Technical Interaction Design (Autumn 2022)

KSTEIND1KU

Final Report
Chat Application

William Frost Thomsen, wilt@itu.dk
Fabian August Clement Harlang, faha@itu.dk
Georgios Rounis, gero@itu.dk

IT University of Copenhagen, Denmark
Teacher: Bjørn Hjorth Westh

IT UNIVERSITY OF COPENHAGEN

Contents

1 Introduction

With the advent of technological improvements, web applications are now used on a large scale. For the purpose of this course, we are going to focus on web chat applications, and in particular, we are going to develop a chat application to be used by students and Teaching assistants in ITU. Due to the fact that the current communication platforms for ITU students can provoke a lot of frustration among students that want to receive help as soon as possible and TAs that want to help students with their questions but at the same time do it whenever they have available time, we identified the need to develop a chat application that addresses those expectations and make communication easier. The implementation part of the application will make use of markup language HTML, CSS styling, and client-side programming languages using JavaScript React libraries and the technical solution will be described in a later section. Prior to the implementation specifics, we will describe in the following sections the design steps we followed to build a chat application that both covers the need of ITU students and creates a seamless experience for the end user.

2 Empathy research

2.1 User group and target group

Understanding the use context is important as the success of the software highly depends on how well it fits its environment and the use context (Parnas, 1999). As we build the project on the chat application the focus will be to provide a channel with a user interface experience where the users can communicate with each other and respond or react to the messages whenever necessary in real-time. After doing a brainstorming session to pick the user group, we agreed on the students and Teaching Assistants at the IT University of Copenhagen. We proceeded with this selection because there are two different user groups. On the first hand, we have students who are seeking help in their selected courses, and on the other hand, we have TAs who wanna help them as quickly as possible whenever they have the time to do so.

2.2 Empathy research

Empathy research helps undertake research to improve your comprehension of your users (Bjørn H. Westh, slides). We have chosen to conduct interviews with our target users defined above to understand their needs and find out which features they consider the most desired ones for our chat application. It is very essential before interviewing to sufficiently prepare to ensure a smooth run and cover all the important areas in one sitting. For that reason, before the interview date, we followed the below steps to plan our interview:

1. Brainstorm the interview questions with the members of the team project
2. Group the potential questions into areas and decide on a smooth flow
3. Polish and refine our questions as well as remove the repeated ones

2.3 Conduct empathy research

Conduct empathy research We interviewed 3 students and 2 Teaching assistants from ITU to gain useful information for ITU chat application. The interview questions asked are listed below:

Question 1: How do you usually contact other students from ITU? (Applicable for both students and TAs)

Question 2: How do you usually contact your TAs when you have questions? (Applicable for students)

Question 3: What is your opinion on the current ways to communicate with your TAs? Do you find it easy to get answers from them as fast you need them? (Applicable for students)

Question 4: What is your opinion on the current ways to communicate with students and answer their questions? Do you find it easy to reply them quickly? (Applicable for TAs)

Question 5: What do you think about a central ITU chat application that students will be able to view all available TAs per subscribed course to answer their questions? (Applicable for both students and TAs)

2.4 Findings from our empathy research

After the interviews with both students from ITU and TAs, we managed to gain useful information for our target users.

Regarding the students, according to their answers, they mostly use either Microsoft Teams or Facebook messenger to contact other students and their TAs. While it is relatively easy to find them using these apps, it is often quite hard to get fast responses to their questions, and therefore they need to spend time find for other TAs that are available at the moment to help. This makes them spend a lot of time texting different people till someone is ready to help. Students were quite positive about the idea of an application that will indicate the availability of every TA to help per subscribed course.

As for the TAs, they also find it quite frustrating the part to answer students' questions fast only when they have available time and very often they are stopping what they do at the moment to reply. Lastly, both TAs were excited with the idea of an application that they will be able to set their status as available or busy based on whether they can help at the moment or not.

3 Problem Domain

3.1 Personas

Persona one

Name: Ben Hopkins

At a glance
age: 28
location: Northwest
life stage: In relationship
occupancy: study at IT-University of Copenhagen (1st year Master)

Motivators:
Find time to spend with his friends, doing good at school and listen to his favorite 70's Rock band Black Sabbath

Behaviors:
Daily life concerns - Spend a lot of time studying, attending the lectures, trying to not falling behind in semester courses and trying to find balance between ITU and social life.

Volunteers - At Analog cafe twice per week.

Spends time with his own Rock band - He plays keyboard and participates in rehearsal with his band before a concert.

Needs
- Social Connection
- Spend time with his beloved ones
- Take his university degree in a timely manner

Persona two

Name: Maira Lowe

At a glance
age: 26
location: Southeast
life stage: in a relationship
occupancy: Pursuing her master in IT-University of Copenhagen (2nd year, being a teacher assistant in Introductory Programming)

Motivators:
Being helpful and appreciated for it without pushing her own needs. In addition, she would like to gain knowledge through explaining complex topics in a way that is comprehensible to inexperienced students.

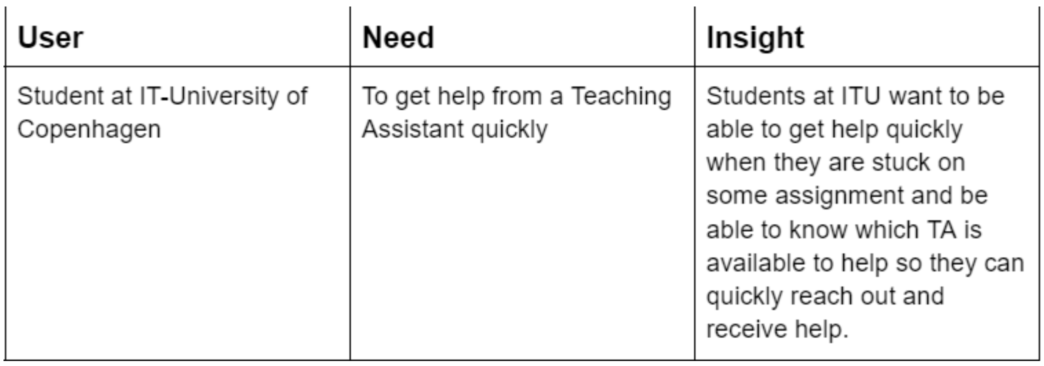
Behaviors:
Daily life concerns - Spend a lot of time studying, likes exercising and plays handball with her friends.

Volunteers - At Scrollbar every Friday.
Spends time playing League of Legends online.

Needs
- Follow his favorite football team every Sunday
- Spend time with her boyfriend
- Take his university degree next summer

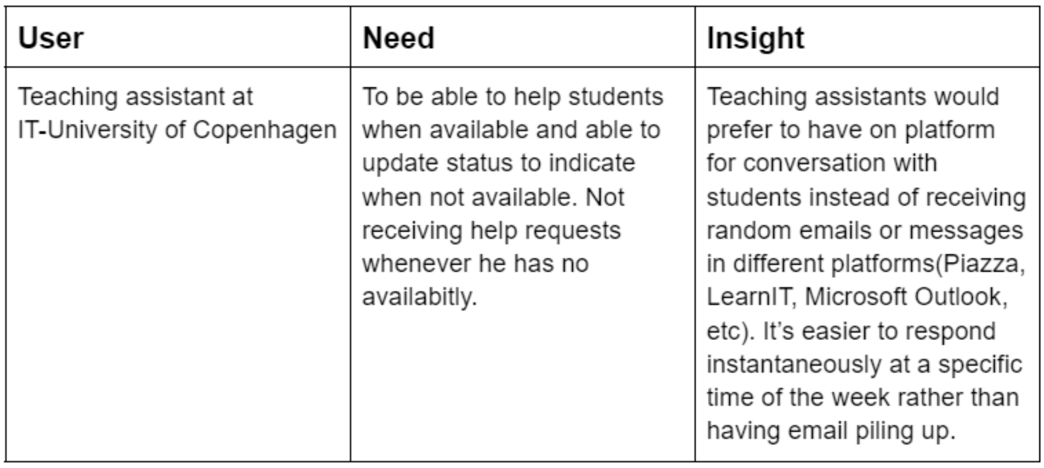
3.2 POVs

Student POV



User	Need	Insight
Student at IT-University of Copenhagen	To get help from a Teaching Assistant quickly	Students at ITU want to be able to get help quickly when they are stuck on some assignment and be able to know which TA is available to help so they can quickly reach out and receive help.

