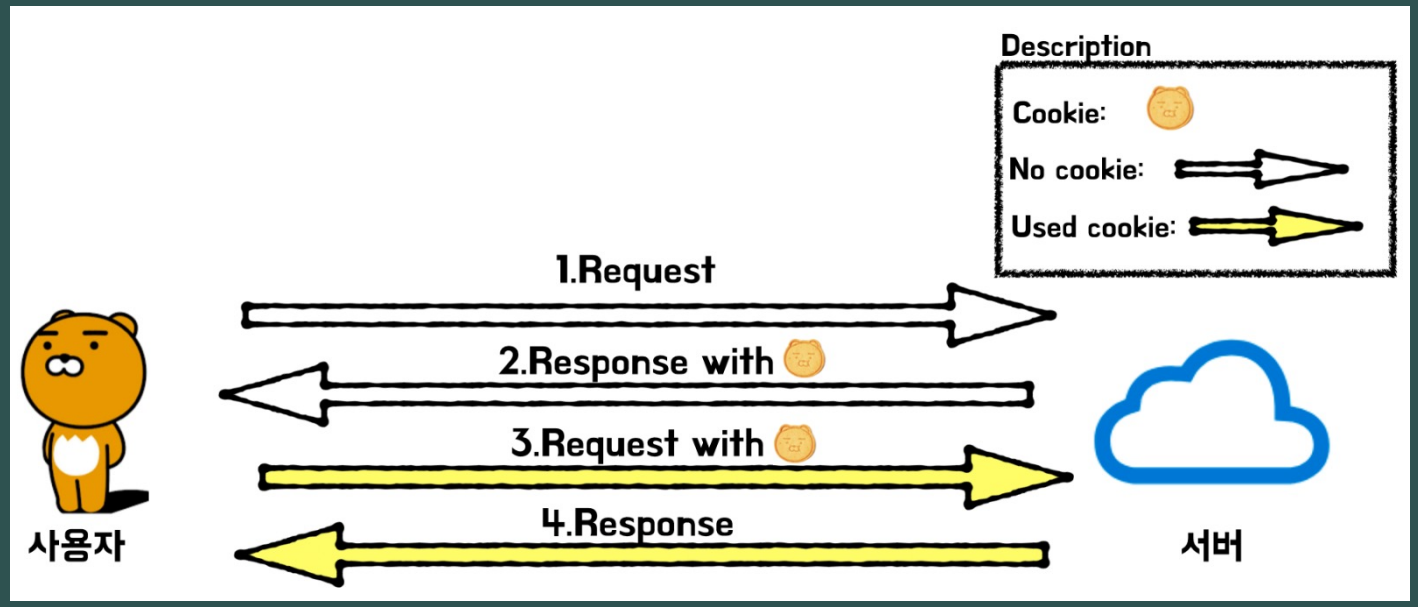
Weekly Progress 5

Capstone Design Project

TEAM B

2021.11.23

Session, Cookie



JWT(JSON WEB TOKEN)

