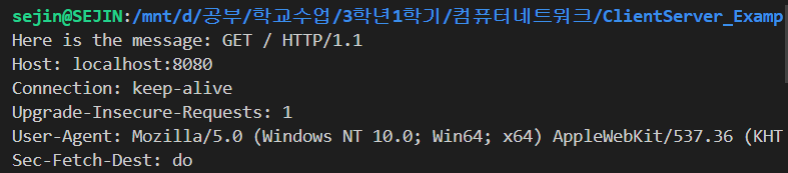
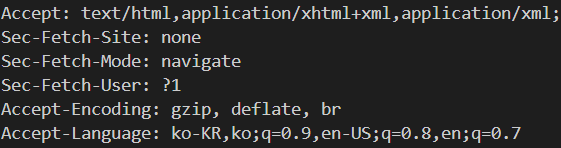
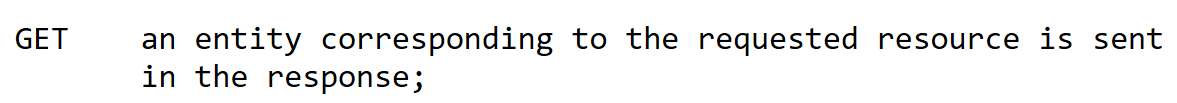
**컴퓨터 네트워크 프로젝트 레포트**

Part I.

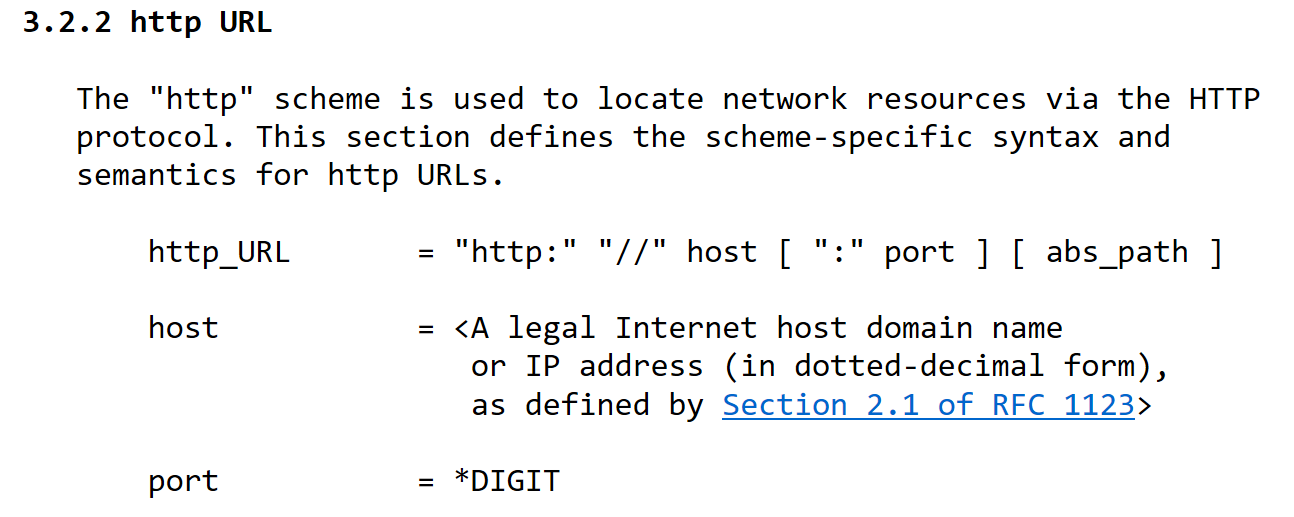




Server를 실행시켜 <http://localhost:8080/>로 접속했을 때 client에서 보내는 request 메시지는 위와 같다. 먼저 첫번째 줄에 GET / HTTP/1.1이 있다. 

RFC1945 Page 32를 보면 다음과 같은데, 그 말인 즉슨, GET 뒤에 오는 파라미터 즉 요청된 resource와 일치하는 요소를 client에게 전송해 준다. 뒤의 HTTP/1.1은 HTTP request가 1.1 version이라는 것을 의미한다.

그 다음 Host를 살펴보도록 한다.



