2020. 5. 1.

소프트웨어공학

2차 프로젝트

이 기 훈 교수님

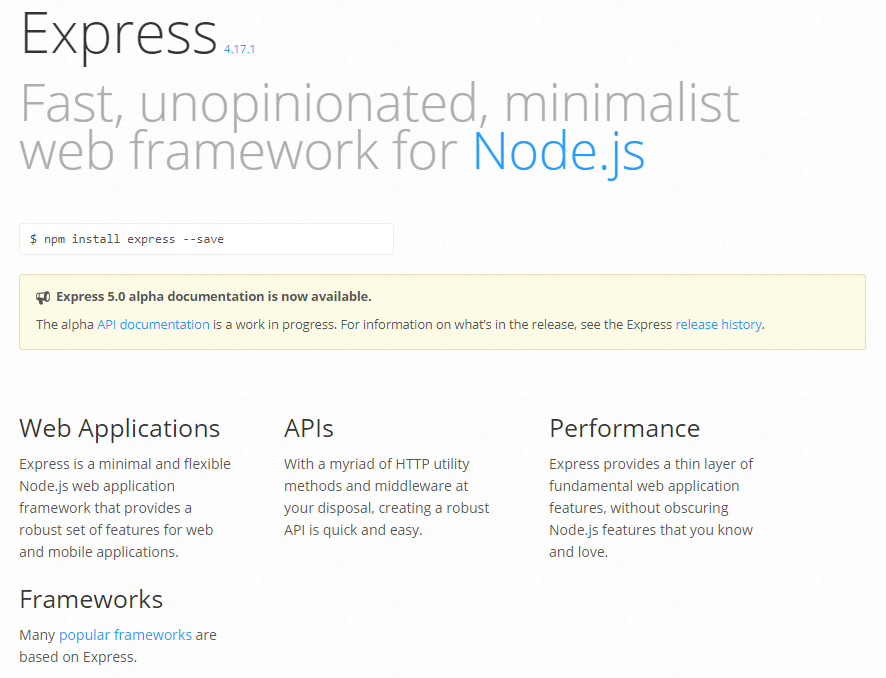
2015722087 컴퓨터정보공학부

김민철

**1. Node.js – MVC Framework에 대한 조사 및 설명**

Framework라는 것은 해석하면 ‘뼈대’이다. 애플리케이션 개발에 바탕이 되는 템플릿과 같은 역할을 하는 클래스와 인털페이스의 집합이며, Web Framework는 Web을 개발하는데 사용되는 프레임워크이며, 그 중에서도 MVC Framework는 MVC 패턴을 지원하는 Framework이다. Framework를 사용하게 되면, 모든 애플리케이션에 적용되는 공통적인 부분을 구현하지 않고 사용하기만 하도 되어 개발 시간과 비용이 단축되기 때문에, Framework를 사용하는 것은 매우 권장되는 사항이다.

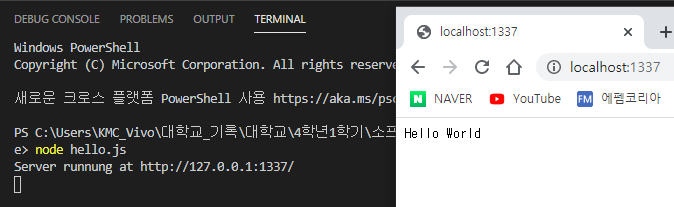
Node.js의 Framework에는 많은 종류가 있다. Express.js, Hapi.js, nest.js, sails.js 등이 있지만, 이중에서 MVC 패턴을 사용하는 것은 Express, Adonis.js, Sails, Derby, Mean 등이 있다. 이중에서 Express.js가 가장 많이 쓰이며 MVC 패턴으로 동작하는 Web Framework이다.



