포팅 메뉴얼

1. 개발 환경

1-1. 프로젝트 기술 스택

- Frontend
  - 프로젝트 환경

    ```
    VisualStudioCode: 1.92.1
    Android Studio: 2.024.1.1
    ```

  - 의존성 및 버전

    ```
    autoprefixer: 10.4.19
    axios: 1.7.2
    js-cookie: 3.0.5
    postcss: 8.4.39
    react: 18.3.1
    react-dom: 18.3.1
    react-draggable: 4.4.6
    react-icons: 5.2.1
    react-modal: 3.16.1
    react-router-dom: 6.25.1
    sockjs: 0.3.24
    sockjs-client: 1.6.1
    stompjs: 2.3.3
    swiper: 11.1.6
    tailwindcss: 3.4.6
    websocket: 1.0.35
    zustand: 4.5.4
    @types/react: 18.3.3
    @types/react-dom: 18.3.0
    @vitejs/plugin-react: 4.3.1
    eslint: 8.57.0
    eslint-config-prettier: 9.1.0
    eslint-plugin-prettier: 5.2.1
    eslint-plugin-react: 7.35.0
    eslint-plugin-react-hooks: 4.6.2
    eslint-plugin-react-refresh: 0.4.7
    prettier: 3.3.3
    vite: 5.3.4
    ```

- Backend
  - 프로젝트 환경

    ```
    IntelliJ IDEA: 2024.1.4 (Ultimate Edition)
    JVM: OpenJDK 17 (17-jdk-slim)
    Spring Boot: 3.3.1
    Spring Dependency Management: 1.1.5
    Redis: 7.4.0
    MySql: 8.4.2
    ```

  - 의존성 및 버전

    ```
    Spring Boot Starter Data JPA: 3.3.1
    Spring Boot Starter Data Redis: 3.3.1
    Spring Boot Starter Web: 3.3.1
    MySQL Connector/J: 8.3.0
    Spring Data Redis: 3.3.1
    Lombok: 1.18.32
    Spring Boot DevTools: 3.3.1
    Spring Boot Starter WebSocket: 3.3.1
    Springdoc OpenAPI Starter WebMVC UI: 2.5.0
    Apache Commons Codec: 1.15
    P6Spy Spring Boot Starter: 1.5.6
    Spring Boot Starter Validation: 3.3.1
    Javax Annotation API: 1.3.2
    QueryDSL JPA: 5.0.0 (Jakarta 버전)
    Spring Boot Starter Mail: 3.3.1
    Spring Boot Starter OAuth2 Client: 3.3.1
    Spring Boot Starter Security: 3.3.1
    JJWT API: 0.11.2
    JJWT Impl: 0.11.2
    JJWT Jackson: 0.11.2
    Firebase Admin SDK: 7.1.1
    ```

- CI/CD

  ```
  AWS EC2

  - Nginx: 1.18.0
  - Ubuntu: 20.04.6
  - Docker: 27.1.1
  - Jenkins: 2.452.3
  ```

1-2. 환경변수 설정

- Frontend
  - .env

    ```
    VITE_SERVER_URL=https://i11e205.p.ssafy.io
    ```

  - Dockerfile

    ```
    FROM node:16 AS build

    # 애플리케이션 디렉토리 생성
    WORKDIR /app

    # 애플리케이션 종속성 설치
    COPY package*.json ./
    RUN npm install

    # 애플리케이션 소스 복사
    COPY . .
    ```

```
# 애플리케이션 빌드
RUN npm run build

# Nginx 설정
FROM nginx:alpine
COPY --from=build /app/dist /usr/share/nginx/html
COPY nginx.conf /etc/nginx/nginx.conf

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

- docker-compose-front.yml

```
services:
  web:
    build:
      context: ./took_web
    container_name: took_web
    expose:
      - 5173
    restart: always
    networks:
      took_network:
        ipv4_address: 172.19.0.2

networks:
  took_network:
    external: true
```

- nginx.conf

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    server {
        listen 80;
        server_name localhost; # 필요에 따라 도메인 이름 수정

        root /usr/share/nginx/html; # 도커파일에서 복사한 위치와 일치하도록 수정
        index index.html;

        location / {
            try_files $uri $uri/ /index.html;
        }
    }
}
```

