Global Distributed Software Development

Summer 2021

Milestone 4 - Project Team A

Under the guidance
Rainer Todtenhoefer, CTO and CEO

Team Members

Team Member Name	Role
Jagadesh Devan	Team Lead & Frontend Developer
Ahmad Yar Khan	Frontend Lead & GitHub Master
Taha Bin Aziz	Backend lead
Obada Altnawi	Frontend / Backend Developer

Revision History

Name	Date Submitted	Date Revised	Revision Summary	Version
Jagadesh Devan	27.07.2021	NA	NA	1.0

Contents

1. Product Summary

Motivation & Importance

The real estate market is one of the most complex markets in the entire world since it is in a continuous change, thus making it a very dynamic market. The internet has a lot to offer consumers regarding real estate and as a result it is a great place to start shopping. The buyer and seller have direct access to information about the property in question. This makes other forms of communication between the buyer and the seller obsolete. The internet is easy in comparison to the old-fashioned method of answering dozens of phones calls or setting up numerous meetings.

There is no better, safe, and easier way to search for a home or to sell one than online as the internet has a lot to offer in the real estate market and it is rapidly developing, gaining more and more consumers every day and thus improving your chances for a profitable buy/sell.

Target Audience:

The target of our applications is the customers who cannot spend more time for buying and selling the housing property.

Online real estate has become popular and is consuming are looking to the internet more each day as an easy place to get good information. As a matter of fact, more than 5 million people use the internet for real estate issues every month. With numbers like this it is easy to see how the internet can improve your chances for selling or buying a home.

Another major advantage of real estate moving to the internet is that you won't need a real estate agent to start your search. This is very important because we all know that real estate agents are of value but sometimes you just want to look.

Reasons to use our HomeForMe application:

Using our HomeForMe application for real estate will make you your own real estate agent without having to pay a great sum of money to an agent and you will have full control. If you are a home buyer or seller, it is very easy to search for the perfect house as the online offers are endless.

Unique Selling Point:

There are several applications available in the market for real estate. But our HomeForMe application will allow users to calculate an avg price of a property by adding in their preferences like area, rooms beforehand so it makes the user and provide specific results which will reduces the time for the users. Also, we have made our application user friendly so that you need is very basic computer usage skill.

Major Committed Functions

Actor	Requirement
New User	View Properties
New User	Searching
New User	Filter/Browse Properties
New User	Registration
Buyers	Messaging
Buyers	Dashboard
Buyers	Make a Contract
Agent	Mange Properties
Agent	Dashboard
Agent	Messaging
Admin	Approvals

2. Usability Test Plan

This document describes a test plan for conducting a usability test during the development of HomeForMe The goals of usability testing include establishing a baseline of user performance, establishing, and validating user performance measures, and identifying potential design concerns to be addressed to improve the efficiency, productivity, and end-user satisfaction.

The usability test objectives are:

- To determine design inconsistencies and usability problem areas within the user interface and content areas. Potential sources of error may include:
  - Navigation errors – failure to locate functions, excessive keystrokes to complete a function, failure to follow recommended screen flow.
  - Presentation errors – failure to locate and properly act upon desired information in screens, selection errors due to labeling ambiguities.
  - Control usage problems – improper toolbar or entry field usage.
- Exercise the application or web site under controlled test conditions with representative users. Data will be used to access whether usability goals regarding an effective, efficient, and well-received user interface have been achieved.
- Establish baseline user performance and user-satisfaction levels of the user interface for future usability evaluations.

Participants

The participants' responsibilities will be to attempt to complete a set of representative task scenarios presented to them in as efficient and timely a manner as possible, and to provide feedback regarding the usability and acceptability of the user interface. The participants will be directed to provide honest opinions regarding the usability of the application, and to participate in post-session subjective questionnaires and debriefing.

The intended users

The test will be conducted with targeted category of users, who are the students at university of Fulda. Therefore, we will ask three users of the university to test the website

Usability Testing Phase:

Test Participant: 1

Phase: 1 Screening & Pre-test

Q1. How old are you?
25
Q2. What is the highest level of education you've completed?
Bachelors Degree
Q3. What is your current occupation?
Student(Pursuing Masters)
Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for house online?
5
Q5. When was the last time you looked for buying or selling house online?
In 2018

Phase: 2 Usability Task description

Task 1: Imagine that you are in different country, and you want to view and rent a house in another country where you will be moving in couple of months.
Task 2: Imagine that you own a house, and you want to sell the property online.
Task 3: Imagine you are looking for a house in Frankfurt and you want to know the estimate rent of the house in Frankfurt.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching a property?
Good
Q2. How simple and clean was the interface?
It is very simple and clear with instructions.
Q3. Can you tell me what you think about filtering the property?
I tried multiple times and it worked without any issues.
Q4. How was your experience using the chat application?
Simple and clear.
Q5. How was your overall experience using the website?
It was very easy to use, and I faced an issue in price calculation when I could not find the estimation for some city.

Sr.No	Task Objective	Scale 0/5(0 – OK : 5 – GOOD)
TASK 1	Search for house and viewing house for rent.	5/5
TASK 2	Adding Property for selling	4/5
TASK 3	Estimating the price calculation for a house	3/5

Test Participant: 2

Phase: 1 Screening & Pre-test

Q1. How old are you?
29
Q2. What is the highest level of education you've completed?
Graduate Degree
Q3. What is your current occupation?
SAP Consultant
Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for house online?
5
Q5. When was the last time you looked for buying or selling house online?
April,2021

Phase: 2 Usability Task description

Task 1: Imagine that you are in different country, and you want to view and rent a house in another country where you will be moving in couple of months.
Task 2: Imagine that you own a house, and you want to sell the property online.
Task 3: Imagine you are looking for a house in Frankfurt and you want to know the estimate rent of the house in Frankfurt.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching a property?
Ideal web application to use.
Q2. How simple and clean was the interface?
Not confusing completely clear.
Q3. Can you tell me what you think about filtering the property?
With the available data I was able to fetch the property I wanted is displayed.
Q4. How was your experience using the chat application?
Just like any other messaging application.
Q5. How was your overall experience using the website?
Good application.

Sr.No	Task Objective	Scale 0/5
TASK 1	Search for house and viewing house for rent.	5/5
TASK 2	Adding Property for selling	3/5
TASK 3	Estimating the price calculation for a house	5/5

Test Participant: 3

Phase: 1 Screening & Pre-test

Q1. How old are you?
43
Q2. What is the highest level of education you've completed?
Diploma
Q3. What is your current occupation?
Electrician
Q4. On a scale of 1 to 5 how would you rate your level of confidence in using your laptop/pc for looking for house online?
3
Q5. When was the last time you looked for buying or selling house online?
N/A

Phase: 2 Usability Task description

Task 1: Imagine that you are in different country, and you want to view and rent a house in another country where you will be moving in couple of months.
Task 2: Imagine that you own a house, and you want to sell the property online.
Task 3: Imagine you are looking for a house in Frankfurt and you want to know the estimate rent of the house in Frankfurt.

Phase: 3 Post Test Questions

Q1. How was your overall experience when searching a property?
good
Q2. How simple and clean was the interface?
nice
Q3. Can you tell me what you think about filtering the property?
Easy to use the filters
Q4. How was your experience using the chat application?
Easy to use
Q5. How was your overall experience using the website?
Its good except adding property. It was confusing for me to add property. I had to perform the task twice to finish the scenario.

Sr.No	Task Objective	Scale 0/5
TASK 1	Search for house and viewing house for rent.	3/5
TASK 2	Adding Property for selling	2/5
TASK 3	Estimating the price calculation for a house	3/5

3. QA test plan

Test Objectives

The test objectives are to verify the Functionality of website HomeForMe, the project should focus on testing the search for product to guarantee this functionality can work normally in real business environment and to ensure that the bugs/issues are identified and fixed before going live

Feature to be tested

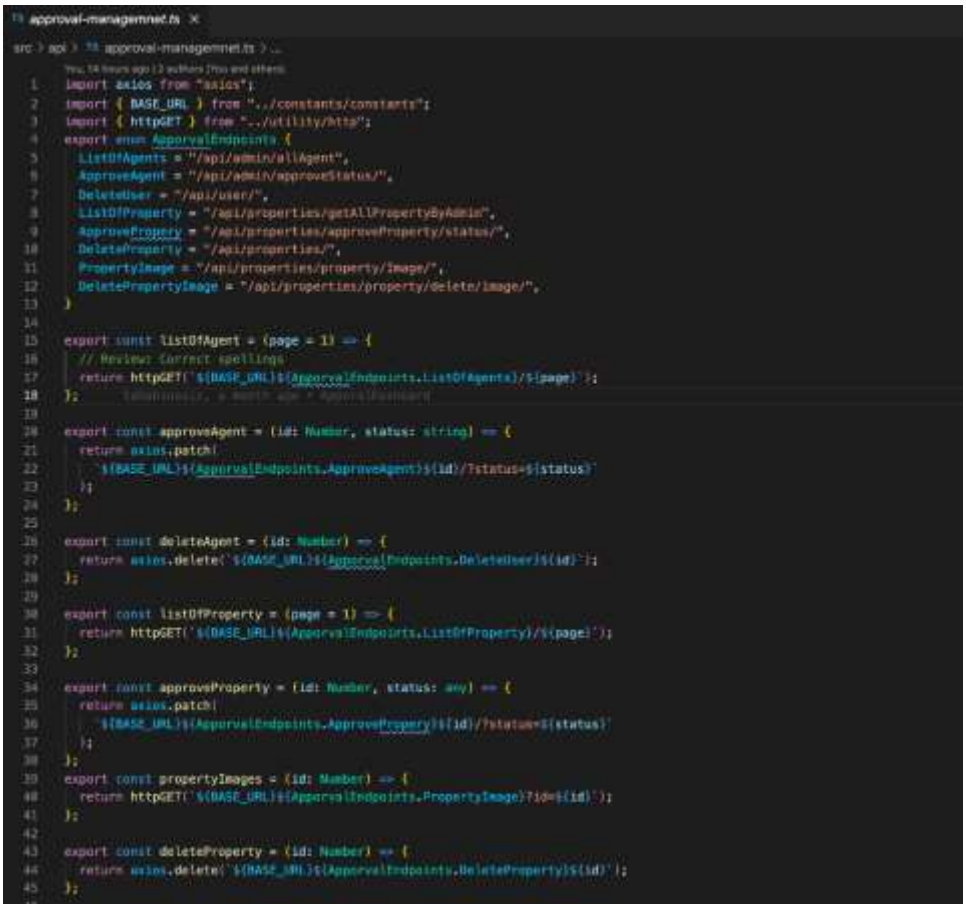
All the functionalities of the website should be tested and verified, but will concentrate on the "search for property", "Messaging", and "Filtering" features

Test cases:

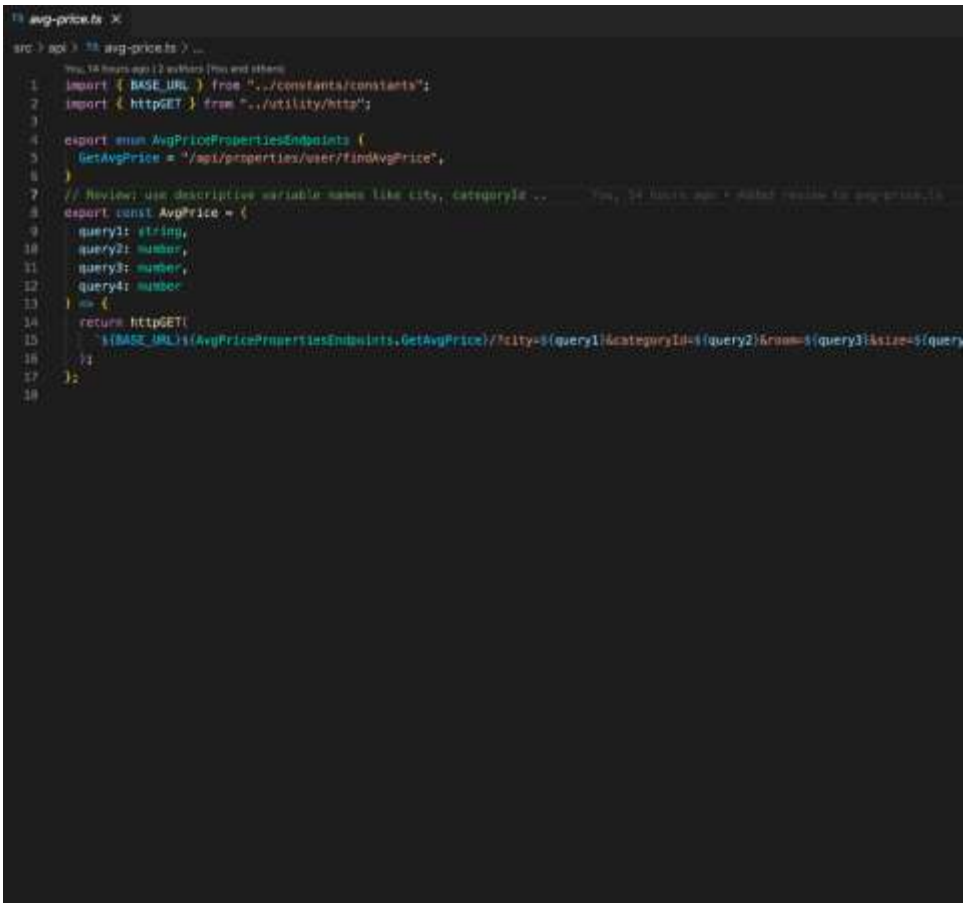
Test Case Id	Test Title	Test Description	Test Input	Expected Result	Test Results
TC_1	Viewing Property	As a new user view the property	Navigate to the view proper page as a new user and verify that new user is able to view the properties.	New should be able to view the properties.	Pass
TC_2	Filter Houses for Rent	As a customer filter the houses for rent.	Navigate to the property page, filter the category by rent.	Customer should be able to see only houses for rent.	Pass
TC_3	Messaging to the House Owner	As a customer contact the house owner.	Navigate to the property details page and click on Contact to message the house owner.	Customer should be able to contact the house owner via chat.	Pass

