facility.
* To provide User Friendly interface that will make various operations easier to perform.
* To provide automatic bill generation.
* To provide automatic stock inward outward.
* To provide automatic stock update after the selling process.
* To provide data validation.
* To provide facility of storing record of Employees, record of sold items, record of pending order, record of completed order, and record of expense.
* To provide facility of instant report generation.

# **SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 FUNCTIONAL REQUIREMENTS**

# **Login to the Application**

# **Introduction** -

As you hit the exe of the application, it will display a login page.

User will enter the user-id and password in the fields provided.

# **Processing**–

The entry will be checked in the database and user will be welcomed to the home page if user is found registered.

# **Outputs**–

User will land on the application home page once it will be verified from the database.

# **Error Handling**-

1. Empty Filed Validation - The fields should not be empty. Empty field validation will be there.

2. Credentials Validation - User will not be allowed to login if password does not match with the password saved in the database

# **User Request for Newspaper**

# **Introduction** –

User Request for Newspaper

# **Inputs –**

User will request for newspaper by selecting newspaper type, language and no of papers.

# **Processing**–

Validate the given details and record the information in to the database.

# **Outputs**–

For the request, token number is generated.

# **Error Handling**-

Empty Filed Validation - The field should be selected from drop down and textbox should not be empty.

# **User track request**

# **Introduction** –

User track request status.

# **Inputs –**

User will enter token number and click on check status button.

# **Processing**–

Validate the given token number and admin update the status of request.

# **Outputs**–

User get request status message

# **Handling**-

Empty Filed Validation –The entered token no should be valid. Invalid token no is not accepted.

# **Admin view stock**

# **Introduction** –

Admin view stock

# **Inputs –**

In Inventory module, Admin click View Stock link.

# **Processing**–

The stock detail is maintained in database.

# **Outputs**–

Admin view stock details.

# **Error Handling**-

Empty Filed Validation –The entered token no should be valid. Invalid token no is not accepted.

# **Admin add/update stock**

# **Introduction** –

Admin add/update stock

# **Inputs –**

In Inventory module, Admin add/update stock by selecting cylinder type and no of cylinder.

# **Processing**–

The stock detail is maintained in database.

# **Outputs**–

Inventory updated successfully message is displayed.

# **Error Handling**-

Empty Filed Validation –The selected field should be selected and text field should not be empty.

# **Change Password**

# **Introduction** –

For security purpose password can be change over a time period.

# **Inputs –**

Old Password, New Password, Confirm Password.

# **Processing**–

Validate the given details and record the information in to the database.

# **Outputs**–

Database Record, Database successfully updated message.

# **Error Handling**-

Empty Filed Validation - The fields should not be empty. Empty field validation will be there.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Event | Trigger | Source | Use Case | Response | Destination |
| 1 | Login | Enter to System Feature | Admin or User | Entering Login username and password | Valid or Invalid Username or Password | Accessing System |
| 2 | Admin want to Verify User | Verify User | Admin | User Verification | User Verified or Not Verified | User Permitted |
| 3 | Admin wants to Check Stock | View Stock | Admin | Stock in Hand | Stock Details | View Stock |
| 4 | Admin wants to Add Stock | Adding Stock | Admin | Stock Updated | Stock Increased | Stock Updated in database |
| 5 | Admin Add or Update Rates | Update Rates | Admin | Rates Modified | Rates Updated | Database Updation |
| 6 | User Login | Login details | User | Entering Login username and password | Valid or Invalid Username or Password | Accessing System |
| 7 | User Request for Newspaper | Request for newspaper | User | Request for a paper | Request number | Request Added |
| 8 | User tracks request | Enters Request Details | User | Search for request number | Request Pending or complete | Details Displayed |

# **Non-Functional Requirements**

Non-functional requirements define the need in terms of performance, reliability, availability, security, maintainability, portability, security.

# **Performance**

Performance requirements define acceptable response times for system functionality. Although the system is developed suiting for the least system performances, the performance of the system will highly depend on the performance of the hardware and software components of the installing computer.

# **Reliability**

The software should provide automatically generated backup.

* + 1. **Availability**

The availability of the GAMS is up to the internet connection of the client. Since this is client-server related application, it shall be attainable all the time. User should have an account to enter the system.

# **Security**

The software used for this gas agency system include the password, so the security is provided. When anyone opens the software it has the provision for entering password. We have to enter the correct password, otherwise we cannot enter into the system. Password is saved in system Registry for more security.

# **Maintainability**

The software product will be easy to maintain and be fully function even when other system upgrades occurs. The system will perform with normalcy under any conditional state.

# **Portability**

The software will be able to support any Window operation system changes. The portability should be effective without disruption from normal performance.

# **LITERATURE SURVEY**

# **PURPOSE OF LITERATURE SURVEY**

# “The Newspaper Agency System” is one can go into almost required solution regarding the Newspaper shop.

# Software we can easily track all the customer detail, supplier detail, newspaper detail, stock detail, bill detail and we can able generate report.

# The main goal of the software is build a good management tool.

# The newspaper agency system will make storing of the customer records, stock records, sales detail, employee information, expense detail and order details in the database it will generate bills and print the bills.

# **METHODOLOGY**

The method being used in developing the system is the system Development Life Cycle (SDLC) The SDLC process includes project identification and selection, project initiation and planning, analysis, design, implementation and maintenance.

**System Development Life Cycle (SDLC)**

The system Development Life Cycle (SDLC) is a traditional methodology used to develop, maintain and replace or enhance information system. The main reason SDLC model is chosen because in SDLC it is possible to complete some activities in one phase in parallel with some activities of another phase. The life cycle can also be interactive that is phase are repeated as required until an acceptable system is found.

**Phase 1: Project identification and selection**

In this phase the project information system needs are identified and analyzed such as identified the title of the project that is Newspaper Agency System, scope and objective of the Newspaper Distributer System.