TAs POV



User	Need	Insight
Teaching assistant at IT-University of Copenhagen	To be able to help students when available and able to update status to indicate when not available. Not receiving help requests whenever he has no availabitly.	Teaching assistants would prefer to have on platform for conversation with students instead of receiving random emails or messages in different platforms(Piazza, LearnIT, Microsoft Outlook, etc). It's easier to respond instantaneously at a specific time of the week rather than having email piling up.

3.3 "How might we"

- How might we improve the communication between students and Teachers Assitants (TAs)?
- How might we avoid delayed and inefficient communitacion via e-mail?
- How might we create an option for students at ITU to get instantaneous help with an exercise/assigment from a TAs?
- How might we ensure only avaialble TAs will be contacted by students?
- How might we create a chat applications with different rooms each representing a specific course?
- How might we enable functionalities of attaching documents and/or sharing screens for easier code troubleshooting or feedback?

4 Ideation Prototype

4.1 Ideation

After the definition of our problem domain in the previous step, it is now time to discover possible solutions to the existing problem: Students at ITU need a smart and flexible way to communicate and get help quickly only from Teaching assistants that are available at the moment to offer help avoiding massive emails and spamming messages. In the Ideation phase, we are going to discover and discuss the ideas which will be finally converted into solutions for our problem.

4.2 Methods - Generation

Our selected method to generate our ideas was a Brainstorming session among the three members of our teams where we set a 30-minute time limit and each one of us noted down their personal ideas. During the session, we tried to ensure that members stayed on topic and continued generating solutions around the core problem statement without going beyond the scope of the discussion. The most constructive part of this method was the building of each other's ideas. Hence, an immature idea generated by one of us later evolved into a solid one. The reason we have chosen the Brainstorming session against other methods is that we use this method in our daily life and we are more familiar with it when it comes to finding solutions. In addition, the open discussion, and the easy building on others' ideas is indeed very useful.

4.3 Methods - Selection

After we list the ideas defined during the generation phase, it is time to select those that are considered by all the team members as the most critical ones. In order to vote for our ideas, we used the Dot Voting method. All the ideas generated in the ideation session were written down on individual Post-its and each member could give 4 votes in total simply using a marker to make a dot on the ideas they like. The reason why we have chosen this method over the others is because of its simplicity.

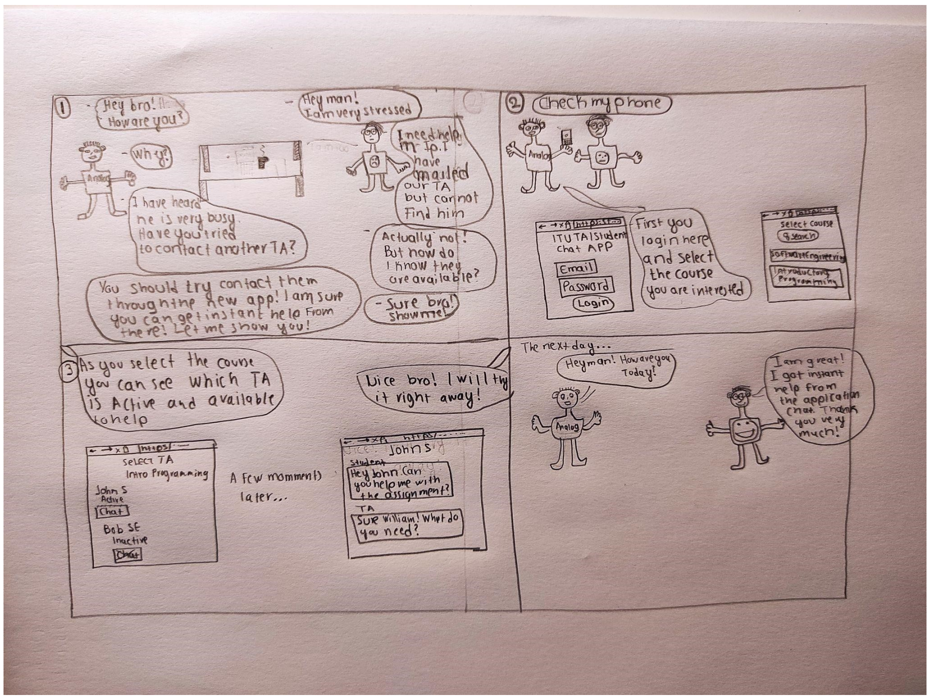
4.4 Our selected ideas

The ideas we finally voted for as the best to fit our problem domain are listed below:

- A chat application to enable direct communication between students and teaching assistants will give the chance to ITU students to get help quickly from their TAs.
- Each teacher assistant should have a status in the chat application indicating whether they are available or not.
- Students should be able to chat only with teacher assistants for the courses that they have registered for.
- Users should be able to create groups in order to chat with multiple users.
- Students before starting the chat should be able to select the severity of their ticket through a 4-scale with the choices Low/ Medium/ High/ Very High.