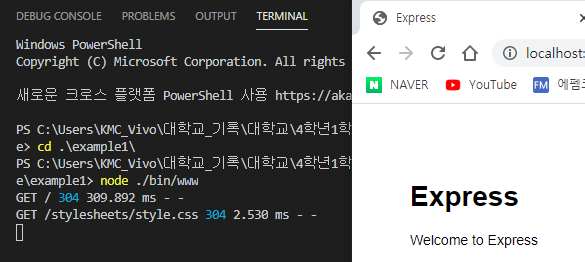
Npm을 통해 간단히 설치하여 사용할 수 있으며, 템플릿 엔진에 ejs 파일을 사용할 수 있다. Web Program을 위한 기본적인 시스템이 구축되어 있으며 개발자별로 필요한 처리를 아주 쉽게 추가하여 사용할 수 있다. 대부분의 다른 Framework도 Express.js를 기반으로 구현되었기 때문에 가장 많이 쓰이며, 간단하고 효율적이라 가장 많이 쓰이는 것 같다.

2020년에 사용될 Node.js 프레임워크 중에서 가장 많이 쓰일 프레임워크로 선정되었으며, 2019년엔 가장 많이 사용된 Node.js 프레임워크로 선정되었다.

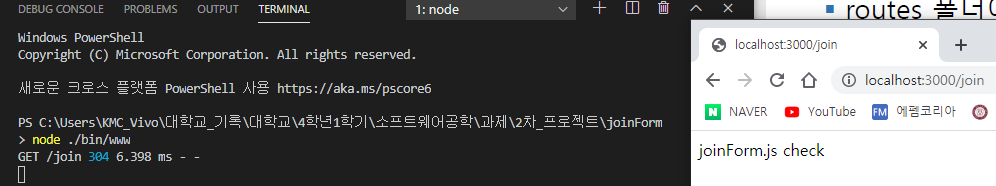
**2. 제공된 튜토리얼 진행**



- 아주 간단한 텍스트를 출력하게하는 웹 서버를 자바스크립트로 구현하여 URL에 localhost:포트번호 입력시 브라우저가 응답을 받아 출력하는 것을 확인하였다.



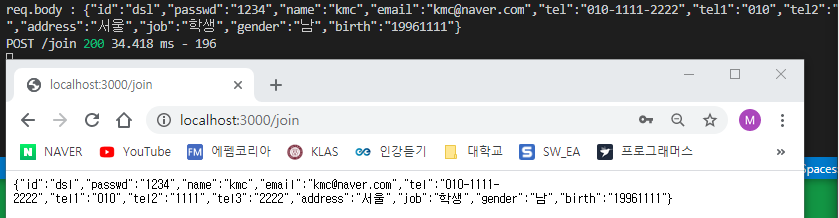
- nodejs와 express를 설치하고 동작하는 것을 확인하였다.



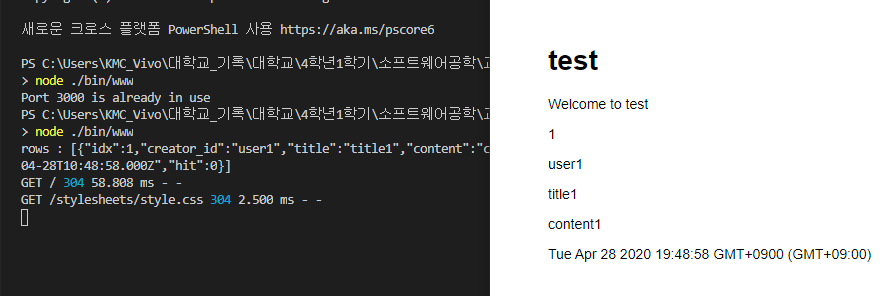
- 회원가입 Form을 만들기 위한 작업을 하는 과정에서 로직을 담당하는 js파일을 생성하고 app.js에서 연결하는 과정을 진행한 뒤에 확인해본 결과이다.



