A307 포팅 매뉴얼

1. 개발환경

- **운영 체제:** Ubuntu 22.04 LTS (AWS EC2)
- **프레임워크:** Spring Boot 3.4.1, Next.js, React
- **데이터베이스:** MySQL 8.0.41 (AWS RDS), MongoDB 6.0.9
- **캐시:** Redis 7.2.3
- **파일(이미지) 저장소:** AWS S3
- **CI/CD:** Jenkins + Docker
- **프로그래밍 언어:** Java 17, JavaScript, TypeScript, HTML5, CSS3
- **버전 관리:** Git(GitLab), Jira
- **웹 서버:** Nginx 1.24.0
- **검색 엔진:** Elasticsearch 8.5.3
- **미디어 서버:** Mediasoup 1.0.0

2. 라이브러리 버전

BackEnd

- Spring Boot: 3.4.1
- Spring Security: 6.4.2
- JPA (Hibernate): 3.4.2
- Redis: 7.4.2
- Lombok: 1.18.36
- WebSocket : 10.1.34

FrontEnd

- shadCN: 1.0.0
- tailwind CSS: 3.4.17
- next.js: 13.5.8
- node.js: 20.18.1
- TypeScript: 5.7.3
- Lucide: 0.475.0

3. 프로젝트 설정 파일 (application.yml)

BackEnd `application.yml` **파일**

```
spring:
  application:
    name: respring

servlet:
  multipart:
    enabled: true
    file-size-threshold: 2KB
    max-file-size: 10MB
    max-request-size: 10MB

config:
  activate:
    on-profile: default

datasource:
  url: ${SPRING_DATASOURCE_URL}
  username: ${SPRING_DATASOURCE_USERNAME}
  password: ${SPRING_DATASOURCE_PASSWORD}
  driver-class-name: com.mysql.cj.jdbc.Driver

data:
  mongodb:
    database: ${MONGODB_DATABASE}
    uri: ${MONGODB_URI}

  redis:
    host: ${REDIS_HOST}
    port: ${REDIS_PORT}
    password: ${REDIS_PASSWORD}


ai:
  openai:
    api-key: ${OPENAI_API_KEY}
    chat:
      options:
        model: gpt-4o-mini # gpt-4o-mini
        temperature: 0.7

jpa:
  hibernate:
```

```
    ddl-auto: update
  show-sql: false
  properties:
    hibernate.dialect: org.hibernate.dialect.MySQLDialect
    hibernate.jdbc.time_zone: Asia/Seoul

elasticsearch:
  uris: ${ELASTICSEARCH_URI}
  username: ${ELASTICSEARCH_USERNAME}
  password: ${ELASTICSEARCH_PASSWORD}


security:
  oauth2:
    client:
      registration:
        google:
          client-id: ${GOOGLE_CLIENT_ID}
          client-secret: ${GOOGLE_CLIENT_SECRET}
          redirect-uri: ${GOOGLE_REDIRECT_URI}
          scope:
            - profile
            - email

        kakao:
          client-id: ${KAKAO_CLIENT_ID}
          client-secret: ${KAKAO_CLIENT_SECRET}

          client-authentication-method: client_secret_post
          authorization-grant-type: authorization_code
          scope:
            - profile_nickname
            - profile_image
            - account_email
          redirect-uri: ${KAKAO_REDIRECT_URI}
          client-name: Kakao

      provider:
        kakao:
          authorization-uri: https://kauth.kakao.com/oauth/authorize
          token-uri: https://kauth.kakao.com/oauth/token
          user-info-uri: https://kapi.kakao.com/v2/user/me
          user-info-authentication-method: header
          user-name-attribute: id

  jackson:
    time-zone: Asia/Seoul

logging:
  level:
    org:
      hibernate:
        orm:
          connections:
            pooling: ERROR

file:
  upload-dir: ${FILE_UPLOAD_DIR:${user.home}/uploads}
```

```
management:
  endpoints:
    web:
      exposure:
        include: "health,info"
  endpoint:
    health:
      show-details: always


cloud:
  aws:
    s3:
      bucket: respring-s3-bucket
    stack.auto: false
    region:
      static: ap-northeast-2
    credentials:
      access-key: ${S3_ACCESSKEY}
      secret-key: ${S3_SECRETKEY}

---
# 🔹 로컬 환경에서 `.env` 파일을 읽도록 설정 (필요한 경우만)
spring:
  config:
    import: optional:file:.env[.properties]
    activate:
      on-profile: local
```

