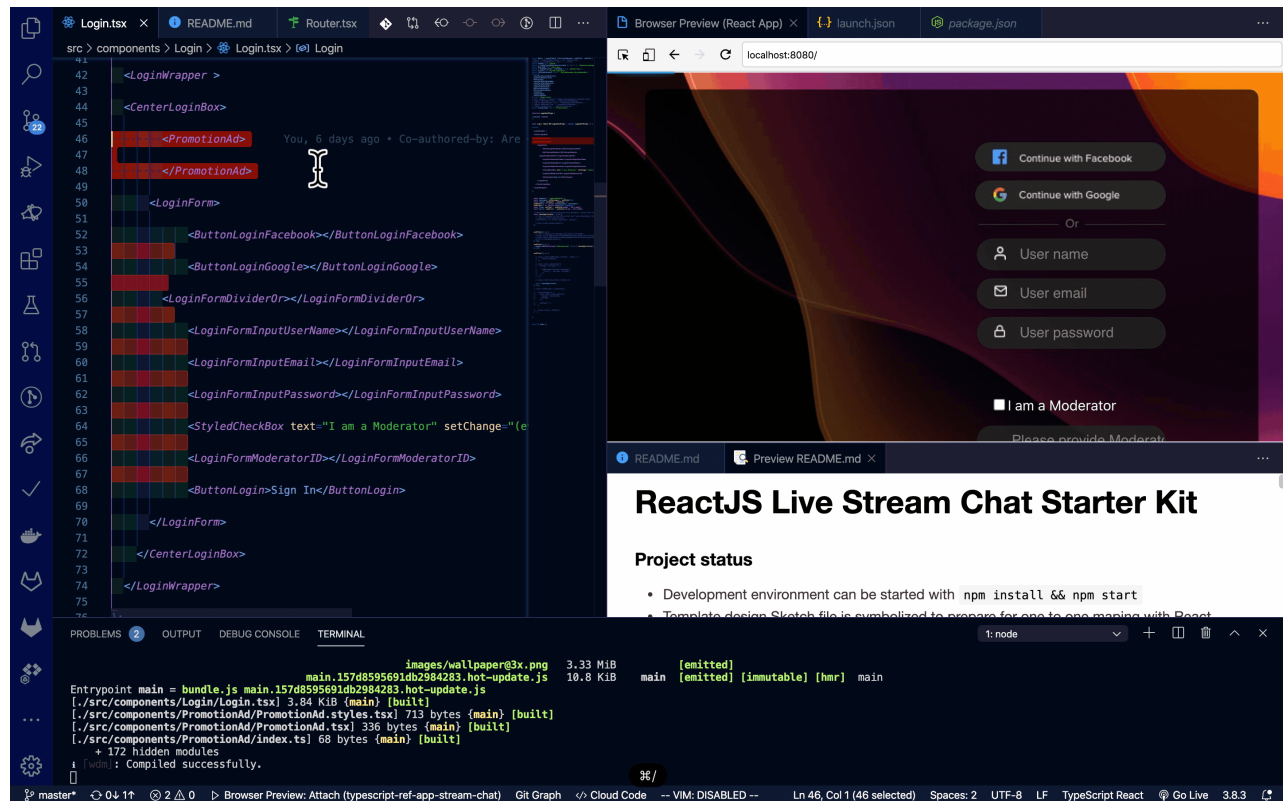


ReactJS Live Stream Chat Starter Kit



Project status

- Development environment can be started with `npm install && npm start`
- Template design Sketch file is symbolized to prepare for one to one mapping with React components.
- Login page is "pixel perfect" beside color and luminosity to be fixed.
- Event page is now more advanced but I had to modify the Sketch file and split the EventInfo Sketch component into two while keeping the first one to comply with Styled Components requirements, but then found a way to make it transparent to events (click events, not only the event being displayed per say).
- Event Page is now accessible upon app run. Login Screen can be bypassed (search for Router.tsx below for more information)

Background

React JS is now the most used framework used by Internet companies to develop the web apps they generate most revenue from.

React offers an easy way to group UI elements into Components so a Chat application can turn from a 100 UI objects, or more, to a 5 UI objects application, and experience less than 250ms delivery time from anywhere to anywhere in the world.

