

P-2847



**ACCESSING WEB SERVER DATABASE USING
MOBILES**

A PROJECT REPORT

Submitted by

S.PRADEEP

71205104031

M.PRADEEP SRINIVASAN

71205104032

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

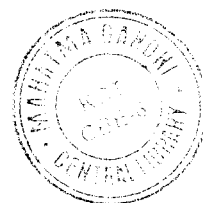
in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI-600 025

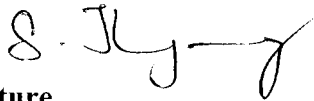
APRIL 2009



BONAFIDE CERTIFICATE

BONAFIDE CERTIFICATE

Certified that this project report entitled “**Accessing web server database using mobiles**” is the bonafide work of S.Pradeep and M.Pradeep Srinivasan, who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported herein does not from part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or other candidate.



Signature


Dr. S.Thangasamy

Dean,

Department of Computer Science & Engineering,

Kumaraguru College of Technology,

Coimbatore – 641 006



Signature

Mr.K.Sivan Arul Selvan, M.E

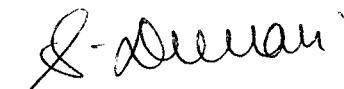
Senior lecturer

Department of Computer Science & Engineering,

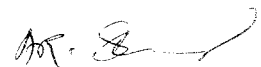
Kumaraguru College of Technology,

Coimbatore – 641 006

The candidates with University Register number 71205104031 and 71205104032 were examined by us in the project viva-voce examination held on 27-4-09.



Internal Examiner



External Examiner

DECLARATION


DECLARATION

We hereby declare that the project “**Accessing web server database using mobiles**” is a record of original work done by us and to the best of my knowledge; a similar work has not been submitted to Anna University or any institution, for fulfillment of the requirement of the course study.

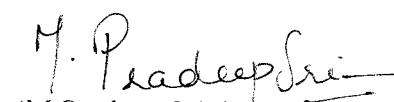
The report is submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Chennai.

Place: Coimbatore

Date: 27-4-09



(S.Pradeep)



(M.Pradeep Srinivasan)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We extend our sincere thanks to our Prof R. Annamalai, Vice Principal, Kumaraguru College of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facilities to work on this project.

We would like to make a special acknowledgement and thanks to Dr. S. Thangasamy, Ph.D., Dean, Department of Computer Science and Engineering, for his support and encouragement throughout the project.

We express deep gratitude and gratefulness to our guide Mr.K.Sivan Arul Selvan M.E, Senior lecturer, Department of Computer Science, for his supervision, enduring patience, active involvement and guidance.

We would like to thank our project coordinator, Mrs P. Devaki, Asst. Professor, for her support during the course of our project.

We would like to convey our honest thanks to all **Members of staff** the Department for their enthusiasm and wealth of experience from which we have greatly benefited.

Table of Contents

1. Abstract	- 10
2. List of Figures and Abbreviations	- 11
3. Introduction	- 2
4. Proposed Methodology	- 7
5. Programming Environment	- 10
6. Architecture	- 12
7. System Design	- 22
8. Implementation	- 27
9. Testing	- 34
10. Conclusion	- 38
Appendix	
- Sample Code.	- 40
- Sample Output.	- 44
11. References	- 51

ABSTRACT

Abstract

This project aims to develop a college website. It helps the students to receive their marks through mobile (via SMS) and email. The website requires authentication for staff entries. In case of registering the new staffs, the staffs need to be authenticated by the administrator. The staff who has administrator rights can edit students attendance and mark details. If the staffs add new details or updates existing details then the mail and message will be automatically sent to the respective end users email id. End users or parents can pose their queries with a particular syntax to receive their wards additional details regarding marks and attendance.

The staffs are also provided with an easy webpage interface to add students, details, marks, attendance etc. which reduces the maintenance of registries, further the user authentication of the database provides a high degree of security.

The website is developed using J2EE (Java 2 Enterprise Edition) which is a platform independent. The database used is MS Access.

LIST OF FIGURES

FIGURE NO.	FIGURE NUMBER	PAGE NO.
1	Mail Transmission architecture	12
2	Mail Transfer Block Diagram	13
3	Simple Structure of Message	15
4	Multipart Structure of Message	16
5	Use case Diagram for entire system	22
6	Sequence Diagram for email query module	23
7	Collaboration Diagram for email query module	24
8	Activity Diagram for entire system	25

LIST OF ABBREVIATIONS

SERIAL NO	ABBREVIATION	EXPANSION
1	HTTP	Hyper Text Transfer Protocol
2	LAN	Local Area Network
3	SMS	Short Messaging Service
4	JSP	Java Server Pages
5	IMAP	Internet Message Access Protocol
6	J2EE	Java 2 enterprise edition

INTRODUCTION

3. INTRODUCTION:

Accessing web server database using mobiles is a result publishing technique to convey the information regarding the students' results and attendance to their parents or guardians. This could be done by sending the marks and attendance percentage through mobiles and e-mails.

3.1. Scope

Accessing webservice database using mobiles can be used to retrieve information from a large repository in order to answer the queries posed by the user. The system can handle queries framed using simple English language with the use of SQL formats. Apart from answering queries, the system also helps the user to frame sensible queries by providing hints to construct queries. This system answers to the queries by analyzing the intent of the user by means of providing links. The system can also be used to update or delete entries in the database.

This system is developed in a much generic sense, by considering all the possible bottlenecks that would arise when the system is tried to be integrated with a different database, such that, the same implementation logic can be used for any database only by including relevant details in the mapping tables and making corresponding modifications in the code.

In order to demonstrate the working of this system, we use a Student Database. The Student Database comprises of Personal Details, Academic Details etc

3.2. Existing System.

In the existing system, the marks and attendance details of students can be conveyed to parents using traditional methodology. Some of the methodologies are given by

3.2.1. Notice Board.

In this method the marks and attendance report are put on the notice boards of institutions which can be viewed by the students.

The disadvantages here are time delay due to crowd, transportation, etc...

3.2.2. Face to face (in personal).

In this methodology the results or mark sheets of students are handed over to students or parents in personal.

The disadvantage here is that the students\parents undergo hardships such as transportation, time inconvenience etc...

3.2.3. Postal.

In this case the results of the students are sent to their respective postal address.

In this method the results might be delayed or may not be delivered due various reasons. Also students might have registered wrong addresses with the institutions.

3.2.4. Telephone.

Here the students can avail the results from the service providers for public examinations.

This technique is limited to very few public examinations and charges are applicable to avail these services. Lines to the numbers allocated to this service might be busy.

3.3. Proposed System

3.3.1 E-mail.

This is one of the latest technique in which the results of the students are sent through electronic mails by the concerned institutions.

One of the most powerful methodologies but might not be available in remote areas. Also at peak times services might not be available as servers might be busy or may face failures in some cases.

3.3.2 Mobile phone (service providers).

Here the students can contact the service providers' through special numbers to know their results.

In this technique the services are limited to certain examinations only. Charges for this service are applicable.

3.4 ADVANTAGES OF PROPOSED SYSTEM

- The users of this system need not have a complete understanding of the schema of the database.
- This system can automatically send marks and attendance details through e-mail and SMS(short message service) to the parents.
- The users are not required to know the format or the structure in which the query has to be constructed in order to obtain the desired information from the database.
- The users are only required to know what information they want to retrieve or know from the database.
- The system also provides extra information relevant to the query apart from providing the details requested by the user.
- The user preference such as intentions, interests and needs in databases are considered while answering the queries.

