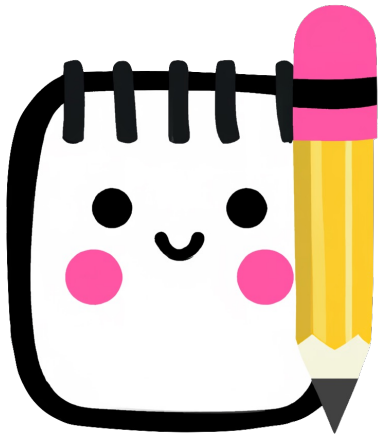


Collab Note



중간 발표

2023년 1학기 WAP WEB TEAM 1

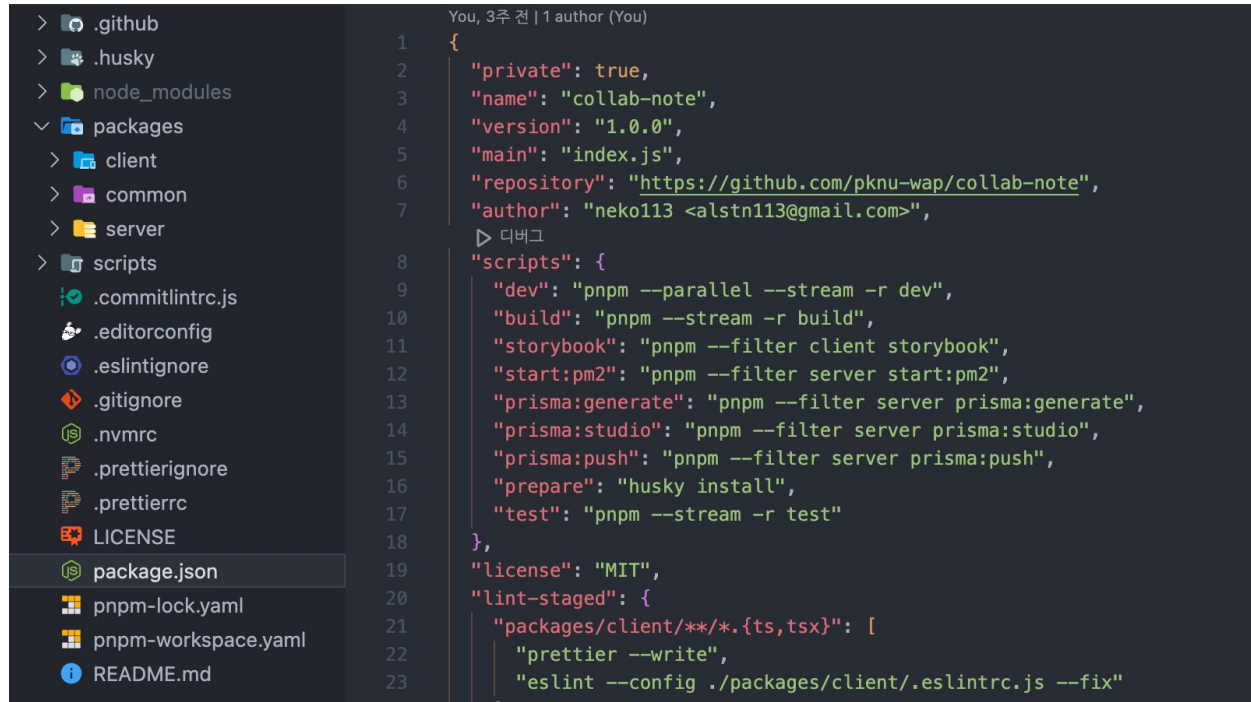
201911430 김민수

목차

1. Monorepo
2. CI
3. CD
4. Server
5. Client

6. Comon Workspace
7. WebRTC
8. CRDT
9. 개발 일정
10. QnA

MONOREPO

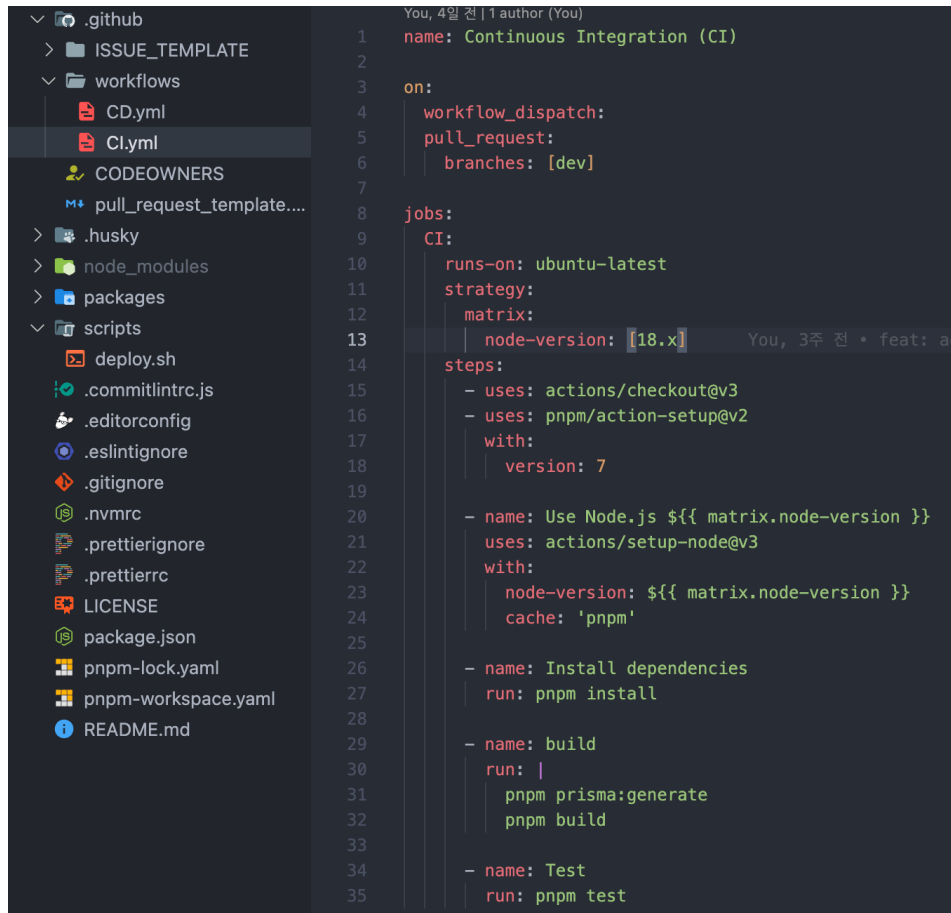


The image shows a file explorer on the left and a code editor on the right. The file explorer displays the root directory of a monorepo with the following files and folders: .github, .husky, node_modules, packages (containing client, common, and server), scripts, .commitlintrc.js, .editorconfig, .eslintignore, .gitignore, .nvmrc, .prettierignore, .prettierrc, LICENSE, package.json, pnpm-lock.yaml, pnpm-workspace.yaml, and README.md. The code editor shows the content of package.json, which is a pnpm workspace configuration. It includes fields for private, name, version, main, repository, author, scripts, license, and lint-staged. The scripts section defines commands for development, building, running storybook, starting the server, generating prisma, pushing prisma, installing husky, and testing. The lint-staged section defines a pre-commit hook that runs prettier and eslint on the client packages.

```
1 {
2   "private": true,
3   "name": "collab-note",
4   "version": "1.0.0",
5   "main": "index.js",
6   "repository": "https://github.com/pknu-wap/collab-note",
7   "author": "neko113 <alstn113@gmail.com>",
8   "scripts": {
9     "dev": "pnpm --parallel --stream -r dev",
10    "build": "pnpm --stream -r build",
11    "storybook": "pnpm --filter client storybook",
12    "start:pm2": "pnpm --filter server start:pm2",
13    "prisma:generate": "pnpm --filter server prisma:generate",
14    "prisma:studio": "pnpm --filter server prisma:studio",
15    "prisma:push": "pnpm --filter server prisma:push",
16    "prepare": "husky install",
17    "test": "pnpm --stream -r test"
18  },
19  "license": "MIT",
20  "lint-staged": {
21    "packages/client/**/*.ts,tsx": [
22      "prettier --write",
23      "eslint --config ./packages/client/.eslintrc.js --fix"
24    ]
25  }
26 }
```

* 공유 코드를 root workspace에 두고, server, client, common으로 분리해서 작업했음. Server는 Nestjs, Client는 React, Common에는 공유 상수, 및 CRDT 코드들이 있습니다.

