포팅 매뉴얼

개요

프로젝트 개요

[사용자 데이터 기반 소비습관 형성 도우미 서비스] - 소비습관 개선 서비스

1. 사용자 소비내역 기반 퍼센트, 절대금액 바탕 강제 벌금
2. 벌금 및 소비내역 대시보드로 한 눈에 사용자의 소비습관 파악
3. AI 분석으로 소비내역 자동 태그 분류 기능
4. 3개월 간 소비 내역 분석 모델이 내놓는 소비 내역 예측 서비스
5. 소비자의 소비 행태 기반 가장 비용을 절감할 수 있는 카드 추천

프로젝트 사용 도구

1. 이슈 관리: JIRA
2. 형상 관리: Gitlab
3. 커뮤니케이션: Mattermost, Notion, Google Docs
4. 디자인: Figma
5. UCC: 모바비
6. CI/CD: Jenkins, Docker-compose

개발 환경

1. IDE
    - Visual Studio Code 1.99.0 (프론트엔드 개발)
    - IntelliJ 2024.3.2 (백엔드 개발)

2. 언어 및 런타임
    - JVM: OpenJDK 17
    - Node.js: 18.20
3. 서버 환경
    - AWS EC2 Ubuntu 22.04.5 LTS
4. 컨테이너 환경
    - Docker Compose
5. 데이터베이스
    - MySQL 8.0.41 (EC2 내 직접 설치)
    - Redis (Docker Compose로 관리)
    - Elasticsearch 8.17.4 (Docker Compsoe로 관리)

외부 서비스

1. Firebase Cloud Messaging
2. Simple Mail Transfer Protocol

빌드 및 실행

환경변수

1. 프론트엔드 .env

```
VITE_BASE_URL=/api
VITE_FRONT_URL=https://j12c207.p.ssafy.io:3000
VITE_API_KEY=AIzaSyBhjnI5Xw916c9qUBuxLInTGTxqiCb3n34
VITE_AUTH_DOMAIN=mmmproject-5c97f.firebaseapp.com
VITE_PROJECT_ID=mmmproject-5c97f
VITE_STORAGE_BUCKET=mmmproject-5c97f.firebasestorage.app
VITE_MESSAGING_SENDER_ID=623476923226
VITE_APP_ID=1:623476923226:web:1ae823850315d54f543163
```

```
VITE_VAPID_KEY=BN1p6wiXgDkcoKGEwCa3zS07I5IzvvhpfABzw_TgUlg
KlVXmO6xxv-VxmcUibUFwM9hjnyI4H80uKjXC4RISUok
VITE_MEASURE_MENT_ID=G-SVFLJWXGWK
```

2. 백엔드 .env

```
SMTP_EMAIL=ocsba990821@gmail.com
SMTP_PASS=hsru xuwr udfj kxzp
REDIS_HOST=redis
REDIS_PORT=6379
REDIS_DURATION=300
DATABASE_URL=jdbc:mysql://j12c207.p.ssafy.io:3306/S12P21C207?use
SSL=false&allowPublicKeyRetrieval=true&serverTimezone=Asia/Seoul&
characterEncoding=UTF-8&useUnicode=true
DATABASE_USERNAME=enugu
DATABASE_PASSWORD=ssafy_c207
SMS_KEY=NCSSTCJHYHUYXZHN
SMS_SECRET=RD2AUB1EEW0RTXSZK0ZJ4SEIESQSSGCK
SMS_PHONE=01092035370
ELASTIC=http://j12c207.p.ssafy.io:9200
ALLOWED_IPS=127.0.0.1,172.26.6.140,172.17.0.0/16,172.18.0.0/16,0:0:0:0:
0:0:0:1
AI_API_URL=https://j12c207.p.ssafy.io/ai
FCM_CERTIFICATION=firebase/mmmproject-5c97f-firebase-adminsdk-
fbsvc-6422dfc52b.json
```

