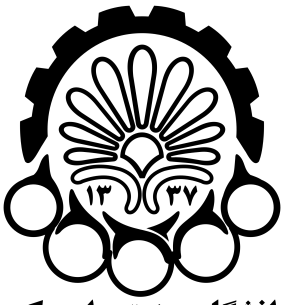
به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

برنامه‌نویسی وب (پاییز ۱۴۰۲)

پروژه پایانی

(فاز سوم)

پیام‌رسان تحت وب

استاد درس:

آقای مهندس الوانی

ددلاین معنوی:

ساعت ۲۳:۵۹ روز ۱۰ بهمن ۱۴۰۲

پیاده‌سازی فرانت‌اند

مقدمه

در این فاز از پروژه قصد داریم که کارکرد[1]ها و API های پیاده‌سازی‌شده در فازهای قبل (Backend) را به صورت بصری در مرورگر نمایش دهیم. برای این منظور لازم است تا از تکنولوژی‌ها و ابزارهای فرانت‌اند مانند HTML, CSS, JS که در درس با آن آشنا شدیم استفاده کنیم تا به صورت تعاملی و مناسب اطلاعات موردنیاز را به کاربر نمایش دهیم.

نکات: توجه داشته باشید که نیاز است تا برنامه پیام‌رسان شما شامل بک‌اند و فرانت‌اند به صورت یکپارچه و با ثبات طراحی شده باشد تا به خوبی کار کند و نیازمندی‌های مطرح‌شده را پاسخگو باشد.
در ضمن در نظر داشته باشید شما می‌توانید برای اجرای قابلیت‌های اضافی و موردنیاز برنامه‌تان علاوه بر موارد مطرح شده API ها و قسمت‌هایی را اضافه انجام دهید البته توجه کنید که این موارد نباید جایگزین موارد اصلی مطرح شده شوند و صرفا می‌تواند به عنوان مکمل آن‌ها به کار برود. در ادامه درباره نحوه نمره‌دهی به این موارد هم توضیحاتی خواهیم داشت.

تعریف کلی این فاز

برنامه‌های پیام‌رسان تحت وب شامل تعدادی عناصر کلیدی جهت ارتباط کاربران با یکدیگر هستند که برخی موجودیت[2]های منطقی آن در فازهای قبل تعریف و اجرا شدند. حالا با توجه به این موجودیت‌ها و نمونه‌های پیام‌رسان‌های تحت‌وب همانند تلگرام و واتسپ وب قرار است مواردی که در ادامه به تفصیل خواهیم گفت را پیاده‌سازی کنیم مانند قرارگیری عناصر برنامه شامل لیست چت‌ها، لیست مخاطبین و سایر موارد.
همینطور در ادامه درباره نحوه کارکرد پیام‌رسان و ارسال و دریافت پیام‌ها در فرانت‌اند و کار کردن با وب‌سوکت در آن توضیحاتی خواهیم داد.

ابزارهای پیاده‌سازی

برای پیاده‌سازی فرانت شما به دو صورت می‌توانید عمل کنید:

1. **پیاده‌سازی با HTML, CSS, Vanilla JS:** که به این معنی است که شما اجازه استفاده از Framework ها و کتابخانه‌های خارجی جاوااسکریپت برای مدیریت Dom مانند jQuery را در این حالت ندارید و صرفا با جاوااسکریپت خام باید پروژه را پیاده کنید. البته استفاده از

[1] Functionality
[2] Entity

کتابخانه‌هایی که برای مدیریت DOM[3] نیستند و نقش کمک‌کننده و تسریع‌بخش روند توسعه را دارند و ساختار و کانفیگ کلی برنامه را تغییر نمی‌دهند مانند axios, moment.js و ... مشکلی ندارد.
دانستن استانداردها و قابلیت‌های مطرح‌شده در نسخه‌های به‌روزتر ECMAScript به شما در پیاده‌سازی قابلیت‌ها و انجام پروژه کمک خواهد کرد.
همچنین استفاده از کتابخانه‌ها و فریمورک‌های CSS مانند Bootstrap, Tailwind مجاز نیست.
برای کسب اطلاعات بیشتر می‌توانید به W3Schools, MDN مراجعه کنید.