TC_4	View Proprty details.	As a customer view the property detials.	Navigate to property page and click on the property to view the property details.	Customer should be able to view the property details.	Pass
TC_5	Managing Properties	As a Agent mange the properties	Login as Agent and navigate to property details page and edit the properties or add new property.	Agent should be able to manage the properties and add new property.	Pass

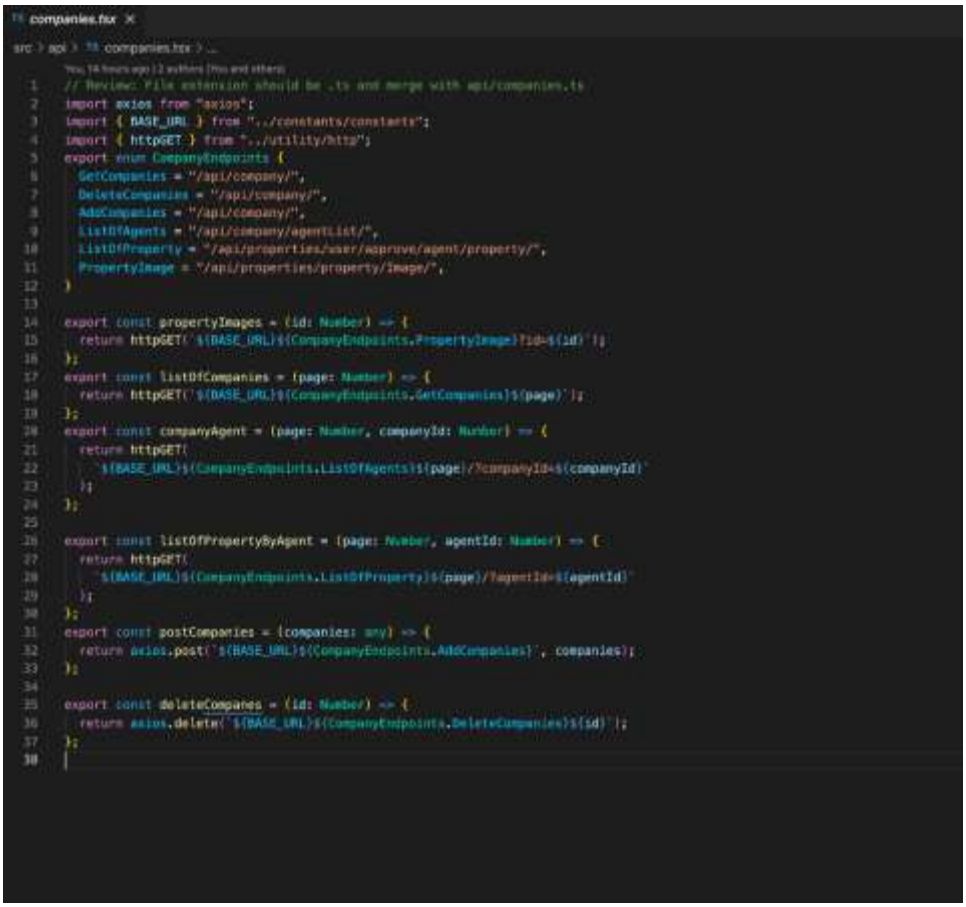
4. Code Review



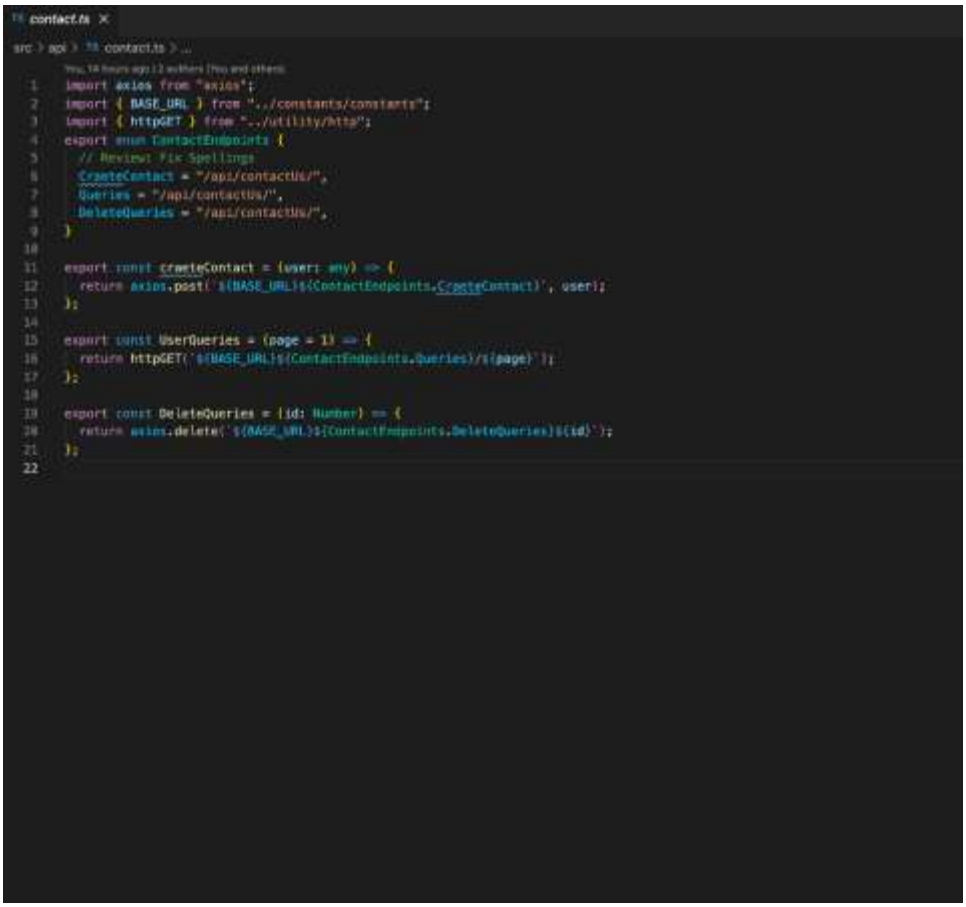
```
approval-managemnet.ts
src > api > approval-managemnet.ts > ...
import axios from "axios";
import { BASE_URL } from "../constants/constants";
import { httpGET } from "../utility/http";
export enum ApporvalEndpoints {
  ListOfAgents = "/api/admin/allAgent",
  ApproveAgent = "/api/admin/approveStatus/",
  DeleteUser = "/api/user/",
  ListOfProperty = "/api/properties/getAllPropertyByAdmin",
  ApproveProgery = "/api/properties/approveProperty/status/",
  DeleteProperty = "/api/properties/",
  PropertyImage = "/api/properties/property/Image/",
  DeletePropertyImage = "/api/properties/property/delete/image/",
}

export const listOfAgent = (page = 1) => {
  // Review Correct spellings
  return httpGET(`${BASE_URL}${ApporvalEndpoints.ListOfAgents}/${page}`);
};

export const approveAgent = (id: Number, status: string) => {
  return axios.patch(
    `${BASE_URL}${ApporvalEndpoints.ApproveAgent}${id}/?status=${status}`
  );
};

export const deleteAgent = (id: Number) => {
  return axios.delete(`${BASE_URL}${ApporvalEndpoints.DeleteUser}${id}`);
};

export const listOfProperty = (page = 1) => {
  return httpGET(`${BASE_URL}${ApporvalEndpoints.ListOfProperty}/${page}`);
};

export const approveProperty = (id: Number, status: any) => {
  return axios.patch(
    `${BASE_URL}${ApporvalEndpoints.ApproveProgery}${id}/?status=${status}`
  );
};
export const propertyImages = (id: Number) => {
  return httpGET(`${BASE_URL}${ApporvalEndpoints.PropertyImage}?id=${id}`);
};

export const deleteProperty = (id: Number) => {
  return axios.delete(`${BASE_URL}${ApporvalEndpoints.DeleteProperty}${id}`);
};
```



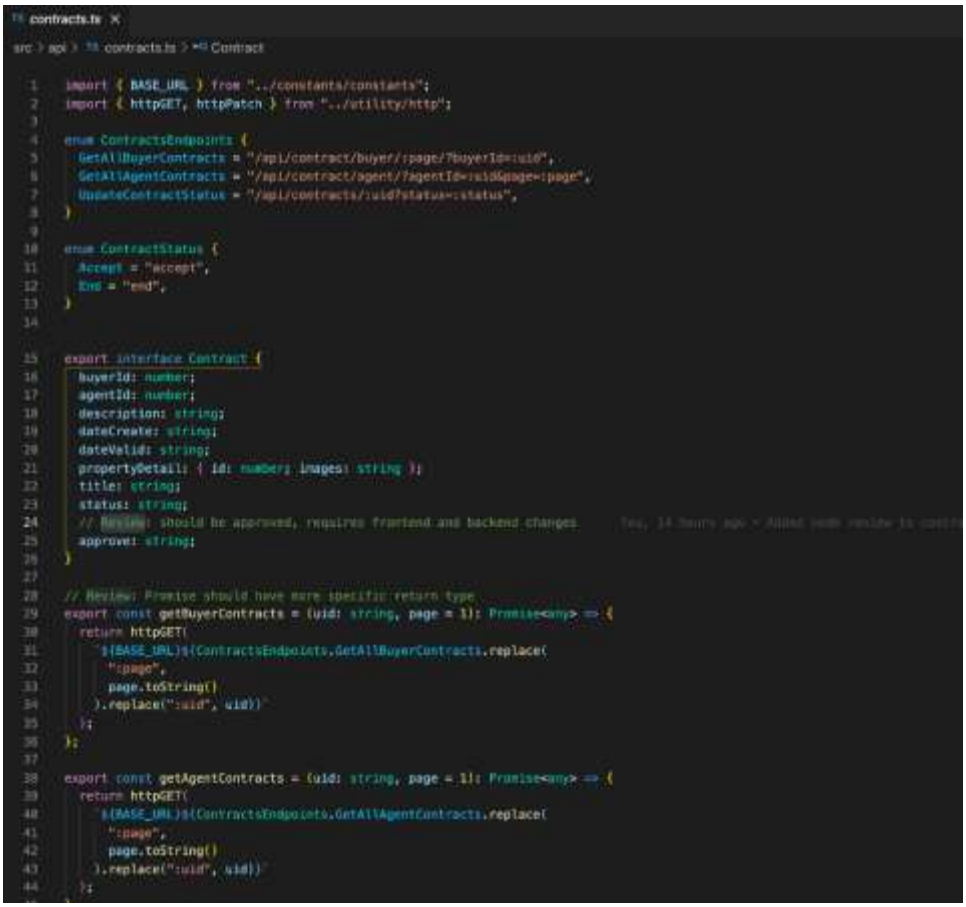
```
avg-price.ts
src > api > avg-price.ts > ...
You, 14 hours ago | 2 authors (You and others)
1   import { BASE_URL } from "../constants/constants";
2   import { httpGET } from "../utility/http";
3
4   export enum AvgPricePropertiesEndpoints {
5     GetAvgPrice = "/api/properties/user/findAvgPrice",
6   }
7   // Review: use descriptive variable names like city, categoryId ..
8   export const AvgPrice = (
9     query1: string,
10    query2: number,
11    query3: number,
12    query4: number
13  ) => {
14    return httpGET(
15      `${BASE_URL}${AvgPricePropertiesEndpoints.GetAvgPrice}/?city=${query1}&categoryId=${query2}&room=${query3}&size=${query
16    );
17  };
18
```



