به نام خدا

مستند سازی پروژه

رضا عرفان آرانی{98105919}

14/11/1398

[مبانی برنامه‌سازی]‌  [دانشگاه صنعتی شریف]

# فاز اول: کلاینت(client)

در این فاز ما باید یک کاربر در یک چت اپلیکیشن را طراحی می‌کردیم، به طوری که بتواند مانند یک کاربر چت اپلیکیشن ثبت نام کند، وارد حساب کاربری خود شود، کانالی برای چت کردن بسازد، به کانالی که از قبل در دیتابیس وجود داشته ملحق شود، در کانال پیام بفرستد، پیام‌های کانال را مشاهده کند، اعضای کانال را ببینند، با جستجو بین اعضای کانال دنبال عضوی خاص بگردند، از کانال بیرون بیایند، و دست آخر به کل از حساب کاربری خود خارج شوند.

همچنین بخش‌هایی امتیازی به مانند نحوه انتخاب گزینه‌ها با استفاده از Arrow Keys و یا با فشردن عدد گزینه و همچنین رنگ آمیزی و خوشگل سازی رابط کاربری در این پروژه پیاده سازی شده اند.

برای طراحی این سیستم ما از یک کتاب‌خانه کمکی بهره بردیم به نام cJSON که ما را در مرتب کردن و سازماندهی پیام‌ها و پاسخ سرور این چت اپلیکیشن بسیار یاری نمود.

## تابع main :

اول باید توجه داشت که reqstring رشته‌ای است که ما هر بار هر درخواستی از سرور داریم در این رشته ذخیره می کنیم و به سرور در تابع Send\_Req ارسال می کنیم.(درباره دو رشته restype و resinside جلو تر در قسمت تابع Send\_Req توضیح داده شده است) چون درخواست‌های کاربر می‌توانند بی نهایت باشند در حلقه‌ای همواره درست انتخاب کاربر برای او تکرار می‌شود. درون حلقه اول ما سه گزینه داریم. گزینه‌ها به صورت ساختن یک حساب کاربری، ورود به حساب کاربری، و خروج از برنامه می باشند. همانطور که در مقدمه این بخش گفته شد انتخاب کردن گزینه‌ها به دو روش استفاده از Arrow Keys و همچنین انتخاب گزینه به صورت مستقیم می باشد.

با انتخاب گزینه sign up به صفحه ای منتقل می شوید که از شما نام کاربری و رمز عبور برای حساب کاربری می‌خواهد. اگر کاربر دیگری با اسم مشابه وجود داشته باشد پیغام خطا چاپ خواهد شد و درغیر این صورت پیغام موفقیت چاپ خواهد شد. باید توجه داشت که سرور از ما نوع خاصی پیام دستوری می خواهد. سرور طوری طراحی شده که به پیغامی با فرمت command message, authentication پاسخ دهد. درواقع ما دراینجا پیغاممان به سرور را به صورت register <username>, <password> می‌فرستیم و سرور پاسخ درخور را برای ما ارسال می کند.

گزینه sign in تا حد بسیار زیادی به‌سان گزینه قبلی عمل میکند. با دریافت یک نام‌کاربری و یک رمز‌عبور اگر خطایی وجود داشته باشد خطا را چاپ می کند و اگر نداشته باشد به صفحه ساخت و الحاق به کانال گفتگو منتقل می کند شما را. در این مرحله شما از طرف سرور یک authentication code دریافت می کنید که باید هر بار می خواهید با این کاربر درخواستی از سرور بکنید از این کد استفاده کنید مگرنه سرور به شما پیغام خطا خواهد داد پس آن را در رشته token ذخیره می‌کنیم.

گزینه Exit نیز شما را از حلقه بیرون می‌اندازد و عملا شما به پایان برنامه می رسید.

بعد از ورود به حساب کاربری در گزینه دوم با منویی جدید با سه گزینه جدید مواجه می شوید. گزینه ها ساخت یک کانال، ملحق شدن به یک کانال، و خروج از حساب کاربری هستند.