**Phase 2: Project initiation and planning**

During this phase the Gantt chart has been developed as a time line to determining the task involve in developing the Newspaper Agency System.

**Phase 3: Analysis**

In the phase, the existing system is studies by collecting the information through the Internet and analyzed the information to get alternatives for the use of proposed system. Determine what the Newspaper Agency System should do.

**Phase 4: Design**

Logical design is the fourth phase in SDLC methodology. The functional features chosen for the proposed system in Analysis phase are described. Part of the logical design of the information system is to devise the user interface. The interface plays an important role to connect the user with the system and is thus extremely important.

# **Detailed Design of The Project**

# **PRODUCT PERSPECTIVE**

Newspaper Agency System is done to make the manual process easier by making it a computerized system for billing and maintaining stock. The Newspaper Agency get the order request through phone calls or by personal from their customers and deliver the newspaper to their address based on their demand and previous delivery date. This process is made computerized and the customer’s name, address and stock detail are stored in a database. Based on this, billing for the customer is made simple and easier, since the customer order for gas can be accepted only after completing a certain period from the previous delivery. This can be calculated and billed easily through this.

# **SOFTWARE INTERFACES**

* Operating System: Windows XP, Window 7 x64 Edition.
* Front End: Java Swing, AWT.
* Back End: MySQL.

# **PRODUCT FUNCTIONS**

Admin (Server side module)

* Reads posted product details.
* Validates.
* Shows product status.
* Allowance of purchasing process

Purchasing Customer (Server-side module)

* Analyse and Checks existence posted product details
* Purchase the required product
* Make payment through credit card over the network.
* Automatic transaction processed the details and update the admin account.

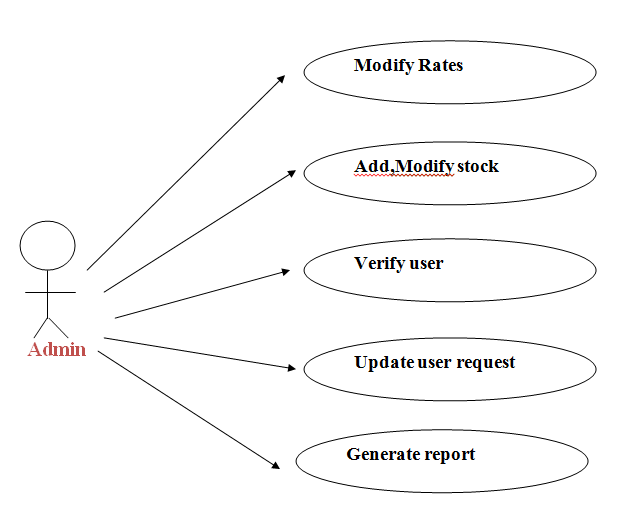
# **ARCHITECTURAL DESIGN**

The software capabilities and requirements specified in GAMS Simulation Software Requirements Specification are transformed into programs that will execute. Software items are partitioned into classes, objects, DFD, ER, Sequence and other diagram.

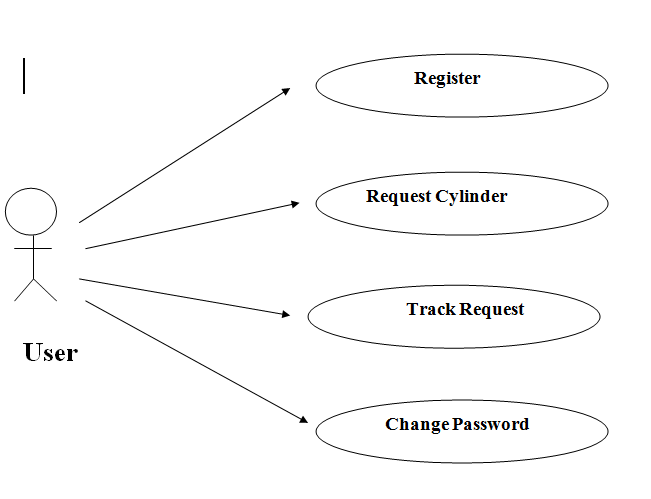
# **USECASE DIAGRAM**

Use Case Diagrams (UCD) can be used to describe the functionality of a system in a horizontal way. That is, rather than merely representing the details of individual features of your system, UCDs can be used to show all of its available functionality.

**USE CASE DIAGRAM FOR ADMIN**

****

**USE CASE DIAGRAM FOR USERS.**

****

# **CLASS DIAGRAM**

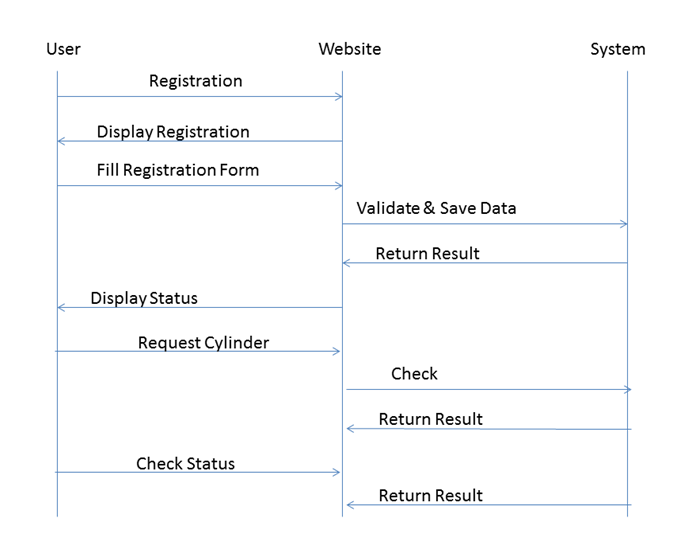
Class diagrams enable us to establish relationship among various classes of the system.