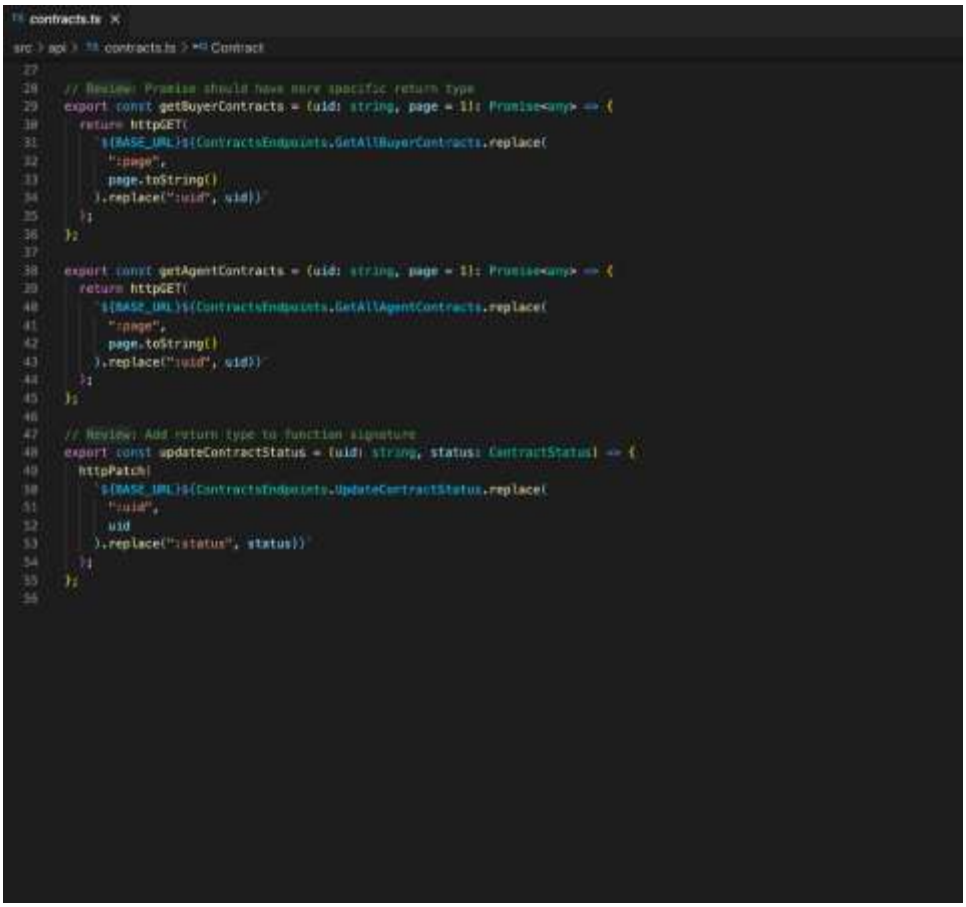
```tsx
// Reviewer: File extension should be .ts and merge with api/companies.ts
import axios from "axios";
import { BASE_URL } from "../constants/constants";
import { httpGET } from "../utility/http";
export enum CompanyEndpoints {
  GetCompanies = "/api/company/",
  DeleteCompanies = "/api/company/",
  AddCompanies = "/api/company/",
  ListOfAgents = "/api/company/agentList/",
  ListOfProperty = "/api/properties/user/approve/agent/property/",
  PropertyImage = "/api/properties/property/image/",
}

export const propertyImages = (id: Number) => {
  return httpGET(`${BASE_URL}${CompanyEndpoints.PropertyImage}?id=${id}`);
};
export const listOfCompanies = (page: Number) => {
  return httpGET(`${BASE_URL}${CompanyEndpoints.GetCompanies}${page}`);
};
export const companyAgent = (page: Number, companyId: Number) => {
  return httpGET(
    `${BASE_URL}${CompanyEndpoints.ListOfAgents}${page}/?companyId=${companyId}`
  );
};

export const listOfPropertyByAgent = (page: Number, agentId: Number) => {
  return httpGET(
    `${BASE_URL}${CompanyEndpoints.ListOfProperty}${page}/?agentId=${agentId}`
  );
};
export const postCompanies = (companies: any) => {
  return axios.post(`${BASE_URL}${CompanyEndpoints.AddCompanies}`, companies);
};

export const deleteCompanies = (id: Number) => {
  return axios.delete(`${BASE_URL}${CompanyEndpoints.DeleteCompanies}${id}`);
};
```



```
contact.ts
src > api > contact.ts > ...
You, 14 hours ago | 2 authors (You and others)
import axios from "axios";
import { BASE_URL } from "../constants/constants";
import { httpGET } from "../utility/http";
export enum ContactEndpoints {
  // Reviewr Fix Spellings
  CrateContact = "/api/contactUs/",
  Queries = "/api/contactUs/",
  DeleteQueries = "/api/contactUs/",
}

export const crateContact = (user: any) => {
  return axios.post(`${BASE_URL}${ContactEndpoints.CrateContact}`, user);
};

export const UserQueries = (page = 1) => {
  return httpGET(`${BASE_URL}${ContactEndpoints.Queries}/${page}`);
};

export const DeleteQueries = (id: Number) => {
  return axios.delete(`${BASE_URL}${ContactEndpoints.DeleteQueries}${id}`);
};

```



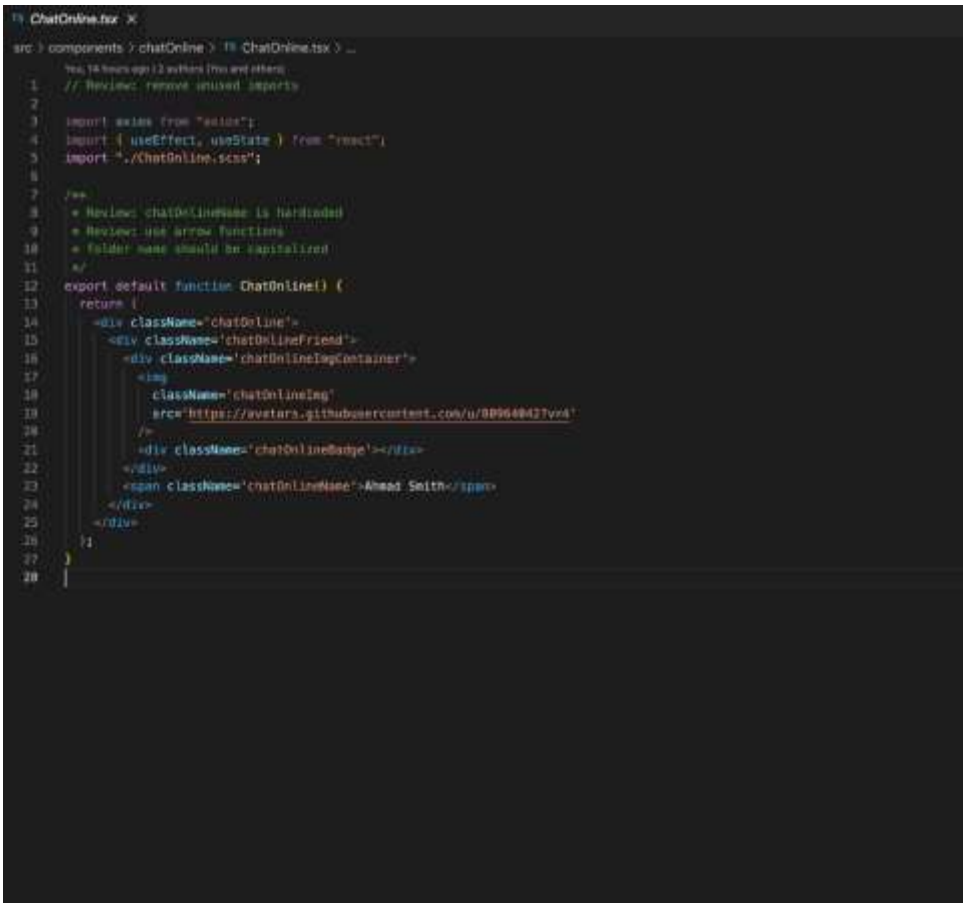
```ts
import { BASE_URL } from "../constants/constants";
import { httpGET, httpPatch } from "../utility/http";

enum ContractsEndpoints {
  GetAllBuyerContracts = "/api/contract/buyer/:page/?buyerId=:uid",
  GetAllAgentContracts = "/api/contract/agent/?agentId=:uid&page=:page",
  UpdateContractStatus = "/api/contracts/:uid?status=:status",
}

enum ContractStatus {
  Accept = "accept",
  End = "end",
}


export interface Contract {
  buyerId: number;
  agentId: number;
  description: string;
  dateCreate: string;
  dateValid: string;
  propertyDetail: { id: number; images: string };
  title: string;
  status: string;
  // Review: should be approved, requires frontend and backend changes
  approve: string;
}

// Review: Promise should have more specific return type
export const getBuyerContracts = (uid: string, page = 1): Promise<any> => {
  return httpGET(
    `${BASE_URL}${ContractsEndpoints.GetAllBuyerContracts.replace(
      ":page",
      page.toString()
    ).replace(":uid", uid)}`
  );
};

export const getAgentContracts = (uid: string, page = 1): Promise<any> => {
  return httpGET(
    `${BASE_URL}${ContractsEndpoints.GetAllAgentContracts.replace(
      ":page",
      page.toString()
    ).replace(":uid", uid)}`
  );
};
```


contracts.ts
src > api > contracts.ts > Contract
// Review: Promise should have more specific return type
export const getBuyerContracts = (uid: string, page = 1): Promise<any> => {
return httpGET(
`${BASE_URL}${ContractsEndpoints.GetAllBuyerContracts.replace(
":page",
page.toString()
).replace(":uid", uid)}`
);
};
export const getAgentContracts = (uid: string, page = 1): Promise<any> => {
return httpGET(
`${BASE_URL}${ContractsEndpoints.GetAllAgentContracts.replace(
":page",
page.toString()
).replace(":uid", uid)}`
);
};
// Review: Add return type to function signature
export const updateContractStatus = (uid: string, status: ContractStatus) => {
httpPatch(
`${BASE_URL}${ContractsEndpoints.UpdateContractStatus.replace(
":uid",
uid
).replace(":status", status)}`
);
};

Carousel.tsx

src > components > Carousel > Carousel.tsx > ...

