

# ENHANCED CHAT APPLICATION WITH RICH UI

**R Sai Suryanarayana Sastry, Nimmala Harsha Vardhan**

**Mr. S. Maruthi (Guide)**

**Sreenidhi Institute of Science and Technology**

**Hyderabad**

---

## Abstract

The primary goal of a web-based chat programme is to provide a chat programme that enables users to talk with one another via a server linked. A server and numerous clients are supported by this web-based chat programme. Prior to connecting clients, the server needs to be started. Clients can communicate with each other in both directions using this application. The application's user interface would be distinct from that of the current chat programmes, which is what makes this project unique. In this modern era, many applications have been used to communicate long distances, both for telephone and chat. UI design in applications affects users, as most of the younger generation prefers an attractive, innovative, and many-choice design, while most of the older generation prefers a simple design appearance.

## Introduction

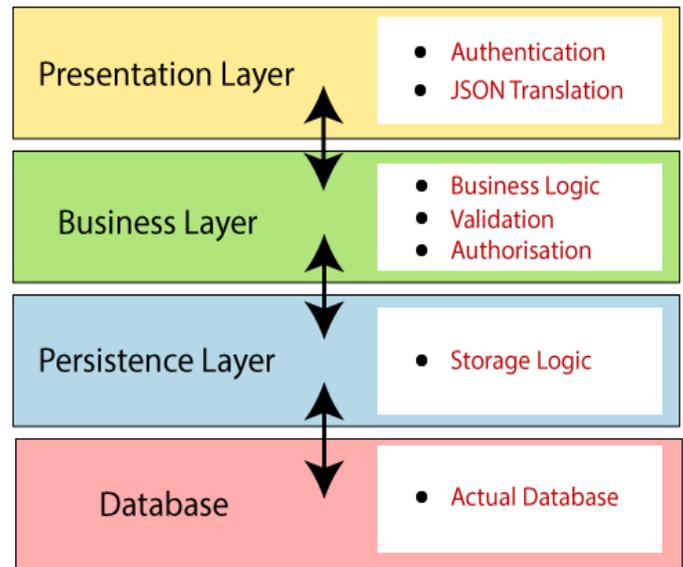
Currently, all chat applications have the same general user interface. In contrast to this idea, we have tried to build a new outlook for our chat programme in user interface. Users can send private messages as well as group messages. While online, individuals can instantly receive messages. Any machine running the JVM can host the server-side application, which uses a web socket to communicate with the client side. The programme is created to function as a web application. Anyone can use it as the foundation for offering instant messaging features because it provides a general architecture for chat apps. The software's primary goal is to give users access to an instant messaging tool that, when necessary, can easily handle multiple users at once. The distribution of information promptly is one of the project's top priorities. Choosing the appropriate communication technology has thus become the project's main priority. Through the internet, people can communicate with one another.

## Literature Survey

1. “The influence of design aesthetics in usability testing: Effects on user performance and perceived usability”.
2. “A Study on UI Design of Social Networking Service Messenger by Using Case Analysis Model”.
3. “The Influence of UI UX Design to Number of Users Between ‘Line’ and ‘WhatsApp’”

The above papers signify the importance of a user interface in a web or mobile application. These papers explain about various aspects which we need to keep in mind while developing a user interface for a web or mobile chat application. They have also discussed the case studies between the existing chat applications and what makes one better than the other. They have also discussed the parameters which makes the user interface more usable. One of the above papers discussed the user behaviour towards the mobile application based on certain age group (14-17 years) stating that the more attractive user interface will catch an eye of the user in that particular age group.

## Methodology



## Spring Architecture

1. **PresentationLayer:** The presentation layer handles the HTTP requests, translates the JSON parameter to object, and authenticates the request and transfer it to the business layer. In short, it consists of **views** i.e., frontend part.
2. **Business Layer:** The business layer handles all the **business logic**. It consists of service classes and uses services provided by data access layers. It also performs **authorization** and **validation**.
3. **Persistence Layer:** The persistence layer contains all the **storage logic** and translates business objects from and to database rows.
4. **Database Layer:** In the database layer, **CRUD** (create, retrieve, update, delete) operations are performed. Now we

have validator classes, view classes, and utility classes.

## Web Sockets

WebSocket is bidirectional, a full-duplex protocol that is used in the same scenario of client-server communication, unlike HTTP it starts from `ws://` or `wss://`. It is a stateful protocol, which means the connection between client and server will keep alive until it is terminated by either party (client or server). After closing the connection by either of the client and server, the connection is terminated from both ends.

## STOMP Protocol



1. STOMP is the Simple Text Orientated Messaging Protocol.
2. STOMP provides an interoperable wire format so that STOMP clients can communicate with any STOMP message broker to provide easy and widespread messaging interoperability among many languages, platforms, and brokers.
3. STOMP is a very simple and easy to implement protocol, coming from the HTTP school of design; the server side may be hard to implement well, but it is very easy to write a client to get yourself connected.

4. Many developers have told us that they have managed to write a STOMP client in a couple of *hours* to in their language, runtime, or platform into the STOMP network. So, if your favored language/runtime of choice does not offer a good enough STOMP client don't be afraid to write one.
5. The protocol is broadly like HTTP, and works over TCP using the commands like CONNECT, SEND, BEGIN and ABORT etc.

