BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS 299

SUMMER TRAINING REPORT

Mustafa Göktan Güdükbay

21801740

Performed at

Software Research & Development and Consultancy (SRDC) Corporation

29 June 2020 - 27 July 2020

Table of Contents

Table of Figures

1 Introduction

I performed my internship at SRDC (Software Research & Development Consultancy) company. The duration of my summer internship was four weeks. The company works on areas like health, security, web development, and big data [1]. They do research, work on several EU commission projects, and design products for industrial needs. I did my summer internship at SRDC because I was curious about web development and database systems. Together with me, there were seven interns, and they were computer engineering students in METU.

I worked on three assignments during my summer internship. Their purpose was the same; developing user management and messaging tool. However, I used different technologies and applications for the development of each assignment.

2 Company Information

2.1 About the company

SRDC (Software Research & Development Consultancy) is a company that works on software research and development. They are located in the METU technopark. Some of the European Commission Projects they participated are ADLIFE, EFPF, I2PANEMA, FAIR4HEALTH, C3-Cloud, Medolution and, so on [2]. The company focuses on several areas including eHealth solutions, semantic web technologies, eGovernment solutions, conformance testing, emergency management security, big data, data analytics, and business intelligence [1]. Prof. Dr. Asuman Doğac is the general manager of SRDC and in the past, she worked in METU. Her research areas are electronic health solutions, business applications, and database systems.

2.2 About your department

In SRDC, there are not any departments.

2.3 About the hardware and software systems

The company uses several technologies on web development and database applications. Programming languages that I used in this internship were Java, HTML, CSS, JavaScript, TypeScript. I used were Angular API for client-side programming, Node.js for server-side programming, and Express.js for route programming. The database technologies that I used were PostgreSQL and MongoDB. The company also uses Java build tools such as Ant, Maven, and IDE’s like IntelliJ IDEA, Microsoft Visual Studio Code, and Eclipse.

2.4 About your supervisor

My advisor in SRDC was Dr. Mustafa Yüksel. Dr.Yüksel started working in SRDC in 2005, and he is one of the co-founders of SRDC. Dr. Yüksel is working as a project manager and senior researcher. Dr.Yüksel got his BSc degree in Computer Engineering at METU in 2006. Later, he started MSc in METU and finalized his thesis on Sharing Electronic Healthcare Records Across Country Borders in 2008. He also got his Ph.D. degree from METU, and his Ph.D. was on Semantic Interoperability Framework for Reinforcing Post Market Safety Studies in 2013 [3].

3 Work Done

There were three assignments given during the internship. The purpose of all the assignments was to create a user management and messaging application with different technologies.

3.1 First Assignment: User Management & Messaging Tool

The purpose of this assignment was to create a user management and messaging application using the command-line interface. In this application, the server has a default list of admins and these admins can perform create, read, update, and delete operations on users. The users logged in with a username and a password. All users can message each other and read their inboxes and outboxes.

This assignment was an example of multi-threaded socket programming in Java. Socket programming is a method for communicating using two nodes on a network. One socket listens on a port, and another port connects to that port to exchange information. There should be a client handler to handle multiple clients at the same time to create a multi-threaded server [4].

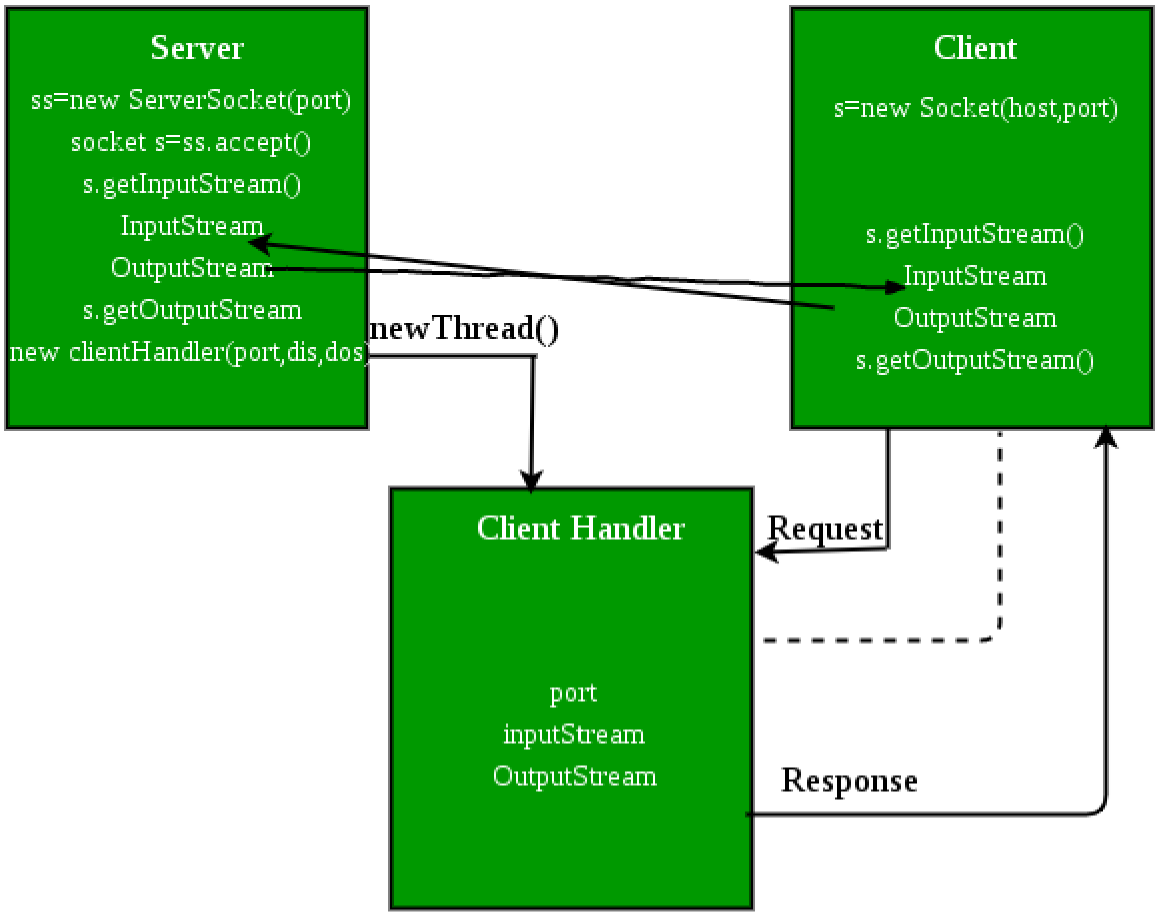
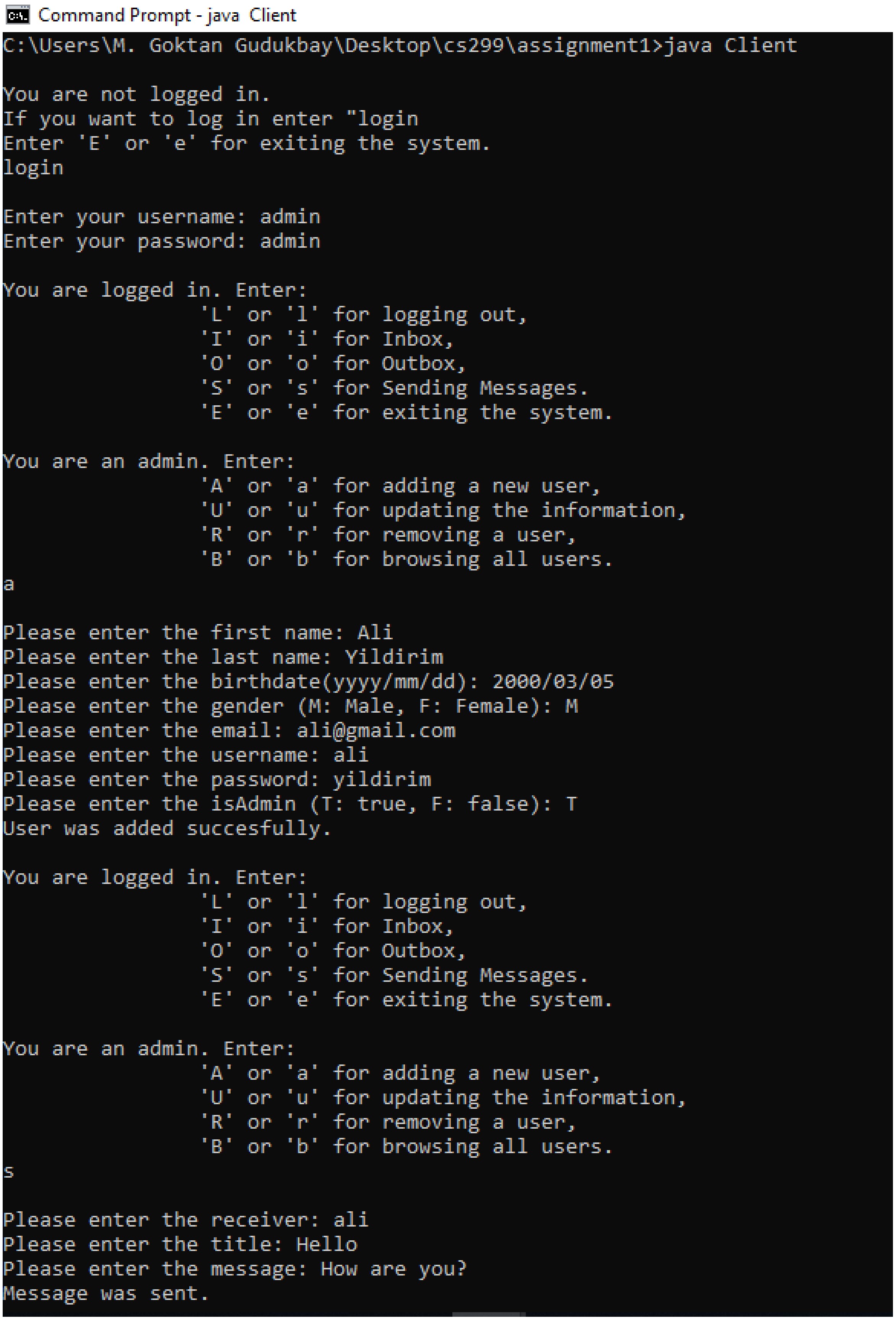


Figure 1. Flow Diagram of Multi-threaded Socket Programming [4].

A protocol was created to exchange information between the client and the server. The vertical bar "|" was used as the protocol. When the client made a request to the server the operation to be performed (read inbox, read outbox, send a message, create a user, update a user, delete a user, list all users) and the necessary information for that operation was sent. When the server sent a response to the request made by the client the information to be displayed was sent.

PostgreSQL was used as a database to hold information of the users. There were two different tables for users and messages. Users table held information on username, first name, last name, birth date, gender, email, and admin info. The primary key for the users table was the username variable. On the other hand, the messages table held information on the sender username, receiver username, message date, a message title, and a message. The messages table had two foreign key definitions for the sender username and the receiver username. Later this was used to retrieve sender and receiver information from the users table for displaying the messages in inboxes and outboxes. The foreign key definition also helped me when I checked the validity of each receiver.

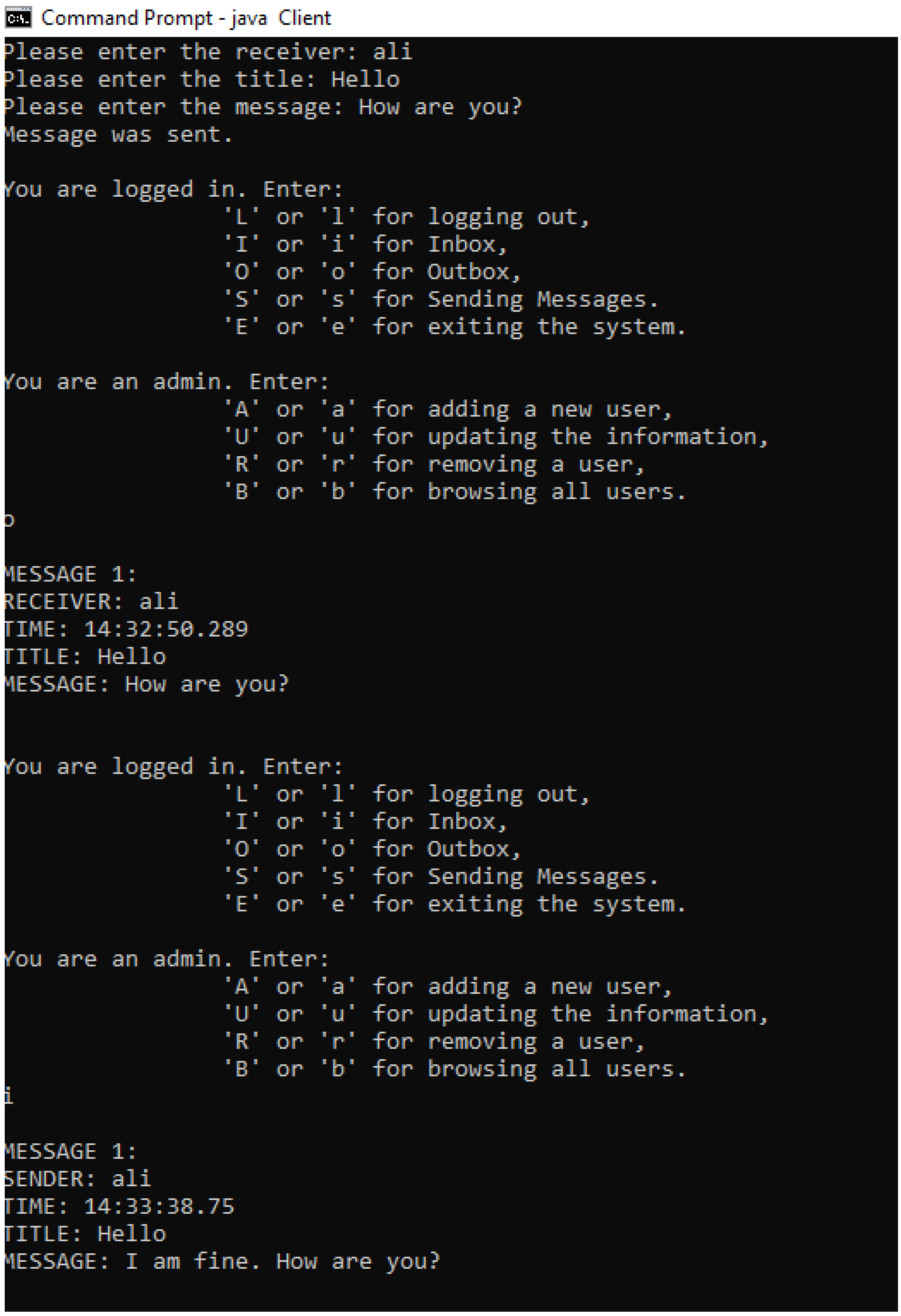
JDBC (Java Database Connection) was used to connect with the database and exchange information. There was a separate class for connecting and exchanging information with the database. The program was done using Object-Oriented Programming. The assignment aimed to communicate within the local network but I tried the application with two different computers connected to the same WI-FI. IntelliJ IDEA was used to develop and edit code.



Command Prompt - java Client

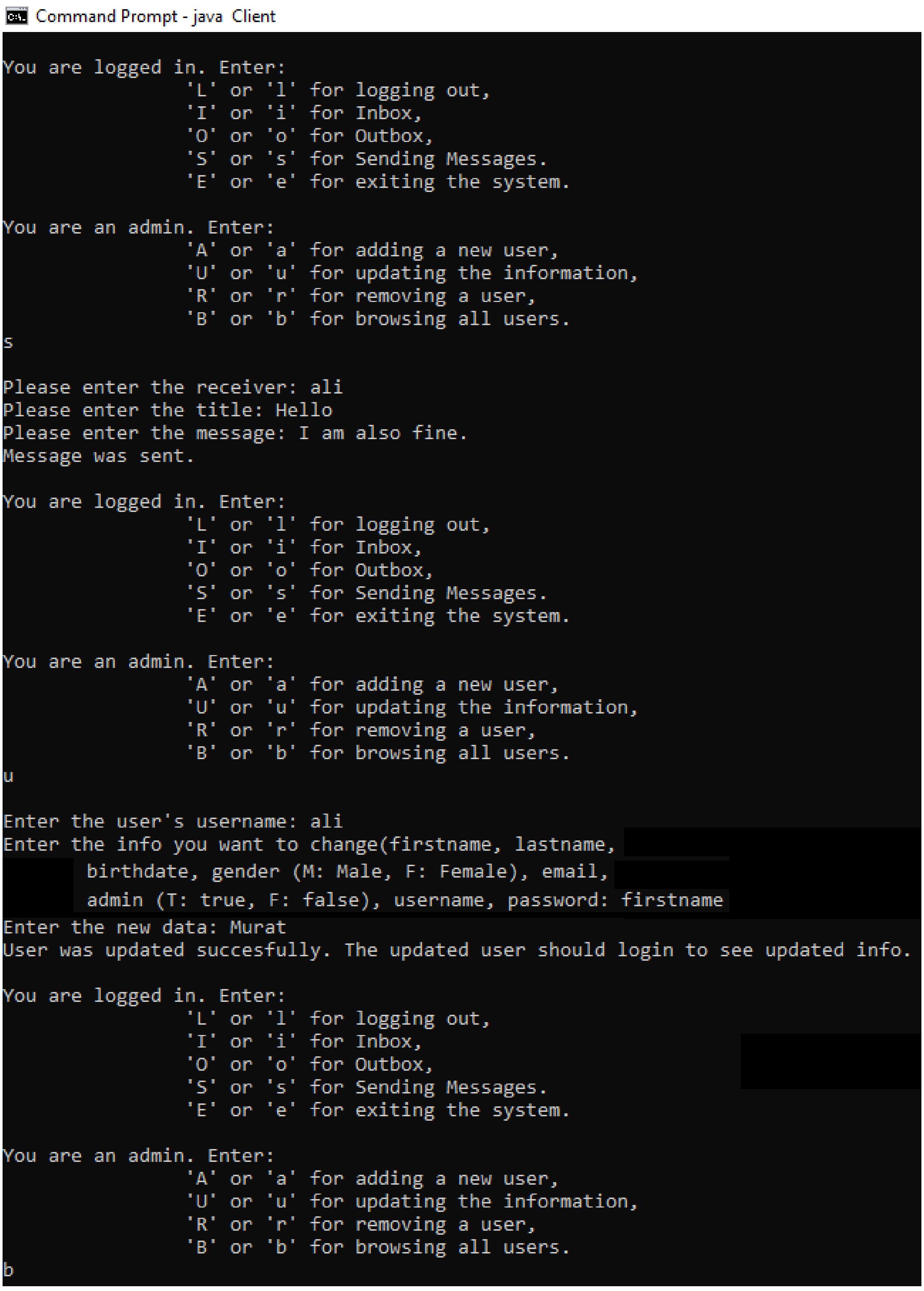
```
C:\Users\M. Goktan Gudukbay\Desktop\cs299\assignment1>java Client

You are not logged in.
If you want to log in enter "login
Enter 'E' or 'e' for exiting the system.
login

Enter your username: admin
Enter your password: admin

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
a

Please enter the first name: Ali
Please enter the last name: Yildirim
Please enter the birthdate(yyyy/mm/dd): 2000/03/05
Please enter the gender (M: Male, F: Female): M
Please enter the email: ali@gmail.com
Please enter the username: ali
Please enter the password: yildirim
Please enter the isAdmin (T: true, F: false): T
User was added succesfully.

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
s

Please enter the receiver: ali
Please enter the title: Hello
Please enter the message: How are you?
Message was sent.
```

Figure 2.The screenshot of the Client application for user creation.



```
Command Prompt - java  Client
Please enter the receiver: ali
Please enter the title: Hello
Please enter the message: How are you?
Message was sent.

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
o

MESSAGE 1:
RECEIVER: ali
TIME: 14:32:50.289
TITLE: Hello
MESSAGE: How are you?


You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
i

MESSAGE 1:
SENDER: ali
TIME: 14:33:38.75
TITLE: Hello
MESSAGE: I am fine. How are you?
```

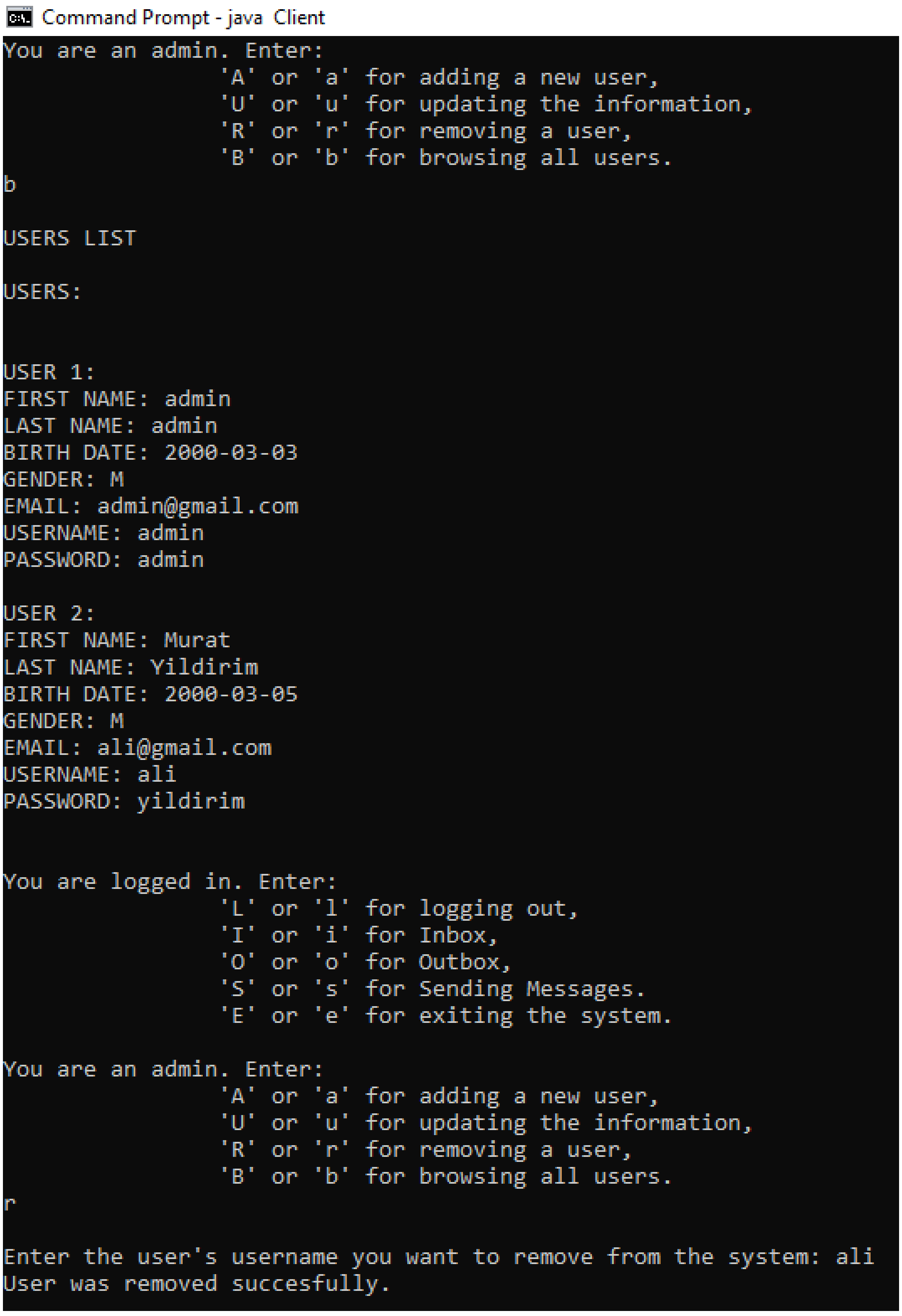
Figure 3. The screenshot of the Client application for message sending.



Command Prompt - java Client

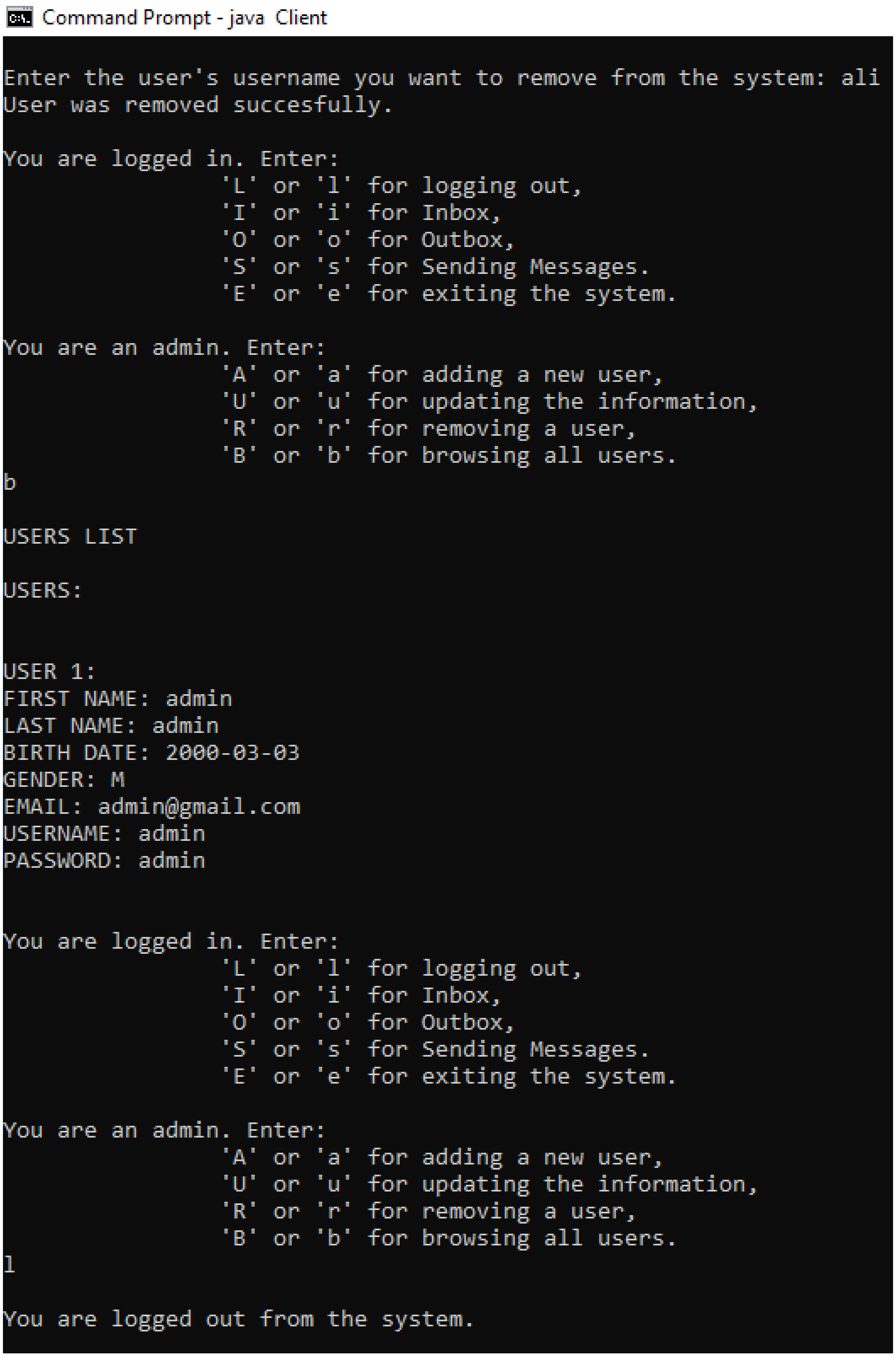
```
You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
s

Please enter the receiver: ali
Please enter the title: Hello
Please enter the message: I am also fine.
Message was sent.

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
u

Enter the user's username: ali
Enter the info you want to change(firstname, lastname,
       birthdate, gender (M: Male, F: Female), email,
       admin (T: true, F: false), username, password: firstname
Enter the new data: Murat
User was updated succesfully. The updated user should login to see updated info.

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
b
```

Figure 4. The screenshot of the Client application for message sending and updating user information.



```
You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
b

USERS LIST

USERS:


USER 1:
FIRST NAME: admin
LAST NAME: admin
BIRTH DATE: 2000-03-03
GENDER: M
EMAIL: admin@gmail.com
USERNAME: admin
PASSWORD: admin

USER 2:
FIRST NAME: Murat
LAST NAME: Yildirim
BIRTH DATE: 2000-03-05
GENDER: M
EMAIL: ali@gmail.com
USERNAME: ali
PASSWORD: yildirim


You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
r

Enter the user's username you want to remove from the system: ali
User was removed succesfully.
```

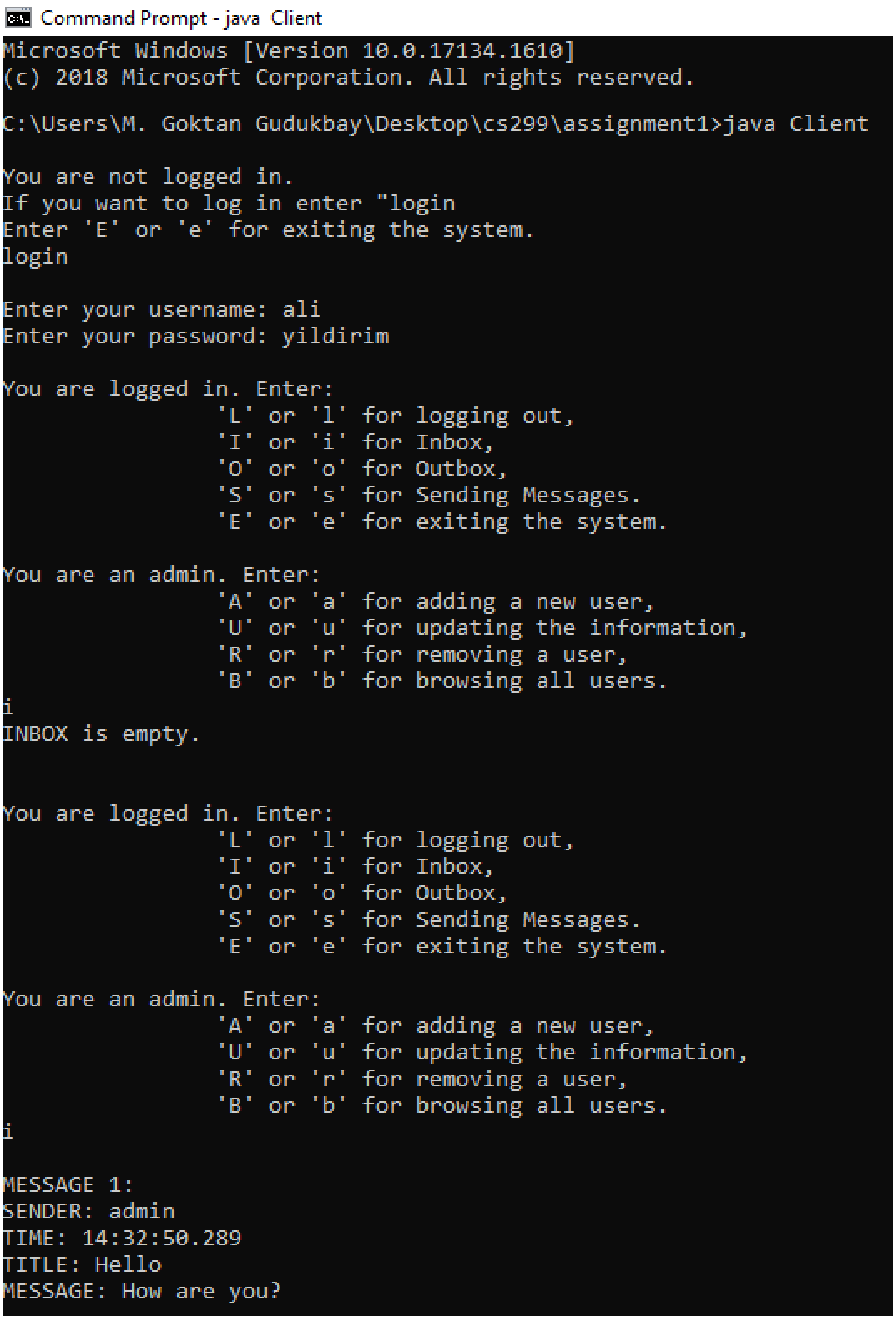
Figure 5. The screenshot of the Client application for removing a user.



Command Prompt - java Client

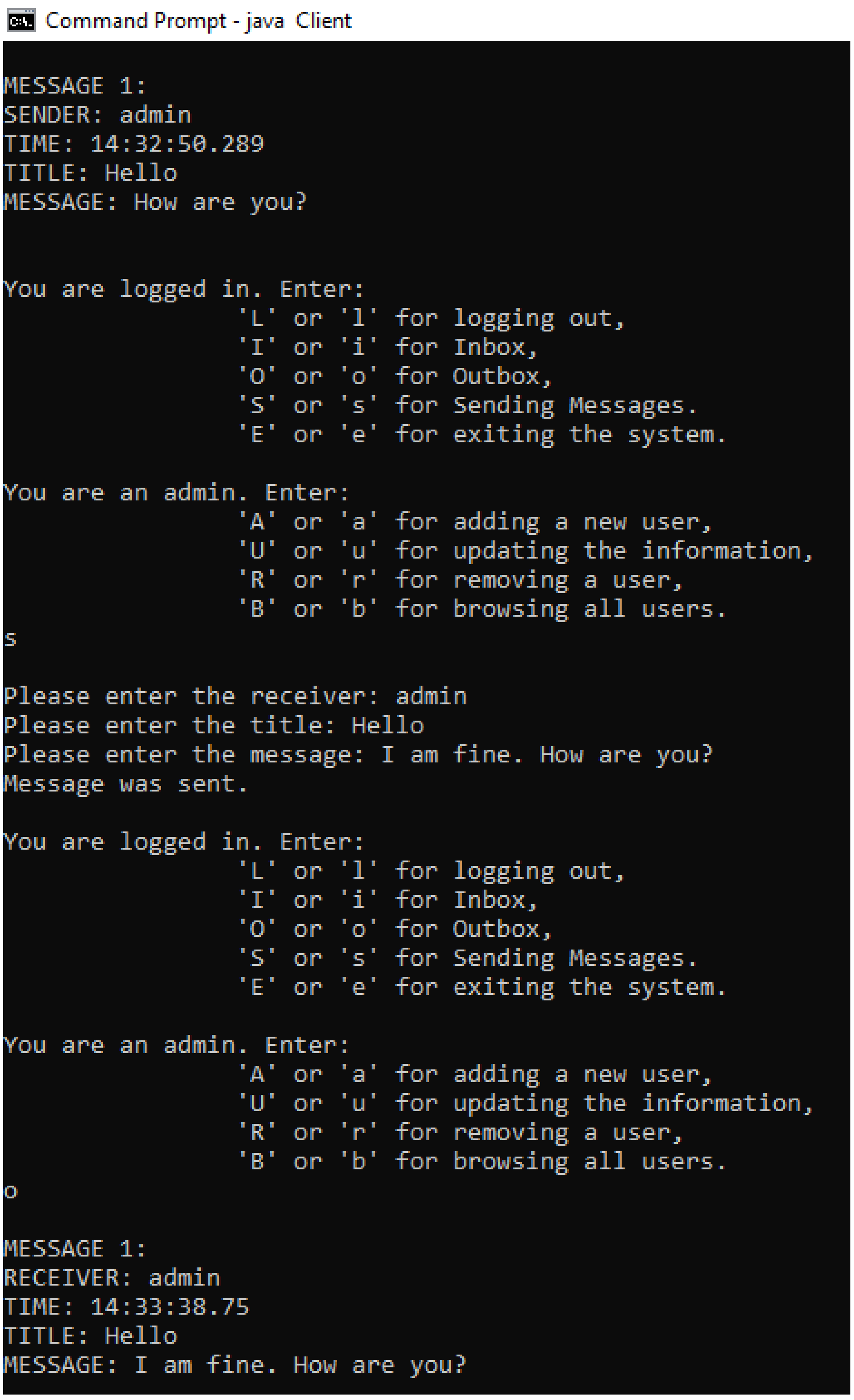
```
Enter the user's username you want to remove from the system: ali
User was removed succesfully.

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
b

USERS LIST

USERS:


USER 1:
FIRST NAME: admin
LAST NAME: admin
BIRTH DATE: 2000-03-03
GENDER: M
EMAIL: admin@gmail.com
USERNAME: admin
PASSWORD: admin


You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
l

You are logged out from the system.
```

Figure 6. The screenshot of the Client application for listing users after removing the user with username “Ali”.



```
Command Prompt - java  Client
Microsoft Windows [Version 10.0.17134.1610]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\M. Goktan Gudukbay\Desktop\cs299\assignment1>java Client

You are not logged in.
If you want to log in enter "login
Enter 'E' or 'e' for exiting the system.
login

Enter your username: ali
Enter your password: yildirim

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
i
INBOX is empty.


You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
i

MESSAGE 1:
SENDER: admin
TIME: 14:32:50.289
TITLE: Hello
MESSAGE: How are you?
```

Figure 7. The screenshot of the Client application for listing messages in the Inbox of the user with username “Ali”.



```
Command Prompt - java  Client

MESSAGE 1:
SENDER: admin
TIME: 14:32:50.289
TITLE: Hello
MESSAGE: How are you?


You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
s

Please enter the receiver: admin
Please enter the title: Hello
Please enter the message: I am fine. How are you?
Message was sent.

You are logged in. Enter:
                'L' or 'l' for logging out,
                'I' or 'i' for Inbox,
                'O' or 'o' for Outbox,
                'S' or 's' for Sending Messages.
                'E' or 'e' for exiting the system.

You are an admin. Enter:
                'A' or 'a' for adding a new user,
                'U' or 'u' for updating the information,
                'R' or 'r' for removing a user,
                'B' or 'b' for browsing all users.
o

MESSAGE 1:
RECEIVER: admin
TIME: 14:33:38.75
TITLE: Hello
MESSAGE: I am fine. How are you?
```

Figure 8. The screenshot of the Client application for sending a message and seeing the Outbox for the user “admin”.

