**Graphic Era University**

**Dehradun**

PROJECT REPORT

Chat Bot

**April, 2011**

**REVISION LIST**

| Ver.Rev | Date | Authors | Description |
| --- | --- | --- | --- |
| 1.0 | 13-April-2011 | Avijatya Vashishtha,  Avin Singhal,  Balaji &  Chetan Melkani | Project Report |

**TABLE OF CONTENTS**

1. INTRODUCTION…………………………………………………………………….…..4
2. FUNCTIONAL REQUIREMENTS…………………………………………………….,.5
   1. REQUIREMENTS IN SCOPE………………………………………...................5
   2. REQUIREMENTS OUT OF SCOPE …………………………………………….5
3. USER INTERFACE………………………………………………………………………5
4. SYSTEM SNAPSHOTS ………………………………………………………...………..6
5. EXTERNAL SYSTEM INTERFACE REQUREMENTS………………………………11
6. NON FUNCTIONAL REQUIREMENTS………………………………………………12
7. HIGH LEVEL DESIGN……..………………………………………………………….13
8. DETAILED DESIGN…….……………………………………………………………..15
9. HARDWARE AND SOFTWARE REQWIREMENTS…………………………….….25
10. SUmmery……………………………………………………………………………...26
11. References…………………………………………………………………….….…26
12. **INTRODUCTION**

This project is a web application. The product is used to create an impression of a live help through the medium of text chat. Here one person asks a question and the program processes it and generates a suitable answer for the question asked. A chatbot is a computer program designed to simulate an intelligent conversation with one or more human users via auditory or textual methods, primarily for engaging in small talk. Since the product is available on Internet, it is accessible only through a secure log in IDs (users/admin). The admin will have access rights to edit the knowledge base on the product and see the error log file and make the corresponding changes in the database. For regular user the access to the portal is through single sign on.

The system decided to build using JEE technology and built with the help of JSP. This help the development team to concentrate on functional requirements and non functional requirements would be taken care by the system. This Java technology that helps to serve dynamically generated web pages based on HTML and XML.

# FUNCTIONAL REQUIREMENTS

## REQUIREMENTS IN SCOPE

One Module takes care of Role Based Access Control (RBAC) through login page from an XML file which has the credentials stored.

**Role: Admin**

1. Add Knowledge Base :

Here the admin must be able to improve the knowledge base used to answer a posted question to an XML or a flat file.

1. Get the failed replies :

Here the admin must get the reports about the failed cases in the form of a CSV file with appropriate fields where the bot couldn’t reply.

**Role: Customer**

The user will have a console (web browser) to post his questions and the replies coming from the program will also be displayed in the same console with proper messages. Whenever the bot is not able to generate a reply a default apology is to be generated replied and the case should be logged.

## REQUIREMENTS OUT OF SCOPE

The user can’t set the complexity of the chat (categories include layman and skilled user).

## USER INTERFACE

**Login** : Has a username and password field for the authentication and a sign up option.

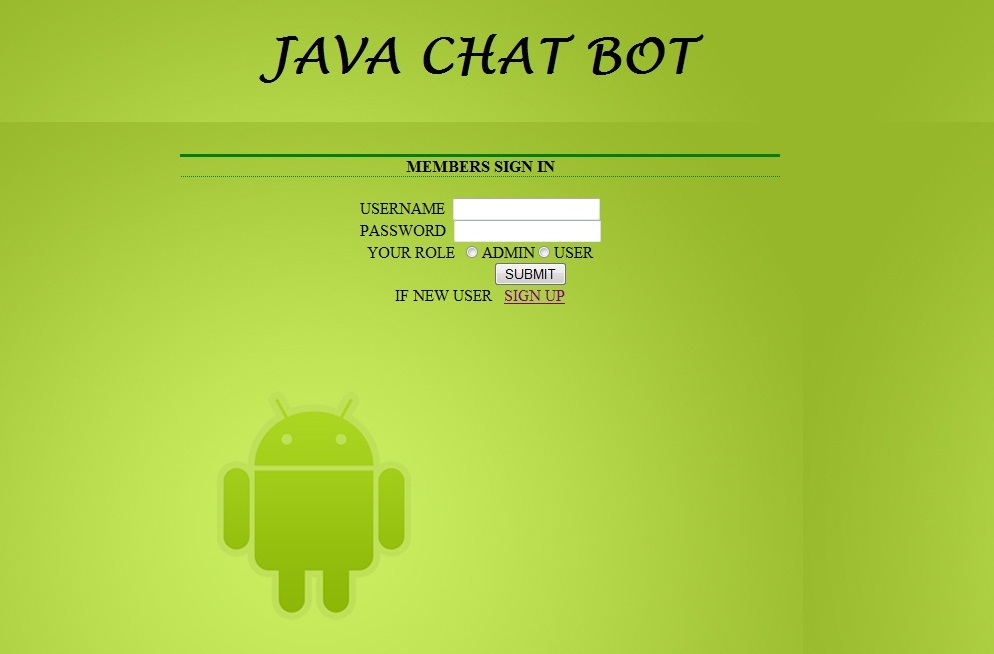
**Add Knowledge Base :** Has a console where the admin can improve the knowledge base.