## Sock JS

A JavaScript package called SockJS makes full-duplex communication possible. It enables low latency communication between the client and server and is cross-browser compatible. A WebSocket-like object is made available by the JavaScript library for browsers called SockJS. You may construct a low latency, full duplex, cross-domain communication channel between the browser and the web server using SockJS, which provides you with a consistent, cross-browser JavaScript API. Internally, SockJS supports a variety of transport protocols. Depending on the type of transportation being used, you can encounter restrictions.

## Technologies Used ReactJS

1. ReactJS tutorial provides basic and advanced concepts of ReactJS. Currently, ReactJS is one of the most popular

- JavaScript front-end libraries which has a strong foundation and a large community.
2. The React. is framework being an open-source JavaScript framework and library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript.
  3. ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front-end library which is responsible only for the view layer of the application.

### Spring Boot

1. Spring Boot is an open-source Java-based framework used to create a micro-Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications.
2. Spring Boot is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring framework.
3. Spring Boot is one of the best backend frameworks that developers count on while performing backend web development tasks.

4. It is a Spring-based framework that allows developers to write production-grade backend web applications in Java.

### MySQL

1. MySQL is an open-source relational database management system.
2. MySQL is currently the most popular database management system software used for managing the relational database.
3. It is open-source database software, which is supported by Oracle Company.

### Inversion of control container (dependency injection)

Central to the Spring Framework is its inversion of control (IoC) container, which provides a consistent means of configuring and managing Java objects using reflection. The container is responsible for managing object lifecycles of specific objects: creating these objects, calling their initialization methods, and configuring these objects by wiring them together.

Objects created by the container are also called managed objects or beans. The container can be configured by loading XML (Extensible Markup Language) files or detecting specific Java annotations on configuration classes. These data sources contain the bean definitions that provide the information required to create the beans.

Objects can be obtained by means of either dependency lookup or dependency injection. Dependency lookup is a pattern where a caller asks the container object for an object with a specific name or of a specific type. Dependency injection is a pattern where the container passes objects by name to other objects, via either constructors, properties, or factory methods.

In many cases one need not use the container when using other parts of the Spring Framework, although using it will likely make an application easier to configure and customize. The Spring container provides a consistent mechanism to configure applications and integrates with almost all Java environments, from small-scale applications to large enterprise applications.

The container can be turned into a partially compliant EJB (Enterprise JavaBeans) 3.0 container by means of the Pitchfork project. Some criticize the Spring Framework for not complying with standards. However, Spring Source doesn't see EJB 3 compliance as a major goal, and claims that the Spring Framework and the container allow for more powerful programming models. The programmer does not directly create an object, but describes how it should be created, by defining it in the Spring configuration file. Similarly, services and components are not called directly; instead, a Spring configuration file defines which services and components must be called. This IoC is

intended to increase the ease of maintenance and testing.

### **Aspect-oriented programming framework**

The Spring Framework has its own Aspect-oriented programming (AOP) framework that modularizes cross-cutting concerns in aspects. The motivation for creating a separate AOP framework comes from the belief that it should be possible to provide basic AOP features without too much complexity in either design, implementation, or configuration. The Spring AOP framework also takes full advantage of the Spring container.

The Spring AOP framework is proxy pattern-based, and is configured at run time. This removes the need for a compilation step or load-time weaving. On the other hand, interception only allows for public method-execution on existing objects at a join point.

Compared to the AspectJ framework, Spring AOP is less powerful, but also less complicated. Spring 1.2 includes support to configure AspectJ aspects in the container. Spring 2.0 added more integration with AspectJ; for example, the pointcut language is reused and can be mixed with Spring AOP-based aspects. Further, Spring 2.0 added a Spring Aspects library that uses AspectJ to offer common Spring features such as declarative transaction management and dependency injection via AspectJ compile-time or load-time weaving.

SpringSource also uses AspectJ AOP in other Spring projects such as Spring Roo and Spring Insight, with Spring Security also offering an AspectJ-based aspect library.

Spring AOP has been designed to make it able to work with cross-cutting concerns inside the Spring Framework. Any object which is created and configured by the container can be enriched using Spring AOP.

The Spring Framework uses Spring AOP internally for transaction management, security, remote access, and JMX.

Since version 2.0 of the framework, Spring provides two approaches to the AOP configuration:

schema-based approach and

@AspectJ-based annotation style.

### **Model–view–controller framework**

The Spring Framework features its own model–view–controller (MVC) web application framework, which was not originally planned. The Spring developers decided to write their own Web framework as a reaction to what they perceived as the poor design of the (then) popular Jakarta Struts Web framework, as well as deficiencies in other available frameworks. In particular, they felt there was insufficient separation between the

presentation and request handling layers, and between the request handling layer and the model.

Like Struts, Spring MVC is a request-based framework. The framework defines strategy interfaces for all of the responsibilities that must be handled by a modern request-based framework. The goal of each interface is to be simple and clear so that it's easy for Spring MVC users to write their own implementations, if they so choose. MVC paves the way for cleaner front end code. All interfaces are tightly coupled to the Servlet API. This tight coupling to the Servlet API is seen by some as a failure on the part of the Spring developers to offer a high-level abstraction for Web-based applications[citation needed]. However, this coupling makes sure that the features of the Servlet API remain available to developers while also offering a high abstraction framework to ease working with it.