PROPOSED METHODOLOGY

4. Proposed Methodology

A process model for developing any project is chosen based on its nature and application, methods and tools to be used, controls and deliverables that are required. The proposed line of attack for “Accessing web server database using mobiles” is conversion of English Query to SQL Query and the objective is achieved using one of the significant software engineering models called the “Incremental Model”.

The Incremental model combines the elements of Linear Sequential Model or Waterfall Model with the iterative philosophy of prototyping. The Incremental model delivers software in small but usable pieces called “Increments”. In general, each increment builds on those that have already been delivered. Early increments are stripped down versions of the final product, but they do provide the capability that serves the user and also provide a platform for evaluation by the user.

When an incremental model is used the first increment is called the ‘CORE PRODUCT’. Here the basic requirements are addressed, but many supplementary features remain undelivered. After evaluation a plan is developed for the next increment. The plan addresses the modification of the core product to better meet the needs of the customer and delivery of additional features and functionalities. The process is repeated following the delivery of each increment, until the complete product is produced.

4.1. INCREMENTAL STAGES IN THE DEVELOPMENT OF THIS SYSTEM

4.1.1. CORE PRODUCT

- 3.1.1.1. Sending Mail.
- 3.1.1.2. Sending SMS(Short message service).
- 3.1.1.3. Querying through mail.

4.1.2. PHASE II

- 3.1.2.1. Database designing
 - 3.1.2.1.1. Table Creation
 - 3.1.2.1.2. User Authentication

4.1.3. PHASE III

- 3.1.3.1. Complete Website designing

4.1.4. PHASE IV

- 3.1.4.1. Providing additional details like keyword analyzing
- 3.1.4.2. Adding more details in database

PROGRAMMING ENVIRONMENT

5. PROGRAMMING ENVIRONMENT

The requirements for this system are classified into categories and are given below:-

5.1 Hardware interfaces:

Processor: Pentium IV or higher

Minimum RAM: 128 MB or more

Minimum Hard Drive capacity: 1 GB or more

5.2. Software Interfaces

Platform required: Windows XP

Database: Microsoft Access

Language: JDK1.5

Tomcat server is needed to host the web site.

5.3. Communication Interfaces:

A browser is needed to send requests to server in client side.

High speed Internet connection is needed.



ARCHITECTURE

6. ARCHITECTURE

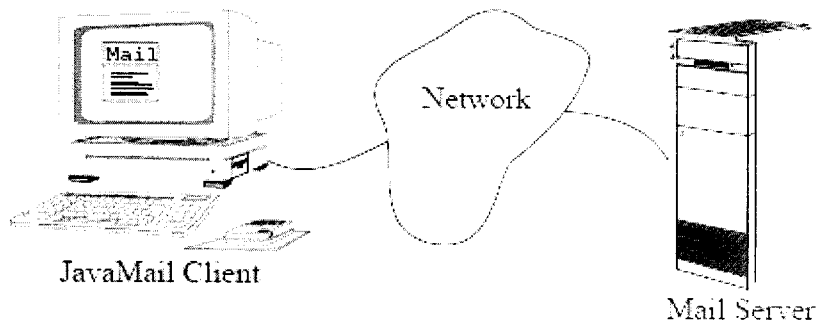


Fig no 1. Mail Transmission architecture

The Enterprise API is composed of a set of abstract classes that model the various pieces of a typical mail system. These classes include,

- Message—Abstract class that represents an electronic mail message.

Enterprise implements the RFC822 and MIME Internet messaging standards. The MimeMessage class extends Message to represent a MIME-style email message.

- Store—Abstract class that represents a database of messages maintained by a mail server and grouped by owner. A Store uses a particular access protocol.
- Folder—Abstract class that provides a way of hierarchically organizing messages. Folders can contain messages and other folders. A mail server provides each user with a default folder, and users can typically create and fill subfolders.
- Transport—Abstract class that represents a specific transport protocol. ATransport object uses a particular transport protocol to send a message.

As a service provider, you implement abstract classes in terms of your specific protocol or system. For example, an IMAP provider implements the Enterprise API

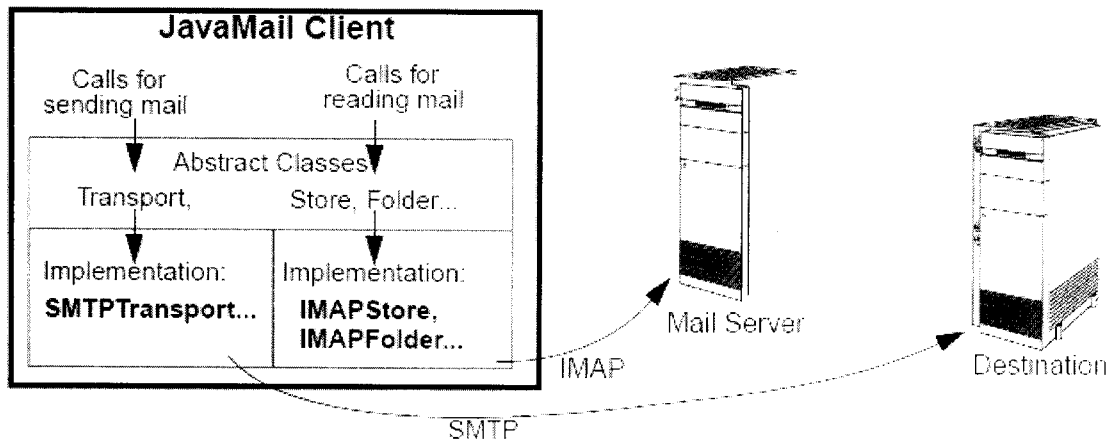


Fig no 2. Mail Transfer Block Diagram

This Service Provider's Guide shows you how to develop and package a JavaMail service provider for your clients. It is meant to be used in conjunction with the Javadoc provided with the JavaMail API and the JavaMail API Specification.

This guide covers:

- Creating messages
- Storing and retrieving messages
- Sending a message
- Communicating with a client (for example, notifying the client of new mail)
- Packaging your implementation

The descriptions of the first three tasks show how to subclass the appropriate abstract classes and implement their abstract methods. In addition, the task descriptions point out the methods that have default implementations that you might choose to override for the sake of efficiency.

6.1 EMAIL & MESSAGE

Messages are central to any electronic mail system. This chapter discusses how to implement them for your provider. If your provider allows only for the common case of creating and sending MIME style messages, then your provider can use the pre-written JavaMail message implementation: the `javax.mail.internet.MimeMessage` class. Implementations that furnish a protocol like SMTP fall into this category.

If your implementation does not fall into the previous category, you will have to implement your own Message subclass. This chapter

- Explains the structure of a Message object
- Explains how Message objects use the JavaBeans™ Activation Framework
- Shows you how to develop a Message subclass

6.1.1 The Structure of a Message

The Message class models an electronic mail message. It is an abstract class that implements the Part interface. The Message class defines a set of attributes and content for an electronic mail message. The attributes, which are name-value pairs, specify addressing information and define the structure of the message's content (its content type). Messages can contain a single content object or, indirectly, multiple content objects. In either case, the content is held by a DataHandler object.