3. AI .env

```
KMP_DUPLICATE_LIB_OK=TRUE
OPENAI_API_KEY=sk-proj-Ao_xTDp7JIXoblcxrmvKTB_JGA1gDVmDVrnu
hOO1fx1ZXR8ZAp2ssodGEyPijOX_F2i3Ci0_lBT3BlbkFJjtFGiR8n7dUlN8e
KgWw1RgGg6cb57O7LcHG_3BL67M5OtyCsCp3t0KSJKwZ4BZVUz3Pd
Rqw3UA
```

4. application.properties

```
spring.application.name=TMT
server.servlet.context-path=/api
```

```
spring.jackson.time-zone=Asia/Seoul

# database
spring.datasource.url= ${DATABASE_URL}
spring.datasource.username= ${DATABASE_USERNAME}
spring.datasource.password= ${DATABASE_PASSWORD}
spring.datasource.driver-class-name= com.mysql.cj.jdbc.Driver

## mybatis
mybatis.config-location=classpath:mapper/config/sqlmap-config.xml
mybatis.mapper-locations=mapper/*.xml

# Redis
spring.data.redis.host=${REDIS_HOST}
spring.data.redis.port=${REDIS_PORT}
spring.data.redis.duration=${REDIS_DURATION}

## email
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=${SMTP_EMAIL}
spring.mail.password=${SMTP_PASS}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

# JPA
spring.jpa.hibernate.ddl-auto=none
#spring.jpa.show-sql=true
#spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Di
alect

# coolSMS
coolsms.apikey=${SMS_KEY}
coolsms.apisecret=${SMS_SECRET}
coolsms.fromnumber=${SMS_PHONE}

# log
```

```
logging.level.org.springframework.web=DEBUG

# session
server.servlet.session.timeout=60m

# Elastic
spring.data.elasticsearch.repositories.enabled=true
#spring.data.elasticsearch.url=${ELASTIC}
spring.elasticsearch.uris=${ELASTIC}



# actuator
management.endpoints.web.exposure.include=health
management.endpoint.health.show-details=always

# mongoDB, rabbitMQ
management.health.mongo.enabled=false
management.health.rabbit.enabled=false
spring.autoconfigure.exclude=org.springframework.boot.autoconfigur
e.mongo.MongoAutoConfiguration

ai.api.url=${AI_API_URL}

app.tomcat.redirect.enabled=false

## HTTPS ?? (?: PKCS12 ??? ???? ??)
#server.ssl.key-store=classpath:localhost.p12
#server.ssl.key-store-password=changeit
#server.ssl.key-store-type=PKCS12
#server.ssl.key-alias=1
#
## HTTPS ??? ??? ??? ?? (?? HTTPS ??? 8443)
#server.port=8443
```

Docker

1. Docker Compose

```
services:
  nginx:
    build: /home/ubuntu/nginx # Dockerfile 경로
    container_name: nginx-container
    ports:
    - "80:80"
    - "443:443"
    volumes:
    - /etc/letsencrypt:/etc/letsencrypt:ro # SSL 인증서를 컨테이너에 제공
    - /home/ubuntu/nginx/conf.d:/etc/nginx/conf.d # Nginx 설정 파일을
컨테이너에 제공
    restart: always
    networks:
    - app-network

  frontend:
    image: chaaaaaaanmi/mmm-frontend:latest
    container_name: react-container
    restart: always
    networks:
    - app-network

  backend:
    image: chaaaaaaanmi/mmm-backend:latest
    container_name: backend
    restart: always
    networks:
      - app-network
    depends_on:
      - redis
      - elasticsearch

  ai:
    image: chaaaaaaanmi/mmm-ai:latest
    container_name: ai
    volumes:
      - ./python_back/data:/app/data  # FAISS 인덱스 파일을 위한 볼륨
    restart: always
```

```
    networks:
      - app-network

  redis:
    image: redis:latest
    container_name: redis
    restart: always
    networks:
      - app-network

  jenkins:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: jenkins-container
    ports:
    - "9090:8080"
    - "50000:50000"
    volumes:
    - ./jenkins_home:/var/jenkins_home # Jenkins 설정과 작업 이력 저장 (컨테이너가 재시작되더라도 데이터 유지됨)
    - /var/run/docker.sock:/var/run/docker.sock # 호스트의 Docker 데몬에 접근할 수 있도록 (Jenkins 컨테이너 내에서 Docker 명령어 실행)
    restart: always
    networks:
    - app-network

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.17.4
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - "ES_JAVA_OPTS=-Xms512m -Xmx512m"
      - xpack.security.enabled=false # 보안 기능 비활성화
    ports:
      - "9200:9200"
    volumes:
      - elasticsearch-data:/usr/share/elasticsearch/data
```

```
        networks:
            - app-network

volumes:
    elasticsearch-data:

networks:
    app-network:
        external: true
```

