Live-Chat in HealthCare Domain

A COURSE PROJECT REPORT

By

Tarun Prasad (RA2011003011172)
Dishita Sibal (RA2011003011162)
Nitin Kumar (RA2011003011143)
Lakshay Vijay(RA2011003011157)

Under the guidance of

Rajalakshmi M

In partial fulfilment for the Course

of

18CSC302J - COMPUTER NETWORKS

in C-Tech

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY

Kattankulathur, Chenpalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report "**File-Server System**" is the bonafide work of **Tarun Prasad (RA2011003011172), Dishita Sibal (RA2011003011162), Nitin Kumar (RA2011003011147) and Lakshay Vijay (RA2011003011157)** who carried out the project work under my supervision.

SIGNATURE

Rajalakshmi M
Assistant Professor
Computer Networking
SRM Institute of Science and Technology

ABSTRACT

A program has to be designed for a small business organization. The organization hosts a File Transfer Server which is accessible to internet users using TCP/IP and FTP with IP address and Port number.

A network for the same was designed using Socket Programming. The requirements were emulated and tested for connectivity. A server was setup, which is accessible on port 1212 with HTTPS connectivity.

The client is connected to the server where it can upload files to the server and the other client of the organization will be able to download the required files.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty Rajalakshmi M , for her assistance, timely suggestion and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project.
Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

CHAPTERS	CONTENTS
1.	ABSTRACT
2.	INTRODUCTION
3.	LITERATURE SURVEY
4.	ARCHITECTURE & IMPLEMENTATION
5.	RESULTS & ANALYSIS
6.	CONCLUSION
7.	REFERENCES

INTRODUCTION

Research says 60% of visits to a doctors are for simple small-scale diseases, 80% of which can be cured at home using simple home remedies.

These diseases mostly include common cold and cough, headache, abdominal pains, etc. They may be caused due to the changes in the weather, intake of improper diet, fatigue, etc. and can be cured without visiting a doctor all the way to the clinic.

There are a number of live-chats which provide services for the healthcare domain. Work is being carried out to enable the patient to communicate in a way similar to the communication carried out between two humans in physical mode.

LITERATURE SURVEY

S.NO	Paper Title	Summary	Methodology / Algorithm used	Limitations
1	**A direct approach for word and character segmentation in run-length compressed documents with an application to word spotting** Year: 2015 Publisher : ICDAR	The paper proposes a method for text segmentation directly in run-length compressed, printed English text documents. Line segmentation is done using the projection profile technique. Further segmentation into words and characters is accomplished by tracing the white runs along the base region of the text line.	Calculating all character spaces	Text segmentation in compressed documents warrants decompression, and needs additional computing resources.
2	**Joint Optimization for Chinese POS Tagging and Dependency Parsing** Year: 2014 Publisher: IEEE/ACM	This paper proposes a solution by jointly optimizing POS tagging and dependency parsing in a unique model. We propose for our joint models several dynamic programming based decoding algorithms which can incorporate rich POS tagging and syntactic features.	Using a graph data structure for the implementation of the parser	Chinese POS tagging has proven to be much more challenging than morphology-richer languages such as English (94% vs. 97% on POS tagging accuracy). This leads to severe error propagation.
3	**Part-of-Speech Tagging by Latent Analogy** Year: 2010 Publisher: IEEE	This paper explores an alternative tagging strategy based on the principle of latent analogy, which was originally introduced in the context of a speech synthesis application. In this approach, locally optimal tag subsequences emerge automatically from an appropriate representation of global sentence-level information.	Using the latent analogy algorithm	High-accuracy taggers (e.g., based on conditional random fields) rely on well chosen feature functions to ensure that important characteristics of the empirical training distribution are reflected in the trained model. This makes them vulnerable to any discrepancy between training and tagging corpora.
4	**Implementation of robot journalism by programming custombot using tokenization and custom tagging** Year: 2017 Publisher: ICACT	The paper introduces a prototype of an algorithm that creates personalized news articles about IT and technology based on each personal preference for a specific theme, criteria, or element.	Using NLTK package which involves inbuilt tokenizer	While processing and analyzing data by inductive reasoning, CustomBot considers the concepts of news angle and filter bubble.
5	**Joint POS Tagging and Dependency Parsing With Transition-Based Neural Networks** Year: 2017 Publisher: IEEE/ACM	In this paper, we propose an approach to joint POS tagging and dependency parsing using transition-based neural networks. Three neural network based classifiers are designed to resolve shift/reduce, tagging, and labeling conflicts.	Using neural network algorithm	While part-of-speech (POS) tagging and dependency parsing are observed to be closely related, existing work on joint modeling with manually crafted feature templates suffers from the feature sparsity and incompleteness problems.