برای ساخت یک کانال(chat-room) از گزینه اول یعنی create channel استفاده می‌شود. با انتخاب این گزینه از شما اسم یک کانال خواسته می شود. اگر کانالی با این اسم وجود داشته باشد به شما پیغام خطا و در غیر اینصورت به شما پیغام صحیح می دهد و مستقیما شما را به صفحه کانال درست شده منتقل می کند. دقت کنید که در این مرحله نیز پیغامی که به سرور فرستاده می‌شود از فرمول command message, authentication پیروی می‌کند و در اینجا authentication همان token حساب می‌شود.

گزینه دیگر الحاق به کانالی از قبل موجود می باشد. مراحل این کار عینا به مانند ساخت کانال است و اگر کانالی با این اسم وجود نداشته باشد به شما پیغام خطا و در غیر این صورت به صفحه کانال منتقل می شوید.

گزینه آخر این صفحه sign out میباشد که با آن می توانید از حساب کاربری خود خارج شده و با حساب کاربری دیگری وارد شوید. فرمول پیغام فرستاده شده در این تابع به خاطر نبودن message با حذف این بخش از فرمول اصلی و به صورت command authentication فرستاده می شود.

## تابع chat:

در اینجا بعد از وارد شدن به یک کانال وارد این صفحه می‌شوید. دراین مکان ما شش گزینه فرستادن پیام، دیدن پیام های جدید، لیست اعضای کانال، جستجو در بین اعضای کانال، جستجو در بین متن پیام های درون کانال، و خروج از کانال مواجه می شویم.

با استفاده از گزینه اول می توانید پیام خود را ارسال کنید تا در کانال ثبت شود. دقت کنید که همچنان فرمول فرستادن درخواست به سرور دارد حفظ می شود.

با انتخاب گزینه دوم می توانید از آخرین باری که پیام ها را Refresh کرده اید تا بدین جا پیام های جدید را به همراه فرستنده آنها مشاهده کنید. بدیهی است که بار اولی که Refresh را بزنید از اولین پیغام کانال تا آخرین پیغام موجود به شما نمایش داده می شود. دقت کنید که در این گزینه ما فرمول فرستادن درخواست به سرورمان کمی تغییر می کند. در این گزینه ما messageای نداریم که به سرور بفرستیم برای همین در این درخواست ما از فرمول command authentication استفاده می‌کنیم.

گزینه سوم نیز درخواست نشان دادن کاربران موجود در کانال است. دقت کنید که این گزینه نیز فرمولی مطابق فرمول Refresh دارد.

گزینه چهارم از کاربر یک اسم کاربری می گیرد. اگر این کاربر در کانال موجود باشد به کاربر True و در غیر این صورت به کاربر False نشان می دهد. فرمول پیغام فرستاده شده به سرور در این قسمت مانند فرمول اصلی است.

گزینه پنجم با دریافت کلمه ای به جستجو در پیغام‌های کانال پرداخته و اگر وجود داشته باشه جمله ای که در آن پیام به کار برده شده و درغیر این صورت پیغام Not Found چاپ می شود. فرمول پیغام ارسالی به سرور به مانند فرمول اصلی است.

گزینه آخر نیز شما را از حلقه درون تابع بیرون آورده و عملا شما از تابع خارج شده و به حلقه بینهایت منوی انتخاب کانال برمی‌گردید. فرمول ارسالی به سرور مانند فرمول sign out می باشد.

## تابع Send\_Req:

این تابع، تابع اصلی ارتباط با سرور است. با ساختن یک socket که در تابع Sock\_Creator ساخته می شود ما به سرور وصل می شویم. در این تابع ما دو نوع جواب دریافتی از سرور خواهیم داشت. نخست باید توجه داشت که تمام جواب‌های سرور از یک type و یک content تشکیل شده‌اند. ما با دریافت type و content از سرور آن ها را به ترتیب در restype و resinside می ریزیم و بعدها ازآنها استفاده می کنیم. Type به سه نوع تحویل ما داده می‌شود: یا successful یا Error و یا List که دو مدل اول به صورت مستقیم تحویل تابع های اولیه می شوند با توجه به restype پیغامشان (Resinside) اصولا یا چاپ می شود یا معادلی برای پیغام به جایشان نوشته می‌شود. در صورتی که type به حالت سوم یعنی List دربیاید ما برای راحتی عملیات چاپ را در همان تابع Send\_Req انجام می‌دهیم. رشته saver به این درد میخورد که بدانیم درخواست کاربر refresh بوده یا channel members بوده. دقت کنید که reqstring بعد از recv از سرور تغییر میکند و دارای پیغام سرور می شود و ما دیگر اطلاعی از دستورات کاربر نداریم. در قسمت list بودن نیز با استفاده از توابع موجود در کتابخانه cJSON ما درخواست مورد نظر را چاپ میکنیم.

