مسابقه بک اند

مقدمه:
میخواهیم یک شبکه از کاربران مختلف داشته باشیم که بتوانند با هم چت کنند.

موجودیت ها: (برای فهم بهتر، لزوما هر کدام یک entity در نرم افزار نخواهند بود)

- **کاربر (User):** همانند هر شبکه دیگری این شبکه نیز از مجموعه ای از کاربران تشکیل شده است.
- **گروه (Group):** اگر هر کاربر را به عنوان یک node در نظر بگیریم. گروه شامل تعدادی از node ها میباشد . (متشکل از یک سری user ها). نکته ای که وجود دارد هر کاربر تنها میتواند در یک گروه عضو باشد پس گروه ها نمیتوانند با هم اشتراک داشته باشند.
- چت باکس (**Chat Box**): هر ۲ یوزری اگر واجد شرایط باشند (که در ادامه به آن ها اشاره خواهد شد) میتوانند با یکدیگر چت کنند.

نکات:
- هر کاربر تنها میتواند در یک گروه عضو باشد. پس گروه ها هیچ اشتراکی ندارند.هم چنین مشخص است که اگر تمامی کاربران عضو گروهی باشند آنگاه گروه ها کاربران را افراز کرده اند.
- هر شخص میتواند به یک گروه وارد شود یا گروه خود را ایجاد کند.
- شخصی که گروهی را ایجاد کند به عنوان ادمین آن گروه در نظر گرفته میشود.
- دو نقش در گروه میتواند وجود داشته باشد.
    a. ادمین: سازنده گروه
    b. کاربر عادی: کسی که سازنده گروه نیست.
- دو گروه میتوانند با یک دیگر کانکت شوند. بدین صورت که ادمین یک گروه به گروه دیگر درخواست کانکشن بفرستد و در گروه دیگر ادمین این درخواست را قبول کند.
- افراد گروه های مختلف در صورتی که گروه هایشان با یکدیگر کانکت باشد میتوانند با یکدیگر چت خصوصی داشته باشند. (چت دو نفره)
- اضافه شدن یک شخص به یک گروه به این صورت خواهد بود که ابتدا شخصی که در گروهی عضو نیست درخواست اضافه شدن به گروه را میفرستد و ادمین میتواند قبول کند.

حال میخواهیم *api* های زیر را برای این پروژه پیاده سازی کنیم..

APIs

نکته مهم: لطفا تمامی api های خود را با توجه به swagger طراحی کنید. یعنی آدرس route، ساختار ریسپانس خروجی، http status code. خطاها و سایز جزییات. حتی یک فیلد اضافه یا کم میتواند باعث شود شما نمره را از تست مربوطه از دست بدهید. همچنین انتظار داشته باشید که فرمت دیتا های ورودی به شکلی باشد که در swagger نشان داده شده است. همچنین برای سادگی کار اگر در یک درخواست مشکلی وجود داشت و به دلیل های مختلف (مانند نداشتن دسترسی کافی، یا اینکه کاربر در گروه دیگری عضو است یا پسورد در لاگین اشتباه است یا آیدی موجود نیست یا ...) فرآیند به مشکل خورد، خطای 400 باید داده شود. (فرمت این خطا هم در swagger مشخص شده است.)

نحوه احراز هویت: بجز دو route ثبت نام و لاگین بقیه روت ها محافظت شده هستند (route هایی که روی آن ها علامت قفل است) و درخواست ها با توکن jwt شناخته خواهند شد و این توکن هم در header با کلید Authorization ست خواهد شد. قابل ذکر است که حتما در payload توکن ادرس ایمیل و آیدی شخص را با کلید های (email, userId) ذخیره کنید.

نکته: متناظر با تمامی route هایی که در زیر به آن ها اشاره شده است، در swagger داکیومنت آن ها آمده است.
نکته:در تمامی route هایی که زمان به عنوان خروجی آمده است فرمت آن timestamp در نظر بگیرید.

- **ثبت نام**

کاربر با این api می تواند ثبت نام کند.

(POST `/api/v1/auth/signup/`)

- **لاگین**

کاربر با این api می تواند لاگین کند.

(POST `/api/v1/auth/login/`)

- **ایجاد گروه**

کاربر با این api میتواند گروه خود را ایجاد کند

(POST `/api/v1/groups/`)

نکته اگر کاربر در گروهی عضو باشد باید خطای 400 دریافت کند.

