

INDIAN INSTITUTE OF TECHNOLOGY,BOMBAY

CS699 Project

by

Sahil Mukund Patki 183059002

Abhishek Singh 18305R010

Suraj Kumar 18305R008

Athul S Nambiar 183059003

November 2018

Contents

List of Figures	ii
-----------------	----

1	Introduction	1
1.1	Sign-Up and Login	1
1.2	Workspace	2
1.3	Channels And Chatting	3
2	Architecture	4
2.1	Django	4
2.2	Django Channels	5
2.3	PostgreSQL	5
3	Database Design	6
4	Tecnology Stack	7
4.1	Frontend	7
4.2	Backend	7
5	References	8

List of Figures

1	Architecture	1
2	Signup Page	2
3	Login Page	2
4	Workspace	3
5	Channels and Messaging	3
6	System Flow	4
7	Sequence Diagram	5
8	Database Design	6

1 Introduction

This project is designed to emulate the Slack collaboration software. Just like Slack, this application contains the concept of workspaces and channels to manage teams and projects. Workspace is at the top level of the hierarchy and can be used for entire Projects. Channels are the second level of hierarchy which can be used to separate different teams like development, testing, etc. For a user, there can be many workspaces and in a workspace, there can be many channels.

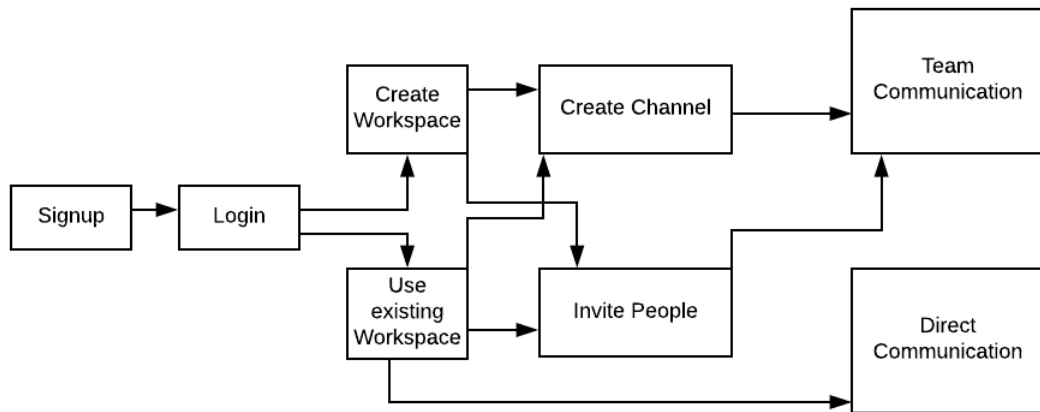
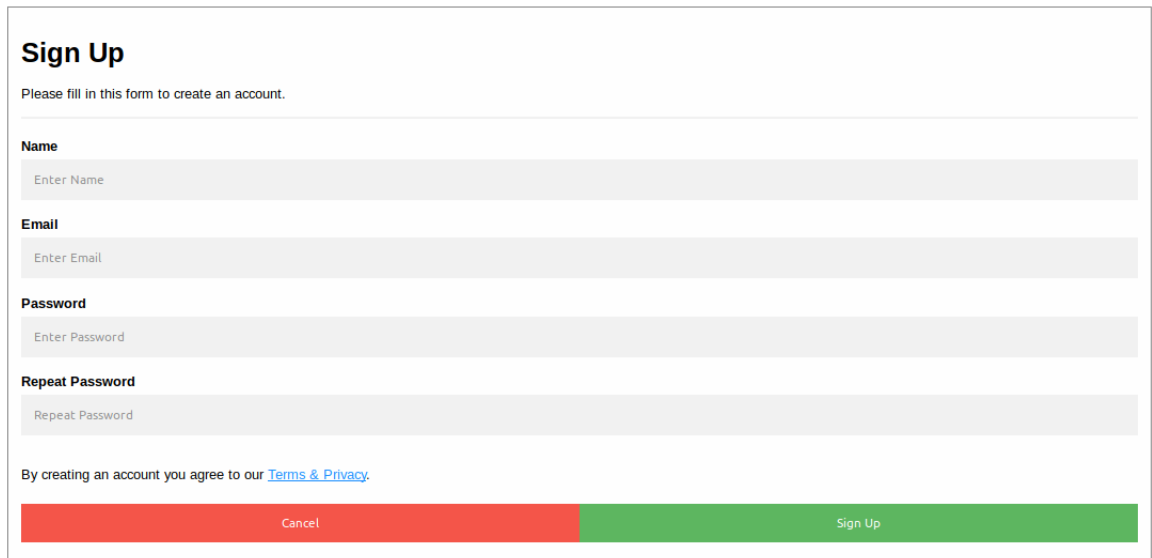


FIGURE 1: Architecture

Our Application is designed in such a way that user can create any number of workspaces and any number of channels in these workspaces and invite users to these channels. But a user who doesn't have admin access cannot create a channel in the workspace.

