비스킷(biscuIT) 포팅 매뉴얼

1. 개요

1.1. 프로젝트 소개

개발자를 위한 `시간 맞춤` 테크 컨텐츠 큐레이션 서비스

1.2. 개발 환경

형상관리

- Gitlab

협업도구

- Discord

UI/UX

- Figma

이슈관리

- Jira
- Notion

IDE

- Visual Studio Code 1.75
- IntelliJ IDEA 2022.3.1
- DataGrip 2022.3.2

Database

- MariaDB 10.11.2
- Redis 7.0.10

Front-end

- React 18.2.0
  - Recoil 0.7.7
  - React-query 4.28.0
- Typescript 4.9.5
- Node 16.18.14
  - npm 9.6.2
  - yarn 1.22.19

Back-end

- JAVA JDK 11.0.9
- SpringBoot Gradle 2.7.9
  - Spring Data JPA
  - Spring Security
  - Lombok

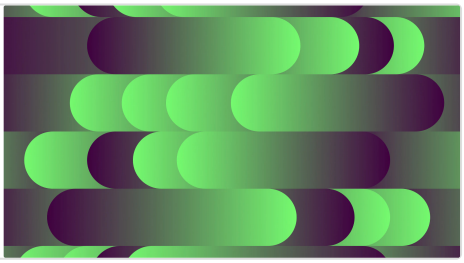
- vite 4.1.4

1.3. 외부 서비스 및 문서

퀴즈 생성 : OpenAI API

컨텐츠 별 사용자의 학습내용을 확인할 수 있는 퀴즈 생성

> Introducing ChatGPT and Whisper APIs
>
> Developers can now integrate ChatGPT and Whisper models into their apps and products through our API.
>
> https://openai.com/blog/introducing-chatgpt-and-whisper-apis



1.4. `.gitignore` 처리한 핵심 키 파일

Spring Boot

- `application.yml` : Spring Boot Application 설정 파일

2. 빌드

2.1. 환경 변수 형태

2.1.1. Spring Boot: `application.yml`

```
spring:
  datasource:
    username:
    password:
  security:
    oauth2:
      client:
        registration:
          google:
            client-id:
            client-secret:
            scope:
          github:
            client-id:
```

```
            client-secret:
            scope:
  redis:
    password:

jwt:
  secret:
```

2.2. 빌드하기

2.2.1. 프론트엔드: React

```
yarn build
```

2.2.2. 백엔드: Spring

```
./gradlew clean build --exclude-task test
```

2.3. 배포하기

2.3.1. Docker 이미지

도커 이미지를 사용하면 아래 과정을 거치지 않아도 됩니다.

```
docker pull evekristin/biscuit-service:api
docker pull evekristin/biscuit-react:front
```

2.3.2. `docker-compose.yml`

```
version: "3.1"

services:
        proxy:
                image: nginx:latest
                ports:
                        - "80:80"
                        - "443:443"
                volumes:
                        - ./proxy/nginx.conf:/etc/nginx/nginx.conf
                        - /var/log/nginx:/var/log/nginx
                        - /etc/letsencrypt:/etc/letsencrypt
```

```
                container_name: proxy
                depends_on:
                        - biscuit-react
                        - biscuit-service
                restart: always
                environment:
                        - TZ=Asia/Seoul

        mariadb:
                container_name: mariadb
                image: mariadb
                environment:
                        MYSQL_DATABASE:
                        MYSQL_USER:
                        MYSQL_PASSWORD:
                        MYSQL_ROOT_PASSWORD:
                        TZ: Asia/Seoul
                command:
                        - --character-set-server=utf8mb4
                        - --collation-server=utf8mb4_unicode_ci
                volumes:
                        - /var/lib/docker/volumes/mariadb/_data:/var/lib/mysql
                ports:
                        - "3306:3306"
                restart: always
        biscuit-service:
                container_name: biscuit-service
                image: evekristin/biscuit-service:api
                expose:
                        - 8080
                depends_on:
                        - mariadb
                restart: always
                environment:
                        TZ: Asia/Seoul

        biscuit-react:
                container_name: biscuit-react
                image: evekristin/biscuit-react:front
                expose:
                       - 80
        jenkins:
                image: jenkins/jenkins
                privileged: true
                restart: always
                ports:
                        - "9090:8080"
                expose:
                        - 50000
                volumes:
                        - '/jenkins:/var/jenkins_home'
                        - '/var/run/docker.sock:/var/run/docker.sock'
                        - '/usr/bin/docker:/usr/bin/docker'
                environment:
                        TZ: "Asia/Seoul"
                user: root

        redis:
```

```
                container_name: redis
                image: redis
                ports:
                        - "6379:6379"
                command:
                hostname: biscuit
```

