

구현 완료 요약

프로젝트 개요

Express + TypeScript + Prisma + AWS RDS MySQL 기반의 카카오톡/구글 소셜 로그인 인증 서버가 성공적으로 구현되었습니다.

완성된 기능

✓ 1. 프로젝트 초기화 및 설정

- Express + TypeScript 프로젝트 구조
- 필요한 모든 의존성 설치
- TypeScript 설정 (`tsconfig.json`)
- Nodemon을 통한 개발 환경 구성
- 환경변수 관리 (`.env`, `.env.example`)

✓ 2. 데이터베이스 설계 (Prisma + MySQL)

- **User 모델**: 사용자 기본 정보 및 프로필
- **SocialAccount 모델**: 소셜 로그인 계정 연결
- **PhoneVerification 모델**: 전화번호 인증 이력
- **JwtToken 모델**: JWT 토큰 발급/사용 이력
- MySQL 최적화 (VARCHAR 사용, 인덱스 설정)

✓ 3. 인증 시스템

OAuth 2.0 소셜 로그인

- **구글 로그인**: passport-google-oauth20 전략
- **카카오 로그인**: passport-kakao 전략
- 소셜 프로필 정보 추출 및 저장

전화번호 인증

- CoolSMS API 연동
- 6자리 인증 코드 생성 및 발송
- 5분 만료 시간 설정
- 스팸 방지 (시도 횟수 제한)
- 개발 모드에서 콘솔 출력

계정 통합 로직

- 전화번호 기준 동일 사용자 판별
- 기존 사용자에게 새 소셜 계정 자동 연결
- 카카오에서 전화번호 제공 시 즉시 계정 생성

✓ 4. JWT 토큰 시스템

- **Access Token:** 15분 유효기간
- **Refresh Token:** 7일 유효기간
- 토큰 발급, 갱신, 검증 기능
- 토큰 무효화 (로그아웃 시)
- 데이터베이스에 토큰 이력 저장
- 만료된 토큰 자동 정리 (1시간마다)

✓ 5. API 엔드포인트

인증 관련

- `GET /auth/google` - 구글 로그인 시작
- `GET /auth/google/callback` - 구글 OAuth 콜백
- `GET /auth/kakao` - 카카오 로그인 시작
- `GET /auth/kakao/callback` - 카카오 OAuth 콜백
- `POST /auth/send-verification` - SMS 인증 코드 발송
- `POST /auth/verify-phone` - 전화번호 인증 확인
- `POST /auth/complete-social-login` - 소셜 로그인 완료
- `POST /auth/refresh` - 토큰 갱신
- `POST /auth/logout` - 로그아웃

사용자 관련

- `GET /auth/me` - 현재 사용자 정보 조회
- `PUT /auth/profile` - 프로필 업데이트
- `DELETE /auth/account` - 계정 비활성화

기타

- `GET /health` - 서버 상태 확인

✓ 6. 보안 기능

미들웨어

- **Helmet:** 보안 헤더 설정
- **CORS:** Cross-Origin 요청 제어
- **Rate Limiting:**
 - 일반 API: 100회/15분
 - 인증 API: 5회/15분
 - SMS 발송: 1회/1분
 - 토큰 갱신: 10회/1분
- **JWT 검증:** Bearer Token 인증

에러 처리

- 통합 에러 핸들러

- Prisma 에러 처리
- JWT 에러 처리
- 운영/개발 환경별 에러 메시지 처리
- 상세한 에러 로깅

✓ 7. 유틸리티 및 헬퍼

- **ResponseUtil**: 표준화된 API 응답
- **JwtUtil**: JWT 토큰 생성/검증
- **PhoneUtil**: 전화번호 정규화 및 검증
- **PrismaService**: 데이터베이스 연결 관리

✓ 8. 타입 정의 (TypeScript)

