DEPARTMENT OF COMPUTER SCIENCE and ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITAPURI CAMPUS

Project Report
15CSE313 Software Engineering
Product Owner : Mr. Sumesh K.J.

Submitted By:
Srinivas Jayanth Yadhati (AM.EN.U4CSE17328)
T Yashwanth (AM.EN.U4CSE17340)
Vaishnavi Gopalasamy (AM.EN.U4CSE17343)
Viswanath V (AM.EN.U4CSE17355)
Y R Abhishek (AM.EN.U4CSE17357)

TEAM - DELTA S

Project - Team Collaboration Web App/Tool

PROJECT PLAN

1. Project Summary

1.1 Project Overview

This Team Collaboration System is a solution for all Engineers looking for a tool to collaborate efficiently for a software project. It allows the user to plan their tasks, schedule their meetings, and host or join public and private chats. This web-based app will make collaboration easier and more efficient.

1.2 Project Scope

The web application provides an easy-to-use platform for users where they can log in, form a team and manage their work. The following are the current core features :

- Login & Authentication
- To-do list
- Meeting List
- Group Chat

All users of a team can view and edit can set up tasks and establish a deadline for completion of each task, in accordance with the timeline dates. They can also manage their meetings and join public chats or join/create their private chat rooms.

1.3 Development Process

We follow the agile methodology of software development. The product will be developed in two sprints.

Sprint 1

The core modules will be developed in this sprint which would include the following functions:

1. Users can register and log in, and access their dashboard for the features listed below.

2. Users can access a to-do list where they list their tasks, and manage them accordingly to meet deadlines.

3. Cross-check their meeting schedule and make changes accordingly

4. Basic chat server

Note: The product after the first sprint is a complete working system with basic UI.

Sprint 2

This iteration would complete the product in its full functionality with professional UI. The following enhancements will be done in this sprint:
The chat apps will be fully developed.
The UI will be made to look simple and easy to use.

2. Requirements Specification

2.1Functional Requirements

- Sign Up & Log-in
- Public Chat
- Chat Rooms for Private Channels
- To-do list
- Meeting scheduling
- Administrative access on the backend

2.2Non-Functional Requirements

- Accessibility
- Adaptability
- Stability

2.3User Characteristics

The main users of this system will be Software Engineers and university students.

2.4General Constraints

The user may need to check deadlines on their tasks separately to organize tasks efficiently (this constraint may be removed after further development on the app, to improve user experience).

2.5 General Assumptions and Dependencies

Users need to register to access the application's features.
Groups will be managed by an admin(leader).

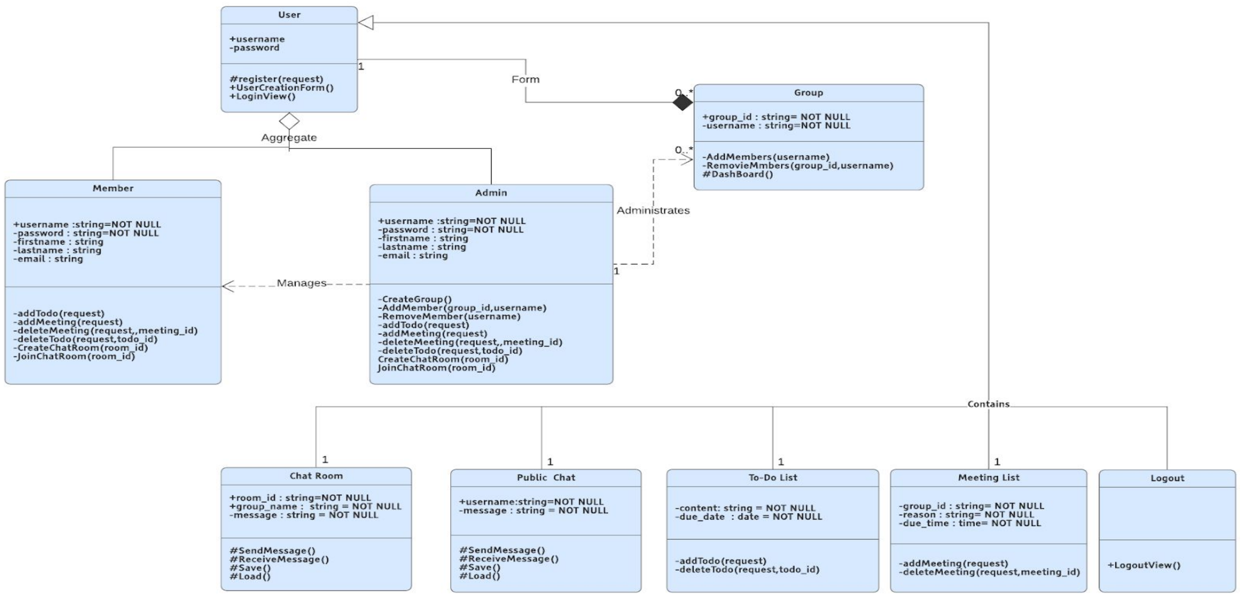
3. Team Organization

The team comprises the following persons. We will have a small team, hence we use a flat team structure of peers with one person having an additional role of scrum master.

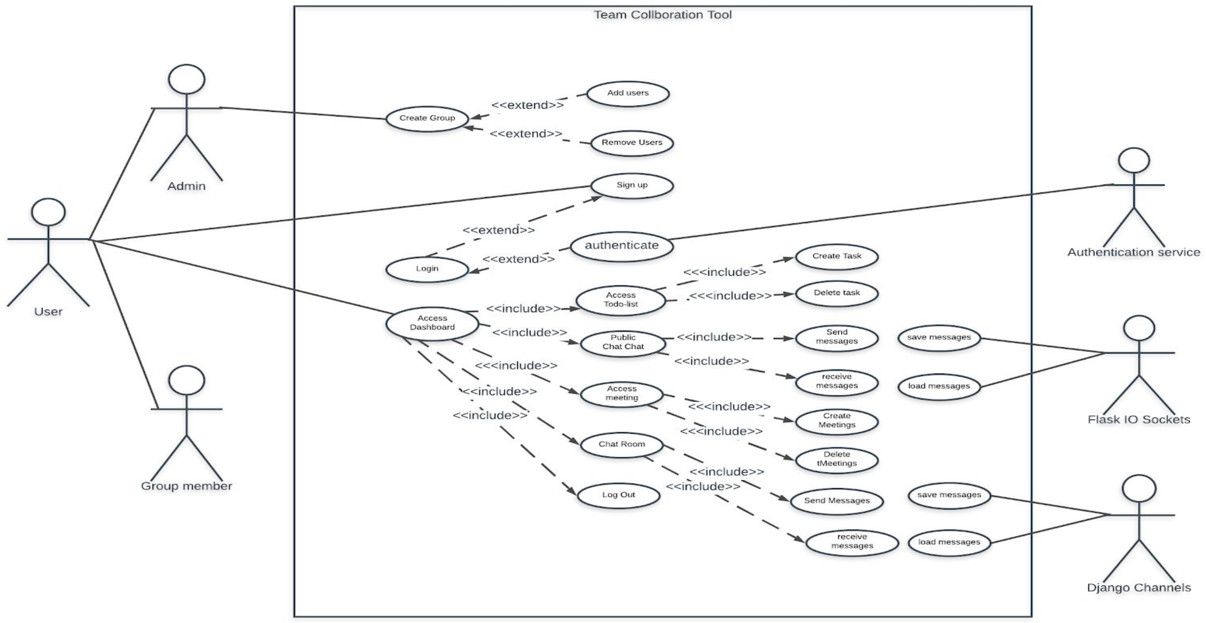
Name	**Role**
Mr. Sumesh	**Product owner**
Sreenivas Jayanth Y	**Scrum master/Developer, Tester**
Yashwanth T.	**Tester, Developer**
Vaishnavi Gopalasamy	**Developer, Tester**
Viashvanath V	**Developer, Tester**
Abshishek Y.R.	**Developer, Tester**

