تمرین سری اول (آشنایی با برخی از خدمات ابری)

محدثه سادات اطیابی ۹۷۲۶۰۰۴

در این پروژه برای ارتباط با هر سرویس ابری، یک فایل py. جداگانه ایجاد کرده‌ام و توابع مورد نیاز برای استفاده از هر سرویس را در قالب کلاس یا تابع در فایل مربوطه پیاده سازی کرده‌ام. در ابتدا به توضیح جداگانه هرکدام از این قایل‌ها می‌پردازم و سپس چگونگی پیاده سازی سرویس‌های مورد نظر را بررسی می‌کنم.

برای برقراری ارتباط با هر سرور و استفاده از خدمات آنها از کدهای داخل documentation هر سایت استفاده کرده‌ام.

- **پایگاه داده به عنوان سرویس)**

برای این قسمت از سرویس MongoDB Atlas به عنوان پایگاه داده استفاده کرده‌ام و کد آن را در فایل mongoDB.py قرار داده‌ام. در ابتدا در این سایت یک حساب کاربری ساختم و سپس پایگاه داده‌ای بنام HW1 ساختم و در آن collectionای بنام hw1 ایجاد کردم که داده‌ها را در آن ذخیره می‌کنم. در این collection ستون‌های id، ایمیل، دسته بندی، متن و وضعیت را داریم. در این فایل کلاسی بنام MongooDB داریم که سه تابع insert، update و show دارد. در ابتدا در __init__ کلاینت را تعریف می‌کنیم و آدرس سروری که ساخته‌ایم را به آن می‌دهیم. سپس متغیرهای db، coll (collection) و id_ را برای آن تعریف می‌کنیم.

- ✓ تابع insert: برای اضافه کردن داده جدید در دیتابیس از این تابع استفاده می‌کنیم که ایمیل و متن را به عنوان ورودی از کاربر دریافت می‌کند. در ابتدا باید یک شناسه یکتا برای داده در نظر بگیریم، به همین دلیل یک عدد رندوم بین ۰ تا ۱۰۰۰۰ تولید می‌کنیم. سپس بررسی می‌کنیم که آیا قبلا این شناسه را ذخیره کرده‌ایم یا نه. تا زمانیکه شناسه‌ها تکراری باشند، یک شناسه جدید تولید می‌کنیم. سپس با اطلاعاتی که داریم فیلدهای شناسه، ایمیل و متن را برای داده پر می‌کنیم و با استفاده از تابع insert_one داده جدید را به دیتابیس اضافه می‌کنیم. در این حالت، وضعیت همه آگهی‌ها بصورت Pending است.
- ✓ تابع update: این تابع شناسه، وضعیت و دسته بندی را به عنوان ورودی دریافت می‌کند. برای آپدیت کردن اطلاعات یک داده از تابع update_one استفاده می کنیم که با استفاده از id، داده مورد نظر را می‌یابد و وضعیت و دسته بندی آن را بروزرسانی می‌کند.
- ✓ تابع show: این تابع شناسه داده مورد نظر را دریافت می‌کند و با استفاده از تابع find_one، اطلاعات آن را برمی‌گرداند.

- **ذخیره ساز شئ S3)**

در این بخش از سرویس ابرآروان استفاده می‌کنم. در ابتدا حساب کاربری ایجاد کردم و در قسمت stoage یک سبد برای ذخیره سازی اطلاعات ساختم. کدهای مربوط به این سرویس را در فایل abrArvan.py و در کلاس AbrArvan ذخیره کرده‌ام. در __init__ این کلاس، کدهای مربوط به متصل شدن به کلاینت را نوشته‌ام. این کلاس دو تابع برای آپلود و دانلود اطلاعات دارد.

- ✓ تابع upload: در این تابع شناسه و آدرس محل داده را به عنوان ورودی دریافت می‌کنیم. سپس با استفاده از upload_file داده مورد نظر را آپلود می‌کنیم. طبق تعریف پروژه، تصاویری که می‌خواهیم آپلود کنیم بنام شناسه‌شان ذخیره شده‌اند. به همین دلیل به جای نام فایل، عبارت id + "jpg" را به تابع می‌دهیم.