4. 사용된 외부 API

- OpenAI의 chatGPT API
  - `gpt-4o-mini`
- web speech api

5. 빌드 및 배포 방법

docker-compose를 이용한 빌드

1. **docker 설치**

```
sudo apt update
sudo apt install -y docker.io
```

2. **docker-compose 설치 및 권한 부여**

```
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

`docker-compose.yml` **파일**

```
services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    environment:
      - SPRING_DATASOURCE_URL=${SPRING_DATASOURCE_URL}
      - SPRING_DATASOURCE_USERNAME=${SPRING_DATASOURCE_USERNAME}
      - SPRING_DATASOURCE_PASSWORD=${SPRING_DATASOURCE_PASSWORD}
      - MONGODB_URI=${MONGODB_URI}
      - MONGODB_DATABASE=${MONGODB_DATABASE}
      - ELASTICSEARCH_URI=${ELASTICSEARCH_URI}
      - ELASTICSEARCH_USERNAME=${ELASTICSEARCH_USERNAME}
      - ELASTICSEARCH_PASSWORD=${ELASTICSEARCH_PASSWORD}
      - REDIS_HOST=${REDIS_HOST}
      - REDIS_PORT=${REDIS_PORT}
      - REDIS_PASSWORD=${REDIS_PASSWORD}
      - S3_ACCESSKEY=${S3_ACCESSKEY}
      - S3_SECRETKEY=${S3_SECRETKEY}
      - S3_BUCKET=${S3_BUCKET}
      - S3_REGION=${S3_REGION}
      - GOOGLE_CLIENT_ID=${GOOGLE_CLIENT_ID}
      - GOOGLE_CLIENT_SECRET=${GOOGLE_CLIENT_SECRET}
      - GOOGLE_REDIRECT_URI=${GOOGLE_REDIRECT_URI}
      - KAKAO_CLIENT_ID=${KAKAO_CLIENT_ID}
      - KAKAO_CLIENT_SECRET=${KAKAO_CLIENT_SECRET}
      - KAKAO_REDIRECT_URI=${KAKAO_REDIRECT_URI}
      - OPENAI_API_KEY=${OPENAI_API_KEY}
      - MEDIASOUP_SERVER=${MEDIASOUP_SERVER}
      - MEDIASOUP_ANNOUNCED_IP=${MEDIASOUP_ANNOUNCED_IP}
    volumes:
      - ~/.gradle:/home/gradle/.gradle
    depends_on:
      db:
        condition: service_healthy
      mongo-db:
        condition: service_healthy
      elasticsearch:
        condition: service_healthy
      redis:
        condition: service_healthy

    healthcheck:
      test: [ "CMD", "wget", "--spider", "-q", "http://localhost:8080/actuator/health" ]
      interval: 10s
      timeout: 5s
      retries: 3
    networks:
      - project-network

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile
      args:
        NEXT_PUBLIC_API_BASE_URL: "https://i12a307.p.ssafy.io/api"
    networks:
```

```
    - project-network
  depends_on:
    - backend


db:
  image: mysql:8.0
  ports:
    - "3306:3306"
  environment:
    MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
    MYSQL_DATABASE: ${MYSQL_DATABASE}
    MYSQL_USER: ${MYSQL_USER}
    MYSQL_PASSWORD: ${MYSQL_PASSWORD}
    MYSQL_ROOT_HOST: "%"
    TZ: Asia/Seoul
  command:
    [
      "sh", "-c",
      "mkdir -p /etc/mysql/conf.d && \
      echo '[mysqld]' > /etc/mysql/conf.d/timezone.cnf && \
      echo 'default-time-zone = \"+09:00\"' >> /etc/mysql/conf.d/timezone.cnf && \
      exec mysqld --defaults-extra-file=/etc/mysql/conf.d/timezone.cnf"
    ]

