1.INTRODUCTION

1.1.PURPOSE

The purpose of this document is to display all the external requirements for the "Travel Bill tracking System". The main objective of preparing this document is to give a detailed description of the analysis and requirements for the system to be automated and this will be a guide in the other phase. The purpose of "Travel Bill Tracking System " to provide the Travelling bill process in an electronic form, using the Internet.

1.2 SCOPE

This document is meant for use by the developers and will be the basis for developing the final delivered system. The requirement changes to the document can be made due to changes in the client needs, change in technology etc.,

1.3 SYSTEM ARCHITECTURE

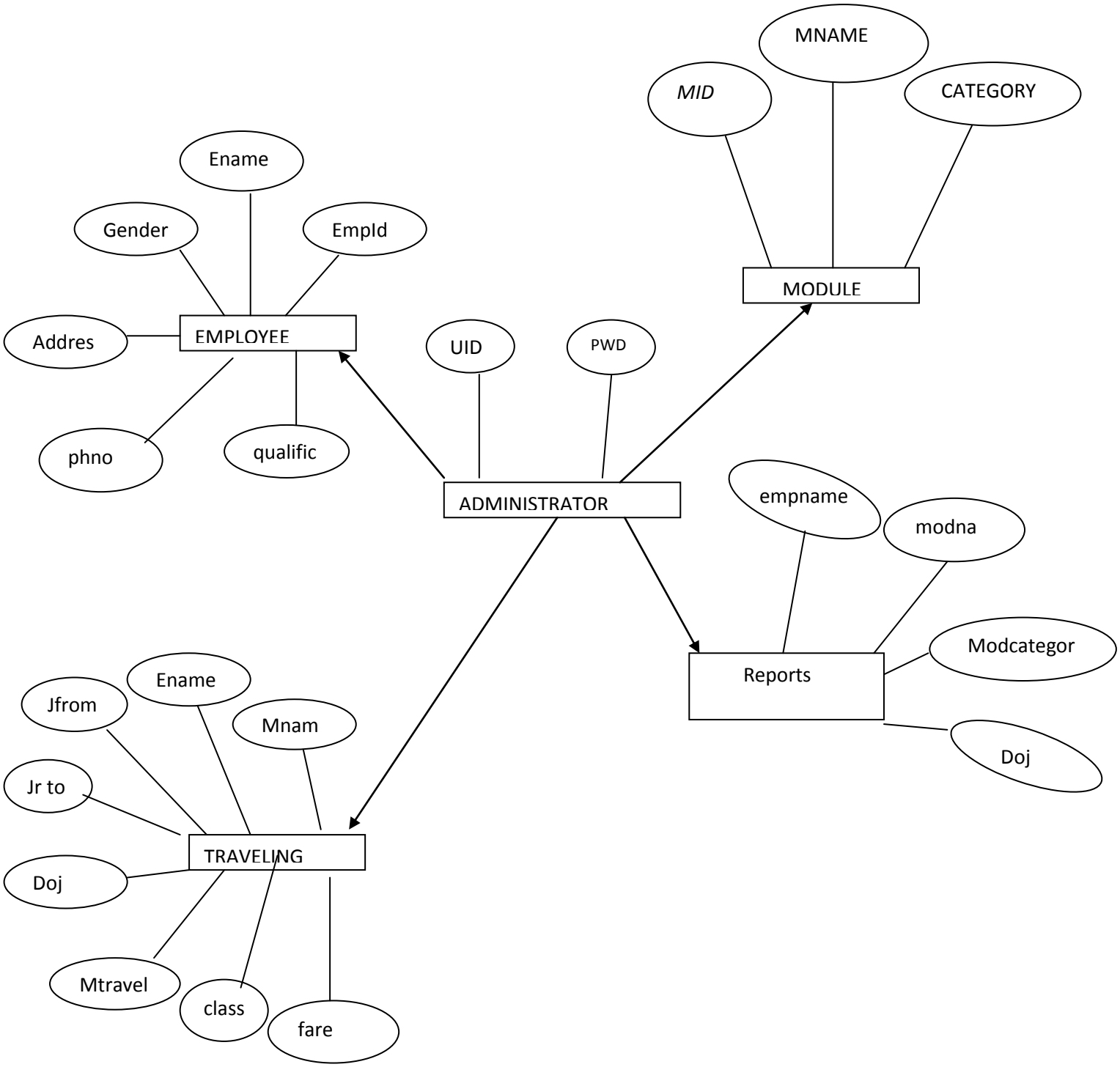


Fig 1.1: Model diagram of TBTS

1.4 OVERVIEW OF THE PROJECT

Travel Bill Tracking System is a web application that lets you maintain employee traveling bill information.

Functional specifications are in order to streamline activities and information flow, computerization on special tasks is most suited option. This shall minimize the paper movements and increase the efficiency, labor as well as time-consuming reports can be generated as when needed. Updating the progress of projects and status of different activities and jobs can be checked easily and regularly.

The process of "Travel Bill Tracking System" is to provide a employee traveling expenditure of different category working in an organization different type of reports on different selected options.

This Travel Bill Tracking System is design to provide communication channel inside an organization between different clients on the network. The main objective of the project is to develop and enhances communication among the members of organizations in a reliable cost effective and secure way. This document is one the describes the requirements of the System. In the System mainly new employee information employee travel information from place to place by the type of travel, cost of particular travelling, different type of report.

The main objective of the document is meant for use by the developers and will be the bases for developing final delivered system. The requirement changes to the document can be made due to changes in client needs, changes in technology etc., and this will be a guide in the other phase.

Thus the system helps the organization to improve the performance of its teams in conducting the organizational works, and design shall be able to provide the following facilities to the users.

The user can access this mailing system for sending the information to other users of the same organization. He is also provided with login and logout services and also provided with a chance to change the pass word of his account to option service design.and is to be designed with user flexible screens through which user can navigate

through the mailing system and access the above services for the basic purpose of communication..

User Interface Requirements : Web Based, User friendly interface

Database Requirement : Centralized

Integration Requirement : Web Enabled, Well integrated with internal system

Preferred Technologies : JAVA/J2EE/MS.NET

2. LITERATURE SURVEY

2.1 ABOUT JAVA

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

- **IMPORTANCE OF JAVA TO THE INTERNET**

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

- **JAVA CAN BE USED TO CREATE TWO TYPES OF PROGRAMS**

APPLICATIONS AND APPLETS**:** An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java’s ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java –compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

2.1.1 FEATURES OF JAVA

- SECURITY

Every time you that you download a "normal" program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scanned them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a "firewall" between a network application and your computer.

When you use a Java-compatible Web browser, you can safely download Java applets without fear of virus infection or malicious intent.

- PORTABILITY

For programs to be dynamically downloaded to all the various types of platforms connected to the Internet, some means of generating portable executable code is needed .As you will see, the same mechanism that helps ensure security also helps create portability. Indeed, Java's solution to these two problems is both elegant and efficient.

- THE BYTE CODE

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its

Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

- JAVA VIRTUAL MACHINE(JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

Overall Description

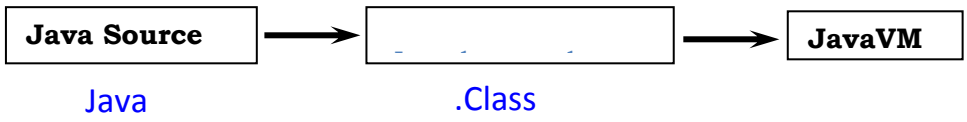


Fig 2.1 Picture showing the development process of JAVA Program

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The. Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code

2.1. 2 JAVA ARCHITECTURE

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a

dynamic system, able to load code when needed from a machine in the same room or across the planet.

2.1.3COMPILATION OF CODE

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

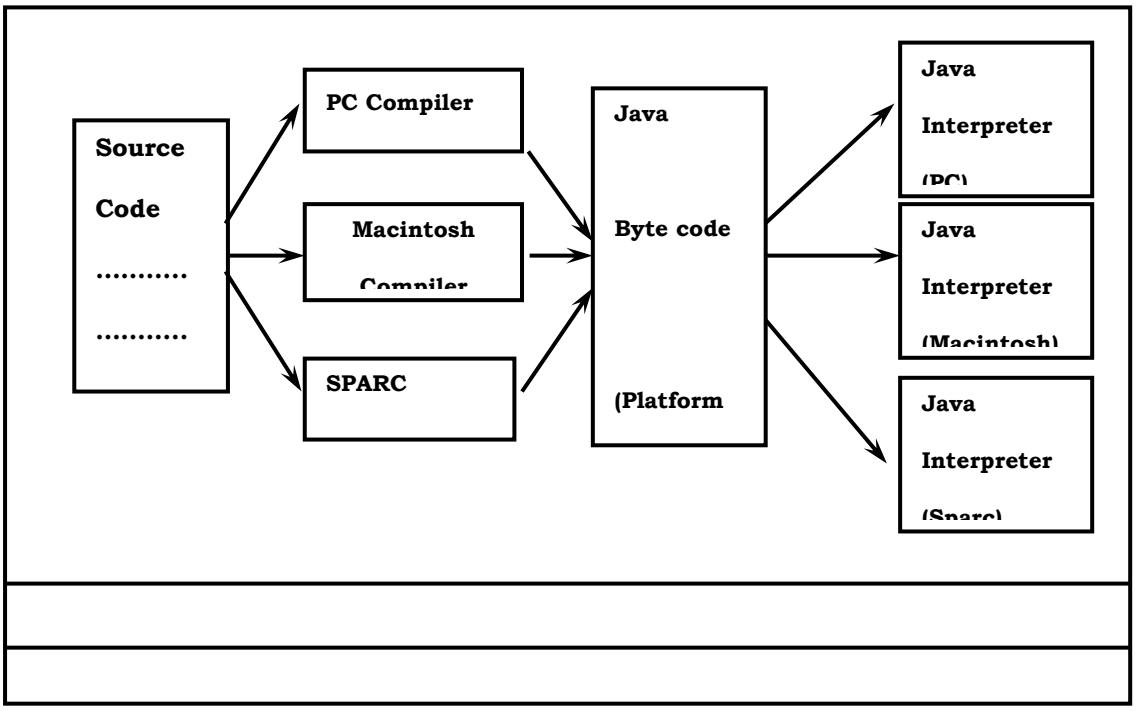


Fig 2.2 compilation of code

During run-time the Java interpreter tricks the bytecode file into thinking that it is running on a Java Virtual Machine. In reality this could be a Intel Pentium Windows 95 or SunSARC station running Solaris or Apple Macintosh running system and all could receive code from any computer through Internet and run the Applets.

- SIMPLE

Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ programmer, learning Java will be even easier. Because Java inherits the C/C++ syntax and many of the object oriented features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

- OBJECT ORIENTED

Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank slate. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

- ROBUST

The multi-platform environment of the Web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java. Java is strictly typed language; it checks your code at compile time and run time.

Java virtually eliminates the problems of memory management and deallocation, which is completely automatic. In a well-written Java program, all run time errors can –and should –be managed by your program.

2.2SERVLETS

- **INTRODUCTION**

The Java web server is JavaSoft's own web Server. The Java web server is just a part of a larger framework, intended to provide you not just with a web server, but also with tools. To build customized network servers for any Internet or Intranet client/server system. Servlets are to a web server, how applets are to the browser.

- **ABOUT SERVLETS**

Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side - object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, plugable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

For example, an HTTP Servlets can be used to generate dynamic HTML content. When you use Servlets to do dynamic content you get the following advantages:

- They're faster and cleaner than CGI scripts
- They use a standard API (the Servlets API)
- They provide all the advantages of Java (run on a variety of servers without needing to be rewritten).

- **ATTRACTIVENESS OF SERVLETS**

There are many features of Servlets that make them easy and attractive to use. These include:

- Easily configured using the GUI-based Admin tool
- Can be loaded and invoked from a local disk or remotely across the network.
- Can be linked together, or chained, so that one Servlets can call another Servlets, or several Servlets in sequence.
- Can be called dynamically from within HTML pages, using server-side include tags.
- Are secure - even when downloading across the network, the Servlets security model and Servlets sandbox protect your system from unfriendly behavior.

- **ADVANTAGES OF SERVLET API**

One of the great advantages of the Servlet API is protocol independence. It assumes nothing about:

- The protocol being used to transmit on the net
- How it is loaded
- The server environment it will be running in

These qualities are important, because it allows the Servlet API to be embedded in many different kinds of servers. There are other advantages to the Servlet API as well. These include:

- It’s extensible - you can inherit all your functionality from the base classes made available to you.
- it's simple, small, and easy to use.

♦ **FEATURES OF SERVLETS**

- Servlets are fast. Since Servlets only need to be loaded once, they offer much better performance over their CGI counterparts.
- Servlets are platform independent.
- Servlets are extensible. Java is a robust, object-oriented programming language, which easily can be extended to suit your needs
- Servlets are secure.
- Servlets can be used with a variety of clients.

♦ **LOADING SERVLETS**

Servlets can be loaded from three places

From a directory that is on the CLASSPATH. The CLASSPATH of the JavaWebServer includes service root/classes/ which is where the system classes reside.

From the <SERVICE_ROOT /Servlets/ directory. This is *not* in the server's classpath. A class loader is used to create Servlets from this directory. New Servlets can be added - existing Servlets can be recompiled and the server will notice these changes.

From a remote location. For this a code base like http: // nine.eng / classes / foo / is required in addition to the Servlets class name. Refer to the admin GUI docs on Servlet section to see how to set this up.

- LOADING REMOTE SERVLETS

Remote Servlets can be loaded by:

1. Configuring the Admin Tool to setup automatic loading of remote Servlets
2. Setting up server side include tags in .shtml files
3. Defining a filter chain configuration

- INVOKING SERVLETS

A Servlet invoker is a Servlet that invokes the "service" method on a named Servlet. If the Servlet is not loaded in the server, then the invoker first loads the Servlet (either from local disk or from the network) and the then invokes the "service" method. Also like applets, local Servlets in the server can be identified by just the class name. In other words, if a Servlet name is not absolute, it is treated as local.

