

California State University, San Bernardino

**CSUSB ScholarWorks**

---

Theses Digitization Project

John M. Pfau Library

---

2006

## Real-time finance management system

Abdul Muqtadir

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>

---

### Recommended Citation

Muqtadir, Abdul, "Real-time finance management system" (2006). *Theses Digitization Project*. 2992.  
<https://scholarworks.lib.csusb.edu/etd-project/2992>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact [scholarworks@csusb.edu](mailto:scholarworks@csusb.edu).

# REAL-TIME FINANCE MANAGEMENT SYSTEM

---

A Project  
Presented to the  
Faculty of  
California State University,  
San Bernardino

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science  
in  
Computer Science

---

by  
Abdul Muqtadir

March 2006

REAL-TIME FINANCE MANAGEMENT SYSTEM

---

A Project  
Presented to the  
Faculty of  
California State University,  
San Bernardino

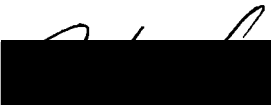
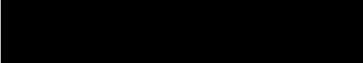
---

by


Abdul Muqtadir



March 2006

Approved by:

  
  
\_\_\_\_\_  
Dr. Ernesto Gomez, Chair, Computer Science

3/6/2006  
Date

  
\_\_\_\_\_  
Dr. Richard Botting

  
  
\_\_\_\_\_  
Dr. Keith Schubert

© 2006 Abdul Muqtadir

## ABSTRACT

Almost everybody earns money in this world, but very few of them actually spend it in a wise way. There is no doubt that managing finances will help anybody to have a secure and safe financial future. I have attempted to solve this problem by creating a Real-Time Finance Management System (RFMS) wherein users can easily manage their finances.

RFMS will also help users to learn more about stocks. They can create a sample portfolio of stocks and monitor them closely to see which direction their picked stocks are headed. Graphs have been provided in the system for them to easily visualize and see the numbers in a clear perspective.

Apart from stocks RFMS helps the users to manage their finances using the different finance management tools and calculators provided.

This is a Java/XML based approach where-in real-time market data from different stock exchanges is fetched and displayed for the user. This is done internally using a data access layer. The information we get is in an XML format, which is taken by the Java program and displayed, using Java Server Pages.

## ACKNOWLEDGMENTS

First and foremost I would like to thank my parents for putting in me the desire and the importance of education. I also thank them for giving me the strength and confidence that I can achieve anything in life.

During the course of my project and writing the thesis there have been people who gave support both academically and professionally. I would like to thank my advisor Dr. Ernesto Gomez and my committee members Dr. Botting and Dr. Schubert for giving me the education and encouragement for my project. I would also like to thank my friends and family for the confidence that they have given me.

## TABLE OF CONTENTS

ABSTRACT .....	iii
ACKNOWLEDGMENTS .....	iv
LIST OF FIGURES .....	viii
CHAPTER ONE: SOFTWARE REQUIREMENTS SPECIFICATION	
1.1 Introduction .....	1
1.2 Purpose of the Project .....	2
1.3 Context of the Problem .....	2
1.4 Related Work .....	2
1.5 Significance of the Project .....	3
1.6 Assumptions and Limitations .....	3
1.7 Definition of Terms .....	4
CHAPTER TWO: SOFTWARE DESIGN	
2.1 Introduction .....	9
2.2 Preliminary Design .....	9
2.3 Detailed Design .....	13
2.4 System Setup .....	17
2.5 Summary .....	19
CHAPTER THREE: SYSTEM VALIDATION	
3.1 Introduction .....	20
3.2 Unit Test Plan .....	20
3.3 Integration Test Plan .....	21
3.4 System Test Plan .....	24
CHAPTER FOUR: MAINTENANCE	
4.1 Introduction .....	25

4.2 Maintenance Guidelines .....	25
4.2.1 Interfaces Management .....	25
4.2.2 Administration .....	26
4.3 Tools Utilized .....	26
4.4 Directory Structure .....	26
CHAPTER FIVE: PROJECT IMPLEMENTATION	
5.1 Introduction .....	28
5.2 System Interfaces .....	29
5.2.1 Login .....	29
5.2.2 Portfolio Management .....	29
5.2.3 Reminder Setup .....	33
5.2.4 Debt Reduction Planning .....	34
5.2.5 Expenditure Analysis .....	35
5.2.6 Account Information .....	36
CHAPTER SIX: CONCLUSIONS AND FUTURE DIRECTIONS	
6.1 Conclusions .....	39
6.2 Future Directions .....	40
APPENDIX: SOURCE CODE OF JAVA CLASSES .....	42
REFERENCES .....	78



## LIST OF FIGURES

Figure 1. Administrator Use Case Diagram .....	11
Figure 2. User Use Case Diagram .....	11
Figure 3. 3-Tier Architecture Diagram .....	13
Figure 4. Model 1 Architecture Diagram .....	15
Figure 5. Login Page .....	29
Figure 6. Portfolio Management Page .....	30
Figure 7. Get Stock Quote Page .....	31
Figure 8. Get Top Trading Stock List .....	32
Figure 9. Time and Sales Report .....	33
Figure 10. Reminder Setup Page .....	34
Figure 11. Debt Reduction Planning Page .....	35
Figure 12. Expenditure Analysis Page .....	36
Figure 13. Account Information Page .....	37
Figure 14. Balance Check Book Page .....	38

## CHAPTER ONE

### SOFTWARE REQUIREMENTS SPECIFICATION

#### 1.1 Introduction

Almost everybody earns money in this world, but very few of them actually spend it in a wise way. There is no doubt that managing finances will help anybody to have a secure and safe financial future. I have attempted to solve this problem by creating a Real-Time finance management system - RFMS, wherein users can easily manage their finances.

RFMS will also help users to learn more about stocks. They can create a sample portfolio of stocks and monitor them closely to see which direction are their picked stocks headed. Graphs have been provided in the system for them to easily visualize and help them see the numbers in a clear perspective.

Apart from stocks RFMS helps the users to manage their finances using the different finance management tools and calculators provided.

This is a Java/XML based approach where real-time market data from different stock exchanges is fetched and displayed for the user. This is done internally using a data access layer. The information we get is in an XML

format which is taken by the Java program and displayed using Java Server Pages.

The project divides finance management into 4 general categories. These include Portfolio Management, Reminders, Debt Management, and Account Management. Each of these categories has different tools to help users with their finances.

### 1.2 Purpose of the Project

The purpose of the project was to develop an online finance management system where the users can learn and manage their stock portfolio and personal finances which would help them have control over their investments, personal finances or debts.

### 1.3 Context of the Problem

The context of the problem was to address issues of personal finance management and problems of having to do so from one place. Also proprietary software had to be installed on every computer the users want to use for managing their finances.

### 1.4 Related Work

There has been lot of work done in the field of personal finance management software, but most of the work

is done for stand-alone systems where the users usually have to buy and install proprietary software on their computer.

### 1.5 Significance of the Project

RFMS was developed for providing a one stop place for dealing with most of the problems in personal finance management. The users are under no obligation to use this tool a set number of times. Based on their present situation the users can use the different tools provided in the system.

### 1.6 Assumptions and Limitations

The following assumptions and limitations were made regarding the project:

1. To manage accounts using RFMS users need to enter their daily transactions into the system.
2. Users have all the required data for Credit Card Pay-Down such as Beginning Balance, Annual Percentage Rate and the Minimum Payment amounts.
3. User should have at least a very basic knowledge about stocks and managing portfolios.
4. The interest calculations used in the system are either Simple Interest or Compound Interest Calculations.

5. J2EE - Java 2 Platform, Enterprise Edition.  
Sun's Java platform for multi-tier server oriented enterprise applications.
6. JDK - Java Development Kit, A free Sun Microsystems product which provides the environment required for programming in Java.  
The JDK is available for variety of platforms, such as Sun Solaris, Microsoft Windows and Linux.
7. JVM - Java Virtual Machine. It is a software that interprets and executes the byte code in Java class files.
8. JDBC - Java DataBase Connectivity. A programming interface that lets Java applications access a database via the SQL Language.
9. JSP - JavaServer Page, an extension to the Java servlet technology from Sun Microsystems that provides a simple programming vehicle for displaying dynamic content on a web page.
10. JavaBean - A component architecture for the Java programming language, developed initially by Sun, but now available from several other vendors.  
JavaBeans components are called "Beans".
11. JavaScript - A Scripting Language that is widely supported in the web browsers and other web

tools. It adds interactive functions to HTML pages which are otherwise static.

12.XML - The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language for creating special-purpose markup languages. It is a simplified subset of SGML, capable of describing many different kinds of data. Its primary purpose is to facilitate the sharing of data across different systems, particularly systems connected via the Internet. Languages based on XML (for example, RDF, RSS, MathML, XHTML, SVG, and cXML) are defined in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form.

13.JAXP - JAXP or Java API for XML Parsing is an optional API provided by Javasoft. It provides basic functionality for reading, manipulating, and generating XML documents through pure Java APIs. It is a thin and lightweight API that provides a standard way to seamlessly integrate any XML-compliant parser with a Java application.

14.JDOM - JDOM is a Java-based document object model for XML that integrates with Document Object

- Model (DOM) and Simple API for XML (SAX) and uses parsers to build the document.
- 15.XERCES - Xerces is a set of parsers compatible with Extensible Markup Language (XML). Xerces parsers are available for Java and C++, implementing World Wide Web Consortium (W3C) XML, Document Object Model (DOM), and Simple API for XML (SAX) standards.
  - 16.SAX - SAX or Simple API for XML is an event-driven, serial-access mechanism for accessing XML documents. Provides a standardized interface for the interaction of applications with many XML tools. SAX uses two basic types of objects, Parser and Document Handlers.
  - 17.SOAP - Simple Object Access Protocol SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope, a set of encoding rules, and a convention for representing remote procedure calls and responses.
  - 18.DOM - Document Object Model (DOM) is a form of representation of structured documents as an object-oriented model. DOM is the official World

Wide Web Consortium (W3C) standard for representing structured documents in a platform- and language-neutral manner. The DOM is also the basis for a wide range of application programming interfaces, some of which are standardized by the W3C.

19.REST - Representational State Transfer is a model for web services based solely on HTTP. REST takes the view that the Web already has everything necessary for web services, without having to add extra specifications like SOAP and UDDI. Any item can be made available (i.e. represented) at a URI, and, subject to the necessary permissions, it can be manipulated using one of the simple operations defined within HTTP (GET to retrieve information, PUT and POST to modify it, DELETE to remove it).

20.RFMS - Real-Time Finance Management System. This is the name of the directories where the .jsp files of RFMS are stored in tomcat. The same name has been used for the database of RFMS as well.



## CHAPTER TWO

### SOFTWARE DESIGN

#### 2.1 Introduction

Chapter Two consists of a discussion of the software design. Specifically, RFMS is a JSP based system that resides on three-tier architecture. The front-end was based on JSP, and the back-end was a MySQL database. The connections to the database are made through the Java Beans with a JDBC connection.

#### 2.2 Preliminary Design

RFMS is a system designed for users to manage their finances using the tools provided in the system. It was developed to overcome the following issues:

1. Getting the price for one stock at a time and basically managing your own portfolio of stocks.
2. Using actual money to learn and play around with stocks. This is an expensive mistake; users should have a lot of knowledge before they get into the stock market.
3. Using different reminders systems and still forgetting to pay bills on time.
4. Paying more interest than required in loans such as Student Loans, Car Loans, and Mortgages.