  healthcheck:
    test: [ "CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "${MYSQL_USER}", "-p${MYSQL_PASSWORD}" ]
    interval: 10s
    timeout: 5s
    retries: 3
  volumes:
    - db_data:/var/lib/mysql
  networks:
    project-network:
      aliases:
        - database


jenkins:
  build:
    context: ./jenkins
    dockerfile: Dockerfile
  user: root
  ports:
    - "8081:8080"
    - "50001:50000"
  volumes:
    - jenkins_home:/var/jenkins_home
    - /var/run/docker.sock:/var/run/docker.sock
  networks:
    - project-network

nginx:
  image: nginx:latest
  ports:
    - "80:80"
    - "443:443"
  volumes:
```

```
    - ./nginx/nginx.conf:/etc/nginx/conf.d/default.conf
    - /etc/letsencrypt:/etc/letsencrypt:ro
  depends_on:
    backend:
      condition: service_healthy
    frontend:
      condition: service_started
  healthcheck:
    test: [ "CMD", "curl", "-f", "http://localhost" ]
    interval: 10s
    timeout: 5s
    retries: 3
  networks:
    - project-network

mongo-db:
  image: mongo:latest
  ports:
    - "27017:27017"
  environment:
    MONGO_INITDB_ROOT_USERNAME: ${MONGODB_USER}
    MONGO_INITDB_ROOT_PASSWORD: ${MONGODB_PASSWORD}
    MONGO_INITDB_DATABASE: ${MONGODB_DATABASE}
  volumes:
    - mongo_data:/data/db
  healthcheck:
    test: [ "CMD", "mongosh", "-u", "${MONGODB_USER}", "-p", "${MONGODB_PASSWORD}", "--authenticationDatab
ase", "${MONGODB_DATABASE}", "--eval", "db.runCommand({ ping: 1 })" ]
    interval: 10s
    timeout: 5s
    retries: 3
  networks:
    project-network:
      aliases:
        - mongodb

elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:8.5.3
  container_name: elasticsearch
  environment:
    - discovery.type=single-node
    - xpack.security.enabled=true
    - ELASTIC_PASSWORD=${ELASTICSEARCH_PASSWORD}
    - network.host=0.0.0.0
    - http.host=0.0.0.0
    - ES_JAVA_OPTS=-Xms1g -Xmx2g  # 서버 메모리 관리를 위해 힙 메모리 제한

  ports:
    - "9200:9200"
  volumes:
    - elasticsearch_data:/usr/share/elasticsearch/data
    - elasticsearch_plugins:/usr/share/elasticsearch/plugins
  networks:
    - project-network
  healthcheck:
    test: [ "CMD-SHELL", "curl -u elastic:${ELASTICSEARCH_PASSWORD} -fsSL http://localhost:9200/_cluster/health |
grep -q '\"status\":\"green\"' || exit 1" ]
    interval: 15s
```

```
    timeout: 10s
    retries: 5

redis:
  image: redis:latest
  container_name: redis
  restart: always
  command: redis-server --requirepass ${REDIS_PASSWORD} --appendonly yes
  ports:
    - "6379:6379"
  environment:
    - REDIS_PASSWORD=${REDIS_PASSWORD}
  volumes:
    - redis_data:/data
  networks:
    - project-network
  healthcheck:
    test: [ "CMD", "redis-cli", "-a", "${REDIS_PASSWORD}", "ping" ]
    interval: 10s
    timeout: 5s
    retries: 3

nodejs:
  build:
    context: ./nodejs
    dockerfile: Dockerfile
  ports:
    - "4000:4000"
  environment:
    - MEDIASOUP_SERVER=${MEDIASOUP_SERVER}
    - MEDIASOUP_PORT=4001  # 내부 통신용 포트 지정