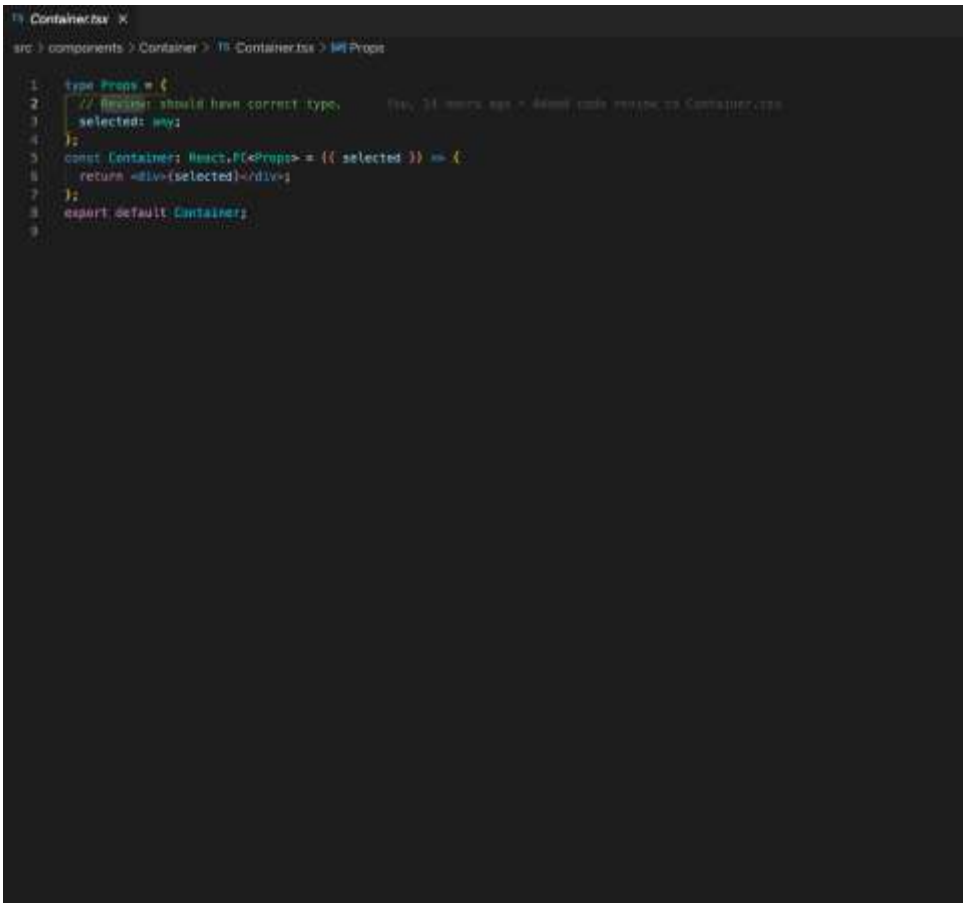
```tsx
import React from "react";
import Carousel from "react-material-ui-carousel";
import { BASE_URL } from "../../constants/constants";
import "./Carousel.scss";

interface CarouselProps {
  // Review: images should have proper types
  images?: any[];
  rounded?: boolean;
}
const CarouselComponent: React.FC<CarouselProps> = ({
  images = [],
  rounded = false,
}) => {
  return (
    <div
      className={`carousel-component carousel-component--${
        rounded ? "rounded" : ""
      }`}>
      <Carousel
        animation='slide'
        interval={3000}
        cycleNavigation
        autoPlay
        indicators={false}>
        {images.map((item, i) => (
          <div className='slide' key={i}>
            <img src={`${BASE_URL}/${item}`}></img>
          </div>
        ))}
      </Carousel>
    </div>
  );
};

export default CarouselComponent;
```


ChatOnline.tsx
src > components > chatOnline > ChatOnline.tsx > ...
You, 14 hours ago | 2 authors (You and others)
// Reviewer: remove unused imports
import axios from "axios";
import { useEffect, useState } from "react";
import "./ChatOnline.scss";
/**
* Reviewer: chatOnlineName is hardcoded
* Reviewer: use arrow functions
* Folder name should be capitalized
*/
export default function ChatOnline() {
return (

<img
className='chatOnlineImg'
src='https://avatars.githubusercontent.com/u/80964027?v=4'
/>

Ahmad Smith

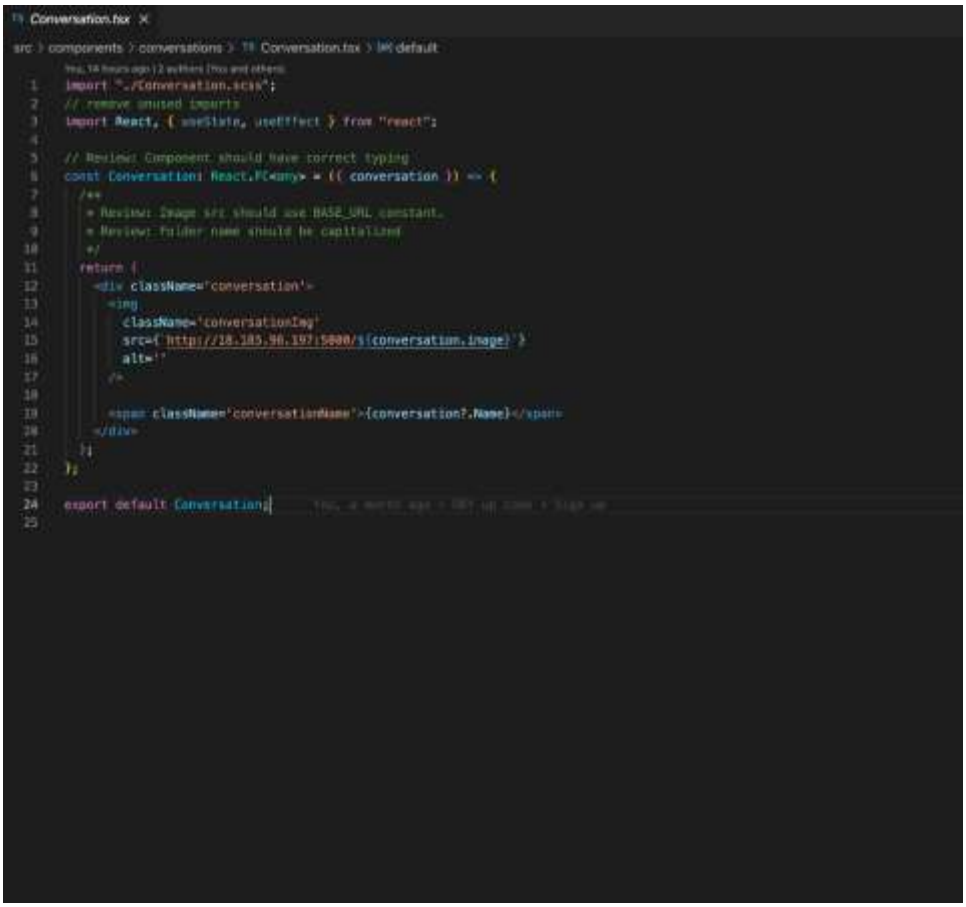
);
}


Container.tsx
src > components > Container > Container.tsx > Props
type Props = {
// Reviewer should have correct type.
selected: any;
};
const Container: React.FC<Props> = ({ selected }) => {
return {selected} ;
};
export default Container;

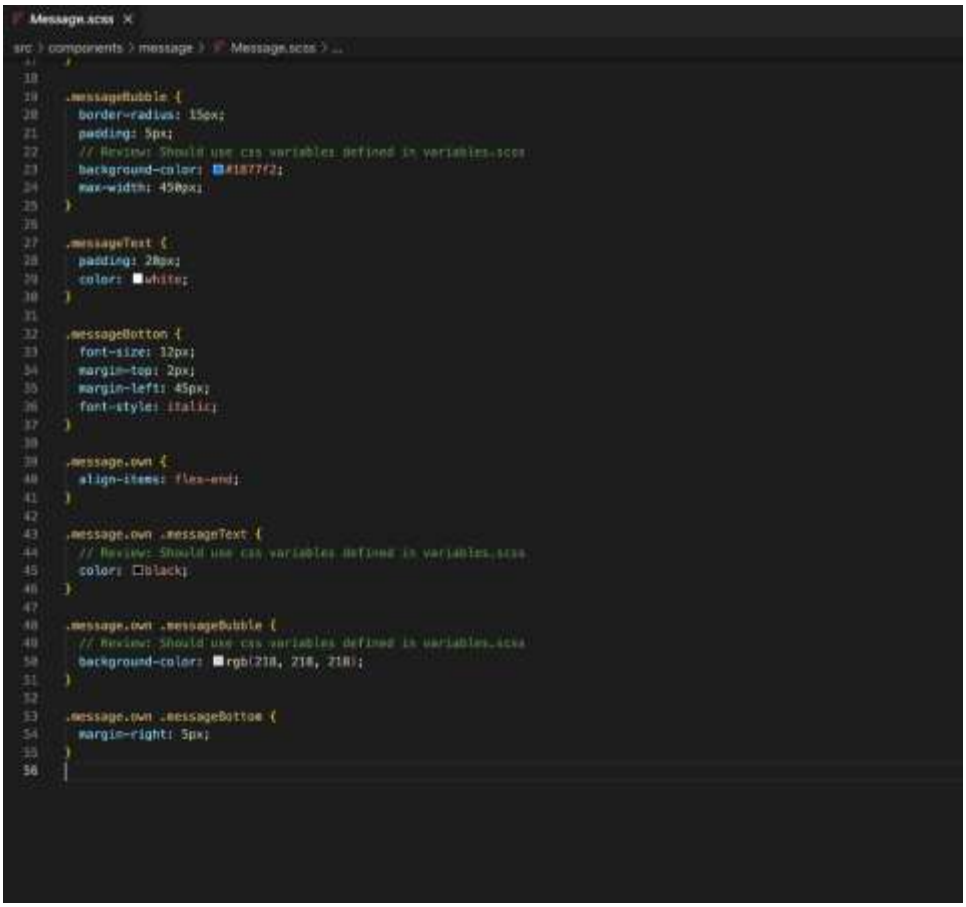
Conversation.scss

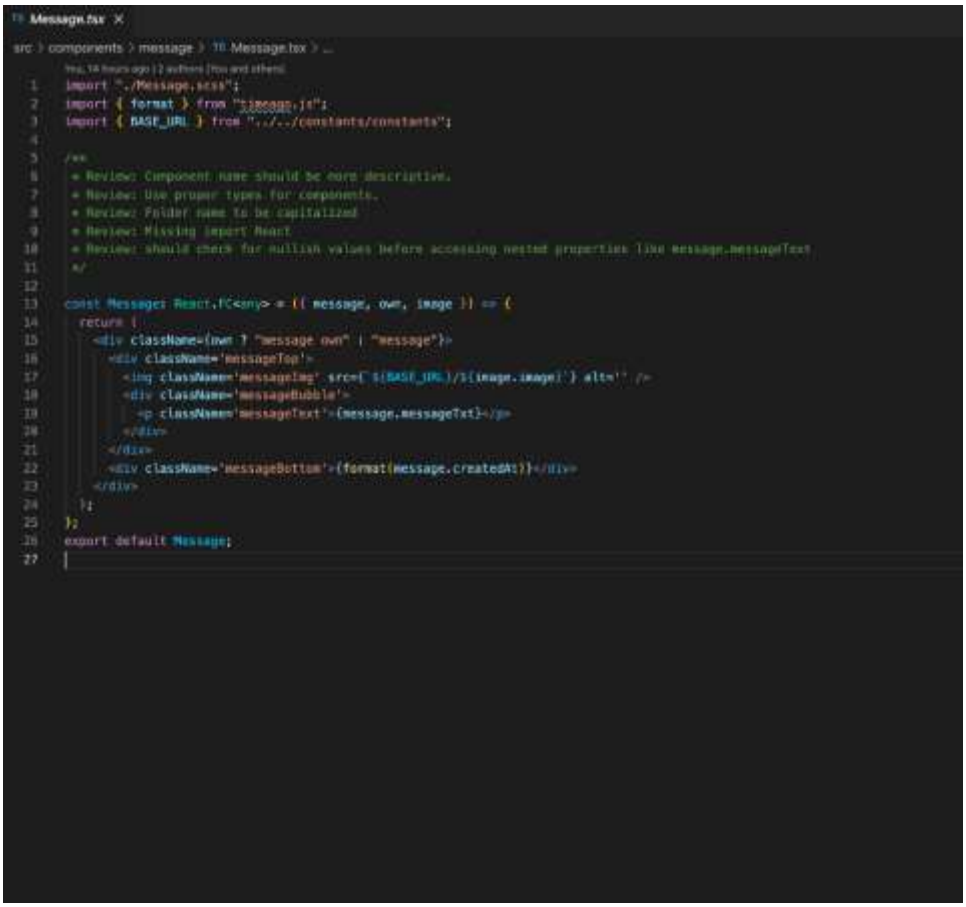
src > components > conversations > Conversation.scss > ...

```scss
.conversation {
  display: flex;
  border-radius: 15px;
  align-items: center;
  padding: 10px;
  cursor: pointer;
  margin-top: 20px;
}

.conversation:hover {
  // Review: Should use css variables defined in variables.scss
  background-color: rgba(190, 189, 189, 0.548);
}

.conversationImg {
  width: 40px;
  height: 40px;
  border-radius: 50%;
  object-fit: cover;
  margin-right: 20px;
}

.conversationName {
  font-weight: 500;
}

@media screen and (max-width: 768px) {
  .conversationName {
    display: none;
  }
}
```