6.1.2. Simple Messages;

A simple message has a single content object, which is wrapped by a DataHandler

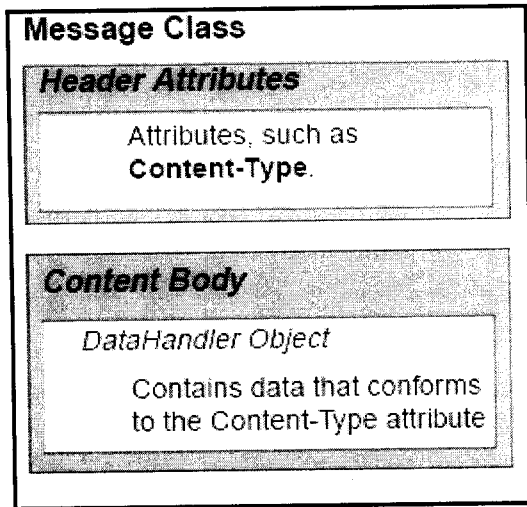


Fig no 3. Simple Structure of Message

6.1.3. Multipart Messages;

In addition to the simple structure shown in FIGURE 3, messages can also contain multiple content objects. In this case the DataHandler object contains a Multipart object, instead of merely a a single block of content data. A Multipart object is a container of BodyPart objects. The structure of a BodyPart object is similar to the structure of a Message object, because they both implement the Part interface. Each BodyPart object contains attributes and content, but the attributes of a Bodypart object are limited to those defined by the Part interface. An important attribute is the content-type of this part of the message content. The content of a BodyPart object is a DataHandler that contains either data or another Multipart object.

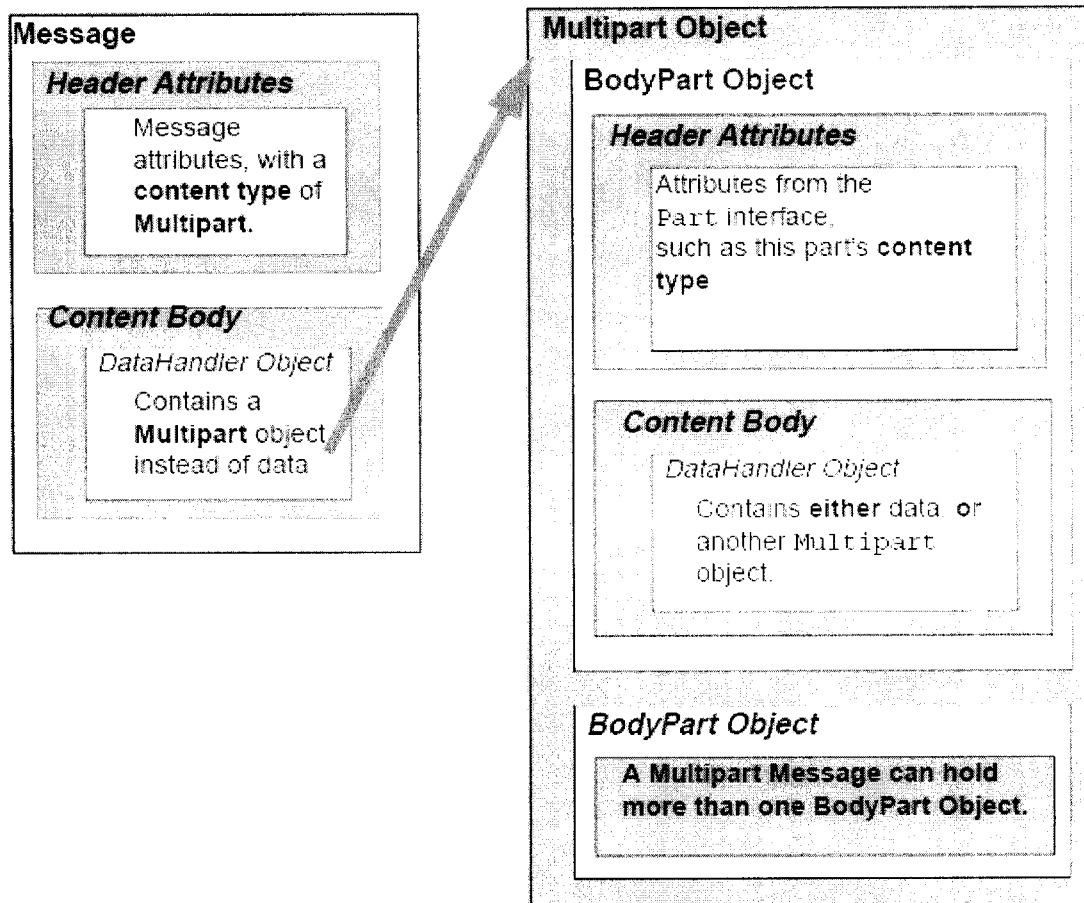


Fig no 4. Multipart Structure of Message

6.2. Message Storage and Retrieval

Users interact with message stores to fetch and manipulate electronic mail messages. This chapter discusses how to implement the classes that allow clients this access. If you are creating a JavaMail service provider that allows a client to send mail, but does not interface with a mail store, you do not have to implement this functionality. To provide message storage and retrieval, you must implement some abstract classes:

- Store, which models the message database and the access protocol used to retrieve messages. I

- Folder, which represents a node in the message storage hierarchy used to organize messages.

6.2.1 Store

The Store class models a message database and its access protocol. A client uses it to connect to a particular message store, and to retrieve folders (groups of messages). To provide access to a message store, you must extend the Store class and implement its abstract methods. In addition, you must override the default implementation of at least one method that handles client authentication. The next sections cover how to write these methods. They begin with authentication, since it precedes retrieval when the provider is used.

6.2.2 Authentication

JavaMail provides a framework to support both the most common style of authentication, (username, passphrase), and other more sophisticated styles such as a challenge-response dialogue with the user. To furnish the username, passphrase) style authentication in your provider, override the protocolConnect method. To use another style of authentication, you must override the version of the connect method that takes no arguments.

6.2.3 The protocol Connect Method;

The Store class provides a set of methods that establish a connection with a message store. Establishing a connection typically involves setting up a network connection to a host and authenticating the user with the message store installed on that host. The protocol Connect method handles these tasks.

The signature of the protocol Connect method is:

Boolean protocol Connect(String host, int port, String user, String password)

The method returns true if the connection and authentication succeed. If the connection fails, it throws a Messaging Exception. If the authentication fails, it returns false. The default implementation of protocol Connect returns false, indicating that the authentication failed. You should provide an implementation that connects to the given host at the specified port, and performs the service-specific authentication using the given username and passphrase. The simplest implementation, for message stores that do not require authentication, merely has this method return true. An example of such a message store is one that is local file-based. Note that clients do not call the protocolConnect method directly. Instead, the protocolConnect method is invoked when clients call one of the connect methods.]

6.2.3 The connect Method

To provide authentication schemes more sophisticated than (username, passphrase), you must override the version of the connect method that takes no arguments. The connect method takes no arguments and uses an Authenticator object to obtain information from the user if the information is not already available. (The client provides the Authenticator object.) It then uses that information to connect to the message store and authenticate the user. Finally, if the connection is successful, it delivers an OPENEDConnectionEvent. (For more information about events, see Chapter 5: Events.)

6.2.4 Folder Retrieval

A message store stores messages, and often allows users to further group their messages. These groups of messages are called folders, and are represented by the abstract class, Folder. The Store class provides abstract methods for allowing the user to retrieve a folder:

- getDefaultFolder
- getFolder

The `getDefaultFolder` method must return the default folder. The returned folder must be in a closed state. (This is the default initial state of a folder.)

The `getFolder` methods return the specified folders:

- Folder `getFolder(String name)`
- Folder `getFolder(URLConnection urlname)`

These methods return the requested folders whether or not they exist in the store. (This is similar to the `java.io.File` API.) Do not validate the folder's existence in the `getFolder` methods. The folders returned by the `getFolder` methods must be in a closed state. (The default initial state of a folder is closed.) Note – The Store object should not cache Folder objects. Invoking the `getFolder` method multiple times on the same folder name must return that many distinct Folder objects.