- ✓ تابع دانلود: در این تابع مقدار شناسه را به عنوان ورودی دریافت می‌کنیم. سپس با استفاده از تابع download_file عکس مورد نظر را دانلود می‌کنیم و نام فایلی که می‌خواهیم دانلود کنیم و مسیری که می‌خواهیم عکس را در آن ذخیره کنیم را به آن می‌دهیم.

- **سرویس RabbitMQ:**

در این قسمت از سرویس RabbitMQ استفاده می‌کنیم و در ابتدا در آن یک جساب کاربری ایجاد می‌کنیم. استفاده از این سرویس به دو بخش تقسیم می‌شود که بخش اول مربوط به سرویس اول و بخش دوم مربوط به سرویس دوم می‌باشد. بخش اول را در فایل rabbitPika.py و بخش دوم را در فایل rabbitPikaServer.py پیاده سازی می‌کنیم.

در فایل rabbitPika.py تنها یک تابع به نام send داریم که شناسه را دریافت می‌کند و آن را به صف ارسال می‌کند. برای این کار ابتدا آدرس URL را مشخص می‌کنیم و به آن یک ارتباط ایجاد می‌کنیم و صف مورد نظر را تعیین می‌کنیم. سپس شناسه را به صف می‌فرستیم و سپس ارتباط را خاتمه می‌دهیم.

در فایل rabbitPikaServer.py دو تابع داریم.

- ✓ تابع receiver: در این تابع نیز URL را تعیین می‌کنیم و سپس ارتباط را برقرار می‌کنیم و اگر داده‌ای روی صف وجود داشته باشد، آن را دریافت می‌کند و تابع callback را فراخوانی می‌کند. در نهایت اگر داده‌ای روی صف وجود نداشته باشد، ارتباط را ادامه می‌دهیم تا داده جدید روی صف قرار بگیرد.
- ✓ تابع callback: این تابع پس از دریافت داده از روی صف فراخوانی می‌شود و در آن، اقدامات مربوط به سرویس دوم انجام می‌شود. در ابتدا مقدار شناسه را از body دریافت شده استخراج می‌کنیم. سپس تصویر دخیره شده در ابرآروان را با استفاده از تابع download، دانلود می‌کنیم. سپس تصویر دانلود شده را به تابع tag از فایل imagga می‌دهیم تا تصویر را برچسب گذاری کند و نتیجه را در متغیرهای tag و description ذخیره می‌کنیم.
  - اگر آگهی پذیرفته شده باشد (یعنی description == accepted باشد)، داده ذخیره شده در دیتابیس را آپدیت می‌کنیم و مقدار tag را به عنوان دسته بندی و مقدار Accepted را به عنوان وضعیت آن داده ذخیره می‌کنیم. در نهایت نیز ایمیلی با عنوان Accepted به ایمیل مربوط به آگهی ارسال می‌کنیم و "آگهی شما با شناسه id پذیرفته شده است" را می‌فرستیم.
  - اگر آگهی پذیرفته نشده باشد (یعنی description == rejected باشد)، داده ذخیره شده در دیتابیس را آپدیت می‌کنیم و دسته بندی را برابر با None و وضعیت را برابر با Rejected قرار می‌دهیم. سپس ایمیلی با عنوان Rejected به ایمیل مربوط به آگهی ارسال می‌کنیم و "آگهی شما با شناسه id رد شده است" را می‌فرستیم.

