**INTRODUCTION**

* 1. **Objective of the System:**

The main objective of our project is to enable security in the case of mobile crimes.

1. This Project is designed by analyzing the Correct details of the Customer.
2. Provide more security to overcome the flawful activities.
3. we will trace out suspected mobile calls by detailed investigation.
   1. **Purpose of the System:**

The main Purpose of the System is to

1. Provide a friendly Environment to the Authorized Persons of Mobile Operator to use the service.
2. Also gives updated information to the Customers about newly introduced Top ups, Value Added Services, Alerts, etc.

**SYSTEM ANALYSIS**

**2.1 Existing System:**

The Existing System has some following disadvantages as mentioned below: -

1. The dealer can register the customer details by filling the application manually.
2. SIM Activation may takes more time by validating Customer Identification Details.
3. There’s a change of flawful Customer Identification details by submitting duplicate details.

**2.2 Proposed System:**

The system can be automated, fully functional and web supportive.

1. Customer details can be filled by the Authorized people of Mobile Operator by Checking the Identification details based on Andhra Card ID.
2. Maintaining each and every call records used by the Customers.
3. Also tracks the Call details of the Customer, which helps the Police Departments.
4. Security System can Enhanced.

**2.3 SOFTWARE SPECIFICATIONS**

* Operating system : Windows XP/7.
* Coding Language : ASP.Net with C#.
* Data Base : SQL Server 200**5**
* Web application server : IIS
* Model Design: Rational rose
* IDE : Visual Studio

**2.4 HARDWARE SPECIFICATIONS**

|  |  |  |
| --- | --- | --- |
| **Specification** | **Minimum Requirement** | **Optimal Requirement** |
| System | Pentium IV 2.4 GHz. | Core2Duo 3.1GHz |
| Hard Disk | 40 GB | 320GB |
| Monitor | 15’ VGA Colour | 17” TFT |
| RAM | 512 Mb | 2GB |
| Internet | Mandatory | Mandatory(dedicated 1:1) |

* **SYSTEM REQUIREMENT SPECIFICATION**

A **System Requirements Specification** (**SRS**) - a requirements specification for a software system - is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional (or supplementary) requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

* **FUNCTIONAL REQUIREMENTS**

* In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Generally, functional requirements are expressed in the form “system shall do <requirement>”. The plan for implementing functional requirements is detailed in the system design. In requirements engineering, functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system. A requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request -> feature -> use case -> business rule. The use cases for this project are briefly described under SYSTEM ANALYSIS SECTION.
* **NON FUNCTIONAL REQUIREMENTS**

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing non-functional requirements is detailed in the system architecture. The non-functional requirements are "system shall be <requirement>". Non-functional requirements are often called qualities of a system. The following are the Non functional requirements for our system:

**Availability**

A system's availability, or "uptime," is the amount of time that it is operational and available for use. As it is NATGRID online System used by the most critical agency users, our system must be available always(24X7).

**Efficiency**

Specifies how well the software utilizes scarce resources: CPU cycles, disk space, memory, bandwidth, etc. All of the above mentioned resources can be effectively used by performing most of the validations at client side

**Flexibility**

If the organization intends to increase or extend the functionality of the software after it is deployed, that should be planned from the beginning; it influences choices made during the design, development, testing, and deployment of the system. New modules can be easily integrated to our system without disturbing the existing modules or modifying the logical database schema of the existing applications. As it is real time project involving the authorities from agencies and departments, it requires lot of collaboration between them and requires number of evolutions. As it is a distributed application, users can access and utilize the NATGRID interface from anywhere in the world with an internet connection. Database is designed in NATGRID in such a way that the minimum required attributes for interlinking the databases were already considered; hence new attributes can be added without re-defining the design.

**Portability**

Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run.

As NATGRID is a web-application it requires no installation, and they can start accessing NATGRID interface through any of the Web browser with an internet connection.

**Integrity**

Integrity requirements define the security attributes of the system, restricting access to features or data to certain users and protecting the privacy of data entered into the software.

Agencies are controlled to access only the necessary databases based on their functionality. Database users must be able to access only their own database. Access can be disabled by providing appropriate login screens for users and administrator separately.

**Scalability**

Software that is scalable has the ability to handle a wide variety of system configuration sizes. The nonfunctional requirements should specify the ways in which the system may be expected to scale up (by increasing hardware capacity, adding machines, etc.). Our system can be easily expandable. Any additional requirements such as hardware or software which increase the performance of the system can be easily added.

Generic implementation of NATGRID facilitates to include new departments and agencies to be the part of the NATGRID project. An additional server at every department facilitates backup.

**Usability**

Ease-of-use requirements address the factors that constitute the capacity of the software to be understood, learned, and used by its intended users. Hyperlinks will be provided for each and every service the system provides through which navigation will be easier. A system that has high usability coefficient makes the work of the user easier.

**Performance**

The performance constraints specify the timing characteristics of the software. Certain tasks very easy and less time taking to perform like placing an order, adding new products, etc., Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. The requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system

**Feasibility Report:**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operation Feasibility
* Economical Feasibility

**3.1 Technical Feasibility:**

In the feasibility study first step is that the organization or company has to decid that what technologies are suitable to develop by coensidering existing system.

Here in this application used the technologies like Visual Studio 2008 and Sql Server 2005. These are free software that would be downloaded from web.

Visual Studio 2008 –it is tool or technology.

### Economic Feasibility

Assessing the economic feasibility of an implementation by performing a cost/benefit analysis, which as its name suggests compares the full/real costs of the application to its full/real financial benefits.  The alternatives should be evaluated on the basis of their contribution to net cash flow, the amount by which the benefits exceed the costs, because the primary objective of all investments is to improve overall organizational performance.

3.3 **Operational Feasibility**

Not only must an application make economic and technical sense, it must also make operational sense. Very often you will need to improve the existing operations, maintenance, and support infrastructure to support the operation of the new application that you intend to develop.  To determine what the impact will be you will need to understand both the current operations and support infrastructure of your organization and the operations and support characteristics of your new application.To operate this application END-TO-END VMS. The user no need to require any technical knowledge that we are used to develop this project is Asp.net ,C#.net. That the application providing rich user interface by user can do the operation in flexible manner.