A client can invoke Servlets in the following ways:

- The client can ask for a document that is served by the Servlet.
- The client (browser) can invoke the Servlet directly using a URL, once it has been mapped using the Servlet Aliases section of the admin GUI.
- The Servlet can be invoked through server side include tags.

- The Servlet can be invoked by placing it in the Servlets/ directory.
- The Servlet can be invoked by using it in a filter chain.

2.3 JAVASCRIPT

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then updates the browser's display accordingly

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is

almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags

<SCRIPTS>..</SCRIPT>.

<SCRIPT LANGUAGE = “JavaScript”>

- **JAVASCRIPT STATEMENTS**

</SCRIPT>

Here are a few things we can do with JavaScript :

- Validate the contents of a form and make calculations.
- Add scrolling or changing messages to the Browser’s status line.
- Animate images or rotate images that change when we move the mouse over them.
- Detect the browser in use and display different content for different browsers.
- Detect installed plug-ins and notify the user if a plug-in is required.

We can do much more with JavaScript, including creating entire application.

- **JAVASCRIPT VERSUS JAVA**

JavaScript and Java are entirely different languages. A few of the most glaring differences are:

- Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
- While JavaScript is best suited o simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

- **ADVANTAGES**
  - JavaScript can be used for Sever-side and Client-side scripting.
  - It is more flexible than VBScript.

- JavaScript is the default scripting languages at Client-side since all the browsers supports it.

- **HYPERTEXT MARKUP LANGUAGE**

Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

- **BASIC HTML TAGS**

<!-- -->	Specifies comments
<A>…….</A>	Creates hypertext links
<B>…….</B>	Formats text as bold
<BIG>…….</BIG>	Formats text in large font.
<BODY>…</BODY>	Contains all tags and text in the HTML document
<CENTER>...</CENTER>	Creates text

<DD>…</DD>	Definition of a term
<DL>...</DL>	Creates definition list
<FONT>…</FONT>	Formats text with a particular font
<FORM>...</FORM>	Encloses a fill-out form
<FRAME>...</FRAME>	Defines a particular frame in a set of frames
<H#>…</H#>	Creates headings of different levels
<HEAD>...</HEAD>	Contains tags that specify information about a document
<HR>...</HR>	Creates a horizontal rule
<HTML>…</HTML>	Contains all other HTML tags
<META>...</META>	Provides meta-information about a document
<SCRIPT>…</SCRIPT>	Contains client-side or server-side script
<TABLE>…</TABLE>	Creates a table
<TD>…</TD>	Indicates table data in a table
<TR>…</TR>	Designates a table row
<TH>…</TH>	Creates a heading in a table

- **ADVANTAGES**

✓ A HTML document is small and hence easy to send over the net. It is small because it does not include formatted information.

✓ HTML is platform independent.

✓ HTML tags are not case-sensitive.

2.4JDBC

JDBC is a Java API for executing SQL statements. (As a point of interest, JDBC is a trademarked name and is not an acronym; nevertheless, JDBC is often thought of as standing for Java Database Connectivity. It consists of a set of classes and interfaces written in the

Java programming language. JDBC provides a standard API for tool/database developers and makes it possible to write database applications using a pure Java API.

Using JDBC, it is easy to send SQL statements to virtually any relational database. One can write a single program using the JDBC API, and the program will be able to send SQL statements to the appropriate database. The combinations of Java and JDBC lets a programmer write it once and run it anywhere.

- **FUNCTIONS OF JDBC**

Simply put, JDBC makes it possible to do three things:

- Establish a connection with a database
- Send SQL statements
- Process the results.

- **JDBC VERSUS ODBC**

At this point, Microsoft's ODBC (Open Database Connectivity) API is that probably the most widely used programming interface for accessing relational databases. It offers the ability to connect to almost all databases on almost all platforms.

So why not just use ODBC from Java? The answer is that you can use ODBC from Java, but this is best done with the help of JDBC in the form of the JDBC-ODBC Bridge, which we will cover shortly. The question now becomes "Why do you need JDBC?" There are several answers to this question:

1. ODBC is not appropriate for direct use from Java because it uses a C interface. Calls from Java to native C code have a number of drawbacks in the security, implementation, robustness, and automatic portability of applications.
2. A literal translation of the ODBC C API into a Java API would not be desirable. For example, Java has no pointers, and ODBC makes copious use of them, including the notoriously error-prone generic pointer "void *". You can think of JDBC as ODBC translated into an object-oriented interface that is natural for Java programmers.
3. ODBC is hard to learn. It mixes simple and advanced features together, and it has complex options even for simple queries. JDBC, on the other hand, was designed to keep simple things simple while allowing more advanced capabilities where required.

4. A Java API like JDBC is needed in order to enable a "pure Java" solution. When ODBC is used, the ODBC driver manager and drivers must be manually installed on every client machine. When the JDBC driver is written completely in Java, however, JDBC code is automatically installable, portable, and secure on all Java platforms from network computers to mainframes.

- TWO-TIER AND THREE-TIER MODELS

The JDBC API supports both two-tier and three-tier models for database access.
In the two-tier model, a Java applet or application talks directly to the database. This requires a JDBC driver that can communicate with the particular database management system being accessed. A user's SQL statements are delivered to the database, and the results of those statements are sent back to the user. The database may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the database as the server. The network can be an Intranet, which, for example, connects employees within a corporation, or it can be the Internet.

In the three-tier model, commands are sent to a "middle tier" of services, which then send

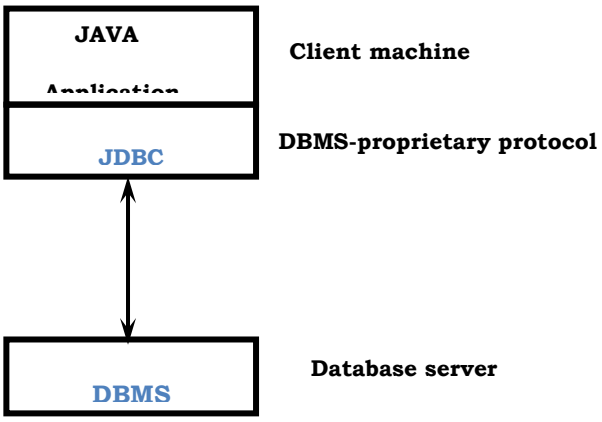


Fig 2.3 Three tier model

SQL statements to the database. The database processes the SQL statements and sends the results back to the middle tier, which then sends them to the user. MIS directors find the three-tier model very attractive because the middle tier makes it possible to maintain control over access and the kinds of updates that can be made to corporate data. Another advantage is

that when there is a middle tier, the user can employ an easy-to-use higher-level API which is translated by the middle tier into the appropriate low-level calls. Finally, in many cases the three-tier architecture can provide performance advantages

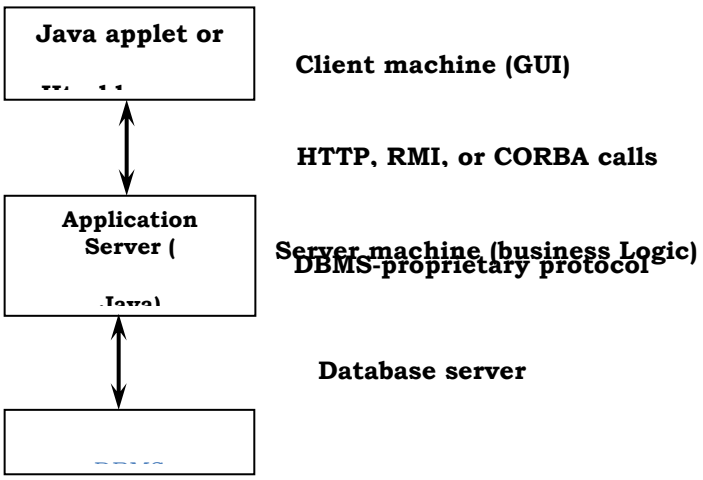


Fig 2.4 Middle tier architecture

Until now the middle tier has typically been written in languages such as C or C++, which offer fast performance. However, with the introduction of optimizing compilers that translate Java byte code into efficient machine-specific code, it is becoming practical to implement the middle tier in Java. This is a big plus, making it possible to take advantage of Java's robustness, multithreading, and security features. JDBC is important to allow database access from a Java middle tier.

2.5 JAVA SERVER PAGES(JSP)

Java server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards, and a mature re-usable component model .The Java Server Pages architecture enables the separation of content generation from content presentation. This separation not eases maintenance headaches, it also allows web team members to focus on their areas of expertise. Now, web page designer can concentrate on layout, and web application designers on programming, with minimal concern about impacting each other’s work.

FEATURES OF JSP

- PORTABILITY

Java Server Pages files can be run on any web server or web-enabled application server that provides support for them. Dubbed the JSP engine, this support involves recognition, translation, and management of the Java Server Page lifecycle and its interaction components.

- COMPONENTS

It was mentioned earlier that the Java Server Pages architecture can include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans, and Servlets.

- PROCESSING

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. The Java Server Pages file has a JSP extension to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a Servlet on the server side. The Servlet that is generated outputs real content in straight HTML for responding to the client.

- ACCESS MODELS

A Java Server Pages file may be accessed in at least two different ways. A client's request comes directly into a Java Server Page. In this scenario, suppose the page accesses reusable Java Bean components that perform particular well-defined computations like accessing a database. The result of the Beans computations, called result sets is stored within the Bean as properties. The page uses such Beans to generate dynamic content and present it back to the client.

In both of the above cases, the page could also contain any valid Java code. Java Server Pages architecture encourages separation of content from presentation.

STEPS IN THE EXECUTION OF A JSP APPLICATION

1. The client sends a request to the web server for a JSP file by giving the name of the JSP file within the form tag of a HTML page.

2. This request is transferred to the Java Web Server. At the server side Java Web Server receives the request and if it is a request for a jsp file server gives this request to the JSP engine.
3. JSP engine is program which can understands the tags of the jsp and then it converts those tags into a Servlet program and it is stored at the server side. This Servlet is loaded in the memory and then it is executed and the result is given back to the Java Web Server and then it is transferred back to the result is given back to the Java Web Server and then it is transferred back to the client.

CONNECTIVITY

The JDBC provides database-independent connectivity between the J2EE platform and a wide of tabular data sources. JDBC technology allows an Application Component Provider range to:

- Perform connection and authentication to a database server
- Manager transactions
- Move SQL statements to a database engine for preprocessing and execution
- Execute stored procedures

3. SYSTEM ANALYSIS

3.1 Definition and reason for Condition Analysis

System analysis will be performed to determine if it is feasible to design an information based on policies and plans of the organization and on user requirements and to eliminate the weaknesses of the present system.

General requirements

1. The new system should be cost effective.

1. To augment management, improve productivity and services.
2. To enhance User/System interface.
3. To improve information qualify and usability.
4. To upgrade system's reliability, availability, flexibility and growth potential.

Identification of Need

3.2 EXISTING SYSTEM

3.2.1 DISADVANTAGES:

In the existing system billing of the employer traveling is done manually.

Time Consuming: - As this system needs lots of manpower and as papers will be moving from one place to another manually, lots of time is consumed. So time constraints maintenance is very difficult.

Less Security: - As these records move from place to department the security provided to the data is very less. As this is manually done the data cannot be very accurate. Even there can be chance that intermediate person can leak the proposals.

Billing cannot be done round the clock: - As this is entirely manual work the billing should be available round the clock and from any place to any place.but this is not possible with the manual system.

Updating of database is done only after Billing: - As this is a manual process only after the uploading is completed the database is updated with all the details.

Progress of Traveling bill not known instantaneously: - In this process as the database is not updated time 10 time, the progress of the travel billing is hard to know and 10 get details instantaneously can be forgotten.

3.3 PROPOSED SYSTEM

3.3.1 ADVANTAGES

In this we discuss about the proposed Travel bill tracking System.

General Description Of Inputs & Outputs:

The system will be getting the employee information, employee traveling information, category information different type of reports.

Fully Automated System: - Our proposed system is an automated format of the above Slated paper based work. Here the entire process is computerized thus by reducing the man power and the entire manual work.

Fast and Accurate: - As this is automated system and fully computerized so it is far more fast and accurate than manual work.

Data security: - In our system the data is updated in database time 10 time and data is nor sent via any paper work so data is secured as only the administrator has the access to the database and no one else can modify the database,

Availability of system round the clock: - As this is an automated system, system is available all the time, so no need for the official availability so no delay in work. The process continues automatically does not need to wait for anyone to keep the proposal.

Continuous updating of database: -

Database is updated from time to time after each effect on the proposal. So data is more accurate and perfect as all the updations are done simultaneously as the process.

Instantaneous retrieval of data: -

As the database is update from time to time the data can he retrieved at any time.

Generation of reports for decision-making: -

In our project we generate reports so that the higher officials can know the progress of the proposal placed and thus by also take the important decisions regarding the proposal handling.

How does it work?

This document is one that describes the requirements of the system. In this system mainly new employee information, employee traveling information from place to place by the type of travel, cost of particular traveling ,different type of reports.

\

4. REQUIREMENT SPECIFICIATIONS

4.1 SOFTWARE REQUIREMENT SPECIFICIATIONS

OPERATING SYSTEM	: WINDOWS XP with SP2.
LANGUAGE (FRONT END)	:JAVA (JDK:1.5/1.6)
SERVER	: APACHE TOMACAT 5.5/6.0.
WEB TECHNOLOGY	:HTML,JAVASCRIPT, CSS.
DATABASE (BACK END)	: ORACLE 10G
ARCHITECTURE	: 3 TIER ARCHITECTURE

4.2 HARDWARE REQUIREMENTS SPECIFICATIONS

PROCESSOR	: INTEL 2.0 GHz OR ABOVE
HARDDISK	: 80 GB
RAM	: 512 MB

5.SYSTEM DESIGN

5.1 SYSTEM SPECIFICATIONS

Sytem specification documents most predominantly contain information on basic website requirements which include:

- Performance levels
- Reliability
- Quality
- Interfaces
- Security and Privacy
- Constraints and Limitations
- Functional Capabilities
- Data Structures and elements

System specifications are:

1. Pentium 4 with minimum 1.x GHz processor or equivalent processor
2. Minimum 128 MB RAM (1 GB RAM recommended)
3. Hard disk with minimum 1 GB free space
4. NIC (network interface card) connected to network pentium III
5. Ram : 64 MB.
6. Hard disk : 10.2 GB.
7. Monitor : SVGA color monitor.
8. Keyboard : 105 standard mouse.

