*A*

*Mini Project Report Of*

**CHAT APPLICATION**

**Submitted By: -**

**Arun Kumar & Ravi Ranjan Kumar**

**2018PGCACA47**

**2018PGCACA40**



**Under Supervision of**

Mr. Chandrashekhar Azad

**DEPARTMENT OF COMPUTER APPLICATIONS**

**NIT JAMSHEDPUR – 831014**

**November, 2018**

**ACKNOWLEDGEMENT**

I am highly indebted *Dr. Chandrashekhar Azad* for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

Finally, I am indebted to all my team members who have contributed in this report work at National Institute of Technology, Jamshedpur.

**Arun Kumar**

**Ravi Ranjan Kumar**

**MCA 1st Semester**

**Contents**

|  |  |
| --- | --- |
| 1. **Introduction** |  |
| * 1. Introduction to Project |  |
| * 1. Project Category (Client Server) |  |
| * 1. Objective |  |
| * 1. Existing System |  |
| * 1. Proposed System |  |
| * 1. Features of System |  |
| 1. **Requirement Analysis and System Specification** |  |
| * 1. Feasibility Study |  |
| * 1. Software Requirement and Specifications Document |  |
| * + 1. Data Requirements |  |
| * + 1. Functional Documents |  |
| * + 1. Hardware Requirements |  |
| * 1. Validations |  |
| * 1. SDLC Model Used |  |
| 1. **System Design** |  |
| * 1. System Design |  |
| * + 1. ER Diagrams     2. Data Flow Diagrams |  |
| 1. **Implementation** |  |
| * 1. Introduction to Languages, IDE’s Tools and Technologies Used for Implementation |  |
| * 1. Test Plan and Test Activities |  |
| * + 1. Test Plan |  |
| * + 1. Test Activities |  |
| 1. **Results and Discussions** |  |
| * 1. Snapshots of System |  |
| * + 1. Client Application     2. Server Application     3. Back End Representation  1. **Conclusions and Future Scope**    1. Future Scope    2. Conclusions 2. **References** |  |

**Chapter 1**

**Introduction**

**1.1 Introduction to Project**

Teleconferencing or Chatting, is a method of using technology to bring people and ideas “together” despite of the geographical barriers. The technology has been available for years but its acceptance was quite recent. The project is an example of a chat application. It is made up of two interfaces the client interface and server interface, which runs on any computer on the network. To start chatting client should get connected to each other where they can exchange text messages as chatting.

**1.2 Project Category (Client Server Networking)**

Chat Application is a type of networking project working on the client server model. Here the client is the user who chats with the other users and server is the admin interface where the all the data of the users is stored.

In Computer science, client-server is a software architecture model consisting of two parts, client systems and server systems, both communicating over a computer network or on the same computer. A client-server application is a distributed system made up of both client and server software. Client server application provide a better way to share the workload. The client process always initiates a connection to the server, while the server process always waits for requests from any client. When both the client process and server process are running on the same computer, this is called a single seat setup.

**1.3 Objective**

The proposed system’s objectives are to overcome all the limitations and drawbacks of the existing system. The online chat application is user-friendly C application. The main objective of the application is its simplicity of design and ease of implementation.

The main objectives of the proposed system can be enumerated as follows:

* To exchange text messages.
* Converse with family and friends.
* Communicate within the organization.
* Leisure activities.

**1.4 Existing System**

After a thorough study of the existing system, these are the problems discovered.

* People have to communicate manually or through letters.
* Letters are inefficient and insecure.
* Most of the existing applications work online which may prove insecure.
* Most of the existing chat applications do not allow transfer of big messages.
* Some existing chat applications have slightly complex interface.

**1.5 Proposed System**

These are the advantages of our proposed system over the existing system.

* People can communicate over the local network.
* High levels of security as only two terminals are involved in the chat.
* No limit on the size of text to be send.
* Simple and easy to use interface.

**1.6 Features of System**

The unique features of this application are as follow:

* Used for chatting over Local Area Network.
* Can be used for sharing information within an organization.
* Reliable mode of communication.
* No dedicated server is required over Local Area Network.
* Basically, this application can be used for communication purpose anywhere.

**Chapter 2**

**Requirement Analysis and System Specifications**

**2.1 Feasibility Study**

Depending on the results of the initial investigation, the survey is expanded to a more detailed feasibility study. Feasibility study is a test of system proposal ac- cording to its work ability, impact on the organization, ability to meet user needs, and effective use of resources. The objective for this phase is not to solve the problem but to acquire a sense of scope. During the study, the problem definition is crystallized and aspects of the problem to be included in the system are determined.

The three main points, kept in mind at the time of project, are:

* **Possible** (To build it with the given technology and resources)
* **Affordable** (given the time and cost constraints of the organization)
* **Acceptable** (for use by the eventual users of the system

The three major areas to be considered while determining the feasibility of a project are:

1. ***Technical Feasibility:*** The technical issue usually raised during the feasibility stage of the investigation includes the following:

Does the necessary technology exist to do what is suggested?

Do the proposed equipment have the technical capacity to hold the data required to use the new system?

Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?

Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of Secure Infrastructure Implementation System. The current system developed is technically feasible. It is a web-based user interface. Thus, it provides an easy access to the users. The databases purpose is to create, establish and maintain a work- flow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hardware requirements for the development of this project are not many and are already available as free as open source. The work for the project is done with the current equipment and existing software technology.

1. ***Operational Feasibility:*** Under this category of service we conduct a study to analysis and deter-mine whether your need can be fulfilled by using a proposed solution. The result of our operational feasibility Study will clearly outline that the solution proposed for your business is operationally workable and conveniently solves your problems under consideration after the proposal is implemented. We would precisely describe how the system will interact with the systems and persons around.
2. ***Timeline Feasibility:*** It is important to understand that a need must be fulfilled when it has to be. Some otherwise feasible and highly desirable projects can become non-feasible due to very restrictive timeline constraints. This fact makes it imperative that milestones are clearly linked to the timeline and projects are well conceived with safe unforeseen margins. We make sure that we strictly follow what has been stated above.

**2.2 Software Requirement Specification Document**

**2.2.1 Data Requirements**

Data requirement is meant to be the data that will be used in our application. Data required in this project is all notices that need to be conveyed to the user. This application also requires the name and department of persons in order to register them and sending notification about updates. So, two main requirements are:

* **User Name**
* **Password**

**2.2.2 Functional Requirements**

In order to make this application functional, we require the following:

1. ***Run Server application***: A user should be able to connect to the server in order to chat with another user online for that the server needs to be initialized by the admin.
2. ***User registration***: Given that a server application has initialized, then the user should be able to register in application. The user must provide user name and password.
3. ***User Login***: Given that a user has registered, then the user should be able to log in to the chat application. The log-in information will be stored on the server and the user should be logged in automatically.
4. ***Dashboard:*** Given that a user is logged in to the mobile application, then the first page that is shown should be the dashboard page. The user should be able to select public chat.
5. ***Selecting a Notice***: User can select public chat to start join a public chat portal and chat with all the users online in the public chat portal.
6. ***Message Alert***: When a new message appears on the screen a notification alert sound occurs to highlight it to the user.