DIAGRAMS

1. CLASS DIAGRAM

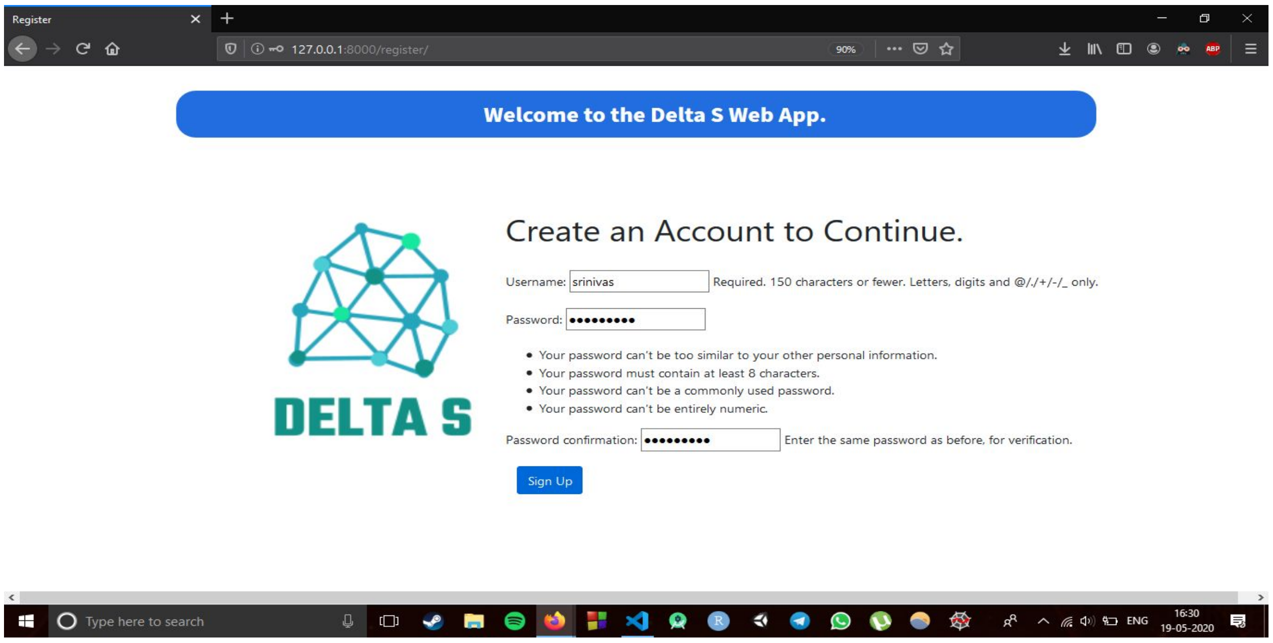

User
+username
-password
#register(request)
+UserCreationForm()
+LoginView()
Form
Group
+group_id : string= NOT NULL
-username : string=NOT NULL
-AddMembers(username)
-RemovieMmbers(group_id,username)
#DashBoard()
Aggregate
Administrates
Member
+username :string=NOT NULL
-password : string=NOT NULL
-firstname : string
-lastname : string
-email : string
-addTodo(request)
-addMeeting(request)
-deleteMeeting(request,,meeting_id)
-deleteTodo(request,todo_id)
-CreateChatRoom(room_id)
-JoinChatRoom(room_id)
Manages
Admin
+username :string=NOT NULL
-password : string=NOT NULL
-firstname : string
-lastname : string
-email : string
-CreateGroup()
-AddMember(group_id,username)
-RemoveMember(username)
-addTodo(request)
-addMeeting(request)
-deleteMeeting(request,,meeting_id)
-deleteTodo(request,todo_id)
CreateChatRoom(room_id)
JoinChatRoom(room_id)
Contains
Chat Room
+room_id : string=NOT NULL
+group_name : string = NOT NULL
-message : string = NOT NULL
#SendMessage()
#ReceiveMessage()
#Save()
#Load()
Public Chat
+username:string=NOT NULL
-message : string = NOT NULL
#SendMessage()
#ReceiveMessage()
#Save()
#Load()
To-Do List
-content: string = NOT NULL
-due_date : date = NOT NULL
-addTodo(request)
-deleteTodo(request,todo_id)
Meeting List
-group_id : string= NOT NULL
-reason : string= NOT NULL
-due_time : time= NOT NULL
-addMeeting(request)
-deleteMeeting(request,meeting_id)
Logout
+LogoutView()

2. USE CASE DIAGRAM



UI SCREENSHOTS

ACCOUNT CREATION



Register

127.0.0.1:8000/register/

Welcome to the Delta S Web App.

Create an Account to Continue.

Username: srinivas Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: •••••••••

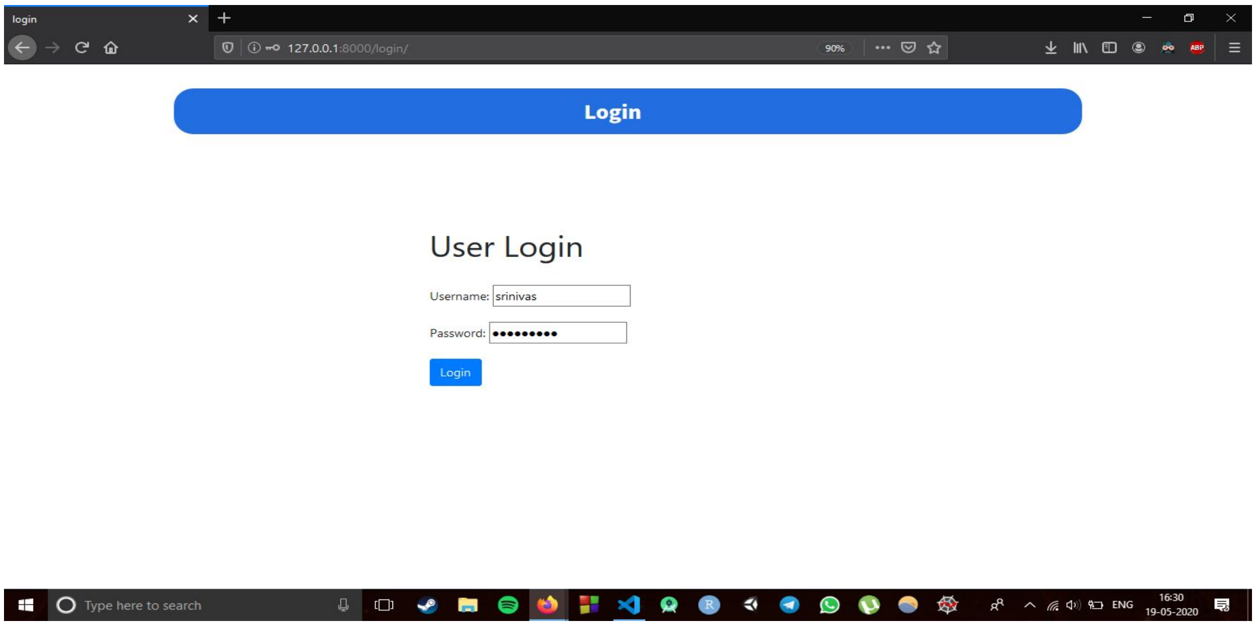
- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: ••••••••• Enter the same password as before, for verification.