2. **پیاده‌سازی با React.js (پیشنهادی):** در این حالت شما یک اپ React ایجاد می‌کنید که در آن تگ‌های HTML را در قالب کد JSX در کتابخانه React پیاده می‌کنید و همچنین منطق برنامه و کار با DOM را به وسیله هوک‌ها و ابزارهایی که React در اختیار شما می‌گذارد انجام می‌دهید.
برای استایل‌دهی کامپوننت‌ها و المان‌های ایجاد شده نیز از CSS استفاده می‌کنید که ترجیح بر استفاده از CSS Module ها برای کامپوننت‌ها و یک سری هم استایل‌دهی‌های گلوبال است.
استفاده از React به دلیل تسهیل کار با DOM به وسیله Virtual DOM، معماری Component-Based و در نتیجه ساختار منظم و قابل استفاده مجدد و همینطور مدیریت وضعیت[4] برنامه با روش بهتر به شما پیشنهاد می‌شود.
برای ساخت یک برنامه React هرچند که طبق گفته داکیومنت رسمی در مدتی قبل استفاده از فریمورک‌هایی مثل Next.js و یا build tool هایی مثل Vite پیشنهاد شده است اما جهت یکپارچگی، در این پروژه باید صرفا از طریق create-react-app ساخته شود. در نظر داشته باشید برای کار با React، بر سیستم خود Node.js و npm را نصب داشته باشید.
نکته: شاید در ابتدا به علت عدم دانش درباره React قصد نداشته باشید با این کتابخانه پروژه را پیاده کنید، اما در نظر داشته باشید که اگر در مدت زمان مناسبی به مقدار موردنیاز آن را یاد بگیرید پیاده‌سازی پروژه برای شما بسیار راحت‌تر خواهد شد.
برای مشاهده داکیومنت‌ها و اطلاعات بیشتر درباره React می‌توانید به این لینک مراجعه کنید.

---
[3] Document Object Model
[4] State Management

دستور کار

پیام‌رسان ما شامل بخش‌هایی است که توضیح آن‌ها را از قبیل کارکرد و محل قرارگیری آن‌ها در ادامه خواهیم دید.

نکته: توجه داشته باشید که مسائلی از قبیل محل قرارگیری برخی عناصر مثل برخی دکمه‌ها و منوها که در ادامه توضیح داده می‌شود پیشنهادی است و شما می‌توانید به طور مناسب آن‌ها را تغییر دهید، البته به نحوی که به کارکردها، صحت برنامه و تجربه کاربری[5] خللی وارد نشود و به طور کلی شما باید ساختار یک پیام‌رسان وب را داشته باشید که قابلیت‌های لازم را داشته باشد.

ایجاد حساب کاربری

در این بخش می‌خواهیم برای کاربر پیام‌رسان تحت وب خودمان حساب کاربری ایجاد کنیم. برای این کار نیاز به دو صفحه داریم. اول یک صفحه Sign Up است که باید فیلدهای زیر را داشته باشد:

- firstname: string
- lastname: string
- phone: string
- username: string
- password: string
- image
- bio: string

به صلاحدید خود می‌توانید تکمیل این اطلاعات را در چند مرحله و صفحه‌های پشت سر هم یا Stepper ها انجام دهید.

برخی فیلدها باید چک شود که به‌ازای هر کاربر یکتا باشد. این فیلدها شامل شماره تلفن همراه، username خواهد بود. می‌توانید برای این کار در صورتی که کاربر از ویژگی یکتا بودن این سه مورد خارج شد پیغام مناسب را برایش چاپ کنید و فیلدها را قرمز کنید تا توجه کاربر را به خود جلب کند.

سپس با زدن ثبت‌نام و در صورت موفقیت آمیز بودن درخواست API، کاربر توکن JWT خود را دریافت کرده و وارد برنامه می‌شود. توضیحات بیشتر درباره این توکن را در بخش بعدی یعنی لاگین می‌بینیم.

