REUSABLE COMPONENT FOR MULTIPLE VISUAL REPRESENTATIONS OF DATA

P-2264

By

T. Kalpana
Registration Number: 71205621015

Of

Kumaraguru College of Technology

KCT LIBRARY Coimbatore - 641 006

A PROJECT REPORT
Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements
for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

ANNA UNIVERSITY
CHENNAI 600 025

June 2008

BONAFIDE CERTIFICATE

Certified that this project report titled **"Reusable Component for Multiple Visual Representations of Data"** is the bonafide work of **Ms. T. Kalpana** (Registration Number: 71205621015) who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SUPERVISOR **HEAD OF THE DEPARTMENT**

Submitted to Project and Viva Examination held on 01.07.2008

INTERNAL EXAMINER EXTERNAL EXAMINER

Infosys Technologies Limited
No. 350, Hebbal Electronics City
Hootagalli, Mysore - 570 018
Tel: 91 821 240 4101 Fax: 91 821 240 4200
www.infosys.com

PROJECT COMPLETION CERTIFICATE

This is to certify that **Miss. T. Kalpana** of **Kumaraguru College of Technology,** Coimbatore, Tamil Nadu-641006, was associated with us as a project trainee to carry out her academic project for the partial fulfillment of award of Master of Computer Applications for the year 2008.

She has successfully completed the project titled **"Reusable Component for Multiple Visual Representations of Data"** as per the requirements.

Senior Project Manager
Infosys Technologies Limited,
Mysore

ABSTRACT

This project titled "**Reusable Component for Multiple Visual Representations of Data**" is focused on presenting data to the user in any of the three forms desired by him. As specified in the title, this entire project is developed as a component, using Java and JEE technology, which can later be plugged into any application which wishes to present data in any of these three forms.

Most projects involve certain amount of data interaction and manipulation. Reusable Component for Multiple Visual Representations of Data (RCMVRD) is a data source independent and a standalone desktop component which will display data in form, grid and tree views. These views will provide the CRUD (create, read, update, delete) functionalities to the end user on the data present in the data source. It would be used as a pluggable module for development projects .Data source independence allows for an easier integration and a better reusability of the RCMVRD component. This component is compatible with databases (such as Oracle, MySQL), CSV files and XML files. Data source details will be specified by the developer in an XML file.

The Form view component is designed keeping in mind the extensive usage and importance associated with it in showcasing data source such as database, CSV or XML file in a presentable manner. Some user friendly features in terms of functionalities like, performing **CRUD** (Create, Read, Update and Delete) operations in the form view can be supported The component is designed in such a way that it can fit into an application which has a module performing form view functionality.

The Tree view component is developed with the view of displaying data in a hierarchical form enabling the user to clearly interpret information. The user initiates the component by clicking any node on the tree. User should be able to expand/collapse, add, delete or update in the Tree View.

The Grid view component is carried with the view of displaying information in the data source such as database, CSV or XML file in the form of tables to the user. Supported operations also include add, delete or update in the Grid which in turn enable edit functionalities, thereby displaying data in a data grid to the user.

ACKNOWLEDGEMENT

First and foremost I thank God for his good will and blessings showered on me throughout the project. The success of this project needs cooperation and encouragement from different quarters. Words are inadequate to express my profound and deep sense of gratitude to those who helped me in bringing out this project successfully.

I wish to express my deep unfathomable feeling of gratitude and indebtedness to **Dr. Joseph V. Thanikal, Ph.D., Principal, Kumaraguru College of Technology, Coimbatore** for the successful completion of the project work.

I am very gladly taking this opportunity to express a special word of thanks to **Dr. M. Gururajan, Ph.D., Head of the Department, Kumaraguru College of Technology, Coimbatore** for encouraging me to do this work.

I am very much indebted to **Mr. A. Muthukumar, M.C.A., Kumaraguru College of Technology, Coimbatore** for his complete assistance, guidance and support given to me throughout my project.

It's always a pleasure and privileges to be associated with a prestigious outstanding esteemed organization **"Infosys Technologies Ltd"**, Mysore. I am very happy and grateful to be a part of **Infosys.**

My hearty thanks to my Project mentor **Mr. P. Rajagopalan** and my **E&R guide Miss. Ashwathi SShiva** of **Infosys Technologies Ltd**, for their valuable guidance throughout the project. Also, I am grateful to my parents and friends who were the real source of my project.

TABLE OF CONTENTS

List of Figures:

List of Tables:

List of Abbreviations:

RCMVRD-Reusable Component for Multiple Visual Representations of Data

CRUD-Create, Read, Update and Delete

XML-Extended Markup Language

CSV-Comma Separated Values

JVM-Java Virtual Machine

CHAPTER-I
INTRODUCTION

1.1 ORGANIZATION PROFILE

Infosys Technologies Ltd. (NASDAQ: INFY) was started in 1981 by seven people with US$ 250. Today is a global leader in the "next generation" of IT and consulting with revenues of over US$ 3 billion.

Infosys defines designs and delivers technology-enabled business solutions that help Global 2000 companies win in a Flat World. Infosys also provides a complete range of services by leveraging our domain and business expertise and strategic alliances with leading technology providers.

Infosys has a global footprint with offices in 23 countries and development centers in India, China, Australia, the UK, Canada and Japan. Infosys has over 80,500 employees covering 66 nationalities.

Vision

"To be a globally respected corporation that provides best-of-breed business solutions, leveraging technology, delivered by best-in-class people."

Mission

"To achieve our objectives in an environment of fairness, honesty, and courtesy towards our clients, employees, vendors and society at large."

Industry Leadership

Infosys history is marked by a series of firsts. They were the first Indian company to list on a US stock exchange and the first Indian company to do a POWL in Japan. In December 2006, they became the first Indian company to be added to the NASDAQ-100 index and became the only Indian company to be part of any of the major global indices.

More recently, Infosys was named among the 'Top 10 Companies for Leaders' by Fortune magazine. Infosys won the prestigious Global Most Admired Knowledge Enterprises (MAKE) award for the fourth year in succession

1.2 PROBLEM DEFINITION

Presently there is no composite system to view and manipulate data. It is tedious to view, update data without a graphical interface. Data can be shown in many forms like form, tree, pie chart, bar chart, grid, bar diagram. So there is a requirement to develop such a component that will show data in different formats. A desktop based generic java language component needed to be developed which would connect to popular data sources such as databases, CSV and XML files for displaying the data residing in them. The display of data would have three views:

- Tree
- Grid, and
- Form

This data source independent component will be able to query data sources and perform create, read, update and delete (CRUD) activities despite having no knowledge of the data source till before runtime.

This component would be used by the project developer who may like to incorporate our component into their larger project. This sub component would have a graphic user interface which would allow the developer to set the data source for the other sub component that would display the data to the end user. The project developer would also have an interface to select the data source for the end user when the file and 2 database details are already set and only one can be shown to the end user at a particular moment.

CHAPTER-II
SYSTEM ANALYSIS

2.1 EXISTING SYSTEM ARCHITECTURE

Existing System would have some disadvantages in representing data in different views. It is difficult to View, update data without a graphical interface. Data can be shown in many forms like form, tree, pie chart, bar chart, grid, bar diagram. So there is a requirement to develop such a component that will show data in different formats.

Study of Existing System (DB2 & MSSQL):

Study of GUI of DB2 & MSSQL database have been done, which represents data in a grid format. The DB2 tools have features that improve accessibility for users with visual representation of data. The user can edit data using mouse click also. User can delete a row by selecting the row .User can add data by clicking on the add button. When a user clicks on the add button one row is inserted in the grid and then user will insert data. User can edit data using double click on the data .These databases also provide facility of query builder where the user can build SQL queries.

Problems in existing system

In above case, it is tedious to view data without a graphical interface. This creates a lot of difficulty for projects to view large number of records from data source. Existing system does not support view of data in different representation which is also not user-friendly. It is difficult to understand by all developers of any other application projects which want to view large number of records in user interface screen without interaction of the data source. The user has to connect to the database to represent data with large number of records; this is difficult in case of developers working in larger projects. The developer should be able to configure database for representing data.

2.2 PROPOSED SYSTEM ARCHITECTURE

Presently there is no composite system to view and manipulate data. It is tedious to view, update data without a graphical interface. Data can be shown in many forms like form, tree, pie chart, bar chart, grid, bar diagram. So there is a requirement to develop such a component that will show data in different formats. A generic software component will be developed, which can be plugged in other applications with some modification to view data in tree, form and grid format. The data can also be inserted, updated and deleted into the database tables, CSV or XML files.

The proposed system has two sub-components in it which would incorporate to represent data from different data sources in different views. Developer component would be used by the project developer who may like to use our component into their larger application projects. This sub component would have a graphic user interface which would allow the developer to set the data source such as CSV, XML or any other databases such as Oracle, MySQL for the other sub component that would display the data to the end user.

The project developer would also have an interface to select the data source for the end user when the file and database details are already set and only one can be shown to the end user at a particular moment. The End User sub component would be used by the end user to view the data present in the data source in tree, form and grid displays as set by the project developer. The end user would thus be able to perform different operations on the data records in the data sources.

This component is plugged into any other application project. Proposed system consists of two separate user interface screens with separate menu bars for each screen. Main user interface screen is developer home in which the developer has to select the type of data source to represent data in all the three views. Another user interface screen will contain the menu bar to represent data from the data source in Tree, Grid and Form

views. End user can navigate to any view with the help of the menu bar in the user interface screen.

Block Diagram of the Use and Importance of RCMVRD in a Large Project

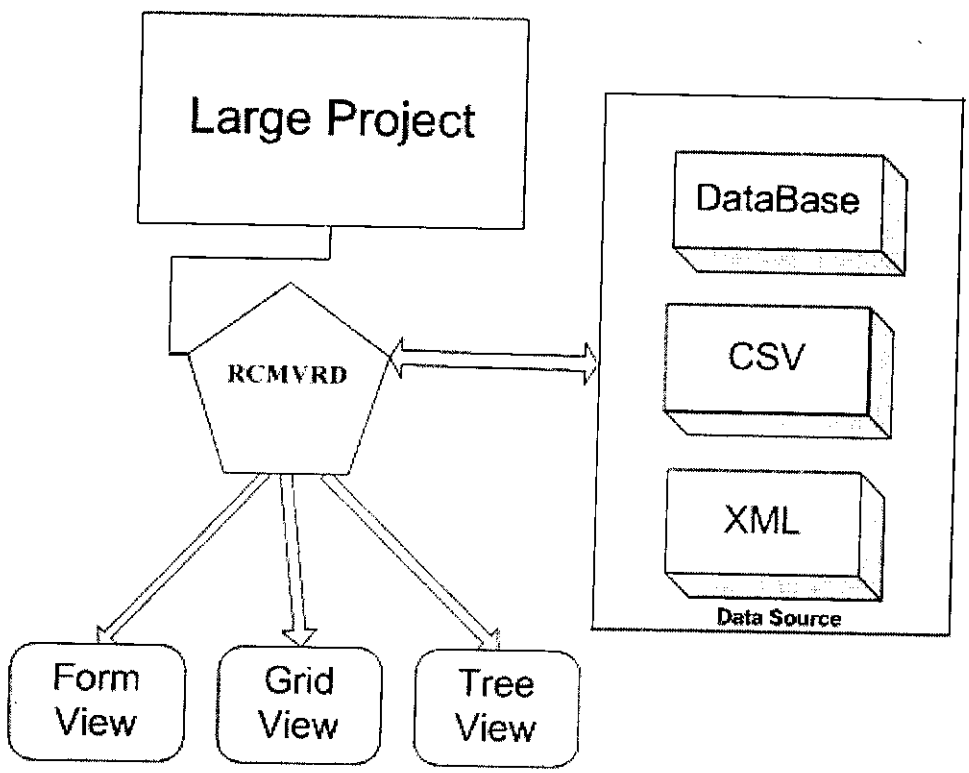


Figure: No: 2.2.1 Block Diagram of the Use And Importance of RCMVRD in A Large Project

As discussed earlier this component is used for interaction between larger projects and data sources used to retrieve data and to display it in different views. Larger

application projects use RCMVRD to view data in Tree, Grid and Form views from different data sources such as CSV/XML or any other database.

2.3 USER INTERFACE REQUIREMENTS

Scope and Assumptions

The proposed project has following three modules:

1. Developer home.
2. User Interface to view the data.
3. Application home to select the type of view.

The objective of developer Home is to select the Type of Datasource from which the user wants to view the data. If the user select database then the user has to set the database connection details in the user interface. If the user selects CSV/XML File as Datasource then the user has to set the CSV/XML File path selected.

The objective of User interface is to display the data retrieved from the datasource in any of the three views Tree, Grid or Form selected by the user.

The objective of Application Home is to provide users with the type of views available. Provide users with the menu to select the type of view.

Functional Requirements

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Non-functional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

Form View

The form view allows the end user to focus on a single record present in a data source. Java's Swing classes provide no support for a form. Adding a single scroll bar and providing scroll buttons required a lot of coding. The mouse click events were captured and a small segment of code was executed to move the text fields and labels vertically.

Grid View

This view will show all the data present in a data source in a grid format. Grid view will be helpful in carrying out data analysis and thus enhance the decision making ability of the end user.

Tree View

The tree view will provide a hierarchical view of the records. This view will emphasize more on the individual records and the fields within them which will contain the actual data.