**Get the failed replies :** Generate the list of the cases where the bot couldn’t generate the replies. Which gets stored in a CSV file.

**Chat :** The interface must depict a conventional chat window with a submit button.

1. **SYSTEM SNAPSHOTS**

**LogIn:**



**LogIn(client side validation):**



**LogIn(server side validation):**



**SignUp form:**



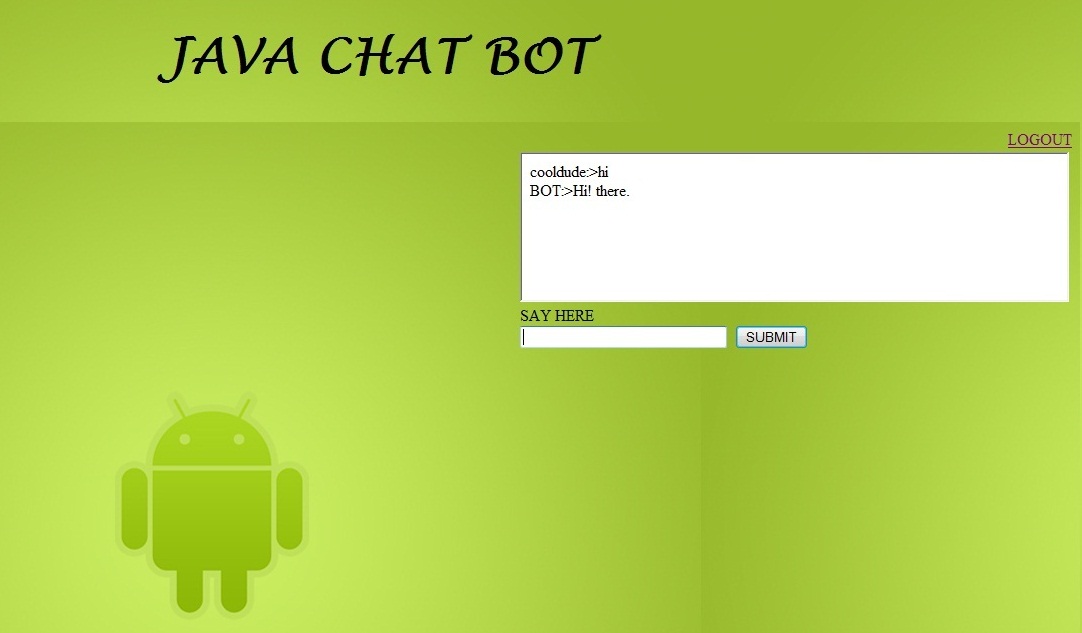
**Successful SignUp:**



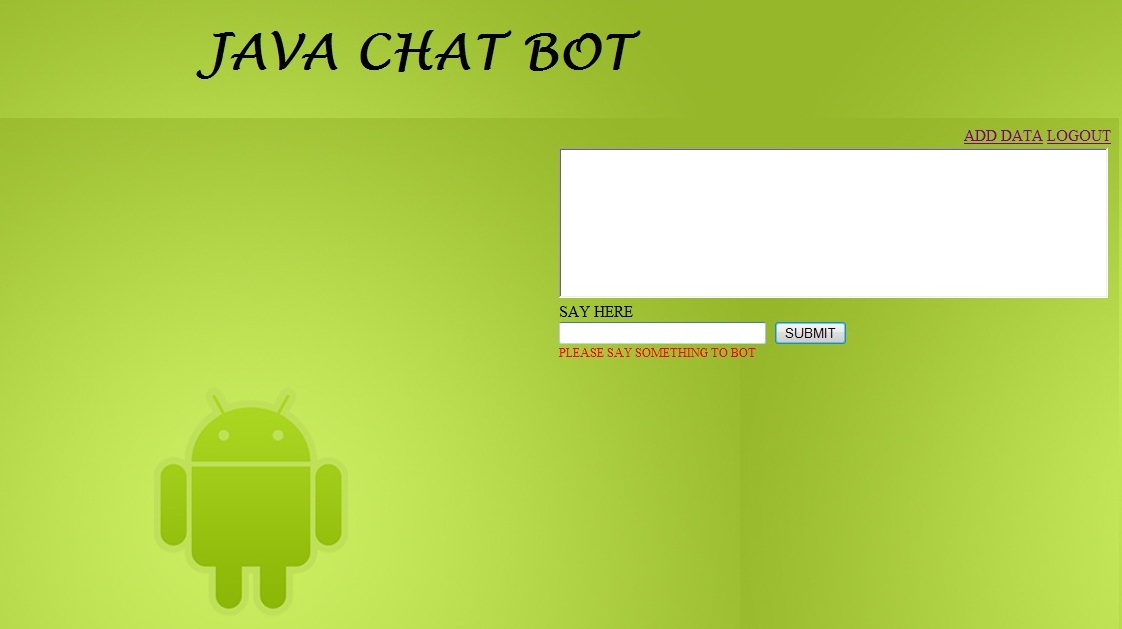
**SignUp Error:**



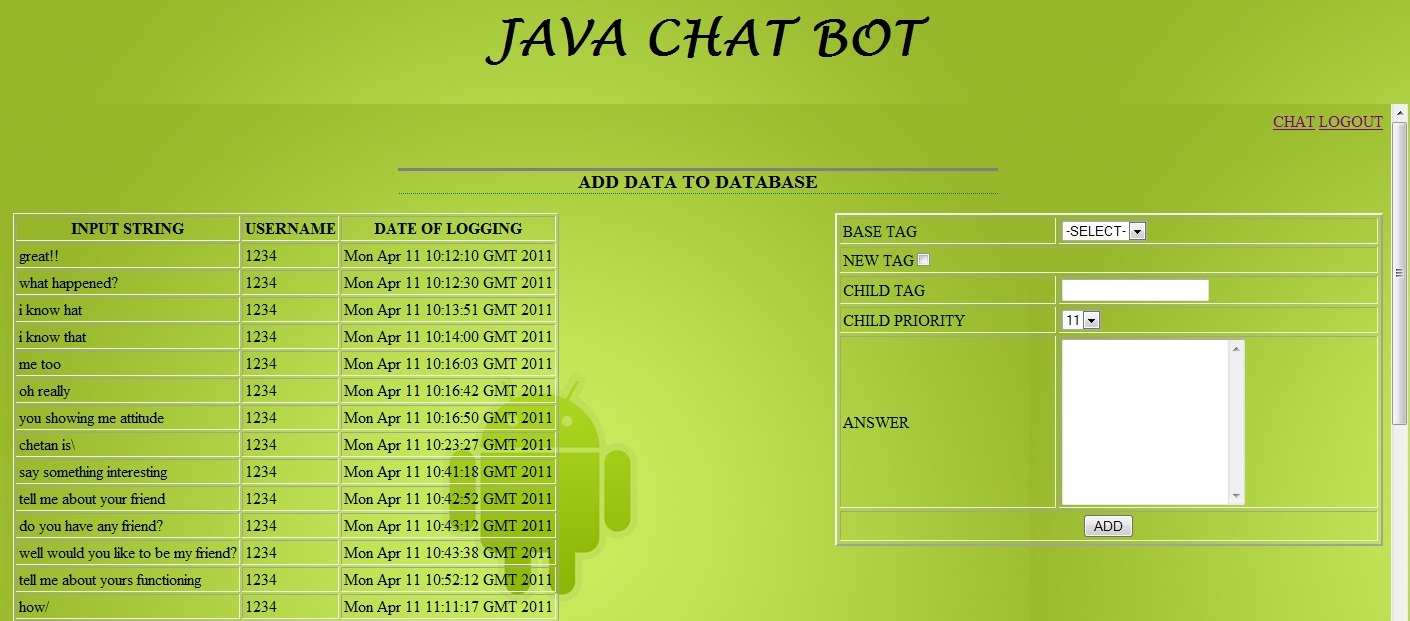
**User Chat Screen:**



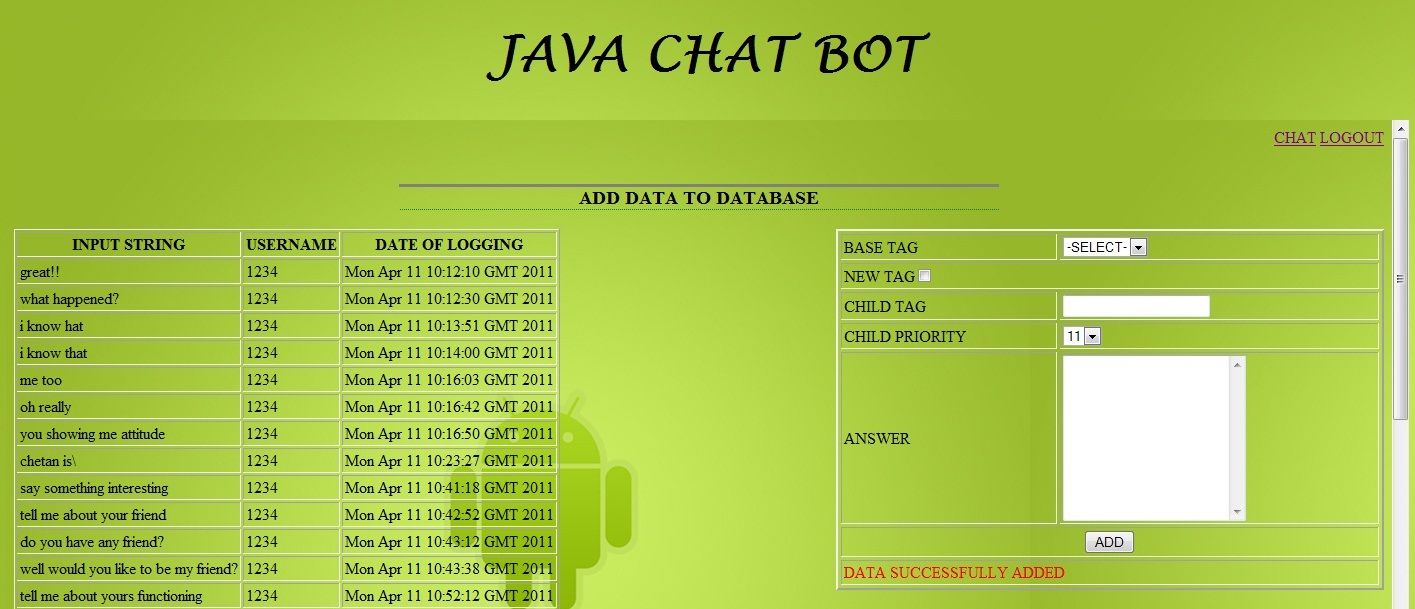
**Admin Chat Screen:**



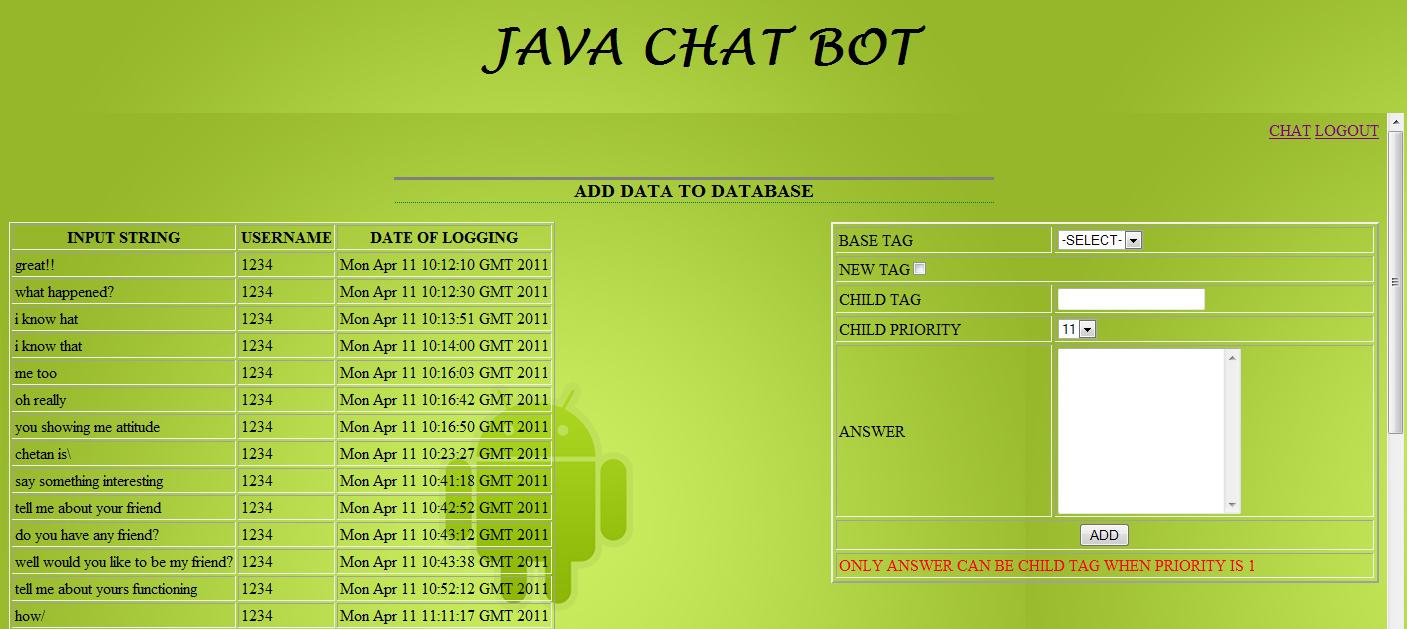
**Admin Screen:**



**Admin Screen(Data added successfully):**



**Admin Screen(Error Message:)**



# EXTERNAL SYSTEM INTERFACE REQUIREMETNS

**NA**

# NON FUNCTIONAL REQUIREMENTS

## Security

A normal level of Role based access control is needed and also the needs as per the product architecture.