Once a developer uses React Components it becomes a matter of hours before a full fledged mission critical chat solution is offered to anyone in the world who wants to connect.

By leveraging open source tools and components it is possible to create a world class, mission critical, and maintainable chat service, surfacing over the web and mobile devices, while relying on an infrastructure provider to deliver messages anywhere on the Internet.

A reliable Chat back-end provider offers message deliverability, presence indicators and other related data in a predictable fashion so the React based front end can not only be suited to users taste and current habits but, no puns intended, also react to network messages without which there is no UI that would serve any user.

What to do with this repository?

This is an example of how to integrate open source components, including React chat components, with one notable message and presence service provider: PubNub. PubNub is offering a solid Java Script SDK to drive the React components along with the business specific logic.

This repository is based, not only on a reliable chat backend provider, already processing billions of messages, but also it is based only on open source tools with significant following on github so there is a better chance to see them maintained in the future.

You should only have to bring the design if you want to change it and the repository does provide some starter ones.

The repository includes the javascript libraries that allow for easier access to the chat service provider without the need to use a lower level interface such as REST.

It also includes standard React components that can be dressed up through standard css files that can be generated by any designer tool suite, including Sketch and Adobe XD. If also using some intelligent tools available to designers, such as Sketch2React, it is also possible to generate the React Chat App, almost entirely, through the designers pushing to a github repository automatically their designs and updates.

The Sketch file in this repository contains reusable UI components that leverage the latest Sketch technology, including "Overrides". Through Sketch Overrides saves time because it allows to use only one Sketch visual component per visual state this component can display once live on the screen. Using Overrides one can simply override each component property, including graphics and see how the new graphics integrate in the live template inside Sketch. If you have professional designers available to you they will certainly appreciate the time it saves them to start from Sketch files including relevant Overrides properties set.

What does this repository contain?

- A sample Sketch design file that maps to the css delivered in the source code so your Chat app can look designer generated.
- React files, and React js libraries, in typescript, allowing to run a web server application for hosting a Chat app. It can run on your computer or served hosted anywhere on the Internet for public access (The goal of this repo is not based on best security practices so you may want to bring in your security expert before doing so).
- A list of third party libraries used in the app is available in the package.json file. The command « yarn install » ran from the terminal will update your code with fresh cloned files for the versions of these libraries indicated right after their names under the header: dependencies in the package.json file located at the top of the project directory.
- A Chat component library to display the chat UI and content.
- PubNub React library to connect people together through the Chat app this project generates.

In the project directory:

1. Install ReactJS if not done already.
2. Clone this repository in your work directory.
3. Try this Web App on your computer. Launch a terminal, cd in the project directory and type each command below, then press enter, the last command should run the server and launch your web browser and connect to the localhost and the right port automatically:

```
yarn install
```

```
yarn start
```