REQUIREMENTS

1.1 Requirement Analysis

From the given scenario, we draw the following requirements:

1. Identifying the appropriate hardware which would be used (Cisco Packet Tracer)
2. Users on the internet should be able to access only https on the e-commerce server.
3. Users on the internet should have access only to the public IP address of the server and not the private IP address.
4. The users in the organization should have full access to the server.
5. TCP/IP Network design with IP addressing
6. Features and configuration required on the hardware with explanation

We need to configure a network design keeping the following requirements in mind.

1.2 Hardware Requirement

1. Processor - I3/Intel Processor
2. Ram - 4GB (min)
3. Hard Disk -160 GB

1. 3 Software System Configuration

1.Operating System : Windows 7/8/10
2. Application Server : Tomcat 9.0
3. Front End : HTML , JSP
4. Scripts : JavaScript
5. Server side Script : Java Server Pages
6. Database : My SQL 6.0
7. Database Connectivity : JDBC

ARCHITECTURE AND DESIGN

DOCTOR

```c
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
//headers for socket and related functions
#include <sys/types.h>
#include <sys/socket.h>
//for including structures which will store information needed
#include <netinet/in.h>
#include <unistd.h>
//for gethostbyname
#include "netdb.h"
#include "arpa/inet.h"
int main()
{
int socketDescriptor;
struct sockaddr_in serverAddress;
char sendBuffer[8000],recvBuffer[8000];
pid_t cpid;
bzero(&serverAddress,sizeof(serverAddress));
serverAddress.sin_family=AF_INET;
serverAddress.sin_addr.s_addr=inet_addr("127.0.0.1");
serverAddress.sin_port=htons(8080);
/*Creating a socket, assigning IP address and port number for that socket*/
socketDescriptor=socket(AF_INET,SOCK_STREAM,0);
/*Connect establishes connection with the server using server IP address*/
connect(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));
printf("\nYour patient is here.\n");
/*Fork is used to create a new process*/
cpid=fork();
if(cpid==0)
{
while(1)
{
bzero(&sendBuffer,sizeof(sendBuffer));
//printf("\nYour patient is here.\n");
/*This function is used to read from server*/
fgets(sendBuffer,80000,stdin);
```

```c
/*Send the message to server*/
send(socketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
//printf("\nMessage sent !\n");
}
}
else
{
while(1)
{
bzero(&recvBuffer,sizeof(recvBuffer));
/*Receive the message from server*/
recv(socketDescriptor,recvBuffer,sizeof(recvBuffer),0);
printf("\nPATIENT: %s\n",recvBuffer);
}
}
return 0;
}
```

PATIENT

```c
#include<sys/types.h>
#include<sys/socket.h>
#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>
int main(int argc,char *argv[])
{
int clientSocketDescriptor,socketDescriptor;
struct sockaddr_in serverAddress,clientAddress;
socklen_t clientLength;
char recvBuffer[8000],sendBuffer[8000];
pid_t cpid;
bzero(&serverAddress,sizeof(serverAddress));
/*Socket address structure*/
serverAddress.sin_family=AF_INET;
serverAddress.sin_addr.s_addr=htonl(INADDR_ANY);
serverAddress.sin_port=htons(8080);
/*TCP socket is created, an Internet socket address structure is filled with
wildcard address & server's well known port*/
socketDescriptor=socket(AF_INET,SOCK_STREAM,0);
/*Bind function assigns a local protocol address to the socket*/
bind(socketDescriptor,(struct sockaddr*)&serverAddress,sizeof(serverAddress));
/*Listen function specifies the maximum number of connections that kernel should queue
for this socket*/
listen(socketDescriptor,5);
printf("%s\n","Your HealthBot Doctor is available. Please type your concers here...\n");
/*The server to return the next completed connection from the front of the
completed connection Queue calls it*/
clientSocketDescriptor=accept(socketDescriptor,(struct
sockaddr*)&clientAddress,&clientLength);
/*Fork system call is used to create a new process*/
cpid=fork();
if(cpid==0)
{
```

```c
while(1)
{
bzero(&recvBuffer,sizeof(recvBuffer));
/*Receiving the request from client*/
recv(clientSocketDescriptor,recvBuffer,sizeof(recvBuffer),0);
printf("\nHEALTHBOT: %s\n",recvBuffer);
}
}
else
{
while(1)
{
bzero(&sendBuffer,sizeof(sendBuffer));
//printf("\nPlease type your concerns here... ");
/*Read the message from client*/
fgets(sendBuffer,80000,stdin);
/*Sends the message to client*/
send(clientSocketDescriptor,sendBuffer,strlen(sendBuffer)+1,0);
//printf("\nMessage sent !\n");
}
}
return 0;
}
```

The source code for both doctor and patient as server and client are displayed.

