پروژه چت بات بوتکمپ ترب

امیرعرفان تیموری - پاییز ۱۴۰۲

طرح پروژه:

- توسعه یک سایت که در آن کاربرها میتوانند برای دیتای مورد نظر خودشون chatBot بسازن. به این ترتیب که ابتدا دیتای مورد نظرشون رو در نرم‌افزار وارد میکنن (knowledge base) و بعد میتونن چت باتی بسازن که با توجه به دیتای وارد شده در مرحله قبل به سوالات کاربرها جواب بده، این سایت صفحات و قابلیت های مختلفی داره.

- **شما میتونین از طریق این لینک سایت دسترسی داشته باشید.**

طرح ذهنی اولیه پروژه:

- برای حل این مسئله، ابتدا باید استراکچر اپلیکیشن خودمون رو با توجه به نیازمندی های پروژه پیاده سازی کنیم.
  این کار شامل:
  - مشخص کردن طرح اولیه صفحات ابتدایی و محتوای هر صفحه
  - مشخص کردن مسیر طی کردن پیج ها توسط کاربر (نقطه شروع، انتخاب ها، بازگشت)
  - مشخص کردن نسخه اولیه مدل های دیتابیس
  - مشخص کردن اپلیکیشن های مورد نیاز و کاربرد وظایف آنها

من برای این پروژه، فقط یک اپ درست کردم که اون هم، chatbot هست.
دلایل؟

- اندازه پروژه آنچنان بزرگ نیست
- ساختن چند اپ دیگر (مثل یوزر) باعث پراکندگی بی دلیل کد ها میشه.
- من رویکردم اینجوری بود که با یک اپ شروع کنم کد نویسی و جایی که حس کردم نیازه، بشکونم اون رو به اپ های کوچکتر، که در این پروژه به همچین نقطه ای نرسیدم. اما اگر بخواد پروژه گسترش پیدا کنه، قطعا نیازه که اپ های بیشتری اضافه بشن
- اضافه کردن اپ دیگه پیچیدگی کد رو بالا میبره، که اینجا بنظرم بی دلیل و trade off خوبی نیست

معرفی دیتابیس و مدل های آن:

- **CustomUser:**
  - این مدل از همون مدل دیفالت استفاده میکنه
    هدف گذاشتنش چی بوده؟
    - اگر فیلد اضافه کردن نیاز بود، انجام آن راحت خواهد بود و نیازی به تغییرات در settings.py نیست.
- **Chatbot:**
  - یک کلید خارجی برای سازنده خود دارد
  - اسم و توضیح دارد که مشخص است.
  - یک پرامپ دلخواه دارد که میتواند کاربر سازنده آن را تغییر دهد. این به عنوان سیستم پرامپ پاس داده میشود.
  - یک فیلد فعال بودن دارد، که اگر ربات فعال نباشد، آن را به کاربر نشان ندهیم.
  - یک فیلد برای عکس دارد.
  - یک تاریخ ساخت دارد.
- **ChatbotContent:**
  - یک کلید خارجی به چت بات مربوط به خود دارد.

- یک فیلد محتوا دارد، که محتوا بصورت متن در آن ذخیره میشود و کاربر می تواند آن را تغییر دهد.
- فیلد نهایی مربوط به امبدینگ است که با توجه به محتوا بصورت خودکار حساب میشود و برای محاسبه شباهت حساب میشود.

- **Conversation:**

- یک کلید خارجی به دو طرف صحبت دارد، یعنی یوزر و چت بات مربوط به گفتگو.
- تاریخ شروع مکالمه
- تاریخ آخرین پیام رد و بدل شده برای نمایش در تاریخچه چت
- عنوان مکالمه که بصورت خودکار محاسبه میشود.