5. Difficulty in analyzing information from budget creation and maintenance.

6. Overdrawing and ruining your credit by simple mistakes of not balancing your check book as it requires too many calculations.

A robust, flexible and user-friendly system was needed and RFMS was an implementation that solved most of the issues.

Choosing the architecture was an important part. RFMS uses the 3 tier architecture that provides the user interfaces through the web browser. All the user interfaces are divided based on user type. The users are of 2 types:

- System Administrator
- User

The use case diagrams showing the individual functions of all the users are given in the following figures:

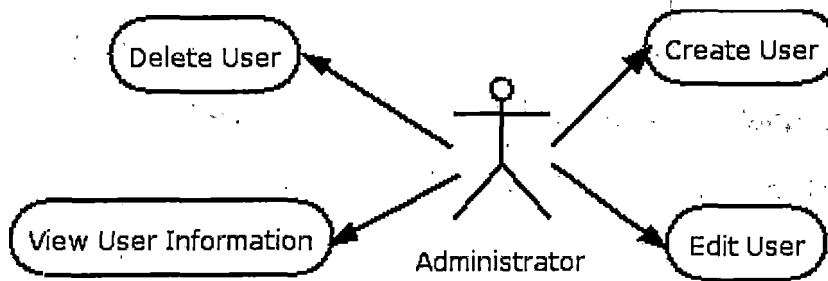


Figure 1. Administrator Use Case Diagram

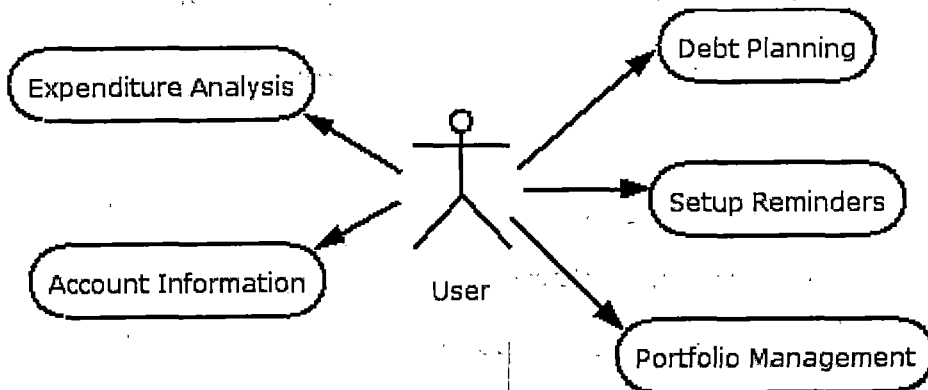


Figure 2. User Use Case Diagram

### 2.3 Detailed Design

Architecture: RFMS has a 3-tier architecture. A 3-tier architecture was considered since the 2-tier architecture combines both the presentation logic with the business logic in one tier called the client side. The other tier called the server provides the database.

The 3-tier architecture separates the business logic from the presentation logic and has the database in the third tier. This architecture is very flexible and has high scalability.

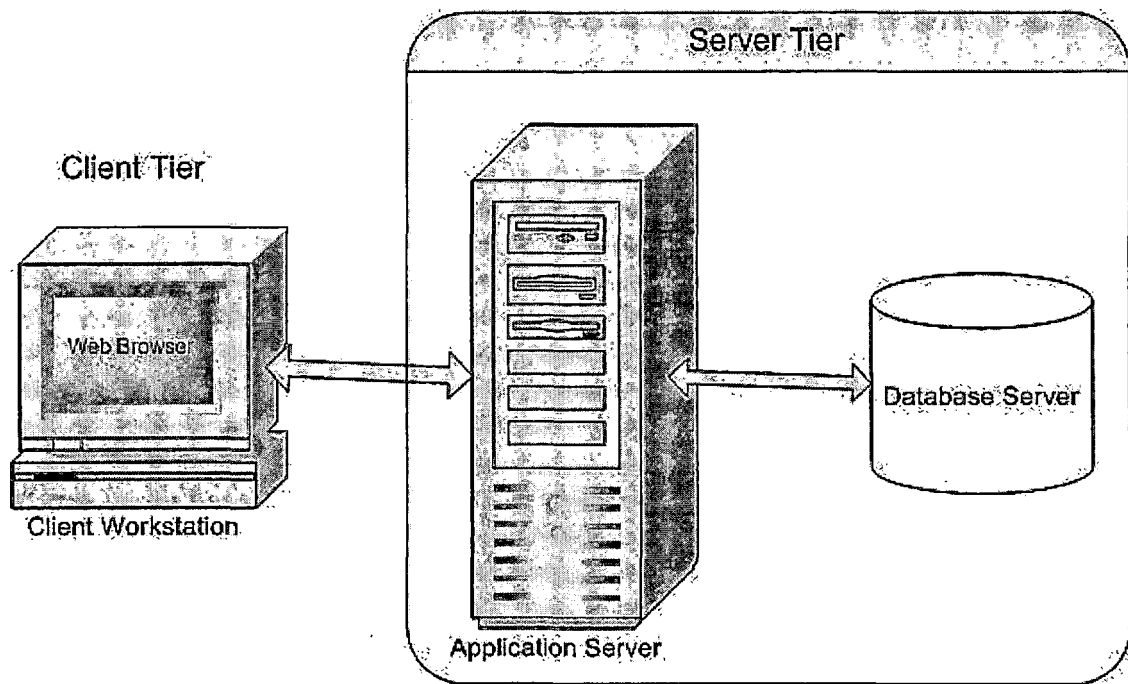


Figure 3. 3-Tier Architecture Diagram

A. Client Tier: The Java enabled web browsers are used as client tier. The user sends requests via the browser. The browser then sends the request to the Java Beans residing at the application server that process the request and send it back to the browser which in turn interprets the information it receives from the server and displays graphically for the client.

B. Middle Tier: The middle tier is also known as the application server. RFMS was designed to run in Jakarta Tomcat-5.0.30 web server. Tomcat supports JSP and JavaBeans which are the programming techniques used in the system.

C. Database Server Tier: The database tier is the back end application server where the data is stored. RFMS uses MySQL version 4.0.23-nt. MySQL provides very fast joins using an optimized one-sweep multi-join. You can connect MySQL to Tomcat using a MySQL driver.

Programming Technique and Language: The programming language used by RFMS was Java and its JSP technology. JSP standard was developed by Sun Microsystems as an alternative to Microsoft's active server page (ASP) technology. JSP pages are similar to ASP pages in the fact that they are compiled on the server, rather than in a user's Web browser.

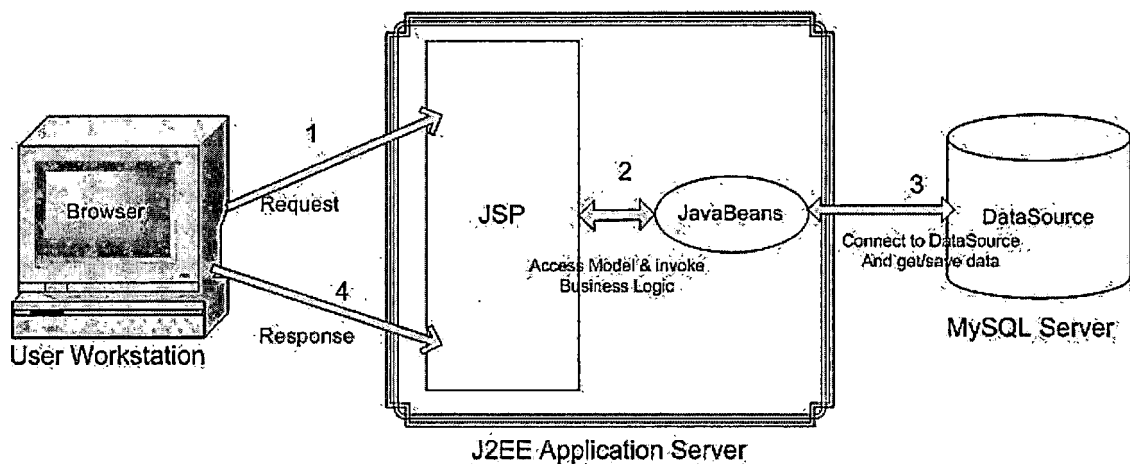


Figure 4. Model 1 Architecture Diagram

However, JSP is Java-based, whereas ASP is Visual Basic-based. JSP pages are useful for building dynamic Web sites and accessing database information on a Web server. Though JSP pages may have Java interspersed with HTML, all the Java code is parsed on the server. Therefore, once the page gets to the browser, it is only HTML. JavaScript, on the other hand, is usually parsed by the Web browser, not the Web server.

The other benefits of JSP that made it a choice over other technologies are:

1. JSP separates program logic from the presentation.
2. JSP pages have the "Write Once, Run Anywhere" property. By virtue of their ultimate translation to Java byte code, JSP pages are platform

independent. This means that JSP pages can be developed on any platform and deployed on any server.

3. JSP pages also make it easy to embed reusable components like JavaBeans that perform specialized tasks. JavaBeans are developed once and can be used in any number of Java Server Pages.
4. JSP technology has become a part of the J2EE which brings Java technology to enterprise computing. Using JSP pages for constructing a website, we can create a front-end component of the type of powerful N-tier applications made possible by J2EE.
5. JSP has custom tag libraries that make it highly extensible.

These numerous characteristics of JSP made it a good choice for RFMS.

The JavaBeans used in RFMS are:

1. StockInfo
2. Users
3. UsersAdmin
4. TimeAndSales
5. TopList

- 6. DateBean
- 7. Reminder
- 8. DebtPlanner
- 9. Expenditure
- 10. AccountInfo

## 2.4 System Setup

RFMS's development system involved the following steps:

1. Installation of J2SDK 1.4.2\_06. The Java 2 SDK is a development environment for building applications, applets, and components using the Java programming language. The Java 2 SDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform. These tools are designed to be used from the command line. Except for the appletviewer, these tools do not provide a graphical user interface.
2. Install Tomcat - Servlet/JSP container. RFMS uses Tomcat version 5.0.30 which implements the Servlet 2.4 and JavaServer Pages 2.0 specifications from the Java Community Process, and includes many additional features that make



it a useful platform for developing and deploying web applications and web services.

3. Install the MySQL database server: RFMS uses MySQL version 4.0.23-nt. The MySQL (R) software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL Server is intended for mission-critical, heavy-load production systems as well as for embedding into mass-deployed software.
4. Install a JDBC Driver: RFMS uses MySQL Connector/J which is a native Java driver that converts JDBC (Java Database Connectivity) calls into the network protocol used by the MySQL database. It lets developers working with the Java programming language easily build programs and applets that interact with MySQL and connect all corporate data, even in a heterogeneous environment. MySQL Connector/J is a Type IV JDBC driver and has a complete JDBC feature set that supports the capabilities of MySQL.
5. Make changes in the server.xml and web.xml files in the application server to establish connectivity between tomcat and MySQL.

## 6. Test your web application.

The client can have any operating system such as Windows 95/98/NT/2000/XP workstation, Mac OS, OS/2, UNIX, Linux etc. The browsers that the client can use for viewing HTML documents and using RFMS are Netscape Navigator 3.0 or higher or Microsoft's Internet Explorer 3.x or higher.