3.2 Second Assignment: Web-based User Management & Messaging Tool (Version 1)

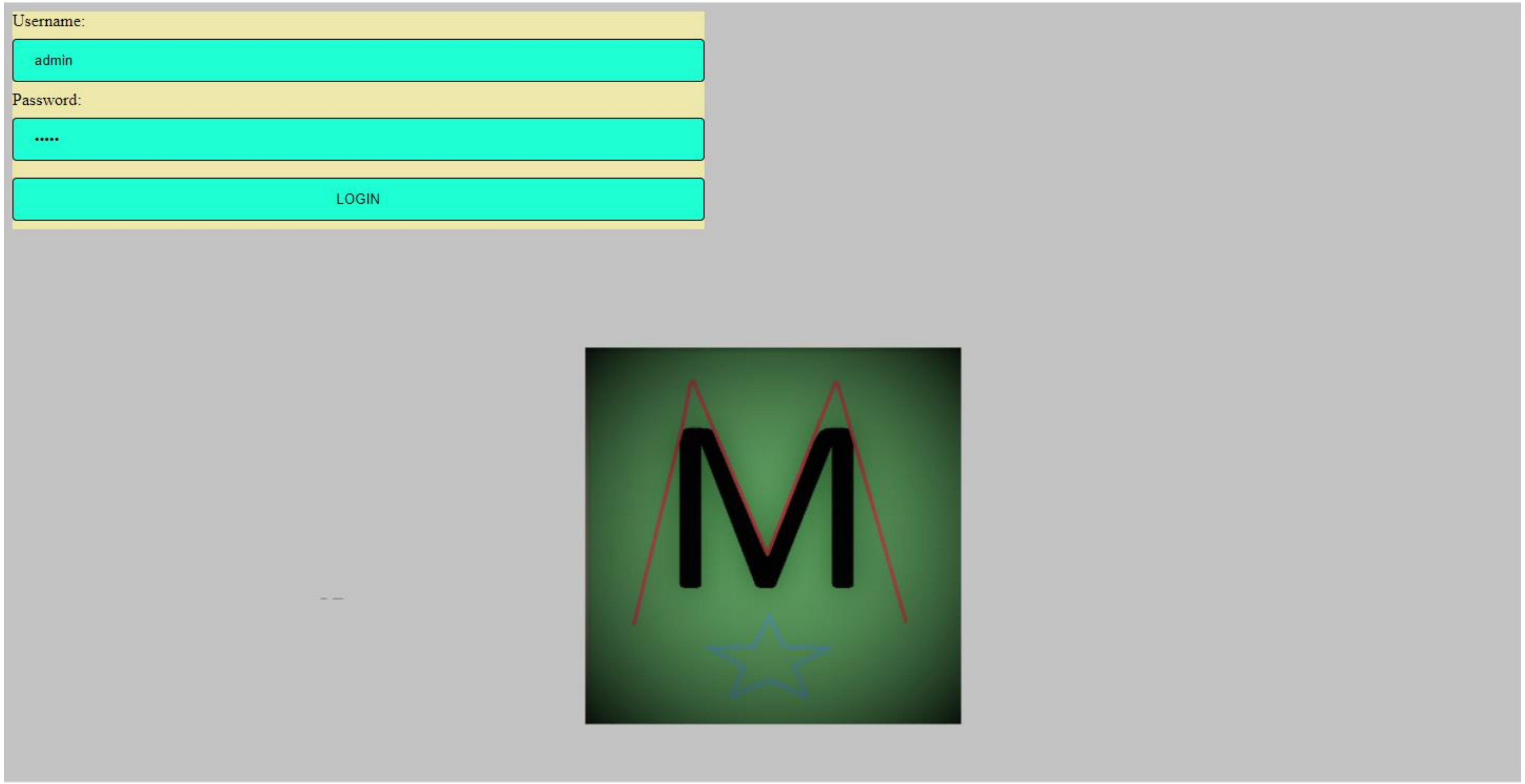
The purpose of this assignment was the same as the previous one; to create a user management and messaging application where the admins can perform create, read, update, and read operations and all users can send message to other users, read their inboxes and outboxes.

This assignment had three parts. The first part of the assignment was creating a server using SOAP (Simple Object Access Protocol) web services and Java [5]. SOAP is a communication protocol for sending and receiving information based on XML (eXtensible Markup Language). XML is used to store and exchange information and it is similar to HTML but more self-descriptive [6]. An XSD (XML Schema Definition) was written to handle client requests and server responses. Consequently, the server had the necessary information for the requests made by the client and it returned the responses in a structured form to the client. An XSD is a document for defining the structure of an XML document [7]. The properties of objects were also defined in the XSD file and by using JAXB (Jakarta XML Binding) Java classes for those objects were automatically generated. JAXB is a framework that allows users to correlate XML schemas and Java representations [8].

PostgreSQL database management system was used to hold and transfer information. The database management system that was designed in the previous assignment was used for this assignment. The dependences were built using Maven. The dependences were produced using a file called POM(Project Object Model). A POM contains the structural dependencies for the project.

The second part of the assignment was about testing the server using SOAPUI and Spring framework. SOAPUI is a technology used to test web service by checking the received XML responses after making get, post, delete, update, and other HTTP requests [9]. Later, client testing was made in Java using Spring framework. Some of the actions were tested by making requests and displaying the responses received on the command-line.

The third part of the assignment was making a web client using Javascript and HTML. The technology that was used in this part was AJAX (Asynchronous JavaScript and XML). The combination of XML objects and usage of JavaScript is called AJAX [10]. Libraries like XMLHttpRequest and DOMParser were used to make requests to the server and parse the responses received. XMLHttpRequest is an object that is used to retrieve information from a server [11]. DOMParser (Document Object Model Parser) is a parser that is used to access XML documents [12]. The development of the website was designed with HTML and CSS. This assignment had an extra part for authentication. Other than logging and logging out, a randomly generated authentication token was used to send requests to the server. This token was saved to the user table in the database. This assignment did not handle CORS (Cross-origin resource sharing) so the application was tested on the same domain, the same port of the network. CORS is a structure that allows browsers to access to resources on a different origin by using extra HTTP headers [13]. IntelliJ IDEA Ultimate Version was used to develop and edit code.


Username:

admin

Password:

•••••

Figure 9. The login screen for the Web-based Messaging Application.

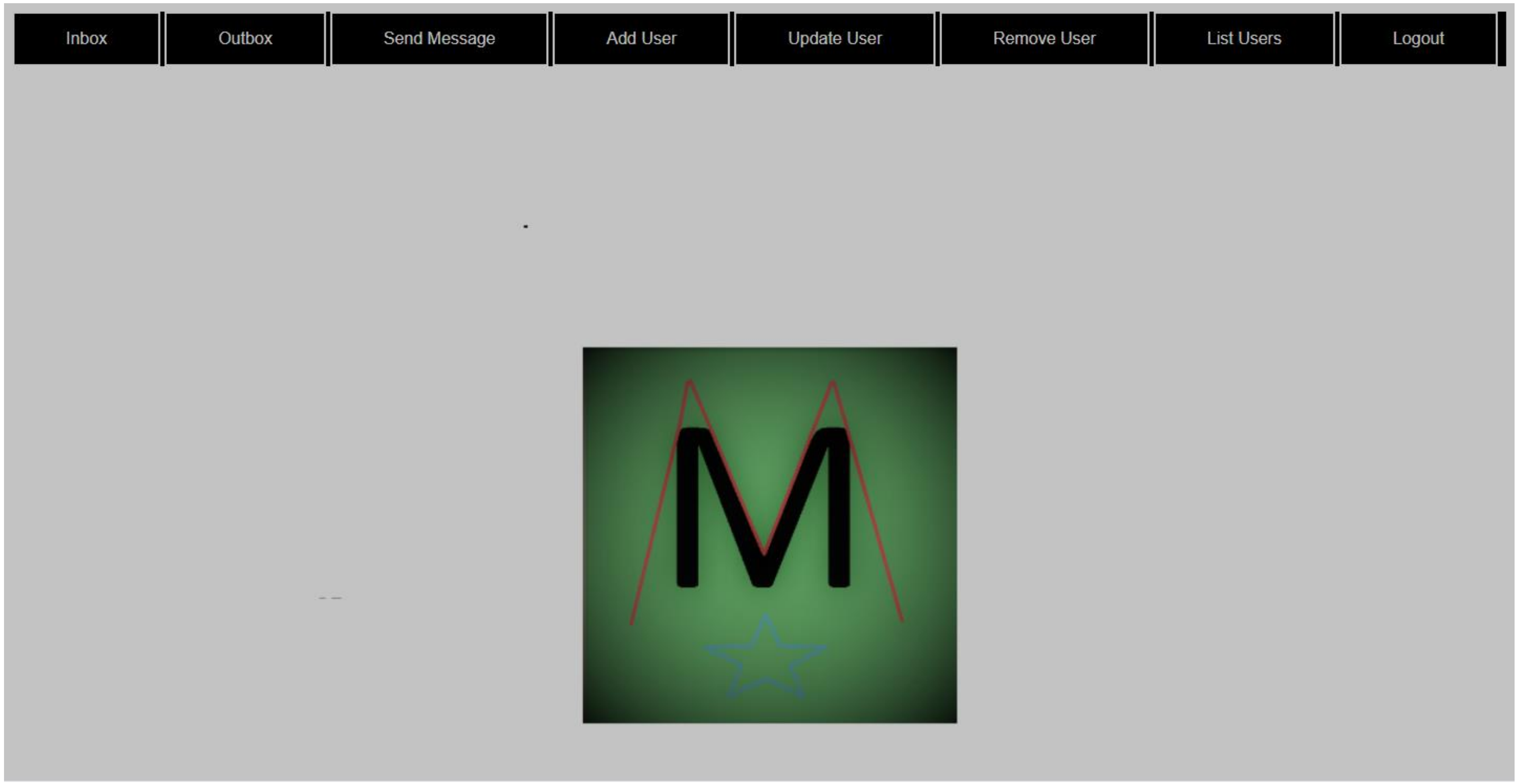
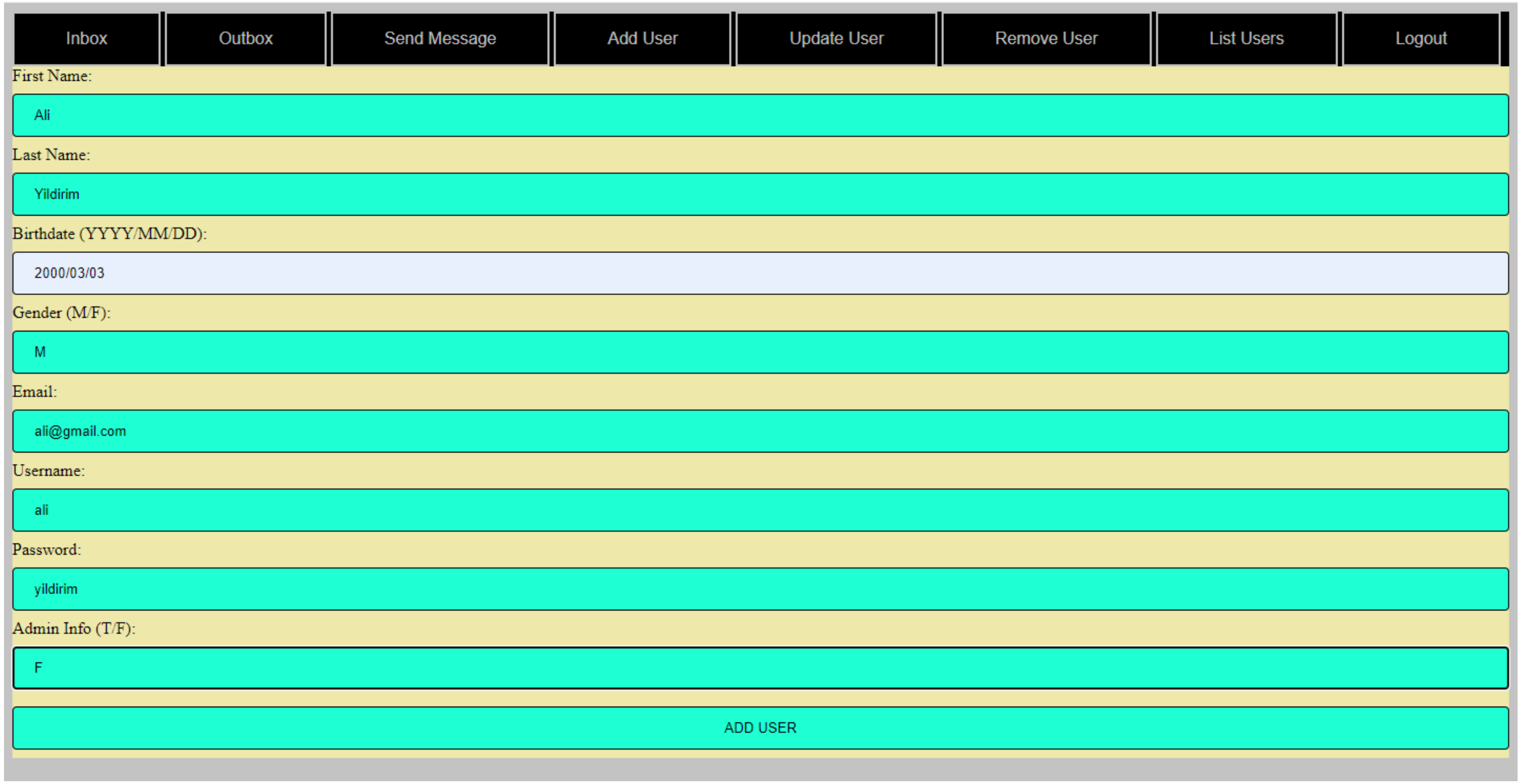
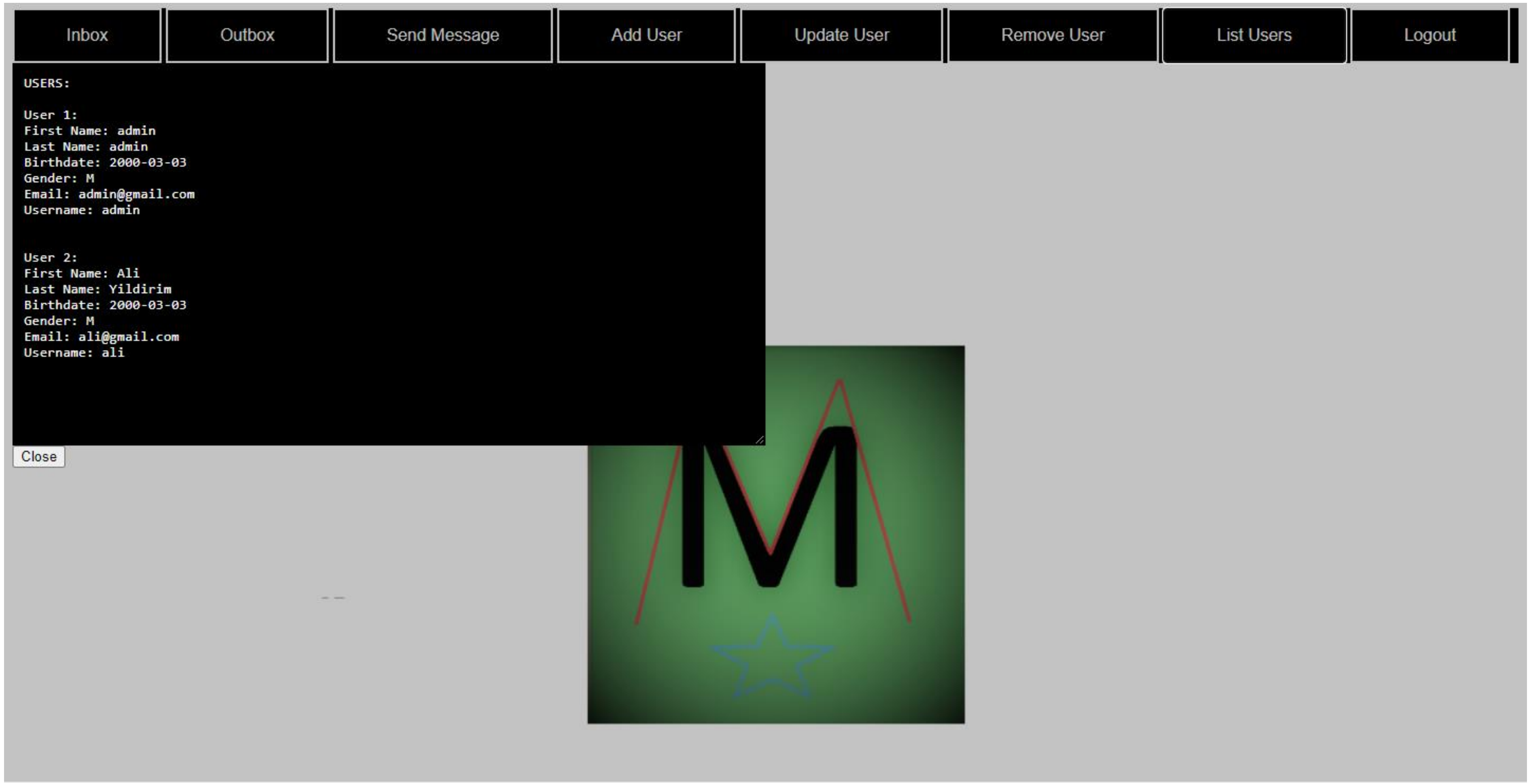


Figure 10. The administrator screen for the Web-based Messaging Application.



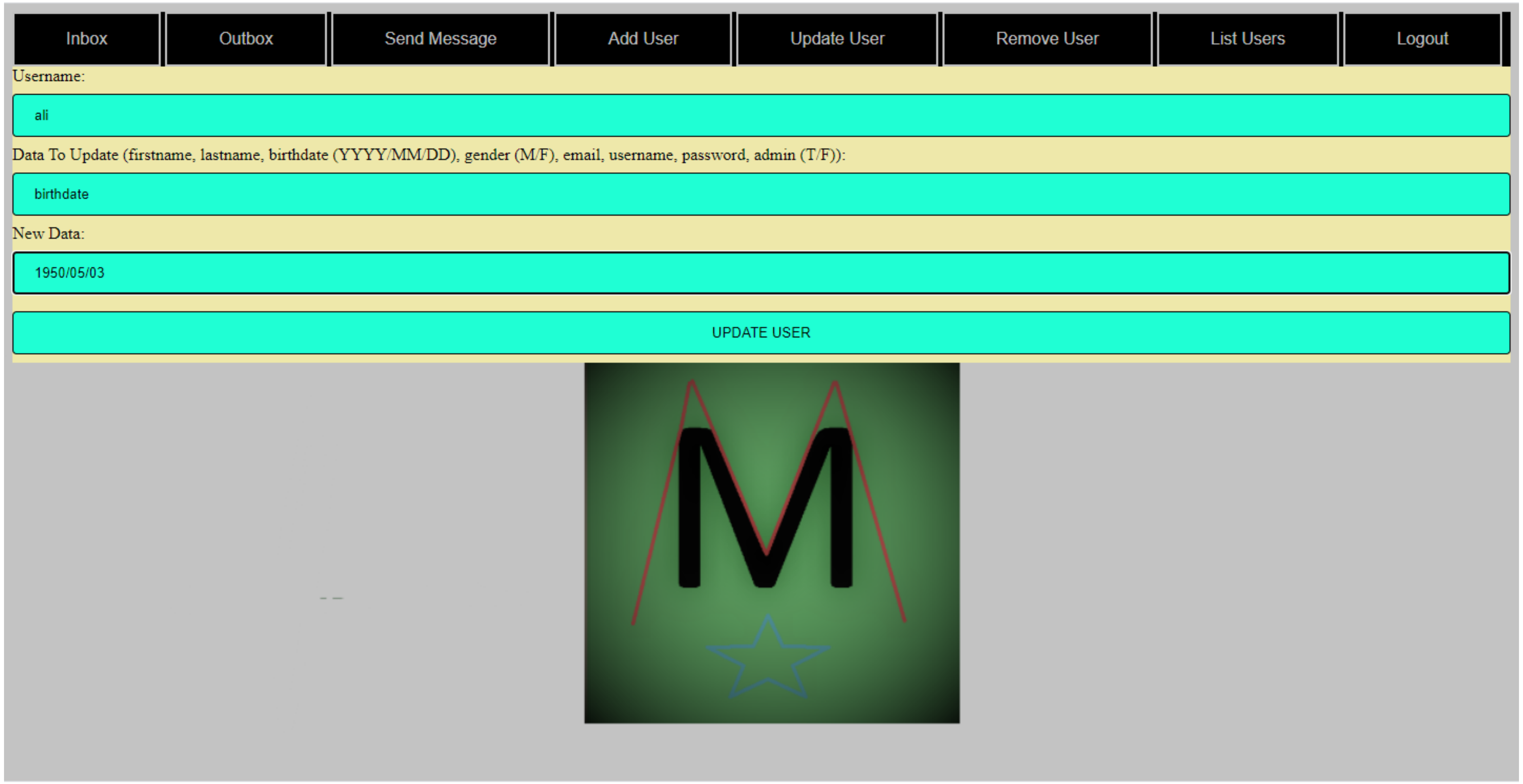
First Name:	Ali
Last Name:	Yildirim
Birthdate (YYYY/MM/DD):	2000/03/03
Gender (M/F):	M
Email:	ali@gmail.com
Username:	ali
Password:	yildirim
Admin Info (T/F):	F

Figure 11. The administrator screen of the Web-based Messaging Application for adding a user. Only the administrator can add a new user.



```
USERS:

User 1:
First Name: admin
Last Name: admin
Birthdate: 2000-03-03
Gender: M
Email: admin@gmail.com
Username: admin


User 2:
First Name: Ali
Last Name: Yildirim
Birthdate: 2000-03-03
Gender: M
Email: ali@gmail.com
Username: ali
```

Figure 12. The administrator screen of the Web-based Messaging Application for listing users.



Username:

ali

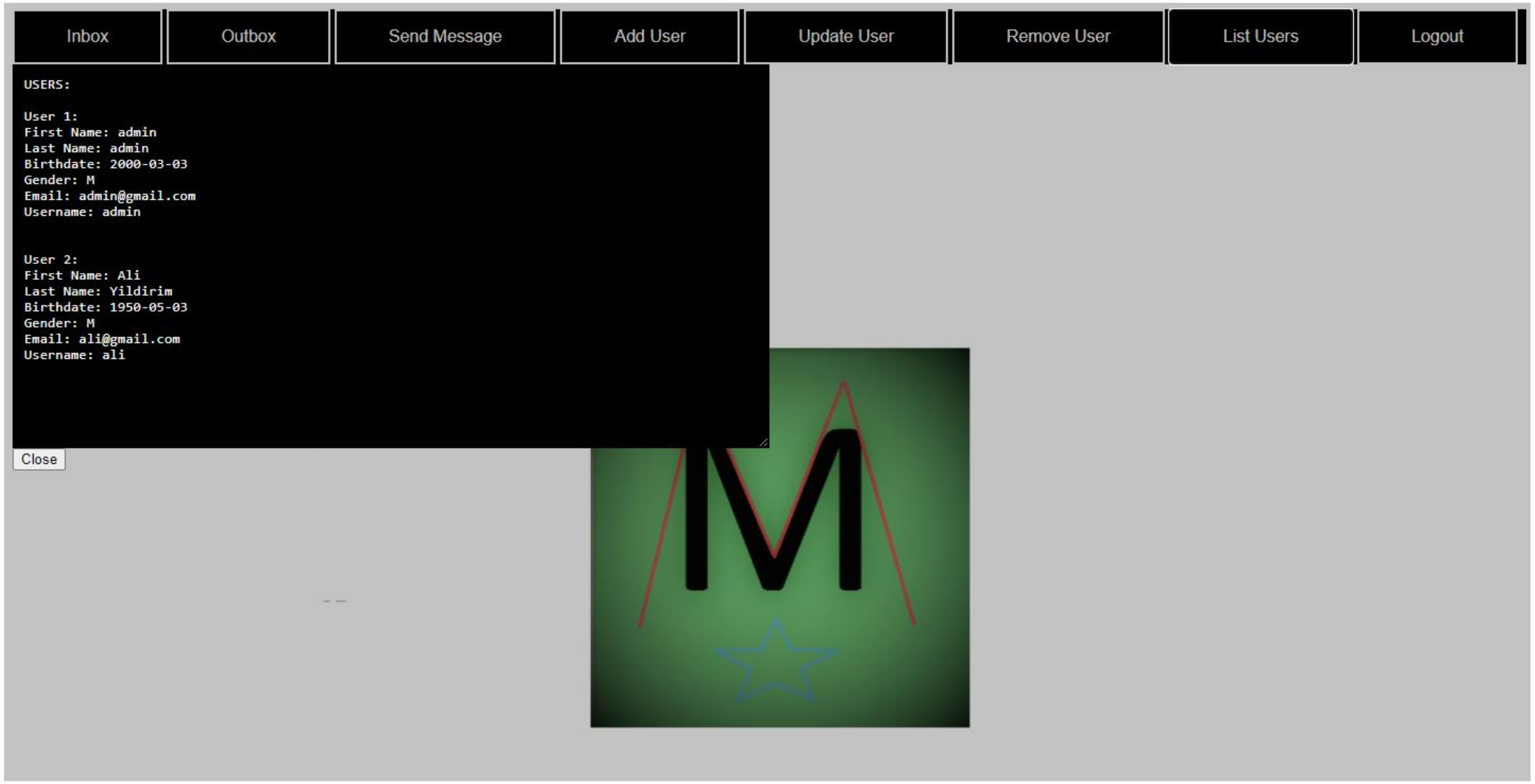
Data To Update (firstname, lastname, birthdate (YYYY/MM/DD), gender (M/F), email, username, password, admin (T/F)):

birthdate

New Data:

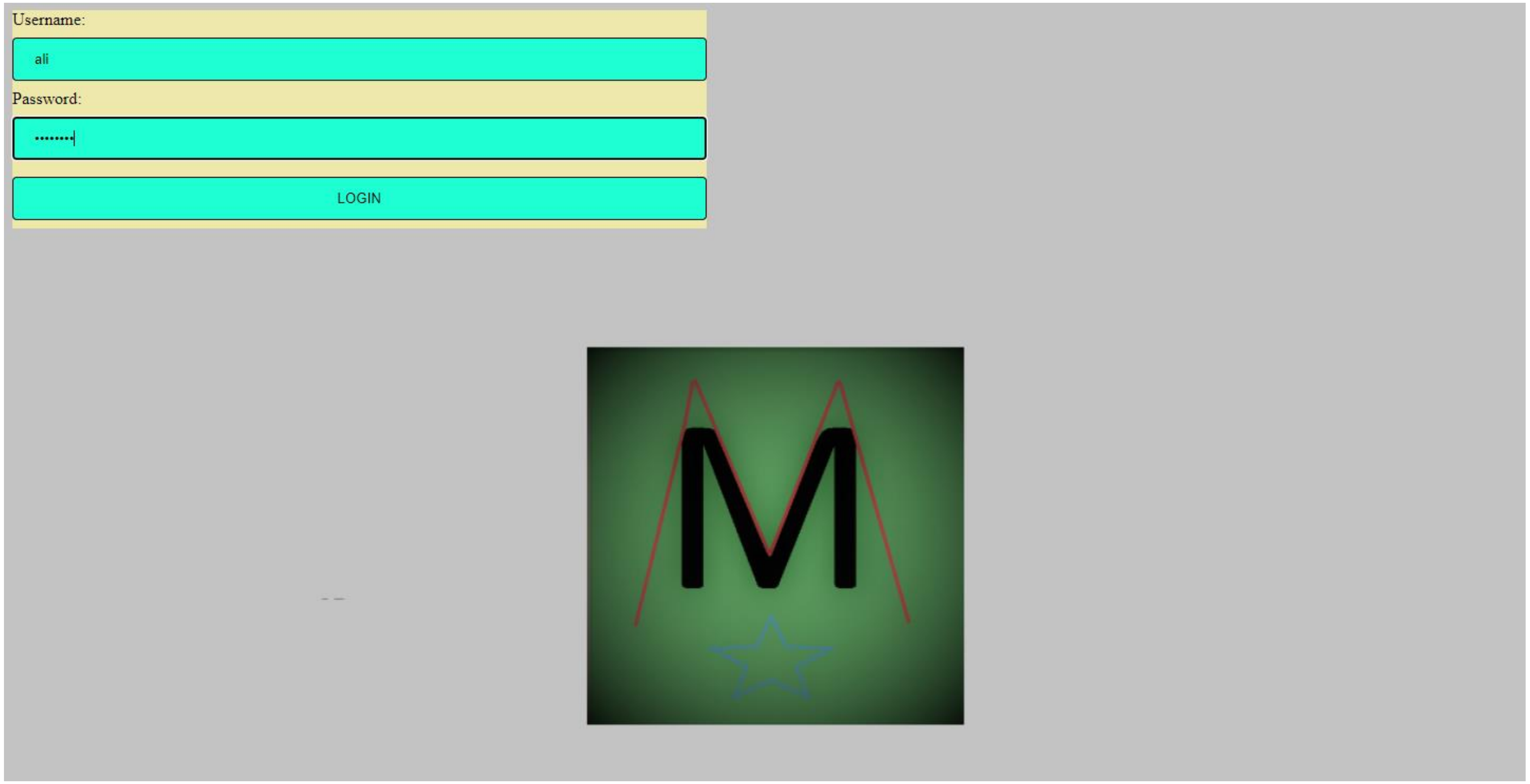
1950/05/03

Figure 13. The administrator screen of the Web-based Messaging Application for updating a user.



```
USERS:

User 1:
First Name: admin
Last Name: admin
Birthdate: 2000-03-03
Gender: M
Email: admin@gmail.com
Username: admin


User 2:
First Name: Ali
Last Name: Yildirim
Birthdate: 1950-05-03
Gender: M
Email: ali@gmail.com
Username: ali
```

Figure 14. The administrator screen of the Web-based Messaging Application after updating the birthdate of the user with username "ali". Only the administrator can update a user.


Username:

ali

Password:

••••••••

Figure 15. The Web-based Messaging Application interface showing the user with username “ali” is logging in.