  networks:
    - project-network



mediasoup:
  image: dmandry/mediasoup-demo:latest
  container_name: mediasoup-demo
  ports:
    - "4001:4000"
    - "40000-40100:40000-40100/udp"
  environment:
    - PROTOO_LISTEN_PORT=4000
    - MEDIASOUP_LISTEN_IP=0.0.0.0
    - MEDIASOUP_ANNOUNCED_IP=${MEDIASOUP_ANNOUNCED_IP}
    - MEDIASOUP_MIN_PORT=40000
    - MEDIASOUP_MAX_PORT=40100
  networks:
    - project-network


networks:
  project-network:
    driver: bridge
```

```
volumes:
  db_data:
  jenkins_home:
  mongo_data:
  redis_data:
  elasticsearch_data:
  elasticsearch_plugins:
```

3. **Jenkins로 CI Pipeline 구성**

```
pipeline {
    agent any
    tools {
        dockerTool 'DefaultDocker'
        nodejs 'NodeJS-20'
    }
    environment {
        SPRING_DATASOURCE_URL = credentials('SPRING_DATASOURCE_URL')
        SPRING_DATASOURCE_USERNAME = credentials('SPRING_DATASOURCE_USERNAME')
        SPRING_DATASOURCE_PASSWORD = credentials('SPRING_DATASOURCE_PASSWORD')
        MONGODB_URI = credentials('MONGODB_URI')
        MONGODB_DATABASE = credentials('MONGODB_DATABASE')
        ELASTICSEARCH_URI = credentials('ELASTICSEARCH_URI')
        ELASTICSEARCH_USERNAME = credentials('ELASTICSEARCH_USERNAME')
        ELASTICSEARCH_PASSWORD = credentials('ELASTICSEARCH_PASSWORD')
        REDIS_HOST = credentials('REDIS_HOST')
        REDIS_PORT = credentials('REDIS_PORT')
        REDIS_PASSWORD = credentials('REDIS_PASSWORD')
        S3_BUCKET= credentials('S3_BUCKET')
        S3_ACCESSKEY = credentials('S3_ACCESSKEY')
        S3_SECRETKEY = credentials('S3_SECRETKEY')
        S3_REGION = credentials('S3_REGION')
        GOOGLE_CLIENT_ID = credentials('GOOGLE_CLIENT_ID')
        GOOGLE_CLIENT_SECRET = credentials('GOOGLE_CLIENT_SECRET')
        GOOGLE_REDIRECT_URI = credentials('GOOGLE_REDIRECT_URI')
        KAKAO_CLIENT_ID = credentials('KAKAO_CLIENT_ID')
        KAKAO_CLIENT_SECRET = credentials('KAKAO_CLIENT_SECRET')
        KAKAO_REDIRECT_URI = credentials('KAKAO_REDIRECT_URI')
        NEXT_PUBLIC_API_BASE_URL = credentials('NEXT_PUBLIC_API_BASE_URL')
    }
    stages {
        stage('Backend Build') {
            steps {
                dir('backend') {
                    sh 'chmod +x ./gradlew'
                    sh './gradlew clean build'
                }
            }
        }
        stage('Frontend Build') {
            steps {
                dir('frontend') {
                    sh '''
                    npm install
```

```
                npm run build
                '''
            }
        }
    }
    stage('Docker Compose (Test)') {
        steps {
            script {
                sh '''
                docker-compose up -d --build --no-deps backend frontend
                '''
            }
        }
    }
    stage('Check Running Containers') {
        steps {
            script {
                sh 'docker ps -a'
            }
        }
    }
    post {
        always {
            script {
                sh '''
                docker-compose down || true
                '''
            }
        }
    }
}
```

6. 프로젝트 실행 방법

백엔드

```
# Build stage
FROM openjdk:17-jdk-slim AS build
# 작업 디렉토리 설정
WORKDIR /app
# 프로젝트 소스 코드 복사
COPY . .
# Gradle Wrapper를 사용하여 애플리케이션 빌드
RUN ./gradlew bootJar --no-daemon
# 실행 stage
FROM openjdk:17-jdk-slim
# 작업 디렉토리 설정
WORKDIR /app