**2.2.3 Hardware Requirements**

* 1.2Ghz 32-bit Processor
* 512 MB RAM
* 100MB Disk Space
* Compatible Keyboard
* Compatible Display

**2.3 Validations**

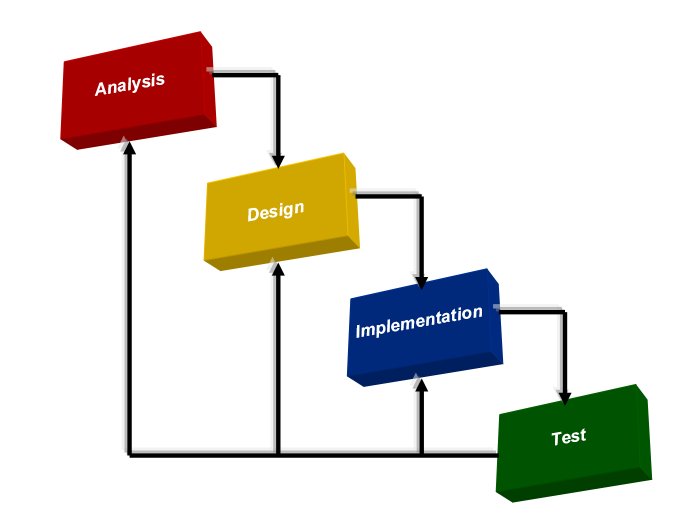
Any application is useless without validation. There should be a way to validate the user input first before sending the user request to the server. Following are the validations implemented in proposed system:

* ***User Password Validation:*** The application should check the user and password fields before sending any request to the server. It should check whether the password contains some specific characters. In this way, there will be less traffic on the server.
* ***Validations during Registration:*** There are a lot of validations that needs to be implemented in the application. They are as follow:

1. ***Username:*** The username can contain only alphabets, digits, underscore, hyphen, @. It should be at least 3 characters long and maximum of 20 characters.
2. ***Password:*** The password can contain only alphabets, digits, underscore, hyphen, @. It should be at least 3 characters long and maximum of 20 characters.

**2.4 SDLC Model Used**

This section describes the project as per the various stages of the Software Development life cycle. The model of software development life cycle used in this project is the waterfall method. The Waterfall Method is comprised of a series of very definite phases, each one run intended to be started sequentially only after the last has been completed, with one or more tangible deliverables produced at the end of each phase of the waterfall method of SDLC. Essentially, it starts with a heavy, documented, requirements-planning phase that outlines all the requirements for the project, followed by sequential phases of design, coding, test-casing, optional documentation, verification (alpha-testing), validation (beta-testing), and finally deployment/release.

****

1. ***Requirement Analysis:*** Existing system is time consuming and it makes difficult to convey huge number of users about any event, class or seminar almost instantly. Also, there is always a big crowd in front of notice board. So, it was hectic to read any useful instruction and information. Thus, all the problems of the existing system are summarized and proposing a new system that works as an online application. It is a value-added solution to the problem. It resolves all the problems stated above. It will provide simple interface to the user to operate on and convey the intended users about events almost instantly, anytime and anywhere.
2. ***Design:*** It includes translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language. The output of the design phase is a design document that acts as an input for all the subsequent SDLC phases. The design of this app is simple and user-friendly containing six main activities, namely:
3. Register
4. Login
5. Dashboard
6. Admin Panel
7. ***Coding/Implementation:*** It includes translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language. The output of the design phase is a design document that acts as an input for all the subsequent SDLC phases. The project is implemented using the Android virtual devise (AVD). This emulator helped to implement the project in a real-like environment and sketch out the details of how it will work on a real hardware. Each activity is linked with another and interconnectivity is transparent and smooth.
8. ***Testing:*** It includes detection of errors in the application. The testing process starts with a test plan that recognizes test-related activities, such as test case generation, testing criteria, and resource allocation for testing. The code is tested and mapped against the design document created in the design phase. The output of the testing phase is a test report containing errors that occurred while testing the application. Testing of the project has not been done on real hardware and also on the emulator or software environment. Testing has been done for each of the individual activities of the project.
9. ***Maintenance:*** It includes implementation of changes that software might undergo over a period of time, or implementation of new requirements after the software is deployed at the customer location. The maintenance phase also includes handling the residual errors that may exist in the software even after the testing phase. The project maintenance is low cost and efficient as user will get this application at free of cost and also this application is shared over network, therefore maintenance is little bit difficult.

**Chapter 3**

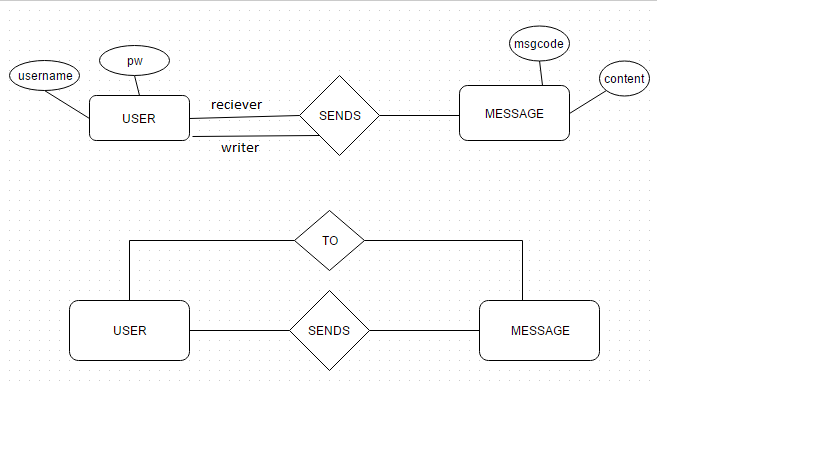
**System Design**

**3.1 System Design**

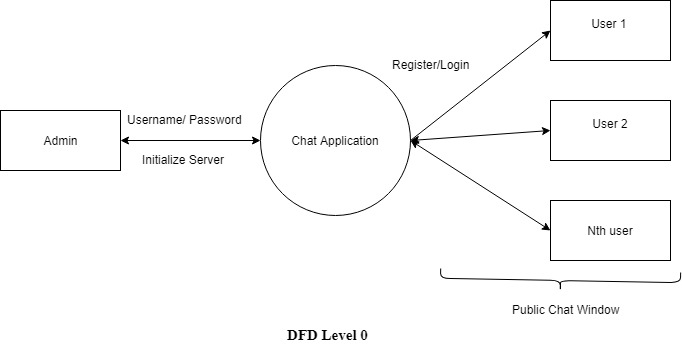
**3.1.1 ER Diagrams**

An ER diagram is a diagram that helps to design databases in an efficient way. Attributes in ER diagrams are usually modeled as an oval with the name of the attribute, linked to the entity or relationship that contains the attribute.

An Entity Relationship Diagram (ERD) is a visual representation of different data using conventions that describe how these data are related to each other. ER diagrams are most often associated with complex databases that are used in software engineering and IT networks. In particular, ER diagrams are frequently used during the design stage of a development process in order to identify different system elements and their relationships with each other. For example, inventory software used in a retail shop will have a database that monitors elements such as purchases, item, item type, and item source and item price.



**3.1.2 Data Flow Diagrams**



**Chapter 4**

**Implementation, Testing and Maintenance**

**4.1 Introduction to Languages, IDE’s, Tools and Technologies Used for Implementation**

The project involves development of two panels namely Admin Panel and a Client Panel. Both of these have been designed using C programming language. The technologies used are as under.

* **Programming languages:**

1. **C Language –** C language is a procedure-oriented programming language. It is developed by Dennis Ritchie between 1969 and 1973. It has been00. used in this project as the main programming language to develop the Chat Application and its front end.
2. **Software Used -**

**CodeBlocks –** CodeBlocks is a cross-platform IDE that supports compiling and running multiple programming languages.

**File Handling –** File Handling is used in the project in the back end to manage the data of users.

**4.2 Test Plan and Activities**

**4.2.1 Test Plan**

A test plan can be defined as a document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. In software testing, a test plan gives detailed testing information regarding an upcoming testing effort, including

* Scope of testing
* Schedule
* Test Deliverables
* Release Criteria
* Risks and Contingencies

It is also be described as a detail of how the testing will proceed, who will do the testing, what will be tested, in how much time the test will take place, and to what quality level the test will be performed.

