



دانشکده مهندسی کامپیوتر

## مرورگر ساده با استفاده از Socket ها

استاد راهنما:

آقای دکتر زمانی فر

تهیه کننده:

سید عمید اسدالهی مجد

بهمن ۱۴۰۰

## فهرست مطالب

1.....	مقدمه
1.....	راه اندازی
1.....	دسته بندی فایل های موجود در پروژه
1.....	ماژول <b>main</b>
2.....	ماژول <b>limited_http_client</b>
2.....	متود <b>tokenize_address</b>
2.....	متود <b>tokenize_response</b>
2.....	متود <b>response_decode</b>
2.....	متود <b>create_get_request</b>
3.....	متود <b>send_request</b>
4.....	اسکرین شات هایی از اجرای پروژه

## مقدمه

در این پروژه سعی بر پیاده‌سازی یک مرورگر بود که یک مرورگر ساده با استفاده از سوکت‌ها (Sockets) و کتابخانه PyQt5 در زبان پایتون پیاده‌سازی شد. درواقع با استفاده از سوکت‌ها پیامی بین دو هاست (کلاینت و سرور) ارسال و دریافت می‌شود و در محیط گرافیکی مرورگر پاسخ سرور به درخواست HTTP کلاینت نمایش داده می‌شود. این مرورگر از گونه‌های مختلف آدرس و همچنین HTTPS (با تایپ پروتکل `https://` قبل از آدرس) پشتیبانی می‌کند.

## راه‌اندازی

برای راه‌اندازی پروژه تنها نیاز به نصب دو کتابخانه زیر می‌باشد:

۱. `magic` : برای تشخیص فرمت فایل با استفاده از `magic number` (مثل `html`, `text`, `jpg`, ...) .

۲. `pyqt5` : برای اجرای محیط گرافیکی

## دسته‌بندی فایل‌های موجود در پروژه

پروژه شامل یک فایل اصلی `limited_http_client` می‌باشد که یک کلاینت است و هسته اصلی مرورگر می‌باشد. فایل `main.ui` در واقع محیط گرافیکی برنامه در فرمت `xml` است که توسط فایل `main.py` لود می‌شود. فایلی که در نهایت با پایتون باید اجرا گردد تا مرورگر همراه با محیط گرافیکی اجرا شود `main.py` نام دارد که محیط گرافیکی مرورگر را به هسته‌ی آن متصل می‌سازد.

## ماژول `main`

برای اجرای مرورگر به صورت گرافیکی باید این فایل با دستور `python main.py` اجرا شود. این فایل شامل یک کلاس `MainWindow` است که در این کلاس یک آبجکت از محیط گرافیکی طراحی شده ساخته می‌شود و سپس با کلیک کاربر بر روی `GO` یا فشردن کلید `Enter` پس از تایپ آدرس مورد نظر، ماژول `limited_http_client` در پس زمینه اجرا شده (با فراخوانی متود `send_request` از ماژول `limited_http_client`) و آدرس کاربر به این ماژول داده می‌شود تا قسمت محتوای (`Content`) نتیجه درخواست (پاسخ سرور) را در محیط مرورگر نمایش دهد. اگر آدرس وارد شده بسیار کوچک باشد (کمتر از سه حرف) یا مشکلی در دریافت پاسخ وجود داشته باشد، به ترتیب پیغام‌های `Invalid Address` و `Requested Address is not Reachable` روی مرورگر مشاهده خواهند شد.

## ماژول `limited_http_client`

این ماژول درواقع قسمت اصلی مرورگر است که آدرس درج شده در محیط گرافیکی را به عنوان درخواست ارسال کرده و پاسخ درخواست را به محیط گرافیکی بر می گرداند.

در این ماژول با صدا زدن متود `send_request` با آدرس درخواستی کاربر، کار بارگیری صفحات آغاز می شود.

### متود `tokenize_address`

این متود URL وارد شده کاربر را دریافت کرده و دیکشنری `address` را بر می گرداند که شامل `host` به عنوان آدرس پایه سرور، `port` به عنوان پورت درخواستی کاربر، `path` به عنوان مسیر درخواستی کاربر و پروتکل می باشد. کاربر امکان وارد کردن آدرس به گونه های مختلف را دارد. مثلاً همراه با `http://` یا بدون آن، با شماره پورت یا بدون آن و ..