**TECHNOLOGIES**

**4**. THE .NET FRAMEWORK

* **Objectives of .Net Framework:**

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.

2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.

3. Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications.

* **Features of .Net (VISUAL STUDIO)**

Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There’s no language barrier with .NET: there are numerous languages available to the developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate.

**The .NET Framework has two main parts:**

1. The Common Language Runtime (CLR).

2. A hierarchical set of class libraries (BCL).

The CLR is described as the “execution engine” of .NET. It provides the environment within which programs run. The most important features are

* Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
* Memory management, notably including garbage collection.
* Checking and enforcing security restrictions on the running code.
* Loading and executing programs, with version control and other such features.
* The following features of the .NET framework are also worth description:

**Managed Code :**

The code that targets .NET, and which contains certain extra Information - “metadata” - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.

**Managed Data:**

With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you’re using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn’t get garbage collected but instead is looked after by unmanaged code.

**Common Type System**

The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn’t attempt to access memory that hasn’t been allocated to it.

**Common Language Specification**

The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

**The Class Library**

.NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of converting value types to object types if and when necessary.

The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O, threading, and so on, as well as XML and database connectivity. The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

* **LANGUAGES SUPPORTED BY .NET**

The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft’s old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family.

Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework.

C# is Microsoft’s new language. It’s a C-style language that is essentially “C++ for Rapid Application Development”. Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own.

Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages.

Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State’s Perl Dev Kit.

**Other languages for which .NET compilers are available include**

* FORTRAN
* COBOL
* Eiffel

|  |  |
| --- | --- |
| ASP.NET  XML WEB SERVICES | Windows Forms |
| Base Class Libraries | |
| Common Language Runtime | |
| Operating System | |

Fig **.**Net Framework

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.

C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET .The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

**Constructors And Destructors:**

Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

**Garbage Collection:**

Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use.

In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

**Overloading:**

Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

**Multi Threading:**

C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

**Structured Exception Handling:**

C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

* SQL-SERVER

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

**SQL-SERVER database consist of six type of objects:**

They are: Table, Query, Form, Report, Macro

**Table:**

A database is a collection of data about a specific topic.

**Views Of Table:**

We can work with a table in two types,

1. Design View

2. Datasheet View

**Design View:**

To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

**Datasheet View:**

To add, edit or analyses the data itself we work in tables datasheet view mode.

**Query:**

A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it. such as deleting or updating.

**SYSTEM DESIGN**

* **Design Methodologies**

The two basic modern design strategies employed in software design are

1. Top Down Design
2. Bottom Up Design

Top down Design is basically a decomposition process, which focuses on the flow of control. At later stages it concerns itself with the code production. The first step is to study the overall aspects of the tasks at hand and to break it into a number of independent modules. The second step is to break each one of these modules further into independent sub-modules. The process is repeated one to obtain modules, which are small enough to group mentally and to code in a straightforward manner. One important feature is that at each level the details of the design at the lower level are hidden.

In a bottom-up design one first identifies and investigates parts of design that are most difficult and necessary designed decision are made the remainder of the design is tailored to fit around the design already chosen for crucial part. It vaguely represents a synthesis process explained in previous section.

One strong point of the top-down method is that it postpones details of the decision until the last stage of the decision. It allows making small design changes when the design is half way through. There is danger that the specifications will be incompatible and this will not be discovered until late in the design process. By contrast the bottom-up strategy first focuses on the crucial part.

In mixing top-down and bottom-up design it often appears that we start in the middle of the problem and work our way both up and down there. In a complex problem; it is often difficult to decide how to modularize the various procedures in such cases one might consider a list of system inputs and decide what functions are necessary to process these inputs. This is called back to front design. Similarly one can start with the required outputs and work backwards evolving so called front-back design. We have applied both the top down and bottom up approach in our design approach.

**Water Fall Model**

Water fall model was being chosen because all requirements were known beforehand and the objective of our software development is the computerization/automation of an already existing manual working system

**Communicated Requirements**

**Requirements Specification**

**Design Specification**

**Executable Software Modules**

**Integrated Software Product**

**Delivered Software Product**

**Changed Requirements**

**Requirements Engineering**

**Design**

**Programming**

**Integration**

**Delivery**

**Maintenance**

**Fig 3.1.1 Water Fall Model**

The stage of planning the development process involves defining a define, develop, test, deliver, operate, and maintain a software product. Different lifecycle models emphasize different aspects of the lifecycle and no single lifecycle model is suitable for all software products. A lifecycle model that is understood and accepted by all concerned parties improves project communication and enhances project manageability, resource allocation, cost control, and product quality.

**The Phased Life Cycle Model**

The phased lifecycle model represents software lifecycle as a series of successive activities. Each phase requires well-defined input information, utilizes well-defined processes and results in well-defined products. The phased model consists of following phases.

**Analysis, Design, Implementation, System Testing and Maintenance**

This model is sometimes called the Waterfall Chart, the implication being that the products cascade from one level to another in smooth progression.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Analysis | Design | Implementation | System Testing | Maintenance |
| Planning,  User needs  Definition | Design  Details | Code, debug and Test | Integration & Acceptance | Enhance, Fix Adapt |

The **Analysis Stage** consists of Planning and Requirements definition Major include understanding the customer’s problem, performing a feasibility study, developing a recommended solution strategy, determining the acceptance criteria and planning development process. The products of planning are a System definition and a project plan.

The **Software Design** follows analysis. Design is concerned with its software components, specifying relationships among components specifying some structure, maintaining a record of design decisions and providing blueprint implementation phase. Design consists of detailed design and Architectural design.

The **Implementation** phase of software development involves translation design specification into source code, and debugging, documentation and unit testing the source code. To enhance the quality of the software the methods are structured control constructs, built in and user defined data types, secure type checking, flexible scope rules exception handling mechanism, concurrency constructs and separates compilation modules.

