# KSP 57Invest – Starter Monorepo (Frontend + Backend + DB + API Spec)

Monorepo awal siap dikloning untuk KSP 57Invest: mobile (Expo React Native), admin panel (Next.js), backend API (NestJS + Prisma + PostgreSQL), OpenAPI spec, seed data, dan pipeline dev lokal.

## 📁 Struktur Folder

57invest/  
├── README.md  
├── package.json # pnpm workspaces  
├── pnpm-workspace.yaml  
├── .gitignore  
├── .env.example # variabel lingkungan root  
├── apps/  
│ ├── admin/ # Next.js 14 (app router)  
│ │ ├── package.json  
│ │ ├── next.config.mjs  
│ │ ├── tsconfig.json  
│ │ ├── src/  
│ │ │ ├── app/  
│ │ │ │ ├── layout.tsx  
│ │ │ │ ├── page.tsx  
│ │ │ │ ├── api-client.ts  
│ │ │ │ └── (features)/  
│ │ │ │ ├── members/page.tsx  
│ │ │ │ ├── loans/page.tsx  
│ │ │ │ ├── savings/page.tsx  
│ │ │ │ ├── deposits/page.tsx  
│ │ │ │ └── reports/page.tsx  
│ │ └── .env.example  
│ └── mobile/ # Expo React Native  
│ ├── package.json  
│ ├── app.json  
│ ├── babel.config.js  
│ ├── app/  
│ │ ├── \_layout.tsx  
│ │ ├── index.tsx # Login/Register  
│ │ ├── savings.tsx  
│ │ ├── loans.tsx  
│ │ ├── deposits.tsx  
│ │ └── profile.tsx  
│ └── .env.example  
├── services/  
│ └── api/ # NestJS + Prisma  
│ ├── package.json  
│ ├── tsconfig.json  
│ ├── nest-cli.json  
│ ├── prisma/  
│ │ ├── schema.prisma  
│ │ └── seed.ts  
│ ├── src/  
│ │ ├── main.ts  
│ │ ├── app.module.ts  
│ │ ├── common/  
│ │ │ ├── guards/jwt-auth.guard.ts  
│ │ │ ├── decorators/user.decorator.ts  
│ │ │ └── dto/pagination.dto.ts  
│ │ ├── auth/  
│ │ │ ├── auth.module.ts  
│ │ │ ├── auth.controller.ts  
│ │ │ ├── auth.service.ts  
│ │ │ └── jwt.strategy.ts  
│ │ ├── members/  
│ │ │ ├── members.module.ts  
│ │ │ ├── members.controller.ts  
│ │ │ └── members.service.ts  
│ │ ├── savings/  
│ │ │ ├── savings.module.ts  
│ │ │ ├── savings.controller.ts  
│ │ │ └── savings.service.ts  
│ │ ├── loans/  
│ │ │ ├── loans.module.ts  
│ │ │ ├── loans.controller.ts  
│ │ │ └── loans.service.ts  
│ │ ├── deposits/  
│ │ │ ├── deposits.module.ts  
│ │ │ ├── deposits.controller.ts  
│ │ │ └── deposits.service.ts  
│ │ ├── reports/  
│ │ │ ├── reports.module.ts  
│ │ │ ├── reports.controller.ts  
│ │ │ └── reports.service.ts  
│ │ └── parameters/  
│ │ ├── parameters.module.ts  
│ │ ├── parameters.controller.ts  
│ │ └── parameters.service.ts  
│ └── .env.example  
└── specs/  
 └── openapi.yaml # definisi API

## 🧪 Menjalankan Secara Lokal

### Prasyarat

* Node.js 18+
* pnpm npm i -g pnpm
* PostgreSQL 14+ (lokal atau via Docker)

### 1) Clone & Install

pnpm i

### 2) Bootstrap Database

Set variabel di services/api/.env:

DATABASE\_URL="postgresql://postgres:postgres@localhost:5432/ksp57invest?schema=public"  
JWT\_SECRET="devsecret"  
PORT=4000