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

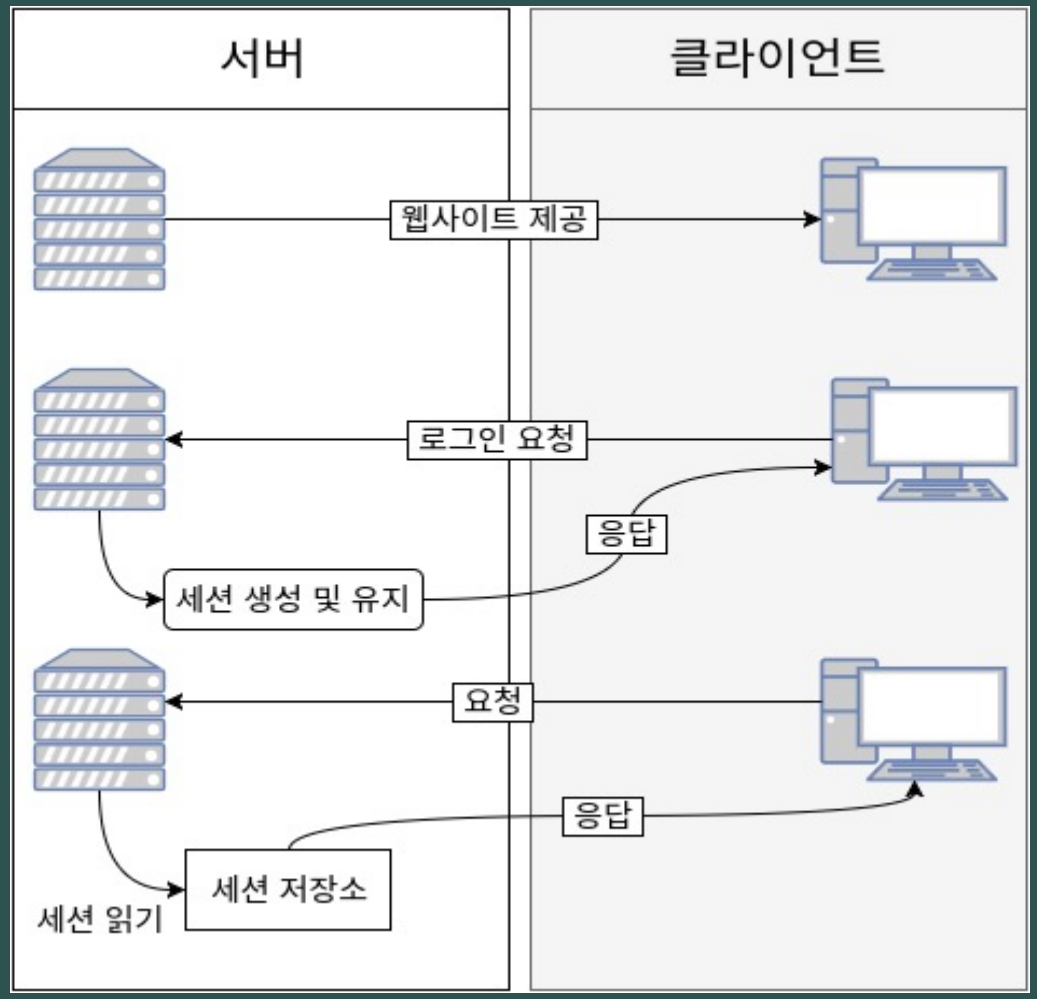
JWT(JSON WEB TOKEN)