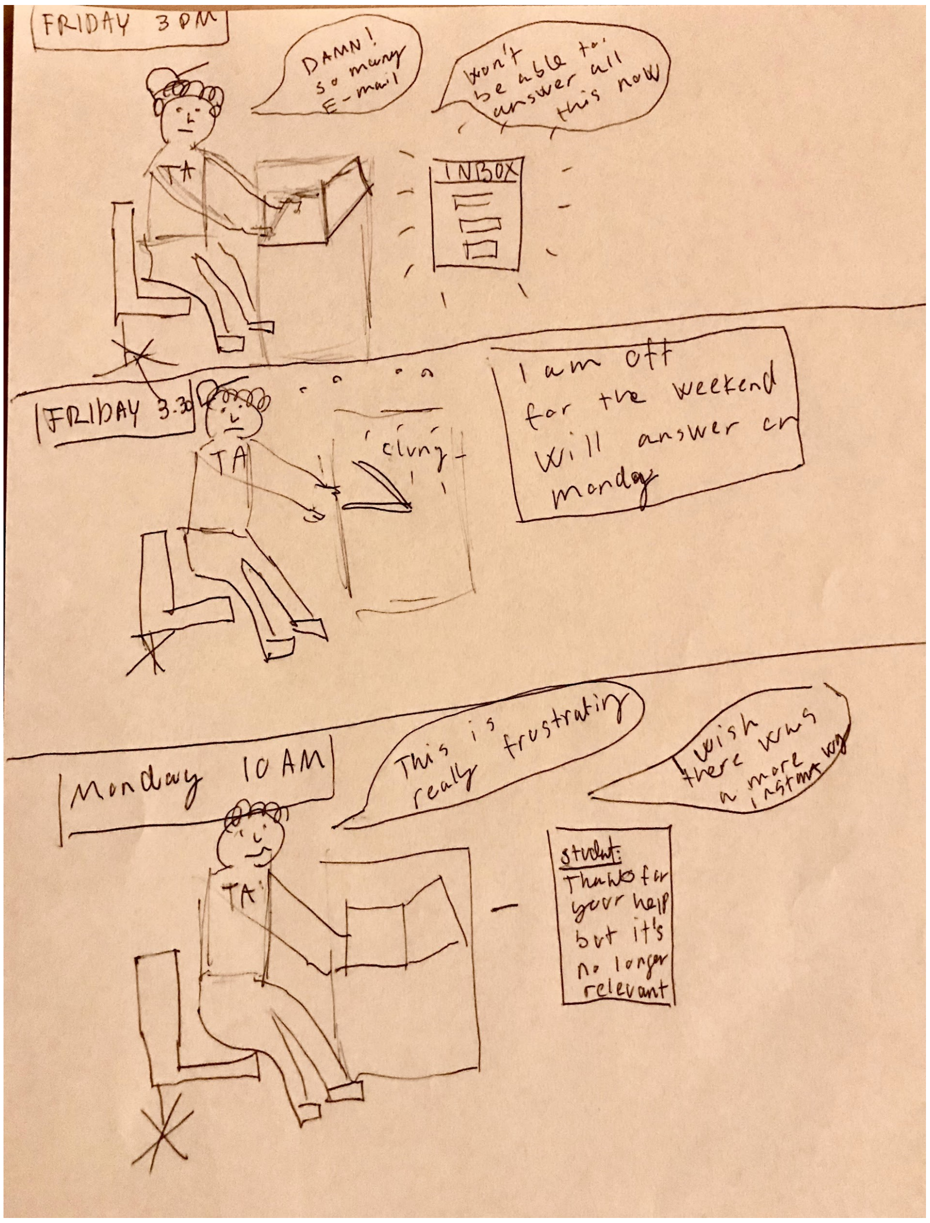
4.5 Prototyping

We decided to use sketches and wireframes for our prototyping.



1
Hey bro! How are you?
Hey man! I am very stressed
Why?
I need help in Ip. I have mailed our TA but cannot find him
I have heard he is very busy. Have you tried to contact another TA?
Actually not! But how do I know they are available?
You should try contact them throughthe new app! I am sure you can get instant help from there! Let me show you!
Sure bro! Show me!
Analog
2
Check my phone
Analog
ITU TA|Student Chat APP
Email
Password
Login
First you login here and select the course you are interested
Select course
Search
Software Engineering
Introductory Programming
3
As you select the course you can see which TA is Active and available to help
Select TA
Intro Programming
John S
Active
Chat
Bob SE
Inactive
Chat
A few momments later...
Nice bro! I will try it right away!
John S
Student
Hey John Can you help me with the assignment?
TA
Sure William! What do you need?
The next day...
Hey man! How are you Today?
Analog
I am great! I got instant help from the application chat. Thank you very much!

Figure 1: Sketch 1. Student


FRIDAY 3PM
DAMN! so many E-mail
won't be able to answer all this now
INBOX
TA
FRIDAY 3:30
TA
'clung'
I am off for the weekend will answer on monday
Monday 10 AM
This is really frustrating
wish there was a more instant way
TA
student: Thanks for your help but it's no longer relevant

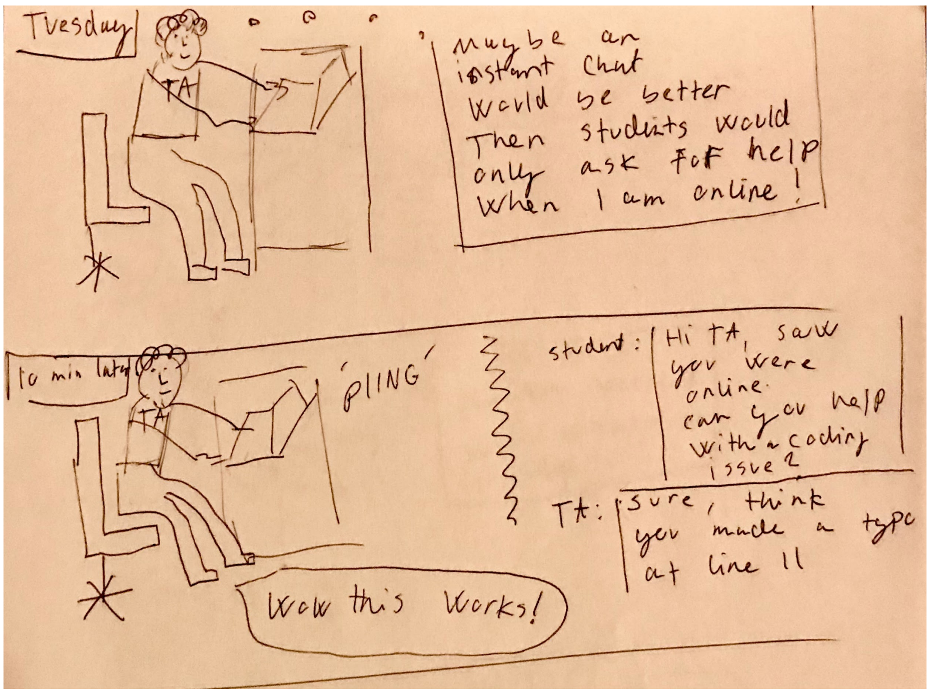

Tuesday
TA
Maybe an instant chat would be better Then students would only ask for help when I am online!
10 min later
TA
'PLING'
student: Hi TA, saw you were online. can you help with a coding issue?
TA: Sure, think you made a typo at line 11
Wow this works!

Figure 2: Sketch 2. TA

We use wireframes as it offers a highly flexible solution for displaying lo-fi versions of the product which we will develop and how it will be displayed for the user. As we are still early in the process and are still in the process of designing the product to fit the problem domain etc, wireframing gives us the ability to easily display some initial idea about how the interface and product might look like, but simultaneously gives us the opportunity to easily change the design to comply with the user feedback or if our user functionality will change later on. We decided, unlike the sketches, to do the wireframe with the online-tool Balsamiq as we expect to update it as more functionalities and sections come into account.