RUN apt update && apt install -y wget curl && rm -rf /var/lib/apt/lists/*

# 빌드된 JAR 파일 복사
COPY --from=build /app/build/libs/*.jar app.jar
```

```
# 애플리케이션 실행
ENTRYPOINT ["java", "-jar", "app.jar"]
```

프론트엔드

```
# 1. Node.js를 사용한 빌드 단계
FROM node:18-alpine AS build
WORKDIR /app

# 환경 변수 설정을 위한 ARG 추가
ARG NEXT_PUBLIC_API_BASE_URL
ENV NEXT_PUBLIC_API_BASE_URL=${NEXT_PUBLIC_API_BASE_URL}

COPY package*.json ./
RUN npm install

COPY . .
RUN npm run build

FROM node:18-alpine AS production
WORKDIR /app

COPY --from=build /app/package.json /app/package.json
COPY --from=build /app/node_modules /app/node_modules
COPY --from=build /app/.next /app/.next
COPY --from=build /app/public /app/public

ENV NODE_ENV=production
ENV NEXT_PUBLIC_API_BASE_URL=${NEXT_PUBLIC_API_BASE_URL}


EXPOSE 3000

CMD ["npm", "run", "start"]
```

Nginx

```
server {
    listen 80;
    server_name i12a307.p.ssafy.io;

    # Certbot 인증 요청 처리
    location /.well-known/acme-challenge/ {
        root /var/www/certbot;
    }

    # HTTPS로 리디렉션
    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name i12a307.p.ssafy.io;
```

```
# SSL 인증서 경로
ssl_certificate /etc/letsencrypt/live/i12a307.p.ssafy.io/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/i12a307.p.ssafy.io/privkey.pem;


client_max_body_size 20M;

# 프론트엔드 리버스 프록시
location / {
    proxy_pass http://frontend:3000;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /api/ {
    rewrite ^/api/(.*)$ /$1 break;
    proxy_pass http://backend:8080/;
    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-For $proxy_add_x_forwarded_for;

    proxy_buffering off;
    proxy_cache off;
    proxy_read_timeout 3600;
}

location /swagger-ui/ {
    proxy_pass http://backend:8080/swagger-ui/;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /swagger-ui.html {
    proxy_pass http://backend:8080/swagger-ui.html;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}

location /v3/api-docs {
    proxy_pass http://backend:8080/v3/api-docs;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Access-Control-Allow-Origin *;
    proxy_set_header Access-Control-Allow-Methods "GET, POST, OPTIONS, DELETE, PUT, PATCH";
    proxy_set_header Access-Control-Allow-Headers "Content-Type, Authorization";
}

location /oauth2/success {
    proxy_pass http://backend:8080/oauth2/success;
    proxy_set_header Access-Control-Allow-Origin http://i12a307.p.ssafy.io;
```

```
        proxy_set_header Access-Control-Allow-Credentials on;
    }

    location /login/oauth2/code/ {
        proxy_pass http://backend:8080/login/oauth2/code/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }

    location /socket.io/ {
        proxy_pass http://nodejs:4000;
        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-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 3600;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Jenkins

```
FROM jenkins/jenkins:lts

USER root

RUN apt-get update && apt-get install -y docker.io

RUN getent group docker || groupadd -g 122 docker

RUN usermod -aG docker jenkins

RUN curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)"
    && chmod +x /usr/local/bin/docker-compose \
    && ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

USER jenkins
```

7. `.gitignore` 파일에 포함된 내용

Frontend

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
/node_modules
/.pnp
.pnp.*
.yarn/*
!.yarn/patches
```

