نویسنده : علی ونکی فراهانی

فاز ۱ :

در این فاز منو اصلی , منو ورود و عضویت در کانال و منو مربوط به هر کانال در int main( ) کار شده است و بقیه کارها در تابع های مربوطه انجام می‌شود .

تابع int socket_saz( )

در این تابع یک سوکت ساخته شده و تحویل داده می‌شود . در ادامه کار این سوکت برای ارتباط بین سرور و کلاینت مورد نیاز است و از آن استفاده می‌شود .

تابع void send_and_recive_array (char request[] , char type[10000] , char content[10000])

در این تابع در ابتدا با استفاده از تابع socket_saz یک سوکت ساخته می‌شود . هم چنین درخواست که به حالت استاندارد درآمده است هم به عنوان وروردی به تابع داده می‌شود . سپس با استفاده از سوکت ساخته شده , درخواست مورد نظر به سرور ارسال می‌شود و منتظر پاسخ سرور باقی می‌ماند . بعد از دریافت پاسخ سرور آن را تجزیه و تحلیل می‌کند و type و content پیام دریافتی را در دو آرایه ذخیره می‌کند . نکته مهم این است که در این بخش type و content هر دو ساده می‌باشند و به صورت آرایه نیستند .

تابع void SetColor(int ForgC)

این تابع که از داخل نت یافت شده است برای تنظیم رنگ برنامه مورد استفاده قرار می‌گیرد . با استفاده از این تابع و با دادن یک عدد به عنوان ورودی به آن (هر عدد بیانگر یک رنگ می‌باشد که لیست آن در نت یافت می‌شود .) می‌توان رنگ متن قسمت‌های مختلف پروژه را تغییر داد .

تابع void send_and_recive_array (char[]request , char type[10000] ,char content [10000])

عملکرد این تابع همانند تابع send_and_recieve می‌باشد یعنی با ساختن سوکت مورد نظر به سرور متصل شده و درخواست استانداردی که از قبل طراحی و به عنوان ورودی به آن داده شده است را به سرور ارسال می‌کند و منتظر پاسخ سرور می‌ماند . سپس پاسخ سرور را تجزیه تحلیل می‌کند . این تابع زمانی کاربرد دارد که content ارسالی از سمت سرور به صورت آرایه باشد که در این حالت این تابع آن را تجزیه تحلیل می‌کند .

تابع void send_and_recive_members(char request[] , char type[10000] , char content[10000])

این تابع دقیقا مانند تابع send_and_recieve_array کار می‌کند و تنها تفاوت این است که زمانی که سرور لیست افراد مورد نظر را ارسال می‌کند این تابع صدا زده می‌شود .

تابع void send_and_recive_Find(char request[] , char type[10000] , char content[10000])

این تابع مانند تابع send_and_recieve_members کار می‌کند و زمانی استفاده می‌شود که به دنبال یک پیام خاص در بین پیام‌های دیگر هستیم .

تابع های دیگر

در تابع‌های دیگر , درخواستی که می‌خواهیم به سرور بفرستیم را به حالت استاندارد تبدیل می‌کنیم , یعنی به حالتی تبدیل می‌کنیم که برای سرور قابل فهم باشد . سپس با استفاده از توابع بالا (هر تابع بر حسب نیاز) تابع مورد نظر را فراخوانده , درخواست خود را به سرور ارسال و جواب سرور را دریافت می‌کنیم . در این توابع با توجه به جواب سرور پاسخ مناسب به کاربر ارسال می‌شود .

فاز ۲ :

در این فاز در قسمت اصلی برنامه (در int main( )) یک سوکت ساخته می‌شود سپس با استفاده از این سوکت درخواستی از جانب کلاینت را دریافت می‌کنیم سپس با تجزیه تحلیل درخواست دریافتی تابع‌های مناسب به آن را صدا زده و پیام سرور به کلاینت را با استفاده از این تابع‌ها و سوکتی از قبل ساختیم را ساخته و ارسال می‌کنیم . هم چنین در قسمت اصلی برنامه دایرکتوری‌های مورد نظر ساخته می‌شود و فایل‌های مورد نظر هم در صورت عدم وجود ساخته می‌شود . از این فایل‌ها می‌توان به Members_List.txt و Channel_List اشاره کرد که به ترتیب اسامی کاربرها و اسامی کانال‌های مختلف را در خود ذخیره می‌کنند . هم چنین در ابتدا شروع به کار سرور , لیست کاربران به همراه اطلاعات آنها در یک struct ذخیره می‌شود . در ادامه هم اگر کاربری ثبت نام کند به این struct اضافه می‌شود . (منظور از استراکت , آرایه‌ای از استراکت‌ها است) .