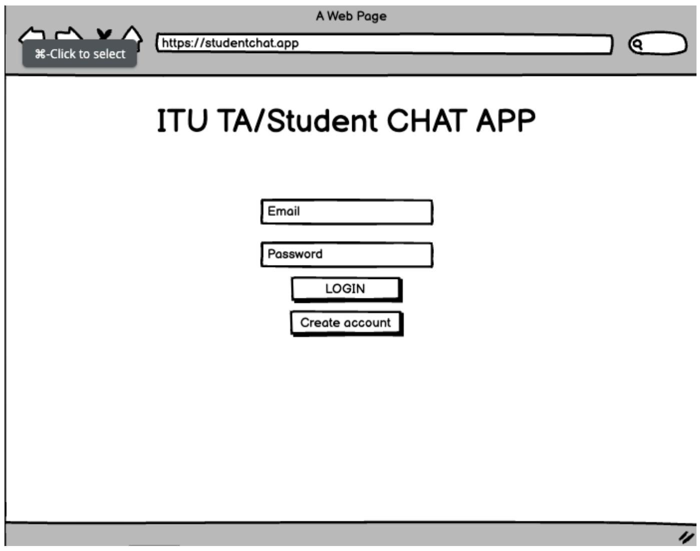


Figure 3: Login / Signup page

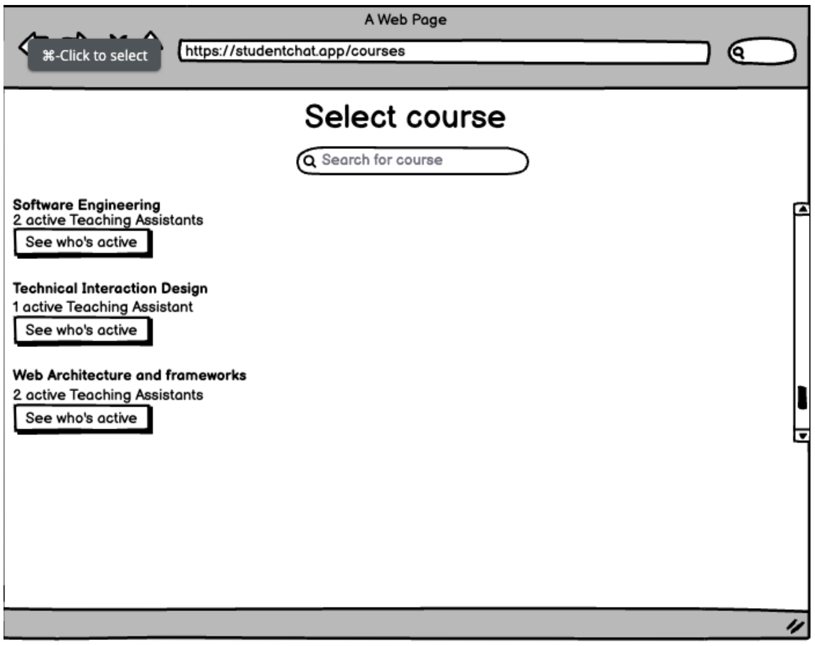

A Web Page
⌘-Click to select
https://studentchat.app/courses
Select course
Search for course
Software Engineering
2 active Teaching Assistants
See who's active
Technical Interaction Design
1 active Teaching Assistant
See who's active
Web Architecture and frameworks
2 active Teaching Assistants
See who's active

Figure 4: Course Overview

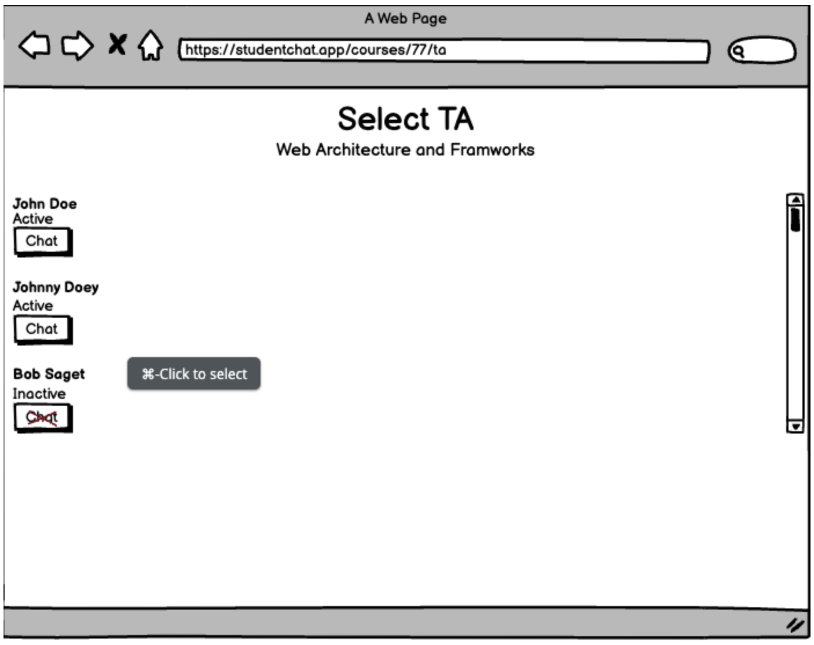
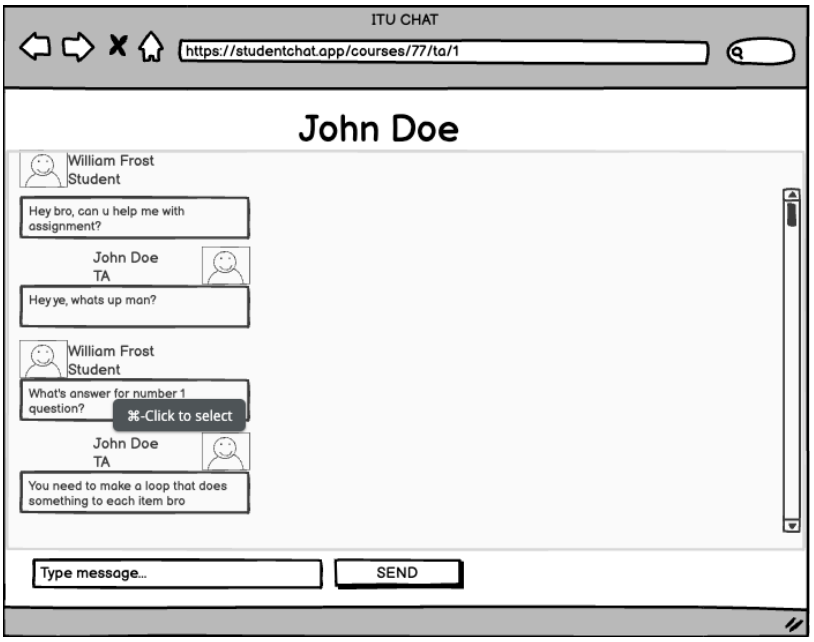


Figure 5: TA overview and availability



Hey bro, can u help me with assignment?

Hey ye, whats up man?

What's answer for number 1 question?

You need to make a loop that does something to each item bro

Figure 6: Chat window

5 Usability Test

5.1 Usability Test of Low-Fidelity Prototype

According to Jacob Nielsen, Thinking Aloud Testing is considered to be the most valuable usability engineering method because it is cheap, flexible, easy to learn, and unlike quantitative usability studies where the slightest mistake might have misleading results, even with poorly run research we can get insightful findings. For all the above reasons, we decided to proceed with Thinking Aloud Testing of our Low-Fidelity Prototype using the wireframes we created last week.

In order to conduct Thinking Aloud Testing, we followed the below three steps:

- Recruit representative users
- Prepare a set of tasks
- Conduct Think Aloud test

Recruit Representative Users
We managed to recruit four users in total- three Software Design students and one Teaching Assistant.

Our defined set of tasks
Using the wireframes we created last week, we decided on some tasks for our testing users.

Task 1:
Description: Using the wireframe from Figure 1, the users need to answer which is the expected result of performing the Login and Create Account actions.

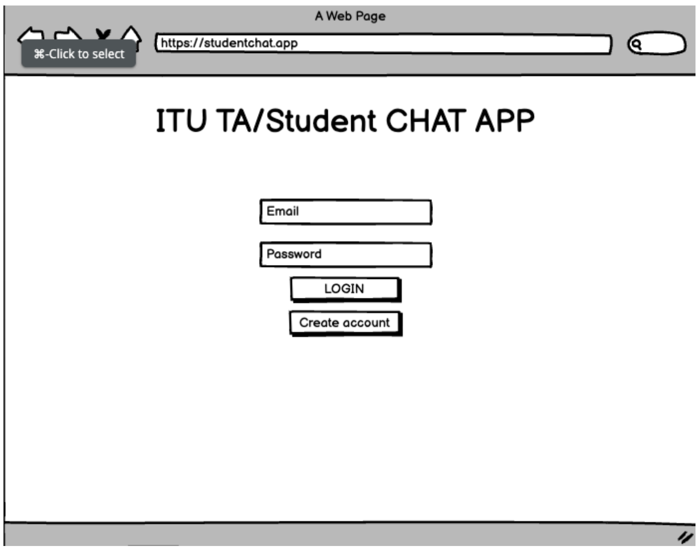
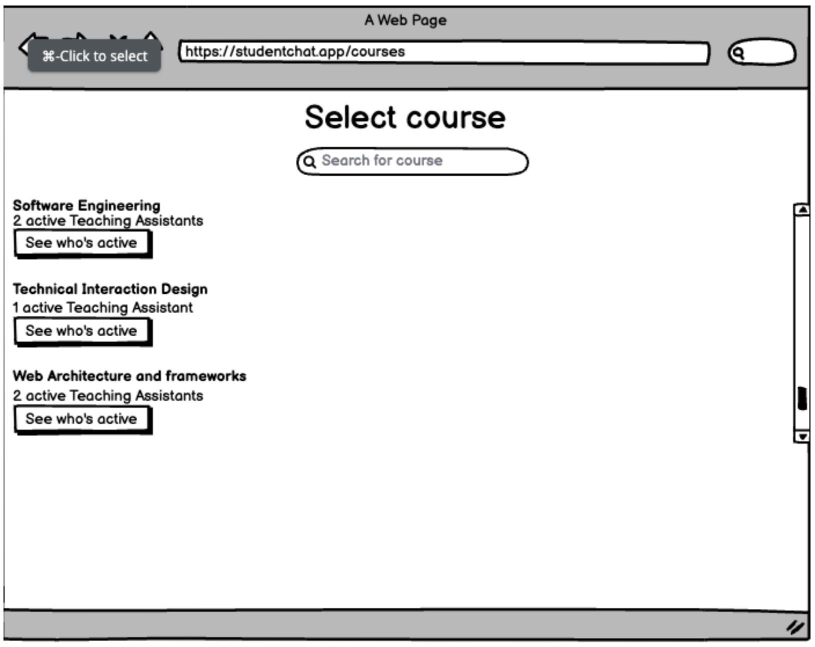


