2019. 4. 12

OPEN SOURCE SOFTWARE



DDok-Ddok

한국인을 위한 멀티 플랫폼 스트리밍 용 챗봇

TEAM 12 Final Report

20145523 김상헌, 20146290 김성민

20145822 김영현, 20141261 송제웅

20141001 이지호

1. **프로젝트 주제**

**스트리머에게 필요한 다양한 기능과 욕설 채팅 필터링을 강화한 한국형 오픈소스 스트리밍 챗봇 서비스 개발.**

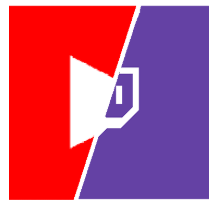


FIGURE 1. DDOK DDOK의 로고

1. **프로젝트 소개**
   1. **프로젝트의 필요성**
      1. 최근 1인 개인방송의 인기가 급증하면서 다양한 플랫폼들을 통해 많은 사람들이 개인 방송을 송출하고, 또 시청하고 있다. 이러한 시대적인 흐름에 맟춰서 유튜브, 트위치와 같은 여러 개인 방송 플랫폼들의 시장이 커지고 있고, 더 많은 시청자들을 끌어들이기 위해 여러 기능들을 내보이고 있다.
      2. 개인방송을 진행하는 스트리머 혹은 유튜버는 자신의 방송을 시청하는 시청자들과의 소통을 방송을 유지함에 있어 굉장히 중요하다고 생각한다. 하지만, 시청자의 수가 늘어남에 따라 수많은 채팅이 오고 가기 때문에 일일이 모든 채팅에 대한 피드백을 주기 어렵다. 스트리머는 이러한 상황을 악용하여 욕설 및 도배 등의 악질적인 채팅을 하는 시청자들에 의해 방송에 피해를 입게 된다.
      3. 위 문제를 해결하기 위해 개인 방송 플랫폼에서는 채팅과 관련한 API를 제공하고 있으며, 스트리머들은 API들을 이용한 NightBot, 싹둑 등과 같은 ‘채팅 봇’을 사용하여 채팅창의 관리를 진행한다. 하지만, 성장하는 한국 개인방송 시장에 비해, 활발하게 쓰이는 채팅 봇들은 한글에 대한 지원을 하지 않거나 채팅 관리 기능이 부실하거나 하는 등의 문제점들이 있다.
   2. **목적**
      1. 위에 설명한 필요성에 따라 우리는 새로운 채팅 봇을 제작하기로 결정하였다. 프로젝트에서 구현할 새로운 채팅 봇은 한글을 지원하고, 기존 봇들이 지원하지 않는 욕설 필터링 및 도배 방지 등의 기능을 제공할 것이다.
      2. 그 기능들 중 욕설 필터링의 강화를 중점적인 목표로 설정하였다. 악질적인 채팅을 하는 시청자들에 대한 피드백과 그 시청자들에 대한 처벌(타임아웃 또는 밴 등)을 방송을 진행함과 동시에 확인할 수 있는 프로그램을 만드는 것이 주된 목적이다.
      3. 프로젝트를 오픈소스화하여 버전을 관리하고, 프로젝트를 유지 및 보수할 것이다. 그리고 유저들의 Contribution을 통해 챗봇 기능에 대한 수요와 1인 스트리밍 시장에 대한 정보를 확인하고, 우리 프로그램의 기능을들 추가 및 보완할 계획이다.
2. **관련 프로그램 소개 및 차별성**

위의 필요성 항목에서 이미 언급했듯이, Nightbot이라는 전세계적으로 많이 활용되고 있는 프로그램이 이미 시장에 존재했다. 그러나 세밀한 조사를 진행한 결과, 시장에 있는 제품들 모두 다른 특징들이 존재함을 발견하였고, 이번 프로젝트를 진행하면서 제품마다 존재하는 각각의 단점들을 모두 보완하며 새로운 기능을 추가하여 기존 채팅 봇과의 차별점을 가지고자 하였다. 기존 채팅 봇 프로그램들의 특징 및 단점은 다음 Figure 2와 같다.