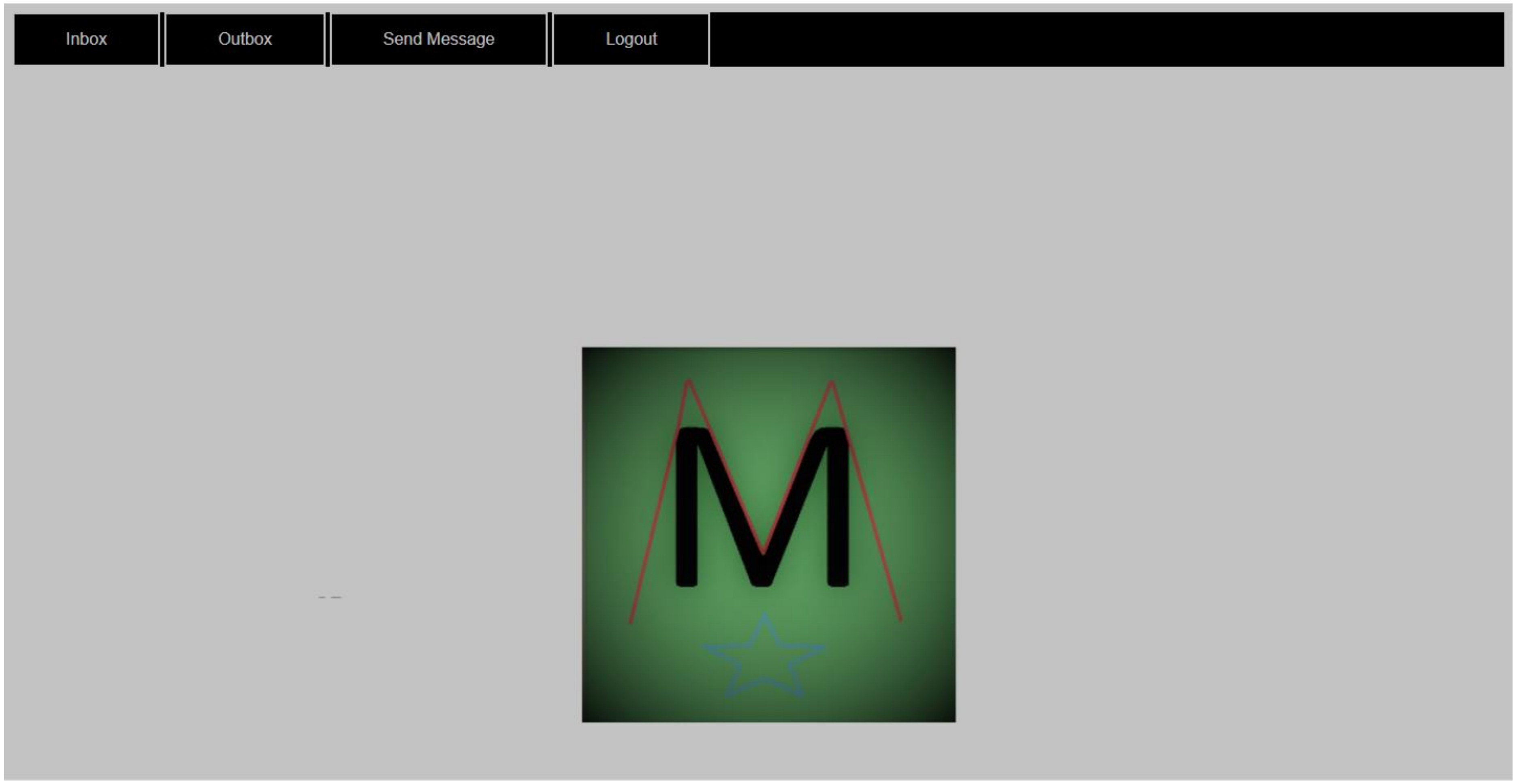
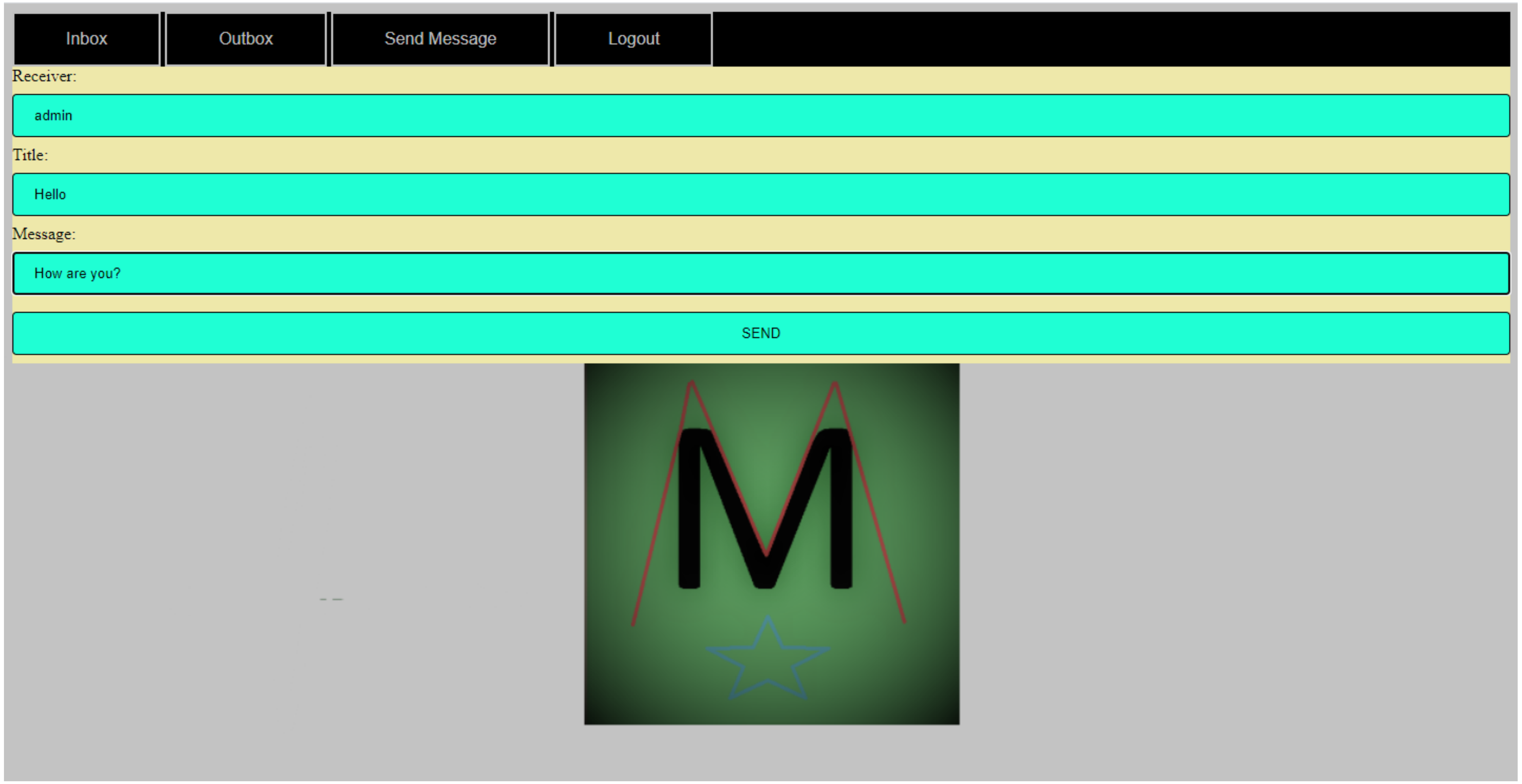


Figure 16. The user interface of the Web-based Messaging Application for users without administrator privileges.



Receiver:

admin

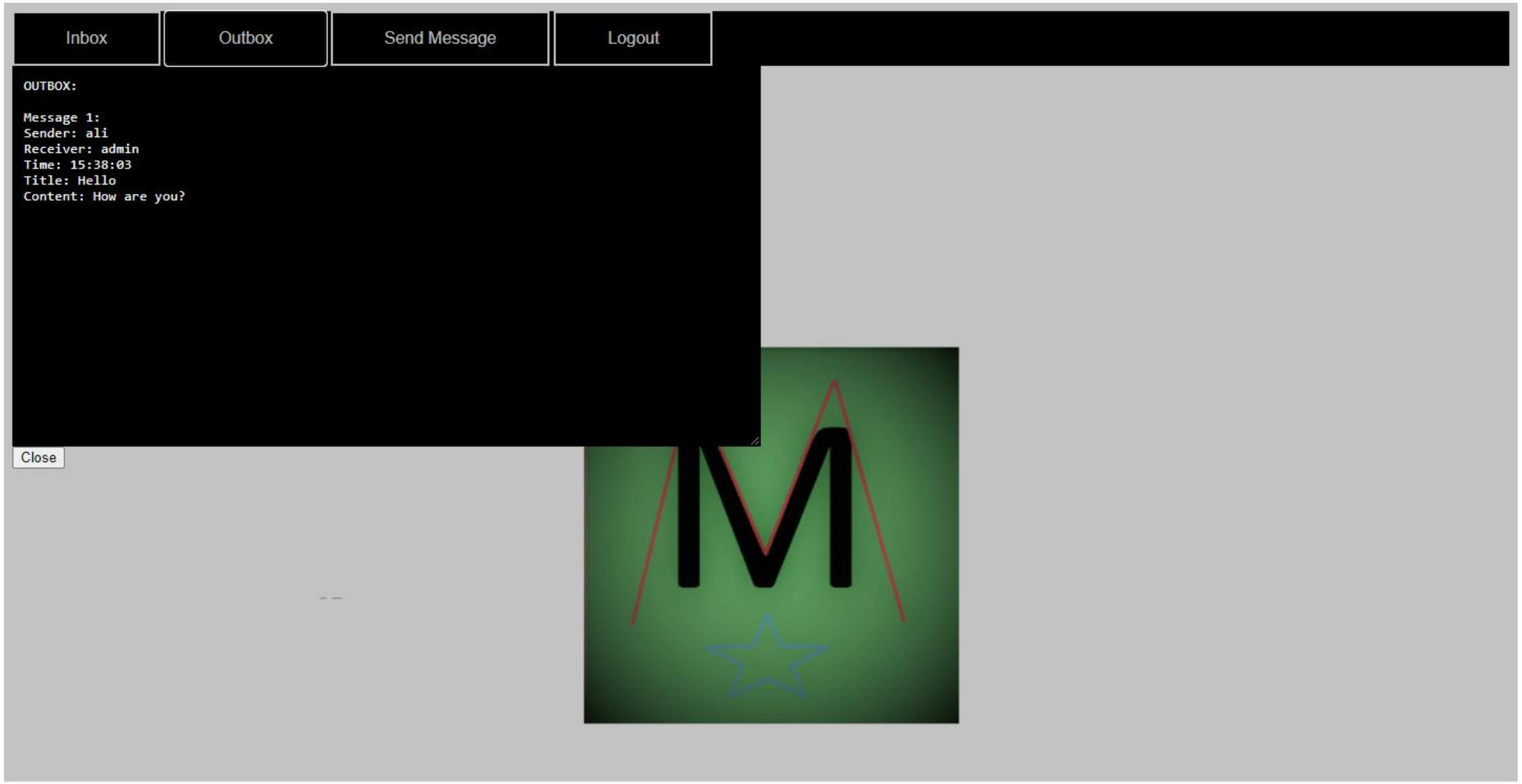
Title:

Hello

Message:

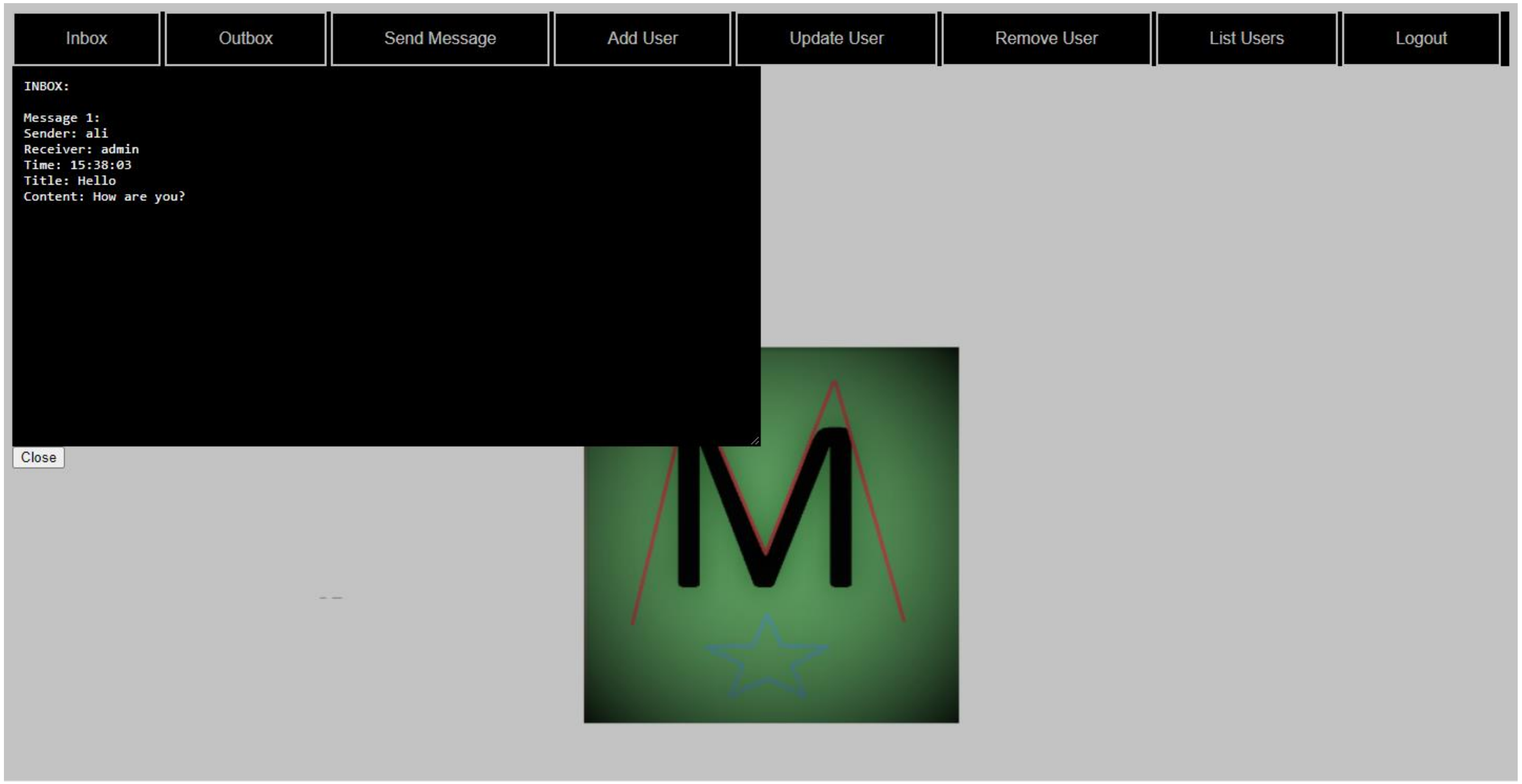
How are you?

Figure 17. The user interface of the Web-based Messaging Application for sending a message.



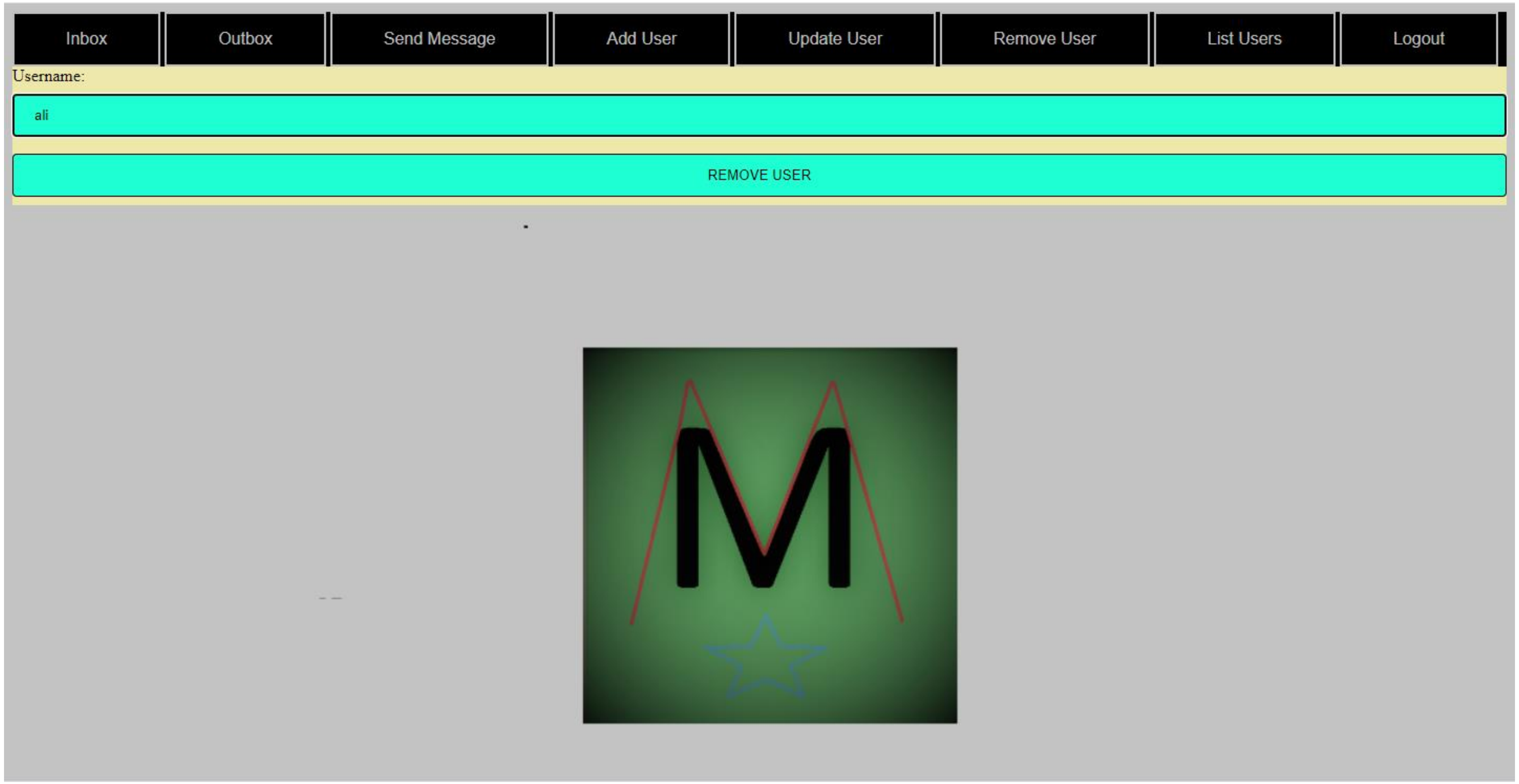
```
OUTBOX:

Message 1:
Sender: ali
Receiver: admin
Time: 15:38:03
Title: Hello
Content: How are you?
```

Figure 18. The user interface of the Web-based Messaging Application for seeing the Outbox of the user “ali” after sending a message.



```
INBOX:

Message 1:
Sender: ali
Receiver: admin
Time: 15:38:03
Title: Hello
Content: How are you?
```

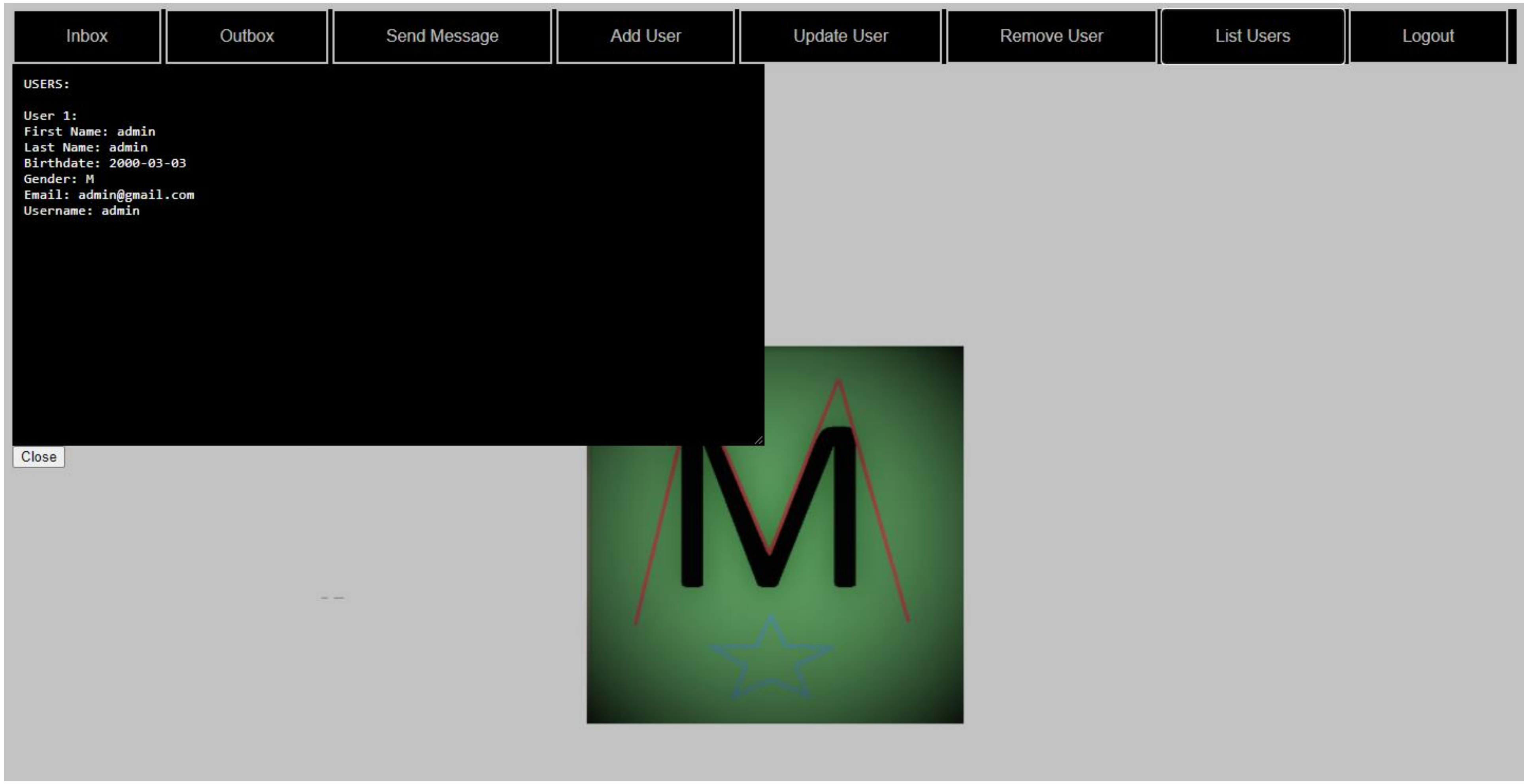
Figure 19. The user interface of the Web-based Messaging Application for seeing the Inbox of the user “admin” after receiving the message.



Username:

ali

Figure 20. The user interface of the Web-based Messaging Application for removing a user. Only the administrator can remove a user.



```
USERS:

User 1:
First Name: admin
Last Name: admin
Birthdate: 2000-03-03
Gender: M
Email: admin@gmail.com
Username: admin
```

Figure 21. The user interface of the Web-based Messaging Application after removing the user with username “ali”.

3.3 Third Assignment: Web-based User Management & Messaging Tool (Version 2)

This assignment had the same purpose as the previous assignments. Admins were able to perform create, read, update, delete operations on users, and users were able to message other users and read their inboxes and outboxes. This time different web technologies and a different database management system were used. Rather than SOAP web services, REST (Representational State Transfer) was used. REST implies statelessness. Therefore, the server does not need to know anything about the client's state or the client does not need to know anything about the state of the server [14]. This time there were no CORS problems because REST solved this problem. Therefore, different ports were used for network connection testing in this assignment.

The database used in this assignment was MongoDB. MongoDB is not a relational database; it is a document-based database. It stores data similar to JSON (JavaScript Object Notation) format [15]. JSON is a format to store data that is easy to read and write [16]. An application called Robo 3T was used to create the database, collections (tables in an RDBMS), and schemas.

The server was written in JavaScript using node.js and express.js libraries. Node.js was used for creating the server, and express.js was used for routing services. Node.js is a source for running a server outside a browser [17]. The server handled each request and gave a proper response. For each request, there was a JavaScript function written. Each function made a connection with the database to retrieve information. A JWT (JSON Web Token) was used to create a random token for each login attempt, and tokens were used by the client-side for each request. JWT is a standard used to transmit information in an encoded and secure way. JWT has a header, payload, and signature. The header is used to define the algorithm to encode the data. The payload is the data to be encoded, and the signature is a random 64-byte string [18].

The server was tested using the Postman application where get, post, delete, and update requests were made to the server and the received responses in JSON format were checked.

The client-side was written in typescript using Angular API. Angular is a platform made by Google for building web applications [19]. The user interface was better compared to the previous assignment because some libraries of Angular was used for graphical user interface components. For example, the table for displaying users and the tables for displaying messages had a sorting option in the column headers. Also, admins can delete or update the users when they open the list, and they do not need to remember their usernames, unlike the previous assignment. When admins update or add a user they perform the process on a user-friendly form. This assignment had an additional feature on browser information. Admins were able to see users' login information, the browser they used, and their IP address. I used Microsoft Visual Studio for editing and developing.

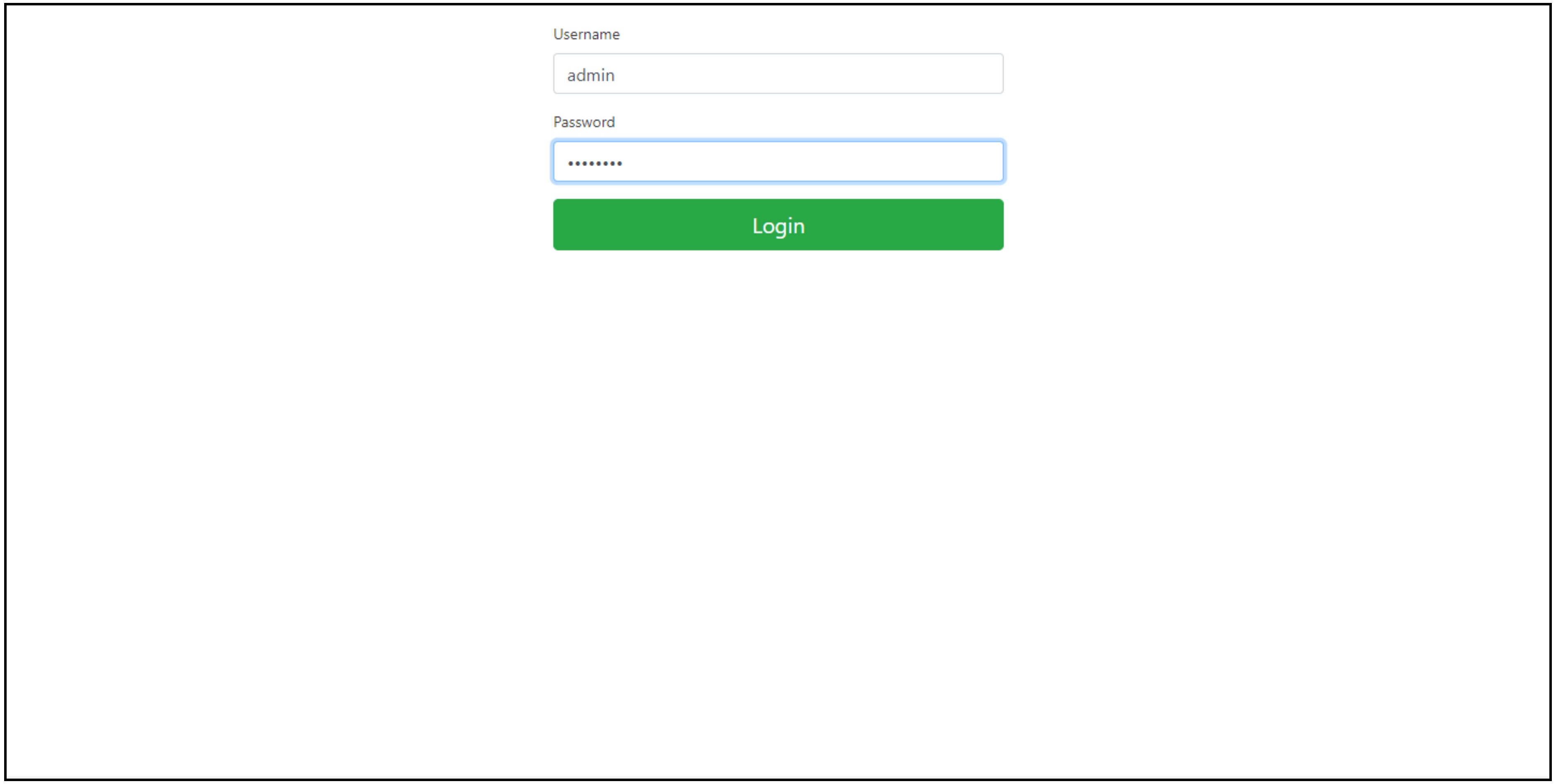


Figure 22. The login screen of the Web-based Messaging Application 2.

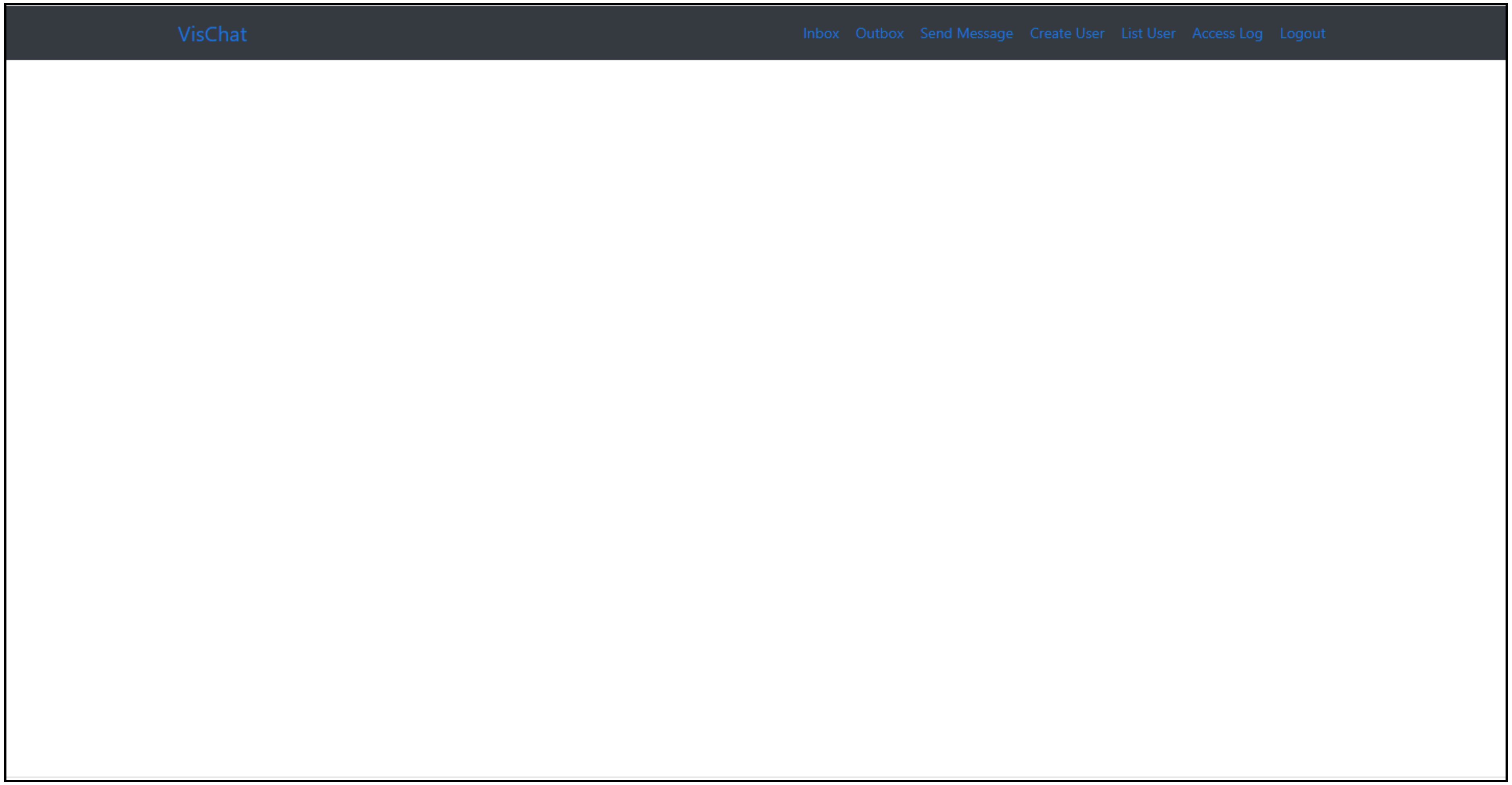
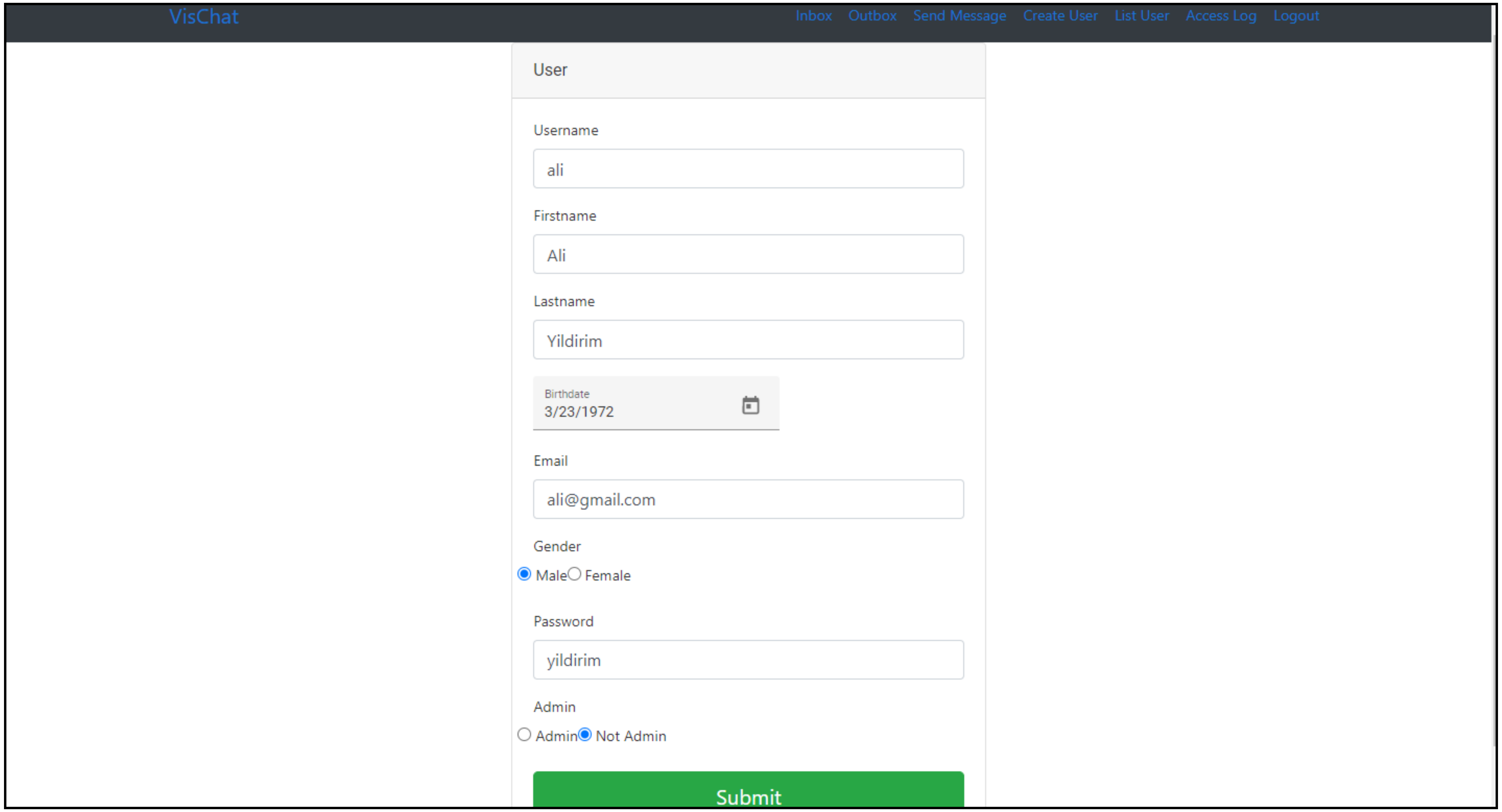


Figure 23. The administrator screen for the Web-based Messaging Application 2.