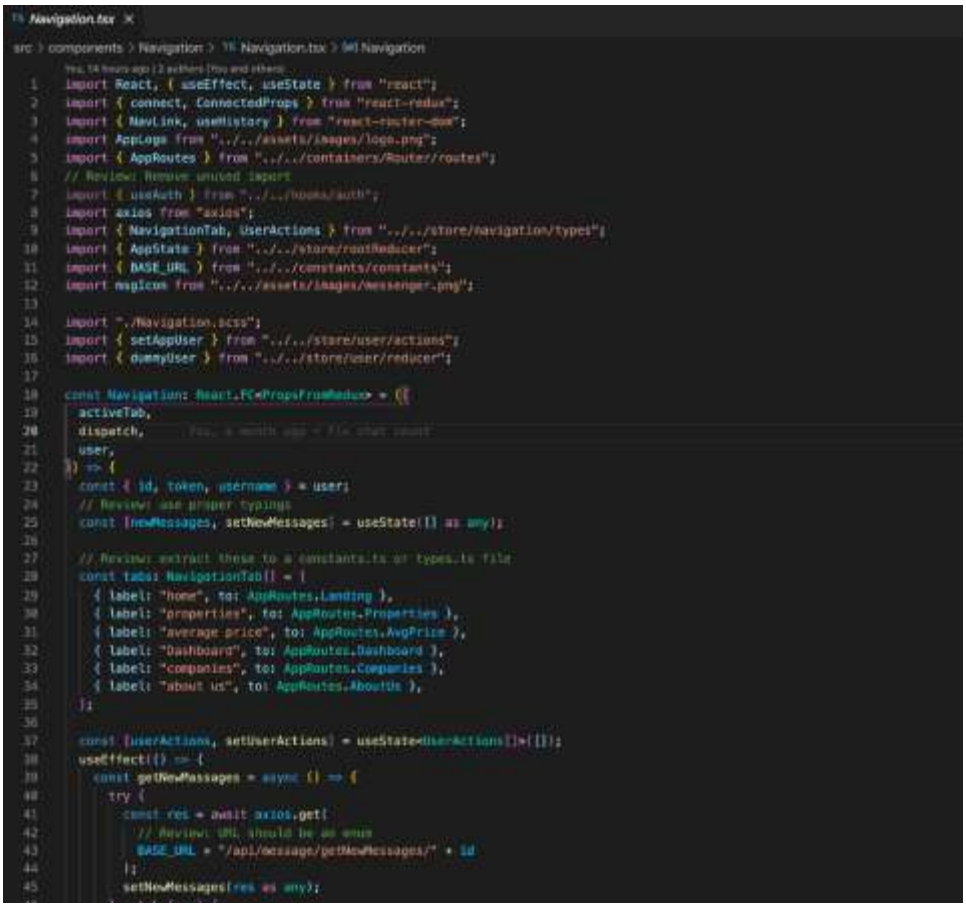


```
Conversation.tsx
src > components > conversations > Conversation.tsx > default
  You, 14 hours ago | 2 authors (You and others)
1 import "./Conversation.scss";
2 // remove unused imports
3 import React, { useState, useEffect } from "react";
4
5 // Review: Component should have correct typing
6 const Conversation: React.FC<any> = ({ conversation }) => {
7   /**
8    * Review: Image src should use BASE_URL constant.
9    * Review: Folder name should be capitalized
10   */
11   return (
12     <div className='conversation'>
13       <img
14         className='conversationImg'
15         src={`http://18.185.98.197:5000/${conversation.image}`}
16         alt=''
17         />
18
19       <span className='conversationName'>{conversation?.Name}</span>
20     </div>
21   );
22 };
23
24 export default Conversation;
25
```

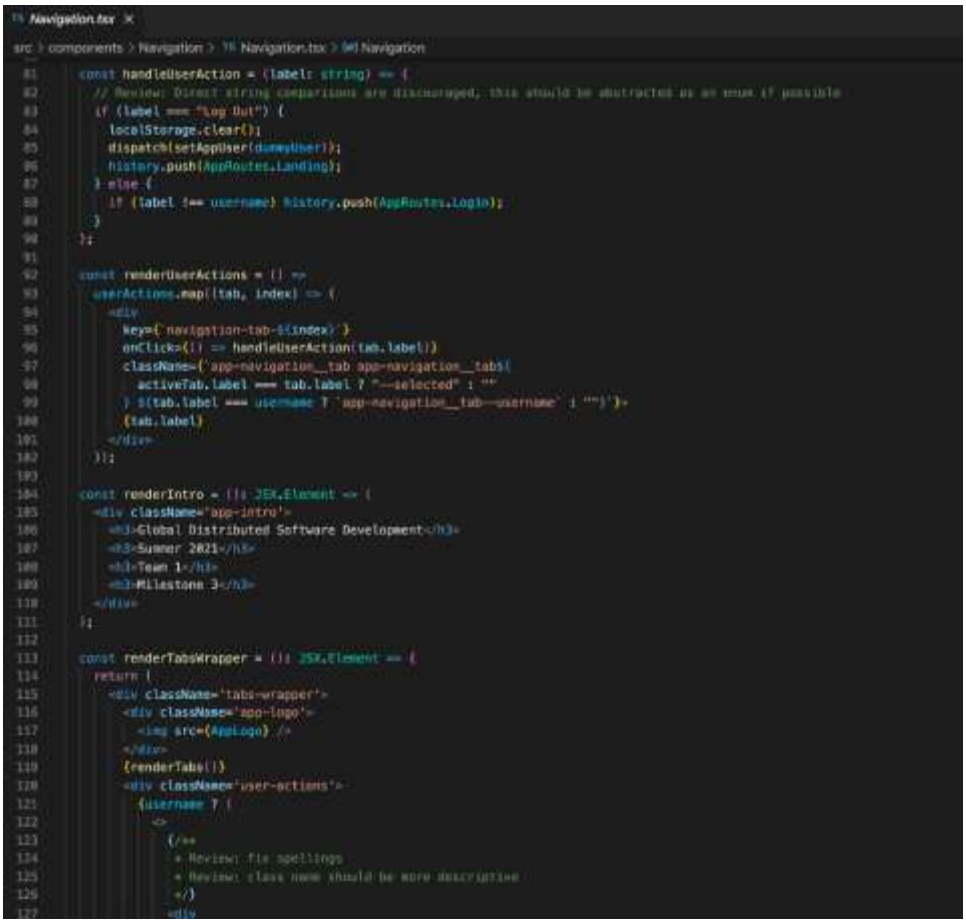

Message.scss
src > components > message > Message.scss > ...
.messageBubble {
border-radius: 15px;
padding: 5px;
// Review: Should use css variables defined in variables.scss
background-color: #1877f2;
max-width: 450px;
}
.messageText {
padding: 20px;
color: white;
}
.messageBotton {
font-size: 12px;
margin-top: 2px;
margin-left: 45px;
font-style: italic;
}
.message.own {
align-items: flex-end;
}
.message.own .messageText {
// Review: Should use css variables defined in variables.scss
color: black;
}
.message.own .messageBubble {
// Review: Should use css variables defined in variables.scss
background-color: rgb(218, 218, 218);
}
.message.own .messageBottom {
margin-right: 5px;
}



```
Message.tsx
src > components > message > Message.tsx > ...
You, 14 hours ago | 2 authors (You and others)
import "./Message.scss";
import { format } from "timeago.js";
import { BASE_URL } from "../../constants/constants";

/**
 * Review: Component name should be more descriptive.
 * Review: Use proper types for components.
 * Review: Folder name to be capitalized
 * Review: Missing import React
 * Review: should check for nullish values before accessing nested properties like message.messageText
 */

const Messages React.FC<any> = ({ message, own, image }) => {
  return (
    <div className={own ? "message own" : "message"}>
      <div className='messageTop'>
        <img classNamen='messageImg' src={`${BASE_URL}/${image.image}`} alt='' />
        <div classNamen='messageBubble'>
          <p classNamen='messageText'>{message.messageTxt}</p>
        </div>
      </div>
      <div classNamen='messageBottom'>{format(message.createdAt)}</div>
    </div>
  );
};
export default Message;
```



```
Navigation.tsx
src > components > Navigation > Navigation.tsx > Navigation
You, 14 hours ago | 2 authors (You and others)
import React, { useEffect, useState } from "react";
import { connect, ConnectedProps } from "react-redux";
import { NavLink, useHistory } from "react-router-dom";
import AppLogo from "../../assets/images/logo.png";
import { AppRoutes } from "../../containers/Router/routes";
// Review: Remove unused import
import { useAuth } from "../../hooks/auth";
import axios from "axios";
import { NavigationTab, UserActions } from "../../store/navigation/types";
import { AppState } from "../../store/rootReducer";
import { BASE_URL } from "../../constants/constants";
import msgIcon from "../../assets/images/messenger.png";

import "./Navigation.scss";
import { setAppUser } from "../../store/user/actions";
import { dummyUser } from "../../store/user/reducer";

const Navigation: React.FC<PropsFromRedux> = ({
  activeTab,
  dispatch,      You, a month ago • Fix chat count
  user,
}) => {
  const { id, token, username } = user;
  // Review: use proper typings
  const [newMessages, setNewMessages] = useState([] as any);

  // Review: extract these to a constants.ts or types.ts file
  const tabs: NavigationTab[] = [
    { label: "home", to: AppRoutes.Landing },
    { label: "properties", to: AppRoutes.Properties },
    { label: "average price", to: AppRoutes.AvgPrice },
    { label: "Dashboard", to: AppRoutes.Dashboard },
    { label: "companies", to: AppRoutes.Companies },
    { label: "about us", to: AppRoutes.AboutUs },
  ];

  const [userActions, setUserActions] = useState<UserActions[]>([]);
  useEffect(() => {
    const getNewMessages = async () => {
      try {
        const res = await axios.get(
          // Review: URL should be an enum
          BASE_URL + "/api/message/getNewMessages/" + id
        );
        setNewMessages(res as any);
```


Navigation.tsx
src > components > Navigation > Navigation.tsx > Navigation
const handleUserAction = (label: string) => {
// Review: Direct string comparisons are discouraged, this should be abstracted as an enum if possible
if (label === "Log Out") {
localStorage.clear();
dispatch(setAppUser(dummyUser));
history.push(AppRoutes.Landing);
} else {
if (label !== username) history.push(AppRoutes.Login);
}
};
const renderUserActions = () =>
userActions.map((tab, index) => (
<div
key={`navigation-tab-${index}`}
onClick={() => handleUserAction(tab.label)}
className={'app-navigation__tab app-navigation__tabs'
activeTab.label === tab.label ? "--selected" : ""
} ${tab.label === username ? 'app-navigation__tab--username' : ""}`}>
{tab.label}

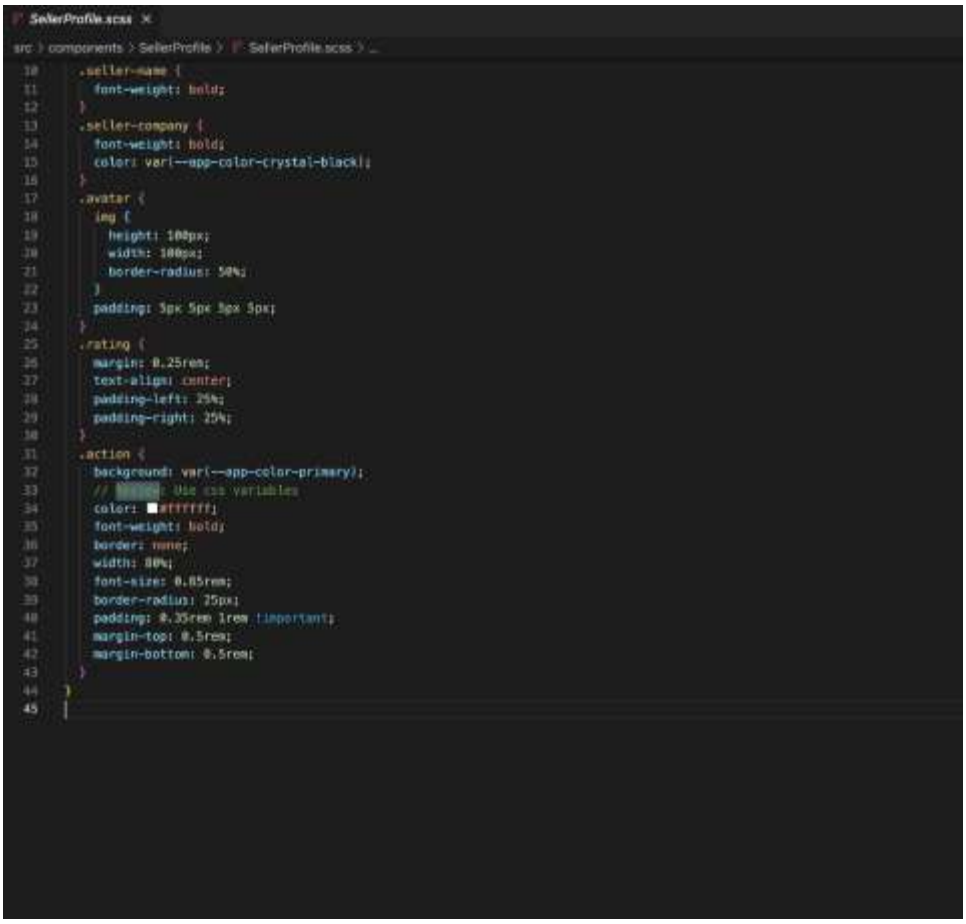
));
const renderIntro = (): JSX.Element => (