Process flow

Process flow of the system is represented as follows:

The system should give the user interface to select the type of data source to view the data. The developer can be able to select the data source. The developer should be able to set the database connection or select CSV/XML File. XML Parser is used to parse the database details and the type of data source selected. CSV Parser is used to parse the CSV data source data and to represent in the three views. XML DOM Parser is used to parse the XML data source data and to represent in all the three views. The End user can be able to view the data in any of the three views by selecting the view from the Application Home. The End user can be able to Add, Update or Delete data in all of the

three views. The user can perform different operations on all the three views. Data stored in one data source would be available in other data source during the representation of data. Menu bar helps in navigating between different views for updating or doing other operations in the data source.

Module Description:

This will be a standalone system, which will use a data source specified by the component user and show data of that data source. It will show data in tree, form or grid view.

1. Tree View

Tree view is one of the most important aspects for representation of data and is dynamic. The tree view differs according to the users requirements. Data is retrieved from the data source and should be displayed in the tree format. User can perform different operations on the tree as mentioned below:

- Add a Node
- Expand/Collapse
- Delete/Update

Add a node:

This functionality is used to create a new node in the tree view. The new node added will contain all the child nodes with the respective column names from the data source. Parsers are used to retrieve the column names from the data source and display it in the child nodes with blank nodes to insert values. The user enters some values in the column child nodes and click on the respective button to complete the operation. If the value entered is not valid then it should give appropriate error message in the separate label at the top the user interface screen for tree view.

Expand/Collapse:

This functionality is used to Expand/Collapse nodes in the tree view. The expand option will unfold all the child nodes. The collapse option will fold all the child nodes with the parent node. A plus button will appear if the parent is in collapsed mode and a minus button will appear if it is expanded to display all the child nodes.

Delete/Update:

This functionality is used to delete or update node in the tree view. If the user does not select any node and click on the delete button will display the appropriate error message. If the user select a node for deleting and click on the delete button will delete data from the data source. For updating a node the user have to select node and enter new value for updating. If the value entered is valid then the data will be updated in the data source. If the user entered some inappropriate value then appropriate error message will be displayed in the user interface screen.

2..Form View

Form view is another aspect to display data from the data source. It is used to display data one by one. We can add controls in the form to perform different operations. User can perform different operations on Form as mentioned below:

- Add Data.
- Update Data.
- Delete Data.
- Move Between Data (First/Last/Previous/Next)

Add data:

This functionality is used to add new record to the data source. In form view new form is created to add new record in the data source. Form view is provided with scroll bar to view all data if the column size from the data source exceeds screen size of the form. When the user clicks on the save button new record will be inserted in the data

source if the user entered some wrong value then appropriate error information will be displayed on the user interface screen for form.

Update data:

This functionality is used to update data in the data source from the form view. The user can update value by entering the correct value in the appropriate text box and then click on the update button to update data in the data source. If the user entered some wrong value and click on the update button will display appropriate error information in the user interface screen.

Delete data:

This functionality is used to delete data from the data source in the form view. For deleting a record user select a record and click on the delete button then the record selected will be deleted from the data source. If there is no record in the data source delete button will be disabled.

Move between data (First/Next/Last/Previous):

This functionality is used to move between records in the form view. User can be able to view First/Last/Next/Previous records in the form view y clicking on the respective buttons in the user interface screen. If there is no record in the data source then all the buttons are disabled in the form view.

3. Grid View

Grid view is used to display set of data from the data source. User can perform different operations on Grid as mentioned below:

- Add Data.
- Delete Data.
- Update Data.

P-2264

Add data:

This functionality is used to add a new record in the data source. For adding new record in the Grid view user has to create a new row by clicking on the insert button on the user interface screen. The user then inserts the appropriate value in the new row created, click on the save button will add new record in the data source.

Delete data:

This functionality is used to delete an existing row from the data source. The user has to select a row for deleting a record from the data source. If user clicks on the delete button without selecting a row then appropriate error message will be displayed on the user interface screen. If the row selected is valid then the selected record will be deleted from the data source.

Update data:

This functionality is used to update data in the Grid view. The user has to select the row and enter the new value to update data in the data source from the grid view. If the user entered some wrong value then appropriate error message will be displayed on the user interface screen.

System features

Development projects often refer to data sources for processing data and providing information as well as analysis to the end user. For larger development projects data retrieval may form yet another module along with more time consuming and complex modules like algorithm development and their implementation Therefore, a component that:

- is generic in design,
- provides multiple visual representations of data like form, tree and grid,
- can be easily integrated and plugged into a larger project,

- can work with data sources such as databases, CSV and XML,
- fires queries or carries out file input/output operations independent of the data source details, and

Would be of great benefit and help to project developers.

This component would then be called upon to do the data operations for the popular data sources, and reduce project development man hours. It is envisaged that in the long run it will cost effective and efficient project component.

This component will provide easy insertion, updating and deleting data from the data source using the user interface screen. This component is used in larger projects where in need for representing data from the data source which consists of large number of records which is difficult to maintain in the data source.

NON FUNCTIONAL (SYSTEM) REQUIREMENTS

Non-functional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

Performance

Response time should be stable for any number of tables or records in the CSV File. The component should be able to retrieve data from any database, CSV or XML File. Retrieval of data from databases or files should be faster.

Scalability

The Component should work with large input data set or large number of participating nodes. The Component should be easily pluggable for other projects.

Error Logging

If the user has selected some inappropriate operation, then the system should handle the error and display the appropriate error message.

Availability/Reliability

The system should be ready for use at any specified time. The system or component should perform its required functions under stated conditions for a specified period of time.

Security

Security is the condition of being protected against danger or loss. In the general sense, security is a concept similar to safety. The nuance between the two is an added emphasis on being protected from dangers that originate from outside. Individuals or actions that encroach upon the condition of protection are responsible for the breach of security.

The component is more secured that no one can break the system. The user cannot be able to switch over between the user interface screens during the

operations performed in one screen. Effort must be made to utilize standards compliant interfaces in the operating environment software to support this security implementation.

Adaptability

RCMVRD is capable of being readily changed to adjust to new environment and alliances with any data base or data source community. Furthermore, just as there is a need to structure for change, there may be a need to revert to a previous structure when the data base does not provide support for specific database.

Maintainability:

RCMVRD components must be able to be introduced, modified, and replaced with minimum impact on the application environment. Application design must support the maintenance of application and underlying operating environment components with a view to minimizing any loss of availability.

Manageability:

Functionality must be provided In the RCMVRD application to assure the application developers can easily and directly plug-in the component into their project for easy and effective representation of data from different data source.

Usability:

Usability is a term used to denote the ease with which people can employ a particular tool or other human-made object in order to achieve a particular goal. Usability can also refer to the methods of measuring usability and the study of the principles behind an object's perceived efficiency or elegance. This component is very ease of use and can be used by any kind of user who wants to view data from the data source.

CHAPTER-III
DEVELOPMENT ENVIRONMENT

3.1 H/W ENVIRONMENT

The component has been developed using the Main Processor Intel Pentium IV with the RAM speed of 512 MB. The Hard Disk Capacity of the system is 80 Giga Bytes. Monitor size of the system is 17" of *hp* and the keyboard used is *hp* 106 keys.

3.2 S/W ENVIRONMENT

I. Technologies

- Advanced Java

II. Tools

- Microsoft Visio
- Eclipse IDE

III. Data sources

- Oracle9i
- MySQL
- XML
- CSV

Advanced JAVA:

Java was conceived by James Gosling, Patrick Naughton, Chir Warth, Ed Frank and Mike Sheridan at Sun Microsystems in 1991. The language originally named as "oak" but was renamed as Java in 1995. Now Java is one of the most widely used language in the world of internet and programming.

The original impetus for java was not the internet; instead the primary motivation was the need for a platform independent language that could be used to create software to be embedded in various electronics devices.

Java Virtual Machine:

JVM is used to solve both the security and portability problems. Bytecode which is the output of any java program is a highly optimized set of instructions designed to be executed by the Java Run Time System, which is called JVM.

Eclipse

Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle.

A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform.

Eclipse is an open-source Integrated development environment (IDE) written primarily in Java. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its capabilities by installing plug-

ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Java Applet:

An Applet is an application designed to be transmitted over the internet and executed by a java compatible Web Browser. An Applet is a tiny java program, dynamically downloaded across the network just like an image. Application is a program that can react to user input and dynamically change. An **applet** is a software component that runs in the context of another program, for example a web browser. An applet usually performs a very narrow function that has no independent use.

Examples of applets are Java applets and Flash movies. Another example is the Windows Media Player applet that is used to display embedded video files in Internet Explorer (and other browsers that support the plug-in). Some plug-in also allow for displaying various 3D model formats in a web browser, via an applet that allows the view of the model to be rotated and zoomed. Many browser games are applet-based, though some may develop into fully functional applications that require installation.

JAVA Features:

Java is the platform-independent language and has many applications. The following are some of the features of Java:

Java is Simple:

Java was designed to make it much easier to write bug free code. According to Sun's Bill Joy, shipping C code has, on average, one bug per 55 lines of code. The

most important part of helping programmers write bug-free code is keeping the language simple.

Because Java is simple, it is easy to read and write. Obfuscated Java isn't nearly as common as obfuscated C. There aren't a lot of special cases or tricks that will confuse beginners.

Java is Object-Oriented:

When a program is run messages are passed back and forth between objects. When an object receives a message it responds accordingly as defined by its methods.

Object oriented programming is alleged to have a number of advantages like Simpler, easier to read programs, more efficient reuse of code, faster time to market and more robust, error-free code.

Java is Multithreaded:

Java is inherently multi-threaded. A single Java program can have many different threads executing independently and continuously. Three Java applets on the same page can run together with each getting equal time from the CPU with very little extra effort on the part of the programmer.

This makes Java very responsive to user input. It also helps to contribute to Java's robustness and provides a mechanism whereby the Java environment can ensure that a malicious applet doesn't steal all of the host's CPU cycles.

Java is Dynamic:

Java does not have an explicit link phase. Java source code is divided into .java files, roughly one per each class in your program. The compiler compiles these

into .class files containing byte code. Each .java file generally produces exactly one .class file.

More importantly, classes that were unknown to a program when it was compiled can still be loaded into it at runtime. For example, a web browser can load applets of differing classes that it's never seen before without recompilation.

Java is Robust:

The type checking of Java is at least as strong as that of C++, so that Java programs are well-checked at compile-time. Strong run-time checking by the interpreter catches many other errors. Type checking and interpretation make Java programs crash-proof. The program can fail, but not crash the rest of the system. Many of the errors that hang systems, such as bad subscripts and bad pointers, are specifically detected, preventing a Java program from damaging the rest of the system.

Java is Automatic Garbage Collection:

You do not need to explicitly allocate or deallocate memory in Java. Memory is allocated as needed, both on the stack and the heap, and reclaimed by the *garbage collector* when it is no longer needed. There's no malloc(), free(), or destructor methods. There are constructors and these do allocate memory on the heap, but this is transparent to the programmer.

Java is Architecture-neutral:

The Java compiler does this by generating byte code instructions which have nothing to do with particular computer architecture. Rather, they are designed to be both easy to interpret on any machine and easily translated into native machine code

on the fly. To enable a Java application to execute anywhere on the network, the compiler generates an architecture-neutral object file format--the compiled code is executable on many processors, given the presence of the Java runtime system.

ORACLE 9i

The ORACLE or *Oak Ridge Automatic Computer and Logical Engine,* an early computer built by Oak Ridge National Laboratory, was based on the Institute for Advanced Study (IAS) architecture developed by John von Neumann. Oracle is 100% ANSI SQL compatible relational Database management System.

Oracle data base offers capabilities of both relational and object oriented database system. Managing large amount of data could present administrative and performance challenges. This oracle data partitioning helps to minimize the program.

Each of the partition can be managed individually, thereby allowing more efficient management of the database. In Oracle RDBMS all information is stored in simple tables consisting of rows and columns.

Oracle9*i* Database provides efficient, reliable, secure data management for high-end applications such as high-volume on-line transaction processing (OLTP) environments, query-intensive data warehouses, and demanding Internet applications.

Oracle also offers several additional optional database products that enhance the capabilities of Oracle9*i* Database for specific application requirements.Oracle9i has been designed with focus on certain key development areas.

MySQL:

MySQL is a relational database management system (RDBMS) which has more than 11 million installations. The program runs as a server providing multi-user access to a number of databases. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now a subsidiary of Sun Microsystems, which holds the copyright to most of the codebase. The project's source code is available under terms of the GNU General Public License, as well as under a variety of proprietary agreements.

SEQUEL was an earlier IBM database language, a predecessor to the SQL language. The company does not take issue with the pronunciation "My sequel" or other local variations. MySQL is popular for web applications and acts as the database component of the LAMP, BAMP, MAMP, and WAMP platforms (Linux/BSD/Mac/Windows-Apache-MySQL-PHP/Perl/Python), and for open-source bug tracking tools like Bugzilla. Its popularity for use with web applications is closely tied to the popularity of PHP and Ruby on Rails, which are often combined with MySQL. PHP and MySQL are essential components for running popular content management systems

CHAPTER-IV

SYSTEM DESIGN

After finishing system analysis next step is the system development is system design. The design phase focuses on the detailed implementation of the system recommended in the feasibility study. The design phase is a transition from a user–oriented document to a document oriented to the programmers or database personnel. Systems design goes through two phases of development:

- Logical Design
- Physical Design

The data flow diagram shows the logical flow of a system and defines the boundaries of the system. For a candidate system, it describes the inputs (source), outputs (destination), database (files) and procedures (data flow), all in a format that meet the user's requirements.

In logical design we specify the users needs at a level of detail it virtually determines the information flow in and out of a system and the required data resources. The logical design and physical design produce the working system flow by defining the design specifications that tell programmers exactly what the candidate system must do.

In turn we write the necessary programs or modify the software packages that accept input from the user. Then perform the necessary operations through the existing file and produce the reports.