5.2 SYSTEM COMPONENTS

5.2.1 MODULES DESCRIPTION

There are 4 types of Modules. They are:

1. Employee Module.
2. Modules Information.
3. Traveling details.
4. Reports Module.

1. EMPLOYEE INFORMATION:

- Creating new employee.
- We can View the complete details of an employee.
- Regarding Emp ID ,name, Gender, phone no, Address, D.O.B and Qualification
- Here we can Update the details of an Employee.
- The employee information will be added by entering the details of an employee in the employee registration.

2.**MODULES INFORMATION:**

- Creating new module.
- By entering the Module ID, Module name and Category we can add the new module.
- And if the data is true means it will display that Data is inserted successfully.

3. TRAVELLING INFORMATION:

- Entering employee traveling details.
- In this we should enter the details regarding emp name, module name, journey from, journey to, Date of journey, Mode of travel, Class, Fare etc.
- Here we can Search the Travelling Information.
- And also we can delete the details.

5. **REPORTS MODULE:**

- Generating reports basing upon the choice.

- And we can view the Details of the Reports.

5.3 UML DIAGRAMS

\A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.

5.1 use case fig

An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

They are helpful in exposing requirements and planning the project..

Use Case Diagram for Administrator:

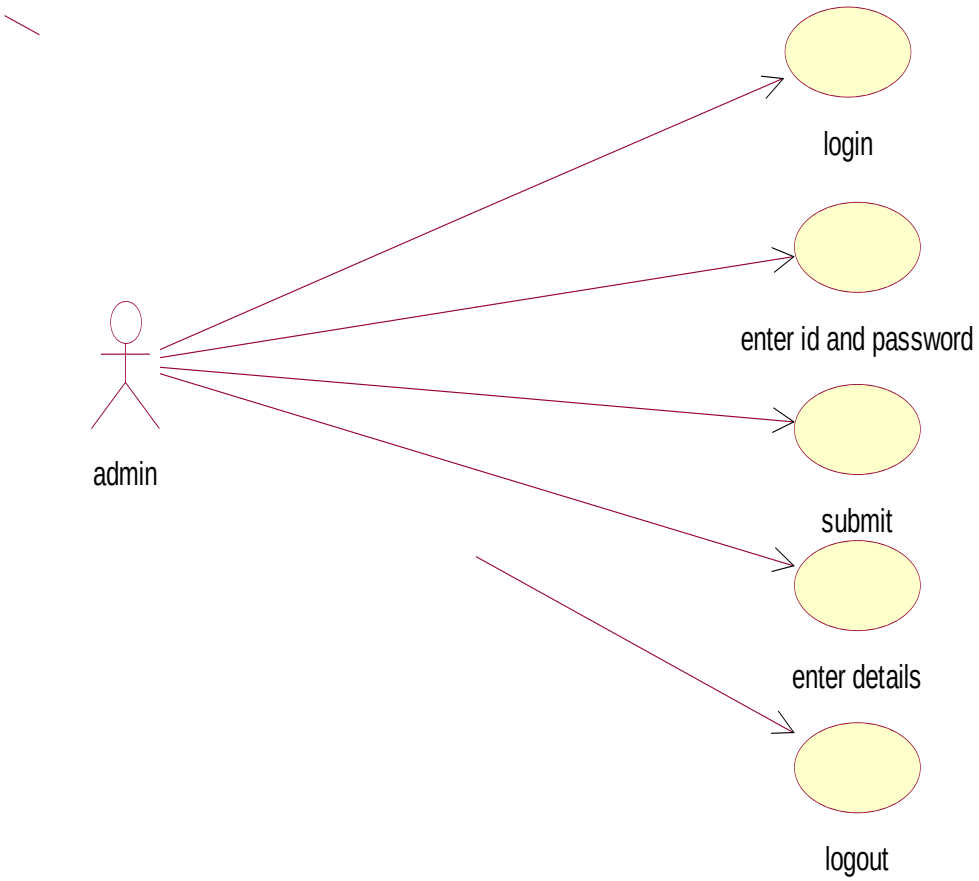


Fig 5.2 usecase diagram for admin module

SEQUENCE DIAGRAM FOR ADMINSTRATOR

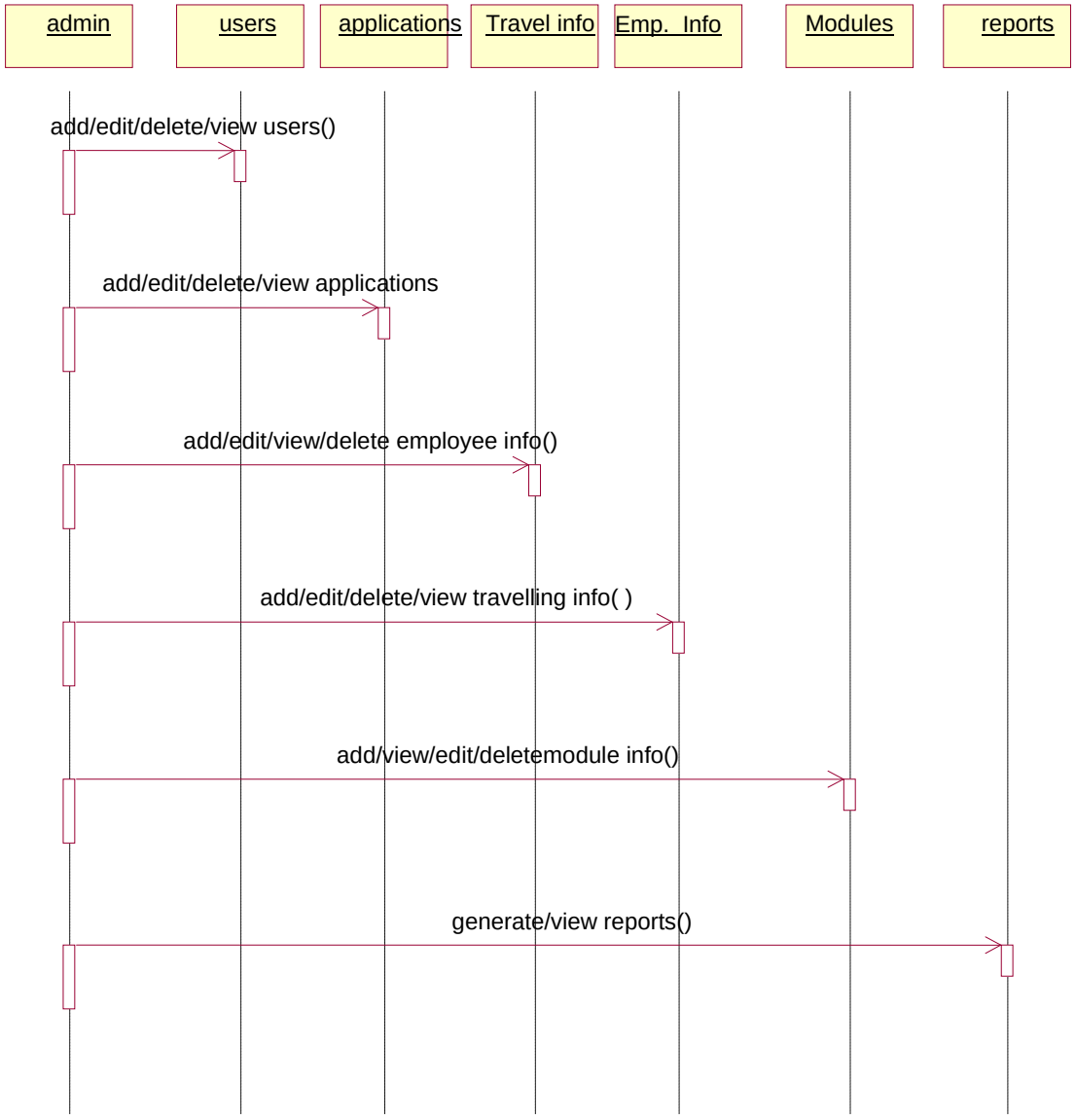


Fig 5.3 sequence diagram for admin module

USE DIAGRAM FOR EMPLOYEE INFO MODULE

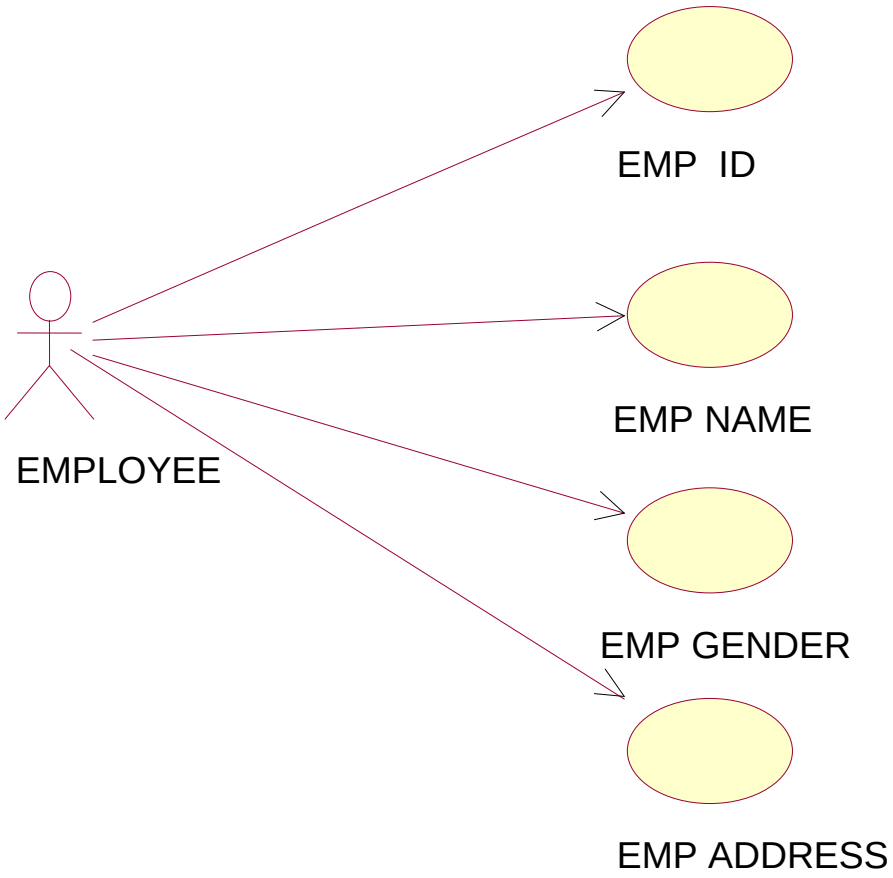
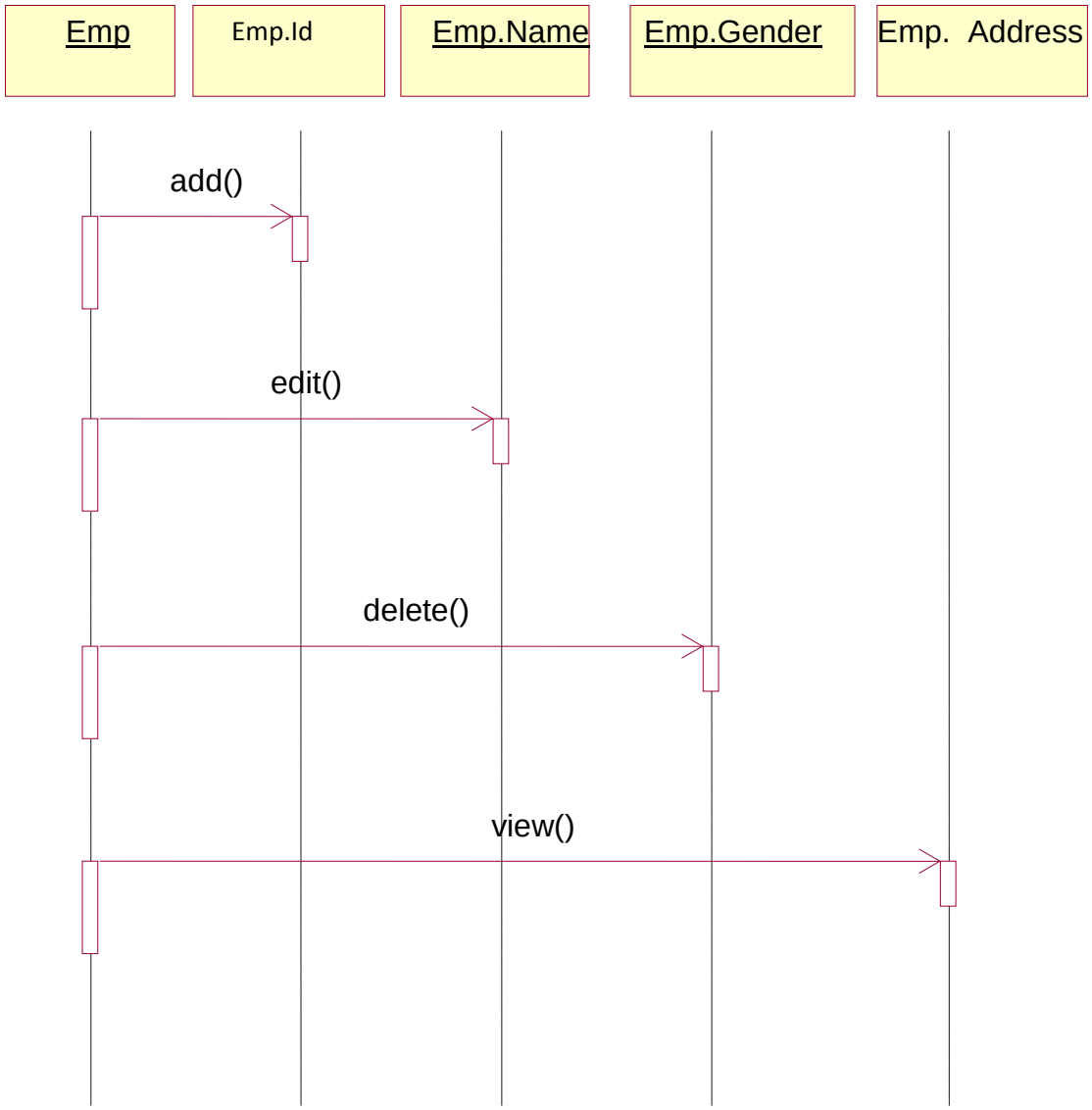


