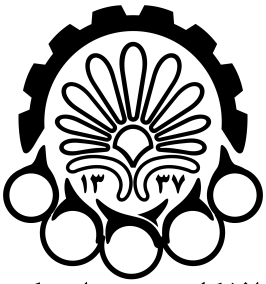
به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

برنامه‌نویسی وب (پاییز ۱۴۰۲)

پروژه پایانی

(فاز های اول و دوم)

پیام‌رسان تحت وب

استاد درس:

آقای مهندس الوانی

ددلاین معنوی:

ساعت ۲۳:۵۹ روز ۸ دی ۱۴۰۲

پیام‌رسان تحت وب

هدف اصلی این پروژه آشنایی با روند کامل توسعه و پیاده‌سازی یک‌اند و فرانت‌اند یک web app، در قالب یک پیام‌رسان تحت وب، مشابه با پلتفرم‌های محبوبی مانند تلگرام، است. در این فایل به توضیح کامل یک‌اند در قالب دو فاز پرداخته می‌شود و در ادامه فایل توضیحات بخش فرانت‌اند را نیز دریافت خواهید کرد. این بخش به صورت مجزا ددلاین و تحویل ندارد اما برنامه زمانی پیشنهادی خود را در قالب ددلاین معنوی مطرح کردیم تا ذهنیتی از زمان مورد نیاز برای توسعه یک‌اند داشته باشید. ددلاین نهایی با توجه به آخرین مهلت قطعی کردن نمرات اعلام خواهد شد. توصیه می‌شود ابتدا یک بار دستورکار را به صورت کامل مطالعه کنید و سپس پیاده‌سازی را شروع کنید (در غیر این صورت نکات تحویل را حتما مطالعه کنید).

تعریف کلی از فازهای پروژه

فاز اول

در اولین فاز از طراحی سیستم پیام‌رسان‌مان می‌خواهیم به سراغ پیاده‌سازی بخش منطقی پروژه (یک-اند) برویم. از آنجایی که اکثر سیستم‌های تحت‌وب نیاز به مدیریت داده‌های کاربران و ذخیره‌سازی آن‌ها در یک پایگاه داده دارند، وجود یک مولفه برای برطرف کردن این نیازمندی‌ها ضروری است. البته این مولفه صرفا برای ذخیره‌سازی داده‌های کاربران مورد استفاده قرار نمی‌گیرد و با توجه به کاربردهای سیستم می‌تواند کارهای دیگری مانند احراز هویت کاربران، فرستادن ایمیل، ایجاد سطح دسترسی برای داده‌ها و ... را انجام دهد.
روش‌های زیادی برای پیاده‌سازی این مولفه وجود دارد؛ اما در این پروژه ما می‌خواهیم برای مولفه منطقی سیستم یک سرور HTTP پیاده‌سازی کنیم. شما با پروتکل HTTP در درس شبکه‌های کامپیوتری آشنا شده‌اید؛ همچنین در رابطه با نحوه ارسال درخواست HTTP، ویژگی‌های آن درخواست و دریافت پاسخ آن آشنا هستید.
در فاز اول ما از شما می‌خواهیم تا با استفاده از زبان برنامه‌نویسی دلخواه خود، یک سرور HTTP را پیاده‌سازی کنید که درخواست‌های HTTP را دریافت کند و بر اساس ویژگی‌های هر درخواست عملی را انجام دهد.
شما باید در این سرور HTTP تعدادی اندپوینت (endpoint) تعریف کنید. این اندپوینت‌ها در اصل توابعی هستند که به ازای درخواست‌های ارسال شده به سرور صدا زده می‌شوند. مدیریت منطق سیستم در این توابع پیاده‌سازی می‌شود. این توابع با توجه به سناریوهای کاربران که نحوه چگونگی استفاده از سیستم را مشخص می‌کنند، باید پیاده‌سازی شوند.
در این فاز در ابتدا در رابطه با مدل‌های سیستم توضیحی ارائه می‌دهیم. در ادامه لیستی از اندپوینت‌های موردنیاز سیستم را تعریف می‌کنیم و در آخرین بخش محدودیت‌ها و سناریوهای کاربران سیستم را شرح می‌دهیم.