System design gives more detailed view of the system in graphical representation. It gives the over all view of the system through use-case diagrams. Data flows are

represented through data flow diagram. As this component is using sample database the data flow diagram would have only one table to access data from the data source.

4.1 DATA MODEL

4.1.1 PHYSICAL DATA DESIGN

This component is built with a sample data base table administrator which consists of fields of different data types. The physical database design for the administrator table is given as follows:

TABLE NAME: ADMINISTRATOR

COLUMN NAME	DATA TYPE	SIZE
ADMINID	NUMBER	5
FIRSTNAME	VARCHAR2	10
MIDDLENAME	VARCHAR2	10
LASTNAME	VARCHAR2	10
SURNAME	VARCHAR2	10
DEPARTMENT	VARCHAR2	10
DOB	DATE	NOT APPLICABLE
ADDRESS	VARCHAR2	25
PINCODE	NUMBER	6
TELEPHONE	NUMBER	12

TABLE: NO: 4.1.1-ADMINISTRATOR

This table is created in oracle and in MySQL to represent data from the data base data source. This table consists of different data types such as number, varchar2 and date. In this table adminid is treated as primary key which is made non-editable in data representation views.

4.2 PROCESS MODEL

4.2.1 CONTEXT ANALYSIS

Requirements are gathered from the user (other application projects) for the proposed system. Preparation of Software Requirement Specification process has been completed. Preparation of High Level Design and Detailed Level Design documentation has been completed. In the High Level Design document Use case diagram, Activity diagram and Sequence diagram has been created according to the requirements.

In the Detailed Level Design document class diagram and its descriptions are given. All the forms required for this project has been designed according to the requirements of the user (other application projects). Sample Database design has been completed. The system can be used with data source such as CSV/XML or any Database such as Oracle, MySQL.. Client side coding has been generated as per the form designs.

All the valid and invalid conditions are checked through Unit test plan and Integration test plan. Through Defect tracking System defects are removed and appropriate solutions are given for the recorded defects. Boundary Value Analysis has been performed for various input data involved in this project. Various validators such as Required Field Validator, Compare Validator, and Range Validator are used for this process. Then the project has been valuated through Unit testing and Integration testing.

In testing phase various test cases have been identified and the component is unit tested and integration testing has been done. Unit Test Plan and Integration Test Plan documents are prepared based on the different problems encountered during testing or during the development of the component.

4.2.2 USE CASE DIAGRAM

This section lists the use cases or scenarios from the use case model which depict significant, central functionality of the final system.

4.2.2.1 Use-Case: Reusable Component for Multiple Visual Representations of Data

Use-Case Description:

This Use-Case diagram is to represent overall activities in the system. It gives details such as how to provide the data source to represent data in different views. It gives details about how to generate Tree, Grid and Form views to represent data from the data source. The use-case diagram depicts the interaction between the user and the system.

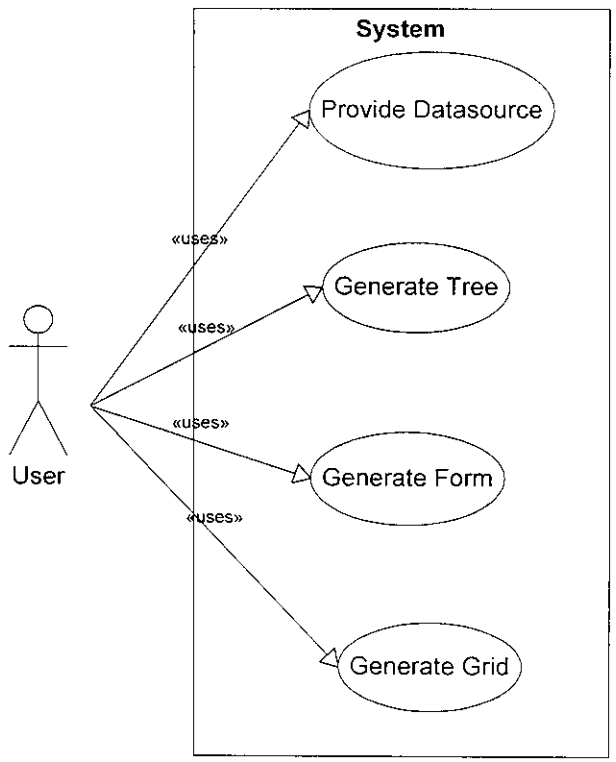


Figure: No-4.2.2.1 Use-Case: Reusable Component for Multiple Visual Representations of Data

4.2.2.2 Use-Case: Provide Datasource

Use-Case Description:

This Use-Case diagram is to represent how to provide different data sources available. This component is mainly developed to represent data from different datasources in different views. There are different datasources such as Database, CSV or XML. If the user wants to represent the data from database then database connection

details is to be provided. If the user wants to represent the data from CSV or XML file then appropriate file path details has to be provided.

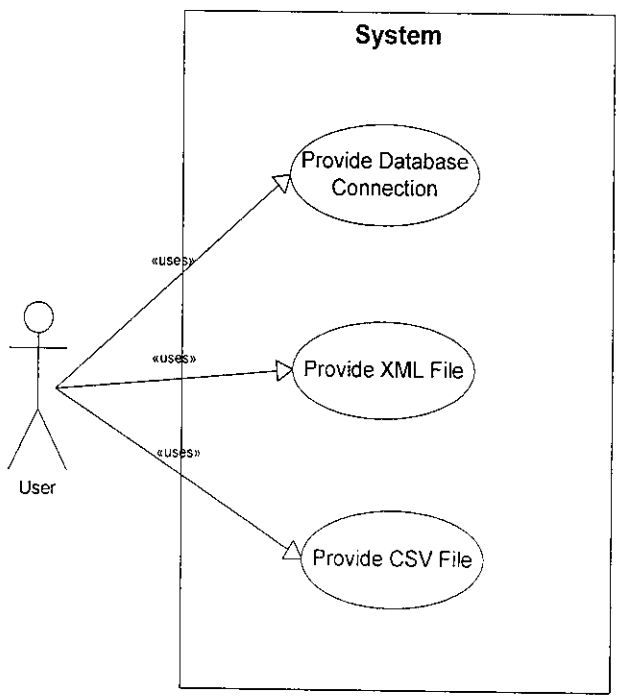


Figure: No-4.2.2.2: Provide Datasource

4.2.2.3 Use-Case: Tree Generation

Use-Case Description:

This Use-Case diagram is to provide details about generating Tree and representing data in Tree view. It provides details about the operations performed on the Tree view. In Tree view user can add, delete and update data. The user should be able to Expand/Collapse nodes in the tree view. The user can easily interact with Tree component and can be to do all the primary operations provided by the tree component.

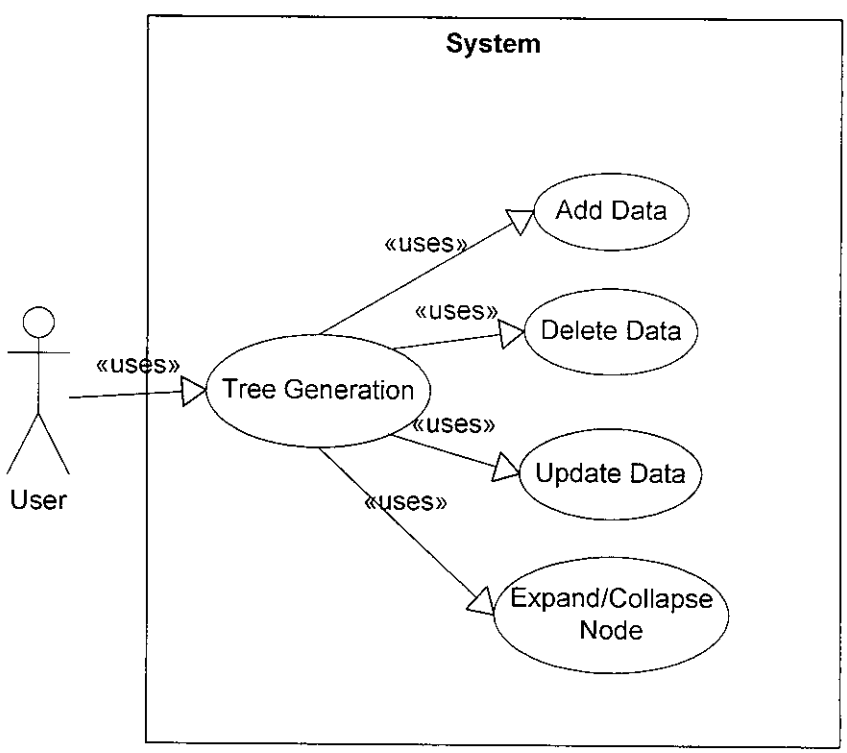


Figure: No-4.2.2.3: Tree Generation

4.2.2.4 Use-Case: Grid Generation

Use-Case Description:

This Use-Case diagram is to provide details about generating Grid and representing data in Grid view. It provides details about the operations performed on the Grid view. In Grid view user can add, delete and update data. The user should be able to add new rows in the Grid view. The user can easily interact with Grid component and can be to do all the primary operations provided by the Grid component.

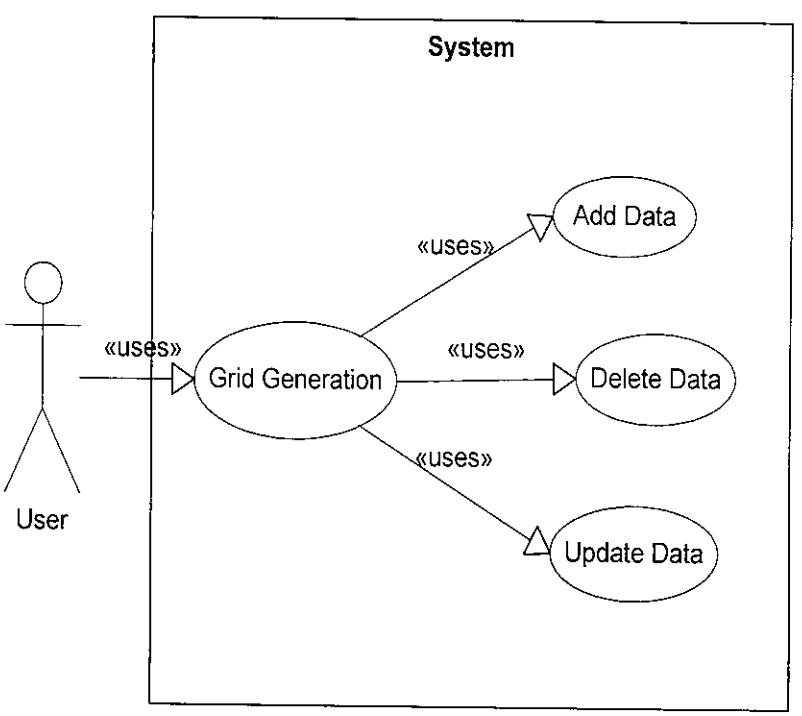


Figure: No-4.2.2.4: Grid Generation

4.2.2.5 Use-Case: Form Generation

Use-Case Description:

This Use-Case diagram is to provide details about generating Form and representing data in Form view. It provides details about the operations performed on the Form view. In Form view user can add, delete and update data. The user should be able to add new record using a separate window in the Form view. The user can easily interact with Form and can be to do all the primary operations provided by the Form. The user can be able to easily move between the records such as First/Next/Last/Previous.

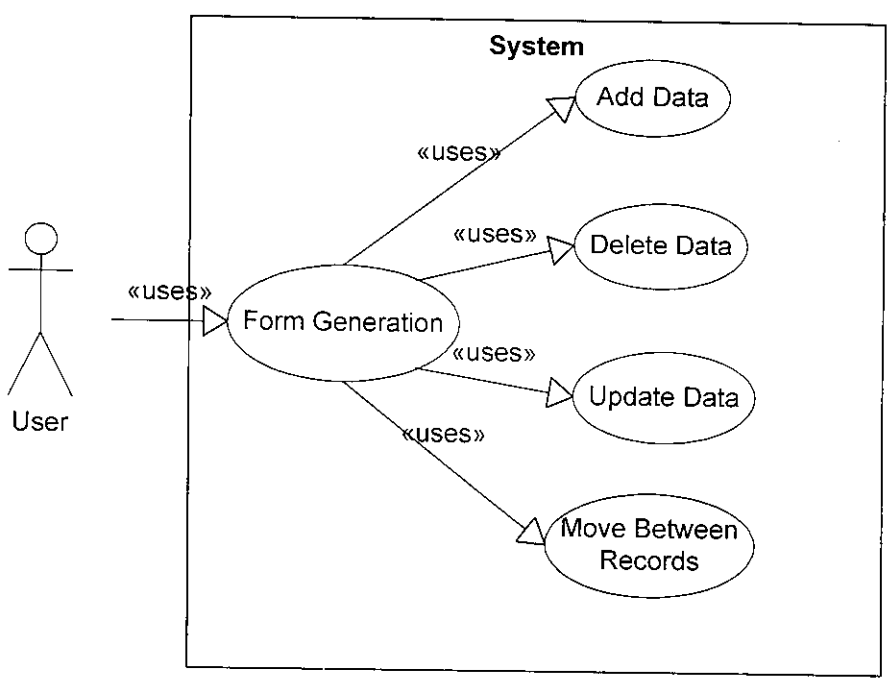


Figure: No-4.2.2.5: Form Generation

4.2.2.6 Use-Case: Move between Records

Use-Case Description:

This Use-Case diagram is to provide details about move between records in the form view. This Use-Case is used to show First/Next/Previous/Last records in then form view. The user will be provided with separate buttons to perform each and every operation. The buttons are enabled and disabled based on the number of records in the datasource and also based on the datasource selected by the user.