Continuous Integration

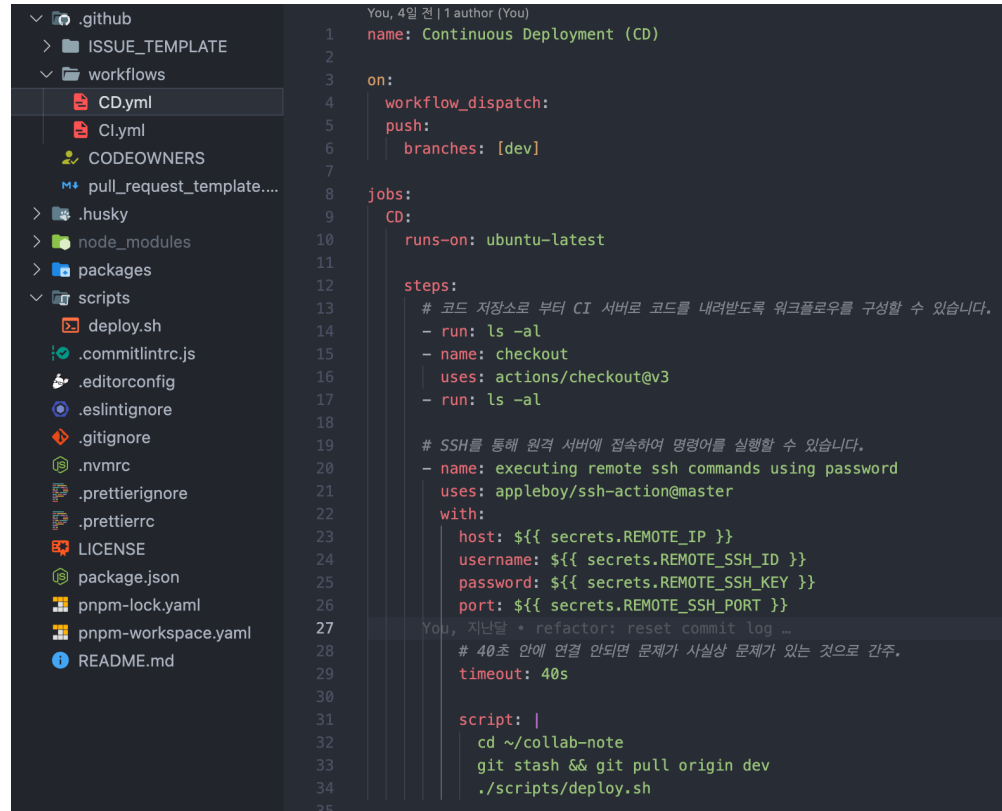


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a repository structure with files like .github, ISSUE_TEMPLATE, workflows, CD.yml, CI.yml, CODEOWNERS, pull_request_template..., .husky, node_modules, packages, scripts, deploy.sh, .commitlintrc.js, .editorconfig, .eslintignore, .gitignore, .nvmrc, .prettierrc, LICENSE, package.json, pnpm-lock.yaml, pnpm-workspace.yaml, and README.md. The code editor shows the content of CI.yml, which is a GitHub Actions workflow. The workflow is triggered on pull requests to the dev branch. It includes a job named CI that runs on ubuntu-latest, using a matrix strategy to test different Node.js versions (18.x). The steps include checking out the code, setting up pnpm, setting up Node.js, installing dependencies, building the project, and running tests.

```
1 name: Continuous Integration (CI)
2
3 on:
4   workflow_dispatch:
5   pull_request:
6     branches: [dev]
7
8 jobs:
9   CI:
10    runs-on: ubuntu-latest
11    strategy:
12      matrix:
13        node-version: [18.x]
14    steps:
15      - uses: actions/checkout@v3
16      - uses: pnpm/action-setup@v2
17        with:
18          version: 7
19      - name: Use Node.js ${{ matrix.node-version }}
20        uses: actions/setup-node@v3
21        with:
22          node-version: ${{ matrix.node-version }}
23          cache: 'pnpm'
24      - name: Install dependencies
25        run: pnpm install
26      - name: build
27        run: |
28          pnpm prisma:generate
29          pnpm build
30      - name: Test
31        run: pnpm test
```

* 지속적 통합 (CI)를 적용해서 dev branch에 pull_request를 할 경우 git actions를 통해서 server, common, client의 test code를 작동시킨다.

