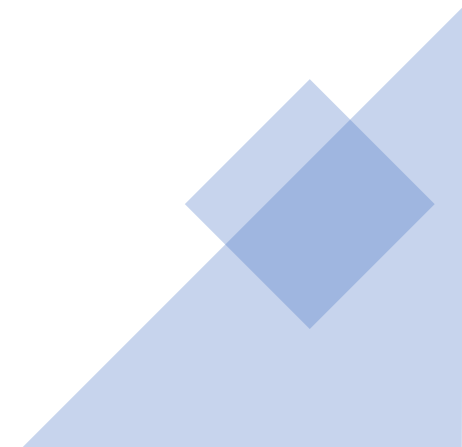


5월 4일 발표자료 - Web

1871062 김정한
1871069 김진웅

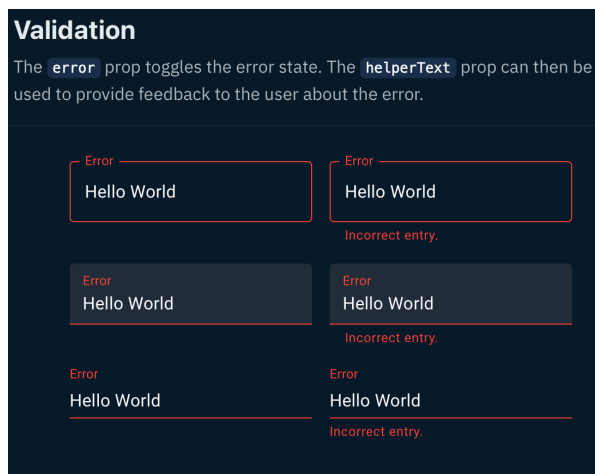


목차


- 1. 한 주간의 진행 상황
 - 1-1. 어떠한 것을 공부했는가
 - 1-2. 부딪힌 이슈 & 해결사항
 - 2. 고민하고 있는 issue
 - 3. 앞으로의 계획
 - 4. 기타
- 

1. 한 주간의 진행 상황

- 1-1. 어떠한 것을 공부했는가
 - 비밀번호 확인 기능 구현 완료



Mui library 이용



이때 모해?

이름 *

이메일 *

비밀번호 *

비밀번호 확인 *


생년월일

이메일 인증하기

인증번호 입력 *

회원가입

비밀번호가 같을 경우



이때 모해?

이름 *

이메일 *

비밀번호 *

비밀번호 확인 *

생년월일

이메일 인증하기

인증번호 입력 *

회원가입

비밀번호가 다를 경우

1. 한 주간의 진행 상황

- 1-1. 어떠한 것을 공부했는가
 - 비밀번호 확인 기능 구현 완료

```
<TextField className={styles.textField}
  error={passwordCheck}
  helperText={passwordCheck === true ? "비밀번호가 일치하지 않습니다." : undefined}
  margin="normal"
  required
  fullWidth
  name="passwordcheck"
  label="비밀번호 확인"
  type="password"
  id="passwordcheck"
  onInput={handleInputPasswordCheck}
/>
```

삼항연산자를 사용하여 passwordCheck 값이 true일 경우
error message, false일 경우 undefined 값 출력

```
const [passwordCheck, setPasswordCheck] = useState(false);
```

useState를 이용해 passwordCheck 상태값 설정
(에러 메시지는 초기에는 나타나지 않아야 하기 때문에 초기값은 false)

```
const handleInputPasswordCheck = (event) => {
  const { value } = event.target
  if (value !== passwordInput) {
    setPasswordCheck(true)
  } else {
    setPasswordCheck(false)
  }
}
```

passwordInput(비밀번호 입력 값) 값과 일치하는지 비교하여
passwordCheck값의 상태를 true or false로 실시간으로 변환