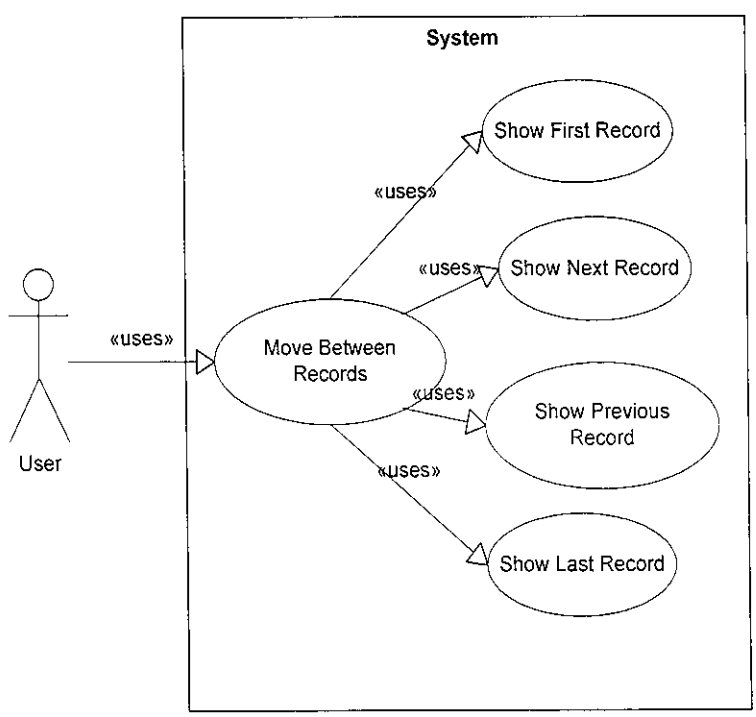
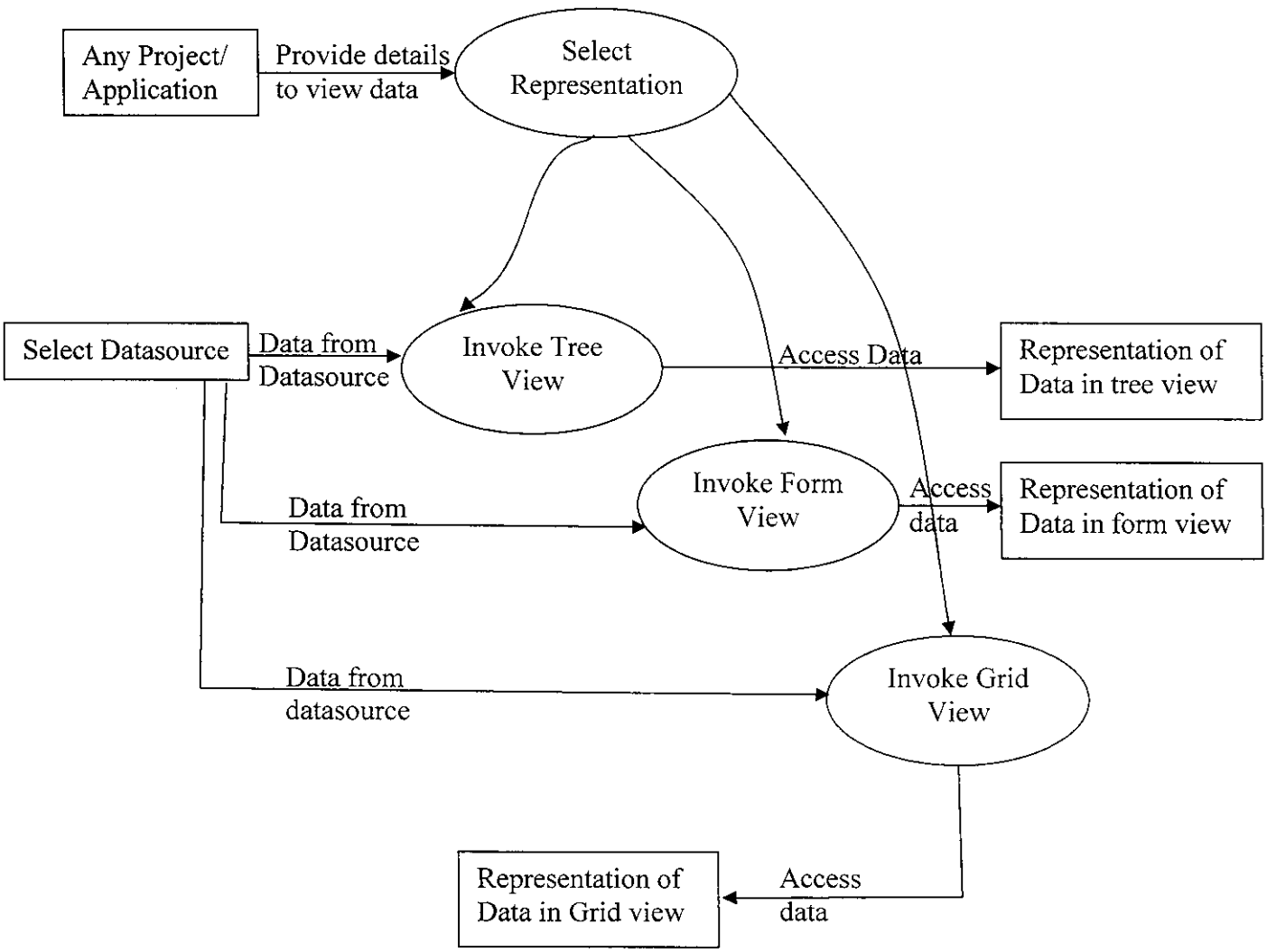


Figure: No-4.2.2.6: Move between Records

4.2.3 DATA FLOW DIAGRAM



CHAPTER-V

ARCHITECTURAL DETAILS

5.1 MVC ARCHITECTURE:

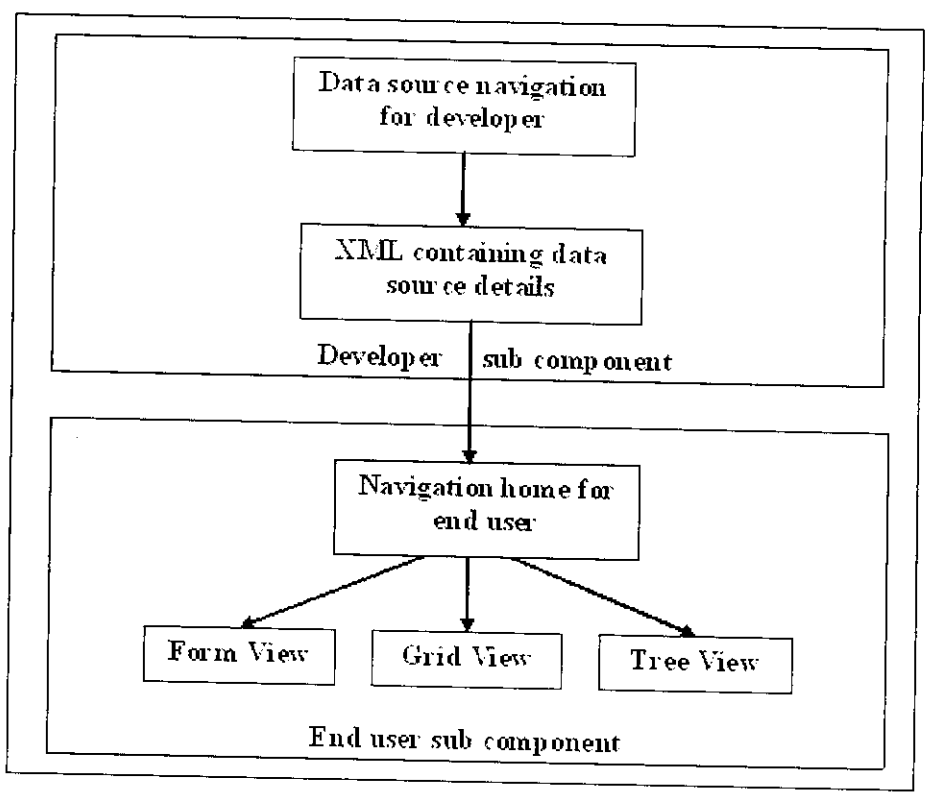


Figure: No: 5.1-MVC Architecture:

Unlike the other frameworks, Java EE defines a pattern for model objects.

Model

The model is commonly represented by entity beans, although the model can be created by a servlet using a business object framework such as spring.

View

The view in a Java EE application may be represented by a Java Server Page, which may be currently implemented using JavaServer Faces Technology (JSF). Alternatively, the code to generate the view may be part of a servlet.

Controller

The controller in a Java EE application may be represented by a servlet, which may be currently implemented using JavaServer Faces (JSF).

In this system the action part is separated from the view part to make users easily understandable. The system has been built with two separate components as

- a developer sub component and
- an end user sub component

The first sub component would be used by the developers to customize the RCMVRD data source (databases, CSV or XML) details and therefore provide the end user with the data that needs to be shown. This sub component will have a graphic interface which will allow the developers to provide appropriate data source connection strings, usernames, passwords, table names and file paths that would be written into a connections detail XML file.

The other sub component will be the one that the end user will use to view the data present in the data source set by the developer in the XML file. The users would be able to perform create, read, update and delete operations on the data because the XML file containing connection particulars will be parsed to obtain the type of data source and other details and then the appropriate methods and functions would be executed that allowed CRUD functionalities on that particular data source.

As this component is very generic in nature and easily pluggable into a larger data source projects, only the data types that are common and present in all databases like varchar, char, number, date and null will be handled by RCMVRD.

RCMVRD was developed using the Java Swing classes. The Java Swing package provides graphic user interface development tools. All Java Swing classes imports from the javax.swing package. Swing classes provide features such as labels, buttons, text boxes, checkboxes, combo boxes, panels, tables, trees and sliders. The display of the component was modified to the Microsoft Windows look by setting the UIManager's look and feel method to Windows look and feel.

CHAPTER-VI

TESTING

6.1 SYSTEM TESTING:-

Testing plays a perfect role to reach the perfectness in any system. It is a major quality control measure used to determine the status and usefulness of the system.

During the testing phase we carried out comprehensive tests on the component, Reusable Component for Multiple Visual Representations of Data (RCMVRD). We tested RCMVRD with all the supported data sources. The database testing was carried out with two vey popular databases, Oracle and MySQL. The common data types like varchar, char, date, number and null were tested and provided positive results.

Database integrity constraints were also tested and upon violating them an easily understandable error message was shown to the end user who would then have to re enter values that adhere to the constraints of the database. CSV and XML data source tests were carried out and CRUD operations were tested. CSV files even allowed the end user to enter commas as values in the comma separated fields. An appropriate error message was displayed to the user when the CSV/XML file was empty or the path provided by the developer in the connections XML was invalid.

The objectives of testing are:-

Testing is the process of executing a program with the intention of finding an error. A good test is one that has a high probability of finding an undiscovered error. A successful test is one that covers an undiscovered error.

Test Execution Procedure:

Test cases are prepared according to design / specifications. Each Test case and the items under test are documented in detail in a set format. The input and the initial

values are then filled in the Unit Test Plan (UTP) as per each test case & theoretical values evaluated. The UTP is then reviewed and executed.

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing plays a vital role in the success of the system. In order to reduce the testing period and the cost of the project, software that is being developed should be carefully designed and should be iterated at every stage of the development in order to ensure that product is being developed without any bugs.

6.1.1 UNIT TESTING

Unit testing focuses verification effort on the smallest unit of software design, software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within boundary of the module. In unit testing each and every small software components and the controls like text fields are validated for errors.

In this component unit testing is done on all the modules. Each and every field is checked in all the three views. In Form view the text fields are validated with different data types available in the database. In Grid view the displays of data in all the columns have been checked. In tree view display of data in nodes have been checked.

6.1.2 MODULE TESTING:

In module testing main modules are tested for errors and failures. After unit testing the modules, which contain the small software component have to be tested. Giving the data to check whether the data entered in the user interface should get saved in the relevant database.

Module Testing is performed to check whether the data entered in one view should get saved in the database and retrieved to see the data in other views. Likewise all modules in the system are system is checked for its performance.

6.1.3 INTEGRATION TESTING:

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. There are two types in integration testing:

- Top-Down approach
- Bottom-Up approach

Integration testing involves test cases designed to validate that end-to-end processes are properly working together. It's a logical extension of unit testing and is performed to establish whether the components interact with each other according to the specification.

Integration testing is accomplished through the execution of predefined business flows, or scenarios, that emulate how the system will run the business. Integration test plan documents provide different test cases that have been tested in the component. Defect Tracker has been identified for both the unit and integration test cases. Defect tracker have been developed for future verification of the component

6.1.4 VALIDATION TESTING:

Validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrates conformity with requirement. Deviation or errors discovered at this project is corrected

prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies.

Various validation tests have been applied in this project. A separate function itself has been written to facilitate easy validation checking. Whenever validation checking is needed the function will be called. The function mainly checks whether the data entered is a valid date and so on.

The purpose of defining a separate function is to facilitate the easy understanding and also to avoid repetition of the same code in different modules. Hence since validation checking is needed in all the places where the user enters some input it is easier to call the function whenever necessary. If any invalid data is entered by the user it prompts the mistake and request to enter the valid data. Thus the proposed system has been tested by using validation testing.

6.2 SYSTEM IMPLEMENTATION:

System Implementation is the process of making the newly designed system fully operational. The system is implemented after completed testing. This is also the phase where there is maximum interaction between other application projects. The most crucial stage is delivering a successful system that will work efficiently and effectively.