**System Testing** involves two kinds of testing integration testing and acceptance testing. Developing a strategy for integrating the components of a software system into a functioning whole requires careful planning so that modules are available for integration when needed. Acceptance testing involves planning and execution of various tests in order to demonstrate that the implemented system satisfies the requirements document.

The **Maintenance** phase comes after the acceptance by the customer and release of the system for production work. Maintenance activities include enhancements of capabilities, adaptation of software to new processing environments, and correction of software bugs.

**This project** follows the Phased Life Cycle Model or the Water Fall model to a large extent. The analysis stage consisted of listening to the needs and requirements of the examination department obtaining the required format of the system as desired by them, taking the required data to be stored for future use etc., In the design stage the structure of the system was designed and all the required screens were formatted. The coding and debugging was done even after this stage certain changes were made as made as requested by the guide. The testing was done to check for any errors or bugs or unwanted behavior in the system.

**5.0 SCENARIO BASED MODELING**

Use-oriented techniques are widely used in software requirement analysis and design. Use cases and usage scenarios facilitate system understanding and provide a common language for communication. This paper presents a scenario-based modeling technique and discusses its applications. In this model, scenarios are organized hierarchically and they capture the system functionality at various abstraction levels including scenario groups, scenarios, and sub-scenarios. Combining scenarios or sub-scenarios can form complex scenarios. Data are also separately identified, organized, and attached to scenarios. This scenario model can be used to cross check with the UML model. It can also direct systematic scenario-based testing including test case generation, test coverage analysis with respect to requirements, and functional regression testing.

**5.1USE CASE DIAGRAM**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram. If this interaction is essential to a coherent description of the desired behavior, perhaps the system or use case boundaries should be re-examined.

**Actors**

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

**System boundary boxes (optional)**

A rectangle is drawn around the use cases, called the system boundary box, to indicate the scope of system. Anything within the box represents functionality that is in scope and anything outside the box is not. Four relationships among use cases are used often in practice.

**Include**

In one form of interaction, a given use case may include another. "Include is a Directed Relationship between two use cases, implying that the behavior of the included use case is inserted into the behavior of the including use case. The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description. The notation is a dashed arrow from the including to the included use case, with the label "«include»". There are no parameters or return values.

**Extend**

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "«extend»". Modelers use the «extend» relationship to indicate use cases that are "optional" to the base use case.

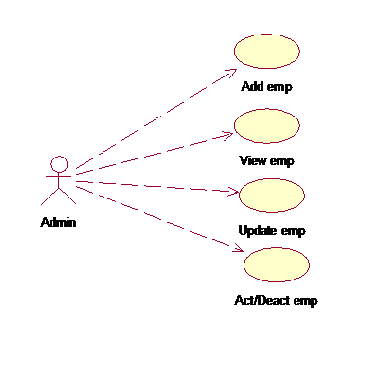


Fig: Use case diagram for Admin

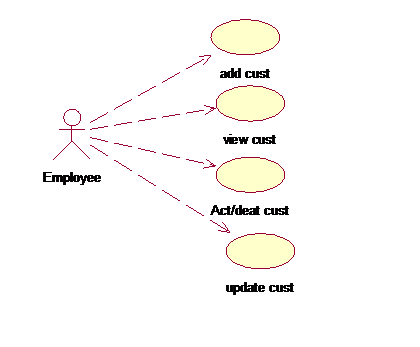
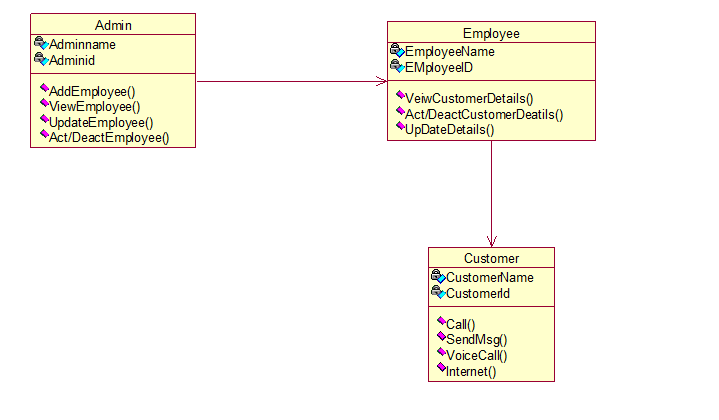


Fig: Use case diagram for Employee



Fig: use case diagram for customer

**Class Diagram:**

****

**5.2 SEQUENCE DIAGRAM**

A sequence diagram in Unified Modeling Language (UML) 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 diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

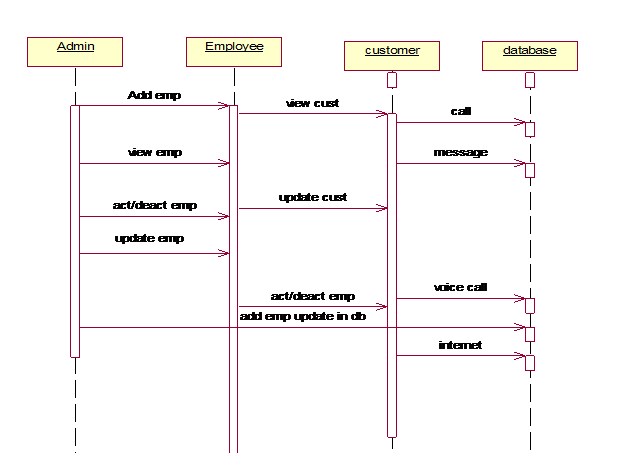


Fig: Sequence diagram

**5.3 ACTIVITY DIAGRAM:**

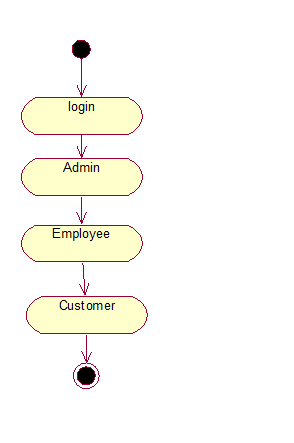


Fig: Activity diagram

IMPLEMENTATION

6.1 MODULES

Admin Module

Employee Module

Customer Module

6.2 MODULES DESCRIPTION

Admin Module: In this module administrator have all the rights on employee & Customer. In our project he only interacts with Employee. In this admin can view all the employee information, add new employee, deactivate employee, update employee information. Admin can view all the employee’s information just click on view employee button it will display all the information about all employee’s involved in organization.