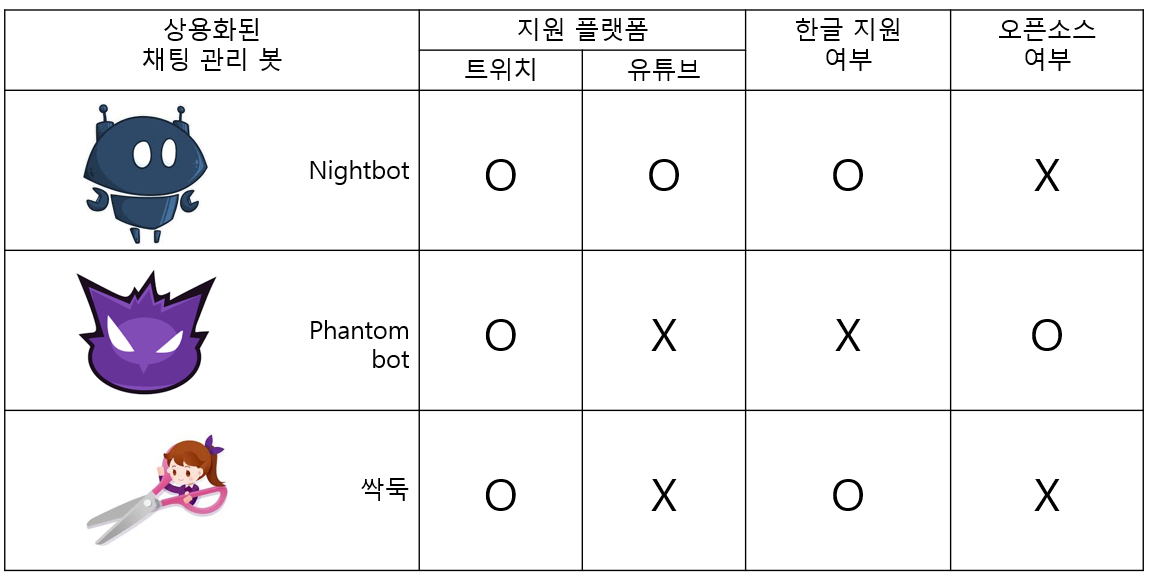


FIGURE 2. 기존 채팅 봇의 특징 및 문제점

* 1. **기존 채팅 봇 프로그램의 단점**.
     1. Figure 2에 존재하는 3개의 프로그램은 가장 많이 쓰이는 대표적인 챗봇들에 대해 조사를 하고 정리한 표이다. NightBot의 경우 Github Repository가 존재하나 프로그램 사용과 관련한 API 문서들만 존재하고 있고 소스 코드는 공개하고 있지 않아 우리 프로젝트와는 큰 차이점을 갖고 있다.
     2. Phantom Bot의 경우 Github에서 twitch chatbot이라는 카테고리로 검색 시, 가장 많은 스타를 보유하고 있다고 나오는 프로젝트인데, 북미권에서 자주 사용되는 플랫폼들에 대한 지원만 하다보니 한국에서 많이 사용하는 유튜브에는 따로 챗봇기능을 제공하지 않고 있으며 한글 명령어 혹은 한글 채팅에 대한 피드백 기능을 제공하지 않고 있어 우리 프로젝트와는 큰 차이점을 갖고 있다.
     3. 싹둑의 경우 한국 BJ들에게 유명한 한글 지원 채팅 봇으로 도표에 나와있듯, 트위치만 지원하고 있고 오픈소스를 통한 코드 공개를 하지 않아 우리 프로젝트와는 큰 차이점을 가진다.
  2. **차별점을 둔 새로운 기능들**
     1. 비속어 필터링과 관련한 기능이 추가된다. 한국인 스트리머들을 대상으로 한 프로젝트이다 보니, 다양한 언어권의 비속어가 아닌, 흔히 접할 수 있는 모국어의 비속어들의 필터링 기능을 넣어 채팅창 관리에 도움이 되게 할 것이다. 특정 유저가 지속적으로 비속어를 사용하게 되면 경고 누적 후 스트리머에게 밴을 추천하도록 하는 기능을 추가할 것이다.
     2. 채팅창 인원관리 기능이 추가된다. 한국인 스트리머들을 대상으로 하다 보니 비슷한 부류의 소위 말하는 ‘네임드’라는 유명인사들을 한정 시킬 수 있다. 이러한 점을 활용하여 채팅창에 네임드가 나타나면 방송을 진행하는 스트리머에게 알려 적절한 리액션을 통해 방송을 보는 시청자들에게 흥미를 유발할 수 있다는 차별점을 가지게 된다.