**C:\Users\na267003\Downloads\class.png**

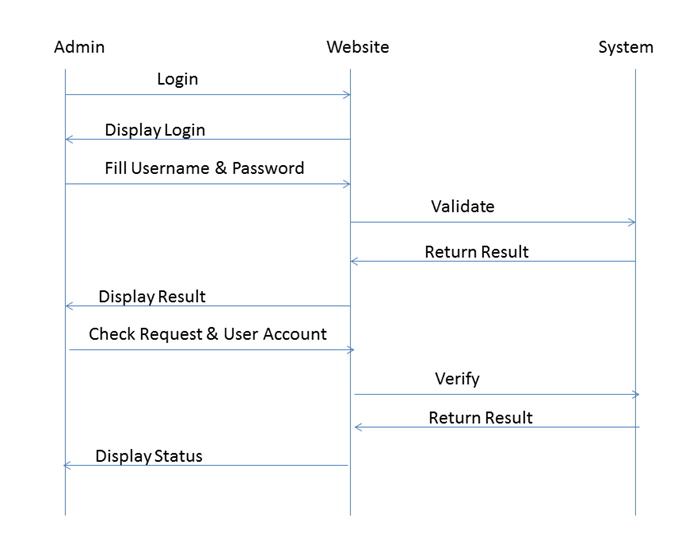
# **SEQUENCE DIAGRAM**

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

**Sequence Diagram for User**



**Sequence Diagram for Admin**



# **Implementation of System**

# **DETAILED MODULE DESCRIPTION**

# **5.1.1 Introduction**

The **GAS Agency Management System**’s objective is to make the manual process easier by making it a computerized system for billing and maintaining stock. Without this automation the management of the gas agency has become a tedious task. The system will be able to handle many services to take care of all customers in an easy and timely manner. The Main Objective of the Gas Agency is to provide a facility to customer by registering with required valid information for customer’s registration ID and for his identity. So customer can register his requirements by posting its valid details. They can request for a cylinder and track the request.

# **Process**

The various users participating in the system are as follows:

The Administrator

The User

1. **Administrator Role**

Admin is super user of this application. He is owner of this application. The admin will authenticate through system.

* + Administrator verifies the user.
  + Administrator maintains stock by adding/ updating / viewing stock detail.
  + Administrator can update the status of the request depending on the availability of stock.
  + Administrator can update the price of the cylinder.
  + Administrator can generate report of the transaction.

1. **User Role**

New users will first register into the system. At the time of registration, the user should provide all the required information.

* User can login with the registered Userid and password.
* User can request the cylinder which includes the type of cylinder and the unit of cylinder.
* When the user post the request token number will be generated.
* User can check the status of the request by entering the token number.

# **Existing System**

The existing system is a manual system. In the present Gas Agency system, all the activities are done manually. All data entry is performed by writing data into the book, paper documents. When there is need for retrieving details searching is unavoidable this is a difficult task searching the records manually .This is also too much time consuming when we want to retrieve details according to some specific conditions.

# **Proposed System**

The development of this system contains the following activities, which tries to automate the entire processes keeping in the view of database integration approach

* Users will receive better and quick service.
* Security is ensured by protecting the system with passwords.
* Normalized database tables eliminate data redundancy.
* Provision for minimizing errors in data entry.
* Efficient data storage.
* Real-time response and user-friendliness.
* Time saving.
* User Friendly
* Speed and Accuracy
* Efficiency and flexibility
* Formatted Output

# **Modules**

# **New User Registration Screen**

New Customer in the portal have to registered them for going through the services of the application.

# **Login to the Application**

Here we have 2 different login screen one for Admin and other for User. User can login into the application with registered credential.

The entry will be checked in the database and user will be welcomed to the home page if user is found registered. User will land on the application home page once it will be verified from the database.

# **Cylinder Request by User Screen.**

Depending on user type, after login the page will displayed. When the user logs in as a user, this screen will show up. In this screen, user can post request for the cylinder.

# **Check Status of Request Screen**

User can check the status of the request by entering the token no generated during the cylinder request.

# **Inventory Screen**

Admin can manage the stock by adding/ viewing the stock.

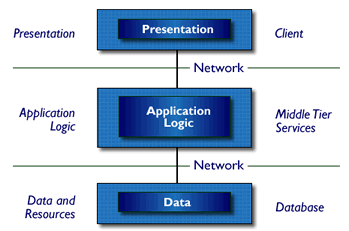
# **Change Password**

This Module provides the facility to change the password for the credential. It will be helpful for the security reasons.

# **System Architecture**

The current application is being developed by taking the 3-tier architecture as a prototype. The 3-tier architecture is the most common approach used for web applications today. In the typical example of this model, the web browser acts as the client, IIS handles the business logic, and a separate tier MS-SQL Server handles database functions.

Although the 3-tier approach increases scalability and introduces a separation of business logic from the display and database layers, it does not truly separate the application into specialized, functional layers. For prototype or simple web applications, the 3-tier architecture may be sufficient. However, with complex demands placed on web applications, a 3-tiered approach falls short in several key areas, including flexibility and scalability. These shortcomings occur mainly because the business logic tier is still too broad- it has too many functions grouped into one tier that could be separated out into a finer grained model.



Three-Tier Architecture

* Tier 1: the client contains the presentation logic, including simple control and user input validation. This application is also known as a thin client. The client interface is developed using ASP.Net Server Controls and HTML controls in some occasions
* Tier 2: the middle tier is also known as the application server, which provides the business processes logic and the data access. The business logic/ business rules can be written either with C#.Net or VB.Net languages. These business runes will be deployed as DLL’s in IIS web server.
* Tier 3: the data server provides the business data. MS-SQL server acts as Tier-3, which is the database layer.

These are some of the advantages of three-tier architecture:

* It is easier to modify or replace any tier without affecting the other tiers.
* Separating the application and database functionality means better load balancing.
* Adequate security policies can be enforced within the server tiers without hindering the clients.

The proposed system can be designed perfectly with the three tier model, as all layers are perfectly getting set as part of the project. In the future, while expanding the system, in order to implement integration touch points and to provide enhanced user interfaces, the n-tier architecture can be used.

