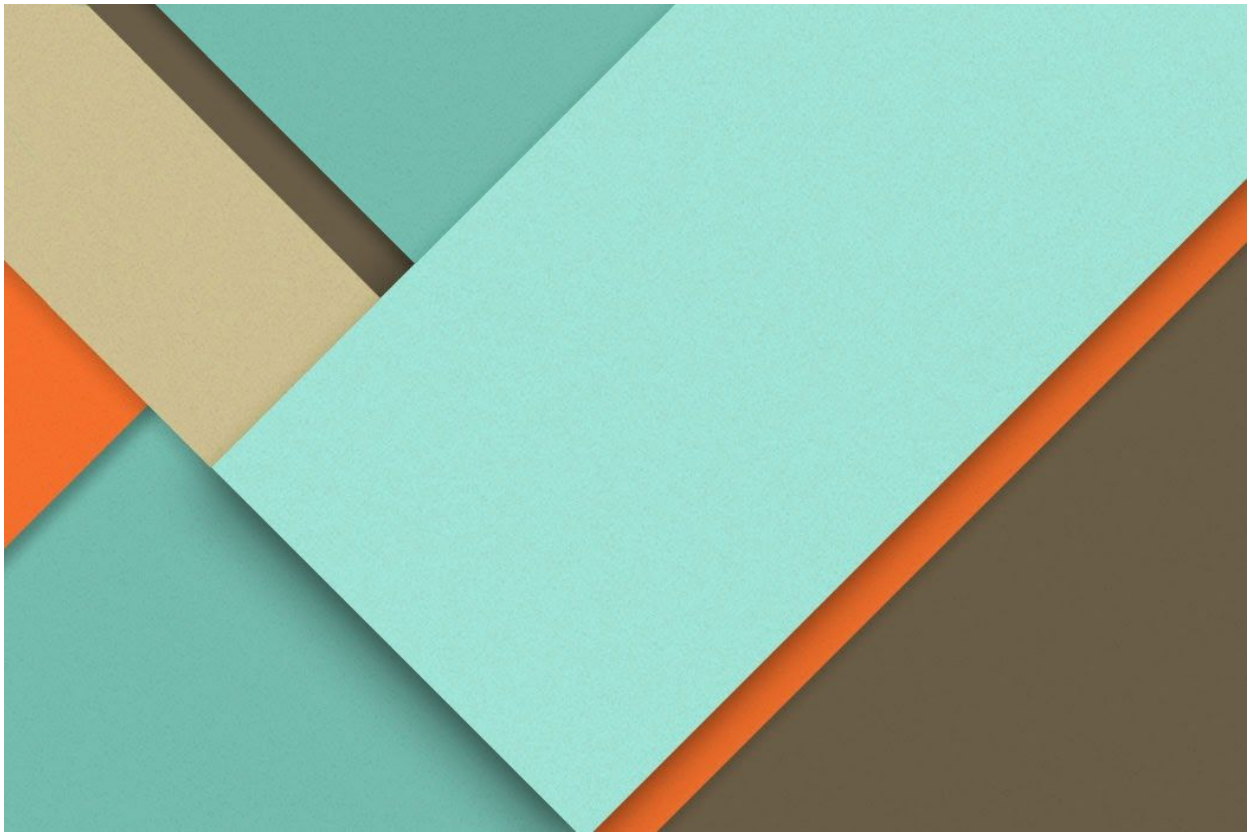

پروژه مبانی برنامه سازی

نیمسال اول 98-99

دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف

فاز2 : چت اپلیکیشن (سرور)

فهرست

۰.مروری بر فاز ۱، شروعی بر فاز ۲!

همانطور که احتمالا می‌دانید و از تیتر این داک هم پیداست، در این فاز از پروژه باید سرور چت‌اپلیکیشن خود را بنویسید. همان سروری که در فاز ۱ ما در اختیار شما قرار دادیم. حالا قرار است که بعد از کلاینت، سرور را هم خودتان بسازید تا چت‌اپلیکیشنتان آماده‌ی بهره برداری و افتتاح شود!