- **Message:**

- کلید خارجی به گفتگویی که این پیام بهش مربوطه.
- محتوای پیام.
- بردار سرچ برای پیاده سازی full text search (میاد حروف اضافه و غیر مهم رو حذف میکنه و برای کلمه های اصلی این متن، میاد یک وزن در نظر میگیره و اونارو ذخیره میکنه، این وزن هارو با توجه به ویژگی های مختلف میسازه، مثل اینکه هرچی یک کلمه کم تکرار تر باشه، وزن بیشتری میگیره)
- نقش پیام:
  - اینکه این پیام رو چت بات ساخته
  - کاربر اونو فرستاده
  - یا اینکه کانتکست گرفته شده است و از جنس کانتکست عه
  - راجع به این مورد اگر بخوام بیشتر توضیح بدم، هر بات یک مجموعه کانتکست میتونه داشته باشه (Chatbot Content) که هربار کاربر پیام میفرسته، میاد یکی ازون کانتکست های مرتبط رو بر می گردونه و خب نقشش رو همین میذاره.
- تعداد لایک ها و دیسلایک های یک پیام

- **محتوای اصلی:**

- وقتی کاربر دیسلایک میکنه که پیام دوباره جنریت شه، پیام جدید میره توی فیلد محتوا قرار میگیره و محتوای فعلی، میره توی این فیلد قرار میگیره. حالا اگر باز هم دیسلایک داد کاربر، همین روند تکرار میشه، یعنی پیام دفعه اول از بین میره و ورژن ۲ و ۳ باقی میمونن.

حالا چرا این فیلد رو قرار دادم؟

- چون من یک فیچر به پروژه اضافه کردم، اونم جابجایی بین پیام اولیه و دوباره جنریت شده است. احیانا اگر کاربر درخواست تولید دوباره پیام رو داد ولی پیام اول بهتر بود، اینجوری میتونه به ورژن قبلی برگرده!

- فیلد آخر هم مشخص میکنه که الان پیام اصلی رو نشون بده یا پیام دوباره جنریت شده.

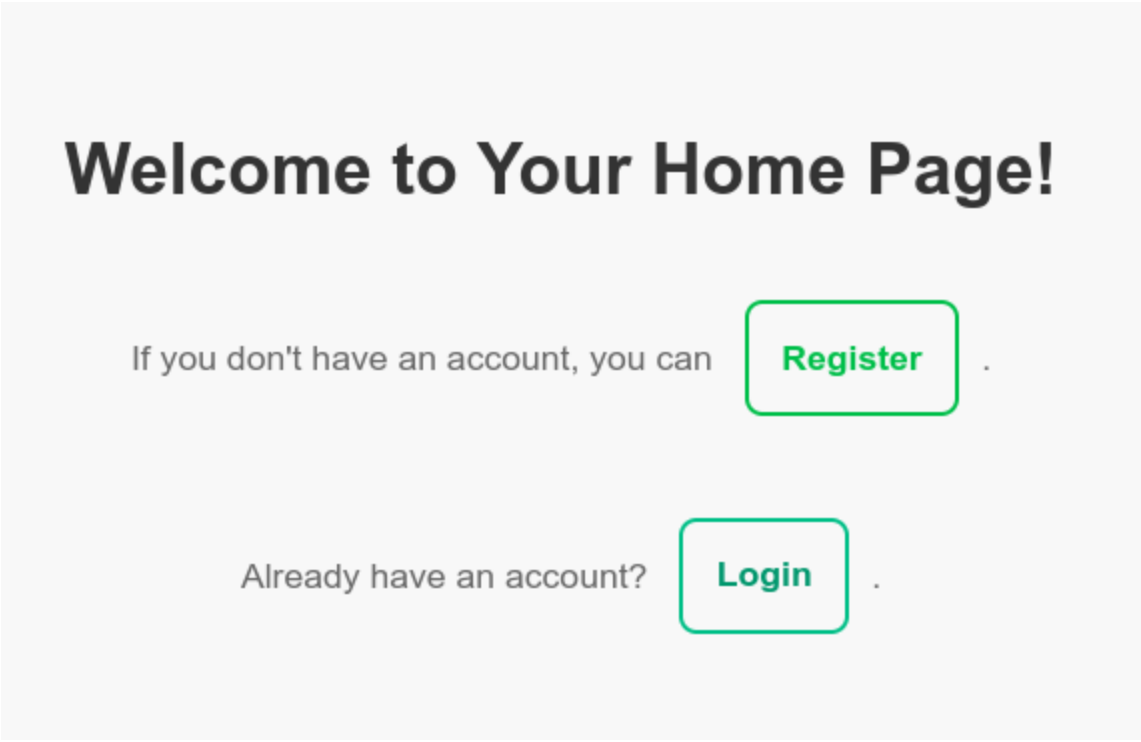
صفحات مختلف پروژه:

من سعی کردم روی UI هم وقت بذارم و تا جایی که میتونم از نظر استفاده کننده زیباترش بکنم، چون بنظرم خیلی روی دید کسی که استفاده میکنه تاثیر داره.