The process of defining a test project so that it can be properly measured and controlled. The test planning process generates a high-level test plan document that identifies the software items to be tested, the degree of tester independence, the test environment, the test case design and test measurement techniques to be used, and the rationale for their choice.

A testing plan is a methodological and systematic approach to testing a system such as a machine or software. It can be effective in finding errors and flaws in a system. In order to find relevant results, the plan typically contains experiments with a range of operations and values, including an understanding of what the eventual workflow will be.

Test plan is a document which includes introduction, assumptions, list of test cases, and list of features to be tested, approach, deliverables, resources, risks and scheduling. A test plan is a systematic approach to testing a system such as a machine or software. The plan typically contains a detailed understanding of what the eventual workflow will be. A record of the test planning process detailing the degree of tester independence, the test environment, the test case design techniques and test measurement techniques to be used, and the rationale for their choice.

**4.2.2 Test Activities**

Various Testing Activities are as follow:

1. ***Black box testing -*** Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.
2. ***White box testing -*** This testing is based on knowledge of the internal logic of an applications code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.
3. ***Unit testing -*** Testing of individual software components or modules. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses.
4. ***Incremental integration testing -*** Bottom up approach for testing i.e. continuous testing of an application as new functionality is added; Application functionality and modules should be independent enough to test separately. Done by programmers or by testers.
5. ***Integration testing -*** Testing of integrated modules to verify combined functionality after integration. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.
6. ***Functional testing -*** This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.
7. ***System testing -*** Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.
8. ***End-to-end testing -*** Similar to system testing, involves testing of a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.
9. ***Acceptance testing -*** Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer does this testing to determine whether to accept application.

**Chapter 5**

**Results and Discussions**

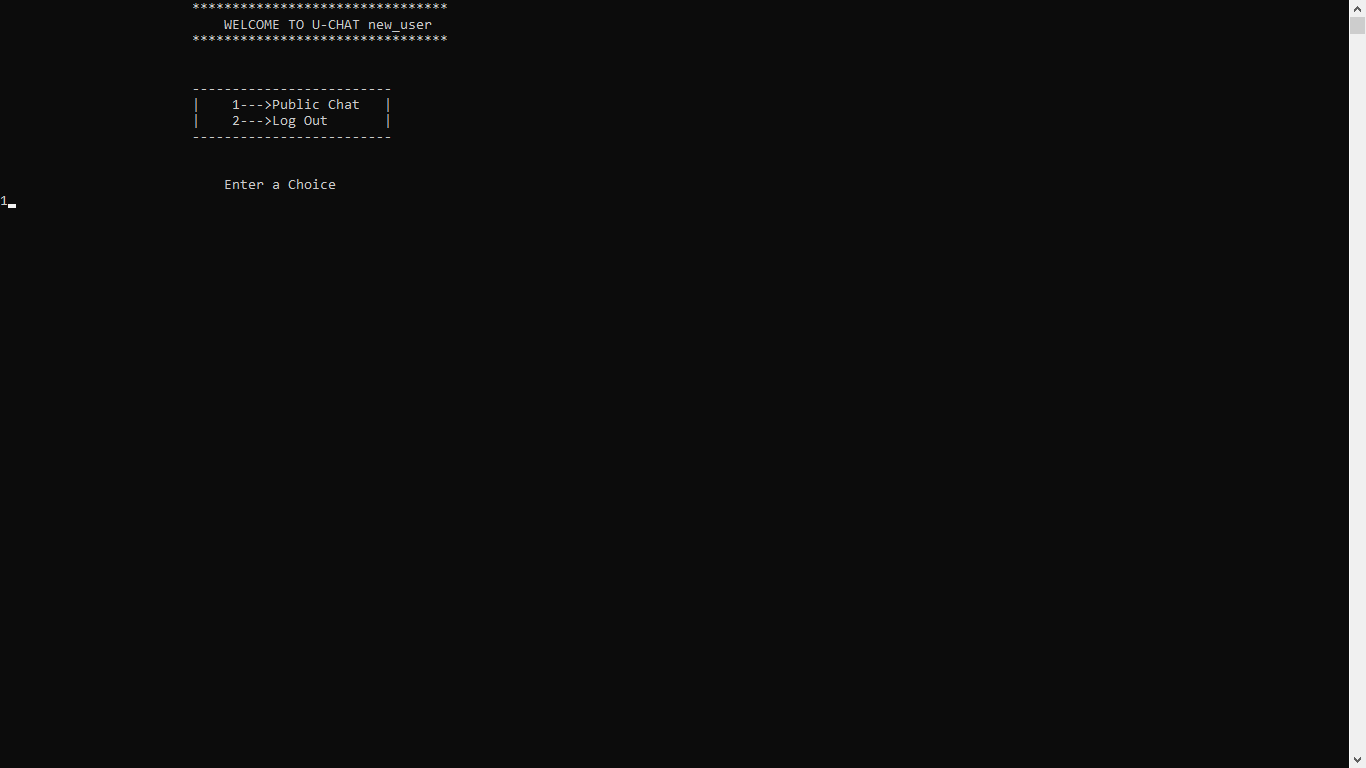
**5.1 Snapshots of the system**

**5.1.1 Client Application**

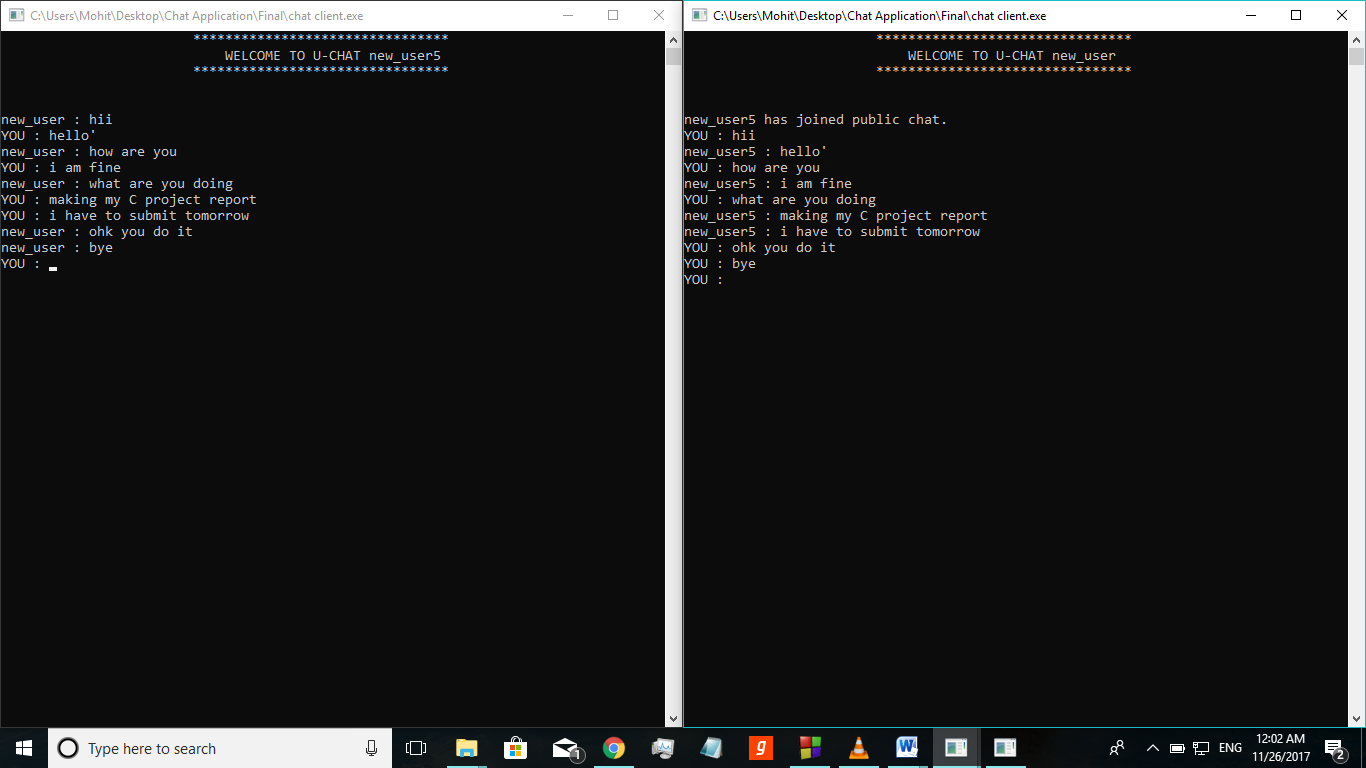
Following are the snapshots of working client application:



***User Login***



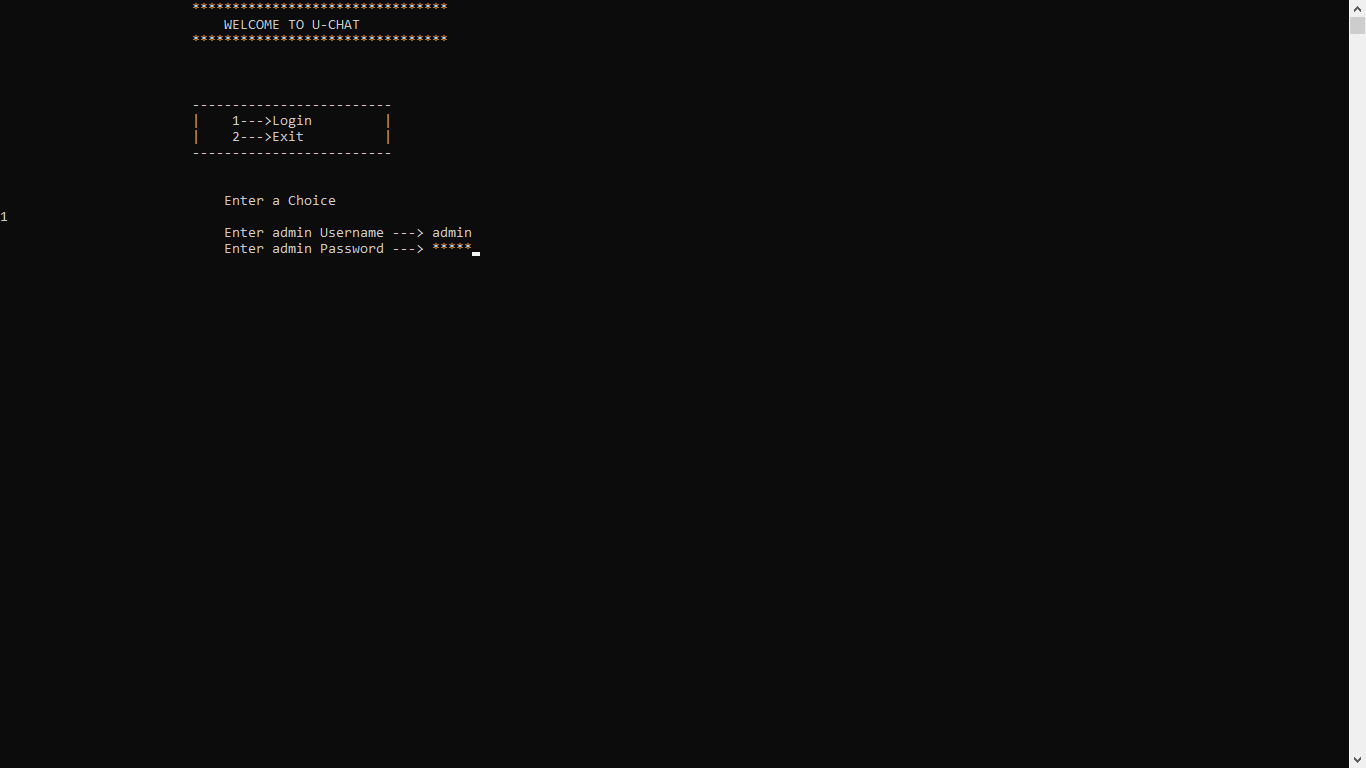
***User Dashboard***



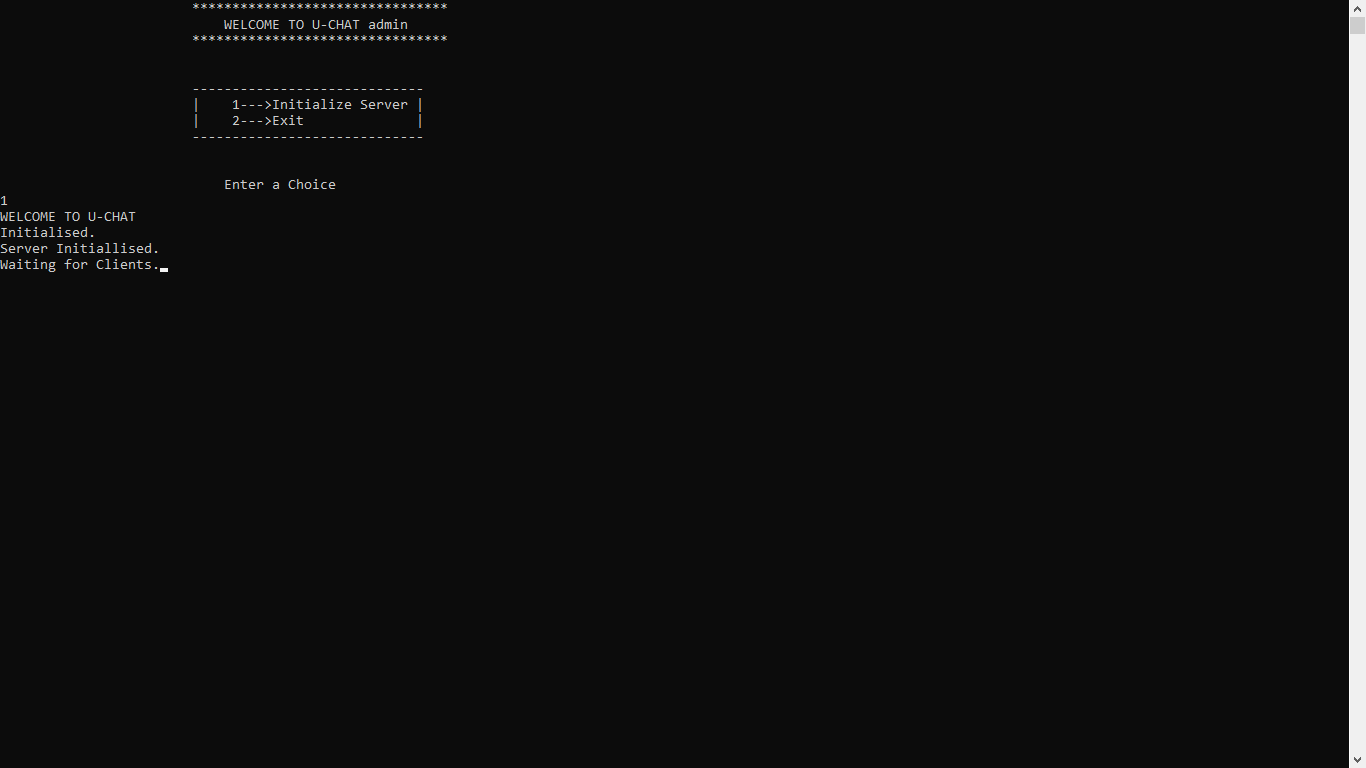
***Chat Between two users***

**5.1.2 Server Application**

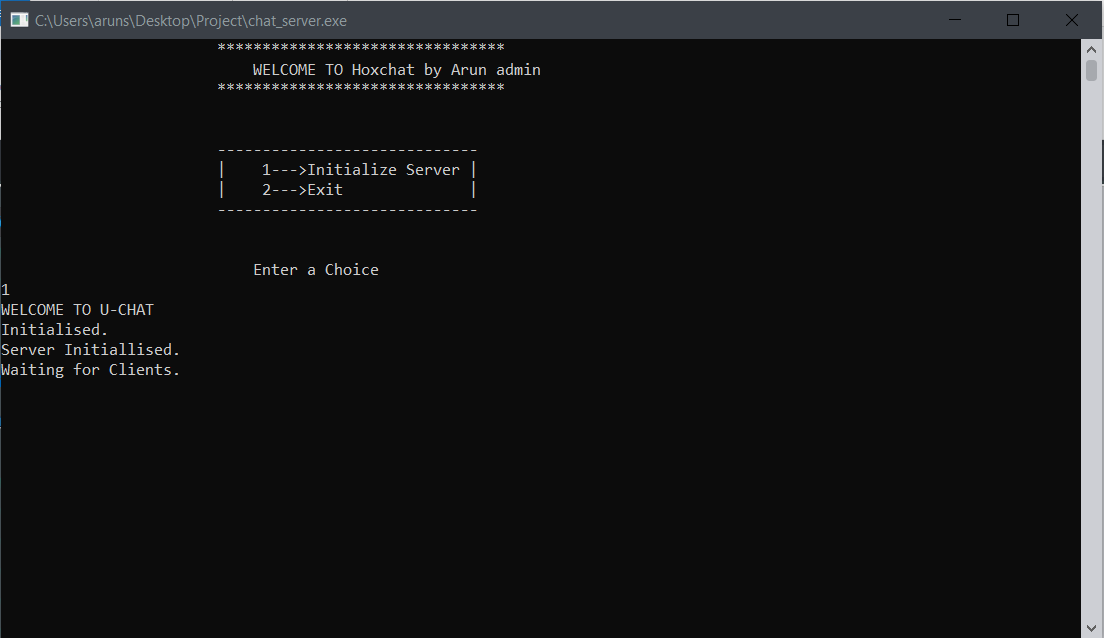
Following are the snapshots of working server application:



***Admin Login***

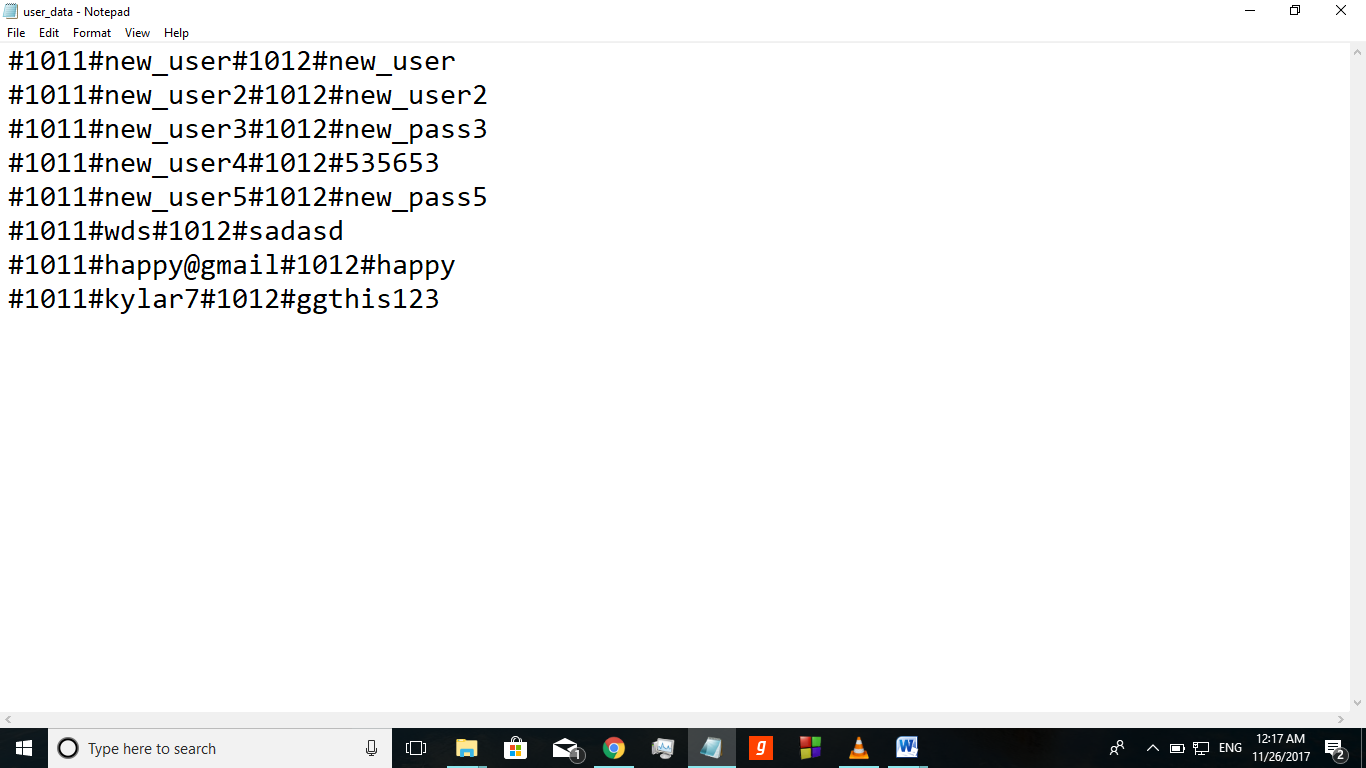


***Server Initialized***



**5.1.3 Back End Representation**

The backend storage is done using file handling. Data is stored as text in notepad.



**Username and password file**

**5.2 Coding**

**5.2.1 Client Application**

#include<dos.h>

#include<stdio.h>

#include<winsock2.h>

#include <stdlib.h>

#include<strings.h>

#include<windows.h>

#define MSG\_SIZE 512

int choice, menu, rcvd\_size, activity;

char user[20]={'\0'};

SOCKET sock;//Network Socket to communicate with the server

struct sockaddr\_in server;

char send\_msg[MSG\_SIZE], \*rcvd\_msg, ip\_server[30], msg[MSG\_SIZE];

DWORD WINAPI ReadingThread(LPVOID param);

int show\_menu();

void home\_page();

void exit\_user();

int register\_user();

int verify\_entry();

int login\_user();

void dashboard();

void public\_chat();

void private\_chat();

void retext();

int main()

{

WSADATA wsa;//Windows Socket Application Data variable

rcvd\_msg = (char\*) malloc((MSG\_SIZE + 1) \* sizeof(char));

printf("WELCOME TO U-CHAT\n");

if (WSAStartup(MAKEWORD(2,2),&wsa) != 0)//Winsock Initialization

{

printf("\nWinsock Initialization Failed. Error Code : %d ",WSAGetLastError());

printf("\nPress any key to exit.");

getch();

return 1;

}

//Create a network socket

if((sock = socket(AF\_INET , SOCK\_STREAM , 0 )) == INVALID\_SOCKET)

{

printf("\nCould not create socket : %d" , WSAGetLastError());

printf("\nPress any key to exit.");

getch();

return 1;