## 2.5 Summary

The software design of the project was presented in Chapter Two and it the underlying structure of the application developed. RFMS was a 3-tier architecture based web application that provides interfaces in HTML for the clients. The data is retrieved from a MySQL database through JavaBeans and sent to the browsers by JSP pages.

## CHAPTER THREE

### SYSTEM VALIDATION

#### 3.1 Introduction

This chapter provides the procedures in which RFMS - Real-Time Finance Management System was tested and the results gathered from it. It was tested on Microsoft's Internet Explorer Browser.

#### 3.2 Unit Test Plan

RFMS - A Real-Time Finance Management System was a web-based application that helps users manage their finances by using the tools provided in the system. The users can input their financial information online and analyze it irrespective of where they are.

As part of the Unit Test Plan, RFMS was tested based on the user types since each user has its own set of interfaces. All the interfaces are web based and available through the browser.

As part of the unit test plan these steps were taken and results found:

1. All the hyperlinks available in each set of interfaces of RFMS were checked, irrespective of the content that they provide. It was found that all of them do work.

2. User data input was given to check if the presentation logic worked. The presentation logic was implemented in JavaScript, which checks if the input taken from the users was valid. The display was in HTML. The Interfaces of RFSM that required user input successfully validated it.
3. There are error pages that display the errors based on the functionality.
4. JavaScript displays the errors in the input of entries in alert boxes.
5. The basic rules of web based applications of readability and presentation for each set of Interfaces was verified. The data and content was readable and displayed in a user-friendly manner.

### 3.3 Integration Test Plan

As part of the Integration Test Plan, RFMS was tested for functionality based on its user-type. The users have their own interfaces and after being tested as single units and their presentation they were tested based on functionality.

The functions of each user are as follows:

A. Systems Administrator:

- Create user account

- Edit user account
- Delete user account
- View User Information

#### B. User

- Manage Portfolio
- Setup Reminders
- Debt Reduction Planning
- Account Information

Every set of functions is available on the left hand side of the browser as a menu.

The following tests were performed and results obtained:

1. RFMS's user interfaces were tested now for content across each page depending on the functionality.
2. Every function of each user was tested.
3. A user was created as part of the administrator pages. After the information was entered the functions of edition, deleting and final viewing of all information was done. RFMS's Systems Administrator pages tests passed successfully.

4. The user functionality was tested for bugs and limitations. Each function was checked for content and right response.
5. Certain limitations were found which have been documented and the bugs were fixed.
6. The portfolio management section was checked if it was able to get data over the internet from the inetat website. This passed successful.
7. The Reminder section was thoroughly tested to see if it was able to send outgoing email at the exact specified time, it was able to do it without any problem.
8. The different tools in the debt reduction planning were checked to see if all the formulas used in there were correct. The system was able to perform these tasks even with wrong input values.
9. The account information page was checked to see if the balance check book tool matches a real balance book.

### 3.4 System Test Plan

As part of the System Test Plan, the following tests were performed:

1. The JSP server and the MySQL server were started and the system start page was opened in the browser successfully.
2. Tested to see that unauthorized users cannot view pages. They cannot get past the login page.
3. Tested to see if XML queries are being processed properly, they were being processed as expected.
4. Tested to see if reminder email system was able to send out emails and found that it doing it without any problems.
5. The graphs for the expenditure analysis page needs data for two months to be able to display the comparison, it gives an error message if data for at least two months is not available.
6. Balance check book sections was tested to see if users modified an earlier entry then the system should be able to recalculate all the entries again. The test was passed.

## CHAPTER FOUR

### MAINTENANCE

#### 4.1 Introduction

This chapter provides the measures that need to be taken to maintain RFMS - A Real-Time Finance Management System. Maintenance needs to be done if:

1. Issues arise on the front-end of the application or the back-end programming.
2. Issues with the tools utilized for development.
3. Issues with the directory structure of the application.

#### 4.2 Maintenance Guidelines

These guidelines aim at providing the information for maintenance issues that might arise depending on the user-type and the front end of the application as well as depending on the programming and servers at the backend of the application.

##### 4.2.1 Interfaces Management

The programming has been done in JSP and Java and RFMS has been tested for all the functionality. But if issues arise then the Webmaster will have to know Java, JSP, JavaScript, XML and MySql to troubleshoot problems in the business logic of the application.



#### 4.2.2 Administration

A Systems Administrator/webmaster is needed to create, edit or delete accounts.

#### 4.3 Tools Utilized

RFMS was developed using the following tools:

1. Tomcat Web Server
2. MySQL Database Server
3. Java 2 Standard Development Kit
4. JDBC Driver
5. Macromedia Dreamweaver

#### 4.4 Directory Structure

RFMS used Apache's Jakarta Tomcat server as the JSP/Servlet container. The web application was developed in the "webapps" sub-directory of the Jakarta Tomcat main directory. It was stored in a directory under "webapps" called "rfms".

The files present at each directory level:

##### rfms directory

All the .jsp and .html files are saved here. The naming convention followed can be explained by an example of a create user admin page:

1. Presentation page: createUser.jsp.

2. JavaScript Code for user-input validation:

`createUserCheck.jsp`.

3. JSP servlet page for communication with the

JavaBeans that insert the data:

`createUserInsert.jsp`.

The common files used by all the users are placed directly under "rfms", but all the other pages are placed in sub-directories based on their user-type first and then their functionality. The sub-directories under "rfms" are:

1. admin - The administrator pages.

2. portfolio - The stock management pages.

3. reminder - The email reminder pages.

4. debt - The debt reduction planning pages.

5. expenditure - The expenditure analysis pages.

6. accountInfo - The account information pages.

4. web-inf - This directory has the .java files stored in it and the respective .class files are stored in another sub-directory package in it called rfms\_modules.

## CHAPTER FIVE

### PROJECT IMPLEMENTATION

#### 5.1 Introduction

This chapter provides details about the implementation of the project by describing some of the interfaces used in the project. The RFMS's front-end was tried to be made as user-friendly as possible. JavaScript was used on the client side to check for the validity of user's input. Error messages were reported appropriately wherever it was needed.

The Real-Time Finance Management System - RFMS was developed to help users control and plan their finances. Various aspects of finance are considered in the project. These include:

1. Portfolio Management
2. Reminder Setup
3. Debt Reduction Planning
4. Expenditure Analysis
5. Account Information

Apart from these pages there was a system login process which was required to authenticate users.

## 5.2 System Interfaces

### 5.2.1 Login

The login process authenticates users based on the username and password provided by them.

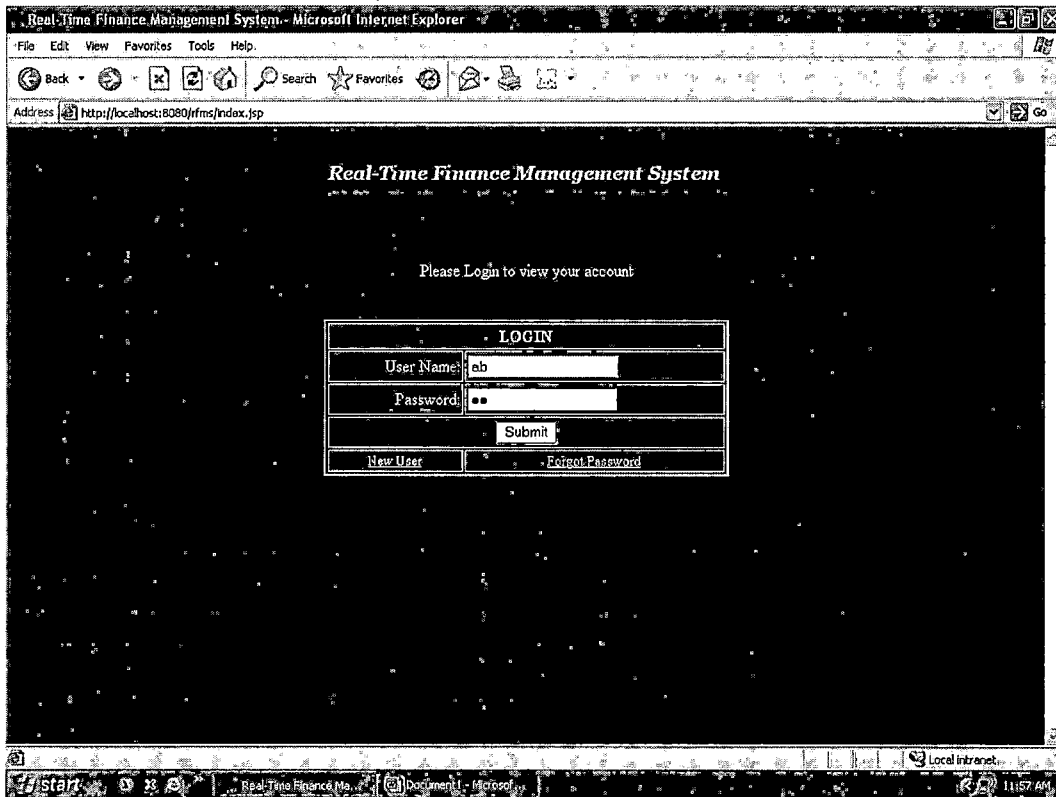


Figure 5. Login Page

### 5.2.2 Portfolio Management

This section had three tools present in it:

1. Get Quote
2. Get Top Trading List
3. Get Time and Sales Report for a Stock

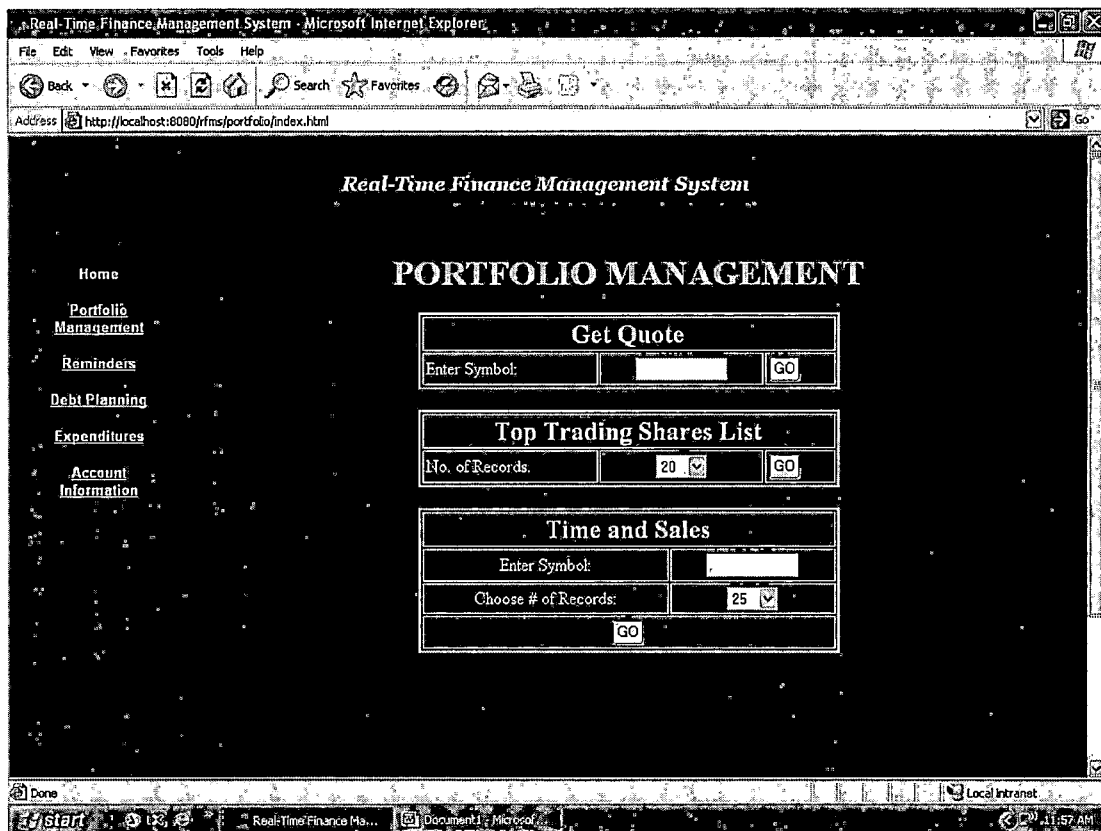


Figure 6. Portfolio Management Page

The three tools provided in this section gets updated Real-Time Market information when the markets are open.