1. 한 주간의 진행 상황

• 1-1. 어떠한 것을 공부했는가

- 회원가입 페이지의 생년월일 표시기 설정, 데이터 값 저장 완료



이때 모해?

```
import { DemoContainer } from '@mui/x-date-pickers/internals/demo';
import { LocalizationProvider } from '@mui/x-date-pickers/LocalizationProvider';
import { AdapterDayjs } from '@mui/x-date-pickers/AdapterDayjs';
import { DatePicker } from '@mui/x-date-pickers/DatePicker';
import dayjs from 'dayjs';
```

```
<LocalizationProvider dateAdapter={AdapterDayjs}>
  <DemoContainer components={['DatePicker', 'DatePicker']} sx={{ width: '100%', '& .MuiFormControl-root': { width: '100%', mt: 1 } }}>
    <DatePicker
      label="생년월일"
      value={value}
      onChange={(newValue) => setValue(newValue)}
    />
  </DemoContainer>
</LocalizationProvider>
```

MUI 라이브러리 활용.

```
const [value, setValue] = useState(dayjs());
```

dayjs를 통해 날짜를 불러올 수 있다.
() 안에 아무 값도 집어넣지 않는다면



Default 값은 오늘 날짜로

1. 한 주간의 진행 상황

• 1-1. 어떠한 것을 공부했는가

- 로그인 페이지의 Key 값들을 Server 쪽으로 Post, Get 완료

```
import axios from 'axios'
```

Axios 라이브러리 활용

```
<Button
  type="submit"
  fullWidth
  variant="contained"
  sx={{ backgroundColor: '#0040ff', mt: 3, mb: 2 }}
  onClick={handleSubmit}
>
  로그인
</Button>
```

로그인 버튼 클릭 시 handleSubmit 핸들러 작동

```
const handleSubmit = async (event) => { // await을 사용하기 위해 async 함수로 선언
  event.preventDefault();
  const response = await axios.post('http://localhost:8080/auth/signin', { // axios는 항상 await과 함께 사용
    userId: emailInput,
    password: passwordInput
  });

  const { status, data } = response;
};
```

로그인 버튼을 누르면 서버에 값들을 보낼
handleSubmit 함수 설정

```
// 로그인 요청, POST
localhost:8080/auth/signin
{
  "userId" : "terry8408@gmail.com",
  "password": "yeah"
}
```

서버 프로토콜 통신 규칙에 맞춤.

1. 한 주간의 진행 상황

• 1-1. 어떠한 것을 공부했는가

- 회원가입 페이지의 Key 값들을 Server 쪽으로 Post, Get 완료

```
<Button
  type="submit"
  fullWidth
  variant="contained"
  sx={{ backgroundColor: '#0040ff', mt: 2, mb: 1, height: '45px' }}
  onClick={handleSubmit}
>
  회원가입
</Button>
```

회원가입 버튼 클릭 시 handleSubmit 핸들러 작동

```
const handleSubmit = async (event) => { //회원가입 버튼 이벤트 핸들러
  event.preventDefault();

  const responseEmail = await axios.post('http://13.125.122.132:5005/auth/email-verify', { //이메일 인증요청
    userId: emailInput,
    emailAuthCode: codeInput
  })

  if (responseEmail.data.includes('success')) { //만약 서버에서 보내준 값에 "success" 값이 있을 경우
    window.alert('이메일 인증에 성공했습니다!') //성공 alert를 띄운 후 그대로 실행
  } else {
    window.alert('이메일 인증에 실패했습니다!') //실패 alert를 띄운 후 그대로 종료
    return
  }

  //데이터값들을 서버 요청 양식에 맞게 재가공
  let month = birthInput.$M + 1 //월
  let day = birthInput.$D //일

  if (month < 10) { //10월 이전 생월일 경우
    month = '0' + month
  }

  if (day < 10) { //10일 이전 생일일 경우
    day = '0' + birthInput.$D
  }

  const birth = birthInput.$y + '-' + month + '-' + day //서버 요청 양식에 맞춘 새로운 birth값 생성

  const response = await axios.post('http://13.125.122.132:5005/auth/signup', { //회원가입 버튼 클릭시 서버로 post
    username: nameInput,
    userId: emailInput,
    password: passwordInput,
    birth: birth
  })

  const { status, data } = response;

  console.log(response)

  window.history.pushState('', '', 'localhost:3000/calendar') //calendar 페이지로 이동
};
```

```
SignUp
{
  sx: {},
  sy: 1998,
  $M: 11,
  $D: 11,
  $W: 5,
  $H: 0,
  $m: 36,
  $s: 48,
  $ms: 0,
  parse: f () {}
}
```

```
// 회원가입 시, POST
localhost:8080/auth/signup
{
  "userId": "terry8408@gmail.com",
  "password": "yeah",
  "userName": "윤재",
  "birth": "2010-10-22"
}
```

서버 프로토콜 통신 규칙에 맞춤.