1.1 Sign-Up and Login

To use our application first the user has to create an account. Account creation can be done in two ways, either by signing up or by invitation. If a user signs up, then they have to provide their details like name, email and password. Otherwise, if the user was invited then they get a mail containing the registration link and a temporary randomly generated password. If they follow this link and login then they have to change their password on the first login. After changing the password, the user will be redirected to the login page.



Sign Up

Please fill in this form to create an account.

Name

Email

Password

Repeat Password

By creating an account you agree to our [Terms & Privacy](#).

Cancel Sign Up

FIGURE 2: Signup Page



SLACK

Email

Password

Login

FIGURE 3: Login Page

1.2 Workspace

When a user logs in, the first page contains a list of workspaces they created or are a part of. The can create new workspace here, search for a workspace(if there are a lot of them) on this page. There is also a logout and change password options available on this page.

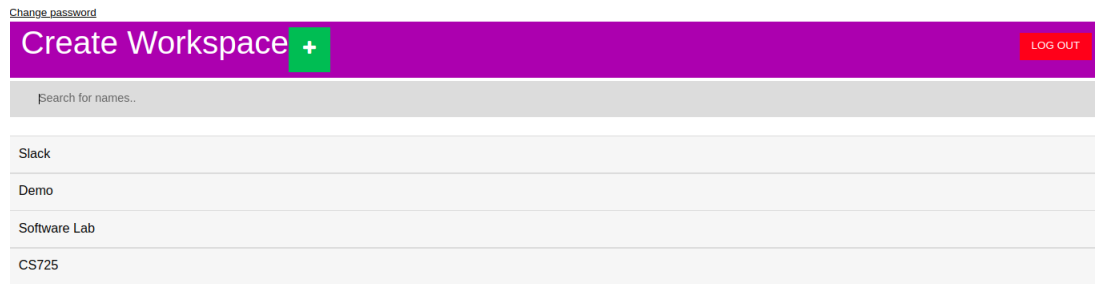


FIGURE 4: Workspace

1.3 Channels And Chatting

When the user clicks on a workspace, a new page containing the list of channels is loaded. This page also contains the interface for adding new channels and inviting new users to the workspace. On clicking either a channel or a user the chats occurred in this space is shown. We can send or receive messages here. There are buttons available to reply on or delete a message. Delete button simply deletes the message, whereas on clicking the reply button a new thread starts in a new page where the user can either send a reply or see all the replies to that specific message. At present only one level of commenting is possible but it can be easily extended.

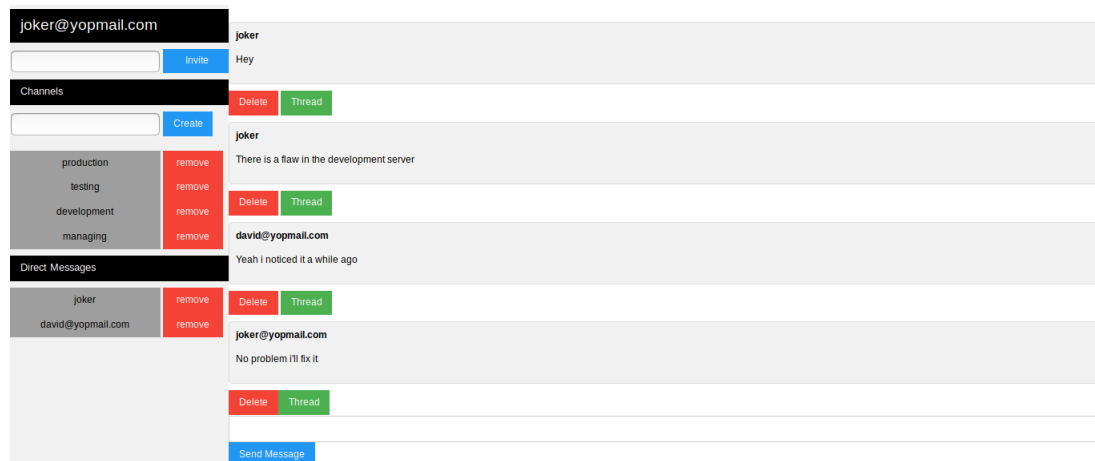


FIGURE 5: Channels and Messaging

2 Architecture

2.1 Django

Django is a free and open-source web development framework based on Python programming language. This framework gained popularity in recent times due to its simplicity and intuitive design architecture. Django follows MVT(Model, View, Template) architecture. It uses apps for encapsulating different components of the project. Using apps ensures code reusability and pluggability.