Sign Up

Type here to search ENG 16:30 19-05-2020

USER LOGIN



login

127.0.0.1:8000/login/

Login

User Login

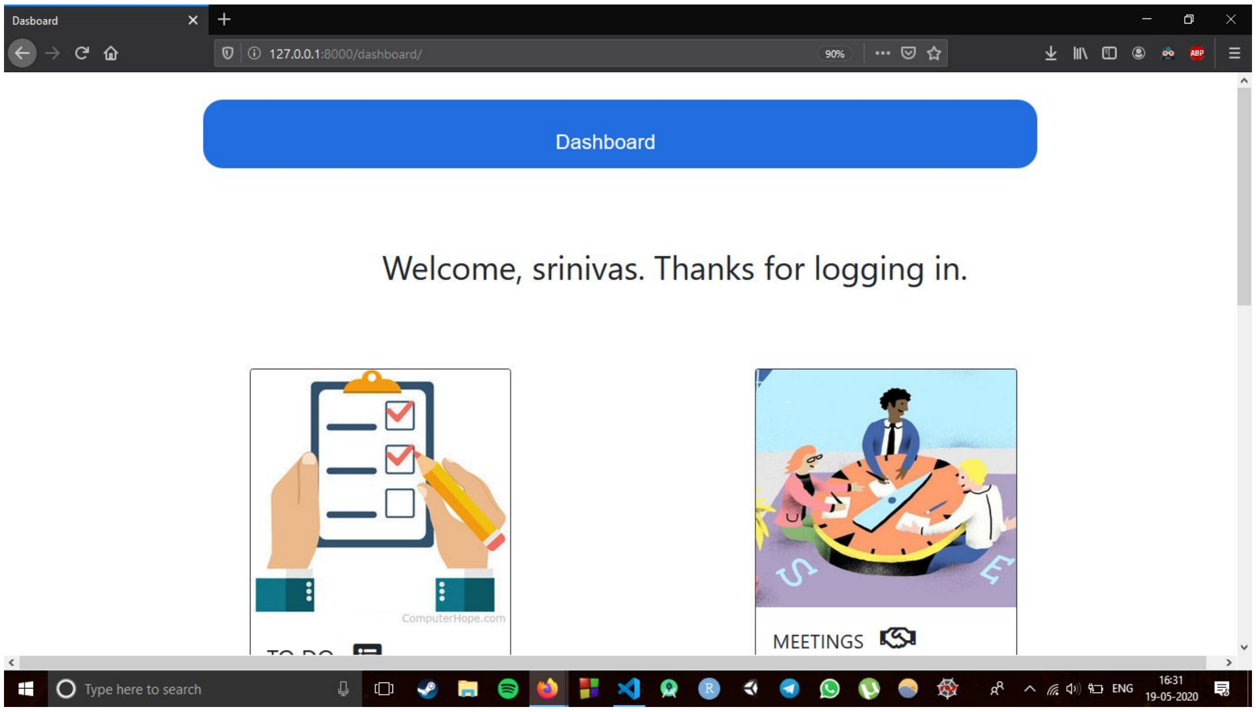
Username: srinivas

Password: •••••••••

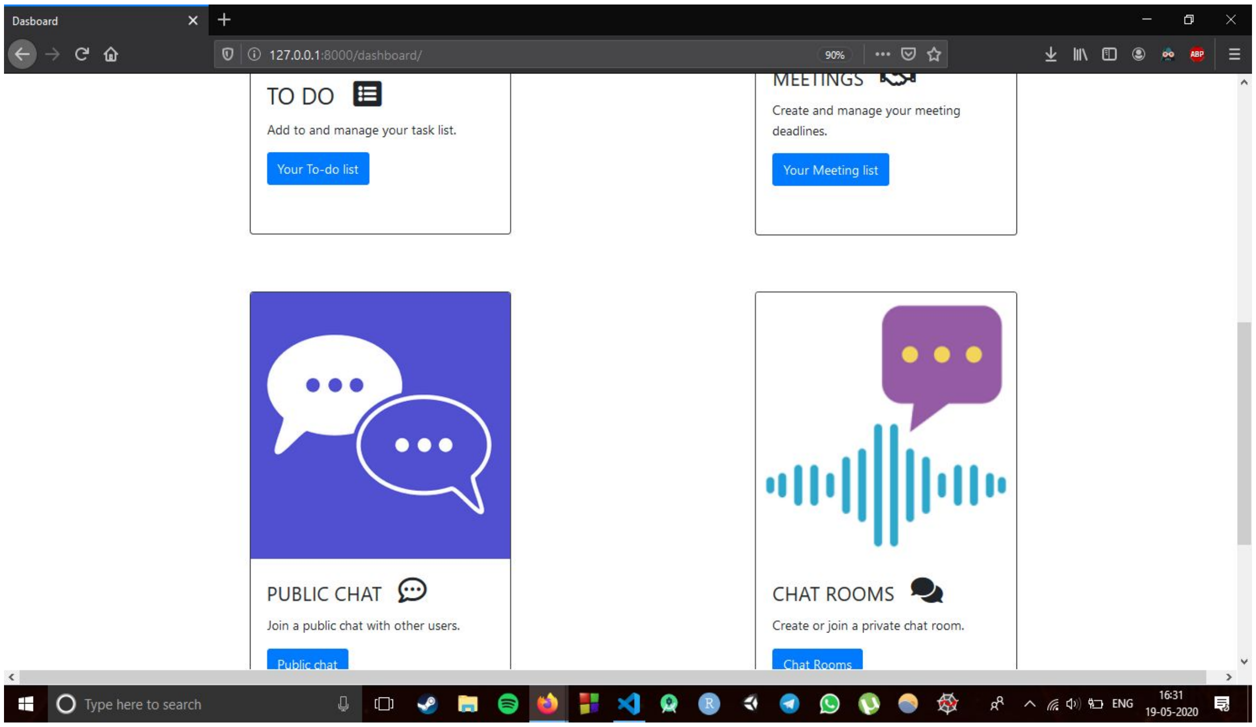
Login

Type here to search ENG 16:30 19-05-2020

DASHBOARD



Welcome, srinivas. Thanks for logging in.



TO DO

Add to and manage your task list.

MEETINGS

Create and manage your meeting deadlines.

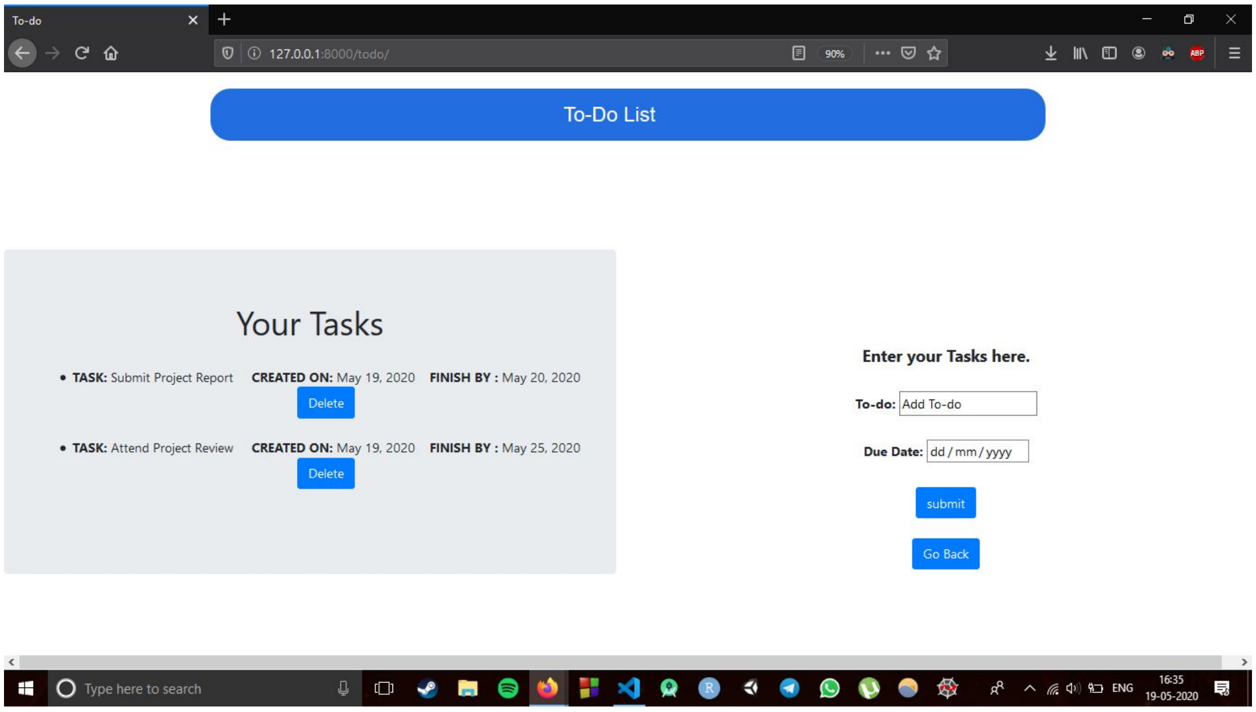
PUBLIC CHAT

Join a public chat with other users.

CHAT ROOMS

Create or join a private chat room.

TO-DO LIST


To-do

127.0.0.1:8000/todo/

To-Do List

Your Tasks

- **TASK:** Submit Project Report **CREATED ON:** May 19, 2020 **FINISH BY :** May 20, 2020 Delete
- **TASK:** Attend Project Review **CREATED ON:** May 19, 2020 **FINISH BY :** May 25, 2020 Delete

Enter your Tasks here.