The DispatcherServlet class is the front controller of the framework and is responsible for delegating control to the various interfaces during the execution phases of an HTTP request.

The most important interfaces defined by Spring MVC, and their responsibilities, are listed below:

Controller: comes between Model and View to manage incoming requests and redirect to proper response. Controller will map the http request to corresponding methods. It acts as a gate that

directs the incoming information. It switches between going into model or view.

**HandlerAdapter:** execution of objects that handle incoming requests

**HandlerInterceptor:** interception of incoming requests comparable, but not equal to Servlet filters (use is optional and not controlled by DispatcherServlet).

**HandlerMapping:** selecting objects that handle incoming requests (handlers) based on any attribute or condition internal or external to those requests

**LocaleResolver:** resolving and optionally saving of the locale of an individual user

**MultipartResolver:** facilitate working with file uploads by wrapping incoming requests

**View:** responsible for returning a response to the client. Some requests may go straight to view without going to the model part; others may go through all three.

**ViewResolver:** selecting a View based on a logical name for the view (use is not strictly required)

Each strategy interface above has an important responsibility in the overall framework. The abstractions offered by these interfaces are powerful, so to allow for a set of variations in their implementations, Spring MVC ships with implementations of all these interfaces and together offers a feature set on top of the Servlet API. However, developers and vendors are free to write other implementations. Spring MVC uses the Java `java.util.Map` interface as a data-oriented

abstraction for the Model where keys are expected to be string values.

The ease of testing the implementations of these interfaces seems one important advantage of the high level of abstraction offered by Spring MVC. DispatcherServlet is tightly coupled to the Spring inversion of control container for configuring the web layers of applications. However, web applications can use other parts of the Spring Framework—including the container—and choose not to use Spring MVC.

### **Lazy loading**

Lazy loading is not a new concept. It has been available for quite some time. In essence, lazy loading means that a component or a part of code must get loaded when it is required. It is also referred to as code splitting and data fetching.

Talking about React specifically, it bundles the complete code and deploys all of it at the same time. Now, usually, that's not a bad idea, since React SPAs (Single page application) are quite small and do not affect the performance. But what if we have a gigantic application, like a content management system with a customer portal, admin portal etc. In such a case, it does not seem like a smart idea to load the complete application.

- It will be a huge application and will cost a lot of unnecessary data transfer which can lead to slow loading of the website.

- A customer login, will not have access to admin specific features, so loading it is a waste of memory and time.

In this post, I will try to explain the advantages of lazy loading and how to implement it in React.

### React Component Life Cycle

Each component in react has a life cycle which you can monitor and manipulate during the main three phases. The three phases are: Mounting, Updating, and Unmounting.

#### Mounting:

Mounting means putting react elements into the HTML DOM.

React has some built in methods which are called, in a particular order to achieve mounting

Constructor(), `getDerievedStateFromProps()`,  
`render()`,  
`compoentDidmount()`

constructor is a method which is called first before anything else, when a component is initiated, and it is the natural place for us to set up the initial state and the other initial values. The constructor method is called with the props and arguments and we should always start with calling the super constructor `super(props)` before anything else, this will initiate the parent's constructor method and allows the component to inherit from its parent.

#### `getDerivedStateFromProps`

The `getDerivedStateFromProps()` method is called right before rendering the element(s) in the DOM.

This is the natural place to set the state object based on the initial props.

It takes state as an argument, and returns an object with changes to the state.

#### Render

The `render()` method is required, and is the method that actually outputs the HTML to the DOM.

#### `componentDidMount`

The `componentDidMount()` method is called after the component is rendered.

This is where you run statements that requires that the component is already placed in the DOM.

#### Updating:

The next phase in the lifecycle is when a component is updated.

A component is updated whenever there is a change in the component's state or props.

React has five built-in methods that gets called, in this order, when a component is updated:

`getDerivedStateFromProps()`  
`shouldComponentUpdate()`  
`render()`  
`getSnapshotBeforeUpdate()`  
`componentDidUpdate()`

The render() method is required and will always be called, the others are optional and will be called if you define them

### getDerivedStateFromProps

Also at updates the getDerivedStateFromProps method is called. This is the first method that is called when a component gets updated.

This is still the natural place to set the state object based on the initial props.

### shouldComponentUpdate

In the shouldComponentUpdate() method you can return a Boolean value that specifies whether React should continue with the rendering or not.

### Render

The render() method is of course called when a component gets *updated*, it has to re-render the HTML to the DOM, with the new changes.

### getSnapshotBeforeUpdate

In the getSnapshotBeforeUpdate() method you have access to the props and state before the update, meaning that even after the update, you can check what the values were before the update.

If the getSnapshotBeforeUpdate() method is present, you should also include the componentDidUpdate() method, otherwise you will get an error.

### componentDidUpdate

The componentDidUpdate method is called after the component is updated in the DOM.

### Unmounting

The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.