2.3.3. React Container: `Dockerfile`

`docker build -f Dockerfile -t evekristin/biscuit-react:front .`

`docker run -d -p 3000:80 --name biscuit-react evekristin/biscuit-react:front`

```
###### builder stage ######
FROM node:18.12.1 as builder

# a. 작업 폴더를 만들고 npm 설치
#RUN mkdir /usr/src/app
WORKDIR /usr/src/app
ENV PATH /usr/src/app/node_modules/.bin:$PATH
COPY package.json /usr/src/app/package.json

#RUN npm install yarn --silent
RUN npm install npm@9.6.2
RUN yarn install

# b. 소스를 작업폴더로 복사하고 빌드
COPY . /usr/src/app
RUN yarn add tailwindcss@3.2.7
RUN yarn build

EXPOSE 3000

###### run stage ######
FROM nginx:latest
# c. nginx의 기본 설정을 삭제하고 앱에서 설정한 파일을 복사
RUN rm -rf /etc/nginx/conf.d
COPY conf /etc/nginx

# d. 위에서 생성한 앱의 빌드산출물을 nginx의 샘플 앱이 사용하던 폴더로 이동
COPY --from=builder /usr/src/app/dist /usr/share/nginx/html

# e. [원하는 포트]번 포트 내부에 오픈하고 nginx 실행
    # nginx라서 기본 80번
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

2.3.4. Spring Container: `Dockerfile`

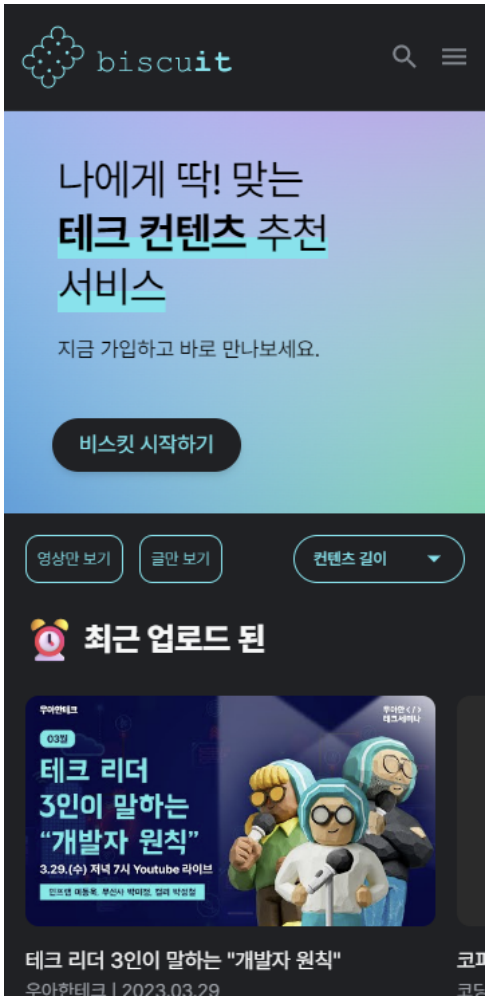
`docker build -t evekristin/biscuit-service:api .`

`docker run -d -p 8080:8080 --name biscuit-service evekristin/biscuit-service:api`

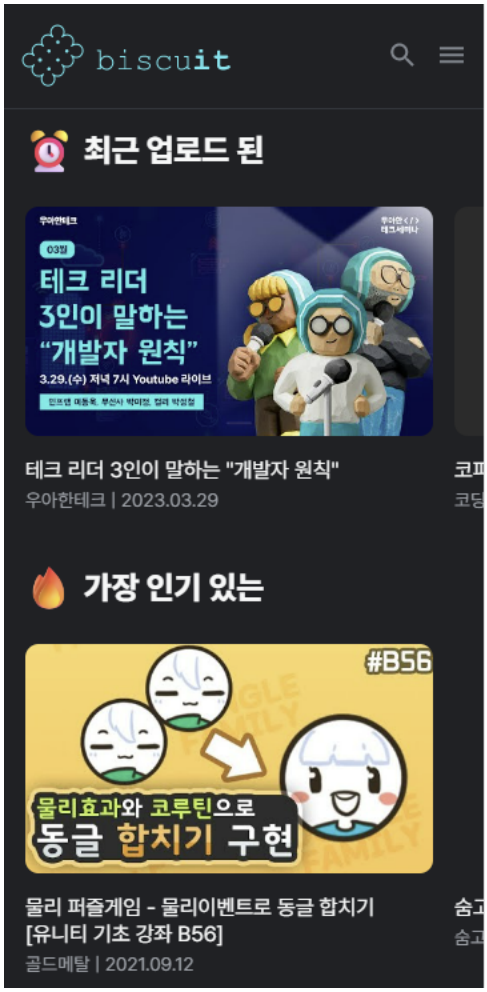
```
FROM gradle:jdk11 as builder

