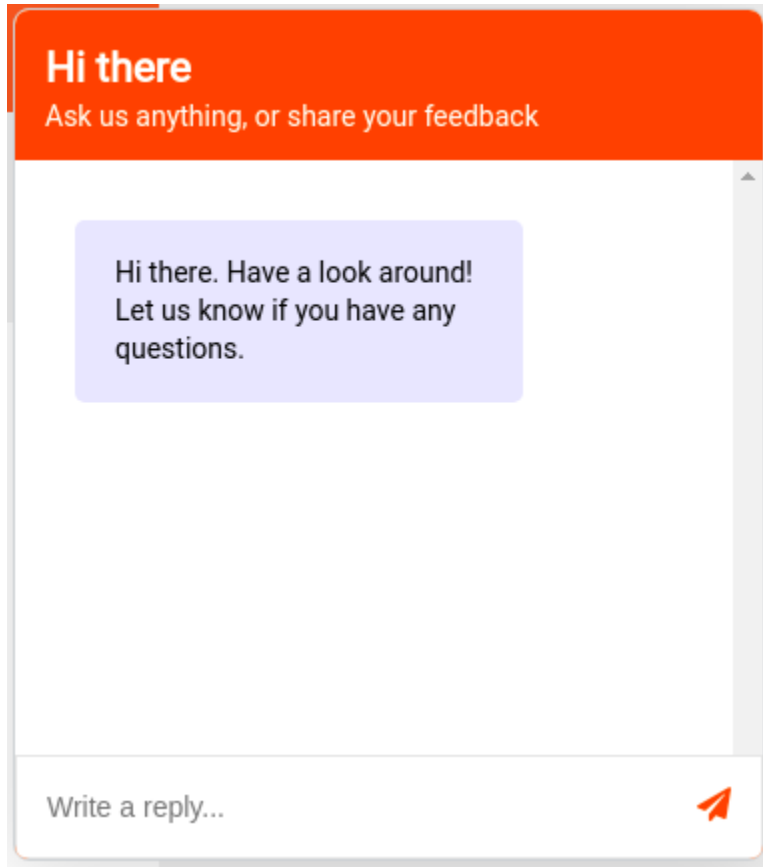
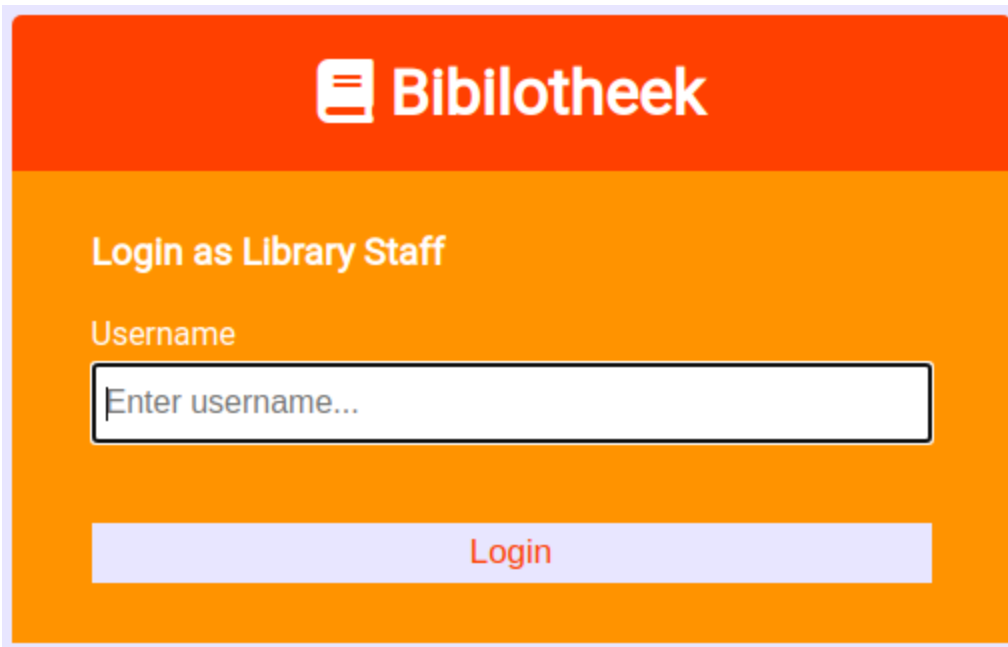


Chat Application

Features explained



Since the focus of this project is to build a frontend for chat. We have only built a prototype of how the backend would look like to demonstrate the working and integration. The staff user can login by going to <https://aqueous-journey-60276.herokuapp.com/staff/>



The image shows a login interface for 'Bibilothek'. At the top, there is a red header with the 'Bibilothek' logo and name in white. Below this, the text 'Login as Library Staff' is displayed in white. Underneath, the label 'Username' is shown in white. A white text input field with a black border contains the placeholder text 'Enter username...'. At the bottom, there is a wide, light blue button with the word 'Login' in red text.

Note that no login mechanism was provided as this component was only built to show the working.