To-do: Add To-do

Due Date: dd / mm / yyyy

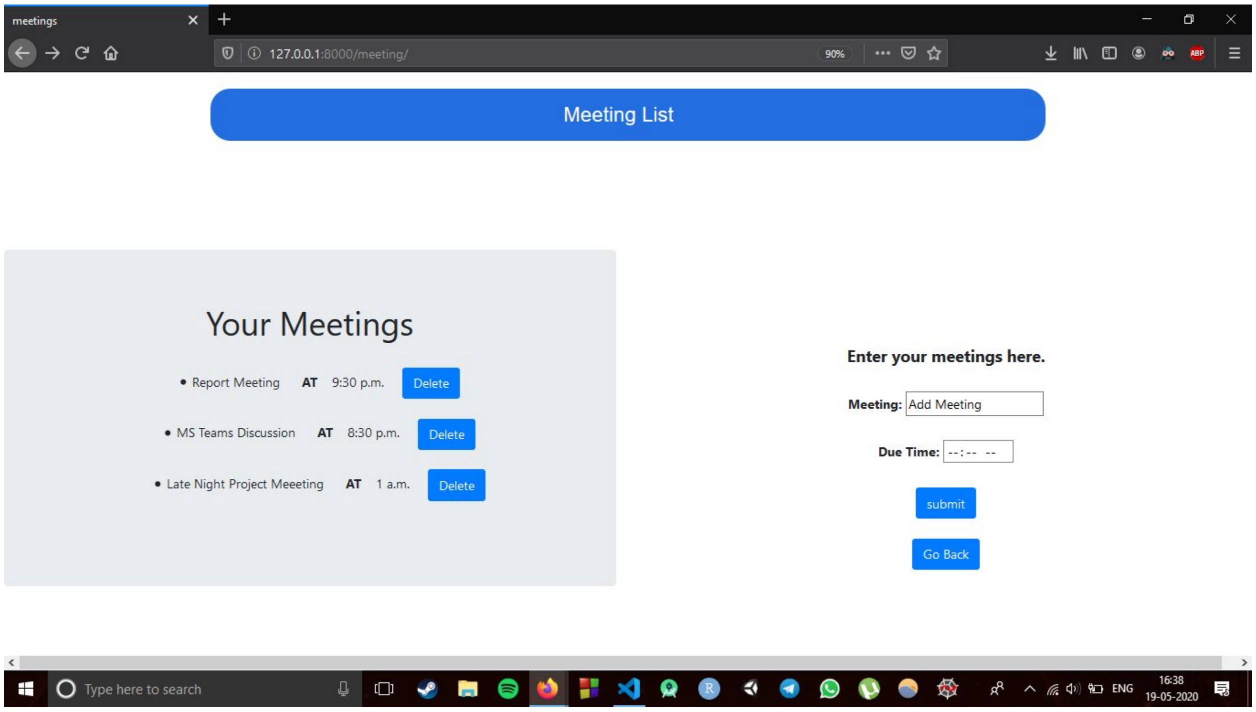
submit

Go Back

Type here to search

16:35 19-05-2020

MEETING LIST


meetings

127.0.0.1:8000/meeting/

Meeting List

Your Meetings

- Report Meeting **AT** 9:30 p.m. Delete
- MS Teams Discussion **AT** 8:30 p.m. Delete
- Late Night Project Meeeting **AT** 1 a.m. Delete

Enter your meetings here.

Meeting: Add Meeting

Due Time: --:-- --

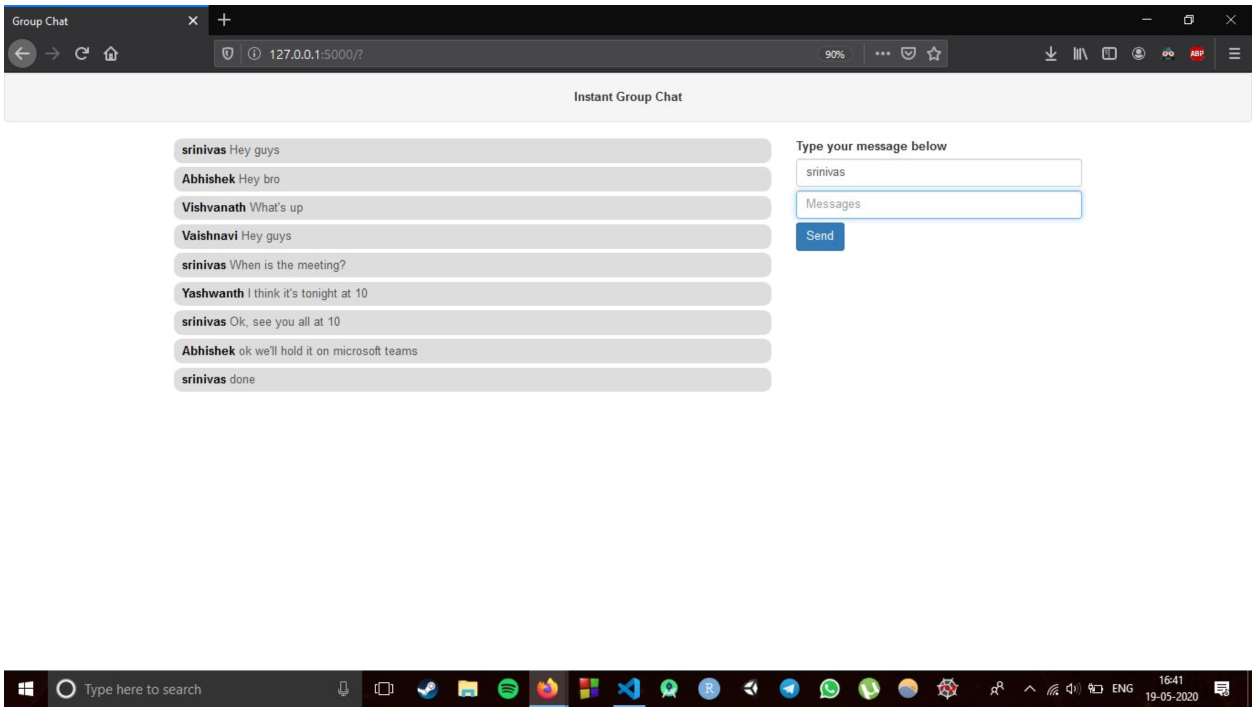
submit

Go Back

Type here to search

16:38 19-05-2020

PUBLIC GROUP CHAT



Instant Group Chat

srinivas Hey guys

Abhishek Hey bro

Vishvanath What's up

Vaishnavi Hey guys

srinivas When is the meeting?