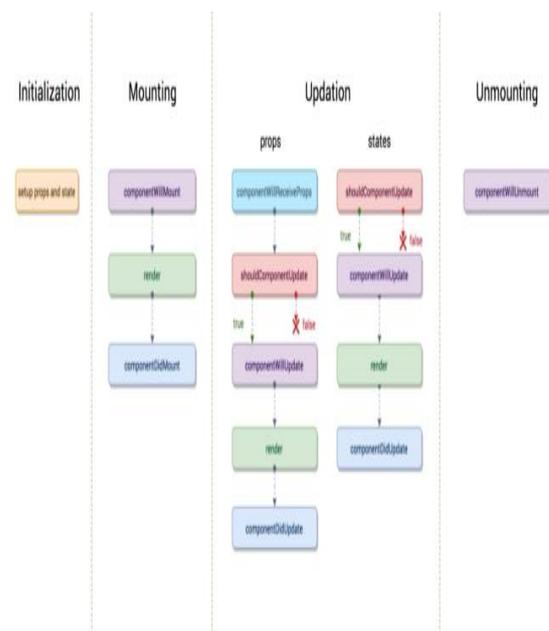
React has only one built-in method that gets called when a component is unmounted:

- componentWillUnmount()

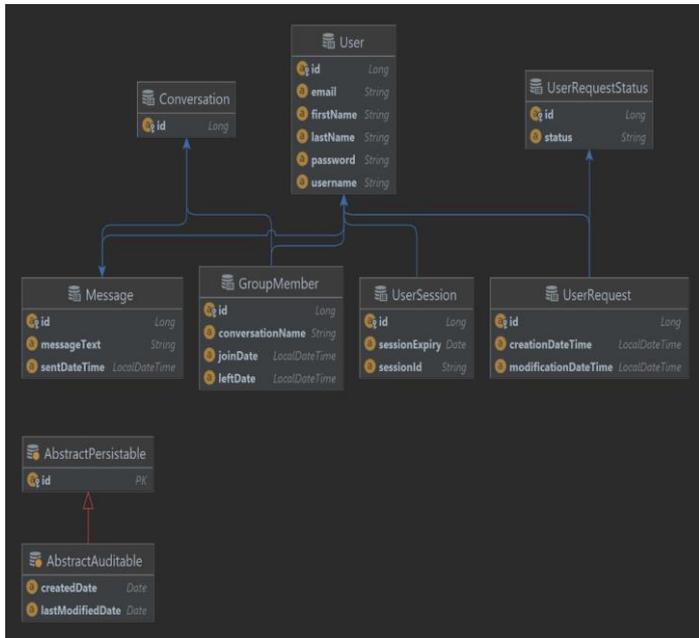
### componentWillUnmount

The componentWillUnmount method is called when the component is about to be removed from the DOM.