1. 한 주간의 진행 상황

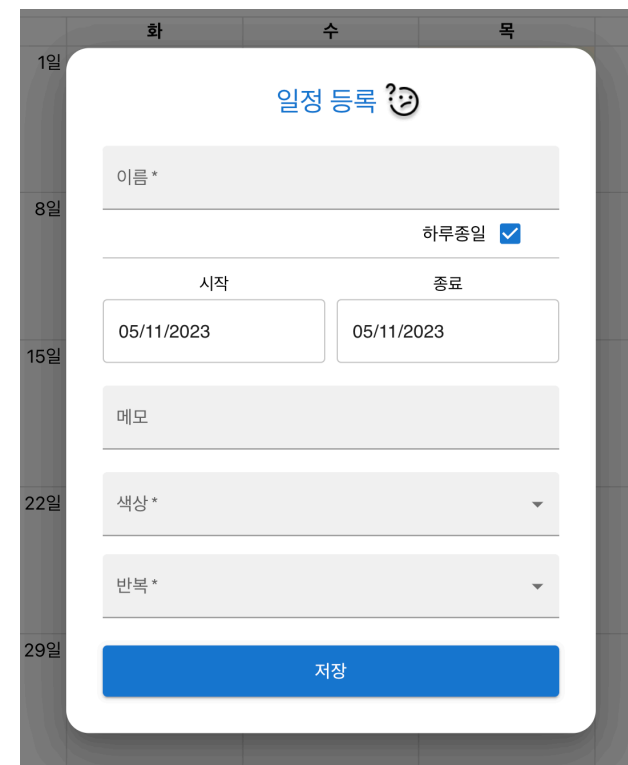
• 1-1. 어떠한 것을 공부했는가

- 캘린더 모달의 UI 구현 완료
- 하루종일 토글 클릭 시 오른쪽 시간 표시가 사라진다.



The image shows a calendar modal titled "일정 등록 ?" (Schedule Registration ?). It features a form with the following elements: a text input for "이름 *" (Name *), a checkbox for "하루종일" (All day) which is unchecked, two time selection boxes for "시작" (Start) and "종료" (End) both set to "04/24/2023 11:42 PM", a text area for "메모" (Memo), a dropdown for "색상 *" (Color *), and another dropdown for "반복 *" (Repeat *). A blue "저장" (Save) button is at the bottom. The background shows a calendar grid with dates from 1 to 24.

하루종일 토글이 체크가 안 되어 있을 경우



The image shows the same calendar modal as the previous one, but with the "하루종일" (All day) checkbox checked. In this state, the "시작" (Start) and "종료" (End) time selection boxes are disabled and show the date "05/11/2023". The "저장" (Save) button remains at the bottom. The background calendar grid shows dates from 1 to 29.

하루종일 토글이 체크 되어 있을 경우

1. 한 주간의 진행 상황

• 1-2. 부딪힌 이슈 & 해결사항

- 프론트에서 서버로 데이터를 전송할 경우 CORS 오류 발생
- bearer 을 코드에 넣어 오류 해결

```
✖ Access to XMLHttpRequest at 'http://13.125.122.132:8080/auth/email-send' from origin 'http://localhost:3000' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is present on the requested resource.
✖ ▶ POST http://13.125.122.132:8080/auth/email-send net::ERR_FAILED xhr.js:247
✖ ▶ Uncaught (in promise) Signup.jsx:104
  AxiosError {message: 'Network Error', name: 'AxiosError', code: 'ERR_NETWORK',
  config: {...}, request: XMLHttpRequest, ...}
```