- **سرویس پردازش عکس (برچست زنی عکس‌ها)**

در این قسمت از سرویس Imagga استفاده می‌کنیم و کدهای مربوطه را در فایل imagga.py ذخیره می‌کنیم. در این فایل یک تابع با عنوان tag داریم. در این تابع با استفاده از تابع post، عکس مورد نظر را به سرویس ارسال می‌کنیم (درواقع آن را در سایت imagga آن را آپلود می‌کنیم) و پاسخ آن را در متغیر response ذخیره می‌کنیم. حال تمام تگ‌هایی که به عنوان پاسخ دریافت کرده‌ایم را بررسی می‌کنیم و اگر تگ vehicle با درصد اطمینان بیش از ۵۰ یافتیم، مقدار accepted را به همراه تگی که بیشترین درصد اطمینان دارد (به عنوان دسته بندی تصویر) برمی‌گردانیم. اما اگر چنین برچسبی را نیافتیم، مقدار rejected و دسته بندی None را برمی‌گردانیم.

- **سرویس ارسال ایمیل)**

برای راسال ایمیل از سرویس Mailgun استفاده می‌کنیم و کد آن را در فایل mailgun.py ذخیره می‌کنیم. این فایل دارای یک تابع با عنوان send_simple_message است که ایمیل، موضوع ایمیل و متن آن را به عنوان ورودی دریافت می‌کند. برای ارسال درخواست به سایت، از تابع post استفاده می‌کنیم و آدرس domain را به آن می‌دهیم. سپس در قسمت from، اطلاعات ارسال کننده ایمیل را وارد می‌کنیم که نام ارسال کننده Mohadeseh Atyabi و ایمیل آن atyabi2000@gmail.com است. در سمت to آدرس ایمیل گیرنده را وارد می‌کنیم (که به عنوان ورودی دریافت کرده‌ایم). برای عنوان ایمیل مقدار subject و برای متن ایمیل، مقدار description را قرار می‌دهیم.

تا اینجا کدهای مربوط به سرویس‌های مختلف را بررسی کردیم. عملیات مربوط به سرویس دوم تماماً در فایل rabbitPikaServer.py اجرا می‌شود. حال به بررسی کدهای مبوط به سرویس اول می‌پردازیم که در فایل main.py قرار دارد. در این سرویس باید دو سرویس API پیاده سازی کنیم که برای این کار از flask استفاده می‌کنیم و سرور را بصورت local اجرا می‌کنیم.

- در ابتدا اگر بعد از آدرس سرور (که بصورت 127.0.0.1 + port number است و بعد از اجرای کد پایتون به ما داده می‌شود) عبارتی وارد نشود یا تنها یک / قرار گیرد، تنها نوشته "Hey! Welcome to my local host :(" نمایش داده می‌شود.
- API ثبت آگهی: برای ثبت آگهی باید بعد از آدرس سرور، عبارت "/register" وارد شود و پس از آن اطلاعات ایمیل، متن و آدرس تصویر آگهی داده شود. داخل تابع handle_register اقدامات مربوط به این بخش هندل می‌شود. در ابتدا مقادیر ایمیل، متن و آدرس تصویر که از کاربر دریافت کرده‌ایم را در متغیرهای مناسب ذخیره می‌کنیم. سپس ایمیل و متن را به تابع insert از کلاس mongoDB می‌دهیم و id آگهی که تابع برمی‌گرداند را در متغیر id ذخیره می‌کنیم. حال با استفاده از آدرس تصویر، تصویر را با نام id در ابرآروان آپلود می‌کنیم. سپس مقدار id را به وسیله تابع send از فایل rabbitPika در صف قرار می‌دهیم تا بررسی شود. در نهایت نیر عبارت "آگهی شما با شناسه id ثبت شد" را به کاربر نمایش می‌دهیم.
- API دریافت آگهی: برای دریافت آگهی باید بعد از آدرس سرور، عبارت "/receive" وارد شود و پس از آن شناسه آگهی‌ای که می‌خواهیم آن را ببینیم، داده شود. در تابع handle_receive که اقدامات مربوط به این بخش هندل می‌شود، با استفاده از تابع show از کلاس mongoDBB، اطلاعات آگهی با شناسه id را دریافت می‌کنیم. حال بر اساس عبارتی که برای وضعیت آگهی داریم، به سه حالت می‌رسیم.
  - اگر وضعیت آگهی برابر Pending باشد، عبارت "آگهی شما در صف بررسی است" نمایش داده می‌شود.
  - اگر وضعیت آگهی برابر Rejected باشد، عبارت "آگهی شما رد شده است :(" نمایش داده می‌شود.
  - اگر وضعیت آگهی برابر Accepted باشد، تابع show_advertisement فراخوانی می‌شود. این تابع شناسه و اطلاعات داده دریافتی از دیتابیس را به عنوان ورودی دریافت می‌کند. در فولدر پروژه، یک فولدر به نام static داریم که در آن فولدر IMG قرار دارد و عکس آگهی‌هایی که پذیرفته شده‌اند در آن ذخیره شده‌اند. مسیر این دو پوشه را در IMG_FOLDER ذخیره می‌کنیم و سپس آدرس عکس با نام id.jpg را در Flask_Logo ذخیره می‌کنیم. حال برای نمایش اطلاعات آگهی به همراه تصویر آن، باید این داده‌ها را به کد HTML بدهیم.