### System Design

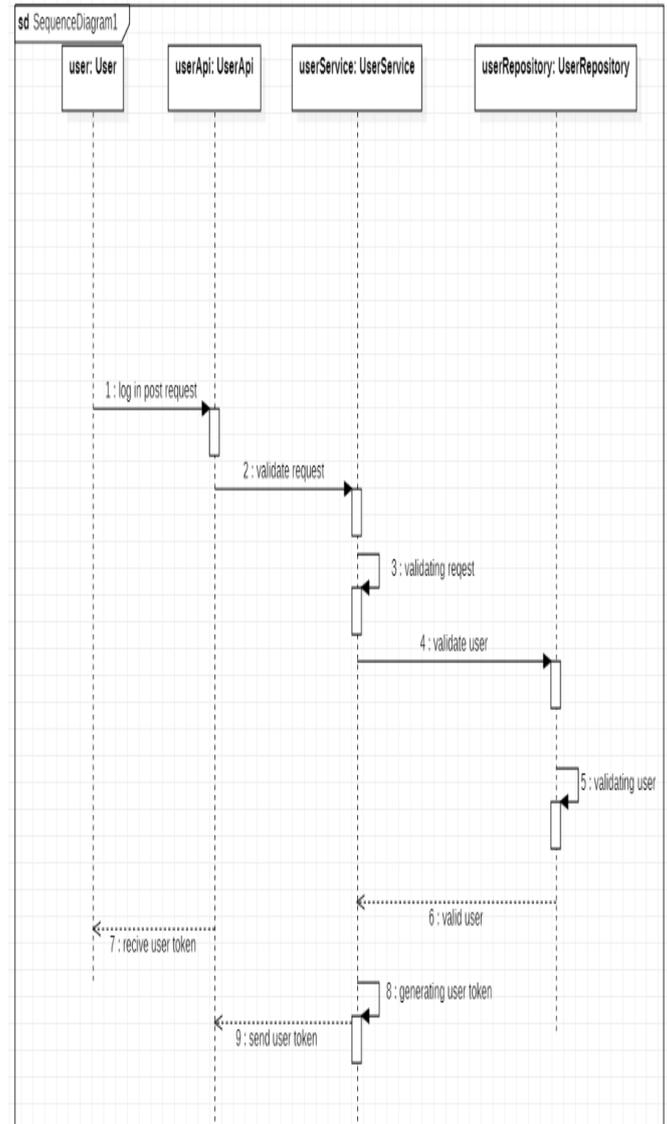


### Class Diagram

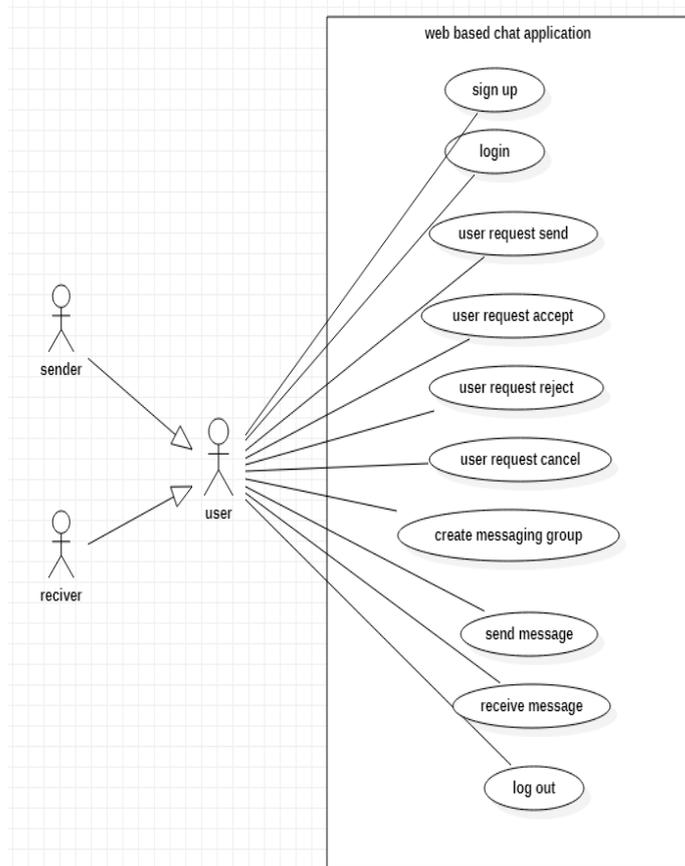


### Sequence Diagram

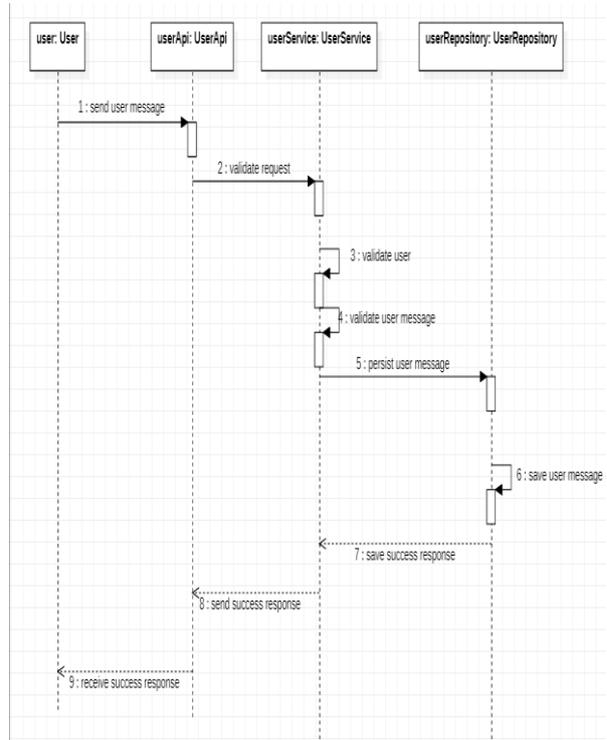
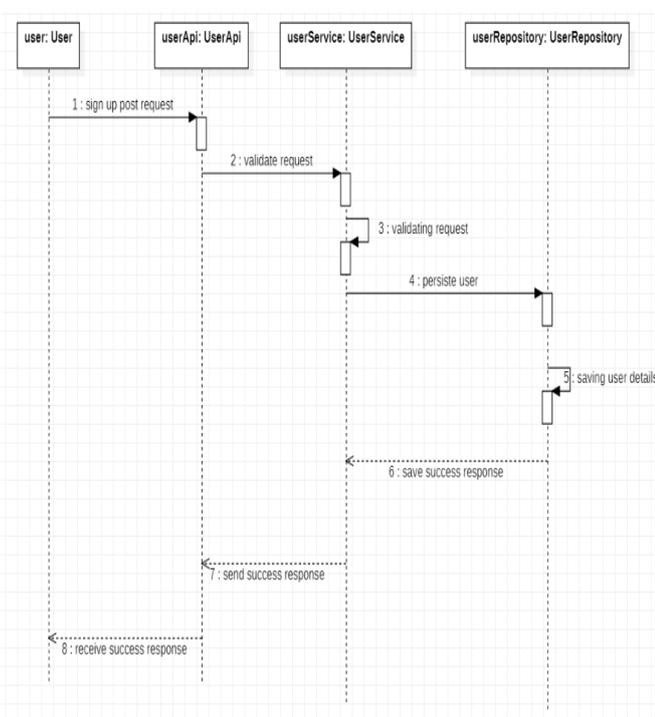
#### For Login