6.2.5 Folders

The Folder class models a node in a mail storage hierarchy. Folders can contain messages or subfolders or both. Each user has a folder that has the case-insensitive name INBOX. Providers must support this name. Folders have two states: they can be closed (operations on a closed folder are limited) or open. Since Folder is an abstract class, you must extend it and implement its abstract methods. In addition, some of its methods have default implementations that, depending on your system, you can override. Get the byte stream of the message, and transmit the message using its `writeTo` method.

6.2.6 Send a Transport Event

The `TransportEvent` indicates the delivery status of the message. The possible event types are `MESSAGE_DELIVERED`, `MESSAGE_NOT_DELIVERED`, and `MESSAGE_PARTIALLY_DELIVERED`.

6.2.7 Throw an exception if the delivery is unsuccessful

If the delivery fails, completely or partially, you must throw a suitable `MessagingException` or `SendFailedException`

SYSTEM DESIGN

7. SYSTEM DESIGN

The design of the proposed system is shown in the following diagrams.

7.1 Use case Diagram

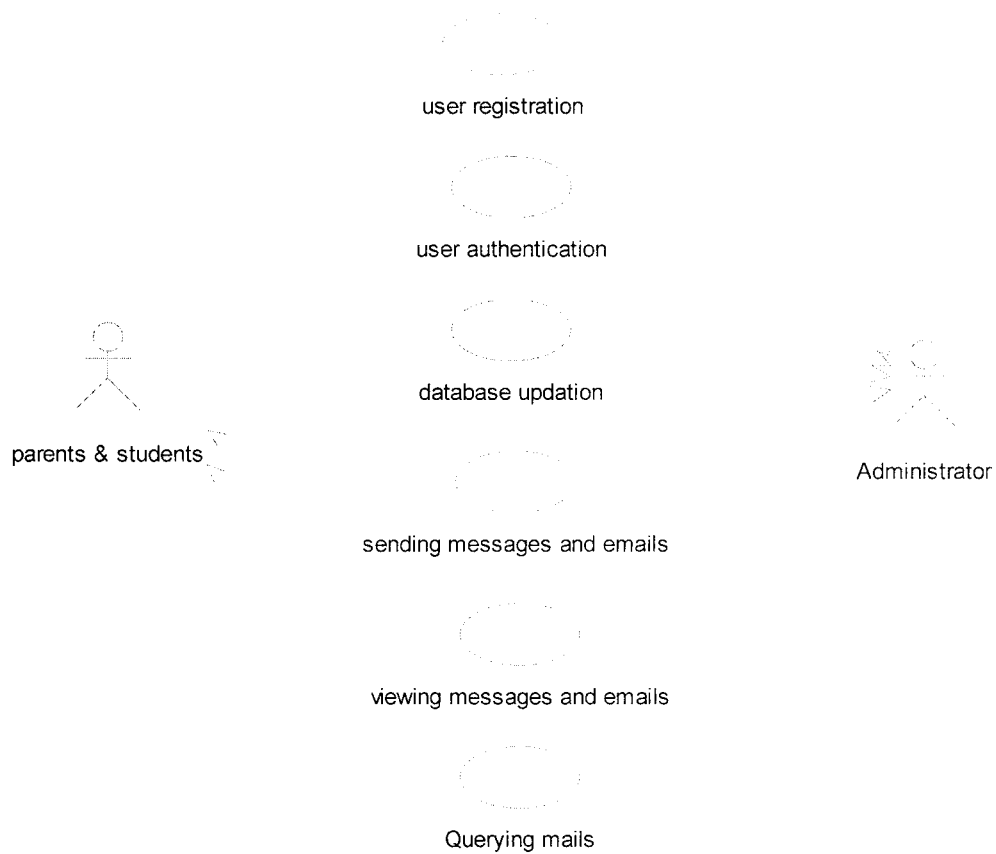


Fig no 5. Use case diagram for entire system

The above diagram shows the users in the system and the process each one of them is associated with

