오픈소스SW기여 최종보고서

프로젝트 명 : webConvertor

이름	고현우 / 소완열
학과	소프트웨어 공학과
학번	32170171 / 32172110
과목명	오픈소스SW기여
분반	1분반
교수님	송인식 교수님
제출일자	2023 / 06 / 13

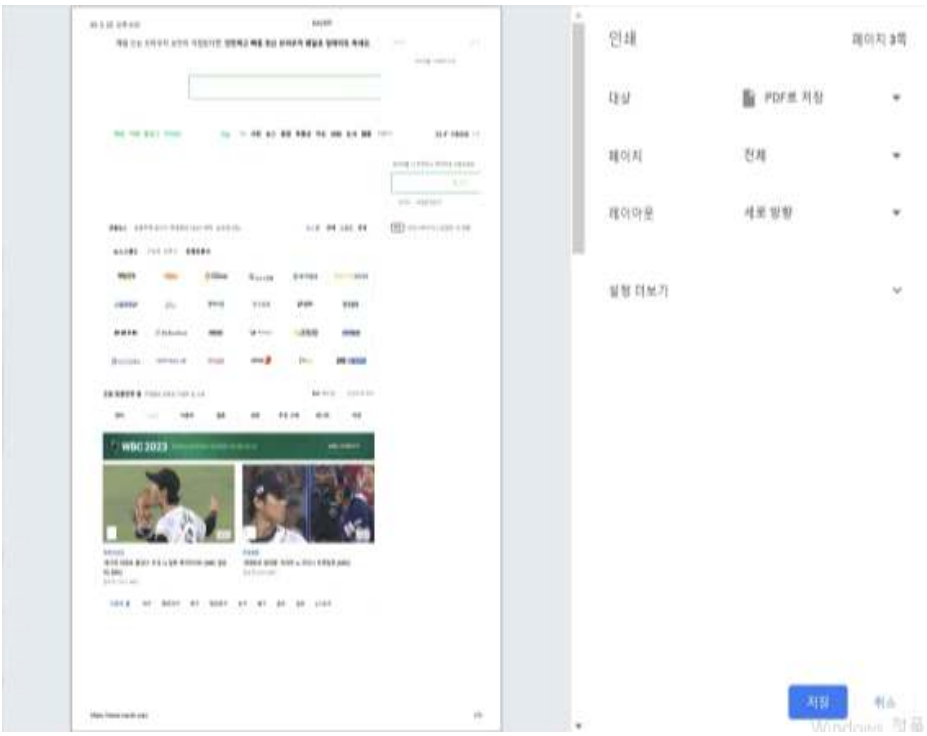
목 차

1. 프로젝트 개요

(1) 프로젝트 진행 배경

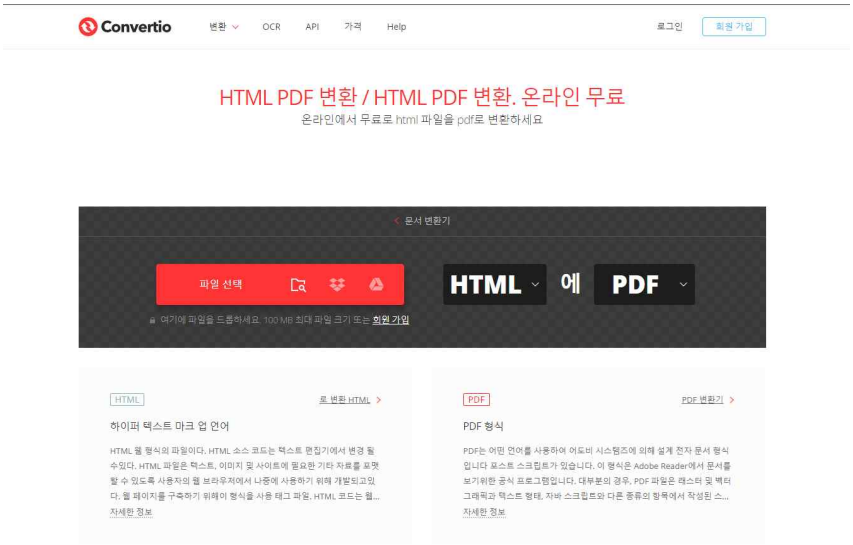
기존의 로컬 파일(html, xml)과 웹 URL을 PDF로 변환하려면, 사용자가 직접 웹을 열어 'PDF로 인쇄하기' 기능을 사용했어야 합니다.