Fig 5.4 Usecase diagram for employee module

SEQUENCE DIAGRAM FOR EMPLOYEE INFORMATION MODULE



5.5 sequence diagram for Employee Information Module

USECASE DIAGRAM FOR MODULE INFO

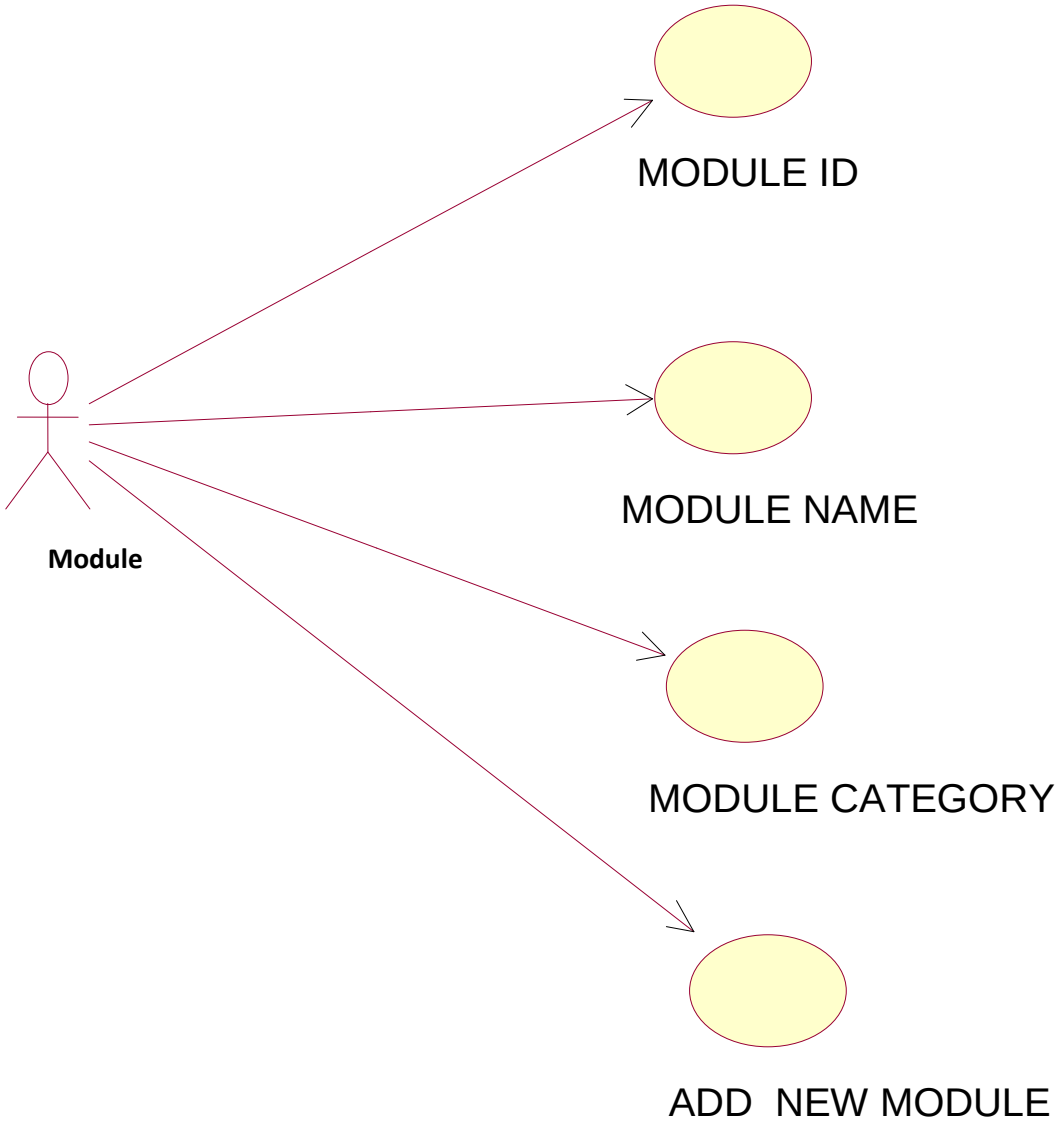
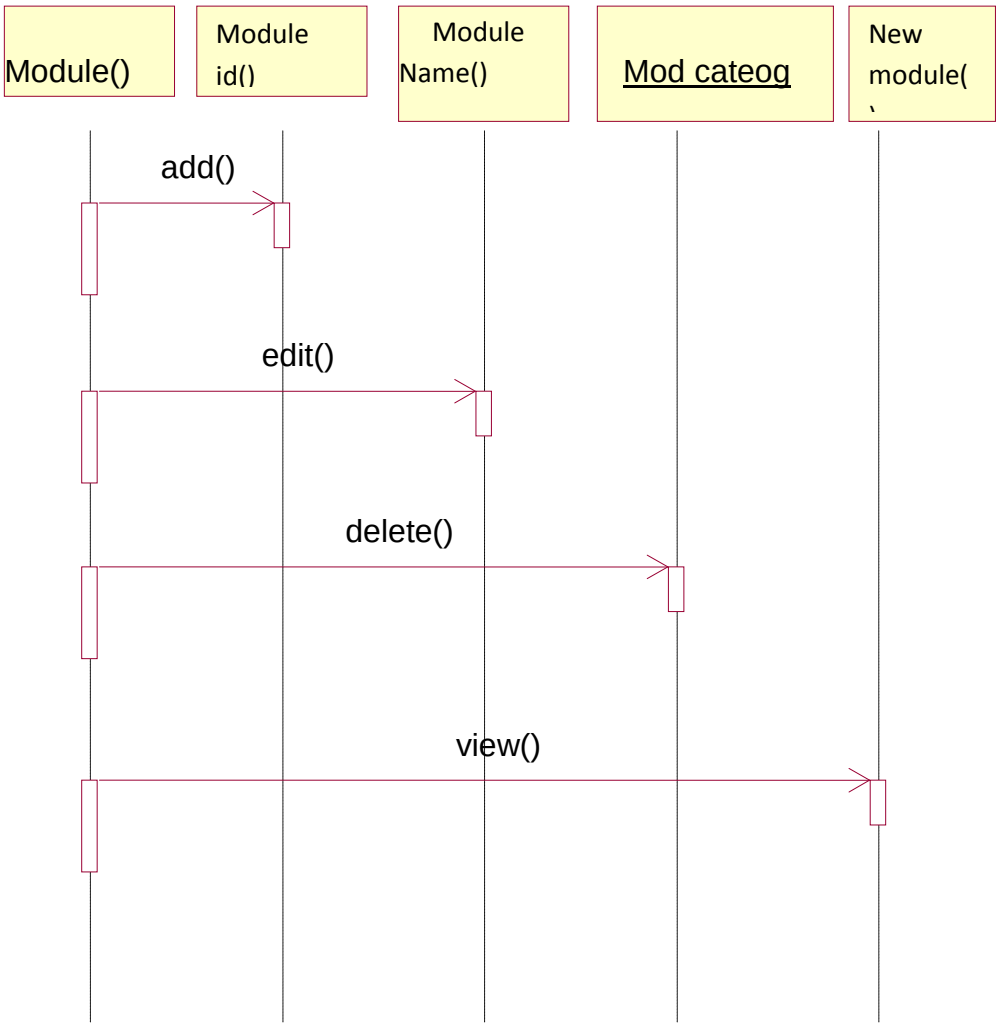