همچنین این متود اگر پورتهی وارد نشده باشد پورت 80 را برای پروتکل HTTP و پورت 443 را برای پروتکل HTTPS در نظر می گیرد که برای درخواست `https` باید `https://` قبل از آدرس تایپ شود.

### متود `tokenize_response`

این متود پاسخ دریافت شده از سمت سرور را جدا سازی کرده و در نهایت یک دیکشنری شامل کلید هایی مثل `content – protocol – status – status_code` و ... بر می گرداند.

### متود `response_decode`

این متود وظیفه دیکود کردن اطلاعات با فرمت UTF-8 را بر عهده دارد.

### متود `create_get_request`

این متود بدنه درخواست را با مقادیر ساخته شده می سازد و به باینری دیکود می کند.

## متود send\_request

این متود درواقع متود اصلی کلاس Client است که مراحل زیر را برای هر آدرس ورودی اجرا می‌کند:

1. ساختن یک سوکت با پروتکل TCP و IP از ورژن ۴
2. اجرای جدا ساز آدرس بر روی ورودی کاربر (tokenize\_address)
3. با توجه به پروتکل معین شده در مرحله دوم، ایجاد اتصال (ساختن connection) به آدرس درخواستی کاربر با پروتکل HTTP به صورت پیش فرض و یا ایجاد اتصال امن با پروتکل HTTPS با استفاده از سوکت ساخته شده در مرحله اول (به ازای هر آدرس ورودی)
4. ساختن بدنه درخواست HTTP/HTTPS با اجرای متود (create\_get\_request) و ارسال این درخواست با استفاده از سوکت
5. در حلقه while بی نهایت تا وقتی که سرور داده‌ای برای ارسال داشته باشد و اندازه تکه (chunk) دریافتی از مقادیر دریافتی قبلی کمتر نباشد، به دریافت از طریق بافر ادامه می‌دهد. اما وقتی نشانه رسمی پایان فایل در chunk یافت شود به این حلقه خاتمه داده می‌شود و عمل دریافت پایان می‌یابد. همچنین اگر میزان timeout در نظر گرفته شده سپری شود و دریافتی وجود نداشته باشد و ساینز chunk نیز کوچکتر از مقادیر دریافتی قبلی باشد، حلقه while شکسته می‌شود.
6. تکه‌های دریافتی chunk به هم چسبانده شده و با اجرای متود tokenize\_response جداسازی می‌شود.
7. عملیات دیکود روی پاسخ دریافتی اجرا می‌شود تا پاسخ از فرمت باینری به رشته تبدیل شود.
8. پاسخ دریافتی به عنوان نتیجه متود send\_request برگردانده می‌شود.