Yashwanth I think it's tonight at 10

srinivas Ok, see you all at 10

Abhishek ok we'll hold it on microsoft teams

srinivas done

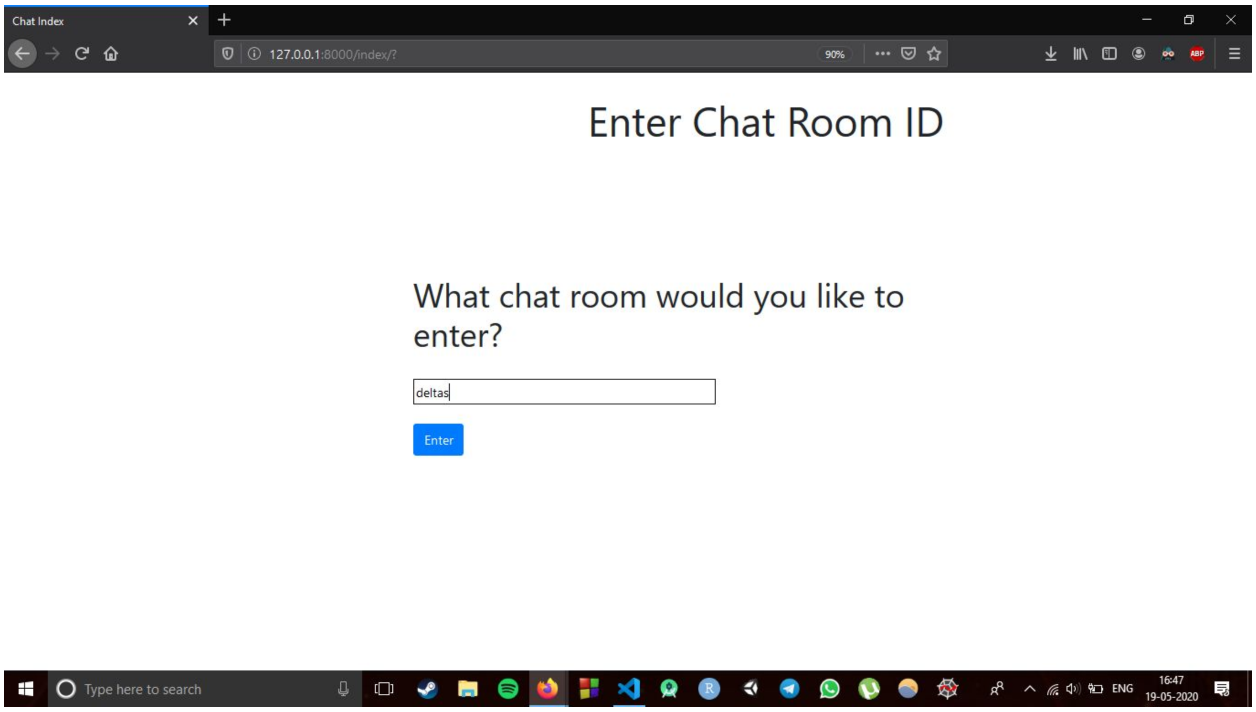
Type your message below

srinivas

Messages

Send

ACCESSING A CHAT ROOM



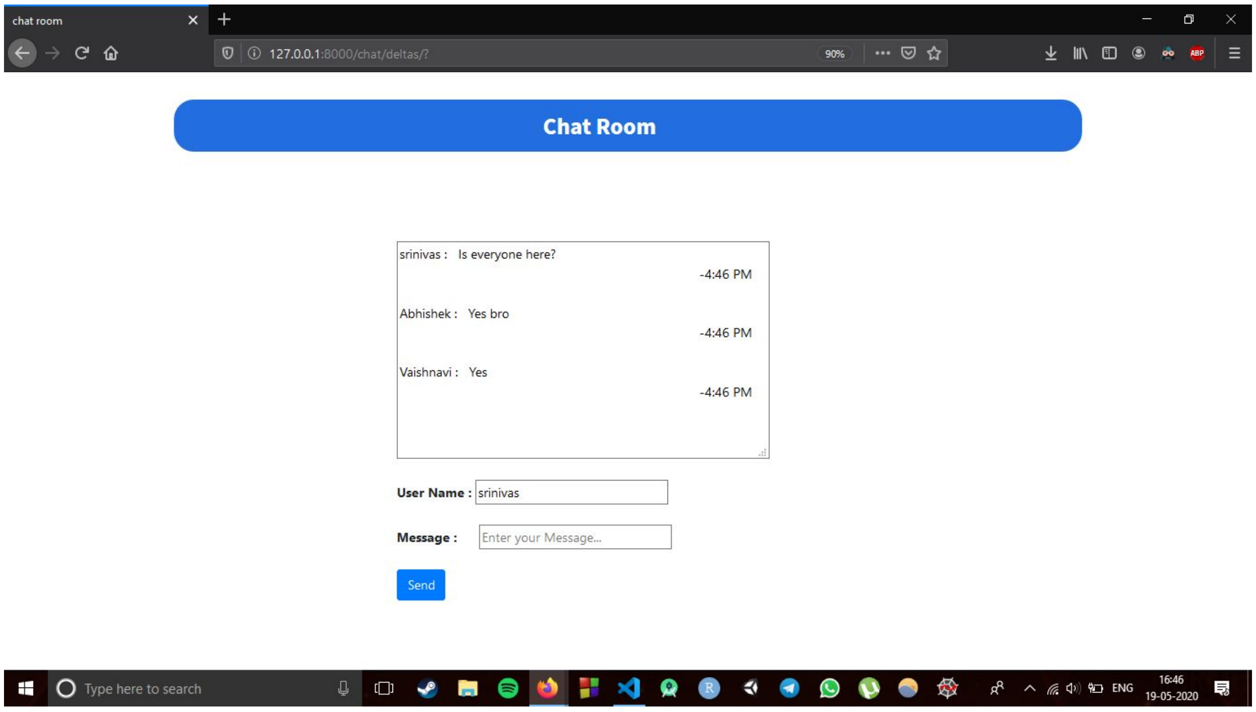
Enter Chat Room ID

What chat room would you like to enter?

deltas

Enter

CHAT ROOM



srinivas : Is everyone here?
-4:46 PM

Abhishek : Yes bro
-4:46 PM

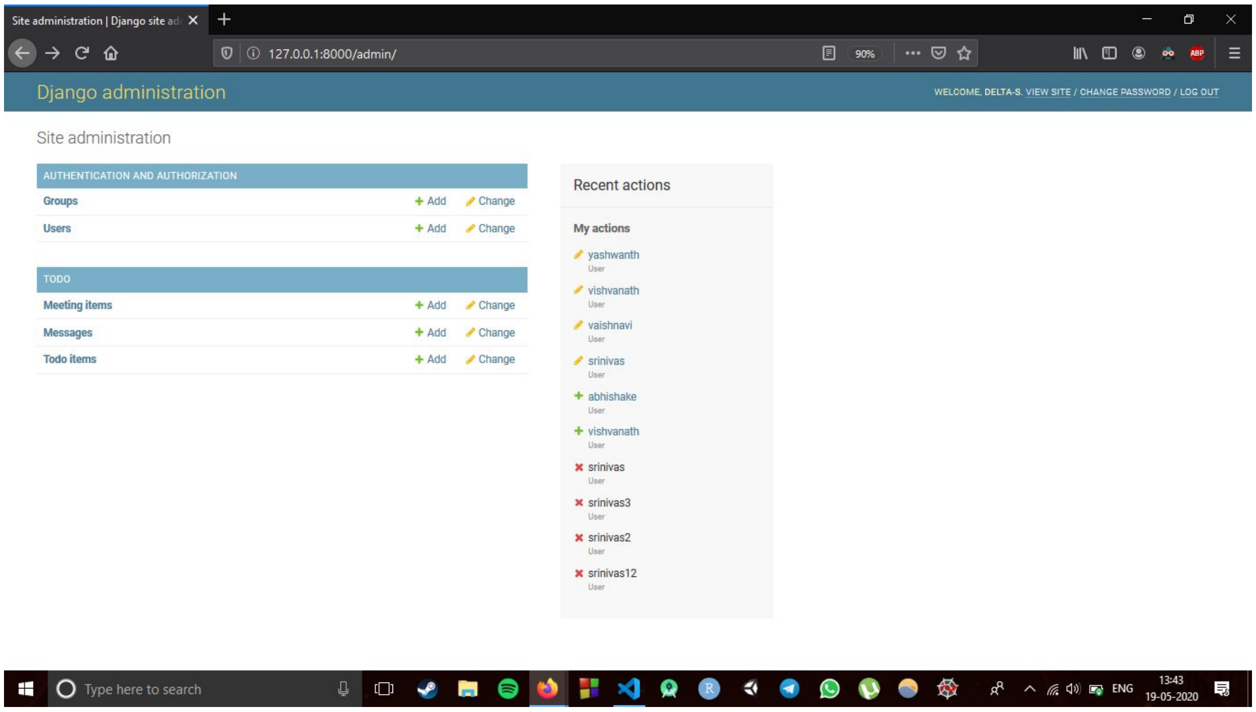
Vaishnavi : Yes
-4:46 PM

User Name : srinivas

Message : Enter your Message...

Send

SITE ADMINISTRATION


Site administration | Django site ad
127.0.0.1:8000/admin/
Django administration
WELCOME, DELTA-S. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	Add	Change
Users	Add	Change

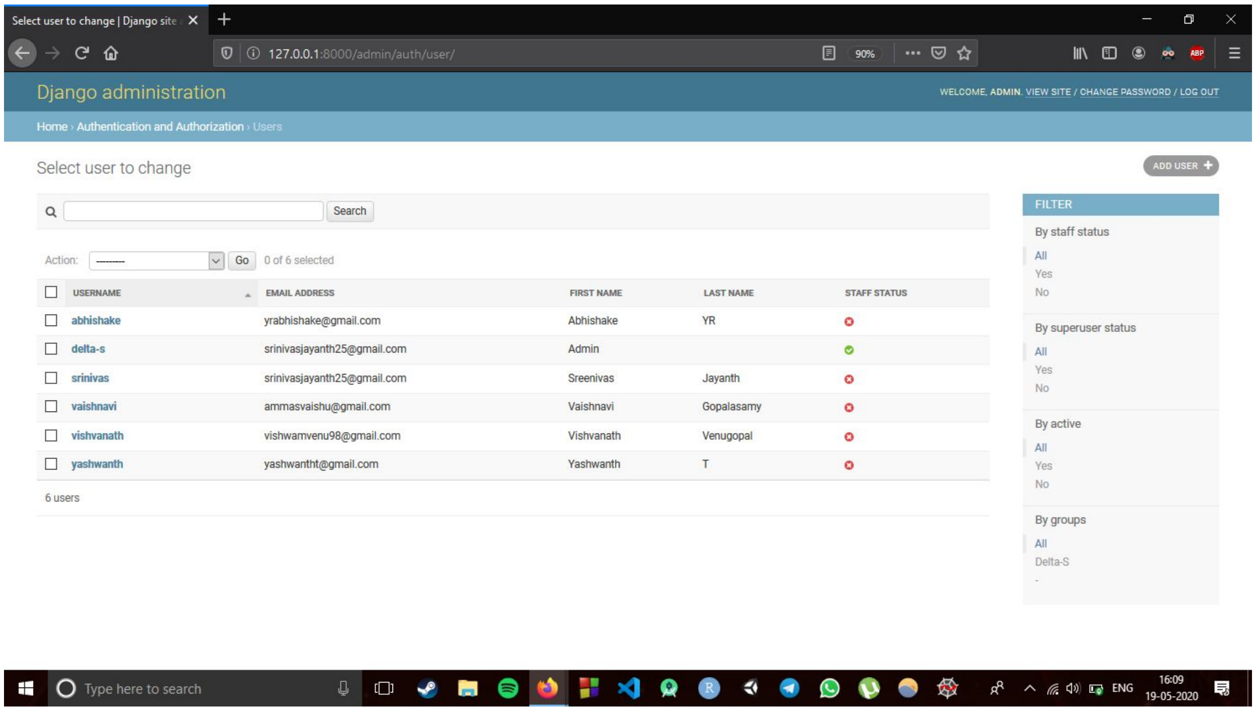
TODO		
Meeting items	Add	Change
Messages	Add	Change
Todo items	Add	Change