Fig 5.6 usecase diagram for module info

SEQUENCE DIAGRAM FOR MODULE INFORMATION



(

Fig 5.7 sequence diagram for module information

USECASE DIAGRAM FOR REPORTING MODULE

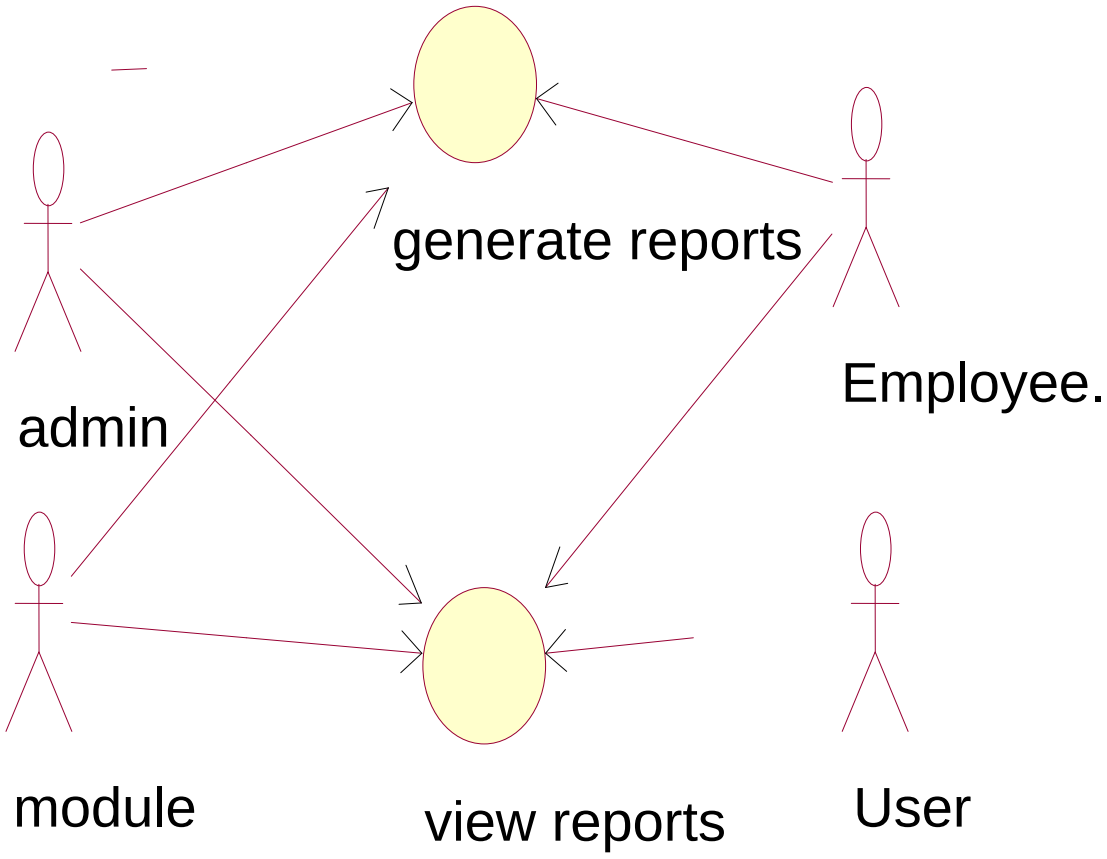


Fig 5.8 usecase diagram for reporting module

SEQUENCE DIAGRAM FOR REPORTING MODULE

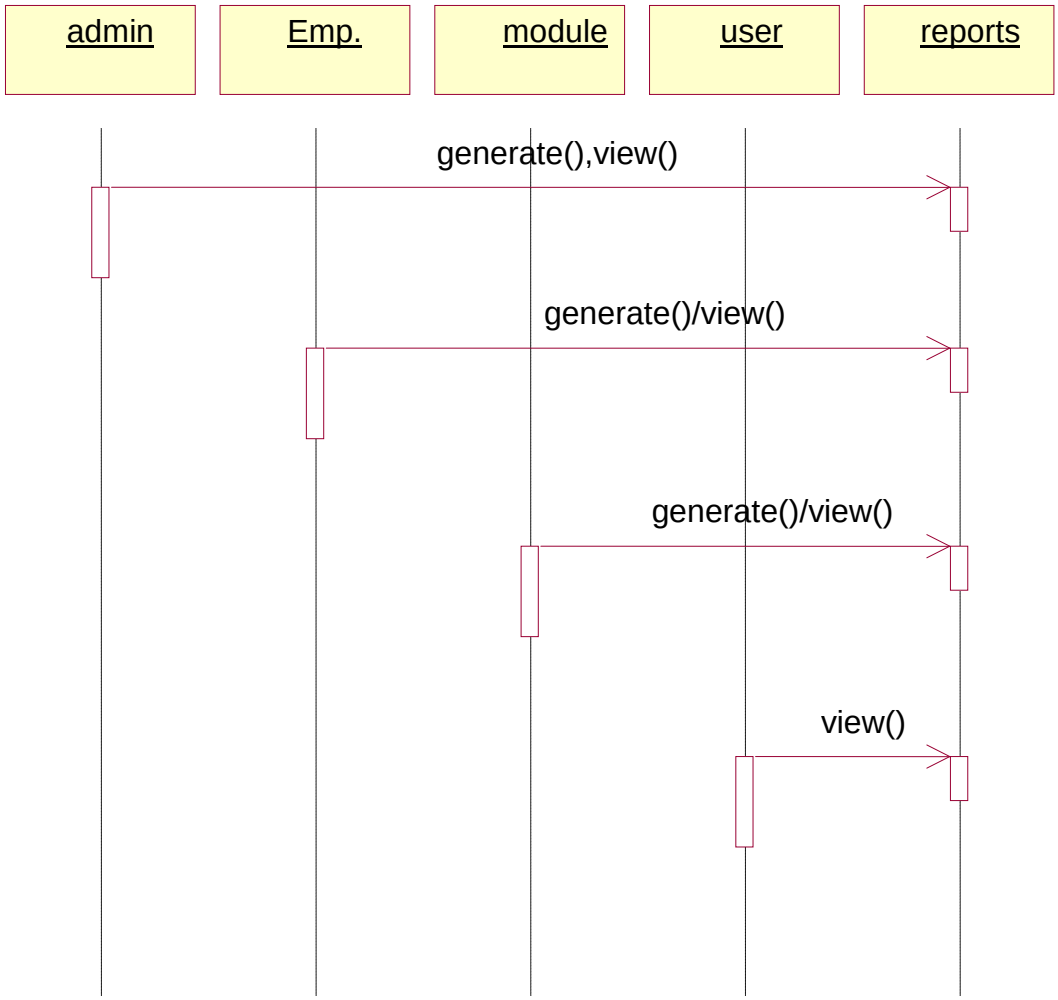


Fig 5.9 sequence diagram for reporting module

USECASE DIAGRAM FOR TRAVELLING INFORMATION MODULE

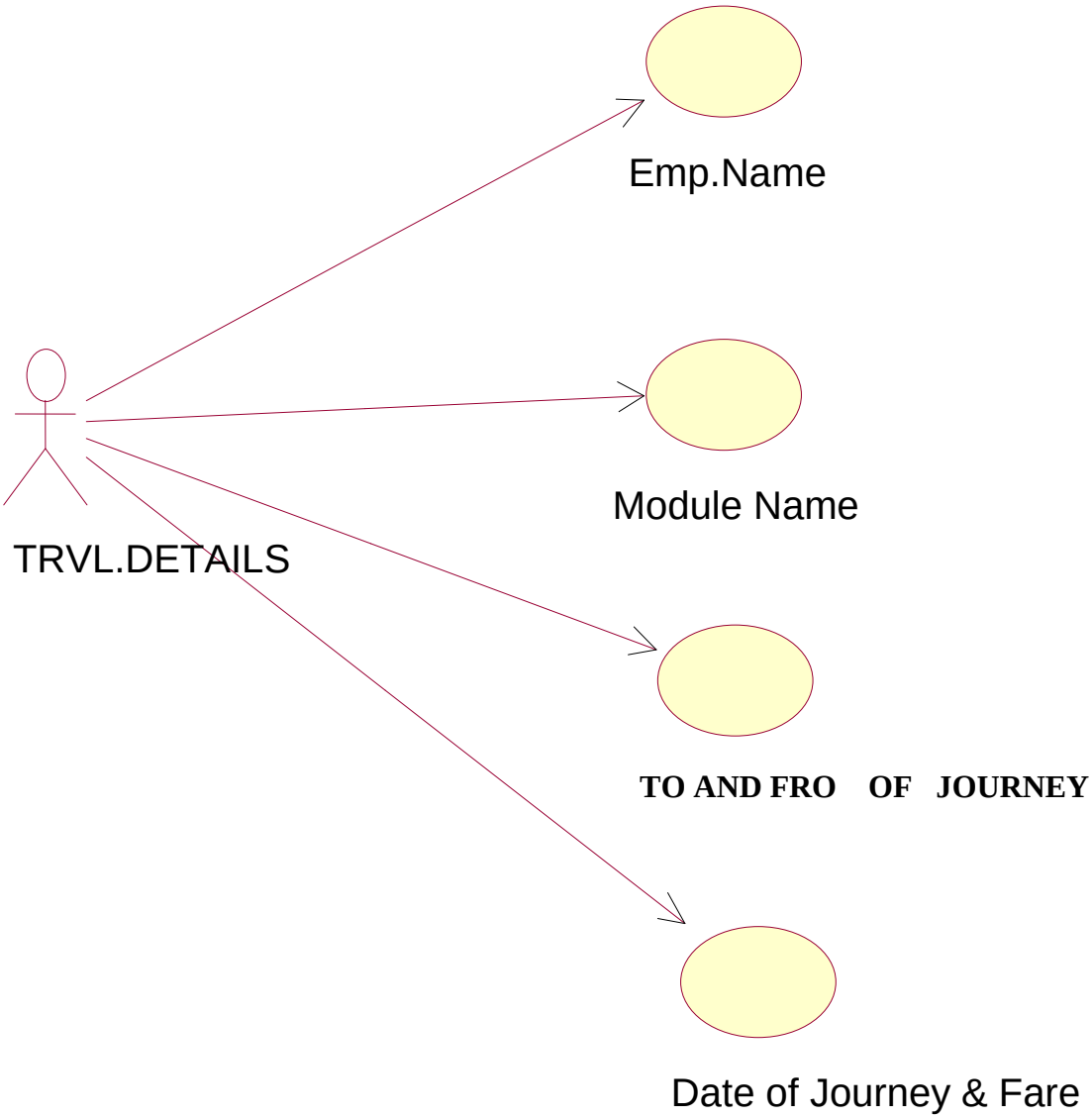


Fig 5.10 usecase diagram for travelling information module

SEQUENCE DIAGRAM FOR TRAVELLING INFORMATION MODULE

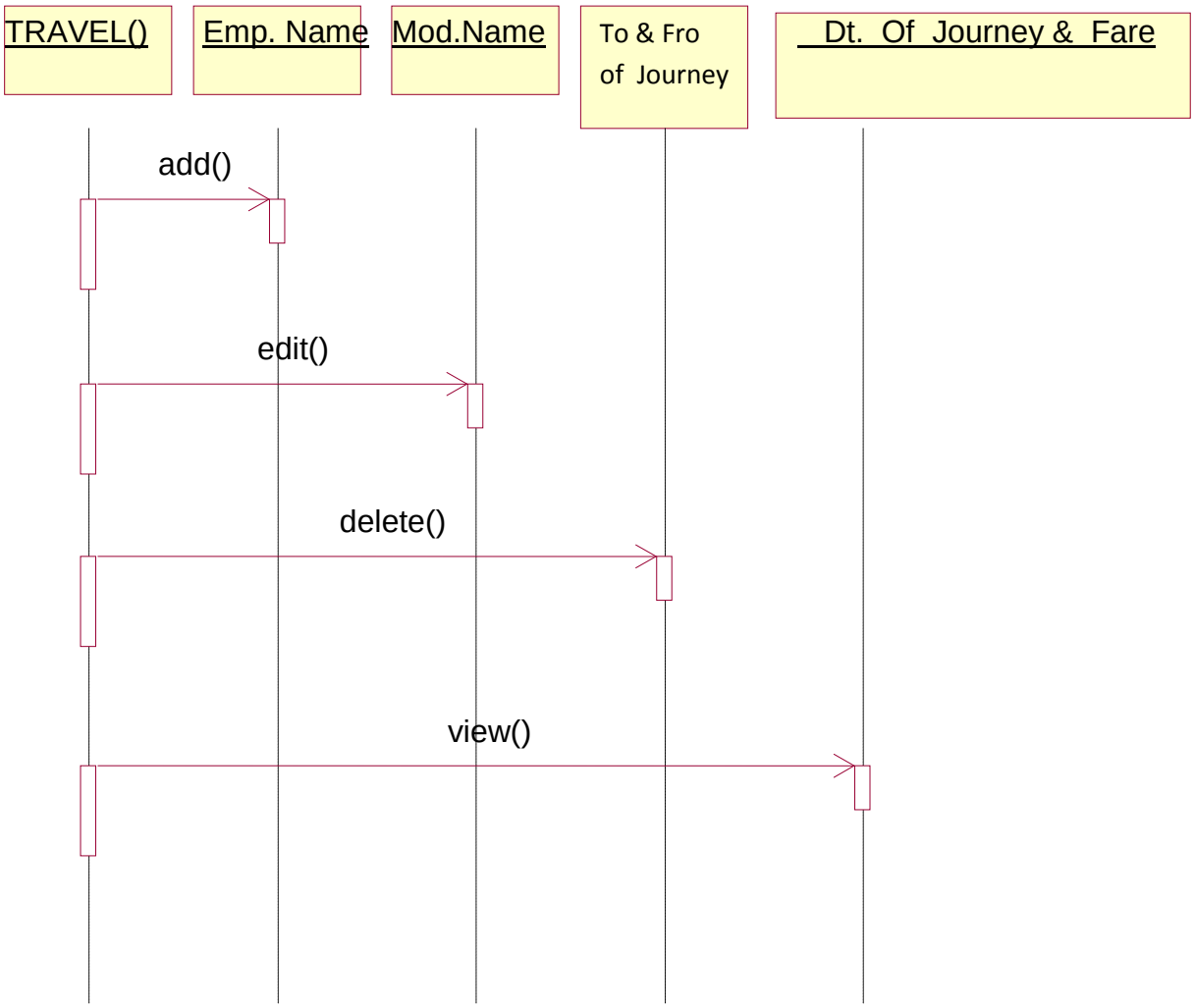


Fig 5.11 Sequence diagram for travelling information module

USECASE DIAGRAM FOR TBTS MODULE

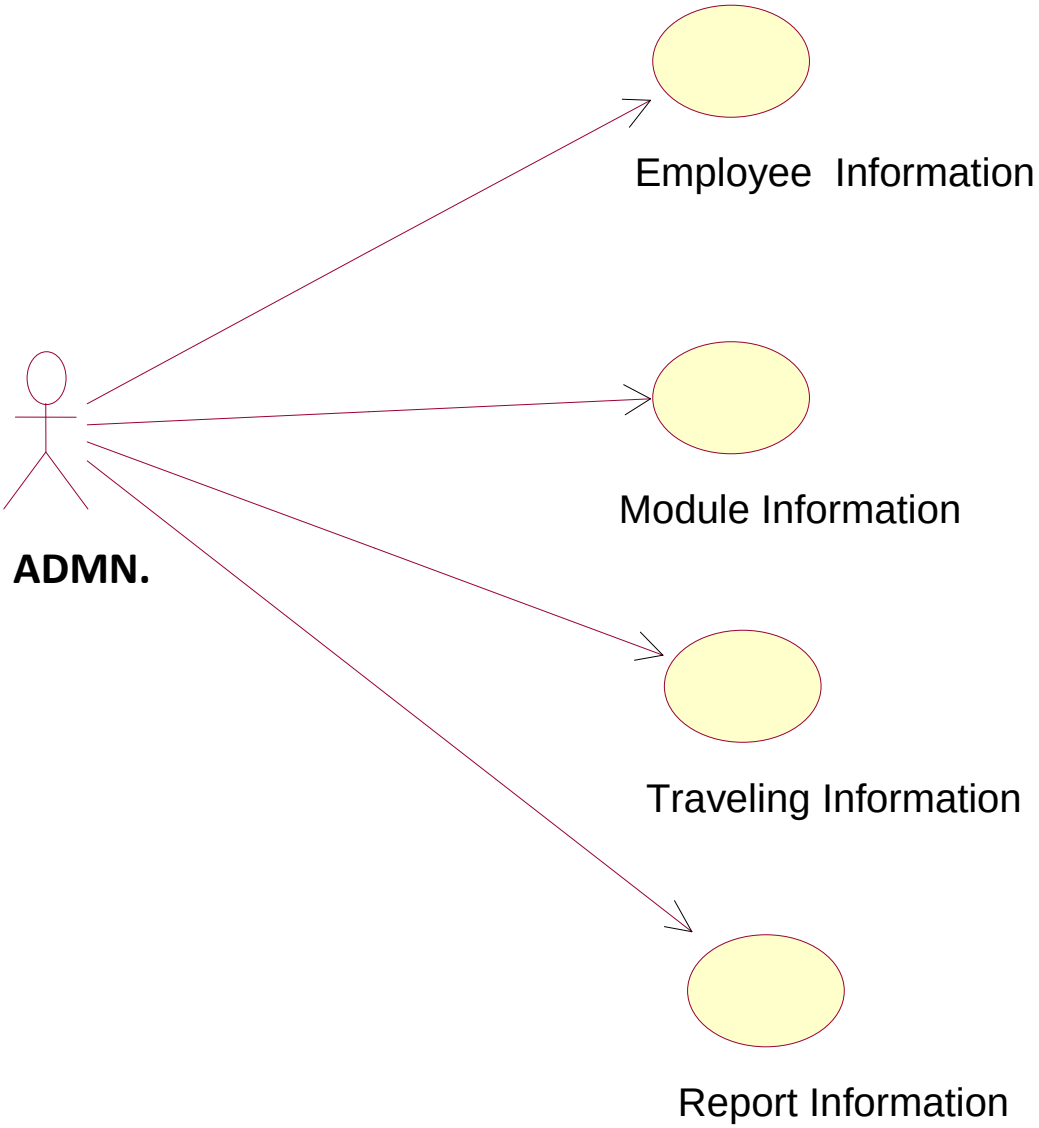


Fig 5.12 Use case diagram for TBTS module

SEQUENCE DIAGRAM FOR TBTS

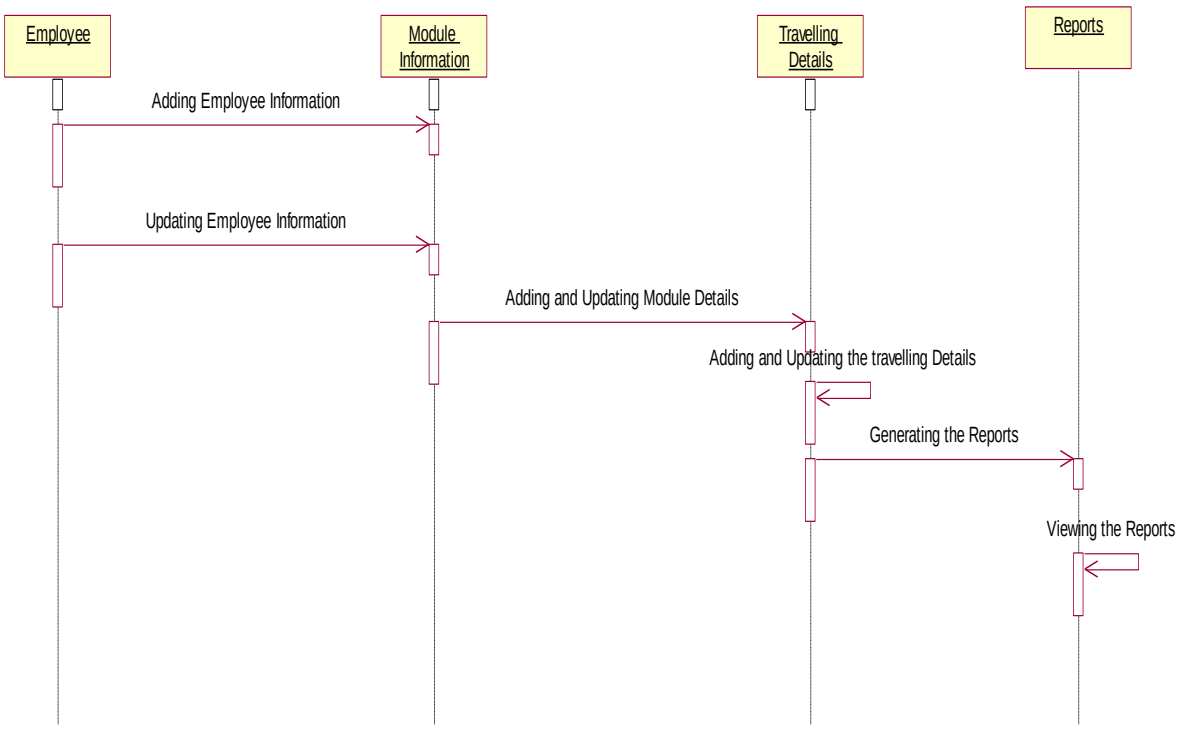


Fig 5.13 Sequence diagram for TBTS module

5.4 DATA DICTIONARY

The logical characteristics of current systems data stores, including name, description, aliases, contents, and organization, identifies processes where the data are used and where immediate access to information required, Serves as the basis for identifying database requirements during system design.

Uses of Data Dictionary:

1. To manage the details in large systems.
2. To communicate a common meaning for all system elements.
3. To Document the features of the system.
4. To facilitate analysis of the details in order to evaluate characteristics and determine where system changes should be made.

ADMINLOGIN

Name	Null?	Type
USERNAME	NOT NULL	VARCHAR2(30)
PWD	NOT NULL	VARCHAR2(30)

EMPLOYEE_INFO

Name	Null?	Type
EMP_ID	NOT NULL	VARCHAR2(20)
NAME	NOT NULL	VARCHAR2(50)
GEN	NOT NULL	VARCHAR2(7)
PHNO		NUMBER(12)
ADDRESS	NOT NULL	VARCHAR2(100)
QLFY	NOT NULL	VARCHAR2(20)
DOJ	NOT NULL	DATE

\

MODULES_INFO

Name	Null?	Type
MID	NOT NULL	NUMBER(3)

MODULENAME	VARCHAR2(100)
MODULECAT	VARCHAR2(100)

TRAVEL_DETAILS

Name	Null?	Type
TRID	NOT NULL	NUMBER(4)
EMP_ID		VARCHAR2(20)
MID		NUMBER(4)
JRFRM		VARCHAR2(100)
JRTO		VARCHAR2(100)
DOJ		DATE
MOT		VARCHAR2(20)
CLS		VARCHAR2(10)
FARE		NUMBER(12,2)

6. IMPLEMENTATION

6. 1. *SOURCE CODE:*

LOG IN

```
<html>

<head>

<p><img src="C:\Documents and Settings\Prithvi\Desktop\New Folder\umu.gif"width="980"height="150"alt="umu"></p>

<title>LOGIN</title>

</head>

<body>

<form>

<br>username:<input type="text"name="username"><br>

<br>password:<input type="text"name="password"><br>

<br><input type="submit"value="login">

</form>

</html>
```

CREATE :

```
<html>

<head>

<title>Travel Bill tracking system</title>

</head><body>
```

<form>

** <img src="C:\Documents and Settings\Prithvi\Desktop\New Folder\umu.gif"width="980"height="150"alt="umu"> **

** <h2><u>Enter The Following Details</u></h2> **

<table align="center" width="100%">

<tr>

<td align "center">First Name </td>

<td><input type="text"name="First Name"></td>

</tr>

<tr>

<td>Last Name</td>

<td><input type="text"name="Last Name"></td>

</tr>

<tr><td>Module</td>

<td><select name="module">

<option value="1st">-------------------------</option>

<option value="2nd">Health care</option>

<option value="3rd">Education & Adult Literacy</option>

<option value="4th">Water, Environment Sanitaion</option>

<option value="5th">Livelihoods</option>

<option value="6th">Virtual Delivery of Services</option>

<option value="7th">Gram IT</option>

**</select></td></tr>
**

```
<tr>
<td>Journey from</td>
<td><input type="text"name="Journey From"></td>
</tr>
<tr>
<td>Journey To</td>
<td><input type="text"name="Journey to"></td>
</tr>
<tr>
<td>Date of Journey</td>
<td><input type="text"name="date"><br></td>
</tr>
<tr>
<td>Mode of travel</td>
<td><input type="text"name="mode of travel"><br></td>
</tr>
<tr>
<td>Class</td>
<td><input type="text"name="class"><br></td>
</tr>
<tr>
<td>Fare</td>
<td><input type="text"name="fare"><br><td>
```

</tr>

<tr>

<td></td>

<td><input name="submit"type="submit"value="submit"><input name="reset"type="reset"value="reset"></td>

</tr>

** **

</form>

</body>

</html>

OPTIONS

<html>
<head>
** <img src="C:\Documents and Settings\Prithvi\Desktop\New Folder\umu.gif"width="980"height="150"alt="umu"> **
<title>Welcome To Travel Bill Tracking System</title>

DELETE

</head>
** please choose an option **
<href="create.html">
**
Create**
**
<href="retrieve.html">**
**
Retrieve**
**
<href="delete.html">**
**
Delete**

```
</form>
</body>
</html>
```

RETRIEVE

```
<html>
<head>
<title>Retrieve a bill</title>
</head>
<body>
<form>
<p><img src="C:\Documents and Settings\Prithvi\Desktop\New Folder\umu.gif"width="980"height="150"alt="umu"></p>
<p><b>search by:</b></p>
<br>Module:<select name="module"><br>
<option value="1st">Health</option>
<option value="2nd">School education</option>
<option value="3rd">Adult Literacy</option>
<option value="4th">Drinking Water</option>
</select><br>
<br>name of partner:<input type="text"name="name of partner"><br>
<br>date:<select name="date">
<option value="1st">1</option>
<option value="2nd">2</option>
<option value="3rd">3</option>
<option value="4th">4</option>
<option value="5th">5</option>
<option value="6th">6</option>
<option value="7th">7</option>
<option value="8th">8</option>
<option value="9th">9</option>
<option value="10th">10</option>
```

<option value="11th">11</option>

<option value="12th">12</option>