## Audit Trail

Audit Trial will capture every action or specifically the chats and the situations in which the bot could not generate an answer to the question posted.

## Error logging

A flat file is generated to store the logs when the system generates an error.Whis is only visible to the admin.

## Multi language Support

System currently supports only English-US as the chatting language. As enhancement the product must support other locales like German, Spanish and French as well.

## Performance

The system is expected to service the requests of 50 users. Around 10 users can access the system concurrently. The product should be scalable to further enhancements and must be able to include changes as per requirements. The modules must be loosely coupled.

## Scalability

The application must cater to the request from 50 users at a time without any performance degradation.

## Availability / reliability

The application is online hence must be available round the clock.

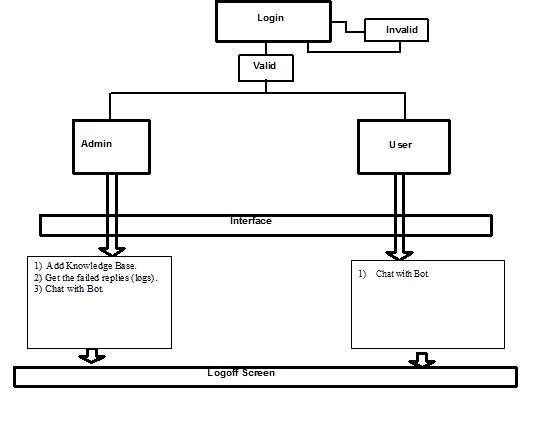
## Data migration

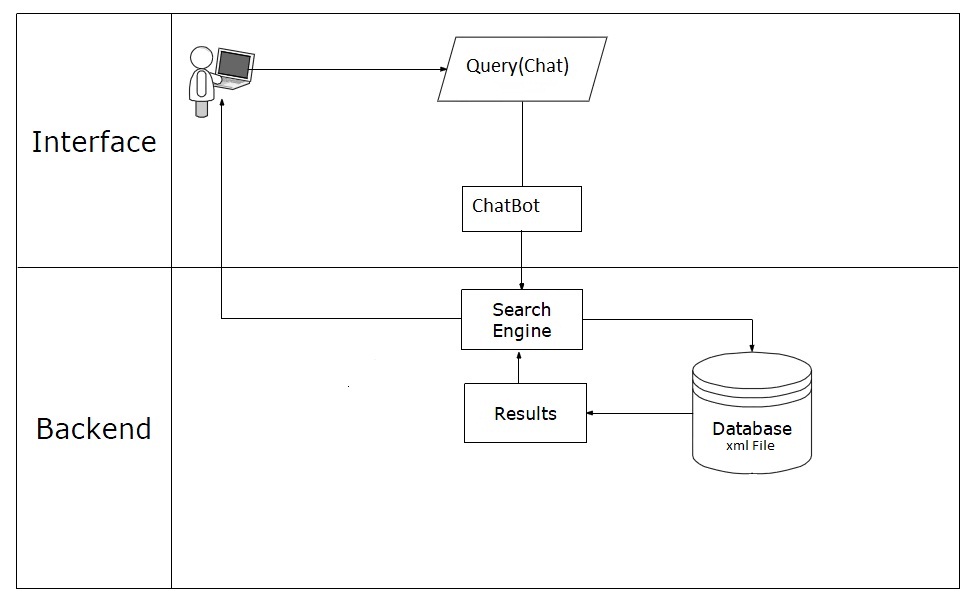
This project will be developed from the scratch and hence no data migration is required.

## Legal/ regulatory requirements

System should take care of personal data protection.

1. **HIGH LEVEL DESIGN ARCHITECTURE**





## Subsystems

**LogIn :** The existing members can login to the system with their username and password which are matched with are stored in a xml file. The members can be either a user or a admin according to their role. If a new user is using the system then the user can signup to the system and the details are stored in the same xml file.