- google-service.json

```
{
  "project_info": {
    "project_number": "433232049429",
    "project_id": "took-a85bc",
    "storage_bucket": "took-a85bc.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:433232049429:android:3a5c56d8b147d30a68651c",
        "android_client_info": {
          "package_name": "com.example.myfcmapp"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": "AIzaSyCpfPT9A_R1TFpC2eQ87R8YRGchgPA8bjQ"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    },
    {
      "client_info": {
        "mobilesdk_app_id": "1:433232049429:android:b139762e395ee86d68651c",
        "android_client_info": {
          "package_name": "com.example.took_app"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": "AIzaSyCpfPT9A_R1TFpC2eQ87R8YRGchgPA8bjQ"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

- Backend
  - application.properties

```
# Redis
spring.data.redis.host=i11e205.p.ssafy.io
spring.data.redis.port=6380


# MySQL
```

```
spring.datasource.url=jdbc:mysql://db:3306/took
spring.datasource.username=turtle
spring.datasource.password=took5678%^&*
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#spring.datasource.url=jdbc:mysql://localhost:3306/test
#spring.datasource.username=root
#spring.datasource.password=ssafy
#spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver


# JPA
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.open-in-view=true

# Swagger
springdoc.version=@project.version@
springdoc.api-docs.path=/api-docs
springdoc.default-consumes-media-type=application/json
springdoc.default-produces-media-type=application/json
springdoc.swagger-ui.operations-sorter=alpha
springdoc.swagger-ui.tags-sorter=alpha
springdoc.swagger-ui.path=/api/swagger-ui/index.html
springdoc.swagger-ui.disable-swagger-default-url=true
springdoc.swagger-ui.display-query-params-without-oauth2=true
springdoc.swagger-ui.doc-expansion=none


# API KEY
kakao.api.key=f10f59b2a284e660b2b416dc63f9bf6f
sms.api.key=NCSA8ZZQOIFAMBS1
sms.api.secret=F5CRLDG3CBFY0ZNDT7JXTRS4ONGGJP2C


#Gmail
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=devjaechan@gmail.com
spring.mail.password=lioz rges mfgm rdhr
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

#Kakao OAuth2
spring.security.oauth2.client.registration.kakao.client-id=290fedb94c4d6ee266dfe2e4e56f563a
spring.security.oauth2.client.registration.kakao.client-secret=7Czf997AslCjhspjg2SjNEDBs19Ufhox
spring.security.oauth2.client.registration.kakao.redirect-uri=https://i11e205.p.ssafy.io/api/oauth2/callback/{registrationId}
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.client-authentication-method=client_secret_post
spring.security.oauth2.client.registration.kakao.scope=profile_nickname

# Google Oauth2
spring.security.oauth2.client.registration.google.client-id=951724536921-v4kbthfc7tcjkfql4dr909ch3j51ubvl.apps.googleusercontent.com
spring.security.oauth2.client.registration.google.client-secret=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


#KaKao Oauth2 provider
spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attribute=id

# Jwt
secret-key=ThisStatementIsJwtSecretKeyDoNotUseThisState

# http utf-8
server.servlet.encoding.charset=UTF-8
server.servlet.encoding.enabled=true
server.servlet.encoding.force=true

# application.properties
distance.threshold=10000.0

# timezone
spring.jackson.time-zone=Asia/Seoul

# CORS ( swagger-ui )
server.forward-headers-strategy=FRAMEWORK
```

- Dockerfile

```
# Use a base image with JDK
FROM openjdk:17-jdk-slim

# Install tzdata package and configure timezone
RUN apt-get update && apt-get install -y tzdata && \
    ln -fs /usr/share/zoneinfo/Asia/Seoul /etc/localtime && \
    dpkg-reconfigure --frontend noninteractive tzdata

# Add a directory for the application
WORKDIR /app

# Copy the JAR file into the container
COPY build/libs/*.jar app.jar

# Run the JAR file
ENTRYPOINT ["java", "-jar", "app.jar"]
```

- docker-compose.yml

```
services:
  db:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: took5678%^&*
      MYSQL_DATABASE: took
      MYSQL_USER: turtle
      MYSQL_PASSWORD: took5678%^&*
    ports:
      - "3307:3306"
    networks:
      took_network:
        ipv4_address: 172.19.0.3

  redis:
```

```
    image: redis:latest
    ports:
      - "6380:6379"
    networks:
      took_network:
        ipv4_address: 172.19.0.4