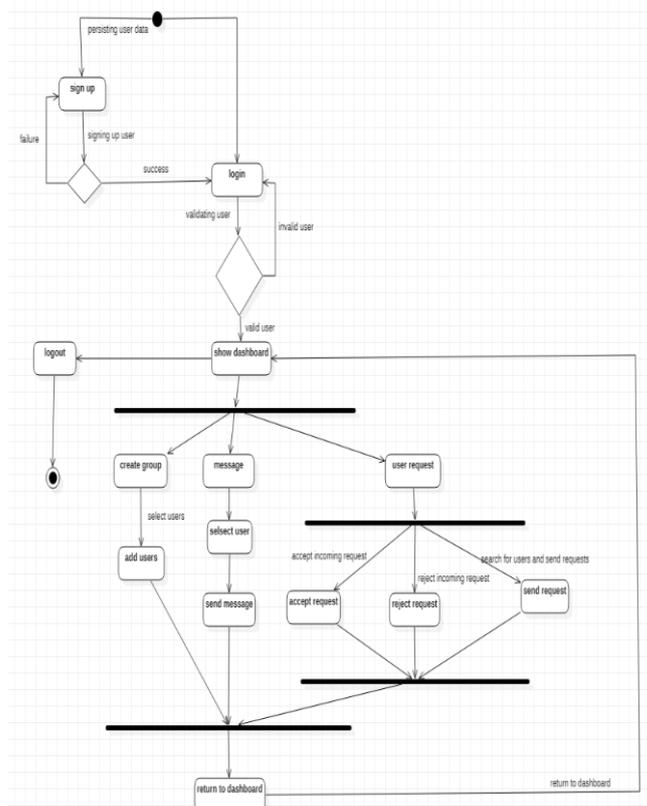
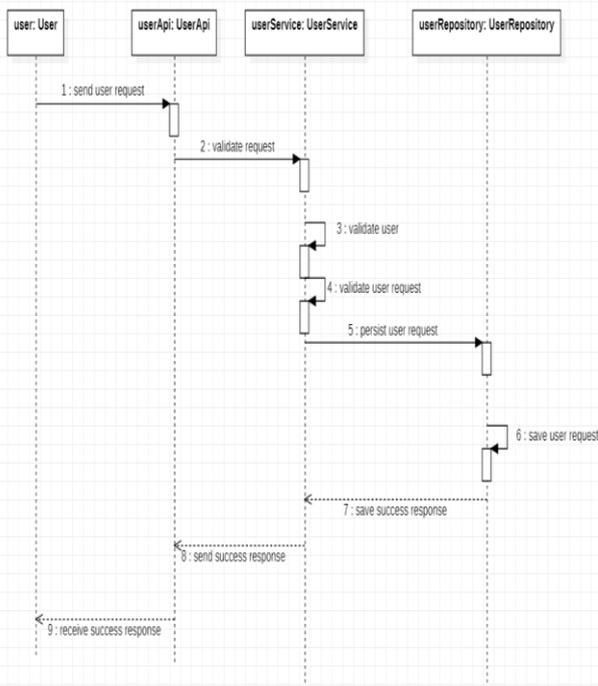
### Use Case Diagram