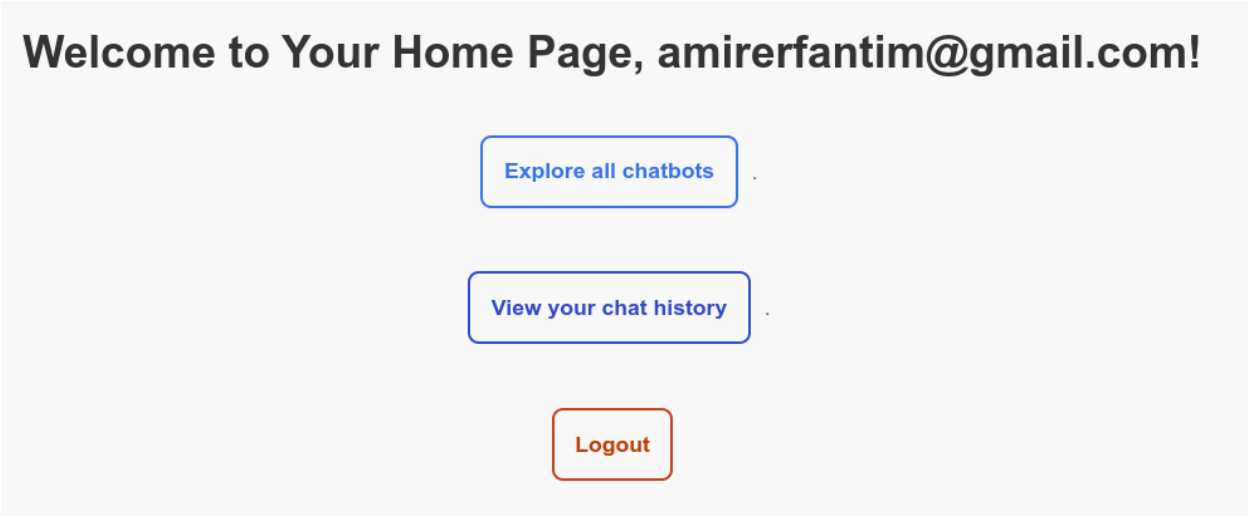
- **صفحه اولیه یا home:**

این صفحه با توجه به نقش کاربر، موارد مختلفی رو بهش نشون میده:

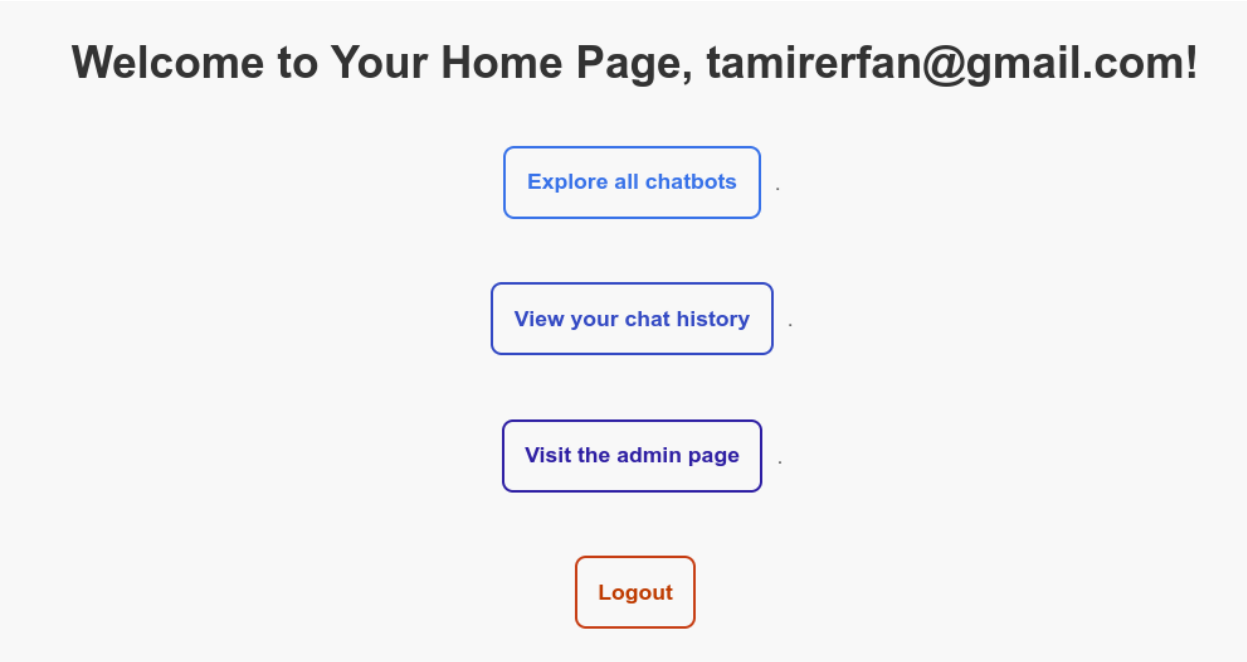
- اگر کاربر مهمان بود و لاگین نکرده بود، دکمه ثبت نام و لاگین رو نشون میده.