Figure 7: Login / Signup page

Task 2:

Description: Using the wireframe from Figure 2, the users should decide which is the expected result of checking the available courses and see which TA is available.



Select course

Software Engineering
2 active Teaching Assistants

Technical Interaction Design
1 active Teaching Assistant

Web Architecture and frameworks
2 active Teaching Assistants

Figure 8: Course Overview

Task 3:
Description: Using the wireframe from figure 3, the user should decide which is the expected behavior if he selects to chat with an active or inactive user.

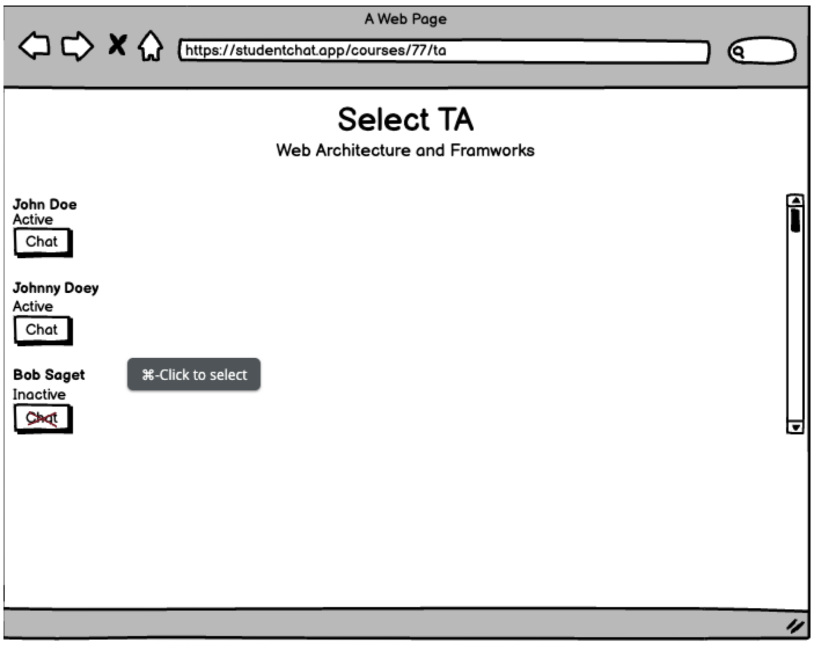


Figure 9: TA overview and availability

Task 4:
When the user selects to study with an active TA should be redirected to a page similar to Figure 4. Which is the expected behavior when the user types a message and clicks send? What will happen if the other person responds back?

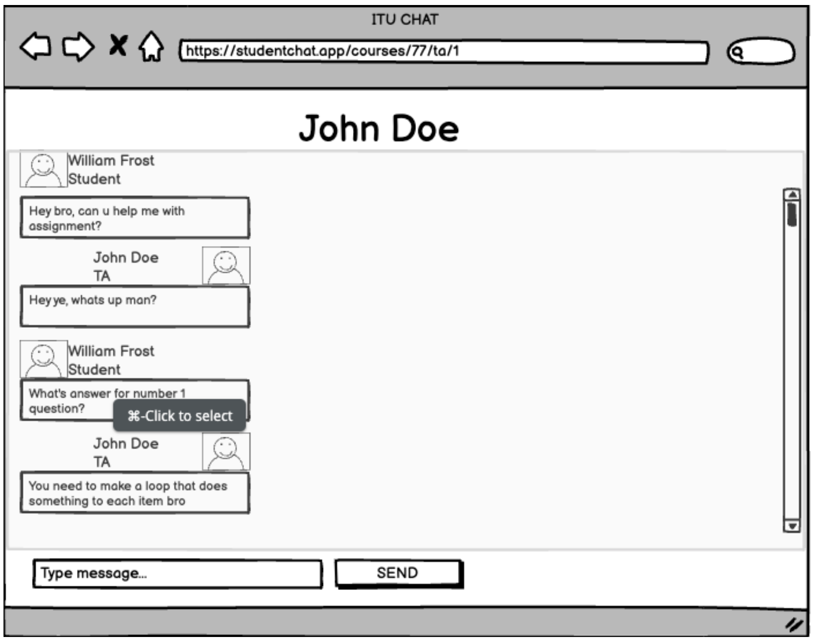

ITU CHAT
https://studentchat.app/courses/77/ta/1
John Doe
William Frost
Student
Hey bro, can u help me with assignment?
John Doe
TA
Hey ye, whats up man?
William Frost
Student
What's answer for number 1 question?
⌘-Click to select
John Doe
TA
You need to make a loop that does something to each item bro
Type message...
SEND

Figure 10: Chat window

5.2 Conduct Think Aloud test

On Thursday, 29th of September we conducted the Think Aloud test with our users, and both findings and feedback we received are presented below:

- **#Task 1:** Users guessed most of the time correctly 75% about the expected behavior of our login/signup prototype 25% expected the redirect to ITU to authorize their account). This means that our design at this stage ensures intuitiveness.
- **#Task 2:** All four users correctly expected the courses in which they are currently enrolled to be displayed in this view. It made sense to them that available TAs and their availability status were listed below the course name.
- **#Task 3:** All users correctly guessed that clicking on an online TA would redirect them to the chat window (Figure 4). 75% of the users expected to be redirected to the TAs chat window regardless of the availability status. The remaining 25% of the users expected the application to disable redirection to the chat window of the offline TA and display an error message.
- **#Task 4:**All users expected the message to be sent and delivered and have similar behavior to other well-known chat applications. Half of the users expected the message to the offline TA to be sent anyway. This is not entirely correct, however. The message will not behave like a normal instant chat message, but would instead open a new "ticket" with a status unsolved. If some other TA then solves the problem in the meantime he/she can close the ticket and avoid the offline TA to answer already solved issues. This is a design decision to avoid excessive work for the TAs and ensure more efficient communication for the students. Only 25% of the users expected this behavior. The remaining 25% expected the message not to be sent.