Continuous Deployment

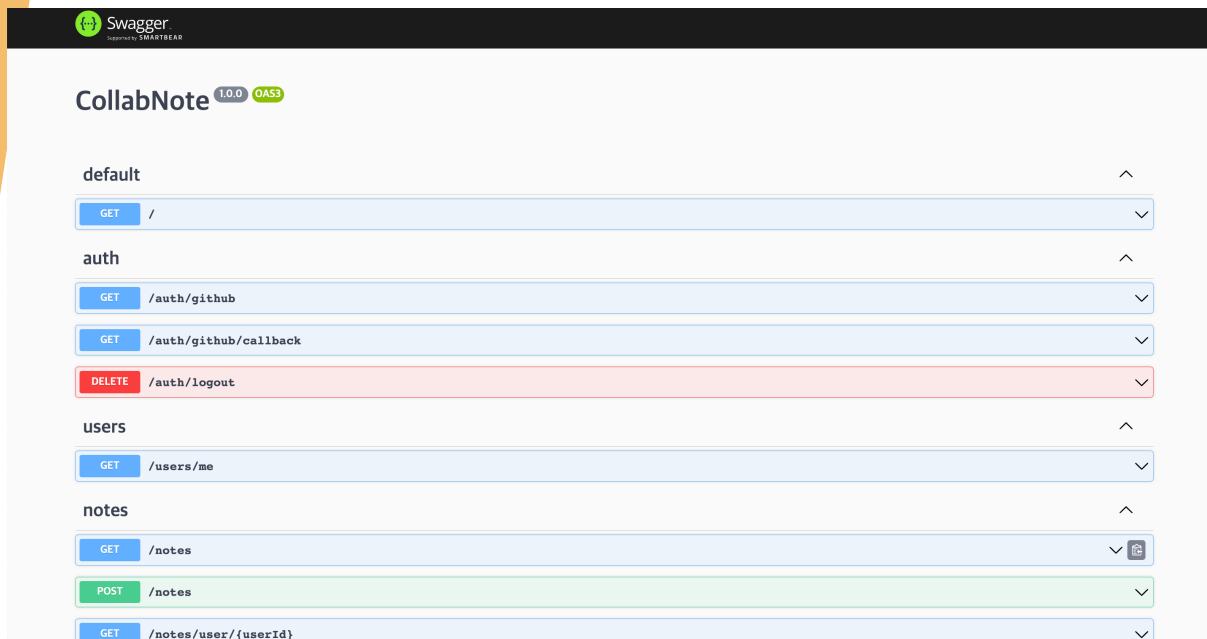


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a repository structure with files like .github, ISSUE_TEMPLATE, workflows, CD.yml, CI.yml, CODEOWNERS, pull_request_template..., .husky, node_modules, packages, scripts, deploy.sh, .commitlintrc.js, .editorconfig, .eslintignore, .gitignore, .nvmrc, .prettierignore, .prettierrc, LICENSE, package.json, pnpm-lock.yaml, pnpm-workspace.yaml, and README.md. The code editor shows the content of CD.yml, which is a GitHub Actions workflow for Continuous Deployment. The workflow is triggered on a push to the dev branch. It has a job named CD that runs on ubuntu-latest. The job has two steps: a checkout step and an SSH step that runs a script to deploy the code to a remote server.

```
1  name: Continuous Deployment (CD)
2
3  on:
4    workflow_dispatch:
5    push:
6      branches: [dev]
7
8  jobs:
9    CD:
10     runs-on: ubuntu-latest
11
12     steps:
13       # 코드 저장소로부터 CI 서버로 코드를 내려받도록 워크플로우를 구성할 수 있습니다.
14       - run: ls -al
15       - name: checkout
16         uses: actions/checkout@v3
17       - run: ls -al
18
19       # SSH를 통해 원격 서버에 접속하여 명령어를 실행할 수 있습니다.
20       - name: executing remote ssh commands using password
21         uses: appleboy/ssh-action@master
22         with:
23           host: ${ secrets.REMOTE_IP }
24           username: ${ secrets.REMOTE_SSH_ID }
25           password: ${ secrets.REMOTE_SSH_KEY }
26           port: ${ secrets.REMOTE_SSH_PORT }
27
28       # 40초 안에 연결 안되면 문제가 사실상 문제가 있는 것으로 간주.
29       timeout: 40s
30
31       script: |
32         cd ~/collab-note
33         git stash && git pull origin dev
34         ./scripts/deploy.sh
```