RFC1945 Page 14를 살펴보면 host는 인터넷 호스트 도메인 이름이나 ip adress를 뜻한다고 한다. 즉 저 위의 경우에는 localhost 또는 IP주소인 127.0.0.1을 의미한다. host:port로 구성 되어있는데 port는 생략할 수 있다. 생략한 경우에는 port를 80번으로 설정해 연결된다. 그리고 port 뒤에 abs\_path가 비어있다면, “/”로 대체되어 들어간다.

Connection은 서버와 클라이언트의 연결상태를 의미하는데, keep-alive는 현재 socket이 닫히지 않고 계속 연결되어 request와 response를 전달할 수 있다는 것을 말한다.

User-Agent는 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.163 Whale/2.7.99.20 Safari/537.36로 보여지는데, request를 보낸 user agent에 대한 정보를 담고 있다. 이는 통계적인 목적, 프로토콜 위반 추적, 그리고 특정 user agent limitation들을 피하기 위해서 존재한다.

Accept-Encoding는 response에 허용하는 content-coding 값을 제한한다.

Accept-Language는 accept와 비슷하지만 request에 대한 response로 선호되는 자연어를 제한한다.

Part II.

**서버 디자인**

서버의 디자인은 맨 처음에는 서버가 소켓을 만들어주고, bind를 해준다.

그리고 listen()을 통해 클라이언트의 연결을 기다린다. accept()가 클라이언트의 연결이 들어올 때 까지 blocking 되어있다가 연결 요청이 오면 accept를 실행한다. 이 때 반환된 클라이언트 소켓 파일 디스크립터를 저장해준다. 그리고 서버는 read를 통해 newsockfd로 전송된 request 메시지를 읽게 된다.

메시지 중 첫 줄인 GET /file HTTP/1.1에서 file을 따로 분리해서 일단 그 파일이 서버에 존재하는지 access를 통해 검사한다. 만약 존재한다면 file의 확장자 역시 strtok를 통해 분리해 ext로 저장해준다. 그리고 FILE stream으로 해당 파일을 fopen 해준다. 각각 html, jpeg, mp3, gif, pdf 마다 Header에서 각각 text/html, image/jpeg, audio/mpeg, image/gif, application/pdf로 다르게 Content-type을 지정해주고 우선 먼저 send를 통해 해당 header를 보내준다. 그리고 fopen으로 열린 파일을 loop를 돌며 fread를 해서 buffer에 저장한 후 client에게 send를 해준다.

작업이 모두 끝난 후에는 FILE steram을 닫아주고, request에 대한 response를 모두 보내주었으니 newsockfd 역시 닫아준다.

그리고는 위를 반복해준다.

스크린샷, 쥐고있는이(가) 표시된 사진

자동 생성된 설명

도식은 간단히 다음과 같다.

**구현시 어려웠던 점**

C언어를 많이 사용해보지 않아서 문자열을 다루는데 어려움이 있었다. 구글링과 따로 학습을 통해 char 배열과 char pointer의 사용법과 string을 다루는 함수들에 대해 익숙해 진 후 접근하니 한결 수월했다.

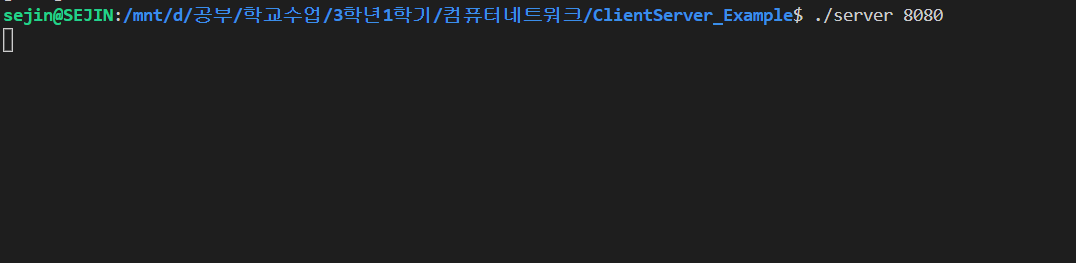
html파일을 전송할 때는 괜찮았지만 jpg파일을 전송할 때 클라이언트에서 요청한 사진이 나오지 않고 검은 네모사진 같은 쓰레기 값이 넘어가서 한참을 고민하다가 단순히 fgetc로 char 단위로 가져오는 방법에서 fread로 읽어와 buffer에 저장한 후 send를 해주니 문제가 해결되었다.