- **دیدن گروه ها**

کاربر با این api می تواند لیست تمامی گروه ها را از جدید به قدیم مشاهده کند

(GET `/api/v1/groups/`)

- **دیدن اطلاعات گروهی که شخص در آن عضو است**

کاربر با این api می تواند اطلاعات گروهی که در آن عضو است را ببیند.

(GET `/api/v1/groups/my/`)

نکته: دقت کنید که اعضا از جدید ترین به قدیمی ترین باید نمایش داده شوند. (تاریخ وارد شدن به گروه)
نکته: اگر کاربری در گروهی عضو نبود باید خطای 400 دریافت کند.
نکته:کاربری که owner نیست باید نقشش normal در نظر گرفته شود.

- **ارسال درخواست عضو شدن**

کاربر با این api میتواند به گروهی درخواست عضو شدن ارسال کند.
(POST `/api/v1/join_requests/`)
نکته: اگر کاربر در گروهی عضو بود باید خطای 400 بگیرد

- **دیدن درخواست های عضویت خود**

کاربر با این api میتواند تمامی درخواست های عضویت خود را که به سایر گروه ها ارسال کرده است را مشاهده کند.
(GET `/api/v1/join_requests/`)
نکته: جدیدترین به قدیمی ترین.

- **دیدن درخواست های داده شده به گروه**

کاربر میتواند با این api درخواست های داده شده به گروهش را مشاهده کند.
(GET `/api/v1/join_requests/group`)

نکته: جدید ترین به قدیمی ترین.
نکته: اگر کاربر صاحب گروهی نبود باید خطای 400 دریافت کند.

- **قبول درخواست داده شده به گروه برای عضویت**

کاربر میتواند با این api درخواست عضویت در گروهش توسط شخص دیگری را قبول کند .
(POST `/api/v1/join_requests/accept/`)

نکته: اگر کاربر صاحب گروهی نبود باید خطای 400 دریافت کند.

- **فرستادن درخواست اتصال به گروهی دیگر**

کاربر می تواند با این api به گروه دیگری درخواست اتصال بدهد

(POST `/api/v1/connection_requests/`)

نکته: در صورتی که کاربر صاحب گروه نباشد باید خطای 400 دریافت کند

- **دریافت لیست درخواست های اتصال به گروه خود**

کاربر می تواند با این api درخواست های اتصال به گروه خود را مشاهده کند

(GET `/api/v1/connection_requests/`)

نکته: در صورتی که کاربر صاحب گروه نباشد باید خطای 400 دریافت کند
نکته: جدیدترین به قدیمی ترین
نکته: لیست درخواست های دریافت شده به گروه خود در این api قابل مشاهده است.

- **قبول درخواست اتصال گروه دیگر به گروه خود**

کاربر می تواند با این api درخواست اتصال به گروه خود را از جانب گروه دیگر را بپذیرد.

(POST `/api/v1/connection_requests/accept`)

نکته: در صورتی که کاربر صاحب گروه نباشد باید خطای 400 دریافت کن

- **چت کردن با شخصی دیگر**

کاربر می تواند با این api به شخص دیگری پیام دهد.

(POST `/api/v1/chats/{user_id}`)

نکته: در صورتی که دو کاربر در گروه هایی عضو باشند که آن دو گروه با یکدیگر connect نباشد آنگاه باید خطای 400 داده شود

- **دریافت چت های موجود با شخصی دیگر**

کاربر می تواند با این api چت های خود را با کاربری دیگر مشاهده کند.

(GET `/api/v1/chats/{user_id}`)

نکته:از جدیدترین به قدیمی ترین
نکته: اگر با آن شخص تا به حال پیامی نداشت یا آن شخص موجود نبود خطای 400 داده شود.

- **دریافت لیست چت های خود با دیگران**

مسابقه بک اند

کاربر می تواند با این api چت های خود را با کاربری دیگر مشاهده کند.

(GET `/api/v1/chats`)

نکته:از جدیدترین به قدیمی ترین (آخرین پیام دریافتی یا ارسالی ملاک است)

نکته: اگر با آن شخص تا به حال هیچ کس چتی نداشته است. همان فرمت آرایه خالی برگردانده شود.