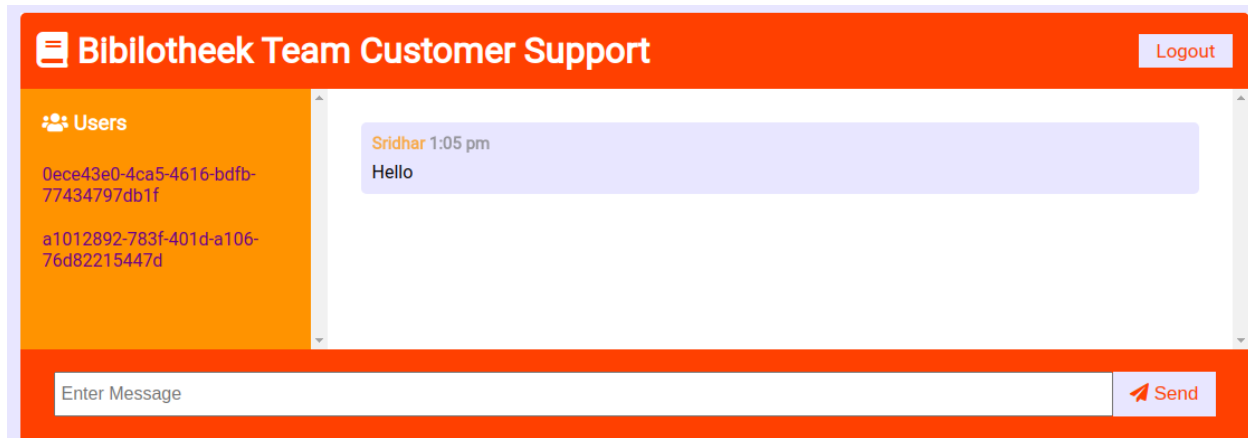
After we login to the Staff portal, we can see the following screen.



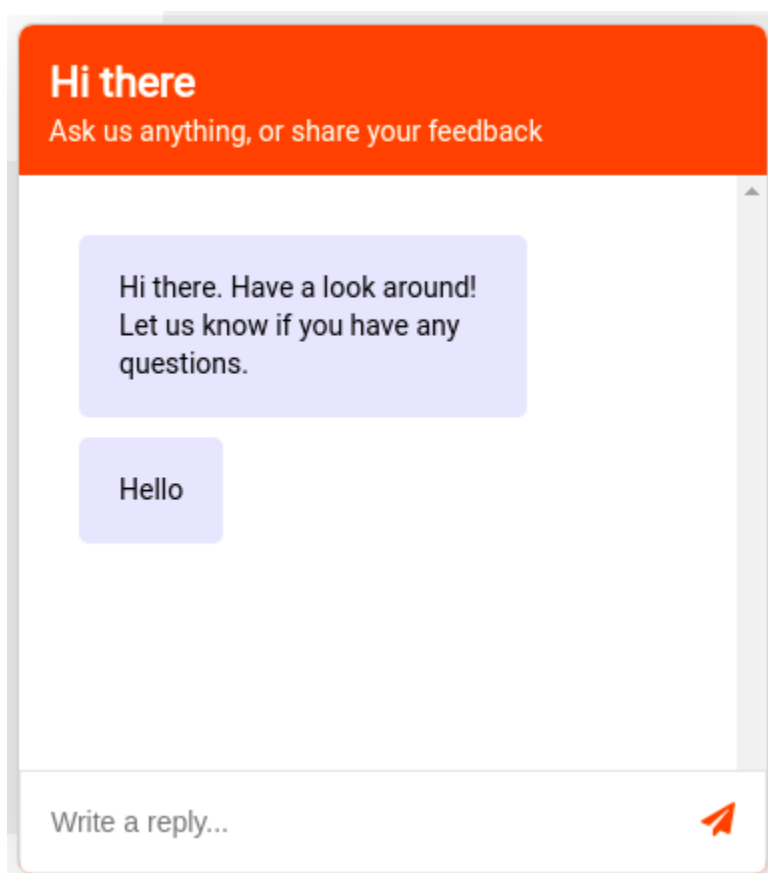
The image shows a 'Bibilothek Team Customer Support' interface. The top has a red header with the title 'Bibilothek Team Customer Support' on the left and a 'Logout' button on the right. The main area is divided into two panes. The left pane, titled 'Users' with a person icon, lists two users with their UUIDs: '0ece43e0-4ca5-4616-bdfb-77434797db1f' and 'a1012892-783f-401d-a106-76d82215447d'. The right pane is empty. At the bottom, there is a red bar containing a text input field with the placeholder 'Enter Message' and a 'Send' button with a paper plane icon.

The left-hand pane shows the users who are currently logged in. Every logged in user is assigned with a unique UUID. The staff can chat live with currently logged in users by clicking on the user.