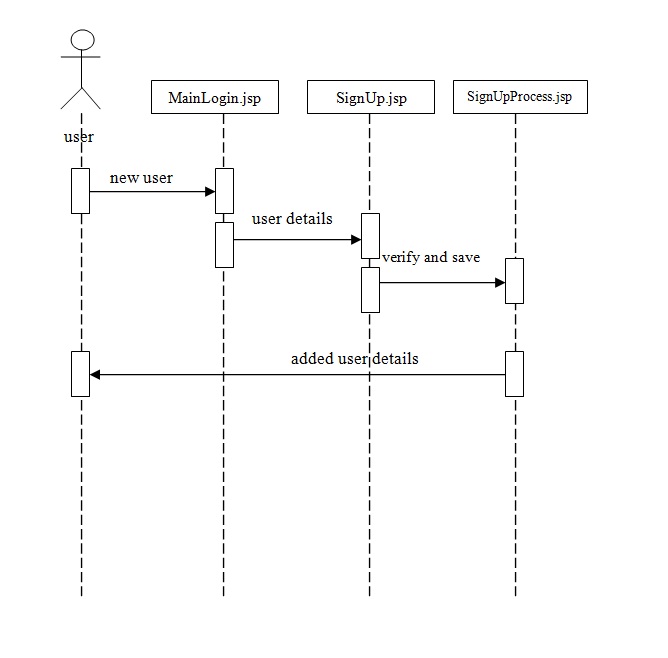
**User :** The user has a console to post his questions and the replies coming from the bot will also be displayed in the same console with proper responses. Whenever the bot is not able to generate a reply a default apology is to be generated replied and the case is logged.

**Admin :** The admin can add Knowledge Base i.e. the admin must be able to improve the knowledge base used to answer a posted question to an XML file and also he will get the failed replies i.e.The admin must get the reports about the failed cases in the form of a CSV file with appropriate fields where the bot couldn’t reply.

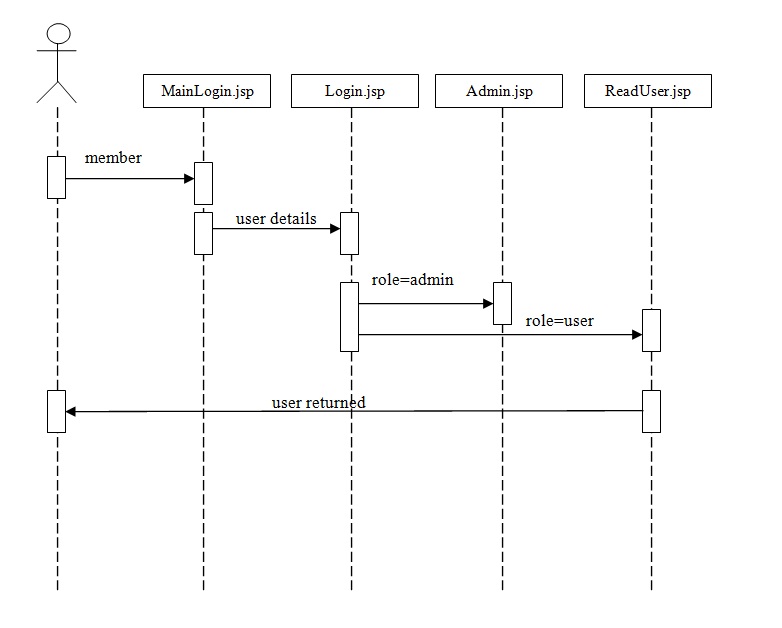
## Modeling

Prototyping Model has been used to develop the system.

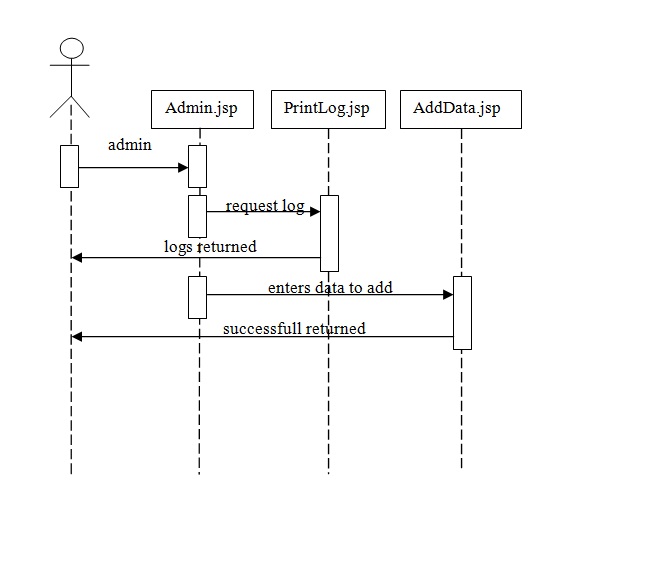
1. **DETAILED DESIGN**
2. **Sequence Diagrams**
3. **Sign Up**

****

1. **Log In**

****

1. **Admin**

****

1. **Parsing**

Input from user

Break the input.

Store in String Array

Keywords

Scan Array for Keywords(Root Nodes).

Found?

Display Output from Random Tag

**NO**

**YES**

Keywords?

Priority?

**MORE THAN ONE**

Select all the tags with different priority.

Select the tag which is aligned last in the list

**SAME DIFF**

**ONLY ONE**

Scan the array for child tags of the Root Node.