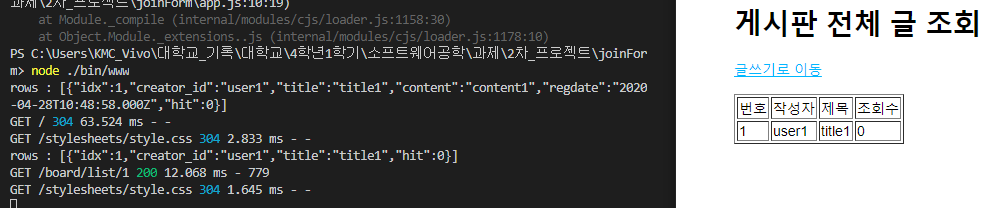
- 연결된 로직을 담당하는 joinForm.js파일에서 응답 메시지로 joinForm.ejs에 title 값을 입력하여 렌더링한 결과를 보내주게 하였다.



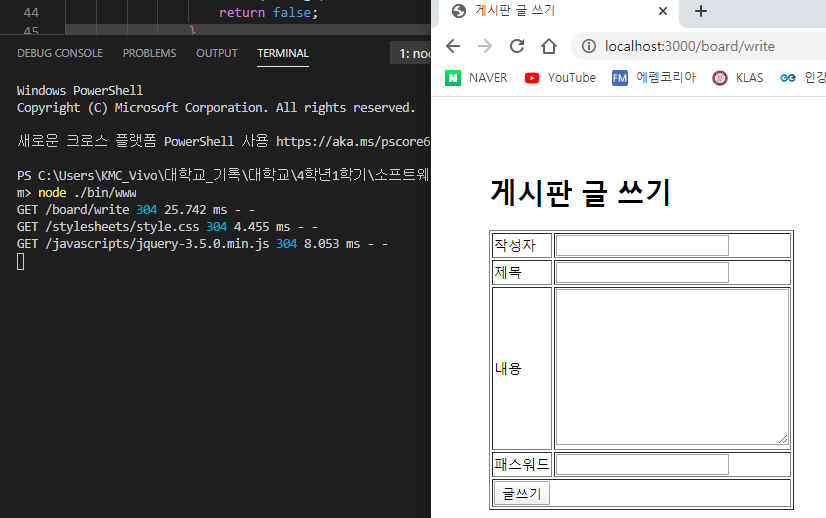
- 이전에는 GET method의 동작이었다면 이번에는 POST method의 동작을 해보았다. 이전에 브라우저에서 joinForm.ejs에 작성한 데이터들을 request 메시지에 담아 POST 방식으로 request하면, 서버에서 해당 request를 받아 메시지에 담긴 내용을 json 형식으로 변경하여 응답 메시지에 담아 전송하였다.



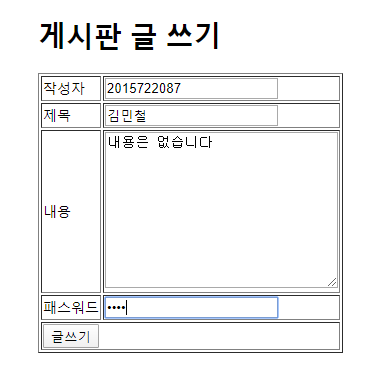
- MySQL의 데이터베이스와 Nodejs로 구현된 웹서버를 연동하여 데이터베이스에 담긴 내용을 응답하는 동작을 구현해보았다.



- 이전에 했던 단계들을 기반으로 게시판을 구축해보았다. App.js 파일에 board라는 새로운 router를 연결하였고, 글 목록을 출력하는 list.ejs 파일을 생성하여 응답 메시지로 list.ejs 파일에 title과 데이터베이스의 board 테이블의 데이터들을 입력하여 렌더링한 결과를 전송하였다.