* 지속적 배포 (CD)를 적용해서 dev branch에 pull_request를 할 경우 git actions를 통해서 AWS EC2의 서버에서 script를 작동시킨다. 이후 server, common, client를 빌드하고 pm2를 통해서 서버를 실행한다.

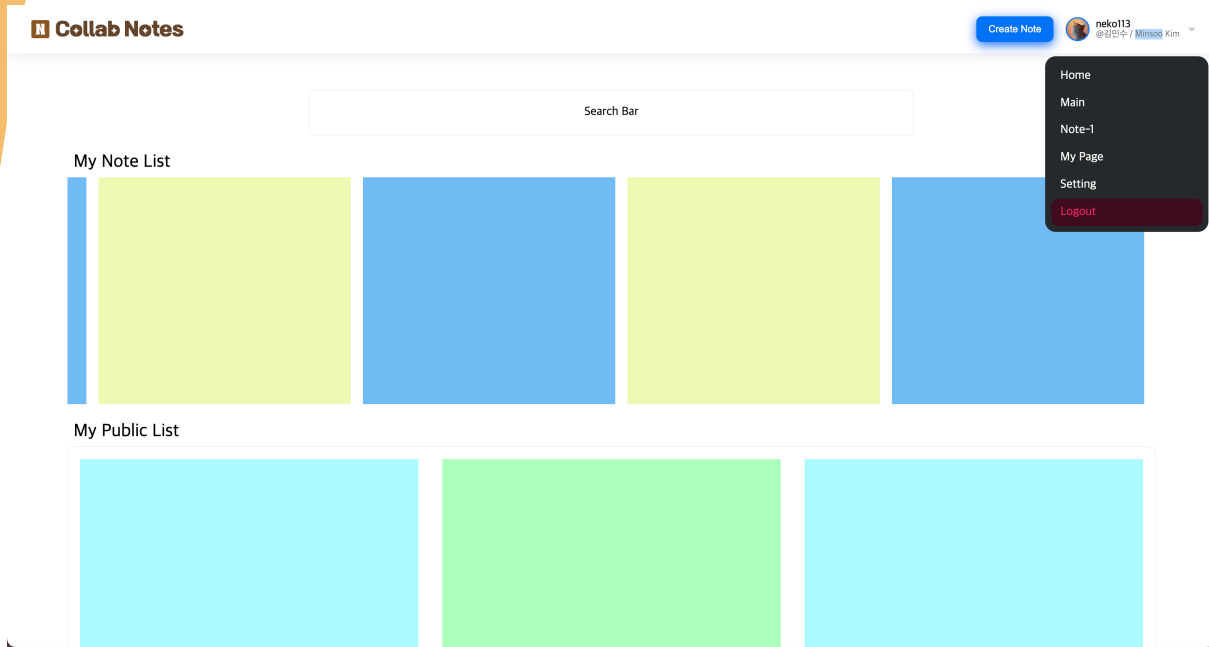
Server Workspace



* Nestjs 프레임워크에서 Prisma ORM을 사용했고, Socket 라이브러리와 common workspace에 있는 코드로 CRDT 코드를 구성했다. 또한 Client의 WebRTC를 위한 Socket 코드도 있다.

* Swagger를 적용해 서버에서 각 route를 실행하고 테스트했다.

