프로젝트 보고서

1. 팀 소개

남현수 (201504012): Back-end 개발 , 전체 시스템 설계 , 모델링
김바울 (201707004): Front-end 개발 , 리엑트(material-ui) , API 설계 (프로필 부분)
손은빈 (201701032): Back-end 개발 , 로그인 구현, 리덕트, 리엑트 훅 적용

2. 프로젝트 소개

Nodejs를 사용한 웹 프로젝트입니다. 배운 기술을 활용하여 social network service를 만들었습니다. 또한, 필요한 기술들을 학습하여 프로젝트에 적용하였습니다.

사용자 등록 및 인증, 게시물 및 코멘트 등록, 사용자 프로필 수정 등의 기능을 제공합니다. 리액트를 사용하여 자연스러운 페이지 이동과 세련된 사용자 경험을 제공합니다. 무한 스크롤을 사용하여 편리한 서비스 이용환경을 제공합니다.

3. 개발환경

Back-end: NodeJS(express), MongoDB
Front-end: ReactJS(redux, react-hook)

4. API 명세서

User API

	API	Method	Role	Parameters	
1	/api/users/profile	post	사용자 정보 요청 (프로필)	userID	사용자 id
	/api/users/profileUpdate		사용자 정보 업로드 (프로필)	userID	사용자 id
				location	거주지역 정보
				birthday	사용자 생일
				job	사용자 직업
				profileImg	프로필 사진
	/api/users/following		팔로우 관계 생성	followingID	팔로우 하는 사용자의 ID
	/api/users/getFollowing		팔로잉 친구 정보 요청	followerID	팔로워 정보를 얻으려는 사용자 ID

Auth API

	API	Method	Role	Parameters	
1	/api/auth/auth	GET	사용자 정보 인증	_id	사용자 id
				isAdmin	관라자 여부
				isAuth	인증 여부
				email	사용자 이메일
				name	사용자 이름
				location	거주 지역
				birthday	생일
				job	직업
				profileImg	프로필 이미지
				following	팔로잉
	/api/auth/logout		사용자 로그아웃	isAuth	인증 여부
				_id	사용자 id
2	/api/auth/login	POST	사용자 로그인/ 회원가입	email	이메일
				password	비밀번호
	/api/auth/signup			email	이메일
				name	이름
				passward	비밀번호