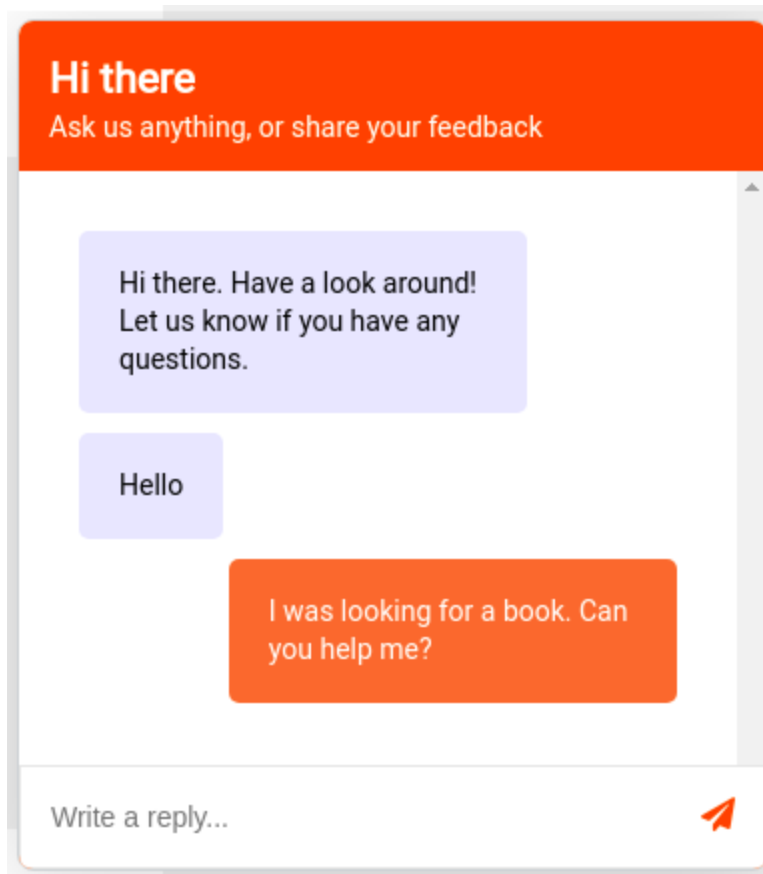
For demonstration, I click on one of the users and type a message.



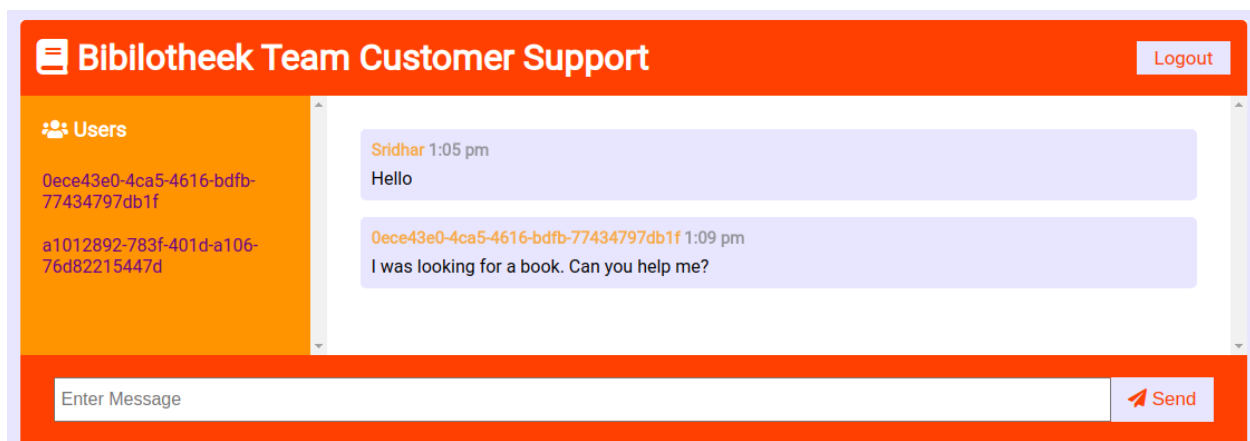
The user would instantly receive this message and can reply back. The chat between the staff and the user is recorded in a MySQL database.



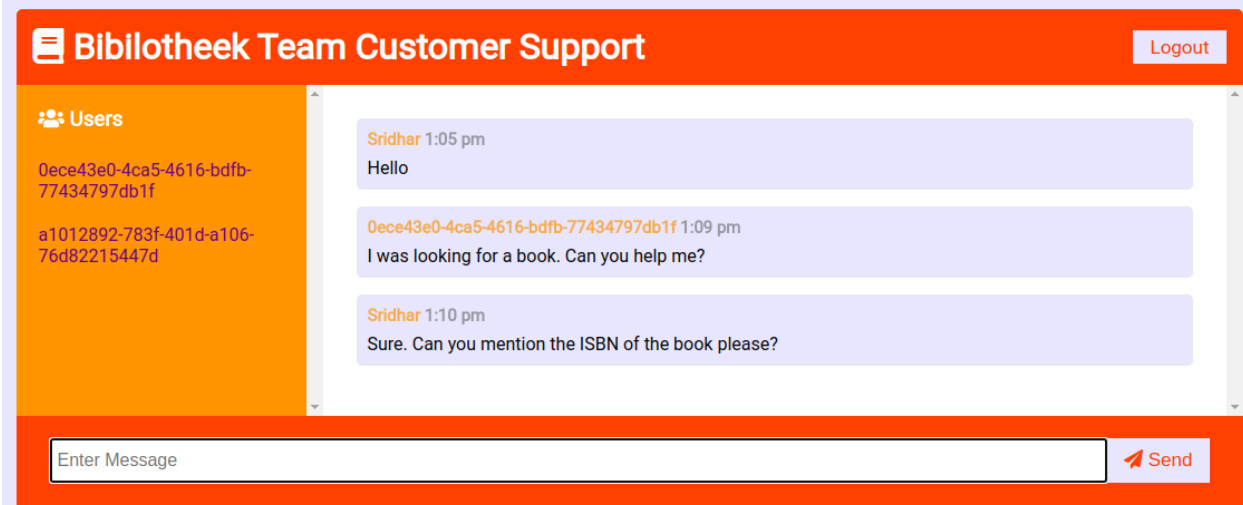
The user responds to this and enters his query.