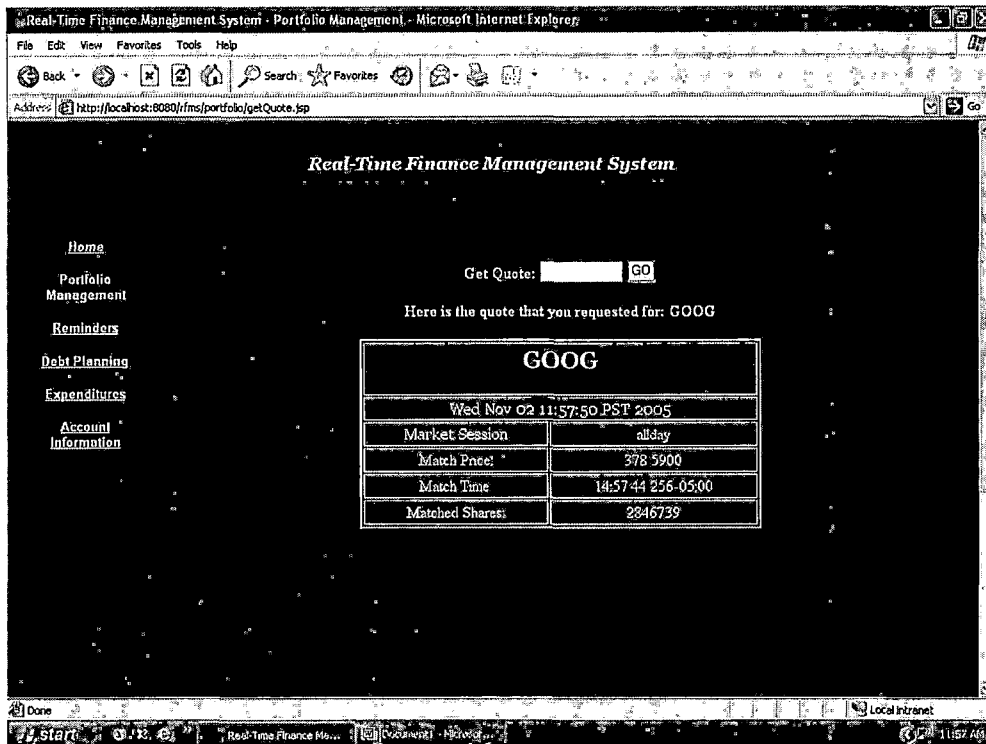


Figure 7. Get Stock Quote Page

The get quote section retrieves a quote and market information based on the symbol provided by the user.

Real-Time Finance Management System - Portfolio Management - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://localhost:8080/fms/portfolio/getTopList.jsp

Get Quote:  GO

### Top 20 List

RANK	SYMBOL	MARKET	PRICE	TIME	MATCHED SHARES
1	QQQQ	NM	32.2240	14:58:22.784-05:00	33836924
2	SUNW	NM	3.6910	14:58:28.377-05:00	28370925
3	SYMC	NM	12.1500	14:58:28.819-05:00	24254904
4	SPY	AM	121.2400	14:58:24.538-05:00	22177769
5	MSFT	NM	26.4000	14:58:28.222-05:00	20502338
6	SIRI	NM	6.8760	14:58:29.183-05:00	19976370
7	JDSU	NM	2.2010	14:58:29.320-05:00	16289272
8	CSCO	NM	17.5800	14:58:25.278-05:00	15000740
9	INTC	NM	23.1300	14:58:21.872-05:00	14174479
10	ORCL	NM	12.5000	14:58:18.488-05:00	10392451
11	AMAT	NM	16.3300	14:58:11.270-05:00	8391704
12	DELL	NM	29.0300	14:58:25.67-05:00	8069563
13	AAPL	NM	59.7000	14:58:29.681-05:00	7983171
14	MERQ		24.5000	14:58:26.926-05:00	6894778
15	IWM	AM	65.0400	14:58:22.888-05:00	6160217
16	SEBL	NM	10.4300	14:58:28.733-05:00	4999812
17	QCOM	NM	40.1700	14:58:23.567-05:00	4863533
18	CNXT	NM	2.1200	14:58:15.76-05:00	4765056
19	CMCSA	NM	28.5100	14:58:27.944-05:00	4734616
20	SMH	AM	34.1300	14:58:20.648-05:00	4631600

Done Local intranet 11:58 AM

Figure 8. Get Top Trading Stock List

This page gets the top 20 trading list by default. If needed, we can ask the system to get up to 500 top trading stocks at that time.

Real-Time Finance Management System - Portfolio Management - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Reload Search Favorites

Address http://localhost:8080/rfms/portfolio/getTimeAndSales.jsp

Get Quote:  GO

## Time and Sales

The last 25 Sales for GOOG

#	MATCHED SHARES	PRICE	TIME	MATCH TYPE	REFERENCE NUMBER
1.	100	378.2500	14:59:56.398-05:00	B	2121616
2.	308	378.1300	14:59:55.239-05:00	S	2121521
3.	34	378.1300	14:59:55.185-05:00	S	2121512
4.	66	378.1300	14:59:55.143-05:00	S	2121511
5.	100	378.1600	14:59:52.153-05:00	S	2120705
6.	100	378.2400	14:59:51.714-05:00	B	2120668
7.	350	378.1200	14:59:50.769-05:00	S	2120599
8.	50	378.1200	14:59:50.439-05:00	S	2120351
9.	100	378.1900	14:59:49.853-05:00	S	2120162
10.	400	378.5900	14:59:49.332-05:00	S	2120082
11.	198	378.3100	14:59:48.733-05:00	S	2120021
12.	100	378.3400	14:59:46.507-05:00	S	2119776
13.	100	378.3500	14:59:46.478-05:00	S	2119772
14.	2	378.3500	14:59:46.475-05:00	S	2119769
15.	50	378.3600	14:59:46.369-05:00	B	2119764
16.	10	378.2500	14:59:45.733-05:00	S	2119716
17.	100	378.2300	14:59:44.724-05:00	S	2119528

Home  
Portfolio Management  
Reminders  
Debt Planning  
Expenditures  
Account Information  
Expense Planner  
Bills & Dues

Done Local Internet 12:00 PM

Figure 9. Time and Sales Report

This page gets the last 25 matched deals for a particular stock. Again, 25 is just the default the maximum number of matched deals it can retrieve in one call is 100.

### 5.2.3 Reminder Setup

This section consists of setting up reminders for oneself. The system sends out an email at the particular day the reminder is set up to.



Real-Time Finance Management System - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Search Favorites

Address http://localhost:8080/rms/reminders/index.html Go

Real-Time Finance Management System

Home

Portfolio Management

Reminders

Debt Planning

Expenditures

Account Information

REMINDERS

Setup New Reminders

Event	
Date:	
Time:	
Description:	

Reminder 1		Reminder 2	
Date:		Date:	
Time:		Time:	
Email:		Email:	

Create Event Reminder

Start Real-Time Finance Man... Document1 - Microsoft... Local intranet 12:01 PM

Figure 10. Reminder Setup Page

Here the users were able to set reminders for themselves. The system used to allow them to set up to two reminders for each event.

#### 5.2.4 Debt Reduction Planning

This sections a loan calculator to figure out the details of any loan. You could get information like how much interest will you pay over the term of the loan or how much will your loan payments per month will be.



Figure 11. Debt Reduction Planning Page

Just by entering a few details such as loan amount, the term and the interest rate users could get valuable information such as the monthly payments and the total interest they will be paying over the life of the loan.

#### 5.2.5 Expenditure Analysis

This is the section where the users enter their monthly expenses every month and then they can use this information to see graphs and charts on their expenses so that they can better analyze their budget for the upcoming months.

Real-Time Finance Management System - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://localhost:8080/rms/expenditures/index.html>

**Real-Time Finance Management System**

**EXPENDITURE ANALYSIS**

Entering budget information for each month will help you analyze your spending

**Enter your Expenditures**

1. Housing (Mortgage/Rent)	<input type="text"/>
2. Utilities (Water, Gas, Electric, etc.)	<input type="text"/>
3. Auto Loans	<input type="text"/>
4. Auto Insurance	<input type="text"/>
5. Credit Card Payments	<input type="text"/>
6. Miscellaneous	<input type="text"/>
7. Other	<input type="text"/>

**Income**

Enter your total monthly income from all sources

**Submit**

Figure 12. Expenditure Analysis Page

Here the users enter their monthly expenses based on the categories provided. The users also had the ability to add new categories that are specific to them.

#### 5.2.6 Account Information

This section has a balance check book tool. This facilitates the users to balance their check book from anywhere. Calculations are done for the user by the system. All the user has to do is to enter the transactions.

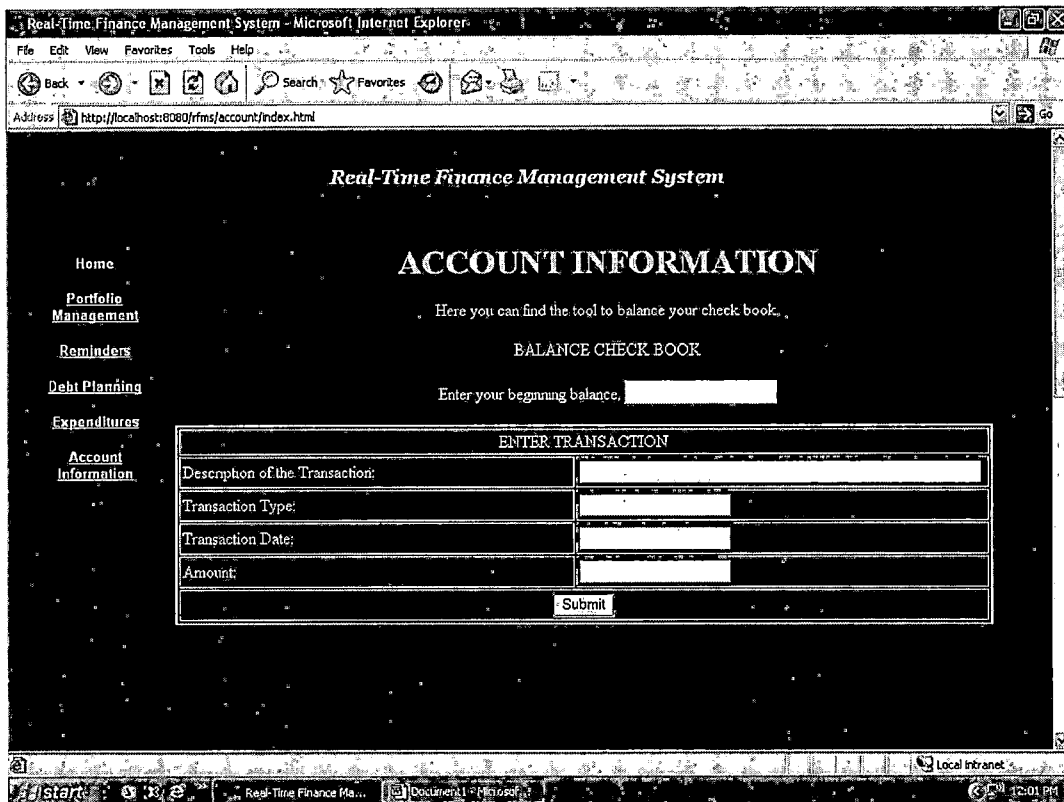


Figure 13. Account Information Page

This is the page where the users enter their transactions. Based on the transaction type the system determines if it was a debit/credit transaction and calculations are done on that basis.

