Chat Developer Manual

Version: 2.0.0

University of California, Irvine
EECS 22L, TEAM 13

Justin Han
Raymond Yu
Vivek Hatte
Daisuke Otagiri
Hao Ming Chiang

Table of Contents

Glossary

Add User - function where user can add other users to their contacts list

Application Program Interface (API) - a defined set of functions that enables data exchange between software applications. The API determines how programs should interact between the operating system or protocol.

Contacts List - a list of users maintained by the server to keep track of who added who using the "Add User" function

Data Types - a list of variable types and structure definitions our software will use

GUI (Graphical User Interface) - allow users to interact with electronic devices through graphical icons and indicators instead of command labels.

GTK (GIMP Toolkit) - free and open-source cross-platform widget toolkit that is used to create graphical user interfaces

Instant Messaging - online chat function that allows for real-time text transmission. Short messages are transmitted over the server and sent back with little to no lag.

Log in - user enters information from registration and server checks if information matches original registration information.

Module Diagram - a diagram that displays the structure of the program and the connection between each module of the program

Program Control Flow - the steps the program will execute in given user input once program opened

Register - user enters information and a record is sent and kept on the server

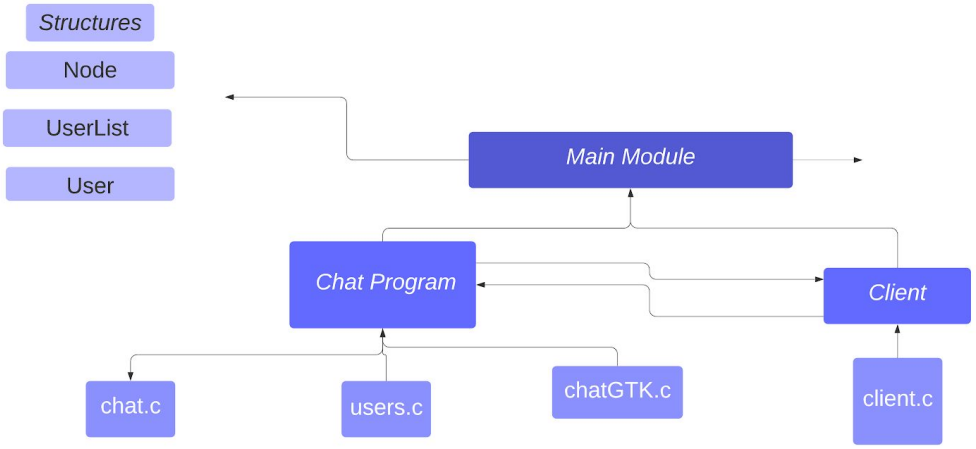
Usage Scenario - An example of how our software will run from the user's perspective

Client Software Architecture Overview

Main Data Types and Structures

- GtkWidget

Major Software Components



Module Interfaces

- chat_client.c
  - int main(int argc, char* argv);
  - Void * getData (void * client);
- chat_gtk.c
  - int main (int argc, char *argv[]);
  - GtkWidget *Create_ChatWindow(int *argc, char **argv[]);
  - GtkWidget *Create_RegisterWindow(int *argc, char **argv[]);
  - GtkWidget *Add_UserWindow(int argc, char *argv[]);
  - GtkWidget *Create_LoginWindow(int *argc, char **argv[]);
  - void createAccountButton(void);
  - void listContacts(char* username);
  - void chat(char* username);

Program Control Flow

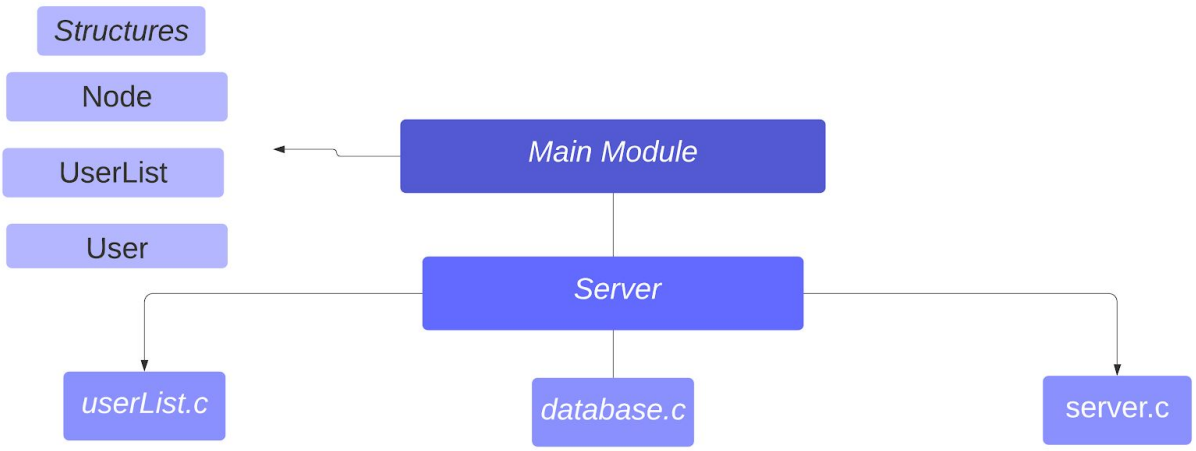
- Main Chat Window
    - UI shows main chat window with text field underneath main chess program
    - Chat takes any keyboard input from user
        - Ex.
            - User types “SEND 2 Hello”
            - Main chat window will send “Hello” to client 2
    - Add User Feature
        - Clicking on “Add User” button will pop another window
        - Window will have text field for user to input another user’s username
    - Sign in Feature
        - Click on “Sign in” button will pop another window
        - Window will have 2 text fields
            - Username text field - user types in their username
            - Password text field - user types in their password
        - Server will check data to see if there is a matching username and password
            - If not, sign in is denied
    - Register Feature
        - Click on “Register” button will pop another window
        - Window will have two fields:
            - Username - user types in an username for future use
            - Password - user types in password for future use
        - Server will write the information to the data