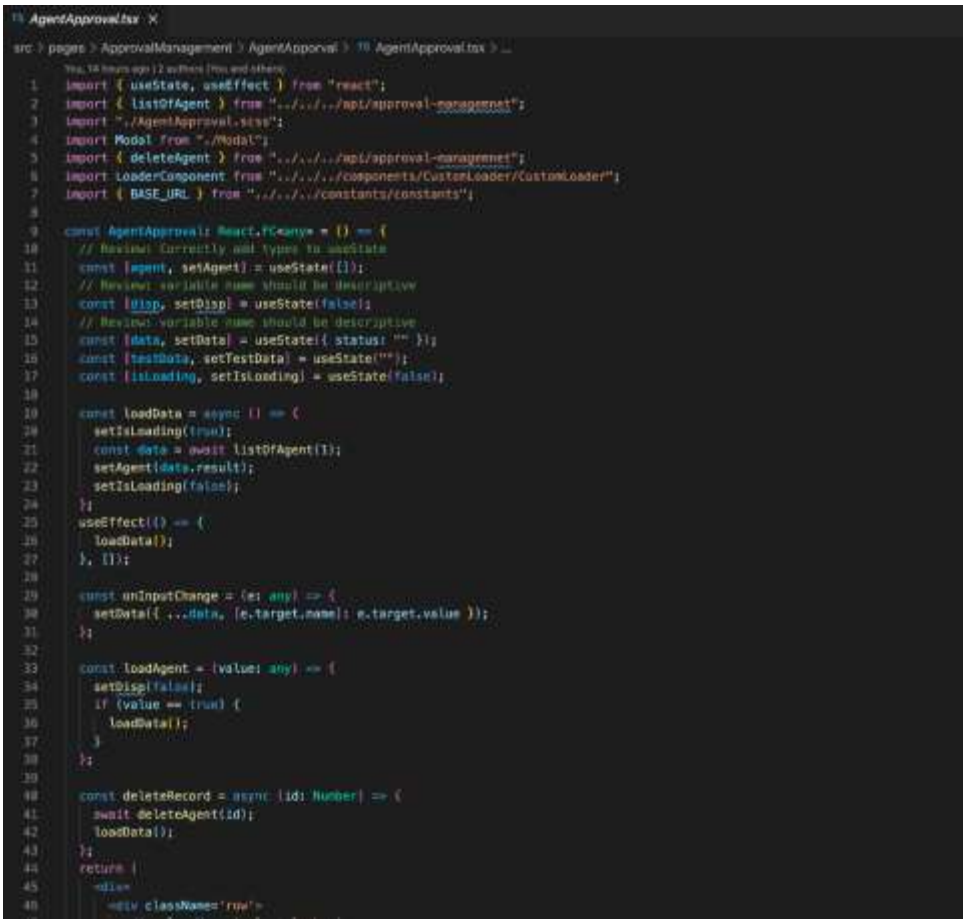
<h3>Global Distributed Software Development</h3>
<h3>Summer 2021</h3>
<h3>Team 1</h3>
<h3>Milestone 3</h3>

);
const renderTabsWrapper = (): JSX.Element => {
return (

{renderTabs()}

{username ? (
<>
{/**
* Review: fix spellings
* Review: class name should be more descriptive
*/}
<div


SellerProfile.scss
src > components > SellerProfile > SellerProfile.scss > ...
.seller-name {
font-weight: bold;
}
.seller-company {
font-weight: bold;
color: var(--app-color-crystal-black);
}
.avatar {
img {
height: 100px;
width: 100px;
border-radius: 50%;
}
padding: 5px 5px 3px 5px;
}
.rating {
margin: 0.25rem;
text-align: center;
padding-left: 25%;
padding-right: 25%;
}
.action {
background: var(--app-color-primary);
// Todo: Use css variables
color: #ffffff;
font-weight: bold;
border: none;
width: 80%;
font-size: 0.85rem;
border-radius: 25px;
padding: 0.35rem 1rem !important;
margin-top: 0.5rem;
margin-bottom: 0.5rem;
}
}

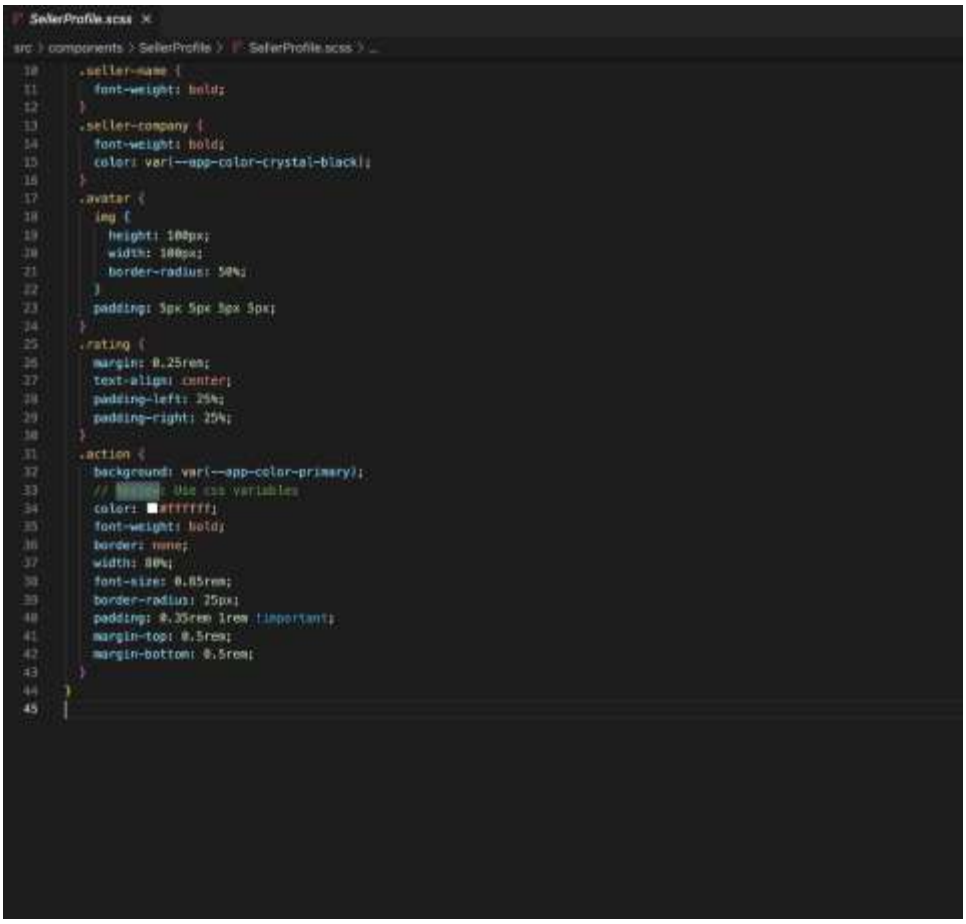


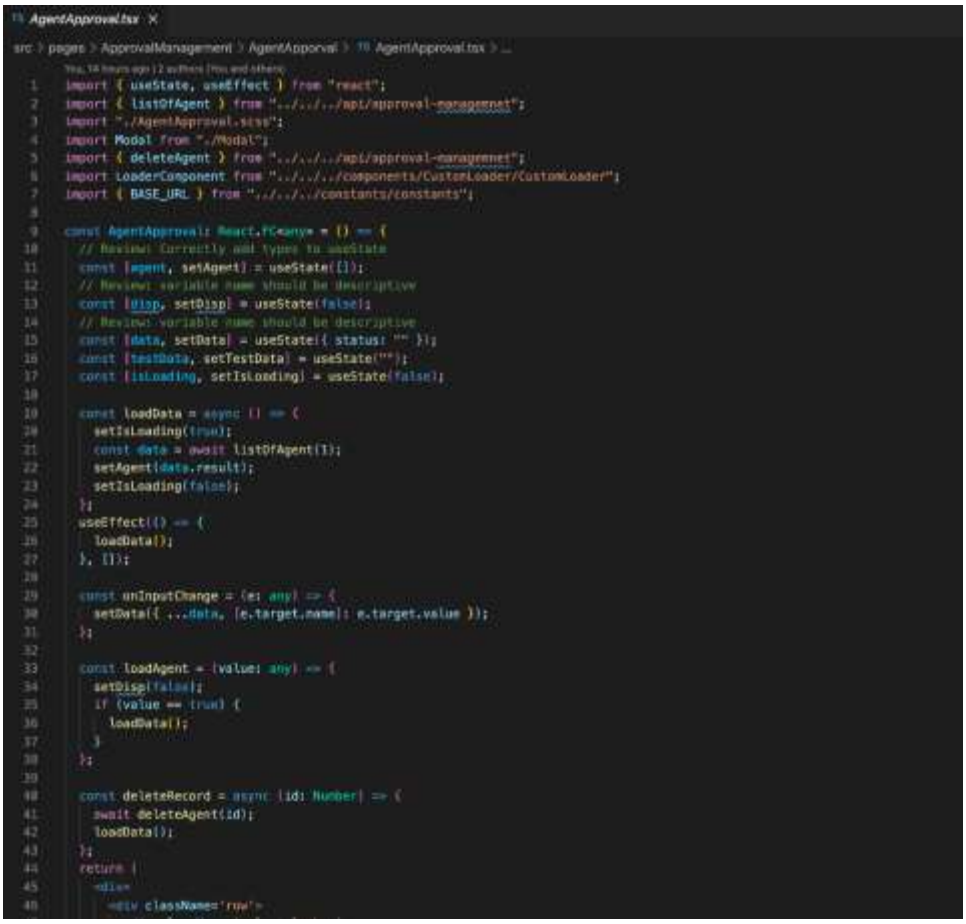
```
AgentApproval.tsx
src > pages > ApprovalManagement > AgentApporval > AgentApproval.tsx > ...
You, 14 hours ago | 2 authors (You and others)
import { useState, useEffect } from "react";
import { listOfAgent } from "../../../api/approval-managemnet";
import "./AgentApproval.scss";
import Modal from "./Modal";
import { deleteAgent } from "../../../api/approval-managemnet";
import LoaderComponent from "../../../components/CustomLoader/CustomLoader";
import { BASE_URL } from "../../../constants/constants";

const AgentApproval: React.FC<any> = () => {
  // Review: Correctly add types to useState
  const [agent, setAgent] = useState([]);
  // Review: variable name should be descriptive
  const [disp, setDisp] = useState(false);
  // Review: variable name should be descriptive
  const [data, setData] = useState({ status: "" });
  const [testData, setTestData] = useState("");
  const [isLoading, setIsLoading] = useState(false);

  const loadData = async () => {
    setIsLoading(true);
    const data = await listOfAgent(1);
    setAgent(data.result);
    setIsLoading(false);
  };
  useEffect(() => {
    loadData();
  }, []);

  const onInputChange = (e: any) => {
    setData({ ...data, [e.target.name]: e.target.value });
  };

  const loadAgent = (value: any) => {
    setDisp(false);
    if (value == true) {
      loadData();
    }
  };

  const deleteRecord = async (id: Number) => {
    await deleteAgent(id);
    loadData();
  };
  return (
    <div>
      <div className='row'>
```