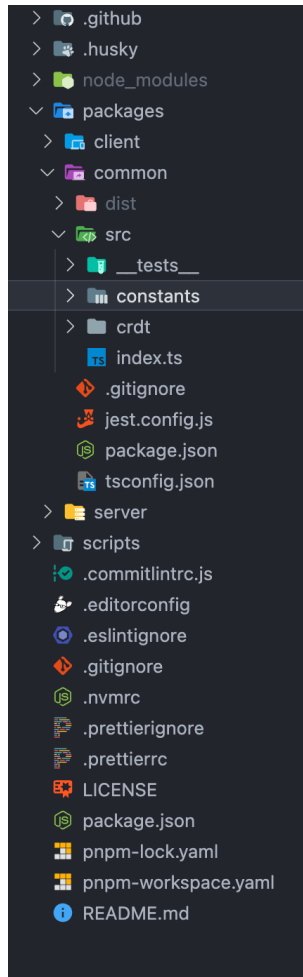
Client



* Client의 Home Page로
My Note List는 Carousel
Component를 만들어 수평
스크롤로 보이게 할 예정입니다.

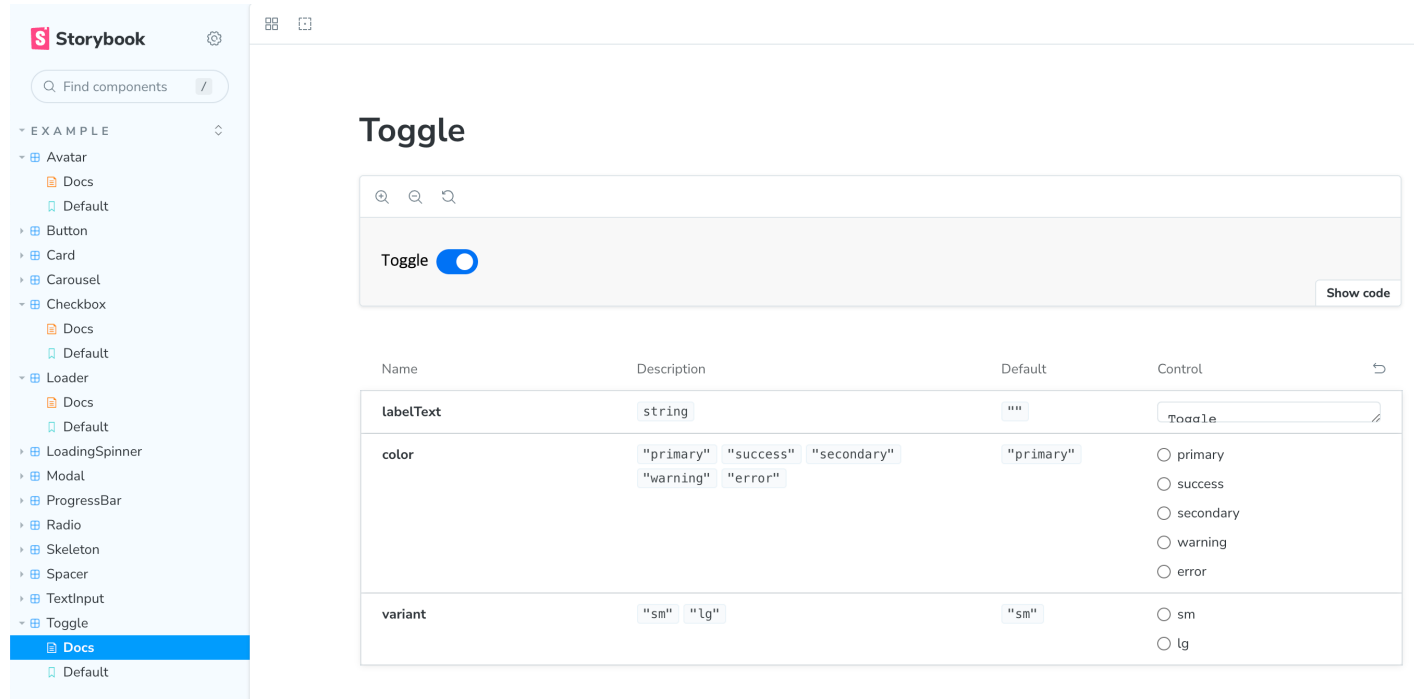
* My Public List는 수직 스크롤로
밑으로 부한 스크롤되게 할
예정입니다.