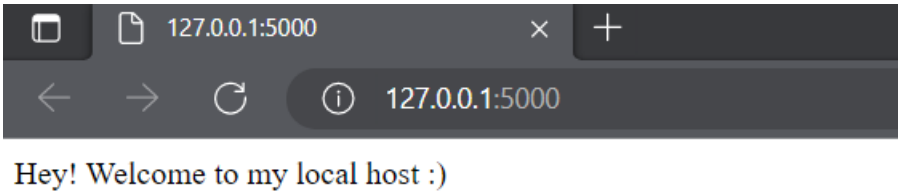
این فایل HTML با نام index.html در پوشه templates در فولدر پروژه ذخیره شده است. برای این کار از تابع render_template استفاده می‌کنیم و نام فایل HTML را به آن می‌دهیم. سپس اطلاعاتی که برای نمایش آگهی نیاز داریم را به عنوان ورودی، به تابع می‌دهیم. این اطلاعات شامل موارد زیر است:

- Flask_Logo: آدرس قرارگیری تصویر آگهی را در خود ذخیره کرده است.
- Description: متن آگهی را مشخص می‌کند.
- Category: دسته بندی آگهی را مشخص می‌کند.
- State: وضعیت آگهی را تعیین می‌کند.

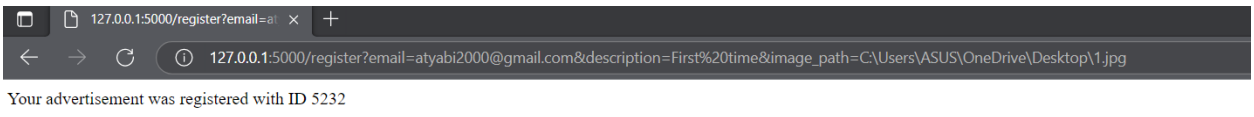
اطلاعات مورد نیاز برای این بخش را از متغیر data که به عنوان ورودی دریافت کردیم، تامین می‌شود.
در فایل HTML، در تگ p اطلاعات آگهی را نمایش می‌دهیم و محتوای هر بخش را از متغیرهایی که در کد پایتون ارسال کرده‌ایم، تامین می‌کنیم. تصویر آگهی را با استفاده از تگ img نمایش می‌دهیم و برای آدرس تصویر، در قسمت src متغیر Flask_Logo را قرار می‌دهیم.

امتحان عملکرد کد)

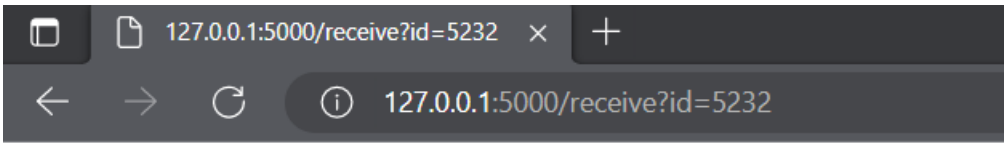
در ادامه نمونه‌ای از روند ثبت و دریافت آگهی را بررسی می‌کنیم. در ابتدا کد main را در پایتون اجرا می‌کنیم و آدرس 127.0.0.1:5000 به عنوان آدرس سرور local بازگردانده می‌شود. در ابتدا که این آدرس را باز می‌کنیم، با تصویر زیر مواجه می‌شویم.