오류 메세지

```
instance.interceptors.request.use(function (config) {
  const token = cookie.get('token') //쿠키의 토큰값을 가져와서 header에 넣어줌
  console.log('token', token)
  config.headers = {
    Authorization: `Bearer ${token}` //Authorization에 token값 넣기
  }

  // console.log(config)
  //요청이 전달되기 전에 작업 수행
  return config;
})
```

Bearer 코드 삽입

2. 고민하고 있는 이슈들

- 2-1. UI적인 요소들만 우선 전부 구현 후 나중에 기능적인 부분 구현 vs 기능적인 부분들 우선 전부 구현 후 나중에 UI적인 요소들 수정

3. 앞으로의 계획

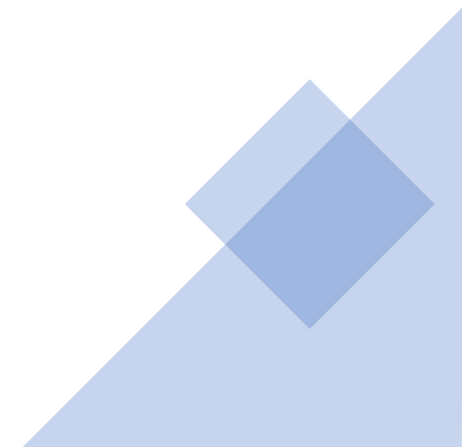
- 일정 추가, 수정, 삭제 기능 구현 후 실제 Server에 data가 잘 들어가는지 확인할 예정
- 일정을 추가하면 캘린더에 일정이 나타나는 기능 구현 예정
- 그룹 UI 작업 + logic 구현 예정

