TCSS 558: Applied Distributed Computing

Homework 2

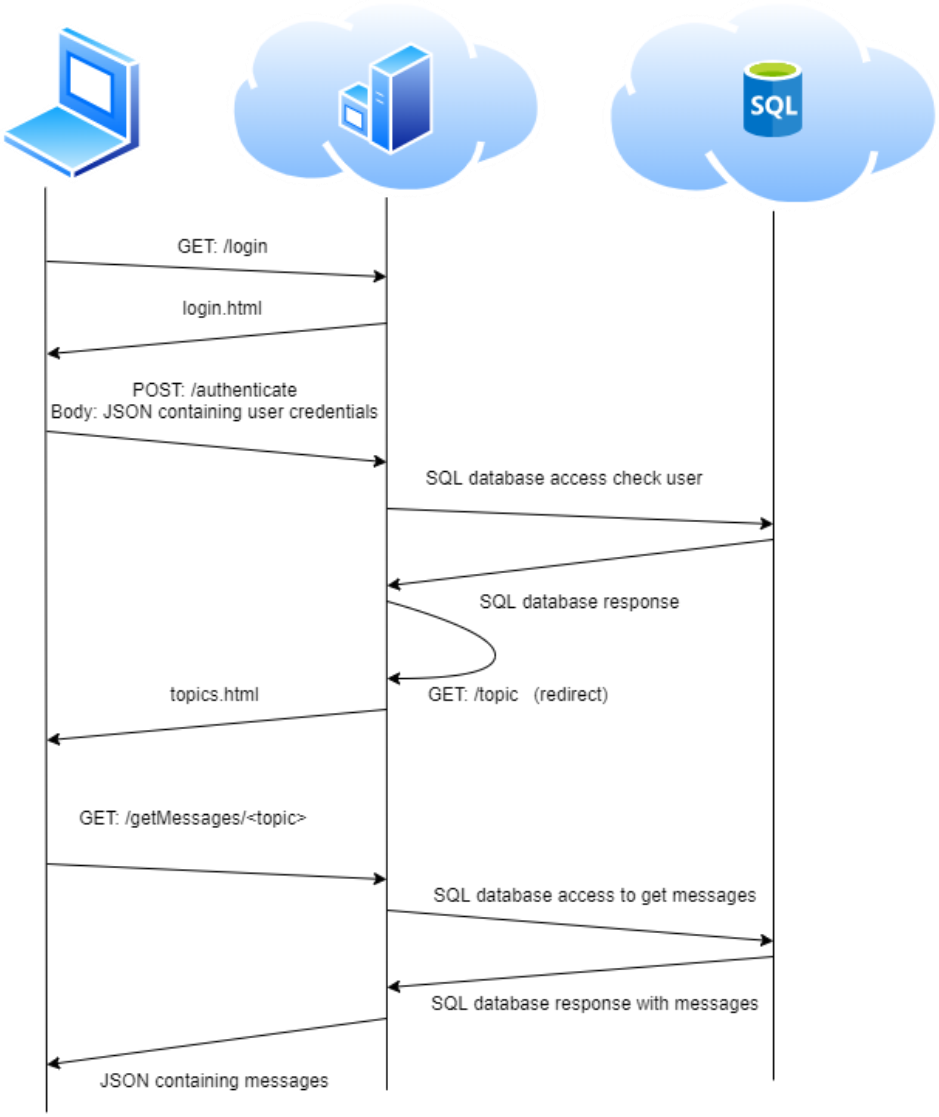
Submitted by: Ankit Singh, Ayush Bandil, Vaishali Girdhar

Title

A cloud-based fully functioning chat application server implementing REST API in a request-response architectural

Implementation and workflow

A user can log on to the web and open the chat application. On the homepage it displays the Login Screen where a user can enter their credentials which is then authenticated with a database in the cloud. After authentication, the user is directed to Chatroom Selection Page where the user can choose what chatroom they want to join. Upon selecting a chatroom, a chat page is displayed where user can interact with other users of the chatroom. There is also an auto-login feature that enables users to automatically login without needing to entering credentials every time. New users can register using the register page. Registration button is available on the account login page.



REST API Calls

Type	URL Endpoint	Request Body	Response Body	Function
GET	/login	N/A	Web page: login.html	Get login page

POST	/authenticate	{
		"username":<username>,
		"password":<password>
		}
			{
			"code":<http code>,
			"status":<valid/invalid>
			}
				Process login and then redirect to /topics page if login is authenticated i.e. user credentials match user in database

GET	/topics	N/A	Web page: topics.html	Get topics page

GET	/getMessages/<topic>	N/A	{
			"message":{
			"topic":<topic>,
			"username":<username>,
			"time":<time>,
			"messages":<message>
			}
			}
				Get user messages from database corresponding to the current topic

PUT	/sendMessage	{
		"message":<message>,
		"username":<username>,
		"url":<url> (for topic)
		}
			{
			"code":<http code>,
			"status":<success/failure>
			}
				Put messages to messages database

GET	/register	N/A	Web page: Register.html	Get register page view

POST	/register	{
		"username":<username>,
		"password":<password>
		}
			{
			"code":<http code>,
			"status":<success/failure>
			}
				Register a new user in the database

Get	/static/<filepath>	N/A	Return corresponding file	Get .js and .css files needed by the HTML web pages

Key Features

I. The chat server and associated Microsoft SQL Server database is hosted on the cloud
Note: Our team has deployed the project on Cloud (Azure IoT hub) while giving the presentation during the class. Currently the server is not running, please let us know if you want to check on cloud.

II. **Technologies:**

- *MS SQL Server* is used for database.
- API uses *JSON* content. *JSON* file also has a message object.
- Chatbox is refreshed every few seconds using *AJAX* without needing to refresh the entire page. *Html* and *CSS* are using for web designing.
- The server side and databased integration is written using *java.*

III. Chat Application Workflow Features:

- Auto-login feature that enables users to automatically login without needing to entering credentials every time.
- Active Status of users shown as whether they are ONLINE or OFFLINE.
- A new user can register using the Register page, the corresponding entry is saved in the database.
- After logging-in a user can select the chat room from three options: Movies, Food and Sports. The user can then read the precious conversations of the topic.
- Fairly lightweight application that loads up pretty fast