Core Django framework consists of MVC(Model, View and Controller) architecture. A model consists of a python class which defines data models. These models are mapped to a relational database using Object Relational Mapper (ORM). View part contains functions to handle different HTTP requests with an HTML templating system. A controller is a URL dispatcher which maps URLs to specific functions in view.

Django can work on most servers but we are using default server provided by Django to reduce the amount of setup required.

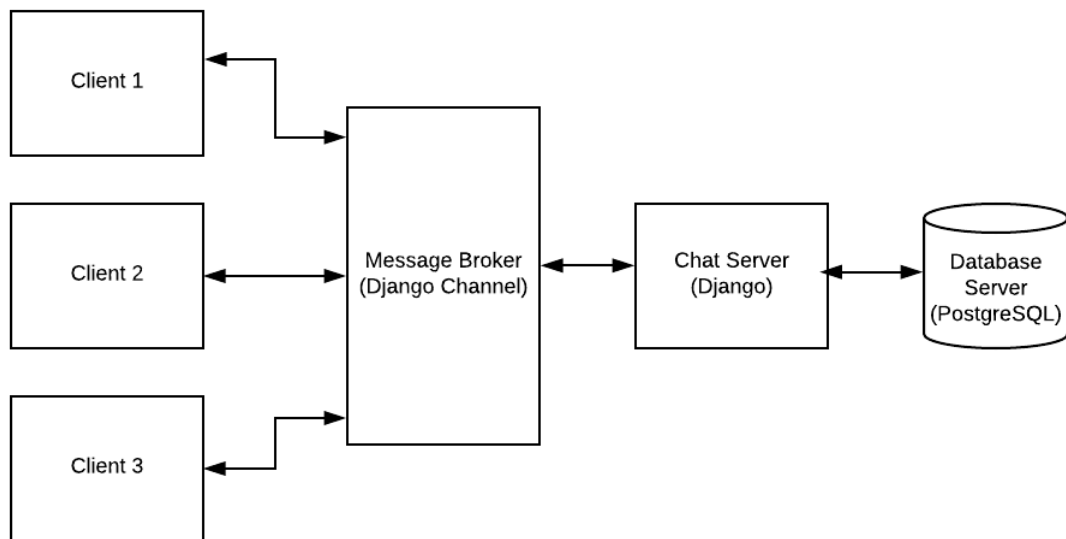


FIGURE 6: System Flow

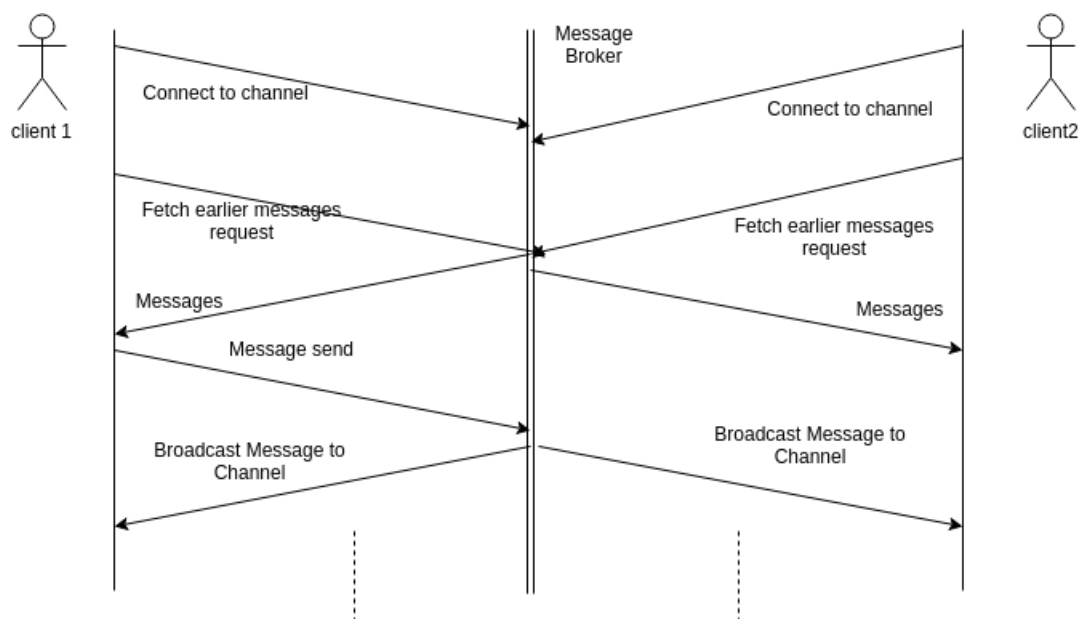


FIGURE 7: Sequence Diagram

2.2 Django Channels

The interaction between the client and server is handled by a middleware called **message broker**. In this chat app, we are using a message broker to keep messages in a queue and forward it to the client when the client is active. The main reason a message broker is required is due to different frameworks being used on the client and server side. Message brokers are the fundamental building blocks of Messaging middlewares.