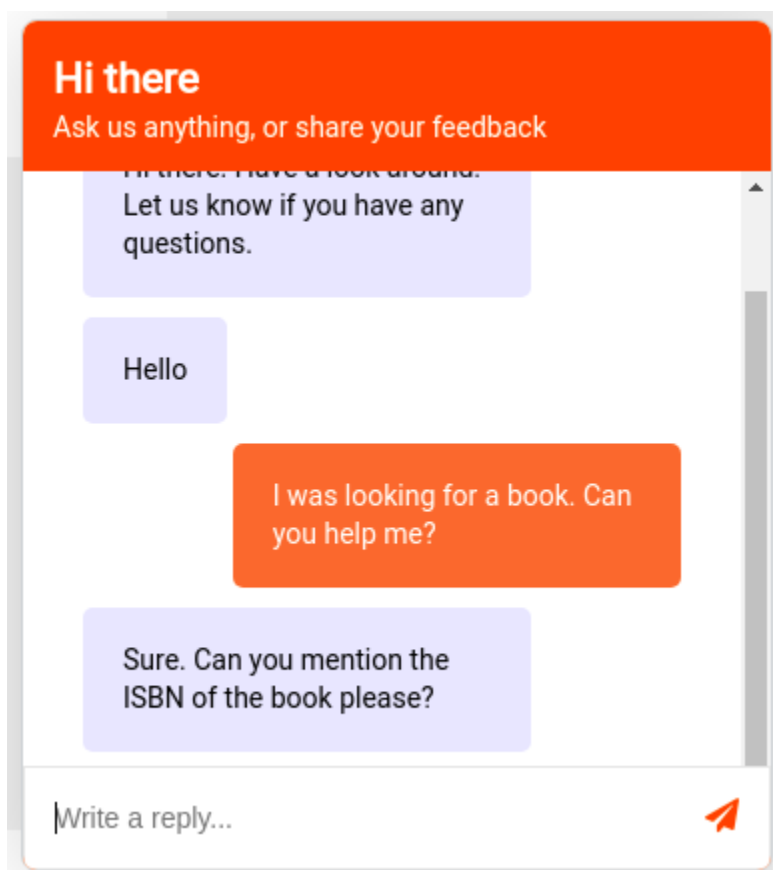
The support staff can view the response real time.



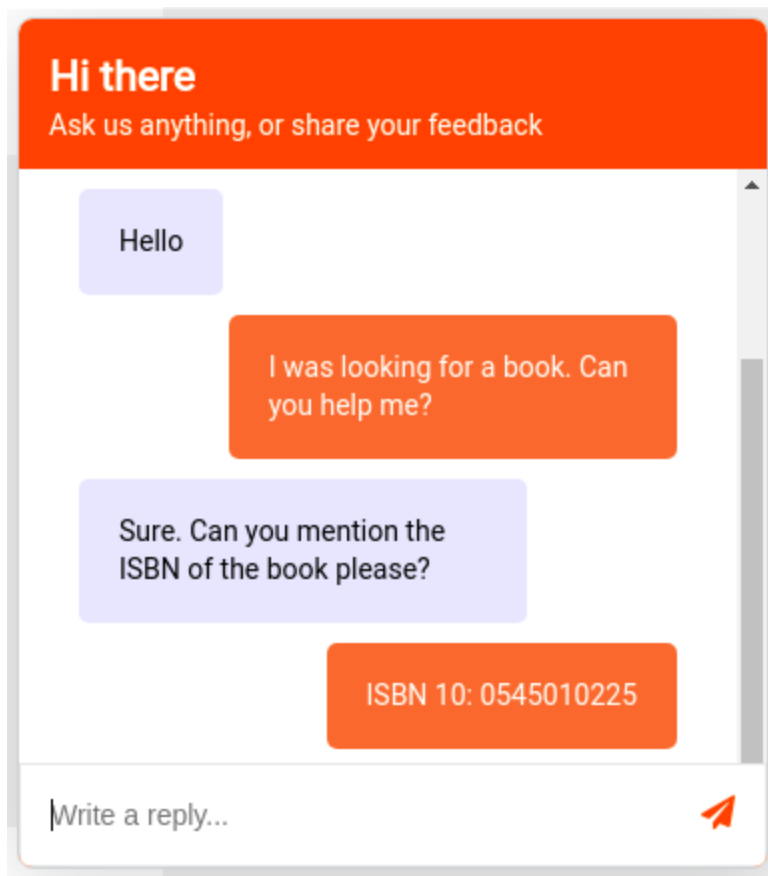
And the conversation continues.



