شایان محمدی زاده مستندسازی پروژه مبانی 98102273

**فاز یک (کلاینت)**

در این قسمت از پروژه، قسمت مربوط به کلاینت پیاده سازی شده است. در این قسمت درخواست های مربوط به کلاینت به سرور فرستاده میشود و همچنین گاسخ کاربر به این درخواست ها تجزیه و تحلیل میشوند.

**تابع SetColor** : این تابع تنها برای رنگی کردن outputها می­باشد. ورودی تابع یک int است(عددی بین 0 تا 16) که بیانگر رنگ output می­باشد.

**تابع arrow** : این تابع برای جابجایی بین منوها و انتخاب گزینه­ها با enter به جای اعداد است. ورودی این تابع تعداد گزینه­ها و خروجی آن یک int است که نشان دهنده گزینه انتخاب شده است. برای جابجایی بین گزینه ها متغییر what\_to\_do تعریف شده که نشان دهنده خطی است که کاربر درحال حاضر روی آن قرار دارد! با استفاده از این متغییر و تابع SetColor نشانگر مورد نظر ساخته می­شود.

**تابع socket\_create**: با استفاده از این تابع سوکت ساخته می­شود که برای ارتباط کلاینت و سرور به کار می­رود.

**تابع sendreq**: همان طور که میدانیم رابطه­ی سرور و کلاینت همچون رابطه گارسون و مشتری است!! این تابع برای ارسال درخواست کلاینت و دریافت پاسخ سرور کاربرد دارد. در رشته req درخواست کاربر ذخیره می­شود، در رشته reqtype و reqtoken پاسخ سرور به این درخواست ها ذخیره میشود. در همین تابع نیز این درخواست­ها بررسی میشوند. بررسی پاسخ سرور به معنای parse کردن cJSON ها و نمایش آنها به کاربر است. این تابع به نوعی مهمترین تابع برنامه به حساب می­آید.

**تابع account\_menu**: این تابع صفحه اول چت اپلیکیشن را به نمایش میگذارد که در آن کاربر میتواند اکانت بسازد(register) یا وارد اکانت خود شود(log in).

**تابع main\_menu**: در این تابع کاربر میتواند کانال بسازد، وارد کانالی از پیش ساخته شده شود، یا از حساب خود خارج شود. در این تابع همچنین درخواست مورد نظر با توجه به گزینه انتخابی کاربر نیز ساخته می­شود.

**تابع chat\_menu**: در این تابع نیز کاربر میتواند پیام ارسال کند، پیام های فرستاده شده را ببیند، اعضای کانال را مشاهده کند، و همچنین از کانال خارج شود. در این تابع درخواست مورد نظر با توجه به گزینه انتخابی کاربر نیز ساخته می­شود.

**فاز دو (سرور)**

در این قسمت از پروژه، بخش مربوط به سرور پیاده سازی شده است. در این قسمت درخواست های کلاینت دریافت شده و با توجه به آنها پاسخ مناسب به سمت کلاینت فرستاده میشود. تمامی این پاسخ ها از دو بخش type و content تشکیل شده­اند. Type میتواند برابر error یا list باشد و در content به نوعی اطلاعات مربوط به آن type قرار میگیرد.

**تابع socket\_maker**: همانطور که در فاز یک گفته شد، راه ارتباطی کلاینت و سرور سوکت میباشد. در این تابع سوکت مورد نظر ساخته می­شود.

**تابع recvreq**: گفته شد که کلاینت درخواست های خود را به سرور ارسال میکند، این درخواست ها در این تابع دریافت شده و با توجه به درخواست مورد نظر توابعی همچون register\_func و login\_func فراخوانده میشوند. در انتها نیز پاسخ سرور به کلاینت فرستاده می­شود.

**تابع register\_func**: در این تابع کلاینت درخواست ساخت یک حساب برای کاربر را دارد. کاربر با وارد کردن یک نام کاربری(username) و رمز عبور(password) منتظر پاسخ سرور است. سرور در اغاز این دو را دریافت میکند، سپس با توجه به حساب های کاربری که از قبل ساخته شده­اند باید چک کند که نام کاربری وارد شده قبلا استفاده نشده باشد، اگر چنین نام کاربری وجود نداشت، حسابی برای کاربر ساخته میشود. ورودی ها در دو رشته name و pass ذخیره میشوند و با استفاده از فایل و توابع کتابخانه cJSON ، در فایلی با نام آن کاربر قرار میگیرند.