فاز دوم

در فاز دوم پروژه و بخش دوم پیاده سازی یک-اند، می خواهیم یک کانکشن دو سویه (Bidirectional) با استفاده از وب سوکت (web-socket) ایجاد کنیم که امکان ارتباط کلاینت با سرور و برعکس را به صورت همزمان فراهم می‌سازد.
در پایان این بخش می‌خواهیم کاربر قادر باشد به کاربر دیگر پیام متنی ارسال کند، وضعیت آنلاین بودن کاربر دیگر را دریافت کند و در یک چت فعال indicator درحال تایپ بودن را ارسال و دریافت کند.
(اضافه کردن قابلیت ارسال فایل به کاربر دیگر نمره امتیازی دارد)

شرح کامل فازهای پروژه

دستور فاز اول

پایگاه داده و تعریف ساختار داده‌های سیستم

در اولین بخش از کار باید مدل‌های پایگاه‌داده سیستم را طراحی کنید. برای راحت‌تر شدن کار شما، ما یک ساختار پیشنهادی و کلی برای مدل‌های مورد نیاز سیستم تعریف کرده‌ایم. الزامی ندارد که مدل‌های شما به این شکل طراحی شوند، اما باید ویژگی‌های مدل‌های تعریف شده را شامل شوند.

حساب کاربری

کاربر یکی از اصلی‌ترین اجزای پیام‌رسان ما است. هر کاربر باید برای ورود یک حساب کاربری داشته باشد، که شامل مشخصات آن فرد می‌باشد. این اطلاعات شامل نام، نام خانوادگی، شماره تلفن منحصر به فرد کاربر، نام کاربری یکتا، عکس حساب کاربری و بیوگرافی فرد است. توجه کنید که عکس پروفایل کاربر باید در یک جایی از سرور قرار بگیرد و در ادامه لینک آن در پایگاه داده ذخیره شود.

- id: integer
- firstname: string
- lastname: string
- phone: string (unique)
- username: string (unique)
- password: string [hashed]
- image: string (unique)
- bio: string

کلمه عبور باید با الگوریتم‌های ایمن رمزنگاری شده باشد. ذخیره متن خام کلمه عبور در پایگاه داده یک اشتباه بزرگ امنیتی است. (اندکی عمیق‌تر: الگوریتم‌های رمزنگاری ایمن بر اساس سرعت عملکرد و بهینگی به دو دسته کلی سریع و کند تقسیم می‌شوند. استفاده از الگوریتم‌های کند ولی با پیچیدگی محاسباتی بالا در ذخیره سازی اطلاعات بیشتر توصیه شده است. دلیل این توصیه سخت‌تر شدن کار شکستن کلمه‌های عبور در مواقع لو رفتن پایگاه داده است. توضیحات بیشتر درمورد جزئیات این الگوریتم‌ها را در این لینک می‌توانید پیدا کنید)

لیست مخاطبین

هر کاربر دارای یک لیست مخاطبین است که مشاهده و یا آغاز چت را برایش آسان‌تر می‌کند. کاربر برای هر مخاطب خود یک نام انتخاب می‌کند. کلید این جدول composite بوده و شامل آیدی کاربر و مخاطب او است. (اضافه کردن قابلیت‌های حریم شخصی مانند نمایش و عدم نمایش شماره، عکس پروفایل و ... جزو موارد امتیازی است)

- user_id: integer
- contact_id: integer
- contact_name: string

چت

هر کاربری که بخواهد با کاربر دیگر ارتباط برقرار کند، باید یک چت برای این ارتباط ایجاد کند. هر چت مختص دو نفر از کاربران سیستم است و برای تعداد بیشتر از دو نفر باید یک گروه یا کانال ایجاد شود که در بخش‌های بعدی به آن‌ها می‌پردازیم. این چت‌ها باید یک مشخصه یکتا، مولفه‌هایی برای مشخص کردن طرفین چت و زمان ایجاد شدن آن چت را دارا باشند. دو کاربر می‌توانند تنها یک چت با یک دیگر داشته باشند. لازم به ذکر است که یک چت شامل تعدادی پیام است. ایجاد رابطه بین جدول چت و کاربر (در صورت استفاده از پایگاه‌داده رابطه‌ای) دارای نمره امتیازی است.

- ID: integer

- People: array of integers
- Created_at: date

پیام

بین هر دو کاربر موجود در سیستم یک سری پیام رد و بدل می‌شود. این پیام‌ها شامل یک آیدی برای مشخص کردن پیام، مولفه‌ای برای اینکه مشخص شود این پیام متعلق به کدام چت می‌باشد، مولفه‌ای برای مشخص کردن فرستنده پیام، مولفه‌ای برای مشخص کردن گیرنده پیام، محتوای پیام و زمان ارسال آن است. توجه کنید که پیام‌ها می‌توانند مختص به یک گروه یا کانال نیز باشند؛ برای مدیریت کردن آن‌ها می‌توانید از ویژگی‌های گیرنده یا فرستنده پیام استفاده کنید.

- ID: integer
- Chat_id: integer
- Sender: integer
- Receiver: integer
- Content: string
- Created_at: date

برای استفاده از پایگاه داده سیستم شما می‌توانید از پایگاه داده‌های زیر استفاده کنید.