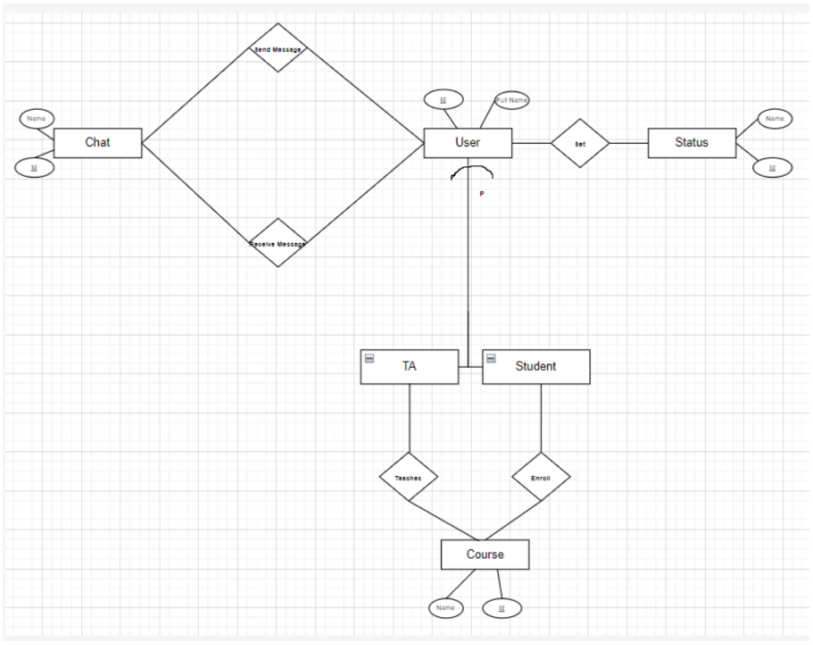
- We missed a wireframe showing the ability of a TA to set the status to either available or unavailable. We furthermore weren't able to provide a prototype (wireframe) of the "ticket" section.
- The feedback overall was very good and useful. All our users agreed an instant chat application is a nice idea that would make both lives of TAs and students easier

6 Hi-Fi Prototype

Please find below the link for our Figma: `https://www.figma.com/file/nD2yOOqgh8pfd6ZQus4qCz/Group-8---Chat-Application?node-id=3%3A3`

7 Data Model

Below you can find our ER Diagram:



8 Technical Implementation

For the implementation we decided to use the frameworks and programming languages taught in the Technical Interaction Design course i.e React JS, HTML, CSS for the front end and the low-code service back4app for the back end. We did the version control and collaboration using a GitHub repository:

`https://github.itu.dk/faha/chatITU/tree/master/client(back4app)`

8.1 Sprint 1

Initially, we implemented a simple navigation component in HTML and CSS and Fontawesome icons. We then installed and set up the react-router-dom in the App.js file to make the link render the proper components using the Link component. This worked as expected with only minimum effort. We then started to implement the generic pages i.e home page, signup, login.

For the homepage, we used CSS flexbox to make two big "columns". One with a welcome message with a button to 'get started' and another with a GIF animation showing a chat window.

The signup page is simply made up by an HTML form with some input and a submit button. The input captures the data with the onChange event and set the state of the const username, password using react useState.

```
<label for="email">Username</label>
 <input
   type="text"
   placeholder="email"
   name="email"
   onChange={(e) => setUsername(e.target.value)}
   value={username}
```

We then use the back4app parser to create a new user in the database:

```
const newUser = new Parse.User();
newUser.set("username", username);
newUser.set("password", password);
newUser.set("teachingAssistant", teachingAssistant);

try {
  // Since the signUp method returns a Promise, we need to call it using await
  const createdUser = await newUser.signUp();
  alert(
    `Success! User ${createdUser.getUsername()} was successfully created!`
  );
```

The login page has a similar implementation but this time we compare the user input and parse it with the 'logIn' function to check if a matching record was found in the database. If a match was successfully found the user is redirected to the chat otherwise an error is thrown.

```
const loggedInUser = await Parse.User.logIn(usernameValue, passwordValue);
```

The forms are styled in a similar fashion to our Figma project. We decided however to not center the forms and instead keep them to one side and have some graphics on the other side. Sign up form and login are similar in design but flipped so that the login form is on the left side and the opposite for the sign up.

Our process was straightforward as we have done similar implementations before, but we did have to change something as we initially mistakenly utilized the CSS framework Bootstrap not knowing this wasn't allowed. We manage however to refactor and got a similar design with custom HTML elements and CSS flexbox etc.

8.2 Sprint 2

In this sprint, we did the further setup of our back4app. In our database, we created 4 classes: role, session, user, and chat. This is more or less aligned with our ER diagram except for the fact we didn't restrict chatting to 'courses' that the user must be enrolled in beforehand. Instead, we allowed users and TA to create new custom chats with any subject and allowed them to be either public or for specific users added by the chat creator. This implementation was easier to do and also provided more freedom for the users.

As explained above we create users using the signup form and look for matching records in the login form. We didn't do all CRUD operations at this point, but could easily be done in a later version e.g. if we create functionality for the users to update their passwords or delete accounts.

In this sprint we started implementing a 'chat dashboard' (see wChatSetup.js for reference) this page/component should aid the user to create or join a new chat and give an overview of previous/ongoing chat threads. For better usability, we've added sections indicated with different font awesome icons. Registered users are color-coded with either red or green circles indicating the status of the user. For each user the "add user" button is present. Clicking it will return the message 'user added' with a green check mark. The status can be changed from the drop-down menu in the navigation.

8.3 Sprint 3

In this sprint, we implemented the chat window i.e the main function of the app. Messages are fetched from back4app using the parser (please see component wChatWindow.js for more details). The chat window is a large div element with the css property 'overflow' set to 'scroll' to make all messages visible by scrolling. We created a function "autoScroll" that automatically scrolls down to the most recent message whenever a new message is sent. This will make the usability of the chat more intuitive as the user doesn't need to scroll manually

```
   function scrollAuto() {
  setTimeout(() => {
    let obj = document.getElementsByClassName("messages")[0];
    obj.scrollTop = obj.scrollHeight;
  }, 500);
}
```

We also added a key down handler enabling the user to send (submit the input value) on enter. This is a very default behavior of most chat applications and is something most users by intuition would expect to be possible.

```
    useEffect(() => {
  const keyDownHandler = (event) => {
    console.log("User pressed: ", event.key);

    if (event.key === "Enter" && messageInput !== "") {
      event.preventDefault(); (...)
```

Each message bubble has some color coding and border-radius and padding. We have made conditional coloring based on if the message is sent by the currently logged-in user or not. Similar behavior to Facebook messenger the outbound messages are blue while the inbound messages are grey. The messages have some padding and margin so they easily distinguish from one another. We also enable word wrap to avoid too-long lines in horizontal overflow. Below each message, we've added the sender and the timestamp with a smaller font size. We would have liked to enhance our chat window with a widget showing users currently in this ”room” similar to our idea in Figma. This implementation has however been pushed to the next sprint. We also discussed options of attachment and screen sharing as these would be convenient for especially coding support. We agreed, however, that this was an implementation beyond the scope of the course. In this sprint, we also did some refactoring in our routing allowing the behavior to be more conditional and show menu items based on whether the user is logged in or not i.e hiding redundant pages like login and sign up for logged-in users and instead displaying chat / my profile drop-down nav items.

8.4 Collaboration

We used GitHub for version control and collaboration and had several meetings at ITU and online. We've worked on different branches and files and succeeded with that without too many merge conflicts. By the end of each sprint, we did online meetings reviewing and explaining the progress we did on the code, suggestions for improvements, and other ideas. All together we think the collaboration went well and we all believe we improved our understanding of React JS. We also agreed that back4app is a great and intuitive tool to use for building a fast backend for any application, we would however have been keen to become acquainted with something like MongoDB and MERN-stack development in general.

9 Design to App

Component 1 - Login Page:
Below you can find a screenshot of the component as was designed during our Hi-Fi prototype.