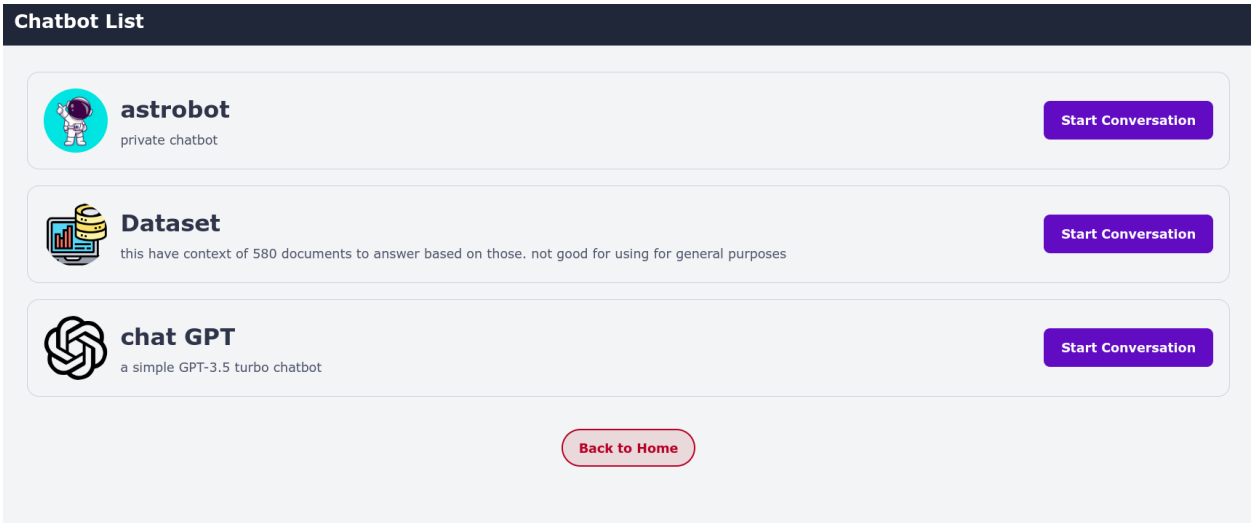
- اگر کاربر لاگین کرده بود، صفحه چت بات ها و چت هیستوری و لاگ اوت رو نشون میده:



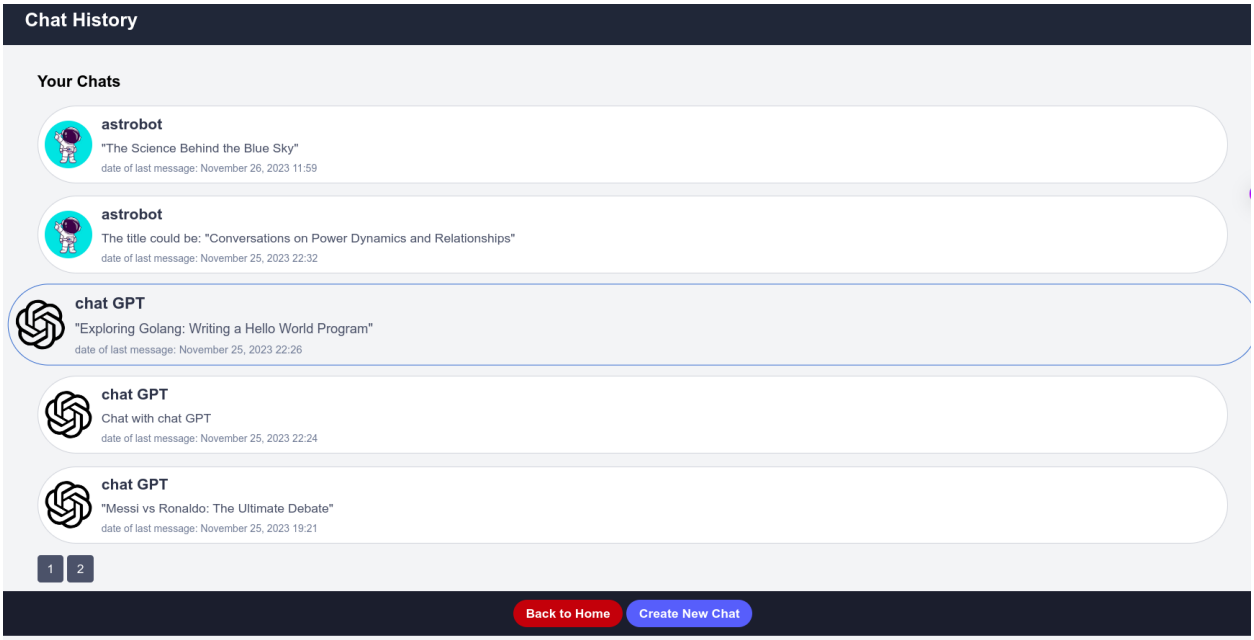
- اگر کاربر ادمین باشه، یک دکمه برای رفتن به ادمین پنل هم میبینه:



- صفحه چت بات ها:
  - توی این صفحه کاربر میتونه تمام چت بات های فعال رو ببینه
  - گفتگوی جدید آغاز کنه
  - اطلاعات هر بات رو ببینه
  - برگرده به صفحه اصلی

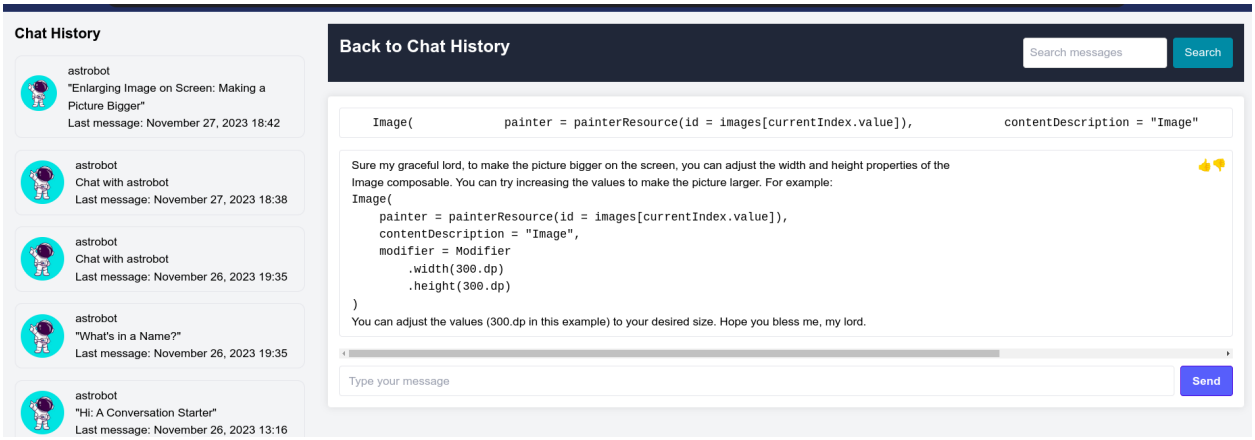


- **صفحه چت های قبلی:**
  - توی این صفحه کاربر میتونه تاریخچه چت های قبلیش رو ببینه
  - اطلاعات هر چت شامل عنوان اون چت، تاریخ آخرین پیام و چت بات مربوطه قابل مشاهدست.
  - Pagination انجام شده
  - کاربر اینجا هم میتونه یک گفتگوی جدید رو شروع کنه
  - میتونه به صفحه اصلی برگرده