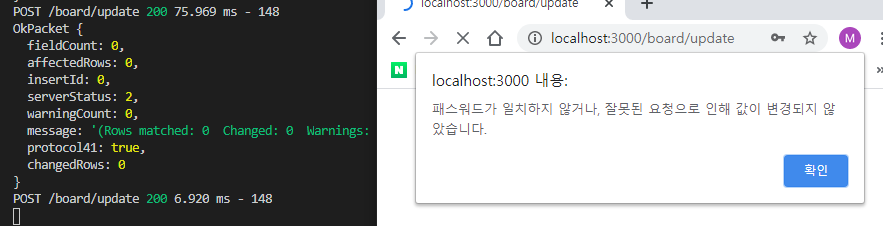
- 이전 단계와 비슷하게 글을 작성하는 페이지를 생성하기 위해 write.ejs 파일을 생성하고, 글을 작성하는 url을 ‘/write’로 하여 router가 ‘/write’ url을 GET method로 받으면 쓰기 페이지를 출력하도록 구현하였다.

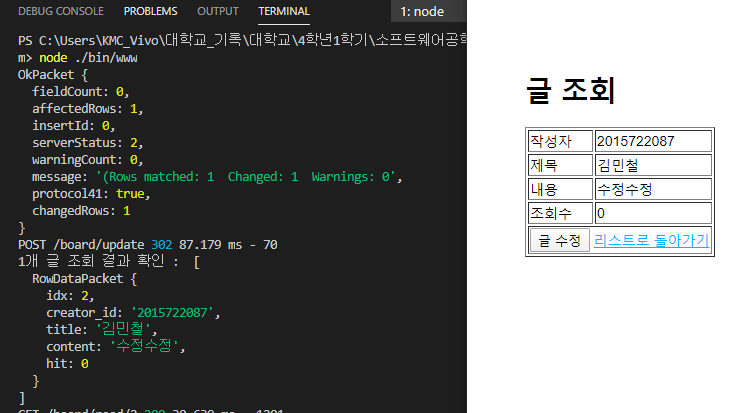
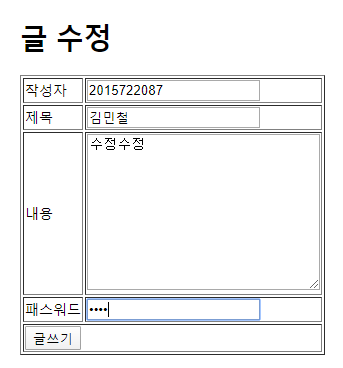


- 게시물을 작성하고 글쓰기 버튼을 클릭하면, 이전에 joinForm에서 했던것과 마찬가지로 ‘/write’ url에 POST 방식으로 받아 입력된 데이터들을 데이터베이스에 작성하도록 처리하였다.



- CRUD의 기능중 Create를 하였고 Read와 Update 동작을 구현하였다. 이전에 했던 방식들과 비슷하게 Read 기능을 위해 read.ejs 파일을 생성하였고, ‘/read/<index>’ url을 GET 방식으로 받으면 해당 index에 해당하는 게시물의 정보를 출력하도록 구현하였다.





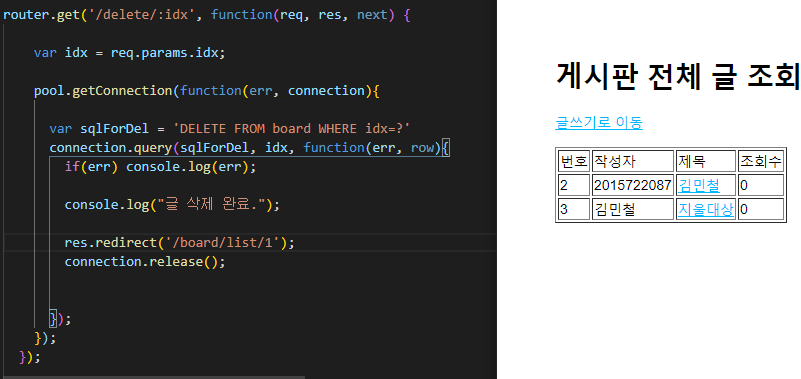
- Read 기능에서 ‘글 수정’ 버튼을 클릭하면 수정하는 페이지로 이동하도록 하였다. 이 과정 역시 update.ejs 파일을 생성하였으며, 이곳에서 작성한 내용들을 POST 방식으로 요청 메시지를 보내어 데이터베이스의 내용을 수정하도록 하였으며, 비밀번호가 맞지 않는다면 수정되지 않도록 처리하였다.