Recent actions

My actions

- yashwanth User
- vishvanath User
- vaishnavi User
- srinivas User
- abhishake User
- vishvanath User
- srinivas User
- srinivas3 User
- srinivas2 User
- srinivas12 User

USER SELECTION


Select user to change | Django site

127.0.0.1:8000/admin/auth/user/

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Users

Select user to change

ADD USER

Action: --------- Go 0 of 6 selected

	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
☐	abhishake	yrabhishake@gmail.com	Abhishake	YR	✗
☐	delta-s	srinivasjayanth25@gmail.com	Admin		✓
☐	srinivas	srinivasjayanth25@gmail.com	Sreenivas	Jayanth	✗
☐	vaishnavi	ammasvaishu@gmail.com	Vaishnavi	Gopalasamy	✗
☐	vishvanath	vishwamvenu98@gmail.com	Vishvanath	Venugopal	✗
☐	yashwanth	yashwantht@gmail.com	Yashwanth	T	✗

6 users

FILTER

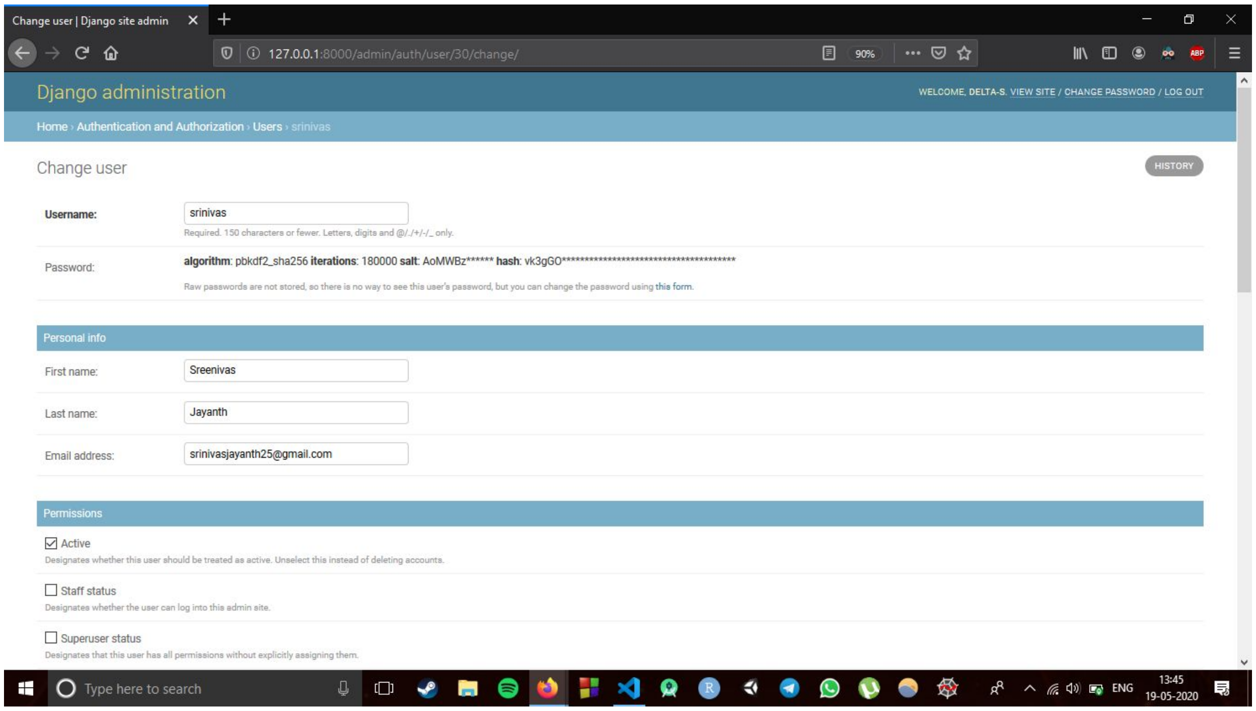
By staff status: All, Yes, No

By superuser status: All, Yes, No

By active: All, Yes, No

By groups: All, Delta-S, -

MODIFYING USER DETAILS


Change user | Django site admin

127.0.0.1:8000/admin/auth/user/30/change/

Django administration

WELCOME, DELTA-S. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Users › srinivas

Change user

HISTORY

Username: srinivas

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm**: pbkdf2_sha256 **iterations**: 180000 **salt**: AoMWBz****** **hash**: vk3gGO**************************************

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name: Sreenivas

Last name: Jayanth

Email address: srinivasjayanth25@gmail.com

Permissions

☑ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

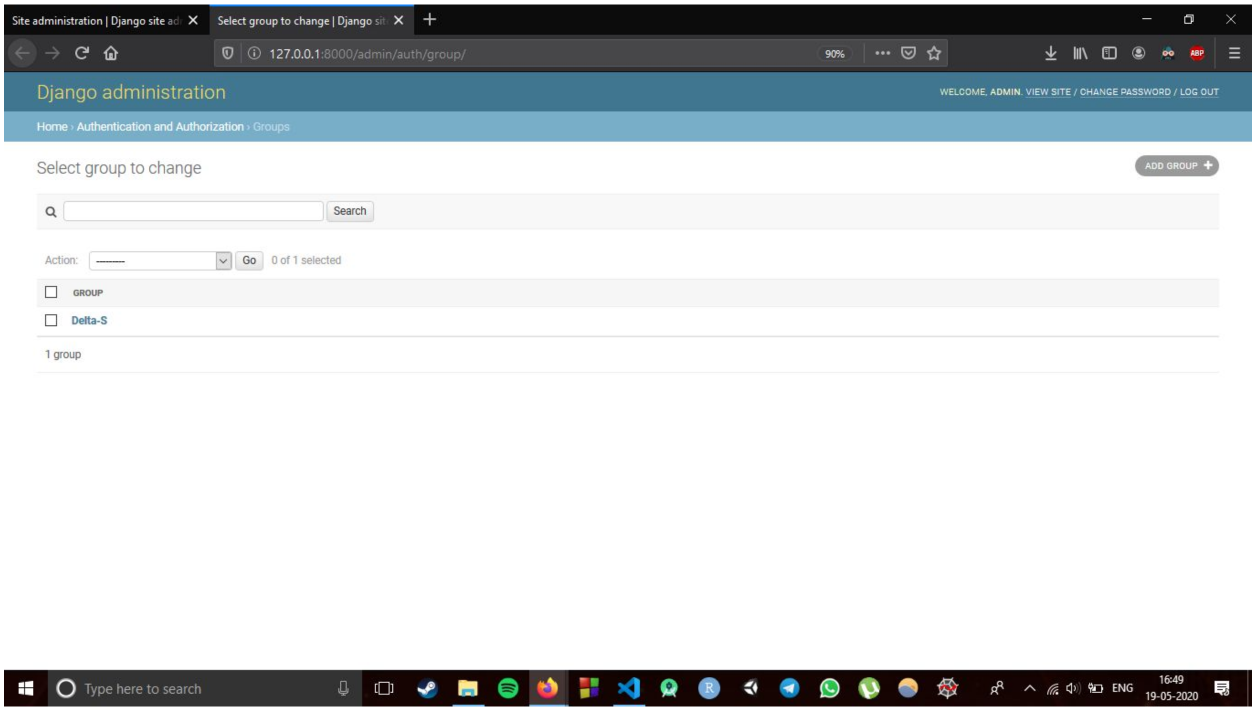
☐ Staff status

Designates whether the user can log into this admin site.