ENV APP_HOME=/apps

WORKDIR $APP_HOME

COPY build.gradle settings.gradle $APP_HOME/

COPY src $APP_HOME/src

RUN gradle clean build

FROM openjdk:11-jdk

ENV APP_HOME=/apps
ARG JAR_FILE_PATH=build/libs/biscuIT-0.0.1-SNAPSHOT.jar

WORKDIR $APP_HOME
COPY --from=builder $APP_HOME/$JAR_FILE_PATH app.jar

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar"]
```

3. 시연 시나리오

3.1. 비회원 홈

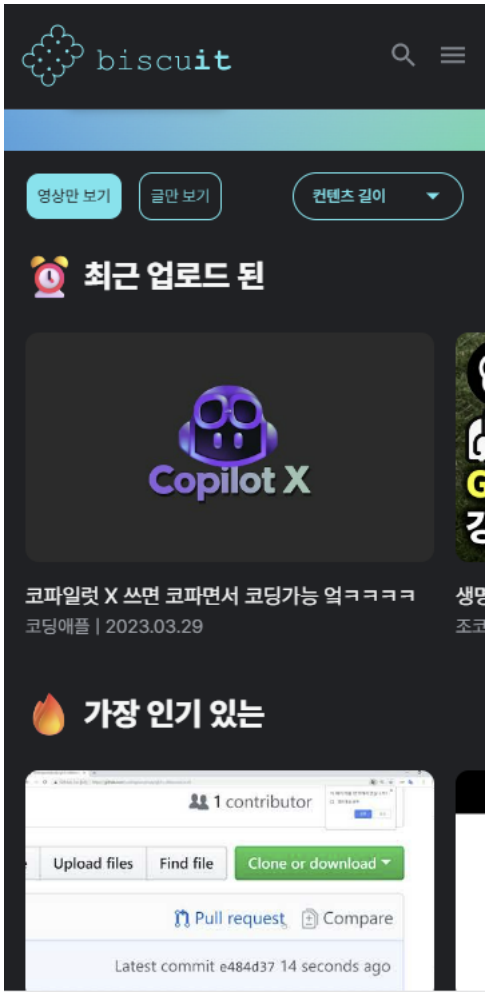


1. 비로그인시 홈 화면

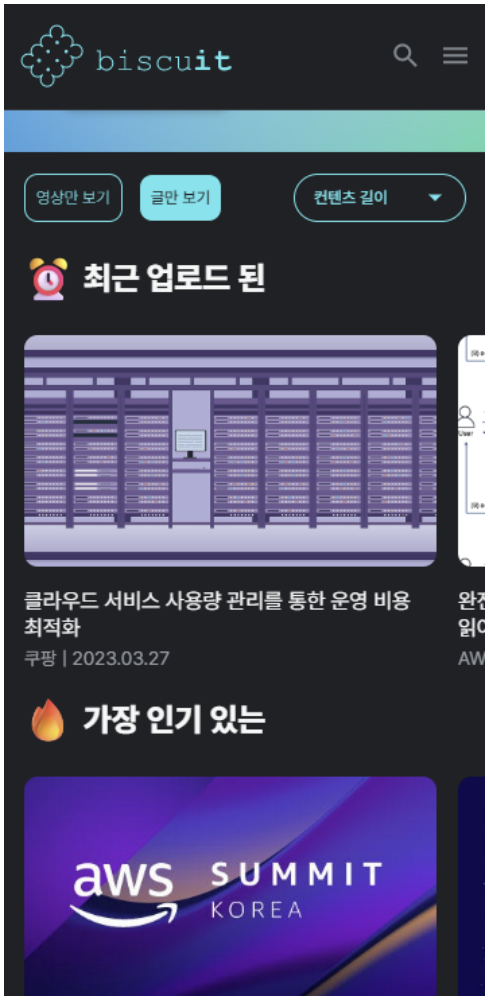


2. 랜덤 카테고리 및 기본 추천

3.1.1. 컨텐츠 종류 필터

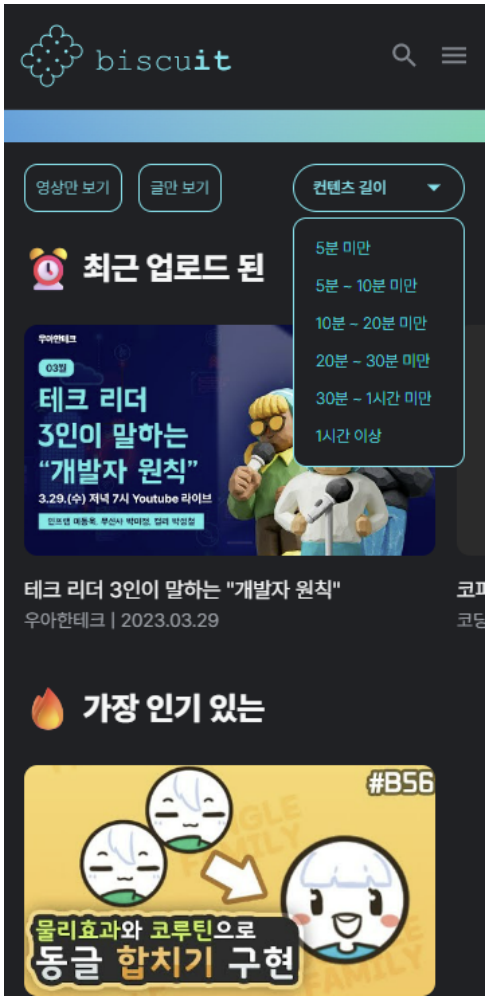


3. [영상만 보기] 필터

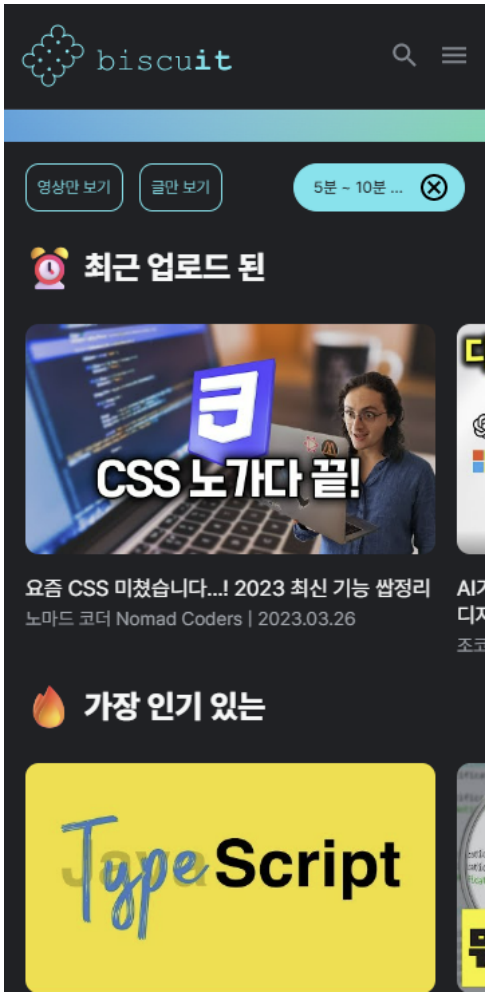


4. [글만 보기] 필터

3.1.2. 시간 필터

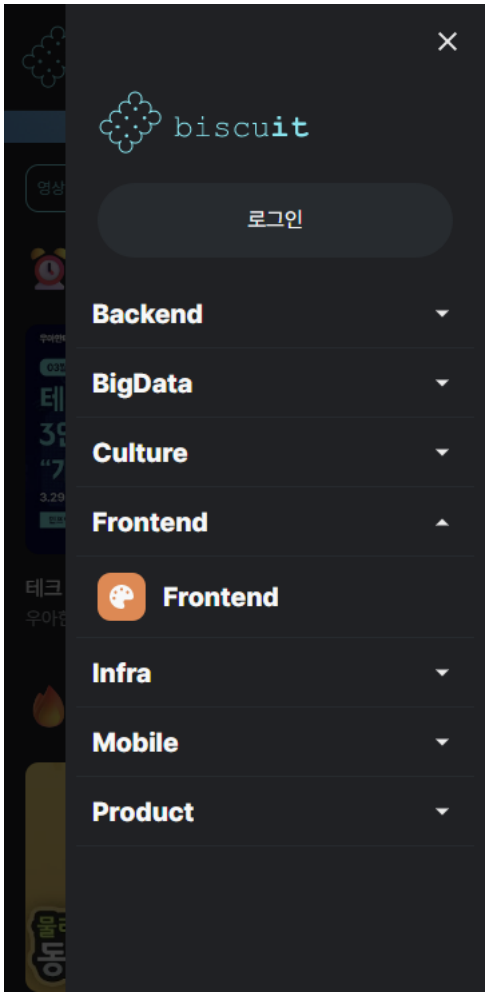