7.2 Sequence Diagram for Email Query module

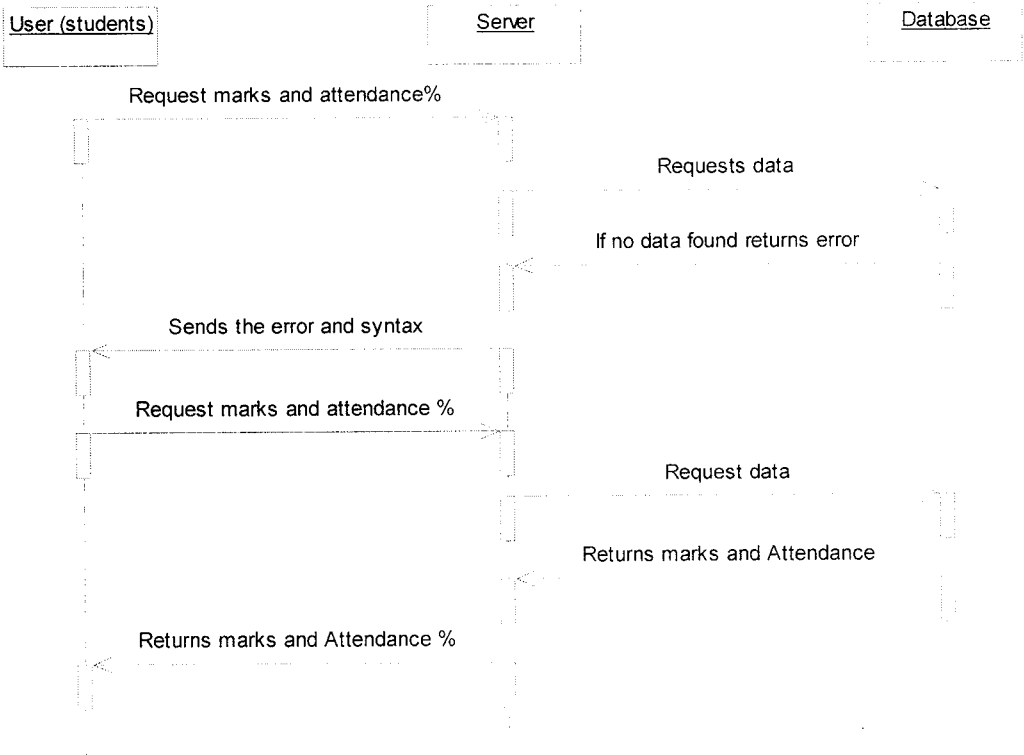


Fig no 6. Sequence Diagram for Email Query module

7.3 Collaboration Diagram for Email Query module

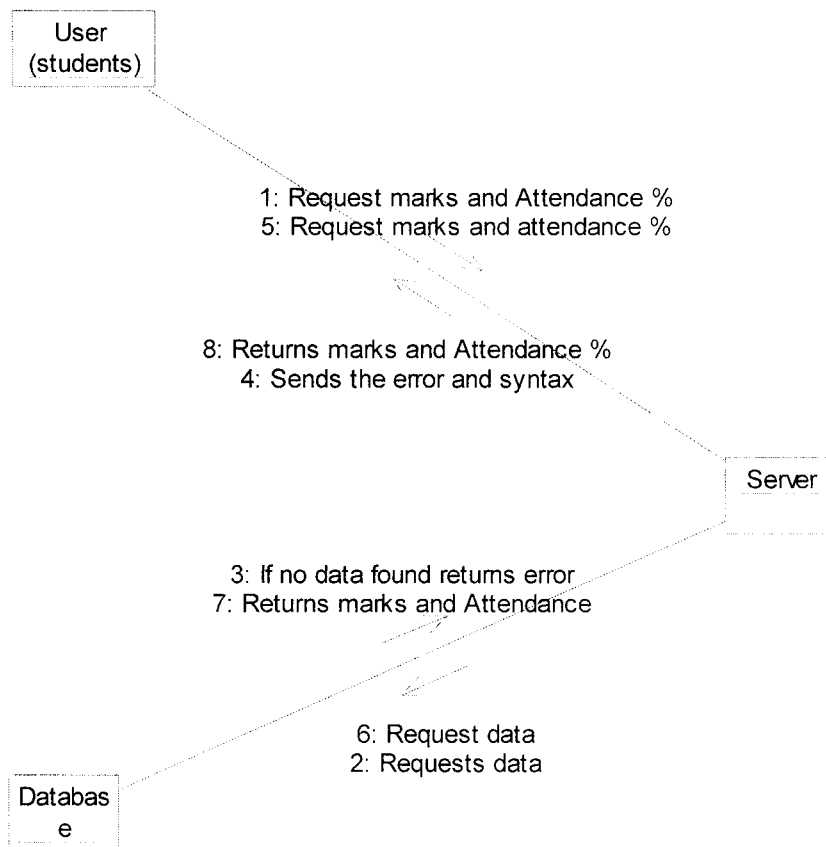


Fig no 7. Collaboration Diagram for Email Query module

7.4 Activity Diagram

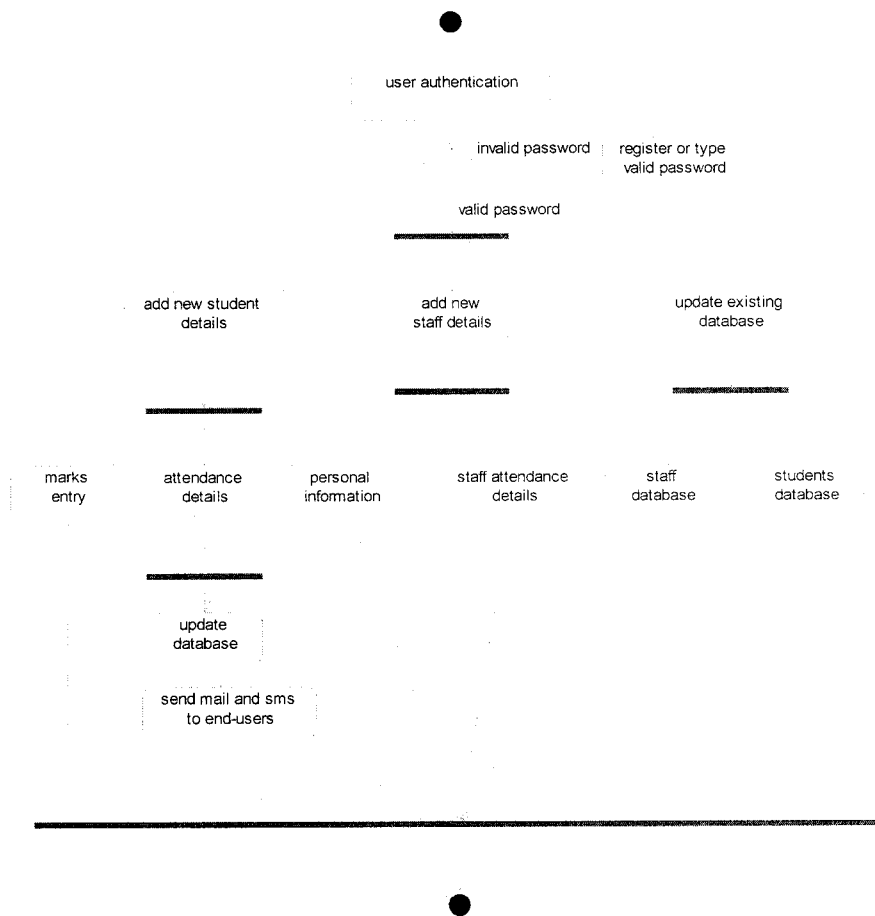


Fig no 8. Activity diagram for entire system

IMPLEMENTATION

8. Implementation:

The general idea of the existing techniques as well as the proposed scheme is described in the above sections. The detailed functioning of the system is described in the following sequence of steps:-

1. First the server page is running on the server system which can be accessed from anywhere.
2. To access the page the user must enter the page address or ip address on the browser.
3. As mentioned this system can be accessed by both the authenticated administrator as well as the end-user.
4. On entering into the project's home page the user is prompted for login and password (authentication is provided only for administrator) or in case of end-user then the system prompts for the students' roll number.
5. Once the authentication process is completed the user is shown the webpage which contains the options available as per the users' designation.
6. In case if the user is an administrator then he is provided with the options such as marks entry, attendance entry and students entry.
7. In the sections marks entry and attendance entry the administrator can enter the students' attendance and marks accordingly. Once the entry process is completed the administrator can send these records to the end-users e-mail id as well as to their mobile phones.
8. In the section students' entry the administrator can update the existing database with new students' entries.
9. Also in the students' entry section the administrator can as well as edit and update the existing students' records.
10. In case if the user is an end-user then he/she can enter their roll numbers and view their marks or attendance percentage accordingly.
11. Also in this system the user is provided with a querying option.

12. In the querying section the end-users can send queries in the form of e-mails and in a pre-determined format..
13. These queries are checked for authentication, and the authenticated queries are stored in the database.
14. The administrator can retrieve these mails from the database and replies to the users are sent automatically.

8.1 Java Server Pages (JSP)

JavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server- and platform-independent.

JavaServer Pages (JSP) technology enables Web developers and designers to rapidly develop and easily maintain, information-rich, dynamic Web pages that leverage existing business systems. As part of the Java technology family, JSP technology enables rapid development of Web-based applications that are platform independent. JSP technology separates the user interface from content generation, enabling designers to change the overall page layout without altering the underlying dynamic content.

Benefits for Developers

If you are a Web page developer or designer who is familiar with HTML, you can:

Use JSP technology without having to learn the Java language:

You can use JSP technology without learning how to write Java scriptlets. Although scriptlets are no longer required to generate dynamic content, they are still supported to provide backward compatibility.

Extend the JSP language:

Java tag library developers and designers can extend the JSP language with "simple tag handlers," which utilize a new, much simpler and cleaner, tag extension API. This spurs the growing number of pluggable, reusable tag libraries available, which in turn reduces the amount of code needed to write powerful Web applications.

Easily write and maintain pages:

The JavaServer Pages Standard Tag Library (JSTL) expression language is now integrated into JSP technology and has been upgraded to support functions. The expression language can now be used instead of scriptlet expressions.

JSP Technology and Java Servlets

JSP technology uses XML-like tags that encapsulate the logic that generates the content for the page. The application logic can reside in server-based resources (such as JavaBeans component architecture) that the page accesses with these tags. Any and all formatting (HTML or XML) tags are passed directly back to the response page. By separating the page logic from its design and display and supporting a reusable component-based design, JSP technology makes it faster and easier than ever to build Web-based applications.

JavaServer Pages technology is an extension of the Java Servlet technology. Servlets are platform-independent, server-side modules that fit seamlessly into a Web server framework and can be used to extend the capabilities of a Web server with minimal overhead, maintenance, and support. Unlike other scripting languages, servlets involve no platform-specific consideration or modifications; they are application components that are downloaded, on demand, to the part of the system that needs them. Together, JSP technology and servlets provide an attractive alternative to other types of dynamic Web scripting/programming by offering: platform independence; enhanced performance; separation of logic from display; ease of administration; extensibility into the enterprise; and, most importantly, ease of use.