Cookie

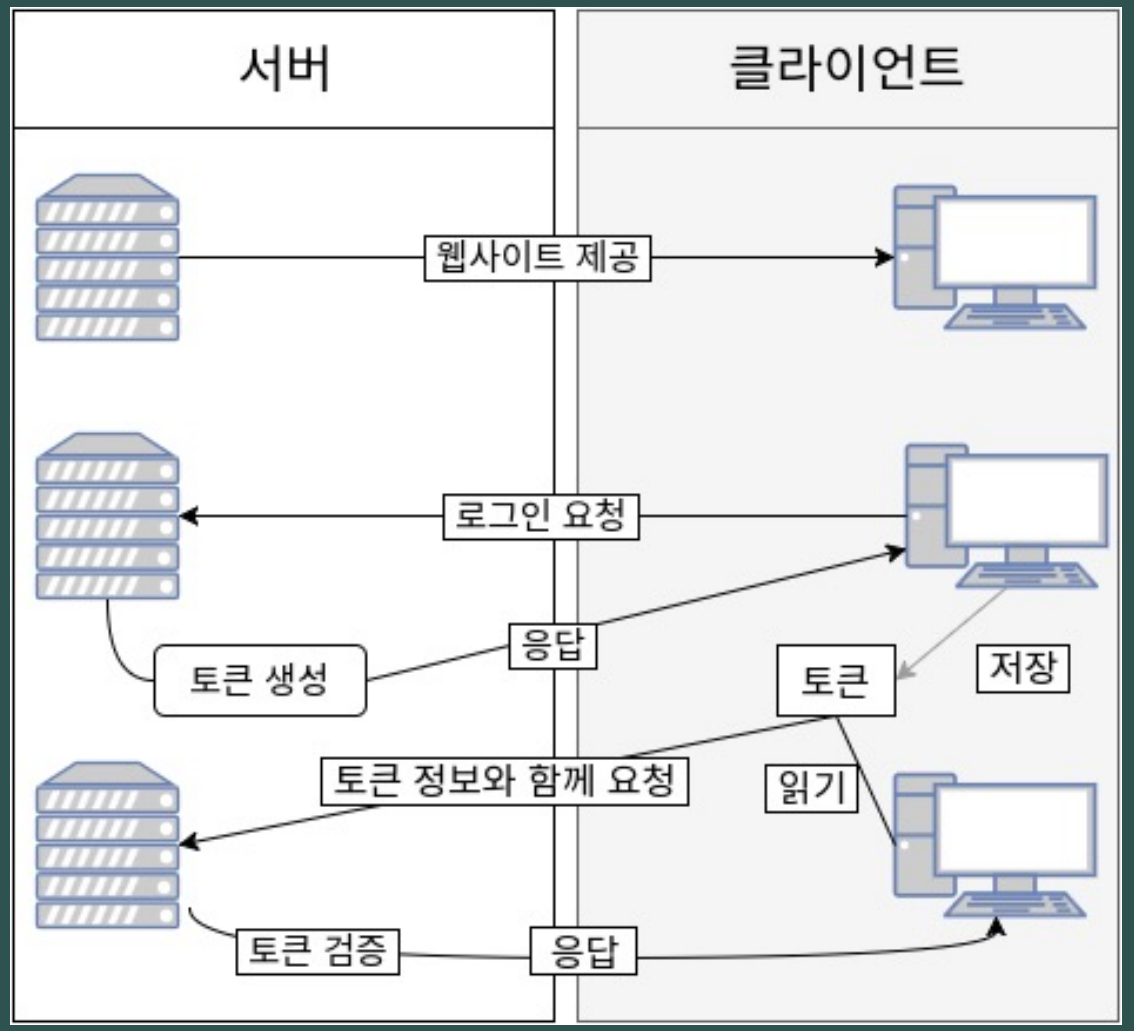
Session

JWT

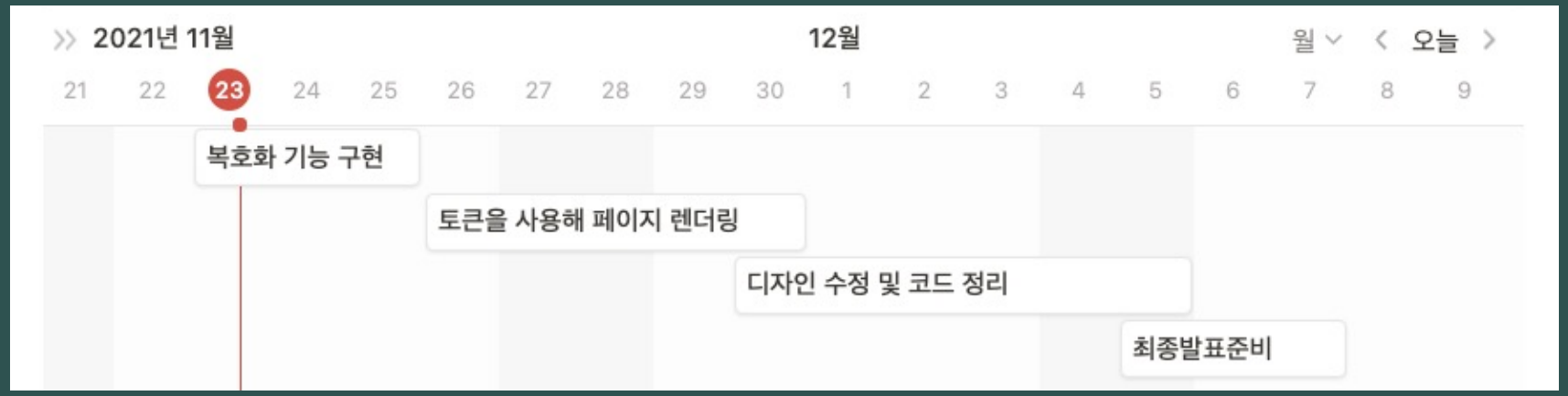
Session, Cookie



JWT(JSON WEB TOKEN)