**3. 게시글 삭제 기능, 이미지 업로드 및 출력 구현 과정 설명 및 캡처**

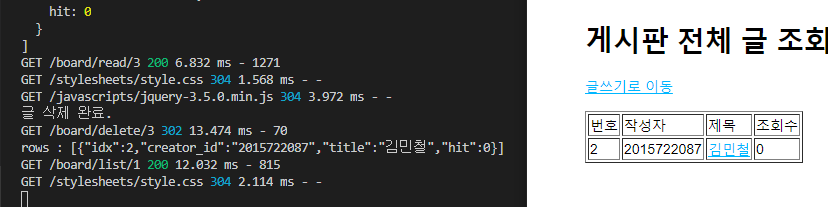
- 게시글 삭제 기능



- 가장 먼저 글을 Read하는 화면에서 삭제 기능을 수행할 수 있도록 하였다.

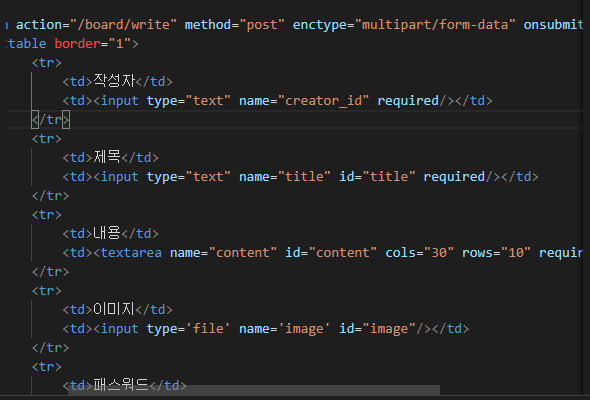


- 위에서 작성한 ‘글 삭제’ 버튼을 클릭하면 이동하게되는 url 주소를 router에서 받으면 삭제 기능을 동작하도록 구현하였다. 이때 어떤 글을 삭제할 것인지 index를 확인하기위해 url 뒤에 idx 값을 추가하여 받도록 구현하였다. 삭제하고 난 후에는 list를 출력하는 화면으로 이동하도록 응답 메시지를 전송하였다.

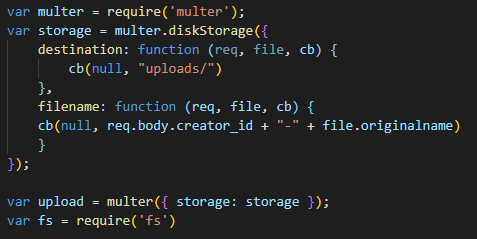


- 삭제하고 난 후의 화면이다.

- 이미지 업로드 & 출력 기능



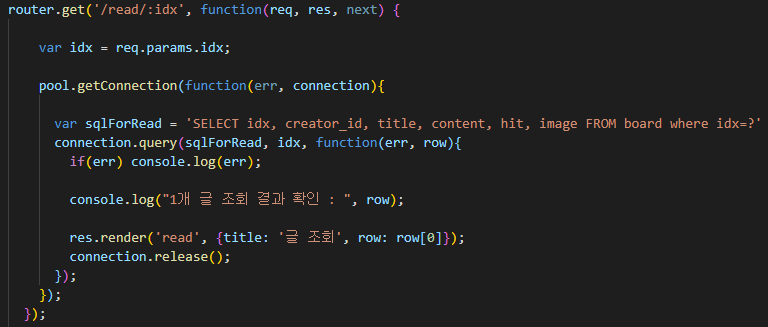
- 우선 글을 작성하는 화면에서 이미지를 업로드할 수 있도록 하기 위해 write.ejs의 Table에 이미지를 업로드하는 요소를 추가하였다.



