Internship Assignment Report: Chat Room using Django Rest Framework

1. Introduction

The aim of this project was to develop a real-time chat application that allows users to send messages instantly to a group, supporting features such as text-based communication, user authentication, and real-time updates. User can register or login on the app after which he is allowed to create or join any group.

2. Approach and Implementation

2.1 Frontend

- Technology Stack: React.js was chosen for the frontend development along with redux and useContext for some simple state management and cloudinary storage for image storage.
- User Interface: A user-friendly interface was designed to facilitate messaging allowing users to toggle the theme into light and dark. It is also mobile responsive and has a manifest.js file which allows it to be installed from browser into local device.
- Libraries Used: Employed notistack for error handling and Chakra UI and Material UI for UI components.

How to use:

Clone the repository locally.

Run npm install to install all the packages.

Run npm start to start the application on port 3000.

2.2 Backend

- Technology Stack: Django REST Framework (DRF) was employed for building the backend API.
- API Endpoints: All the authentication and messaging is persisted in database and has an API for backend connection. Used MySQL for database and created custom user, chat and message models and serializers. APIs are protected by custom auth middleware.
- Web Sockets: Used Django channels with Redis and Daphne for real time messaging and notifications
- Integration: Integrated django-cors-headers to handle Cross-Origin Resource Sharing (CORS) issues.

How to start the server:

Clone the repository locally.

Create a virtual environment by using pipenv. Run the following commands:

```
pip install pipenv
pipenv shell
```

Install the dependencies:

```
pip install -r requirements.txt
```

Create .env file.

Add your MySQL database details in the .env file.

For running the channels you will need redis support. Follow these steps:

Install linux subsystem for windows. You can install ubuntu latest from Microsoft Store.

Open the ubuntu terminal and run these commands one by one:

```
sudo apt-get update
sudo apt-get install redis
sudo service redis-server start
```

Check if the redis is running using these codes:

```
redis-cli
ping
```

If the output is pong, redis is up and running.

Start migrations:

```
python manage.py makemigrations
python manage.py migrate
```

Start server:

```
python manage.py runserver
```

3. Insights Gained

Reflecting on the project experience, insights were gained into the complexities of django channels. The experience and challenges were new and different from node.js and socket.io.

4. Future Enhancements

- Identified potential areas for improvement such as performance optimization of frontend using image compression for both profile photos and static assets.
- More features such as one to one chat can be added for better experience. Users can make friends and chat with them.
- File sharing using cloudinary can also be implemented by tweaking the message model.

**Frontend is deployed on this url: Frontend

**Due to very less availability of free MySQL cloud service supporting latest versions, backend was not deployed.