User

Username: ali

Firstname: Ali

Lastname: Yildirim

Birthdate: 3/23/1972

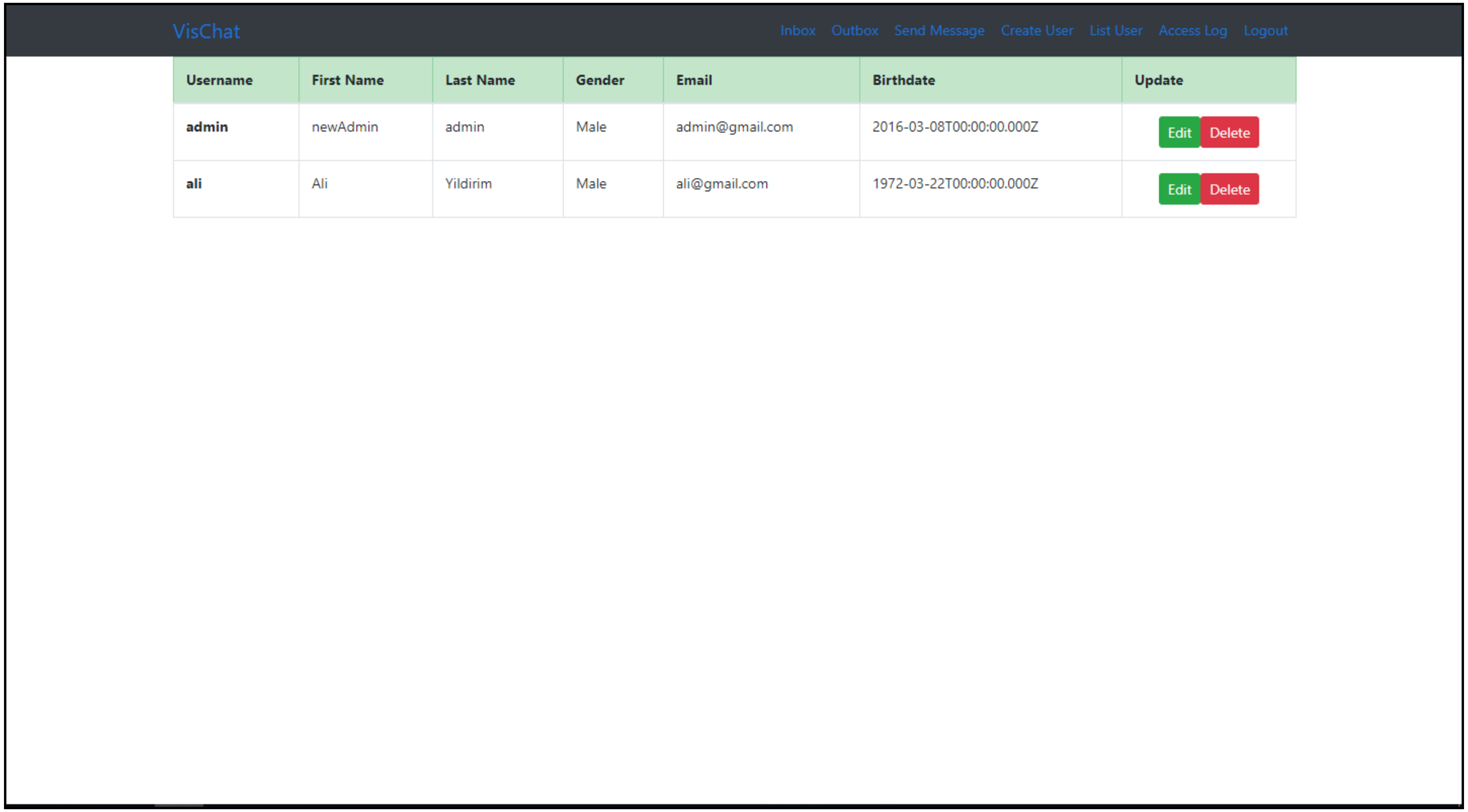
Email: ali@gmail.com

Gender: [✓ Male] Female

Password: yildirim

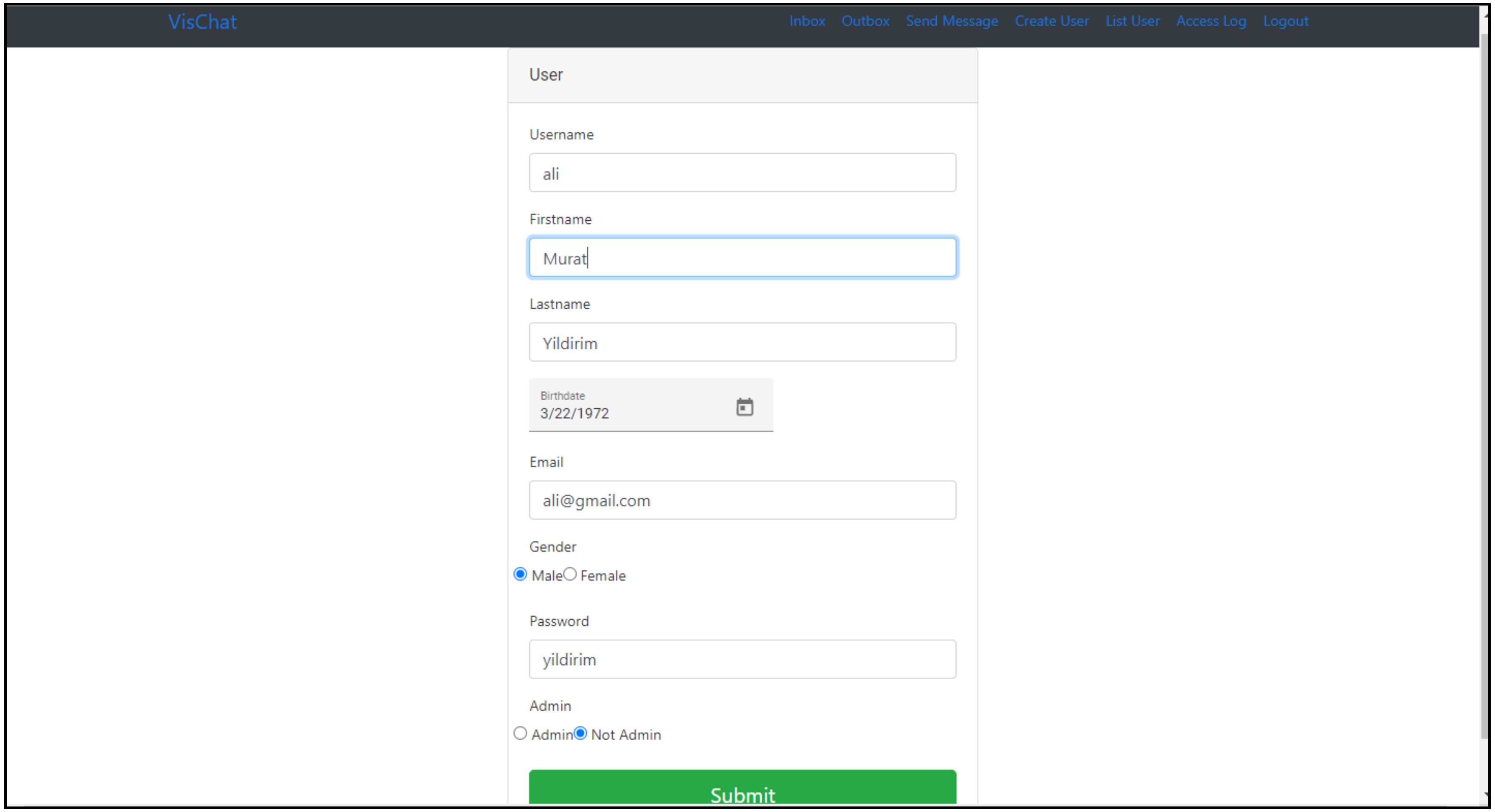
Admin: Admin [✓ Not Admin]

Figure 24. The administrator screen of the Web-based Messaging Application 2 for creating a new user.



Username	First Name	Last Name	Gender	Email	Birthdate	Update
admin	newAdmin	admin	Male	admin@gmail.com	2016-03-08T00:00:00.000Z	Edit Delete
ali	Ali	Yildirim	Male	ali@gmail.com	1972-03-22T00:00:00.000Z	Edit Delete

Figure 25. The administrator screen of the Web-based Messaging Application 2 for browsing users after creating the new user.



User

Username: ali

Firstname: Murat

Lastname: Yildirim

Birthdate: 3/22/1972

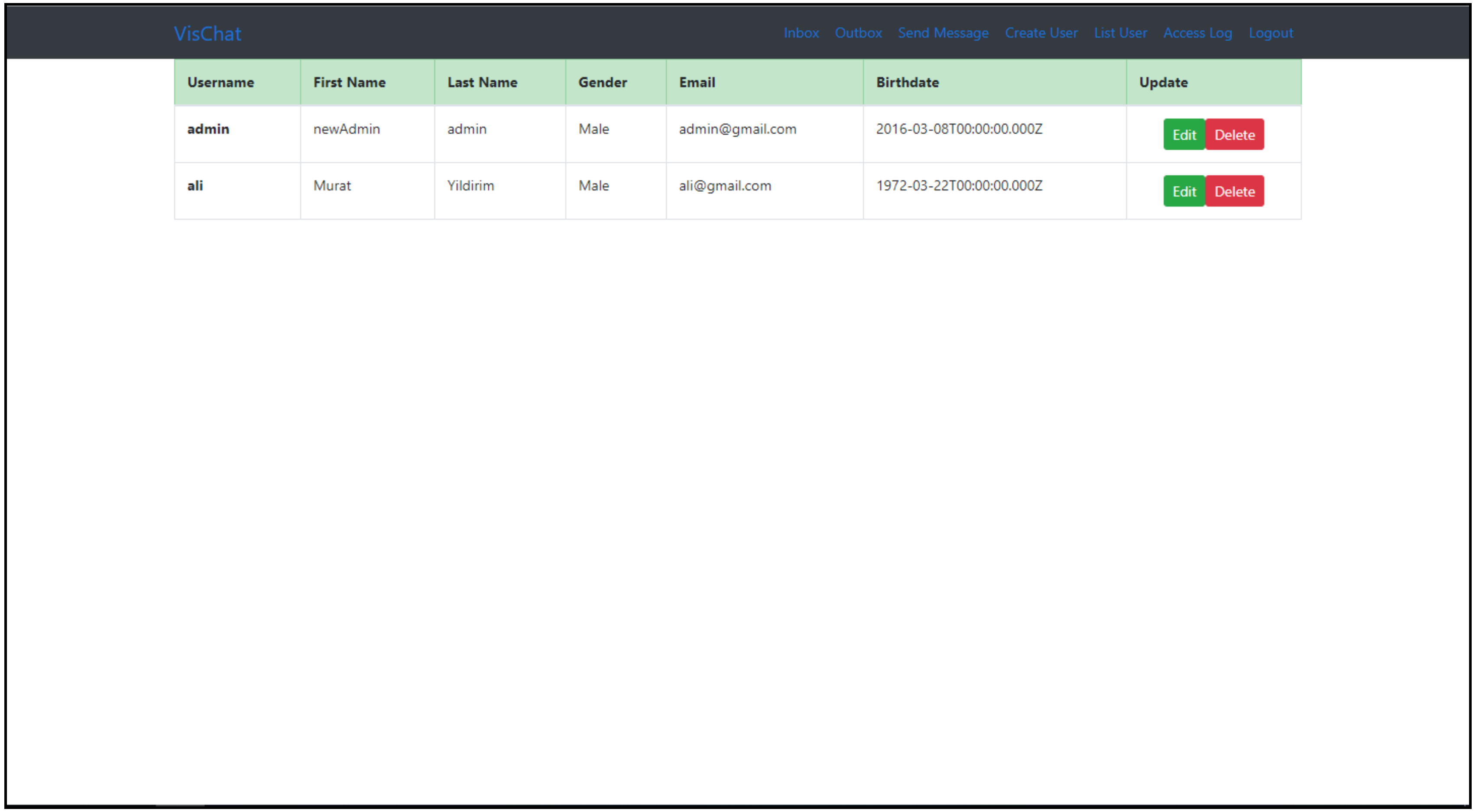
Email: ali@gmail.com

Gender: [✓ Male] Female

Password: yildirim

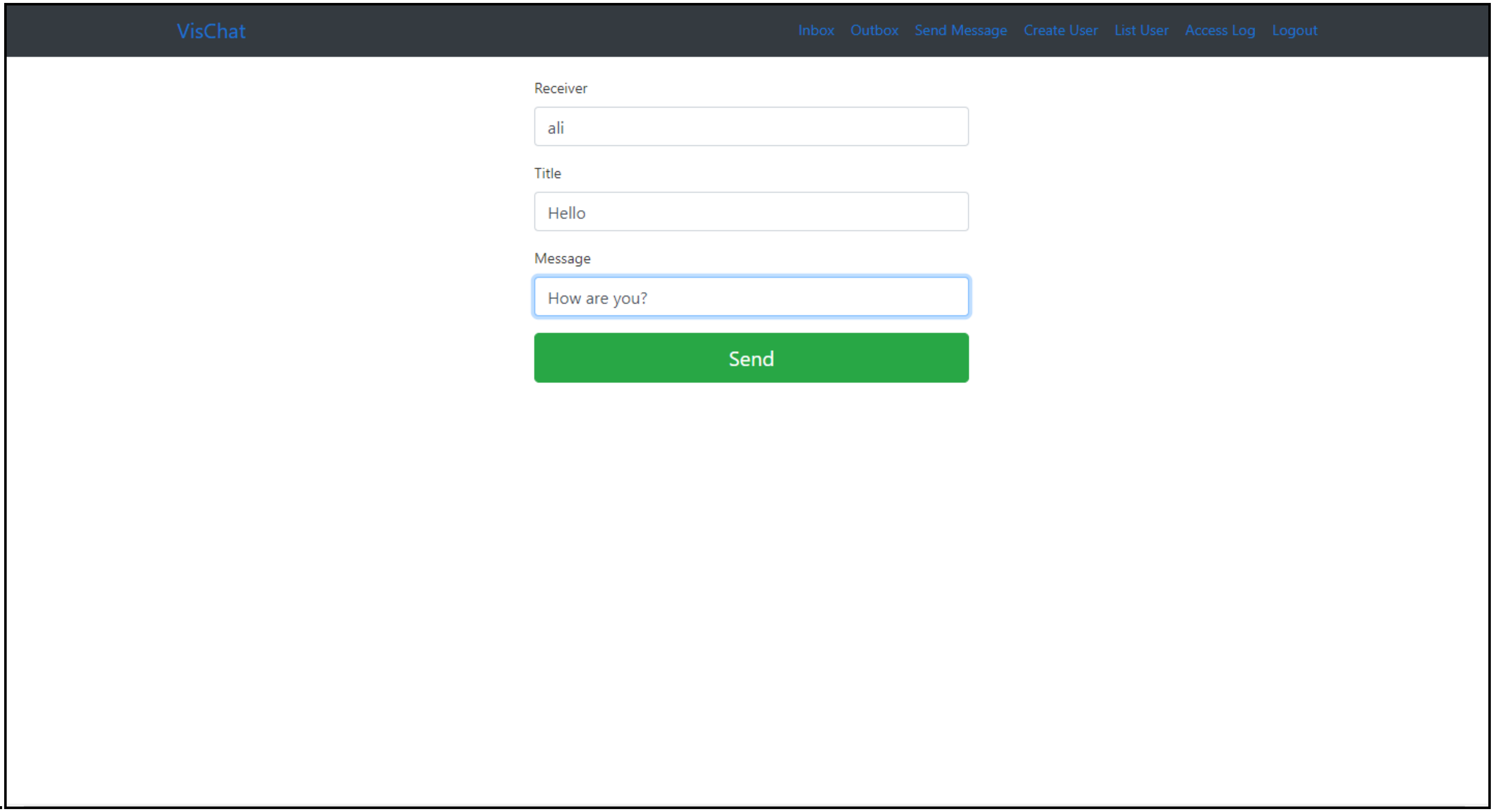
Admin: Admin [✓ Not Admin]

Figure 26. The administrator screen of the Web-based Messaging Application 2 for updating a user.


VisChat
Inbox Outbox Send Message Create User List User Access Log Logout

Username	First Name	Last Name	Gender	Email	Birthdate	Update
admin	newAdmin	admin	Male	admin@gmail.com	2016-03-08T00:00:00.000Z	Edit Delete
ali	Murat	Yildirim	Male	ali@gmail.com	1972-03-22T00:00:00.000Z	Edit Delete

Figure 27. The administrator screen of the Web-based Messaging Application 2 for browsing users after updating the name of the user “ali” as Murat.



Receiver

ali

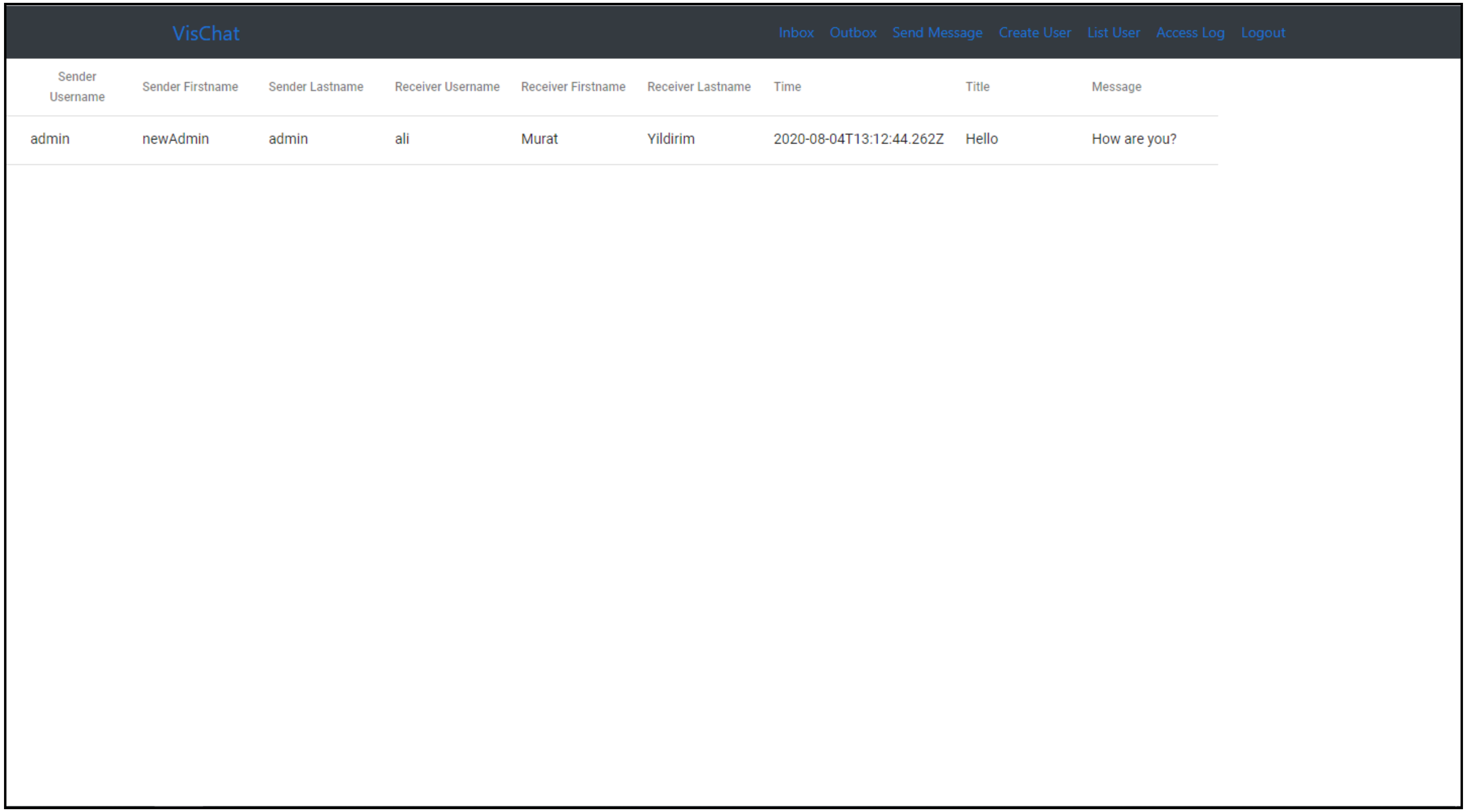
Title

Hello

Message

How are you?

Figure 28. The screen of the Web-based Messaging Application 2 for sending a message (the user is “admin”).



VisChat

Inbox Outbox Send Message Create User List User Access Log Logout

Sender Username	Sender Firstname	Sender Lastname	Receiver Username	Receiver Firstname	Receiver Lastname	Time	Title	Message
admin	newAdmin	admin	ali	Murat	Yildirim	2020-08-04T13:12:44.262Z	Hello	How are you?

Figure 29. The user interface of the Web-based Messaging Application 2 for seeing the Outbox of the user “admin” after sending the message.

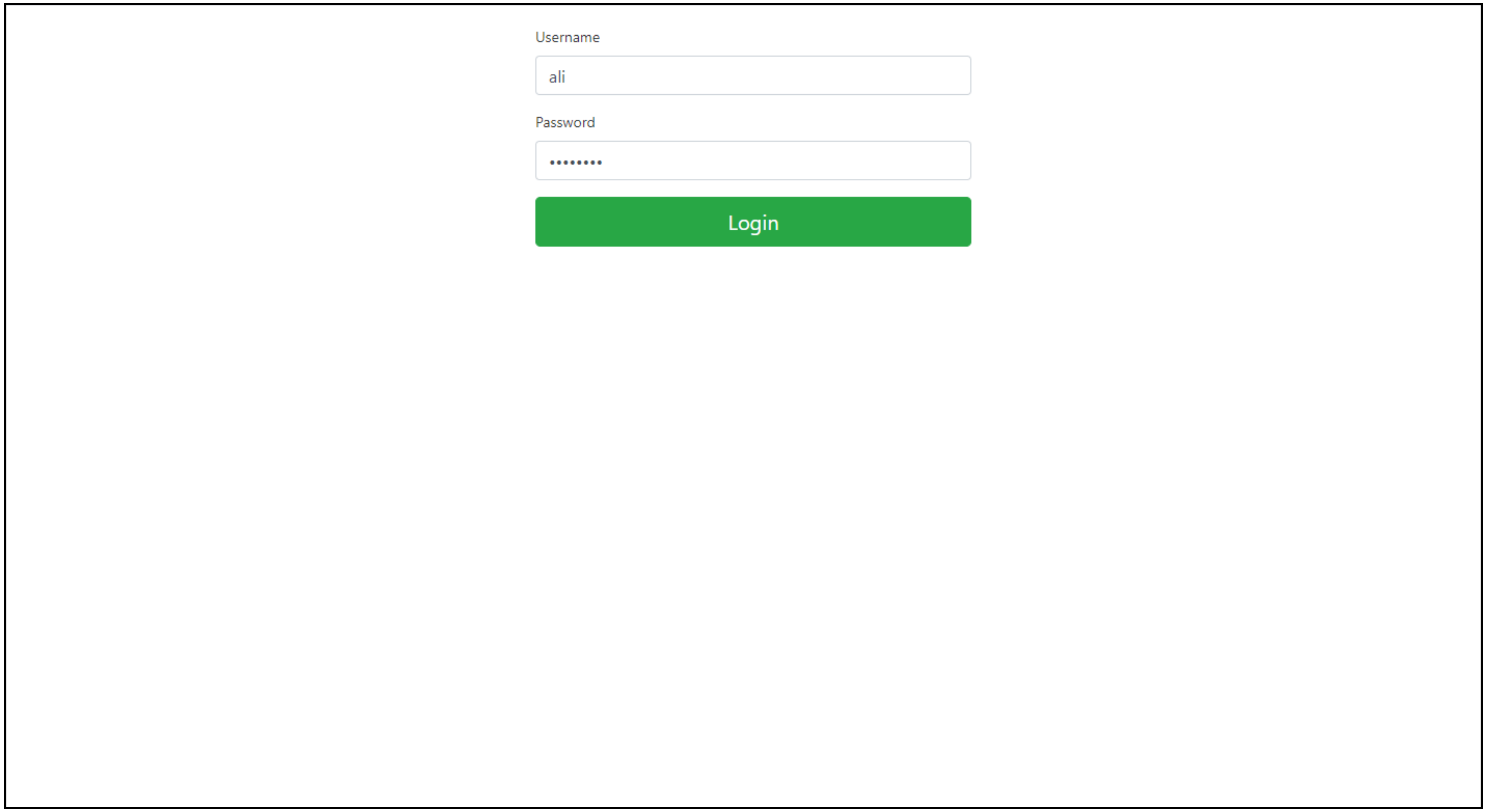
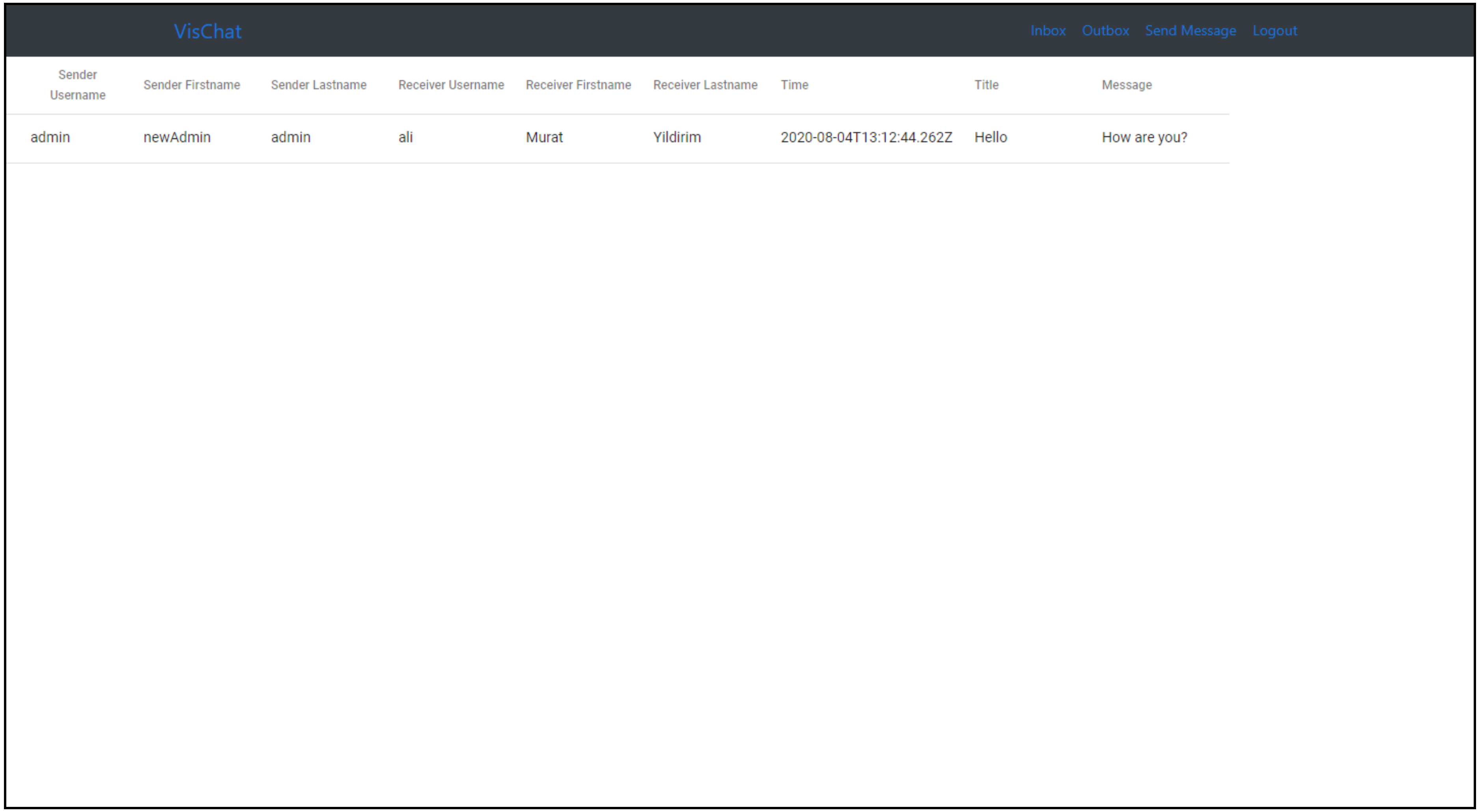
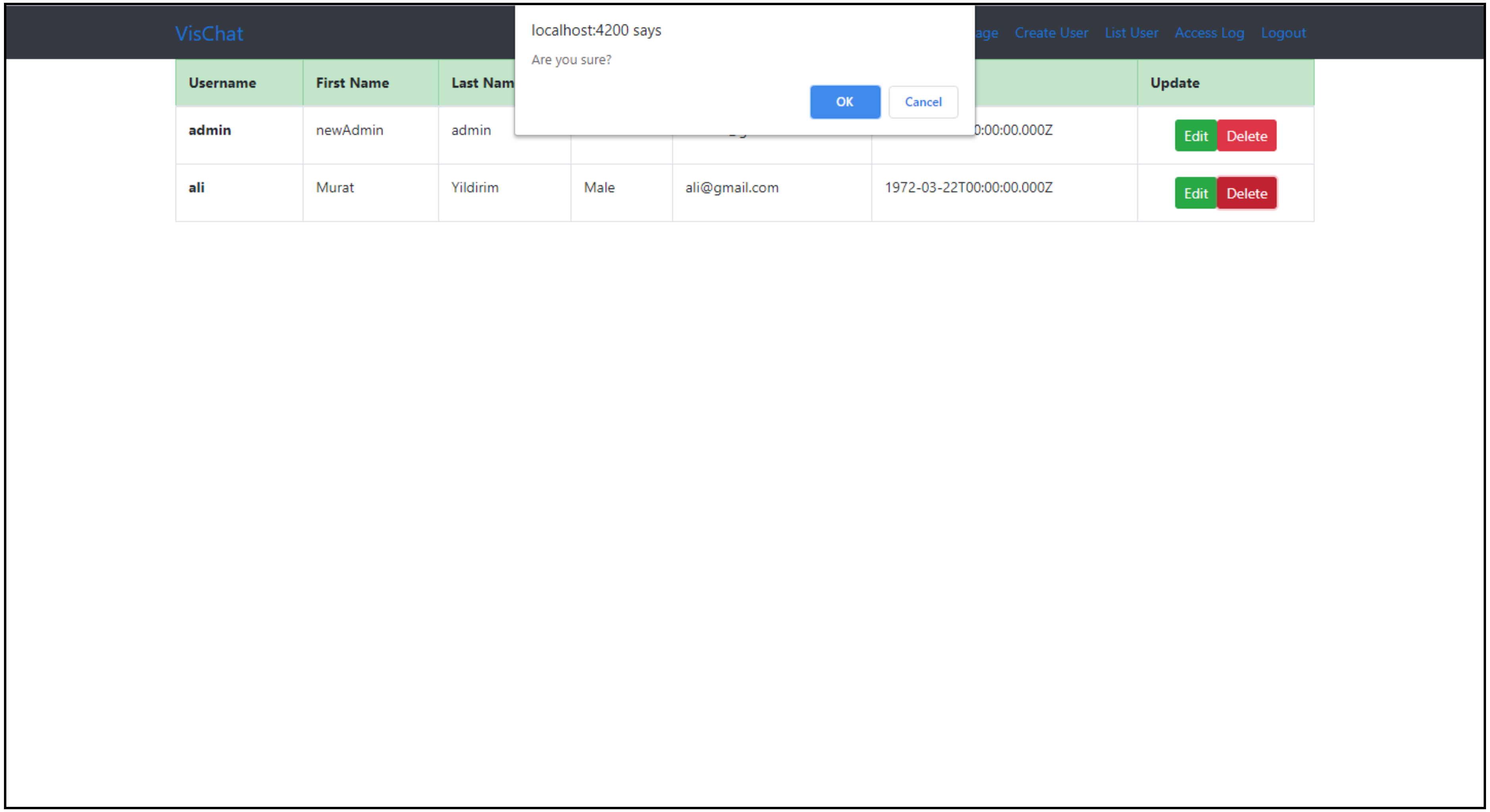


Figure 30. The Web-based Messaging Application 2 interface showing the user with username “ali” is logging in.



Sender Username	Sender Firstname	Sender Lastname	Receiver Username	Receiver Firstname	Receiver Lastname	Time	Title	Message
admin	newAdmin	admin	ali	Murat	Yildirim	2020-08-04T13:12:44.262Z	Hello	How are you?

Figure 31. The user interface of the Web-based Messaging Application 2 for seeing the Inbox of the user “ali”.

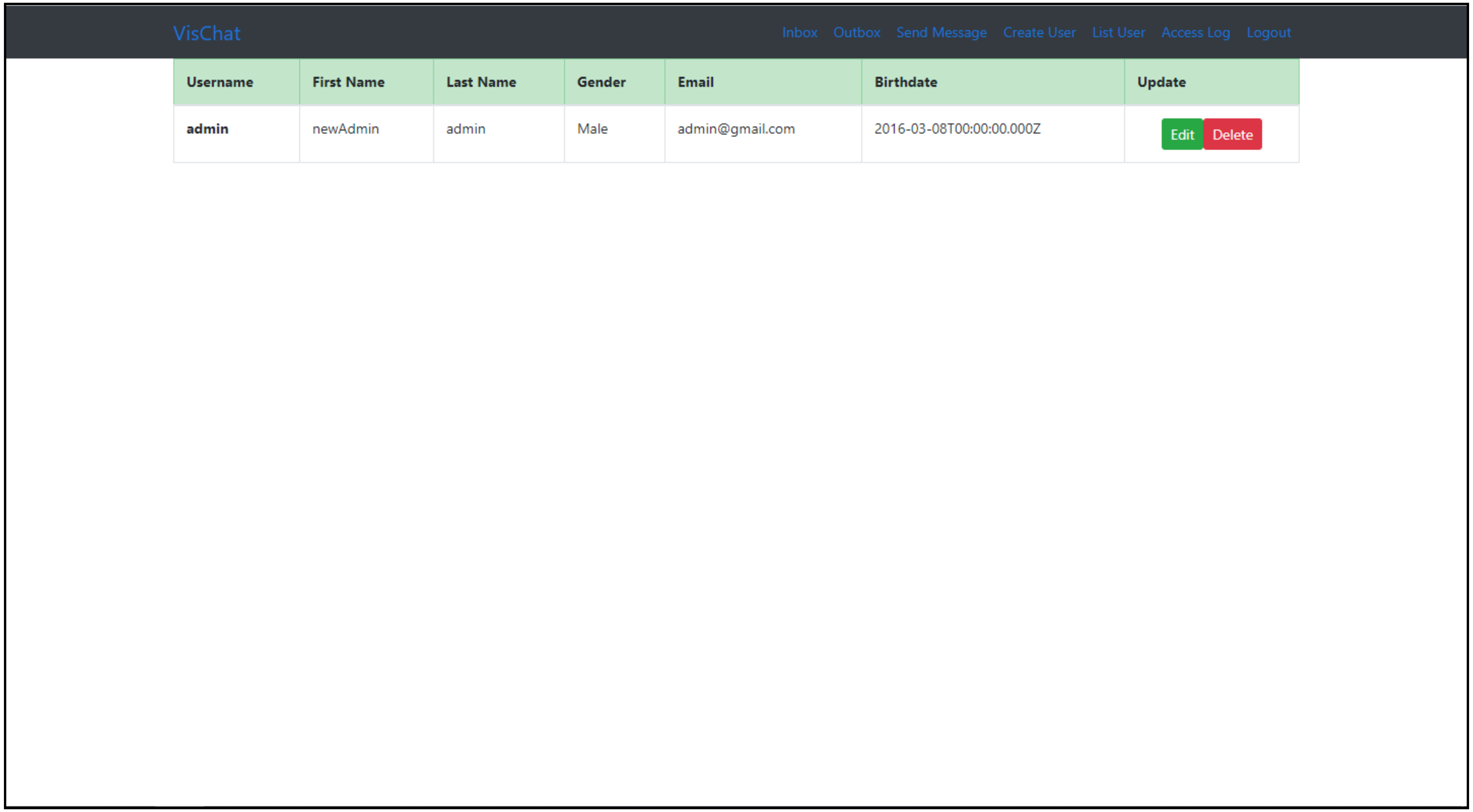


localhost:4200 says

Are you sure?