تابع int Build_Socket(int server_socket)

در این تابع یک سوکت ساخته شده و تحویل داده می‌شود . در ادامه کار این سوکت برای ارتباط بین سرور و کلاینت مورد نیاز است و از آن استفاده می‌شود .

تابع void login()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد وارد حساب کاربری خود شود . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود . در این تابع خطاهای اشتباه بودن رمز , غیرمجاز بودن نام کاربری و عدم وجود نام کاربری چک می‌شود .

تابعvoid login_successful(int n)

در صورتی که ورود با موفقیت قابل انجام باشد , این تابع پیام موفقیت را به کلاینت ارسال می‌کند . هم چنین , یک توکن ساخته و آن را به کاربر اختصاص می‌دهد و در فایل مربوط به کاربر ذخیره می‌کند .

تابع void login_wrong_password()

این تابع در صورتی که ورود ناموفق باشد , پیام مناسب را ساخته و آن را به کلاینت ارسال می‌کند . این تابع زمانی فراخوانده می‌شود که نام کاربری و رمز عبور با هم مطابقت نداشته باشند .

تابع void login_username_not_existed()

این تابع در صورتی که ورود ناموفق باشد , پیام مناسب را ساخته و آن را به کلاینت ارسال می‌کند . این تابع زمانی صدا زده می‌شود که نام کاربری وجود نداشته باشد .

تابع void login_invalid_username()

این تابع در صورتی که ورود ناموفق باشد , پیام مناسب را ساخته و آن را به کلاینت ارسال می‌کند . این تابع زمانی صدا زده می‌شد که نام کاربری غیر مجاز باشد . (احتمالا به دلیل وجود space در آن)

تابع void registeration( )

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد ثبت نام کند . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود .

تابع void register_already_exist()

این تابع در صورتی که ثبت نام ناموفق باشد , پیام مناسب را ساخته و آن را به کلاینت ارسال می‌کند . این تابع زمانی صدا زده می‌شود که نام کاربری از قبل وجود داشته باشد و امکان ثبت نام وجود ندارد .

تابع void register_successful(char* Username , char* Password)

این تابع در صورتی که امکان ثبت نام وجود داشته باشد صدا زده می‌شود . در این تابع ابتدا یک فایل برای کاربر مورد نظر ساخته می‌شود که حاوی اطلاعات کاربر است . سپس , نام کاربری کاربر در Members_List.txt افزوده می‌شود . در نهایت , پیام ثبت نام با موفقیت انجام شد , در سرور ساخته شده و به کلاینت فرستاده می‌شود .

تابع void register_fail_invalid_username()

این تابع در صورتی که ثبت نام ناموفق باشد , پیام مناسب را ساخته و آن را به کلاینت ارسال می‌کند . این تابع زمانی صدا زده می‌شد که نام کاربری غیر مجاز باشد . (احتمالا به دلیل وجود space در آن)

تابع void CreateChannel()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد یک کانال جدید به وجود بیاورد . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود .

تابع void Channel_name_invalid()

در صورتی که نام کانال غیرمجاز باشد (احتمالا شامل space باشد) این تابع فراخوانده می‌شود . در این تابع پیام خطا ایجاد شده و به کلاینت ارسال می‌شود .

تابعvoid CreateChannel_existed()

در صورتی که نام کانال از قبل وجود داشته باشد (یعنی قبلا کانالی با این نام ثبت شده باشد) , این تابع فراخوانده می‌شود . در این تابع پیام خطا ایجاد شده و به کلاینت ارسال می‌شود .

تابعvoid CreateChannel_successful(char *channel_name , char *Creator, int k)

این تابع در صورتی که امکان به وجود آوردن کانال وجود داشته باشد صدا زده می‌شود . در این تابع ابتدا یک فایل برای کانال مورد نظر ساخته می‌شود که حاوی اطلاعات کانال (یک فایل برای پیام‌ها و یک فایل برای فرستنده های پیام ها) است . سپس , نام کانال در Channel_List.txt افزوده می‌شود . در نهایت , پیام ایجاد کانال با موفقیت انجام شد , در سرور ساخته شده و به کلاینت فرستاده می‌شود .