즉, 이러한 PDF 변환 작업은 사용자 의존성이 높고, 서버에서 진행할 수 없는 작업이었습니다.



<그림 1. 기존의 웹페이지 -> PDF 변환 방법 >

또한, 기존의 WEB to PDF 변환을 제공하는 서비스와 솔루션 제품은 사용자가 직접 접속하여 수행하는 방식 뿐이었습니다.



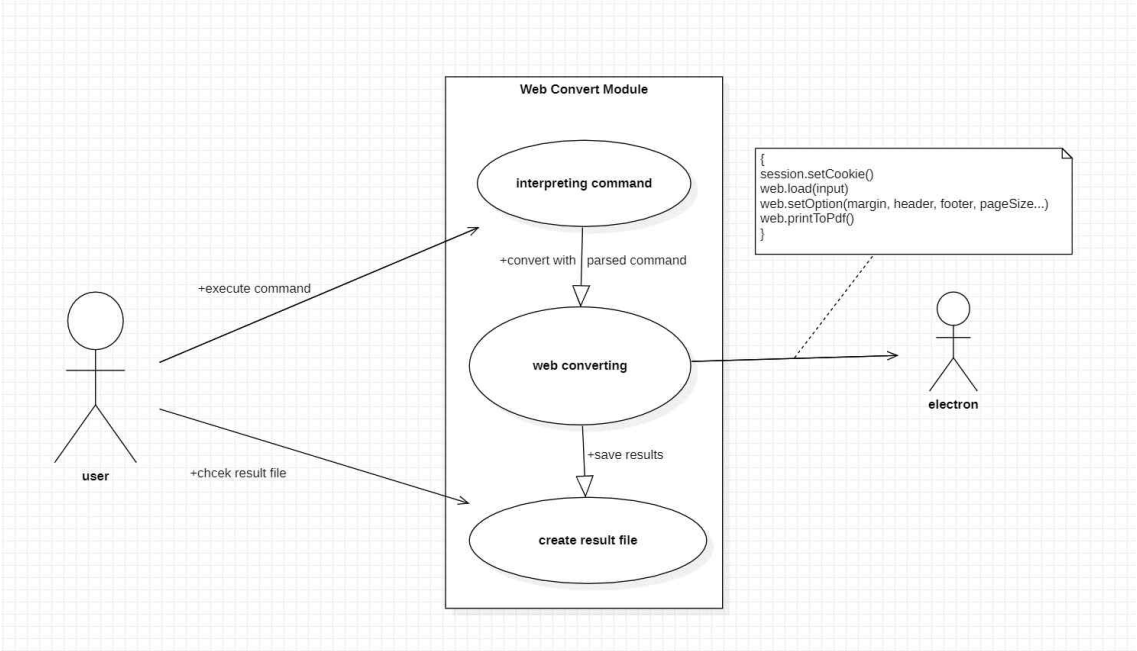
<그림 2. 국내 솔루션 제품 >

저희는 위와 같이 사용자 의존적인 프로세스를 자동화 할 수 있도록 지원하는 콘솔 어플리케이션을 개발하고자 합니다.

이러한 기능을 제공한다면, 이를 활용한 다른 오픈소스 프로젝트의 파급 효과 또한 기대할 수 있을 것으로 예상됩니다.