그리고 처음에는 Header를 작성할 때 Content-length를 임의로 정해서 넣었더니 그 length만큼만 출력이 되어 문제가 생겨 아예 그 요소를 제거했더니 해결되었다.

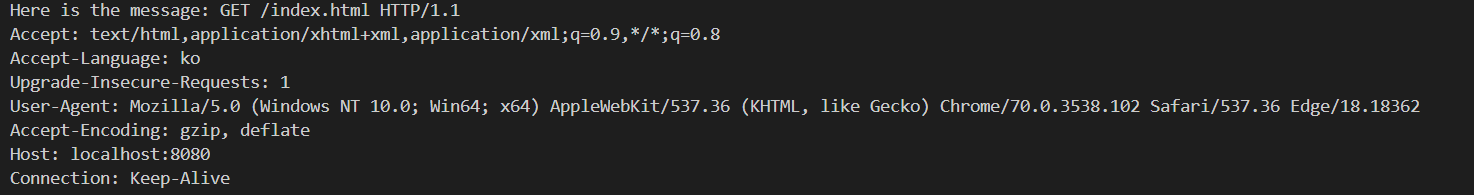
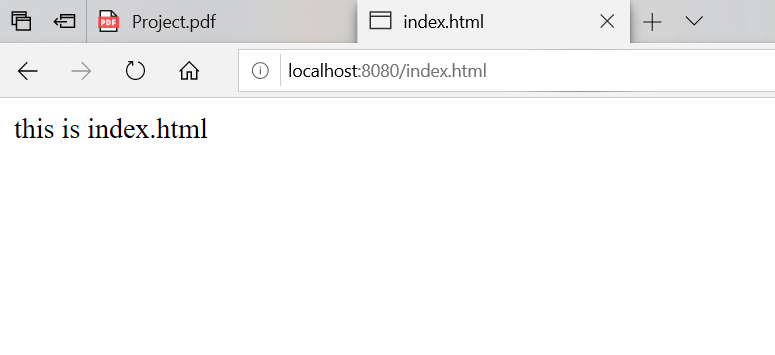
그리고 모두 같은 방법으로 전송하고, Content-Type만 다르기 때문에 그 부분만 조건문을 통해 넣어주려 하였는데, 정상적으로 작동하지 않아서 그냥 모두 header를 새로 선언해주었다.