The patient calls for service and a doctor is connected to him through simple concept of computer networks which uses full duplex chat system using TCP/IP.

IMPLEMENTATION

To implement a TCP/IP day time server (concurrent server) that handles multiple client requests. Once the client establishes connection with the server, the server sends its day-time details to the client which the client prints in its console.

CODE:

Server :

```
#include<netinet/in.h>
#include<sys/socket.h>
#include<stdio.h>
#include<string.h>
#include<time.h>
#include<stdlib.h>
int main()
{
struct sockaddr_in sa;
struct sockaddr_in cli;
int sockfd,conntfd;
int len,ch;
char str[100];
time_t tick;
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
{
printf("error in socket\n");
exit(0);
```

```
}
else
printf("Socket opened");
bzero(&sa,sizeof(sa));
sa.sin_port=htons(5600);
sa.sin_addr.s_addr=htonl(0);
if(bind(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0)
{
printf("Error in binding\n");
}
else
printf("Binded Successfully");
listen(sockfd,50);
for(;;)
{
len=sizeof(ch);
conntfd=accept(sockfd,(struct sockaddr*)&cli,&len);
printf("Accepted");
tick=time(NULL);
snprintf(str,sizeof(str),"%s",ctime(&tick));
printf("%s",str);write(conntfd,str,100);
}
}
```

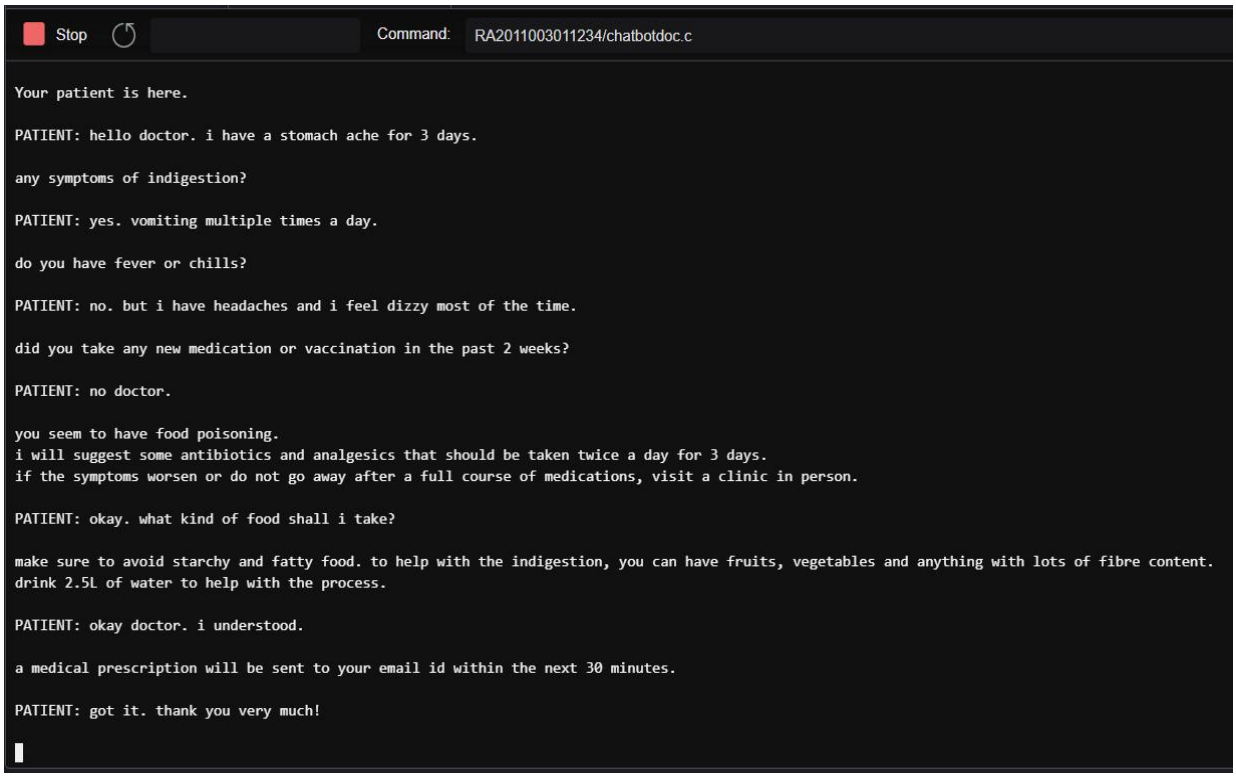
Client :

```
#include <netinet/in.h>
#include <sys/socket.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
```