Child Tags

Priority?

Display the output of tag which is aligned last

Display the output of tag with higher priority.

**DIFFERENT** **SAME**

## Implementation elements

**LogIn :** The existing members can login to the system with their username and password which are matched with are stored in a xml file. The members can be either a user or a admin according to their role. If a new user is using the system then the user can signup to the system and the details are stored in the same xml file.

**User :** The user has a console to post his questions and the replies coming from the bot will also be displayed in the same console with proper responses. Whenever the bot is not able to generate a reply a default apology is to be generated replied and the case is logged.

**Admin :** The admin can add Knowledge Base i.e. the admin must be able to improve the knowledge base used to answer a posted question to an XML file and also he will get the failed replies i.e.

the admin must get the reports about the failed cases in the form of a CSV file with appropriate fields where the bot couldn’t reply.

## Design of algorithms

There are 2 algorithms which have been used.

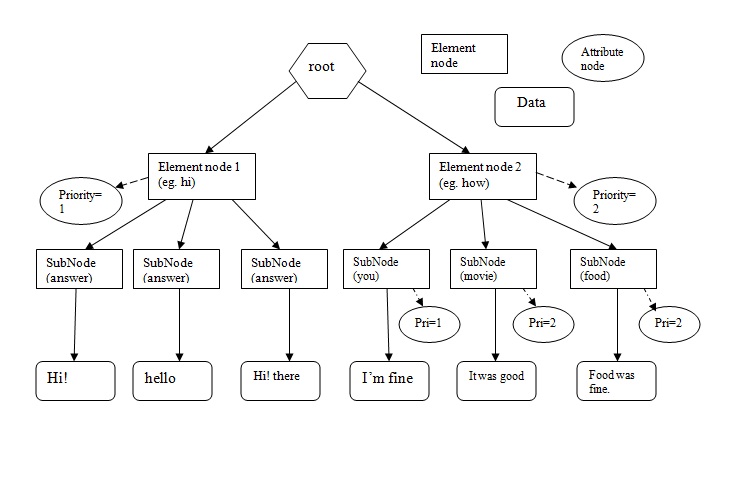
1) **Searching** : This algorithm takes the user input breaks it to individual alphabets according to spaces and stores it in a string. Them compares the alphabets to previously stored list which contains the root nodes of the UserData.xml file. If not found then generates a random output from the same file. If found then stores the child node of the same root node in another list and then matches the remaining alphabets with the list. If there is only one child element then it displays the output corresponding it, else it compares the priority of the child element and displays the output of the child node with the maximum priority.

2) **Insert** : This algorithm is used to append data to the already existing xml file UserData.xml. The administrator views the error logs from the .csv file and the accordingly can add root or child tags as necessary. With it the administrator can also set the priority of the tags that are being added.

Before the new tag is added it is validated to be already present in the database or not. If the tag is already present it displays a error message and if not the a success message is returned to the administrator.

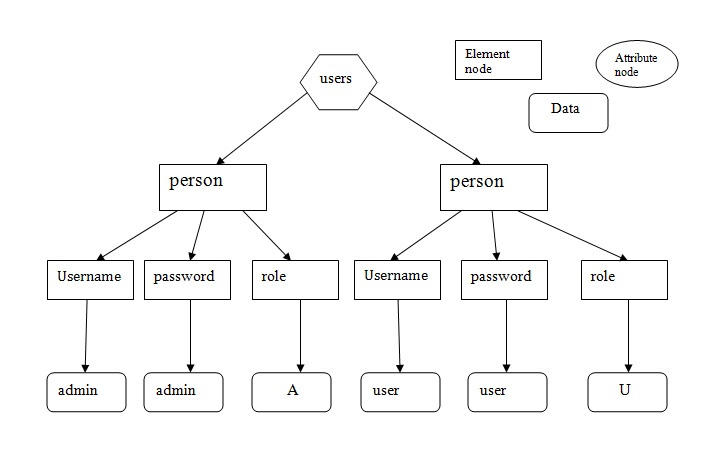
## Database design

The database has been made in a XML file. It has two files one is BotData.xml and the other is UserData.xml. There structure is as shown below.