Navigation.tsx

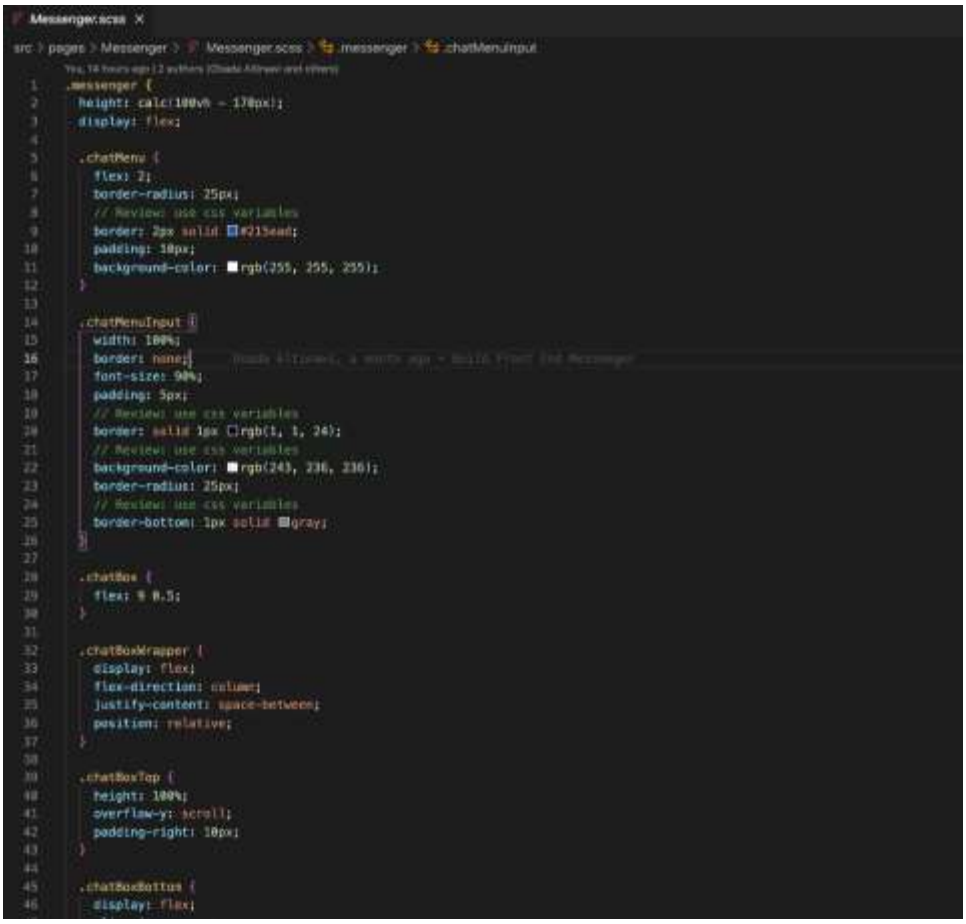
src > components > Navigation > Navigation.tsx > Navigation

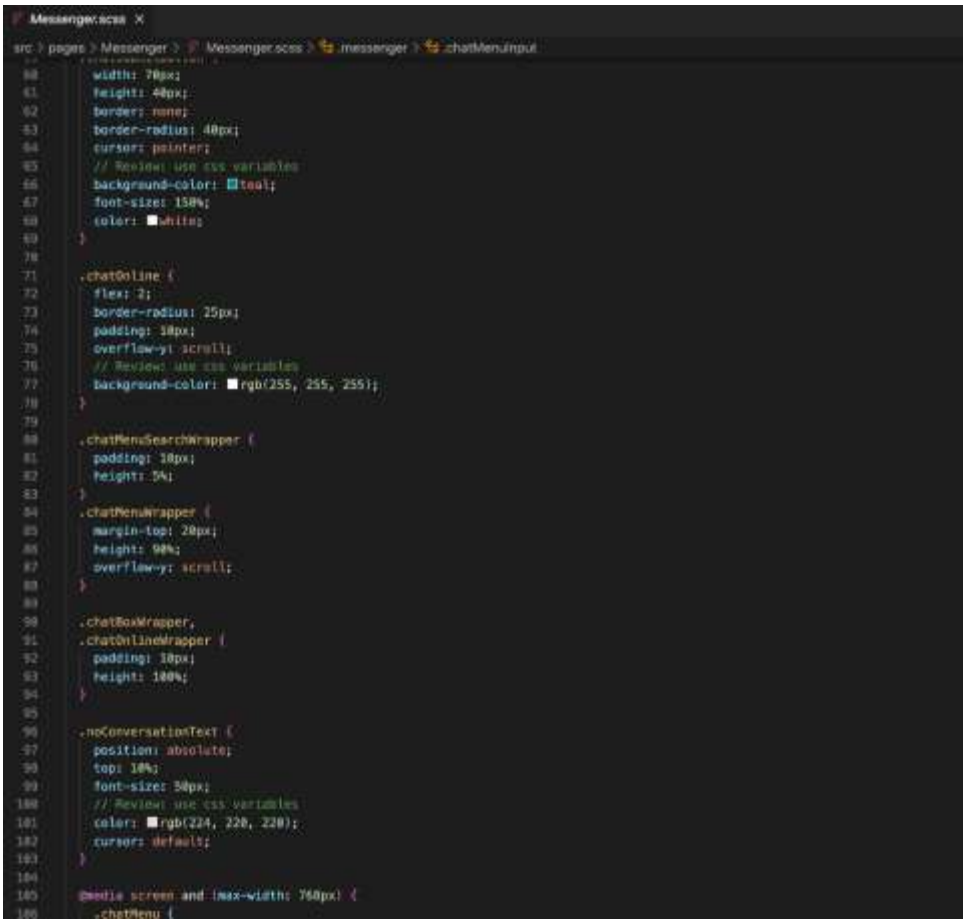
```tsx
  const handleUserAction = (label: string) => {
    // Review: Direct string comparisons are discouraged, this should be abstracted as an enum if possible
    if (label === "Log Out") {
      localStorage.clear();
      dispatch(setAppUser(dummyUser));
      history.push(AppRoutes.Landing);
    } else {
      if (label !== username) history.push(AppRoutes.Login);
    }
  };

  const renderUserActions = () =>
    userActions.map((tab, index) => (
      <div
        key={`navigation-tab-${index}`}
        onClick={() => handleUserAction(tab.label)}
        className={`app-navigation__tab app-navigation__tabs${
          activeTab.label === tab.label ? "--selected" : ""
        } ${tab.label === username ? `app-navigation__tab--username` : ""}`}>
        {tab.label}
      </div>
    ));

  const renderIntro = (): JSX.Element => (
    <div className='app-intro'>
      <h3>Global Distributed Software Development</h3>
      <h3>Summer 2021</h3>
      <h3>Team 1</h3>
      <h3>Milestone 3</h3>
    </div>
  );

  const renderTabsWrapper = (): JSX.Element => {
    return (
      <div className='tabs-wrapper'>
        <div className='app-logo'>
          <img src={AppLogo} />
        </div>
        {renderTabs()}
        <div className='user-actions'>
          {username ? (
            <>
              {/**
               * Review: fix spellings
               * Review: class name should be more descriptive
               */}
              <div
```

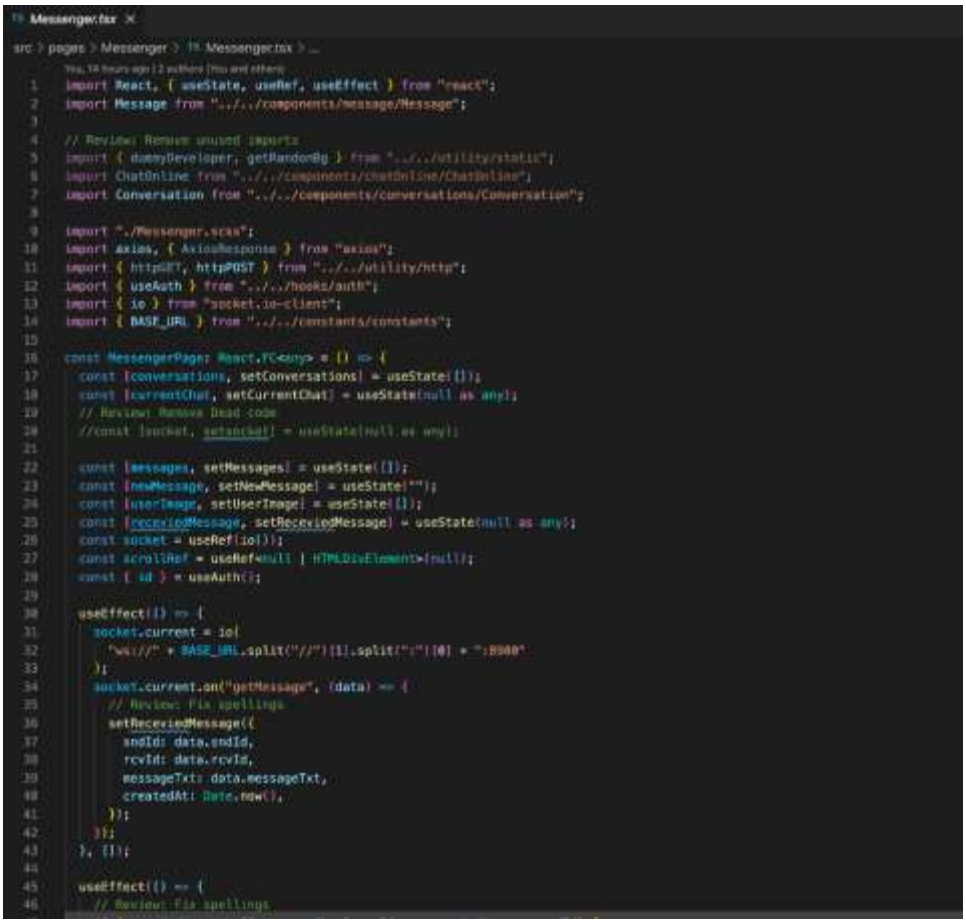

SellerProfile.scss
src > components > SellerProfile > SellerProfile.scss > ...
.seller-name {
font-weight: bold;
}
.seller-company {
font-weight: bold;
color: var(--app-color-crystal-black);
}
.avatar {
img {
height: 100px;
width: 100px;
border-radius: 50%;
}
padding: 5px 5px 3px 5px;
}
.rating {
margin: 0.25rem;
text-align: center;
padding-left: 25%;
padding-right: 25%;
}
.action {
background: var(--app-color-primary);
// Todo: Use css variables
color: #ffffff;
font-weight: bold;
border: none;
width: 80%;
font-size: 0.85rem;
border-radius: 25px;
padding: 0.35rem 1rem !important;
margin-top: 0.5rem;
margin-bottom: 0.5rem;
}
}



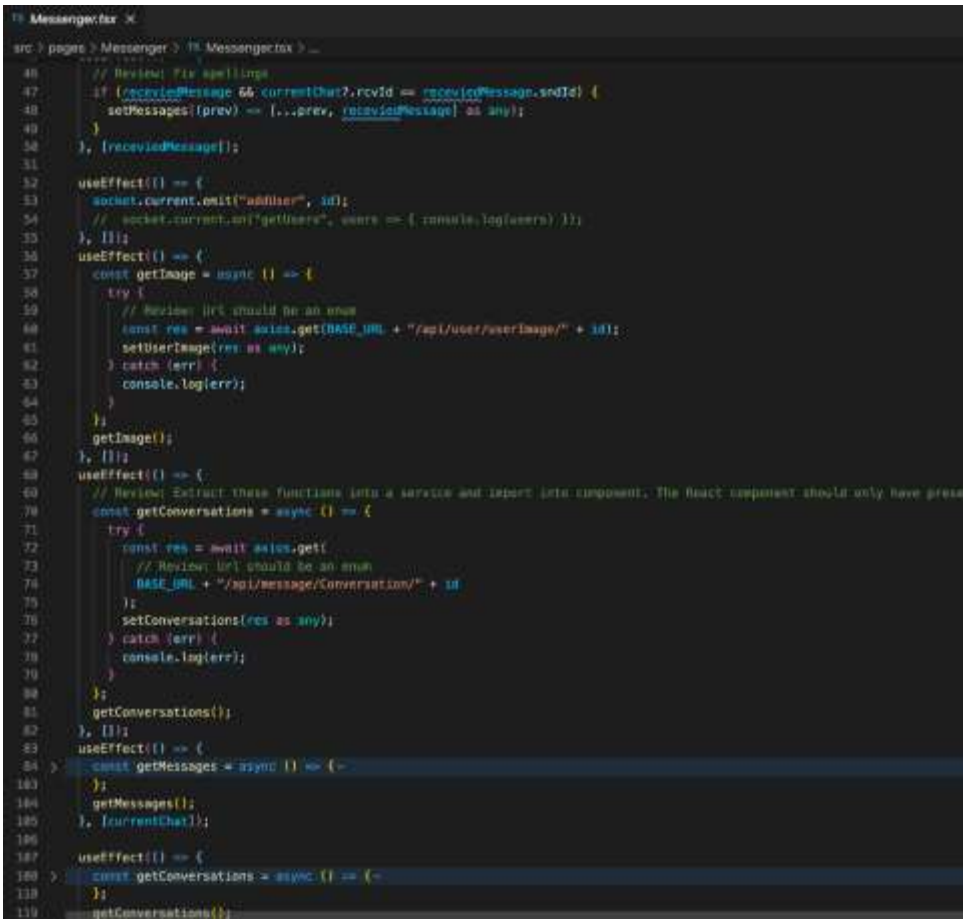
```
AgentApproval.tsx
src > pages > ApprovalManagement > AgentApporval > AgentApproval.tsx > ...
You, 14 hours ago | 2 authors (You and others)
import { useState, useEffect } from "react";
import { listOfAgent } from "../../../api/approval-managemnet";
import "./AgentApproval.scss";
import Modal from "./Modal";
import { deleteAgent } from "../../../api/approval-managemnet";
import LoaderComponent from "../../../components/CustomLoader/CustomLoader";
import { BASE_URL } from "../../../constants/constants";

const AgentApproval: React.FC<any> = () => {
  // Review: Correctly add types to useState
  const [agent, setAgent] = useState([]);
  // Review: variable name should be descriptive
  const [disp, setDisp] = useState(false);
  // Review: variable name should be descriptive
  const [data, setData] = useState({ status: "" });
  const [testData, setTestData] = useState("");
  const [isLoading, setIsLoading] = useState(false);

  const loadData = async () => {
    setIsLoading(true);
    const data = await listOfAgent(1);
    setAgent(data.result);
    setIsLoading(false);
  };
  useEffect(() => {
    loadData();
  }, []);

  const onInputChange = (e: any) => {
    setData({ ...data, [e.target.name]: e.target.value });
  };

  const loadAgent = (value: any) => {
    setDisp(false);
    if (value == true) {
      loadData();
    }
  };

  const deleteRecord = async (id: Number) => {
    await deleteAgent(id);
    loadData();
  };
  return (
    <div>
      <div className='row'>
```