  springboot:
    build:
      context: ./backend
    container_name: took_springboot
    expose:
      - 8080
    depends_on:
      - db
      - redis
    networks:
      took_network:
        ipv4_address: 172.19.0.5

networks:
  took_network:
    external: true
```

- serviceAccountKey.json

```
{
  "type": "service_account",
  "project_id": "took-a85bc",
  "private_key_id": "39e4e83e996390830c9678874529d2ec386ca498",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCM8Ji2EmsNjAjx\nSbaVoMq4f+7COFoiOZmWioZLonDBjRcrS9dEEV9fKYyUcAzWJxBs+sdBTdocV07m\nBqf2SPEF
  "client_email": "firebase-adminsdk-7typg@took-a85bc.iam.gserviceaccount.com",
  "client_id": "115351089823964331132",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-7typg%40took-a85bc.iam.gserviceaccount.com",
  "universe_domain": "googleapis.com"
}
```

1-3. CI/CD 설정 파일

- Nginx
  - 설정 파일 위치

```
/etc/nginx/sites-available/proxy-setting
```

  - proxy-setting

```
server {
    listen 80;
    listen [::]:80;
    server_name i11e205.p.ssafy.io;
    return 301 https://$host$request_uri;
}

server {
         listen 443 ssl;
         listen [::]:443 ssl;
         server_name i11e205.p.ssafy.io;


         ssl_certificate /etc/letsencrypt/live/i11e205.p.ssafy.io/fullchain.pem; # managed by Certbot
         ssl_certificate_key /etc/letsencrypt/live/i11e205.p.ssafy.io/privkey.pem; # managed by Certbot


        location /api {
            # Postman의 User-Agent를 차단하는 설정
            if ($http_user_agent ~* "PostmanRuntime") {
               return 403;
            }

            # Referer 헤더를 확인하여 프론트엔드에서의 요청만 허용
            if ($http_referer !~* "^https://i11e205.p.ssafy.io") {
               return 403;
            }

            proxy_pass http://172.19.0.5:8080/api;

            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Server $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /ws {
            proxy_pass http://172.19.0.5:8080/ws;

            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";

            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Server $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }


        location / {
            proxy_pass http://172.19.0.2;

            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Server $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

}
```

- Jenkins
  - Frontend 파이프라인

```
pipeline {
    agent any

    stages {
        stage('Clone Repository') {
            steps {
                git branch: 'dev-fe', credentialsId: 'zxader', url: 'https://lab.ssafy.com/s11-webmobile1-sub2/S11P12E205'
            }
        }
        stage('Prepare Docker Compose File') {
            steps {
                withCredentials([file(credentialsId: 'docker-compose-front', variable: 'yml')]) {
                    script {
                        sh 'pwd'
                        sh 'chmod +r $yml'
                        sh 'chmod -R 777 ./'
                        sh 'cp $yml docker-compose-front.yml'
                    }
                }
            }
        }
        stage('Prepare env') {
            steps {
                withCredentials([file(credentialsId: 'env', variable: 'env')]) {
                    script {
                        // 현재 작업 디렉토리 출력
                        sh 'pwd'

                        // .env 파일에 읽기 권한 부여
                        sh 'chmod +r $env'

                        // 모든 파일에 대해 777 권한 부여 (주의: 이 권한 설정은 보안에 취약할 수 있음)
                        sh 'chmod -R 777 ./'