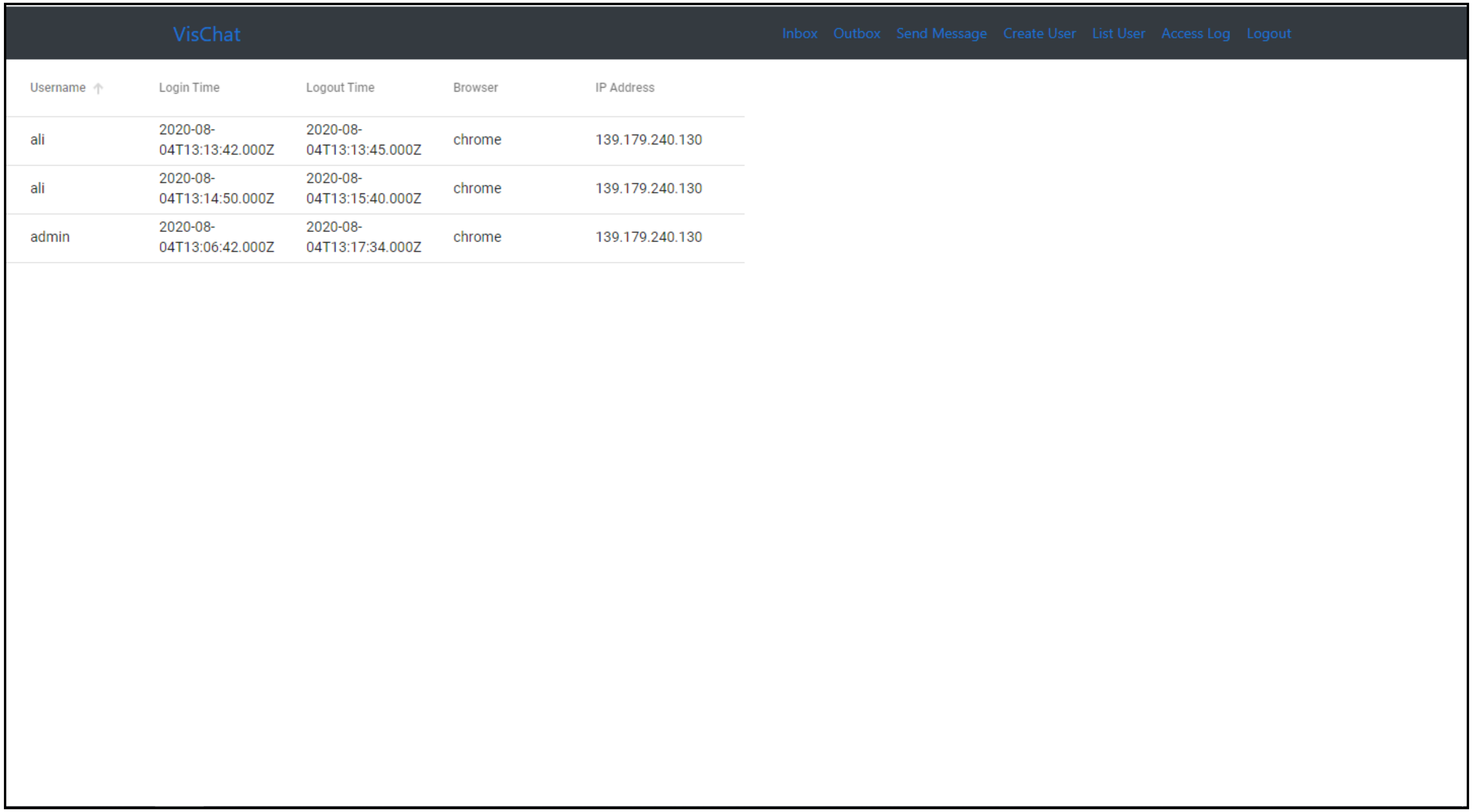
Username	First Name	Last Nam				Update
admin	newAdmin	admin			0:00:00.000Z	Edit Delete
ali	Murat	Yildirim	Male	ali@gmail.com	1972-03-22T00:00:00.000Z	Edit Delete

Figure 32. The administrator screen for the Web-based Messaging Application 2 during the deletion of a user.



Username	First Name	Last Name	Gender	Email	Birthdate	Update
admin	newAdmin	admin	Male	admin@gmail.com	2016-03-08T00:00:00.000Z	Edit Delete

Figure 33. The administrator screen for the Web-based Messaging Application 2 after the deletion of the user with username “ali”.



Username	Login Time	Logout Time	Browser	IP Address
ali	2020-08-04T13:13:42.000Z	2020-08-04T13:13:45.000Z	chrome	139.179.240.130
ali	2020-08-04T13:14:50.000Z	2020-08-04T13:15:40.000Z	chrome	139.179.240.130
admin	2020-08-04T13:06:42.000Z	2020-08-04T13:17:34.000Z	chrome	139.179.240.130

Figure 34. The administrator screen for the Access Log of the Web-based Messaging Application 2.

4 Performance and Outcomes

4.1 Solving Complex Engineering Problems

Throughout the internship, the purpose was to create user management and messaging applications in an efficient and user-friendly way. The first application used a command-line interface so it was not user-friendly. However, there were not any connection problems. In the first assignment, I used Java. In this part, I applied my Java knowledge that I learned from CS-101 and CS-102 courses. At first, I made some algorithms to parse the received strings on both the client-side and server-side because they were separated with a “|” protocol. This made the code longer but later my advisor recommended using built-in split functions in Java.

In the second assignment, the server was developed using the Spring framework. I had a hard time using Maven in this part because it was my first experience. However, in CS-102 we used Intellij IDEA for our project so I was familiar with those tools. I also used my CS-101 and CS-102 knowledge in the first part of the second assignment because I wrote the server in Java.

In the third assignment, I used lots of different libraries, applications, and frameworks which were quite challenging. I did not have prior experience with any of the technologies I used in this part. Therefore, I can say that I only used my general programming knowledge that I acquired in school in this assignment.

4.2 Ethical and Professional Responsibilities

All our assignments had deadlines and at the end of each assignment, I showed my work to my advisor via Zoom. My advisor gave feedback and, I changed my programs according to the given feedback. We communicated via Slack for discussing and asking questions about the assignments. I realized that a deadline was crucial because there was an internship schedule and, everything was following another task.

4.3 Making Informed Judgments

In all assignments, I designed my user interface according to usability. In the first assignment, although I used the command-line interface I wanted the user to access all actions with easy commands so all actions were defined with a letter and inputs necessary for each action were simple and user-friendly. In the second assignment, it was easier to design the website in a user-friendly manner because I used JavaScript, HTML, and CSS. I put the menu at the top of the page to give more space to display information for other actions. However, the user did not have the chance to edit or remove a user from the users’ list, they needed to remember the user’s name to take those actions. This problem was fixed in the third assignment. This user list had buttons near each person for editing and removing.

Besides, data security is an important topic these days. The first assignment did not manage data security but in the second and third assignments tokens were used to exchange information with the server. JWT was used in the third assignment to have more secure data transfer.

4.4 Acquiring New Knowledge by Using Appropriate Learning Strategies

For the first assignment, I learned socket programming. I studied this topic from Geeks for Geeks. When I was learning database management systems for the first assignment I had a hard time because I did not have any experience with it. I solved this problem by watching tutorials from

YouTube and studying relational database management systems from W3Schools Online Web Tutorials.

In the second assignment, I used the Spring framework, SOAP web services, and AJAX for developing the server, and the client and I did not have any prior knowledge of it. I solved this problem by studying SOAP from W3Schools Online Web Tutorials and reading two tutorials recommended by my advisor:

1. Producing a SOAP Web Service With Spring https://spring.io/guides/gs/producing-web-service/,

2. Consuming a SOAP Web Service With Spring https://spring.io/guides/gs/consuming-web-service/.

Besides, I learned JavaScript, HTML, and CSS in this part. I read w3schools about these languages before I started this assignment. In the same assignment, I experienced the CORS problem as I mentioned before. I did not know this issue, so it took me a while to figure out the reason for not connecting to the website. I solved this CORS problem by reading similar problems in Stack Overflow. In this part, I used JavaScript and HTML which was very hard for me because solving problems in these kinds of programming languages is difficult compared to other languages like Java and Python. I used the inspect feature of the browser to look more into the XML requests made and the responses received.

In the third assignment, I learned REST web services. In this assignment, I used JavaScript and TypeScript. I needed a deep understanding of these programming languages because I faced lots of problems during this assignment. I needed to learn MongoDB to use in the server part as a document-based database. Document-based databases were different than relational databases. I used w3schools to understand MongoDB and I used my knowledge on relational databases to understand the system of document-based databases.

Before I started I reviewed a tutorial named: Tutorial for a full MEAN (MongoDB, Express, Angular, Node.js) stack example using Bootstrap

https://www.positronx.io/mean-stack-tutorial-angular-7-crud-bootstrap/

Creating the server was a difficult job but I used Postman API to test the requests made and responses received before I moved to the client-side. On the client-side I faced problems when I was writing in TypeScript because solving syntax problems in this language was quite hard for me. I searched similar errors in Stack Overflow when I faced syntax errors. Also, Angular had lots of libraries but using them was difficult. I used several YouTube videos to design features for my website.

4.5 Applying New Knowledge As Needed

For the first assignment, I learned relational database management systems and I used this knowledge to create a database in PostgreSQL that had two tables in it to store users and messages. In the second assignment, I studied the Spring framework and SOAP web services and I used this knowledge to create a server and consume it using SOAP web services. Also, in this part, I used my JavaScript and HTML knowledge I learned new to create a website for the client-side. In the third assignment, I used my knowledge on REST web services to create a protocol and a server. I used my knowledge on document-based database systems and MongoDB to create a database in MongoDB to store users, messages, and login information for the third assignment.

5 Conclusions

In this internship, I learned important information on web development, computer networks, and database management systems. I wrote huge amounts of code in different languages and developed useful desktop and web applications. My first assignment was making user management and messaging tool using a command-line interface. I learned database management and socket programming in this assignment. My second assignment was creating a server using Spring Boot and a web client application using HTML, JavaScript, and CSS. I learned new languages like HTML, JavaScript, XML, and new protocols like SOAP which was quite challenging for me. In my third assignment, I used a new database called MongoDB for storing and retrieving information. Besides, I used Node.js and Express for developing a server and Angular API for developing a web client. This assignment was the most challenging one for me because the project was so big and there were additional features to add. Overall, the internship was very educational and I gained lots of experience in web development and database management systems. I was happy with my internship at the end.

References

[1] “SRDC (Software Research & Development Consultancy).” www.srdc.com.tr/en/. [Accessed: Sep 20, 2020].

[2] “Completed Projects.” SRDC. www.srdc.com.tr/completed/. [Accessed: Sep 20, 2020].

[3] “Mustafa Yüksel.” SRDC. www.srdc.com.tr/staff-member/mustafa/. [Accessed: Sep 20, 2020].

[4] “Introducing Threads in Socket Programming in Java.” GeeksforGeeks, 8 Feb. 2018, www.geeksforgeeks.org/introducing-threads-socket-programming-java/. [Accessed: Sep 20, 2020].

[5] “XML Soap.” w3schools, www.w3schools.com/xml/xml_soap.asp. [Accessed: Sep 20, 2020].

[6] “XML Tutorial.” w3schools, www.w3schools.com/xml/default.asp. [Accessed: Sep 20, 2020].

[7] “XML Schema Tutorial.” w3schools, www.w3schools.com/xml/schema_intro.asp. [Accessed: Sep 20, 2020].

[8] “Lesson: Introduction to JAXB.” The Java™ Tutorials, Java Architecture for XML Binding (JAXB)), https://docs.oracle.com/javase/tutorial/jaxb/intro/index.html. [Accessed: Sep 20, 2020].

[9] “Accelerating API Quality Through Testing.” SoapUI, https://www.soapui.org/. [Accessed: Sep 20, 2020].

[10] “Getting Started.” MDN Web Docs, https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started. [Accessed: Sep 20, 2020].

[11] “XML HTTP Request.” w3schools, https://www.w3schools.com/xml/xml_http.asp. [Accessed: Sep 20, 2020].

[12] “XML Parser.” w3schools, https://www.w3schools.com/xml/xml_parser.asp. [Accessed: Sep 20, 2020].

[13] “Cross-Origin Resource Sharing (CORS).” MDN Web Docs, https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Accessed: Sep 20, 2020].

[14] “REST.” REST API Tutorial, restfulapi.net/. [Accessed: Sep 20, 2020].

[15] “MongoDB Tutorial.” Tutorialspoint, www.tutorialspoint.com/mongodb/index.htm. [Accessed: Sep 20, 2020].

[16] “Introducing JSON.” JSON, www.json.org/json-en.html. [Accessed: Sep 20, 2020].

[17] “Node.js Introduction.” w3schools, www.w3schools.com/nodejs/nodejs_intro.asp. [Accessed: Sep 20, 2020].

[18] “Introducing JSON.” JSON, www.json.org/json-en.html. [Accessed: Sep 20, 2020].

[19] Angular. https://angular.io. [Accessed: Sep 20, 2020].

Appendices

A. First Assignment: User Management & Messaging Tool

```
//*******************************************************
//                   MUSTAFA GÖKTAN GÜDÜKBAY             *
// SRDC Software Research & Development and Consultancy Corp. *
//*******************************************************

// Java implementation of User class

public class User {

    //properties
    private String name;
    private String lastName;
    private String birthDate;
    //gender: 0 for invalid, M for male, F for female
    private char gender;
    private String email;
    private String username;
    private String password;
    private boolean isAdmin;

    //constructors
    public User(){
        name = "";
        lastName = "";
        birthDate = "";
        gender = '0';
        email = "";
        username = "";
        password = "";
        isAdmin = false;
    }

    // input is in the form
    // |ADDUSER|firstname|lastName|birthdate|gender|email
    // |username|passwordcolumn|isAdmin|

    public User(String input){
        String[] words = input.split("\\|");
        name = words[1];
        lastName = words[2];
        birthDate = words[3];
        gender = words[4].charAt(0);
        email = words[5];
        username = words[6];
        password = words[7];

      if(words[8].charAt(0) == 'T' || words[8].charAt(1) == 't')
            isAdmin = true;
        else
            isAdmin = false;
    }
```

```
public User(String name, String lastName, String birthDate,
             char gender, String email, String username, String
             password, boolean isAdmin) {
        this.name = name;
        this.lastName = lastName;
        this.birthDate = birthDate;
        this.gender = gender;
        this.email = email;
        this.username = username;
        this.password = password;
        this.isAdmin = isAdmin;
    }


    // methods

    // get methods

    public String getName() {
        return name;
    }

    public String getLastName(){
        return lastName;
    }

    public String getBirthDate() {
        return birthDate;
    }

    public char getGender() {
        return gender;
    }

    public String getEmail() {
        return email;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public boolean isAdmin(){
        return isAdmin;
    }

    //set methods

    public void setName(String name) {
        this.name = name;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public void setBirthDate(String birthDate) {
        this.birthDate = birthDate;
    }

    public void setGender(char gender) {
        this.gender = gender;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void setAdmin(boolean isAdmin){
        this.isAdmin = isAdmin;
    }

    public void welcomeUser(){
        System.out.println("\nWelcome " + name + " " + lastName + ".\n");
    }
}
```

```
// Java implementation of Message class

public class Message {

    // properties

    private String sender;
    private String receiver;
    private String time;
    private String title;
    private String content;


    // constructor

    public Message(){
        sender = "";
        receiver = "";
        time = "";
        title = "";
        content = "";
    }

    public Message(String sender, String receiver, String time,
                   String title, String message){
        this.sender = sender;
        this.receiver = receiver;
        this.time = time;
        this.title = title;
        this.content = message;
    }

    //methods

    public String getSender() {
        return sender;
    }

    public void setSender(String sender) {
        this.sender = sender;
    }

    public String getReceiver() {
        return receiver;
    }

    public void setReceiver(String receiver) {
        this.receiver = receiver;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getContent() {
        return content;
    }
    public void setContent(String content){
        this.content = content;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }
}
```

```
// Java implementation of Client side

import java.io.*;
import java.net.*;
import java.util.Scanner;

// Client class

public class Client
{
    public static void main(String[] args) throws IOException
    {
      // variables

      boolean loggedIn;
      User user;
      int index1;
      String temp;
      int index2;
      String received;
      String[] words;
      String un;
      String pw;

      // code

      loggedIn = false;

      user = new User();
      try
      {
          Scanner scn = new Scanner(System.in);

          // getting localhost ip
          InetAddress ip = InetAddress.getByName("localhost");

          // establish the connection with server port 5056
          Socket s = new Socket(ip, 5056);

          // obtaining input and out streams
          DataInputStream dis = new DataInputStream(s.getInputStream());
          DataOutputStream dos = new DataOutputStream(s.getOutputStream());

          // the following loop performs the exchange of
          // information between client and client handler

          while (true)
          {
              if(!loggedIn){
                  System.out.println("\nYou are not logged in. \n" +
                        "If you want to log in enter \"login\n"
                        + "Enter 'E' or 'e' for exiting the system.");
              }
              else{
                  System.out.println("\nYou are logged in. " +
                       "Enter:\n\t\t'L' or 'l' for logging out, " +
                       "\n\t\t'I' or 'i' for Inbox, " +
                       "\n\t\t'O' or 'o' for Outbox," +
                       "\n\t\t'S' or 's' for Sending Messages. " +
                       "\n\t\t'E' or 'e' for exiting the system.\n");
```

```
            if(user.isAdmin()){
                System.out.println("You are an admin. " +
                    "Enter: \n\t\t'A' or 'a' for adding a new user," +
                    "\n\t\t'U' or 'u' for updating the information," +
                    "\n\t\t'R' or 'r' for removing a user, " +
                    "\n\t\t'B' or 'b' for browsing all users.");

            }
        }

        String tosend = scn.nextLine();

        // check for exit

        if(tosend.equals("E") || tosend.equals("e")){
       dos.writeUTF("|EXIT|");
            System.out.println("Closing this connection : " + s);
            s.close();
            System.out.println("Connection closed");
            break;
        }

        if(!loggedIn && tosend.equals("login")){
            un = "";
            pw = "";
            System.out.println();
            System.out.print("Enter your username: ");
            un = scn.nextLine();
            System.out.print("Enter your password: ");
            pw = scn.nextLine();
            tosend = "|LOGIN|" + un + "|" + pw + "|";
            dos.writeUTF(tosend);
            received = dis.readUTF();
            words = received.substring(1).split("\\|");

            // check for the message |loginSuccesful|

            if(words[0].equals("LOGINSUCCESSFUL")) {
                loggedIn = true;

                // received will be in the form
                // |LOGINSUCCESSFUL|name|lastname|birthdate|gender
                // |email|username|password|isAdmin|

                user.setName(words[1]);
                user.setLastName(words[2]);
                user.setBirthDate(words[3]);
                user.setGender(words[4].charAt(0));
                user.setEmail(words[5]);
                user.setUsername(words[6]);
                user.setPassword(words[7]);

                if(words[8].equals("T"))
                    user.setAdmin(true);
                else
                    user.setAdmin(false);
            }
        }
        else {
            switch (tosend) {
```

```
case ("L"):
case ("l"):
    user = new User();
    loggedIn = false;
    System.out.println("\nYou are logged out.");
    break;

case ("I"):
case ("i"):
    tosend = "|INBOX|" + user.getUsername() + "|";
    dos.writeUTF(tosend);
    received = dis.readUTF();

    // inbox will be received in the form
    // |INBOX|SIZE|SENDER|TIME|TITLE|CONTENT
    // |SENDER|TIME|TITLE|CONTENT|...

    words = received.substring(1).split("\\|");
    String inbox = "";
    if (words[1].equals("0"))
        inbox = "INBOX is empty.\n";
    else {
       int index = 2;
       for (int i = 0; i <
              Integer.parseInt(words[1]); i++) {
          inbox += "\nMESSAGE " + (i + 1) + ":\n";
          inbox += "SENDER: " + words[index++] + "\n";
          inbox += "TIME: " + words[index++] + "\n";
          inbox += "TITLE: " + words[index++] + "\n";
          inbox += "MESSAGE: " + words[index++] +"\n";
        }
    }
    System.out.println(inbox);

    break;

case ("O"):
case ("o"):
    tosend = "|OUTBOX|" + user.getUsername() + "|";
    dos.writeUTF(tosend);
    received = dis.readUTF();

    // inbox will be received in the form
    // |OUTBOX|SIZE|RECEIVER|TIME|TITLE|
    // CONTENT|RECEIVER|TIME|TITLE|CONTENT|...

    words = received.substring(1).split("\\|");
    String outbox = "";
    if (words[1].equals("0"))
        outbox = "OUTBOX is empty.\n";
    else {
       int index = 2;
       for (int i = 0; i <
                 Integer.parseInt(words[1]); i++) {
         outbox += "\nMESSAGE " + (i + 1) + ":\n";
         outbox += "RECEIVER: " + words[index++]+"\n";
         outbox += "TIME: " + words[index++] + "\n";
         outbox += "TITLE: " + words[index++] + "\n";
         outbox += "MESSAGE: " + words[index++]+"\n";
        }
    }
    System.out.println(outbox);
    break;

case ("S"):
case ("s"):
    String rc = "";
    String t = "";
    String msg = "";
    System.out.println();
    System.out.print("Please enter the receiver: ");
    rc = scn.nextLine();
    System.out.print("Please enter the title: ");
    t = scn.nextLine();
    System.out.print("Please enter the message: ");
    msg = scn.nextLine();
    tosend = "|SENDMSG|" + user.getUsername() + "|" +
             rc + "|" + t + "|" + msg + "|";
    dos.writeUTF(tosend);
    received = dis.readUTF();
    words = received.substring(1).split("\\|");
    if (words[0].equals("SUCCESSFUL"))
     System.out.println("Message was sent.");
    else
     System.out.println("Message could not be sent.");

    break;

    default:
}

if(user.isAdmin()) {
  switch (tosend) {
   case ("A"):
   case ("a"):
     String fn = "";
     String ln = "";
     String bd = "";
     String g = "";
     String e = "";
     un = "";
     pw = "";
     String iA = "";
     String ib = "";
     String ob = "";

     System.out.println();
     System.out.print("Please enter the first name:");
     fn = scn.nextLine();
     System.out.print("Please enter the last name:");
     ln = scn.nextLine();
     System.out.print("Please enter the birthdate" +
                       "(yyyy/mm/dd): ");
     bd = scn.nextLine();
     System.out.print("Please enter the gender " +
                       "(M: Male, F: Female): ");
     g = scn.nextLine();
     System.out.print("Please enter the email: ");
     e = scn.nextLine();
     System.out.print("Please enter the username: ");
       un = scn.nextLine();
  System.out.print("Please enter the password: ");
  pw = scn.nextLine();
  System.out.print("Please enter the isAdmin" +
                   "(T: true, F: false): ");
  iA = scn.nextLine();

  tosend = "|ADDUSER|" + fn + "|" + ln + "|" +
           bd + "|" + g + "|" + e + "|" + un +
           "|" + pw + "|" + iA + "|";

  dos.writeUTF(tosend);
  String[] words5 =
    dis.readUTF().substring(1).split("\\|");

 if (words5[0].equals("SUCCESSFUL"))
 System.out.println("User was added succesfully.");
 else
 System.out.println("The user was not added.");
 break;

case ("U"):
case ("u"):
  un = "";
  String dtc = "";
  String nd = "";
  System.out.println();
  System.out.print("Enter the user's username: ");
  un = scn.nextLine();
  System.out.print("Enter the info you want to " +
      "change(firstname, lastname, birthdate, " +
      "gender (M: Male, F: Female), " + "email, " +
      "admin (T: true, F: false), " +
      "username, password: ");

  dtc = scn.nextLine();
  System.out.print("Enter the new data: ");
  nd = scn.nextLine();

  tosend = "|UPDATEUSER|" + un + "|" +
           dtc + "|" + nd + "|";

  dos.writeUTF(tosend);
  words = dis.readUTF().substring(1).split("\\|");

  if (words[0].equals("SUCCESSFUL"))
    System.out.println("User was updated " +
        "succesfully. The updated user should " +
        "login to see updated info.");
  else
    System.out.println("The user was not updated.");

  break;

case ("R"):
case ("r"):
  System.out.println();
  un = "";
  System.out.print("Enter the user's username " +
        "you want to remove from the system: ");
  un = scn.nextLine();
  tosend = "|REMOVE|" + un + "|";
```

```
                             dos.writeUTF(tosend);
                             words = dis.readUTF().substring(1).split("\\|");
                             if (words[0].equals("SUCCESSFUL"))
                             System.out.println("User was removed succesfully.");
                             else
                             System.out.println("The user was not removed.");

                             break;

                           case ("B"):
                           case ("b"):
                              System.out.println("\nUSERS LIST\n");
                              tosend = "|LIST|";
                              dos.writeUTF(tosend);
                              words = dis.readUTF().substring(1).split("\\|");
                              int noOfUsers = Integer.parseInt(words[1]);

                              String list = "USERS:\n\n";
                              int index = 2;
                              for (int i = 0; i < noOfUsers; i++) {
                                 list += "\nUSER " + (i + 1) + ":\n";
                                 list += "FIRST NAME: " + words[index++]+"\n";
                                 list += "LAST NAME: " + words[index++] +"\n";
                                 list += "BIRTH DATE: " + words[index++]+"\n";
                                 list += "GENDER: " + words[index++] + "\n";
                                 list += "EMAIL: " + words[index++] + "\n";
                                 list += "USERNAME: " + words[index++] + "\n";
                                 list += "PASSWORD: " + words[index++] + "\n";
                               }
                               System.out.println(list);
                               break;

                            default:
                            }
                        }
                    }
                }

                // closing resources

                scn.close();
                dis.close();
                dos.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

```
// Java implementation of Server side
// It contains two classes: Server and ClientHandler

import java.io.*;
import java.util.*;
import java.net.*;

// Server class

public class Server
{
  public static void main(String[] args) throws IOException
  {
     // server is listening on port 5056
     ServerSocket ss = new ServerSocket(5056);

     Database db = new
  Database("jdbc:postgresql://localhost:5432/UserMessagingAppDatabase",
          "postgres", "Gudukbay1905");

     // running infinite loop for getting
     // client request

     while (true)
        {
          Socket s = null;

          try
          {
              // socket object to receive incoming client requests
              s = ss.accept();

              System.out.println("A new client is connected : " + s);

              // obtaining input and out streams
              DataInputStream dis =
                  new DataInputStream(s.getInputStream());
              DataOutputStream dos =
                  new DataOutputStream(s.getOutputStream());

              System.out.println("Assigning new thread for this client");

              // create a new thread object
              Thread t = new ClientHandler(s, dis, dos, db);

              // Invoking the start() method
              t.start();

            }
            catch (Exception e){
                if(s != null)
                    s.close();
                e.printStackTrace();
            }
        }
    }
}
```

```
// ClientHandler class

class ClientHandler extends Thread {
    private final DataInputStream dis;
    private final DataOutputStream dos;
    private final Socket s;
    private Database db;


    // Constructor

    public ClientHandler(Socket s, DataInputStream dis,
                         DataOutputStream dos, Database db) {
        this.s = s;
        this.dis = dis;
        this.dos = dos;
        this.db = db;
    }

    @Override
    public void run() {
        String received;
        String username;
        String password;
        try {
            db.connection();
            while (true) {
                // Check for the user message
                received = dis.readUTF();
                String[] words = received.substring(1).split("\\|");

               System.out.println(words[0]);

                switch (words[0]) {
                    case "LOGIN":
                        username = words[1];
                        password = words[2];
                        if (db.loginSuccesfulCheck(username, password)) {
                           User user = db.queryAllUserInformation(username);
                           String isAdmin = "F";
                           if (user.isAdmin())
                              isAdmin = "T";

                           // login succesful message
                           dos.writeUTF("|LOGINSUCCESSFUL|" +
                                  user.getName() + "|" + user.getLastName() +
                                  "|" + user.getBirthDate() +
                                  "|" + user.getGender() +
                                  "|" + user.getEmail() +
                                  "|" + user.getUsername() + "|" +
                                  user.getPassword() + "|" + isAdmin + "|");
                        }
                        else
                            dos.writeUTF("|LOGINUNSUCCESSFUL|");
                        break;


                    case "INBOX":
                        // received will be in the format |INBOX|username|
                        username = words[1];
                      ArrayList<Message> inboxMessages = b.getInbox(username);
```

```
  // |INBOX|SIZE|SENDER|TIME|TITLE|CONTENT
  // |SENDER|TIME|TITLE|CONTENT|...

   String inbox = "|INBOX|";
   inbox += "" + inboxMessages.size() + "|";
   for (Message m : inboxMessages)
       inbox += m.getSender() + "|" + m.getTime() +
                "|" + m.getTitle() + "|" +
                m.getContent() + "|";

   dos.writeUTF(inbox);

   break;

 case "OUTBOX":

  // received will be in the format |OUTBOX|username|

  username = words[1];
  ArrayList<Message> outboxMessages=db.getOutbox(username);

  // |INBOX|SIZE|SENDER|TIME|TITLE|CONTENT
  // |SENDER|TIME|TITLE|CONTENT|...

  String outbox = "|OUTBOX|";
  outbox += "" + outboxMessages.size() + "|";
  for (Message m : outboxMessages)
      outbox += m.getReceiver() + "|" +
                m.getTime() + "|" +
                m.getTitle() + "|" +
                m.getContent() + "|";

  dos.writeUTF(outbox);

  break;

 case "SENDMSG":

 // the received string will be in the form
 // "|SENDMSG|SENDER|RECEIVER|TITLE|MESSAGE|"

  Message m = new Message(words[1], words[2],
               java.time.LocalTime.now().toString(),
               words[3], words[4]);

  if (db.sendMessage(m))
      dos.writeUTF("|SUCCESSFUL|");
  else
      dos.writeUTF("|UNSUCCESSFUL|");
  break;

case "ADDUSER":

  // received will be in the format
  // |ADDUSER|firstname|surname|birthdate|gender
  // |email|username|passwordcolumn|isAdmin|

  boolean isAdmin = false;
  if (words[8].equals("T"))
      isAdmin = true;
```

```
                User temp = new User(words[1], words[2],
                words[3], words[4].charAt(0), words[5],
                words[6], words[7], isAdmin);

                if (db.addUser(temp))
                    dos.writeUTF("|SUCCESSFUL|");
                 else
                    dos.writeUTF("|UNSUCCESSFUL|");
                 break;

               case "UPDATEUSER":

                 // received will be in the format
                 // |UPDATEUSER|username|dataToChange|newData|

                 if (db.updateUser(words[1], words[2], words[3]))
                     dos.writeUTF("|SUCCESSFUL|");
                 else
                     dos.writeUTF("|UNSUCCESSFUL|");

                 break;

               case "REMOVE":

                   //received will be in the format |REMOVE|username|

                   if (db.removeUser(words[1]))
                       dos.writeUTF("|SUCCESSFUL|");
                   else
                       dos.writeUTF("|UNSUCCESSFUL|");

                   break;

               case "LIST":
                   ArrayList<User> users = db.listUsers();
                   String list = "|LIST|";
                   list += users.size() + "|";

                   for (User u : users)
                       list +=  u.getName() + "|" +
                                u.getLastName() + "|" +
                                u.getBirthDate() + "|"  +
                                u.getGender() + "|" +
                                u.getEmail() + "|"+
                                u.getUsername() + "|" +
                                u.getPassword() + "|";

                   dos.writeUTF(list);

                   break;

               default:
           }
       }

   } catch (Exception e) {
       e.printStackTrace();
   }
```

```
 try {
            db.closeConnection();
            this.s.close();

            // closing resources
            this.dis.close();
            this.dos.close();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
// Java implementation of Database Class

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

public class Database{

    // properties

    private String url;
    private String user;
    private String password;
    private Connection conn;
    private Statement stmt;

    // constructor

    public Database(String url, String user, String password){
        this.url = url;
        this.user = user;
        this.password = password;
        conn = null;
        stmt = null;
    }

    //methods

    //connection method

    public void connection() throws ClassNotFoundException{
      try {
        Class.forName("org.postgresql.Driver");
        conn = DriverManager.getConnection(url, user, password);
        System.out.println("Connected to the PostgreSQL server successfully.");
        }
        catch (SQLException e) {
            System.out.println(e.getMessage());
      }
    }

    //query methods

    //user information when login is succesful

    public User queryAllUserInformation(String username) throws SQLException {
        String query;
        query = "select * from users where username = \'" + username + "\'";
        User user;
        user = new User();
        try {
             stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query);
        if(rs.next())
           user = new User(rs.getString("firstname"),
                           rs.getString("lastname"),
                           rs.getDate("birthdate").toString(),
                           rs.getString("gender").charAt(0),
                           rs.getString("email"),
                           rs.getString("username"),
                           rs.getString("passcode"),
                           rs.getBoolean("isAdmin"));

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (stmt != null) {
                stmt.close();
            }
        }

     return user;
}


// Check username and password for authentication

public boolean loginSuccesfulCheck(String username,
                 String password) throws SQLException{
    String query;
    boolean success = false;
    try {
      stmt = conn.createStatement();
      query = "select passcode from users where username = \'" +
              username + "\'";
      ResultSet rs = stmt.executeQuery(query);
      if (rs.next()){
          System.out.println(rs.getString("passcode"));
          if (rs.getString("passcode").equals(password))
              success = true;
      }
    }catch (SQLException e) {
          e.printStackTrace();
    } finally {
        if (stmt != null) {
            stmt.close();
        }
    }
    return success;
}

// get inbox for a user

public ArrayList<Message> getInbox(String username) throws SQLException{
    String query;
    ArrayList<Message> inbox = new ArrayList<Message>();
    try{
        stmt = conn.createStatement();
        query = "select * from messagetable where receiver = '" +
                username + "'";
        ResultSet rs = stmt.executeQuery(query);
        Message temp;
        while(rs.next()){
            temp = new Message(rs.getString("Sender"),
                               rs.getString("Receiver"),
                               rs.getString("messageTime"),
                               rs.getString("title"),
                               rs.getString("messageContent"));

            inbox.add(temp);
        }

    }catch(SQLException e){
        e.printStackTrace();
    }finally{
        if(stmt !=  null){
            stmt.close();
        }
    }
    return inbox;
}

// get outbox for a user

public ArrayList<Message> getOutbox(String username) throws SQLException{
    String query;
    ArrayList<Message> outbox = new ArrayList<Message>();
    try{
        stmt = conn.createStatement();
        query = "select * from messagetable where sender = '" +
                username + "'";
        ResultSet rs = stmt.executeQuery(query);
        Message temp;
        while(rs.next()){
            temp = new Message(rs.getString("Sender"),
                              rs.getString("Receiver"),
                              rs.getString("messageTime"),
                              rs.getString("title"),
                              rs.getString("messageContent"));
            outbox.add(temp);
        }
    }catch(SQLException e){
        e.printStackTrace();
    }finally{
        if(stmt !=  null){
            stmt.close();
        }
    }
    return outbox;
}

// send message specified as parameter

public boolean sendMessage(Message m) throws SQLException{
    String query;
    boolean success = false;

    try {
        stmt = conn.createStatement();
        query = "Insert into messagetable Values('" +
                m.getSender() + "', '" + m.getReceiver() + "', '" +
                m.getTitle() + "', '" + m.getTime() + "', '" +
                m.getContent() + "')";
        if (stmt.executeUpdate(query) > 0)
            success = true;

    }catch(SQLException e){
        e.printStackTrace();
    }finally{
        if(stmt !=  null){
            stmt.close();
        }
    }
    return success;
}

