

CHAT APPLICATION USING REACT AND SPRING BOOT

CH. GUNA PRANITH REDDY, S.BAL NARESH, T. VENKANNA

Mr. Pasam Ramu (Guide)

**Sreenidhi Institute of Science and Technology
Hyderabad**

ABSTRACT:

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.

INTRODUCTION

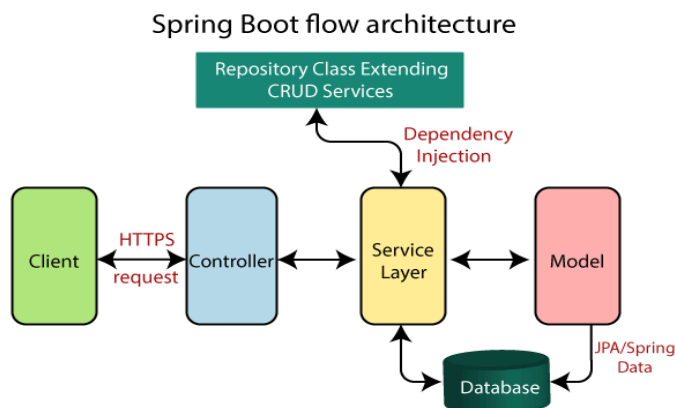
In the present days the chat applications are having the same generic user interface in contrast to this concept we have tried to design a new outlook to our chat application in the aspect of user interface. The users have the capability to do one to one messaging, group messaging. The users can receive messages instantly while online. The server-side application can be hosted on any machine with JVM and communicate with client side via web socket.

The application is designed as a web application. It provides a general architecture for chat applications, and anyone can use it as the basis for providing instant messaging capabilities. The main purpose of the software is to provide users with an instant messaging tool that can

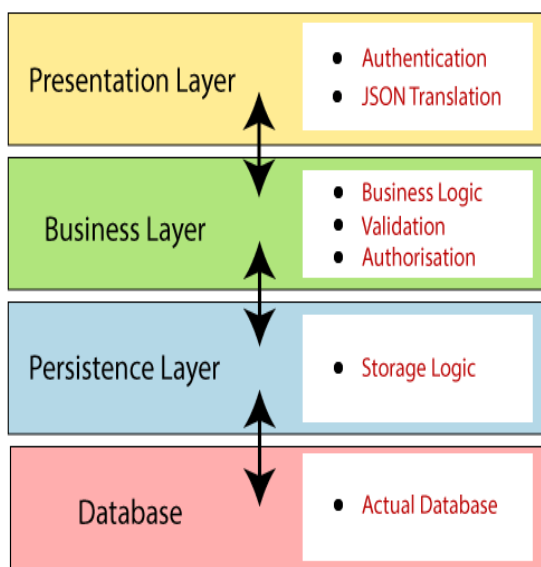
handle large number of users simultaneously when needed and can be easily done.

The one of the most important targets of this project is the delivery information instantly. Thus, picking the right technologies in communicating has become the focus of this project. Web chat is text-based chat between people. People can communicate with each other over the web.

METHODOLOGY



ARCHITECTURE



- 1. Presentation Layer:** 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 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.
- 5.** 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

connection to keep it alive, this new connection will be known as WebSocket. Once the communication link establishment and the connection are opened, message exchange will take place in bidirectional mode until connection persists between client-servers. If anyone of them (client-server) dies or decide to close the connection is closed by both parties. The way in which socket works is slightly



server, the connection is terminated from both ends.

There is the client which is a web browser and a server, whenever we initiate the connection between client and server, the client-server made the handshaking and decide to create a new connection and this connection will keep alive until terminated by any of them. When the connection is established and alive the communication takes place using the same connection channel until it is terminated.

This is how after client-server handshaking, the client-server decides on a new

different from how HTTP works, the status code 101 denotes the switching protocol in WebSocket.

REACT JS

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. 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.

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.

It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram.

The main objective of ReactJS is to develop User Interfaces (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app.

The JavaScript virtual DOM is faster than the regular DOM. We can use ReactJS on the client and server-side as well as with other frameworks.

It uses component and data patterns that improve readability and helps to maintain larger apps.

Before learning ReactJS in-depth, you must have a good knowledge of JavaScript, HTML5, and CSS.

SPRING BOOT

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.

Spring Boot is a Spring module that provides the RAD (Rapid Application Development) feature to the Spring framework.

Spring Boot is one of the best backend frameworks that developers count on while performing backend web development tasks. It is a Spring-based framework that allows developers to write production-grade backend web applications in Java.

Spring Framework offers a dependency injection feature that lets objects define their own dependencies that the Spring container later injects into them. This enables developers to create modular applications consisting of loosely coupled components that are ideal for microservices and distributed network applications.

MYSQL

MySQL is an open-source relational database management system.

MySQL is currently the most popular database management system software used for managing the relational database.

It is open-source database software, which is supported by Oracle Company.

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services.

MySQL is a tool used to manage databases and servers, so while it's not a database, it's widely used in relation to manage and organizing data in databases.

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.