                        // .env 파일을 front_web 디렉토리로 복사
                        sh 'cp $env took_web/.env'
                    }
                }
            }
        }
        stage('Build and Run Docker Compose') {
            steps {
                sh 'docker-compose -f docker-compose-front.yml up -d --build'
            }
        }
    }
    post {
        failure {
            echo 'Build or deployment failed.'
        }
        cleanup {
            script {
                // 모든 경우에 사용하지 않는 이미지 삭제
                sh 'docker image prune -af'
            }
        }
    }
}
```

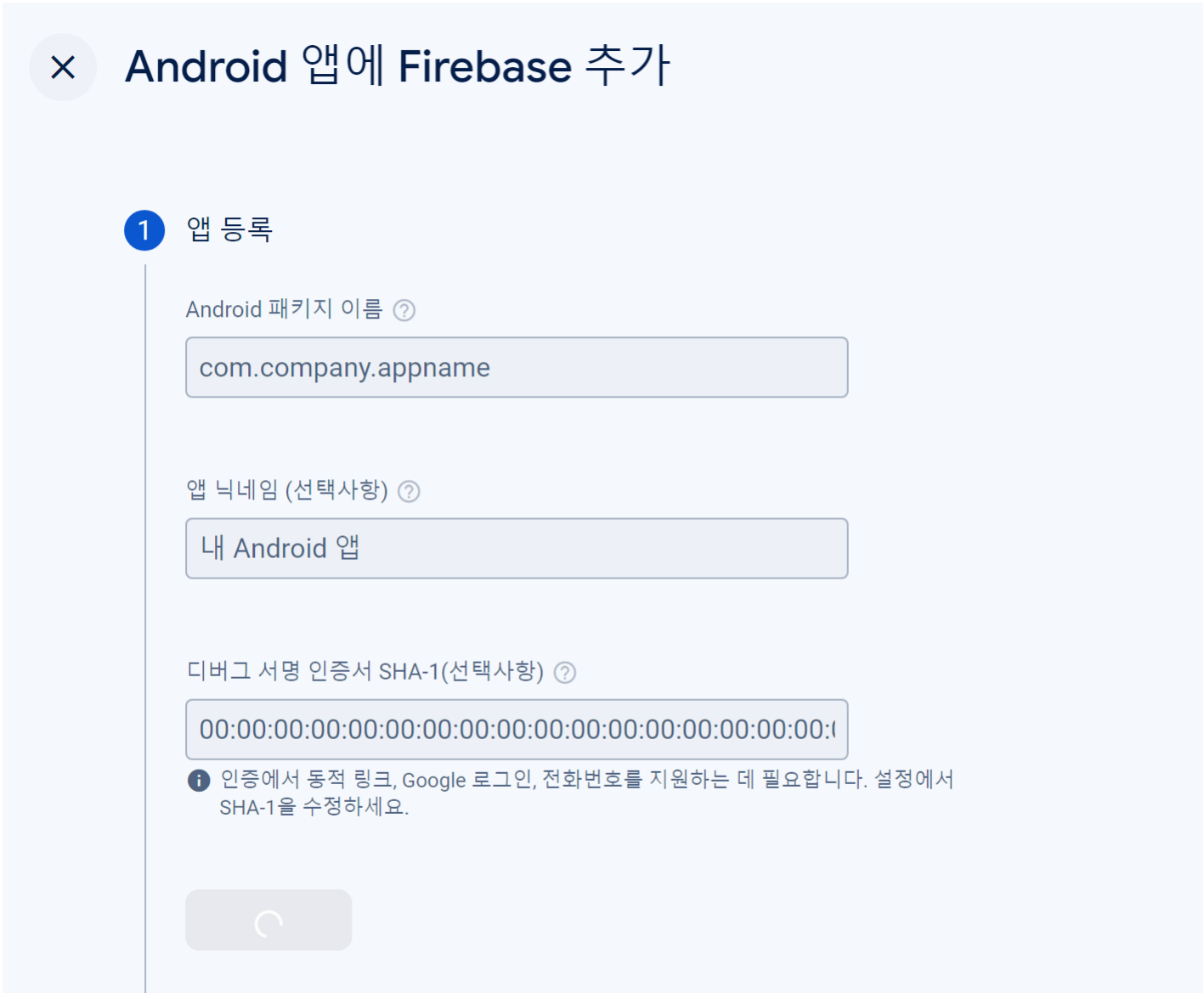
  - Backend 파이프라인

```
pipeline {
    agent any

    stages {

        stage('Clone Repository') {
            steps {
                git branch: 'dev-be', credentialsId: 'zxader', url: 'https://lab.ssafy.com/s11-webmobile1-sub2/S11P12E205'
            }
        }
        stage('application.properties'){
              steps{
              withCredentials([file(credentialsId: 'application', variable: 'properties')]) {
              script {
                    sh 'pwd'
                    sh 'chmod +r $properties'
                    sh 'chmod -R 777 ./backend/src/main/resources'
                    sh 'cp $properties backend/src/main/resources/application.properties'
            }
            }
        }
    }
    stage('serviceAccountKey'){
          steps{
          withCredentials([file(credentialsId: 'serviceAccountKey', variable: 'json')]) {
          script {
                sh 'pwd'
                sh 'chmod +r $json'
                sh 'chmod -R 777 ./backend/src/main/resources'
                sh 'cp $json backend/src/main/resources/serviceAccountKey.json'
            }
            }
        }
    }
    stage('docker-compose'){
          steps{
          withCredentials([file(credentialsId: 'docker-compose', variable: 'yml')]) {
          script {
                sh 'pwd'
                sh 'chmod +r $yml'
                sh 'chmod -R 777 ./'
                sh 'cp $yml docker-compose.yml'
            }
            }
        }
    }
    stage('Build JAR') {
        steps {
            script {
                // JAR 파일 빌드를 위한 디렉토리로 이동
                dir('backend') {
```