[5] User Experience (UX)

لاگین

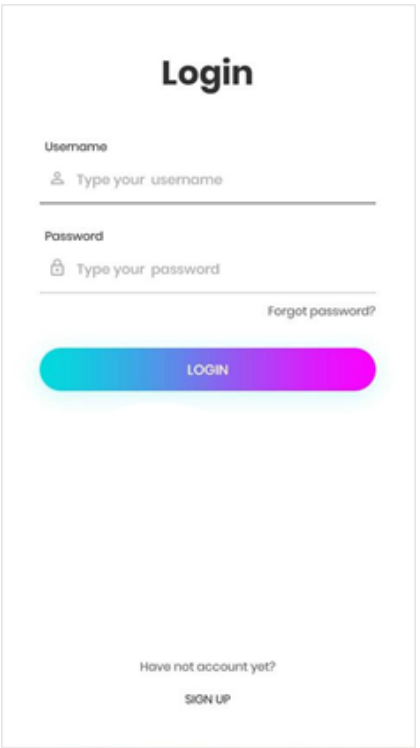
برای بخش لاگین باید از نام کاربری و رمز عبوری که معتبر است استفاده کنید. در صورتی که نام کاربری و پسورد درست نبودند باید به نحو مناسبی به کاربر خطا دهید.

با کلیک روی دکمه لاگین یک ریکوئست برای احراز هویت به بکند ارسال میشود. در بکند این اطلاعات چک شده و اگر درست بود JWT تولید شده و به سمت کلاینت ارسال می شود. بسته به روش مورد استفاده شما (مثلا کوکی) این JWT در استیت کلی فرانت‌اند (جهت استفاده از آن در تمام ریکوئست‌های برنامه که به احراز هویت نیاز دارند) و همچنین مرورگر کاربر (جهت از بین نرفتن با ترک نشست) باید قابل دسترسی باشد و برای درخواست های بعدی به بکند استفاده خواهد شد. به این صورت که در هدر Authorization این توکن قرار داده خواهد شد. در صورت منقضی شدن توکن یا هنگام خوردن خطای unauthorized در هر ریکوئست، کاربر باید به صفحه لاگین ریدایرکت شود تا با لاگین کردن توکن جدید خود را دریافت کند.

صفحه ورود باید شامل موارد زیر باشد:

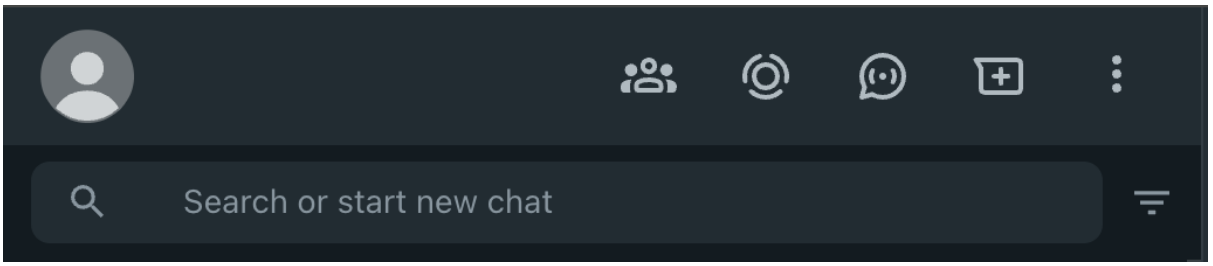
- فیلدی برای وارد کردن نام کاربری
- فیلدی برای وارد کردن رمز عبوری
- دکمه سابمیت فرم لاگین
- دکمه ای برای رفتن به صفحه sign up

نمونه ای از آنچه مد نظر است در تصویر آمده است؛ رنگ‌ها و شِما میتوانند به انتخاب شما تغییر کنند.



مشاهده پروفایل

در پیام‌رسان‌هایی که شما روزانه از آن‌ها استفاده می‌کنید مانند تلگرام و واتسپ شما یک قسمت در گوشه صفحه خود می‌بینید که عکس پروفایل یا آیکن آن قرار دارد (مانند شکل زیر). شما باید این آیکن را قرار دهید و کاربر باید بتواند با کلیک بر روی آن وارد صفحه پروفایل خود شود.



در این صفحه شما می‌توانید اطلاعات صفحه کاربری خود را مشاهده کنید و تغییر دهید. (فیلدها همان فیلدهای صفحه ایجاد حساب کاربری هست). می‌توانید در پایین صفحه هم دکمه‌ای را قرار دهید که در صورت تغییر اطلاعات شخصی خود، نمایان شده و بتوانید تغییرات را ذخیره کنید.