Common Workspace



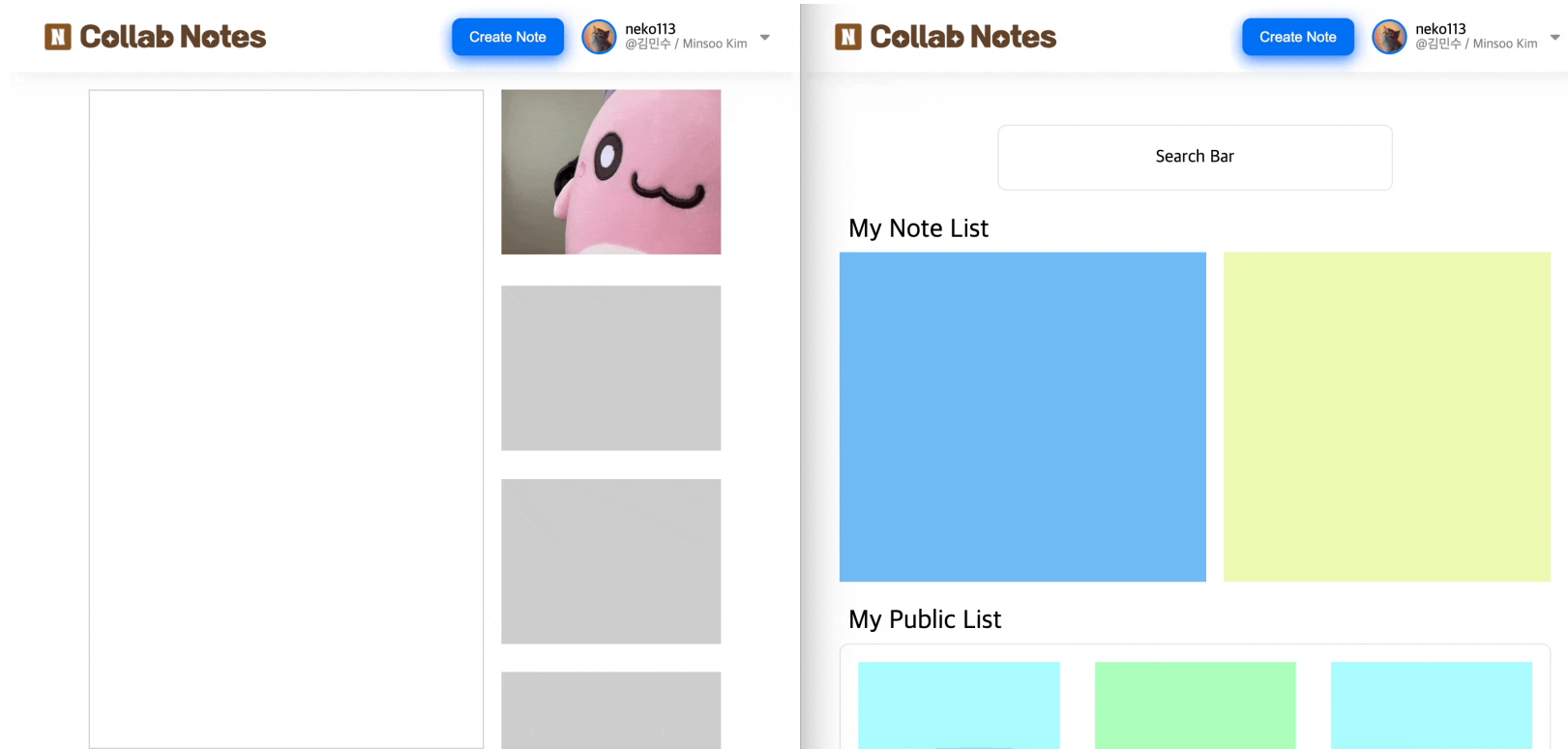
* Common workspace는
다음과 같이
공유 상수가 있고, server와
client에서 같이 사용할 CRDT
class 코드가 있습니다.