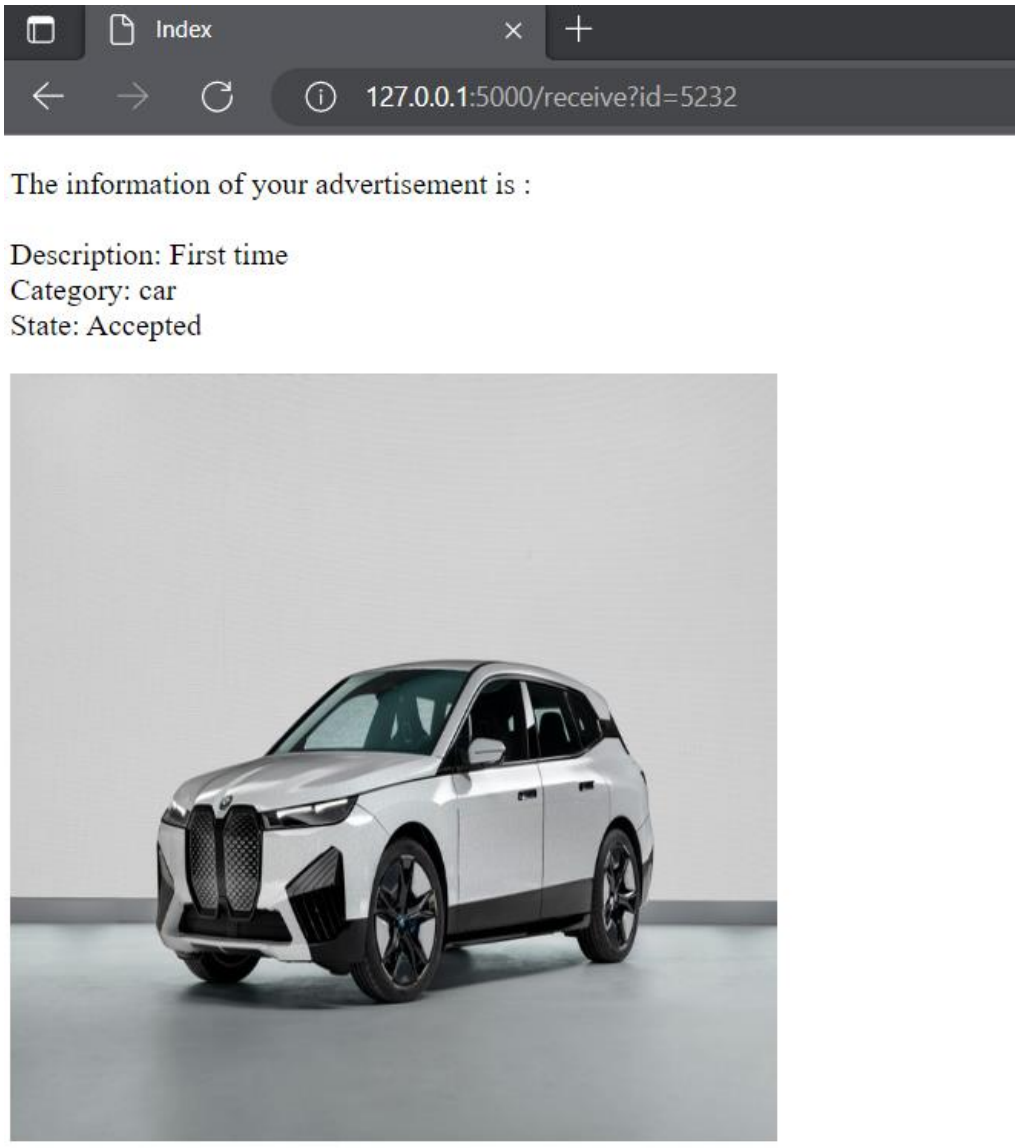
همانطور که مطرح شد، اگر تنها آدرس سرور را وارد کنیم با صفحه خوشامدگویی مواجه می‌شویم. حال می‌خواهیم آگهی‌ای را ثبت کنیم. بعد از اضافه کردن /register، پارامترهای مورد نیاز را مقداردهی می‌کنیم. به عنوان ایمیل مقدار atyabi2000@gmail.com را وارد می‌کنیم و متن "اولین تلاش" را به عنوان متن آگهی ثبت می‌کنیم. آدرس تصویر آگهی را نیز به عنوان مقدار پارامتر image_path وارد می‌کنیم. پس از وارد شدن به این صفحه، تصویر زیر را می‌بینیم که نشان می‌دهد آگهی مورد نظر با شناسه ۵۲۳۲ وارد سامانه شده است.