// add user method input is in the format
 // |ADDUSER|firstname|surname|birthdate|gender
// |email|username|passwordcolumn|isAdmin|inbox|outbox|

public boolean addUser(User user) throws SQLException, ParseException{
    String query;
    boolean success = false;
    try{
        stmt = conn.createStatement();
        query = "insert into users Values('" + user.getName() +
                "', '" + user.getLastName() + "', TO_DATE('" +
                user.getBirthDate() + "','yyyy/mm/dd'), '" +
                user.getGender() + "', '" + user.getEmail() +
                "', '" + user.getUsername() + "', '" +
                user.getPassword() + "', " + user.isAdmin() + ")";

     System.out.println(query);
        System.out.println(query);
        if(stmt.executeUpdate(query)  > 0)
            success = true;

    }catch(SQLException e){
        success = false;
        e.printStackTrace();
    }finally{
        if(stmt !=  null){
            stmt.close();
        }
    }
    return success;
}

// update method in the format Username|DataToChange|NewData|

public boolean updateUser(String username, String dataToChange,
                String newData) throws SQLException, ParseException{
    String query;
    boolean success = false;

    try {
        stmt = conn.createStatement();
        if(dataToChange.equals("birthdate"))
           query = "update users set " + dataToChange + " = TO_DATE('" +
                   newData + "', 'yyyy/mm/dd') where username = '" +
                   username + "'";
        else if(dataToChange.equals("admin")) {
            boolean isadm = false;
            if(newData.charAt(0) == 'T' || newData.charAt(0) == 't')
                isadm = true;
            query = "update users set " + dataToChange + " = " + isadm +
                   " where username = '" + username + "'";
        }
        else
            query = "update users set " + dataToChange + " = \'" +
                    newData + "\' where username = \'" + username + "\'";

        System.out.println(query);
        if(stmt.executeUpdate(query) > 0)
            success = true;
    }catch (SQLException e){
        e.printStackTrace();
    }
    finally{
        if(stmt !=  null){
            stmt.close();
        }
    }
    return success;
}

// remove user method input in the form "|REMOVE|username|"

public boolean removeUser(String username) throws SQLException{
    String query;
    boolean success = false;

    try{
        stmt = conn.createStatement();
        query = "delete from users where username = \'" + username + "\'";
        if(stmt.executeUpdate(query) > 0)
            success = true;
    }
    catch(SQLException e) {
        e.printStackTrace();
    }
    finally {
        if(stmt != null){
            stmt.close();
        }
    }
    return success;
}

// listusers method

public ArrayList<User> listUsers() throws SQLException{
    String query;
    ArrayList<User> list= new ArrayList<User>();
    User temp;
    try{
        stmt = conn.createStatement();
        query = "SELECT * FROM users" + " ORDER BY username ASC";
        ResultSet rs = stmt.executeQuery(query);
```

```
            while(rs.next()){
                temp = new User(rs.getString("firstname"),
                                rs.getString("lastname"),
                                rs.getString("birthdate"),
                                rs.getString("gender").charAt(0),
                                rs.getString("email"),
                                rs.getString("username"),
                                rs.getString("passcode"),
                                rs.getBoolean("isAdmin"));
                list.add(temp);
            }
        }catch(SQLException e){
            e.printStackTrace();
        }finally{
            if(stmt != null)
                stmt.close();
        }
        return list;
    }

    // close database connection

    public void closeConnection() throws SQLException{
        try {
            conn.close();
        } catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

B. Second Assignment: Web-based User Management & Messaging Tool, Version 1

```
//*****************************************************************
//                   MUSTAFA GÖKTAN GÜDÜKBAY                     *
// SRDC Software Research & Development and Consultancy Corp. *
//*****************************************************************

package com.example.producingwebservice;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import asm.*;


public class Database{

    //properties
    private String url;
    private String user;
    private String password;
    private Connection conn;
    private Statement stmt;

    //constructor
    public Database(String url, String user, String password){
        this.url = url;
        this.user = user;
        this.password = password;
        conn = null;
        stmt = null;
    }

    //methods

    //connection method

    public void connection() throws ClassNotFoundException{
       try {
         Class.forName("org.postgresql.Driver");
         conn = DriverManager.getConnection(url, user, password);
         System.out.println("Connected to the PostgreSQL server successfully.");
       }
       catch (SQLException e) {
         System.out.println(e.getMessage());
       }
    }

    //query methods

    //user information when login is succesful
    public User queryAllUserInformation(String username) throws SQLException {
        String query;
        query = "select * from users where username = \'" + username + "\'";
        User user;
        user = new User();

        try {
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            if(rs.next())
                user.setFirstName(rs.getString("firstname"));
                user.setLastName(rs.getString("lastname"));
                user.setBirthDate(rs.getDate("birthdate").toString());
                user.setGender(rs.getString("gender"));
                user.setEmail(rs.getString("email"));
                user.setUsername(rs.getString("username"));
                user.setPassword(rs.getString("passcode"));
                user.setIsAdmin(rs.getBoolean("isAdmin"));
                user.setAuthToken(rs.getString("authenticationToken"));
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (stmt != null) {
                stmt.close();
            }
        }

        return user;
    }


    public boolean loginSuccesfulCheck(String username,
                                String password) throws SQLException{
        String query;
        boolean success = false;
        try {
            stmt = conn.createStatement();
            query = "select passcode from users where username = \'" +
                    username + "\'";
            ResultSet rs = stmt.executeQuery(query);
            if (rs.next()){
                System.out.println(rs.getString("passcode"));
                if (rs.getString("passcode").equals(password))
                    success = true;
            }
        }catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (stmt != null) {
                stmt.close();
            }
        }
        return success;
    }

    public ArrayList<Message> getInbox(String username) throws SQLException{
        String query;
        ArrayList<Message> inbox = new ArrayList<Message>();
        try{
            stmt = conn.createStatement();
            query = "select * from messagetable where receiver = '" +
                    username + "'";
            ResultSet rs = stmt.executeQuery(query);
            while(rs.next()){
                Message temp = new Message();
                temp.setSender(rs.getString("Sender"));
                temp.setReceiver(rs.getString("Receiver"));
                temp.setTime(rs.getString("messageTime"));
                temp.setTitle(rs.getString("title"));
                temp.setContent(rs.getString("messageContent"));
                inbox.add(temp);
            }
        }catch(SQLException e){
            e.printStackTrace();
        }finally{
            if(stmt !=  null){
                stmt.close();
            }
        }
        return inbox;
    }

    public ArrayList<Message> getOutbox(String username) throws SQLException{
        String query;
        ArrayList<Message> outbox = new ArrayList<Message>();
        try{
            stmt = conn.createStatement();
            query = "select * from messagetable where sender = '" +
                    username + "'";
            ResultSet rs = stmt.executeQuery(query);

            while(rs.next()){
                Message temp = new Message();
                temp.setSender(rs.getString("Sender"));
                temp.setReceiver(rs.getString("Receiver"));
                temp.setTime(rs.getString("messageTime"));
                temp.setTitle(rs.getString("title"));
                temp.setContent(rs.getString("messageContent"));
                outbox.add(temp);
            }
        }catch(SQLException e){
            e.printStackTrace();
        }finally{
            if(stmt !=  null){
                stmt.close();
            }
        }
        return outbox;
    }



    public boolean sendMessage(Message m) throws SQLException{
        String query;
        boolean success = false;

        try {
            stmt = conn.createStatement();
            query = "Insert into messagetable Values('" + m.getSender() +
                    "', '" + m.getReceiver() + "', '" + m.getTitle() +
                    "', '" + m.getTime() + "', '" + m.getContent() + "')";

            if (stmt.executeUpdate(query) > 0)
                success = true;
        }catch(SQLException e){
            e.printStackTrace();
        }finally{
            if(stmt !=  null){
                stmt.close();
            }
        }
        return success;
    }



    // add user method input is in the format
    // |ADDUSER|firstname|surname|birthdate|gender
    // |email|username|passwordcolumn|isAdmin|inbox|outbox|

    public boolean addUser(User user) throws SQLException, ParseException{
        String query;
        boolean success = false;
        try{
            stmt = conn.createStatement();
            query = "insert into users Values('" + user.getFirstName() +
                    "', '" + user.getLastName() + "', TO_DATE('" +
                    user.getBirthDate() + "','yyyy/mm/dd'), '" +
                    user.getGender() + "', '" + user.getEmail() + "', '" +
                    user.getUsername() + "', '" + user.getPassword() + "', " +
                    user.isIsAdmin() + ",' " + user.getAuthToken() + "')";


            System.out.println(query);
            System.out.println(query);
            if(stmt.executeUpdate(query)  > 0)
                success = true;

        }catch(SQLException e){
            success = false;
            e.printStackTrace();
        }finally{
            if(stmt !=  null){
                stmt.close();
            }
        }
        return success;
    }

    //update method in the format Username|DataToChange|NewData|

    public boolean updateUser(String username, String dataToChange,
                        String newData) throws SQLException, ParseException{
        String query;
        boolean success = false;

        try {
            stmt = conn.createStatement();
            if(dataToChange.equals("birthdate"))
                query = "update users set " + dataToChange + " = TO_DATE('" +
                        newData + "', 'yyyy/mm/dd') where username = '" +
                        username + "'";
            else if(dataToChange.equals("admin")) {
                boolean isadm = false;
                if(newData.charAt(0) == 'T' || newData.charAt(0) == 't')
                    isadm = true;
                query = "update users set " + dataToChange + " = " + isadm +
                        " where username = '" + username + "'";
            }
            else
                query = "update users set " + dataToChange + " = \'" +
                        newData + "\' where username = \'" + username + "\'";
            System.out.println(query);
            if(stmt.executeUpdate(query) > 0)
                success = true;
        }catch (SQLException e){
            e.printStackTrace();
        }
        finally{
            if(stmt !=  null){
                stmt.close();
            }
        }
        return success;
    }


    //remove user method input in the form "|REMOVE|username|"

    public boolean removeUser(String username) throws SQLException{
        String query;
        boolean success = false;
        try{
            stmt = conn.createStatement();
            query = "delete from users where username = \'" + username + "\'";
            if(stmt.executeUpdate(query) > 0)
                success = true;
        }
        catch(SQLException e) {
            e.printStackTrace();
        }
        finally {
            if(stmt != null){
                stmt.close();
            }
        }
        return success;
    }

    //listusers method

    public ArrayList<User> listUsers() throws SQLException{
        String query;
        ArrayList<User> list= new ArrayList<User>();
        try{
            stmt = conn.createStatement();
            query = "SELECT * FROM users" + " ORDER BY username ASC";
            ResultSet rs = stmt.executeQuery(query);

            while(rs.next()){
                User user = new User();
                user.setFirstName(rs.getString("firstname"));
                user.setLastName(rs.getString("lastname"));
                user.setBirthDate(rs.getDate("birthdate").toString());
                user.setGender(rs.getString("gender"));
```

```
                user.setEmail(rs.getString("email"));
                user.setUsername(rs.getString("username"));
                user.setPassword(rs.getString("passcode"));
                user.setIsAdmin(rs.getBoolean("isAdmin"));
                list.add(user);
            }
            System.out.println("ffdsjlaa");
            for(User u:list)
                System.out.println(u.getUsername());
        }catch(SQLException e){
            e.printStackTrace();
        }finally{
            if(stmt != null)
                stmt.close();
        }
        return list;
    }

    //returns true if the token exists on the table

    public boolean searchToken(String token) throws SQLException{
        String query = "SELECT * FROM users WHERE authenticationtoken = \'" +
                        token + "\'";
        System.out.println(query);
        try{
            stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);
            if (rs.next())
                return true;
            else
                return false;

        }catch (SQLException e){
            e.printStackTrace();
        }
        return false;
    }
    public void closeConnection() throws SQLException{
        try {
            conn.close();
        } catch(SQLException e){
            e.printStackTrace();
        }
    }
}
```

```
// WebServiceConfig Class Implementation

package com.example.producingwebservice;

import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.ws.config.annotation.EnableWs;
import org.springframework.ws.config.annotation.WsConfigurerAdapter;
import org.springframework.ws.transport.http.MessageDispatcherServlet;
import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;

@EnableWs
@Configuration
public class WebServiceConfig extends WsConfigurerAdapter {
  @Bean
  public ServletRegistrationBean
        messageDispatcherServlet(ApplicationContext applicationContext) {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(applicationContext);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean(servlet, "/ws/*");
    }

  @Bean(name = "chat")
  public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema chatSchema) {
      DefaultWsdl11Definition wsdl11Definition =
                  new DefaultWsdl11Definition();
      wsdl11Definition.setPortTypeName("ChatPort");
      wsdl11Definition.setLocationUri("/ws");
      wsdl11Definition.setTargetNamespace(
                  "http://spring.io/guides/gs-producing-web-service");
      wsdl11Definition.setSchema(chatSchema);
      return wsdl11Definition;
    }

    @Bean
    public XsdSchema countriesSchema() {
        return new SimpleXsdSchema(new ClassPathResource("chat.xsd"));
    }
}
```

```
// ChatEndpoint Class Implementation

package com.example.producingwebservice;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.util.Assert;
import org.springframework.ws.server.endpoint.annotation.Endpoint;
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
import org.springframework.ws.server.endpoint.annotation.RequestPayload;
import org.springframework.ws.server.endpoint.annotation.ResponsePayload;

import asm.*;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;


@Endpoint
public class ChatEndpoint {
    private static final String NAMESPACE_URI =
               "http://spring.io/guides/gs-producing-web-service";

    private Database db;

    @Autowired
    public ChatEndpoint() {
      db =
      new Database("jdbc:postgresql://localhost:5432/UserMessagingAppDatabase",
                   "postgres", "Gudukbay1905");
       try{
           db.connection();
       } catch (ClassNotFoundException e) {
         e.printStackTrace();
       }
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "getUserRequest")
    @ResponsePayload
    public GetUserResponse getUser(@RequestPayload GetUserRequest request) {
        System.out.print("dsadsjkdjkasjks");
        GetUserResponse response = new GetUserResponse();
        boolean succesfulLogin;
        String username = request.getUsername();
        String password = request.getPassword();
        try {
            succesfulLogin = db.loginSuccesfulCheck(username, password);
            if(succesfulLogin)
                response.setUser(db.queryAllUserInformation(username));
        }catch (SQLException e){
            e.printStackTrace();
        }
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "inboxRequest")
    @ResponsePayload
    public InboxResponse getInbox(@RequestPayload InboxRequest request) {
        InboxResponse response = new InboxResponse();
        String username = request.getUsername();
        String token = request.getToken();
        try {
            if(db.searchToken(token)) {
                ArrayList<Message> inbox = db.getInbox(username);
                response.getMessage().addAll(inbox);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return response;
    }



    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "outboxRequest")
    @ResponsePayload
    public OutboxResponse getOutbox(@RequestPayload OutboxRequest request) {
        OutboxResponse response = new OutboxResponse();
        String username = request.getUsername();
        try {
            String token = request.getToken();
            if(db.searchToken(token)) {
                ArrayList<Message> outbox = db.getOutbox(username);
                response.getOutboxMessage().addAll(outbox);
            }
        }catch (SQLException e){
            e.printStackTrace();
        }
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "sendMessageRequest")
    @ResponsePayload
    public SendMessageResponse sendMessageResponse(@RequestPayload
               SendMessageRequest request) {
        SendMessageResponse response = new SendMessageResponse();
        Message m = request.getMessageToSend();
        System.out.println(m.getSender());
        System.out.println(m.getReceiver());
        String token = request.getToken();
        boolean success = false;
        try {
            if(db.searchToken(token))
                success = db.sendMessage(m);
            if(success)
                response.setMessageInfo("SUCCESSFUL");
            else
                response.setMessageInfo("UNSUCCESSFUL");
        }catch (SQLException e){
            e.printStackTrace();
        }
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "addUserRequest")
    @ResponsePayload
    public AddUserResponse
            addUserResponse(@RequestPayload AddUserRequest request) {
        AddUserResponse response = new AddUserResponse();
        User userToBeAdded = request.getUserToAdd();
        boolean success = false;
        String token = request.getToken();
        try {
            if(db.searchToken(token))
                success = db.addUser(userToBeAdded);
            if(success)
                response.setAddInfo("SUCCESSFUL");
            else
                response.setAddInfo("UNSUCCESSFUL");
        }catch (Exception e){
            e.printStackTrace();
        }
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "updateUserRequest")
    @ResponsePayload
    public UpdateUserResponse
            updateUserResponse(@RequestPayload UpdateUserRequest request) {
        UpdateUserResponse response = new UpdateUserResponse();
        boolean success = false;
        String token = request.getToken();
        try {
            if(db.searchToken(token))
                success = db.updateUser(request.getUsername(),
                                        request.getDataToChange(),
                                        request.getNewData());

            if(success)
                response.setUpdateInfo("SUCCESSFUL");
            else
                response.setUpdateInfo("UNSUCCESSFUL");
        }catch (Exception e){
            e.printStackTrace();
        }
        return response;
    }

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "removeUserRequest")
    @ResponsePayload
         public RemoveUserResponse removeUserResponse(@RequestPayload
                  RemoveUserRequest request) {
        RemoveUserResponse response = new RemoveUserResponse();
        boolean success = false;
        String token = request.getToken();
        try {
            if(db.searchToken(token))
                success = db.removeUser(request.getUsername());
            if(success)
                response.setRemoveInfo("SUCCESSFUL");
            else
                response.setRemoveInfo("UNSUCCESSFUL");
        }catch (Exception e){
            e.printStackTrace();
        }
        return response;
    }
```

```
      @PayloadRoot(namespace = NAMESPACE_URI, localPart = "listUsersRequest")
      @ResponsePayload
      public ListUsersResponse
             listUsersResponse(@RequestPayload ListUsersRequest request) {
        ListUsersResponse response = new ListUsersResponse();
        String token = request.getToken();
        try {
            if(db.searchToken(token))
                response.getUser().addAll(db.listUsers());
        }catch (Exception e){
            e.printStackTrace();
        }
        return response;
    }
}




// ProducingWebServiceApplication Class including main method to test

package com.example.producingwebservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class ProducingWebServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProducingWebServiceApplication.class, args);
    }
}
```

```
// XML Schema (XSD) for Handling Server Requests and Responses

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://spring.io/guides/gs-producing-web-service"
           targetNamespace="http://spring.io/guides/gs-producing-web-service"
           elementFormDefault="qualified">

    <xs:element name="getUserRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="username" type="xs:string"/>
                <xs:element name="password" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="getUserResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="user" type="tns:User"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="inboxRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="username" type="xs:string"/>
                <xs:element name="token" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="inboxResponse">
        <xs:complexType>
           <xs:sequence>
              <xs:element name="message" type="tns:Message" maxOccurs="100"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="outboxRequest">
        <xs:complexType>
           <xs:sequence>
               <xs:element name="username" type="xs:string"/>
               <xs:element name="token" type="xs:string"/>
           </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="outboxResponse">
       <xs:complexType>
         <xs:sequence>
          <xs:element name="outboxMessage" type="tns:Message" maxOccurs="100"/>
           </xs:sequence>
       </xs:complexType>
    </xs:element>
    <xs:element name="sendMessageRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="messageToSend" type="tns:Message"/>
            <xs:element name="token" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="sendMessageResponse">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="messageInfo" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="addUserRequest">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="userToAdd" type="tns:User"/>
            <xs:element name="token" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="addUserResponse">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="addInfo" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="updateUserRequest">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="username" type="xs:string"/>
            <xs:element name="dataToChange" type="xs:string"/>
            <xs:element name="newData" type="xs:string"/>
            <xs:element name="token" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="updateUserResponse">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="updateInfo" type="xs:string"/>
            <xs:element name="token" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="removeUserRequest">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="username" type="xs:string"/>
            <xs:element name="token" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
    <xs:element name="removeUserResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="removeInfo" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="listUsersRequest">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="token" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="listUsersResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="user" type="tns:User" maxOccurs="100"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="User">
        <xs:sequence>
            <xs:element name="firstName" type="xs:string"/>
            <xs:element name="lastName" type="xs:string"/>
            <xs:element name="birthDate" type="xs:string"/>
            <xs:element name="gender" type="tns:char"/>
            <xs:element name="email" type="xs:string"/>
            <xs:element name="username" type="xs:string"/>
            <xs:element name="password" type="xs:string"/>
            <xs:element name="isAdmin" type="xs:boolean"/>
            <xs:element name="authToken" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="Message">
        <xs:sequence>
            <xs:element name="sender" type="xs:string"/>
            <xs:element name="receiver" type="xs:string"/>
            <xs:element name="time" type="xs:string"/>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="content" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>

    <xs:simpleType name="char">
        <xs:restriction base="xs:string">
            <xs:pattern value="[A-Z]"/>
        </xs:restriction>
    </xs:simpleType>
</xs:schema>
```

```
// HTML File for the Messaging Client, which includes JavaScript Code

<html lang="en">
<head>
    <title>SOAP JavaScript Client Test</title>
    <link rel="stylesheet" href="client.css">
    <script type="text/javascript">

        var usernameLoggedIn;
        var token;

        function loginRequest() {
            var xmlhttp = new XMLHttpRequest();
            xmlhttp.open("POST", "http://localhost:8080/ws", true);

            var enteredUsername =
                document.getElementById("txt_username").value;
            var enteredPassword =
                document.getElementById("txt_password").value;

            // build SOAP request
            var sr = '<?xml version="1.0" encoding="utf-8"?>' +
             '<soapenv:Envelope xmlns:soapenv=
              "http://schemas.xmlsoap.org/soap/envelope/"' +
             ' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">'+
                '<soapenv:Header/>' +
             '<soapenv:Body>' +
             '<gs:getUserRequest>' +
             '<gs:username>' + enteredUsername + '</gs:username>' +
             '<gs:password>' + enteredPassword + '</gs:password>' +
             '</gs:getUserRequest>'+
             '</soapenv:Body>'+
             '</soapenv:Envelope>';

            xmlhttp.onreadystatechange = function () {
              if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
                var xmlDoc = xmlhttp.responseXML;
                var a = xmlDoc.getElementsByTagName('ns2:username');
                usernameLoggedIn = a[0].childNodes[0].nodeValue;

                console.log(document.getElementById("txt_username").value);
                console.log(a[0].childNodes[0].nodeValue);
                var str = '' + a[0].childNodes[0].nodeValue;

                if(a[0].childNodes[0].nodeValue.localeCompare(enteredUsername)
                    === 0){
                   usernameLoggedIn =  enteredUsername;
                   token = xmlDoc.getElementsByTagName('ns2:authToken')[0].
                              childNodes[0].nodeValue;
                   console.log(token);
                   menu(xmlDoc.getElementsByTagName("ns2:firstName")[0].
                        childNodes[0].nodeValue,
                   xmlDoc.getElementsByTagName("ns2:lastName")[0].
                        childNodes[0].nodeValue,
                   xmlDoc.getElementsByTagName("ns2:isAdmin")[0].
                        childNodes[0].nodeValue);
                    }
                }
            }
            // Send the POST request
            xmlhttp.setRequestHeader('Content-Type', 'text/xml');
            xmlhttp.send(sr);
        }

        function menu(fn, ln, admin){

            document.getElementById("buttons").style = "display:block";
            document.getElementById("buttons").style ="display:inlineBlock";
            document.getElementById("buttons").className ="buttonContainer";

            document.getElementById("login_form").style="display:none";

            document.getElementById("inboxButton").disabled = false;
            document.getElementById("outboxButton").disabled = false;
            document.getElementById("sendButton").disabled = false;
            document.getElementById("logoutButton").disabled = false;

            console.log(admin);
            console.log(admin.localeCompare('true'));
            if(admin.localeCompare('true') == 0){
                document.getElementById("addButton").disabled = false;
                document.getElementById("updateButton").disabled = false;
                document.getElementById("removeButton").disabled = false;
                document.getElementById("listButton").disabled = false;
                document.getElementById("addButton").style =
                    "display:inline-block";
                document.getElementById("updateButton").style =
                    "display:inline-block";
                document.getElementById("removeButton").style =
                    "display:inline-block";
                document.getElementById("listButton").style =
                    "display:inline-block";
            }
            else{
                document.getElementById("addButton").style = "display:none";
                document.getElementById("updateButton").style =
                    "display:none";
                document.getElementById("removeButton").style =
                    "display:none";
                document.getElementById("listButton").style = "display:none";
            }
        }

        function inboxRequest(){
            document.getElementById("sendMessage_form").style =
                   "display:none";
            document.getElementById("sendMessage_form").reset();
            document.getElementById("sendMessageButton").disabled = false;

            document.getElementById("addUser_form").style = "display:none";
            document.getElementById("addUser_form").reset();
            document.getElementById("addUserButton").disabled = true;

            document.getElementById("update_form").style = "display:none";
            document.getElementById("update_form").reset();
            document.getElementById("updateUserButton").disabled = true;

            document.getElementById("remove_form").style = "display:none";
            document.getElementById("remove_form").reset();
            document.getElementById("removeUserButton").disabled = true;
```

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.open("POST", "http://localhost:8080/ws", true);

// build SOAP request
var sr = '<?xml version="1.0" encoding="utf-8"?>' +
    '<soapenv:Envelope xmlns:soapenv=
     "http://schemas.xmlsoap.org/soap/envelope/"' +
' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">' +
    '<soapenv:Header/>' +
    '<soapenv:Body>' +
    '<gs:inboxRequest>' +
    '<gs:username>' + usernameLoggedIn + '</gs:username>' +
    '<gs:token>' + token + '</gs:token>' +
    '</gs:inboxRequest>'+
    '</soapenv:Body>'+
    '</soapenv:Envelope>';

xmlhttp.onreadystatechange = function () {

    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {

        var xmlDoc = xmlhttp.responseXML;
        var allSenders =
            xmlDoc.getElementsByTagName('ns2:sender');
        var inboxStatement = "INBOX:\n\n";
        if(allSenders.length != 0) {
           var allReceivers =
               xmlDoc.getElementsByTagName('ns2:receiver');
           var allTime = xmlDoc.getElementsByTagName('ns2:time');
           var allTitle =
               xmlDoc.getElementsByTagName('ns2:title');
           var allContent =
               xmlDoc.getElementsByTagName('ns2:content');
           for(var i = 0; i < allSenders.length; i++) {
              inboxStatement += "Message " + (i + 1) + ":\n";
              inboxStatement += "Sender: " +
                allSenders[i].childNodes[0].nodeValue + "\n";
              inboxStatement += "Receiver: " +
                allReceivers[i].childNodes[0].nodeValue + "\n";
              inboxStatement += "Time: " +
                allTime[i].childNodes[0].nodeValue + "\n";
              inboxStatement += "Title: " +
                allTitle[i].childNodes[0].nodeValue + "\n";
              inboxStatement += "Content: " +
                allContent[i].childNodes[0].nodeValue + "\n\n\n";
           }
           document.getElementById("textBoxDivision1").style =
               "display:block";
           document.getElementById("textBox1").className =
               "textArea1";
           document.getElementById("textBox1").value =
               inboxStatement;
           document.getElementById("closeWindowButton").disabled =
               false;
           document.getElementById("closeWindowButton").className=
               ".closeButton";
        }
    }
}
```

```
    // Send the POST request
    xmlhttp.setRequestHeader('Content-Type', 'text/xml');
    xmlhttp.send(sr);
}

function outboxRequest(){
    document.getElementById("sendMessage_form").style =
       "display:none";
    document.getElementById("sendMessage_form").reset();
    document.getElementById("sendMessageButton").disabled = false;

    document.getElementById("addUser_form").style = "display:none";
    document.getElementById("addUser_form").reset();
    document.getElementById("addUserButton").disabled = true;

    document.getElementById("update_form").style = "display:none";
    document.getElementById("update_form").reset();
    document.getElementById("updateUserButton").disabled = true;

    document.getElementById("remove_form").style = "display:none";
    document.getElementById("remove_form").reset();
    document.getElementById("removeUserButton").disabled = true;

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", "http://localhost:8080/ws", true);

    // build SOAP request
    var sr = '<?xml version="1.0" encoding="utf-8"?>' +
     '<soapenv:Envelope xmlns:soapenv=
      "http://schemas.xmlsoap.org/soap/envelope/"' +
     ' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">'+
     '<soapenv:Header/>' +
     '<soapenv:Body>' +
     '<gs:outboxRequest>' +
     '<gs:username>' + usernameLoggedIn + '</gs:username>' +
     '<gs:token>' + token + '</gs:token>' +
     '</gs:outboxRequest>'+
     '</soapenv:Body>'+
     '</soapenv:Envelope>';

    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            var xmlDoc = xmlhttp.responseXML;
            var allSenders =
                xmlDoc.getElementsByTagName('ns2:sender');
            var inboxStatement = "OUTBOX:\n\n";
            if(allSenders.length != 0) {
                var allReceivers =
               xmlDoc.getElementsByTagName('ns2:receiver');
               var allTime = xmlDoc.getElementsByTagName('ns2:time');
               var allTitle =
                   xmlDoc.getElementsByTagName('ns2:title');
               var allContent =
                   xmlDoc.getElementsByTagName('ns2:content');
               for(var i = 0; i < allSenders.length; i++) {
                   inboxStatement += "Message " + (i + 1) + ":\n";
                   inboxStatement += "Sender: " +
                     allSenders[i].childNodes[0].nodeValue + "\n";
                   inboxStatement += "Receiver: " +
                      allReceivers[i].childNodes[0].nodeValue + "\n";
```

```
                    inboxStatement += "Time: " +
                      allTime[i].childNodes[0].nodeValue + "\n";
                    inboxStatement += "Title: " +
                      allTitle[i].childNodes[0].nodeValue + "\n";
                    inboxStatement += "Content: " +
                      allContent[i].childNodes[0].nodeValue + "\n\n\n";
                }

                document.getElementById("textBoxDivision1").style =
                   "display:block";
                document.getElementById("textBox1").className =
                   "textArea1";
                document.getElementById("textBox1").value =
                   inboxStatement;
                document.getElementById("closeWindowButton").disabled=
                   false;
                document.getElementById("closeWindowButton").className=
                   ".closeButton";
            }
        }
    }

    // Send the POST request
    xmlhttp.setRequestHeader('Content-Type', 'text/xml');
    xmlhttp.send(sr);

}

function sendMessageRequest(){
    document.getElementById("sendMessage_form").style =
       "display:none";
    document.getElementById("sendMessage_form").reset();
    document.getElementById("sendMessageButton").disabled = false;

    document.getElementById("addUser_form").style = "display:none";
    document.getElementById("addUser_form").reset();
    document.getElementById("addUserButton").disabled = true;

    document.getElementById("update_form").style = "display:none";
    document.getElementById("update_form").reset();
    document.getElementById("updateUserButton").disabled = true;

    document.getElementById("remove_form").style = "display:none";
    document.getElementById("remove_form").reset();
    document.getElementById("removeUserButton").disabled = true;

    closeButtonMethod1();

    document.getElementById("sendMessage_form").style =
       "display:block";
    document.getElementById("sendMessageButton").disabled = false;

    document.getElementById("sendMessageButton").
       addEventListener("click", sendTheMessage);
}

function sendTheMessage() {
    var receiverXML = document.getElementById("txt_receiver").value;
    var titleXML = document.getElementById("txt_title").value;
    var messageXML = document.getElementById("txt_message").value;
    var today = new Date();
    var time = today.getHours() + ":" + today.getMinutes() + ":" +
               today.getSeconds();

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", "http://localhost:8080/ws", true);

    // build SOAP request
    var sr = '<?xml version="1.0" encoding="utf-8"?>' +
     '<soapenv:Envelope xmlns:soapenv=
      "http://schemas.xmlsoap.org/soap/envelope/"' +
     ' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">'+
     '<soapenv:Header/>' +
     '<soapenv:Body>' +
     '<gs:sendMessageRequest>' +
     '<gs:messageToSend>' +
     '<gs:sender>' + usernameLoggedIn + '</gs:sender>' +
     '<gs:receiver>' + receiverXML + '</gs:receiver>' +
     '<gs:time>' + time + '</gs:time>' +
     '<gs:title>' + titleXML + '</gs:title>' +
     '<gs:content>' + messageXML + '</gs:content>' +
     '</gs:messageToSend>'+
     '<gs:token>' + token + '</gs:token>' +
     '</gs:sendMessageRequest>' +
     '</soapenv:Body>'+
     '</soapenv:Envelope>';

    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            var xmlDoc = xmlhttp.responseXML;
            var messageInfo =
                xmlDoc.getElementsByTagName('ns2:messageInfo');
            if(messageInfo[0].childNodes[0].nodeValue.
               localeCompare("SUCCESSFUL") == 0)
               alert("Message was sent successfully.");
            else
                alert("Message could not been sent.");
        }
    }

    // Send the POST request
    xmlhttp.setRequestHeader('Content-Type', 'text/xml');
    xmlhttp.send(sr);

    document.getElementById("sendMessage_form").style =
       "display:none";
    document.getElementById("sendMessage_form").reset();
    document.getElementById("sendMessageButton").disabled = false;
}

function addUserRequest(){
    document.getElementById("sendMessage_form").style =
       "display:none";
    document.getElementById("sendMessage_form").reset();
    document.getElementById("sendMessageButton").disabled = false;

    document.getElementById("addUser_form").style = "display:none";
    document.getElementById("addUser_form").reset();
    document.getElementById("addUserButton").disabled = true;

    document.getElementById("update_form").style = "display:none";
    document.getElementById("update_form").reset();
    document.getElementById("updateUserButton").disabled = true;

    document.getElementById("remove_form").style = "display:none";
    document.getElementById("remove_form").reset();
    document.getElementById("removeUserButton").disabled = true;

    closeButtonMethod1();

    document.getElementById("addUser_form").style = "display:block";
    document.getElementById("addUserButton").disabled = false;
    document.getElementById("addUserButton").addEventListener("click",
       addTheUser);
}

function addTheUser() {
    var firstNameXML = document.getElementById("txt_fn").value;
    var secondNameXML = document.getElementById("txt_ln").value;
    var birthDateXML = document.getElementById("txt_bd").value;
    var genderXML = document.getElementById("txt_g").value;
    var emailXML = document.getElementById("txt_e").value;
    var usernameXML = document.getElementById("txt_u").value;
    var passwordXML = document.getElementById("txt_p").value;
    var adminXML = document.getElementById("txt_a").value;


    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", "http://localhost:8080/ws", true);
    var result = '';
    var characters =
     'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
    var charactersLength = characters.length;
    for ( var i = 0; i < 10; i++ ) {
        result += characters.charAt(Math.floor(Math.random() *
                  charactersLength));
    }
    console.log(result);

    // build SOAP request
    var sr = '<?xml version="1.0" encoding="utf-8"?>' +
     '<soapenv:Envelope xmlns:soapenv=
      "http://schemas.xmlsoap.org/soap/envelope/"' +
     ' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">'+
     '<soapenv:Body>' +
     '<soapenv:Header/>' +
     '<gs:addUserRequest>' +
     '<gs:userToAdd>' +
     '<gs:firstName>' + firstNameXML + '</gs:firstName>' +
     '<gs:lastName>' + secondNameXML + '</gs:lastName>' +
     '<gs:birthDate>' + birthDateXML + '</gs:birthDate>' +
     '<gs:gender>' + genderXML + '</gs:gender>' +
     '<gs:email>' + emailXML + '</gs:email>' +
     '<gs:username>' + usernameXML + '</gs:username>' +
     '<gs:password>' + passwordXML + '</gs:password>' +
     '<gs:isAdmin>' + adminXML + '</gs:isAdmin>' +
     '<gs:authToken>' + result + '</gs:authToken>' +
     '</gs:userToAdd>' +
     '<gs:token>' + token + '</gs:token>' +
     '</gs:addUserRequest>' +
     '</soapenv:Body>' +
     '</soapenv:Envelope>';
    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            var xmlDoc = xmlhttp.responseXML;
            var addInfo = xmlDoc.getElementsByTagName('ns2:addInfo');
            if (addInfo[0].childNodes[0].nodeValue.
                localeCompare("SUCCESSFUL") == 0)
                alert("User was added successfully.");
            else
                alert("User could not been added.");
        }
    }