تابع void CreateChannel_invalid_AuthToken()

این تابع درصورتی که توکن گرفته شده از کلاینت در سرور موجود نباشد (یعنی کاربری با آن توکن موجود نباشد) فراخوانده می‌شود و پیام خطای مناسب را به کلاینت ارسال می‌کند .

تابع void JoinChannel()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد وارد یک کانال بشود . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود . از جمله خطاهای احتمالی آن , این است که اصلا کانالی با آن نام وجود نداشته باشد و یا توکن مفروض در سرور موجود نباشد .

تابع void JoinChannel_successful(char *channel_name,char *Joiner,int k)

در صورتی که امکان ورود وجود داشته باشد , در استراکت کاربر نام کانال وارد شده و ذخیره می‌شود . هم چنین در قسمت فایل‌ها , یک پیام از سمت سرور ارسال می‌شود که کاربر مفروض در کانال عضو شد . هم چنین پیام موفقیت آمیز بودن ورود به کلاینت ارسال می‌شود .

تابعvoid JoinChannel_not_existed()

این تابع زمانی صدا زده می‌شود که کانالی که کاربر می‌خواهد موجود نباشد . در این تابع پیام خطای مناسب ایجاد شده و به کلاینت ارسال می‌شود .

تابعvoid JoinChannel_invalid_AuthToken()

این تابع درصورتی که توکن گرفته شده از کلاینت در سرور موجود نباشد (یعنی کاربری با آن توکن موجود نباشد) فراخوانده می‌شود و پیام خطای مناسب را به کلاینت ارسال می‌کند .

تابعvoid Logout()

در این تابع ابتدا چک می‌کند که آیا کاربری با توکن داده شده وجود دارد یا نه . اگر کاربر وجود داشته باشد , توکن آن را صفر می‌کند و اگر وجود نداشته باشد تابع مورد نظر را ارسال می‌کند . به طور کلی از اینجا به بعد همواره توکن چک می‌شود و در صورت عدم وجود تابع مناسب صدا زده می‌شود .

تابعvoid Send_Message()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد یک پیام ارسال کند . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود .

تابعvoid Send_Message_successful(char *Message,int k)

در این تابع پیام موفقیت ایجاد شده و به کلاینت ارسال می‌شود . هم چنین متن پیام و فرستنده آن به ترتیب در فایل‌هایی که از قبل به وجود آمده‌اند ذخیره می‌شوند .

تابعvoid Channel_members()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد یک اعضای کانال را مشاهده کند . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود .

تابعvoid Channel_members_successful(int n)

در صورتی که مشاهده اعضای کانال با مشکل رو به رو نشود این تابع فراخوانده می‌شود . در این تابع پیام موفقیت ایجاد شده و به کلاینت ارسال می‌شود . هم چنین لیست افراد کانال هم در قالب cJSON به کلاینت ارسال می‌شود .

تابعvoid Channel_members_not_existed()

در صورتی که کاربر در هیچ کانالی عضو نباشد این تابع صدا زده می‌شود . در این تابع , پیام خطای مناسب برای کلاینت ایجاد و ارسال می‌شود .

تابع void Leave()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد کانالی که در آن حضور دارد , خارج بشود . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود .

تابع void Leave_successful(int n)

در این تابع پیام موفقیت ایجاد شده و به کلاینت ارسال می‌شود . هم چنین در استراکت کاربر , نام کانال (که نشان می‌داد کاربر در چه کانالی عضو می‌باشد .) تماما پاک می‌شود . هم چنین در لیست پیام‌های کانال یک پیام از جانب سرور ارسال می‌شود که کاربر مدنظر از کانال خارج شد .

تابع void Refresh()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد پیام‌های کانال را رفرش کند . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود .

تابع void Refresh_successful(int k)

در این تابع پیام موفقیت ایجاد شده و به کلاینت ارسال می‌شود . هم چنین , نقطه رفرش را برای کاربر در استراکت جا به جا می‌کند . (هر کاربر در استراکت موجود در سرور دارای یک مقدار int می‌باشد که این مقدار نشان دهنده نقطه شروع رفرش می‌باشد .)

تابع void FindUser()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد نام یک کاربر را میان افراد یک کانال جستجو کند . در این تابع خطاهای احتمالی را بررسی می‌کند و در صورت پیدا کردن خطایی در آن تابع مناسب را صدا می‌زند . در صورتی که هیچ خطایی یافت نشود تابع مناسب فراخوانده می‌شود .

