

INTERNET TECHNOLOGIES ASSIGNMENT REPORT

Name: ABHISHEK DE

Roll no: 001710501027

Group: A1

Assignment: 2

PROBLEM STATEMENT

Write a multi-client chat application consisting of both client and server programs. In this chat application simultaneously several clients can communicate with each other. For this you need a single server program that clients connect to. The client programs send the chat text or image (input) to the server and then the server distributes that message (text or image) to all the other clients. Each client then displays the message sent to it by the server. The server should be able to handle several clients concurrently. It should work fine as clients come and go.

Develop the application using a framework based on Node.JS.

How are messages handled concurrently?

Which web application framework(s) did you follow?

Prepare a detailed report of the experiments you have done, and your observations on the protocol layers thus created.

SOLUTION APPROACH

The server side of the application has been designed using **NodeJS** and the client side using **ReactJS**. To send messages from client to server **socket.io** has been used. Messages are of two types:

Broadcast messages: These messages are received from a single client and the server transmits the message to all available clients.

Private messages: These messages are received from the client as a <message, receiver> pair. The server transmits the message only to the socket associated with the receiver. The receiver name and socketID are mapped together using a dictionary.

To keep the messages persistent, so that they can be retrieved later, a MongoDB database has been used. The database stores the following data:

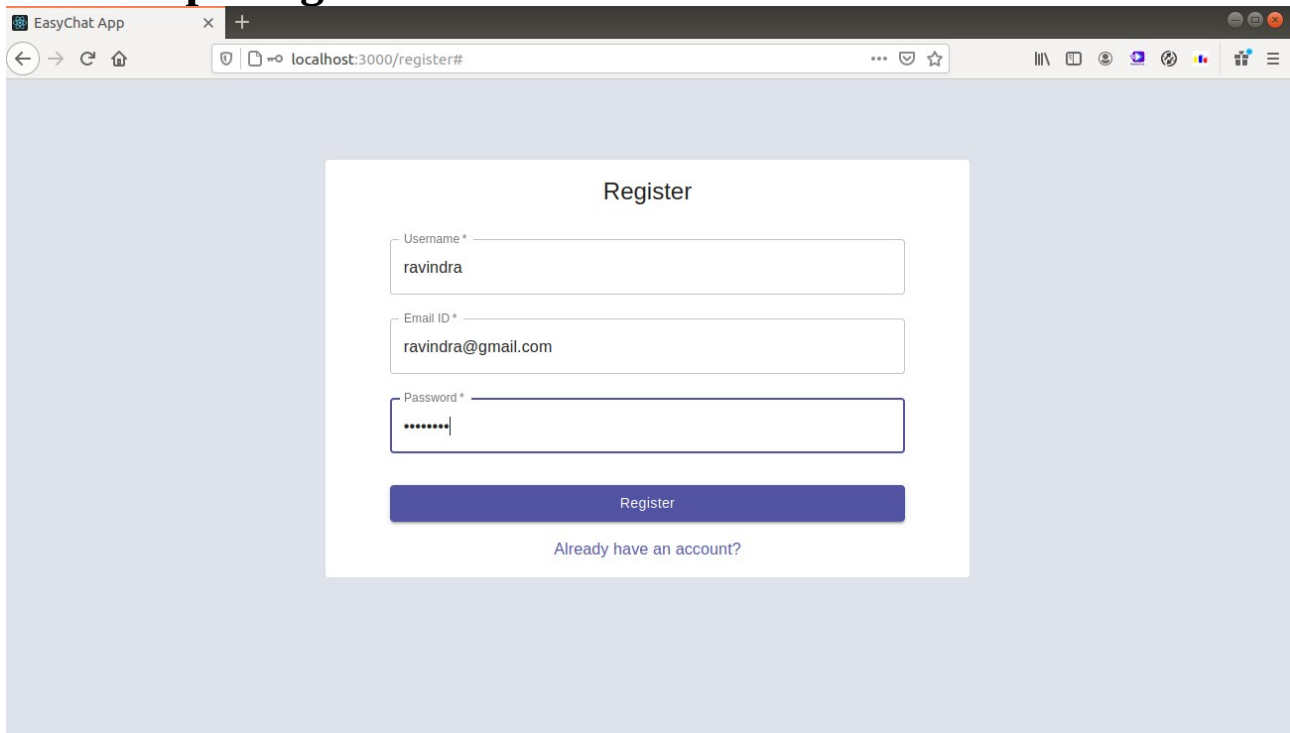
- **Users:** Information of all users who have registered in the application.
- **Messages:** Sender, receiver, and message content of each message.
- **BroadcastMessages:** Sender and message content of broadcast messages.

SALIENT FEATURES UNIQUE TO MY APPLICATION

- Several clients can be handled concurrently.
- Support both broadcast and unicast messages.
- Messages are persistent and can be fetched later.
- Supports user login and registration.
- If user A messages user B for the first time, A gets added to B's contacts list and vice versa.
- A user can search for all existing users using a search bar.