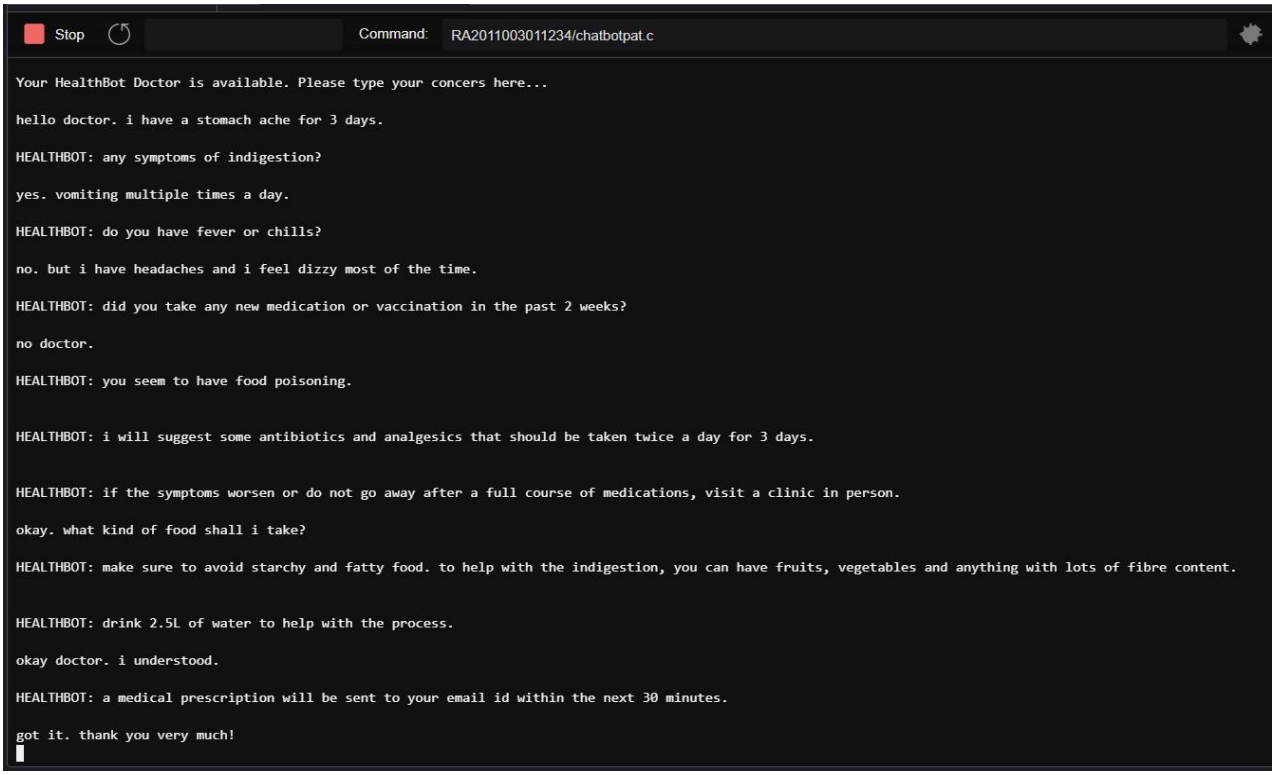
```
struct sockaddr_in sa,cli;
int n,sockfd;
int len;char buff[100];
sockfd=socket(AF_INET,SOCK_STREAM,0);
if(sockfd<0)
{
 printf("\nError in Socket");
exit(0); }
else
printf("\nSocket is Opened");
bzero(&sa,sizeof(sa));
sa.sin_family=AF_INET;
sa.sin_port=htons(5600);
if(connect(sockfd,(struct sockaddr*)&sa,sizeof(sa))<0)
{
printf("\nError in connection failed");
exit(0);
}
else
printf("\nconnected successfully");
if(n=read(sockfd,buff,sizeof(buff))<0)
{
printf("\nError in Reading");
exit(0);
}
else
{
printf("\nMessage Read %s",buff);
}
}
```

RESULTS AND DISCUSSION

DOCTOR



PATIENT



The Live-chat between a doctor and a patient online on a socket network is represented

CONCLUSION AND FUTURE ENHANCEMENT

A huge amount of cost and time for patients can be reduced by implementing a Live-Chat system with trained and verified doctors on your phone, laptops, etc.

And to get the response on the question asked, the patient doesn't even require to strain a muscle to visit a hospital or a clinic , he can just connect with a doctor online.

This would help patient avoid long waiting lines for mildest of issues and consultation and also would ultimately benefit doctors to save their time for more complex and important health-related cases.

REFERENCES

I. https://ieeexplore.ieee.org/document/7333755

II. https://dl.acm.org/doi/10.1109/TASLP.2013.2288081

III. https://ieeexplore.ieee.org/document/5570877

IV. https://ieeexplore.ieee.org/document/7890154

V. https://dl.acm.org/doi/10.1109/TASLP.2017.2788181