# **CODE**

**Admin\_Cyl\_Req.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class Admin\_Cyl\_Req : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string u = Session["User"].ToString();

if (u == "")

{

Response.Redirect("Admin\_Login.aspx");

}

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (DropDownList1.SelectedIndex == 2)

{

int exstk = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Cylinder\_Nos from Stock where Cylinder\_Type='" + TextBox4.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

exstk = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

exstk = exstk - int.Parse(TextBox5.Text);

if (exstk > 0)

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "update Stock set Cylinder\_Nos='" + exstk.ToString() + "' where Cylinder\_Type='" + TextBox4.Text + "'";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "update User\_Request set Req\_Status='" + DropDownList1.SelectedItem.Text + "' where Req\_ID='" + TextBox1.Text + "'";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

}

else

{

Label1.Text = "Stock Low. Cannot Update.";

}

}

else

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "update User\_Request set Req\_Status='" + DropDownList1.SelectedItem.Text + "' where Req\_ID='" + TextBox1.Text + "'";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

}

TextBox1.Text = "";

TextBox2.Text = "";

TextBox3.Text = "";

DropDownList1.SelectedIndex = 0;

Label1.Text = "Status Updated Successfully";

SqlDataSource1.SelectCommand = "select \* from User\_Request";

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

TextBox1.Text = GridView1.SelectedRow.Cells[1].Text;

TextBox2.Text = GridView1.SelectedRow.Cells[2].Text;

TextBox3.Text = GridView1.SelectedRow.Cells[3].Text;

TextBox4.Text = GridView1.SelectedRow.Cells[4].Text;

TextBox5.Text = GridView1.SelectedRow.Cells[5].Text;

DropDownList1.SelectedItem.Text = GridView1.SelectedRow.Cells[6].Text;

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("pgRepRequest.aspx");

}

protected void Button3\_Click(object sender, EventArgs e)

{

Session["Req"] = TextBox1.Text;

Response.Redirect("pgBill.aspx");

}

}

**Admin\_Login.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class Admin\_Login : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

int cnt = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select count(\*) from Admin\_Login where Username='" + TextBox1.Text + "' and Password='" + TextBox2.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

cnt = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

if (cnt == 0)

{

Label1.Text = "Invalid Username / Password";

TextBox1.Text = "";

TextBox2.Text = "";

TextBox1.Focus();

}

else

{

Session["User"] = TextBox1.Text;

Response.Redirect("Admin\_Home.aspx");

}

}

}

**Admin\_Rate.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class Admin\_Rate : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string u = Session["User"].ToString();

if (u == "")

{

Response.Redirect("Admin\_Login.aspx");

}

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

if (!IsPostBack)

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Cylinder\_Rate from Rate where Cylinder\_Type='Domestic'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

TextBox1.Text = rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Cylinder\_Rate from Rate where Cylinder\_Type='Commercial'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

TextBox2.Text = rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "update Rate set Cylinder\_Rate='" + TextBox1.Text + "' where Cylinder\_Type='Domestic'";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "update Rate set Cylinder\_Rate='" + TextBox2.Text + "' where Cylinder\_Type='Commercial'";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

Label1.Text = "Rates Updated Successfully";

}

}

**Default.aspx**

using System;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class \_Default : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

int cnt = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select count(\*) from User\_Login where Username='" + TextBox1.Text + "' and Password='" + TextBox2.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

cnt = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

if (cnt == 0)

{

Label1.Text = "Invalid Username / Password";

TextBox1.Text = "";

TextBox2.Text = "";

TextBox1.Focus();

}

else

{

string status = "";

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select U\_Status from User\_Details where Username='" + TextBox1.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

status = rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

if (status == "Off")

{

Label1.Text = "Account Suspended. Please Contact Admin.";

TextBox1.Text = "";

TextBox2.Text = "";

TextBox1.Focus();

}

else

{

Session["User"] = TextBox1.Text;

Response.Redirect("User\_Home.aspx");

}

}

}

}

**pgAdminCP.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class pgAdminCP : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string u = Session["User"].ToString();

if (u == "")

{

Response.Redirect("Admin\_Login.aspx");

}

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

int cnt = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select count(\*) from Admin\_Login where Username='" + Session["User"].ToString() + "' and Password='" + TextBox1.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

cnt = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

if (cnt == 0)

{

Label1.Text = "Invalid Old Password";

TextBox1.Text = "";

TextBox2.Text = "";

TextBox3.Text = "";

TextBox1.Focus();

}

else

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "update Admin\_Login set Password='" + TextBox3.Text + "' where Username='" + Session["User"].ToString() + "'";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

Label1.Text = "Password Updated Successfully";

TextBox1.Text = "";

TextBox2.Text = "";

TextBox3.Text = "";

TextBox1.Focus();

}

}

}

**pgBill.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class pgBill : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string reqid = Session["Req"].ToString();

if (reqid != "")

{

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

Label4.Text = reqid;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select U\_Name, U\_Address, U\_Contact from User\_Details where Username = (select Username from User\_Request where Req\_ID='" + reqid + "')";

rs = cmd.ExecuteReader();

while (rs.Read())

{

Label1.Text = rs.GetValue(0).ToString();

Label2.Text = rs.GetValue(1).ToString();

Label3.Text = rs.GetValue(2).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Cylinder\_Type, Cylinder\_Nos from User\_Request where Req\_ID='" + reqid + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

Label5.Text = rs.GetValue(0).ToString();

Label6.Text = rs.GetValue(1).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Cylinder\_Rate from Rate where Cylinder\_Type='" + Label5.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

Label7.Text = rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

int nos = int.Parse(Label6.Text);

double rt = int.Parse(Label7.Text);

double tot = rt \* nos;

Label8.Text = tot.ToString();

}

else

{

Response.Redirect("Admin\_Cyl\_Req.aspx");