```
                        // gradlew에 실행 권한 부여
                        sh 'chmod +x gradlew'
                        // JAR 파일 빌드
                        sh './gradlew clean build'
                    }
                }
            }
        }
        stage('Build and Deploy Docker Image') {
            steps {
                script {
                    // Docker Compose를 사용하여 Docker 이미지 빌드 및 실행
                    sh 'docker-compose -f docker-compose.yml up -d --build'
                }
            }
        }
    }
    post {
        success {
            echo 'Build and deployment successful.'
        }
        failure {
            echo 'Build or deployment failed.'
        }
        cleanup {
            script {
                // 모든 경우에 사용하지 않는 이미지 삭제
                sh 'docker image prune -af'
            }
        }
    }
}
```

2. 외부 서비스

- FireBase Cloude Messaging



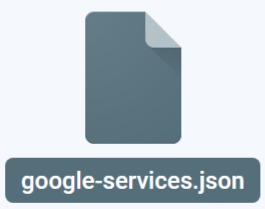
2 구성 파일 다운로드 후 추가

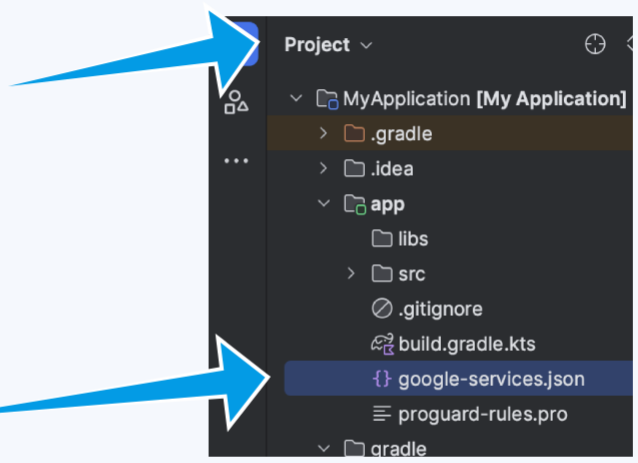
Android 스튜디오에 대한 안내(아래 참조) | Unity C++

google-services.json 다운로드

Android 스튜디오에서 프로젝트 뷰로 전환하여 프로젝트 루트 디렉터리를 확인합니다.

다운로드한 `google-services.json` 파일을 모듈(앱 수준) 루트 디렉터리로 이동합니다.





다음

3 Firebase SDK 추가

Gradle 안내 | Unity C++

★ 여전히 `buildscript` 구문을 사용하여 플러그인을 관리하고 있나요? 해당 구문을 사용하여 Firebase 플러그인을 추가하는 방법을 알아보세요.

1. Firebase SDK가 `google-services.json` 구성 값에 액세스할 수 있도록 하려면 Google 서비스 Gradle 플러그인이 필요합니다.

   ◉ Kotlin DSL(build.gradle.kts) ○ Groovy(build.gradle)

   프로젝트 수준의 `build.gradle.kts` 파일에 플러그인을 종속 항목으로 추가합니다.

   루트 수준(프로젝트 수준) **Gradle** 파일(`<project>/build.gradle.kts`):

```
plugins {
  // ...