```
!.yarn/plugins
!.yarn/releases
!.yarn/versions

# testing
/coverage

# next.js
/.next/
/out/

# production
/build

# misc
.DS_Store
*.pem

# debug
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.pnpm-debug.log*

# env files (can opt-in for committing if needed)
.env*

# vercel
.vercel

# typescript
*.tsbuildinfo
next-env.d.ts

public/sw.js
public/sw.js.map
public/workbox-*.js
public/workbox-*.js.map
```

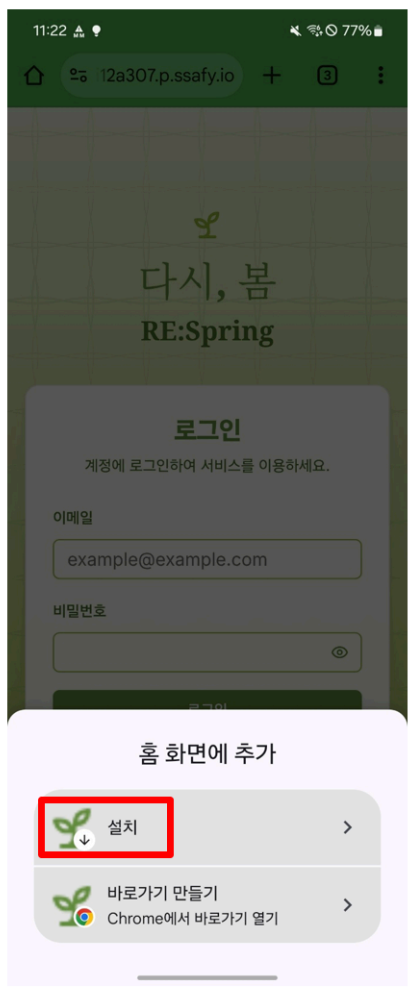
Backend

```
HELP.md
.gradle
.DS_Store
build/
!gradle/wrapper/gradle-wrapper.jar
!**/src/main/**/build/
!**/src/test/**/build/

### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
```

```
!**/src/main/**/bin/
!**/src/test/**/bin/

### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr
out/
!**/src/main/**/out/
!**/src/test/**/out/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