**동작예시**

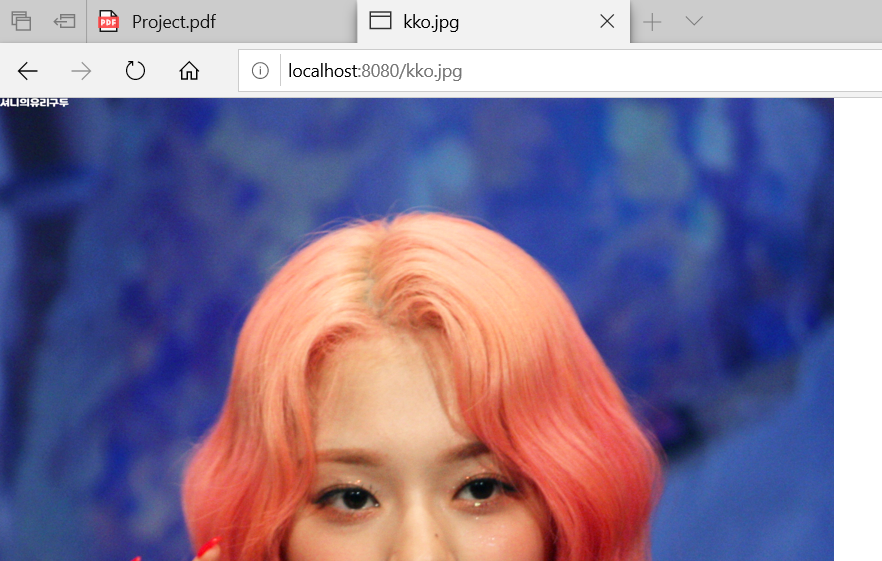
1. **초기상태**

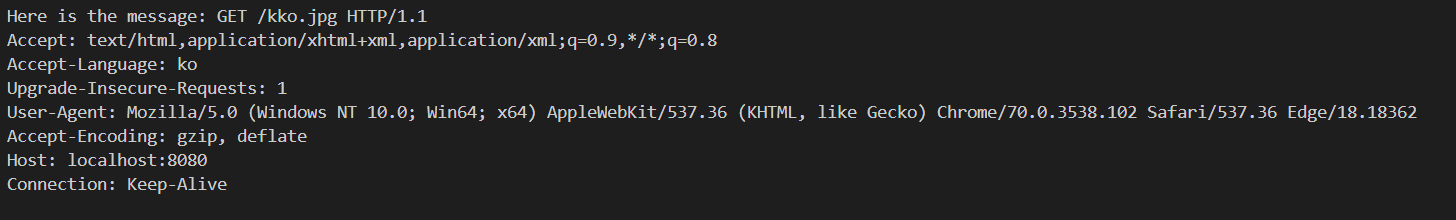
****

1. **Index.html 요청시**

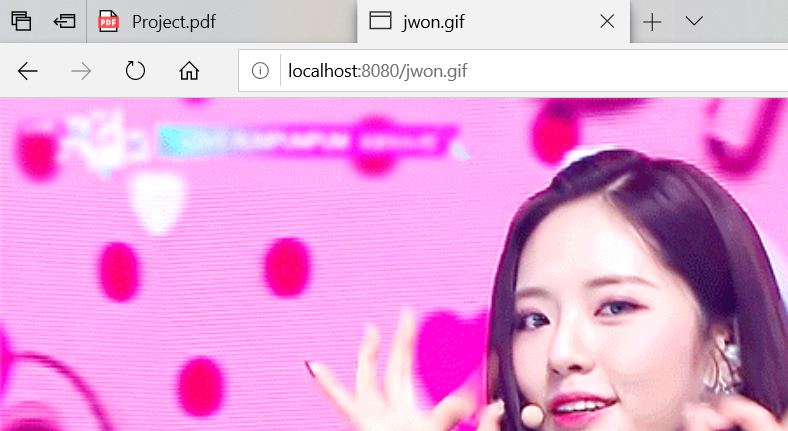