دو دکمه Log out, Delete account هم باید یا در این صفحه قرار بگیرند برای مثال در انتهای صفحه یا در منویی مجزا مثل منوی سه‌نقطه در تصویر بالا. توجه کنید در هنگام logout باید توکن و اطلاعات کاربر از استیت برنامه و مرورگر حذف شده باشد.

لیست چت

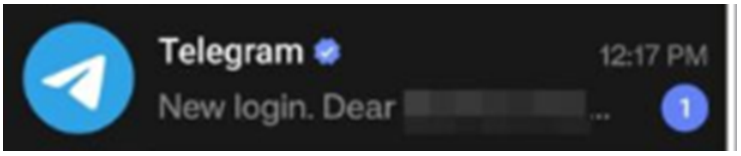
همان‌طور که در پیام‌رسانی مثل تلگرام وب مشاهده کرده‌اید، بخشی از صفحه به نمایش لیست چت‌ها اختصاص داده می‌شود و بخش دیگر صفحه نیز برای نمایش صفحه چت در نظر گرفته شده است. در این بخش می‌خواهیم ستون مربوط به لیست چت را پیاده‌سازی کنیم.

در قسمت بالای این ستون، یک Search Bar وجود دارد که امکان جستجو در میان نام‌ها و پیام‌های مخاطبان و گروه‌ها را برای کاربر فراهم می‌کند. در نتیجه‌ی جستجوی کاربر، باید لیست چت فیلتر شود و نتایج جستجو به کاربر نمایش داده شود.

در بخش زیرین این Search Bar، لیست چت کاربر وجود دارد. این لیست، یک لیست قابل پیمایش است. گروه‌ها و چت‌های مختلف با مخاطبان در این لیست نمایش داده می‌شوند. هر یک از آیتم‌های این لیست، باید شامل موارد زیر باشد:

- تصویر: در هر آیتم از لیست چت، باید آواتار مخاطب و یا تصویر تنظیم شده برای گروه مربوطه نمایش داده شود.

- **نام**: در هر آیتم از لیست چت، باید نام مخاطب و یا نام گروه مربوطه نمایش داده شود.
- **زمان**: در هر آیتم از لیست چت، باید زمان آخرین پیام نمایش داده شود. توجه کنید آخرین پیام ممکن است پیام دریافت شده از مخاطب باشد و یا پیام ارسال شده توسط خود کاربر. در صورتی که در یک گروه پیامی ارسال نشده باشد، باید زمان تشکیل گروه در این قسمت نمایش داده شود.
- **محتوای آخرین پیام**: در هر آیتم از لیست چت، محتوای آخرین پیام باید با توجه به طول پیام نمایش داده شود؛ در صورتی که پیام کوتاه است، همه‌ی پیام و در صورتی که پیام طولانی است، بخش ابتدایی پیام باید قابل مشاهده باشد.
- **وضعیت کاربر**: در هر آیتم از لیست چت، وضعیت آنلاین، آفلاین و یا در حال تایپ بودن کاربر باید قابل مشاهده باشد. برای مثال می‌توانید یک دایره‌ی کوچک رنگی در نظر بگیرید و رنگ آن را با توجه به وضعیت کاربر تغییر دهید. هر روش دیگری که وضعیت کاربر را به درستی نشان دهد نیز قابل قبول است.
- **تعداد پیام‌های خوانده نشده**: در صورتی که پیام‌هایی وجود دارند که هنوز توسط کاربر مشاهده نشده‌اند، باید تعداد این پیام‌های خوانده نشده نیز در این بخش نمایش داده شود.



نکات: وقتی یک چت از لیست چت انتخاب شده و فعال است، باید به طریقی قابل تشخیص بوده و متمایز از بقیه آیتم‌های لیست باشد. برای مثال می‌توانید رنگ پس‌زمینه آیتم را تغییر دهید. همچنین توجه داشته باشید لیست چت باید به صورت real-time آپدیت شود و همواره آخرین و به روزترین اطلاعات را نمایش دهد.

صفحه چت

پس از انتخاب یک چت از لیست چت، باید محتوای آن چت یا گروه در قسمت دیگر صفحه نمایش داده شود. چت انتخاب شده می‌تواند چت با مخاطب و یا گروه باشد که هر کدام در ادامه توضیح داده خواهد شد.

