Messaging Application

Chandana Dayapule

Agenda

- Problem Statement
- Current architecture design
- Scalable architecture design
- Future enhancements/Extended Requirements

Problem Statement

- Build a simple messaging app that allows users to send and receive messages to and from other users
- **App requirements**
  - Login screen
  - Save authentication token for skipping login after first time login
  - List of messages received
  - List of messages sent
  - Delete individual message
  - Compose a new message which includes recipient, title, and body
  - Error messages if the requests are invalid

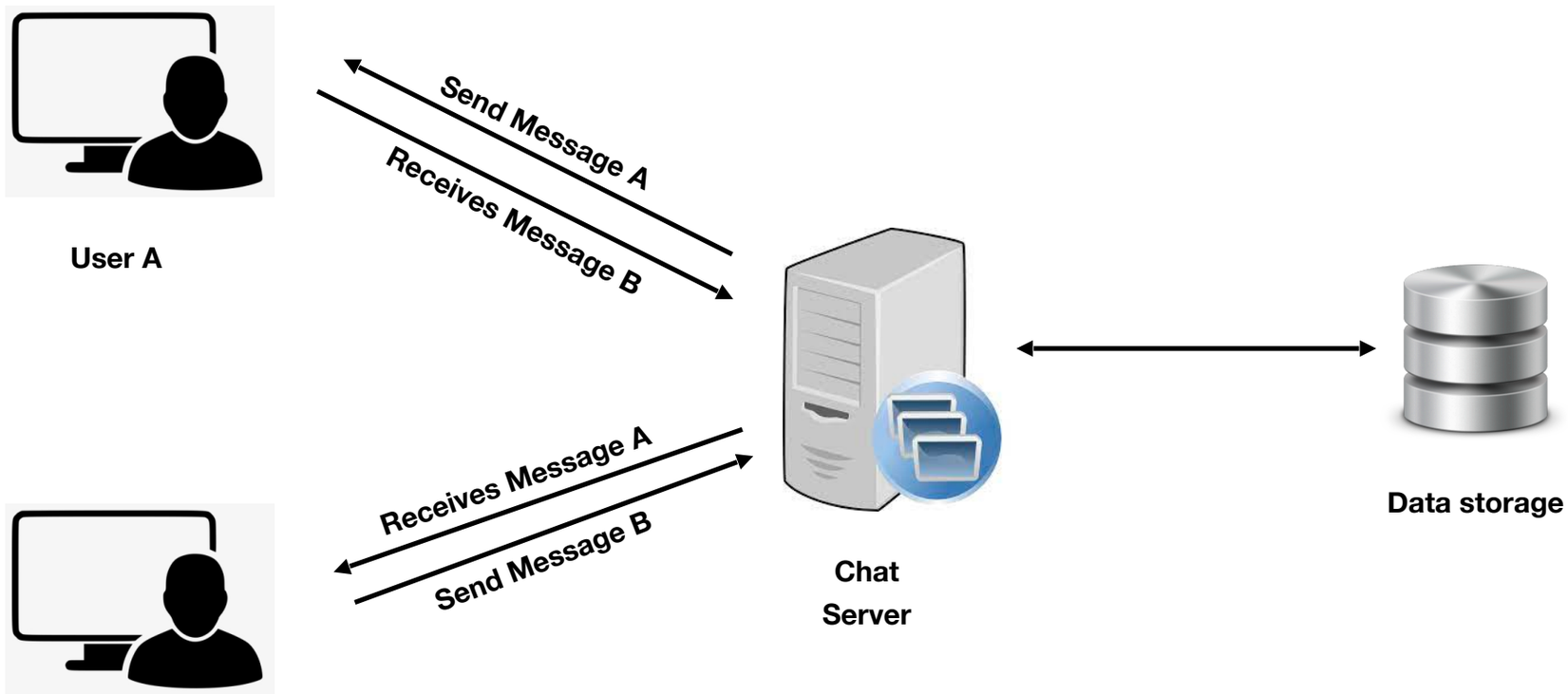
User workflow

- Login -> Select a user -> Send a message/delete a message
- -> Receive message -> See the message
- -> Add a new user -> Send a message

Features available

- Login/logout
- List of conversations with users
- Select an user for starting a conversation
- Sending a message api
- Listing historical conversations between user A, user B
- Delete API for a user message(if the message is already sent then the message will be deleted from one user)

Implemented architecture



The detailed workflow would look like this:

1. User-A sends a message to User-B through the chat server.
2. The server receives the message
3. The server stores the message in its database and sends the message to User-B.
4. User-B receives the message