서버 설정

1. EC2 포트 정보
    - 22: SSH
    - 80: HTTP(Nginx)
    - 443: HTTPS(Nginx)
    - 8080: backend
    - 3306: MySQL
2. NginX 설정 파일

```
# HTTP → HTTPS 리디렉션
server {
    listen 80;
    listen [::]:80;
    server_name j12c207.p.ssafy.io;

    return 301 https://$host$request_uri;
}

# HTTPS
server {
    listen 443 ssl;
    server_name j12c207.p.ssafy.io;
```

```
    # SSL 인증서 파일 경로 설정
    ssl_certificate /etc/letsencrypt/live/j12c207.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j12c207.p.ssafy.io/privkey.pe
m;

    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certb
ot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Cert
bot

    # Front 요청 처리 (React)
    location / {
        proxy_pass http://react-container:3000;  # React 컨테이너 연결
        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 X-Forwarded-Proto $scheme;
    }

    # Actuator 요청 제한
    location /api/actuator/ {
        # 내부 접근만 허용
        allow 127.0.0.1;        # 로컬호스트 (EC2 내부에서 직접 접근)
        allow 43.201.45.107;    # EC2 퍼블릭 IP (SSH 접속 후 외부 도메인으로
접근 시)
        allow 172.17.0.0/16;    # Docker 네트워크 대역 (젠킨스 컨테이너 포함)
        allow 172.18.0.0/16;    # 추가 Docker 네트워크 대역
        deny all;               # 그 외 모든 접근 차단

        # 프록시 설정
        proxy_pass http://backend:8080/api/actuator/;  # Spring Boot 컨테
이너 연결
        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 X-Forwarded-Proto $scheme;
    }
```

```
    # Back API 요청 처리 (Spring Boot)
    location /api/ {
        proxy_pass http://backend:8080/api/;  # Spring Boot 컨테이너 연결
        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 X-Forwarded-Proto $scheme;

        # 🔥 쿠키 경로 수정 (세션 쿠키 전달 문제 해결)
        proxy_cookie_path /api/ /;
    }

    # AI API 요청 처리 (Fast API)
    location /ai/ {
        proxy_pass http://ai:8100/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

3. NginX 도커 파일