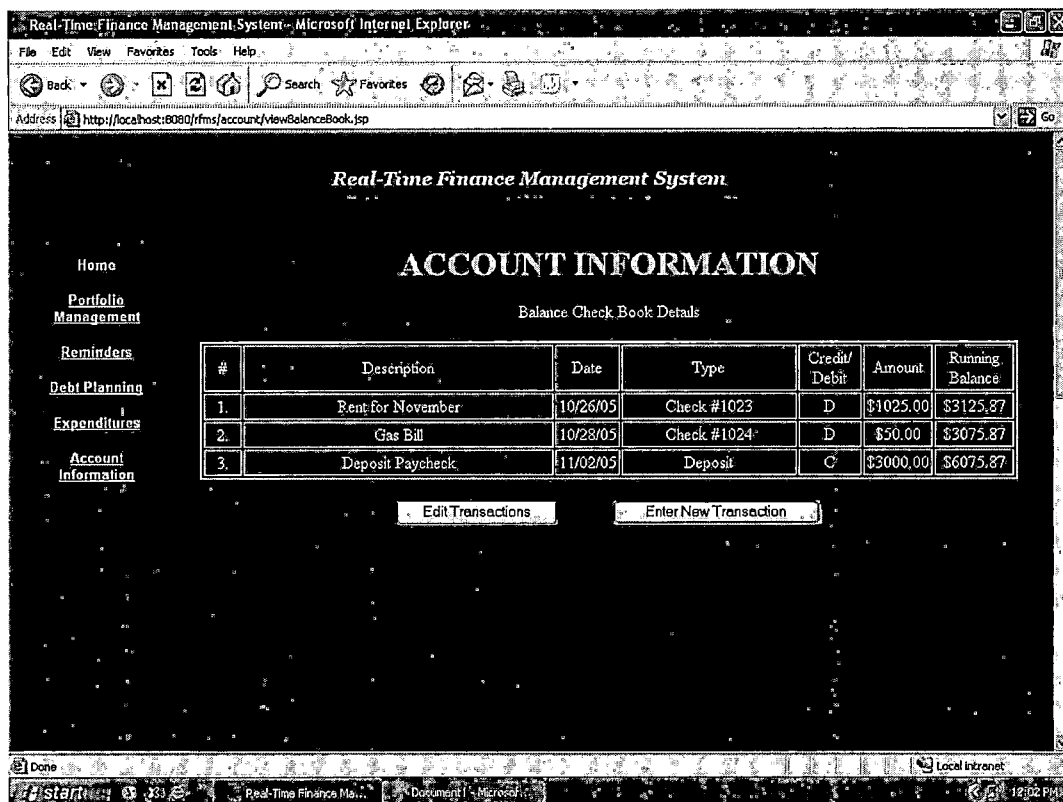


Figure 14. Balance Check Book Page

This page displays the users balance book. It also gives the ability to edit a previous transaction or to enter new transactions.

## CHAPTER SIX

### CONCLUSIONS AND FUTURE DIRECTIONS

#### 6.1 Conclusions

RFMS - A Real-Time Finance Management System was a personal finance management system which provides real-time market data from stock exchanges. When the stock exchange is open the data is delayed by less than 15 minutes. Users can use RFMS to learn more about stocks in general and learn to trade by using virtual money.

RFMS also provides users with other tools such as reminder system which sends email to remind you about events, a few loan calculators, a budgeting system that graphs the user's expenses and a balance check book tool.

Passing XML data over HTTP is the latest in EDI (Electronic Data Interchange) standards. Java Combined with XML provides developers with powerful tools to develop internet based applications.

The real-time stock data is retrieved using REST architecture. REST - Representational State Transfer is a model for web services based solely on HTTP. REST takes the view that the Web already has everything necessary for web services, without having to add extra specifications like SOAP and UDDI. RFMS used Java and XML to get this

data using REST architecture. Part of this data was stored in a database. MySql database was used to store information on user's calculations so that they don't have to enter it the next time they use the system.

## 6.2 Future Directions

Personal Software Management Software is used by almost everyone these days to keep track of their finances. A typical person could have multiple bank accounts, multiple credit cards, a few investment accounts, and lots of bills. So this field is here to stay and many more tools will be required by people in the future.

In future not only stock information but also bank, credit card and bills information can be transferred over the internet using the REST architecture. This will help users to transport their data from one system to another without any problems.

Technology usage wise, Servlets or Java Struts were not used in RFMS, but if the application is expanded to include more tools then using those would benefit a lot. Also trying to utilize XML to the fullest will make development and/or maintenance easy to manage.

Tools: the more the better. There is no limit on the number of different tools that could be provided for the users. Dealing with variable loans, mortgages and creating amortization tables for those will help users see what part of their money is actually going towards the principal.

Taxes: One more limitless field. Tools could be provided which keeps track of user's expenses and at the end of the year prepares a statement for the user which would help the users file their taxes.

Finally and once again, use XML to send data back and forth between applications. Using XML cannot be stressed enough; XML makes the task of transferring data from one application to another a breeze.



APPENDIX  
SOURCE CODE OF JAVA CLASSES

Here is the source code from some of the files used in Real-Time Finance Management System.

**database.java**

```
//java document
package rfms_modules;
import java.sql.*;
public class database
{
    private static String url="jdbc:mysql://localhost/rfms";
    private static String driver="com.mysql.jdbc.Driver";
    private static String user="mac";
    private static String password="rfms";

    public static Connection getConnection() throws Exception
    {
        Class.forName(driver).newInstance();
        //System.out.println("driver Loaded");
        Connection connection = DriverManager.getConnection(url,user,password);
        //System.out.println("connection established");
        return connection;
    }
}
```

### **DateBean.java**

```
// Java Document
package rfms_modules;

import java.io.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
import java.util.*;

public class DateBean {

    public static void main(String[] args) {
        //System.out.println(getSingleQuote("MSFT"));
    }
    public static Date getDate(){
        return(new java.util.Date());
    }
    public static String getSingleQuote(String symbol) {
        String name = "";
        return (name);
    }
}
```

## **StockInfo.java**

```
// Java Document for qoute.xml
package rfms_modules;

import java.io.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
import java.util.*;
import org.apache.xerces.parsers.*;

public class StockInfo {

    public static String marketSession = "";
    public static String matchedShares = "";
    public static String matchPrice = "";
    public static String matchTime= "";

    public StockInfo() {
        marketSession = "";
        matchedShares = "";
        matchPrice = "";
        matchTime= "";
    }

    public static String getMarketSession() {
        return marketSession;
    }

    public static String getMatchedShares() {
        return matchedShares;
    }

    public static String getMatchPrice() {
        return matchPrice;
    }

}
```

```

    public static String getMatchTime() {
        return matchTime;
    }

    public static void setMarketSession(String ms) {
        marketSession = ms;
    }

    public static void setMatchedShares(String ms2) {
        matchedShares = ms2;
    }

    public static void setMatchPrice(String mp) {
        matchPrice=mp;
    }

    public static void setMatchTime(String mt) {
        matchTime = mt;
    }

    public static void main(String[] args) {
        System.out.println(getSingleQuote("ORCL"));
        System.out.println(""+getMatchPrice());
        System.out.println("Date: "+getDate());
    }

    public static Date getDate(){
        return(new java.util.Date());
    }

    public static String getSingleQuote(String symbol) {
        String name = "asdf1111";

        try {
            String x =
"http://xml.island.com/ws/xml/quote.xml?token=TXHE3PCsMdCGBWd7&sy
mbol="+symbol;
            Document d = new SAXBuilder().build(x);

```

```

        Element e = new Element("islandData");
        List children = d.getRootElement().getChildren();
        //for (Iterator i=children.iterator();i.hasNext();) {
        for (int i=0, size=children.size(); i<size;i++) {
            //Object child = i.next();
            Element child = (Element)children.get(i);

/*      if (child instanceof Element){
            //name +=child.toString();
            Element e2 = (Element) child;
            name += e2.getName();
        }
    */
        name +=child.getName();
        name += "<br>";
    }
    Element temp = d.getRootElement().getChild("stock").getChild("matched");
    setMarketSession(temp.getAttribute("marketSession")
    .getValue());
    setMatchedShares(temp.getChild("matchedShares")
    .getValue());
    setMatchPrice(temp.getChild("lastMatch")
    .getChild("matchPrice").getValue());
    setMatchTime(temp.getChild("lastMatch")
    .getChild("matchTime").getValue());
}

catch (Exception e) {
    e.printStackTrace();
    return("There was a exception"+e);
}

return (name);
}
}

```

**TimeAndSales.java**

```

// Java Document for timeandsales.xml
package rfms_modules;

import java.io.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
import java.util.*;
import org.apache.xerces.parsers.*;

public class TimeAndSales {

    public static void main(String[] args) {
        Document d1 = buildDocument("MSFT",20);
        //displayTopList(d1);
    } // end main

    public static Date getDate() {
        return(new java.util.Date());
    } // end getDate

    public static Document buildDocument(String symbol, int recNumb) {

        if (!(recNumb>0)) { recNumb = 20; }
        try {
            String timeAndSales =
"http://xml.island.com/ws/xml/timeandsales.xml?token=TXHE3PCsMdCGB
Wd7&symbol="+symbol+"&recNumb="+recNumb;
            Document doc = new SAXBuilder().build(timeAndSales);
            return (doc);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```
        String str = "There was an exception"+e;
        Element error = new Element("ERROR");
        error.addContent(str);
        Document d = new Document(error);
        return(d);
    }
} // end buildDocument
} // end class TimeAndSales
```