}

server.sin\_family = AF\_INET;//IPv4 Protocol

server.sin\_port = htons( 7777 );//Port Number to be connected

ip\_reconnect://goto label to reconnect to ip address

printf("\nReady to Connect. Enter the IP Address of the server to connect.\n");

scanf("%s",&ip\_server);//Input IP of the server

server.sin\_addr.s\_addr = inet\_addr(ip\_server);//IP of the server to be connected

//Connect to remote server

if (connect(sock , (struct sockaddr \*)&server , sizeof(server)) < 0)

{

printf("\nCould not connect to the IP given : %d" , WSAGetLastError());

printf("\nPress 1 to enter IP of the server again");

printf("\nPress any other key to exit.");

scanf("%d",&choice);

if(choice==1)

goto ip\_reconnect;

getch();

return 1;

}

printf("\nConnected to the Server");

system("cls");

home\_page();

}

int show\_menu()

{

switch(menu)

{

case 0:

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t WELCOME TO U-CHAT %s\n",user);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

break;

case 1:

printf("\t\t\t \n");

printf("\t\t\t-------------------------\n");

printf("\t\t\t| 1--->Login |\n");

printf("\t\t\t| 2--->Register |\n");

printf("\t\t\t| 3--->Exit |\n");

printf("\t\t\t-------------------------\n\n\n");

printf("\t\t\t Enter a Choice\n");

break;

case 2:

printf("\t\t\t-------------------------\n");

printf("\t\t\t| 1--->Public Chat |\n");

printf("\t\t\t| 2--->Log Out |\n");

printf("\t\t\t-------------------------\n\n\n");

printf("\t\t\t Enter a Choice\n");

break;

}

if(menu==0)

choice=100;

else

scanf("%d",&choice);

return choice;

}

void home\_page()

{

menu=0;

show\_menu();

menu=1;

choice=show\_menu();

switch(choice)

{

case 1:

if(!login\_user())

dashboard();

else

break;

case 2:

if(!register\_user())

printf("Registration Successful");

else

printf("Registration Unsuccessful");

home\_page();

break;

case 3:

exit\_user();

break;

default:

home\_page();

}

}

void exit\_user()

{

exit(0);

}

int register\_user()

{

char uname[20];

char pass[20];

char ch;

int i;

re\_uname:

printf("\t\t Only alphabets, digits, '@', '\_', '.' are allowed maximum 20 characters\n");

printf("\t\t\t Enter a Username ---> ");

fflush(stdin);

gets(uname);

if(verify\_entry(uname))

{

printf("\t\t\t INVALID USERNAME. PLEASE TRY AGAIN.\n");

goto re\_uname;

}

re\_pass:

printf("\t\t Only alphabets, digits, '@', '\_', '.' are allowed maximum 20 characters\n");

printf("\t\t\t Enter your Password ---> ");

fflush(stdin);

i=0;

while (1)

{

if (i < 0)

{

i = 0;

}

ch = getch();

if (ch == 13)

break;

if (ch == 8) /\*ASCII value of BACKSPACE\*/

{

printf("\b \b");

i--;

pass[i] = '\0';

continue;

}

pass[i++] = ch;

ch = '\*';

printf("%c",ch);

}

pass[i]='\0';

if(verify\_entry(pass))

{

printf("\t\t\t INVALID PASSWORD. PLEASE TRY AGAIN.\n");

goto re\_pass;

}

strcpy(send\_msg,"#1011#");

strcat(send\_msg,uname);

puts(send\_msg);

if( (send(sock , send\_msg , strlen(send\_msg) , 0) < 0) || ((rcvd\_size = recv(sock , rcvd\_msg , 1 , 0)) == SOCKET\_ERROR))

{

printf("Could not connect to server : %d" , WSAGetLastError());

getch();

printf("\nPress any key to exit.");

return 1;

}

if(rcvd\_msg[0]=='1')

{

printf("\t\t\t USERNAME ALREADY EXISTS. PLEASE TRY AGAIN.\n");

goto re\_uname;

}

strcpy(send\_msg,"#1012#");

strcat(send\_msg,pass);

if( (send(sock , send\_msg , strlen(send\_msg) , 0) < 0) || ((rcvd\_size = recv(sock , rcvd\_msg , 1 , 0)) == SOCKET\_ERROR))

{

printf("Could not connect to server : %d" , WSAGetLastError());

getch();

printf("\nPress any key to exit.");

return 1;

}

return 0;

}

int verify\_entry(char entry[20])

{

int i,x;

if(strlen(entry)>20)

return 1;

for(i=0;i<strlen(entry);i++)

{

x=entry[i];

if((x>=64&&x<=90)||(x>=97&&x<=122)||(x>=48&&x<=57)||(x==46)||(x==95))

continue;

return 1;

}

return 0;

}

int login\_user()

{

char uname[20], pass[20];

int len,i=0;

char ch;

int k;

re\_uname:

printf("\t\t\t Enter a Username ---> ");

fflush(stdin);

gets(uname);

if(verify\_entry(uname))

{

printf("\t\t\t INVALID USERNAME. PLEASE TRY AGAIN.\n");

goto re\_uname;

}

re\_pass:

printf("\t\t\t Enter your Password ---> ");

fflush(stdin);

i=0;

while (1)

{

if (i < 0)

{

i = 0;

}

ch = getch();

if (ch == 13)

break;

if (ch == 8) /\*ASCII value of BACKSPACE\*/

{

printf("\b \b");

i--;

pass[i] = '\0';

continue;

}

//putchar(ch)

pass[i++] = ch;

ch = '\*';

printf("%c",ch);

}

pass[i]='\0';

if(verify\_entry(pass))

{

printf("\t\t\t INVALID PASSWORD. PLEASE TRY AGAIN.\n");

goto re\_pass;

}

strcpy(send\_msg,"#1013#");

strcat(send\_msg,uname);

strcat(send\_msg,"#1012#");

strcat(send\_msg,pass);

if( (send(sock , send\_msg , strlen(send\_msg) , 0) < 0) || ((rcvd\_size = recv(sock , rcvd\_msg , 1 , 0)) == SOCKET\_ERROR))

{

printf("Could not connect to server : %d" , WSAGetLastError());

getch();

printf("\nPress any key to exit.");

return 1;

}

if(rcvd\_msg[0]=='1')

{

printf("\t\t\t USERNAME OR PASSWORD INCORRECT. PLEASE TRY AGAIN.\n");

goto re\_uname;

}

strcpy(user,uname);

return 0;

}

void dashboard()

{

system("cls");

menu=0;

show\_menu();

menu=2;

show\_menu();

switch(choice)

{

case 1:

strcpy(send\_msg,"#1021#");

if( send(sock , send\_msg , strlen(send\_msg) , 0) < 0)

{

printf("Could not connect to server : %d" , WSAGetLastError());

getch();

printf("\nPress any key to exit.");

exit(0);

}

else

public\_chat();

break;

case 2:

strcpy(user,"U-CHAT");

system("cls");

printf("LOG OUT SUCCESSFULL\n");

home\_page();

break;

default:

printf("\t\t\t WRONG CHOICE. PLEASE TRY AGAIN.\n");

}

}

void public\_chat()

{

HANDLE hThread;

DWORD dwThreadID;

system("cls");

menu=0;

show\_menu();

hThread = CreateThread(NULL, 0, &ReadingThread, (void\*)sock, 0, &dwThreadID);

if (!hThread)

{

printf("Could not create thread : %d" , WSAGetLastError());

printf("\nPress any key to exit.");

getch();

exit(EXIT\_FAILURE);

}

while(1)

{

getch();

retext();

printf("YOU : ");

//Send some data

fflush(stdin);

gets(msg);

strcpy(send\_msg,"#1022#");

if(strcmp(msg,"#1001#")==0)

strcpy(send\_msg,"#1023#");

else

strcat(send\_msg,msg);

if( send(sock , send\_msg , strlen(send\_msg) , 0) < 0)