Our component, RCMVRD supports CSV and XML data sources as well as databases and thus can provide an easy to use user interface for the customer to perform the basic data manipulation functions in these in these common data sources. Most importantly this component would reduce the repetitive coding in larger data source projects if the project developers chose to plug in this component into their projects.

CHAPTER-VII

PERFORMANCE AND LIMITATONS

A performance-based approach permits the use of any scientifically appropriate method that demonstrates the ability to meet established method performance criteria (e.g., accuracy, sensitivity, bias, precision) and complies with specified data quality needs or requirements. The system pertains to enhancing method technology: development of better, faster, less expensive or new methods to satisfy new or modified programs.

7.1 MERITS OF THE SYSTEM

Merits of the system are as follows:

- This component can have a variety of uses because of its generic nature.
- These days a lot of focus has been given towards the relatively newer data storage formats like CSV's and XML's.
- Our component, RCMVRD supports CSV and XML data sources as well as databases and thus can provide an easy to use user interface for the customer to perform the basic data manipulation functions in these in these common data sources.
- Most importantly this component would reduce the repetitive coding in larger data source projects if the project developers chose to plug in this component into their projects.
- This system is developed with enhanced technology, faster retrieval of data.

- This system is less expensive when used in larger projects which reduces large amount of coding.
- Data is represented in tree and grid view by using the components already available so it is easy to work with the tree and grid view.

7.2 LIMITATIONS OF THE SYSTEM

Limitations of the system are as follows:

- System will not provide support for data types such as CLOB, BLOB.
- Components are used for creating Tree and Grid views.
- More functions are included to represent millions of records from the data source.

7.3 FUTURE ENHANCEMENTS

The system will be able to work in larger projects as it is plugged into any other application projects. The end user of the component should provide some changes in the coding to suite for their application projects. The database connection details are to be provided by end user based on the type of data base the application project is using.

The system should have future enhancements as

- The system should provide support for data types such as CLOB, BLOB.
- Tree and Grid view are created using Components. So that should be changed as per user needs.
- In Tree nodes should be made Non-Editable.

CHAPTER-VIII
APPENDICES

8.1 SAMPLE SCREENS

Screen Description:

This user interface screen is used to select the type of data source such as Database, CSV or XML File path. This screen contains option buttons to select the type of data source to represent the data. The user can be able to select option and click OK button to set the data source selected.

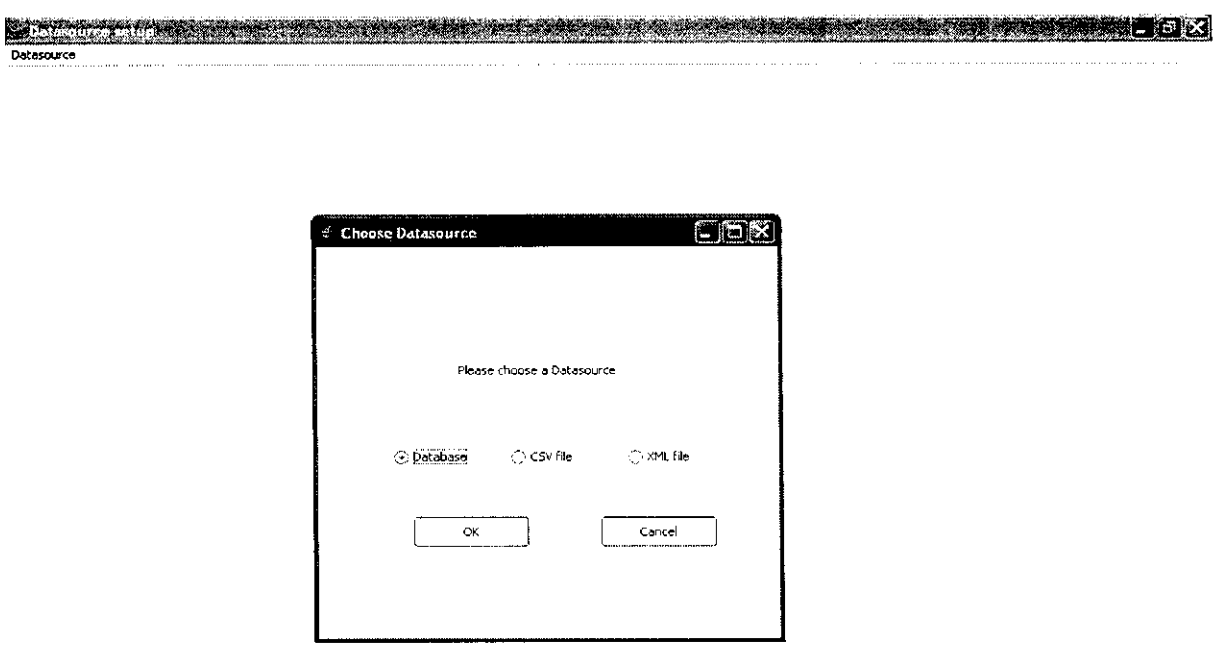


Figure: No: 8.1.1-Selecting the data source

Screen Description:

When the data source chosen is database then set database connection details. The users have to enter the database connection details based on the database selected. The database connection details are different for different databases.

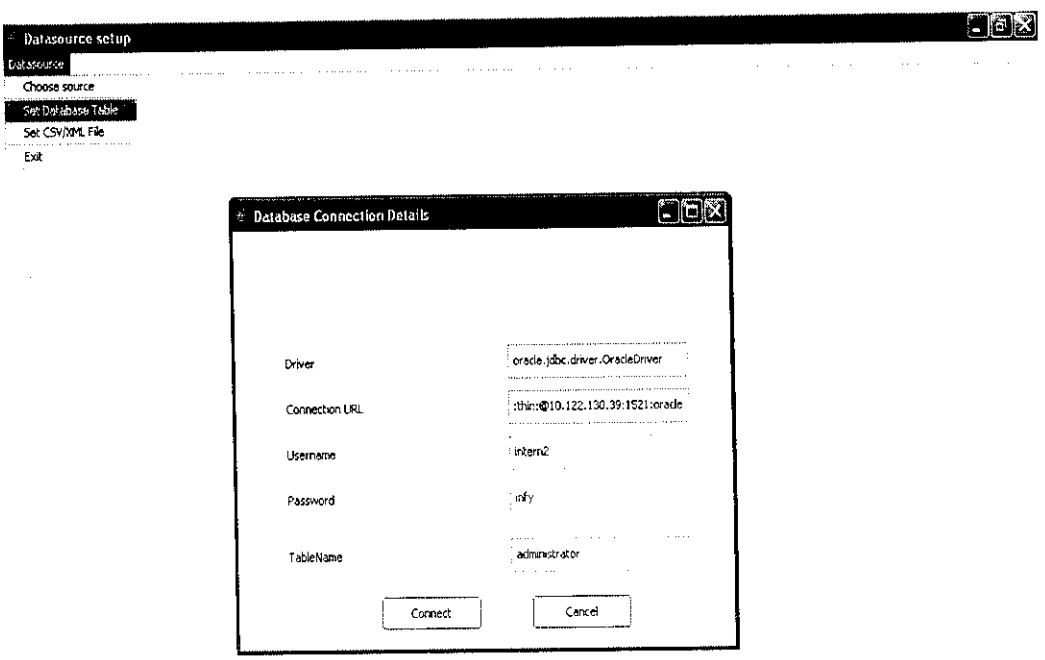


Figure: No: 8.1.2- When the data source chosen is database then set connection string

Screen Description:

When the data source chosen is CSV/XML file then the user has to select the file from which the data is to be represented in all the three views. The user clicks on the connect button to represent the data from the data source in all the views.

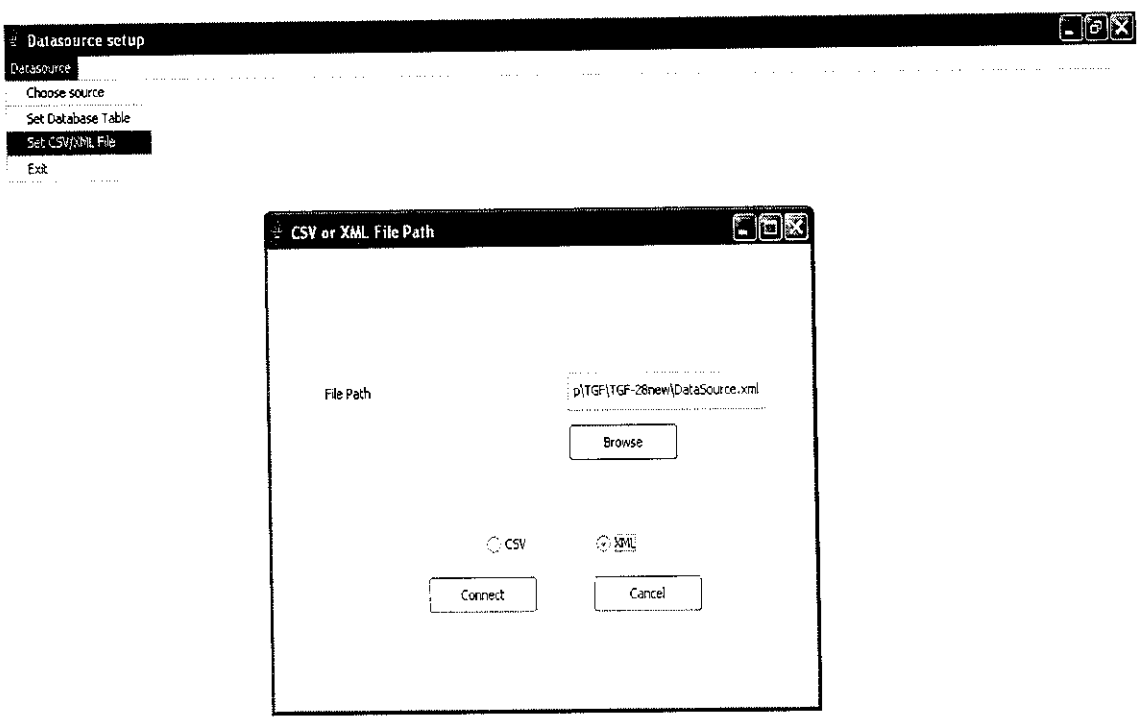


Figure: No: 8.1.3-When data source chosen is XML/CSV file then set the file path

Screen Description:

This user interface screen is to represent data in form view. This display column names and data corresponding to those columns. This screen contains scrollbar for easy view of data if the number of columns is greater than screen size. First/Previous/Next/Last buttons are used to move between records in the form view. This user interface screen is used to update or delete data from the data source.

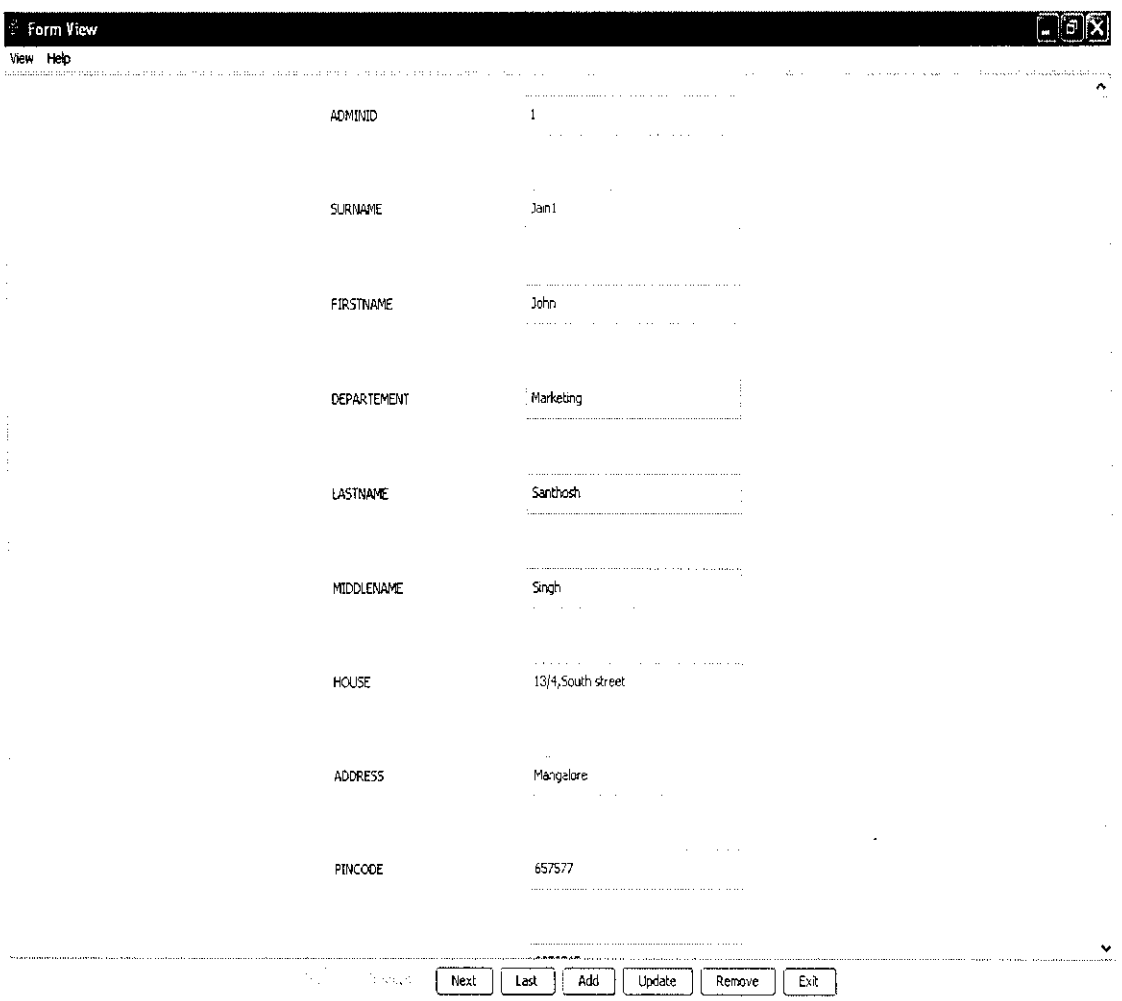


Figure: No: 8.1.4-Data representation using Form View

Screen Description:

This user interface screen is used to add a new record in the data source. New form or window is created to add a new record in the data source. This screen is also provided with the scroll for easy view of data

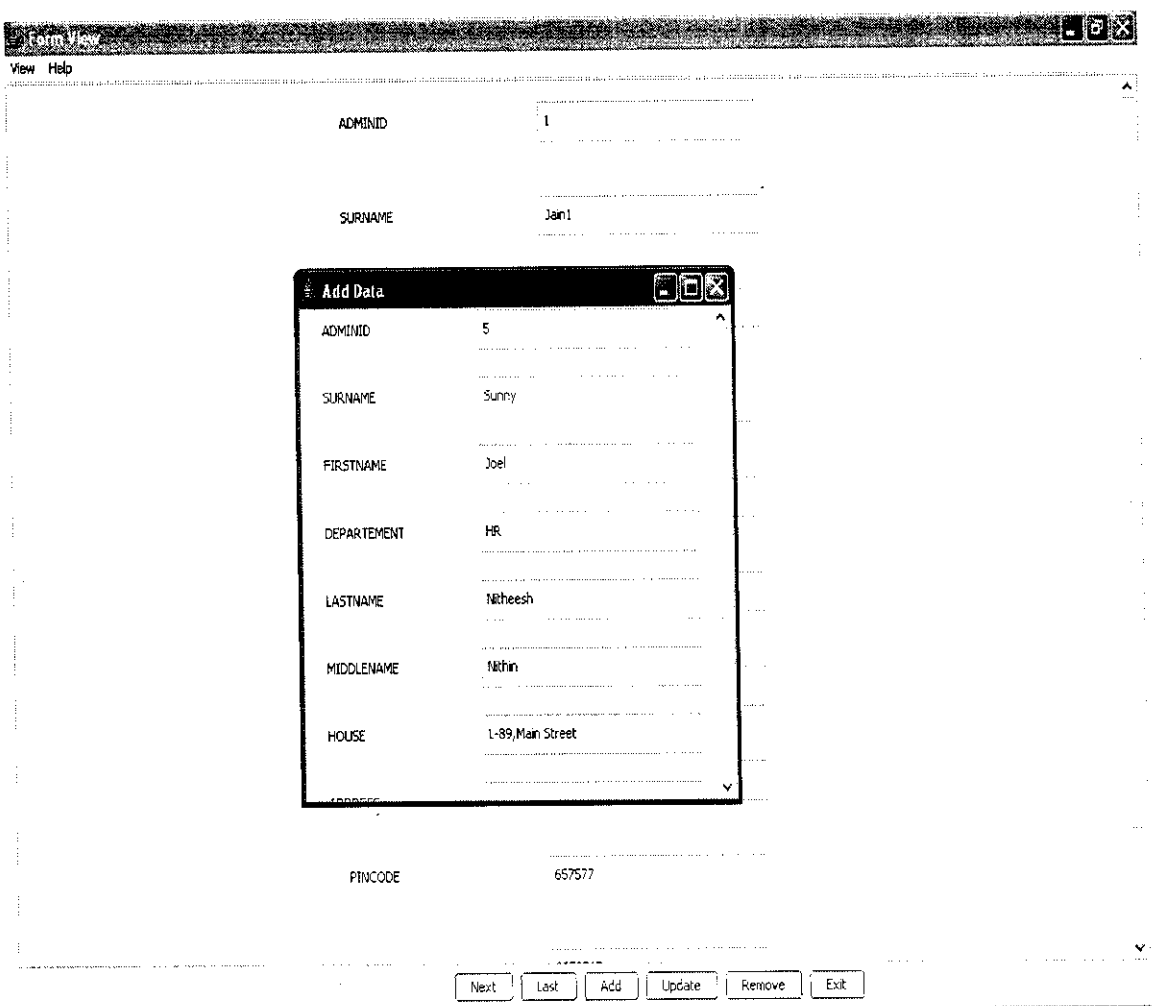


Figure: No: 8.1.5- Adding new record in Form View

Screen Description:

This user interface screen displays the error message in separate label if the user has given any wrong data that is not supported by the data source selected.

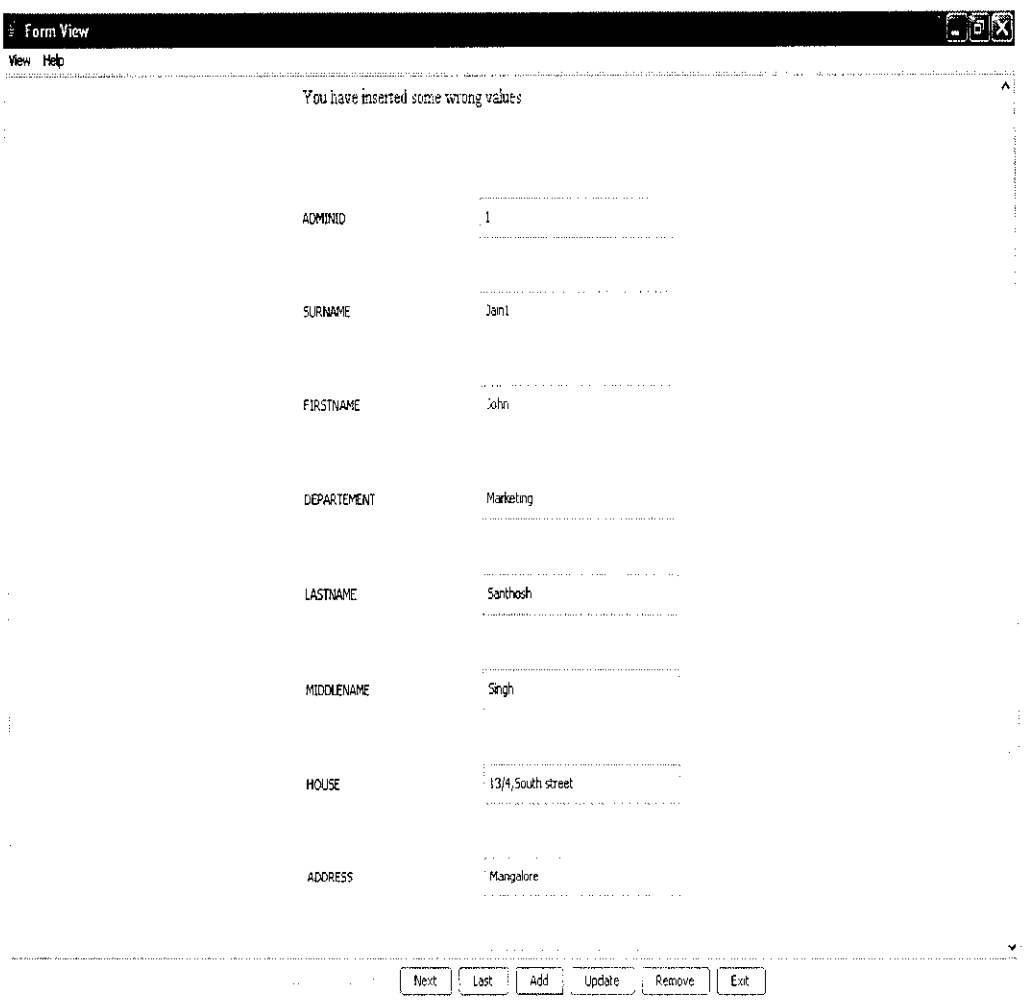


Figure: No: 8.1.6-To Display the error message when wrong or null values are entered in

Form View

Screen Description:

This user interface screen is used to delete the selected row of data from the data source in the form view. Once the user clicks on the delete button the system will prompt the user to confirm deleting the selected row of data.

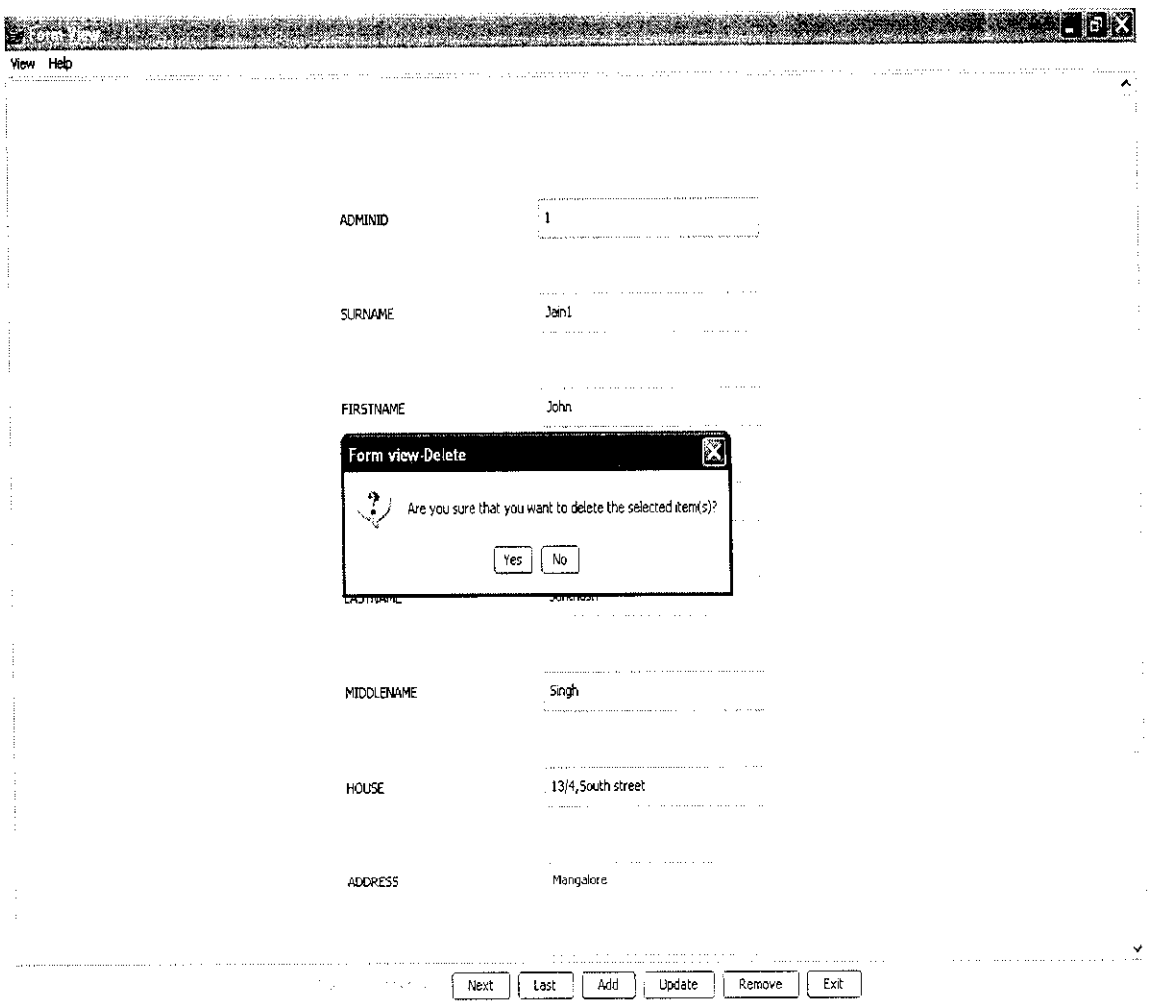


Figure: No: 8.1.7- Deleting a existing record in Form View

Screen Description:

This user interface display data in the Grid format. If the user clicks on the add button a new row will be created to add data in the data source. Save button is clicked to save the data. Update button in the screen is used to update data in the data source.