****

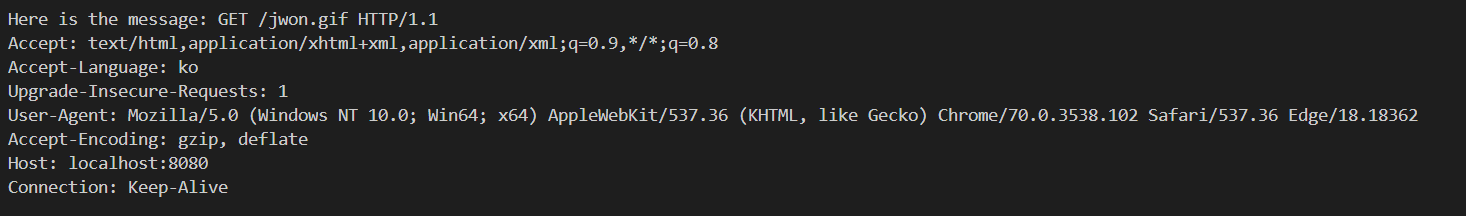
1. **kko.jpg 요청시**



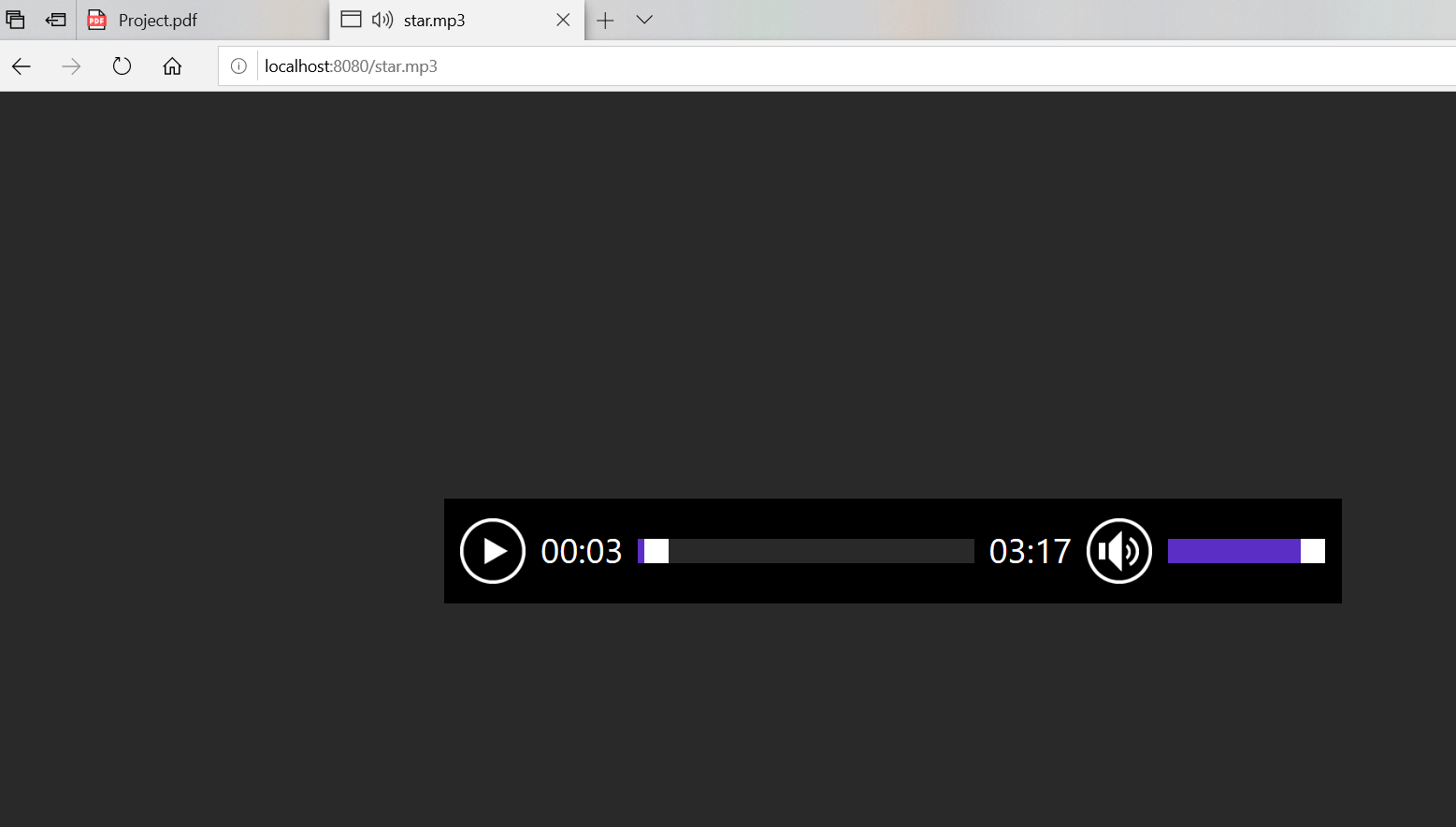
****

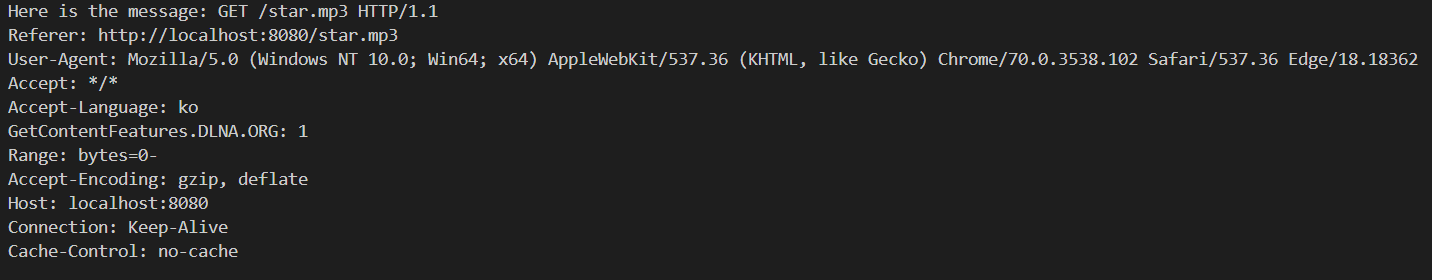
1. **jwon.gif 요청 시**