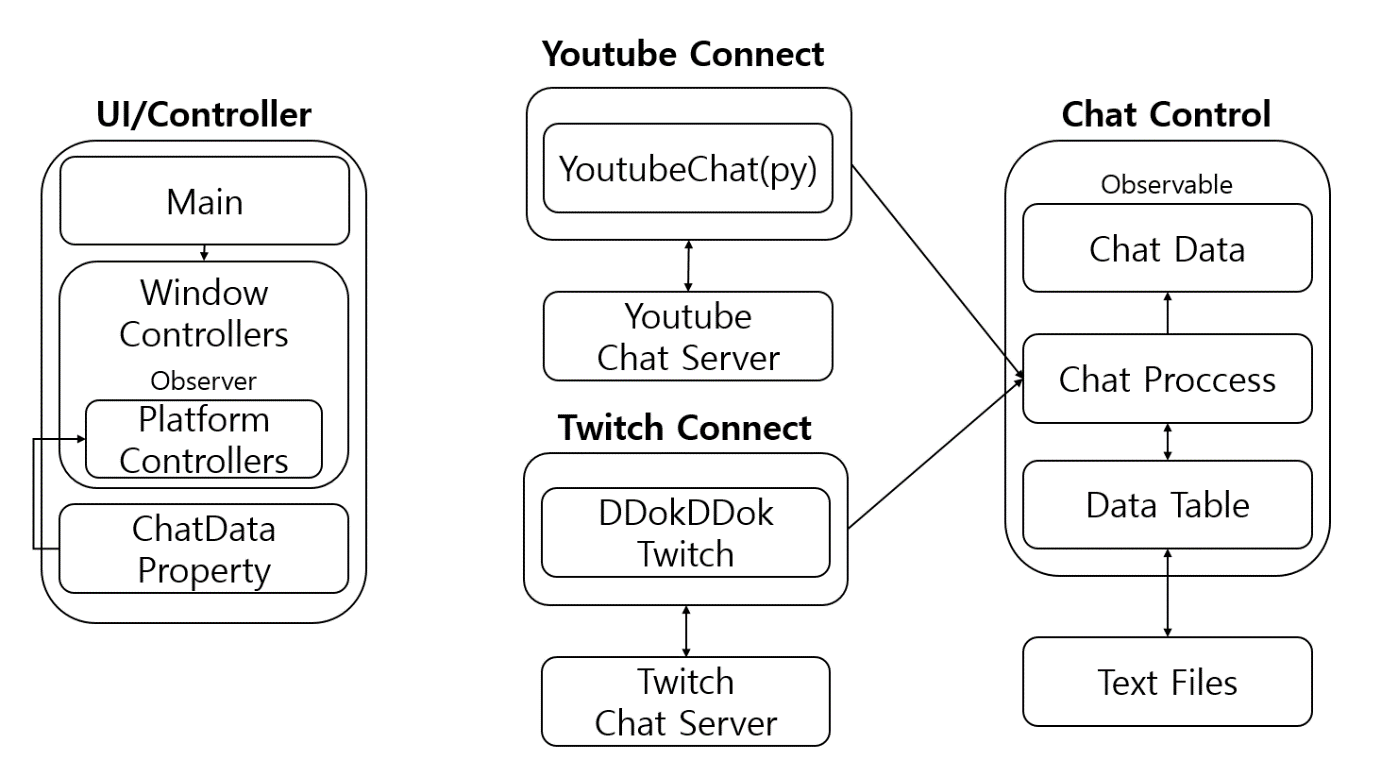
1. **구현물에 대한 설명**
   1. **아키텍처**

Figure . 프로그램 아키텍처

* 1. **Workflow**
     1. Twitch : 입력 후 트위치를 선택하고 봇이 OAuth 인증을 통해 IRC Chat Server로 접속한다. 접속 한 후, 챗봇 관리 Dashboard에서 지속적으로 IRC서버에 chatlist를 받아오는 request를 보내고, 각 유저가 보내는 chat에 대해 비속어인지, 명령어인지 parsing을 진행한다. 그 이후, chat parser가 각 chat을 처리한 후 스트리머에게 비속어인지, 유명인이 입장하였는지 등을 알려준다. 명령어의 경우 사용자/개발자/스트리머가 사전에 해둔 설정에 의해 적절히 반응한다.
     2. Youtube : 메인 대쉬보드에서 Youtube Chat ID를 입력한 후 유튜브를 선택하고 봇이 Google OAuth 2.0인증을 통해 Youtube LiveStreaming Chat Server로 접속한다. 그 후, 일정한 간격으로 API Functin Call을 통해 Chatlist를 받아온 후, 각 Chat에 대해 Parsing을 진행한 다음 비속어와 명령어에 대한 처리를 진행한다.

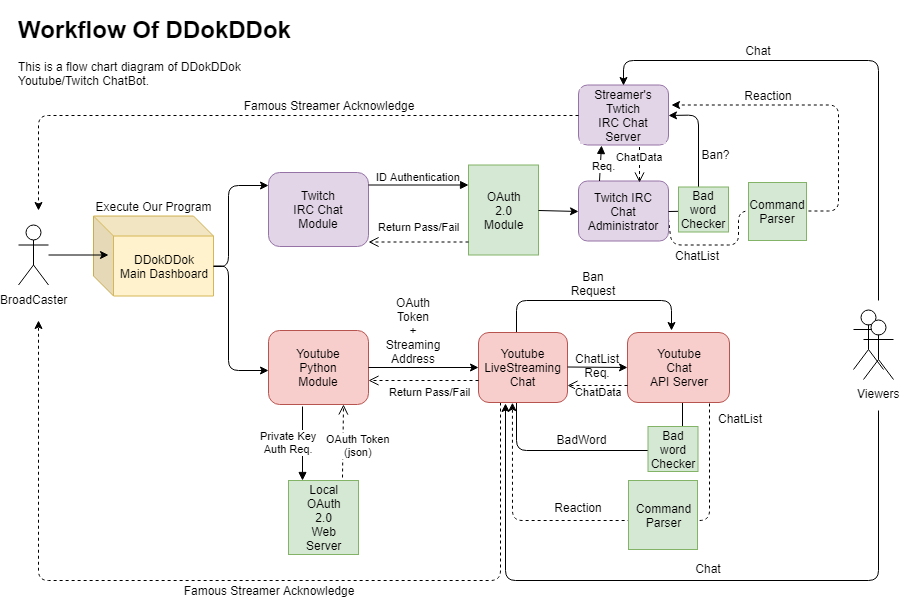


Figure 4. 프로그램의 Workflow

* 1. **핵심 기술**
     1. 욕설검사

먼저 우리 프로젝트에서 스트링 처리에 관한 특성은 다음과 같다.

* + - 1. 매 초당 들어오는 입력 채팅의 수가 매우 많아질 수 있으며, 최악의 경우에도 이 모든 입력에 대한 처리가 새로운 입력이 들어오기 전에 끝나야 한다.
      2. 입력 스트링 검사 시 한 번만 일치되면 그 뒷부분은 검사할 필요 없이 욕설로 분류된다.
      3. 욕설 키워드의 첫 글자들이 뭉쳐있는 경향이 있으나, (개, 씨, 섹 등..) 동일 글자로 시작하는 욕설의 개수는 중복을 제외하면 30개를 넘지 않는다.
      4. 욕설 키워드는 default 약 2000개이나, 사용자가 임의로 추가/변경함에 따라 무한정 증가할 수 있다.
    1. 일반적인 단순 반복문을 이용한 스트링에서의 키워드 디텍팅은 한 개의 입력 당 input string의 길이(n) \* 키워드 개수(m)만큼의 스트링 비교 연산을 수행하게 된다. 트위치의 최대 입력 제한이 500자, 우리의 욕설 키워드가 기본 2000개 + 사용자 지정 @임을 고려했을 때 한 개의 채팅 당 최대 10만 번 이상의 스트링 비교 연산이 수행될 수 있다는 것이다.
    2. 스트링 처리에 관한 특성 1에 의해 우리는 새로운 키워드 디텍팅 알고리즘을 적용해야 할 필요를 느꼈고, 다음과 같은 방식을 고안했다. 키워드를 첫 글자별로 묶은 디렉토리를 생성하여 저장을 해 두고 해당 디렉토리를 찾아가 그 안에서만 검색을 하는 방식으로, 스트링의 한 글자를 디렉토리에 매칭시키는 과정(directory search)이 추가되기는 하나 특성3)에 의해 최악의 경우 시간 복잡도가 약 1/60 미만의 수준으로 줄어들게 될 것으로 예상되었다.