5. 시간 선택 필터

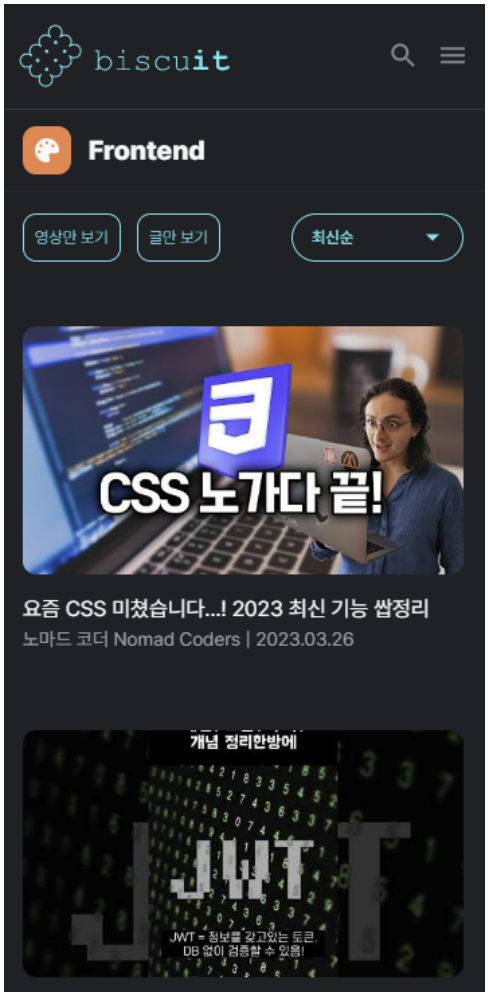


6. 해당 범위 시간의 컨텐츠만 제공

3.2. 카테고리

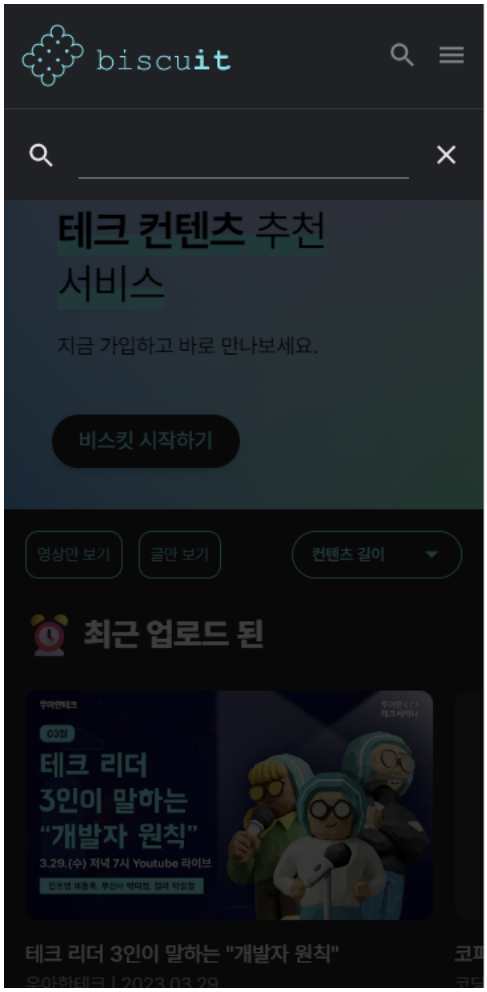


1. 측면 메뉴를 통해 카테고리 조회

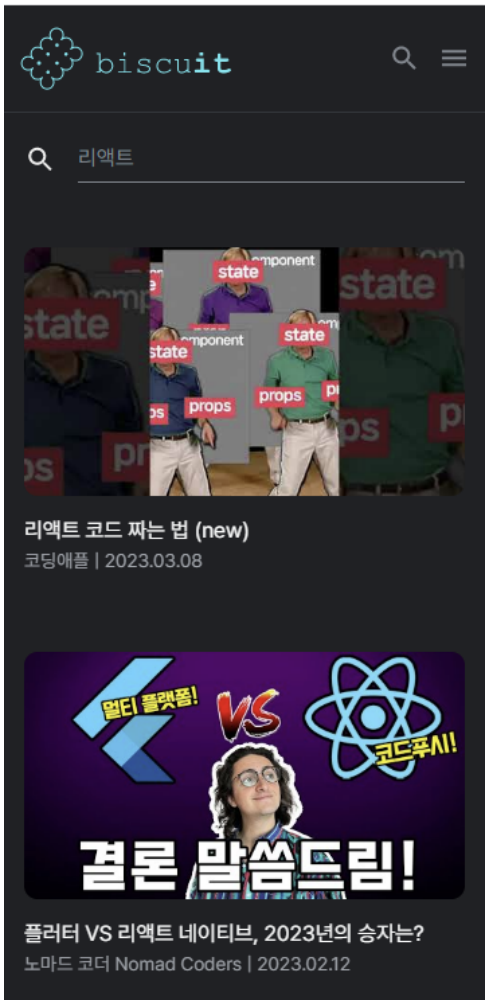


2. 카테고리별 컨텐츠 조회

3.3. 검색

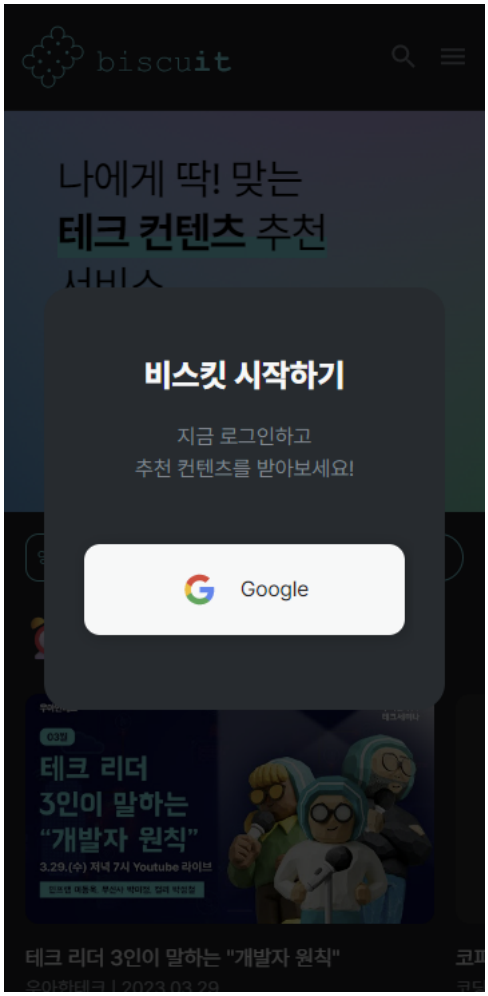


1. 글로벌 네브바를 통해 검색 가능



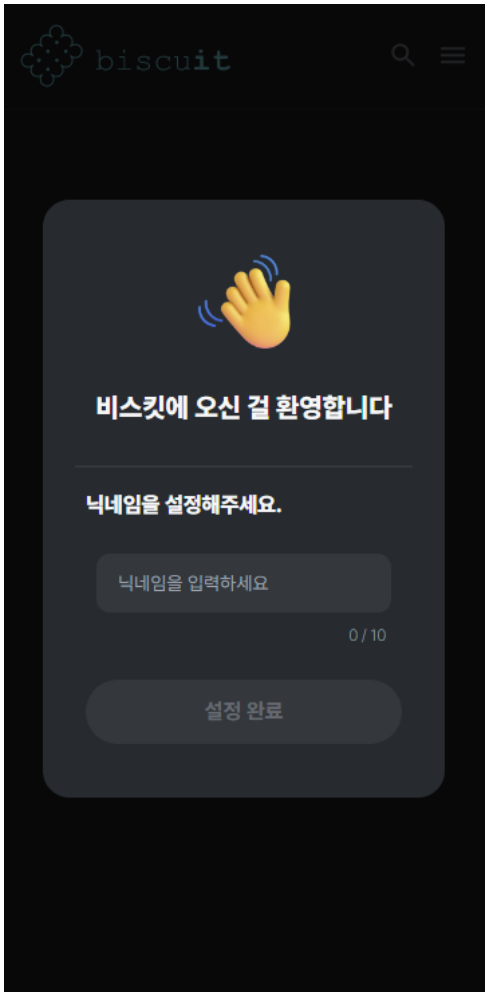