}

}

}

**pgCheckStatus.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class pgCheckStatus : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string u = Session["User"].ToString();

if (u == "")

{

Response.Redirect("Default.aspx");

}

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

Label1.Text = "";

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Req\_Status from User\_Request where Req\_ID='" + TextBox1.Text + "' and Username='" + Session["User"].ToString() + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

Label1.Text = "Request Status : " + rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

if (Label1.Text == "")

{

Label1.Text = "Invalid Details Provided";

TextBox1.Text = "";

TextBox1.Focus();

}

}

}

**pgInventory.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class pgInventory : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string u = Session["User"].ToString();

if (u == "")

{

Response.Redirect("Admin\_Login.aspx");

}

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

int cnt;

string id, fid;

int temp;

temp = 0;

cnt = 0;

id = "";

fid = "";

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select count(\*) from Inventory";

rs = cmd.ExecuteReader();

while (rs.Read())

{

cnt = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

if (cnt > 0)

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select max(Order\_ID) from Inventory";

rs = cmd.ExecuteReader();

while (rs.Read())

{

id = rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

temp = int.Parse(id.Substring(1, 4));

temp++;

id = "";

id = temp.ToString();

if (id.Length == 1)

{

id = "I000" + id;

}

else if (id.Length == 2)

{

id = "I00" + id;

}

else if (id.Length == 3)

{

id = "I0" + id;

}

else if (id.Length == 4)

{

id = "I" + id;

}

fid = id;

}

else

{

fid = "I0001";

}

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "insert into Inventory values('" + fid + "','" + System.DateTime.Now.ToShortDateString() + "','" + DropDownList1.SelectedItem.Text + "','" + TextBox1.Text + "')";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

int exstk = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Cylinder\_Nos from Stock where Cylinder\_Type='" + DropDownList1.SelectedItem.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

exstk = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

exstk = exstk + int.Parse(TextBox1.Text);

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "update Stock set Cylinder\_Nos='" + exstk.ToString() + "' where Cylinder\_Type='" + DropDownList1.SelectedItem.Text + "'";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

Label1.Text = "Inventory Updated";

SqlDataSource1.SelectCommand = "select \* from Inventory";

}

protected void Button2\_Click(object sender, EventArgs e)

{

Response.Redirect("pgRepInventory.aspx");

}

}

**pgUserReq.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class pgUserReq : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string u = Session["User"].ToString();

if (u == "")

{

Response.Redirect("Default.aspx");

}

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

DateTime prevdate = new DateTime();

int chk = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Req\_Date from User\_Request where Username='" + Session["User"].ToString() + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

prevdate = DateTime.Parse(rs.GetValue(0).ToString());

chk = 1;

}

rs.Close();

cmd.Dispose();

cn.Close();

int nod = 0;

if (chk == 1)

{

DateTime todate = new DateTime();

TimeSpan t = todate.Subtract(prevdate);

nod = t.Days;

}

else

{

nod = 20;

}

if (nod <= 7)

{

Label1.Text = "Cannot Re-Request within 7 days of Previous Request";

}

else

{

int cnt;

string id, fid;

int temp;

temp = 0;

cnt = 0;

id = "";

fid = "";

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select count(\*) from User\_Request";

rs = cmd.ExecuteReader();

while (rs.Read())

{

cnt = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

if (cnt > 0)

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select max(Req\_ID) from User\_Request";

rs = cmd.ExecuteReader();

while (rs.Read())

{

id = rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

temp = int.Parse(id.Substring(1, 4));

temp++;

id = "";

id = temp.ToString();

if (id.Length == 1)

{

id = "R000" + id;

}

else if (id.Length == 2)

{

id = "R00" + id;

}

else if (id.Length == 3)

{

id = "R0" + id;

}

else if (id.Length == 4)

{

id = "R" + id;

}

fid = id;

}

else

{

fid = "R0001";

}

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "insert into User\_Request values('" + fid + "','" + Session["User"].ToString() + "','" + System.DateTime.Now.ToShortDateString() + "','" + DropDownList1.SelectedItem.Text + "','" + DropDownList2.SelectedItem.Text + "','Pending')";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

Label1.Text = "Request Accepted. Your Token No is : " + fid;

}

}

}

**Register.aspx**

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

using System.Data.SqlClient;

public partial class Register : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

int cnt = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select count(\*) from User\_Details where Username='" + TextBox8.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

cnt = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

if (cnt == 0)

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "insert into User\_Details values('" + TextBox8.Text + "','" + TextBox1.Text + "','" + TextBox7.Text + "','" + TextBox2.Text + "','" + TextBox3.Text + "','" + TextBox6.Text + "','Off')";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "insert into User\_Login values('" + TextBox8.Text + "','" + TextBox5.Text + "')";

cmd.ExecuteNonQuery();

cmd.Dispose();

cn.Close();

TextBox1.Text = "";

TextBox2.Text = "";

TextBox3.Text = "";

TextBox4.Text = "";

TextBox5.Text = "";

TextBox6.Text = "";

TextBox7.Text = "";

TextBox8.Text = "";

Label2.Text = "User Registered Successfully";

}

else

{

Label2.Text = "Username already in Use";

TextBox8.Text = "";

TextBox8.Focus();

}

}

}

**Retrive\_Password.aspx**

using System;

using System.Data;

using System.Configuration;

using System.Collections;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Web.UI.HtmlControls;

using System.Data.SqlClient;

public partial class Retrive\_Password : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

}

protected void Button1\_Click(object sender, EventArgs e)

{

int cnt = 0;

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Count(\*) from User\_Details where Username='" + TextBox1.Text + "' and U\_Email='" + TextBox2.Text + "' and Password\_Hint='" + TextBox3.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

cnt = int.Parse(rs.GetValue(0).ToString());

}

rs.Close();

cmd.Dispose();

cn.Close();

if (cnt == 0)

{

Label1.ForeColor = System.Drawing.Color.Maroon;

Label1.Text = "Invalid Details Provided";