حال درخواست دریافت آگهی را وارد می‌کنیم. به این منظور پس از /receive، به عنوان id مقدار شناسه آگهی مورد نظر که برابر با ۵۲۳۲ است را وارد می‌کنیم. مشاهده می‌کنیم که آگهی هنوز بررسی نشده است و همچنان در صف انتظار قرار دارد.



پس از گذشت مدتی مجدداً درخواست دریافت آگهی را وارد می‌کنیم و با صفحه زیر مواجه می‌شویم که اطلاعات آگهی از جمله متن، وضعیت و دسته بندی آن را به همراه تصویر آگهی نمایش می‌دهد. می‌بینیم که آگهی در وضعیت Accepted است و دسته بندی آن، ماشین است.



در ادامه در قسمت spam ایمیل نیز می‌بینیم که ایمیل زیر ارسال شده است.

Accepted Spam ×

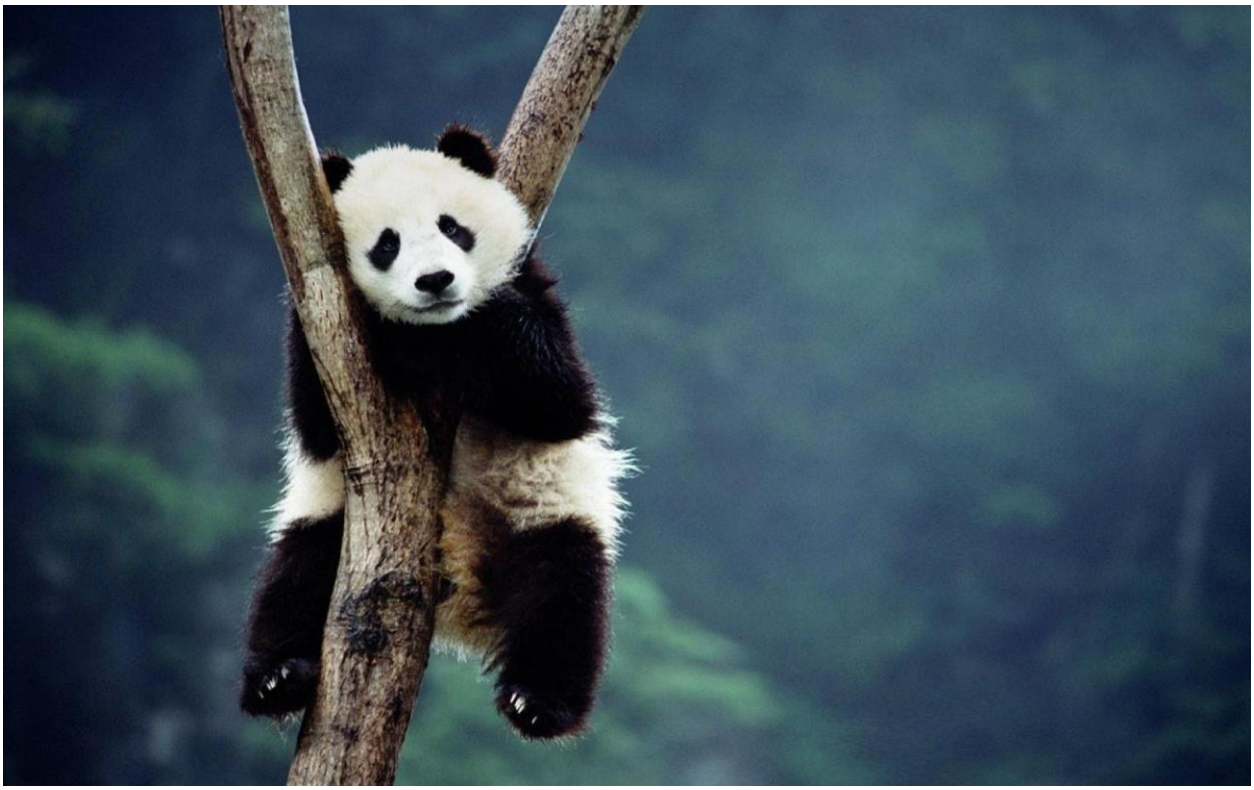
Mohadeseh Atyabi via sandbox.mgsend.net
to me