- SQL server
- MySQL
- MariaDB
- PostgreSQL
- MongoDB
- Cassandra

دوستانی که از پایگاه داده NoSQL استفاده می‌کنند، برای ارزیابی پروژه باید ساختار مدل‌های خود را در فایلی جدا بنویسند.
در نظر گرفتن مقیاس پذیری و انتخاب درست زیرساخت ذخیره سازی و دریافت/ارسال اطلاعات با در نظر گرفتن محدودیت‌ها و شرایط دارای نمره امتیازی است.

تعریف endpointهای سرور HTTP برای عملیات CRUD

حال که مدل‌های پایگاه داده را طراحی کرده‌اید، باید اندپوینت‌های سرور HTTP را نیز طراحی و پیاده‌سازی کنید. در ابتدا توضیح دادیم که برای ارتباط با سرور و مدیریت داده‌ها باید یک سری توابع را تعریف کنیم که به ازای هر درخواست HTTP یکی از این توابع صدا زده می‌شود. در این بخش ما جزئیات این درخواست‌های HTTP را تعریف می‌کنیم. ما در این بخش فقط متد و مسیر درخواست HTTP را تعریف می‌کنیم. نحوه پیاده‌سازی توابع مرتبط با هر درخواست به عهده خودتان است. ورودی‌های سیستم نیز با توجه به پیاده‌سازی خودتان ممکن است متفاوت باشند، اما خروجی‌های سیستم باید به فرم JSON باشند. همچنین لازم است که تمام اطلاعات مدل‌ها در خروجی قابل دیدن باشد.

برای شرح اندپوینت‌ها ما آن‌ها را دسته‌بندی می‌کنیم. هر دسته بر اساس یک مدل پایگاه داده در نظر گرفته شده است. به عنوان مثال اندپوینت‌هایی که نیازمند اجرای عملیات بر روی مدل حساب کاربری هستند، در دسته users قرار گرفته‌اند. ما به هر یک از این دسته‌ها resource group می‌گوییم. برتری این روش پیاده‌سازی، قابل فهم بودن وظیفه هر اندپوینت از روی مسیر و متد درخواست آن می‌باشد. همچنین منطق پیاده‌سازی هر بخش می‌تواند مجزا از بخش‌های دیگر باشد و این ویژگی در فرآیند توسعه کار را راحت‌تر می‌کند.

اندپوینت‌های کاربر

ایجاد حساب کاربر

Method: POST
Path: /api/register

ورود کاربر

Method: POST
Path: /api/login

دریافت اطلاعات حساب کاربری

Method: GET
Path: /api/users/{user_id}

تغییر حساب کاربری

Method: PATCH
Path: /api/users/{user_id}

پاک کردن حساب کاربری

Method: DELETE
Path: /api/users/{user_id}

جستجوی یک کاربر با استفاده از یک کلید واژه

Method: GET
Path: /api/users?keyword=”key”

اندپوینت‌های لیست مخاطبین

دریافت لیست مخاطبین کاربر

Method: GET
Path: /api/users/{user_id}/contacts

اضافه کردن مخاطب جدید به لیست مخاطبین

Method: POST
Path: /api/users/{user_id}/contacts

حذف کردن یک مخاطب از لیست مخاطبین

Method: DELETE
Path: /api/users/{user_id}/contacts/{contact_id}

اندپوینت‌های چت

ایجاد چت

Method: POST
Path: /api/chats

دریافت لیست چت‌ها

Method: GET
Path: /api/chats

دریافت یک چت به همراه محتوای آن

Method: GET
Path: /api/chats/{chat_id}

حذف یک چت

Method: DELETE
Path: /api/chats/{chat_id}

پاک کردن پیام ارسال شده

Method: DELETE
Path: /api/chats/{chat_id}/messages/{message_id}

اندپوینت‌های گروه (امتیازی)

ایجاد گروه

Method: POST

Path: /api/groups

حذف گروه

Method: DELETE
Path: /api/groups/{group_id}

اضافه کردن کاربر جدید به گروه

Method: PATCH
Path: /api/groups/{group_id}

حذف کردن کاربر از گروه

Method: DELETE
Path: /api/groups/{group_id}/{user_id}

احراز هویت کاربران با استفاده از JWT

JSON Web Token یا به اختصار JWT یک استاندارد باز برای ارسال اطلاعات به شکل مستقیم و امضاشده بین دو طرف است. در مفهوم احراز هویت، JWT به‌طور معمول برای ارسال اطلاعاتی مانند اطلاعات کاربر و اطلاعات مربوط به احراز هویت استفاده می‌شود. برای آشنایی با ساختار آن کد و دیکد کردن نمونه داده‌هایی می‌توانید از این سایت شروع کنید.