در فاز اول، روال کار به این صورت بود که کلاینت شما اطلاعات را از کاربر می‌گرفت و آن را در قالبی خاص برای سرور می‌فرستاد و سرور هم پاسخ مناسب را به کلاینت می‌داد. حال باید سرور شما نیز فرایندی را مشابه سروری که به شما دادیم، طی بکند. یعنی بعد از دریافت هر درخواست از کلاینت، پاسخ مناسب را ارسال کند. همچنین متناسب با درخواست ارسال شده توسط کلاینت، گاهی لازم است سرور تغییراتی در داده‌هایی که در خود نگه می‌دارد اعمال کند. به این داده‌ها در اصطلاح پایگاه داده (Database) گفته می‌شود. در ادامه به شرح جزییات بیشتری از این فاز خواهیم پرداخت.

۱.پایگاه داده:

سرور باید در پایگاه داده‌ی خود یا همان دیتابیس‌اش، اطلاعاتی را نگه‌دارد. تا زمانی که سرور قطع شد، با وصل کردن دوباره‌ی آن، حساب‌های کاربری و کانال‌های ایجاد شده از بین نروند. این اطلاعات شامل لیست کاربرها، لیست کانال‌ها و پیام‌های ارسال شده در هر کانال هستند. توجه کنید که نیازی به ذخیره‌ی اعضای کانال در دیتابیس نیست. زیرا با قطع شدن سرور، عملا دیگر هیچ کاربری هم آنلاین نیست که بخواهد در کانالی باشد. پس داشتن لیستی از اعضای کانال زمانی کاربرد دارد که سرور بالا باشد. سرور باید این اطلاعات را در چند فایل ذخیره کند و در زمان مناسب، تغییراتی در آن‌ها اعمال کند. در واقع مدیریت دیتابیس از کارهایی‌ست که سرور باید انجام دهد.

۲. دریافت درخواست از کلاینت و ارسال پاسخ به آن

همانطور که در بخش قبل نیز گفته شد، ارسال درخواست در این فاز مانند فاز ۱ و در همان قالب‌ها صورت می‌گیرد. سرور شما در وهله‌ی اول باید بتواند درخواست‌هایی که کلاینت‌ها برایش می‌فرستند را دریافت کند. در وهله‌ی بعدی، سرور باید با توجه به درخواست ارسال شده، کارهایی بکند و سپس پاسخ مناسب را برگرداند. توجه کنید که پاسخ‌های سرور شما باید همانند سرور فاز ۱ به صورت JSON باشد.

فرم کلی پاسخ سرور به شکل زیر است:

{"<type": "<response type>", "content": <response content"}

۲.۱. پیغام خطا

بعد از دریافت هر درخواست، اگر خطایی وجود داشته باشد، سرور باید پاسخی به فرم زیر ارسال کند.

{"type": "Error", "content": "<error_message>"}

۲.۲. ثبت کاربر جدید

فرمت درخواست کلاینت برای ثبت یک کاربر جدید به شکل زیر بود:

register <username>, <password>

سرور با دریافت این درخواست، ابتدا نام کاربری ارسال شده در درخواست را در بین نام‌های کاربری ثبت شده در دیتابیس جست‌وجو می‌کند. اگر نام کاربری وجود داشت، پیغام خطا برمی‌گرداند و اگر قبلا کاربری با این نام ثبت نام نکرده باشد، اطلاعات کاربر جدید را به لیست کاربرها اضافه می کند و یک پیام اجرای موفق به کلاینت بر می گرداند. پاسخ سرور در صورت موفقیت آمیز بودن ثبت کاربر جدید، به شکل زیر خواهد بود:

{"type": "Successful", "content": ""}

۲.۳. ورود کاربر

درخواست کلاینت برای ورود به حساب کاربری در قالب زیر بود:

login <username>, <password>

پس از دریافت این درخواست، سرور در دیتابیس به دنبال کاربری با نام کاربری ارسال شده می‌گردد. اگر کاربری وجود نداشت، پیغام خطا برمی‌گرداند. اگر کاربری با آن نام کاربری پیدا شد ولی رمز عبور اشتباه وارد شده بود، باز هم پیغام خطایی متناسب به کلاینت می‌فرستند. اگر کاربر مورد نظر وجود داشت و رمز عبور هم درست بود، auth token جدید و یکتایی تولید می‌کند آن را به کلاینت برمی‌گرداند.

{"type": "AuthToken", "content": "<Auth Token>"}

توجه کنید که توکن تولید شده، شناسه‌ی هر کاربر آنلاین است.