## توابع gotoxy و SetColor:

این توابع، توایع اصلی نیستند و تنها برای قسمت های امتیازی مورد استفاده قرار گرفته اند. از gotoxy برای حرکت فلش در انتخاب گزینه ها توسط Arrow Keys استفاده شده و از دومی نیز برای تغییر رنگ نوشته ها.

# فاز دوم: سرور(server):

در این فاز ما سرور یک چت اپلیکیشن را پیاده سازی کردیم. سرور به درخواست های ارسالی از کلاینت قرار است پاسخ مناسب دهد و کارش را راه بیاندازد و همچنین اطلاعات را در دیتابیس خود ذخیره سازی کند. سرور با ساخت سوکتی منتظر پیغام کلاینت می‌ماند و وقتی که کلاینت پیغام خود را بدهد با تجزیه و تحلیل و یک سری عملیات به کلاینت جواب می دهد. دقت کنید که سرور هیچوقت exit نمیشود و باید 24 ساعته آنلاین باشد و جواب کاربران را بدهد بنابرین:

## تابع main:

در تابع main، ما بعد از وصل اولیه به کلاینت به یک حلقه بینهایت بدون وجود هرگونه break می‌رسیم. این برای این است که سرور در هر لحظه باید آماده پاسخ‌گویی باشد. تابع main چیز خاصی ندارد و هر بار فقط تابع Rec\_Req را صدا می‌زند که در آن تابع همه اتفاقات اصلی می افتد.

## تابع Rec\_Req:

تابع با دریافت اطلاعاتی مربوط به ساختن سوکت شروع می‌شود. با استفاه از تابع Sock\_Creator سوکت را ساخته و کلاینت وصل شده و درخواست آن را دریافت می‌کنیم.

بعد از دریافت درخواست متناسب با درخواست کاربر به یکی از توابع موجود مراجعه می کنیم و عملیات را با توجه به درخواست کاربر انجام میدهیم.

دست آخر نیز پیغام خود را برای کلاینت می فرستیم تا تکلیف کلاینت مشخص شده و ادامه کار خود را انجام دهد.

## تابع registering:

اولا که باید توجه داشت به خاطر وجود داشتن تابعی به نام register در توابع کتابخانه ای مجبور به تغییر نام این تابع و برای ریتمیک بودن با توابع دیگر به ته بقیه توابع نیز یک –ing اضافه شده است. در این تابع ما فایل مربوط به این کاربر را در پوشه ساخته شده users می سازیم. فایل ها به نام حساب کاربری درخواست شده ساخته می شوند، بدین گونه اگر فایلی با اسم مشابه وجود داشته باشد یعنی این اسم برای حساب کاربری دیگری گرفته شده و پیفام error برای کلاینت ارسال می‌شود. در غیر این‌صورت فایل با نام کاربر جدید ساخته شده و محتویاتش نام کاربری و رمز عبور کاربر است و سپس پیغام موفقیت فرستاده میشود.

## تابع logining:

دراین تابع ما چند چیز را چک میکنیم. اولا باید چک شود که کاربری با این اسم وجود دارد یا نه که اگر نداشت error مربوطه فرستاده می شود. در مرحله دوم نوبت به چک کردن password می رسد. با خواندن رمزعبور از فایل اگر مشکلی وجود داشته باشد خطای مربوطه فرستاده می شود و در غیر این صورت ها پیغام موفقیت فرستاده میشود.

در این جا ما با استفاده از تابع token\_creator توکن مورد نظر که در بخش فاز یک درباره آن توضیح داده شد را می سازیم. تابع token\_creator سرور ما رشته ای 32 رقمی تنها از حروف بزرگ انگلیسی را تشکیل می‌دهد. پیغام موفقیت نیز به صورت type=”Authtoken” و content=TOKEN فرستاده می شود. توکن و اسم یوزر کاربر که الان به وضعیت آنلاین درآمده نیز در یک خانه از struct users ذخیره میشود. اگر در درخواست کاربر نام کاربری ای وارد شود که اسمش از قبل در این استراکت هست یعنی آن کاربر آنلاین بوده از قبل و پیغام خطای مربوطه فرستاده می شود.