نحوه عملکرد JWT برای احراز هویت

1. **ورود به سیستم:**
   - کاربر با ارسال اطلاعات احراز هویت (مثلاً نام کاربری و رمز عبور) به اندپوینت ورود به سیستم درخواست ارسال می‌کند.
2. **احراز هویت:**
   - در سمت سرور، اطلاعات ارسال شده توسط کاربر چک می‌شود (بررسی صحت نام کاربری و رمز عبور).
   - اگر اطلاعات صحیح باشند، یک JWT با اطلاعات مربوط به کاربر (مانند شناسه کاربری) ایجاد می‌شود.
   - توکن JWT امضا می‌شود تا اطمینان حاصل شود که توسط سرور ایجاد شده است.
3. **ارسال توکن به کاربر:**
   - توکن JWT به کاربر ارسال می‌شود. این توکن به‌طور معمول در سرآیند درخواست‌های آینده قرار می‌گیرد.
4. **احراز هویت در درخواست‌های آینده:**
   - کاربر با ارسال درخواست به سرور، توکن JWT را در سرآیند درخواست قرار می‌دهد.
   - سرور از امضای توکن بررسی می‌کند و اطلاعات کاربری را از داخل توکن استخراج می‌کند.
   - اگر توکن معتبر باشد و اطلاعات کاربری به درستی استخراج شود، کاربر به درخواست دسترسی دارد.

نکات مهم:

- توکن JWT حاوی اطلاعات قابل خواندن است و می‌تواند به‌صورت Base64 درخواست‌ها ارسال شود.
- اطلاعات موجود در توکن JWT قابل خواندن است و اطلاعات حساس را در آن قرار ندهید. (می‌توان آنها را نیز رمزنگاری کرد تا قابل خواندن برای کسانی که کلید مورد نیاز را ندارند قابل خواندن نباشد (JWE) ولی پیش‌فرض در JWT اطلاعات کاملا قابل خواندن است)
- امنیت توکن از اهمیت بسزایی برخوردار است. کلید مخفی برای امضای توکن باید محفوظ نگهداری شود(به طور مثال در یک keystore).
- این توکن به دلیل قرار گرفت در سرآیند درخواست محدودیت اندازه دارد و بهتر است حجم اطلاعات درون آن تا اندازه ممکن کم باشد.
- می توان از cookieهای مرورگر برای ذخیره این توکن در مرورگر کاربر استفاده کرد اما استفاده پرچم‌ها ایمن‌سازی آن مورد نیاز آن خواهد بود تا حدود زیادی کاربر از حملاتی نظیر XSS، MITM و ... در امان باشد.

نحوه استفاده از JWT در پروژه

1. **وارد کردن JWT در توکن‌ها:**
   - در مرحله ورود به سیستم، یک تابع باید توکن JWT را بسازد و به کاربر ارسال کند.
2. **بررسی توکن در هر درخواست:**
   - در هر درخواست ارسالی از کاربر، سرور باید توکن JWT را از سرآیند درخواست یا مکان مشخصی (با کم‌ترین ریسک ممکن) بررسی کند.
   - اگر توکن معتبر بود، سرور می‌تواند اطلاعات موجود در آن را استخراج کند و به درخواست دسترسی بدهد.
3. **تنظیم انقضا توکن:**
   - توکن‌ها باید یک زمان انقضای مشخص داشته باشند تا از مشکلات امنیتی جلوگیری شود. توکن‌های JWT به طور پیش‌فرض خودبه‌خود منقضی می‌شوند و در صورت نیاز به انقضای اجباری/دستی آن توسط کاربر نیاز به فرایند بررسی جداگانه ای توسط سرور/برنامه احراز هویت کاربر در مرحله قبل است.

امتیازی: استفاده از مکانیزمی برای مقابله با حمله CSRF که اجازه می‌دهد هکر با استفاده از درخواست جعل شده در مرورگر کاربر خود را جای او جا بزند.

امتیازی: پیاده سازی بخش سرور برای استفاده از واسطهای ذخیره و بازیابی کلید از طریق مرورگرها برای تشخیص هویت (authentication)؛ از طریق secure element گوشی‌ها یا windows hello در سیستم‌عامل ویندوز و ...، که با یکبار ذخیره رمز یا توکن کاربر در فضای امن سیستم کاربر، اجازه تشخیص هویت یکپارچه را به کاربر شما را می‌دهد (استفاده از تصویر، اثرانگشت و توکن‌های سخت‌افزاری و ...). نمونه متن باز این فرایند را در این سایت می‌توانید مشاهده کنید.

با این توضیحات، شما می‌توانید JWT را به عنوان یک ابزار برای احراز هویت در پروژه‌ی خود استفاده کنید.