```
FROM nginx:alpine

# 기존 설정 파일 제거
RUN rm -rf /etc/nginx/conf.d/*

# 커스텀 설정 파일 복사
COPY conf.d/ /etc/nginx/conf.d/

EXPOSE 80
EXPOSE 443

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

빌드 및 배포

1. Jenkins 파이프라인

```
pipeline {
    agent any

    environment {
        GITLAB_URL = 'https://lab.ssafy.com/s12-fintech-finance-sub1/S12
P21C207.git'
        BRANCH = 'deploy'
        DOCKER_HUB_USER = 'chaaaaaaanmi'
        IMAGE_NAME_FRONTEND = 'mmm-frontend'
        IMAGE_NAME_BACKEND = 'mmm-backend'
        IMAGE_NAME_AI = 'mmm-ai'
        IMAGE_TAG = 'latest'
        SERVER_USER = 'ubuntu'
        SERVER_HOST = 'j12c207.p.ssafy.io'
        MAX_RETRIES = 5
        RETRY_INTERVAL = 10
    }

    stages {
        stage('Checkout') {
            steps {
                echo '📥 GitLab에서 최신 코드 가져오기'
                sh "pwd"
                git branch: env.BRANCH,
                    url: env.GITLAB_URL,
                    credentialsId: 'gitlab-http-credentials'
            }
        }

        stage('Docker Login') {
            steps {
                echo '🔑 Docker Hub 로그인'
                withCredentials([usernamePassword(credentialsId: 'docker-h
```

```
ub-credentials',
                passwordVariable: 'DOCKER_HUB_PASS', usernameVariable:
'DOCKER_HUB_USER')]) {
                    sh "echo \$DOCKER_HUB_PASS | docker login -u \$DOCKE
R_HUB_USER --password-stdin"
                }
            }
        }

        stage('Build Frontend') {
            steps {
                echo '🏗️ 프론트엔드 이미지 빌드 및 푸시'
                withCredentials([file(credentialsId: 'FRONT_ENV_FILE', variabl
e: 'FRONT_ENV_FILE')]) {
                    sh """
                        # cat 명령어로 내용 복사
                        cat \$FRONT_ENV_FILE > ./Front/MMM/.env

                        # .env 삭제 예약 (스크립트 종료 시 실행)
                        trap 'rm -f ./Front/MMM/.env' EXIT

                        # 프론트엔드 이미지 빌드
                        cd ./Front/MMM
                        docker build -t ${DOCKER_HUB_USER}/${IMAGE_NAME_
FRONTEND}:${IMAGE_TAG} .

                        # Docker Hub 푸시
                        docker push ${DOCKER_HUB_USER}/${IMAGE_NAME_F
RONTEND}:${IMAGE_TAG}

                        # 정리
                        docker image prune -f
                    """
                }
            }
        }

        stage('Build Backend') {
```

```
        steps {
            echo '🏗️ 백엔드 이미지 빌드 및 푸시'
            withCredentials([file(credentialsId: 'BACK_ENV_FILE', variable:
'BACK_ENV_FILE')]) {
                sh """
                    # cat 명령어로 내용 복사
                    cat \$BACK_ENV_FILE > ./Back/MMM/.env

                    # .env 삭제 예약 (스크립트 종료 시 실행)
                    trap 'rm -f ./Back/MMM/.env' EXIT

                    # 백엔드 이미지 빌드
                    cd ./Back/MMM
                    docker build -t ${DOCKER_HUB_USER}/${IMAGE_NAME_
BACKEND}:${IMAGE_TAG} .

                    # Docker Hub 푸시
                    docker push ${DOCKER_HUB_USER}/${IMAGE_NAME_B
ACKEND}:${IMAGE_TAG}