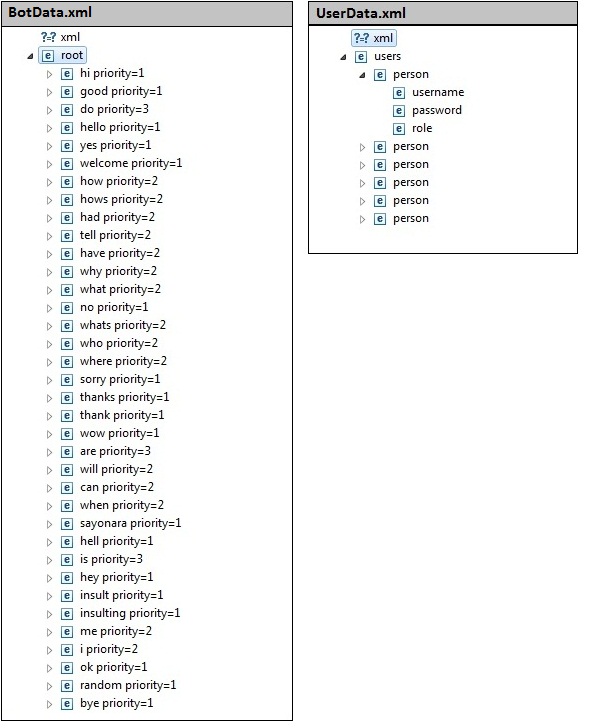
**BotData.xml**

It holds the entire database through which the bot understands the users queries and holds their responses. It is a dynamic file which can be edited by the Admin

****

**UserData.xml**

It holds the username and his corresponding password with the users role which can either be ‘U’(user) or ‘A’(admin).

****

## Optimization of algorithms and data access

The use of DOM parser is both boon and a bit troublesome.

DOM parser is helpful in files that change dynamically but the system processing gets slower when the database size grows.

## Screen field validations, defaults and attribute tables

Validations have been used in 2 phases one on client side through javascript and the other on server side through jsp.

## Error messages

The system undergoes two validation phases one on the client side which is implemented by using javascript and other is server side validation which is done in jsp pages. In both these validation phases whenever the user or admin generates an error an appropriate error message is displayed corresponding to the error.

**Client Side Validations** **:**

1. **LogIn** : If the user directly presses the submit button leaving the username and password field blank then an error message pops up “UserName is Must” directing the user not to leave these fields blank.
2. **SignUp** : In the SignUp page if the user clicks the submit button before filling the entries of the form then the user is prompted with a message “Field can’t be left blank” directing the user not to leave the form blank.
3. **Admin** : If the admin tries to input a new data to the database and misses any field then he is correspondingly prompted to fill the field.

**Server Side Validation :**

1. **LogIn** : When a user logs in if the username and password don’t match or if the role of the user dosen’t match with his username then an error message is displayed to the user showing the failure.
2. **SignUp** : When a new User sighup then if his username matches with any of the user name already present in the database the user is displayed an error message showing that the user already exists.
3. **Admin** : When the admin tries to enter a child tag that already exists for a root tag the system displays a message telling the admin that the tag already exists.

## Special processing notes

### Assumptions

* The user dosen’t use slangs. Proper english words are required to communicate.
* While entering data in database it is assumed that the admin has proper knowledge of the structure of the database.
* When two queries are connected together they should be separated with proper spaces.
* If any absurd input or any input that bot cannot find in its database a random answer for that particular input.
* Bot can only reply correctly only for simple queries.

### Limits

* The user can’t have an intelligent conversation with the bot.
* The bot can’t save any of the user inputs.
* The user can’t change his username and password on the server data.

### Exception handling

* **IOException**

Message:-Some Exception Occurred

* **FileNotFoundException**

Message:-Sorry! No Log File For Current Month Is there

* **NullPointerException**

Message:-Some Exception Occured

* **SAXException**(Hnadled By Parent Class Exception)

Message:-Some Exception Occured

* **KeyWordsNotFound**

Message:-No Particular Message for this exception is defined. A random reply of bot to user has been generated when this exception is raised.

# HARDWARE AND SOFTWARE REQUIREMENTS

## 9.1 Deployment Environment Requirements

## 

**Hardware requirements**

Processor/RAM/HDD : Athlon64 Dual Core 5200+ 2.60GHz / 2GB / 150GB

Web server : Tomcat 6.0

Database Server : NA

**Software requirements**

OS for Web server : Tomcat 6.0

OS for Database Server : NA

DBMS : NA

Third Party S/W’s : JDK 1.6 or above.

## Development Environment Requirements

IDE : Eclipse Ganymede or higher version.

Processor/RAM/HDD : Athlon64 Dual Core 5200+ 2.60GHz / 2GB / 80GB

# 

# Summary

We have created a system that enables the bot to conduct a full and maximally logic conversation with a human user, utilizing rich vocabulary of words. Wide knowledge base written in XML was included, providing a spacious range of information to be talked about with the bot. The core of bot’s essence (personality, character, etc.) was also stored in XML-written file, providing an easy access for the administrator. All in all, makes it a friendly robot, with a wide knowledge base, that can conduct a conversation almost with any human being, talking to him about subjects as one wishes.

# REFERENCES

1. http://alice.pandorabots.com/
2. http://www.ai-programming.com/
3. http://www.alicebot.org/downloads/programs.html