<option value="13th">13</option>

<option value="14th">14</option>

<option value="15th">15</option>

<option value="16th">16</option>

<option value="17th">17</option>

<option value="18th">18</option>

<option value="19th">19</option>

<option value="20th">20</option>

<option value="21st">21</option>

<option value="22nd">22</option>

<option value="23rd">23</option>

<option value="24th">24</option>

<option value="25th">25</option>

<option value="26th">26</option>

<option value="27th">27</option>

<option value="28th">28</option>

<option value="29th">29</option>

<option value="30th">30</option>

<option value="31st">31</option>

</select>

month:<select name="mm">

<option value="1st">jan</option>

<option value="2nd">feb</option>

<option value="3rd">mar</option>

<option value="4th">apr</option>

<option value="5th">may</option>

<option value="6th">jun</option>

<option value="7th">jul</option>

<option value="8th">aug</option>

<option value="9th">sep</option>

<option value="10th">oct</option>

<option value="11th">nov</option>

<option value="12th">dec</option>

</select>

year:<select name="yy">

<option value="1st">2006</option>

<option value="2nd">2007</option>

<option value="3rd">2008</option>

<option value="4th">2009</option>

<option value="5th">2010</option>

<option value="6th">2011</option>

<option value="4th">2012</option>

<option value="4th">2013</option>

<option value="4th">2014</option>

<option value="4th">2015</option>

</select>

**

<input type="submit"value="search">**

</form>

</body>

</html>

WEB:

<html>

<head>

<title>

Person Form

</title>

</head>

<body>

```
<form action="Seven.jsp" method="post">
<center><h2><u>Input person details and press SUBMIT
 button</u></h2></center>
<table align="center" width="100%">
<tr>
<td align "center"> Enter Person name</td>
<td><input type=text name=stName size=10 maxlength=10></td>
</tr>  <tr>
<td align "center"> Enter Person age</td>
<td><input type= int name=stAge size=20 maxlength=15></td>
</tr><tr>
<td align "center"> Enter Person weight</td>
<td><input type= int name=stWeight size=10
 maxlength=14></td>
</tr>
</table>
<center><input type="submit" value="submit"></center>  </form>
</body>
</html>
```

INPUT, SELECT,TEXTAREA

{

```
  color          : #000066;
  font-family    : Tahoma;
  font-size      : x-small;
  border-width   : 1px;
  border-style   : solid;
  border-color   : ##000066;
}
<style>
a
{
        color:"Black";
        text-decoration:none;
}
a:hover
{
         color:white;
         text-decoration:underline;
}
</style>
```

```
INPUT.fullWidth, TEXTAREA.fullWidth, SELECT.fullWidth

{

  /* for some reason, 100% will truncate right edge */

  width          : 100%;

}



BODY

{

        FONT-SIZE: 10px; BACKGROUND: #fffddd; VISIBILITY: visible; COLOR: #000000; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif

}

.topPage {

        PADDING-RIGHT: 0px; PADDING-LEFT: 0px; BACKGROUND: url(Img/fBg.png) #000066; PADDING-BOTTOM: 0px; MARGIN: 0px; PADDING-TOP: 0px

}

.rightPage {

        MARGIN: 20px

}

.leftPage {

        BACKGROUND: #dcebff
```

}

.title {

PADDING-RIGHT: 0px; PADDING-LEFT: 0pt; BACKGROUND: #000066; VISIBILITY: visible; PADDING-BOTTOM: 0px; MARGIN: auto; VERTICAL-ALIGN: middle; WIDTH: 100%; LINE-HEIGHT: 52px; PADDING-TOP: 0px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif; HEIGHT: 52px

}

.title1 {

PADDING-RIGHT: 0px; DISPLAY: block; PADDING-LEFT: 10px; FONT-WEIGHT: bold; FONT-SIZE: 16pt; FLOAT: left; VISIBILITY: visible; MARGIN: auto; VERTICAL-ALIGN: middle; COLOR:#ffddd; LINE-HEIGHT: 52px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif

}

.separator1 {

DISPLAY: block; FONT-SIZE: 5pt; BACKGROUND: #ffffff; LEFT: 6px; FLOAT: left; VISIBILITY: visible; MARGIN-LEFT: 6px; VERTICAL-ALIGN: middle; WIDTH: 10pt; COLOR: #ffffff; LINE-HEIGHT: 3px; MARGIN-RIGHT: 6px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif; POSITION: relative; TOP: 26px; HEIGHT: 3px

}

.title2 {

PADDING-RIGHT: 0px; DISPLAY: block; PADDING-LEFT: 10px; FONT-WEIGHT: normal; FONT-SIZE: 16pt; FLOAT: left; VISIBILITY: visible; PADDING-BOTTOM: 0px; VERTICAL-ALIGN: middle; COLOR: #ffffff; LINE-HEIGHT: 52px; PADDING-TOP: 0px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif

}

.separator2 {

DISPLAY: none; FONT-SIZE: 20pt; FLOAT: left; VISIBILITY: visible; VERTICAL-ALIGN: middle; WIDTH: 20pt; COLOR: #ffffff; LINE-HEIGHT: 52px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif

}

.titleimage {

PADDING-RIGHT: 10px; VISIBILITY: visible; COLOR: #ffffff; LINE-HEIGHT: 20px; POSITION: relative; TOP: 6pt; TEXT-ALIGN: right

}

.topMenu {

PADDING-RIGHT: 0pt; BORDER-TOP: white 1px solid; DISPLAY: block; PADDING-LEFT: 0pt; BACKGROUND: url(Img/A04.png); PADDING-BOTTOM: 0pt; MARGIN: 0pt; WIDTH: 100%; LINE-HEIGHT: 22px; PADDING-TOP: 0pt; BORDER-BOTTOM: black 1px solid

}

.topMenu A {

PADDING-RIGHT: 6pt; DISPLAY: block; PADDING-LEFT: 6pt; FONT-SIZE: 8pt; BACKGROUND: url(Img/A04.png); VERTICAL-ALIGN: middle; COLOR: white; LINE-HEIGHT: 22px; WHITE-SPACE: nowrap; HEIGHT: 22px; TEXT-DECORATION: none

}

.topMenu A:hover {

DISPLAY: block; BACKGROUND: url(Img/A04.png); LINE-HEIGHT: 22px; HEIGHT: 22px; TEXT-DECORATION: underline

}

.selected {

PADDING-RIGHT: 6pt; DISPLAY: block; PADDING-LEFT: 6pt; FONT-SIZE: 8pt; BACKGROUND: url(Img/A02.png); MARGIN: 0pt; COLOR: white; LINE-HEIGHT: 22px; WHITE-SPACE: nowrap