TextBox1.Text = "";

TextBox2.Text = "";

TextBox3.Text = "";

TextBox1.Focus();

}

else

{

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select Password from User\_Login where Username='" + TextBox1.Text + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

Label1.ForeColor = System.Drawing.Color.Navy;

Label1.Text = "Your Password : " + rs.GetValue(0).ToString();

}

rs.Close();

cmd.Dispose();

cn.Close();

}

}

}

**User\_Home.aspx**

using System;

using System.Collections;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.UI;

using System.Web.UI.HtmlControls;

using System.Web.UI.WebControls;

using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

using System.Data.SqlClient;

public partial class User\_Home : System.Web.UI.Page

{

SqlConnection cn = new SqlConnection();

SqlCommand cmd = new SqlCommand();

SqlDataReader rs;

protected void Page\_Load(object sender, EventArgs e)

{

string uid = Session["User"].ToString();

if (uid != "")

{

cn.ConnectionString = "Data Source=DELL-PC;Initial Catalog=Gas\_Agency;Integrated Security=True";

cn.Open();

cmd.Connection = cn;

cmd.CommandText = "select U\_Name from User\_Details where Username='" + uid + "'";

rs = cmd.ExecuteReader();

while (rs.Read())

{

Label1.Text = "Hi " + rs.GetValue(0).ToString() + ", Welcome...";

}

rs.Close();

cmd.Dispose();

cn.Close();

}

else

{

Response.Redirect("Default.aspx");

}

}

}

# **SCREEN SHOTS**

**New User Registration with validations**

****

**New User Registration**

****

**User Login with invalid username and password**

****

**Account Suspended by Admin**

****

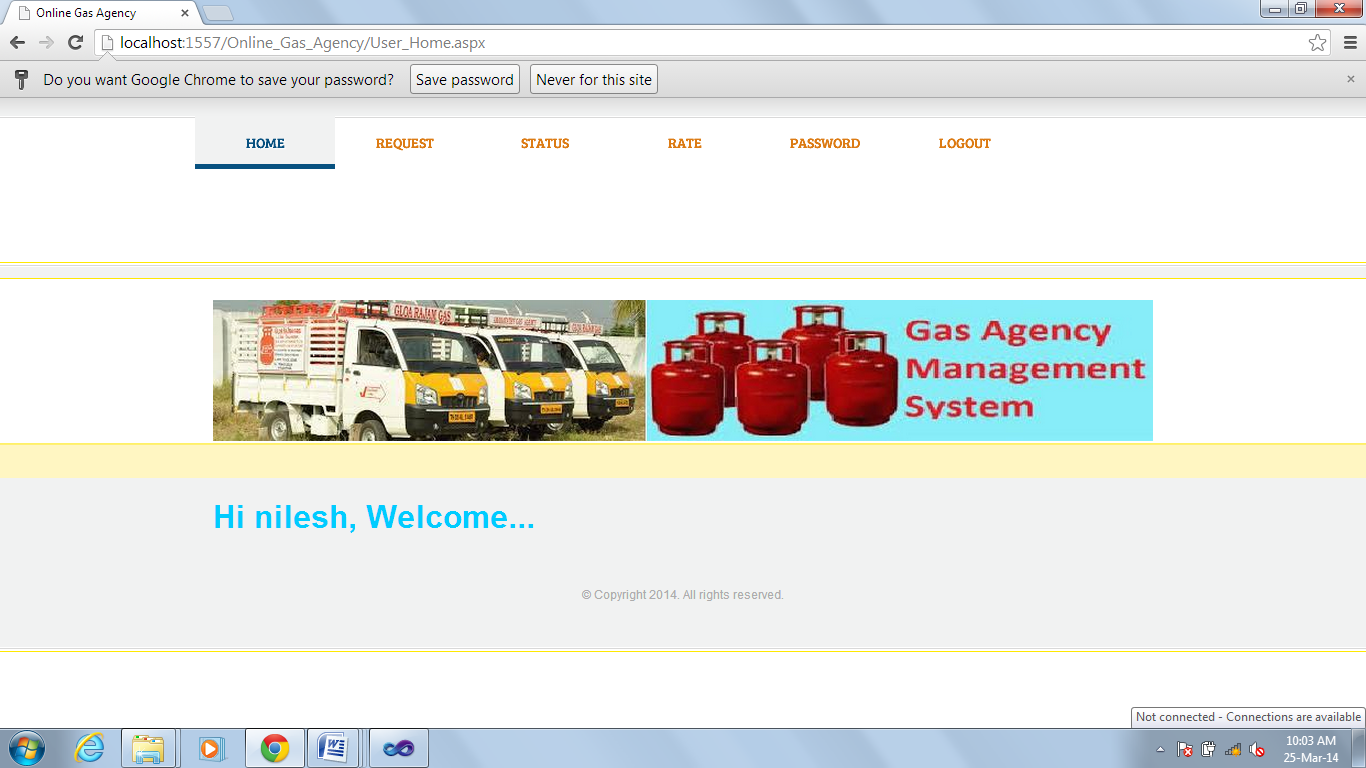
**Admin Login**

****

**User status On by Admin**

****

**User Logins successfully**

****

**Cylinder Request by user**

****

**User requests again for cylinder**

****

**User checks Status**

****

**Admin Completes the request**

****

**Inventory**

****

**View Stock**



# **TESTING AND DOCUMENTATION**

# **TEST PLAN**

# **6.1.1 INTRODUCTION**

A Software Test Plan is a document describing the testing scope and activities. It is the basis for formally testing any software/product in a project. A document describing the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.

This is a Test Plan Document for GAS Management System (GAMS) .The document outlines the test strategies, responsibilities and metrics. It also contains the current library of test cases that will be used for testing purposes and to keep development teams on track.

Potential users should focus on the user interface testing to be able to judge the usability.

Customer should have a thorough understanding of the document, as it will enable to have an even better understanding of the use cases that will be implemented and their black box working style.

