# Requirement

Build a simple java chat server that handles one-to-one textual IMs, contact lists and presence.

### Architecture

A simple chat server is built using JAVA EE 7 (EJB 3.2, JMS 2.0, JAX-RS 2.0) technologies. Java Message Service is used as a chat room and HornetQ is acted as a Message Oriented Middleware. Point-To-Point messaging is used and three queues (contacts, chat & presence) are used for this purpose. Application is deployed in WildFly (Jboss/redhat) application server.

Users are represented as Stateful Session Beans (SSB), message producers to queues, maintaining the contacts list, chat messages and presence information. Message Driven Beans (MDB) is used to consume messages from queues. Singleton Session Bean is used to cache all the users’ SSB.

User1

Message Driven Bean

Stateful Session Bean

Queue

Produce Consume

User2

Stateful Session Bean

For instance when User1 wants to send a friend request to User2, User1 SSB sends a message comprising request to “contacts queue”. MDB receives this message, decodes the message and calls User2 SSB and updates its contact list. The same approach is followed for sending chat message and updating presence status.

### Testing

Restful web services are used for testing. For simplicity, HTTP GET is used to send required information (username, friend, message, presence, operation). These services call Singleton Session Bean to get access to particular user’s Stateful Session Beans for business processing. The application has been deployed in Amazon Web Services (AWS) cloud and can be tested using following urls. If you are interested in using this application, please contact me to know the Amazon instance name.

1. Contact Request – Required fields (username, friend, operation)

<http://xxx.compute-1.amazonaws.com/IMServer/resources/ms?username=raj&friend=konk&operation=1>

1. Message Request - Required fields (username, friend, message, operation)

<http://xxx.compute-1.amazonaws.com/IMServer/resources/ms?username=raj&friend=konk&message=msg-rk-1&operation=2>

1. Presence Request - Required fields (username, presence, operation)

<http://xxx.compute-1.amazonaws.com/IMServer/resources/ms?username=raj&presence=4&operation=3>

1. Obtain All Users’ information - Required fields (operation)

<http://xxx.compute-1.amazonaws.com/IMServer/resources/ms?operation=4>

### Assumptions

1. Users need to be active (user’s SSB is created) to initiate any request among them.
2. State is maintained in memory. Otherwise, during server shutdown or SSB inactiveness, the state is stored in disk during PrePassivate and restored during PostActivate methods.
3. WildFly Application server is used. Otherwise, with little configuration changes, any application server can be used.
4. Basic positive testing is performed.