}

.bookmark {

FONT-WEIGHT: normal; FONT-SIZE: 8pt; VISIBILITY: visible; WIDTH: 100%; PADDING-TOP: 14px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif; TEXT-ALIGN: left

}

.bookmark A {

PADDING-RIGHT: 2px; DISPLAY: block; PADDING-LEFT: 17px; FONT-WEIGHT: normal; FONT-SIZE: 8pt; VISIBILITY: visible; PADDING-BOTTOM: 2px; WIDTH: 100%; COLOR: #333333; LINE-HEIGHT: 12px; PADDING-TOP: 2px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif; HEIGHT: 12px; TEXT-ALIGN: left; TEXT-DECORATION: none

}

.bookmark A:link {

}

.bookmark A:visited {

}

.bookmark A:active {

}

.bookmark A:hover {

FONT-WEIGHT: bold; COLOR: #000000}

.caption1 {

MARGIN-TOP: 14px; FONT-WEIGHT: bold; FONT-SIZE: 15pt; BACKGROUND: #227CA8; VISIBILITY: visible; PADDING-BOTTOM: 1px; COLOR: #ffffff; LINE-HEIGHT: 20px; PADDING-TOP: 1px; BORDER-BOTTOM: #333333 1pt solid; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif; TEXT-ALIGN: center

}

.width770 {

WIDTH: 770px

}

.caption2 {

PADDING-RIGHT: 0px; PADDING-LEFT: 0px; FONT-WEIGHT: bold; FONT-SIZE: 10pt; VISIBILITY: visible; PADDING-BOTTOM: 5px; MARGIN: 15px 0px 5px; COLOR: #ff6600; PADDING-TOP: 5px; FONT-FAMILY: Verdana, Geneva, Arial, Helvetica, sans-serif

}

.tabformat {

BORDER-RIGHT: #cccccc 1px solid; BORDER-TOP: #cccccc 1px solid; BORDER-LEFT: #cccccc 1px solid; BORDER-BOTTOM: #cccccc 1px solid

}

top; COLOR: black

}

.selectedLeft {

BORDER-TOP: #ffffff 1px solid; DISPLAY: block; PADDING-LEFT: 6px; FONT-WEIGHT: bold; FONT-SIZE: 8pt; BACKGROUND: white; VERTICAL-ALIGN: middle; WIDTH: 194px; COLOR: #ff6600; LINE-HEIGHT: 20px; TEXT-DECORATION: none

}

.leftPageLeft {

BACKGROUND: #dcebff

}

TABLE:

create dsn name with tbts;

connect system/javasoft;

create user tbts identified by tbts;

grant dba to tbts;

connect tbts/tbts;

create table employee_info(emp_id varchar2(20) constraint emp_id_pk primary key,

name varchar2(50) not null,

gen varchar2(7) not null,

phno number(12) ,

address varchar2(100) not null,

qlfy varchar2(20) not null,

doj date not null);

create table adminlogin(username varchar2(30) not null,

pwd varchar2(30) not null);

```
create table modules_info(mid number(3)  primary key,
modulename varchar2(100),modulecat varchar2(100));
create table travel_details(trid number(4) primary key,
emp_id varchar2(20) references employee_info(emp_id),
mid number(4)  references modules_info(mid),
jrfrm varchar2(100),
jrto varchar2(100),
doj date,
mot varchar2(20),
cls varchar2(10),
fare number(12,2));
```

7. SYSTEM TESTING

System Testing is a process of executing a program with the explicit intention of finding errors, which cause program failure. There are two general strategies for testing software. They are :

7.1Code Testing

7.2 Specification testing

7.1 CODE TESTING

This strategy examines the logic of a program and has been carried out to identify three levels of correctness of programs. Possible correctness is first achieved by giving arbitrary inputs. Then the inputs are carefully selected to obtain predicted output. This gives the probable correctness. All potentially problematic areas are checked in this way for the software to achieve probable correctness. Absolute correctness can be demonstrated by a test involving every possible combination of inputs. However, this cannot be performed with the software but to the existence of the various possible combinations of the inputs and due to time restrictions.

7.2 SPECIFICATION TESTING

The specifications are examined which states what the program should do and how it should perform under various conditions. Then test cases are developed for each condition or combinations of conditions and submitted for processing. By examining the results, it is determined whether the program performs according to its specified requirements.

7.3 LEVELS OF TESTING

The two levels of Testing are

- Unit Testing
- System Testing

7.3.1 UNIT TESTING

Unit testing is done for the programs making up the systems. It is focused to find out module errors and enables to detect errors in coding and logic that are contained in the module. Unit testing is performed from bottom-up, starting with the smallest and lowest levels modules and proceeding one.

7.3.2 SYSTEM TESTING

At a time System Testing finds out the discrepancies between the system and its original objective, current specifications and systems documentation.

The training session consists of getting the users used to software by asking them to perform data entry in our presence and look into the problems if encountered .

Testing can be done in two ways.

1. Sample Tests
2. Real Tests

7.4 SAMPLE TESTS

The software was tested with sample data that we randomly selected. I tested all functions with such random data and I was successful in getting accurate results. It was at this time I got to know certain intricacies of the system that I had overlooked.Without much delay however, I got over the problems and managed to perfect the software at least to the extent possible.

7.5 REAL TEST

For the real test, I have planned to do in due course. I initialized the software and creation of entities through the updation module, transaction entries through the transaction module and generated reports with the estimations. The various information retrieval functions as per user need are also implemented

7.6TEST CASES

FOR LOGIN PAGE

<table>
<tr><td>Test case#:1</td><td>Priority(H,L):High</td></tr>
<tr><td colspan="2">Test Objective:Correct login details</td></tr>
<tr><td colspan="2">Test Description:User id and password checked</td></tr>
<tr><td colspan="2">Requirements verified:User id and password are checked in database</td></tr>
<tr><td colspan="2">Test Environment:Java</td></tr>
<tr><td colspan="2">Test setup or pre-conditions:User initiates any control mechanism like Submit or Go buttons</td></tr>
<tr><td>Actions</td><td>Expected Results</td></tr>
<tr><td rowspan="3">Incorrect login

Correct login</td><td>A message “Invalid user id/password” will be displayed

Enter into application</td></tr>
<tr><td>Actual Results</td></tr>
<tr><td>A message “valid user id/password” will be displayed

Enter into application</td></tr>
<tr><td colspan="2">Pass:Yes Conditional Pass: Fail:</td></tr>
<tr><td colspan="2">Problems/Issues:Nil</td></tr>
</table>

Fig:7.1 Test case for login page

FOR LOGIN STATUS

Test case#:2	**Priority(H,L):**High
Test Objective: Login user only allowed requesting servicesprovided by the system	
Test Description: Accessing permissions	
Requirements verified: Login status should be verified	
Test Environment: Java	
Test setup or pre-conditions: If user tries to request services	
Actions	**Expected Results**
In login status	User should be able to request services.
If not in the login status	All services are disabled.
	Actual Results
	User should be able to request services.
	All services are available.
Pass: Yes **Conditional Pass:** **Fail:**	
Problems/Issues: Nil	

Fig:7.2 Test case for login status

FOR REGISTRATION PAGE

Test case#:3	**Priority(H,L):**High
Test Objective: To register a new user.	
Test Description: The user selects the option "User Registration" on the mobile phone;as a result it shows the fields to fill as a form.	
Requirements verified: Whether user already exists or not is to be verified.	
Test Environment: Java	
Test setup or pre-conditions: If user tries to register	
Actions	**Expected Results**
The Registration form	Fill the form details.
	Actual Results
	Fill the form details
Pass: Yes **Conditional Pass:** **Fail:**	
Problems/Issues: Nil	

Fig:7.3 Test case for Registration page

7.7 DISCUSSION OF RESULTS

SCREENSHOT FOR LOGIN PAGE

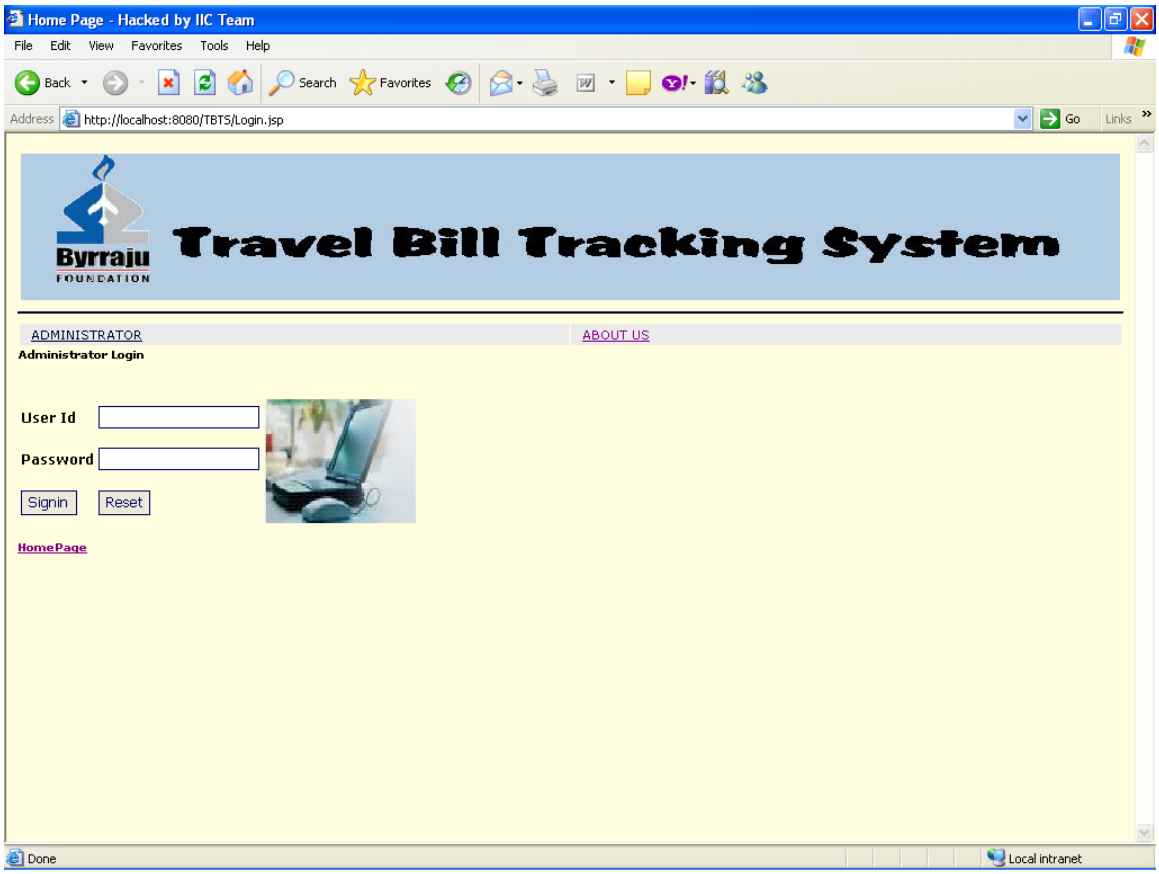


Fig 7.4 Screenshot for Login Page

- This screen shot is for login page where we can see the module of Travel Bill Tracking system to login into home page.From this option the admin can login into the website.The users can also login using their id and password.New user has to select the signup option to create an user account and then he can login.

SCREENSHOT FOR EMPLOYEE INFORMATION PAGE

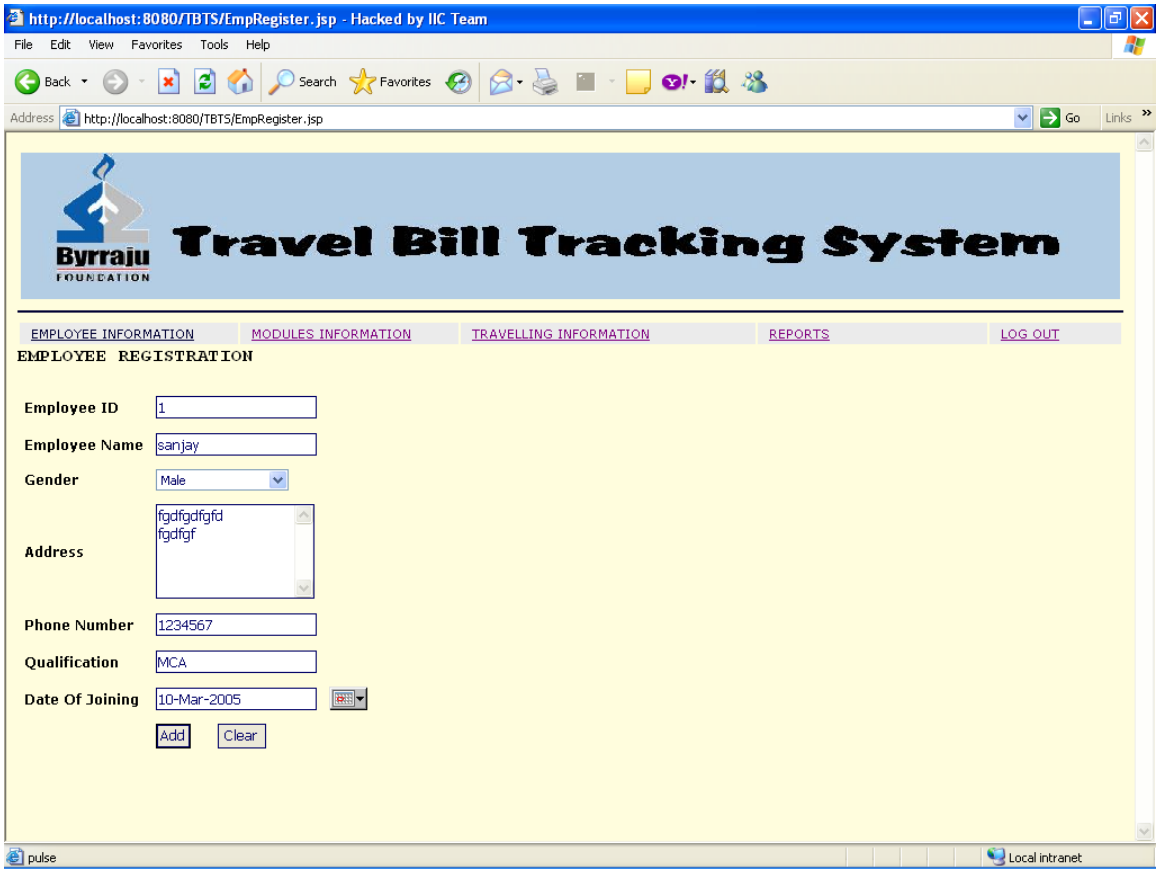


Fig 7.5 Screenshot for Employee Information Page

- This screenshot is for main page of admin where after login we get all the detailof various employee information.,module information , travelling information, reports and This screenshot is for main page of employee info where after login we get all the detailsof empid,name,gender,phone num,address,qualification etc..,