Plan

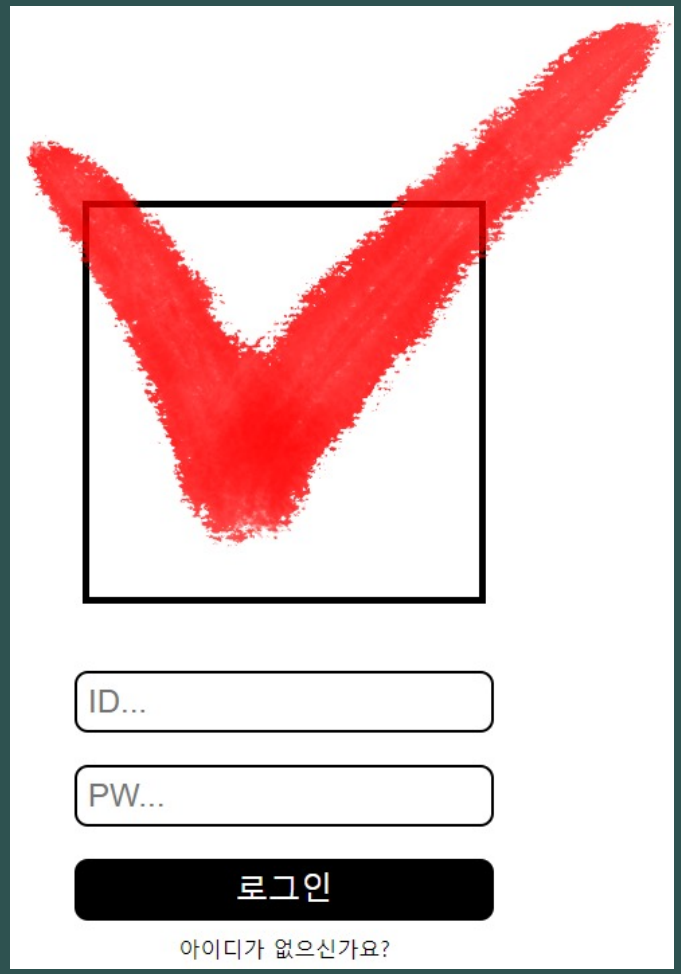


- JWT를 사용해 사용자에게 토큰을 부여하고 쿠키에 해당 내용을 저장해 사용자 별 페이지를 관리할 수 있도록 함
- 비밀번호 암호화 기능 구현

→

- 복호화 기능 구현
- 토큰을 통해 사용자 별로 페이지 렌더링
- 디자인 수정 및 코드 정리
- 최종 발표 준비

JWT - summary



id, pw

Token

서버

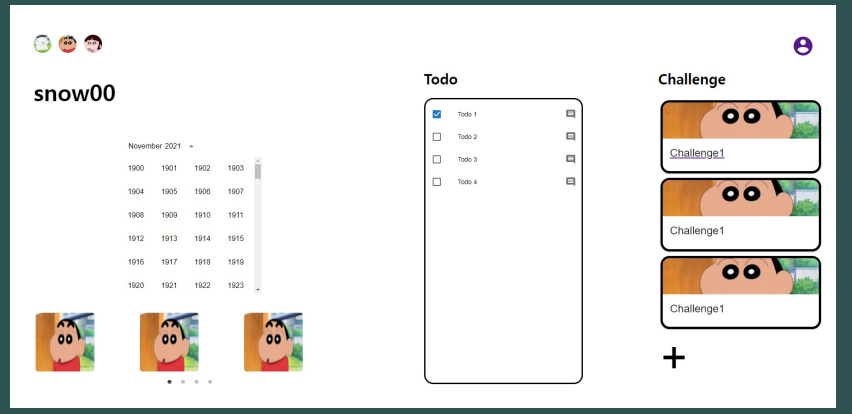
Server

SQL 문

DB

Token

data



JWT - code

```
48  const onClickLogin = ()=>{
49    axios
50      .post("http://localhost:5000/api/login", {
51        inputId,
52        inputPw,
53      })
54      .then((response) => {
55        const token = response['data']['accessToken'];
56
57        setCookie('myToken', token);
58
59        onInsertToggle();
60      })
61      .catch((error) => {
62        console.log(error);
63        onInsertToggle2();
64      });
65  };
```