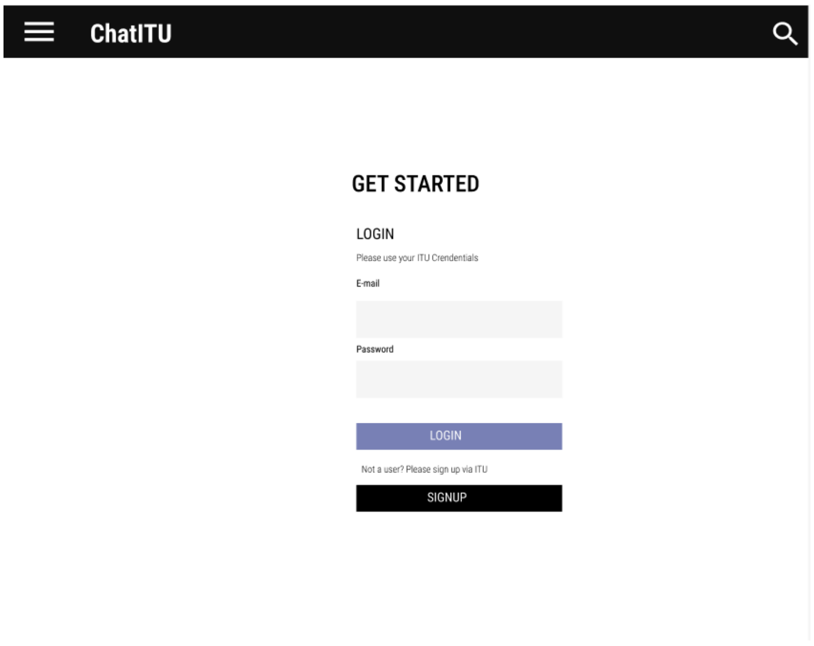

ChatITU
GET STARTED
LOGIN
Please use your ITU Crendentials
E-mail
Password
LOGIN
Not a user? Please sign up via ITU
SIGNUP

Figure 11: Login page designed in our Figma

The login page in our application looks like the following screenshot.

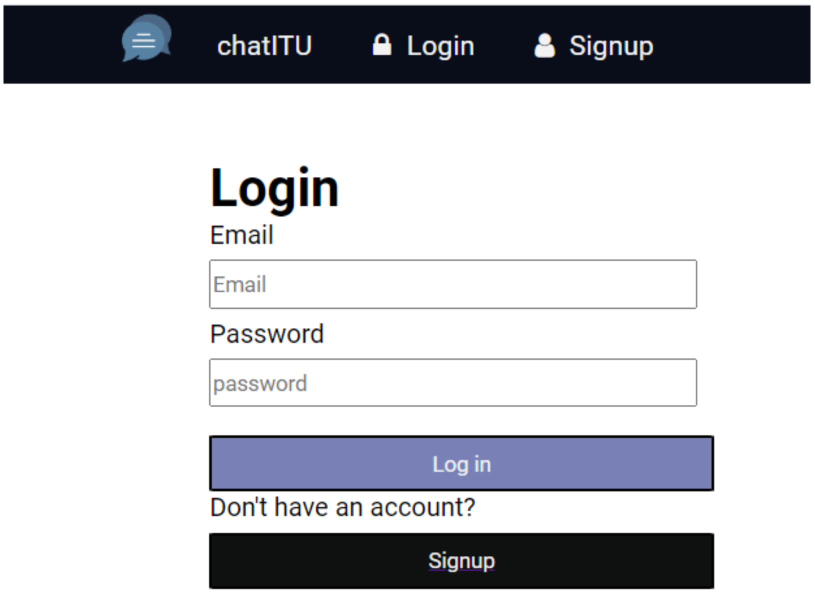
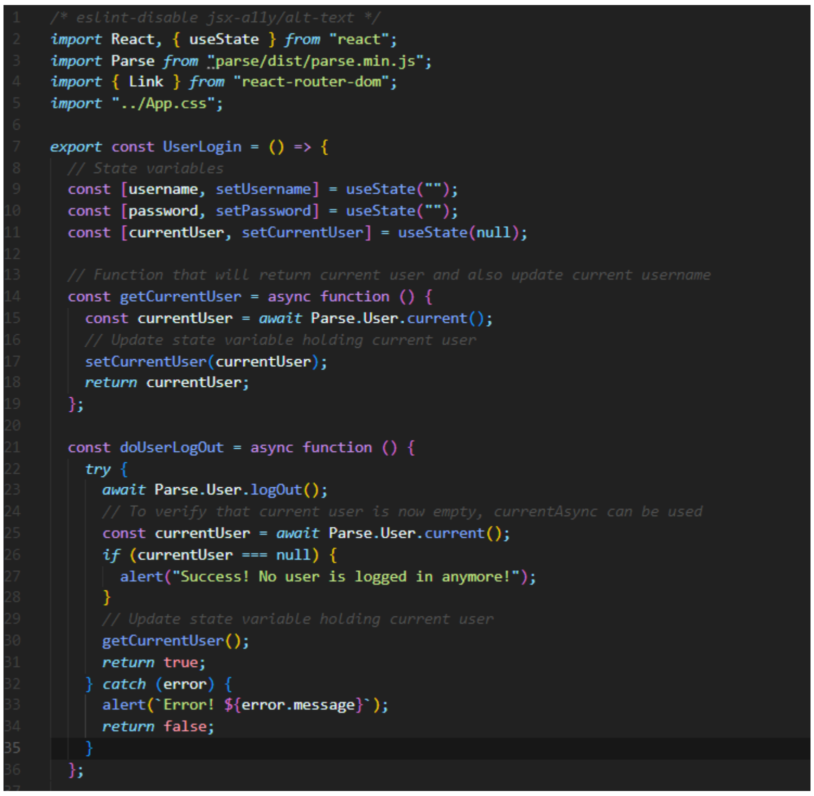


Figure 12: Login page from our application

The source code snippet to implement the Login component can be found below:

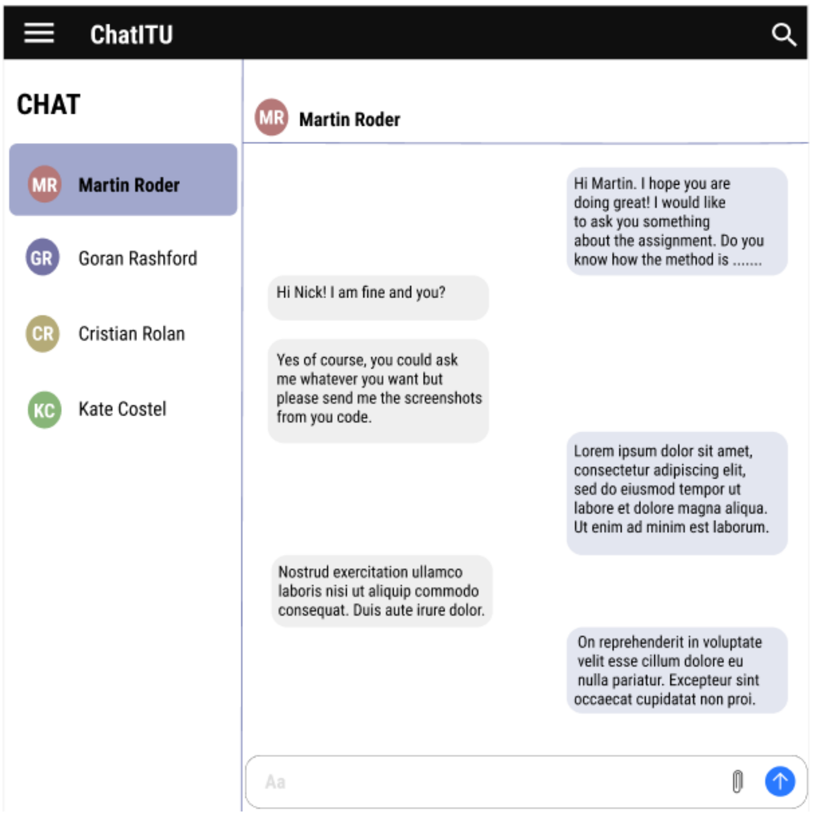


```
/* eslint-disable jsx-a11y/alt-text */
import React, { useState } from "react";
import Parse from "parse/dist/parse.min.js";
import { Link } from "react-router-dom";
import "../App.css";

export const UserLogin = () => {
  // State variables
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const [currentUser, setCurrentUser] = useState(null);

  // Function that will return current user and also update current username
  const getCurrentUser = async function () {
    const currentUser = await Parse.User.current();
    // Update state variable holding current user
    setCurrentUser(currentUser);
    return currentUser;
  };

  const doUserLogOut = async function () {
    try {
      await Parse.User.logOut();
      // To verify that current user is now empty, currentAsync can be used
      const currentUser = await Parse.User.current();
      if (currentUser === null) {
        alert("Success! No user is logged in anymore!");
      }
      // Update state variable holding current user
      getCurrentUser();
      return true;
    } catch (error) {
      alert(`Error! ${error.message}`);
      return false;
    }
  };
```

Figure 13: Source code for our Login Page

Component 2 - Chat Room:

The chat window component as was designed during our HI-Fi prototype can be found below:



Hi Martin. I hope you are doing great! I would like to ask you something about the assignment. Do you know how the method is ....

