

포팅매뉴얼

개요

프로젝트 소개

Moobeing - 마음을 움직이는 생활 플랫폼

Moobeing은 사용자의 금융 생활을 관리하고 개선하는 것을 목표로 한 통합 플랫폼입니다. 대출 상환, 소비 분석, 자산 파악 등을 시각화하고, 사용자에게 맞춤형 금융 서비스를 제공하여 더 나은 금융 결정을 내릴 수 있도록 돕습니다.

프로젝트 목적

Moobeing은 사용자가 자신의 금융 생활을 보다 쉽게 관리하고, 대출 상환 및 자산 관리를 효율적으로 할 수 있도록 돕는 것을 목표로 합니다. 또한, 개인의 소비 습관과 자산 관리를 개선하여 금융적 목표 달성에 기여합니다.

타겟 사용자

- 금융 대출 상환 계획을 관리하고자 하는 사용자
- 여러 통장을 관리하며 자산 현황을 한눈에 파악하고 싶은 사용자
- 소비 내역을 분석하고 자신만의 소비 성향을 알고 싶은 사용자
- 금융 관련 정보를 쉽게 얻고자 하는 사용자

서비스 개요

- 1. 대출 상환 내역 시각화 및 중도상환**
 - 대출 상품의 상환 과정을 선그래프 형태로 시각화
 - 전체 및 특정 대출 상품에 대한 연도별 그래프 제공
 - 또래와의 비교 기능 추가
- 2. 통장 모아보기 기능**
 - 여러 통장의 자산을 통합하여 한 화면에서 조회
- 3. 소비 내역 분석 및 성향 시각화**
 - 소비 내역을 파이 차트와 달력으로 시각화
 - 소비 성향을 분석하여 PDF 보고서 제공
- 4. 무캡슐(타임캡슐) 기능**
 - 대출 상환 및 퀴즈를 통해 얻은 포인트로 소비 기록을 타임캡슐로 저장
 - 특정 위치에서 타임캡슐을 심고, 해당 위치로 이동하면 수확 가능
- 5. 금융 지식 및 서비스 챗봇**
 - AI 기반 챗봇을 통해 금융 관련 정보 제공
 - 랭체인 및 RAG 기술을 사용하여 신뢰성 있는 응답 제공
- 6. 지출 내역 분석을 통한 대출 상환 장려**
 - 지출 내역 분석을 통해 대출 상환을 독려

타 서비스와의 차별성

- 1. 무심기(소비 내역 타임캡슐)**
 - 소비 내역을 기반으로 타임캡슐을 심고 수확
 - 사진과 글을 함께 저장 가능하며, 수확 시 포인트 제공

2. MooBTi(소비 유형 분석)

- 사용자의 월 소비 내역을 분석하여 MooBTi 소비 유형 제공
- 공유 기능을 통해 사용자 트래픽 증가 및 맞춤형 금융 서비스 제공

3. 대출 상환 및 소비 시각화

- 대출 상환 및 소비 내역을 그래프와 차트로 시각화하여 이해도 향상

4. 금융 지식 챗봇

- RAG 기술을 활용하여 금융 지식을 신뢰성 있게 제공

협업 툴

- 이슈 관리 : JIRA
- 형상 관리: Gitlab
- 커뮤니케이션 : Notion , Mattermost
- 디자인 : Figma

개발 환경

- Frontend
 - VS code : 1.90.2
 - Node.js : v20.15.1(LTS)
 - NPM : 10.7.0
 - Vite : 5.3.1
 - axios : 1.7.2
- Backend
 - IntelliJ : 17.0.11+1-b1207.24 amd64
 - JDK : Java17 OpenJDK
 - Spring boot : 3.3.1

외부 서비스

- KakaoMap <https://apis.map.kakao.com/web/>

환경 변수 및 설정 파일 목록

- Frontend

- .env

```
REACT_APP_NAVER_MAP_CLIENT_ID=  
REACT_APP_NAVER_MAP_CLIENT_SECRET=  
REACT_APP_KAKAO_MAP_CLIENT_ID=  
REACT_APP_KAKAO_MAP_APP_KEY=  
REACT_APP_BASE_URL=https://j11a404.p.ssafy.io/api
```