He can add new employee by click on add employee button, the registration form will appear, fill all the details and click submit button it will be stored on employee database table, just execute the query once it automatically display on employee table. He can deactivate the employee just click on deactivate employee button, just specify the employee id. If he update the information of particular employee just give id of employee and update the information, click on update button it automatically store employee table.

Employee Module: In this module employee have all the rights on customers. Employee can perform the following tasks: View all the customer details, update the customer details, active & deactivate the customer details, Add new plan, update the plan, change the password of customer. In this view all customer information based on date, mobile number. Update customer information based on id he update the information. Activate/ Deactivate the customer based on id. He add new plan just click on new plan button all specify all the details. If he want to update the existing plan click on update button and update the information.

Customer: In this module customer can register based on registration form of customer. After register customer can call, send messages, use internet, voice call based on transaction id. Customer choose call they select service type like local, std etc. Based on customer selection money will detect from available balance. Customer can see the how much money available in his mobile, customer can recharge.

SAMPLE CODING

In Customer registration form after filling all the details & click Submit Button

Customer Submition code:

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

{

int x = 0;

if (FileUpload1.PostedFile != null && FileUpload1.PostedFile.FileName != "" )

{

byte[] imageSize = new byte

[FileUpload1.PostedFile.ContentLength];

HttpPostedFile uploadedImage = FileUpload1.PostedFile;

uploadedImage.InputStream.Read

(imageSize, 0, (int)FileUpload1.PostedFile.ContentLength);

string simno = "0";

cmd = new SqlCommand("select count(\*) from mobcust", con);

SqlDataReader dr;

con.Open();

dr = cmd.ExecuteReader();

if (dr.Read() == true)

{

simno = dr[0].ToString();

}

con.Close();

TextBox16.Text = (Convert.ToInt32(simno) + 893573221008).ToString();

cmd = new SqlCommand("select \* from Aadhar where Uid='" + TextBox2.Text + "'", con);

SqlDataReader d;

con.Open();

d = cmd.ExecuteReader();

if (d.Read() == true)

{

if (TextBox1.Text == d[1].ToString() && TextBox7.Text==d[4].ToString())

{

x = 1;

}

}

con.Close();

if (x > 0 && Label4.Text == "UserName Available")

{

cmd = new SqlCommand("insert into mobcust values('" + TextBox6.Text + "','" + TextBox1.Text + "','" + TextBox2.Text + "','" + TextBox3.Text + "','" + TextBox4.Text + "','" + TextBox16.Text + "','" + TextBox7.Text + "','" + TextBox8.Text + "','" + TextBox9.Text + "','" + TextBox10.Text + "','" + DropDownList1.Text + "','" + TextBox11.Text + "','" + DropDownList2.Text + "','" + TextBox12.Text + "','" + TextBox13.Text + "','" + TextBox14.Text + "','" + TextBox15.Text + "','" + RadioButtonList1.Text + "','" + DropDownList3.Text + "',0,'none',@Image,0,0,NULL,NULL)", con);

SqlParameter UploadedImage = new SqlParameter

("@Image", SqlDbType.Image, imageSize.Length);

UploadedImage.Value = imageSize;

cmd.Parameters.Add(UploadedImage);

con.Open();

int result = cmd.ExecuteNonQuery();

con.Close();

if (result > 0)

{

Label1.Visible = true;

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

Label1.Text = "Image Uploaded & Registered Successfully";

Label2.Visible = true;

Label2.Text = "Sim no is:";

TextBox16.Visible = true;

}

Label4.Visible = false;

Label3.Visible = false;

}

else if (x != 1)

{

Label3.Visible = true;

Label3.ForeColor = System.Drawing.Color.Red;

Label3.Text = "invalid Aadhar details";

}

else

{

Label4.ForeColor = System.Drawing.Color.Brown;

Label4.Text = "User name not Available";

Label3.Visible = false;

}

}

}

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

{

Response.Redirect("home.aspx");

}

protected void TextBox3\_TextChanged(object sender, EventArgs e)

{

cmd = new SqlCommand("select \* from mobcust where userid= '"+TextBox3.Text+"'",con);

con.Open();

SqlDataReader dr = cmd.ExecuteReader();

if (dr.HasRows)

{

Label4.Visible = true;

Label4.Text="UserName not Available";

}

else

{

Label4.Visible = true;

Label4.Text="UserName Available";

}

con.Close();

}

}

After filling the details of Employee & click Submit Button:

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

{

if (FileUpload1.PostedFile != null && FileUpload1.PostedFile.FileName != "")

{

byte[] imageSize = new byte

[FileUpload1.PostedFile.ContentLength];

HttpPostedFile uploadedImage = FileUpload1.PostedFile;

uploadedImage.InputStream.Read

(imageSize, 0, (int)FileUpload1.PostedFile.ContentLength);

cmd = new SqlCommand("insert into mobemp values('" + empid.Text + "','" + name.Text + "','" + uid.Text + "','" + usid.Text + "','" + pswd.Text + "','" + dsg.Text + "','" + doj.Text + "','" + dob.Text + "','" + fadd.Text + "','" + mobno.Text + "','" + email.Text + "','" + wloc.Text + "',@Image,'none')", con);

SqlParameter UploadedImage = new SqlParameter

("@Image", SqlDbType.Image, imageSize.Length);

UploadedImage.Value = imageSize;

cmd.Parameters.Add(UploadedImage);

con.Open();

int result = cmd.ExecuteNonQuery();

con.Close();

if (result > 0)

{

Label1.Visible = true;

Label1.Text = "Image Uploaded & Registered Successfully";