2. 검색 결과 컨텐츠 조회

3.4. 소셜 로그인 : 구글 OAuth

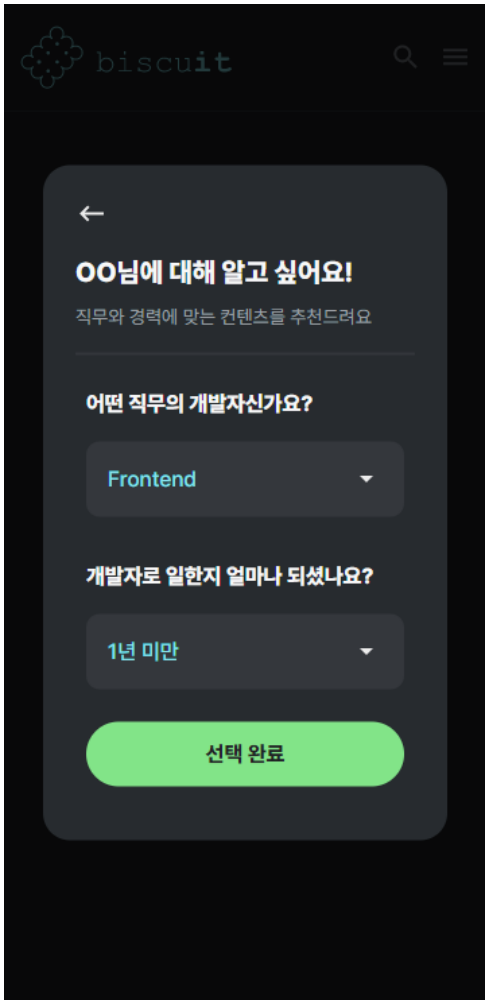


1. 구글 OAuth를 활용한 소셜 로그인

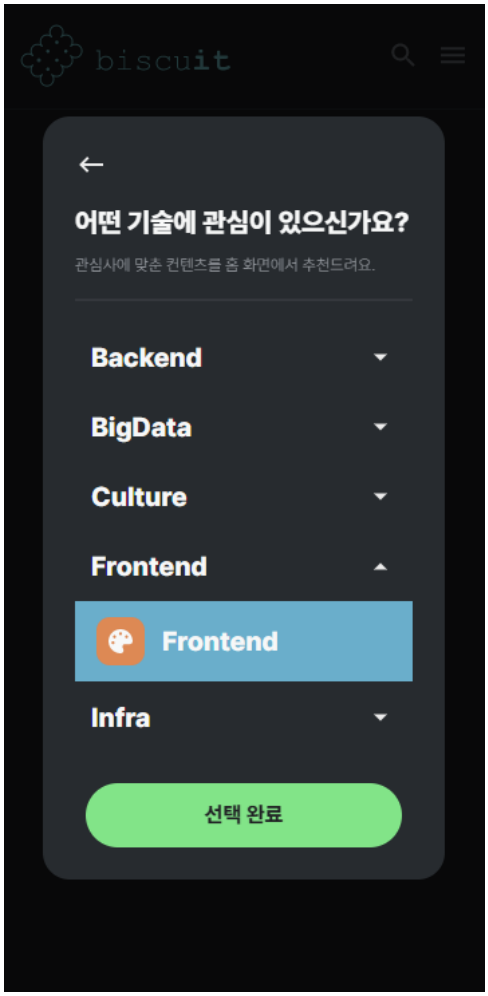
3.5. 온보딩 : 최초 회원 가입시 시행



1. 닉네임 설정

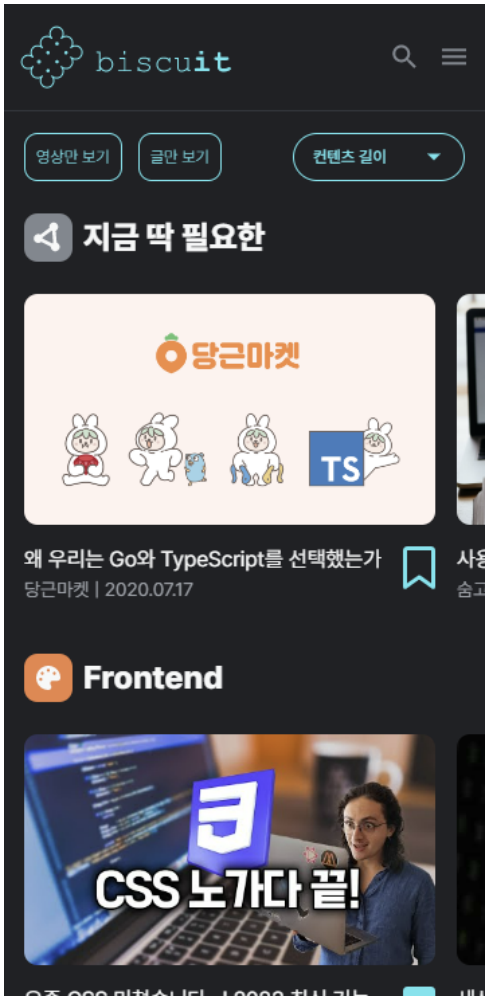


2. 직무 및 경력 설정

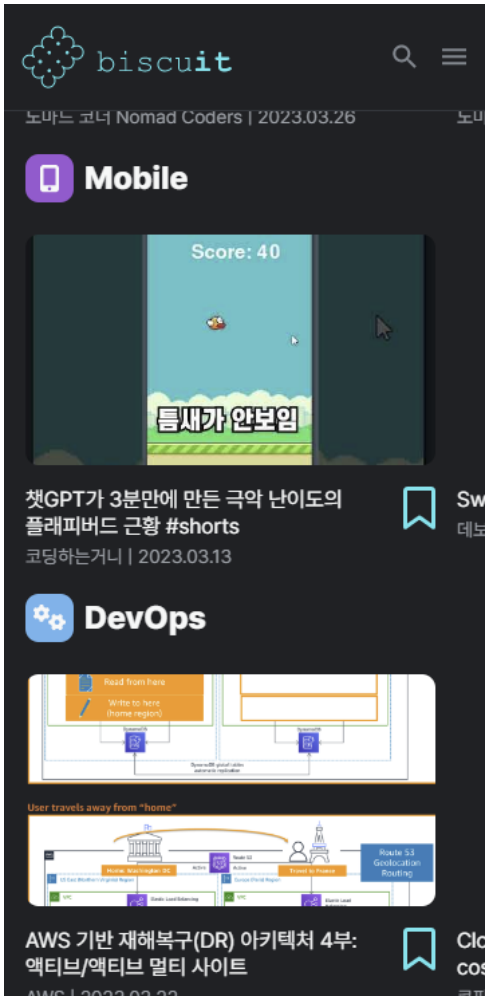


3. 다양한 관심 분야 설정

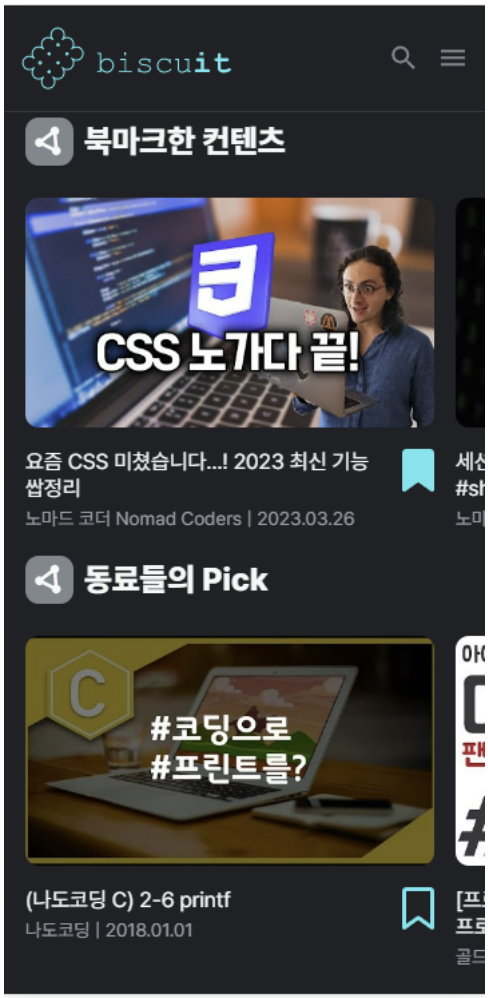