8.2 J2EE (Java 2 Enterprise Edition)

J2EE is the technology which any computer programmer can learn without spending money from his pocket. If you ask why, the answer is simple - everything is free in the j2ee world. This site showcases more on J2EE materials, tutorial downloads, examples, Struts, EJB, Servlets, interview questions, tips and tricks etc.. to propel your J2EE know-how swiftly . J2EE (Java 2 Enterprise Edition) is a specification for developing enterprise and distributed applications from JavaSoft (Sun Microsystems). J2EE is not one thing; it encompasses a large set of technologies ranging from JSP to CORBA. In expectancy of the last release of the J2EE 1.0 specification and reference implementation at the end of 1999, a smaller minority of forward thinking companies began to focus on offering a complete suite of J2EE products, from web server to EJB application server.”

In the commercial world, we use Java 2 Enterprise Edition (J2EE) to solve business problems, to develop commercial software, or to provide contract services to other businesses' projects.

J2EE applications are made up of components. A J2EE component is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and that communicates with other components. The J2EE specification defines the following J2EE components:

- Application clients and applets are components that run on the client.
- Java Servlet and JavaServer Pages (JSP) technology components are web components that run on the server.
- Enterprise JavaBeans (EJB) components (enterprise beans) are business components that run on the ser

J2EE components are written in the Java programming language and are compiled in the same way as any program in the language. The difference between J2EE

components and "standard" Java classes is that J2EE components are assembled into a J2EE application, are verified to be well formed and in compliance with the J2EE specification, and are deployed to production, where they are run and managed by the J2EE server.

Some of the key features and services of J2EE:

- At the client tier, J2EE supports pure HTML, as well as Java applets or applications. It relies on Java Server Pages and servlet code to create HTML or other formatted data for the client.
- Enterprise JavaBeans (EJBs) provide another layer where the platform's logic is stored. An EJB server provides functions such as threading, concurrency, security and memory management. These services are transparent to the author.
- Java Database Connectivity (JDBC), which is the Java equivalent to ODBC, is the standard interface for Java databases.
- The Java servlet API enhances consistency for developers without requiring a graphical user interface.

TESTING

9. Testing

Testing is the process of executing a program with the intent of finding any errors. A good test of course has the high probability of finding a yet undiscovered error. A successful testing is the one that uncovers a yet undiscovered error.

A test is vital to the success of the system. System test makes a logical assumption that if all parts of the system are correct, then goal will be successfully achieved. The candidate system is subjected to a variety of tests online like responsiveness, its value, stress and security. A series of tests are performed before the system is ready for user acceptance testing.

TYPES OF TESTING

The different types of testing are: -

- Unit Testing
- Integration Testing
- Validation Testing
- Output Testing
- User Acceptance Testing

UNIT TESTING

Here we test each module individually and integrate the overall system. Unit testing focuses verification efforts even in the smallest unit of software design in each module. This is also known as “Module Testing”. The modules of the system are tested separately. This testing is carried out in the programming style itself. In this testing each module is focused to work satisfactorily as regard to expected output from the module. There are some validation checks for the fields.

INTEGRATION TESTING

Data can be lost across an interface, one module can have an adverse effect on the other sub-functions, when combined may not produce the desired functions. Integrated testing is the systematic testing to uncover the errors within the interface. This testing is done with simple data and the developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance.

VALIDATION TESTING

At the culmination of the black box testing, software is completely assembled as a package. Interfacing errors have been uncovered and correct and final series of test, i.e., validation test begins. Validation test can be defined with a simple definition that validation succeeds when the software functions in a manner that can be reasonably accepted by the customer.

USER ACCEPTANCE TESTING

User acceptance testing of the system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of development and making change whenever required. This is done with regard to the input screen design and output screen design.

OUTPUT TESTING

After performing validation testing, the next step is output testing of the proposed system. Since the system cannot be useful if it does not produce the required output. Asking the user about the format in which the system is required tests the output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is on screen format and another one is printed format. The output format on the screen is found to be corrected as the format was designed in the system phase according to the user needs. As for the hard copy the output comes according to the specification requested by the user. Here the output testing does not result in any correction in the system.

Taking various kinds of data plays a vital role in system testing. After preparing the test data, system under study is tested using the test data. While testing, errors are again uncovered and corrected by using the above steps and corrections are also noted for future use. The system has been verified and validated by running test data and live data. First the system is tested with some sample test data that are generated with the knowledge of the possible range of values that are required to hold by the fields. The system runs successfully for the given test data and for the live data.

CONCLUSION

10. CONCLUSION

This web service provides round the clock service to the students and also for the parents who wish to view details about their wards. The parents are not required to visit the college to collect details about their ward. They can pose their queries in simple English statements which are already mentioned in the webpage to the provided address for which they will be responded immediately.

The staffs are also provided with an easy webpage interface to add students, details, marks, attendance etc. which reduces the maintenance of registries, further the user authentication of the database provides a high degree of security.

APPENDIX

APPENDIX

Sample code for Sending Mail:

```
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;

class Sendmail
{
    public static void main(String args[])
    {
        final String SMTP_HOST_NAME =
"smtp.gmail.com";
        final String
SMTP_AUTH_USER="Conveyer4u@gmail.com";
        final String SMTP_AUTH_PWD="012345abc";
        String emailMsgTxt="test mail";
        String subject="test";
        String from="Conveyer4u@gmail.com";

        from =SMTP_AUTH_USER;

        String[] recipients={"sudhakarvel@gmail.com"};
    }
}
```

```

class SMTPAuthenticator extends javax.mail.Authenticator
{
    public PasswordAuthentication
getPasswordAuthentication()
    {
        String username =
SMTP_AUTH_USER;
        String password =
SMTP_AUTH_PWD;
        return new
PasswordAuthentication(username, password);
    }
}

try
{
    boolean
debug = false;

    boolean sessionDebug = false;

    Properties
props = new Properties();

    java.security.Security.addProvider(new
com.sun.net.ssl.internal.ssl.Provider());

    props =
System.getProperties();

    props.put("mail.smtp.host", SMTP_HOST_NAME);
}

```

```

        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketF
actory");
        props.put("mail.smtp.socketFactory.fallback", "false");
        Authenticator auth = new SMTPAuthenticator();
        javax.mail.Session sessionMail = javax.mail.Session.getInstance(props,auth);
        MimeMessage message = new MimeMessage(sessionMail);
        Transport transport = sessionMail.getTransport("smtp");
        transport.connect(SMTP_HOST_NAME,SMTP_AUTH_USER,SMTP_AUT
        H_PWD);

```

```

        sessionMail.setDebug(sessionDebug);
        Message msg = new MimeMessage(sessionMail);
        InternetAddress addressFrom = new InternetAddress(from);

```

```

        msg.setFrom(addressFrom);

```

```

        InternetAddress[]      addressTo      =      new
        InternetAddress[recipients.length];

```

```

        for (int i = 0; i < recipients.length; i++)
        {
            addressTo[i] = new InternetAddress(recipients[i]);
        }