    // Send the POST request
    xmlhttp.setRequestHeader('Content-Type', 'text/xml');
    xmlhttp.send(sr);

    document.getElementById("addUser_form").style = "display:none";
    document.getElementById("addUser_form").reset();
    document.getElementById("addUserButton").disabled = true;
}

function removeRequest(){
    document.getElementById("sendMessage_form").style =
       "display:none";
    document.getElementById("sendMessage_form").reset();
    document.getElementById("sendMessageButton").disabled = false;

    document.getElementById("addUser_form").style = "display:none";
    document.getElementById("addUser_form").reset();
    document.getElementById("addUserButton").disabled = true;

    document.getElementById("update_form").style = "display:none";
    document.getElementById("update_form").reset();
    document.getElementById("updateUserButton").disabled = true;

    document.getElementById("remove_form").style = "display:none";
    document.getElementById("remove_form").reset();
    document.getElementById("removeUserButton").disabled = true;

    closeButtonMethod1();

    document.getElementById("remove_form").style = "display:block";
    document.getElementById("removeUserButton").disabled = false;
    document.getElementById("removeUserButton").
       addEventListener("click", removeTheUser);
}

function removeTheUser() {
    var userToBeRemovedXML = document.getElementById("txt_uR").value;

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", "http://localhost:8080/ws", true);

    // build SOAP request
    var sr = '<?xml version="1.0" encoding="utf-8"?>' +
     '<soapenv:Envelope xmlns:soapenv=
      "http://schemas.xmlsoap.org/soap/envelope/"' +
     ' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">'+
     '<soapenv:Header/>' +
     '<soapenv:Body>' +
     '<gs:removeUserRequest>' +
     '<gs:username>' + userToBeRemovedXML + '</gs:username>' +
     '<gs:token>' + token + '</gs:token>' +
     '</gs:removeUserRequest>' +
     '</soapenv:Body>' +
     '</soapenv:Envelope>';

    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {

            var xmlDoc = xmlhttp.responseXML;
            var messageInfo =
                xmlDoc.getElementsByTagName('ns2:removeInfo');
            if (messageInfo[0].childNodes[0].nodeValue.
                localeCompare("SUCCESSFUL") == 0)
                alert("User was removed successfully.");
            else
                alert("User could not been removed.");
        }
    }
    // Send the POST request
    xmlhttp.setRequestHeader('Content-Type', 'text/xml');
    xmlhttp.send(sr);

    document.getElementById("remove_form").style = "display:none";
    document.getElementById("remove_form").reset();
    document.getElementById("removeUserButton").disabled = true;
}

function updateRequest(){
    document.getElementById("sendMessage_form").style =
       "display:none";
    document.getElementById("sendMessage_form").reset();
    document.getElementById("sendMessageButton").disabled = false;

    document.getElementById("addUser_form").style = "display:none";
    document.getElementById("addUser_form").reset();
    document.getElementById("addUserButton").disabled = true;

    document.getElementById("update_form").style = "display:none";
    document.getElementById("update_form").reset();
    document.getElementById("updateUserButton").disabled = true;

    document.getElementById("remove_form").style = "display:none";
    document.getElementById("remove_form").reset();
    document.getElementById("removeUserButton").disabled = true;

    closeButtonMethod1();

    document.getElementById("update_form").style = "display:block";
    document.getElementById("updateUserButton").disabled = false;
    document.getElementById("updateUserButton").
       addEventListener("click", updateTheUser);
}

function updateTheUser(){
    var userToBeUpdatedXML = document.getElementById("txt_uU").value;
    var dataToBeUpdatedXML = document.getElementById("txt_dU").value;
    var newDataToBeUpdatedXML =
        document.getElementById("txt_nU").value;
```

```
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", "http://localhost:8080/ws", true);

    // build SOAP request
    var sr = '<?xml version="1.0" encoding="utf-8"?>' +
     '<soapenv:Envelope xmlns:soapenv=
      "http://schemas.xmlsoap.org/soap/envelope/"' +
     ' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">'+
     '<soapenv:Header/>' +
     '<soapenv:Body>' +
     '<gs:updateUserRequest>' +
     '<gs:username>' + userToBeUpdatedXML + '</gs:username>' +
     '<gs:dataToChange>' + dataToBeUpdatedXML + '</gs:dataToChange>' +
     '<gs:newData>' + newDataToBeUpdatedXML + '</gs:newData>' +
     '<gs:token>' + token + '</gs:token>' +
     '</gs:updateUserRequest>' +
     '</soapenv:Body>' +
     '</soapenv:Envelope>';

    xmlhttp.onreadystatechange = function () {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
            var xmlDoc = xmlhttp.responseXML;
            var updateInfo =
                xmlDoc.getElementsByTagName('ns2:updateInfo');
            if (updateInfo[0].childNodes[0].nodeValue.
                localeCompare("SUCCESSFUL") == 0)
                alert("User was updated successfully.");
            else
                alert("User could not been updated.");
        }
    }
    // Send the POST request
    xmlhttp.setRequestHeader('Content-Type', 'text/xml');
    xmlhttp.send(sr);

    document.getElementById("update_form").style = "display:none";
    document.getElementById("update_form").reset();
    document.getElementById("updateUserButton").disabled = true;
}

function listRequest(){
    document.getElementById("sendMessage_form").style =
       "display:none";
    document.getElementById("sendMessage_form").reset();
    document.getElementById("sendMessageButton").disabled = false;

    document.getElementById("addUser_form").style = "display:none";
    document.getElementById("addUser_form").reset();
    document.getElementById("addUserButton").disabled = true;

    document.getElementById("update_form").style = "display:none";
    document.getElementById("update_form").reset();
    document.getElementById("updateUserButton").disabled = true;

    document.getElementById("remove_form").style = "display:none";
    document.getElementById("remove_form").reset();
    document.getElementById("removeUserButton").disabled = true;

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.open("POST", "http://localhost:8080/ws", true);
```

```
// build SOAP request
var sr = '<?xml version="1.0" encoding="utf-8"?>' +
 '<soapenv:Envelope xmlns:soapenv=
  "http://schemas.xmlsoap.org/soap/envelope/"' +
 ' xmlns:gs= "http://spring.io/guides/gs-producing-web-service">'+
 '<soapenv:Header/>' +
 '<soapenv:Body>' +
 '<gs:listUsersRequest>' +
 '<gs:token>' + token + '</gs:token>' +
 '</gs:listUsersRequest>' +
 '</soapenv:Body>'+
 '</soapenv:Envelope>';

xmlhttp.onreadystatechange = function () {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        var xmlDoc = xmlhttp.responseXML;
        var allUsers =
            xmlDoc.getElementsByTagName('ns2:username');
        var userStatement = "USERS:\n\n";
        if(allUsers.length != 0) {
           var allFirstName =
               xmlDoc.getElementsByTagName('ns2:firstName');
           var allLastName =
               xmlDoc.getElementsByTagName('ns2:lastName');
           var allBirthDate =
               xmlDoc.getElementsByTagName('ns2:birthDate');
           var allGender =
               xmlDoc.getElementsByTagName('ns2:gender');
           var allEmail =
               xmlDoc.getElementsByTagName('ns2:email');
           for(var i = 0; i < allUsers.length; i++) {
               userStatement += "User " + (i + 1) + ":\n";
               userStatement += "First Name: " +
                  allFirstName[i].childNodes[0].nodeValue + "\n";
               userStatement += "Last Name: " +
                  allLastName[i].childNodes[0].nodeValue + "\n";
               userStatement += "Birthdate: " +
                  allBirthDate[i].childNodes[0].nodeValue + "\n";
               userStatement += "Gender: " +
                  allGender[i].childNodes[0].nodeValue + "\n";
               userStatement += "Email: " +
                  allEmail[i].childNodes[0].nodeValue + "\n";
               userStatement += "Username: " +
                  allUsers[i].childNodes[0].nodeValue + "\n\n\n";

           }
           document.getElementById("textBoxDivision1").style =
              "display:block";
           document.getElementById("textBox1").className =
              "textArea1";
           document.getElementById("textBox1").value =
              userStatement;
           document.getElementById("closeWindowButton").disabled=
              false;
           document.getElementById("closeWindowButton").className=
              ".closeButton";
        }
    }
}
```

```
            // Send the POST request
            xmlhttp.setRequestHeader('Content-Type', 'text/xml');
            xmlhttp.send(sr);
        }


        function closeButtonMethod1(){
            document.getElementById("textBoxDivision1").style =
               "display:none";
        }

        function logout() {
            document.getElementById("login_form").style = "login_form";
            document.getElementById("buttons").style = "display:none";
            document.getElementById("textBoxDivision1").style =
               "display:none";
            document.getElementById("addUser_form").style = "display:none";
            document.getElementById("sendMessage_form").style =
               "display:none";
            document.getElementById("update_form").style = "display:none";
            document.getElementById("remove_form").style = "display:none";
        }
    </script>
</head>


<div id = "buttons" class = "buttonContainer" style = "display:none">
    <button class = "buttonEnabled" id = "inboxButton"
onclick="inboxRequest()" disabled>Inbox</button>
    <button class = "buttonEnabled" id = "outboxButton"
onclick="outboxRequest()" disabled>Outbox</button>
    <button class = "buttonEnabled" id = "sendButton"
onclick="sendMessageRequest()" disabled>Send Message</button>

    <button class = "buttonEnabled" id = "addButton"
onclick="addUserRequest()" disabled>Add User</button>
    <button class = "buttonEnabled" id = "updateButton"
onclick="updateRequest()" disabled>Update User</button>
    <button class = "buttonEnabled" id = "removeButton"
onclick="removeRequest()" disabled>Remove User</button>
    <button class = "buttonEnabled" id = "listButton" onclick="listRequest()"
disabled>List Users</button>
    <button class = "buttonEnabled" id = "logoutButton" onclick="logout()"
disabled>Logout</button>
</div>

<form name ="login"  class =
      "sendMessageContainer" id="login_form" action="#"  autocomplete="off">

    Username: <input type="text" name=
      "Username" id="txt_username" class="textForm"/>

    Password: <input type="password" name=
      "Password" id="txt_password" class="textForm"/>

    <input type="button" name="login" value=
      "LOGIN"  onclick="loginRequest()" id="loginBtn" class="textForm">
</form>
```

```
<div id = "textBoxDivision1" class="textarea-container" style =
    "display:none">
    <textarea id ="textBox1" name="textBox1" class = "textArea1"></textarea>
    <button class = "buttonDisabled" id = "closeWindowButton"
       onclick="closeButtonMethod1()" disabled>Close</button>
</div>

<form name ="sendMessage"  class = "sendMessageContainer"
id="sendMessage_form" action="#" style="display:none">

    Receiver: <input type="text" name=
       "Receiver" id="txt_receiver" class="textForm"/>

    Title: <input type="text" name="Title" id="txt_title" class="textForm"/>

    Message: <input type="text" name=
       "Message" id="txt_message" class="textForm">

    <input type="button" name="send" value=
       "SEND"  id="sendMessageButton" disabled class="textForm"/>
</form>

<form name ="addUser"  class = "sendMessageContainer" id="addUser_form"
action="#" style="display:none">

    First Name: <input type="text" name=
       "Receiver" id="txt_fn" class="textForm"/>

    Last Name: <input type="text" name=
       "Title" id="txt_ln" class="textForm"/>

    Birthdate (YYYY/MM/DD): <input type="text" name=
       "Message" id="txt_bd" class="textForm">

    Gender (M/F): <input type="text" name=
       "Gender" id="txt_g" class="textForm"/>

    Email: <input type="text" name=
       "Email" id="txt_e" class="textForm"/>

    Username: <input type="text" name=
       "Username" id="txt_u" class="textForm">

    Password: <input type="text" name=
       "Password" id="txt_p" class="textForm"/>

    Admin Info (T/F): <input type="text" name=
       "Admin Info" id="txt_a" class="textForm"/>

    <input type="button" name="add" value=
       "ADD USER"  id="addUserButton" disabled class="textForm"/>
</form>

<form name ="removeUser"  class = "sendMessageContainer" id=
       "remove_form" action="#" style="display:none">

    Username: <input type="text" name="UsernameToBeRemoved" id=
       "txt_uR" class="textForm"/>

    <input type="button" name="remove" value=
       "REMOVE USER"  id="removeUserButton" disabled class="textForm"/>
</form>
```

```
<form name ="updateUser"  class = "sendMessageContainer" id=
       "update_form" action="#" style="display:none">

    Username: <input type="text" name="UsernameToBeUpdated" id=
       "txt_uU" class="textForm"/>
    Data To Update (firstname, lastname, birthdate (YYYY/MM/DD),
       gender (M/F), email, username, password, admin (T/F)):
       <input type="text" name="UsernameToBeUpdated" id=
       "txt_dU" class="textForm"/>
    New Data: <input type="text" name="UsernameToBeUpdated" id=
       "txt_nU" class="textForm"/>

    <input type="button" name="update" value=
       "UPDATE USER"  id="updateUserButton" disabled class="textForm"/>
</form>
</html>
```

C. Third Assignment : Web based User Management & Messaging Tool, Version 2

```
# README File describing how to run the messaging application


# MessageApp

This project was generated with [Angular
CLI](https://github.com/angular/angular-cli) version 10.0.4.

## Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`.
The app will automatically reload if you change any of the source files.

## Code scaffolding

Run `ng generate component component-name` to generate a new component.
You can also use `ng generate
directive|pipe|service|class|guard|interface|enum|module`.

## Build

Run `ng build` to build the project. The build artifacts will be stored
in the `dist/` directory. Use the `--prod` flag for a production build.

## Running unit tests

Run `ng test` to execute the unit tests via [Karma](https://karma-
runner.github.io).

## Running end-to-end tests

Run `ng e2e` to execute the end-to-end tests via
[Protractor](http://www.protractortest.org/).

## Further help

To get more help on the Angular CLI use `ng help` or go check out the
[Angular CLI README](https://github.com/angular/angular-
cli/blob/master/README.md).
```

```
//**********************************************************
//                   MUSTAFA GÖKTAN GÜDÜKBAY                *
// SRDC Software Research & Development and Consultancy Corp. *
//**********************************************************

// JavaScript implementation of the server

let express = require('express'),
   path = require('path'),
   mongoose = require('mongoose'),
   cors = require('cors'),
   bodyParser = require('body-parser'),
   dbConfig = require('./database/db');

// Connecting with Mongo db
mongoose.Promise = global.Promise;
mongoose.connect(dbConfig.db, {
   useNewUrlParser: true
}).then(() => {
      console.log('Database sucessfully connected')
   },
   error => {
      console.log('Database could not connected: ' + error)
   }
)

// Setting up port with express js
const userRoute = require('../backend/routes/user.route')
const app = express();
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
   extended: false
}));

app.use(cors());
app.use(express.static(path.join(__dirname, 'dist/mean-stack-crud-app')));
app.use('/', express.static(path.join(__dirname, 'dist/mean-stack-crud-app')));
app.use('/api', userRoute)

// Create port
const port = process.env.PORT || 4000;
const server = app.listen(port, () => {
  console.log('Connected to port ' + port)
})

// Find 404 and hand over to error handler
app.use((req, res, next) => {
   next(createError(404));
});

// error handler
app.use(function (err, req, res, next) {
  console.error(err.message); // Log error message in our server's console

  if (!err.statusCode) err.statusCode = 500;
  // If err has no specified error code, set error code to
  // 'Internal Server Error (500)'
  res.status(err.statusCode).send(err.message);
  // All HTTP requests must have a response, so let's send back an error with
  // its status code and message
});
```

```
// JavaScript implementation of the methods for the messaging application

const express = require('express');
const crypto = require('crypto');
const jwt = require('jsonwebtoken');
const app = express();
const userRoute = express.Router();
var cors = require('cors');
app.use(cors());
dbConfig = require('C:/Program Files/angular-1.8.0/backend/database/db');

app.use(express.json());

var secretNumber = crypto.randomBytes(64).toString('hex');

// Models

let User = require('../models/User');
let Message = require('../models/Message');
let AccessLog = require('../models/AccessLog');


// login method (will return information of the user)

userRoute.route('/login').post((req, res, next) => {
  User.findOne({username: req.body.username, password: req.body.password},
(error, data) => {
    if (data == null || error) {
     return next(error);
    }
   else {
     data.authToken = "";
     let o = {"username":data.username, "isAdmin":data.isAdmin}
     var accessToken = jwt.sign(o, secretNumber, {algorithm: "HS256"});
     User.findOneAndUpdate({'username': req.body.username},
             {authToken: accessToken}, {new: true},(err, result) => {
      result = result.toObject();
      console.log(result);
      res.json(result);
       })
     }
  })
})

// get user method (will return information of the user for update

userRoute.route('/:username').get((req, res, next) => {
  var receivedToken = req.headers.authorization;
  let p = new Promise((resolve, reject) => {
     if (receivedToken.startsWith("Bearer ")){
       receivedToken = receivedToken.substring(7, receivedToken.length);
     }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
  })
  p.then((message) => {
     console.log(req.params.username);
  User.findOne({username: req.params.username}, (error, data) => {
    if (data == null || error)
     return next(error);
   else
      res.json(data);
     })
 }).catch((message) => {
    return next(error);
 })
})

// send message (will return a message if it is succesful or not)

userRoute.route('/sendMessage').post((req, res, next) => {
  var receivedToken = req.headers.authorization;
  let p = new Promise((resolve, reject) => {
     if (receivedToken.startsWith("Bearer ")){
       receivedToken = receivedToken.substring(7, receivedToken.length);
     }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
   })

  p.then((message) => {

     var receiverId = -1, senderId = -1;
     let p2 = new Promise((resolve, reject) => {
        console.log(req.body.senderUsername + req.body.receiverUsername);
        User.findOne({username: req.body.senderUsername}, (error, data) =>{
        if(data != null )
         senderId = data._id;
        else
           reject('failed2');
        })
        User.findOne({username: req.body.receiverUsername}, (error, data) =>{
        if(data != null ){
         receiverId = data._id;
         resolve('success');
        }
        else
           reject('failed3');
        })
     })
     p2.then((message) => {
        if(receiverId == -1 || senderId == -1){
         res.status(401).send({ error: "Wrong sender or receiver." });
         console.log("Failed to send the message.");
        }
        else{
           var dt = new Date();
           var messageToPut = {
            sender_id: senderId,
             receiver_id: receiverId,
             time: dt,
             title: req.body.title,
             message: req.body.message
            };

            Message.create(messageToPut, (error, data) => {
               if(error)
                  return next(error);
               else
                  res.json(data);
            })
         }
      }).catch((message2) => {
         res.status(500).send("Message was not sent.");
       })

   }).catch((message) => {
       console.log("Token Error");
   })
})


// read inbox

userRoute.route('/inbox/:username').get((req, res, next) => {
   var receivedToken = req.headers.authorization;
   console.log(receivedToken);
   let p = new Promise((resolve, reject) => {
     if (receivedToken.startsWith("Bearer ")){
       receivedToken = receivedToken.substring(7, receivedToken.length);
     }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
   })
  p.then((message) => {
   usernameOfUser = req.params.username;
   Message.aggregate([
   {
      $lookup:
      {
         from: 'users',
         localField: 'sender_id',
         foreignField: '_id',
         as: 'messageDetailsSender'
      }
   },
   { $unwind: "$messageDetailsSender"},
   {
      $lookup:
      {
         from: 'users',
         localField: 'receiver_id',
         foreignField: '_id',
         as: 'messageDetailsReceiver'
      }
   },
   {$unwind: "$messageDetailsReceiver"},

   // condition
   {
      $match:{
         $and:[{"messageDetailsReceiver.username" : usernameOfUser}]
      }
   },

      // which fields
   {
      $project:{
         _id : 1,
         senderUsername: "$messageDetailsSender.username",
         senderFirstName: "$messageDetailsSender.firstname",
         senderLastName: "$messageDetailsSender.lastname",
         receiverUsername: "$messageDetailsReceiver.username",
         receiverFirstName: "$messageDetailsReceiver.firstname",
         receiverLastName: "$messageDetailsReceiver.lastname",
         time: 1,
         title: 1,
         message: 1,
      }
   }
   ], (error, result) => {
      if(error)
         res.status(500).send("Error");
      else{
         res.status(200).json(result);
      }
   }
   )
  }).catch((message) => {
      console.log("Token error");
  })
})


// read outbox

userRoute.route('/outbox/:username').get((req, res, next) => {
   var receivedToken = req.headers.authorization;
   let p = new Promise((resolve, reject) => {
     if (receivedToken.startsWith("Bearer ")){
       receivedToken = receivedToken.substring(7, receivedToken.length);
     }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
   })
  p.then((message) => {
   var usernameOfUser = req.params.username;
   Message.aggregate([
   {
        $lookup:
        {
           from: 'users',
           localField: 'sender_id',
           foreignField: '_id',
           as: 'messageDetailsSender'
        }
     },
     { $unwind: "$messageDetailsSender"},
     {
        $lookup:
        {
           from: 'users',
           localField: 'receiver_id',
           foreignField: '_id',
           as: 'messageDetailsReceiver'
        }
     },
     {$unwind: "$messageDetailsReceiver"},

     // condition
     {
        $match:{
           $and:[{"messageDetailsSender.username" : usernameOfUser}]
        }
     },

        //which fields
     {
        $project:{
           _id : 1,
           senderUsername: "$messageDetailsSender.username",
           senderFirstName: "$messageDetailsSender.firstname",
           senderLastName: "$messageDetailsSender.lastname",
           receiverUsername: "$messageDetailsReceiver.username",
           receiverFirstName: "$messageDetailsReceiver.firstname",
           receiverLastName: "$messageDetailsReceiver.lastname",
           time: 1,
           title: 1,
           message: 1,
        }
     }
     ], (error, result) => {
        if(error)
           return next(error);
        else{
           res.json(result);
        }
     }
     )
    }).catch((message) => {
        console.log("Token error");
    })
})

// add user

userRoute.route('/').post((req, res, next) => {
  var receivedToken = req.headers.authorization;

  let p = new Promise((resolve, reject) => {
       if (receivedToken.startsWith("Bearer ")){
         receivedToken = receivedToken.substring(7, receivedToken.length);
       }
     User.findOne({authToken: receivedToken}, (error, data) => {
     if((data == null) || error || !data.isAdmin){
        reject('failed');
        return next(error);
     }
     else
        resolve('success');
     })
     })
  p.then((message) => {
       var userToBeInserted = req.body;
       userToBeInserted.birthdate =
               new Date(userToBeInserted.birthdate.substring(0, 10));
  User.create(userToBeInserted, (error, data) => {
       if (error) {
        return next(error);
       } else {
         res.json(data);
       }
  })
  }).catch((message) => {
  })
})

// list users

userRoute.route('/').get((req, res, next) => {
  var receivedToken = req.headers.authorization;
  var wrongToken = false;
     let p = new Promise((resolve, reject) => {
       if (receivedToken.startsWith("Bearer ")){
         receivedToken = receivedToken.substring(7, receivedToken.length);
       }
     User.findOne({authToken: receivedToken}, (error, data) => {
     if((data == null) || error || !data.isAdmin){
        reject('failed');
        return next(error);
     }
     else
        resolve('success');
     })
     })

  p.then((Message) => {
       var sort = {};
       sort[req.query.sortField] = req.query.sortType;
  User.find((error, data) => {
       if (error) {
         return next(error);
       } else {
         res.json(data)
       }
     })

     }).catch((Message) => {
        console.log("Token error");
     })
})
```

```
// update user

userRoute.route('/').put((req, res, next) => {

   var receivedToken = req.headers.authorization;
   let p = new Promise((resolve, reject) => {
     if (receivedToken.startsWith("Bearer ")){
       receivedToken = receivedToken.substring(7, receivedToken.length);
     }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error  || !data.isAdmin){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
   })

  p.then((Message) => {
   var userToBeInserted = req.body;
   userToBeInserted.birthdate =
                new Date(userToBeInserted.birthdate.substring(0, 10));

   User.findOneAndUpdate({username: userToBeInserted.username},
   {$set:userToBeInserted},
   function(error, result){
      if(error)
         return next(error);
      else{
         res.json(result);

      }
   });
   }).catch((Message) => {
      console.log("Token error");
   })
})

// remove user

userRoute.route('/:username').delete((req, res, next) => {
   var receivedToken = req.headers.authorization;
   let p = new Promise((resolve, reject) => {
     if (receivedToken.startsWith("Bearer ")){
       receivedToken = receivedToken.substring(7, receivedToken.length);
     }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error  || !data.isAdmin){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
   })
  p.then((Message) => {
   User.deleteOne({ username: req.params.username}, (error, result) => {
      if(error)
         return next(error);
      else
         res.json(result);
  })
  }).catch((Message) => {
      console.log("Token error");
  })
})

// post to access log

userRoute.route('/log').post((req, res, next) => {
  var receivedToken = req.headers.authorization;

  let p = new Promise((resolve, reject) => {
    if (receivedToken.startsWith("Bearer ")){
      receivedToken = receivedToken.substring(7, receivedToken.length);
    }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
   })

  p.then((message) => {
  AccessLog.create(req.body, (error, data) => {
    if (error) {
     return next(error);
    } else {
      res.json(data);
    }
  })
  }).catch((message) => {
    return next(error);
  })
})

// get access log

userRoute.route('/log').get((req, res, next) => {
   var receivedToken = req.headers.authorization;
   var wrongToken = false;
   let p = new Promise((resolve, reject) => {
     if (receivedToken.startsWith("Bearer ")){
       receivedToken = receivedToken.substring(7, receivedToken.length);
     }
   User.findOne({authToken: receivedToken}, (error, data) => {
   if((data == null) || error || !data.isAdmin){
      reject('failed');
      return next(error);
   }
   else
      resolve('success');
   })
   })
  p.then((Message) => {
   AccessLog.find((error, data) => {
     if (error) {
       return next(error);
     } else {
       res.json(data)
     }
   })

  }).catch((Message) => {
      console.log("Token error");
  })
})

module.exports = userRoute;
```

```
// Message JavaScript Collection and Schema

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

// Define collection and schema
let Message = new Schema({
   sender_id: {
      type: Schema.ObjectId,
     ref: 'User'
   },
   receiver_id: {
      type: Schema.ObjectId,
     ref: 'User'
   },
   time: {
      type: Date
   },
   title: {
      type: String
   },
   message: {
     type: String
   }
}, {
   collection: 'messages'
})

module.exports = mongoose.model('Message', Message)
```

```
// User JavaScript Collection and Schema

const mongoose = require('mongoose');
const Schema = mongoose.Schema;
mongoose.set('useFindAndModify', false);

// Define collection and schema

let Message = new Schema({
  sender: {
     type: String
  },
  receiver: {
     type: String
  },
   time: {
     type: Date
   },
    title: {
     type: String
   },
    message: {
     type: String
   }
})


let User = new Schema({
   username: {
     type: String
   },
   firstname: {
     type: String
   },
   lastname: {
     type: String
   },
   birthdate: {
     type: Date
   },
   email: {
    type: String
   },
   gender: {
    type: String
   },
   password: {
    type: String
   },
   isAdmin: {
     type: Boolean
   },
   authToken: {
    type: String
   }
}, {
   collection: 'users'
})


module.exports = mongoose.model('User', User)
```

```
// AccessLog JavaScript Collection and Schema

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

// Define collection and schema
let AccessLog = new Schema({
   username: {
      type: String
   },
   login_Time: {
    type: Date
   },
   logout_Time: {
     type: Date
   },
   ip: {
      type: String
   },
   browser: {
    type: String
   }
}, {
   collection: 'AccessLogs'
})

module.exports = mongoose.model('AccessLog', AccessLog)
```

```
// Message Class

export class Message {
    time: Date;
    title: string;
    message: string;
    senderUsername: string;
    senderFirstName: string;
    senderLastName: string;
    receiverUsername: string;
    receiverFirstName: string;
    receiverLastName: string;
}

// User Class

export class User {
    username: string;
    firstname: string;
    lastname: string;
    birthdate: Date;
    email: string;
    gender: string;
    password: string;
    isAdmin: boolean;
}

// User Specification

import { User } from './user';

describe('User', () => {
  it('should create an instance', () => {
    expect(new User()).toBeTruthy();
  });
});
```

```
// Typescript Code for API Service

import { Injectable } from '@angular/core';
import { Observable, throwError } from 'rxjs';
import { catchError, map } from 'rxjs/operators';
import { HttpClient, HttpHeaders, HttpErrorResponse } from
'@angular/common/http';

@Injectable({
  providedIn: 'root'
})

export class ApiService {

  baseUri:string = 'http://localhost:4000/api';
  headers = new HttpHeaders().set('Content-Type', 'application/json');

  constructor(private http: HttpClient) { }

  // Create User

  createUser(data, token): Observable<any> {
    let url = `${this.baseUri}/`;
    var reqHeader = new HttpHeaders({
      'Content-Type': 'application/json',
      'Authorization': 'Bearer ' + token
   });
    return this.http.post(url, data, {headers: reqHeader})
      .pipe(
        catchError(this.errorMgmt)
      )
  }

  // Get all Users

  getUsers(token) {
    let url = `${this.baseUri}/`;
    var reqHeader = new HttpHeaders({
      'Content-Type': 'application/json',
      'Authorization': 'Bearer ' + token
   });
    return this.http.get(url, {headers: reqHeader})
    .pipe(
      catchError(this.errorMgmt)
    )
  }

  // Get one User for Update

  getUser(token, username){
    let url = `${this.baseUri}/${username}`;
    var reqHeader = new HttpHeaders({
      'Content-Type': 'application/json',
      'Authorization': 'Bearer ' + token
   });
   return this.http.get(url, {headers: reqHeader}).pipe(
    catchError(this.errorMgmt)
   )
  }
```

```
// Login method

 login(data): Observable<any> {
   let url = `${this.baseUri}/login`;
   return this.http.post(url, data).pipe(
     map((res: Response) => {
       return res || {}
     }),
     catchError(this.errorMgmt)
   )
 }

 // Update user

 updateUser(data, token): Observable<any> {
   let url = `${this.baseUri}/`;
   var reqHeader = new HttpHeaders({
     'Content-Type': 'application/json',
     'Authorization': 'Bearer ' + token
  });
   return this.http.put(url, data, { headers: reqHeader }).pipe(
     catchError(this.errorMgmt)
   )
 }

 // Delete user

 deleteUser(username, token): Observable<any> {
   let url = `${this.baseUri}/${username}`;
   var reqHeader = new HttpHeaders({
     'Content-Type': 'application/json',
     'Authorization': 'Bearer ' + token
  });

   return this.http.delete(url, { headers: reqHeader }).pipe(
     catchError(this.errorMgmt)
   )
 }

 // Send Message

 sendMessage(data, token): Observable<any> {
   let url = `${this.baseUri}/sendMessage`;
   var reqHeader = new HttpHeaders({
     'Content-Type': 'application/json',
     'Authorization': 'Bearer ' + token
  });

  console.log(token);
  console.log(data);
  return this.http.post(url, data, {headers: reqHeader}).pipe(
   catchError(this.errorMgmt)
  )
 }

 // Read Inbox

 readInbox(token, username): Observable<any> {
   let url = `${this.baseUri}/inbox/${username}`;
```

```
     var reqHeader = new HttpHeaders({
       'Content-Type': 'application/json',
       'Authorization': 'Bearer ' + token
    });
    return this.http.get(url, {headers: reqHeader}).pipe(
     catchError(this.errorMgmt)
    )
   }


// Read Outbox

readOutbox(token, username): Observable<any> {
 let url = `${this.baseUri}/outbox/${username}`;
  var reqHeader = new HttpHeaders({
    'Content-Type': 'application/json',
   'Authorization': 'Bearer ' + token
 });
  return this.http.get(url, {headers: reqHeader}).pipe(
    catchError(this.errorMgmt)
   )
}

// Create new access

newAccess(data, token): Observable<any> {
  console.log("a");
  console.log(data);
  console.log(token);
  let url = `${this.baseUri}/log`;
  var reqHeader = new HttpHeaders({
    'Content-Type': 'application/json',
    'Authorization': 'Bearer ' + token
 });
 console.log(url);
 console.log(reqHeader);

  return this.http.post(url, data, {headers: reqHeader})
    .pipe(
      catchError(this.errorMgmt)
    )
}

// Get access log

listAccess(token): Observable<any> {
  let url = `${this.baseUri}/log`;
  var reqHeader = new HttpHeaders({
    'Content-Type': 'application/json',
    'Authorization': 'Bearer ' + token
 });
  return this.http.get(url, {headers: reqHeader})
    .pipe(
      catchError(this.errorMgmt)
    )
}

// Error handling

errorMgmt(error: HttpErrorResponse) {
  let errorMessage = '';
```

```
    if (error.error instanceof ErrorEvent) {
      // Get client-side error
      errorMessage = error.error.message;
    } else {
      // Get server-side error
      errorMessage = `Error Code: ${error.status}\nMessage: ${error.message}`;
    }
    console.log(errorMessage);
    return throwError(errorMessage);
  }