☐ Superuser status

Designates that this user has all permissions without explicitly assigning them.

GROUP SELECTION


Site administration | Django site adm | Select group to change | Django site

127.0.0.1:8000/admin/auth/group/

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Groups

Select group to change

ADD GROUP

Search

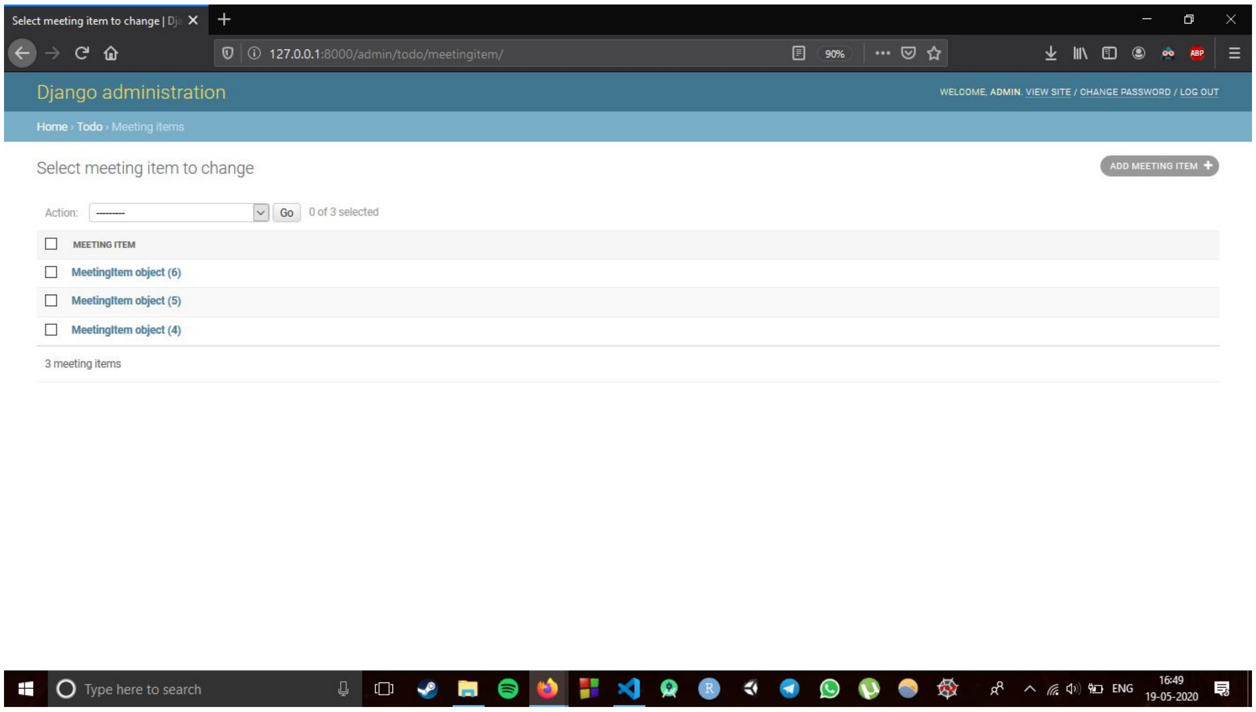
Action: --------- Go 0 of 1 selected

GROUP

Delta-S

1 group

MEETINGS DATABASE


Select meeting item to change | Dja

127.0.0.1:8000/admin/todo/meetingitem/

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Todo › Meeting items

Select meeting item to change

ADD MEETING ITEM

Action: --------- Go 0 of 3 selected

MEETING ITEM

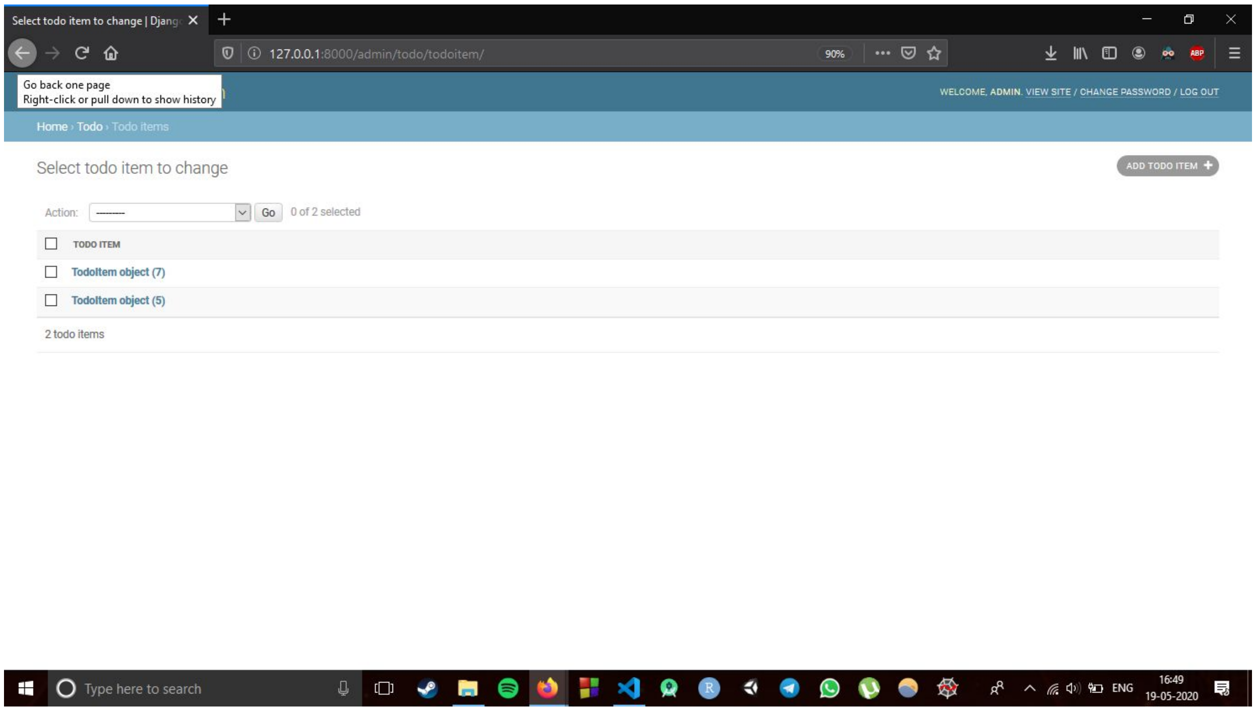
MeetingItem object (6)

MeetingItem object (5)

MeetingItem object (4)