Tech stack

- **Backend** - Django framework (Python 3.8)
- **Frontend** - ReactJs framework
- **Database** - Postgresql
- **IDE** - pycharm, visual studio
- **API client** - Insomnia, postman
- **Documentation** - Confluence, git readme

Current PostgreSQL tabular design

User table

Column name	Description
id	unique user Id
password	encrypted password
last_login	Last login timestamp
is_superuser	If he/she is a super user - currently not in use
username	unique username for indivuduals registering app
first_name	first name of the user
last_name	last name of the user
email	email address
is_staff	is the user a staff or not
is_active	active or not
date_joined	when was the user joined

Message table

Column Name	Description
id	id for converation
message	actual message between user A, user B
image	Image attachment link (for future use)
file	File attachment link (for future use)
conversation_id	Unique conversation Id
conversation_subject	Subject of the conversation (currently its empty)
archived	Is the conversation Archieved
publish	Date of the conversation publish
timestamp	Timestamp of the coversation with timezone
updated	If the conversation was updated (currently not in use)
msg_receiver_id	User Id of the reciever
msg_sender_id	User Id of the sender
delivered	[Future use] to track the message was delivered
read	[Future use] to track the message was read

Resources

Git hub link:

https://github.com/chandanadayapule/messaging-app

User login flow

Enter Username and password

test1

••••

Log in

Messaging app

Home

Compose

test3
You: Hi, test3

test2
You: Hi, sending the message from API

test1 about 23 hours ago
hi

test1 about 23 hours ago
hi I am test1

test2 about 23 hours ago
hi, messaging from test2

test1 about 14 hours ago
Hi test2, messaging from test1

test1 about 1 hour ago
Hi, sending the message from API

test1 33 minutes ago
Hi, sending the message from API

Profile

Log out

Latest conversations

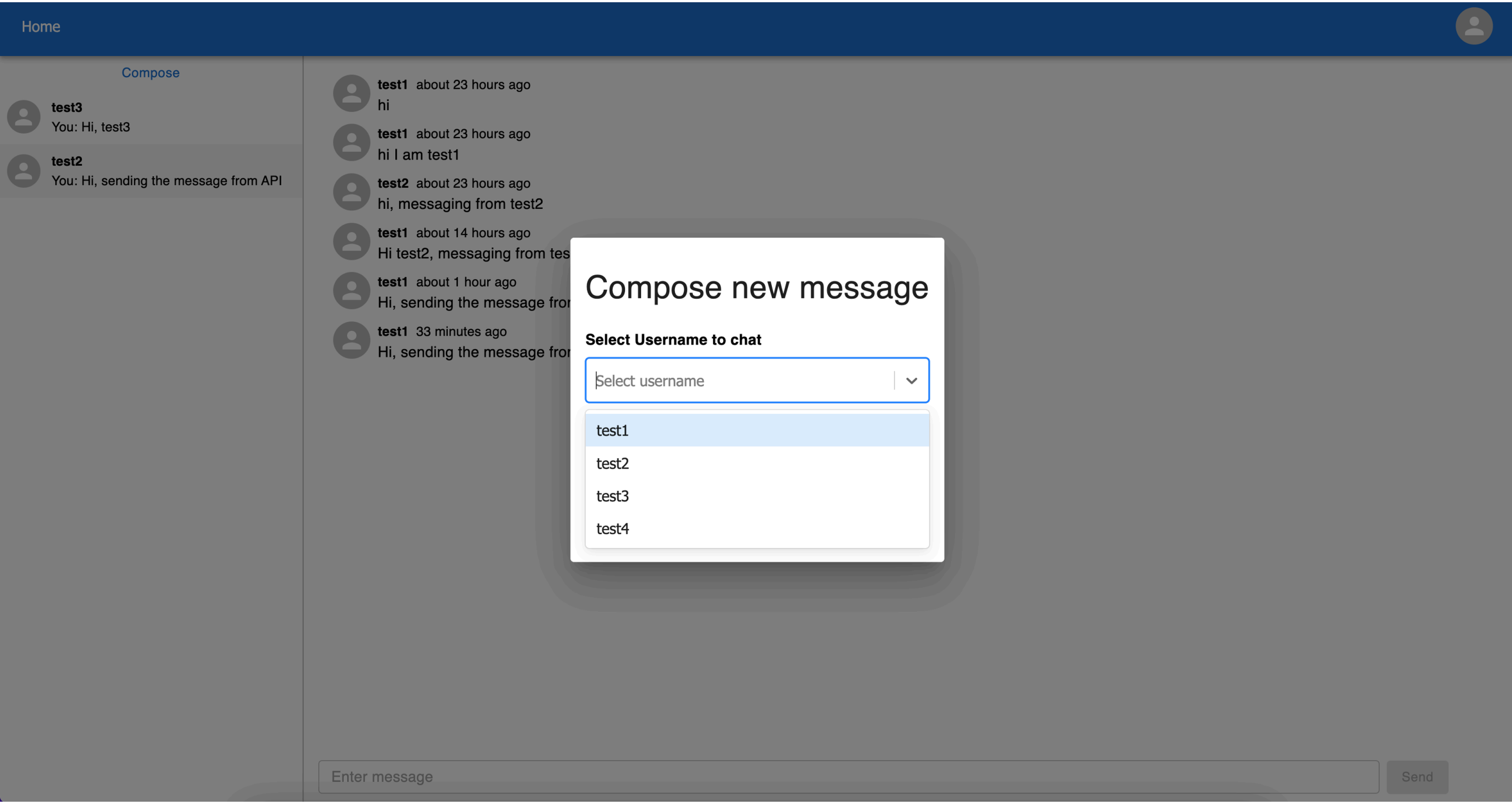
Chat box -

1. **Can view conversation between user A, user B**
2. **Processed timestamp (33 min s ago, 23 hours ago)**
3. **Text box, send button**

Enter message

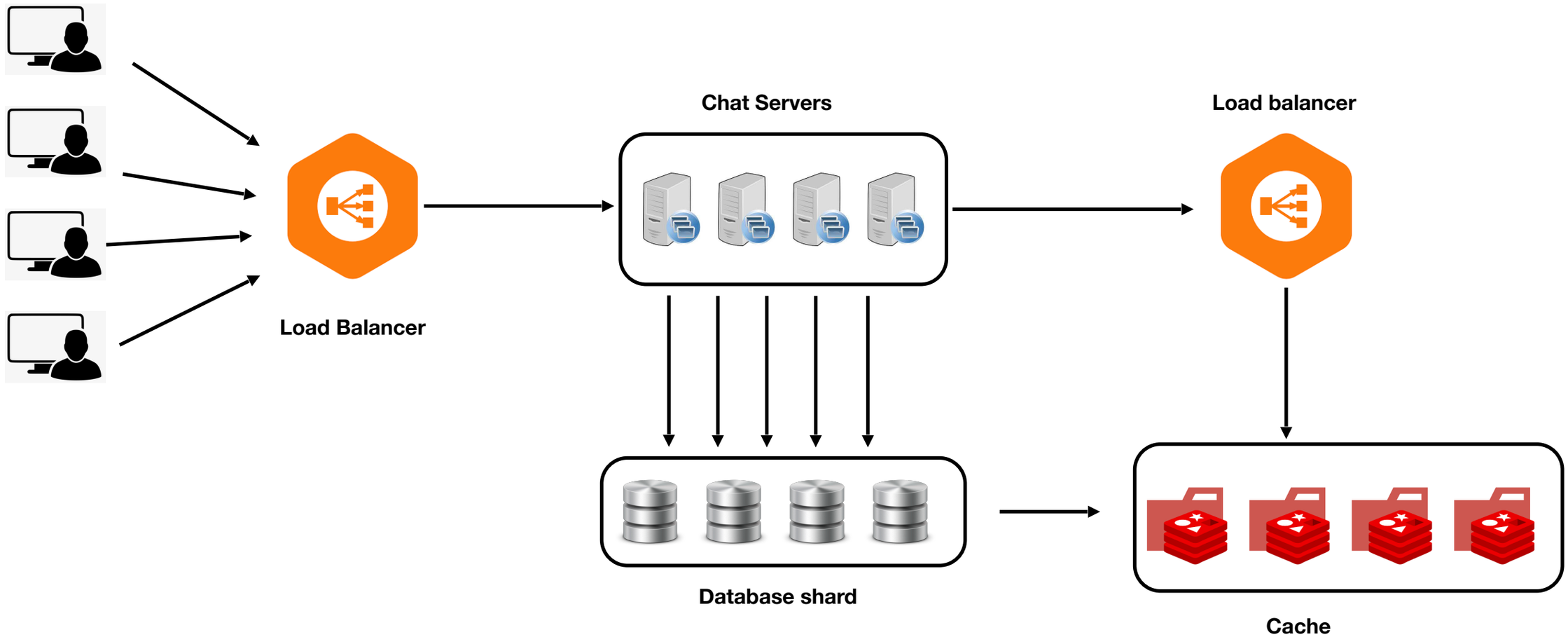
Send

Selecting user for composing a new message



Appendix

Scalable architecture



Extending features for the simple messaging app

- Real-time chatting experience with minimum latency
- System should be highly consistent, users should see the same messages history
- The application should have high availability

Future enhancements

- Group chat
- Push notifications
- Image/File uploads
- User profile
- User archive
- Managing user status - last seen/active/offline