Why is this message in spam? It is similar to messages that were identified as spam in the past.

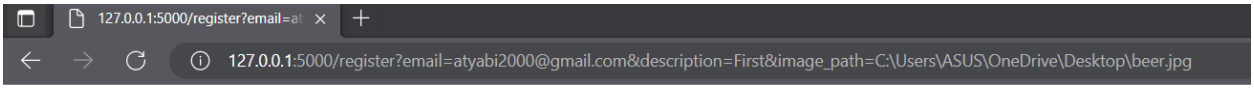
Report not spam

Your advertisement with ID 5232 is accepted :)

حال آگهی دیگری می‌خواهیم ثبت کنیم که برای تصویر آن، عکس زیر را آپلود می‌کنیم.

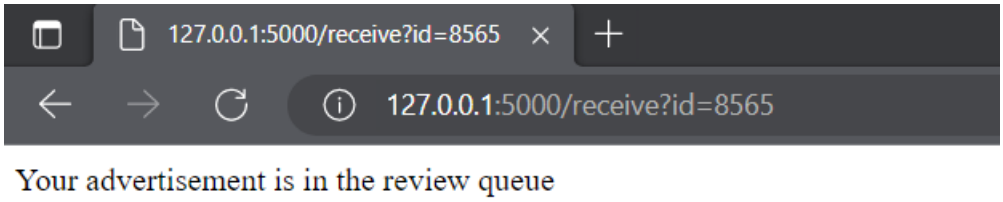


مشابه حالت قبل اطلاعات ایمیل و متن و آدرس تصویر را به عنوان ورودی به URL می‌دهیم. آگهی با شناسه ۸۵۶۵ ثبت شده است.

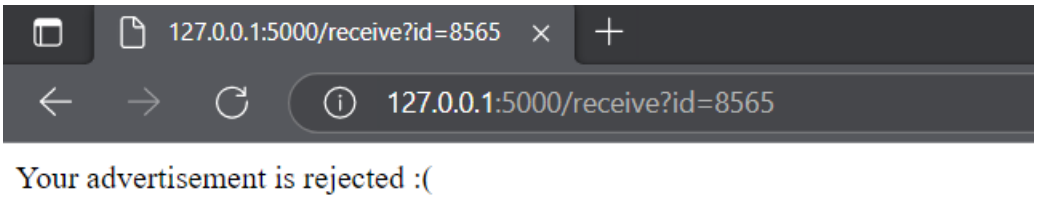


Your advertisement was registered with ID 8565

حال درخواست دریافت آگهی را وارد می‌کنیم و می‌بینیم که آگهی در صف بررسی قرار دارد.



پس از گذشت مدتی، دوباره درخواست دریافت آگهی را وارد می‌کنیم. می‌بینیم که آگهی رد شده است و اطلاعات آن نمایش داده نمی‌شود.



در ادامه نیز ایمیل زیر را برای این آگهی دریافت کرده‌ایم.