# **6.1.2 OBJECTIVE**

The main objective of the Test Plan for the GAS Management System are as follows:

* To identify the feature of the system that will be tested.
* To identify and define all the activities necessary to prepare for and conduct the testing on the GAS Management System.
* To discuss the testing techniques being used to test the GAS Management System.

# **DEFINITIONS**

The following are some terms and definitions that are related to the test plan of the GAS Management System:

* **Pass/Fail Criteria:** Decision rules that are determine whether a software item passes or fails a test.
* **Test:** A collection of one or more test cases.
* **Test Item:** A software item that is an objective of testing.
* **Test Plan:** A document describing the scope, approach, resources, and schedule of the indented testing activities.
* **Test Summary Report:** A document summarizing the testing activity and results.
* **Testing:** The process of analyzing a software item to detect the differences between the existing and required conditions.

# **TEST ITEMS**

The section of the Test Plan lists the all items of the GAS Management System that will be tested:

* Registration
* Login
* Request for Cylinder
* Track request
* Admin view/modify stock
* Change Password

# **APPROACH**

This section of the test plan describes the overall approach for testing the GAS Management System project. The approach followed for the testing the GAS Management System ensures that all the major feature of the project are adequately tested.

# **Unit Testing**

The Unit Testing is a test that tests each single module of the software to check for the errors. This is mainly done to discover errors in code of the GAS Management System. The main goal of the unit testing would be to isolate each part of the program and to check the correctness of the code. In the case of GAS Management System all the forms will be tested. There are many benefits for this unit testing:

* The Unit Testing facilitates change in the code.
* It allows testing to be done in a bottom up fashion.

At the same time, unit testing has some disadvantages such as it might not identify each and every error in the system.

# **Integration Testing:**

In Integration testing, the individual modules are combined and tested as a whole unit. The integration testing generally follows unit testing where each module is tested as a separate unit. The main purpose of the integration testing is to test the functional and performance requirements on the major items of the project. All the modules of the project developed individually would be combined together and tested as a whole system in the integration testing.

# **Regression Testing:**

The Regression testing is done whenever modifications are made to the source code of the project. The Regression testing can also be defined as a process of testing changes made to the computer program and also makes sure that the older programming still works with the new changes.

So, before any new version of the software product is released, the old test cases for the project will be run against the software with the changes made, to make sure that the old functionalities of the project still work.

# **Acceptance Testing:**

By performing acceptance tests on an application the testing team will deduce how the application will perform in production. This test mainly qualifies the project and decides if it will be accepted by the users of the system. The users or the customers of the project are responsible for the test.

# **System Testing:**

The system testing is mainly done on the whole integrated system to make sure that the project that has been developed meets all requirements. The test cases for the system testing will be the combination of unit and integration tests.

# **TEST DESIGN/TEST CASE SPECIFICATIONS**

# **Registration Successful**

|  |  |
| --- | --- |
| **Test Case** | **Registration Successful** |
| Actor | User |
| Pre-conditions | Database created, Interface created |
| Detailed Description | Tests if the correct information is entered in the database and user enters appropriate information |
| Test Procedure | Enter required data into input fields |
| Expected Results | Username will be verified in the database with the information submitted through the input fields from the interface |
| Expected Interface Output | Registration Successful |
| Test Results | Passed |

# **Registration Failed**

|  |  |
| --- | --- |
| **Test Case** | **Registration Successful** |
| Actor | User |
| Pre-conditions | Database created, Interface created |
| Detailed Description | Tests if the correct information is entered in the database and user enters appropriate information |
| Test Procedure | Enter required data into input fields with blank or wrong detail |
| Expected Results | Displays Invalid details message against incorrect input field. |
| Expected Interface Output | Registration Failed. |
| Test Results | Passed |

# **Login Successful**

|  |  |
| --- | --- |
| **Test Case** | **Login Successful** |
| Actor | Admin/User |
| Pre-conditions | Database created, Interface created |
| Detailed Description | Tests if the Userid is created in the database when Admin enters appropriate information |
| Test Procedure | Enter required data into input fields which are username and password |
| Expected Results | Username will be verified in the database with the information submitted through the input fields from the interface |
| Expected Interface Output | Login Successful |
| Test Results | Passed |

# **Login Failed**

|  |  |
| --- | --- |
| **Test ID** | **Failed to Login** |
| Actor | Admin/User |
| Pre-conditions | Database created, Interface created |
| Detailed Description | Tests if the Userid is created in the database when Admin/User enters appropriate information |
| Test Procedure | Enter required data into input fields with blank or wrong username and password |
| Expected Results | Username will be verified in the database and displays the message Invalid Username/Password. |
| Expected Interface Output | Login Failed |
| Test Results | Passed |

# **Change Password Successful**

|  |  |
| --- | --- |
| **Test Case** | **Change Password Successful** |
| Actor | Admin/User |
| Pre-conditions | Database created, Interface created |
| Detailed Description | Tests if the Userid is created in the database when Admin/Owner enters appropriate information |
| Test Procedure | Enter required data into input fields which are username and password |
| Expected Results | Username will be verified in the database with the information submitted through the input fields from the interface |
| Expected Interface Output | Change Password Successful |
| Test Results | Passed |

# **Change Password Failed**

|  |  |
| --- | --- |
| **Test Case** | **Change Password Failed** |
| Actor | Admin/User |
| Pre-conditions | Database created, Interface created |
| Detailed Description | Tests if the Userid is created in the database when Admin/Owner enters appropriate information |
| Test Procedure | Enter required data into input fields which are username and password |
| Expected Results | Password does not changed |
| Expected Interface Output | Change Password Failed |
| Test Results | Passed |