                    # 정리
                    docker image prune -f
                """
            }
        }
    }

    stage('Build AI Backend') {
        steps {
            echo '🏗️ AI 백엔드 이미지 빌드 및 푸시'
            withCredentials([file(credentialsId: 'AI_ENV_FILE', variable: 'AI
_ENV_FILE')]) {
                sh """
                    echo "현재 경로: \$(pwd)"

                    # python_back 디렉토리 존재 확인 및 생성
                    mkdir -p ./python_back
```

```
                    # 디렉토리 권한 설정
                    chmod -R 777 ./python_back

                    # cat 명령어로 내용 복사
                    cat \$AI_ENV_FILE > ./python_back/.env

                    # .env 파일 권한 설정
                    chmod 644 ./python_back/.env

                    # .env 삭제 예약 (스크립트 종료 시 실행)
                    trap 'rm -f ./python_back/.env' EXIT

                    # AI 백엔드 이미지 빌드
                    cd ./python_back
                    docker build -t ${DOCKER_HUB_USER}/${IMAGE_NAME_
AI}:${IMAGE_TAG} .

                    # Docker Hub 푸시
                    docker push ${DOCKER_HUB_USER}/${IMAGE_NAME_A
I}:${IMAGE_TAG}

                    # 정리
                    docker image prune -f
                """
            }
        }
    }

    stage('Deploy') {
        steps {
            echo '🚀 배포 시작'
            withCredentials([sshUserPrivateKey(credentialsId: 'jenkins-ss
h-key', keyFileVariable: 'SSH_KEY')]) {
                sh """
                    ssh -o StrictHostKeyChecking=no -i \$SSH_KEY ${SERV
ER_USER}@${SERVER_HOST} '
                        cd /home/ubuntu/S12P21C207 &&
                        docker-compose pull &&
```

```
                    docker-compose down frontend backend ai nginx redi
s &&
                    docker-compose up -d frontend backend ai nginx redi
s
                '
            """
        }
    }
}

    stage('Health Check') {
        steps {
            echo '🩺 애플리케이션 헬스 체크 수행'
            withCredentials([sshUserPrivateKey(credentialsId: 'jenkins-ss
h-key', keyFileVariable: 'SSH_KEY')]) {
                sh """
                    ssh -o StrictHostKeyChecking=no -i \$SSH_KEY ${SERV
ER_USER}@${SERVER_HOST} '
                    echo "Performing health check..."
                    MAX_RETRIES=${MAX_RETRIES}
                    RETRY_INTERVAL=${RETRY_INTERVAL}
                    HEALTH_CHECK_URL="https://j12c207.p.ssafy.io/api/a
ctuator/health"

                    for i in \$(seq 1 \$MAX_RETRIES); do
                        echo "Health check attempt \$i of \$MAX_RETRIE
S..."
                        # HTTP 상태 코드 확인
                        HTTP_STATUS=\$(curl -s -o /dev/null -w "%{http_co
de}" \$HEALTH_CHECK_URL)

                        if [ "\$HTTP_STATUS" = "200" ]; then
                            echo "✅ 애플리케이션 정상 응답 (HTTP 200)"
                            exit 0
                        else
                            echo "⚠️ 헬스 체크 실패. HTTP 상태 코드: \$HTTP_S
TATUS"
                            echo "재시도 중... (\$RETRY_INTERVAL초 대기)"
```