그러나 사용자가 새로운 필터링 단어를 추가할 수 있는 기능(스트링 처리에 관한 특성4)을 적용함에 따라, 디렉토리의 개수가 유니코드의 범위인 65536까지 증가할 수 있는 일말의 가능성이 존재한다. 따라서 우리는 directory search에 따른 오버헤드를 제거하기 위하여 가상 메모리 기법에서 메모리 주소를 이용한 직접 접근을 모티브로 다음 방식을 고안하였다. 유니코드의 범위인 65536개의 ArrayList<String> 배열을 생성하고 단어의 첫 글자를 유니코드로 바꾸어 배열의 인덱스로 사용한다. 해당 인덱스데 대한 접근 방법은 다음과 같다.

**Arr[(int)’안’] : ‘안’으로 시작하는 욕설들의 ArrayList가 존재**

* + 1. 입력 스트링의 한 글자마다 위와 같은 방식으로 접근하여, 만약 해당 주소의 ArrayList의 길이가 0이라면 해당 글자로 시작하는 욕설이 없다는 의미이므로 작업을 중지하고 다음 글자로 넘어가게 된다.

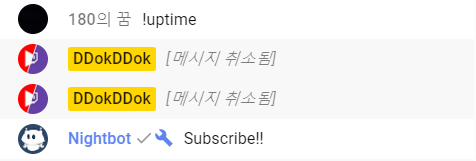
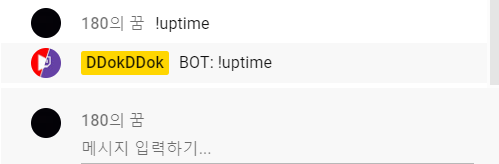
이러한 방식은 공간 복잡도를 증가시키기는 하나 directory search의 시간 복잡도를 상수시간으로 줄이는 것이 가능함에 따라 우리 프로젝트의 특성상 실시간 처리에 대한 이점이 더욱 크다 판단하여 채택하였다. 또한 채팅 검사 뿐만 아니라 사용자가 새로운 필터링 단어를 추가할 때도 directory search가 필요 없도록 해 주는 장점이 있다.

위와 같은 극단적인 경우에 대비하여 알고리즘을 구현하였으나, 실제 보통 채팅의 경우 앞서 언급하였던 프로젝트의 특성상의 실시간 처리에 대한 이점과 스트링 처리에 관한 특성 2 에 의해 수행되는 연산의 회수가 최악의 경우에 비해 매우 낮다고 판단되어진다.

* + 1. 이 알고리즘의 구현을 위해 DataTable의 BWTable 자료구조를 생성하였으며, 해당 처리는 “Figure 3” 및 “4. A. 아키텍처”에 서술한 ChatProc 클래스에서 수행된다.
  1. **Implementation 이후로 개선된 부분**

1. **최종 구현물**
   1. **성능 비교 및 평가**
      1. 반응속도

Figure 5. Night Bot과 똑똑의 반응속도 비교



위의 “Figure 5”는 유튜브 스트리밍 상황 중, ‘똑똑’과 ‘Night Bot’을 동시에 입장시키고 반응을 확인한 것이다. 왼쪽의 사진은 명령어를 입력한 지 1초가 지나지 않아 ‘똑똑’이 반응을 한 모습이며 그 후 1분 30초정도가 소요된 후 ‘night bot’이 반응을 한 것을 오른쪽의 사진에서 확인할 수 있다. 반응속도의 차이는 유튜브 스트리밍 환경 쪽에서 훨씬 크게 나타났으며, 트위치의 경우 5초정도의 차이로 역시 ‘똑똑’이 빠르게 반응했다