Hi Nick! I am fine and you?

Yes of course, you could ask me whatever you want but please send me the screenshots from you code.

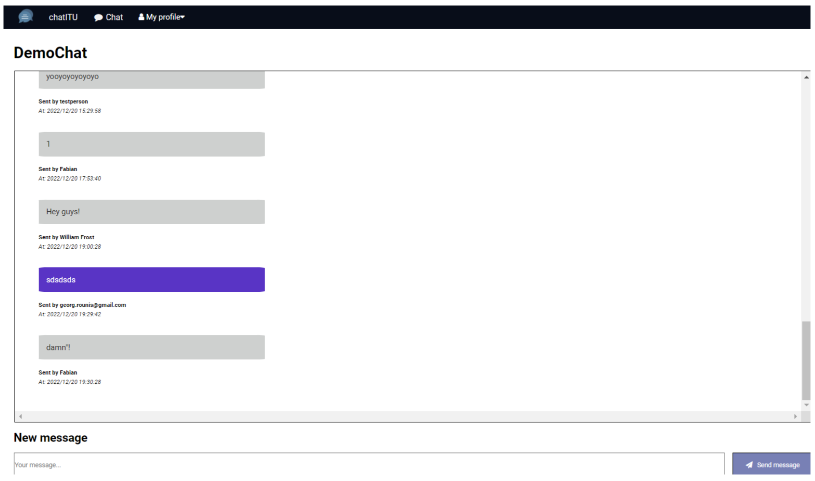
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor ut labore et dolore magna aliqua. Ut enim ad minim est laborum.

Nostrud exercitation ullamco laboris nisi ut aliquip commodo consequat. Duis aute irure dolor.

On reprehenderit in voluptate velit esse cillum dolore eu nulla pariatur. Excepteur sint occaecat cupidatat non proi.

Figure 14: Chat window designed in our Figma

The chat window was finally implemented with a different layout than the designed one but offered the same functionality.



yooyoyoyoyoyo

Sent by testperson
At: 2022/12/20 15:29:58

1

Sent by Fabian
At: 2022/12/20 17:53:40

Hey guys!

Sent by William Frost
At: 2022/12/20 19:00:28

sdsdsds

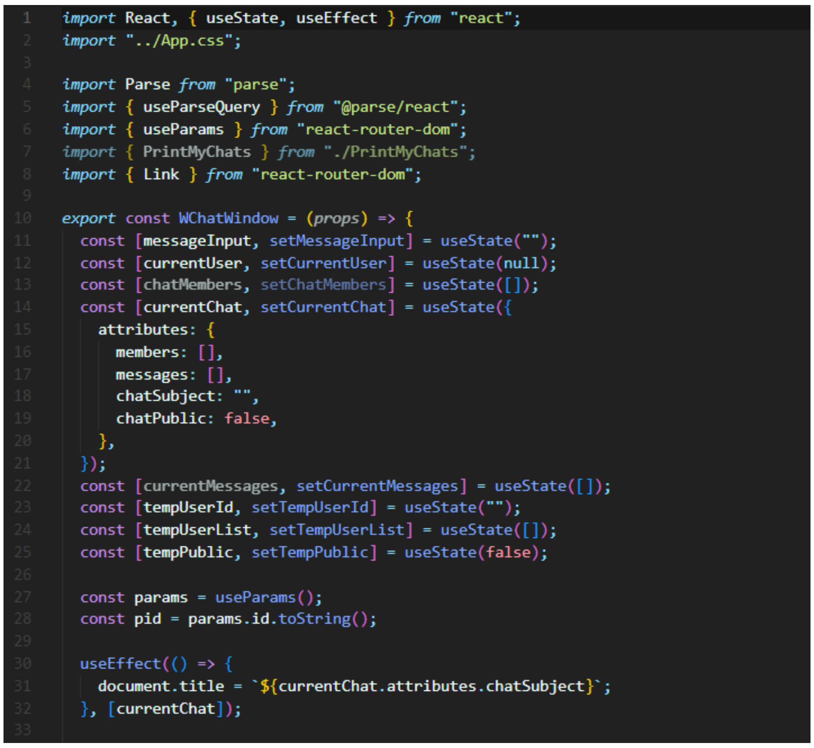
Sent by georg.rounis@gmail.com
At: 2022/12/20 19:29:42

damn"!

Sent by Fabian
At: 2022/12/20 19:30:28

Figure 15: Chat window designed in our Figma

Below we present a code snippet from our source code implementation of the chat window:



```
import React, { useState, useEffect } from "react";
import "../App.css";

import Parse from "parse";
import { useParseQuery } from "@parse/react";
import { useParams } from "react-router-dom";
import { PrintMyChats } from "./PrintMyChats";
import { Link } from "react-router-dom";

export const WChatWindow = (props) => {
  const [messageInput, setMessageInput] = useState("");
  const [currentUser, setCurrentUser] = useState(null);
  const [chatMembers, setChatMembers] = useState([]);
  const [currentChat, setCurrentChat] = useState({
    attributes: {
      members: [],
      messages: [],
      chatSubject: "",
      chatPublic: false,
    },
  });
  const [currentMessages, setCurrentMessages] = useState([]);
  const [tempUserId, setTempUserId] = useState("");
  const [tempUserList, setTempUserList] = useState([]);
  const [tempPublic, setTempPublic] = useState(false);

  const params = useParams();
  const pid = params.id.toString();

  useEffect(() => {
    document.title = `${currentChat.attributes.chatSubject}`;
  }, [currentChat]);
```

Figure 16: chat window source code snippet

10 Final reflection

Throughout the implementation phase, we realized how important is the design phase, as well as the gathering of requirements and needs that the end-user has, and our application should meet. Due to the fact that the first four design steps (Empathy research, Problem Domain, Ideation Prototype, and Usability Test) collect the requirements and specifications through the deep understanding of the end-user and lead to our HI-Fi prototype which is in essence the mockup of our implementation, we found out that it is very crucial to take it very seriously and dedicate time to execute those steps. The better requirement gathering and analysis we make, will lead to better system design avoiding redundant work, change requests, and fewer implementation changes. Hence, there will be no delay in each sprint and the application will be deployed in a timely manner. Another learning experience we had in our project and all of us agreed that initially, we didn't pay a lot of attention to it, was the Hi-Fi prototype. Fortunately, our TAs explained to us the importance of it and we tried to make it nice and distinguish the different components that we are going to need for our implementation. Due to time constraints, we didn't implement all the features we were initially planning to focus more on the mandatory ones for this assignment(Login page, chat with one user, create a group chat, etc). Finally, even though the collaboration of the group was very good, we found it a bit challenging the part to find a common time to meet and work altogether due to the strict schedule of each team member.