* لازم به ذکر است که در انتهای هر تابع، رشته­ی پاسخ سرور به کلاینت(res) با استفاده از توابع کتابخانه cJSON ساخته می­شود.
* در برخی توابع رشته­ای به نام htgt(how to get there) تعریف شده است. این رشته برای تعیین مسیر فایل و دسترسی به اطلاعات کاربران یا فایل کانال است.

**تابع login\_func**: در این تابع با توجه به درخواست کلاینت، کاربر باید با وارد کردن اطلاعات خود وارد صفحه اصلی چت اپلیکیشن شود. در ابتدا وجود نام کاربری و در ادامه صحیح بودن پسوورد وارد شده بررسی میشوند. اگر هردو درست بودند، باید توکن­ای برای کاربر ارسال شود که در حقیقت این توکن یک راستی آزمایی برای کاربر است که کاربر همانی است که باید باشد! برای ساخت این توکن تابع create\_token فراخوانی میشود که در ادامه نحوه کار آن توضیح داده میشود. در پایان نیز رشته cJSON مورد نظر ساخته می­شود.

**تابع create\_token**: این تابع در حقیقت یک رمز برای کاربر میسازد. این رمز با استفاده از تابع rand ساخته شده و متشکل از حروف انگلیسی بزرگ، حروف انگلیسی کوچک و اعداد صفر تا نه می­باشد.

* لازم به ذکر است کلاینت در ادامه ی مراحل توکن داده شده را همرا ه با درخواست خود به سرور ارسال میکند که در همه­ی توابعی که در ادامه مشاهده می­شود دزستی و مطابقت این توکن با توکن داده شده بررسی میشود.

**تابع logout\_func**: زمانی که کاربر درخواست خروج از حساب خود را دارد این تابع فراخوانی می­شود و کاربر از حساب خود بیرون میرود.

**تابع create\_channel\_func**: در این تابع فایلی با نام کانال مورد نظر ساخته میشود. در این فایل پیام های ارسال شده قرار دارند که در ادامه برای refresh مورد استفاده قرار میگیرند. در این تابع پیام “username” created “channel name”، با استفاده از رشته name و کتابخانه string.h در فایل کانال ذخیره میشود. تکراری نبودن نام کانال نیز در این تابع بررسی میشود.

**تابع join\_channel\_func**: در این تابع کاربر به کانال دلخواه وارد میشود. در اینجا باید وجود کانال مورد نظر با استفاده از اطلاعات ذخیره شده در فایل ها(نام فایل ها همان اسم کانال ها هستند)، وجود کانال نیز بررسی میشود. پیام مورد نظر( “user name” joined “channel name”) نیز همانند ساخت پیام قبل در تابع create\_channel\_func، در این تابع ساخته میشوند. نکته­ی مهم در این زمینه این است که ابتدا پیام های قبلی موجود در کانال و در نتیجه موجود در فایل کانال خوانده شده و سپس پیام جدید به آنها اضافه میشود. از این کار در تابع send\_message\_func نیز استفاده می­شود. نحوه خواندن اطلاعات از فایل با استفاده از کار با فایل ها و نحوه parse کردن و اضافه کردن به آن نیز با استفاده از کتابخانه cJSON صورت میگیرد.

**تابع send\_message\_func**: در این تابع کاربر رشته­ای را به عنوان پیام خود ارسال میکند. سرور با استفاده از توابع و کتابخانه های مورد نظر این رشته را به فایل کانال اضافه میکند.

**تابع channel\_members\_func**: ابتدا لازم به ذکر است که هنگام وارد شدن یک کاربر به کانال مورد نظر در توابع create\_channel\_func و join\_channel\_func، اطلاعات کاربر از جمله نام، توکن و نام کانالی که در آن قرار دارد در یک struct ذخیره می­شوند. در اغاز با توجه به توکن ارسالی توسط کلاینت، کاربر مورد نظر را پیدا میکنیم. سپس با توجه به داشتن نام کانال مورد نظر، کانال بقیه کاربران آنلاین را بررسی کرده و با استفاده از توابع کتابخانه cJSON آنها را به کلاینت ارسال میکنیم.