The message broker we are using is **Django Channels**. The server is running on python and the client uses javascript. Django Channels helps Django to extend its abilities beyond HTTP to handle web sockets and chat protocols. It creates a layer of asynchronous communication under Django's synchronous communication layer. So we can use fully synchronous, fully asynchronous or a mixture of both for communication. Channels provides integrations with Django's auth, session and other systems making it easier to integrate with Django.

2.3 PostgreSQL

PostgreSQL is an object-relational database management system. We are using PostgreSQL because we thought of creating some analytics interface for the app (which we were unable to do), which could have benefitted from the stored procedures and custom aggregate functions available in PostgreSQL. Otherwise, it is similar to other popular SQL database management systems available to us.

3 Database Design

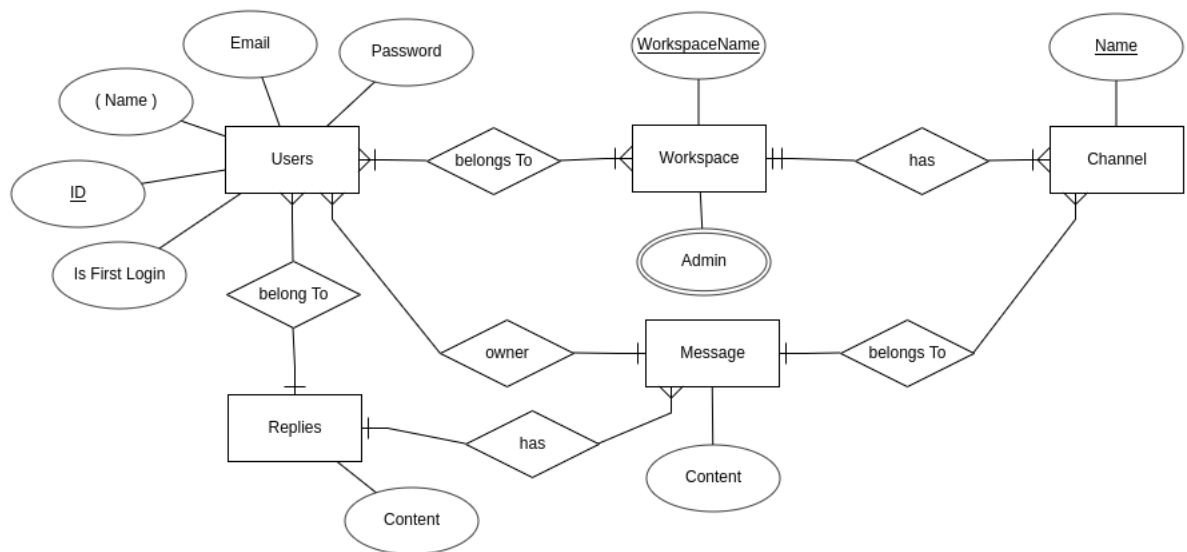


FIGURE 8: Database Design

4 Technology Stack

4.1 Frontend

1. HTML
2. CSS
3. JavaScript

4.2 Backend

1. Python
2. Django
3. Django Channels
4. Redis
5. PostgreSQL

5 References

- **Python docs:** <https://docs.python.org/3/>
- **Django:** <https://docs.djangoproject.com/en/2.1/>
- **Django Channels:** <https://channels.readthedocs.io/en/latest/>
- **PostgreSQL:** <https://www.postgresql.org/docs/>
- **Redis:** <https://redis.io/documentation>