# **User Request Cylinder Successful**

|  |  |
| --- | --- |
| **Test Case** | **User Request Cylinder Successful** |
| Actor | Admin/User |
| Pre-conditions | Database created, Interface created, request details created already in database |
| Detailed Description | Tests if the stock available and generate token number once users post request with cylinder type and no of cylinder. |
| Test Procedure | Select required data from input fields which are cylinder type and cylinder number. |
| Expected Results | Request is accepted and token number is generated |
| Expected Interface Output | Cylinder request Successful |
| Test Results | Passed |

# **User Request Cylinder Failed**

|  |  |
| --- | --- |
| **Test Case** | Re request cylinder within 7 days of previous request. |
| Actor | Admin/User |
| Pre-conditions | Database created, Interface created, request details created already in database |
| Detailed Description | Tests if there is any response from posting request more than once in a month into the database. |
| Test Procedure | Select required data from input fields which are cylinder type and cylinder number |
| Expected Results | Nothing in the database should be edited. |
| Expected Interface Output | Returns Message as “Cannot Re-request within 7 days of Previous Request”. |
| Test Results | Passed |

# **User check status Successful**

|  |  |
| --- | --- |
| **Test Case** | **User check Cylinder status Successful** |
| Actor | User |
| Pre-conditions | System is active and cylinder request has been posted. |
| Detailed Description | Tests if system will provide the status of the request with the valid token number. |
| Test Procedure | Input valid token number in the status page. |
| Expected Results | Status of the request should be updated |
| Expected Interface Output | Returns status: Pending or Completed |
| Test Results | Passed |

# **Maintain Stock**

|  |  |
| --- | --- |
| **Test Case** | **Add/View Stock successful** |
| Actor | Admin |
| Pre-conditions | Database created, Interface created |
| Detailed Description | Tests if the Admin is able to add/view stock into the database. |
| Test Procedure | Add/View no of cylinder depending on type of cylinder |
| Expected Results | Database now contains added stock |
| Expected Interface Output | Stock addition successful |
| Test Results | Passed |

# **User Interface Testing:**

|  |  |
| --- | --- |
| **Test Case** | User Interface Creation |
| Actor | Admin/Owner |
| Pre-conditions | Database created |
| Detailed Description | Web view is correctly created |
| Test Procedure | Log in as an administrator.  Navigate to all the tabs and screens |
| Expected Results | The web forms should be all correctly served |
| Expected Interface Output | All forms are correctly displayed |
| Test Results | Not Tested |

|  |  |
| --- | --- |
| **Test Case** | Icon Rendering |
| Actor | Admin/Owner |
| Pre-conditions | Database created |
| Detailed Description | All icons in the web forms are correctly displayed |
| Test Procedure | Log in as an administrator.  Navigate to all the tabs and screens that have icons.  Verify that all icons are shown and that the right icons are shown in the right locations |
| Expected Results | The web forms should have all the right icons shown in the correct places |
| Expected Interface Output | All forms are correctly displayed |
| Test Results | Not Tested |

# **Pass or Fail Criteria**

The test cases executed on the GAS Management System will pass if they meet the specific requirements mention in the document. A Test cases said to be fail, if the desired functionality is not satisfied by the system.

# **TEST REPORTS**

# **Introduction**

The Test Results presents the results obtained by the GAS Management System.

# **Test Case Result Summary**

The summary of the Test case result has been depicted in the table shown below:

These are the test cases Admin page. The major of the testing has been concentrated on the Admin pages.

|  |  |  |
| --- | --- | --- |
| **Test Case #** | **Description** | **Results/Comments** |
| TC#1 | Registration | Passed |
| TC #2 | Admin Login | Passed |
| TC #3 | User request cylinder | Passed |
| TC #4 | User check status | Passed |
| TC #5 | Admin view/add stock | Passed |

# **Detailed Test Results**

# **Manual Testing**

To start with, I have performed manual testing on the GAS Management System. Manual Testing is one of the oldest and rigorous methods of software testing. This testing strategy gives the best opportunity to check every page thoroughly and make sure it works as expected manner. Due to the complexity of the various automation tools and time available for testing the entire application, I have preferred to use Manual Testing based on the fact that is one of the best methods of testing suggested for a beginner. All the test cases mentioned in the test plan will be tested here. The result of the manual testing are represented in the following tables:

# **TC # 1 – Registration**

|  |  |  |
| --- | --- | --- |
| **Test Unit** | **Test Case** | **Result** |
| **Registration Button** | Wrong format of information entered into the data fields of the Registration page | The system generates an error message to the user indicating that the wrong format of data is entered and to re-enter the data. |
| **Registration Button** | A valid input field entered by the user | The users gets registered successfully once the admin verifies the details. |

# **TC # 2 – Login**

|  |  |  |
| --- | --- | --- |
| **Log In Button** | An invalid username or password entered by the user | The system generates a message “invalid user” or “invalid password” |
| **Log In Button** | A valid username or password entered by the user | The system logs on the user and transfers him to the home page. |

# **TC # 3 – User Cylinder Request.**

|  |  |  |
| --- | --- | --- |
| **Test Unit** | **Test Case** | **Result** |
| **User Cylinder request** | Placed Re-request within 7 days of previous request. | The system generates an error message to the user indicating that the “Cannot Re-request within 7 days of Previous Request”. |
| **User Cylinder request** | Selected valid cylinder type and cylinder no | Request is accepted and Token No is generated. |

# **TC # 4 – User check status.**

|  |  |  |
| --- | --- | --- |
| **Test Unit** | **Test Case** | **Result** |
| **User check status** | An invalid Token No entered by the user. | The system generates an error message to the user indicating that the “Token no does not exist”. |
| **User check status** | A valid Token No entered by the user. | Request is accepted and status of request is displayed. |

# **TC # 5 – Admin Add/View Stock.**

|  |  |  |
| --- | --- | --- |
| **Test Unit** | **Test Case** | **Result** |
| **Admin Add Stock** | Selected valid cylinder type and cylinder no | The system generates a message: “Inventory Updated” and added to database. |
| **Admin View Stock** | Admin click on View Stock link | Admin can view the stock details. |