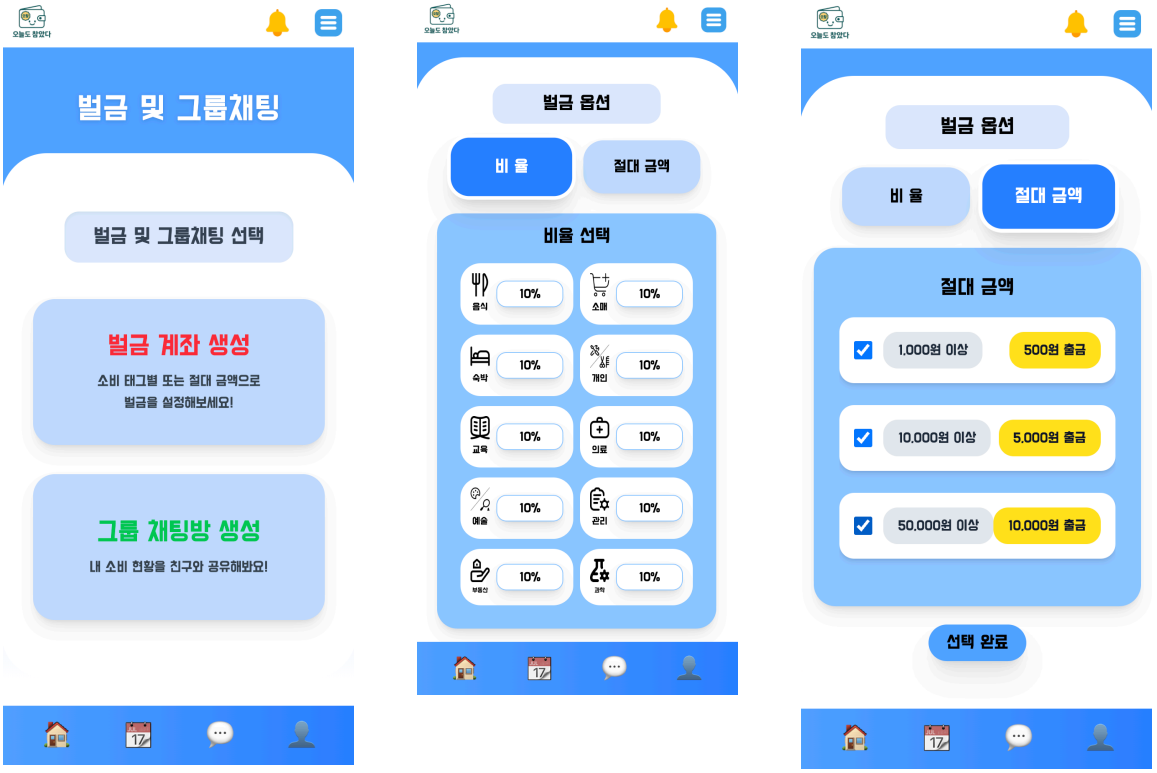
```
                        sleep \$RETRY_INTERVAL
                    fi
                done

                echo "❌ \$MAX_RETRIES번 시도. 애플리케이션이 응답하지 않습니다."
                exit 1
            '
        """
      }
    }
  }


  post {
    always {
      script {
        // 공통 변수를 환경 변수로 설정
        env.AUTHOR_ID = sh(script: "git show -s --pretty=%an", returnStdout: true).trim()
        env.AUTHOR_NAME = sh(script: "git show -s --pretty=%ae", returnStdout: true).trim()
        env.COMMIT_MESSAGE = sh(script: "git log -1 --pretty=%B", returnStdout: true).trim()
      }
      echo '🧹 로그아웃 및 정리 작업'
      sh 'docker logout || true'
    }
    success {
      echo '🎉 배포가 성공적으로 완료되었습니다!'
      mattermostSend(color: 'good',
        message: "### 🤖 배포 성공 | ${env.JOB_NAME} #${env.BUILD_NUMBER}\n" +
                 "**개발자**: ${env.AUTHOR_ID} (${env.AUTHOR_NAME})\n" +
                 "**커밋 내용**:\n```\n${env.COMMIT_MESSAGE}\n```\n" +
                 "[상세 정보](${env.BUILD_URL})",
```

```
            endpoint: 'https://meeting.ssafy.com/hooks/tretq3tc3bgb5y5
zqkx6kxuzpo',
            channel: 'Jenkins'
        )
    }
    failure {
        echo '❌ 배포 중 오류가 발생했습니다'
        mattermostSend(color: 'danger',
            message: "### 🤖 배포 실패 | ${env.JOB_NAME} #${env.BUIL
D_NUMBER}\n" +
                "**개발자**: ${env.AUTHOR_ID} (${env.AUTHOR_NAM
E})\n" +
                "**커밋 내용**:\n```\n${env.COMMIT_MESSAGE}\n```\n" +
                "[상세 정보](${env.BUILD_URL})",
            endpoint: 'https://meeting.ssafy.com/hooks/tretq3tc3bgb5y5
zqkx6kxuzpo',
            channel: 'Jenkins'
        )
    }
  }
}
```