3.6. 회원 홈



1. 지금 딱 필요한 : 사용자의 관심, 직무, 히스토리, 난이도 등을 고려한 컨텐츠 추천

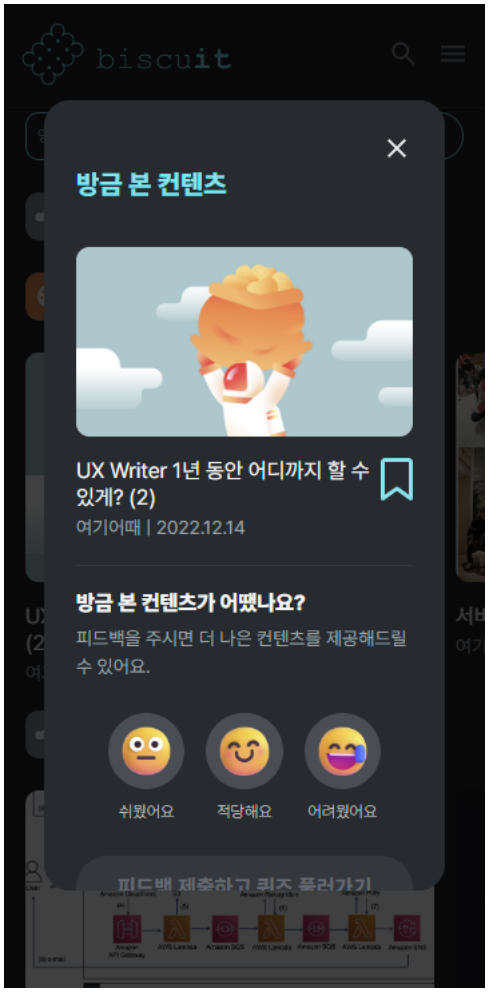


2. 관심 있는 분야에 속한 컨텐츠 추천



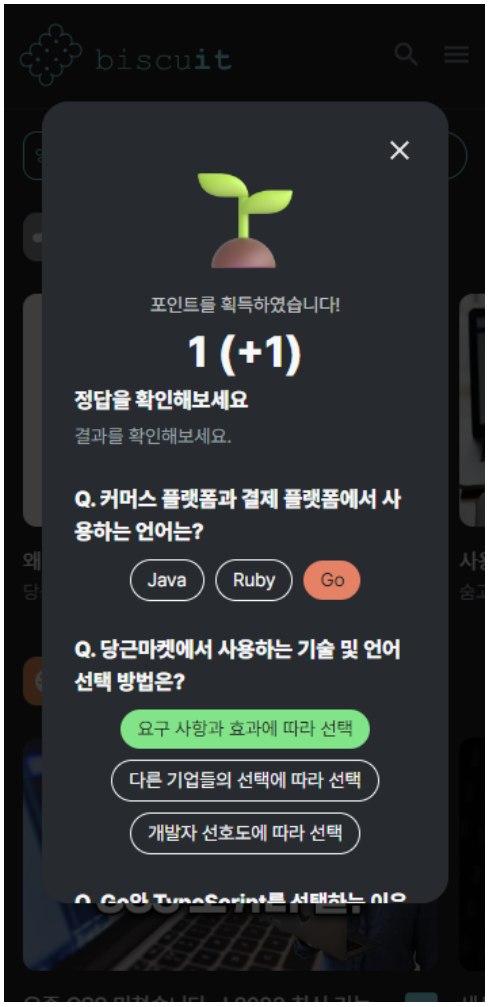
3. 북마크 : 북마크 해놓은 컨텐츠 출력
동료들의 Pick : 유사한 직무 및 경력의 사용자들이 많이 본 컨텐츠 추천

3.7. 피드백 및 퀴즈



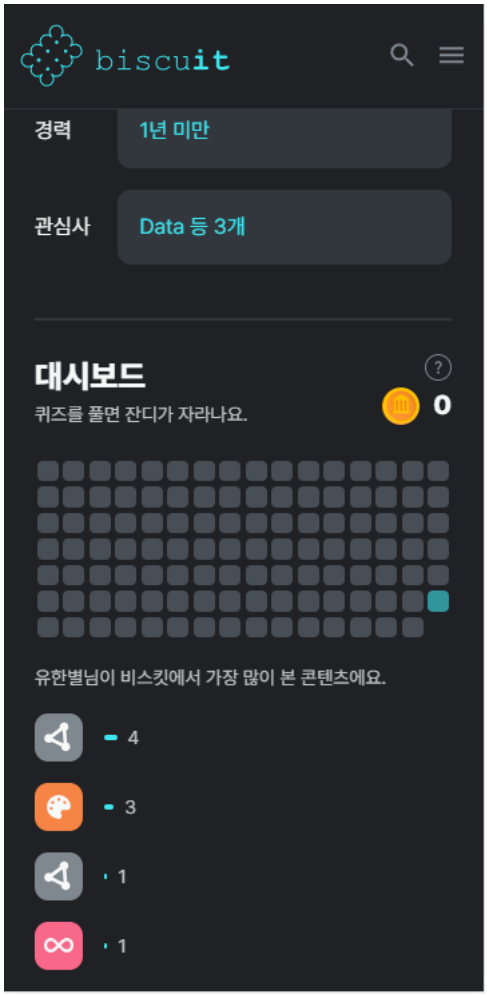
1. 해당 글을 읽는데 걸린 시간 + 난이도 피드백 수집