### VS Code ###
.vscode/

../.env
.env

src/main/generated/
```

8. 설치 및 실행 - PWA 설치

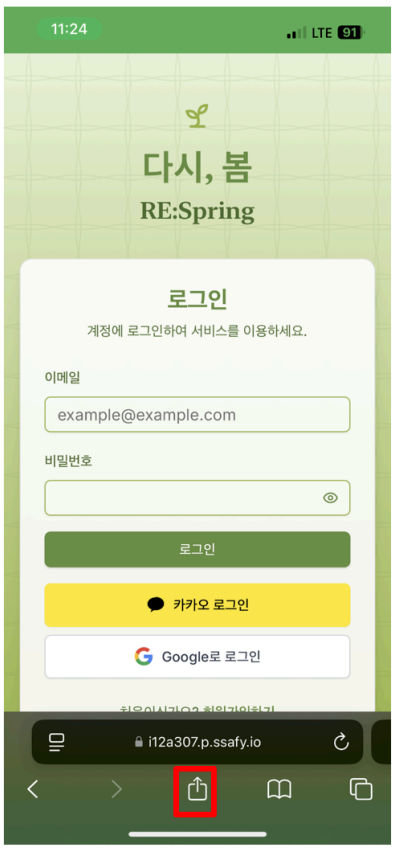
갤럭시(안드로이드)

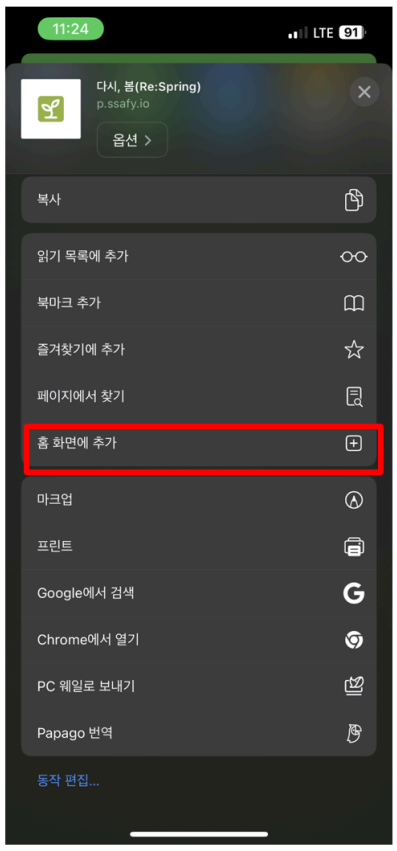


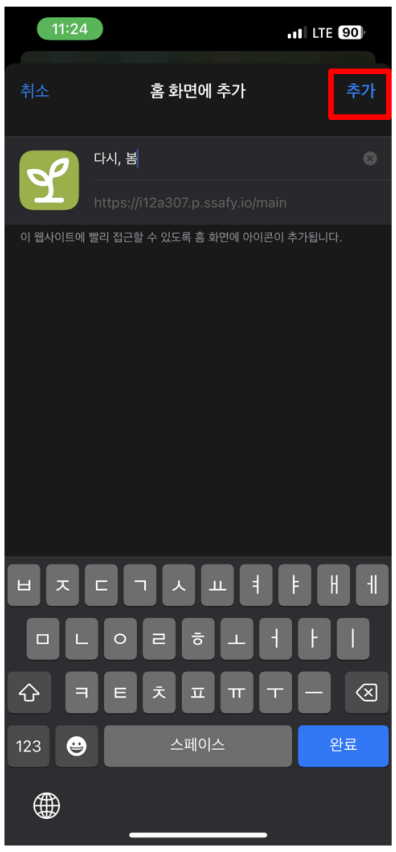
크롬(Chrome) 기준으로 상단 메뉴 중 더보기 아이콘을 탭하여 메뉴에 접근합니다.

홈 화면에 추가를 클릭한 후 설치 버튼을 누르면 홈 스크린에 “다시, 봄” 앱이 설치됩니다.

아이폰(IOS)







하단 브라우징 메뉴 중 중간에 있는 공유 아이콘을 탭하여 메뉴에 접근합니다.

`홈 화면에 추가`를 클릭 한 후 `추가` 버튼을 누르면 홈 스크린에 “다시, 봄” 앱이 설치됩니다.

9. 시연 시나리오

시연 시나리오

메인페이지

- 첫 사용자들이 경험하게 될 UX를 보여준다
- 온보딩을 하며 앱의 목적, 각 메뉴 기능들을 보여준다
- 회원가입 페이지를 예시로 보여주고, 미리 가입해둔 계정을 이용해 로그인한다

어제

- 메뉴에 들어가면 다양한 봄날의 서를 볼 수 있다는 것을 보여준다
- "난중일기"를 예시로 검색 기능 시연한다
- 각 서의 상세 페이지에 들어가 챕터, 댓글 등을 보여준다
- 뷰어를 체험하며 폰트 커스터마이징, 검색 기능 등을 시연한다
- 글조각 쓰기 메뉴에 들어가 글조각 조회 및 상세 보기 시연한다
- 글조각 작성과 타임라인 생성 메뉴를 보여준다
- 미리 작성된 글조각을 이용해 AI 엮기 기능, 미리보기, 수정, 표지 선택, 편찬을 시연한다

오늘

- 메뉴에 들어가면 다양한 정보 및 질문 글들을 볼 수 있다는 것을 보여준다
- "PWA" 게시글을 들어가 어떻게 프로젝트에 사용됐는지 보여주고 좋아요 버튼을 시연한다

내일

- 메뉴에 들어가면 참여중인 챌린지와 다양한 도전들을 볼 수 있다는 것을 보여준다

- 아직 참여하지 않은 챌린지에 들어가 참가하고 오늘치 완수 버튼을 시연한다
- 오픈 채팅에 들어가 다른 참여자들과의 채팅을 시연한다

나의 봄

- 메뉴에 들어가면 나의 프로필을 볼 수 있다는 것을 보여준다
- 각 탭을 눌러 타임라인, 활동 내역 기능 등을 보여준다

알림 메뉴

- 실시간으로 알림을 받아 알림의 출처에 방문한다
- 알림 메뉴에 들어가면 지금까지 받은 알림 내역을 볼 수 있다는 것을 보여준다

채팅

- 채팅방에 들어가서 기존에 생성된 채팅방에 들어간다
- 다른 사용자의 프로필에 들어가 새로운 채팅방을 생성한뒤 채팅을 시연한다
- 끝