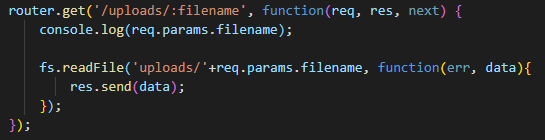
- 이미지를 업로드하기 위해 ‘multer’ 모듈을 install하여 사용하였다. 파일을 받으면 uploads/ 경로에 ‘작성자id-파일이름’ 으로 파일을 저장하도록 설정하였다.



- wrtie하는 과정에서 업로드한 파일을 받아 경로인 req.file.path를 데이터베이스에 입력하도록 write 동작을 수행하는 부분을 수정하였다.



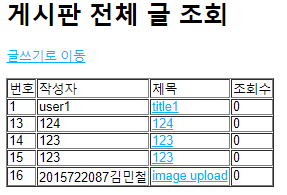
- 데이터베이스에 파일의 경로를 저장했으니 Read하는 과정에서도 역시 해당 경로를 데이터베이스에서 읽어오는 수정 과정을 수행하였다.

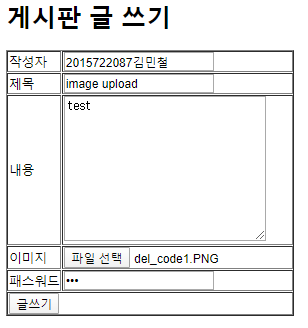


- 파일을 읽어 데이터를 보내주는 동작을 수행하기 위해 ‘/uploads/<filename>’의 URL을 받으면 해당 파일의 데이터를 응답 메시지에 담아 보내주는 동작을 수행하는 코드를 작성하였다.

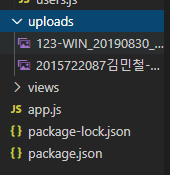


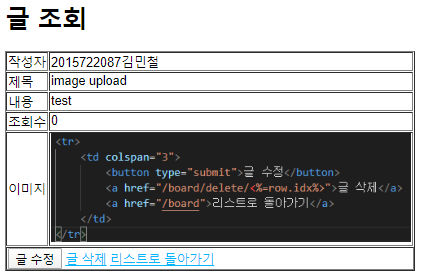
- read.ejs에서 이미지를 출력하는 화면을 생성하고, image의 source 주소를 데이터베이스에서 추출한 경로로 하였다.





- 글을 작성하는 과정에서 파일을 업로드할 수 있게되었음을 확인하였다.



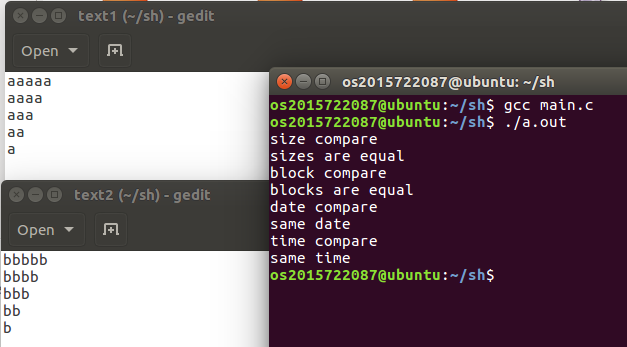


- 파일을 업로드하면 uploads 폴더에 작성자-파일이름 으로 복사되어 저장되는 것을 확인하였고, 글을 Read하는 기능을 수행해보니 이미지가 정상적으로 출력되는 것을 확인하였다.