* + 1. 차별화된 필터링 기능

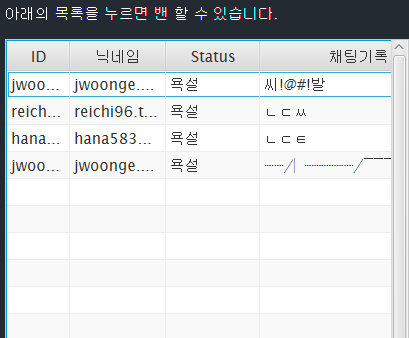
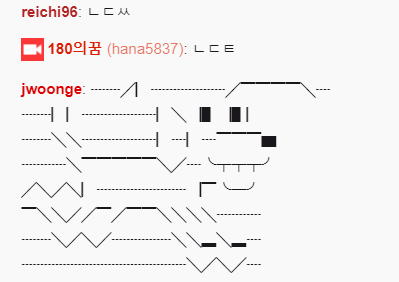
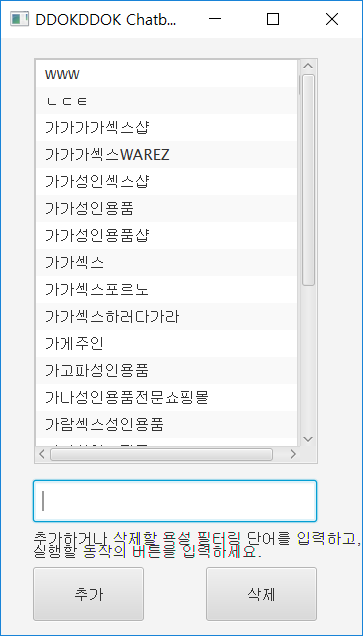
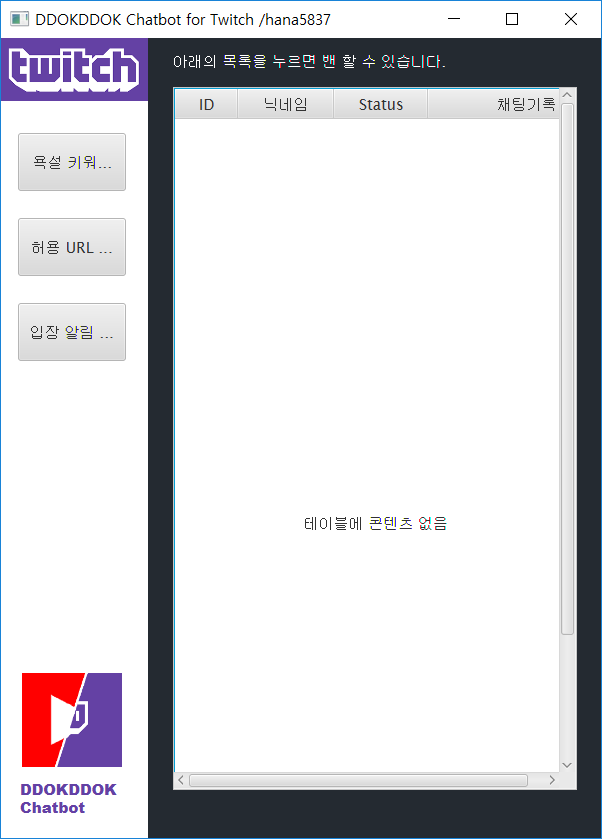
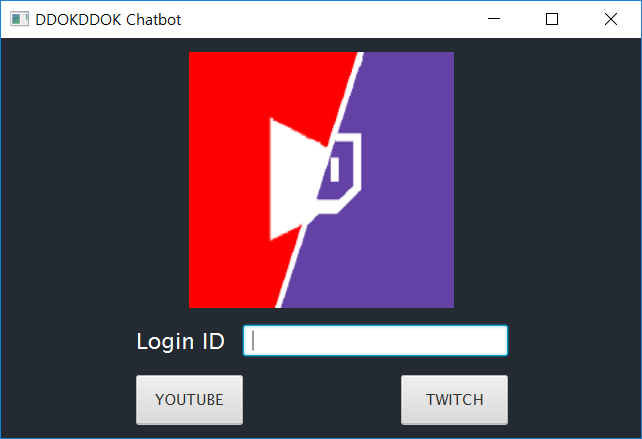


Figure 6. 욕설 및 스팸 이모티콘 적발

“Figure 6”는 Twitch 방송중 똑똑이가 사용자가 등록한 한글 욕설, 스팸 이모티콘을 적발한 모습이다. 각각 밴리스트에 등재되어 있는 것을 확인할 수 있으며, 욕설 사이에 숫자, 특수문자를 입력해 밴을 피해가고자 했던 사람도 적발된 모습이다. 이는 ‘싹둑’, ‘phantom bot’ 등 다른 챗봇이나 트위치에서 기본으로 제공하는 욕설 필터링에서 제공하지 않는 기능이고 이 모든 과정은 시간을 1초 이상 소요하지 않고 즉각적으로 처리된다.

* + 1. UI

Figure 7. UI 캡처 화면



똑똑의 UI는 “Figure 7”과 같다.