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(),
getDerivedStateFromProps(), render(),
componentDidMount()

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

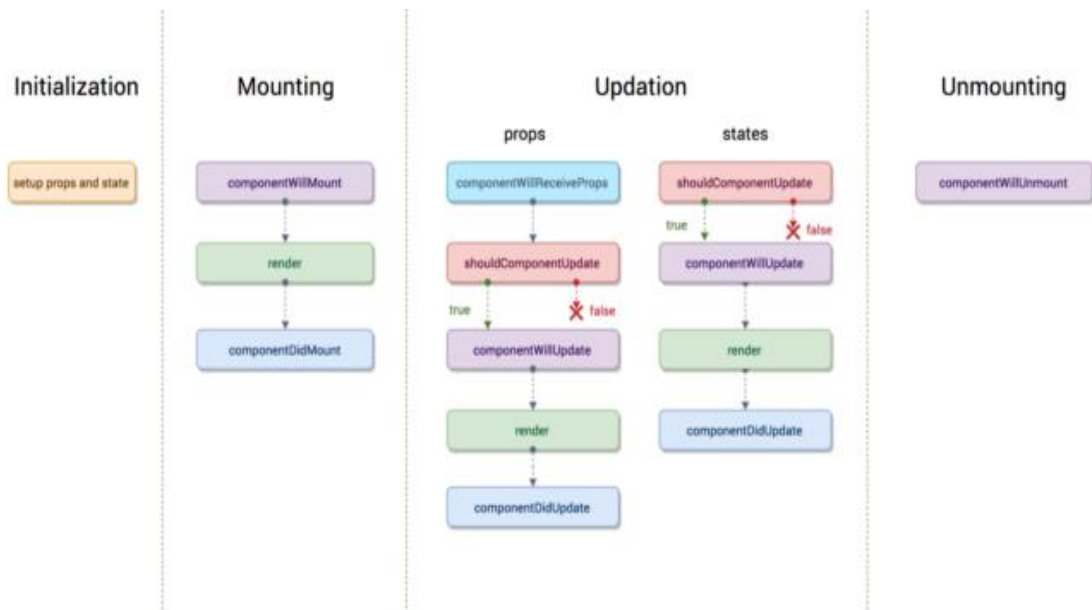
The next phase in the lifecycle is when a component is removed from the DOM, or *unmounting* as React likes to call it.

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.



RESULTS

Output screen running in local host

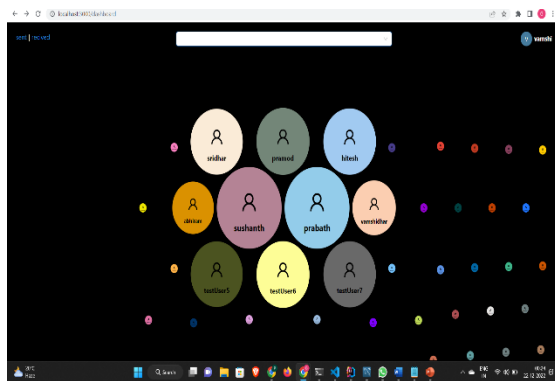


Fig9: user interface

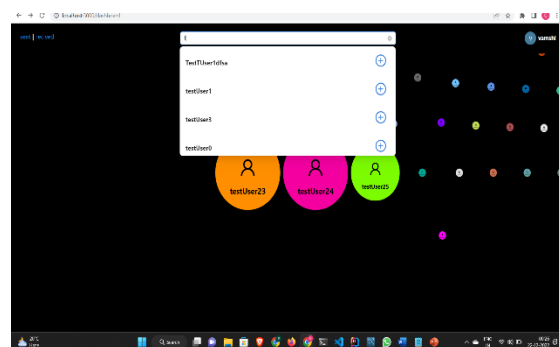
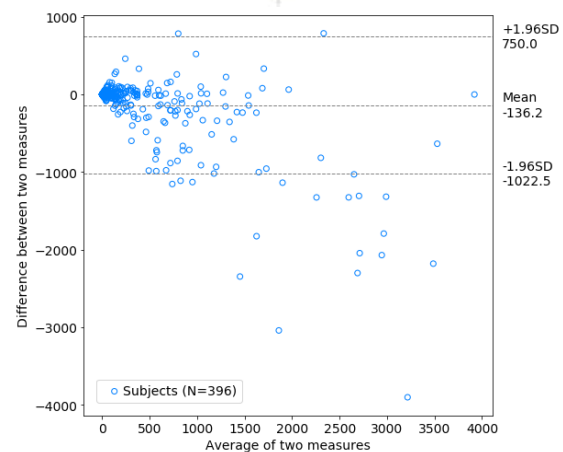


Fig10: adding new user

MERITS

With unique user interface the possibility of attracting more users is high and using better application with faster speeds improves the effectiveness and efficiency in users day to day life.

SOFTWARE AND HARDWARE REQUIREMENTS

In this project we have used Windows 10, Intel Core i7-9700 CPU of 3.00 GHz, 32.0 GB RAM, and NVIDIA GeForce RTX 2080 graphics card.

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

ACKNOWLEDGMENTS

We would like to express our special thanks to our mentor Dr. Pasam Ramu who gave us a golden opportunity to do this wonderful project on this topic which also helped us in doing a lot of research and we came to know about so many new things. We are really thankful to them.

Secondly, We would also like to thank my friends who helped us a lot in finalizing this project within the limited time frame.