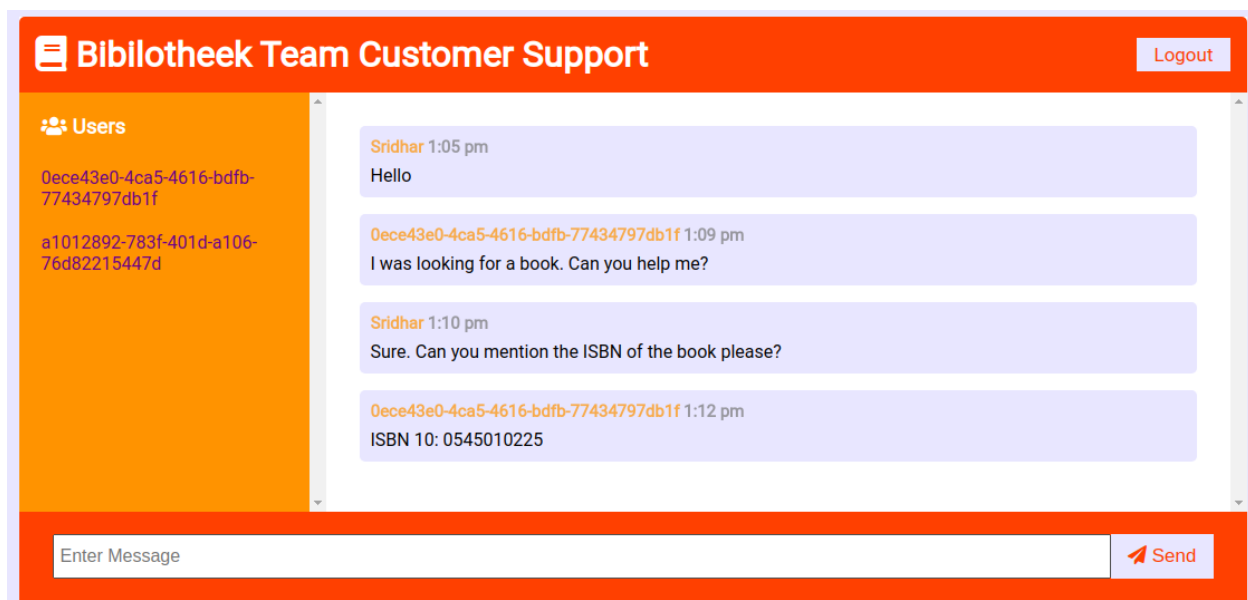
The user receives the message and responds back:



To which the user responds.



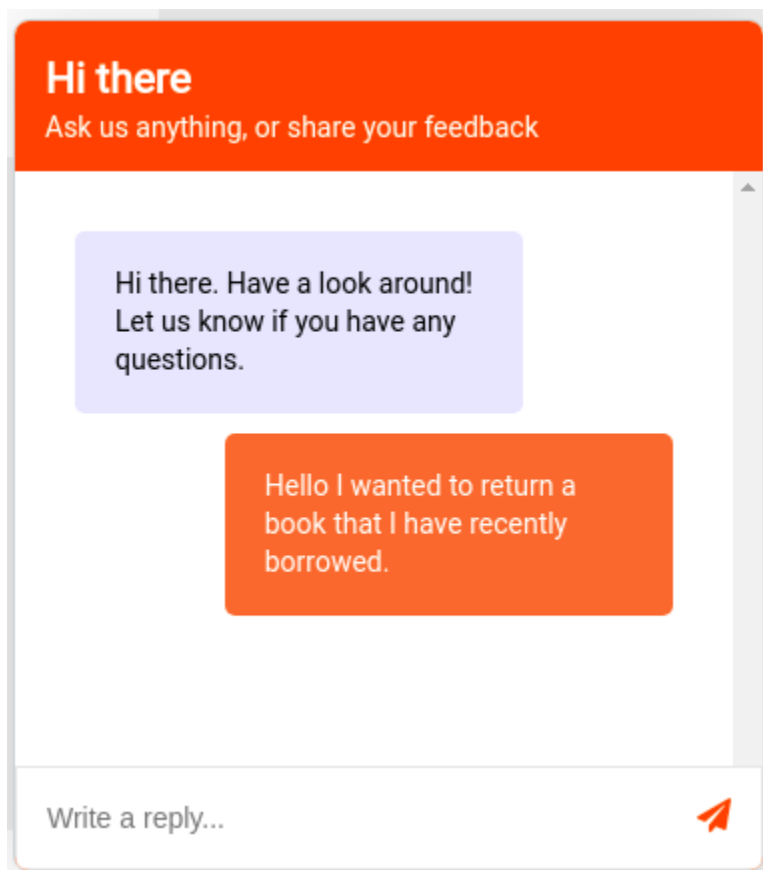
The staff user receives the message.



How long does the user session stay? The user session is not stored in cookies. As soon as the user refreshes the page, the session is not available any more for the staff to respond. However, we can extend this app to support the functionality of sticking a session even after a user refreshes the page.

It is also possible to add feature to show if any staff is currently online. In case there are no staff available, we can show a message to the user that there are no staff currently available to take questions. The user can still go ahead and enter questions and get response later, provided we collect their email address. From the chat we can also collect the email address from the user.

We will now see what happens when a new user starts the chat application on a different browser.



The staff can see that a new user has been added.

Bibilotheek Team Customer Support

Logout

Users

a1012892-783f-401d-a106-76d82215447d

ba553880-3dd6-4eba-bd07-63175b4696fb

ba553880-3dd6-4eba-bd07-63175b4696fb 1:19 pm

Hello I wanted to return a book that I have recently borrowed.

Enter Message

Send

In the current implementation we have kept the prototype simple to demonstrate. However, it is easy for us to show when a new user comes, we can show a notification to the staff to get his/her attention.