* 1. **데모**

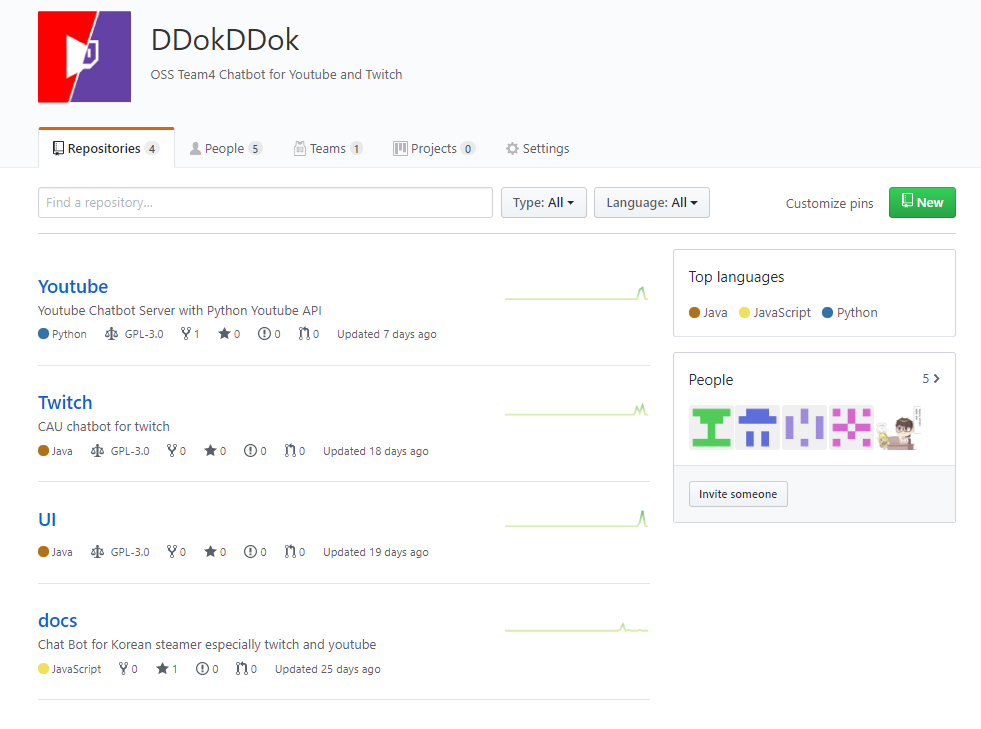
1. **Github**
   1. **Repositories**

Figure 5. Github Organization

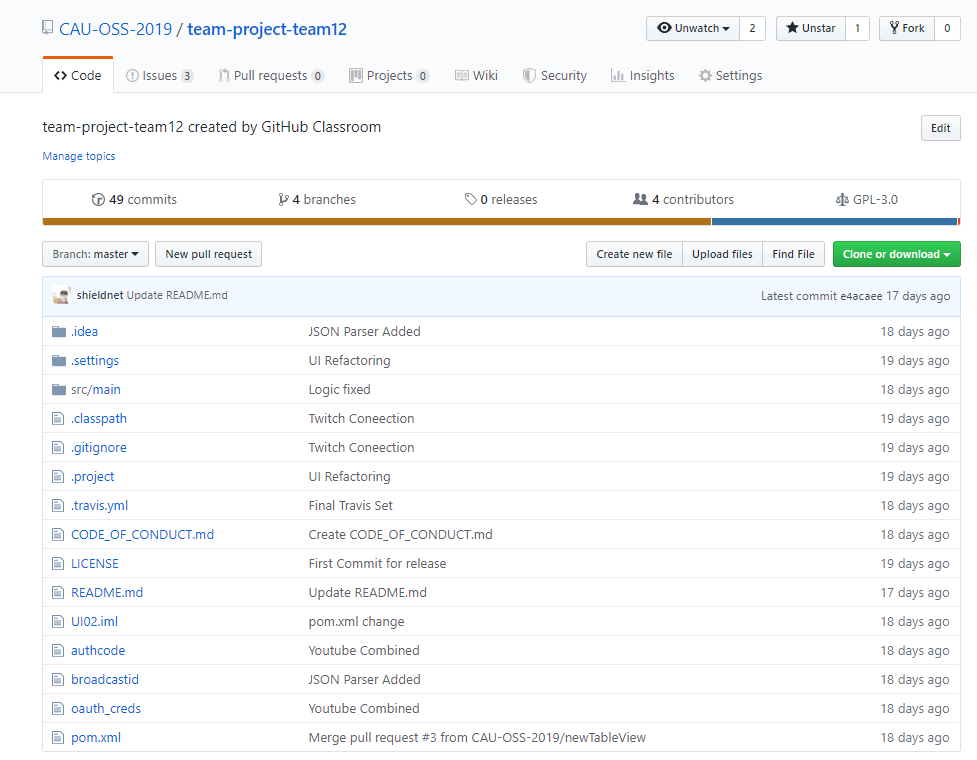
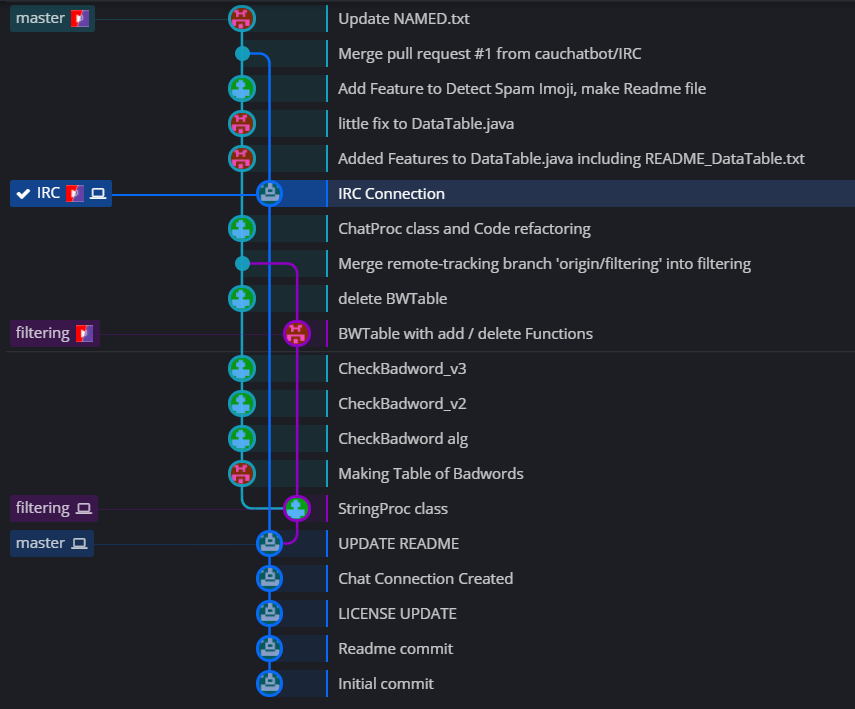
프로젝트 초기에 기존에 있는 유사한 오픈 소스인 Phantombot을 참고하였는데, 당시 설계로는 다른 플랫폼인 Twitch와 Youtube용 application을 따로 만들어야 할 것으로 판단되었다. 그래서 Github의 organization기능을 사용하여 위의 사진과 같이 Twitch/Youtube/UI/docs 로 repository를 나누어 개발을 진행하게 되었다. 개발을 진행하다 보니, UX 관점과 개발의 편의성이라는 점에서 하나의 Application에서 사용을 원하는 플랫폼을 선택하게 하는 방식이 더 합리적이라고 판단되어 추후에는 Figure 5의 모습처럼 프로젝트용 repository에 통합된 코드를 올려 개발을 진행하게 되었다.

Figure 8. 통합 Repository

* 1. **Twitch implementation**

Figure 9. Twitch Repository Commit Log



초기의 Twitch repository에서 commit 내용이다. 좌측의 사진은 Github 홈페이지에서 Insights의 contributors를 통해 나타낸 모습이고 우측의 사진은 Git Kraken 이라는 3rd party 제품을 사용하여 Branch와 커밋 내용을 시각화하여 나타낸 모습이다. Comnamu18이라는 아이디를 가진 김상헌 학우는 Twitch와의 연결을 담당하여 IRC 서버와 연결하는 작업을 따로 branch를 만들어 작업을 진행하였고 jwoonge아이디를 가진 송제웅 학우와 jiho 아이디를 가진 이지호 학우가 chatting filtering 관련 작업을 따로 branch를 만들어 진행하였다.