2. 해당 글의 내용을 바탕으로 복습 퀴즈 생성



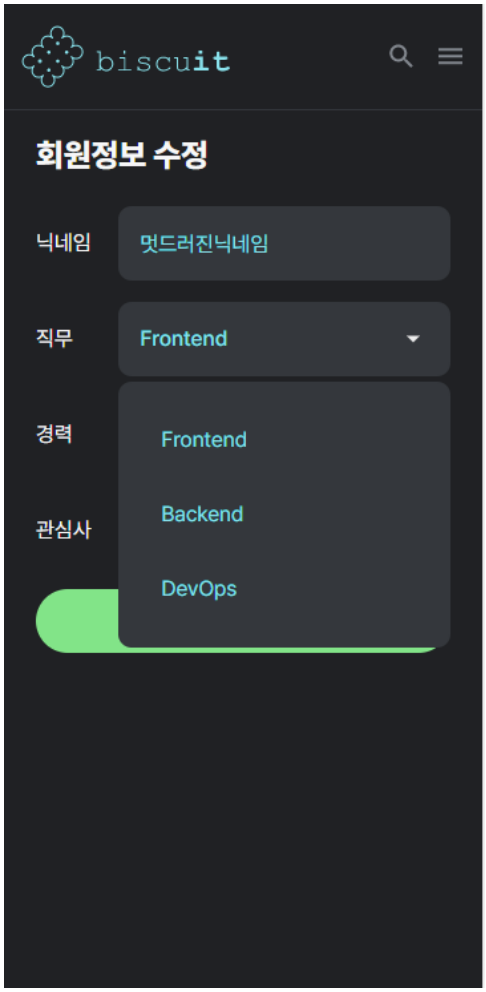
3. 실시간 채점 및 맞춘 문제만큼 포인트 획

3.8. 마이페이지 : 대시보드

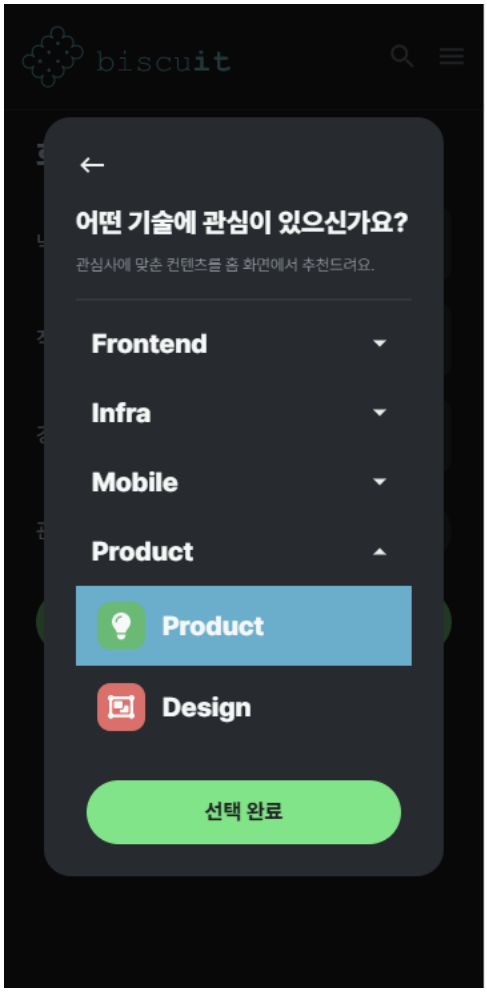


1. 잔디, 포인트, 그래프 기능

3.9. 계정 설정

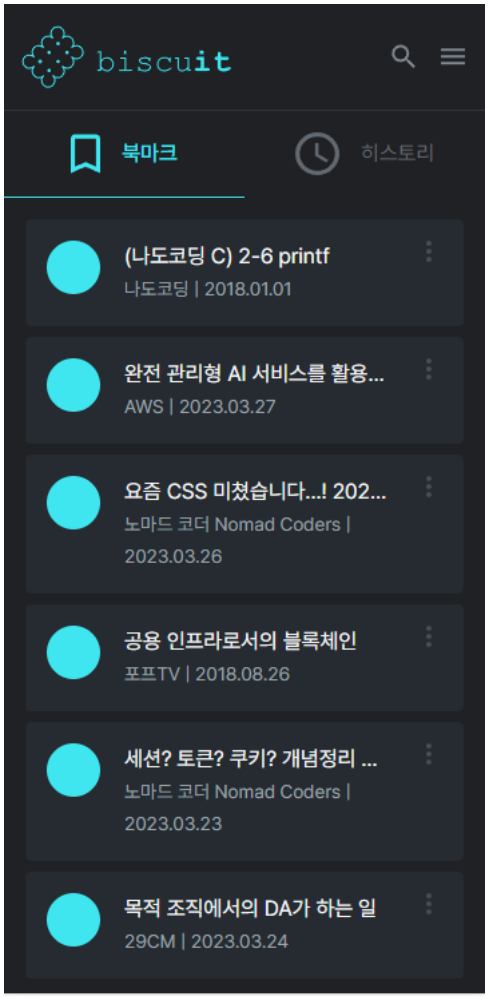


1. 닉네임, 직무, 경력 변경 기능



2. 관심사 변경으로 컨텐츠 추천 새롭게 갱신

3.10. 북마크/히스토리



1. 북마크한 컨텐츠 모아보기

2. 지금까지 시청한 컨텐츠 자동 기록