<그림 3. 어플리케이션 사용 예시 >

(2) System Sequence Diagram



<그림 4. useCase>

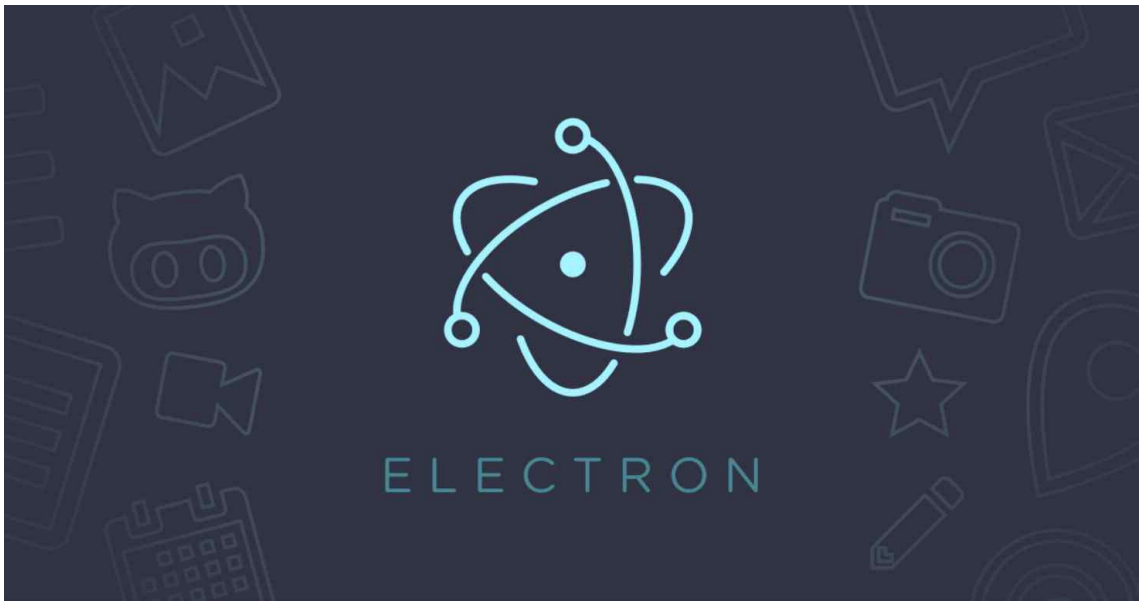
어플리케이션 사용 유스 케이시의 시퀀스 다이어그램입니다.

해당 그림에서 확인할 수 있듯이 사용자는 커맨드 실행 및 결과 파일의 생성 여부만을 확인하며, 이외의 다른 액션은 필요하지 않습니다.

또한 어플리케이션에서는 입력받은 커맨드 라인으로부터 필요한 설정 데이터를 파싱하여 데이터 세팅 후 PDF로 변환을 진행합니다.

2. Eletron 활용 기능

(1) Electron 이란?



Eletron은 Javascript, HTML, CSS를 이용하여 Desktop Application을 만드는 프레임워크입니다. NPM을 통한 Node Package들을 사용할 수 있도록 지원합니다.

(2) 사용

1. load

http주소나 html, mht, xml와 같이 로컬에 있는 파일을 electron상의 web contents로 로드하는 기능

This module cannot be used until the `ready` event of the `app` module is emitted.

```
// In the main process.
const { BrowserWindow } = require('electron')

const win = new BrowserWindow({ width: 800, height: 600 })

// Load a remote URL
win.loadURL('https://github.com')

// Or load a local HTML file
win.loadFile('index.html')
```

<그림 5. load 함수>

2. printToPdf

webContents 상의 내용을 웹 브라우저 내의 pdf로 인쇄와 동일하게 pdf로 인쇄 진행하는 기능

An example of `webContents.printToPDF`:

```
const { BrowserWindow } = require('electron')
const fs = require('fs')
const path = require('path')
const os = require('os')

const win = new BrowserWindow()
win.loadURL('http://github.com')

win.webContents.on('did-finish-load', () => {
  // Use default printing options
  const pdfPath = path.join(os.homedir(), 'Desktop', 'temp.pdf')
  win.webContents.printToPDF({}).then(data => {
    fs.writeFile(pdfPath, data, (error) => {
      if (error) throw error
      console.log(`Wrote PDF successfully to ${pdfPath}`)
    })
  }).catch(error => {
    console.log(`Failed to write PDF to ${pdfPath}: `, error)
  })
})
```

<그림 6. printToPdf 함수>

3. show / openDevTools

디버깅용 툴로, webContents가 로드 된 후 스크립트로 조작할 때 이슈가 발생하거나, 의도와는 다른 결과물이 생성된 경우 이를 통하여 트래킹