- Backend

- Application.yml

```
server:  
  servlet:  
    context-path: /api  
  session:
```

```

        cookie:
            same-site: none
            secure: true

spring:
    application:
        name: moobeing
    servlet:
        multipart:
            max-file-size: 50MB # 파일당 최대 크기
            max-request-size: 50MB # 전체 요청의 최대 크기

    sql:
        init:
            mode: always

    config:
        import: optional:file:.env[.properties]

    datasource:
        driver-class-name: com.mysql.cj.jdbc.Driver
        url: jdbc:mysql://${HOST}:${PORT}/${DATABASE_NAME}?${OPTIONS}
        username: ${USER_NAME}
        password: ${USER_PASSWORD}

    jpa:
        show-sql: true
        hibernate:
            ddl-auto: create-drop
        defer-datasource-initialization: true
        properties:
            hibernate:
                globally_quoted_identifiers: false
                format_sql: true
                dialect: org.hibernate.dialect.MySQL8Dialect
            open-in-view: true

    logging:
        level:
            com.app: debug

SSAFY_APIKEY: ${SSAFY_APIKEY}

openai:
    api:
        key: ${OPENAI_KEY}

fcm:
    type: ${FCM_TYPE}
    project-id: ${FCM_PROJECT_ID}
    private-key-id: ${FCM_PRIVATE_KEY_ID}
    private-key: ${FCM_PRIVATE_KEY}
    client-email: ${FCM_CLIENT_EMAIL}
    client-id: ${FCM_CLIENT_ID}
    auth-uri: ${FCM_AUTH_URI}
    token-uri: ${FCM_TOKEN_URI}
    auth-provider-x509-cert-url: ${FCM_AUTH_PROVIDER_X509_CERT_URL}

```

```
client-x509-cert-url: ${FCM_CLIENT_X509_CERT_URL}
universe-domain: ${FCM_UNIVERSE_DOMAIN}
```

- `.env`

```
HOST=localhost
PORT=3306
DATABASE_NAME=
USER_NAME=
USER_PASSWORD=
OPTIONS=serverTimezone=UTC
SSAFY_APIKEY=
MINIO_ACCESS_KEY=
MINIO_SECRET_KEY=
MINIO_URL=
MINIO_REGION=
MINIO_BUCKET=
OPENAI_KEY=
FCM_TYPE=
FCM_PROJECT_ID=
FCM_PRIVATE_KEY_ID=
FCM_PRIVATE_KEY=
FCM_CLIENT_EMAIL=
FCM_CLIENT_ID=
FCM_AUTH_URI=
FCM_TOKEN_URI=
FCM_AUTH_PROVIDER_X509_CERT_URL=
FCM_CLIENT_X509_CERT_URL=
```

빌드 및 배포

도커 설정

- 공통

```
apt-get update
apt-get install docker

docker network create bookkoo-net

# 사용할 MySQL DB 컨테이너들 생성
docker run -d \
  --name mysql_container \
  -e MYSQL_USER={DB 패스워드} \
  -e MYSQL_PASSWORD={DB 패스워드} \
  -e MYSQL_DB={DB 명} \
  -v {사용할 볼륨}

# Nginx Proxy Manager
docker run -d \
  --name nginx-proxy-manager \
  --restart always \
  -p 80:80 \
  -p 443:443 \
  -p {외부에서 접속할 포트}:81 \
  -e DB_MYSQL_HOST={npm DB 주소} \
```

```
-e DB_MYSQL_PORT={npm DB 주소} \  
-e DB_MYSQL_USER={npm DB 주소} \  
-e DB_MYSQL_PASSWORD={npm DB 주소} \  
-e DB_MYSQL_NAME={npm DB 주소} \  
-v {사용항볼륨} \  
jc21/nginx-proxy-manager:latest  
  
# Front, Backend 폴더 안에서  
docker run -d \  
    --name {서비스명} \  
    -p {외부에서 접속할 포트}:{내부포트} \  

```