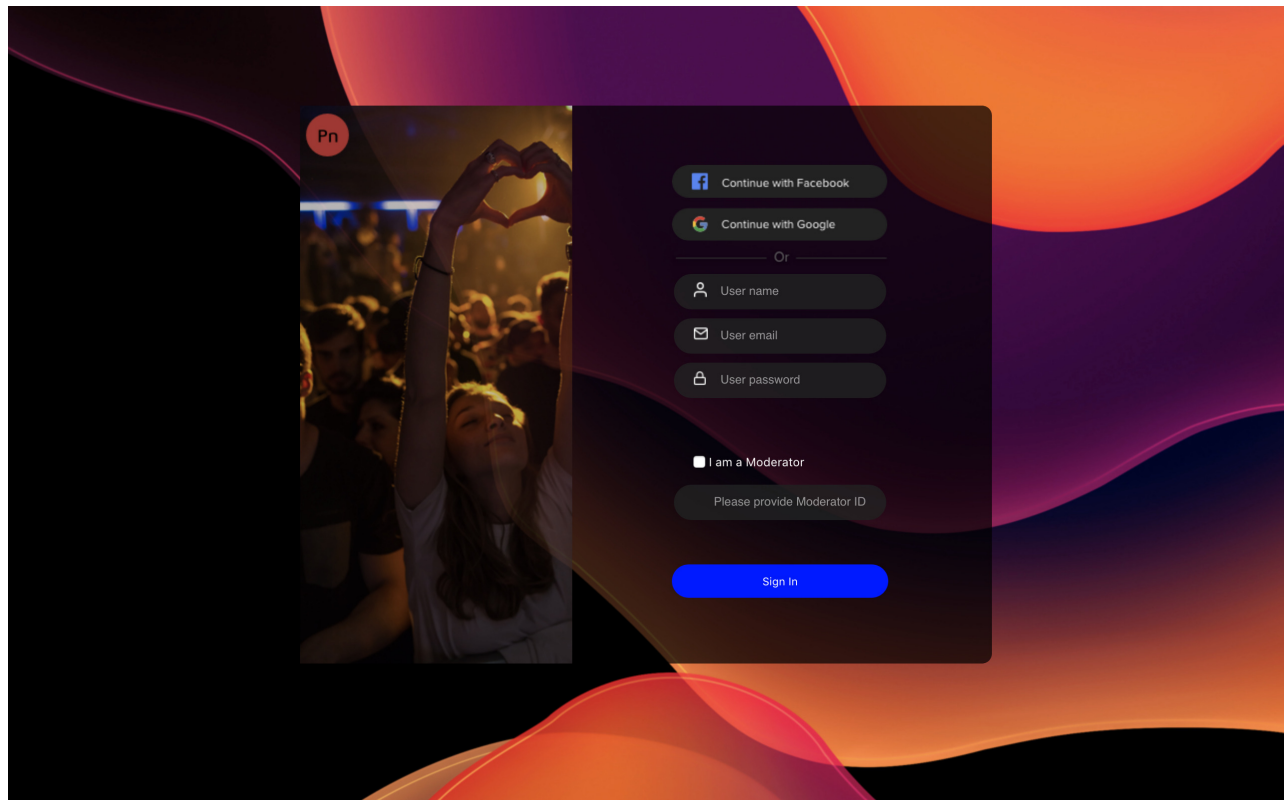
To learn more about the file structure and which file you have to modify so you can make your own Live Stream Event Chat App with your own design please read the Customization section included here.

If you want to bypass our beautiful login screen and access the Live Event selection process directly you can edit the Router.tsx file around line 36 and set the directToEvent variable to false (no need to restart since webpack reloads the chat server files automatically when detecting a change in one of them):

```
directToEvent = true;
```

Visuals

Login:



The code to customize the Login screen with your design can be found in `/src/components/PromotionAd/PromotionAd.styles.tsx`. To change the design all you need to do is edit the design file included inside the directory: The name of the file is `Login.style.tsx`. You can try it out by adding your own image file instead of the one on the login screen for example:

1. Drop a file into the `/src/img` directory and remember its name because you will need it in the next step.
2. Then simply change the name of the image file in the `PromotionAd.style.tsx` where

it shows: `background: url("images/promotionAd.png") no-repeat;`

Live Event

The Live Event screen itself is divided into multiple panels, each panel matches a component, which itself can contain more components in order to make the UI more modular when needed.

Live Event Detail (TBC)

```
1  return( <EventDetailWrapper>
2
3      <EventDetailsTopWrapper>
4          <div></img></div>
5          <TitleWrapper><span>eSport: Las Vegas Tournament 2020</spa
6      </EventDetailsTopWrapper>
7
8
9      <EventDetailsBottomWrapper>
10         <BottomShareButtonWrapper>
11             <div>
12                 </img>
13             </div>
14
15         </BottomShareButtonWrapper>
16
17         <BottomFollowButtonWrapper>
18             <div>
19                 </img>
20             </div>
21
22         </BottomFollowButtonWrapper>
23     </EventDetailsBottomWrapper>
24
25 </EventDetailWrapper>
26 );
```

LiveFeedPanel

This is the part of the screen where the event itself is to be displayed. This is a simple example using a YouTube stream that is simple passed inside the LiveFeedPanelWrapper tags. The

parameters to control the display of the YouTube stream source is passed using props. These props are passed from a higher level component in the component tree where you have defined them, so there should be no need to edit this file unless you want to add a streaming provider of your own or change the default provider.

```
1 | return(  
2 |   <LiveFeedPanelWrapper>  
3 |     <YouTube videoId={props.videoID} opts={props.opts} onReady={prop  
4 |   </LiveFeedPanelWrapper>  
5 | );
```