SCREENSHOTS

1. User opening an account

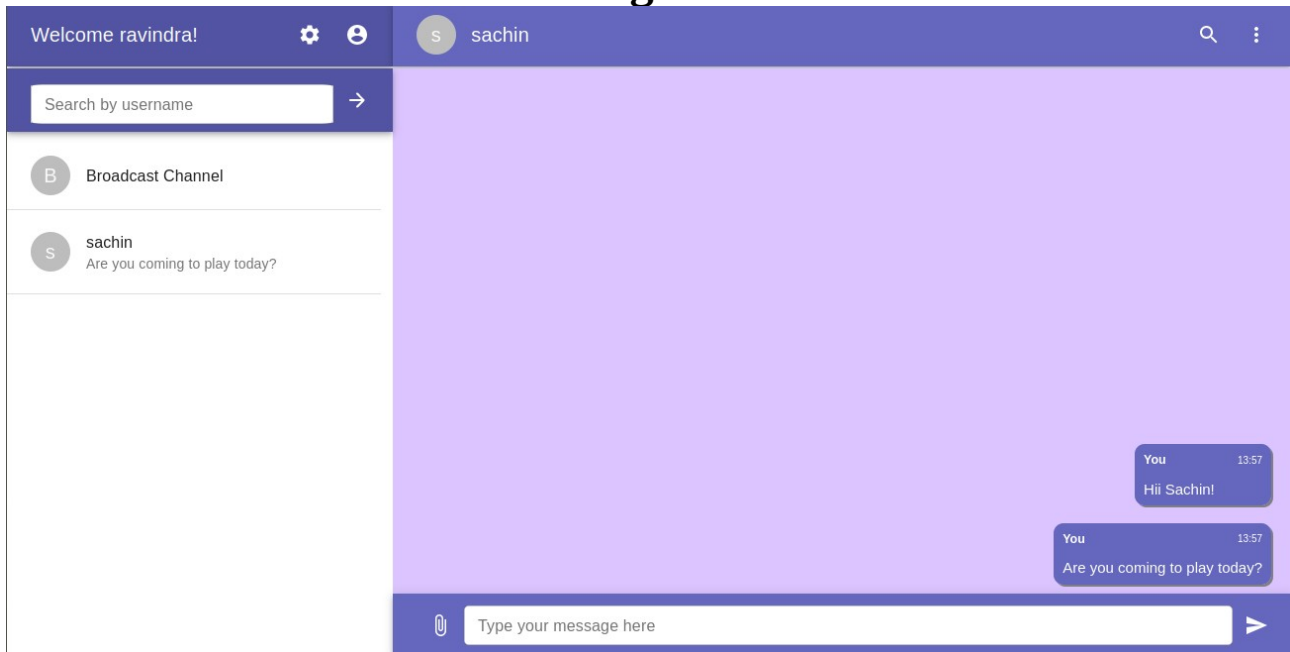


The screenshot shows a web browser window with the title 'EasyChat App'. The address bar displays 'localhost:3000/register#'. The main content area features a 'Register' form with the following fields:

- Username ***: A text input field containing the text 'ravindra'.
- Email ID ***: A text input field containing the text 'ravindra@gmail.com'.
- Password ***: A password input field with masked characters '*****'.

Below the input fields is a blue 'Register' button. Underneath the button is a link that says 'Already have an account?'.

2. User 'ravindra' sends message to user 'sachin'



The screenshot shows the chat interface of the EasyChat App. The top header bar is dark blue and contains the text 'Welcome ravindra!' on the left, a settings gear icon and a user profile icon in the center, and a search icon and a menu icon on the right. Below the header, on the left side, is a search bar with the placeholder text 'Search by username' and a right arrow. Below the search bar is a list of chat items:

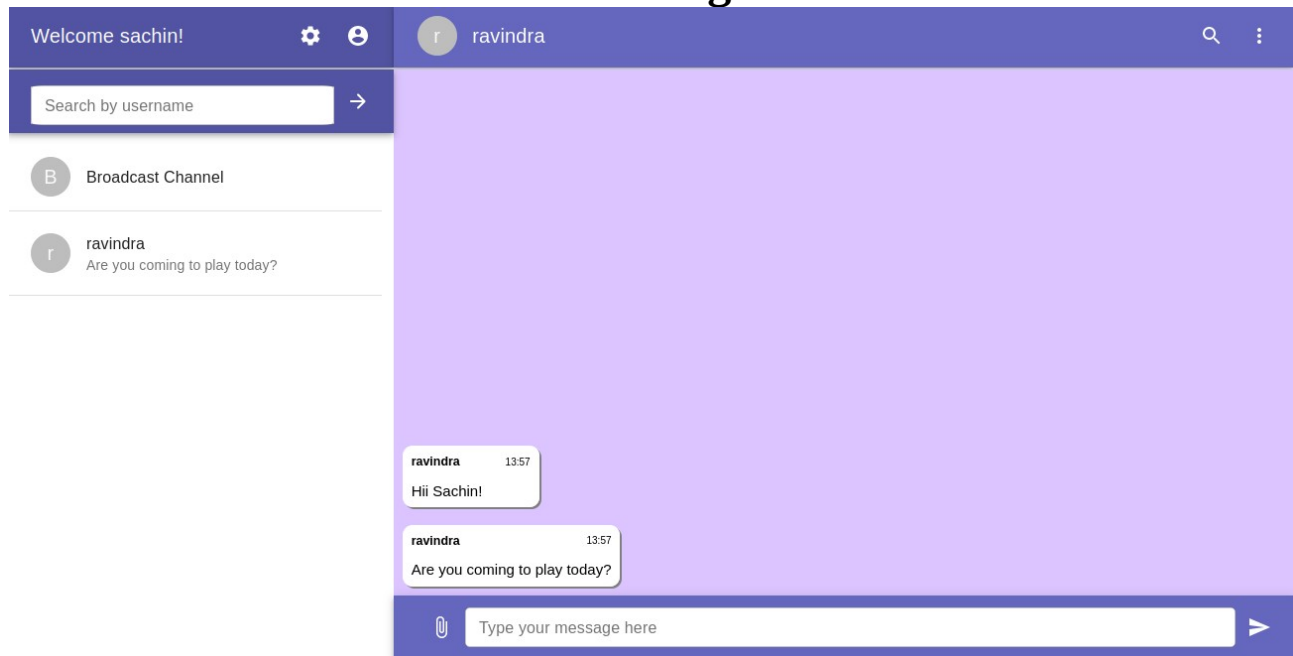
- A 'Broadcast Channel' item with a circular icon containing the letter 'B'.
- A chat item for 'sachin' with a circular icon containing the letter 'S' and the text 'Are you coming to play today?' below it.

The main chat area on the right has a light purple background. It displays two outgoing messages from 'You' (the user 'ravindra') to 'sachin':

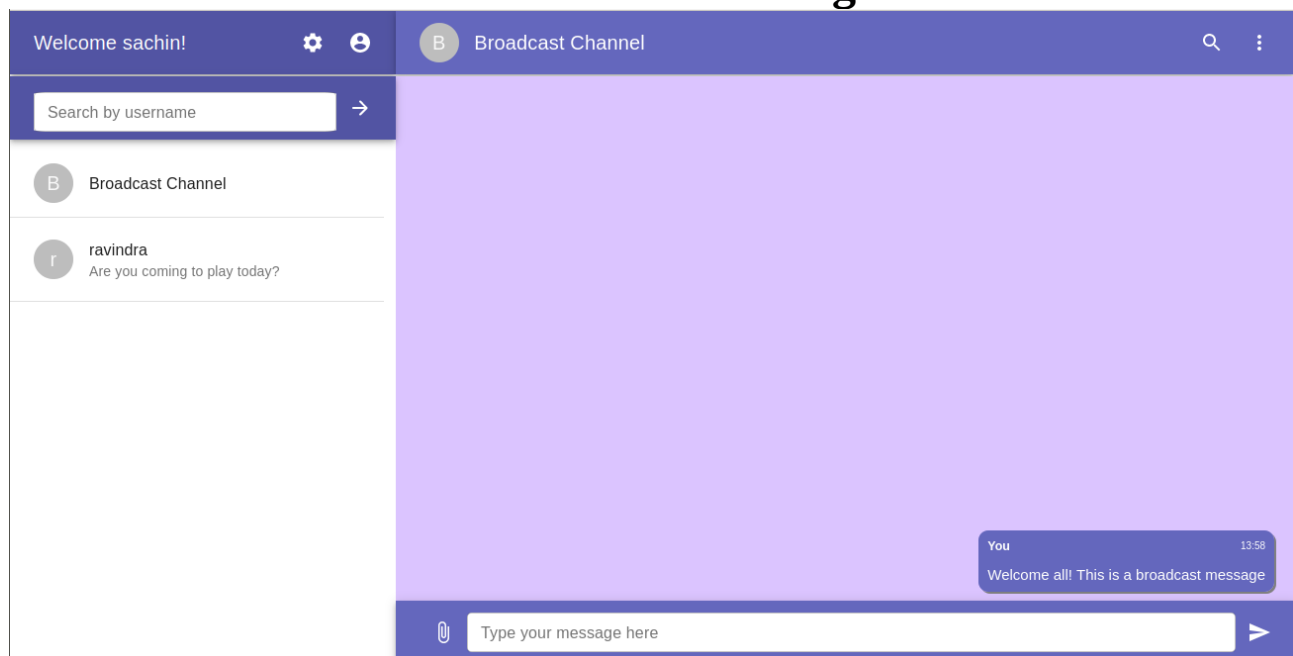
- The first message says 'Hii Sachin!' and is timestamped '13:57'.
- The second message says 'Are you coming to play today?' and is also timestamped '13:57'.

At the bottom of the chat area is a dark blue input bar with a paperclip icon on the left, a text input field with the placeholder 'Type your message here', and a right arrow on the right.

3. User 'sachin' receives the message



4. User 'sachin' sends a broadcast message



5. Other users receiving the broadcast message