SCREENSHOT FOR MODULE INFORMATION

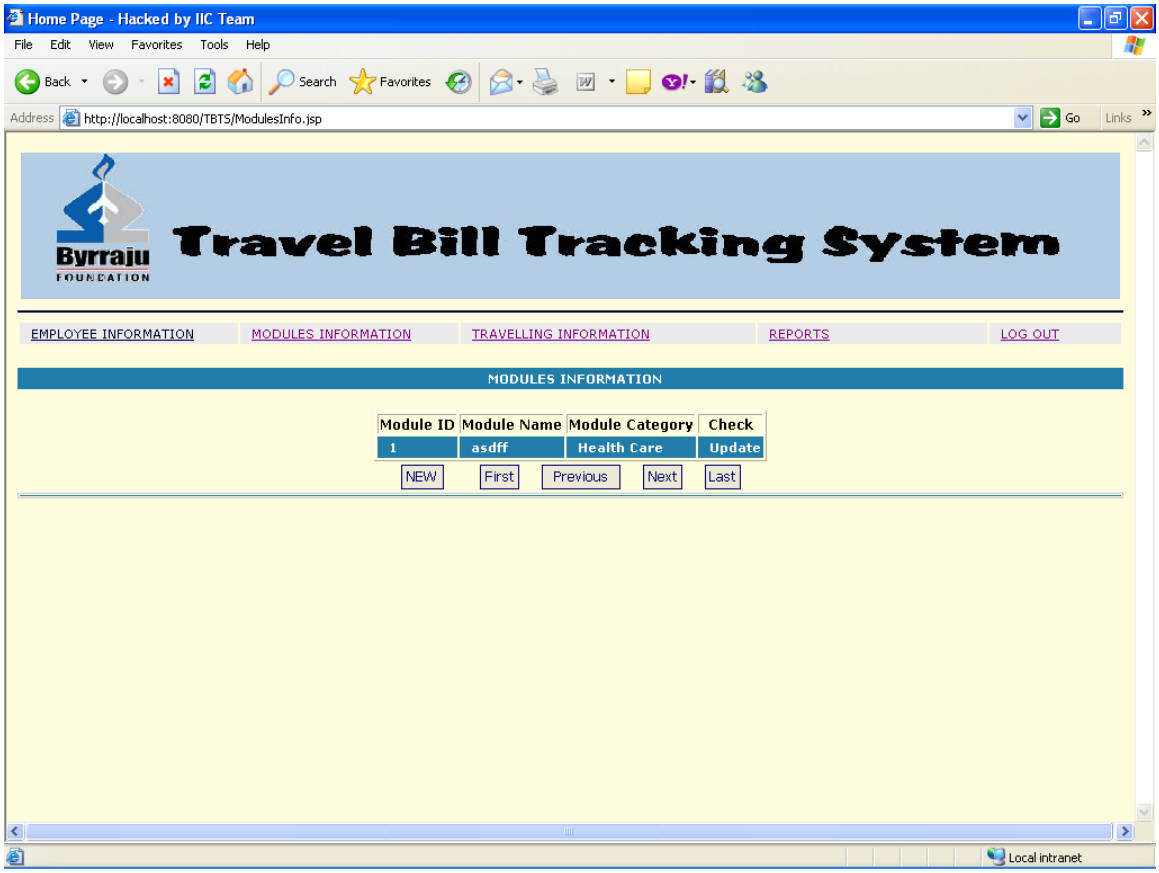


Fig 7.6 Screenshot for Module Information Page

- This screenshot is for viewing the module information.It contains the details like module ID,module name,,,module category and check and we can add the new modules that are introduced by selecting new option.

SCREENSHOT FOR TRAVELLING INFORMATION

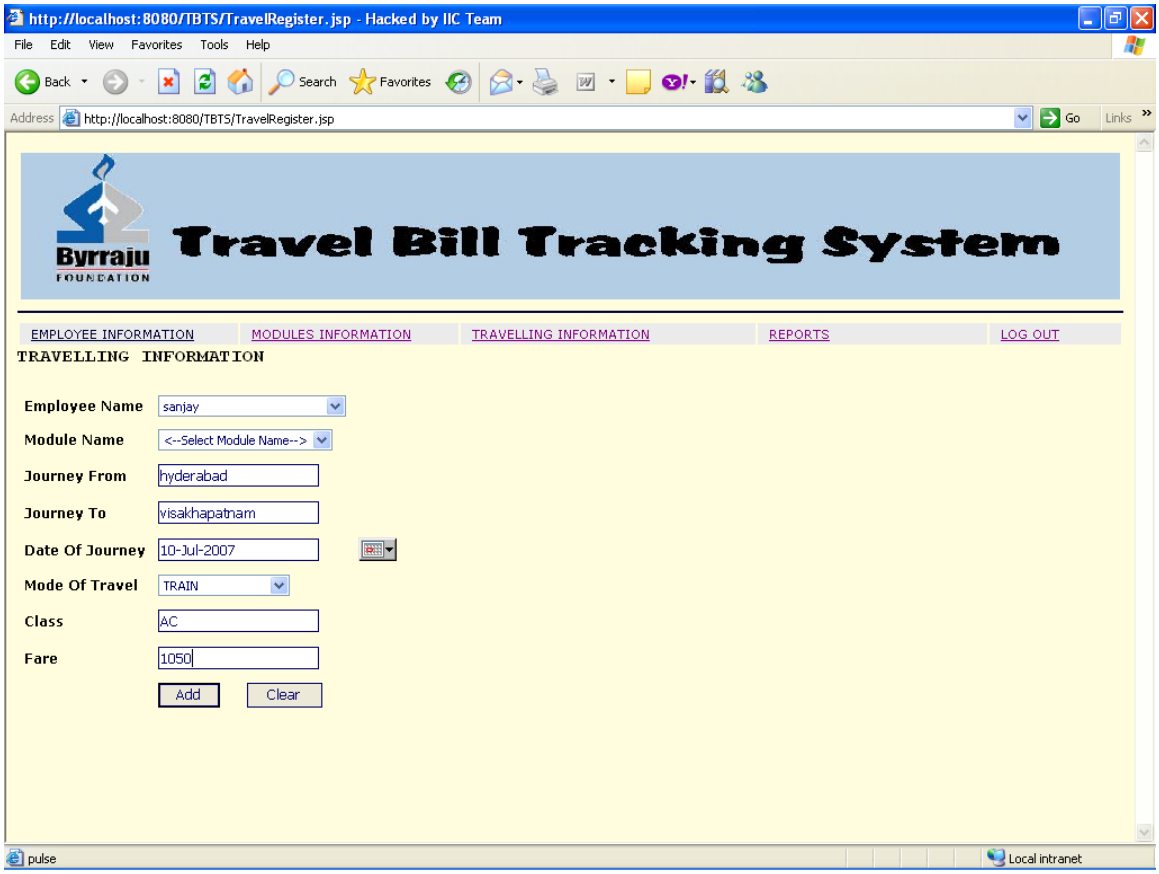


Fig 7.7 Screenshot for Travelling Information Page

- This screenshot is for viewing the Travelling information of Travel BillTracking System..It contains the details like employee name,module name,journey from, journey to date of journey mode of travel, class,fare …and we can add the new tavelling details of the other user.

SCREENSHOT FOR TRAVEL INFORMATION

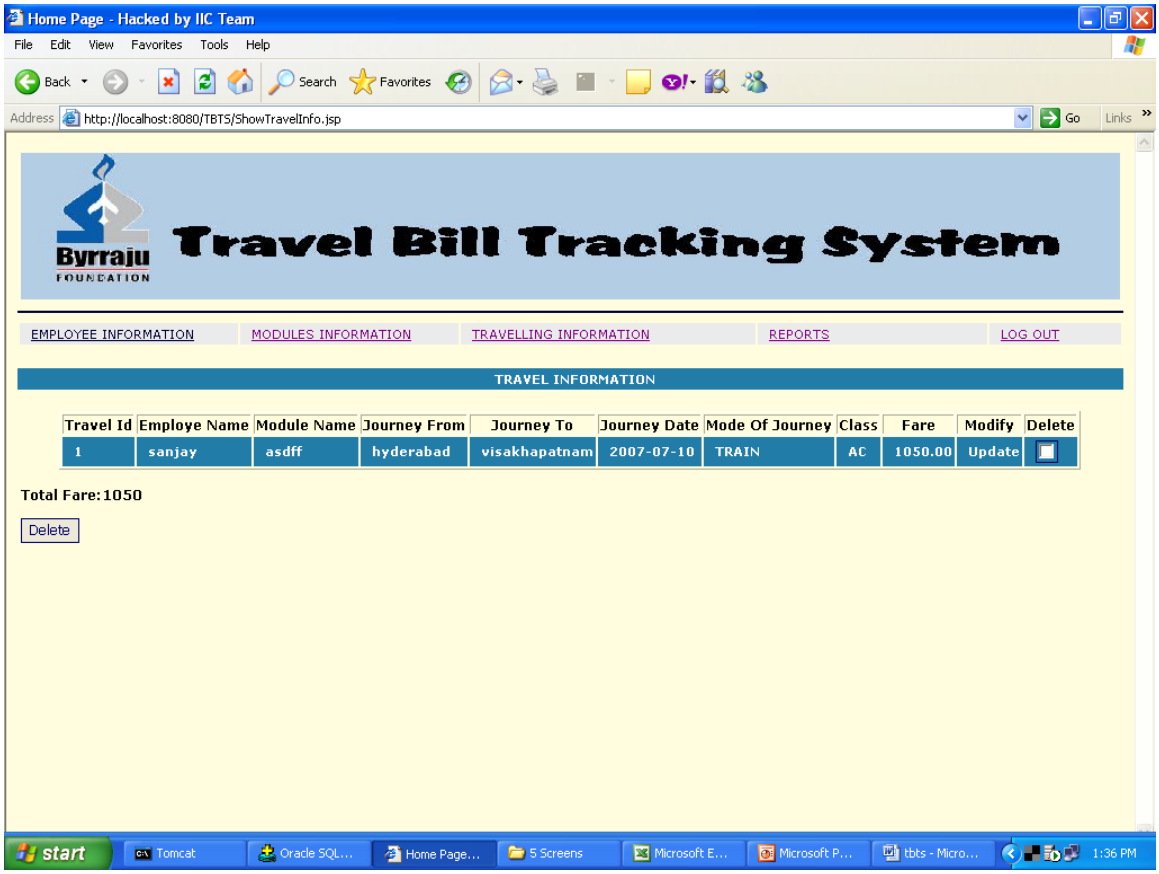


Fig 7.8 Screenshot for Travel Information Page Of TBTS

- This screenshot is for viewing the Travel information of Travel BillTracking System..It contains the details like travel ID, employee name,,module name,journey from, journey to date of journey mode of travel, class,fare and modify the details and wish to delete it . the total fare is updated...we can delete the new tavelling details of the user by selecting delete option..

8. CONCLUSION AND FUTURE ENHANCEMENTS

8.1. CONCLUSION:

The main objective of the project is to automate the"TRAVEL BILL TRACKING SYSTEM". By using JAVA as front end and DATABASE as back end under WINDOWS XP environment..

The efficiency of any system designed to suit an organization depends cooperation during the implementation stage and also flexibility of the system to adopt itself to the organization. Travel Bill Tracking Systems Software takes care of the activities related to the Travel information provided .. The major responsibility is to take care of the proper utilization of different schemes in a transparent way.

This system is used for efficient functioning of the travel details information within an organization. It deals with maintaining administrator, travel information, module information, report information..

Reliable and accurate reports could be available within a very short time, which is done manually..

8.2 FUTURE ENHANCEMENTS:

The entire project has been developed and deployed as per requirements stated by the user, it is found to be bug free as per the testing standards that are implemented.. Any specification untraced errors will be concentrated in coming versions, which are planned to be developed in near future.

This system is ever ready to attend the following needs of future:

- Interaction with external software.
- Cross platform functionality.
- Facility to send any report by E-mail.
- Web enabling of student's progress, so that parents can retrieve the information on web.
- Any other feature that client needs

9. REFERENCES

- Padala Rama Reddy(1990), Detailed Standard Specification and General Principles of Engineering Contracts, 6th Edition, M/S Panchayat publications, Hyderabad.
- Ian Sommerville (1999), Detailed Standard Specification and General Principles of Engineering Contracts, 6th Edition, M/S Panchayat publications, Hyderabad.
- Walker Royce(1996), Software Project Management, 3rd Edition, Pearson Education Private Limited, Singapore.
- Larne Pedowsky(2000), Java Server Pages, 1st Edition, Pearson Education Private Limited, Singapore.
- Kevin Loney(2002), Oracle8i The Complete Reference, 2nd Edition, Tata McGraw Hill Publishing Company Limited, New Delhi.

www.eprocurement.gov.in

www.C1India.com

www.tendersonline.com

www.artimadevelopment.com