دستور فاز دوم

چرا websocket؟

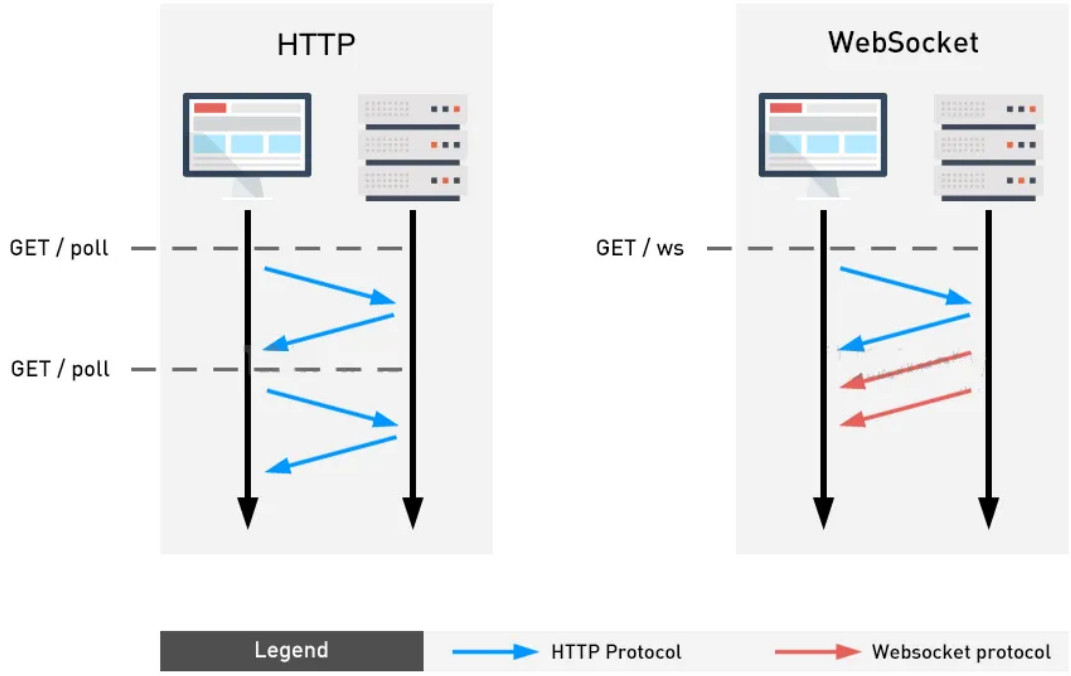
برای پاسخ به این سوال باید HTTP و websocket را مقایسه کنیم تا ببینیم برای یک برنامه چت انلاین کدام مناسب‌تر است.

- HTTP request همواره از طرف کاربر ارسال میشود.
- هر درخواست تنها یک پاسخ از سرور میگیرد و پس از آن کانکشن بین کلاینت و سرور بسته میشود.
- HTTP پروتکلی stateless است و اطلاعاتی درمورد درخواست‌های قبلی یا استیتی که کلاینت مشاهده میکند، نداریم.

در نتیجه، در صورت استفاده از HTTP، اگر یک نفر بخواهد صد پیام برای کاربر دیگر بفرستد، باید صد request جداگانه بفرستد و همچنین کاربر دوم باید هر بار برای دیدن پیام صفحه خود را رفرش کند! در یک برنامه چت انلاین چنین چیزی اصلا خوشایند نیست.

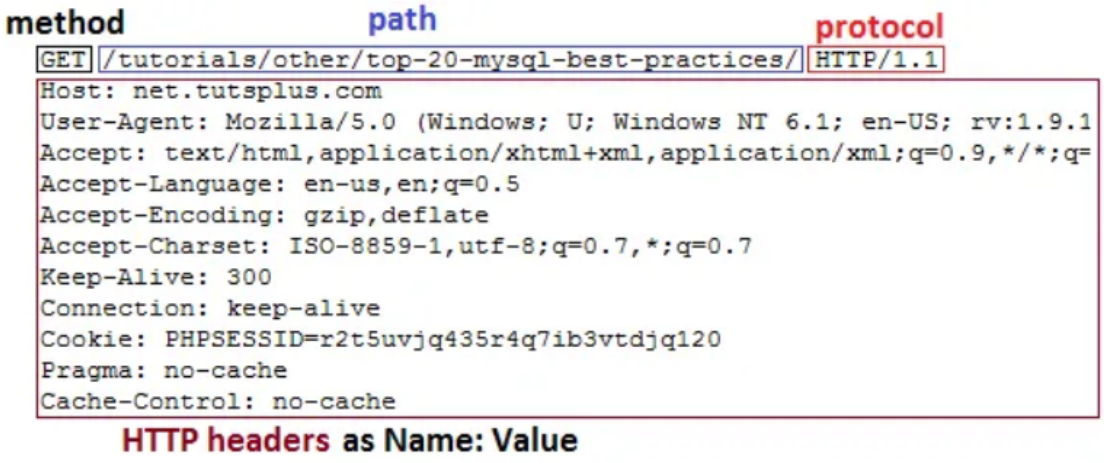
ما می خواهیم سرور از استیت کلاینت با خبر باشد و سرور و کلاینت هرکدام مستقل از هم و بدون نوبت بتوانند request ارسال و response دریافت کنند.