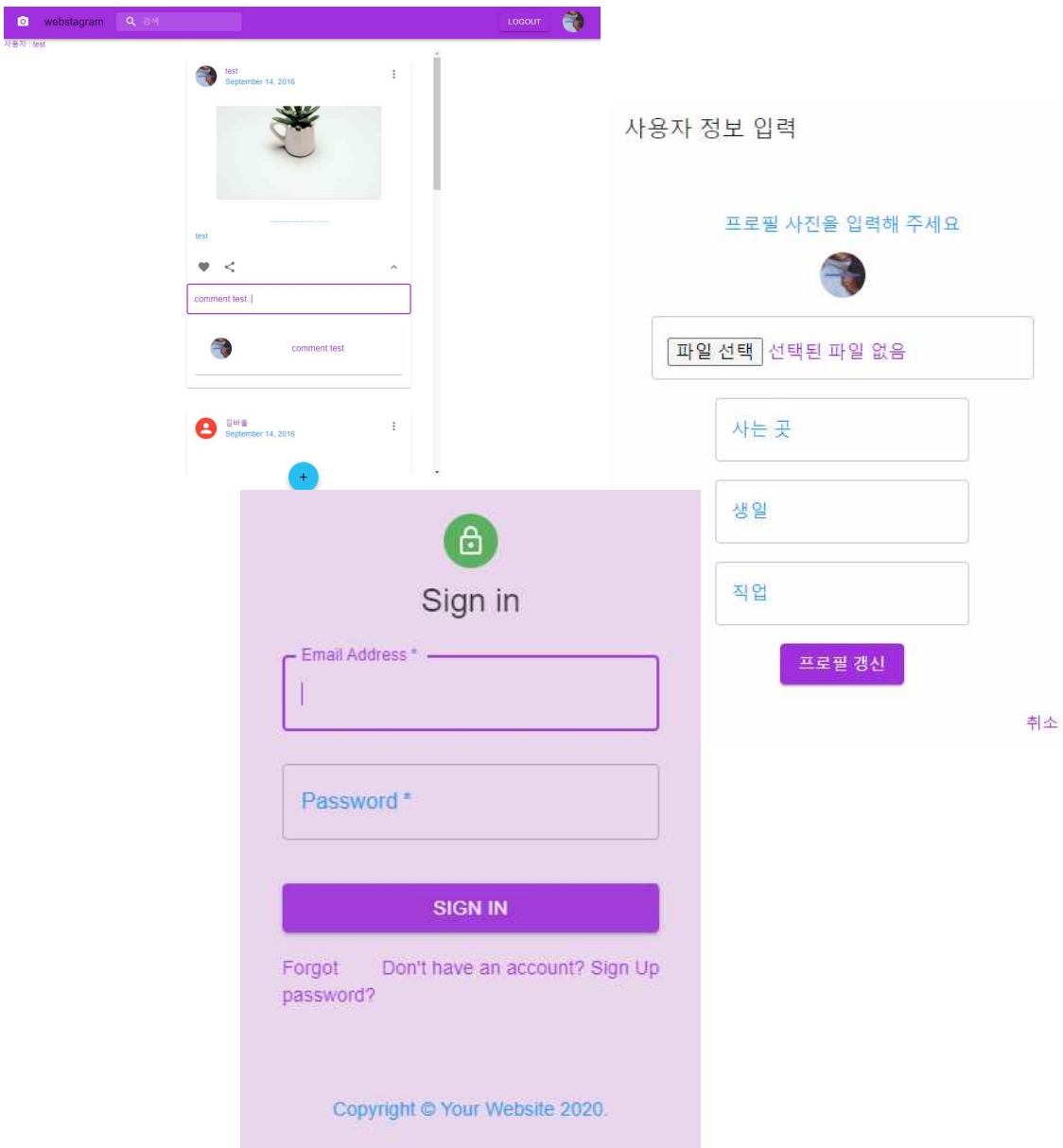
Post API

	API	Method	Role	Parameters	
1	/api/posts/get	GET	게시물 정보 요청	-	-
2	/api/posts/img	POST	이미지 업로드	img	이미지
				url	이미지 주소
2	/api/posts/create		게시물 생성 요청	userID	게시글 작성자
				content	게시글 내용
				postImg	게시글 이미지
2	/api/posts/like		좋아요 정보 생성 요청	postID	해당 글 id
				userID	사용자 id

Comment API

	API	Method	Role	Parameters	
1	/api/comments /create	POST	댓글 생성 요청	userID	작성자 id
				postID	해당글 id
				comment	댓글 내용
2	/api/comments /get		댓글 정보 요청	postID	해당글 id

5. 프로젝트 결과 화면 캡쳐



6. 앞으로의 프로젝트 개발 방향

1. 게시물 삭제 및 수정 , 언팔로우
2. 좋아요 기능의 보완
3. 동영상 업로드
4. 게시물 검색
5. 해시태그 등록 및 검색
6. 카카오, 페이스북 소셜로그인

7. 소스코드

```
const express = require('express')
const router = express.Router();

const { auth } = require('../middleware/auth') //미들웨어 역할을 하는 모듈을 직접 만들어 임포트한다.
const { User } = require('../models/Users');


router.post('/login', (req, res) => {
    console.log(req.body)  //요청된 이메일을 데이터베이스에서 있는지 찾는다.
    User.findOne({ email: req.body.email }, (err, user) => {
      if(!user) {
        console.log("등록되지 않은 이메일입니다.");   //alert("제공된 이메일에 해당하는 유저가 없습니다.");
        return res.json({
          loginSuccess: false,
          message: "등록되지 않은 이메일입니다."
        })
      }
      user.comparePassword(req.body.password, (err, isMatch) => {  //요청된 이메일이 데이터베이스에 있다면 비밀번호가 맞는 비밀번호인지 확인.
        if(!isMatch) {  //만약 비밀번호가 맞지 않다면,
        console.log("비밀번호가 틀렸습니다.");
        //alert("비밀번호가 틀렸습니다.");
        return res.json({ loginSuccess: false, message: "비밀번호가 틀렸습니다."})
        }
        //비밀번호까지 맞다면 토큰을 생성하기.
        user.generateToken((err, user) => {
          if(err) return res.status(400).send(err);

          console.log("로그인에 성공했습니다.")
          //토큰을 저장한다. 어디에? 쿠키, 로컬스토리지 여러가지 방법이 있고 각각 장단점이 존재한다.
          res.cookie("x_auth", user.token) //첫번째 인자로 쿠키의 이름, 두번째 인자로 쿠키의 값이 전달된다.
            .status(200)
            .json({ loginSuccess: true, userId: user._id })
        })
        console.log("로그인에 성공했습니다.")
      })
    })
    //비밀번호까지 맞다면 토큰을 생성하기.
  })
```

[로그인 부분]

```
router.post('/following', async (req, res) => {

  const following = await User.find({ _id: req.body.followingID }, { _id : 0})
  if(following){    // 각 관계에서 팔로워와 팔로잉을 받고 이를 서로 이은 하나의 스키마에 넣어준다.

    const follow = new Follow({ followerID: req.body.followerID , followingID : req.body.followingID });

    follow.save((err, userInfo) => {   // 데이블(컬렉션)에 하나의 요소를 만든다.
      if(err) return res.json({ success: false, err })
      return res.status(200).json({
          success: true
      })
    })

  }else {  // 사람을 찾지 못했을 때는 이렇게 처리한다.
  res.status(404).send('no user');
  }

});

router.post('/getFollowing', async (req, res) => {

  // 팔로워의 정보를 불러올 때 팔로우 컬렉션에서 팔로워의 id로 모두 찾아서 반환한다.
  const followingUser = await Follow.find({ followerID : req.body.userID } , { _id : 0 ,  followingID : 1}).populate('followingID');
  if(followingUser){
  res.json(followingUser);  // 클라이언트 쪽으로 보낸다.

  }else { // 사람을 찾지 못했을 때는 이렇게 처리한다.
  res.status(404).send('no user');
  }

});


module.exports = router;
```

[팔로잉 부분]