**Frameworks:**

### Frontend Frameworks

#### 1. AngularJS

AngularJS is a JavaScript-based open-source front-end web application framework. It is maintained by Google.

* Strengths:
  + Two-way Data Binding: Changes in the user interface instantly influence the application objects and vice versa.
  + MVC Architecture: AngularJS uses Model-View-Controller architecture, which helps in maintaining the separation of concerns.
  + Dependency Injection: AngularJS has a built-in dependency injection subsystem that helps the developer by making the application easier to develop, understand, and test.
* Weaknesses:
  + Performance: For very complex and dynamic applications, the two-way data binding can lead to performance issues.
  + Verbosity: It can be verbose and may require you to write more code than other frameworks/libraries.

#### 2. ReactJS

ReactJS is a JavaScript library for building user interfaces, maintained by Facebook.

* Strengths:
  + Virtual DOM: ReactJS creates a virtual DOM, which improves the performance and efficiency of the application.
  + Component-Based Architecture: React encourages the use of reusable components, making the code more maintainable.
  + One-Way Data Binding: React follows one-way data binding, which makes the code more stable.
  + Strong Community and Ecosystem: React has a large community of developers and a vast array of third-party libraries.
* Weaknesses:
  + Learning Curve: Although React’s learning curve is generally considered to be less steep than AngularJS, it can still be challenging, particularly when it comes to the ecosystem and advanced features.
  + Integration: React is only a library for building UIs, so it needs to be integrated with other tools and frameworks for full functionality.

### Backend Frameworks

#### 1. Node.js

Node.js is not a framework; it's a runtime environment for executing JavaScript code on the server side.

* Strengths:
  + Non-blocking I/O: Node.js uses a non-blocking I/O model, which makes it suitable for building scalable applications.
  + Large Ecosystem: NPM, the Node.js package manager, provides access to a large number of libraries and tools.
* Weaknesses:
  + Young and Unstable: Some NPM packages are not well-maintained or stable.

#### 2. Spring Boot

Spring Boot is a Java-based framework used to create a microservice. It is developed by Pivotal Team.

* Strengths:
  + Rapid Development: Spring Boot offers a fast way to set up, configure, and run applications.
  + Microservices Architecture: It is well-suited for building microservices.
  + Strong Integration: Spring Boot can be easily integrated with various databases and third-party libraries.
  + Comprehensive: Like AngularJS, Spring Boot is a comprehensive framework that provides a wide range of functionality.
* Weaknesses:
  + Learning Curve: Spring Boot has a steep learning curve, particularly for developers new to the Spring ecosystem.
  + Memory Consumption: Spring Boot applications can consume more memory and have a slower startup time compared to other frameworks.

**M, V and C (or MVC’s) of application:**

#### M (Model):

* Location: Implicitly on the Server-side
* Description: Instead of having explicit data models on the server, the data might be received and immediately passed through to the client. The server acts as a thin layer, potentially just handling the connection and routing of messages.
* Reason: This approach simplifies the server-side architecture and reduces the need for extensive data modeling, which might be suitable for a lightweight chat application.

#### V (View):

* Location: Client-side (AngularJS for Application 1, ReactJS for Application 2)
* Description: The view is responsible for rendering the user interface and displaying the chat messages to the user.
* Reason: The client-side frameworks are used to create a dynamic and responsive user interface, providing a good user experience.

#### C (Controller):

* Location: Client-side (AngularJS for Application 1, ReactJS for Application 2)
* Description: The controller on the client-side handles user interactions, updates the view, and communicates with the server to send/receive messages.
* Reason: Having the controller on the client-side allows for immediate response to user interactions, creating a responsive application.

#### Server-side Logic:

* Location: Server-side (Spring Boot for Application 1, Node.js for Application 2)
* Description: The server handles incoming connections, routes messages between users, and ensures that messages are delivered in real-time using STOMP.
* Reason: Placing this logic on the server ensures that the real-time communication is managed centrally, providing consistency across all connected clients.