Bibilotheek Team Customer Support

Logout

Users

a1012892-783f-401d-a106-76d82215447d

ba553880-3dd6-4eba-bd07-63175b4696fb

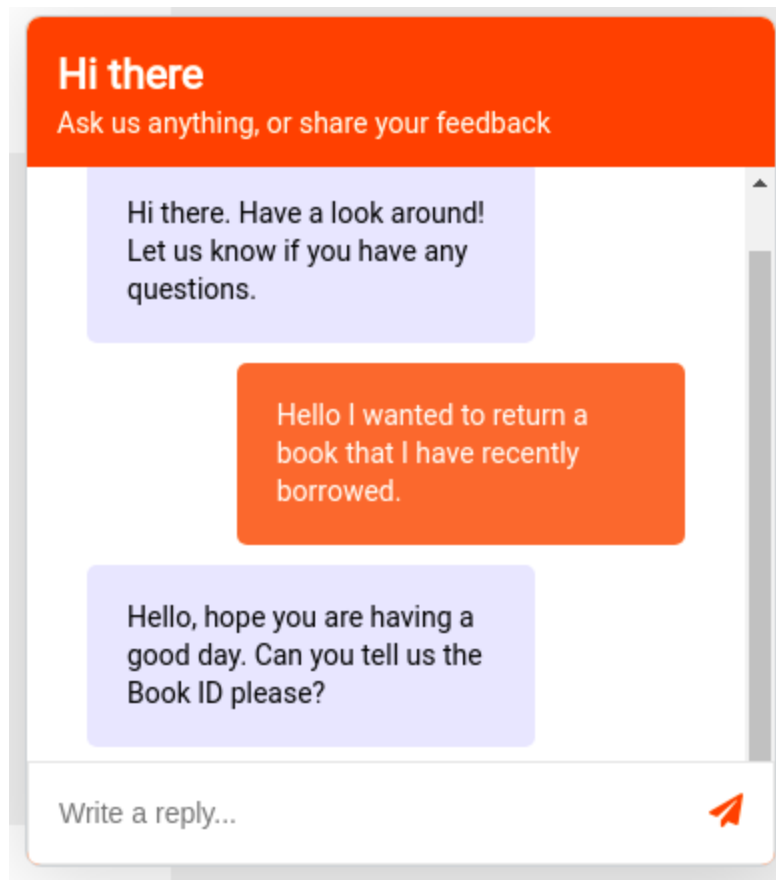
ba553880-3dd6-4eba-bd07-63175b4696fb 1:19 pm

Hello I wanted to return a book that I have recently borrowed.

Hello, hope you are having a good day. Can you tell us the Book ID please?

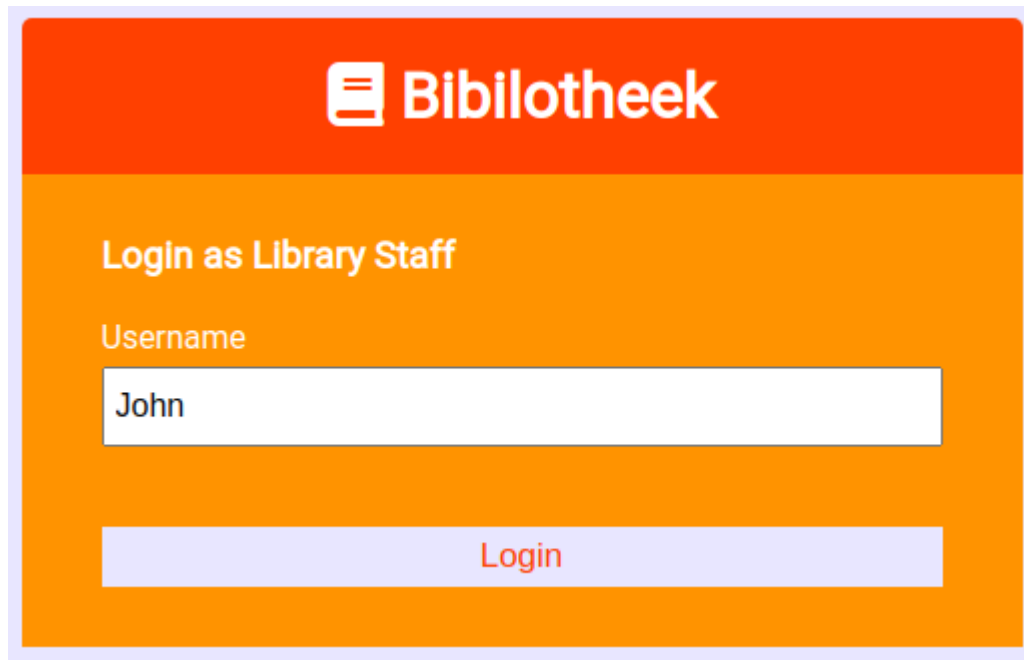
Send

And the conversation goes on with the new user too.



Now let us see the scenario when a new staff user would like to login. We support multiple staff users logged in and they can simultaneously respond to the same user if need be.

I login to the staff portal with a new user.



The image shows a login interface for 'Bibilotheek'. At the top, there is a red header with the 'Bibilotheek' logo and name in white. Below this, the text 'Login as Library Staff' is displayed in white on an orange background. A 'Username' label is followed by a white text input field containing the name 'John'. At the bottom, there is a wide, light purple button with the word 'Login' in red text.