Lalu jalankan:

pnpm -C services/api prisma:dev # generate + migrate + seed  
pnpm -C services/api start:dev # run API http://localhost:4000

### 3) Jalankan Admin Panel (Next.js)

Set apps/admin/.env:

NEXT\_PUBLIC\_API\_BASE\_URL="http://localhost:4000"

Lalu:

pnpm -C apps/admin dev # http://localhost:3000

### 4) Jalankan Mobile (Expo)

Set apps/mobile/.env:

EXPO\_PUBLIC\_API\_BASE\_URL="http://localhost:4000"

Lalu:

pnpm -C apps/mobile start

## 📦 Root Configs

**package.json (root)**

{  
 "name": "57invest",  
 "private": true,  
 "packageManager": "pnpm@9.0.0",  
 "scripts": {  
 "dev:api": "pnpm -C services/api start:dev",  
 "dev:admin": "pnpm -C apps/admin dev",  
 "dev:mobile": "pnpm -C apps/mobile start"  
 },  
 "workspaces": ["apps/\*", "services/\*"]  
}

**pnpm-workspace.yaml**

packages:  
 - "apps/\*"  
 - "services/\*"

## 🗄️ Prisma Schema (services/api/prisma/schema.prisma)

datasource db {  
 provider = "postgresql"  
 url = env("DATABASE\_URL")  
}  
  
generator client {  
 provider = "prisma-client-js"  
}  
  
model Member {  
 id String @id @default(cuid())  
 code String @unique // M-0001, dll  
 name String  
 email String @unique  
 password String // hashed  
 phone String?  
 address String?  
 idNumber String?  
 bank String?  
 accountNo String?  
 status MemberStatus @default(ACTIVE)  
 createdAt DateTime @default(now())  
 updatedAt DateTime @updatedAt  
  
 savings Saving[]  
 loans Loan[]  
 deposits Deposit[]  
}  
  
enum MemberStatus { ACTIVE INACTIVE }  
  
enum SavingType { IN OUT }  
  
model Saving {  
 id String @id @default(cuid())  
 memberId String  
 type SavingType  
 amount Int  
 date DateTime @default(now())  
 offline Boolean @default(false)  
 member Member @relation(fields: [memberId], references: [id])  
}  
  
model Loan {  
 id String @id @default(cuid())  
 memberId String  
 amount Int  
 tenor Int // months  
 rateYear Float // %/year  
 status LoanStatus @default(PENDING)  
 createdAt DateTime @default(now())  
 member Member @relation(fields: [memberId], references: [id])  
 schedules LoanSchedule[]  
}  
  
enum LoanStatus { PENDING APPROVED REJECTED }  
  
model LoanSchedule {  
 id String @id @default(cuid())  
 loanId String  
 dueDate DateTime  
 amount Int  
 paid Boolean @default(false)  
 loan Loan @relation(fields: [loanId], references: [id])  
}  
  
model Deposit {  
 id String @id @default(cuid())  
 memberId String  
 principal Int  
 tenor Int // months  
 rateYear Float // %/year compound monthly  
 startDate DateTime @default(now())  
 maturityDate DateTime  
 active Boolean @default(true)  
 closedAt DateTime?  
 payout Int?  
 closedEarly Boolean? @default(false)  
 member Member @relation(fields: [memberId], references: [id])  
}  
  
model Parameter {  
 id Int @id @default(1)  
 loanRate Float @default(18)  
 depositRate Float @default(6)  
 lateFee Float @default(2)  
}

## 🌱 Seed Data (services/api/prisma/seed.ts)

import { PrismaClient } from "@prisma/client";  
import bcrypt from "bcryptjs";  
  
const prisma = new PrismaClient();  
  
async function main() {  
 const password = await bcrypt.hash("123456", 10);  
 const member = await prisma.member.upsert({  
 where: { email: "member@ksp57invest.co.id" },  
 update: {},  
 create: {  
 code: "M-0001",  
 name: "Member Demo",  
 email: "member@ksp57invest.co.id",  
 password,  
 bank: "BCA",  
 accountNo: "1234567890",  
 status: "ACTIVE",  
 },  
 });  
  
 await prisma.parameter.upsert({  
 where: { id: 1 },  
 update: {},  
 create: { id: 1, loanRate: 18, depositRate: 6, lateFee: 2 },  
 });  
  
 console.log({ member });  
}  
  
main().finally(() => prisma.$disconnect());