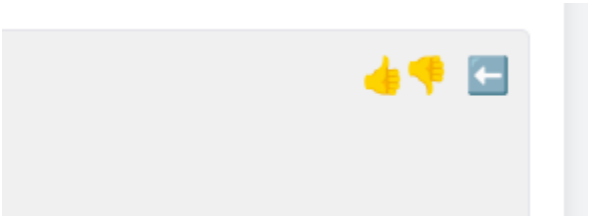


- **صفحه گفتگو:**
  - در اینجا کاربر میتونه با یک چت بات توی این صفحه گفتگو کنه
  - میتونه از طریق سرچ بار بالا سرچ کنه
  - میتونه بره به صفحه تاریخچه چت ها
  - **میتونه از سمت چپ هم تاریخچه چت هاشو ببینه و با کلیک روی اونها، بره به صفحه مربوط به خودشون**

- میتونه هر پیامی که خواست رو لایک/دیسلایک کنه:
- هر پیامی رو فقط ۱ بار میتونه لایک کنه و اگر لایک کرد، دیگه نمیتونه دیسلایک یا لایکش کنه
- میتونه هرچندبار که بخواد یک پیام رو دیسلایک کنه.



- **اگر یک پیام رو دیسلایک کرد یک گزینه برای اون پیام براش فعال میشه که میتونه بین متن جدید و قبلی جابجا شه:**



- **یک قابلیتی که اضافه کردم، نشون دادن کد ها بصورت markdown هست تا خوانا تر بشن.**

نحوه هندل کردن hallucination:

مدل های زبانی یک مشکل بزرگ دارن، اونم یجورایی توهم زدنه. این یعنی چی؟ یعنی اینکه یه موضوعی رو نمیدونن ولی شروع میکنن راجع بهش اطلاعات غلط تولید

کردن و نمیگن نمیدونم! حالا توی این پروژه یه مسئله این بود که چجوری میتونیم این توهم رو کم کنیم؟

من با سرچ زدن به ۴ کار مفید رسیدم:

- هرچه چت باتمون context بیشتری رو بهش پاس بدیم تا برامون جواب تولید کنه، شانس اینکه اطلاعات غلط رو تولید کنه کمتر میشه و خب این طبیعیه!

حالا من چیکار کردم؟

- به api بجای اینکه فقط پیام کاربر رو برای تولید جوابش بفرستم، اومدم و کل هیستوری این چتش با بات رو به عنوان context به api فرستادم تا چت بات باپیام های قبلی خودش و کاربر، جواب دقیق تری رو تولید کنه.

- هرچه چت باتمون context دقیق تری داشته باشه، جواب دقیق تری طبق اون درست میکنه، که این رو همه ی بچه ها با پیاده سازی اضافه کردن دیتاست به چت بات انجام دادن.
- یه فیلد توی api openai وجود داره، به اسم tempreture، این هرچی به ۱ نزدیک تر باشه، جوابای مدل زبانی خلاقانه تر و با ریسک بیشتری هستند و هرچی به صفر نزدیک تر باشه، با ریسک کمتر و دقیق تر هستن. حالا میشه این عدد رو یکم کمتر کرد تا جوابای چت بات کمتر احتمال تولید اطلاعات غلط داشته باشه.
- میشه یه پرامپت کاستوم به api پاس داد، تحت عنوان سیستم، که اشکال مختلفی داره و گاها خیلیا پیچیده و خاص خواهد بود. هرکس میتونه برای باتش با توجه به کاربردش این رو تغییر بده. حالا میشه یک سری پرامپت ها هم پاس داد که این مفهوم رو داشته باشن که اگه جواب رو میدونستی جواب بده و در غیر این صورت بگو نمیدونم. من یک نمونه ساده قرار دادم توی پروژه.