  // Add the dependency for the Google services Gradle plugin
  id("com.google.gms.google-services") version "4.4.2" apply false
}
```

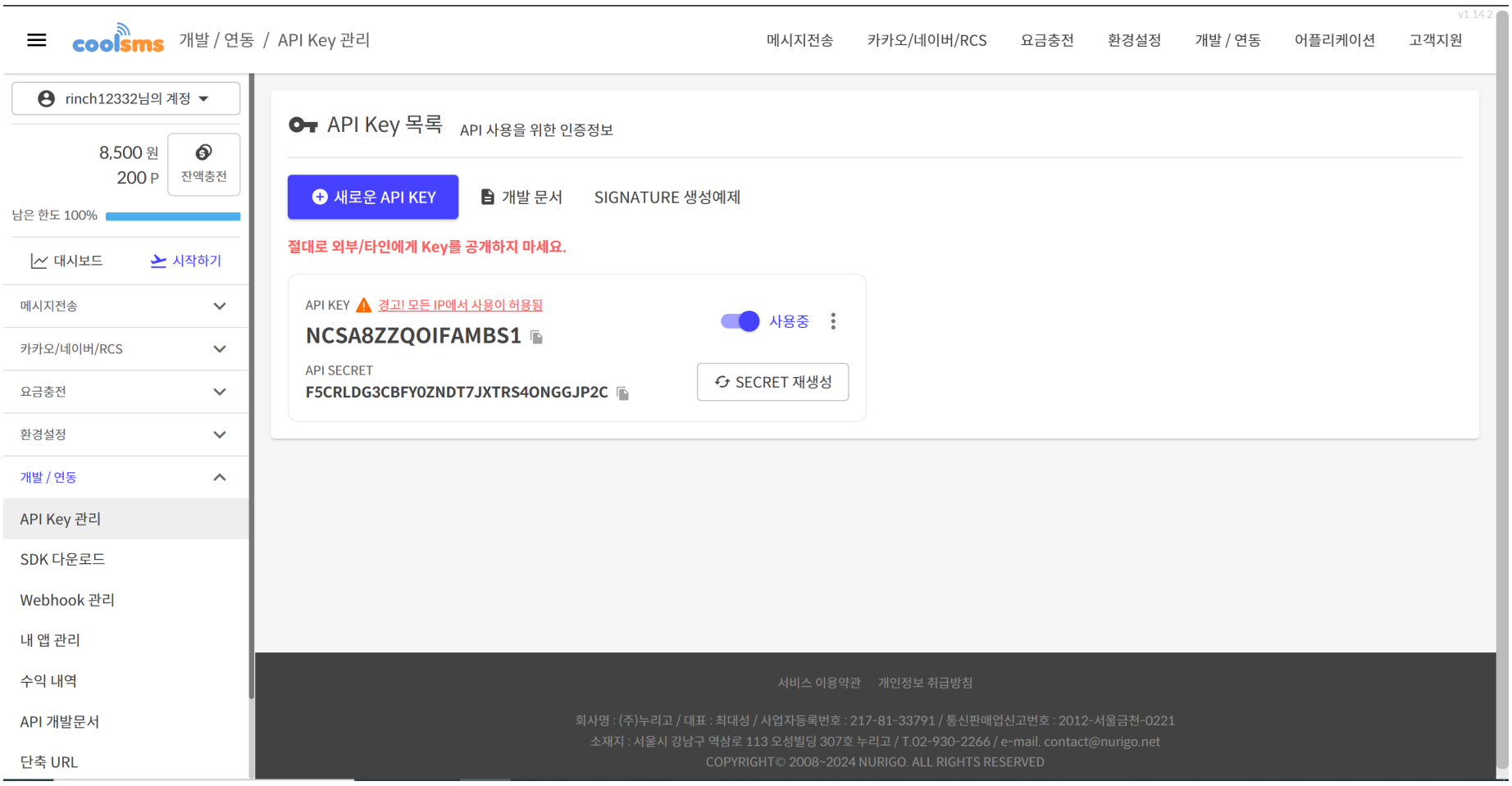
2. 그런 다음 모듈(앱 수준) `build.gradle.kts` 파일에서 `google-services` 플러그인과 앱에서 사용할 Firebase SDK를 모두 추가합니다.

   모듈(앱 수준) **Gradle** 파일(`<project>/<app-module>/build.gradle.kts`):

```
plugins {
  id("com.android.application")
  // Add the Google services Gradle plugin
  id("com.google.gms.google-services")
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation(platform("com.google.firebase:firebase-bom:33.1.2"))

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation("com.google.firebase:firebase-analytics")

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

- 문자 전송 서비스 (쿨에스엠에스)

```
https://console.coolsms.co.kr/ 회원가입 후 API 키 발급 및 SECRET 생성
```



- 카카오 모빌리티

```
https://developers.kakao.com/ 에서  APP 등록 및 API Key 발급

https://developers.kakaomobility.com/product/api 문서 참고해서 API 호출
```