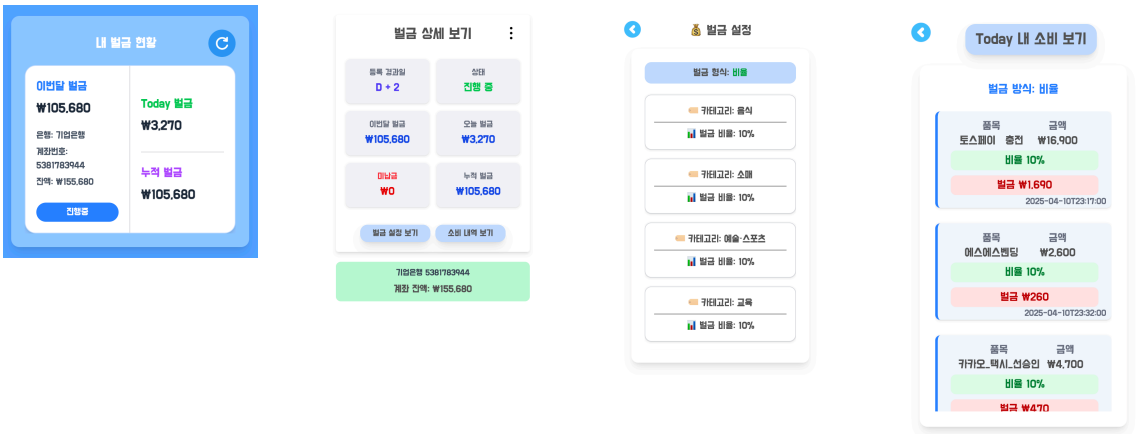
사용자 시나리오

[벌금]

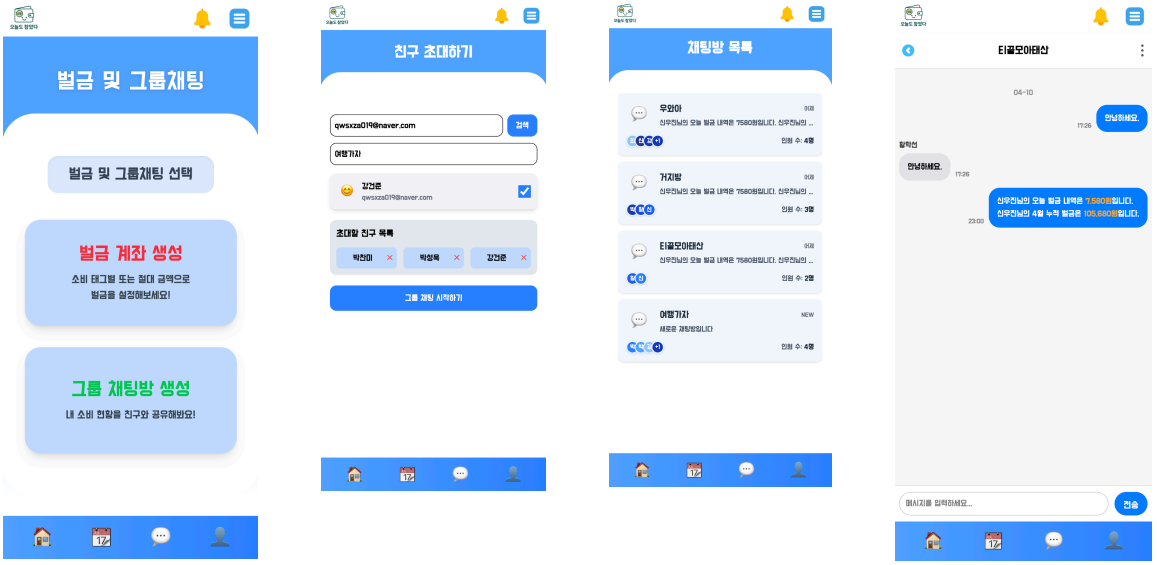
1. 벌금 계좌 생성 및 벌금 방식 선택



2. 벌금 현황 및 상세 조회



3. 매일 오후 11시 사용자의 소비내역을 바탕으로 벌금액이 산정됩니다.
4. 그룹 채팅방을 생성하여 친구와 함께 자유로운 소통 및 소비 현황 공유