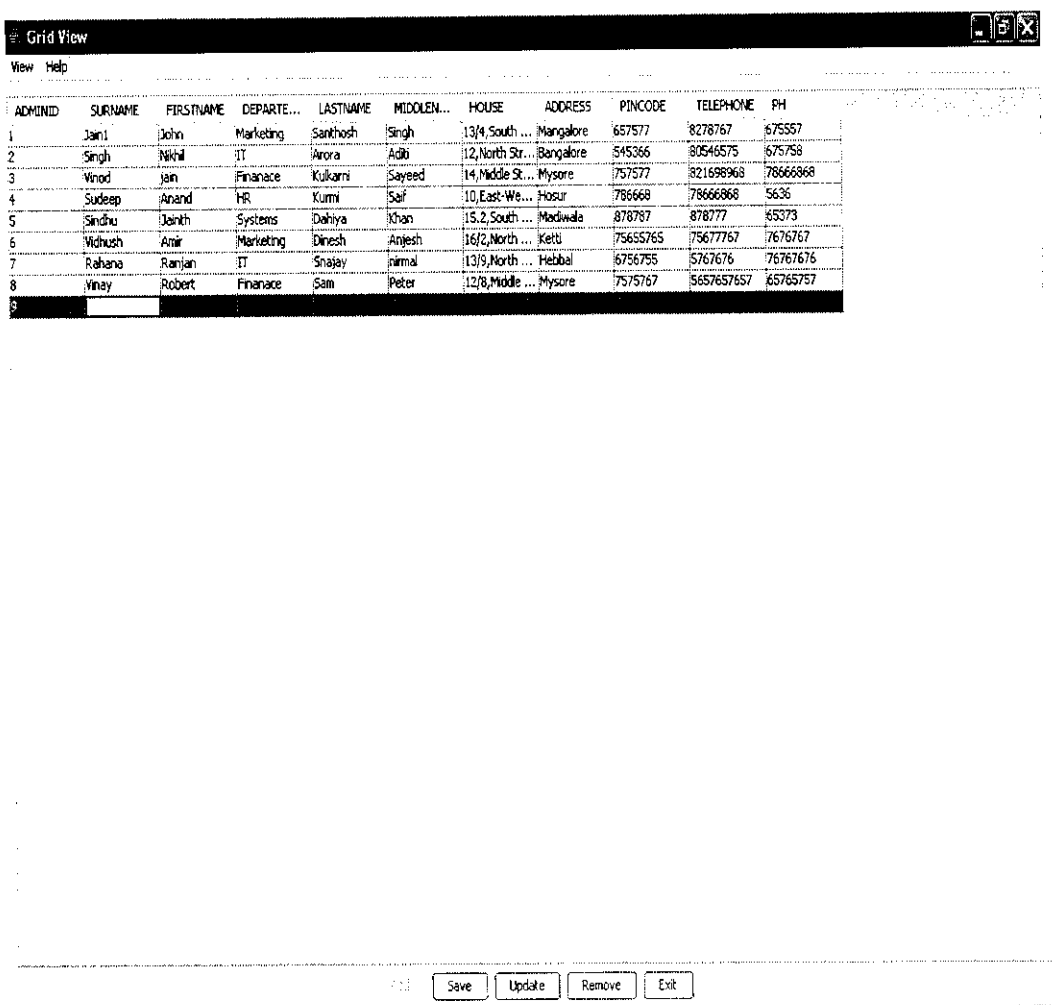


Figure: No: 8.1.8- Data Representation in Grid View

Screen Description:

This user interface screen will delete data from the data source if the row selected is valid for deleting. If the row selected is not valid then it will display the appropriate error message in the user interface screen.

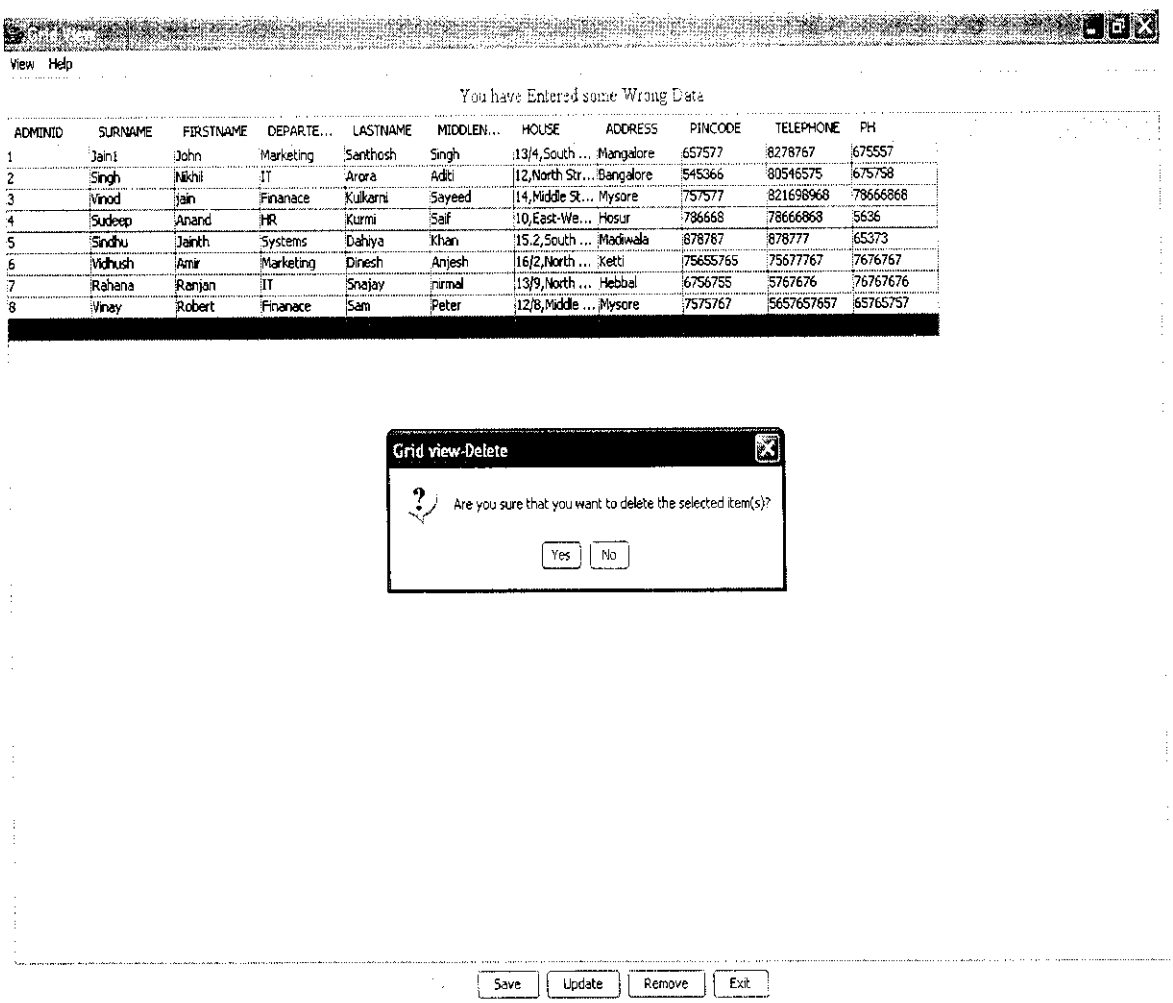


Figure: No: 8.1.9-Deleting an Existing record in Grid View

Screen Description:

This user interface screen is used to insert data in the data source. If the user enters null value in the grid view column then it will display the appropriate error message on the user interface screen.

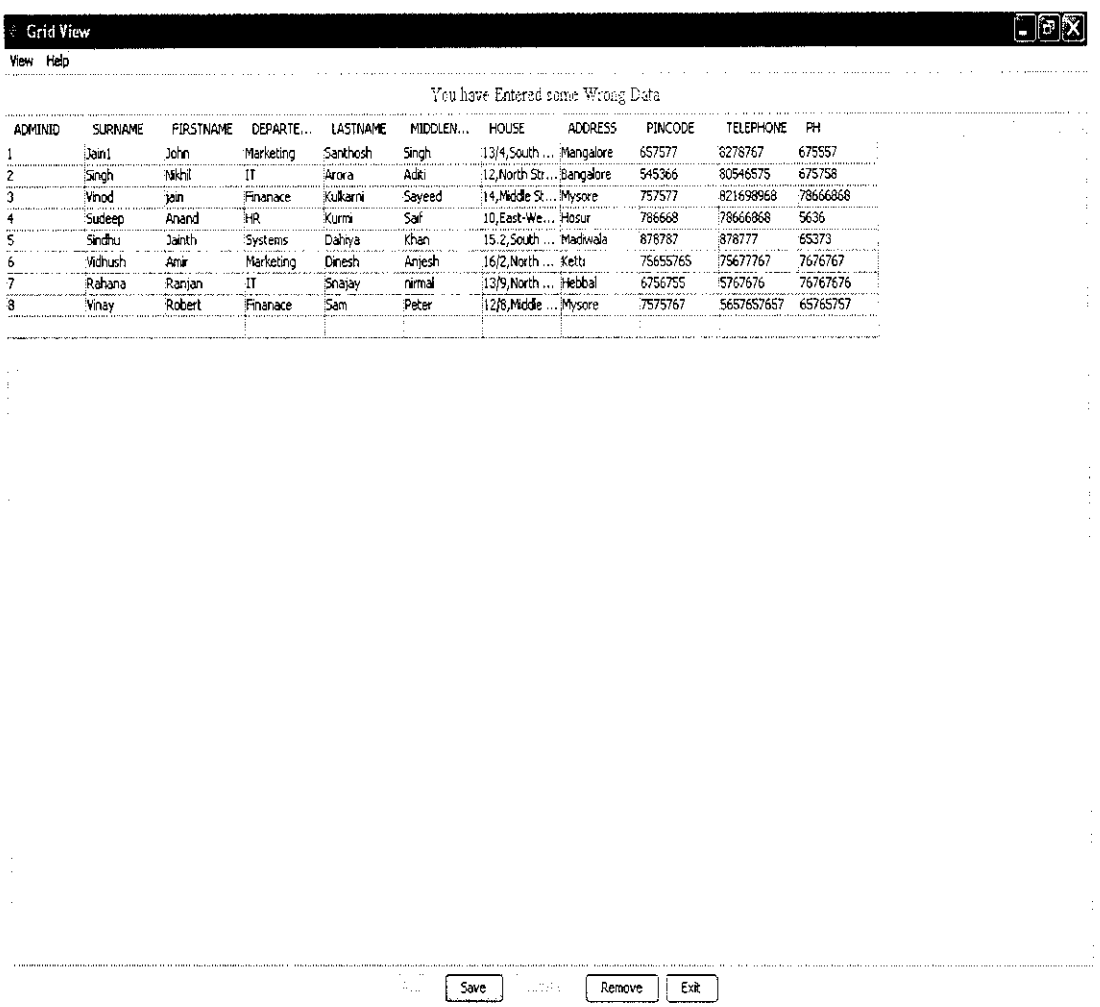


Figure: No: 8.1.10- To display the error message wrong or null values in Grid View

Screen Description:

This user interface screen is to represent data in the tree view. This screen is to depict the scroll bar display in the tree view if the number of columns in the data source exceeds the screen size.

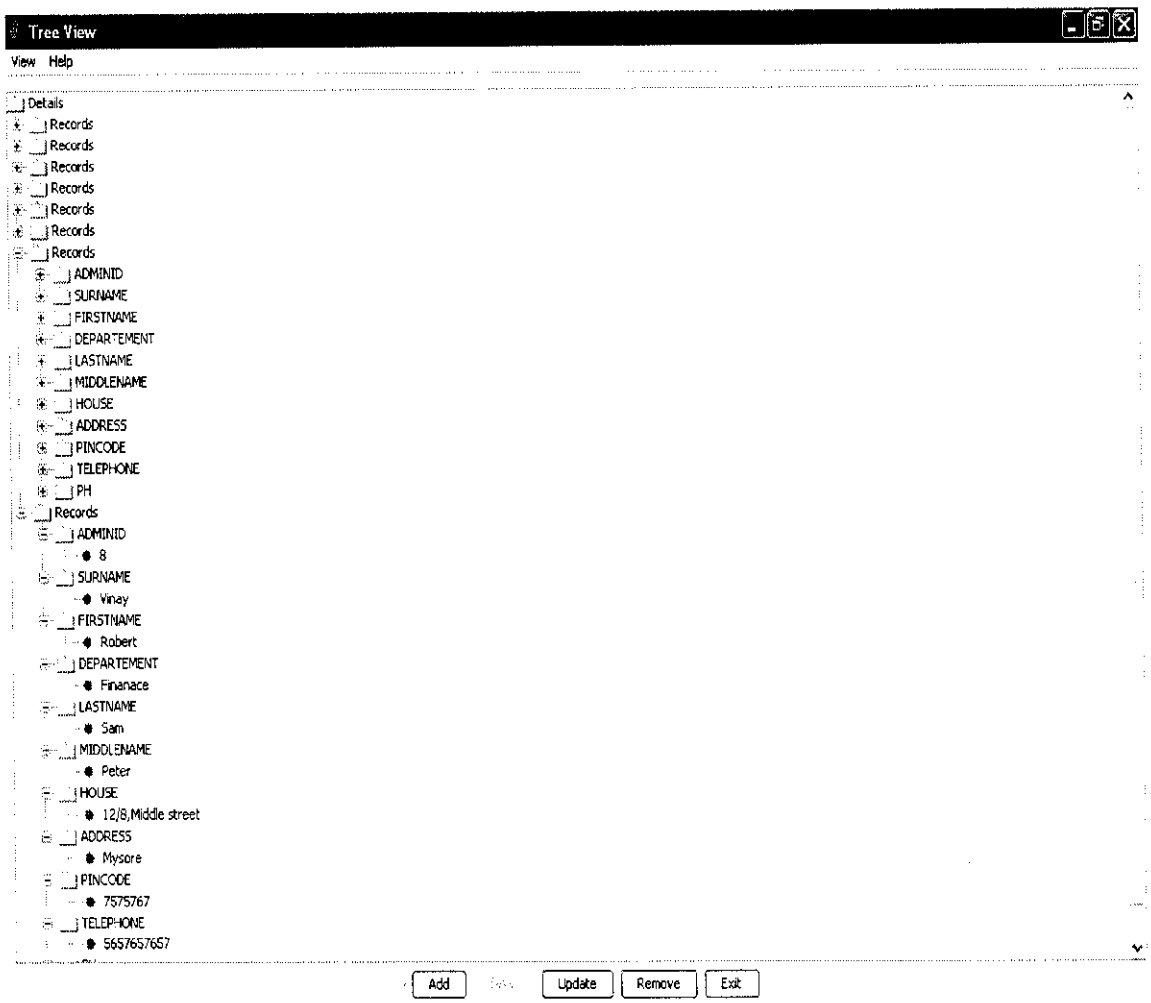


Figure: No: 8.1.11-Data Representation in Tree View

Screen Description:

This user interface is used to represent data in tree view. The user can be able to add data new node in the tree view. Save button is clicked to save data in the data source. The user can be able to update or delete data from the data source.