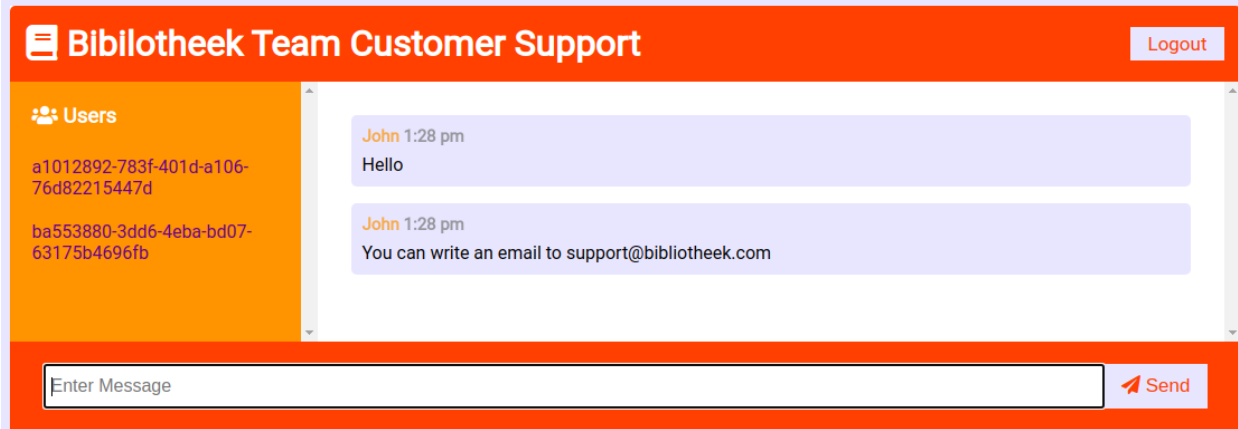
After login, I see this screen



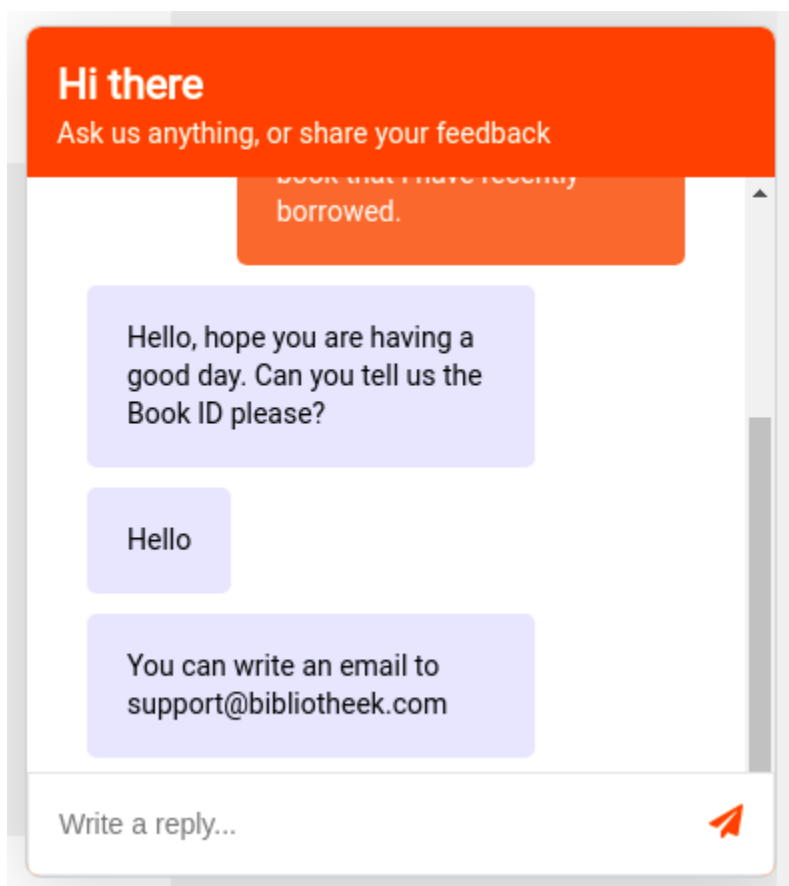
The image shows a chat interface titled 'Bibilotheek Team Customer Support'. The top bar is red with the title on the left and a 'Logout' button on the right. On the left side, there is an orange sidebar with a 'Users' section containing two user IDs: 'a1012892-783f-401d-a106-76d82215447d' and 'ba553880-3dd6-4eba-bd07-63175b4696fb'. The main area is a large white chat window with a vertical scrollbar. At the bottom, there is a red bar containing a text input field with the placeholder 'Enter Message' and a 'Send' button with a red paper plane icon.

The newly logged in staff user, John can continue the conversation. In the current demo, we have not retrieved all the messages exchanged between the user and staff from the database. It is technically possible to have a feature to show all the messages exchanged between the staff and a user. We are storing every conversation between users and staff in our MySQL database.

John writes



The new user won't know who is responding to his query, whether it is the first staff who responded or the newly logged in support member.



If there is a requirement, we can easily extend our chat to show the name of the staff member responding to the query, show a picture of the person responding. We can also add a feature if any staff member has seen the query typed by the user.

We could have a feature to store the conversations within the chat window. So that the logged in user can refer to the previous chat he/she has done with the support staff.

Technical details

The components can be broadly classified into two parts

Components

1. Chat frontend
2. Chat backend

Chat frontend

The frontend is a chat window that can be integrated easily into any website without any limitations. The chat is hosted on a heroku server at <https://aqueous-journey-60276.herokuapp.com/chat/>

We embed this page into the website where we would like the chat to appear using the following code in the website where we would like to see the chat.

```
<embed src="https://aqueous-journey-60276.herokuapp.com/chat/" title="Chat Application" />
```

The real time communication happens using web sockets technology. We use the socket.io library

Quoting from [the Wikipedia article](#) on socket.io

“Socket.IO is a JavaScript library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers. It has two parts: a client-side library that runs in the browser, and a server-side library for Node.js. Both components have a nearly identical API. Like Node.js, it is event-driven. Socket.IO primarily uses the WebSocket protocol with polling as a fallback option, while providing the same interface. Although it can be used as simply a wrapper for WebSocket, it provides many more features, including broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O.”