****

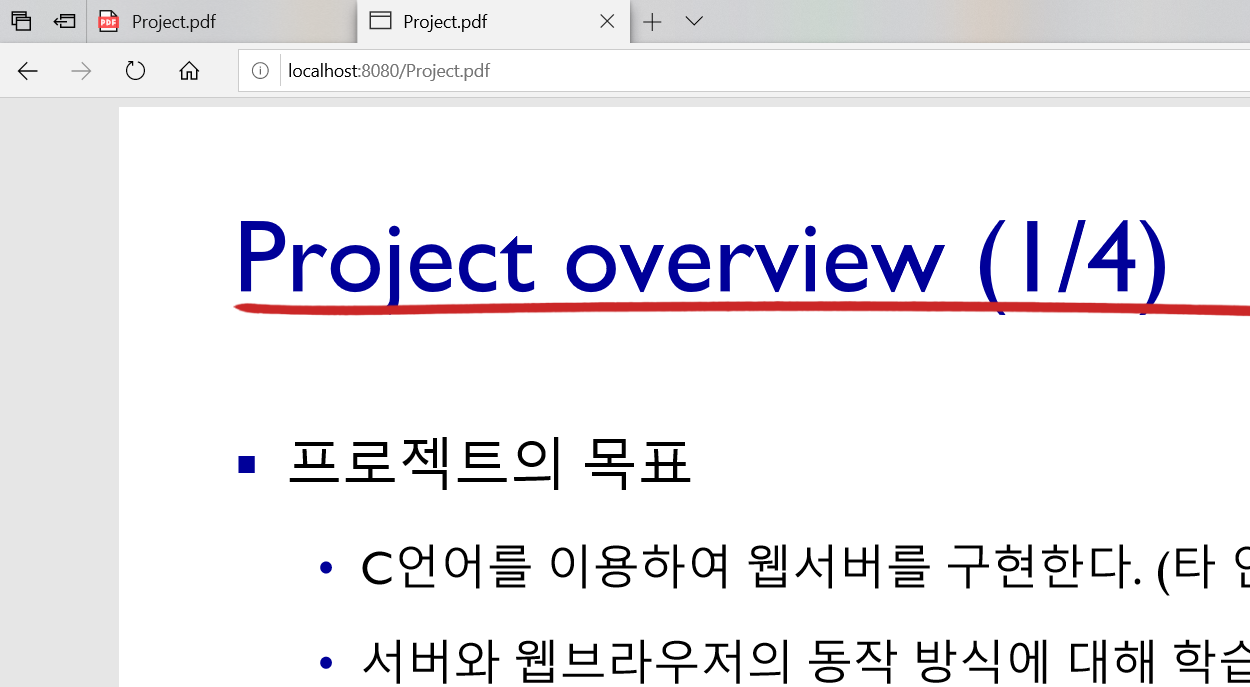
****

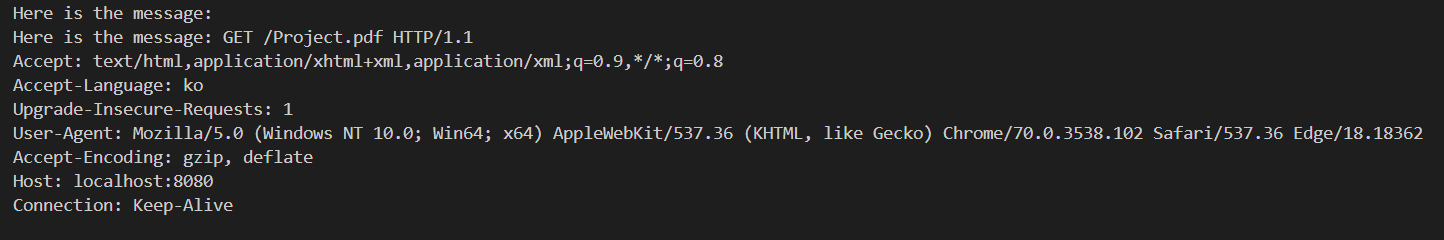
1. **star.mp3 요청시**

****

****

1. **Project.pdf 요청시**

****

****