تابع void FindUser_Fail()

این تابع زمانی صدا زده می‌شود که پس از جسنجو , کاربری با ویزگی‌های داده شده در کانال موجود نباشد . در این تابع پیام خطا درست شده و به کلاینت ارسال می‌شود .

تابع void FindUser_successful(char *User , int k)

در این تابع پیام موفقیت ایجاد شده و به کلاینت ارسال می‌شود .

تابع void FindUser_invalid_username()

این تابع در صورتی که ثبت نام ناموفق باشد , پیام مناسب را ساخته و آن را به کلاینت ارسال می‌کند . این تابع زمانی صدا زده می‌شد که نام کاربری غیر مجاز باشد . (احتمالا به دلیل وجود space در آن)

تابعvoid FindMessage()

این تابع زمانی استفاده می‌شود که سرور بعد از تجزیه تحلیل درخواست کلاینت به این نتیجه برسد که کاربر می‌خواهد میان پیام‌های یک کانال دنبال پیام‌هایی باشد که شامل کلمه ای مشخص باشد . در این تابع , آرایه‌ای از جنس cJSON ها به وجود می‌آید که شامل پیام‌های یافت شده است و آن به کلاینت ارسال می‌شود .

فاز ۳ :

در این فاز در ابتدا یک استراکت به نام cJSON تعریف می‌کنیم که شامل ۵ متغیر می‌باشد . یک Int وجود دارد که نشان می‌دهد که cJSON ما از چه نوعی می‌باشد . هم چنین شامل دو آرایه از استرینگ ها و شامل دو پوینتر به خود استراکت است که اولی نشان دهنده عضو بعدی (next) و دیگری شامل زیرمجموعه (child) می‌باشد .

تابعvoid cJSONInit(cJSON* source)

این تابع در واقع مقادیر موجود در استراکت را NULL می‌کند م

تابعcJSON *cJSON_CreateObject()

این تابع برای ایجاد یک Object به کار می‌رود . کاری که این تابع می‌کند این است که یک مقدار فضای خالی به Object موردنظر اختصاص می‌دهد و سپس با استفاده از تابعی دیگر مقدار آن را NULL می‌کند .

تابعcJSON *cJSON_CreateString(char *val)

این تابع برای ایجاد یک String به کار می‌رود . کاری که این تابع می‌کند این است که یک مقدار فضای خالی به String موردنظر اختصاص می‌دهد و سپس با استفاده از تابعی دیگر مقدار آن را NULL می‌کند . البته در این تابع مقدار Valuestring داخل استراکت را برابر استرینگ مدنظر قرار می‌دهیم .

تابعcJSON *cJSON_CreateArray()

این تابع برای ایجاد یک Array به کار می‌رود . کاری که این تابع می‌کند این است که یک مقدار فضای خالی به Array موردنظر اختصاص می‌دهد و سپس با استفاده از تابعی دیگر مقدار آن را NULL می‌کند .

تابعvoid cJSON_AddItemToObject(cJSON *sourceObject, char *name, cJSON *item)

این تابع یک عضو به یک Object اضافه می‌کند و این کار را با جستجوی فضای خالی در میان child و next انجام می‌دهد .

تابع void cJSON_AddItemToArray(cJSON *sourceArray, cJSON *item)

این تابع یک عضو به یکَ Array اضافه می‌کند و این کار را با جستجوی فضای خالی در میان child و next انجام می‌دهد .

تابع cJSON *cJSON_GetArrayItem(cJSON *array, int index)

این تابع از یک آرایه از جنس cJSON عضو شماره Index ام را دریافت می‌کند .

تابع cJSON *cJSON_GetObjectItem(cJSON *source, char *name)

این تابع از یک آبجکت از جنس cJSON , دنبال عضوی می‌گردد که شامل استرینگ مد نظر باشد .

تابع char *cJSON_PrintUnformatted (cJSON *source)

این تابع یک *cJSON را به عنوان ورودی گرفته و پس از انجام عملیات روی آن , آن را به یک رشته قابل فهم و قابل تجزیه تحلیل تبدیل می‌کند .

تابع cJSON *cJSON_Parse(char *source) , cJSON *cJSON_ParseFlag(char *source, int *i)

با استفاده از این دو تابع می‌توان یک رشته دریافت کرده و آن را به حالت cJSON تبدیل کنیم .