۲.۴. ایجاد کانال جدید

قالب درخواست کلاینت برای ایجاد یک کانال تازه به شکل زیر بود:

create channel <channel_name>, <auth_token>

درخواست‌هایی که در آن‌ها auth token هم فرستاده می‌شود، اولین مرحله چک کردن auth token می‌باشد. از اشاره به این مرحله در درخواست‌های دیگر خودداری می‌کنیم. اگر کاربری با آن توکن وجود نداشت، پیغام خطا برمی‌گرداند. اگر قبل تر کانالی با این نام در دیتابیس وجود داشت، پیغام خطای مناسب را برمیگرداند. در غیر این صورت یک کانال می‌سازد و آن را به لیست کانال‌ها اضافه می‌کند و پیام موفقیت آمیز بودن فرایند را برمی‌گرداند:

{"type": "Successful", "content": ""}

۲.۵. اضافه شدن به کانال

درخواست برای اضافه شدن به یکی از کانال های موجود با فرم زیر صورت می‌گرفت.

join channel <channel_name>, <auth_token>

سرور Auth token را بررسی می‌کند. اگر کانالی با این نام وجود نداشت پیغام خطای مناسب را برمی‌گرداند. در غیر اینصورت، کاربر را به اعضای کانال اضافه می‌کند و پاسخی به نشانه‌ی موفقیت آمیز بودن فرایند به کلاینت ارسال می‌کند.

{“type”: “Successful”, “content”: “”}

۲.۶. ارسال پیام در کانال فعلی

ارسال پیام جدید با درخواست زیر از سرور صورت می‌گرفت.

send <message>, <auth_token>

ابتدا اعتبار auth token بررسی می شود. همچنین اگر کاربر در کانالی عضو نبود پیغام خطای مناسب برگرداند. در غیر اینصورت پیام را به لیست پیامهای کانال اضافه کند و پیام اجرای موفق برگرداند:

{“type”: “Successful”, “content”: “”}

۲.۷. تازه سازی

کلاینت برای دریافت پیام‌های دیده نشده از سرور، ار درخواست زیر استفاده می‌کرد.

Refresh <auth_token>

سرور Auth token را بررسی می‌کند. اگر کاربر در کانالی عضو نبود، پیغام خطای مناسب فرستاده شود. در غیر این صورت، پیام‌هایی که کاربر فرستنده‌ی درخواست ندیده‌است را در قالب زیر برایش می‌فرستد.

```
{
        “type”: “List”,
        “content”: [
        {“sender”: “<username>”, “content”: “<message_content>”},
        {“sender”: “<username>”, “content”: “<message_content>”},
        {“sender”: “<username>”, “content”: “<message_content>”},
```

...

]

}

۲.۸. لیست اعضای کانال

برای گرفتن لیست اعضای فعلی کانال، باید درخواستی به شکل زیر به سرور ارسال می‌شد.

channel members <auth_token>

اگر کاربر در کانالی عضو نبود، پیغام خطای مناسب را می‌فرستند. در غیر اینصورت، لیست افراد کانال را در قاب زیر ارسال می کند.

{“type”: “List”, “content”: [“<username1>”, “<username2>”, “<username3>”, ...]}

۲.۹. خروج از کانال

خروج از کانال با درخواست زیر انجام می‌شدو

leave <auth_token>

ابتدا سرور بررسی می‌کند که آیا کاربر فرستنده‌ی درخواست اکنون در کانالی عضو هست یا نه. اگر عضو نبود، پاسخی به عنوان خطا در خروج از کانال به کلاینت می‌فرستد. اگر کاربر در یکی از کانال‌ها حضور داشت، سرور کاربر فرستنده را از اعضای کانال حذف کرده و پاسخ اجرای موفقیت آمیز فرایند خروج از کانال را ارسال می‌کند.

{“type”: “Successful”, “content”: “”}

۲.۱۰. خروج کاربر

برای خروج از اکانت کاربری از درخواست زیر استفاده می‌کردیم.

logout <auth_token>

بعد از بررسی auth token، سرور توکن کاربر را از بین می‌برد و کاربر را از کاربرهای آنلاین حذف می‌کند. پاسخ سرور در صورت انجام درخواست خروج به شکل زیر است.

{“type”: “Successful”, “content”: “”}