## **TopList.java**

```
// Java Document for toplist.xml
package rfms_modules;

import java.io.*;
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
import java.util.*;
import org.apache.xerces.parsers.*;

public class TopList {

    //this is for test purposes only
    public static void main(String[] args) {
        Document d1 = buildDocument(20);
        displayTopList(d1);
    } // end main

    public static Date getDate() {
        return(new java.util.Date());
    } // end getDate

    public static Document buildDocument(int recNumb) {

        if (recNumb==0) { recNumb = 20; }
        try {
            String topListUrl =
"http://xml.island.com/ws/xml/toplist.xml?token=TXHE3PCsMdCGBWd7&r
ecNumb="+recNumb;

            Document doc = new SAXBuilder().build(topListUrl);
            return (doc);
        }
    }
}
```

```

        catch (Exception e) {e.printStackTrace();
            String str = "There was an exception"+e;
            Element error = new Element("ERROR");
            error.addContent(str);
            Document d = new Document(error);
            return(d);
        }
    } // end buildDocument

    public static void displayTopList(Document d) {
        System.out.println("\n\n\n\n\n");
        //System.out.println(d.getRootElement());
        Element stockList = d.getRootElement().getChild("stockList");
        List stocks = stockList.getChildren();
        for (int i=0, size=stocks.size(); i<size;i++) {
            Element stock = (Element)stocks.get(i);
            System.out.print(stock.getAttribute("rank").getValue());
            System.out.println(stock.getAttribute("symbol").getValue());
            System.out.println(stock.getChild("market").getValue());
            System.out.println(stock.getChild("booked").getChild("bookedShares").getValue());

        }

    } // end displayTopList
} // end class TopList

```

## loginVerify.jsp

```
<%@ page contentType="text/html; charset=iso-8859-1" language="java"
import="java.sql.*" errorPage="" %>
<%@ page import="rfms_modules.*"%>
<%@ page session="true"%>
<%
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    boolean foundRecord = false;
    boolean matched = false;

    try {
        Connection con = database.getConnection();
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("select password,role from user where
username='"+username+"'");

        foundRecord = rs.first();

        if (password.equals(rs.getString("password"))) {
            matched = true;
        } else {
            matched = false;
        }

        if (foundRecord&&matched) {

session.setAttribute("username",username);

                if (rs.getString("role").equals("admin")) {
                    response.sendRedirect("admin.jsp");
                }
                else {
                    response.sendRedirect("user.jsp");
                }
            }
        }
    }
}
```

```
    }  
    else {  
  
        response.sendRedirect("index.jsp?login=0");  
    }  
}  
catch(SQLException e){  
    response.sendRedirect("index.jsp?login=0");  
}  
  
%>
```

## getTopList.jsp

```
<%@ page import="rfms_modules.*"%>
<%@ page import="org.jdom.*, org.jdom.input.*, org.apache.xerces.parsers.*,
java.sql.*, java.util.List"%>
<%@ page language="java" contentType="text/html"%>
<%@ page autoFlush = "true"%>
<%@ page session = "false"%>
<jsp:useBean id="tl" scope="page" class="rfms_modules.TopList"/>

<html>
<head>
<title>Real-Time Finance Management System - Portfolio Management</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body vLink=#ffe4ca aLink=#ffffff link=#ffffff
background="../images/bluebgound.gif" text="#FFFFFF">
<TABLE width=500 border=0 align="center" cellPadding=0 cellSpacing=0>
  <TBODY>
    <TR>
      <TD height="74" align="middle" vAlign="center"> <div align="center"><font
color="#0066FF" size="4" face="Georgia, Times New Roman, Times,
serif"><strong><em><font color="#FFCCFF">Real-Time
      Finance Management System</font><br>
      </em></strong></font>
      <hr width="400" color="#FF0000">
      <font color="#0066FF" size="4" face="Georgia, Times New Roman,
Times, serif"><strong><em>
      </em></strong></font></div></TD>
    </TR>
  </TBODY>
</TABLE>
<br>
```

```

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

        <td width="14%">
<table align="left" cellpadding="8" cellspacing="2" width="100%">
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a
href="../index.html">Home</a></font></strong></font></div></td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif">Portfolio<br>
        Management</font></strong></font></div></td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a
href="reminders/index.html">Reminders</a></font></strong></font></div>
    </td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a href="debt/index.html">Debt
        Planning</a></font></strong></font></div></td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a
href="expenditures/index.html">Expenditures</a></font></strong></font><
    /div></td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a href="account/index.html">Account

```

```

        Information</a></font></strong></font></div></td>
    </tr>
    <tr>
        <td>
            <p>&nbsp;</p>
        </td>
    </tr>
</table>

<p>&nbsp;</p></td>
        <td width="86%"><table width="100%"><tr><td>
            <p align="center">
                <form name="form1" method="post" action="getQuote.jsp">
                    <p align="right"><font color="#FFFFFF"><strong><font size="-1">Get
                        Quote: </font></strong></font> <font size="-1">
                            <input name="stockName" type="text" id="stockName" size="10">
                                <input type="submit" name="Submit" value="GO">
                                    </font> </p>
                                </form>
                            <div align="center">
                                <p><strong><font color="#FF6666" size="6">Top
                                    <%out.print(" ");
                                        out.print(request.getParameter("recNumb"));
                                        out.print(" ");%>
                                    List</font></strong></p>
                                <table width="100%" border="2" align="center" cellpadding="2"
cellspacing="2">
                                    <tr>
                                        <td><div align="center"><strong><font size="-1"> RANK</font>
</strong></div></td>
                                        <td><div align="center"><strong><font size="-1">
SYMBOL</font></strong></div></td>
                                        <td><div align="center"><strong><font size="-1">
MARKET</font></strong></div></td>

```

```

        <td><div align="center"><strong><font size="-1">
PRICE</font></strong></div></td>
        <td><div align="center"><strong><font size="-1">
TIME</font></strong></div></td>
        <td><div align="center"><strong><font size="-1">MATCHED SHARES
</font></strong></div></td>
                </tr>
<%
Document d = tl.buildDocument(Integer.parseInt(request.getParameter("recNumb")));
        Element root = d.getRootElement();
        Element stockList = root.getChild("stockList");
        List stocks = stockList.getChildren();
        for (int i=0, size=stocks.size(); i<size;i++) {
                Element stock = (Element)stocks.get(i); %>
<tr>
        <td><div align="center"><font size="-1">
<%out.print(stock.getAttribute("rank").getValue());%></font></div></td>
        <td><div align="center"><font size="-1">
<%out.print(stock.getAttribute("symbol").getValue());%></font></div></td>
        <td><div align="center"><font size="-
1"><%out.print(stock.getChild("market").getValue());%></font></div></td>
        <td><div align="center"><font size="-
1"><%out.print(stock.getChild("matched").getChild("lastMatch").getChild("mat
chPrice").getValue());%></font></div></td>
        <td><div align="center"><font size="-
1"><%out.print(stock.getChild("matched").getChild("lastMatch").getChild("mat
chTime").getValue());%></font></div></td>
        <td><div align="center"><font size="-1">
<%out.print(stock.getChild("matched").getChild("matchedShares").getValue());
%></font></div></td>
</tr>
<%
        }
%>

</table>

```



```

        <p>&nbsp;</p>
    </div>
    <p align="center"></form> </p>
    <p align="center">&nbsp;</p>
</td>
</tr></table>
</td></tr>
</table>

</body>
</html>

```

```
// end getTopList.java
```

### **getQuote.jsp**

```

<%@ page import="rfms_modules.*"%>
<%@ page import="org.jdom.input.*"%>
<%@ page import="java.sql.*"%>
<%@ page language="java" contentType="text/html"%>
<%@ page autoFlush = "true"%>
<%@ page session = "false"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Real-Time Finance Management System - Portfolio Management</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body vLink=#ffe4ca aLink=#ffffff link=#ffffff
background="../images/bluebgound.gif">
<TABLE width=500 border=0 align="center" cellPadding=0 cellSpacing=0>
  <TBODY>
    <TR>

```

```

        <TD height="74" align="middle" vAlign="center"> <div align="center"><font
color="#0066FF" size="4" face="Georgia, Times New Roman, Times,
serif"><strong><em><font color="#FFCCFF">Real-Time
        Finance Management System</font><br>
        </em></strong></font>
        <hr width="400" color="#FF0000">
        <font color="#0066FF" size="4" face="Georgia, Times New Roman, Times,
serif"><strong><em>
        </em></strong></font></div></TD>
    </TR>
</TBODY>
</TABLE>
<br>
<table align="center" width="100%">
    <tr>

        <td width="14%">
<table align="left" cellpadding="8" cellspacing="2" width="100%">
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a
href=" ../index.html">Home</a></font></strong></font></div></td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif">Portfolio<br>
        Management</font></strong></font></div></td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a
href="reminders/index.html">Reminders</a></font></strong></font></div></td>
    </tr>
    <tr>

```

```

        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a href="debt/index.html">Debt
        Planning</a></font></strong></font></div></td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a
href="expenditures/index.html">Expenditures</a></font></strong></font></div></
td>
    </tr>
    <tr>
        <td><div align="center"><font color="#FFFFFF"><strong><font size="-1"
face="Arial, Helvetica, sans-serif"><a href="account/index.html">Account
        Information</a></font></strong></font></div></td>
    </tr>
</table>

```

```

<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

```

```

        <td width="86%"><table width="100%"><tr><td>
        <p align="center">
        <form name="form1" method="post" action="getQuote.jsp">
        <p align="center"><font color="#FFFFFF"><strong>Get Quote:
</strong></font>
        <input name="stockName" type="text" id="stockName" size="10">
        <input type="submit" name="Submit" value="GO">

```

```

    </p>
</form> <p align="center"><font color="#FFFFFF"><strong>Here is the
quote that you requested for:
    <% out.print(request.getParameter("stockName"));;%>
</strong></font></p>
<p align="center"><font color="#FFFFFF">
    <jsp:useBean id="date" class="DateBean">
</jsp:useBean>
    <jsp:useBean id="si" class="StockInfo">
</jsp:useBean>
    </font> </p>
<p align="center"><font color="#FFFFFF">
    <%
        //StockInfo si = new StockInfo();
        String stockQuote =
si.getSingleQuote(request.getParameter("stockName"));
        //out.println("My Name is Muku");
        //out.println(stockQuote);
        //out.println("<br>");
        //out.println(date.getDate());
        %>
    </font></code> </p>
<div align="center">
    <table width="50%" border="2" align="center" cellpadding="2"
cellspacing="2">
        <tr>
            <td colspan="2"><div align="center">
                <h2> <font color="#66FFFF" face="Georgia, Times New Roman, Times,
serif">
                    <%out.print(request.getParameter("stockName"));;%>
                </font></h2>
            </div></td>
        </tr>
    </table>

```

```

        <td colspan="2"><div align="center"><font color="#66FFFF"
face="Georgia, Times New Roman, Times, serif">
        <%out.print(date.getDate());%>
        </font></div>
        <div align="center"></div></td>
</tr>
<tr>
        <td><div align="center"><font color="#66FFFF" face="Georgia, Times New
Roman, Times, serif">Market
        Session </font></div></td>
        <td><div align="center"><font color="#66FFFF">
        <%out.print(si.getMarketSession());%>
        </font></div></td>
</tr>
<tr>
        <td><div align="center"><font color="#66FFFF">Match
Price:</font></div></td>
        <td><div align="center"><font color="#66FFFF">
        <%out.print("'" + si.getMatchPrice());%>
        </font></div></td>
</tr>
<tr>
        <td><div align="center"><font color="#66FFFF">Match Time:
</font></div></td>
        <td><div align="center"><font color="#66FFFF">
        <%out.print(si.getMatchTime());%>
        </font></div></td>
</tr>
<tr>
        <td><div align="center"><font color="#66FFFF">Matched Shares:
        </font></div></td>
        <td><div align="center"><font color="#66FFFF">
        <%out.print(si.getMatchedShares());%>
        </font></div></td>
</tr>