```

```
msg.setRecipients(Message.RecipientType.TO, addressTo);

msg.setSubject(subject);

msg.setContent(emailMsgTxt,"text/plain");

Transport.send(msg);

System.out.println("Sucessfully Sent mail to All Users");
    }

catch(Exception e)
    {

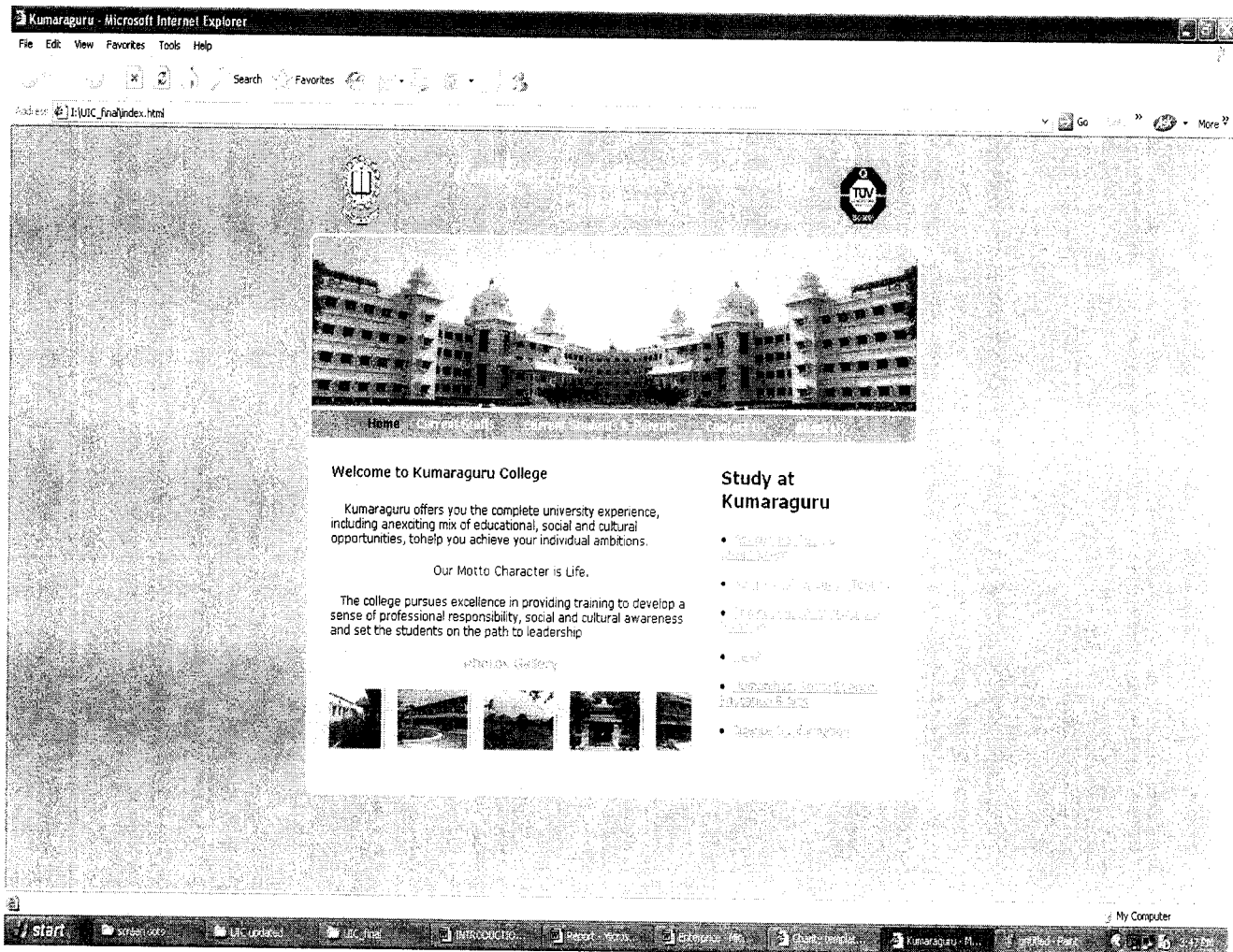
System.out.println(e.getMessage());
    }

}

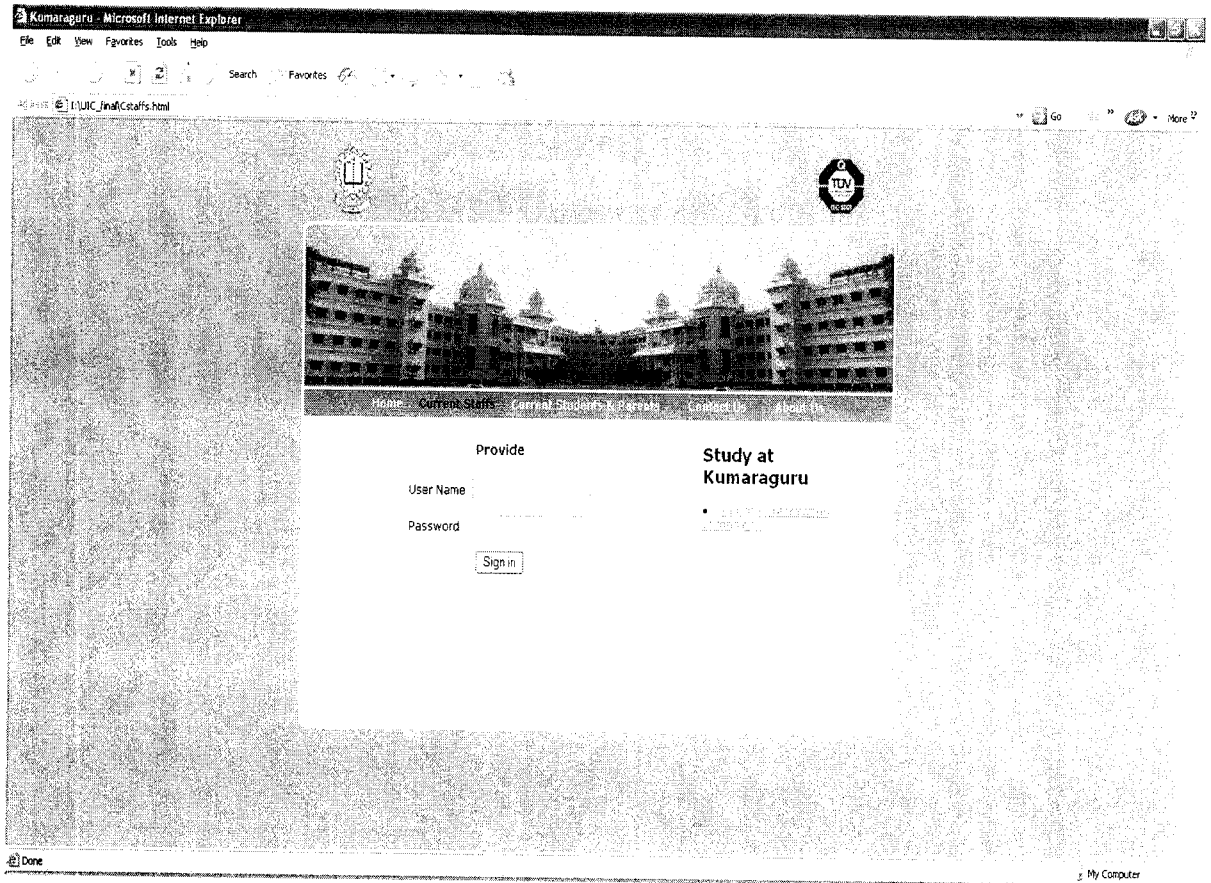
}
```