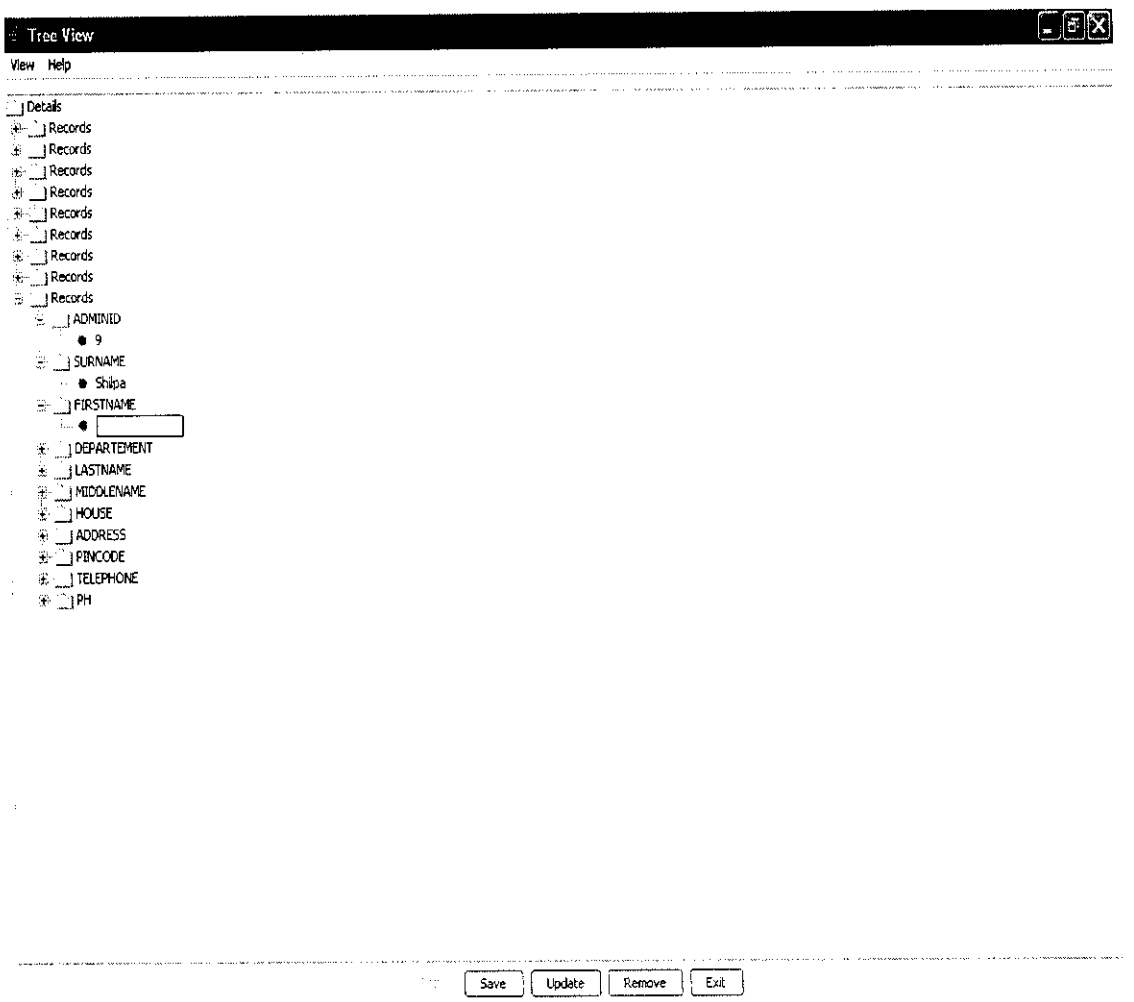


Figure: No: 8.1.12-Data Representation in Tree View

Screen Description:

This user interface screen is to display the error message if the user entered some wrong value in the child nodes of the tree node. When the user clicks on the save button error message will be displayed on the screen to enter valid data in the child nodes.

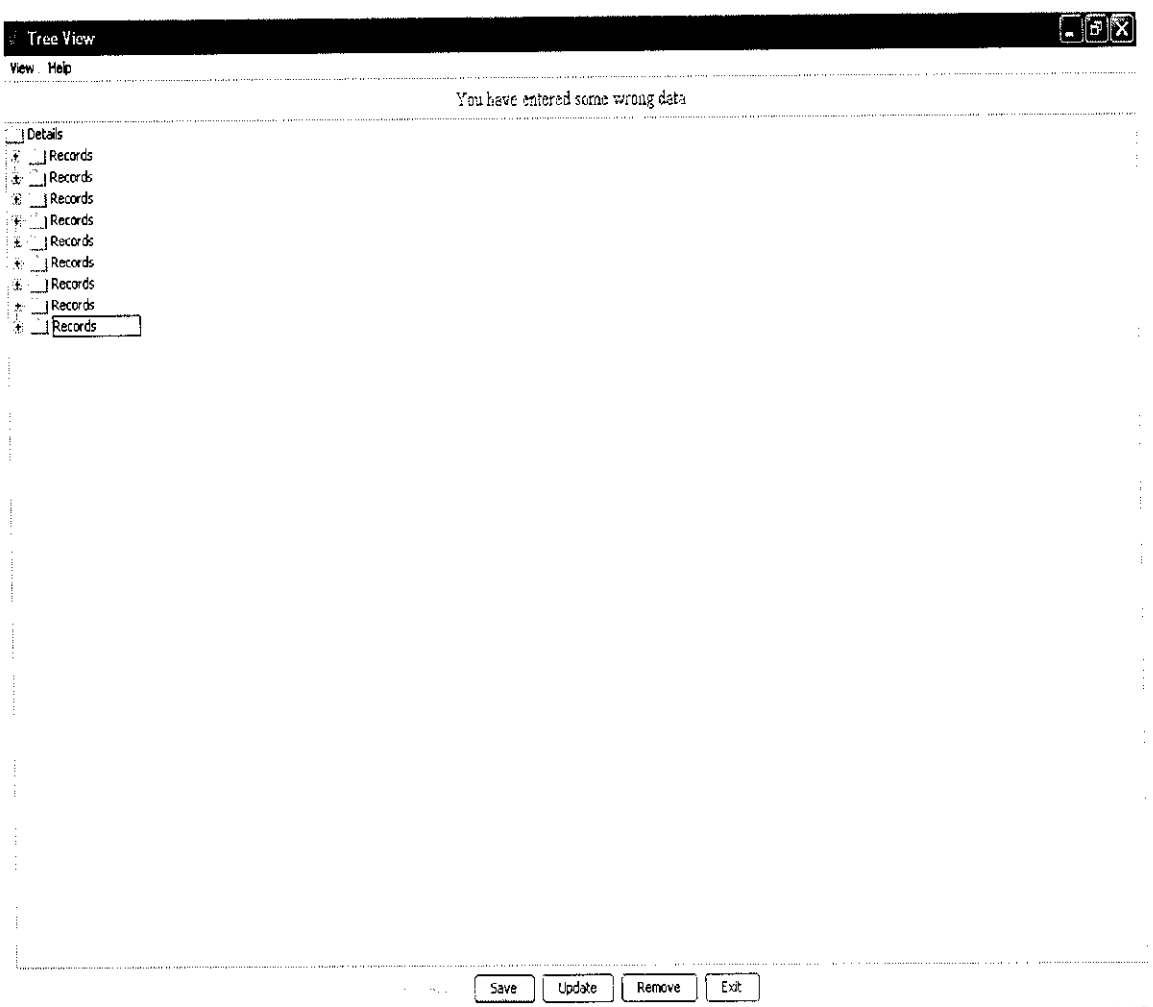


Figure: No: 8.1.13-To display error Message when wrong data is entered in Tree View

Screen Description:

This user interface screen is used to display error message if no node is selected for deleting a particular node. The users have to select the node for deleting or updating the particular node. Then the appropriate button is clicked to update data in the data source selected.

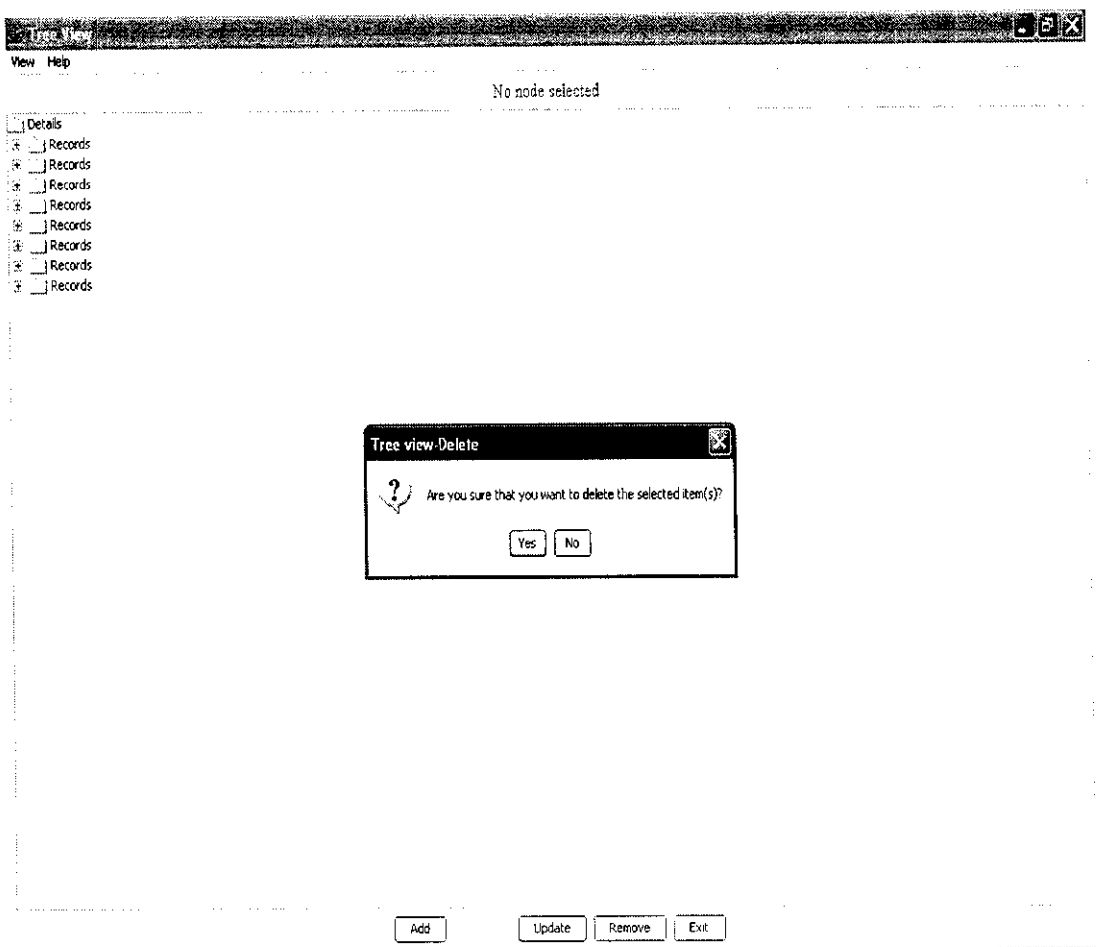


Figure: No: 8.1.14-To display the error message if no node is selected for removing

8.2 USER MANUAL

This component **"Reusable Component for Multiple Visual Representations of Data"** is a data source independent and a standalone desktop component which will display data in form, grid and tree views. These views will provide the CRUD (create, read, update, delete) functionalities to the end user on the data present in the data source. It would be used as a pluggable module for development projects .Data source independence allows for an easier integration and a better reusability of the RCMVRD component. This component is compatible with databases (such as Oracle, MySQL), CSV files and XML files. Data source details will be specified by the developer in an XML file.

This component will provide the GUI for each and every process that is to be done with this. This is most widely used in large projects which need to view large number of data from the datasource.

The GUI used in this component is:

- Choose Datasource GUI => to select the type of Datasource
- Set Database Table => to set database connection if the datasource selected is database
- Set CSV/XML File Path => to set the CSV/XML File Path if the datasource selected is CSV/XML File.

Once the datasource is selected then the user can view the data in all the three views Tree, Grid and Form views

- Tree View will provide the facilities such as Add, Delete or Update data.
- Form View will provide the facilities such as Add, Delete or Update data. In form it is possible to move between records.
- Grid View will provide the facilities such as Add, Delete or Update data

CHAPTER-IX

REFERENCES

[1] XML and DOM Parser pdf document, Available:*http://www.informatik.hsfurtwangen.de/~reich/XMLWebService.WS0708/index.html,* Accessed: 15 April, 2008.

[2] Eckstein et al (Feb-1999), Java swing, Study of Swing classes, Publisher: O'Reilly pp.121-155

[3] David Flanagan (Jan-1994), Java in a Nutshell, Publisher: O'Reilly pp.14-23

[4] Dr. Satyaraj Pantham, Pure JFC Swing, Publisher: Sun Microsystems

[5] Joshua. Bloch (2004)?, "Effective Java Programming", Publisher: Addison-Wesley Professional.

[6] Infosys Technologies Limited, K-shop. Available: http://sparsh/V1/ Accessed: 19 April, 2008.

[7] XML , Available: http://www.simonstl.com/articles/whyxml.htm Accessed:22 April, 2008

[8] C The Comma Separated Value (CSV) File Format, Available: http://www.creativyst.com/Doc/Articles/CSV/CSV01.htm, Accessed : 20 April, 2008