Using the `ready-to-show` event

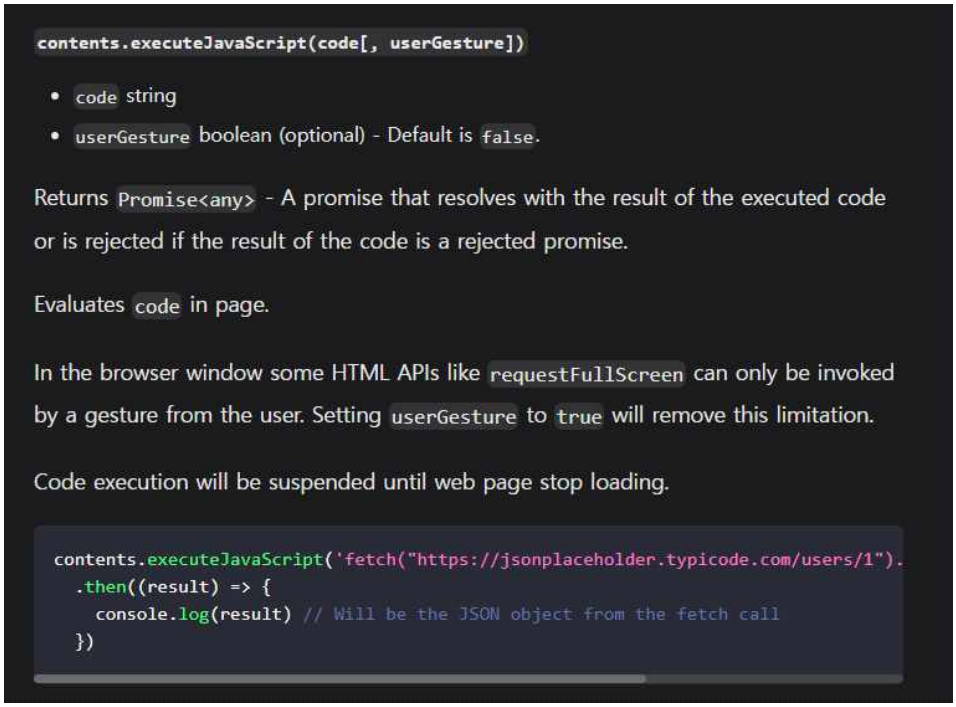
While loading the page, the `ready-to-show` event will be emitted when the renderer process has rendered the page for the first time if the window has not been shown yet. Showing the window after this event will have no visual flash:

```
const { BrowserWindow } = require('electron')
const win = new BrowserWindow({ show: false })
win.once('ready-to-show', () => {
  win.show()
})
```

<그림7. show/ opendevTools 함수>

4. executeJavaScript

web contents에서 파라미터로 전달한 js를 실행하는 함수로, 페이지의 margin이나, footer / header등의 문구를 추가하는 기능



`contents.executeJavaScript(code[, userGesture])`

- `code` string
- `userGesture` boolean (optional) - Default is `false`.

Returns `Promise<any>` - A promise that resolves with the result of the executed code or is rejected if the result of the code is a rejected promise.

Evaluates `code` in page.

In the browser window some HTML APIs like `requestFullScreen` can only be invoked by a gesture from the user. Setting `userGesture` to `true` will remove this limitation.

Code execution will be suspended until web page stop loading.

```
contents.executeJavaScript('fetch("https://jsonplaceholder.typicode.com/users/1").
  .then((result) => {
    console.log(result) // Will be the JSON object from the fetch call
  })
```

<그림8. executeJavaScript 함수>

3. 상세 설계

(1) 아규먼트 정의

1. 필수 항목

(1) --input="Input URL or local file path"

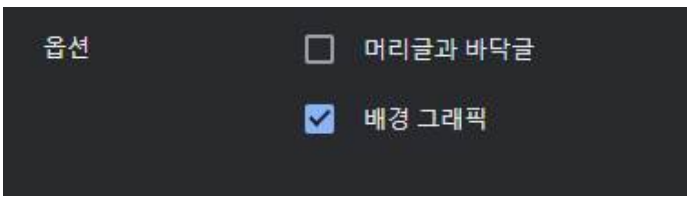
(2) --output="Result PDF File path"

2. 선택 항목

(1) --delay=[millisecond] : Wait this time after the page loads.

(2) --printBackground

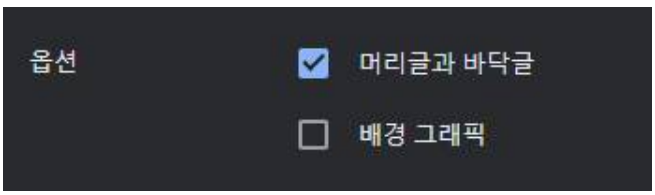
- electron의 printBackground 옵션에 매핑
- 브라우저의 "배경 그래픽" 옵션과 동일
- switch용 옵션으로 없으면 default 값 true



<그림 9. 배경그래픽 옵션>

(3) --footer="some text" / --header="some text"

- 브라우저의 "머리글과 바닥글" 옵션과 동일
- 머리글과 바닥글에 들어갈 텍스트 각각 설정 가능



<그림 10. 머리글과 바닥글 옵션>

(4) --landscape

- 인쇄 용지의 방향 설정. default값은 false(세로)

(5) --margin=[no-margin|minimum|n,n,n,n]

- 옵션이 없는 경우 default는 no-margin
- 브라우저의 최소마진 혹은 Top,Left,Bottom,Right 순으로 입력 가능

(6) --timeout=[millisecond]

- 변환 작업이 특정 시간 이상으로 소요될 경우 강제로 종료

(7) --pageSize=[A4|A3 ...]