در تصویر زیر می توانید رفتار هرکدام را مشاهده و مقایسه کنید.

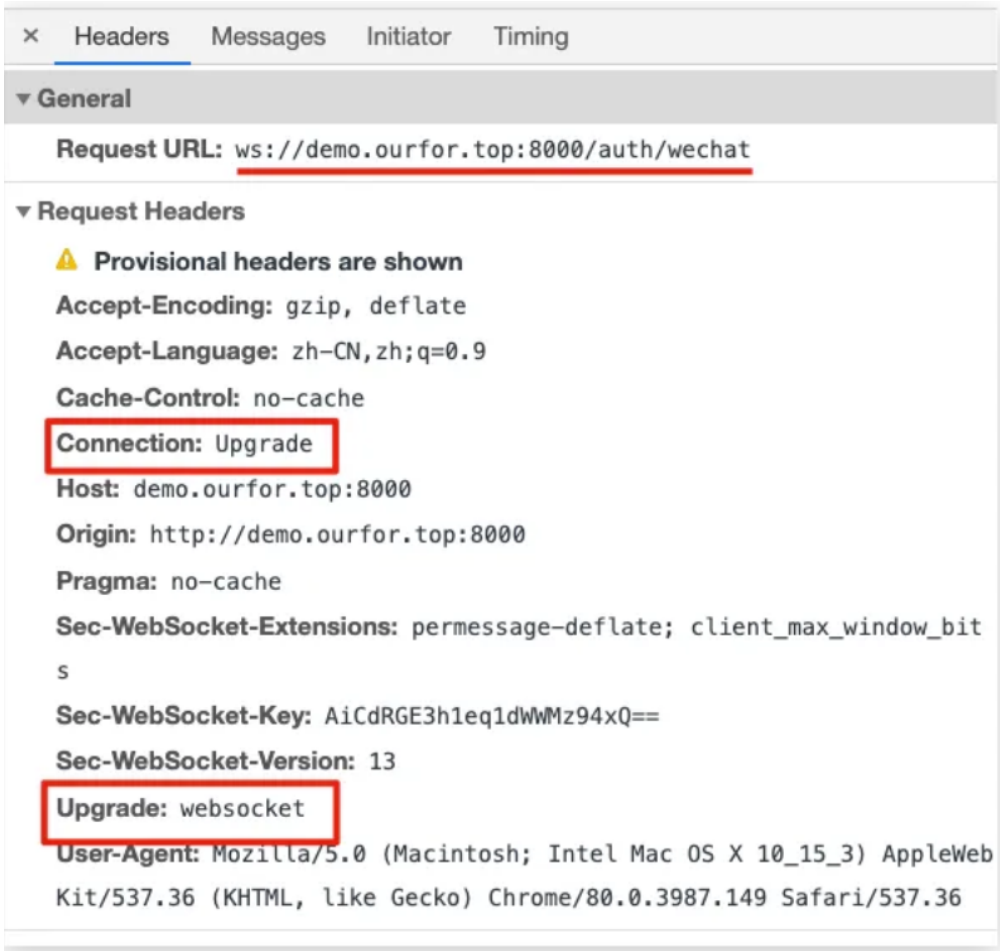


همانطور که مشخص است، وب سوکت‌ها اجازه می‌دهند بدون ارسال درخواست از سوی کلاینت، اتصال بین کلاینت و سرور باز بماند در حالی که با HTTP برای باز ماندن اتصال نیاز به ارسال دوباره درخواست از سوی کلاینت داریم.
در یک درخواست نرمال HTTP، کلاینت یک درخواست به سرور می‌فرستد و به سرور اطلاع میدهد که با یک ریسورس خاص میخواهد کاری انجام دهد. همچنین کلاینت اطلاعاتی در مورد اینکه url در کدام ریسورس یافت میشود ارسال میکند. این اطلاعات داخل درخواست هدر، همراه با جزئیات پروتکل HTTP مورد استفاده، ارسال می‌شود. سپس سرور پاسخی با یک استاتوس کد که اطلاعاتی در مورد موفقیت درخواست، ارسال محتوا و خود محتوا ارسال میکند.
تمام اینها روی یک سوکت TCP/IP انجام می شود. وب سوکت‌ها روی لایه TCP (یک پروتکل انتقال داده که وابسته به این است که دو هاست قبل از ارسال هرگونه داده ای ارتباط داشته باشند) ساخته شده اند و سوکت TCP / IP را تغییر می دهند تا کلاینت و سرور بتوانند روی باز ماندن سوکت توافق کنند.
باز نگه داشتن سوکت قدم اساسی در ممکن ساختن ارتباط bidirectional است. بدون آن، وب سوکت معنایی ندارد و استفاده از وب همچنان به استفاده از مراحل request-response که همیشه یوزر درخواستی برای انجام کاری به سرور می فرستد محدود می ماند.
تا زمانی که کانکشن برقرار است، تنها توافقی در مورد چگونگی جابجایی دیتا با هر دو سمت باقی می ماند. به این اصطلاحا handshake وب سوکت گفته می‌شود. handshake در websocket شبیه یک HTTP GET REQUEST است، با این تفاوت که این درخواست حاوی یک upgrade header نیز هست که درخواست می دهد سرور به یک پروتکل باینری سوئیچ کند و از وب سوکت استفاده کند و همچنین اطلاعاتی در مورد ارتباط کانکشن وب سوکت به همراه دارد.
برای درک و مقایسه بهتر چگونگی ارسال درخواست به تصاویر زیر توجه کنید:

این یک درخواست HTTP است که از HTTP 1.1 استفاده شده و در نتیجه کانکشن keep-alive است. این اساس اولیه وب سوکت است.



شکل زیر نمونه‌ای از upgrade header است. دقت کنید که url به جای $HTTP$: // با ws: // جایگزین شده که به این معناست که درخواست برای استفاده از پروتکل وب سوکت طراحی شده.



همانطور که دیدیم، می توان با ارسال یک GET ریکوئست کانکشن را به یک یا چند پروتکل آپگرید کرد.

GET /index.html HTTP/1.1
Host: www.example.com
Connection: upgrade
Upgrade: example/1, foo/2

توجه کنید که Connection: upgrade باید زمانی که Upgrade ارسال می شود، ارسال شود.
سرور میتواند به هر دلیلی ریکوئست را ایگنور کند و پاسخ دهد که هدر آپگرید ارسال نشده (مثلا با یک Upgrade پاسخ دهد) و یا در صورت تصمیم به آپگرید کانکشن، می تواند با یک هدر ۱۰۱ پاسخ دهد تا سوییچ کردن پروتکل و باز بودن وب سوکت را تایید کند،