}

}

}

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

{

Response.Redirect("admin.aspx");

}

}

Based on mobile no show how many cust added:

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

{

cmd = new SqlCommand("SELECT cid,name,uid,simno,mobno,doa,imei,type,state,Balance,status,Bill FROM mobcust WHERE mobno = '" + TextBox2.Text + "'", con);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

GridView3.DataSource = reader;

GridView3.DataBind();

con.Close();

}

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

{

cmd = new SqlCommand("select cid,name,uid,simno,mobno,doa,imei,type,state,Balance,status,Bill from mobcust", con);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

GridView1.DataSource = reader;

GridView1.DataBind();

con.Close();

}

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

{

Calendar1.Visible = true;

}

protected void back\_Click1(object sender, EventArgs e)

{

Response.Redirect("employ.aspx");

}

}

Based on Date Employee can view the Customer list:

Based on date how many cust added:

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

{

cmd = new SqlCommand("SELECT cid,name,uid,simno,mobno,doa,imei,type,state,Balance,status,Bill FROM mobcust WHERE CAST(FLOOR(CAST( [doa] AS float)) AS datetime) = '" + TextBox1.Text + "'", con);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

GridView2.DataSource = reader;

GridView2.DataBind();

con.Close();

}

protected void Calendar1\_SelectionChanged(object sender, EventArgs e)

{

TextBox1.Text = Calendar1.SelectedDate.ToShortDateString();

Calendar1.Visible = false;

}

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

{

cmd = new SqlCommand("SELECT cid,name,uid,simno,mobno,doa,imei,type,state,Balance,status,Bill FROM mobcust WHERE mobno = '" + TextBox2.Text + "'", con);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

GridView3.DataSource = reader;

GridView3.DataBind();

con.Close();

}

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

{

cmd = new SqlCommand("select cid,name,uid,simno,mobno,doa,imei,type,state,Balance,status,Bill from mobcust", con);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

GridView1.DataSource = reader;

GridView1.DataBind();

con.Close();

}

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

{

Calendar1.Visible = true;

}

protected void back\_Click1(object sender, EventArgs e)

{

Response.Redirect("employ.aspx");

}

}

Based on mobile number view customer details:

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

{

cmd = new SqlCommand("SELECT cid,name,uid,simno,mobno,doa,imei,type,state,Balance,status,Bill FROM mobcust WHERE mobno = '" + TextBox2.Text + "'", con);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

GridView3.DataSource = reader;

GridView3.DataBind();

con.Close();

}

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

{

cmd = new SqlCommand("select cid,name,uid,simno,mobno,doa,imei,type,state,Balance,status,Bill from mobcust", con);

con.Open();

SqlDataReader reader = cmd.ExecuteReader();

GridView1.DataSource = reader;

GridView1.DataBind();

con.Close();

}

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

{

Calendar1.Visible = true;

}

protected void back\_Click1(object sender, EventArgs e)

{

Response.Redirect("employ.aspx");

}

}

**SYSTEM TESTING**

**8.1 INTRODUCTION TO TESTING**

**Introduction to Testing:**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**8.2 TESTING IN STRATEGIES**

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two Strategies:**

**Black Box Testing:**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness.

The logical flow of the data is not checked.

**White Box testing :**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been Executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.

**Integrating Testing :**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together. In this case the communication between the device and Google Translator Service.

**System Testing :**

Involves in-house testing in an emulator of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

**Acceptance Testing :**

It is a pre-delivery testing in which entire system is tested in a real android device on real world data and usage to find errors.

#### Test Approach :

**Testing can be done in two ways:**

* Bottom up approach
* Top down approach

**Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

Test Case1:

Action: Registration

Input: Fill all required details.

Expected Output: Fill all the fields & click submit button then it will store on database.

Test Case2:

Action: Login

Input: Username, password

Expected Output: If the username and password are correct the control will be transferre to the user’s next page otherwise an error page will be displayed and the user can retry ti login..

Test Case3:

Action: check user details

Input: Mobile number, Calendar.

Expected Output: Based on mobile number or based on calendar employee view the customer details.

Test Case4:

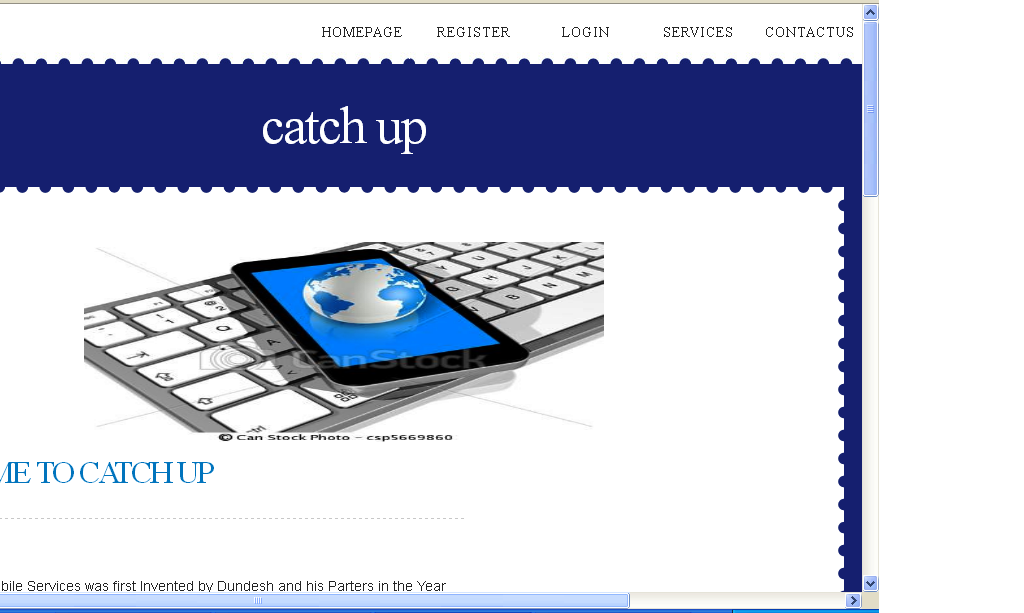
Action: Check employee details.