The frontend has been written in plain JavaScript and HTML, to keep the chat application light weight.

Moment JS has been used to format Date and Time while exchanging messages between the server and client.

The frontend code is organized as follows

- public – this folder is rendered as the output HTML by express JS
 - chat – contains the chat window client-side component (HTML, JS and CSS)
 - staff – contains the code for the support staff component (HTML, JS and CSS)

- index.html is the entry point for staff where a username can be entered
- staff.html is the page for support staff to see the users and respond to them
- test – contains unit tests for the frontend chat component written in Jasmine

Chat Backend

The backend has been implemented using Node JS. Alongside Node JS, express JS has been used to serve the client-side JavaScript and other static files such as HTML and CSS.

The code for backend is organized as follows

- server.js - the main entry point for the Node JS Web Server running on Express JS. This file contains the functions handling web socket events orchestrating the communication between the client and the server. This file also contains the functions to save the messages exchanged between users and the staff to MySQL database.
- util
 - messages.js - utility functions to format the messages exchanged between server and the client
 - users.js - utility for handling all the functions related to user such as maintaining an in-memory list of users and staff currently active

NPM – node package manager has been used for dependency management. The dependencies used for the project can be found in the file named package.json

JavaScript libraries used in the Backend

nodemon has been used for development time auto reload of the Express Web Server upon changes in the IDE. Use the following script to start the nodemon.

```
npm run dev
```

mysql - Node.js module for connecting to MySQL database. The MySQL database is hosted on hostinger.com


The credentials for connecting to the database are:

```
host: '45.87.80.205',
port: 3306,
user: 'u657149837_chatuser',
password: 'palesmoke92',
database: 'u657149837_chat'
```

Database structure

There is only one table named messages to store the messages exchanged between the staff and users.

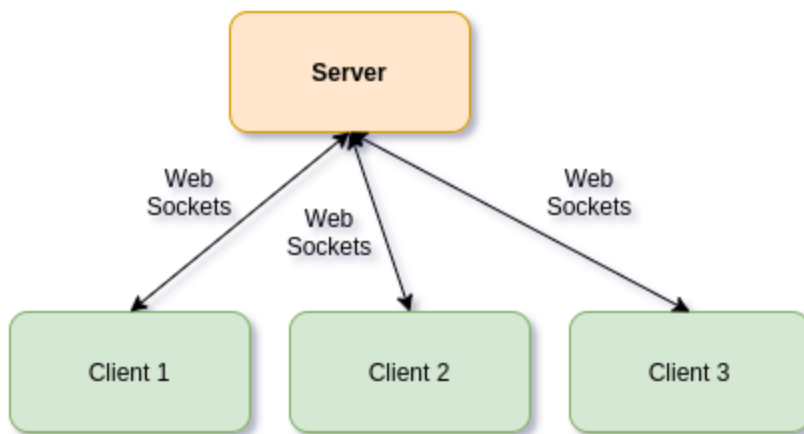
The table has the following attributes

Name	Type
message_id 	int(11)
user_id	varchar(45)
user_name	varchar(45)
staff_id	varchar(45)
staff_name	varchar(45)
sent_time	timestamp
direction	varchar(45)
message	varchar(45)

Explanation of each column in the messages table

- message_id is the primary key for the message stored in this table
- user_id is the unique id which happens to be the socket id of the user connected. The socket id is necessary for the server to respond, when a staff sends a message to the user.
- user_name is the unique UUID generated for each user logged in to the chat. In future we can store real username in this column.
- staff_id is the unique UUID generated for each staff logged in.
- staff_name is the name of the staff entered at the time of login by the staff
- sent_time is the timestamp when the message was sent. This can be used for sorting the messages in chronological order
- direction is either to/from. 'to' indicates that this message was addressed to the user by the staff. 'from' indicates that this message was written by the user as a query to the staff.

Architecture of chat application



The architecture goes like this. The server is running on Node JS and socket.io

The frontend is using JavaScript and socket.io client

Each time a user opens a chat window. There are a series of messages exchanged between the server and the client.

The following table summarizes the messages exchanged:

Event	Description	Who sends?	Who receives?
userJoined	A unique identifier UUID is created to identify the user and is sent as a message in this event	The Frontend Client (Client-Side JavaScript running on the browser)	The Backend Server (Node JS with express JS and socket.io)
message (received by the user)	A welcome message to the newly joined user in the beginning. Then on every message sent by the support staff is forwarded by the server as a message event.	The Backend Server	The Frontend Client
message (received by the staff)	Every message sent by a user to the staff is forwarded by the server back to the user for his/her copy, and sent to a room named 'staff' so that all the staff users can see this message.	The Backend Server	The Frontend Client
chatMessage	The messages sent by the user by typing in the chat window	The Frontend Client	The Backend Server
staffReplyMessage	The messages sent by the support staff addressing the user	The Frontend Client	The Backend Server

roomUsers	Every time a new user joins the chat, this message is being sent by the server to the support staff client with a list of updated users currently active.	The Backend Server	The Frontend Client
-----------	---	--------------------	---------------------