```

```

        </table>
    </div>
    <p align="center">&nbsp;</p>
    <p align="center"></form> </p>
    </td>
</tr></table>

```

```

        </td>
    </tr></table>

```

```

<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

```

```

<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
</body>
</html>

```

```

    // end getQuote.jsp

```

### **getTimeAndSales.jsp**

```

<%@ page import="rfms_modules.*"%>
<%@ page import= "org.jdom.*, org.jdom.input.*,
org.apache.xerces.parsers.*, java.sql.*,java.util.List"%>
<%@ page language="java" contentType="text/html"%>
<%@ page autoFlush = "true"%>
<%@ page session = "false"%>
    <jsp:useBean id="ts" scope="page"
class="rfms_modules.TimeAndSales"/>

```

```

<html>

```

```

<head>
<title>Real-Time Finance Management System - Portfolio
Management</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
</head>

<body vLink=#ffe4ca aLink=#ffffff link=#ffffff
background="../images/bluebgound.gif" text="#FFFFFF">
<TABLE width=500 border=0 align="center" cellpadding=0 cellspacing=0>
  <TBODY>
    <TR>
      <TD height="74" align="middle" vAlign="center"> <div
align="center"><font color="#0066FF" size="4" face="Georgia, Times
New Roman, Times, serif"><strong><em><font color="#FFCCFF">Real-Time
Finance Management System</font><br>
</em></strong></font>
<hr width="400" color="#FF0000">
<font color="#0066FF" size="4" face="Georgia, Times New
Roman, Times, serif"><strong><em>
</em></strong></font></div></TD>
    </TR>
  </TBODY>
</TABLE>
<br>
<table align="center" width="100%">
  <tr>

    <td width="14%">
<table align="left" cellpadding="8" cellspacing="2" width="100%">
  <tr>
    <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif"><a
href="../index.html">Home</a></font></strong></font></div></td>
  </tr>
  <tr>
    <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif">Portfolio<br>
Management</font></strong></font></div></td>
  </tr>
  <tr>
    <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif"><a
href="reminders/index.html">Reminders</a></font></strong></font></div>
</td>
  </tr>
  <tr>
    <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif"><a
href="debt/index.html">Debt
Planning</a></font></strong></font></div></td>
  </tr>
  <tr>
    <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif"><a
href="expenditures/index.html">Expenditures</a></font></strong></font>
</div></td>
  </tr>

```

```

</tr>
<tr>
  <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif"><a
href="account/index.html">Account
  Information</a></font></strong></font></div></td>
</tr>
<tr>
  <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif"><a
href="expense/index.html">Expense
  Planner</a></font></strong></font></div></td>
</tr>
<tr>
  <td><div align="center"><font color="#FFFFFF"><strong><font
size="-1" face="Arial, Helvetica, sans-serif"><a
href="bills/index.html">Bills
  & Dues</a></font></strong></font></div></td>
</tr>
</table>

<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<td width="86%"><table width="100%"><tr><td>
  <p align="center">
    <form name="form1" method="post" action="getQuote.jsp">
      <p align="right"><font color="#FFFFFF"><strong><font
size="-1">Get
        Quote: </font></strong></font> <font size="-1">
        <input name="stockName" type="text" id="stockName"
size="10">
        <input type="submit" name="Submit" value="GO">
      </font> </p>
    </form>
    <div align="center">
      <p><strong><font color="#FF6666" size="6">Time and
Sales</font><br><br>
        <font color="FF6666" size="4">The last
        <%out.print("  " );
          String recNumbString=
request.getParameter("recNumb");
          int recNumbInt=
Integer.parseInt(recNumbString);
          String symbol =
request.getParameter("symbol");
          out.print(recNumbString);
          out.print("  " );

```



```

        if (recNumbInt>1) {out.print("Sales");} else
{out.print("Sale");}
        out.print(" for "+symbol);
        %>
        </font></strong></p>
        <table width="100%" border="2" align="center"
cellpadding="2" cellspacing="2">
        <tr>
        <td><div align="center"><strong><font size="-
1">#</font></strong></div></td>
        <td><div align="center"><strong><font size="-
1">MATCHED<br>SHARES</font></strong></div></td>
        <td><div align="center"><strong><font size="-
1">PRICE</font></strong></div></td>
        <td><div align="center"><strong><font size="-
1">TIME</font></strong></div></td>
        <td><div align="center"><strong><font size="-
1">MATCH<br>TYPE</font></strong></div></td>
        <td><div align="center"><strong><font
size="-1">REFERENCE<br>NUMBER</font></strong></div></td>
        </tr>

        <%
        Document d =
ts.buildDocument(symbol,recNumbInt);
        Element root = d.getRootElement();
        Element matchList =
root.getChild("stock").getChild("matched").getChild("matchList");
        List matches = matchList.getChildren();
        for (int i=0, size=matches.size();

i<size;i++) {
        Element match =
(Element)matches.get(i); %>
        <tr>
        <td><div align="center"><font size="-
1"><%out.print(i+1);%>.</font></div></td>
        <td><div align="center"><font size="-
1"><%out.print(match.getChild("matchedShares").getValue());%></font><
/div></td>
        <td><div align="center"><font size="-
1"><%out.print(match.getChild("matchPrice").getValue());%></font></di
v></td>
        <td><div align="center"><font size="-
1"><%out.print(match.getChild("matchTime").getValue());%></font></div
></td>
        <td><div align="center"><font size="-
1"><%out.print(match.getChild("matchType").getValue());%></font></div
></td>
        <td><div align="center"><font size="-
1"><%out.print(match.getAttribute("refNumb").getValue());%></font></d
iv></td>
        </tr>

        <%      }      %>

        </table>
        <p>&nbsp;</p>

```

```

        </div>
        <p align="center">&nbsp;</p>
        <p align="center"></form> </p>
        </td>
    </tr></table>

```

```

    </td>
</tr></table>

```

```

<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>

```

```

<p>&nbsp;</p>
<p>&nbsp;</p>
</body>
</html>
// end getTimeAndSales.jsp

```

#### **web.xml**

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<!--

```

Copyright 2004 The Apache Software Foundation

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.

-->

```

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd"
  version="2.4">

```

```

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

```

```
<!-- JSPC servlet mappings end -->
```

```
<resource-ref>
  <description>
    Resource reference to a factory for java.sql.Connection
    instances that may be used for talking to a particular
    database that is configured in the server.xml file.
  </description>
  <res-ref-name>
    jdbc/rfms
  </res-ref-name>
  <res-type>
    javax.sql.DataSource
  </res-type>
  <res-auth>
    Container
  </res-auth>
</resource-ref>
```

```
</web-app>
```

```
// end web.xml
```

### **server.xml**

```
<!-- Example Server Configuration File -->
<!-- Note that component elements are nested corresponding to their
parent-child relationships with each other -->
```

```
<!-- A "Server" is a singleton element that represents the entire JVM,
which may contain one or more "Service" instances. The Server
listens for a shutdown command on the indicated port.
```

```
    Note: A "Server" is not itself a "Container", so you may not
    define subcomponents such as "Valves" or "Loggers" at this level.
-->
```

```
<Server port="8005" shutdown="SHUTDOWN">
```

```
  <!-- Comment these entries out to disable JMX MBeans support used for the
  administration web application -->
  <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <Listener className="org.apache.catalina.storeconfig.StoreConfigLifecycleListener" />
```

```
<!-- Global JNDI resources -->
```

<GlobalNamingResources>

<!-- Test entry for demonstration purposes -->

<Environment name="simpleValue" type="java.lang.Integer" value="30"/>

<!-- Editable user database that can also be used by  
    UserDataBaseRealm to authenticate users -->

<Resource name="UserDatabase" auth="Container"  
    type="org.apache.catalina.UserDatabase"  
    description="User database that can be updated and saved"  
    factory="org.apache.catalina.users.MemoryUserDatabaseFactory"  
    pathname="conf/tomcat-users.xml" />

</GlobalNamingResources>

<!-- A "Service" is a collection of one or more "Connectors" that share  
    a single "Container" (and therefore the web applications visible  
    within that Container). Normally, that Container is an "Engine",  
    but this is not required.

Note: A "Service" is not itself a "Container", so you may not  
define subcomponents such as "Valves" or "Loggers" at this level.

-->

<!-- Define the Tomcat Stand-Alone Service -->

<Service name="Catalina">

<!-- A "Connector" represents an endpoint by which requests are received  
    and responses are returned. Each Connector passes requests on to the  
    associated "Container" (normally an Engine) for processing.

By default, a non-SSL HTTP/1.1 Connector is established on port 8080.  
You can also enable an SSL HTTP/1.1 Connector on port 8443 by  
following the instructions below and uncommenting the second Connector  
entry. SSL support requires the following steps (see the SSL Config  
HOWTO in the Tomcat 5 documentation bundle for more detailed  
instructions):

\* If your JDK version 1.3 or prior, download and install JSSE 1.0.2 or  
    later, and put the JAR files into "\$JAVA\_HOME/jre/lib/ext".

\* Execute:

    %JAVA\_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA (Windows)

    \$JAVA\_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA (Unix)

    with a password value of "changeit" for both the certificate and  
    the keystore itself.

By default, DNS lookups are enabled when a web application calls  
request.getRemoteHost(). This can have an adverse impact on  
performance, so you can disable it by setting the  
"enableLookups" attribute to "false". When DNS lookups are disabled,  
request.getRemoteHost() will return the String version of the  
IP address of the remote client.

-->

```

<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector="8080"          maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true" />
<!-- Note : To disable connection timeouts, set connectionTimeout value
to 0 -->

```

<!-- Note : To use gzip compression you could set the following properties :

```

        compression="on"
        compressionMinSize="2048"
        noCompressionUserAgents="gozilla, traviata"
        compressableMimeType="text/html,text/xml"

-->

```

```

<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->
<!--
<Connector port="8443"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" />
-->

```

```

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009"
    enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />

```

```

<!-- Define a Proxied HTTP/1.1 Connector on port 8082 -->
<!-- See proxy documentation for more information about using this. -->
<!--
<Connector port="8082"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" acceptCount="100" connectionTimeout="20000"
    proxyPort="80" disableUploadTimeout="true" />
-->

```

<!-- An Engine represents the entry point (within Catalina) that processes every request. The Engine implementation for Tomcat stand alone analyzes the HTTP headers included with the request, and passes them on to the appropriate Host (virtual host). -->

```

<!-- You should set jvmRoute to support load-balancing via AJP ie :
<Engine name="Standalone" defaultHost="localhost" jvmRoute="jvm1">
-->

```

```

<!-- Define the top level container in our container hierarchy -->
/<!--<Engine name="Catalina" defaultHost="localhost">-->
    <Engine name="Catalina" defaultHost="localhost">
    <!-- The request dumper valve dumps useful debugging information about
    the request headers and cookies that were received, and the response

```

headers and cookies that were sent, for all requests received by this instance of Tomcat. If you care only about requests to a particular virtual host, or a particular application, nest this element inside the corresponding <Host> or <Context> entry instead.

For a similar mechanism that is portable to all Servlet 2.4 containers, check out the "RequestDumperFilter" Filter in the example application (the source for this filter may be found in "\$CATALINA\_HOME/webapps/examples/WEB-INF/classes/filters").

Request dumping is disabled by default. Uncomment the following element to enable it. -->