**تابع refresh\_func**: با توجه به اینکه پیام های نمایش داده شده باید از زمان آخرین رفرش باشند، int با نام previous\_arraaynum تعریف میکنیم. این به نوعی بیانگر نقطه شروع نمایش پیام ها میباشد.

**تابع leave\_channel\_func**: در این تابع کاربر از کانال مورد نظر خارج شده و پیام “username” leaved “channel name” به فایل کانال اضافه میشود.

**فاز سه (کتابخانه cJSON)**

در این فاز برخی از توابع موجود در کتابخانه cJSON.c پیاده سازی می­شوند. نکته­ی اصلی در این فاز وجود link list است. Link list به نوعی وجود یک pointer به یک struct است. ما با استفاده از این پوینتر object ها را میسازیم(در حقیقت با استفاده از آن object های بعد از object اول را میسازیم!)، این کار با تعریف آرایه­ای از پوینتر ها صورت میگیرد که در هریک از خانه های این آرایه پوینتری به یک استراکت قرار دارد. در این برنامه یک struct با نام cJSON داریم. در این استراکت اطلاعاتی از قبیل پوینتر به استراکت بعدی، تایپ cJSON مورد نظر و همچنین content آن موجود است. در متغییر valuestring، رشته های مورد نظر قرار دارند. برای مثال برای پیدا کردن اعضای کانال نام اعضا در این متغییر قرار میگیرد. متغییر hmn نیز نشان دهنده تعداد objectها و arrayهای موجود است.

**توابع cJSON\_CreateObject و cJSON\_CreateArray**: در هر دو تابع به نوعی یک استراکت ایجاد میشود.

**تابع cJSON\_CreateString**: در این تابع رشته­ای به عنوان وروردی داده شده و این رشته در متغییر valuestring یک استراکت زخیره میشود. این استراکت به عنوان خروجی داده میشود.

**تابع cJSON\_AddStringToObject**: یک استراکت و دو رشته ورودی های این تابع میباشند. این دو رشته در متغییر های type و content استراکت مورد نظر ذخیره می­شوند.

**تابع cJSON\_GetObjectItem**: یک استراکت و یک رشته ورودی های این تابع هستند. با جستجوی این رشته در متغییر تایپ استراکت مورد نظر، content آن استراکت را در متغییر valuestring استراکتی دیگر ذخیره کرده و این استراکت را به عنوان خروجی میدهیم.

**تابع cJSON\_GetArrayItem**: آرایه در cJSON در واقع مجموعه ای از objectها است. ورودی این تابع یک استراکت و یک عدد صحیح است. این عدد صحیح در حقیقت نشان دهنده object مورد نظر است. با توجه به توضیحات قبل و وجود آرایه ای از پوینتر ها، پوینتر object مورد نظر پیدا شده و به عنوان خروجی داده میشود.

**تابع cJSON\_AddItemToObject**: در این تابع در حقیقت همان پوینتر به یک تابع در خانه ای از آرایه ای از پوینترها ذخیره می­شود. ورودی های این تابع دو استراکت و یک رشته هستند که پوینتر استراکت دوم در آرایه موردنظر ساخته میشود.

**تابع cJSON\_GetArraySize**: در این تابع تعداد objectهای موجود در یک آرایه(مجموعه ی objectها) به عنوان خروجی داده میشود. ورودی نیز object موردنظر است.

**تابع cJSON\_Parse**: در این تابع تنها یک پوینتر تعریف کرده که در میان حروف رشته ورودی جا به جا شود. سپس تابعی دیگر برای parse کردن فرامی­خوانیم.

تابع cJSON\_Parse2: این تابع باید میان array و object و type یا content فرق بگذارد. برای اینکار سه if نوشته و این حالات را در آنها چک میکنیم. با چک کردن هریک از این حالات باید به ادامه رشته پرداخت که دوباره شامل این سه حالت میشود پس این تابع باید تابعی بازگشتی باشد.