3 meeting items

TO-DO LIST DATABASE



Select todo item to change | Django

127.0.0.1:8000/admin/todo/todoitem/

Go back one page
Right-click or pull down to show history

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Todo › Todo items

Select todo item to change

ADD TODO ITEM

Action: --------- Go 0 of 2 selected

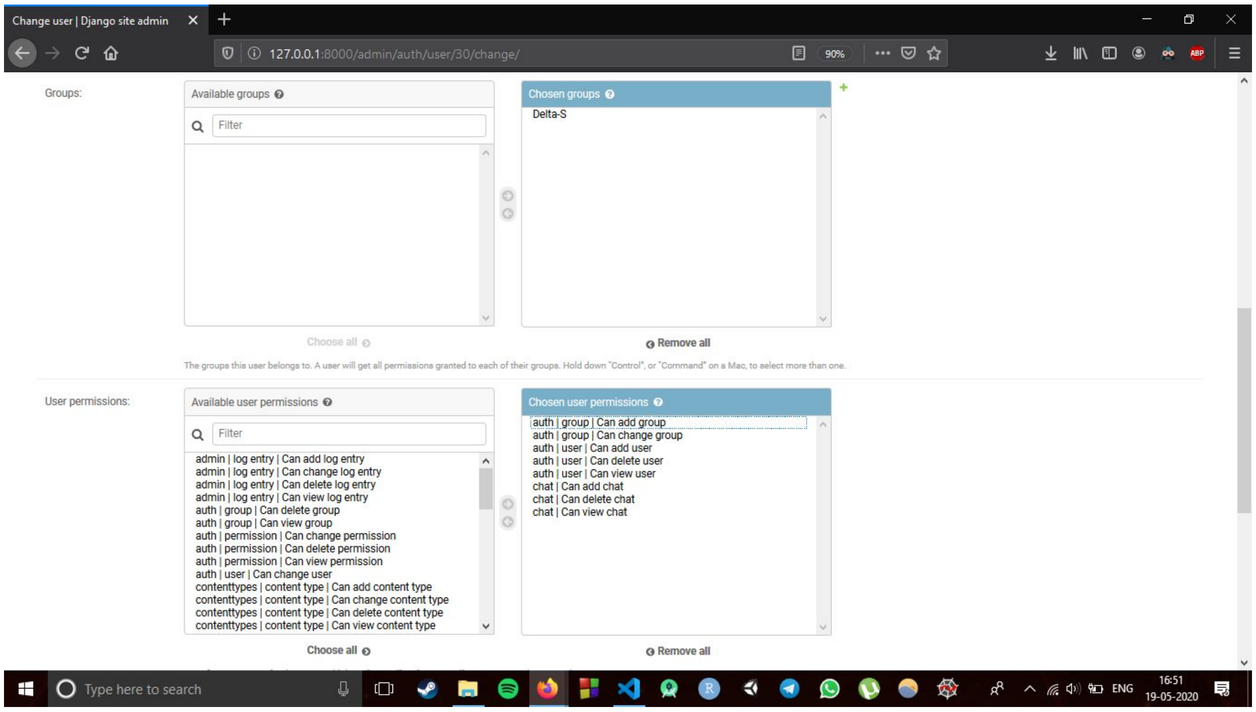
TODO ITEM
TodoItem object (7)
TodoItem object (5)

2 todo items

Type here to search

16:49
19-05-2020

GROUP MANAGEMENT & USER PERMISSIONS



Change user | Django site admin

127.0.0.1:8000/admin/auth/user/30/change/

Groups:

Available groups

Filter

Choose all

Chosen groups

Delta-S

Remove all

The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

Available user permissions

Filter

```
admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry
auth | group | Can delete group
auth | group | Can view group
auth | permission | Can change permission
auth | permission | Can delete permission
auth | permission | Can view permission
auth | user | Can change user
contenttypes | content type | Can add content type
contenttypes | content type | Can change content type
contenttypes | content type | Can delete content type
contenttypes | content type | Can view content type
```

Choose all

Chosen user permissions

```
auth | group | Can add group
auth | group | Can change group
auth | user | Can add user
auth | user | Can delete user
auth | user | Can view user
chat | Can add chat
chat | Can delete chat
chat | Can view chat
```

Remove all

Type here to search

16:51
19-05-2020

TESTING PLAN

As each of our Web-Application's features is of a separate module, and all of these achieve individual goals, we decided to follow a **unit testing** methodology, individually testing all the existing units.
As per the guidelines presented on the internet, this is what we followed :

1. **Analyzing the product**

   Listing the features of the product, and finding out how each feature works in different use cases.
2. **Designing the Test Strategy**

   The following steps were followed :
   1. **Defining scope of the testing:**

      In this phase we decided the **scope** of the testing.

      In Scope:
      - UI elements
      - Chat App testing
      - Entering incomplete data into entry fields (Todo, meeting lists, chat apps)

      Out of Scope:
      - Server side app hosting problems.
      - Unresponsive redis server (Used for chat rooms feature)
   2. **Identifying Testing Type:**
      - We decided to go with the **Unit Testing** method, as we have independent modules for different purposes, including different hosts for the chat apps.

   3. **Risks & Mitigation:**

Risk	**Mitigation**
The app returns an error when one of the entries is null in the To-do list or the meeting list features.	Make it mandatory to enter data in all entry fields.
The app looks different on different browsers, and some elements may not be displayed in some browsers.	Fine tune the UI manually so it is optimised for all the browsers.

The previous messages on the chat apps may not be displayed if the user isn't already connected to the chat server.	Make sure that every member is connected to the server before any messages are sent
The requirements for the app isn't met by one of the team members	Delegate it to the scrum master to work on, as he has the most recent version of the app, and can work on it.

4. **Test Logistics:**

   We **manually tested** the elements whenever we updated the UI of the app, or the backend, like how it looks on different browsers, and **manually fine-tuned** it so it looks its best on all the browsers. Backend wise, we had to log in to the admin page and refresh it every time a database was updated, to ensure that data entry was smooth, and that the frontend and backend were properly linked. Also made sure the code was error-free.

3. **Defining Test Objectives**

   In this phase we listed down all the software's features which may need to be tested, and defined the target of the tests based on the features.

Features that need testing:

- **Performance:** Test the performance for multiple users on the chat server.
- **UI:** Test the working of the UI on different browsers and fine tuning it so it's optimised for all the browsers.
- **All Units :** using print statements & console logs as a way of ensuring that everything was working as expected and that data entry was smooth and error free.
- **Chat App:** Check the load that the server is able to handle for the chat.

4. **Defining Test Criteria**

As we simultaneously worked on the app and tested the units of the app, we employed **Suspension Criteria.** We first built the backend and basic front end of the unit, then tested for errors by identifying and implementing all possible use cases, and individually rectifying each one of them. Once errors were checked, we moved ahead with improving the UI and other things.

5. **Resource Planning**

As everyone was working on the project, he/she was responsible for error checking their own features. So everyone tested their features with print statements and console logs. Front end wise, the rendered templates themselves were enough to check for UI discrepancies, so editing the html, CSS and jQuery script sufficed.

Regarding the System Resources, **we used our own editors and browser consoles, as testing was manual**. Everyone worked in their own network and individual machines and operating systems (Windows, Ubuntu, etc).

6. **Plan Test Environment**

We **manually tested** individual units, so we used our own editors and browser consoles for checking console logs and terminal outputs, under different use cases, like incomplete data entry, and chat message display cases.

7. **Schedule and estimation**

The Tests were carried out as the app was being developed, and once the print statements and console logs worked as expected, we moved on with making the UI for the features.

8. **Determining the Test deliverables**

Test deliverables **before** the testing phase.
- Working model from sprint 1.

Test deliverables **during** the testing phase.
- None.

Test deliverables **after** the testing cycle is over.
- Testing Plan Document (This Report).

PROJECT CODE

CODE

Github : https://github.com/CodeManJay/todoappproto

[INTENTIONALLY LEFT BLANK]

[SCROLL DOWN TO NEXT PAGE FOR OUR EXPERIENCE REPORT]

OUR EXPERIENCE

Team Member	**Takeaway as a Team**	**Takeaway as an Individual**
Sreenivas Jayanth Yadhati	1. Learned how to work long distance in major projects. 2. Learned how to manage the Team accordingly as the Scrum Master, by assigning tasks based on the member's skill-set and capability, and smoothening the workflow.	1. <u>Technical:</u> a. Django b. Django channels for chat server implementation c. Flask IO-sockets d. HTML & jQuery (necessary for the chat apps) 2. <u>Non technical</u> : a. How to diplomatically work on real-world-projects, with proper planning and visualization before implementation. b. Time Management and scheduling.
Yashwanth T.	1. Learned how to work long distance in major projects	1. <u>Technical:</u> a. HTML b. jQuery (necessary for chat apps) 2. <u>Non technical</u> : a. How to diplomatically work on real-world-projects, with proper planning and visualization before implementation. b. Time Management and scheduling.

Vaishnavi Gopalasamy	1. Learned how to work long distance in major projects. 2. Understood how to split tasks and collaborate on work.	1. <u>Technical:</u> a. Django b. Django channels for chat server implementation c. Flask IO-sockets d. jQuery (necessary for the chat apps) 2. <u>Non technical</u> : a. How to diplomatically work on real-world-projects, with proper planning and visualization before implementation. b. Time Management and scheduling.
Vishvanath V.	1. Learned how to work long distance in major projects.	1. <u>Technical:</u> a. jQuery (necessary for the chat apps) b. HTML 2. <u>Non technical</u> : a. How to diplomatically work on real-world-projects, with proper planning and visualization before implementation. b. Time Management and scheduling.
Abhishake Y.R.	1. Learned how to work long distance in major projects.	1. <u>Technical:</u> a. HTML & CSS with Bootstrap b. jQuery (necessary for the chat apps) 2. <u>Non technical</u> : a. How to diplomatically work on real-world-projects, with proper planning and visualization before implementation. b. Time Management and scheduling.

x==== THANK YOU====x

-TEAM DELTA S