5월 4일 발표자료 - Server

1811072 유영재
1871197 이윤재



목차

- 1. 한 주간의 진행 상황
 - 1-1. 어떠한 것을 공부했는가
 - 1-2. 부딪힌 이슈 & 해결사항
 - 2. 앞으로의 계획
- 

1. 한 주간의 진행상황

- 1-1. 어떠한 것을 공부했는가
 - Front-server 연결 test

User 생성 성공

```
mysql> select * from Usr;
+-----+-----+-----+-----+-----+-----+-----+
| originKey | birth | createdAt | emailAuthCode | lastMod |
| ifiedAt | password | token | userId | userName |
+-----+-----+-----+-----+-----+-----+
| 8a8081b187e27c610187e27fe6bc0000 | NULL | 2023-05-03 16:44:05.732768 | 1800 | 2023-05-03 16:44:05.732788 | NULL | NULL | gtrew9606@naver.com | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Schedule 생성 성공

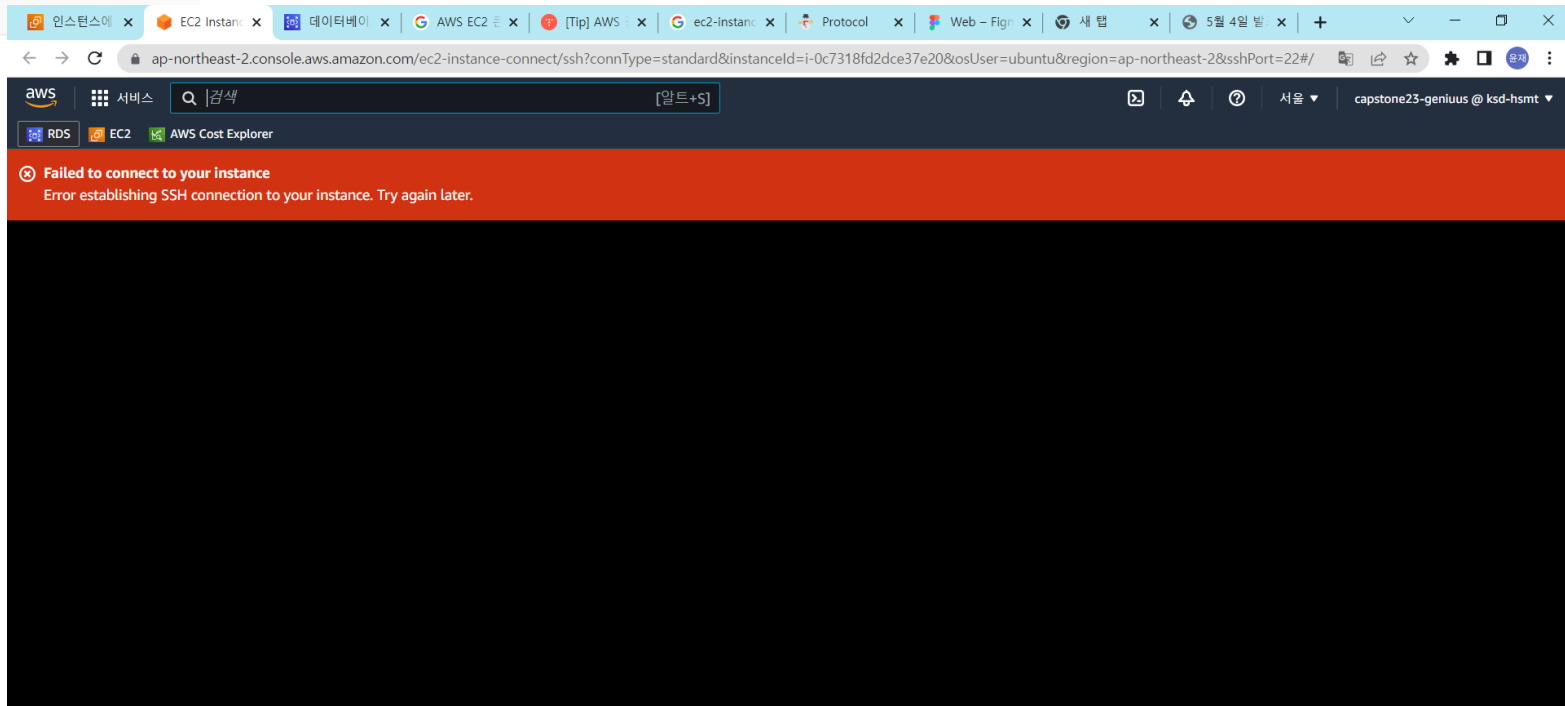
```
mysql> select * from Sche;
+-----+-----+-----+-----+-----+-----+-----+
| originKey | allDayToggle | createdAt | endAt |
| lastModifiedAt | memo | name | notification | startAt |
| userId |
+-----+-----+-----+-----+-----+-----+
| 8a8081b187e27c610187e281b7850001 | false | 2023-05-03 16:46:04.678035 | 2023-05-13T12:00 |
| 2023-05-03 16:46:04.678050 | 김정한 | 0 | 2023-05-11T12:00 | gtrew9606@naver.com |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

1. 한 주간의 진행상황

- 1-1. 어떠한 것을 공부했는가
 - AWS EC2 연결 제한

AWS 콘솔에서의 EC2 연결 차단

```
ubuntu:~$ sudo apt-get remove ec2-instance-connect
```



1. 한 주간의 진행상황

- 1-1. 어떠한 것을 공부했는가
 - Group CRUD 프로토콜
 - 코딩 진행 중...

• Create POST

```
JSON
// 그룹 생성 요청, POST
/group
{
  "groupName" : "뉴진스의 한입보이요",
  "description" : "2023 캡스톤디자인",
  "numOfUsers" : 10,
  "leaderId" : "cross_man@naver.com"
}

// 그룹 생성 응답
{
  "originKey" : "slkfnlarnflwsdjrlwsdjr",
  "groupName" : "뉴진스의 한입보이요",
  "description" : "2023 캡스톤디자인",
  "numOfUsers" : 10,
  "leaderId" : "cross_man@naver.com",
  "createdAt" : "2023-03-17-19:11",
  "lastModifiedAt" : "2023-03-17-19:11"
}
```

• Update PUT

```
JSON
// 수정하고 싶은 그룹의 originKey를 보내주면, 그룹 정보를 수정, 그룹장만 수정 가능, PUT
/group
{
  "originKey" : "slkfnlarnflwsdjrlwsdjr",
  "groupName" : "뉴진스의 한입보이요",
  "description" : "2023 캡스톤디자인"
}

// 그룹 정보 수정 응답
{
  "data" : [
    {
      "originKey" : "slkfnlarnflwsdjrlwsdjr",
      "groupName" : "뉴진스의 한입보이요",
      "description" : "2023 캡스톤디자인",
      "createdAt" : "2023-03-17-19:11",
      "lastModifiedAt" : "2023-03-17-19:11"
    }
  ],
  "status": "succeed",
  "error" : null
}
```