- 모든 API 요청/응답 타입
- 사용자 프로필 타입
- 소셜 프로필 타입
- JWT 페이로드 타입
- Passport 전략 타입

프로젝트 파일 구조

```
dba-portal-auth/
├── src/
│   ├── controllers/
│   │   └── auth.controller.ts      (223줄)
│   ├── middleware/
│   │   ├── auth.middleware.ts      (110줄)
│   │   ├── error.middleware.ts      (112줄)
│   │   ├── passport.middleware.ts   (96줄)
│   │   └── rate-limit.middleware.ts (81줄)
│   ├── routes/
│   │   └── auth.routes.ts          (78줄)
│   ├── services/
│   │   ├── jwt-token.service.ts     (166줄)
│   │   ├── phone-verification.service.ts (110줄)
│   │   ├── prisma.service.ts        (25줄)
│   │   ├── sms.service.ts           (50줄)
│   │   └── user.service.ts           (153줄)
│   ├── types/
│   │   ├── auth.types.ts            (69줄)
│   │   └── passport-kakao.d.ts      (32줄)
│   ├── utils/
│   │   ├── jwt.util.ts              (51줄)
│   │   ├── phone.util.ts            (58줄)
│   │   └── response.util.ts          (36줄)
│   ├── app.ts                       (79줄)
│   └── index.ts                     (88줄)
├── prisma/
│   └── schema.prisma                (87줄)
└── .env
```

```
|— .env.example
|— .gitignore
|— package.json
|— tsconfig.json
|— nodemon.json
|— README.md
|— API_DOCUMENTATION.md
|— DEPLOYMENT.md
|— GETTING_STARTED.md
|— PROJECT_PLAN.md
|— IMPLEMENTATION_SUMMARY.md
```

총 코드 라인 수: 약 **1,650줄** (주석 포함)

코드 품질

모듈화

- 500줄 미만의 모듈로 분리
- 단일 책임 원칙 준수
- 명확한 디렉토리 구조

타입 안정성

- 모든 함수와 변수에 타입 지정
- 인터페이스와 타입 정의 분리
- any 타입 최소화

에러 처리

- try-catch 블록으로 에러 처리
- 의미 있는 에러 메시지
- 로그 시스템 구축

보안

- 환경변수로 민감 정보 관리
- JWT 기반 인증
- Rate Limiting 적용
- 입력값 검증

문서화

README.md

- 프로젝트 개요
- 주요 기능 소개
- 기술 스택
- 설치 및 실행 방법

📄 GETTING_STARTED.md

- 빠른 시작 가이드
- 단계별 설정 방법
- 개발 모드 테스트 방법
- 문제 해결

📄 API_DOCUMENTATION.md

- 모든 API 엔드포인트 상세 설명
- 요청/응답 예시
- 인증 방법
- 에러 코드

📄 DEPLOYMENT.md

- AWS RDS 설정 가이드
- OAuth 설정 방법
- Docker/PM2 배포 방법
- Nginx 설정
- 보안 체크리스트

📄 PROJECT_PLAN.md

- 프로젝트 계획 및 설계
- 구현 단계
- 기술 스택

실행 방법








개발 환경

```
npm install
npm run prisma:generate
npm run prisma:migrate
npm run dev
```

프로덕션 환경

```
npm install
npm run build
npm run prisma:deploy
npm start
```

테스트 가능 기능

1.  Health Check API
2.  전화번호 인증 (콘솔 출력)
3.  구글 OAuth (실제 설정 필요)
4.  카카오 OAuth (실제 설정 필요)
5.  JWT 토큰 발급/갱신
6.  사용자 정보 조회/수정
7.  로그아웃

환경 설정 필요 항목

실제 운영을 위해 다음 항목을 설정해야 합니다:

1. **AWS RDS MySQL**: 데이터베이스 인스턴스 생성
2. **Google OAuth**: 클라이언트 ID/Secret 발급
3. **Kakao OAuth**: 앱 등록 및 API 키 발급
4. **CoolSMS**: API 키 발급 및 발신번호 등록
5. **JWT Secret**: 강력한 랜덤 문자열 생성

특징

설계 원칙 준수

- RESTful API 설계
- 표준 응답 구조
- 에러 처리 일관성
- 코드 재사용성

보안 강화

- JWT 기반 인증
- Rate Limiting
- CORS 제어
- 보안 헤더

유연한 인증

- 다중 소셜 로그인 지원
- 전화번호 기반 계정 통합
- 자동 계정 연결

확장 가능

- 모듈화된 구조
- 서비스 레이어 분리
- 타입 안정성

다음 단계 제안

1. **테스트 코드 작성**: Jest를 사용한 단위/통합 테스트
2. **API 문서 자동화**: Swagger/OpenAPI 통합

3. 로깅 시스템: Winston 또는 Pino 도입
4. 모니터링: Prometheus + Grafana 연동
5. CI/CD 파이프라인: GitHub Actions 설정
6. 역할 기반 권한: Admin, User 등 역할 추가
7. 이메일 인증: 이메일 기반 인증 추가
8. 2FA 인증: 2단계 인증 구현

완료 확인

✅ 모든 TODO 항목 완료 ✅ TypeScript 컴파일 성공 ✅ Prisma 스키마 유효성 검증 통과 ✅ 서버 정상 실행
확인 ✅ 코드 모듈화 (500줄 미만) ✅ 문서화 완료

프로젝트가 성공적으로 완료되었습니다! 🎉