{

printf("Could not connect to server : %d" , WSAGetLastError());

getch();

printf("\nPress any key to exit.");

}

if(strcmp(msg,"#1001#")==0)

break;

}

CloseHandle(hThread);

dashboard();

}

DWORD WINAPI ReadingThread(LPVOID param)

{

SOCKET socks = (SOCKET) param;

int error\_code, i;

fd\_set rcvfds;

while(1)

{

//clear the socket fd set

FD\_ZERO(&rcvfds);

//add master socket to fd set

FD\_SET(socks, &rcvfds);

activity = select( 0 , &rcvfds , NULL , NULL , NULL);

if ( activity == SOCKET\_ERROR )

{

printf("select call failed with error code : %d" , WSAGetLastError());

printf("\nPress any key to exit.");

getch();

exit(EXIT\_FAILURE);

}

if (FD\_ISSET( socks , &rcvfds))

{

rcvd\_size = recv( socks , rcvd\_msg, 512, 0);

if( rcvd\_size == SOCKET\_ERROR)

{

error\_code = WSAGetLastError();

if(error\_code == WSAECONNRESET)

{

//Somebody disconnected , get his details and print

printf("\nConnection Interrupted Unexpectedly %d", error\_code);

printf("\nPress any key to exit.");

getch();

exit(EXIT\_FAILURE);

closesocket( socks );

}

else

{

printf("recv failed with error code : %d" , error\_code);

printf("\nPress any key to exit.");

getch();

exit(EXIT\_FAILURE);

closesocket( socks );

}

}

else if ( rcvd\_size == 0)

{

//Somebody disconnected , get his details and print

printf("\nServer Disconnected %d", error\_code);

printf("\nPress any key to exit.");

getch();

exit(EXIT\_FAILURE);

closesocket( socks );

}

else

{

//adding null character at the end for usage

rcvd\_msg[rcvd\_size] = '\0';

retext();

printf("%s\a\n",rcvd\_msg);

printf("YOU : ");

//printf("YOU : %s",send\_msg);

//strcpy(send\_msg,"\0");

}

}

}

}

void retext()

{

int i;

printf("\r");

for(i=0;i<50;i++)

printf(" ");

for(i=0;i<50;i++)

printf("\b");

strcpy(send\_msg,"\0");

fflush(stdin);

}

**5.2.2 Server Application**

#include<stdio.h>

#include<winsock2.h>

#include <stdlib.h>

#include<strings.h>

#include<windows.h>

#define MSG\_SIZE 512

#define MAX\_CLIENT 30

FILE \*fp;

fd\_set readfds;//set of socket descriptors

char \*rcvd\_msg, send\_msg[MSG\_SIZE], msg[MSG\_SIZE], users[MAX\_CLIENT][20], user[20]={'\0'}, ach[20];

SOCKET master\_socket , new\_socket , client\_socket[MAX\_CLIENT] , sock;

struct sockaddr\_in client, address;

int activity, addrlen, rcvd\_size, public\_chatters[MAX\_CLIENT], tracker, menu;

int show\_menu();

DWORD WINAPI ReadingThread(LPVOID param);

int getHash(char \*hash);

char callHash(int hash);

void openData();

int verify\_entry();

int login\_admin();

int main()

{

WSADATA wsa;

char feed[1];

int choice;

char uname[20], pass[20];

int i;

rcvd\_msg = (char\*) malloc((MSG\_SIZE + 1) \* sizeof(char));

menu=0;

show\_menu();

menu=1;

choice=show\_menu();

do

{

switch(choice)

{

case 1:

login\_admin();

goto cont;

break;

case 2:

exit(0);

break;

default:

printf("\t\t\t WRONG CHOICE. PLEASE TRY AGAIN.\n");

}

}while(1);

cont:

system("cls");

menu=0;

show\_menu();

menu=2;

choice=show\_menu();

do

{

switch(choice)

{

case 1:

goto init;

break;

case 2:

exit(0);

break;

default:

printf("\t\t\t WRONG CHOICE. PLEASE TRY AGAIN.\n");

}

}while(1);

init:

openData();

for(tracker = 0 ; tracker < MAX\_CLIENT;tracker++)

{

client\_socket[tracker] = 0;

public\_chatters[tracker]=0;

}

printf("WELCOME TO U-CHAT\n");

if (WSAStartup(MAKEWORD(2,2),&wsa) != 0)//Winsock Initialization

{

printf("\nWinsock Initialization Failed. Error Code : %d ",WSAGetLastError());

printf("\nPress any key to exit.");

getch();

return 1;

}

printf("Initialised.\n");

//Create a socket

if((master\_socket = socket(AF\_INET , SOCK\_STREAM , 0 )) == INVALID\_SOCKET)

{

printf("\nCould not create socket : %d" , WSAGetLastError());

printf("\nPress any key to exit.");

getch();

return 1;

}

//Prepare the sockaddr\_in structure

client.sin\_family = AF\_INET;

client.sin\_addr.s\_addr = INADDR\_ANY;

client.sin\_port = htons( 7777 );

//Bind

if( bind(master\_socket ,(struct sockaddr \*)&client , sizeof(client)) == SOCKET\_ERROR)

{

printf("Bind failed with error code : %d" , WSAGetLastError());

printf("\nPress any key to exit.");

getch();

return 1;

}

printf("Server Initiallised.\nWaiting for Clients.");

//Listen to incoming connections

listen(master\_socket , 3);

addrlen = sizeof(struct sockaddr\_in);

while(TRUE)

{

//clear the socket fd set

FD\_ZERO(&readfds);

//add master socket to fd set

FD\_SET(master\_socket, &readfds);

//add child sockets to fd set

for ( tracker = 0 ; tracker < MAX\_CLIENT ; tracker++)

{

sock = client\_socket[tracker];

if(sock > 0)

{

FD\_SET( sock , &readfds);

}

}

//wait for an activity on any of the sockets, timeout is NULL , so wait indefinitely

activity = select( 0 , &readfds , NULL , NULL , NULL);

if ( activity == SOCKET\_ERROR )

{

printf("select call failed with error code : %d" , WSAGetLastError());

printf("\nPress any key to exit.");

getch();

return 1;

}

//If something happened on the master socket , then its an incoming connection

if (FD\_ISSET(master\_socket , &readfds))

{

if ((new\_socket = accept(master\_socket , (struct sockaddr \*)&address, (int \*)&addrlen))<0)

{

perror("accept");

printf("\nPress any key to exit.");

getch();

return 1;

}

//inform user of socket number - used in send and receive commands

printf("New connection , socket fd is %d , ip is : %s , port : %d \n" , new\_socket , inet\_ntoa(address.sin\_addr) , ntohs(address.sin\_port));

//add new socket to array of sockets

for (tracker = 0; tracker <MAX\_CLIENT; tracker++)

{

if (client\_socket[tracker] == 0)

{

client\_socket[tracker] = new\_socket;

printf("Adding client to list of clients at index %d \n" , tracker);

break;

}

}

}

//else its some IO operation on some other socket :)

for (tracker = 0; tracker < MAX\_CLIENT; tracker++)

{

sock = client\_socket[tracker];

//if client presend in read sockets

if (FD\_ISSET( sock , &readfds))

{

//get details of the client

getpeername(sock , (struct sockaddr\*)&address , (int\*)&addrlen);

//Check if it was for closing , and also read the incoming message

//recv does not place a null terminator at the end of the string (whilst printf %s assumes there is one).

rcvd\_size = recv( sock , rcvd\_msg, MSG\_SIZE, 0);

if( rcvd\_size == SOCKET\_ERROR)

{

int error\_code = WSAGetLastError();

if(error\_code == WSAECONNRESET)