### For Sign Up



### For User Request

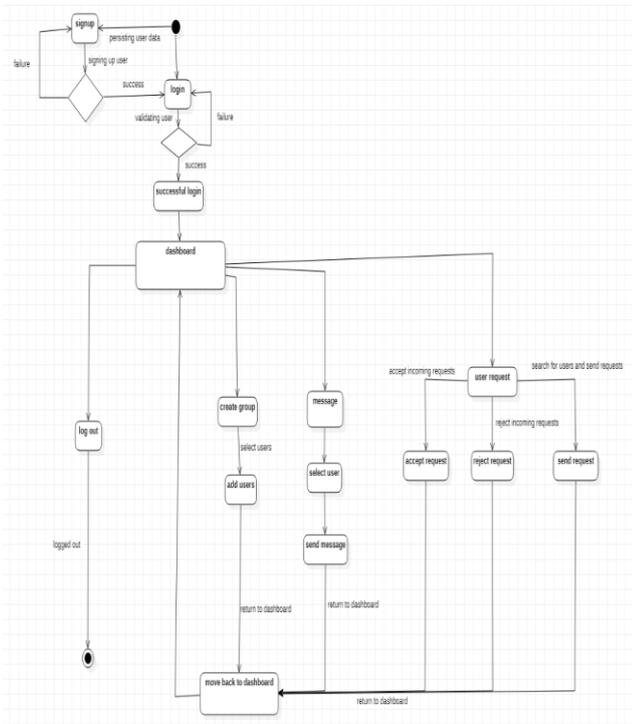


### For Message

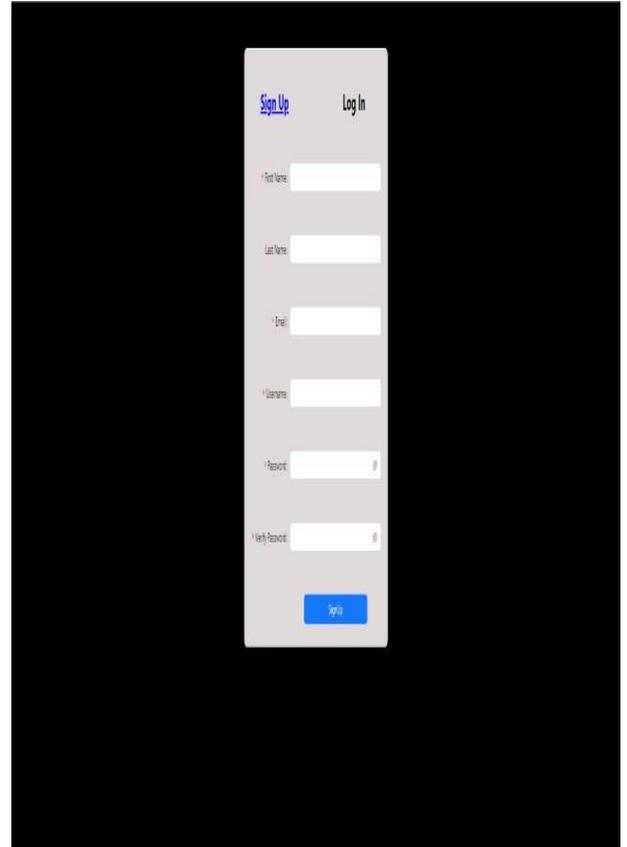
### Activity Diagram

## Results

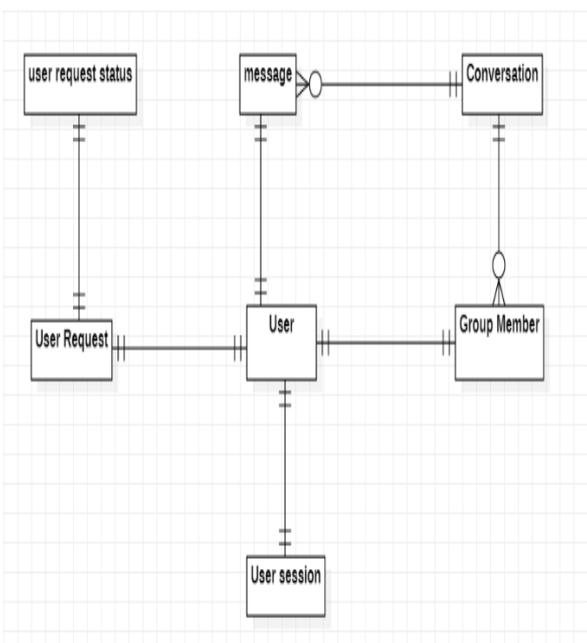
### State Chart Diagram



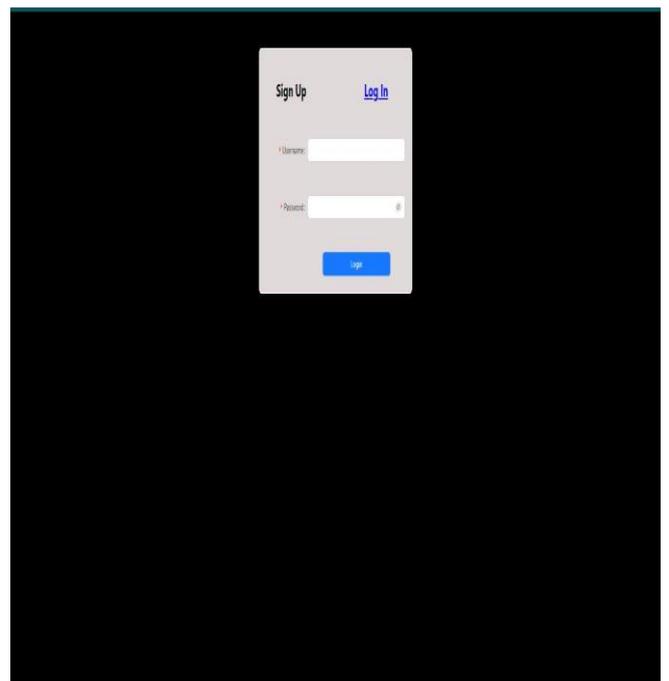
### Sign Up

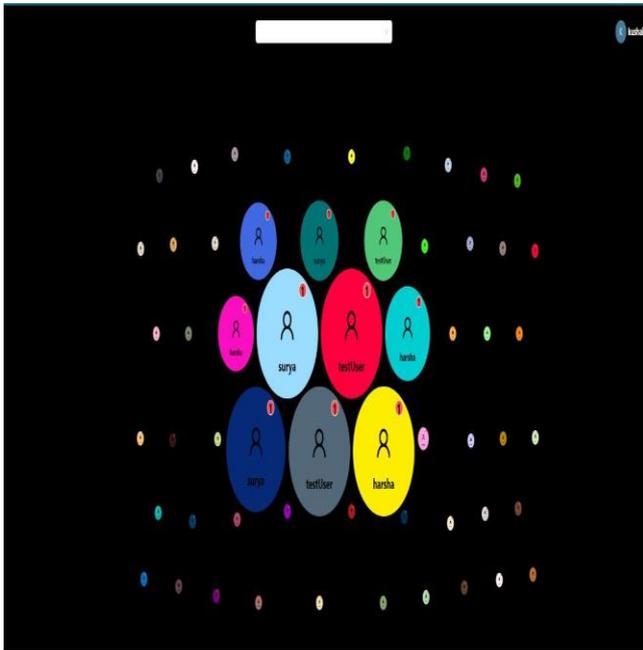


### ER Diagram

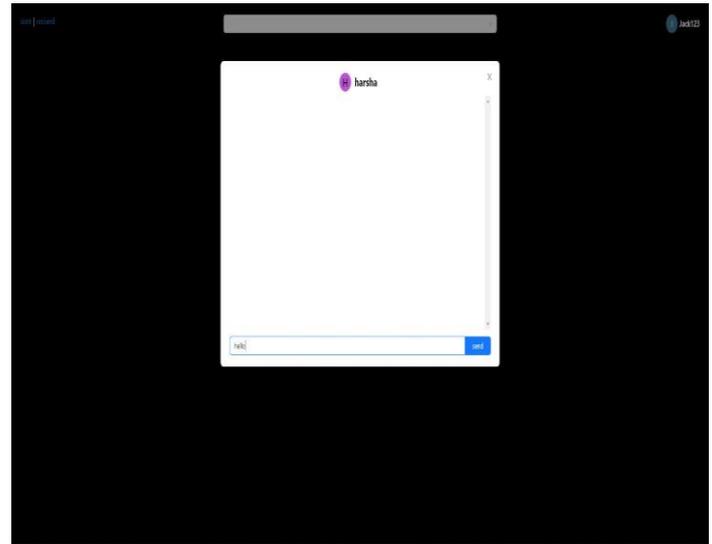


### Login

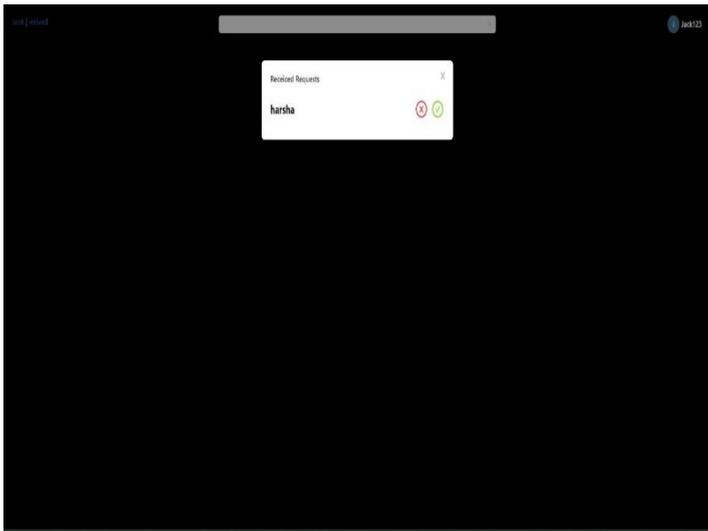




Dashboard

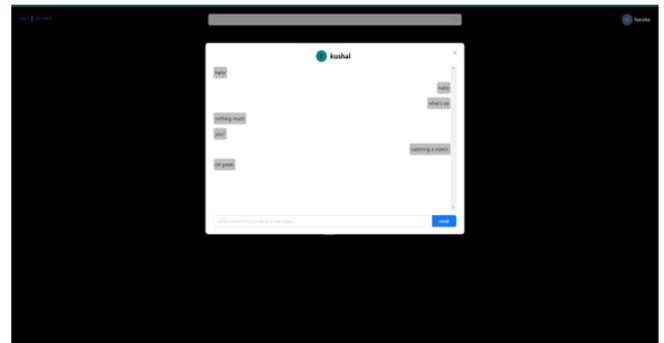


Received Requests



Chat Box

Send Requests



## Conclusion

The chat app offers a more effective and adaptable communication system. created with latest technology to offer a solid system. The system's primary benefits include group chat, improved security, instant messaging, and real-world contact. Users who want to communicate with others privately may find this application to be the most useful, and we can also add some modifications to this current chat application to increase the client base. Using WebSocket's and the STOMP protocol, this application offers users a full-duplex method of communication.

## Future Enhancements

There is always a room for improvements in any apps. Right now, we are just dealing with text communication. There are several chat apps which serve similar purpose as this project, but these apps were rather difficult to use and provide confusing interfaces.

A positive first impression is essential in human relationship as well as in human computer interaction. This project hopes to develop a chat service Web app with high quality user interface. In future we may be extended to include features such as:

- File Transfer
- Voice Message
- Video Message
- Audio Call
- Video Call
- Group Call
- Improving different text style and font size

As this is built using the react as the front end, we can convert this into and mobile application using React Native.

## REFERENCES

[1] Design and Implementation of real time chat interfacing server

<https://ieeexplore.ieee.org/document/7849628>

[2] A Secure Chat Application Based on Pure Peer - to - Peer Architecture

<https://thescipub.com/pdf/jcssp.2015.723.729>

[3] <https://projectsgeek.com/2017/12/simple-chat-room-system-project.html>

[4] <https://reactjs.org/docs/getting-started.html>

[5] <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#native-image.introducing-graalvm-native-images>

[6] <https://docs.spring.io/spring-framework/docs/4.2.x/spring-framework-reference/html/websocket.html>

[7] Chat application with distributed systems

[https://digitalcommons.harrisburgu.edu/cgi/viewcontent.cgi?article=1009&context=csms\\_student-coursework](https://digitalcommons.harrisburgu.edu/cgi/viewcontent.cgi?article=1009&context=csms_student-coursework)

[8] The Influence of UI UX Design to Number of Users Between 'Line' and 'Whatsapp'

<https://ieeexplore.ieee.org/document/9609775>

[9] The influence of design aesthetics in usability testing: Effects on user performance and perceived usability

<https://www.researchgate.net/publication/38069824> The influence of design aesthetics in usability testing Effects on user performance and perceived usability