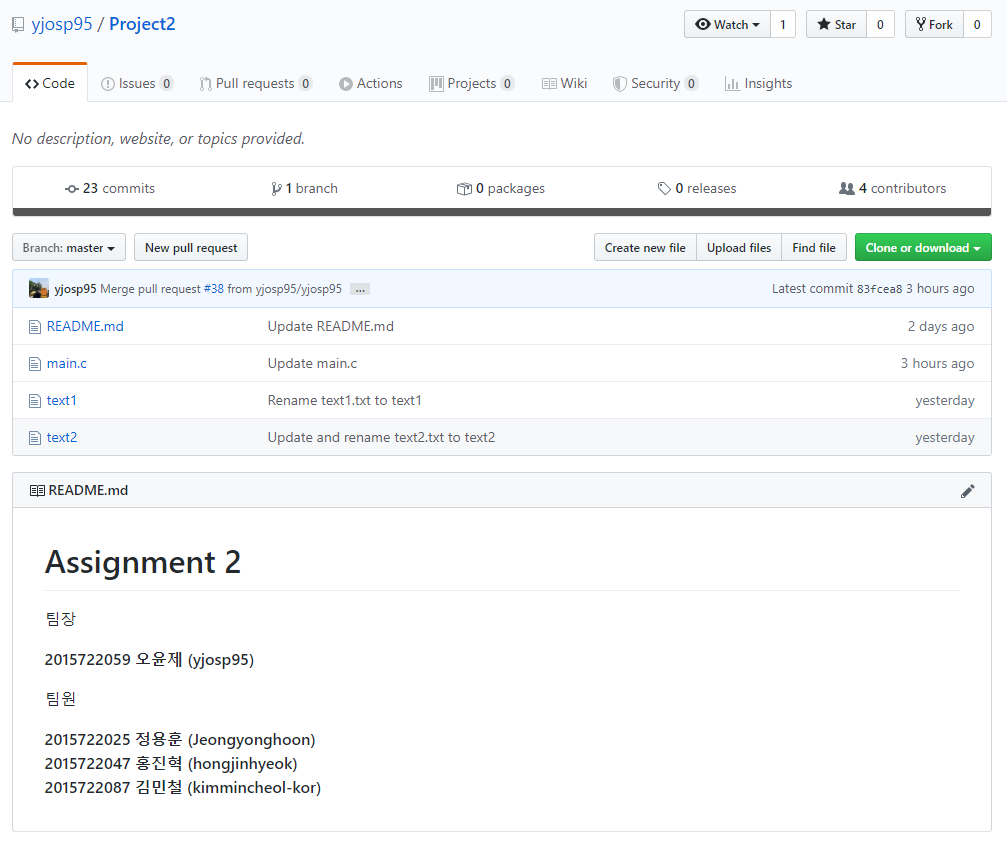
**4. GitHub 공동 작업**

(GitHub Repository : <https://github.com/yjosp95/Project2>)

결과 화면



GitHub Repository 메인 화면



**5. 고찰**

이번 과제에서는 처음으로 Git을 이용하여 실제로 협업을 하였고, 또 처음으로 Node.js와 Express.js를 사용하여 웹 서버를 구현해보았다. 이전에 웹 서버는 ‘시스템프로그래밍’ 강의에서 C언어로 구현 했었는데, 그때 구현했던 웹 서버를 생각하면 정말 단순한 기능을 제공하는 웹 서버임에도 복잡하다고 생각했었는데, 이번에 Node.js와 Express.js를 사용하여 구현해보니 웹 서버를 구현하는 것이 정말 편리해짐으로써 더욱 다양하고 기능이 많은 웹 프로그램들이 나올 수 있는 것 같다는 생각이 들었다. 협업을 하는 과정에서는 기존에 혼자 작업하던 때와는 다르게, 팀원들과 코드를 공유하고 수정 과정을 확인할 수 있게 되면서 협업자가 어디를 수정하였는지 확인하고 진행할 수 있어 편리함을 느낄 수 있었으며 왜 Git을 사용하여 협업하는 것이 개발자에게 있어 선택이 아닌 필수로 분류되는지 조금은 알 것 같은 느낌을 받을 수 있었다.