{

//Somebody disconnected , get his details and print

printf("Host disconnected unexpectedly , ip %s , port %d \n" , inet\_ntoa(address.sin\_addr) , ntohs(address.sin\_port));

//Close the socket and mark as 0 in list for reuse

closesocket( sock );

client\_socket[tracker] = 0;

}

else

{

printf("recv failed with error code : %d" , error\_code);

}

}

else if ( rcvd\_size == 0)

{

//Somebody disconnected , get his details and print

printf("Host disconnected , ip %s , port %d \n" , inet\_ntoa(address.sin\_addr) , ntohs(address.sin\_port));

//Close the socket and mark as 0 in list for reuse

closesocket( sock );

client\_socket[tracker] = 0;

}

//Echo back the message that came in

else

{

//add null character, if you want to use with printf/puts or other string handling functions

rcvd\_msg[rcvd\_size] = '\0';

feed[0]='\0';

feed[0]=callHash(getHash(rcvd\_msg));

if(feed[0]=='0'||feed[0]=='1')

send( sock , feed , 1 , 0 );

}

}

}

}

}

void openData()

{

fp=fopen("Data/user\_data.txt", "r+");

if(fp==NULL)

{

fp=fopen("Data/user\_data.txt", "w+");

if(fp==NULL)

{

printf("\nDatabase Creation Failed.");

printf("\nPress any key to exit.");

getch();

exit(0);

}

}

}

int getHash(char \*hash)

{

int i,x=0;

char p[4];

for(i=1;i<5;i++)

p[i-1]=\*(hash+i);

x=atoi(p);

return x;

}

char callHash(int hash)

{

char ch, ucheck[30],n\_msg[MSG\_SIZE], u\_store[20];

SOCKET ns;

int count=0,i,j;

switch(hash)

{

case 1011:

fseek( fp, 0, SEEK\_SET );

count=0;

for(i=0;(ch=fgetc(fp))!=EOF;i++)

{

ucheck[i]=ch;

if(ch=='#')

{

count++;

if(count==3)

{

ucheck[i]='\0';

if(strcmp(rcvd\_msg,ucheck)==0)

{

return '1';

}

count=0;

strcpy(ucheck,"\0");

i=-1;

}

}

else if(ch=='\n')

{

count=0;

strcpy(ucheck,"\0");

i=-1;

}

}

strcpy(msg,rcvd\_msg);

return '0';

break;

case 1012:

fseek( fp, 0, SEEK\_END );

strcat(msg, rcvd\_msg);

strcat(msg, "\n");

fputs(msg,fp);

fclose(fp);

openData();

printf("%s",msg);

return '0';

break;

case 1013:

fseek( fp, 0, SEEK\_SET );

rcvd\_msg[4]='1';

for(i=0;(ch=fgetc(fp))!=EOF;i++)

{

ucheck[i]=ch;

if(ch=='\n')

{

ucheck[i]='\0';

if(strcmp(rcvd\_msg,ucheck)==0)

{

for(j=6;j<strlen(rcvd\_msg);j++)

{

if(rcvd\_msg[j]=='#')

{

u\_store[j-6]='\0';

strcpy(users[tracker],u\_store);

goto go1;

}

u\_store[j-6]=rcvd\_msg[j];

}

go1:

return '0';

}

strcpy(ucheck,"\0");

i=-1;

}

}

return '1';

break;

case 1021:

public\_chatters[tracker]=1;

strcpy(n\_msg,users[tracker]);

strcat(n\_msg," has joined public chat.");

for(i=0;i<MAX\_CLIENT;i++)

{

ns=client\_socket[i];

if(i!=tracker&&public\_chatters[i]==1)

send( ns , n\_msg , strlen(n\_msg) , 0 );

}

break;

case 1022:

strcpy(n\_msg,users[tracker]);

strcat(n\_msg," : ");

strcat(n\_msg,&rcvd\_msg[6]);

for(i=0;i<MAX\_CLIENT;i++)

{

ns=client\_socket[i];

if(i!=tracker&&public\_chatters[i]==1)

send( ns , n\_msg , strlen(n\_msg) , 0 );

}

break;

case 1023:

public\_chatters[tracker]=0;

strcpy(n\_msg,users[tracker]);

strcat(n\_msg," has left public chat.");

for(i=0;i<MAX\_CLIENT;i++)

{

ns=client\_socket[i];

if(i!=tracker&&public\_chatters[i]==1)

send( ns , n\_msg , strlen(n\_msg) , 0 );

}

break;

}

}

int verify\_entry(char entry[20])

{

int i,x;

if(strlen(entry)>20)

return 1;

for(i=0;i<strlen(entry);i++)

{

x=entry[i];

if((x>=64&&x<=90)||(x>=97&&x<=122)||(x>=48&&x<=57)||(x==46)||(x==95))

continue;

return 1;

}

return 0;

}

int show\_menu()

{

int choice;

switch(menu)

{

case 0:

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\t\t\t WELCOME TO U-CHAT %s\n",user);

printf("\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n\n");

break;

case 1:

printf("\t\t\t \n");

printf("\t\t\t-------------------------\n");

printf("\t\t\t| 1--->Login |\n");

printf("\t\t\t| 2--->Exit |\n");

printf("\t\t\t-------------------------\n\n\n");

printf("\t\t\t Enter a Choice\n");

break;

case 2:

printf("\t\t\t-----------------------------\n");

printf("\t\t\t| 1--->Initialize Server |\n");

printf("\t\t\t| 2--->Exit |\n");

printf("\t\t\t-----------------------------\n\n\n");

printf("\t\t\t Enter a Choice\n");

break;

}

if(menu==0)

choice=100;

else

scanf("%d",&choice);

return choice;

}

int login\_admin()

{

char uname[20], pass[20];

int i=0;

char ch;

int k;

re\_uname:

printf("\t\t\t Enter admin Username ---> ");

fflush(stdin);

gets(uname);

if(verify\_entry(uname))

{

printf("\t\t\t INVALID USERNAME. PLEASE TRY AGAIN.\n");

goto re\_uname;

}

re\_pass:

printf("\t\t\t Enter admin Password ---> ");

fflush(stdin);

i=0;

while (1)

{

if (i < 0)

{

i = 0;

}

ch = getch();

if (ch == 13)

break;

if (ch == 8) /\*ASCII value of BACKSPACE\*/

{

printf("\b \b");

i--;

pass[i] = '\0';

continue;

}

pass[i++] = ch;

ch = '\*';

printf("%c",ch);

}

pass[i]='\0';

if(verify\_entry(pass))

{

printf("\t\t\t INVALID PASSWORD. PLEASE TRY AGAIN.\n");

goto re\_pass;

}

if(strcmp(uname,"admin")==0||strcmp(pass,"admin")==0)

{

strcpy(user,uname);

return 0;

}

printf("\t\t\t USERNAME OR PASSWORD INCORRECT. PLEASE TRY AGAIN.\n");

goto re\_uname;

}

**Chapter 6**

**Conclusion and Future Scope**

**6.1 Future Scope**

The future of the project includes the following features:

* To make the application run over internet.
* To include exchange of emoji with text messages.
* To allow sharing of images and files.
* Include group chat.
* Add message broadcast.
* Add message bulletin board specifically for organizations.

**6.2 Conclusion**

I learned a lot by doing this project.

* Languages used: C
* Backend: File Handling

So, during this project I learned all the above things. Before this project, I had not much idea about socket programming in C for making application. Although I had little bit knowledge of basic C before.

**References**

* https://www.geeksforgeeks.org
* https://www.tutorialspoint.com/cprogramming
* https://msdn.microsoft.com/en-us/library/windows/desktop/ms740673(v=vs.85).aspx