LoginForm.js

JWT - code

```
app.post('/api/login', (req, res) => {
  let isUser = false;

  const userId = req.body.inputId;
  const userPassword = req.body.inputPw;
  const sql = 'SELECT id, password FROM management.user';
  db.query(sql, (err, rows, fields) => {
    if (err) {
      console.log(err);
    } else {
      rows.forEach((info) => {
        if (info.id === userId && info.password === userPassword) {
          isUser = true;
        } else {
          return;
        }
      });
```

```
if (isUser) {
  const accessToken = jwt.sign(
    {
      userId,
    },
    YOUR_SECRET_KEY,
    {
      expiresIn: '1h',
    }
  );
  res.cookie('user', accessToken);
  res.status(201).json({
    result: 'ok',
    accessToken,
  });
} else {
  res.status(400).json({ error: 'invalid user' });
}
```

server.js

JWT - code

```
48  const onClickLogin = ()=>{
49    axios
50      .post("http://localhost:5000/api/login", {
51        inputId,
52        inputPw,
53      })
54      .then((response) => {
55        const token = response['data']['accessToken'];
56
57        setCookie('myToken', token);
58
59        onInsertToggle();
60      })
61      .catch((error) => {
62        console.log(error);
63        onInsertToggle2();
64      });
65  };
```

LoginForm.js

JWT - code

```
67  function FeedPage() {
68    const [user, setUser] = useState('');
69    const token = getCookie('myToken');
70
71    useEffect(() => {
72      console.log(token);
73      async function loadData() {
74        await axios
75          .post('http://localhost:5000/api/feed', {
76            token: token,
77          })
78          .then((res) => {
79            const nickname = res.data.id;
80            console.log(res.data);
81            console.log(nickname);
82            setUser(nickname);
83          })
84          .catch((err) => {
85            console.log(err);
86          });
87      }
88      loadData();
89    });
```

FeedPage.js

JWT - code

```
44  app.post('/api/feed', (req, res) => {
45    const token = req.body.token;
46    let nickname;
47
48    const id = jwt.decode(token, YOUR_SECRET_KEY);
49    const sql = `SELECT nickname FROM management.user_info WHERE user_id = '${id.userId}'`;
50    db.query(sql, (err, row, fields) => {
51      if (err) {
52        console.log(err);
53      } else {
54        nickname = row[0].nickname;
55      }
56      res.status(201).json({
57        result: 'ok',
58        id: nickname,
59      });
60    });
61  });
```

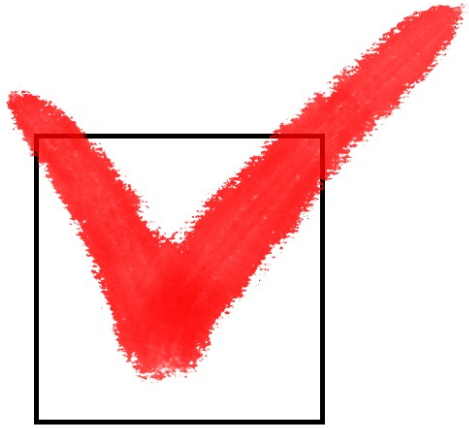
server.js

JWT - code

```
67  function FeedPage() {
68    const [user, setUser] = useState('');
69    const token = getCookie('myToken');
70
71    useEffect(() => {
72      console.log(token);
73      async function loadData() {
74        await axios
75          .post('http://localhost:5000/api/feed', {
76            token: token,
77          })
78          .then((res) => {
79            const nickname = res.data.id;
80            console.log(res.data);
81            console.log(nickname);
82            setUser(nickname);
83          })
84          .catch((err) => {
85            console.log(err);
86          });
87      }
88      loadData();
89    });
```

FeedPage.js

Screen



함께 목표를 이루어봅시다!

이용하러 가기

데이터 암호화

- 회원 가입 정보를 DB에 저장 할 때 Password가 그대로 DB에 저장 되는 문제가 있음
  - Database가 유출 되면 사용자의 정보가 그대로 드러날 것이기 때문에 보안상 매우 취약함

jin	test1
jinhwan	jinhwan
kim	kimPw

데이터 암호화

- bycrypt 방법을 사용하여 비밀번호를 암호화 하기로 결정

why bycrypt

- Hashing만을 사용하여 Rainbow table attack에 취약함

- Salting을 사용하는 암호화 방법 중 결정하자

why bycrypt

- SHA 암호화는 constant password length 문제 및 최신 CPU와 GPU에 취약함

- Blowfish를 추가한 bycrypt를 사용하여 처리하자

데이터 암호화

id	password
test1	0000
createAccount	111

id	password
user1	$2b$10$7tkWHGhL
registerTest	$2b$10$8.9oV2SA

Thank You