Messenger.scss
src > pages > Messenger > Messenger.scss > .messenger > .chatMenuInput
.messenger {
height: calc(100vh - 170px);
display: flex;
.chatMenu {
flex: 2;
border-radius: 25px;
// Review: use css variables
border: 2px solid #215ead;
padding: 10px;
background-color: rgb(255, 255, 255);
}
.chatMenuInput {
width: 100%;
border: none;
font-size: 90%;
padding: 5px;
// Review: use css variables
border: solid 1px rgb(1, 1, 24);
// Review: use css variables
background-color: rgb(243, 236, 236);
border-radius: 25px;
// Review: use css variables
border-bottom: 1px solid gray;
}
.chatBox {
flex: 9 0.5;
}
.chatBoxWrapper {
display: flex;
flex-direction: column;
justify-content: space-between;
position: relative;
}
.chatBoxTop {
height: 100%;
overflow-y: scroll;
padding-right: 10px;
}
.chatBoxBottom {
display: flex;

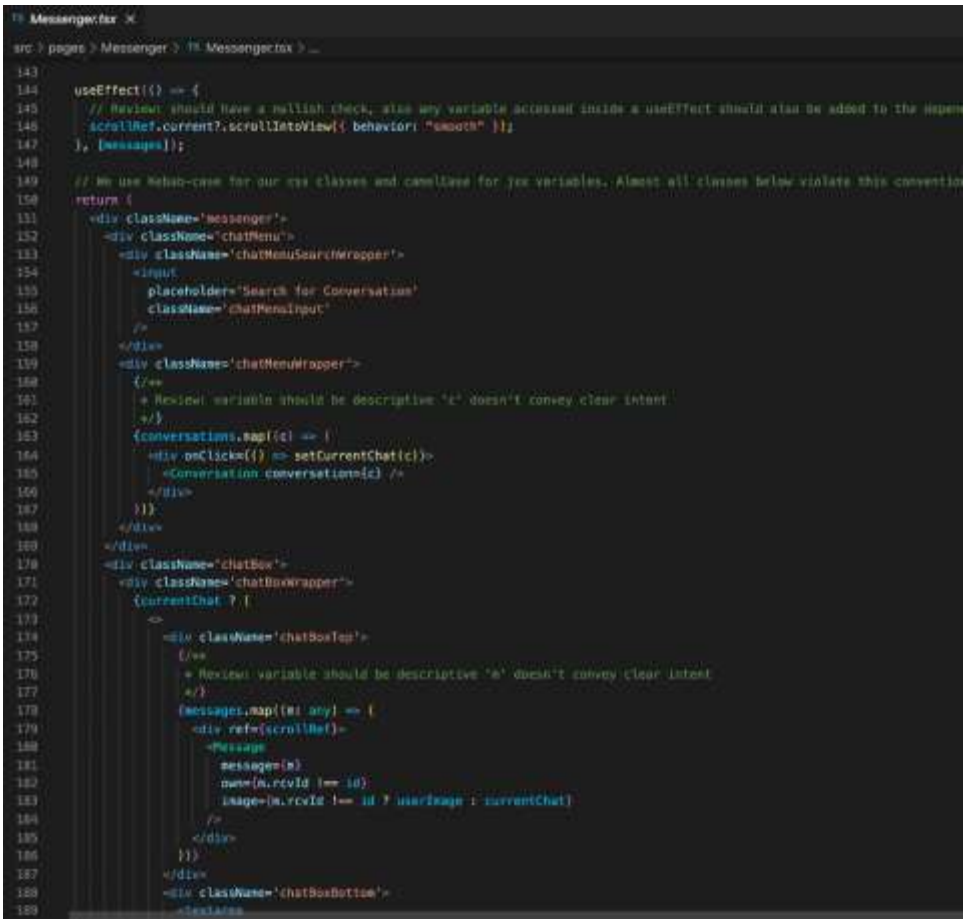

Messenger.scss
src > pages > Messenger > Messenger.scss > .messenger > .chatMenuInput
width: 70px;
height: 40px;
border: none;
border-radius: 40px;
cursor: pointer;
// Review: use css variables
background-color: teal;
font-size: 150%;
color: white;
.chatOnline {
flex: 2;
border-radius: 25px;
padding: 10px;
overflow-y: scroll;
// Review: use css variables
background-color: rgb(255, 255, 255);
.chatMenuSearchWrapper {
padding: 10px;
height: 5%;
.chatMenuWrapper {
margin-top: 20px;
height: 90%;
overflow-y: scroll;
.chatBoxWrapper,
.chatOnlineWrapper {
padding: 10px;
height: 100%;
.noConversationText {
position: absolute;
top: 10%;
font-size: 50px;
// Review: use css variables
color: rgb(224, 220, 220);
cursor: default;
@media screen and (max-width: 760px) {
.chatMenu {



```tsx
Messenger.tsx
src > pages > Messenger > Messenger.tsx > ...
You, 14 hours ago | 2 authors (You and others)
import React, { useState, useRef, useEffect } from "react";
import Message from "../../components/message/Message";

// Review: Remove unused imports
import { dummyDeveloper, getRandomBg } from "../../utility/static";
import ChatOnline from "../../components/chatOnline/ChatOnline";
import Conversation from "../../components/conversations/Conversation";

import "./Messenger.scss";
import axios, { AxiosResponse } from "axios";
import { httpGET, httpPOST } from "../../utility/http";
import { useAuth } from "../../hooks/auth";
import { io } from "socket.io-client";
import { BASE_URL } from "../../constants/constants";

const MessengerPage: React.FC<any> = () => {
  const [conversations, setConversations] = useState([]);
  const [currentChat, setCurrentChat] = useState(null as any);
  // Review: Remove Dead code
  //const [socket, setSocket] = useState(null as any);

  const [messages, setMessages] = useState([]);
  const [newMessage, setNewMessage] = useState("");
  const [userImage, setUserImage] = useState([]);
  const [receviedMessage, setRecevedMessage] = useState(null as any);
  const socket = useRef(io());
  const scrollRef = useRef<null | HTMLDivElement>(null);
  const { id } = useAuth();

  useEffect(() => {
    socket.current = io(
      "ws://" + BASE_URL.split("//")[1].split(":")[0] + ":8900"
    );
    socket.current.on("getMessage", (data) => {
      // Review: Fix spellings
      setRecevedMessage({
        sndId: data.sndId,
        rcvId: data.rcvId,
        messageTxt: data.messageTxt,
        createdAt: Date.now(),
      });
    });
  }, []);

  useEffect(() => {
    // Review: Fix spellings
```



```
Messenger.tsx
src > pages > Messenger > Messenger.tsx > ...
    // Review: Fix spellings
    if (recevieMessage && currentChat?.rcvId == recevieMessage.sndId) {
      setMessages((prev) => [...prev, recevieMessage] as any);
    }
  }, [recevieMessage]);

  useEffect(() => {
    socket.current.emit("addUser", id);
    //  socket.current.on("getUsers", users => { console.log(users) });
  }, []);
  useEffect(() => {
    const getImage = async () => {
      try {
        // Review: Url should be an enum
        const res = await axios.get(BASE_URL + "/api/user/userImage/" + id);
        setUserImage(res as any);
      } catch (err) {
        console.log(err);
      }
    };
    getImage();
  }, []);
  useEffect(() => {
    // Review: Extract these functions into a service and import into component. The React component should only have pres
    const getConversations = async () => {
      try {
        const res = await axios.get(
          // Review: Url should be an enum
          BASE_URL + "/api/message/Conversation/" + id
        );
        setConversations(res as any);
      } catch (err) {
        console.log(err);
      }
    };
    getConversations();
  }, []);
  useEffect(() => {
    const getMessages = async () => {
    };
    getMessages();
  }, [currentChat]);

  useEffect(() => {
    const getConversations = async () => {
    };
    getConversations();
```



5. Self-check on best practices for security

- List major assets you are protecting
  * Protecting the endpoint for backend services from authorization
  * Protecting Page from Unauthorized users.
- List major threats for each asset above
  * No major threats for the asset after the implementation of the high security "auth" function.
- For each asset say how you are protecting it (1-2 lines of text per each)
  * For securing the (server) endpoint of backend services we have made the "auth" function that will protect our endpoint.

The purpose of this auth function will check first that the user is login or not if the user will log in with the right credentials it will pass the token through headers and moreover APIs need tokens from the header to access the resources from the database.

* For protecting client pages we use the private routes for securing the pages.

We have created the "useAuth" function on the client-side it will communicate from the backend and check the token is present in the header or not,

if the token is present it allows the user to route to the other pages but if the token is not present it will redirect to the login page where you have to log in first for the token.

- Confirm that you encrypt PW in the DB
  * We use JWT (JSON Web Token) for the encryption of passwords for login.
  JWT will convert simple strings to hash passwords and also we use the Bcrypt library for comparing hash passwords to simple input passwords and compare the password is matched or not.
- Confirm Input data validation (list what is being validated and what code you used) – we request you validate search bar input for up to 40 alphanumeric characters;

  * From Client side we use validation on input field of "maxlength" attribute
  we use the code: <input type="text" id="search" name="search" maxlength="10"/>

6. Self-check: Adherence to original Non-functional specs

1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team but all tools and servers have to be approved by class CTO). **Done**
2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly. **Done**
   on the two latest versions of two major browsers
3. All or selected application functions must render well on mobile devices. **Pending**
4. Data shall be stored in the database on the team's deployment server. **Done**
5. No more than 50 concurrent users shall be accessing the application at any time **Pending**
6. Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users **Pending**
7. The language used shall be English (no localization needed). **Done**
8. Application shall be very easy to use and intuitive **Done**
9. Application should follow established architecture patterns. **Done**
10. Application code and its repository shall be easy to inspect and maintain. **Done**
11. Google analytics shall be used (optional).
12. No e-mail clients shall be allowed. Interested users can only message to sellers via in-site messaging. **Done**
13. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI. **Done**
14. Site security: basic best practices shall be applied (as covered in the class) for main data items. **Done**
15. Media formats shall be standard as used in the market today. **Done.**
16. Modern SE processes and practices shall be used as specified in the class, including. collaborative and continuous SW development **Done.**
17. The application UI (WWW and mobile) shall prominently display the following exact text on all pages *"Fulda University Software Engineering Fall 2020. For Demonstration Only"* at the top of the WWW page. (Important to not confuse this with a real application). **Done**