Server Software Architecture Overview

Main Data Types and Structures

- SocketInfo
  - socket descriptor (int)
- Client structure
  - { int index; char name[20]; int sockID; struct sockaddr_in ClientAddr; int len }

Major Software Components



Module Interfaces

- server.c
    - int main();
    - void * Exchange Data(void * ClientDetail);

Program Control Flow

- Start the server
- Binds automatically to the ip and the port and begins to listen at port 3333, hardcoded port which can be changed in source code
- Opens a thread for each new client so that they can send messages through the server all at the same time
- Concurrently saves the users socket destination and their data sending in so it can see who to send it too and who is sending the message.

Installation

System Requirement:

- Any OS that allows you to SSH into linux servers
- Standard Library (C Programming)
- Math Library
- C programming compiler (e.g. gcc)

Setup and configuration:

- Download Chat_Beta_src.tar.gz from the Team13 repository in Github
- Log into any accessible linux servers
- Create a directory locally (type mkdir <name_of_directory>)
- Type in the linux terminal:
  > cd <name_of_directory>
  > tar -xzvf Chess_Beta_src.tar.gz
  > make
  > ./bin/chat_server
  -This is to run the server so that the clients will have something to connect to
  > ./bin/GUIlient <your_name>
  -You can open multiple clients and talk between the users. You need to open a client with a name after you run ./client so its know your username
  Ex.
  ./GUIclient Bobby

Uninstalling:

- Login to any linux server on an ssh client (putty, etc)
- Go to the directory where the project was extracted
- Then type
  > make clean
  > rm -rf <your_directory_here>

Documentation of Packages, Modules, Interfaces

- chat_client_final.c
    - int main();
        - Takes in parameters from command line which will be executable followed by the user name
        - There is a forever while loop that runs and is constantly taking in data to see what the user is typing
    - Void * getData(void * client)
        - Continuously gets data from a server stream of data in a new thread and outputs it in the client.

- chat_gtk.c
    - int main(int argc, char *argv[])
        - Responsible for calling the Gtk widgets to open windows for each respective feature of the program (register, log in, add user,...)
    - static void enter_callback(GtkWidget *widget, GtkWidget *entry)
    - void hide(GtkWidget *Widget)
        - Hides any GtkWidgets upon called
    - void LoginButton(GtkWidget *button, gpointer data) (incomplete)
        - Upon user clicking “Login”, typed username and password is checked with the UserList to see if it matches any users.
    - void RegistrationButton(GtkWidget *button, gpointer data)
        - Upon user clicking “Regist
        - er”, a new user will be created in UserList and stored for future Logins
    - void AddUserButton(GtkWidget *button, gpointer data)
        - After user types in another user’s name and clicking “Add”, this function checks if the typed username matches any usernames in UserList and adds them to the respective contact’s list
    - void add_Friend(GtkWidget *widget, gpointer name);
        - Adds friends into the friend list
    - void delete_Friend(GtkWidget *widget, gpointer name);
        - Deletes friends from the friend list
    - void init_list(GtkWidget *list);
        - Initializes the friend list
    - GtkWidget *Create_LoginWindow()
        - Creates the GTK GUI for the login window

        - After the event (button) starts, the terminal will allow the user to input username and password
    - void Create_RegisterWindow();
        - Creates the GUI window for register window
        - After the event (button) starts, the terminal will allow the user to input username and password
        - User will need to retype password to successfully register
    - void Create_ChatWindow();
        - Creates the GUI chat window
        - Enables user to type messages into window and messages to appear on the window
    - void mainMenu()
        - Creates a main menu window with two buttons:
            - Chat
            - Friend
    - void *Add_UserWindow()
        - Creates the add user window
        - Has “Add” button next to text box field
        - “Remove” button
    - void *FriendListWindow()
        - Creates the Friends List Window
        - Lists all users that are in friend’s list
    - void FatalError(const char *ErrorMsg)
    - void destroyWindow(GtkWidget *window)
        - Destroys the GtkWidget window upon being called

- chat_server_final.c
    - int main();
        - Infinite while loop looking to connect to any client that is looking to connect
        - Opens a new thread for each of the active clients that exchanges data between other clients that they want to send data to
    - void *exchangedata(void* client);
        - Takes a client data from a certain client
        - It's a continuous function that is open in a thread that sends data to the person that the client specified.

Development Plan and Timeline

Partitioning of Tasks

1. Client Functions
    a. chat_client_final.c
    b. chat_GTK.c (GUI)
        - Chat Window
        - Contacts List
        - Add Friend
        - Register
        - Sign in
        - Delete
2. Server Functions
    a. chat_server_final.c

Team Member Responsibilities

Justin Han - GUI
Raymond Yu - UserList.c
Vivek Hatte - Server and Client
Daisuke Otagiri - GUI
Hao Ming Chiang - GUI

Complete Server, Client, and chat_GTK.c by end of Week 9 (Alpha Release)
Complete all features by end of Week 10 (adding friends, all GUI functions, …)

Copyright Notice

Published by Team 13 members:
Justin Han, Raymond Yu, Vivek Hatte, Daisuke Otagiri, Hao Ming Chiang

Error Messages

All the error messages will be printed out in the log.
Each error will have a printed message following the error.

Example:
If a player attempts to send a message and loses connection
“Message not sent, please check your connection and try again.”

Index