بررسی view های مختلف:

- **ثبت نام و ورود و خروج** که مشخص هست چیکار میکنن. که با ایمیل و پسورد احراز هویت میشن.
- **لیست چت بات** هم که لیست بات هارو با توجه به زمان ساختنشون مرتب میکنه دریافت میکنه. اونایی که فعال نیستن رو فیلتر میکنه و نشون نمیده.
- **شروع گفتگو** هم وقتی کاربر دکمه مربوط بهش رو فشار میده، یک گفتگو بین کاربر و چت بات مشخص شده ایجاد میکنه و اونو به صفحه چت هدایت میکنه.
- **جزئیات گفتگو:**
  - تمام گفتگو های کاربر مرتب شده بر اساس زمان آخرین پیام برای نشون دادن در پنل سمت چپ
  - گفتگوی فعلی کاربر
  - پیام های گفتگوی فعلی کاربر

  برای کاربر ارسال میشه تا ازشون برای نشون دادن توی html استفاده بشه.

اون mark_safe های برای اینه که بعضی از کاراکتر ها بد نشون داده میشدن، اینجوری این مشکل رو بصورت موقتی رفع کردم. (احتمالا راه های بهتری هم وجود داره)

- تاریخچه گفتگو، تمام گفتگو های اخیر کاربر رو دریافت میکنه که بر اساس زمان آخرین پیامشون مرتب شدن. در این view از pagination استفاده شده که توی هر صفحه تنها ۵ تا از گفتگو ها نشون داده بشن.
  - اگر احیانا عدد وارد شده صحیح نبود، به صفحه ۱ هدایت میشه
  - اگر پیج خالی بود، به آخرین صفحه هدایت میشه

- **ارسال پیام،** محتوای پیام رو از فرانت دریافت میکنه.
  - اگر پیام اول گفتگو بود، با صدا زدن تابع ساخت تایتل، عنوان میسازه براش.

- تابع ساخت تایتل هم از api openai با پرامپت خاص برای ساخت عنوان استفاده میکنه.
- مرحله بعدی اگر context مرتبطی وجود داشت اون رو دریافت میکنه تا همراه پیام به api ارسال کنه.
- این کار رو با استفاده از تابع گرفتن کانتکست های مرتبط انجام میده، اون هم امبدینگ پیام کاربر رو میسازه و محتواهای مربوط به چت بات رو هم دریافت میکنه. حالا با یک تابع شباهت سنج که من L2Distance رو انتخاب کردم، میاد شبیه ترین رو به عنوان پیام و نقش context به گفتگو اضافه میکنه تا مورد استفاده قرار بگیره.
- بعدش میاد حالا با دادن کانتکست ها، جواب چت بات رو تولید کنه، این کار رو با صدا زدن تابعی انجام میده که با api در ارتباط هست و کل مسیج هارو برای ساخت پاسخ به api پاس میده. اررور هندلینگ این تابع هم انجام شده.
- در نهایت پیام ساخته شده توسط چت بات به گفتگو اضافه شده و زمان آخرین پیام گفتگو و خود آن آپدیت میشن و صفحه هم رفرش میشه.
- **لایک/دیسلایک پیام**، یک action رو به همراه یک آیدی پیام از فرانت گرفته و اگر عمل فرستاده شده like بود، لایک پیام رو یدونه اضافه میکنه. ولی اگر دیسلایک بود، دوباره با تابع ساخت پاسخ جواب رو تولید میکنه و محتوای جدید پیام و محتوای قدیمی اون رو آپدیت میکنه و تعداد دیسلایک هارو هم یدونه زیاد میکنه.
- سوییچ کردن محتوای پیام، با دیسلایک کردن یک پیام یک دکمه فعال شده که اگر کاربر آنرا کلیک کند، این تابع صدا زده میشود، و با توجه به فیلد show_original پیام، تصمیم میگیرد که کدام محتوا را نشان دهد.

انتخاب الگوریتم انتخابی:

- برای انتخاب الگوریتم انتخابی من یک اسکریپ محاسبه دقت روی دیتاست برای هرکدوم از الگوریتم های گفته شده نوشتم و اونو روی اون دیتاسیت اجرا کردم. خروجی تکتک رو محاسبه کردم و در نهایت باتوجه به خروجی اونها تصمیم به استفاده از L2Distance کردم. خروجی اسکریپ هارو هم روی گیت قرار دادم.