**scripts (services/api/package.json)**

{  
 "scripts": {  
 "prisma:generate": "prisma generate",  
 "prisma:migrate": "prisma migrate dev --name init",  
 "seed": "tsx prisma/seed.ts",  
 "prisma:dev": "pnpm prisma:generate && pnpm prisma:migrate && pnpm seed",  
 "start:dev": "nest start --watch"  
 }  
}

## 🧰 NestJS Bootstrap (services/api/src)

**main.ts**

import { NestFactory } from '@nestjs/core';  
import { AppModule } from './app.module';  
import { ValidationPipe } from '@nestjs/common';  
import { DocumentBuilder, SwaggerModule } from '@nestjs/swagger';  
  
async function bootstrap() {  
 const app = await NestFactory.create(AppModule, { cors: true });  
 app.useGlobalPipes(new ValidationPipe({ whitelist: true, transform: true }));  
  
 const config = new DocumentBuilder()  
 .setTitle('57Invest API')  
 .setVersion('0.1.0')  
 .addBearerAuth()  
 .build();  
 const doc = SwaggerModule.createDocument(app, config);  
 SwaggerModule.setup('docs', app, doc);  
  
 await app.listen(process.env.PORT || 4000);  
}  
bootstrap();

**app.module.ts**

import { Module } from '@nestjs/common';  
import { AuthModule } from './auth/auth.module';  
import { MembersModule } from './members/members.module';  
import { SavingsModule } from './savings/savings.module';  
import { LoansModule } from './loans/loans.module';  
import { DepositsModule } from './deposits/deposits.module';  
import { ReportsModule } from './reports/reports.module';  
import { ParametersModule } from './parameters/parameters.module';  
  
@Module({  
 imports: [AuthModule, MembersModule, SavingsModule, LoansModule, DepositsModule, ReportsModule, ParametersModule],  
})  
export class AppModule {}

**Auth (ringkas)**

* /auth/login (member email+password → JWT)
* /auth/me (profil)

**Members (ringkas)**

* CRUD anggota (admin)

**Savings (ringkas)**

* POST /savings/deposit|withdraw
* GET /savings/history?memberId=

**Loans (ringkas)**

* POST /loans/apply
* GET /loans (list by member/admin)
* PATCH /loans/:id/approve|reject
* PATCH /loans/:loanId/schedules/:id/togglePaid

**Deposits (ringkas)**

* POST /deposits/create
* PATCH /deposits/:id/redeem (maturity)
* PATCH /deposits/:id/close-early (penalti)

**Reports (ringkas)**

* GET /reports/summary

## 🔐 DTO Contoh (services/api/src/loans/dto)

import { IsInt, Min } from 'class-validator';  
export class ApplyLoanDto {  
 @IsInt() @Min(1) amount: number;  
 @IsInt() @Min(1) tenor: number; // months  
}

## 🧾 OpenAPI Spec (specs/openapi.yaml)

openapi: 3.0.3  
info:  
 title: 57Invest API  
 version: 0.1.0  
servers:  
 - url: http://localhost:4000  
paths:  
 /auth/login:  
 post:  
 summary: Login member  
 requestBody:  
 required: true  
 content:  
 application/json:  
 schema:  
 type: object  
 properties:  
 email: { type: string }  
 password: { type: string }  
 responses:  
 '200': { description: OK }  
 /auth/me:  
 get:  
 security: [{ bearerAuth: [] }]  
 summary: Get profile  
 responses: { '200': { description: OK } }  
  
 /members:  
 get:  
 security: [{ bearerAuth: [] }]  
 summary: List members (admin)  
 responses: { '200': { description: OK } }  
  
 /savings/deposit:  
 post:  
 security: [{ bearerAuth: [] }]  
 summary: Add savings depo