چت با مخاطب

در صفحه چت با مخاطب، در قسمت بالای صفحه چت باید اطلاعات مربوط به مخاطب نمایش داده شود. این اطلاعات شامل تصویر مخاطب، نام مخاطب و وضعیت مخاطب (آنلاین، آفلاین، در حال تایپ) است.

در قسمت وسط صفحه چت، باید پیام‌های ارسال و دریافت‌شده نمایش داده شوند. پیام‌های ارسال‌شده توسط کاربر در یک سمت و با یک رنگ خاص، و پیام‌های ارسال‌شده توسط مخاطب در سمت دیگر و با رنگ متفاوت نمایش داده می‌شوند.

در قسمت پایین صفحه چت نیز یک Text Input وجود دارد که برای تایپ پیام استفاده می‌شود. در این قسمت یک دکمه برای ارسال پیام در نظر بگیرید. همچنین برای ارسال فایل نیز یک دکمه در این بخش وجود دارد.

گروه

در صفحه چت مربوط به گروه، در قسمت بالای صفحه چت باید اطلاعات مربوط به گروه نمایش داده شود. این اطلاعات شامل تصویر گروه، نام گروه و تعداد اعضای گروه است.

در قسمت وسط صفحه چت، باید پیام‌های ارسال و دریافت‌شده نمایش داده شوند. پیام‌های ارسال‌شده توسط کاربر در یک سمت و باقی پیام‌ها در سمت دیگر و با رنگ متفاوت نمایش داده می‌شوند. همچنین در بالای هر پیام، نام ارسال‌کننده پیام و در کنار پیام، آواتار ارسال کننده پیام قرار دارد. (به جز پیام‌های خود کاربر)

در قسمت پایین صفحه چت نیز Text Input وجود دارد که برای تایپ پیام استفاده می‌شود. در این قسمت یک دکمه برای ارسال پیام در نظر بگیرید. همچنین برای ارسال فایل نیز یک دکمه در این بخش وجود دارد.

هم در حالت چت با مخاطب و هم گروه، انتظار می‌رود با کلیک راست روی هر پیام، یک گزینه برای حذف پیام نمایش داده شود.

مخاطبین و گروه

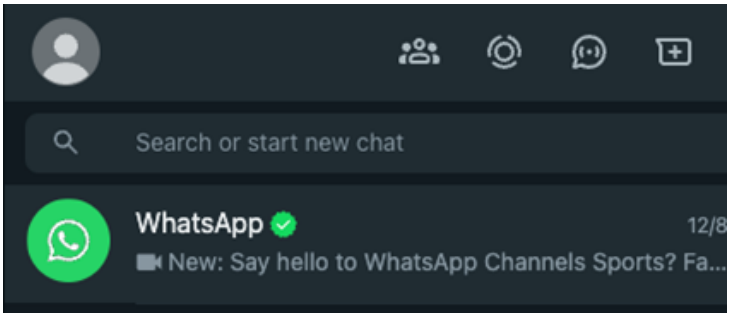
کاربر باید قابلیت اضافه کردن مخاطب یا گروه را در این وب اپلیکیشن داشته باشد. در بالای صفحه سه دکمه وجود دارد.

دکمه افزودن مخاطب: با کلیک بر این دکمه، به کاربر مودالی نشان داده می شود که در این مودال، کاربر میتواند با وارد کردن نام کاربری و بررسی وجود کاربر خود در سیستم، نام او آن را به لیست مخاطبان اضافه کرد.

دکمه افزودن گروه: با کلیک روی این دکمه مجددا مودالی باز می‌شود. در اینجا میتواند اسم و عکسی برای گروه انتخاب کرده و با انتخاب یک یا چند مخاطب گروه را بسازد.

دکمه مشاهده لیست مخاطبان: با کلیک روی این دکمه، کاربر بتواند لیست مخاطبان خود را در ببیند. (لیست مخاطبان باید جایگزین لیست چت‌های فعلی در صفحه شود). همچنین در این صفحه قابلیت های زیر باید وجود داشته باشد:

- قابلیت جستجوی کاربر با کلیدواژه مربوطه با استفاده از فیلدی در بالای لیست مخاطبان
- با کلیک راست بر هر مخاطب در لیست مخاطبین گزینه ای برای حذف آن داریم
- هنگام کلیک بر مخاطب یا گروه مربوطه صفحه چت با مخاطب یا گروه به کاربر نشان داده شود



صفحه اطلاعات کاربر و گروه

صفحه اطلاعات کاربر

طراحی و چیدمان: صفحه با طراحی کاربرپسند و مینیمالیستی، اطلاعات کاربر را در قالبی واضح و آسان برای خواندن نمایش می‌دهد.