* 1. **Youtube implementation**

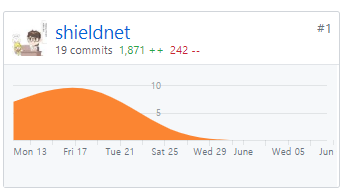
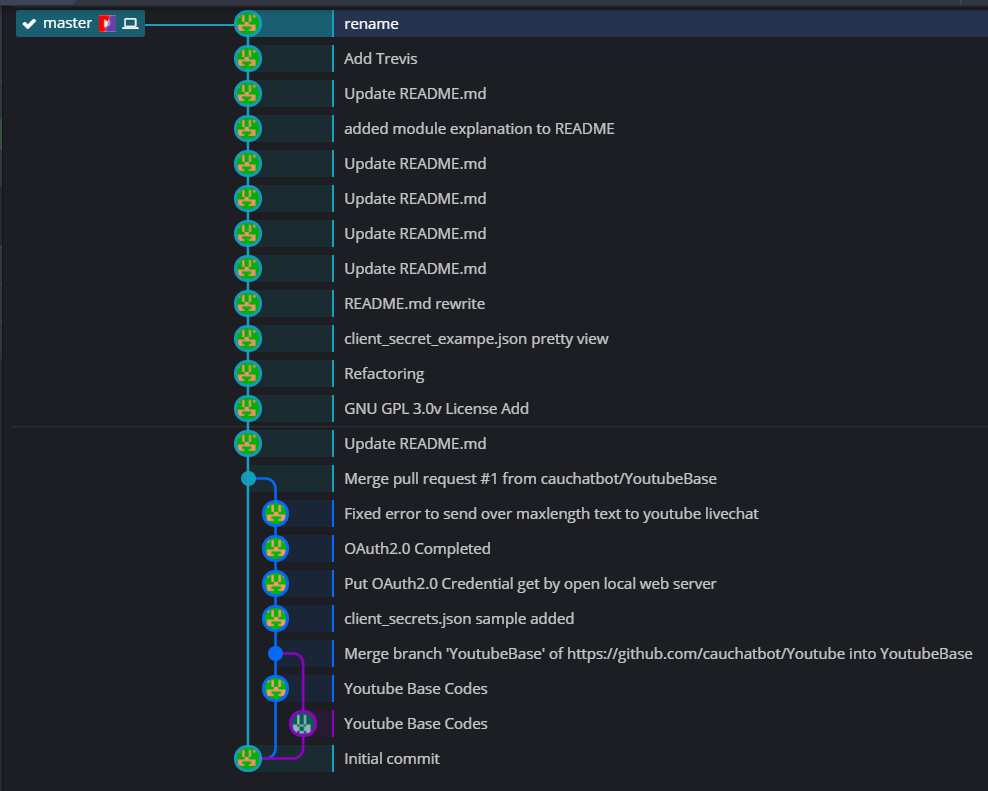


Figure 8. Youtube Repository Commit Log

Youtube repository에서는 shieldnet 아이디를 가진 김성민 학우가 Youtube API를 사용하여 Youtube 채팅방과의 연결부분을 담당하여 작업을 진행하였다. 초기엔 자바 API를 사용하려 하다 추후에 Python API로 변경하게 되어 예상보다 많은 작업이 진행되었다.