• Retrieve GET

```
JSON
// 요청을 보낸 사용자가 속한 그룹 검색 요청, GET
/group
body에 내용 없이 날리면 됨
{

}

// 그룹 검색 응답, 내가 속한 그룹들 반환
{
  "data" : [
    {
      "originKey" : "sdfsdfsdfsdfsdf",
      "groupOriginKey" : "sdfsdf",
      "description" : "2023 캡스톤디자인",
      "numOfUsers" : 10,
      "leaderId" : "cross_man@naver.com",
      "createdAt" : "2023-03-17-19:11",
      "lastModifiedAt" : "2023-03-17-19:11"
    }
  ],
  "status": "succeed",
  "error" : null
}
```

• Delete DELETE

```
JSON
// 그룹 삭제, 그룹장만 가능, DELETE
/group
{
  "originKey" : "slkfnlarnflwsdjrlwsdjr"
}
```


1-2. 부딪힌 이슈 & 해결사항

- 1-2. 부딪힌 이슈
 - Front – Back 연결 이슈(해결)

```
✖ ▶ POST http://13.125.238.167:8080/auth/email-send xhr.js:247 ↕  
net::ERR_CONNECTION_TIMED_OUT  
  
✖ ▶ Uncaught (in promise) Signup.jsx:103  
AxiosError {message: 'Network Error', name: 'AxiosError', code: 'ERR_NETWORK', config: {...}, request: XMLHttpRequest, ...}
```

This generated password is for development use only. Your security configuration must be updated before running your application in production.

```
2023-04-20 13:39:08.108 INFO 1270 --- [main] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web.session.  
isableEncodeUrlFilter@59ce792e, org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@4860827a, org.springframework.security.web.conte  
curityContextPersistenceFilter@5793b87, org.springframework.security.web.header.HeaderWriterFilter@3050ac2f, org.springframework.web.filter.CorsFilter@404db674, com.ex  
e.demo.security.JwtAuthenticationFilter@3f390d63, org.springframework.security.web.authentication.logout.LogoutFilter@4bc33720, org.springframework.security.web.savedre  
t.RequestCacheAwareFilter@512575e9, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@33634f04, org.springframework.security.web.authen  
ation.AnonymousAuthenticationFilter@50f097b5, org.springframework.security.web.session.SessionManagementFilter@15405bd6, org.springframework.security.web.access.Excepti  
anslationFilter@617389a, org.springframework.security.web.access.intercept.FilterSecurityInterceptor@7c2dfa2]  
2023-04-20 13:39:08.695 INFO 1270 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''  
2023-04-20 13:39:08.727 INFO 1270 --- [main] com.example.demo.CapstoneApplication : Started CapstoneApplication in 11.822 seconds (JVM running for 13.09
```

2. 앞으로의 계획

- 2. 계획

- 그룹 CRUD 일정이 한주 밀림
- 그룹& 그룹 일정 CRUD 한번에 해결할 예정

3주차 (3/12 ~ 3/18)	요청 프로토콜 만들거, Controller와 Entity 먼저 구성
4주차 (3/18 ~ 3/25)	EC2 서비스 환경구축, DB 연결, 회원가입/로그인 기능 구현
5주차 (3/25 ~ 4/1)	Firebase Authentication
6주차(4/2 ~ 4/8)	일정 CRUD
7주차(4/9 ~ 4/15)	일정 CRUD
8주차(4/16 ~ 4/22)	그룹 CRUD
9주차(4/23 ~ 4/29)	그룹 CRUD
10주차(4/30 ~ 5/6)	그룹 일정 CRUD
11주차(5/7 ~ 5/13)	그룹 일정 CRUD
12주차(5/14 ~ 5/20)	테스팅
13주차(5/21 ~ 5/27)	테스팅