Event selection:

TO BE COMPLETED

Moderator:

TO BE COMPLETED

Using the Sketch file

PubNub Symbols

TO BE COMPLETED

Updating the Sketch file and generating new elements

TO BE COMPLETED

Sync css and graphics manually from included Sketch designs

TO BE COMPLETED

Continuous integration of updated included Sketch designs

TO BE COMPLETED

Get started with the code template

Using the PubNub Chat Components

Login

TO BE COMPLETED: SHOW VISUAL MAP

In: `src/components/Login.tsx`

```
1  ...
2
3  interface LoginInitProps {
4    simulate: true
5  }
6
7
8  const Login: React.SFC<LoginInitProps> = (props: LoginInitProps) => {
9
10   return(
11
12     <PubNubDesigner>
13
14     <CenterLoginBox>
15
16       <PromotionAd>
17
18       </PromotionAd>
19
20     <LoginForm>
21
22       <ButtonLoginFacebook></ButtonLoginFacebook>
23
24       <ButtonLoginGoogle></ButtonLoginGoogle>
25
26     <LoginFormDividerOr></LoginFormDividerOr>
27
28     <LoginFormInputUserName></LoginFormInputUserName>
29
30     <LoginFormInputEmail></LoginFormInputEmail>
31
32     <LoginFormInputPassword></LoginFormInputPassword>
33
```

```
34         <StyledCheckBox text="I am a Moderator"></StyledCheckBox>
35         <LoginFormModeratorID></LoginFormModeratorID>
36
37         <ButtonLogin>Sign In</ButtonLogin>
38
39     </LoginForm>
40
41 </CenterLoginBox>
42
43 </PubNubDesigner>
44 );
45 ...
```

Using the Sketch file

TO BE COMPLETED

Using the PubNub Styled Chat Components

TO BE COMPLETED

Learn More

TO BE COMPLETED

About React:

TO BE COMPLETED

About TypeScript:

TO BE COMPLETED

About PubNub React library for Java Script:

TO BE COMPLETED

About Styled Components: <https://styled-components.com/docs/basics#styling-any-component>

TO BE COMPLETED

About Sketch:

TO BE COMPLETED

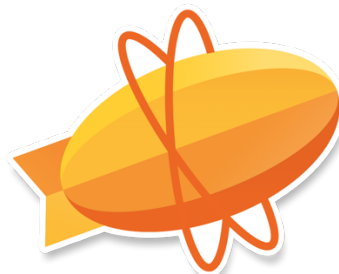
Third party tools

Zeplin

Design



Github



Zeplin

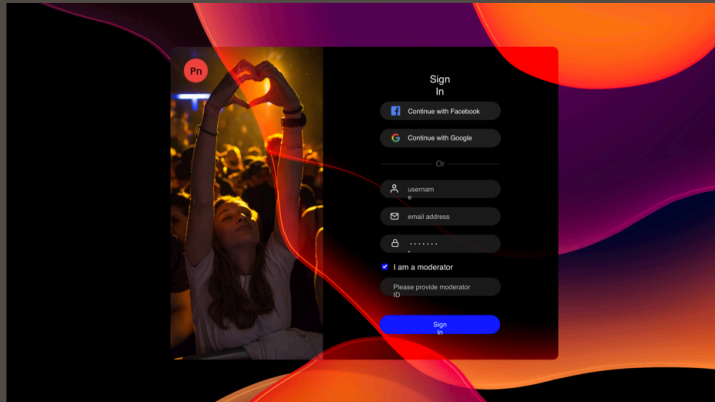


IDE



Zeplin


Browser




PubNub live event chat - login





companyLogo


 Continue with Facebook

 Continue with Google

Or







☒ I am a moderator

login



nfodorpbnub

2w

Description

Got any notes?

Share


Web: zpl.io/bzEPrl7

App: zpl://components?pid=5ec351...

Login

src/components/Login/Login.tsx

```
<Login
  simulate={boolean} />
```

 Open in GitHub

 Open in Default Editor



Size

Width: 217px

Height: 468px

File

Live Event Chat

Local > typescript-ref-app-stream-chat > src > sketch



Used in