HTTP/1.1 101 Switching Protocols
Upgrade: foo/2
Connection: Upgrade

برای توضیحات بیشتر می توانید به این لینک مراجعه بفرمایید.
و یا به ریکوئست اولیه یک پاسخ با پروتکل جدید بدهد. اکنون جابجایی در لحظه بین چندین کلاینت و سرور می‌تواند انجام شود.
به این صورت اگر درخواست موفق باشد، کانکشن وب سوکت ساخته شده و می توان برای ارسال پیام و statusها در لحظه از آن استفاده کرد.

سناریوها و محدودیت‌ها

در این قسمت یک سری از سناریوهای مهم سیستم را شرح می‌دهیم:

- کاربر یک درخواست ایجاد حساب کاربری ارسال می‌کند. اگر شماره تلفن کاربر یا نام کاربری اون از قبل در سیستم ثبت شده باشد، یک خطای مناسب بازگردانده می‌شود. اطلاعات درخواست کاربر در صورت نداشتن خطا در سیستم ذخیره می‌شوند و یک پیام مناسب بازگردانده می‌شود.
- کاربر برای ورود به سیستم نام کاربری و رمز عبور خود را ارسال می‌کند. اگر با اطلاعات درخواست، حساب کاربری یافت نشد باید خطای NotFound بازگردانده شود. در صورتی که حساب کاربری موجود بود اما رمز کاربر نادرست بود، باید خطای رمز نادرست بازگردانده شود. در صورتی که تمامی اطلاعات صحیح بودند، باید یک JWT در پاسخ بازگردانده شود. توجه کنید، کاربری که احراز هویت نشده است نباید بتواند هیچ یک از سناریوهای بعدی را اجرا کند، و باید در پاسخ به درخواست‌های آن شخص خطای Unauthorized بازگردانده شود.
- پس از ورود به سیستم، کاربر با استفاده از یک شماره تلفن یا نام کاربری، اطلاعات حساب کاربری دوست خودش را جستجو می‌کند. پس از یافتن آی‌دی آن شخص، یک چت برای شروع ارتباط برای او ایجاد می‌کند. اگر طرف دیگر ارتباط در سیستم وجود نداشت باید خطای NotFound بازگردانده شود.
- کاربر لیست مخاطبین خود را دریافت می‌کند. بعد از آن می‌تواند با استفاده از شماره تلفن و یا نام‌کاربری، یک کاربر جدید را به لیست مخاطبین خود اضافه کند. همچنین می‌تواند کاربری را از لیست خود حذف کند.
- کاربر لیست چت‌هایی که ایجاد کرده است را مشاهده می‌کند و می‌تواند پیام‌های هرکدام از آن‌ها را مشاهده کند.
- کاربر می‌تواند هرکدام از چت‌ها را (فقط به صورت دو طرفه، یعنی برای هر دو طرف ارتباط آن چت پاک می شود) پاک کند.
- کاربر می‌تواند وارد حساب کاربری خودش شود و اطلاعات شخصی خودش را تغییر دهد. (تمامی اطلاعات قابل تغییر هستند)
- کاربر می‌تواند حساب کاربری خودش را حذف کند. در نتیجه تمامی چت‌های کاربر باید حذف شوند.
- فرستادن پیام در فاز دوم پروژه انجام خواهد شد، اما در فاز اول کاربر می‌تواند یک پیام موجود را پاک کند.
- کاربر می‌تواند یک گروه جدید ایجاد کند.
- کاربر می‌تواند کاربران موردنظر خودش را به آن گروه اضافه کند یا از آن حذف کند.
- کاربر می‌تواند گروه و پیام‌های آن را حذف کند.