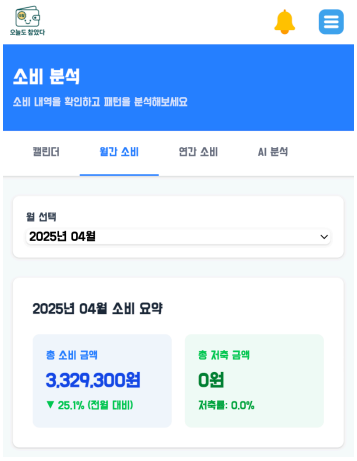


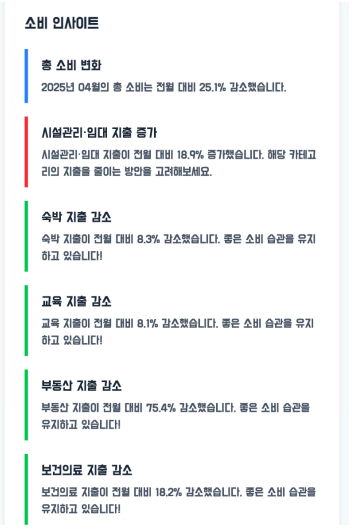
[소비 분석]

1. 캘린더에서 해당 일자의 소비 내역과 그에 따른 벌금액 확인 및 고정지출 확인

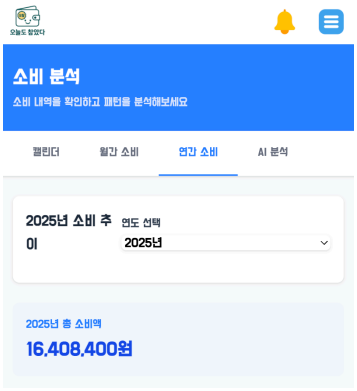

오늘도 참았다
소비 분석
소비 내역을 확인하고 패턴을 분석해보세요
캘린더
월간 소비
연간 소비
AI 분석
2025년 4월
일 월 화 수 목 금 토
30 31 1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 1 2 3
2025년 4월 11일 금요일
오늘의 총 소비
1,100,000원
오늘의 총 저축
10,000원
고정 지출
월세
국민은행 2309145785
1,000,000원
소비 내역 2025-04-11
나무PC
오후 10:46
4,500원
저축: 450원
KB카드출금
오후 08:22
4,800원
저축: 480원
케블2025031
오후 07:32
700원
저축: 70원
네이버페이_DEGICA
오후 01:47
1,000원
저축: 100원

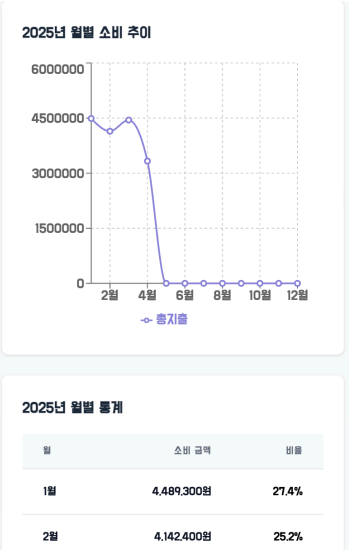
2. 월간 소비 내역 분석



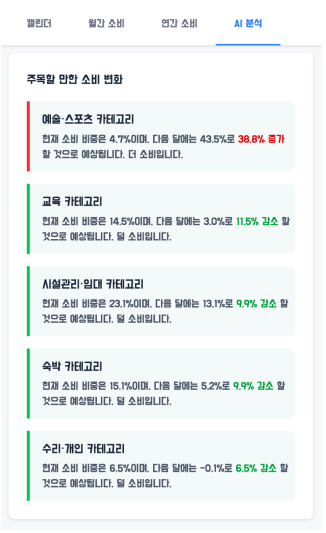


3. 연간 소비 내역 분석





4. AI를 통한 소비 패턴 분석



5. 소비 패턴을 바탕으로 카드 추천