```
<!--
<Valve className="org.apache.catalina.valves.RequestDumperValve"/>
-->

<!-- Because this Realm is here, an instance will be shared globally -->

<!-- This Realm uses the UserDatabase configured in the global JNDI
resources under the key "UserDatabase". Any edits
that are performed against this UserDatabase are immediately
available for use by the Realm. -->
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>

<!-- Comment out the old realm but leave here for now in case we
need to go back quickly -->
<!--
<Realm className="org.apache.catalina.realm.MemoryRealm" />
-->

<!-- Replace the above Realm with one of the following to get a Realm
stored in a database and accessed via JDBC -->

<!--
<Realm className="org.apache.catalina.realm.JDBCRealm"
driverName="org.gjt.mm.mysql.Driver"
connectionURL="jdbc:mysql://localhost/authority"
connectionName="test" connectionPassword="test"
userTable="users" userNameCol="user_name" userCredCol="user_pass"
userRoleTable="user_roles" roleNameCol="role_name" />
-->

<!--
<Realm className="org.apache.catalina.realm.JDBCRealm"
driverName="oracle.jdbc.driver.OracleDriver"
connectionURL="jdbc:oracle:thin:@ntserver:1521:ORCL"
connectionName="scott" connectionPassword="tiger"
userTable="users" userNameCol="user_name" userCredCol="user_pass"
userRoleTable="user_roles" roleNameCol="role_name" />
-->
```

```

<!--
<Realm className="org.apache.catalina.realm.JDBCRealm"
    driverName="sun.jdbc.odbc.JdbcOdbcDriver"
    connectionURL="jdbc:odbc:CATALINA"
    userTable="users" userNameCol="user_name" userCredCol="user_pass"
    userRoleTable="user_roles" roleNameCol="role_name" />
-->

```

```

<!-- Define the default virtual host
    Note: XML Schema validation will not work with Xerces 2.2.
-->

```

```

<Host name="localhost" appBase="webapps"
    unpackWARs="true" autoDeploy="true"
    xmlValidation="false" xmlNamespaceAware="false">

```

<!-- Defines a cluster for this node,  
 By defining this element, means that every manager will be changed.  
 So when running a cluster, only make sure that you have webapps in there  
 that need to be clustered and remove the other ones.  
 A cluster has the following parameters:

className = the fully qualified name of the cluster class

name = a descriptive name for your cluster, can be anything

mcastAddr = the multicast address, has to be the same for all the nodes

mcastPort = the multicast port, has to be the same for all the nodes

mcastBindAddr = bind the multicast socket to a specific address

mcastTTL = the multicast TTL if you want to limit your broadcast

mcastSoTimeout = the multicast readtimeout

mcastFrequency = the number of milliseconds in between sending a "I'm alive" heartbeat

mcastDropTime = the number a milliseconds before a node is considered "dead" if no heartbeat  
 is received

tcpThreadCount = the number of threads to handle incoming replication requests, optimal would  
 be the same amount of threads as nodes

tcpListenAddress = the listen address (bind address) for TCP cluster request on this host,  
 in case of multiple ethernet cards.  
 auto means that address becomes  
 InetAddress.getLocalHost().getHostAddress()

tcpListenPort = the tcp listen port

tcpSelectorTimeout = the timeout (ms) for the Selector.select() method in case the OS  
 has a wakeup bug in java.nio. Set to 0 for no timeout

printToScreen = true means that managers will also print to std.out

expireSessionsOnShutdown = true means that

useDirtyFlag = true means that we only replicate a session after setAttribute,removeAttribute has been called.

false means to replicate the session after each request.

false means that replication would work for the following piece of code: (only for SimpleTcpReplicationManager)

```
<%
```

```
HashMap map = (HashMap)session.getAttribute("map");
```

```
map.put("key", "value");
```

```
%>
```

replicationMode = can be either 'pooled', 'synchronous' or 'asynchronous'.

\* Pooled means that the replication happens using several sockets in a synchronous way. Ie, the data gets replicated, then the request return. This is the same as the 'synchronous' setting except it uses a pool of sockets, hence it is multithreaded. This is the fastest and safest configuration. To use this, also increase the nr of tcp threads that you have dealing with replication.

\* Synchronous means that the thread that executes the request, is also the thread the replicates the data to the other nodes, and will not return until all nodes have received the information.

\* Asynchronous means that there is a specific 'sender' thread for each cluster node, so the request thread will queue the replication request into a "smart" queue, and then return to the client.

The "smart" queue is a queue where when a session is added to the queue, and the same session

already exists in the queue from a previous request, that session will be replaced in the queue instead of replicating two requests. This almost never happens, unless

there is a

large network delay.

```
-->
```

```
<!--
```

When configuring for clustering, you also add in a valve to catch all the requests coming in, at the end of the request, the session may or may not be replicated.

A session is replicated if and only if all the conditions are met:

1. useDirtyFlag is true or setAttribute or removeAttribute has been called AND
2. a session exists (has been created)
3. the request is not trapped by the "filter" attribute

The filter attribute is to filter out requests that could not modify the session, hence we don't replicate the session after the end of this request.

The filter is negative, ie, anything you put in the filter, you mean to filter out, ie, no replication will be done on requests that match one of the filters.

The filter attribute is delimited by ;, so you can't escape out ; even if you wanted to.

filter=".\*\gif;.\*\js;" means that we will not replicate the session after requests with the URI ending with .gif and .js are intercepted.

The deployer element can be used to deploy apps cluster wide.

Currently the deployment only deploys/undeploys to working members in the cluster so no WARs are copied upon startup of a broken node.



The deployer watches a directory (watchDir) for WAR files when watchEnabled="true"  
 When a new war file is added the war gets deployed to the local instance,  
 and then deployed to the other instances in the cluster.  
 When a war file is deleted from the watchDir the war is undeployed locally  
 and cluster wide

-->

<!--

```
<Cluster className="org.apache.catalina.cluster.tcp.SimpleTcpCluster"
  managerClassName="org.apache.catalina.cluster.session.DeltaManager"
  expireSessionsOnShutdown="false"
  useDirtyFlag="true"
  notifyListenersOnReplication="true">
```

```
<Membership
  className="org.apache.catalina.cluster.mcast.McastService"
  mcastAddr="228.0.0.4"
  mcastPort="45564"
  mcastFrequency="500"
  mcastDropTime="3000"/>
```

```
<Receiver
  className="org.apache.catalina.cluster.tcp.ReplicationListener"
  tcpListenAddress="auto"
  tcpListenPort="4001"
  tcpSelectorTimeout="100"
  tcpThreadCount="6"/>
```

```
<Sender
  className="org.apache.catalina.cluster.tcp.ReplicationTransmitter"
  replicationMode="pooled"
  ackTimeout="15000"/>
```

```
<Valve className="org.apache.catalina.cluster.tcp.ReplicationValve"
  filter=".*\.(gif|.*\.(js|.*\.(jpg|.*\.(htm|.*\.(html|.*\.(txt|"/>
```

```
<Deployer className="org.apache.catalina.cluster.deploy.FarmWarDeployer"
  tempDir="/tmp/war-temp/"
  deployDir="/tmp/war-deploy/"
  watchDir="/tmp/war-listen/"
  watchEnabled="false"/>
```

</Cluster>

-->

<!-- Normally, users must authenticate themselves to each web app  
 individually. Uncomment the following entry if you would like  
 a user to be authenticated the first time they encounter a  
 resource protected by a security constraint, and then have that  
 user identity maintained across \*all\* web applications contained  
 in this virtual host. -->

```

<!--
<Valve className="org.apache.catalina.authenticator.SingleSignOn" />
-->

<!-- Access log processes all requests for this virtual host. By
default, log files are created in the "logs" directory relative to
$CATALINA_HOME. If you wish, you can specify a different
directory with the "directory" attribute. Specify either a relative
(to $CATALINA_HOME) or absolute path to the desired directory.
-->
<!--
<Valve className="org.apache.catalina.valves.AccessLogValve"
    directory="logs" prefix="localhost_access_log." suffix=".txt"
    pattern="common" resolveHosts="false"/>
-->

<!-- Access log processes all requests for this virtual host. By
default, log files are created in the "logs" directory relative to
$CATALINA_HOME. If you wish, you can specify a different
directory with the "directory" attribute. Specify either a relative
(to $CATALINA_HOME) or absolute path to the desired directory.
This access log implementation is optimized for maximum performance,
but is hardcoded to support only the "common" and "combined" patterns.
-->
<!--
<Valve className="org.apache.catalina.valves.FastCommonAccessLogValve"
    directory="logs" prefix="localhost_access_log." suffix=".txt"
    pattern="common" resolveHosts="false"/>
-->
    <!--
| Begin MyWebApp context definition.
+-->
<!--
| End MyWebApp context definition.
+-->

<Context path="/rfms" docBase="rfms" reloadable="true" debug="0">

<Logger className="org.apache.catalina.logger.FileLogger"
    prefix="localhost_rfms" suffix=".txt" timestamp="true"/>

<Resource name="jdbc/rfmsDS" auth="Container"
    type="javax.sql.DataSource"/>

<ResourceParams name="jdbc/rfmsDS">
    <parameter>
        <name>factory</name>
        <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>

<!--
| The JDBC connection URL for connecting to your MySQL DB.

```

| The autoReconnect=true argument to the URL makes sure that the  
| MySQL JDBC Driver will automatically reconnect if mysqld closed  
| the connection. mysqld by default closes idle connections after  
| 8 hours.

+-->

```
<parameter>
  <name>url</name>
  <value>jdbc:mysql://localhost/rfms
    autoReconnect=true</value>
</parameter>
```

<!--

| MySQL username and password for DB connections.

+-->

<!--

| Class name for MySQL JDBC driver.

+-->

```
<parameter>
  <name>driverClassName</name>
  <value>com.mysql.jdbc.Driver</value>
</parameter>
```

<!--

| Maximum number of DB connections in pool. Make sure you  
| configure your mysqld max\_connections large enough to handle  
| all of your DB connections. Set to 0 for no limit.

+-->

```
<parameter>
  <name>maxActive</name>
  <value>100</value>
</parameter>
```

<!--

| Maximum number of idle DB connections to retain in pool.  
| Set to 0 for no limit.

+-->

```
<parameter>
  <name>maxIdle</name>
  <value>30</value>
</parameter>
```

<!--

| Maximum time to wait for a DB connection to become available  
| in ms, in this example 10 seconds. An exception is thrown if  
| this timeout is exceeded.  
| Set to -1 to wait indefinitely.

+-->

```
<parameter>
  <name>maxWait</name>
  <value>10000</value>
```

```
</parameter>  
</ResourceParams>
```

```
</Context>
```

```
</Host>
```

```
</Engine>
```

```
</Service>
```

```
</Server>
```

```
// end server.xml
```

## REFERENCES

- [1] Brown et al., Pro JSP, Third Edition, Shroff Publishers & Distributors Pvt. Ltd., 2004.
- [2] Whitehead P.,Friedman-Hill E.,Vanderveer E., Java and XML Your Visual blueprint for creating Java-enhanced Web Programs, Wiley Publishing, 2002.
- [3] Flower and Scott, UML Distilled, A Brief Guide to Standard Object Modelling Language, 1999.
- [4] Naughton P.,Schildt H., Java 2 - The Complete Reference, Tata McGraw Hill 1999.
- [5] Tremblett P., Instant JavaServer Pages, McGraw Hill 2000.