* 1. **UI implementation**

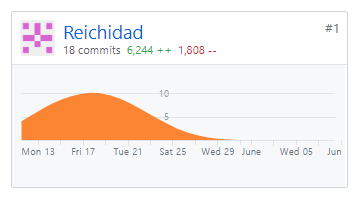
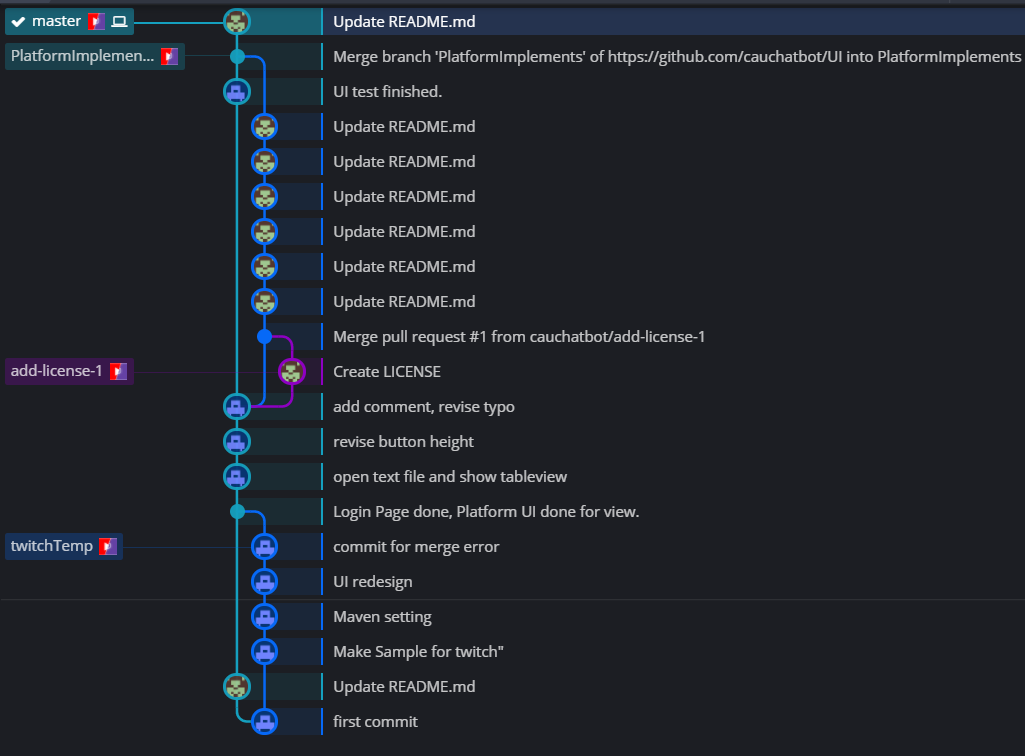
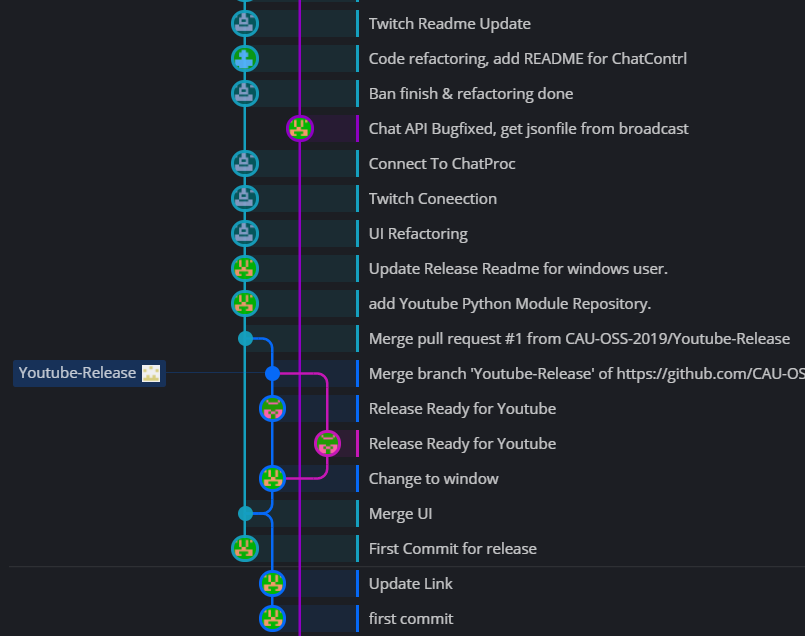
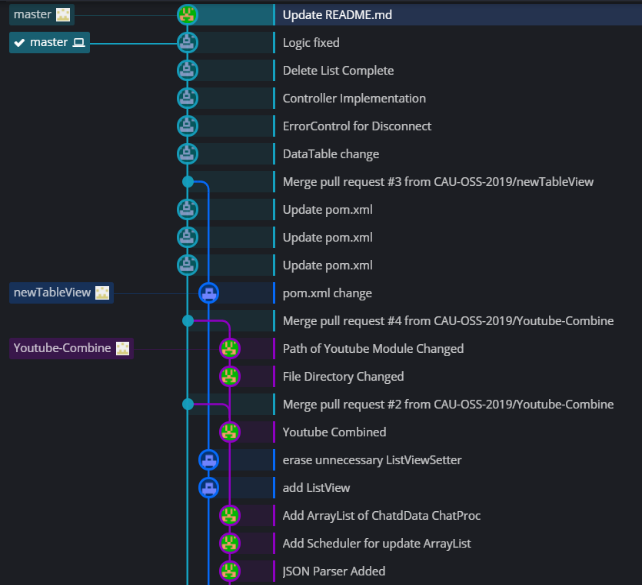
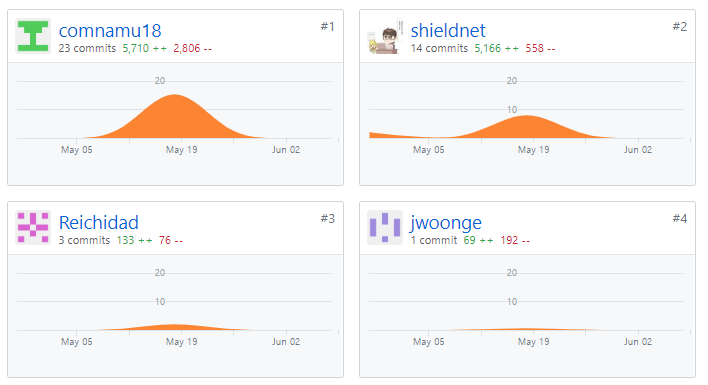


Figure 10. UI Repository Commit Log

UI는 Reichidad라는 아이디를 가진 김영현 학우가 맡아 작업을 진행하였다. 개인 PC에서 작업을 진행하던 와중, Git bash에는 다른 닉네임(YeongHyeonKim)으로 등록이 되어 있어 Git Kraken의 커밋 로그에는 2사람이 작업한 것처럼 보이지만, 좌측을 보면 알겠지만 김영현 학우 혼자 GUI부분을 맡아 작업을 진행했다는 사실을 확인할 수 있다.

* 1. **Final Integration for implementation**

Figure 11. CAU-OSS-2019/team-project-team12-repository Commit Log



추후 CAU-OSS-2019/team-project-team12 repository에 통합된 프로젝트를 올리고 난 후의 커밋 상황이다. 전반적으로 일이 빠르게 끝난 트위치 연결파트를 담당한 김상헌 학우가 UI와 chat filtering 파트 부분의 코드를 합치는 역할을 맡았으며 GUI와 컨트롤러 사이의 통신을 위해 서버와의 연결을 담당한 김상헌, 김성민 학우와 GUI파트인 김영현 학우의 commit이 상대적으로 많은 모습을 확인할 수 있다.