SAMPLE OUTPUT

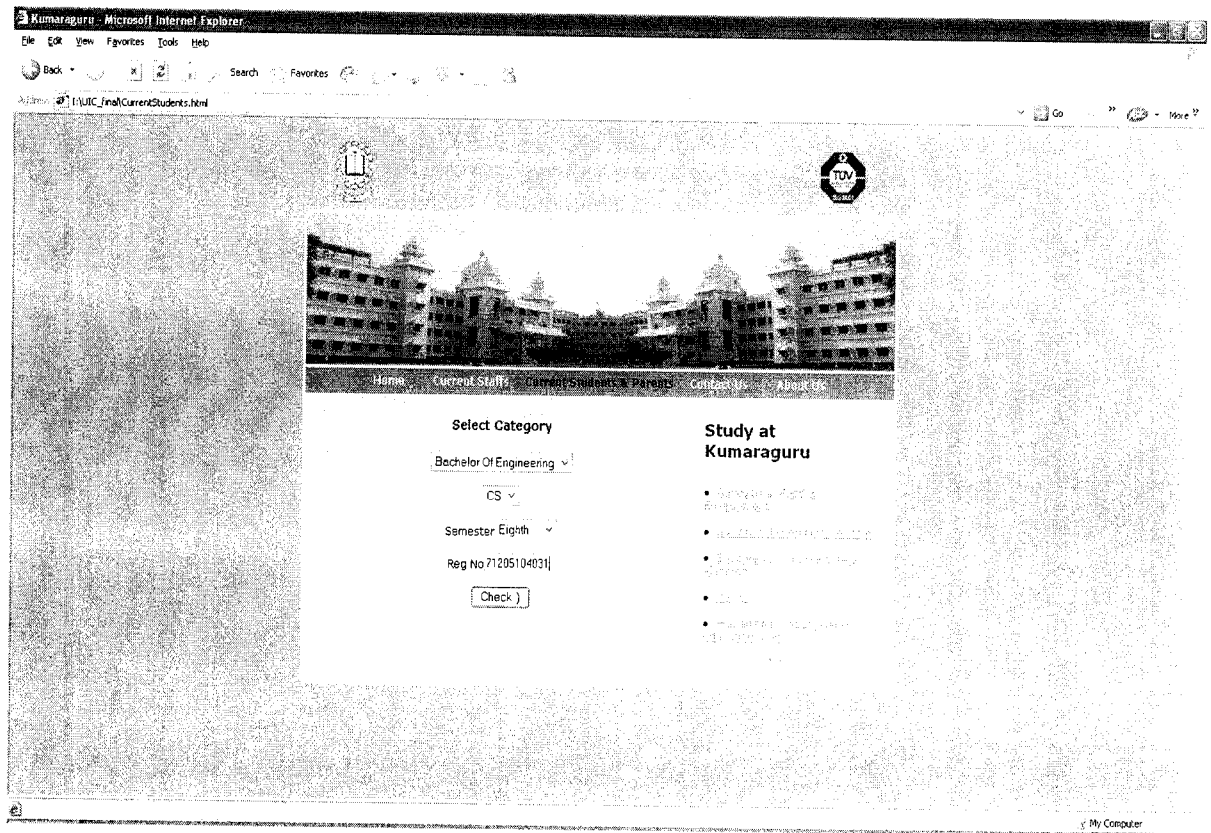
HOME PAGE



STAFF LOGIN PAGE



STUDENTS LOGIN PAGE



END USER MAIL BOX

The screenshot displays a Yahoo! Mail inbox for the user 'n.mc945.mail.yahoo.com'. The interface includes a navigation bar with 'Inbox (2)', 'Drafts', 'Sent', 'Spam', and 'Trash'. A sidebar on the left shows 'My Folders' and 'Search Shortcuts'. The main area contains a list of 14 emails. The 'From' column lists various senders, including 'conveyer4u@gmail.com', 'Monster.com', 'Prabu K B', 'Pradeep C.K.', 'Opinion Technologies', 'monsterindia.com', and 'campus@nuglifeent.com'. The 'Subject' column contains titles such as 'Attendance Details (pradeep sri)', 'Mark Details pradeep', 'Monster.com recommended best practices', '[KOT Placement Cell] Wipro Paper Presentation Contest', '[KOT Placement Cell] Application Form for Indian Air Force', and '[KOT Placement Cell] Tata Curable Quiz on 23rd Feb 2009 at Hotel The Presidency'. The 'Date' and 'Size' columns show the timestamp and file size for each message.

From	Subject	Date	Size
conveyer4u@gmail.com	Attendance Details (pradeep sri)	Thu, 24/09	3KB
conveyer4u@gmail.com	Mark Details pradeep	Thu, 24/09	3KB
Monster.com	Monster.com recommended best practices	Tue, 10/09/09	12KB
Prabu K B	Poster	Wed, 16/09/09	65KB
Pradeep C.K.	[KOT Placement Cell] Wipro Paper Presentation Contest	Wed, 16/09/09	30KB
Monster.com	Prospective employers are skipping your profile!	Tue, 17/09/09	11KB
Yahoo! Member Services	Password changed	Sun, 14/09/09	5KB
Pradeep C.K.	[KOT Placement Cell] Application Form for Indian Air Force	Thu, 12/09/09	129KB
Pradeep C.K.	[KOT Placement Cell] TATA Curable Quiz on 23rd Feb 2009 at Hotel The Presidency	Tue, 10/09/09	75KB
Opinion Technologies	Project list...	Sat, 31/07/09	301KB
monsterindia.com	monsterindia.com: Hi Pradeep, My Monster Login Details	Sat, 31/07/09	3KB
Monster.com	Submit your resume now	Thu, 29/09/09	11KB
Pradeep C.K.	[KOT Placement Cell] Inter-College Events - Feb 09	Wed, 25/09/09	49KB
Pradeep C.K.	[KOT Placement Cell] Inter-College Events - Feb 09	Wed, 24/09/09	49KB
campus@nuglifeent.com	Cognizant in media	Fri, 25/09/09	27KB

RECEIVED MAIL DETAILS

in.mc945.mail.yahoo.com

Contacts Calendar Notepad What's New? Mobile Mail RSS Options

Check Mail Compose Search Mail Search the Web

Previous Next Back to Newsletter

Delete Reply Forward Spam Move...

Attendance Details:pradeep sri
From: "pradeep sri" <pradeepsri@yahoo.com>
To: "pradeep sri" <pradeepsri@yahoo.com>

Name :pradeep sri
RegNo :71205104032
Department :BE-CS
Batch :2005
Semester :First
Main Exam Percentage :35
Over All Attendance Percentage :89

Subjects
C1 87
C2 87
C3 78
C4 57

My Folders [Add - Edit]
Inbox (1)
Drafts
Sent
Spam [Empty]
Trash [Empty]

Search Shortcuts
My Photos
My Attachments

ADVERTISEMENT

Delete Reply Forward Spam Move...

RECEIVED MAIL DETAILS

INDIA

Contacts Calendar Notepad

What's New? Mobile Mail Options

Check Mail Compose

Search Mail Search the Web

Previous | Next | Back to Messages

Mark as Unread | Print

Delete Reply Forward Spam Move...

Mark Details pradeep

From: "conveyent@gmail.com" <conveyent@gmail.com>

To: sri.pradeep@yahoo.co.in

Thursday, 2 April, 2009 4:07 AM

My Folders [Add - Edit]

- sri (8)

Search Shortcuts

- My Photos
- My Attachments

Name pradeep
RegNo 71205104032
Department BE-CS
Batch 2005
Semester First
ClassTest One
InternalTest Nil
Subjects C1 89
C2 90
C3 90
C4 99
Percentage 99
Comments good

Delete Reply Forward Spam Move...

Previous | Next | Back to Messages

Select Message Encoding | Full Screen

REFERENCES

11. References

1. Hans Bergsten, *"Java server pages"*, 3rd Edition, Published by O'Reilly, 2003.
2. Phil Hanna, NetLibrary, Inc, *"JSP: The complete Reference"*, Published by Osborne/McGraw-Hill, 2001.
3. James Keogh, Jim Keogh, *"J2EE: The complete Reference"*, Published by McGraw-Hill/Osborne, 2002
4. Jon Duckett, *"Beginning Web programming with HTML, XHTML, and CSS"*, Published by John Wiley and Sons, 2004.