Input: employee id

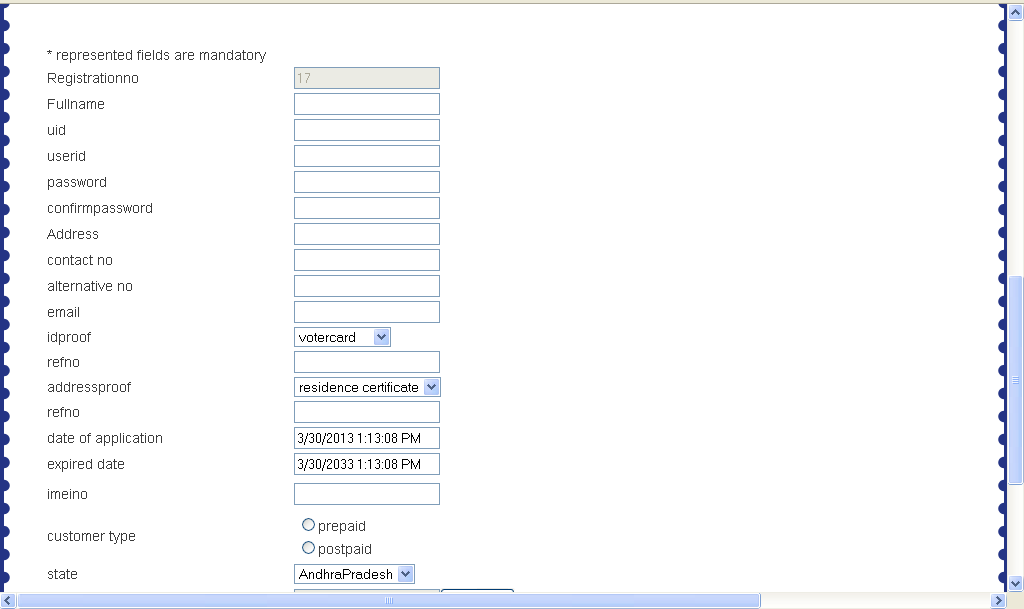
Expected Output: Based on employee id admin can view the employee details.

SCREEN SHOTS:

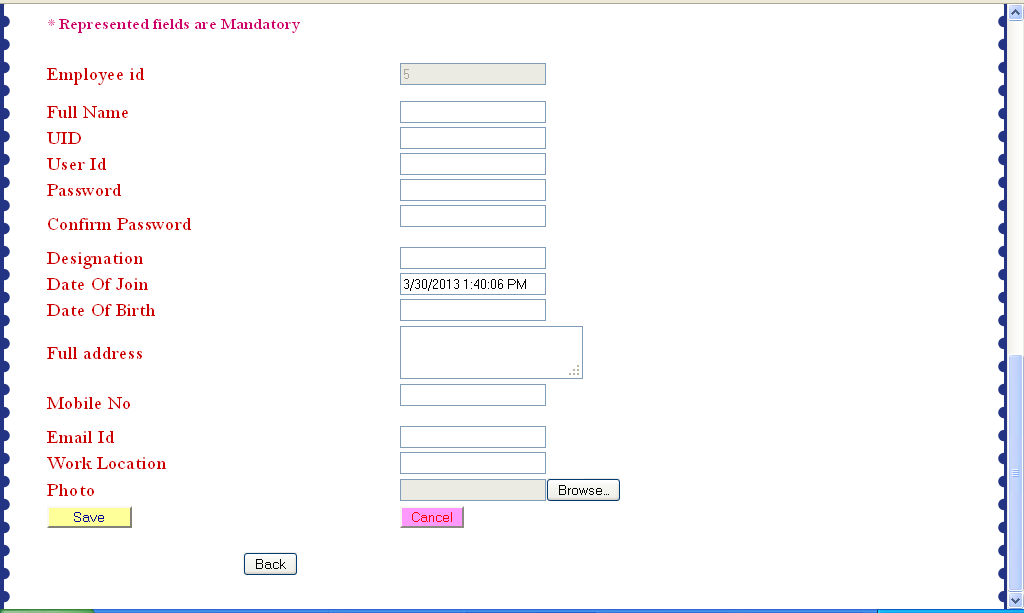
Home Page:



Customer Registration Form:



Employee Registration Form:



Login Form:



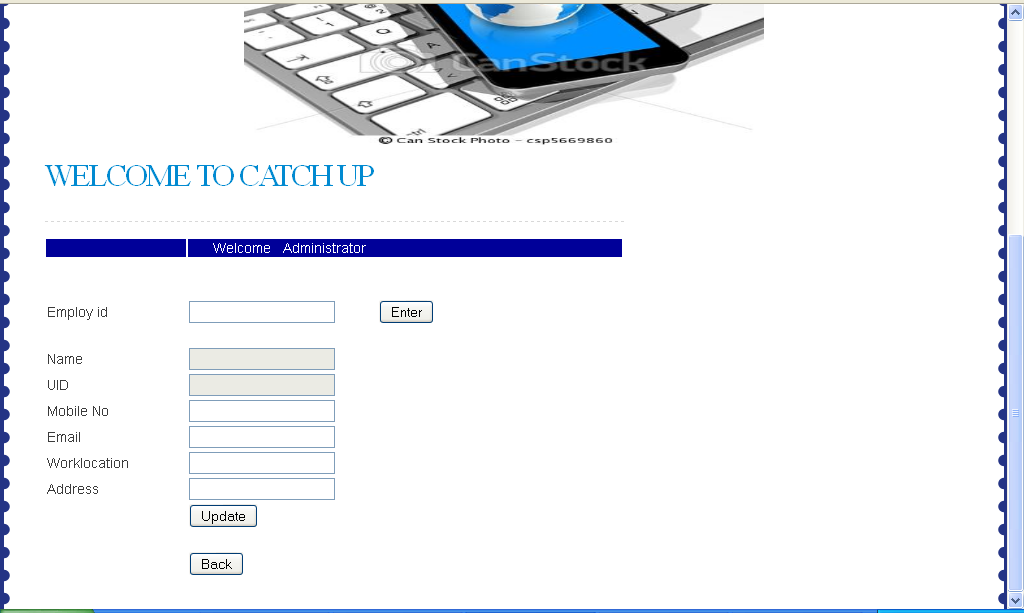
After Admin Login:



View All Employee’s:



Update Employee Information:



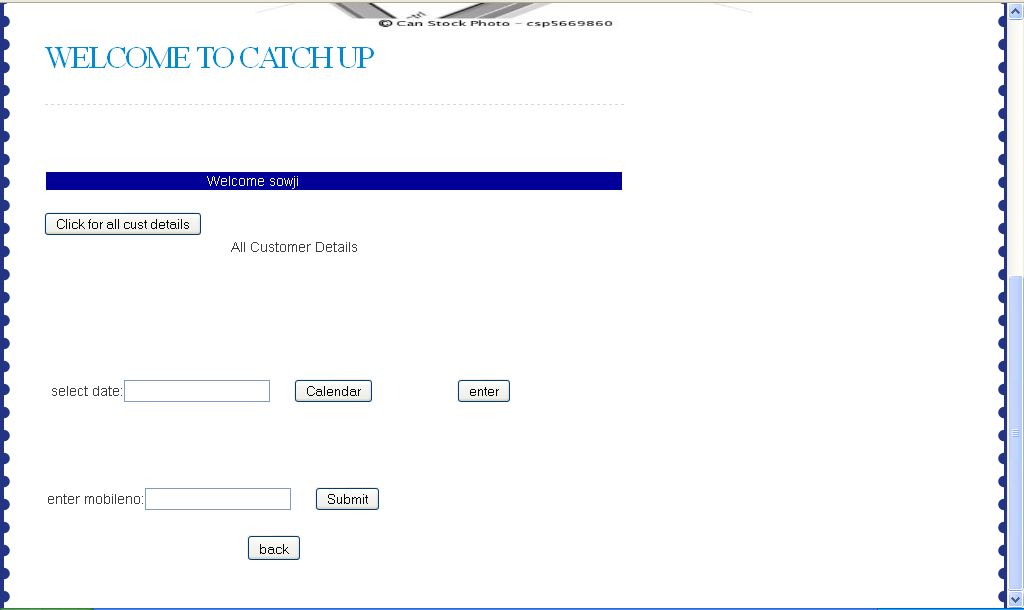
Activate/Deactivate Employee:



After Employee Login:



Employee view the Customer Details Based on Date/Mobile Number:



Activate/Deactivate Customer:



Add new Plan:



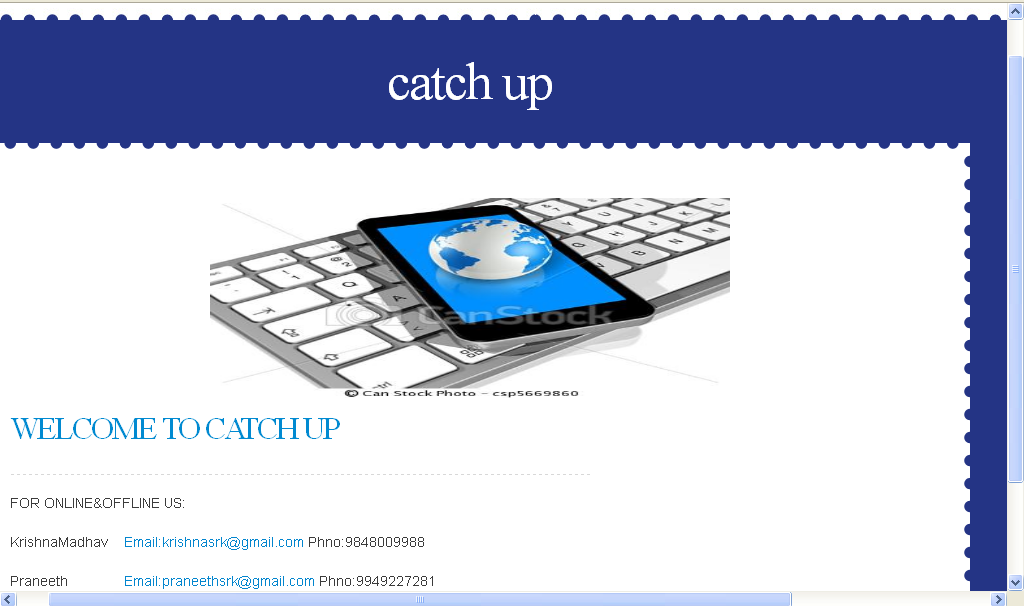
Update Customer Details:



Services:

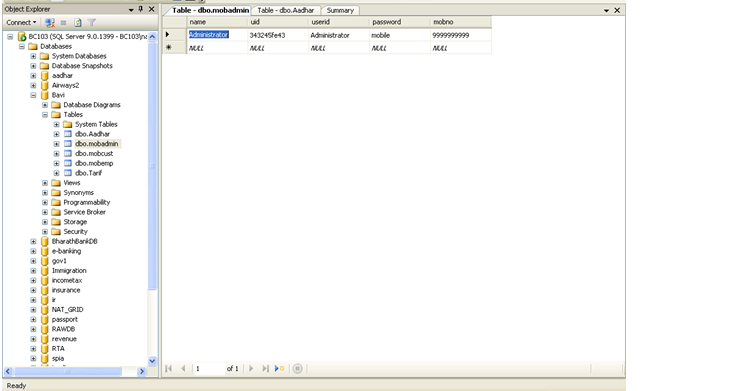


Contact Us:

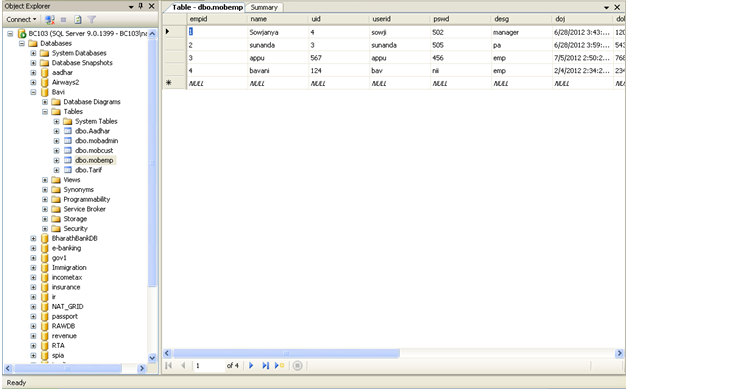


DATABASE TABLES

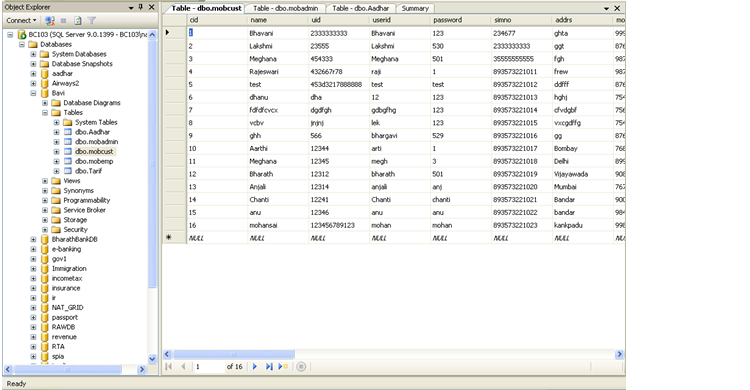
Data Base Table For Admin:



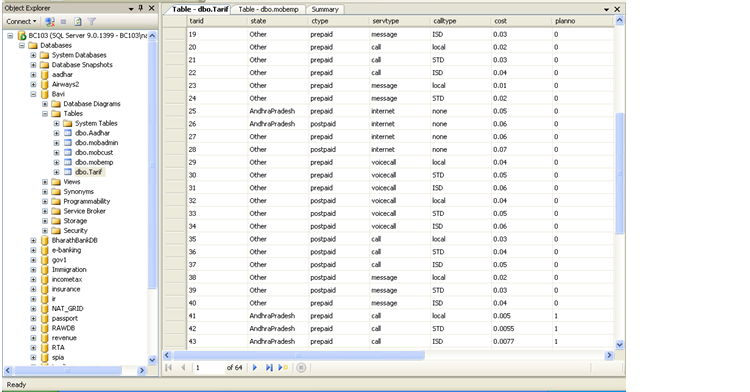
Data Base Table For Employee:



Data Base Table For Customer:



Tariff Data Base Table:



CONCLUSION

In this project we provide services for customer. In this we add new customers, add new employees. Admin have all rights about employees. Admin can add employees, update employee, activate or deactivate employees & view employee details. Employee can view the customer details, update customer details, activate or deactivate customer details. Customer can call; send message, use internet, and voice call. We also provide a friendly environment to the authorized persons of mobile operator to use the service..

Bibliography

 [Scott Guthrie](http://en.wikipedia.org/wiki/Scott_Guthrie) (3 October 2007). ["Releasing the Source Code for the NET Framework"](http://web.archive.org/web/20100907233621/http:/weblogs.asp.net/scottgu/archive/2007/10/03/releasing-the-source-code-for-the-net-framework-libraries.aspx). Archived from [the original](http://weblogs.asp.net/scottgu/archive/2007/10/03/releasing-the-source-code-for-the-net-framework-libraries.aspx) on 07 September 2010. Retrieved 15 September 2010.

 [*MSDN*](http://en.wikipedia.org/wiki/MSDN). [Microsoft](http://en.wikipedia.org/wiki/Microsoft). Retrieved 26 June 2008.

["Scott Guthrie: Silver light and the Cross-Platform CLR"](http://channel9.msdn.com/shows/Going+Deep/Scott-Guthrie-Silverlight-and-the-Cross-Platform-CLR). [Channel 9](http://en.wikipedia.org/wiki/Channel_9_%28discussion_forum%29). 30 April 2007.

 [**^**](http://en.wikipedia.org/wiki/.NET_Framework#cite_ref-4) ["ECMA 335 - Standard ECMA-335 Common Language Infrastructure (CLI)"](http://web.archive.org/web/20080614092650/http:/www.ecma-international.org/publications/standards/Ecma-335.htm). ECMA. 1 June 2006. Archived from [the original](http://www.ecma-international.org/publications/standards/Ecma-335.htm) on 14 June 2008. Retrieved 1 June 2008.

 [**^**](http://en.wikipedia.org/wiki/.NET_Framework#cite_ref-5) ["ISO/IEC 23271:2006"](http://standards.iso.org/ittf/PubliclyAvailableStandards/c042927_ISO_IEC_23271_2006%28E%29_Software.zip). Standards.iso.org. 2006-09-29. Retrieved 2012-04-17.

 [**^**](http://en.wikipedia.org/wiki/.NET_Framework#cite_ref-6) ["Technical Report TR/84 Common Language Infrastructure (CLI) - Information Derived from Partition IV XML File"](http://www.ecma-international.org/publications/techreports/E-TR-084.htm). ECMA. 1 June 2006.

["ECMA-334 C# Language Specification"](http://www.ecma-international.org/publications/standards/Ecma-334.htm). ECMA. 1 June 2006.

["Standard ECMA-372 C++/CLI Language Specification"](http://www.ecma-iointernational.org/publications/standards/Ecma-372.htm). ECMA. 1 December 2005.

 ["Base Class Library"](http://msdn.microsoft.com/netframework/aa569603.aspx). Retrieved 1 June 2008.

 ["Garbage Collection: Automatic Memory Management in the Microsoft .NET Framework"](http://web.archive.org/web/20070703083608/http:/msdn.microsoft.com/msdnmag/issues/1100/GCI/). Archived from [the original](http://msdn.microsoft.com/msdnmag/issues/1100/GCI/) on 3 July 2007. Retrieved 1 June 2008.