Common Component



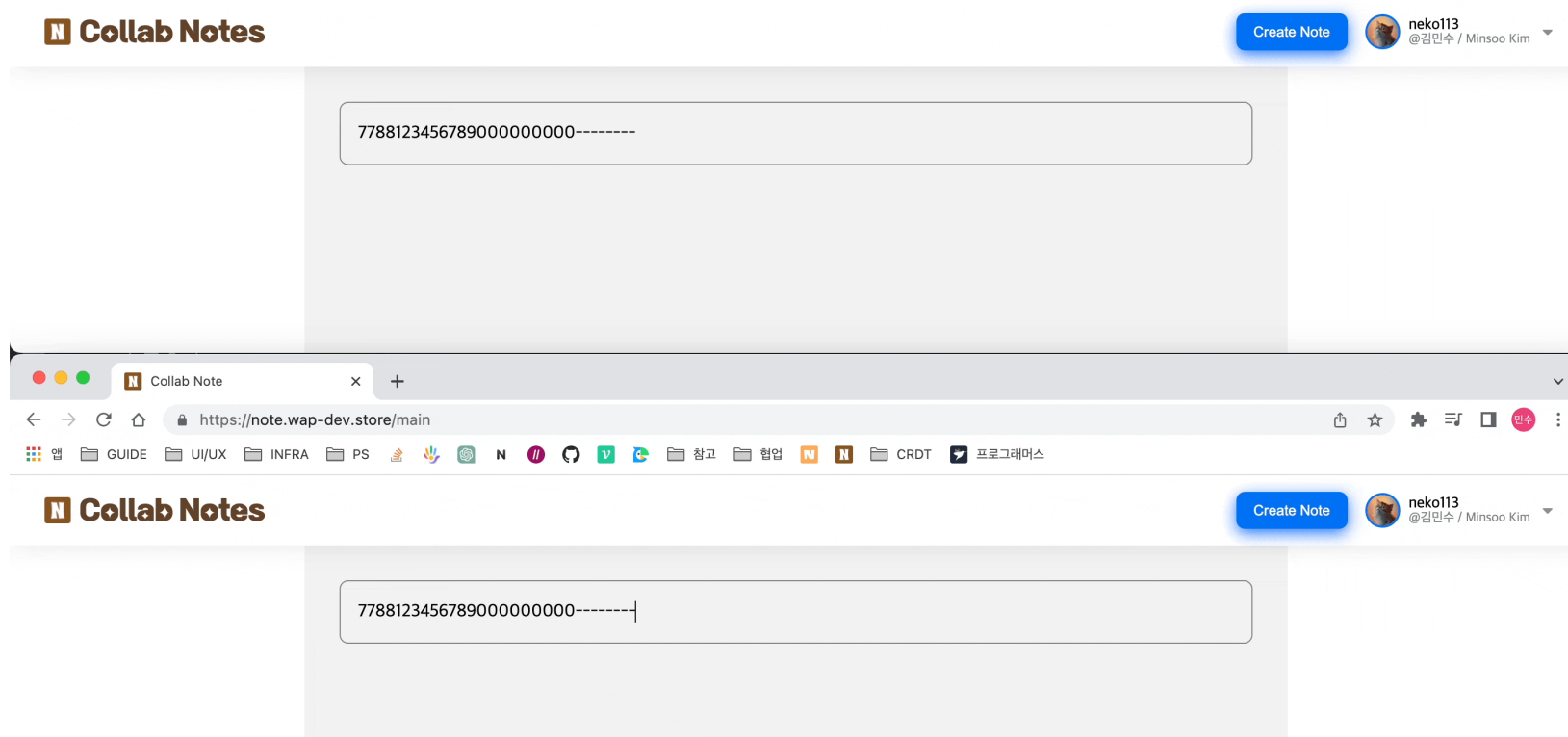
* 재사용 컴포넌트들을 만들고 Storybook으로 확인, 및 공유합니다.

WebRTC



* Client에서 WebRTC를 통해 브라우저 간 영상 공유를 할 수 있다.

CRDT



* Linked List를 통해 CRDT를 만들어서 Server에서 delay없이
동시 편집을 할 수 있다.

개발 일정

	1주차 03/20~03/26	2주차 03/27~04/02	3주차 04/03~04/09	4주차 04/10~04/16	5주차 04/17~04/23	6주차 04/24~04/30
WebRTC	WebRTC 공부		Class 구현		중간 고사	
CRDT	CRDT 공부					
UI	Common Component 제작					

	7주차 05/01~05/07	8주차 05/08~05/14	9주차 05/15~05/21	10주차 05/22~05/28	11주차 05/29~06/04	12주차 06/05~06/11
WebRTC	Hook 구현	Manager 구현	Refactoring			정리
CRDT				Refactoring		
UI	Page. Component Detail					

QnA