## اسکرین شات‌هایی از اجرای پروژه

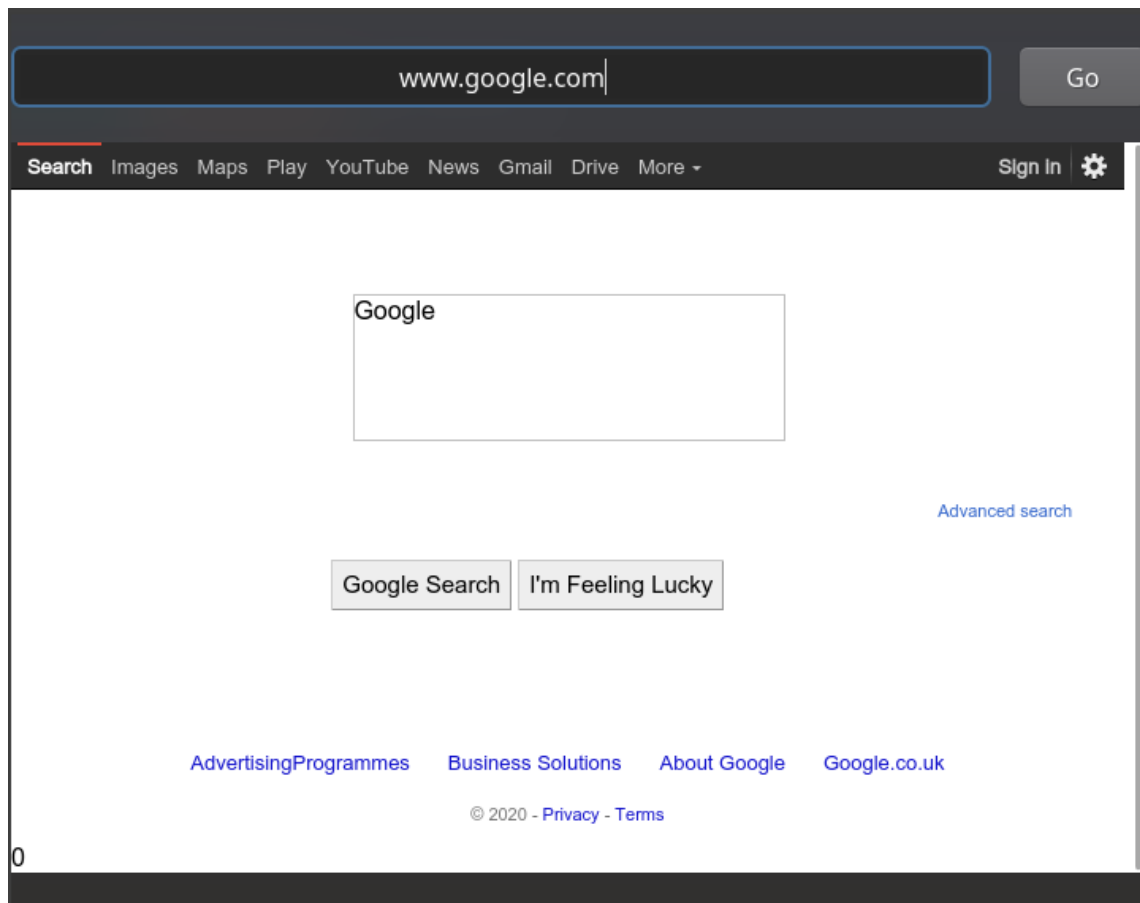
```
loading requested address...
http://www.google.com

Connection created with: www.google.com:80

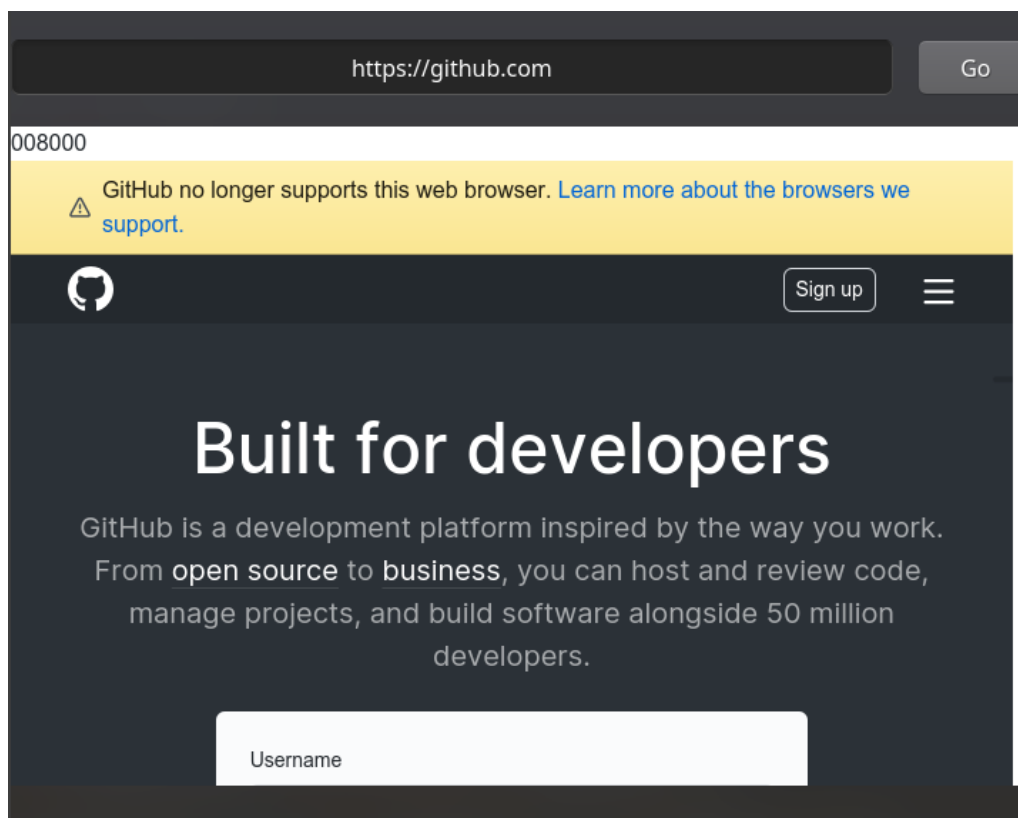
Sending GET request to www.google.com:80 ...
Response : 200 OK
Content-Length : 47011

Client socket closed
```