در ادامه لیستی از محدودیت‌های مورد نظر سیستم را با هم بررسی می‌کنیم:

- برای امنیت بیشتر سیستم، رمز عبور کاربران نباید کمتر از ۸ کاراکتر باشد.
- حجم عکس کاربران نباید بیشتر از یک مگابایت باشد.
- بیوگرافی کاربران نباید بیشتر از صد کاراکتر باشد.
- محتوای پیام‌های ارسالی نباید بیشتر از سیصد کاراکتر باشد.
- در صورتی که کاربر حساب کاربری خودش را پاک کند، چت‌های آن شخص با دیگر افراد نباید به صورت کامل حذف شوند؛ صرفا امکان ارسال پیام در آن چت نباید وجود داشته باشد.
- به هنگام ارسال هر درخواست، اگر JWT منقضی شود، باید خطای Unauthorized برگردانده شود.
- کاربران نمی‌توانند به چت‌هایی که متعلق به خودشان نیست دسترسی داشته باشند.
- کاربران نمی‌توانند اطلاعات حساب کاربری دیگر افراد را تغییر بدهند.
- کاربران نمی‌توانند لیست مخاطبین کاربران دیگر را مشاهده کنند یا آن‌ها را تغییر بدهند.
- کلید واژه‌ای که برای جستجوی کاربران مورد استفاده قرار می‌گیرد فقط شامل حروف و عدد باشد.

ارزیابی

برای ارزیابی این بخش موارد زیر مورد نظر است:

- تعریف کردن مدل‌های سیستم به شکلی که از شما درخواست کردیم
- اتصال صحیح سیستم به پایگاه داده و مدیریت داده‌های سیستم
- دلیل استفاده از پایگاه داده انتخابی
- تعریف endpointهای درخواستی
- آزمایش endpointها با استفاده از سناریوهای داده شده
- رعایت کردن محدودیت‌های سیستم
- نمایش خطاها و پیام‌های مناسب در هر سناریو
- استفاده از JWT برای احراز هویت کاربران
- قابلیت تنظیم کردن سرور قبل از اجرا
- استفاده از روش‌های monitoring برای بررسی وضعیت سیستم و دیباگ کردن خطاها (مانند log, tracing, metrics) (امتیازی)
- امکان بازارسال (فوروارد) پیام به کاربر دیگر
- دریافت پیام بدون نیاز به رفرش کردن صفحه
- دریافت و ارسال وضعیت آنلاین بودن
- دریافت و ارسال وضعیت در حال تایپ بودن برای چت فعال
- امکان ارسال و دریافت پیام در چت گروهی (امتیازی)

نکات تحویل

- تحویل تمامی فازهای پروژه (بکند و فرانت‌اند) همراه با هم انجام می‌شود.
- پروژه باید به درستی بر روی گیت قرار داده شده باشد.
- تقسیم وظیفه در پیاده‌سازی پروژه باید به شکل صحیح و برابر انجام شود.
- هر فرد حداقل باید بر روی کد خود مسلط باشد (تیم بر روی کل پروژه مسلط باشد).
- با توجه به زیاد بودن حجم پروژه، برنامه‌ریزی صحیح برای پیاده‌سازی آن داشته باشید.
- پیاده‌سازی خلاقانه پروژه و اضافه کردن قابلیت‌های جدید به آن دارای نمره امتیازی است اما جایگزین نمره بخش‌های اصلی نمی‌شود.