## تابع logouting:

در این تابع نیز با دریافت token مورد نظر خونه استراکتی که توکنش مشابه است را پیدا می کند و خالی می کند و پیغام موفقیت را برای کلاینت ارسال می‌کند.

## تابع createchanneling:

در اینجا ما فایلی را در پوشه channels میسازیم به نام خود channel. اگر وجود داشته باشد که خطای مربوطه ارسال و در غیر این صورت فایل ایجاد می شود. دقت کنید چند خط اول برای این است که بتوانیم کانال ها را با وجود space در اسمشان ذخیره کنیم. در واقع پس از عملیات رشته nametok که مقدار اولیه اش اسم و توکن بود تنها به اسم کانال و رشته toktok تبدیل به توکنمان خواهد شد. اولین پیام هر کانال نیز ساخته شدن کانال توسط سازنده‌اش است. پیغام ها به صورت آرایه ای از sender و content ها در فایل ذخیره می شوند. پیغام نخست Sender خود server و contentاش <username> created -<channel name>- خواهد بود.اگر عملیات موفق آمیز باشد پیغام موفقیت برای کلاینت ارسال خواهد شد. اسم کانال کاربر نیز در خانه struct مربوطه اش ذخیره می شود.

## تابع joinchanneling:

در این جا نیز شرایط به مانند create است و اگر فایلی با اسم مورد نظر وجود نداشته باشد پیغام خطا و اگر موفقیت آمیز باشد با sender خود server و content به شکل <username> joined -<channel name>- در فایل ذخیره می شود. دقت کنید که فایل باری دیگر از اول رونویسی می شود و این پیغام به پیغام های قبلی درون فایل چسبیده می شود و سپس روی فایل قبلی نوشته می شوند. اسم کانال کاربر میز در خانه struct مربوطه اش ذخیره می شود.

## تابع leaving:

در این تابع ما از کانال بیرون می آییم. تابع با دریافت توکن در struct مربوطه اسم کانال را پاک میکند همچنین previous\_arraaynum نیز در اینجا صفر می شود که اشاره به این دارد که تا کدام پیام در refresh دیده شده بود و با leave دادن از کانال دوباره این پروسه reset می شود و اگر باری دگر در کانالی عضو شوید از اول پیام ها refresh میکند. همچنین در فایل کانال به شکل sender خود server و content به شکل <username> left -<channel name>- پیامی در فایل مربوطه کانال ذخیره می‌شود. درآخر اگر عملیات موفقیت آمیز بود پیغام success به کلاینت برگردانده می شود.

## تابع messagesending:

تابع خیلی شبیه join channel عمل می کند و فقط فرقش اینه که به جای پیغام جوین شدن، متن دریافتی از کاربر را در فایل با sender به نام کاربر فرستنده پیام ذخیره می‌کند.

## تابع channelmembering:

این تابع با استفاده از structای که ساخته بودیم می رود و در بین خونه های struct آن هایی که channelشون با کانال کسی که توکن اش فرستاده شده(شامل خود همان کاربر) به صورت آرایه ای از اسم ها به کلاینت پس داده میشود.

## تابع refreshing:

در این تابع اطلاعات موجود در فایل که شامل پیام ها است خوانده شده و از پیغام شماره previous\_arraaynum که دارای شماره آخرین پیغامی است که فرستاده شده به بعد را در آرایه ای شامل فرستنده پیام به همراه خود پیام به کلاینت فرستاده می شود و اگر پیام جدیدی موجود نباشد NULL می فرستد در آرایه.

## تابع findmembering:

در این تابع ما با پیدا کردن کسایی که کانال یکسانی با ممبر درخواست کننده دارند، به دنبال کسایی که در struct دارای channel یکسان هستند می گردیم اگر اسم خواسته شده با اسم یکی از کسانی که در ُstruct اسم کانالشان شرایط را دارا بود یکی باشد آنگاه به کلاینت content با مضمون True و در غیر این صورت False برگردانده می شود. دقت شود pathg که در چند تابع اخیر استفاده شد دارای مسیر کانال آخرین کسی است که عضوئ کانال شده تا نیاز به ساختن دوباره آن نباشد.