- 인쇄 페이지 크기 설정 default는 A4

(8) --cookies="json file path"

- url 접속 시 필요한 쿠키의 데이터 설정

<그림 11. cookie json 예시>

(9) --requestHeader="text file path"

- url 접속 시 필요한 request Header의 데이터 설정
- text파일 예시

key:value
key:value
...

(10) --logDir="log directory"

- webConverter 사용시에 생성될 로그의 저장 경로

(2) 기능별 함수 설계

1. const checkArgsAvailable = (input ,cookies,logDir)
   입력받은 파일을 체크한 후 없다면 오류코드 출력 후 프로세스 종료

2. const checkFileFormat = (input)
   Input 파일의 확장자가 Web으로 load 가능한 파일인지 확인
   - 허용 가능한 확장자 : [mhtml, html, mht, xml, WebURL]

3. const makeLogFile = (logDir)
   logDir의 위치에 YYYYMMDDHHMMSS.txt 형식으로 전역 변수인 logFile 변수를 설정

4. const logging = (type,context)
   전역변수인 LogFile의 경로에 'fs' 모듈을 통해 컨텐츠를 작성
   'fs' 모듈에서 Sync 오류가 발생 시 오류 코드 출력 후 프로세스 종료

5. const insertCookies = (url, cookies)
   입력받은 cookie 파일을 읽어 JSON 모듈로 파싱한 후, 데이터의 key를 순회하며 Cookie 객체 생성.
   생성한 Cookie 객체를 Eletron의 DefaultSession에 삽입

6. const makeURLOption = (reqHeader)
   reqHeader 경로의 header파일(txt)을 읽어 전체 URL을 만드는 함수

7. const applyAttribute = (margin, header, footer, pageSize)
   margin 옵션을 준 경우, margin 세팅 진행 후, 설정한 pageSize에 margin 값이 문제 없는지 검증
   header 옵션을 준 경우, header 세팅 진행
   footer 옵션을 준 경우, footer 세팅 진행

8. const print = async (input, output, cookies, requestHeader, delay, timeout, margin, printBackground, landscape, header, footer, pageSize, debugMode)
   페이지 로드에 사용하기 위해 전역 변수로 선언된 window init
   debugMode가 켜진 경우 devtool을 킴
   아규먼트로 timeout을 받은 경우 프로세스의 timeout 설정
   (timeout시 APP_TIME_OUT 오류 코드로 프로세스 종료)
   'did-finish-load' 콜백을 받으면 printPage(...) 함수를 호출
   input 파싱 및 로드 진행

9. const printPage = (output, delay, margin, printBackground, landscape, header, footer, pageSize)

print(...)에서 'did-finish-load' 콜백 시 호출되는 함수.
header나 footer가 정의된 경우 displayHeaderFooter 플래그를 킴.
delay가 있거나, footer, header 및 margin 중 하나라도 설정된 경우
기본 delayTime을 설정
setTimeout(printToPdf(..), delayTime) 으로 delay를 준 후 printToPdf(...)
함수에서 실제 변환 진행.

10. const printToPdf = (filePath, margins, printBackground, landscape, pageSize, headerTemplate, footerTemplate, displayHeaderFooter, postCallback)

실제 변환 진행.
postCallback() 함수는 printPage에서 () => app.exit(0)를 전달.

```
window.webContents.printToPDF({
        margins : margins,
        printBackground : printBackground,
        landscape : landscape,
        pageSize : pageSize,
        displayHeaderFooter : displayHeaderFooter,
        headerTemplate : headerTemplate,
        footerTemplate : footerTemplate}).then(data=> {
```

callback 부분에서 'fs'모듈을 통해 로컬에 파일로 저장 및 넘겨받은 callback 함수 호출.

```
fs.writeFile(filePath, data, () => {postCallback();})
```