}


// Typescript Code for API Service Specification

import { TestBed } from '@angular/core/testing';

import { ApiService } from './api.service';

describe('ApiService', () => {
  let service: ApiService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(ApiService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

```
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * -->
<!—* * * * * * * * * * Application Component HTML File * * * * * * * -->
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * -->

<style>
  :host {
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto,
             Helvetica, Arial, sans-serif, "Apple Color Emoji",
             "Segoe UI Emoji", "Segoe UI Symbol";
    font-size: 14px;
    color: #333;
    box-sizing: border-box;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }

  h1,
  h2,
  h3,
  h4,
  h5,
  h6 {
    margin: 8px 0;
  }

  p {
    margin: 0;
  }

  .spacer {
    flex: 1;
  }

  .toolbar {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    height: 60px;
    display: flex;
    align-items: center;
    background-color: #1976d2;
    color: white;
    font-weight: 600;
  }

  .toolbar img {
    margin: 0 16px;
  }

  .toolbar #twitter-logo {
    height: 40px;
    margin: 0 16px;
  }

  .toolbar #twitter-logo:hover {
    opacity: 0.8;
  }

  .content {
    display: flex;
  margin: 82px auto 32px;
  padding: 0 16px;
  max-width: 960px;
  flex-direction: column;
  align-items: center;
}

svg.material-icons {
  height: 24px;
  width: auto;
}

svg.material-icons:not(:last-child) {
  margin-right: 8px;
}

.card svg.material-icons path {
  fill: #888;
}

.card-container {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  margin-top: 16px;
}

.card {
  border-radius: 4px;
  border: 1px solid #eee;
  background-color: #fafafa;
  height: 40px;
  width: 200px;
  margin: 0 8px 16px;
  padding: 16px;
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  transition: all 0.2s ease-in-out;
  line-height: 24px;
}

.card-container .card:not(:last-child) {
  margin-right: 0;
}

.card.card-small {
  height: 16px;
  width: 168px;
}

.card-container .card:not(.highlight-card) {
  cursor: pointer;
}

.card-container .card:not(.highlight-card):hover {
  transform: translateY(-3px);
  box-shadow: 0 4px 17px rgba(0, 0, 0, 0.35);
}
```

```
.card-container .card:not(.highlight-card):hover .material-icons path {
  fill: rgb(105, 103, 103);
}

.card.highlight-card {
  background-color: #1976d2;
  color: white;
  font-weight: 600;
  border: none;
  width: auto;
  min-width: 30%;
  position: relative;
}

.card.card.highlight-card span {
  margin-left: 60px;
}

svg#rocket {
  width: 80px;
  position: absolute;
  left: -10px;
  top: -24px;
}

svg#rocket-smoke {
  height: calc(100vh - 95px);
  position: absolute;
  top: 10px;
  right: 180px;
  z-index: -10;
}

a,
a:visited,
a:hover {
  color: #1976d2;
  text-decoration: none;
}

a:hover {
  color: #125699;
}

.terminal {
  position: relative;
  width: 80%;
  max-width: 600px;
  border-radius: 6px;
  padding-top: 45px;
  margin-top: 8px;
  overflow: hidden;
  background-color: rgb(15, 15, 16);
}

.terminal::before {
  content: "\2022 \2022 \2022";
  position: absolute;
  top: 0;
  left: 0;
  height: 4px;
  background: rgb(58, 58, 58);
  color: #c2c3c4;
  width: 100%;
  font-size: 2rem;
  line-height: 0;
  padding: 14px 0;
  text-indent: 4px;
}

.terminal pre {
  font-family: SFMono-Regular,Consolas,Liberation Mono,Menlo,monospace;
  color: white;
  padding: 0 1rem 1rem;
  margin: 0;
}

.circle-link {
  height: 40px;
  width: 40px;
  border-radius: 40px;
  margin: 8px;
  background-color: white;
  border: 1px solid #eeeeee;
  display: flex;
  justify-content: center;
  align-items: center;
  cursor: pointer;
  box-shadow: 0 1px 3px rgba(0, 0, 0, 0.12), 0 1px 2px rgba(0, 0, 0, 0.24);
  transition: 1s ease-out;
}

.circle-link:hover {
  transform: translateY(-0.25rem);
  box-shadow: 0px 3px 15px rgba(0, 0, 0, 0.2);
}

footer {
  margin-top: 8px;
  display: flex;
  align-items: center;
  line-height: 20px;
}

footer a {
  display: flex;
  align-items: center;
}

.github-star-badge {
  color: #24292e;
  display: flex;
  align-items: center;
  font-size: 12px;
  padding: 3px 10px;
  border: 1px solid rgba(27,31,35,.2);
  border-radius: 3px;
  background-image: linear-gradient(-180deg,#fafbfc,#eff3f6 90%);
  margin-left: 4px;
  font-weight: 600;
```

```
      font-family: -apple-system,BlinkMacSystemFont,Segoe
        UI,Helvetica,Arial,sans-serif,Apple Color Emoji,
        Segoe UI Emoji,Segoe UI Symbol;
    }

    .github-star-badge:hover {
      background-image: linear-gradient(-180deg,#f0f3f6,#e6ebf1 90%);
      border-color: rgba(27,31,35,.35);
      background-position: -.5em;
    }

    .github-star-badge .material-icons {
      height: 16px;
      width: 16px;
      margin-right: 4px;
    }

    svg#clouds {
      position: fixed;
      bottom: -160px;
      left: -230px;
      z-index: -10;
      width: 1920px;
    }


    /* Responsive Styles */
    @media screen and (max-width: 767px) {

      .card-container > *:not(.circle-link) ,
      .terminal {
        width: 100%;
      }

      .card:not(.highlight-card) {
        height: 16px;
        margin: 8px 0;
      }

      .card.highlight-card span {
        margin-left: 72px;
      }

      svg#rocket-smoke {
        right: 120px;
        transform: rotate(-5deg);
      }
    }

    @media screen and (max-width: 575px) {
      svg#rocket-smoke {
        display: none;
        visibility: hidden;
      }
    }
  </style>
```

```
<nav class="navbar navbar-expand-sm bg-dark" *ngIf="router.url != '/login'">
  <div class="container">
      <a class="navbar-brand">VisChat</a>
      <ul class="nav navbar-nav" routerLinkActive="active">
          <li class="nav-item"><a class="nav-link"
                routerLink="inbox">Inbox</a></li>
          <li class="nav-item"><a class="nav-link"
                routerLink="outbox">Outbox</a></li>
          <li class="nav-item"><a class="nav-link"
                routerLink="send-message">Send Message</a></li>
          <li class="nav-item"><a class="nav-link"
                routerLink="create-user"
                      *ngIf='nav.adminOption'>Create User</a></li>
          <li class="nav-item"><a class="nav-link"
                routerLink="user-list"
                      *ngIf='nav.adminOption'>List User</a></li>
          <li class="nav-item"><a class="nav-link" routerLink="access-log"
                      *ngIf='nav.adminOption'>Access Log</a></li>
          <li class="nav-item"><a class="nav-link" (click)="logout()"
                routerLink="login">Logout</a></li>
      </ul>
  </div>
</nav>

<router-outlet></router-outlet>
```

```
// TypeScript code for the Application Module

import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { MatInputModule} from '@angular/material/input';
import { MatTableModule} from '@angular/material/table';
import { MatDatepickerModule} from '@angular/material/datepicker';
import { MatNativeDateModule} from '@angular/material/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { LoginComponent } from './components/login/login.component';
import { MenuComponent } from './components/menu/menu.component';
import { ReadMessageComponent } from './components/read-message/read-
message.component';
import { SendMessageComponent } from './components/send-message/send-
message.component';
import { AddUpdateComponent } from './components/add-update/add-
update.component';
import { ListComponent } from './components/list/list.component';
import { AccessLogComponent } from './components/access-log/access-
log.component';
import { RemoveComponent } from './components/remove/remove.component';
import {MatSort, MatSortModule} from '@angular/material/sort';

import { HttpClientModule } from '@angular/common/http';

import { ApiService } from './service/api.service';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';

@NgModule({
  declarations: [
    AppComponent,
    LoginComponent,
    MenuComponent,
    ReadMessageComponent,
    SendMessageComponent,
    AddUpdateComponent,
    ListComponent,
    AccessLogComponent,
    RemoveComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule,
    MatInputModule,
    MatTableModule,
    BrowserAnimationsModule,
    MatDatepickerModule,
    MatNativeDateModule,
    MatSortModule
  ],
  providers: [ApiService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```
// TypeScript code for the Application Component Module

import { TestBed, async } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  }));

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'message-app'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('message-app');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement;
    expect(compiled.querySelector('.content
span').textContent).toContain('message-app app is running!');
  });
});
```

```
// TypeScript code for the Application Routing Module

import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { LoginComponent } from './components/login/login.component';
import { MenuComponent } from './components/menu/menu.component';
import { ReadMessageComponent } from './components/read-message/read-
message.component';
import { SendMessageComponent } from './components/send-message/send-
message.component';
import { AddUpdateComponent } from './components/add-update/add-
update.component';
import { ListComponent } from './components/list/list.component';
import { AccessLogComponent } from './components/access-log/access-
log.component';
import { RemoveComponent } from './components/remove/remove.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

const routes: Routes = [
  { path: '', pathMatch: 'full', redirectTo: 'login' },
  { path: 'create-user', component: AddUpdateComponent },
  { path: 'edit-user/:username', component: AddUpdateComponent },
  { path: 'user-list', component: ListComponent },
  { path: 'remove/:username', component: RemoveComponent },
  { path: 'inbox', component: ReadMessageComponent },
  { path: 'outbox', component: ReadMessageComponent },
  { path: 'send-message', component: SendMessageComponent },
  { path: 'access-log', component: AccessLogComponent },
  { path: 'menu', component: MenuComponent},
  { path: 'login', component: LoginComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes), FormsModule, ReactiveFormsModule],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

```
<!—Login Component HTML File  -->

<div class="row justify-content-center">
    <div class="col-md-4 login">
      <!-- form card register -->
      <div class="card-body">
        <form [formGroup]="loginForm" (ngSubmit)="onSubmit()">
          <div class="form-group">
            <label for="inputName">Username</label>
            <input class="form-control" type="text"
                  formControlName="username">
            <!-- error -->
            <div class="invalid-feedback" *ngIf="submitted &&
                   myForm.username.errors?.required">
              Username is required.
            </div>
          </div>

          <div class="form-group">
           <label for="inputPassword">Password</label>
           <input class="form-control" type="password"
                  formControlName="password">
           <!-- error -->
           <div class="invalid-feedback" *ngIf="submitted &&
                  myForm.password.errors?.required">
              Password is required.
            </div>
          </div>



          <div class="form-group">
            <button class="btn btn-success btn-lg btn-block"
                  type="submit">Login</button>
          </div>
        </form>

      </div>
    </div><!-- form card register -->
  </div>
```

```
// TypeScript code for the Login Component

import { Component, OnInit, NgZone } from '@angular/core';
import { Router } from '@angular/router';
import { ApiService } from './../../service/api.service';
import { FormGroup, FormBuilder, Validators } from "@angular/forms";
import { HttpClient  } from '@angular/common/http';
import { HeroService } from './../../hero.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {
  submitted = false;
  loginForm: FormGroup;

  constructor(
    public fb: FormBuilder,
    public router: Router,
    public ngZone: NgZone,
    public apiService: ApiService,
    public http: HttpClient,
    public nav: HeroService
  ) {
    this.mainForm();
  }

  ngOnInit(): void {
  }

  mainForm() {
    this.loginForm = this.fb.group({
      username: ['', [Validators.required]],
      password: ['', [Validators.required]]
    })
  }

  // Getter to access form control
  get myForm(){
    return this.loginForm.controls;
  }

  public getIPAddress()
  {
    return this.http.get("http://api.ipify.org/?format=json");
  }

  public getBrowserName() {
    const agent = window.navigator.userAgent.toLowerCase()
    switch (true) {
      case agent.indexOf('edge') > -1:
        return 'edge';
      case agent.indexOf('opr') > -1 && !!(<any>window).opr:
        return 'opera';
      case agent.indexOf('chrome') > -1 && !!(<any>window).chrome:
        return 'chrome';
      case agent.indexOf('trident') > -1:
        return 'ie';
      case agent.indexOf('firefox') > -1:
        return 'firefox';
      case agent.indexOf('safari') > -1:
        return 'safari';
      default:
        return 'other';
    }
}

  onSubmit() {
    this.submitted = true;
    if (!this.loginForm.valid) {
      return false;
    } else {
      this.apiService.login(this.loginForm.value).subscribe(
        (res) => {
          console.log(res);
          sessionStorage.setItem("userInfo", JSON.stringify(res));
          console.log(sessionStorage.userInfo);
          console.log(JSON.parse(sessionStorage.userInfo).isAdmin);
          if(JSON.parse(sessionStorage.userInfo).isAdmin)
            this.nav.show();
          else
            this.nav.hide();
          this.getIPAddress().subscribe((ipResponse:any) =>{
            console.log(ipResponse.ip);
            sessionStorage.setItem("ip", ipResponse.ip);
          });
          console.log(sessionStorage.getItem("ip"));

          sessionStorage.setItem("browserName", (this.getBrowserName()));
          var dt = new Date();
          console.log(dt);
          sessionStorage.setItem("loginTime", dt.toString());
          console.log(sessionStorage.getItem("browserName"));
          this.ngZone.run(() => this.router.navigateByUrl('menu'))
        }, (error) => {
        });
    }
  }

}
```

```
// TypeScript code for the Login Component Specification

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { LoginComponent } from './login.component';

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ LoginComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(LoginComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
<!-- Access Log Component HTML File  -->

<table #dataTable mat-table [dataSource]="dataSource" matSort>
    <!-- COLUMN INFO -->
     <!-- Sender Username Col -->
      <ng-container matColumnDef="username">
        <th style="width: 135px" mat-header-cell
                     *matHeaderCellDef mat-sort-header>Username</th>
        <td  mat-cell *matCellDef="let access"> {{access.username}} </td>
      </ng-container>

      <ng-container matColumnDef="login_Time">
        <th style="width: 125px" mat-header-cell
                     *matHeaderCellDef mat-sort-header>Login Time</th>
        <td  mat-cell *matCellDef="let access"> {{access.login_Time}} </td>
      </ng-container>

      <ng-container matColumnDef="logout_Time">
        <th style="width: 125px" mat-header-cell
                     *matHeaderCellDef mat-sort-header>Logout Time</th>
        <td mat-cell *matCellDef="let access"> {{access.logout_Time}} </td>
      </ng-container>

      <ng-container matColumnDef="browser">
        <th style="width: 125px" mat-header-cell
                     *matHeaderCellDef mat-sort-header>Browser</th>
        <td mat-cell *matCellDef="let access"> {{access.browser}} </td>
      </ng-container>

      <ng-container matColumnDef="ip">
        <th style="width: 125px" mat-header-cell
                     *matHeaderCellDef mat-sort-header>IP Address</th>
        <td mat-cell *matCellDef="let access"> {{access.ip}} </td>
      </ng-container>

    <!-- ROW Info-->
      <tr mat-header-row *matHeaderRowDef="columnsToDisplay" ></tr>
      <tr mat-row *matRowDef="let rowData; columns: columnsToDisplay;" ></tr>
 </table>
```

```
// TypeScript code for the Access Log Component

import { Component, OnInit, ViewChild} from '@angular/core';
import { ApiService } from './../../service/api.service';
import { Router, Data} from '@angular/router';
import {MatSort} from '@angular/material/sort';
import { MatTableDataSource, MatTable } from '@angular/material/table';

@Component({
  selector: 'app-access-log',
  templateUrl: './access-log.component.html',
  styleUrls: ['./access-log.component.css']
})
export class AccessLogComponent implements OnInit {
  @ViewChild('dataTable') dataTable: MatTable<any>;
  dataSource: MatTableDataSource<Access> ;
  @ViewChild(MatSort, {static: true}) sort: MatSort;
  columnsToDisplay = ['username', 'login_Time', 'logout_Time', 'browser', 'ip'];

  Access:any = [];

  constructor(private apiService: ApiService, private router: Router){
    this.dataSource = new MatTableDataSource<Access>();
    this.readAccess();
  }

  ngOnInit(): void {
    this.readAccess();
  }

  readAccess(){
    let dataSamples: Access[];

this.apiService.listAccess(JSON.parse(sessionStorage.userInfo).authToken).subscr
ibe((data) => {
     dataSamples = new Array(data.length);
     for(var i = 0; i < data.length; i++){
      dataSamples[i] = {
        username: data[i].username,
        login_Time : data[i].login_Time,
        logout_Time : data[i].logout_Time,
        browser : data[i].browser,
        ip : data[i].ip,
      };
     }
     this.dataSource = new MatTableDataSource<Access>(dataSamples);
    this.dataSource.sort = this.sort;
    if(this.dataSource){
      this.dataTable.renderRows();
    }
    })
  }
}

export class Access {
  username: string;
  login_Time: Data;
  logout_Time: Date;
  browser: string;
  ip: string;
}
```

```
// TypeScript code for the Access Log Specification Component

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { AccessLogComponent } from './access-log.component';

describe('AccessLogComponent', () => {
  let component: AccessLogComponent;
  let fixture: ComponentFixture<AccessLogComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ AccessLogComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(AccessLogComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
<!—Add Update Component HTML File  -->

<div class="row justify-content-center">
  <div class="col-md-4 register-employee">
    <!-- form card register -->
    <div class="card card-outline-secondary">
      <div class="card-header">
        <h3 class="mb-0">User</h3>
      </div>
      <div class="card-body">
        <form [formGroup]="editForm" (ngSubmit)="onSubmit()">

          <div class="form-group">
            <label for="inputName">Username</label>
            <input class="form-control" type="text" formControlName="username">
            <div *ngIf="submitted && myForm.username.errors?.required">
              Username is required.
            </div>
          </div>

          <div class="form-group">
             <label for="inputName">Firstname</label>
             <input class="form-control" type="text" formControlName="firstname">
              <div *ngIf="submitted && myForm.firstname.errors?.required">
                Firstname is required.
              </div>
          </div>
          <div class="form-group">
            <label for="inputName">Lastname</label>
            <input class="form-control" type="text" formControlName="lastname">
            <div  *ngIf="submitted && myForm.lastname.errors?.required">
               Lastname is required.
             </div>
          </div>

          <mat-form-field appearance="fill">
           <mat-label>Birthdate</mat-label>
           <input matInput [matDatepicker]="picker" formControlName="birthdate">
           <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
              <mat-datepicker #picker></mat-datepicker>
              <div  *ngIf="submitted && picker.errors?.required">
                  Birthdate is required.
              </div>
          </mat-form-field>

          <div class="form-group">
             <label for="inputName">Email</label>
             <input class="form-control" type="text" formControlName="email">
             <div  *ngIf="submitted && myForm.email.errors?.required">
               Email is required.
             </div>
          </div>

          <div class="form-group">
           <label>Gender</label>
           <div class="row">
            <label class="md-check">
            <input type="radio" value="Male" name="gender"
                  formControlName="gender"> Male
                </label>
                <label class="md-check">
```

```
                  <input type="radio" value="Female" name="gender"
                     formControlName="gender"> Female
                </label>
              </div>
              <div  *ngIf="submitted && myForm.gender.errors?.required">
                  Gender is required.
              </div>
          </div>

          <div class="form-group">
              <label for="inputName">Password</label>
              <input class="form-control" type="text" formControlName="password">
              <div *ngIf="submitted && myForm.password.errors?.required">
                Password is required.
              </div>
          </div>
          <div class="form-group">
            <label>Admin</label>
            <div class="row">
              <label class="md-check">
                <input type="radio" value="true" name="isAdmin"
                        formControlName="isAdmin"> Admin
                </label>
                <label class="md-check">
                  <input type="radio" value="false" name="isAdmin"
                        formControlName="isAdmin"> Not Admin
                </label>
              </div>
              <div  *ngIf="submitted && myForm.isAdmin.errors?.required">
                  Email is required.
              </div>
          </div>
          <div class="form-group">
            <button class="btn btn-success btn-lg btn-block"
                  type="submit">Submit</button>
          </div>
        </form>
      </div>
    </div><!-- form  -->
  </div>
</div>
```

```
// TypeScript code for the Add Update Component

import { Component, OnInit, NgZone } from '@angular/core';
import { Router } from '@angular/router';
import { ApiService } from './../../service/api.service';
import { FormGroup, FormBuilder, Validators } from "@angular/forms";
import { ActivatedRoute} from "@angular/router";
import { User} from './../../model/User';

@Component({
  selector: 'app-add-update',
  templateUrl: './add-update.component.html',
  styleUrls: ['./add-update.component.css']
})
export class AddUpdateComponent implements OnInit {
  userData: User[];
  submitted = false;
  editForm: FormGroup;

  constructor(
    public fb: FormBuilder,
    public router: Router,
    private actRoute: ActivatedRoute,
    public ngZone: NgZone,
    public apiService: ApiService

  ) {
    this.mainForm();
    console.log(this.router.url);
    if(this.router.url.localeCompare('/create-user') != 0){
      let username = this.actRoute.snapshot.paramMap.get('username');
      this.apiService.getUser(JSON.parse(sessionStorage.userInfo).authToken,
                                   username).subscribe(
        (data) => {
          var gen;
          var adm;
          if(data['gender'].substring(0, 1).localeCompare("M") == 0)
            gen = "Male";
          else
            gen = "Female";

          if(data['isAdmin'])
            adm = "true";
          else
            adm = "false";
          this.editForm.setValue({
            username: data['username'],
            firstname: data['firstname'],
            lastname: data['lastname'],
            birthdate: data['birthdate'],
            gender: gen,
            email: data['email'],
            password: data['password'],
            isAdmin: adm
          });
        }
      );
    }
   }
  ngOnInit(): void {
  }
  mainForm() {
    this.editForm = this.fb.group({
      username: ['', [Validators.required]],
      firstname: ['', [Validators.required]],
      lastname: ['', [Validators.required]],
      birthdate: [new Date()],
      email: ['', [Validators.required]],
      password: ['', [Validators.required]],
      gender: ['', [Validators.required]],
      isAdmin: ['', [Validators.required]],
    })
  }

  // Getter to access form control
  get myForm(){
     return this.editForm.controls;
  }
  onSubmit() {
    this.submitted = true;
    if (!this.editForm.valid) {
      return false;
    }
    else {
      var adm = false;
      if(this.editForm.value.isAdmin.localeCompare('true') == 0)
        adm = true
      var user =
      {
          "username": this.editForm.value.username,
          "firstname": this.editForm.value.firstname,
          "lastname": this.editForm.value.lastname,
          "birthdate": new Date(this.editForm.value.birthdate),
          "email": this.editForm.value.email,
          "gender": this.editForm.value.gender,
          "password": this.editForm.value.password,
          "isAdmin": adm,
      };
      if(this.router.url.includes('/edit-user')){
        this.apiService.updateUser(user,
               JSON.parse(sessionStorage.userInfo).authToken).subscribe(
          (res) => {
            console.log(res);
            this.ngZone.run(() => this.router.navigateByUrl('menu'))
          }, (error) => {
          });
      }
      else{
        this.apiService.createUser(user,
                JSON.parse(sessionStorage.userInfo).authToken).subscribe(
          (res) => {
            console.log(res);
            this.ngZone.run(() => this.router.navigateByUrl('menu'))
          }, (error) => {
          });
      }
    }
  }

}
```

```
// TypeScript code for the Add Update Component Specification

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { AddUpdateComponent } from './add-update.component';

describe('AddUpdateComponent', () => {
  let component: AddUpdateComponent;
  let fixture: ComponentFixture<AddUpdateComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ AddUpdateComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(AddUpdateComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
<!—List Component HTML File  -->

<div class="container">
    <!-- No data message -->
    <p *ngIf="User.length <= 0"
            class="no-data text-center">There is no user added yet!</p>

    <!-- User list -->
    <table class="table table-bordered" *ngIf="User.length > 0">
      <thead class="table-success">
        <tr>
          <th scope="col">Username</th>
          <th scope="col">First Name</th>
          <th scope="col">Last Name</th>
          <th scope="col">Gender</th>
          <th scope="col">Email</th>
          <th scope="col">Birthdate</th>
          <th scope="col center">Update</th>
        </tr>
      </thead>
      <tbody>
        <tr *ngFor="let user of User; let i = index">
          <th scope="row">{{user.username}}</th>
          <td>{{user.firstname}}</td>
          <td>{{user.lastname}}</td>
          <td>{{user.gender}}</td>
          <td>{{user.email}}</td>
          <td>{{user.birthdate}}</td>
          <td class="text-center edit-block">
            <span class="edit" [routerLink]="['/edit-user/', user.username]">
              <button type="button"
                  class="btn btn-success btn-sm">Edit</button>
            </span>
            <span class="delete" (click)="removeUser(user, i)">
              <button type="button"
                  class="btn btn-danger btn-sm">Delete</button>
            </span>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
```

```
// TypeScript code for the List Component

import { Component, OnInit } from '@angular/core';
import { ApiService } from './../../service/api.service';


@Component({
  selector: 'app-list',
  templateUrl: './list.component.html',
  styleUrls: ['./list.component.css']
})
export class ListComponent implements OnInit {
  User:any = [];

  constructor(private apiService: ApiService) {
    this.readUser();
  }

  ngOnInit(): void {
  }

  readUser(){
    this.apiService.getUsers(JSON.parse(sessionStorage.userInfo).authToken).
         subscribe((data) => {
     console.log(data)
      this.User = data;
    })
  }


  removeUser(user, index) {
    if(window.confirm('Are you sure?')) {
       this.apiService.deleteUser(user.username,
         (JSON.parse(sessionStorage.userInfo).authToken)).subscribe((data) => {
       this.User.splice(index, 1);
        }
      )
    }
  }
}
```

```
// TypeScript code for the List Component Specification

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { ListComponent } from './list.component';

describe('ListComponent', () => {
  let component: ListComponent;
  let fixture: ComponentFixture<ListComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ ListComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(ListComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
// TypeScript code for the Menu Component

import { Component, OnInit, NgZone } from '@angular/core';
import { Router } from '@angular/router';
import { ApiService } from './../../service/api.service';

@Component({
  selector: 'app-menu',
  templateUrl: './menu.component.html',
  styleUrls: ['./menu.component.css']
})
export class MenuComponent implements OnInit {
  constructor() { }

  ngOnInit(): void {
  }

}

// TypeScript code for the Menu Component Specification

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { MenuComponent } from './menu.component';

describe('MenuComponent', () => {
  let component: MenuComponent;
  let fixture: ComponentFixture<MenuComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ MenuComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(MenuComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
<!—Read Message Component HTML File  -->

<table #dataTable mat-table [dataSource]="dataSource" matSort>
  <!-- COLUMN INFO -->
   <!-- Sender Username Col -->
    <ng-container matColumnDef="senderUsername">
      <th style="width: 135px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Sender Username</th>
      <td  mat-cell *matCellDef="let message"> {{message.senderUsername}} </td>
      </ng-container>

      <ng-container matColumnDef="senderFirstName">
        <th style="width: 125px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Sender Firstname</th>
      <td  mat-cell *matCellDef="let message"> {{message.senderFirstName}} </td>
      </ng-container>

      <ng-container matColumnDef="senderLastName">
        <th style="width: 125px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Sender Lastname</th>
      <td mat-cell *matCellDef="let message"> {{message.senderLastName}} </td>
      </ng-container>

      <ng-container matColumnDef="receiverUsername">
      <th style="width: 125px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Receiver Username</th>
      <td mat-cell *matCellDef="let message"> {{message.receiverUsername}} </td>
      </ng-container>

      <ng-container matColumnDef="receiverFirstName">
      <th style="width: 125px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Receiver Firstname</th>
     <td mat-cell *matCellDef="let message"> {{message.receiverFirstName}} </td>
     </ng-container>

     <ng-container matColumnDef="receiverLastName">
     <th style="width: 125px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Receiver Lastname</th>
     <td mat-cell *matCellDef="let message"> {{message.receiverLastName}} </td>
     </ng-container>

     <ng-container matColumnDef="time">
     <th style="width: 190px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Time</th>
     <td mat-cell *matCellDef="let message"> {{message.time}} </td>
     </ng-container>

     <ng-container matColumnDef="title">
       <th style="width: 125px" mat-header-cell
               *matHeaderCellDef mat-sort-header>Title</th>
     <td mat-cell *matCellDef="let message"> {{message.title}} </td>
     </ng-container>
      <ng-container matColumnDef="message">
        <th style="width: 125px" mat-header-cell
                  *matHeaderCellDef mat-sort-header>Message</th>
        <td mat-cell *matCellDef="let message"> {{message.message}} </td>
      </ng-container>

    <!-- ROW Info-->
      <tr mat-header-row *matHeaderRowDef="columnsToDisplay" ></tr>
      <tr mat-row *matRowDef="let rowData; columns: columnsToDisplay;" ></tr>
 </table>
```

```
// TypeScript code for the Read Message Component

import { ApiService } from './../../service/api.service';
import { Router} from '@angular/router';
import { Message } from './../../model/message';
import { MatTableDataSource, MatTable } from '@angular/material/table';
import { Component, ViewChild, OnInit } from '@angular/core';
import {MatSort} from '@angular/material/sort';

@Component({
  selector: 'app-read-message',
  templateUrl: './read-message.component.html',
  styleUrls: ['./read-message.component.css']
})
export class ReadMessageComponent implements OnInit {
  @ViewChild('dataTable') dataTable: MatTable<any>;
  dataSource: MatTableDataSource<Message> ;
  @ViewChild(MatSort, {static: true}) sort: MatSort;

  columnsToDisplay = ['senderUsername', 'senderFirstName', 'senderLastName',
                       'receiverUsername', 'receiverFirstName',
                       'receiverLastName', 'time', 'title', 'message'];

  constructor(private apiService: ApiService, private router: Router) {
    this.dataSource = new MatTableDataSource<Message>();

    if(this.router.url.localeCompare('/inbox') == 0)
      this.readInbox();
    else
      this.readOutbox();

  }

  ngOnInit(): void {
   if(this.router.url.localeCompare('/inbox') == 0)
   this.readInbox();
  else
   this.readOutbox();
  }

  readInbox(){
   let dataSamples: Message[];
   this.apiService.readInbox(JSON.parse(sessionStorage.userInfo).authToken,
   JSON.parse(sessionStorage.userInfo).username).subscribe((data) => {
   dataSamples = new Array(data.length);
    for(var i = 0; i < data.length; i++){
     dataSamples[i] = {
       senderUsername: data[i].senderUsername,
       senderFirstName : data[i].senderFirstName,
       senderLastName : data[i].senderLastName,
       receiverUsername : data[i].receiverUsername,
       receiverFirstName : data[i].receiverFirstName,
       receiverLastName : data[i].receiverLastName,
       time : data[i].time,
       title : data[i].title,
       message : data[i].message,
     };
    }
    this.dataSource = new MatTableDataSource<Message>(dataSamples);
    this.dataSource.sort = this.sort;

    if(this.dataSource){
      this.dataTable.renderRows();
    }
    })

  }

  readOutbox(){
    let dataSamples: Message[];
    console.log("outbox");
    this.apiService.readOutbox(JSON.parse(sessionStorage.userInfo).authToken,
    JSON.parse(sessionStorage.userInfo).username).subscribe((data) => {
    console.log(data.length);
    dataSamples = new Array(data.length);
     for(var i = 0; i < data.length; i++){
      dataSamples[i] = {
        senderUsername: data[i].senderUsername,
        senderFirstName : data[i].senderFirstName,
        senderLastName : data[i].senderLastName,
        receiverUsername : data[i].receiverUsername,
        receiverFirstName : data[i].receiverFirstName,
        receiverLastName : data[i].receiverLastName,
        time : data[i].time,
        title : data[i].title,
        message : data[i].message,
      };
     }

    this.dataSource = new MatTableDataSource<Message>(dataSamples);
    this.dataSource.sort = this.sort;
    if(this.dataSource){
      this.dataTable.renderRows();
    }
   })
  }
}
```

```
// TypeScript code for the Read Message Component Specification

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { ReadMessageComponent } from './read-message.component';

describe('ReadMessageComponent', () => {
  let component: ReadMessageComponent;
  let fixture: ComponentFixture<ReadMessageComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ ReadMessageComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(ReadMessageComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
<div class="row justify-content-center">
    <div class="col-md-4 sendMessage">
      <!-- form card register -->
      <div class="card-body">
        <form [formGroup]="sendMessageForm" (ngSubmit)="onSubmit()">
          <div class="form-group">
            <label for="inputName">Receiver </label>
            <input class="form-control" type="text"
                   formControlName="receiver">
            <!-- error -->
            <div  *ngIf="submitted && myForm.receiver.errors?.required">
              Receiver is required.
            </div>
          </div>

          <div class="form-group">
            <label for="inputName">Title </label>
            <input class="form-control" type="text" formControlName="title">
            <!-- error -->
            <div  *ngIf="submitted && myForm.title.errors?.required">
              Title is required.
            </div>
          </div>

          <div class="form-group">
            <label for="inputName">Message </label>
            <input class="form-control" type="text" formControlName="message">
            <!-- error -->
            <div  *ngIf="submitted && myForm.message.errors?.required">
              Message is required.
            </div>
          </div>

          <div class="form-group">
            <button class="btn btn-success btn-lg btn-block"
                type="submit">Send</button>
          </div>
        </form>

      </div>
    </div><!-- form card register -->
  </div>
```

```
// TypeScript code for the Remove Component

import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-remove',
  templateUrl: './remove.component.html',
  styleUrls: ['./remove.component.css']
})
export class RemoveComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}

// TypeScript code for the Remove Component Specification

import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { RemoveComponent } from './remove.component';

describe('RemoveComponent', () => {
  let component: RemoveComponent;
  let fixture: ComponentFixture<RemoveComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ RemoveComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(RemoveComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
<!— Send Message Component HTML File -->

<div class="row justify-content-center">
    <div class="col-md-4 sendMessage">
      <!-- form card register -->
      <div class="card-body">
        <form [formGroup]="sendMessageForm" (ngSubmit)="onSubmit()">
          <div class="form-group">
            <label for="inputName">Receiver </label>
            <input class="form-control" type="text" formControlName="receiver">
            <!-- error -->
            <div  *ngIf="submitted && myForm.receiver.errors?.required">
              Receiver is required.
            </div>
          </div>

          <div class="form-group">
            <label for="inputName">Title </label>
            <input class="form-control" type="text" formControlName="title">
            <!-- error -->
            <div  *ngIf="submitted && myForm.title.errors?.required">
              Title is required.
            </div>
          </div>

          <div class="form-group">
            <label for="inputName">Message </label>
            <input class="form-control" type="text" formControlName="message">
            <!-- error -->
            <div  *ngIf="submitted && myForm.message.errors?.required">
              Message is required.
            </div>
          </div>

          <div class="form-group">
            <button class="btn btn-success btn-lg btn-block"
                               type="submit">Send</button>
          </div>
        </form>

      </div>
    </div><!-- form card register -->
  </div>
```

```
// TypeScript code for the Send Message Component

import { Component, OnInit, NgZone } from '@angular/core';
import { Router } from '@angular/router';
import { ApiService } from './../../service/api.service';
import { FormGroup, FormBuilder, Validators } from "@angular/forms";


@Component({
  selector: 'app-send-message',
  templateUrl: './send-message.component.html',
  styleUrls: ['./send-message.component.css']
})

export class SendMessageComponent implements OnInit {

  submitted = false;
  sendMessageForm: FormGroup;

  constructor(
    public fb: FormBuilder,
    public router: Router,
    public ngZone: NgZone,
    public apiService: ApiService
  ) {
    this.mainForm();
  }


  mainForm() {
    this.sendMessageForm = this.fb.group({
      receiver: ['', [Validators.required]],
      title: ['', [Validators.required]],
      message: ['', [Validators.required]]
    })
  }



  ngOnInit(): void {
  }

    // Getter to access form control
  get myForm(){
    return this.sendMessageForm.controls;
  }


  onSubmit() {
    this.submitted = true;
    if (!this.sendMessageForm.valid) {
      return false;
    } else {
      var mes =
      {
          "senderUsername": JSON.parse(sessionStorage.userInfo).username,
          "receiverUsername": this.sendMessageForm.value.receiver,
          "title": this.sendMessageForm.value.title,
          "message": this.sendMessageForm.value.message
      };
      console.log(JSON.parse(sessionStorage.userInfo).authToken);
      this.apiService.sendMessage(mes,
            JSON.parse(sessionStorage.userInfo).authToken).subscribe(
        (res) => {
          console.log(res);
          this.ngZone.run(() => this.router.navigateByUrl('menu'))
        }, (error) => {
          console.log("dssads" + error);
        });
    }
  }

}

// TypeScript code for the Send Message Component Specification


import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { SendMessageComponent } from './send-message.component';

describe('SendMessageComponent', () => {
  let component: SendMessageComponent;
  let fixture: ComponentFixture<SendMessageComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ SendMessageComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(SendMessageComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Self-Checklist for Your Report

Please check the items here before submitting your report. This signed checklist should be the final page of your report.

- ✓ Did you provide detailed information about the work you did?
- ✓ Is supervisor information included?
- ✓ Did you use the Report Template to prepare your report, so that it has a cover page, the 8 major sections and 13 subsections specified in the Table of Contents, and uses the required section names?
- ✓ Did you follow the style guidelines?
- ✓ Does you report look professionally written?
- ✓ Does your report include all necessary References, and proper citations to them in the body?
- ✓ Did you remove all explanations from the Report Template, which are marked with yellow color? Did you modify all text marked with green according to your case?

Signature: MUSTAFA GÖKTAN GÜDÜKBAY

[illegible]