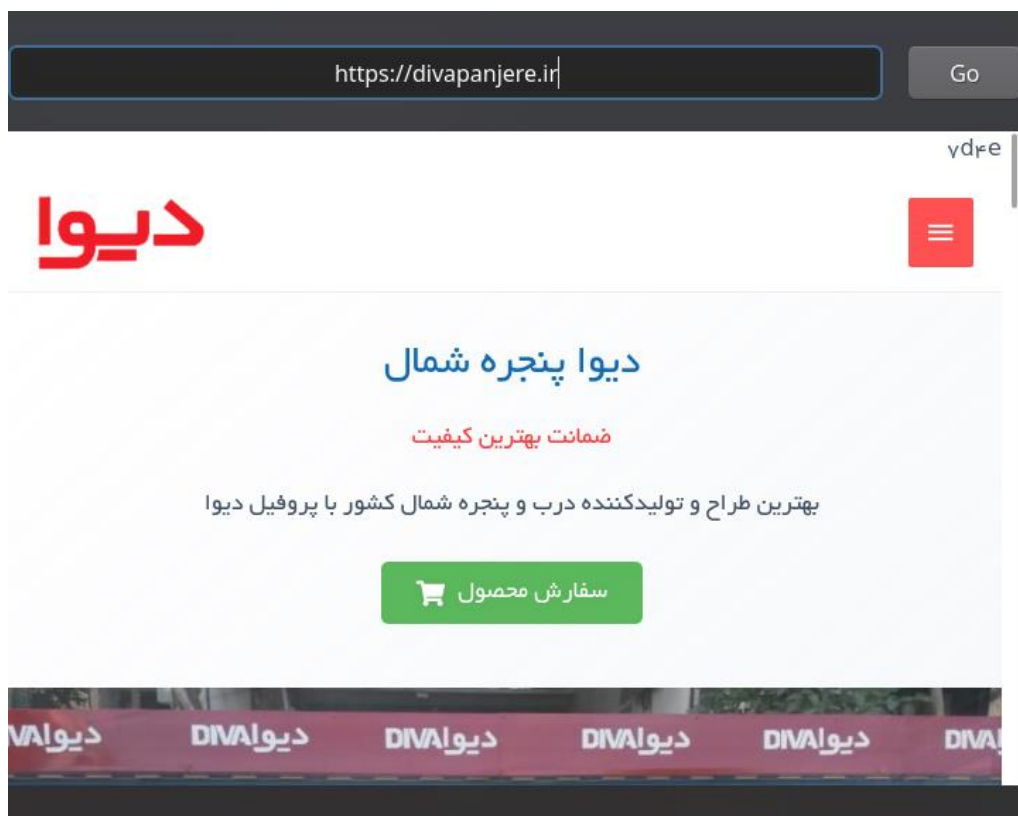
شکل 1: log های چاپ شده توسط مرورگر در command line برای بارگیری سایت google



شکل 2: صفحه نمایش داده شده به کاربر در مرورگر پس از بارگیری سایت google



شکل 3: صفحه نمایش داده شده به کاربر در مرورگر پس از بارگیری سایت **github**



شکل 4: صفحه نمایش داده شده به کاربر در مرورگر پس از بارگیری سایت **divapanjere.ir**