(3) 오류 코드 정의

(1) INPUT_FILE_NOT_FOUND

- input을 로컬에 있는 파일로 설정했을 때 파일ㅇ르 찾지 못한 경우의 오류 코드
- code = 100

(2) UNSUPPORT_INPUT_FILE_TYPE

- input을 지원할 수 없는 경우에 발생하는 오류
- localFile일 수도, url 파일일 수도 있음.
- code = 101

(3) COOKIES_FILE_NOT_FOUND

- cookie 옵션으로 받은 경로에서 cookie 파일을 찾지 못한 경우의 오류 코드
- code = 102

(4) LOG_DIRECTORY_NOT_FOUND

-로그 파일 작성을 위한 logDir 옵션으로 받은 경로를 찾지 못한 경우
- code = 103

(5) WRITE_LOG_SYNC_ERROR

- 로그 파일 작성 중 오류가 발생한 경우의 오류 코드
- code = 104

(6) APP_TIMEOUT

- 정의된 timeout 시간보다 지체되어 프로세스가 종료된 경우의 오류 코드
- code = 105

4. 결과

(1) 단위 테스트 목록

- Delay 기능
- printBackground 기능
- Footer / Header 기능
- landscape 기능
- margin 기능
- Timeout 기능
- pageSize 기능
- cookie 기능
- header 기능
- 로깅 기능

(2) 통합 테스트 시나리오

1. Case A

회사의 요청 등으로 인하여 웹에서 특정 물품 목록의 가격 및 정보를 갖는 페이지를 취합하여 이를 일괄적으로 PDF 문서화 하는 작업

해당 테스트 시나리오에서는 자동화 스크립트 혹은 UI 프로그램을 통해 input 파일들이 위치한 디렉토리를 받아 해당 파일들을 일괄 변환.

2. Case B

특정 파일을 백업 및 열람하는 기능을 제공하는 솔루션에서 html 및 Web URL을 PDF로 변환하여 저장하도록 하는 경우

해당 테스트 시나리오에서는 python3을 통한 간이 서버를 구축하여 API 호출 시 파일을 전달 받아 이를 로컬에 저장 및 모듈을 통하여 변환된 PDF도 함께 저장

(3) 빌드 방법

레포지토리 하위의 build.py 스크립트를 실행하면, dist 디렉토리 하위에 webConvertor-linux-x64.tar.gz, webConvertor-win32-ia32.zip이 생성됩니다.

그리고, webConvertor-win32-ia32.zip에는 윈도우용 테스트 모듈인 webConvertor-tool.exe가 함께 포함되어 있습니다.

5. 시연

(1) 빌드 시연 영상

https://drive.google.com/file/d/1KCoRvovmdur1tiBkGRiRCiH30MMVVXGM/view?usp=drive_link

(2) Case A에 대한 시연 영상

https://drive.google.com/file/d/1kAYc9Ja7lc2llQcBYTj5r1mYlHjqXmaM/view?usp=drive_link

6. 후기

(1) 32170171 고현우

프로젝트를 처음 수행할 때, 어떤 어플리케이션이나 서비스를 개발하면 다양한 사람들에게 도움이 될 것인가에 대해 고민해 보았습니다.
한때, html을 pdf로 변환하기 위해 파일을 직접 열고, 이를 수작업으로 변환하던 것이 생각나 이를 별도의 작업 없이 콘솔 명령 실행으로 제공하는 어플리케이션을 기획하게 되었습니다.

프로젝트를 진행하면서, 어떤 기능을 제공하면 어플리케이션 사용자가 만족할지, 어떤 기능을 추가로 요구할 것인지에 대한 고민을 하였기에 다양한 옵션을 추가하게 되었던 것 같습니다.
그리고, 자주 사용하지 않던 언어인 javascript를 기반으로 프로젝트를 수행하며 익숙하지 않던 언어의 실력을 함께 키울 수 있었습니다.

프로젝트를 마무리하고, 현재 다양한 커뮤니티 (stackOverFlow, Git)등에 저희 모듈을 사람들에게 소개하고, 필요로 하는 사람들이 pull받아 사용할 수 있도록 안내하고 있습니다.
또한, 사용자의 입장에서 서비스의 기능 요구사항을 정리해 보고, 이를 어떻게 구현할지 계획을 세우며 기획자의 안목을 기를 수 있었 던 것 같습니다.

(2) 32172110 소완열

개발 경험이 적음에도 불구하고 고현우 학우의 리드로 프로젝트를 성공적으로 마칠 수 있었습니다. 또한 기존에 없던 오픈소스 프로젝트를 만드는데 기여함으로써 뿌듯함을 느끼게 되었습니다.
앞으로 이 경험을 기반으로 앞으로도 다양한 프로젝트를 개발해 다양한 사람들이 함께 쓸 수 있는 오픈소스를 개발하고 싶어졌습니다.