اطلاعات کاربری:

- نمایش تصویر پروفایل و نام کاربری: در بالای صفحه، تصویر بزرگ پروفایل و نام کاربری به صورت برجسته نمایش داده می‌شود.
- نمایش وضعیت آنلاین/آفلاین و آخرین بازدید: این اطلاعات در کنار تصویر پروفایل قرار دارند.

گزینه‌های ارتباطی و مدیریتی:

- افزودن به مخاطبین/دنبال کردن: یک دکمه برای افزودن کاربر به لیست مخاطبین یا دنبال کردن کاربر
- گزینه‌های حذف و مدیریت: امکان حذف چت و یا حذف کاربر از لیست مخاطبین.

صفحه اطلاعات گروه

نمایش اعضا:

- لیست اعضا: نمایش لیستی از اعضا با تصاویر کوچک و نام کاربری، به همراه وضعیت آنلاین/آفلاین.
- گزینه‌های مدیریتی برای هر عضو: با کلیک راست بر روی هر عضو، منویی با گزینه‌های مختلف مانند حذف عضو از گروه نمایش داده می‌شود. توجه داشته باشید این کار را فقط ادمین بتواند انجام دهد.
- گزینه حذف گروه: ادمین بتواند گروه را حذف کند.
- گزینه خروج از گروه: کاربر گروه بتواند از گروه خارج شود (به وسیله api حذف کردن کاربر از گروه همانند حذف کردن دیگر افراد توسط ادمین)

اضافه کردن اعضا به گروه:

- دکمه اضافه کردن: این دکمه امکان جستجو و انتخاب اعضای جدید را (به وسیله یک مودال یا قرارگیری در قسمتی از صفحه) فراهم می‌کند.
- جستجو: کاربر می‌تواند با استفاده از جستجو در قسمت باز شده، اعضای جدید را بیابد و به گروه اضافه کند.

پیاده‌سازی وب‌سوکت

همانند پیاده‌سازی بکند، برای برقراری یک ارتباط real-time نیاز است تا به جای درخواست‌های HTTP از WebSocket استفاده کنید. برای این منظور بسته به این‌که پیاده‌سازی را با React یا Vanilla JS انجام می‌دهید می‌توانید به صورت‌های زیر عمل کنید:

۱. پیاده‌سازی با Vanilla JS: در این حالت شما از کلاس WebSocket در Javascript آبجکتی می‌سازید و از آن برای فراخوانی توابع سوکت خود استفاده می‌کنید. در تصویر پایین مثال ساده‌ای از استفاده از آن را با هم می‌بینیم. برای مثال هنگام دریافت هر پیام از سمت سرور تابع onmessage فراخوانی می‌شود. برای اطلاعات بیشتر درباره WebSocket می‌توانید به این لینک‌ها مراجعه کنید: javascript info و MDN

```
let socket = new WebSocket("wss://javascript.info/article/websocket/demo/hello");

socket.onopen = function(e) {
  alert("[open] Connection established");
  alert("Sending to server");
  socket.send("My name is John");
};

socket.onmessage = function(event) {
  alert(`[message] Data received from server: ${event.data}`);
};

socket.onclose = function(event) {
  if (event.wasClean) {
    alert(`[close] Connection closed cleanly, code=${event.code} reason=${event.reason}`);
  } else {
    // e.g. server process killed or network down
    // event.code is usually 1006 in this case
    alert('[close] Connection died');
  }
};

socket.onerror = function(error) {
  alert(`[error]`);
};
```

۲. پیاده‌سازی با React:

در این صورت می‌توانید هم از روش بالا و هم روش مخصوص برای خود ری‌اکت به وسیله کتابخانه‌های آن استفاده کنید. برای استفاده از روش بالا و استفاده از کلاس WebSocket درون ری‌اکت برای مثال می‌توانید به صورت زیر عمل کنید:

```
export const Home = () => {
  const connection = useRef(null)

  useEffect(() => {
    const socket = new WebSocket("ws://127.0.0.1:800")

    // Connection opened
    socket.addEventListener("open", (event) => {
      socket.send("Connection established")
    })

    // Listen for messages
    socket.addEventListener("message", (event) => {
      console.log("Message from server ", event.data)
    })

    connection.current = ws

    return () => connection.close()
  }, [])

  return <Chat />
}
```

اما پیشنهاد این است تا از کتابخانه‌هایی نظیر useWebSocket استفاده کنید تا چرا که امکاناتی متناسب با React به شما ارائه می‌دهند که توضیحات مربوطه را در داکیومنت داخل لینک فوق می‌توانید مشاهده کنید.

در تصویر پایین که مثالی ساده را از استفاده از این کتابخانه مشاهده خواهید کرد به وسیله پراپرتی readyState و استفاده از هوک useEffect (جهت مدیریت side effect ها و تشخیص تغییرات یک state از این هوک استفاده می‌کنیم) تغییرات وصل شدن سوکت و با lastJsonMessage هم دریافت پیام‌های جدید را متوجه می‌شویم. از مزایای این کتابخانه امکان استفاده از یک سوکت مشترک در چند کامپوننت بدون نیاز به اتصال مجدد و یا پاک‌سازی آن است که با آپشن share قابل تنظیم است، همینطور دیگر آپشن‌ها که کار با سوکت را در سمت کلاینت ساده‌تر می‌کند که به تفصیل در مستندات آن قابل مشاهده است.

```jsx
import useWebSocket, { ReadyState } from "react-use-websocket";

export const Home = () => {
  const WS_URL = "ws://127.0.0.1:800";
  const { sendJsonMessage, lastJsonMessage, readyState } = useWebSocket(
    WS_URL,
    {
      share: false,
      shouldReconnect: () => true,
    }
  );

  // Run when the connection state (readyState) changes
  useEffect(() => {
    console.log("Connection state changed");
    if (readyState === ReadyState.OPEN) {
      sendJsonMessage({
        event: "subscribe",
        data: {
          channel: "general-chatroom",
        },
      });
    }
  }, [readyState]);

  // Run when a new WebSocket message is received (lastJsonMessage)
  useEffect(() => {
    console.log(`Got a new message: ${lastJsonMessage}`);
  }, [lastJsonMessage]);

  return <Chat lastJsonMessage={lastJsonMessage} />;
};

```

نکات کلی پیاده‌سازی

- یکپارچگی سیستم رعایت شود به این معنی که از المان‌های نامشابه برای کارهای مشابه استفاده نشود و سعی شود از تکرار کد خودداری شود و به جای آن از کامپوننت‌های سیستم در صورت نیاز استفاده مجدد شود.
- هنگام رخداد خطا ادامه کارکرد سیستم نباید مختل شود در نتیجه از روش‌های مناسب برای مدیریت خطا[6] استفاده کنید. هنگام رخداد هرگونه خطا شامل خطاهای مربوط به API یا سیستمی به نحو مناسب به کاربر اطلاع داده شود. برای این منظور می‌توان بسته به موقعیت از Toast، helper text و ... استفاده کرد.
- تمام محدودیت‌های مطرح شده در فازهای قبل برای هر بخش از سیستم مثل حداقل تعداد کاراکتر در رمز عبور و ... لازم است در سمت فرانت هم درنظر گرفته شوند و به طور مناسب مدیریت شوند.
- صحت‌سنجی فیلدهای هر ورودی مثل شماره تلفن به طور مناسبی انجام شود.
- برای فیلدهای که کاربر باید در تعامل با آن‌ها بازخورد دریافت کند مثلا هنگام هاور روی دکمه‌ها تغییرات مناسبی اعمال شود.

ارزیابی

موارد مطرح شده در قسمت‌های قبل هرکدام شامل بخشی از نمره اصلی شما هستند. موارد امتیازی نیز شامل موارد زیر است:

- پیاده‌سازی گروه و قابلیت‌های مربوط به آن
- امکان ارسال فایل در چت‌ها
- ارسال نوتیفیکیشن در مرورگر کاربر در صورت دریافت پیام جدید
- استفاده از افکت‌ها و انیمیشن‌های جذاب و مناسب

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.

---

[6] Error Handling

- 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.