Survey Management Application

A full-stack survey management application built with NestJS backend, React frontend, PostgreSQL database, and Redis for caching. The application allows users to create surveys, manage questionnaires, handle population data, and analyze survey responses.

# 🚀 Quick Start

## Prerequisites

Before running the application, ensure you have the following installed:

- Docker Desktop - https://www.docker.com/products/docker-desktop

- Git - https://git-scm.com/downloads

## Installation & Setup

1. Clone the repository

git clone <repository-url>  
cd <project-directory>

2. Create environment file

touch .env

3. Configure environment variables

# Database Configuration not nessassary when running using docker compose  
POSTGRES\_USER=postgres   
POSTGRES\_PASSWORD=postgres  
POSTGRES\_DB=mydatabase  
POSTGRES\_HOST=postgres  
POSTGRES\_PORT=5432  
  
# Redis Configuration not nessassary when running using docker compose  
REDIS\_HOST=redis  
REDIS\_PORT=6379  
  
# JWT Configuration  
JWT\_SECRET=your-super-secret-jwt-key  
JWT\_EXPIRES\_IN=24h  
  
# Application Configuration  
NODE\_ENV=production  
PORT=3000  
  
# File Upload Configuration  
MAX\_FILE\_SIZE=2000000000  
UPLOAD\_DEST=./uploads  
  
# OPEN AI KEYS   
AZURE\_OPENAI\_API\_KEY=  
AZURE\_OPENAI\_ENDPOINT=  
AZURE\_OPENAI\_API\_VERSION=2024-05-01-preview  
AZURE\_OPENAI\_DEPLOYMENT\_NAME=gpt-4o  
  
# AWS KEYS  
not nessassary when running locally just set STORAGE\_DRIVER="local"  
  
STORAGE\_DRIVER="s3" || "local"  
AWS\_REGION=  
AWS\_ACCESS\_KEY=  
AWS\_SECRET\_KEY=  
AWS\_BUCKET\_NAME=  
  
# GAMIL USER FOR SENDING EMAILS  
not nessassary when running locally  
GMAIL\_USER=  
GMAIL\_PASSWORD= or GMAIL\_APP\_PASSWORD=

4. Start the application

docker-compose up -d

5. Wait for services to start

docker-compose logs -f

## Accessing the Application

Once all services are running, you can access the application at:

- Frontend: http://localhost:80 or http://localhost:8080

- Backend API: http://localhost:3000

- Database: localhost:5432 (PostgreSQL)

- Redis: localhost:6379

# 🔧 Configuration

## Environment Variables

The application uses environment variables for configuration. Key variables include:

- POSTGRES\_\*: Database connection settings

- REDIS\_\*: Redis connection settings

- JWT\_\*: JWT authentication settings

- NODE\_ENV: Application environment

- PORT: Application port

## Database Setup

The PostgreSQL database is automatically initialized with the following default credentials:

- Username: postgres

- Password: postgres

- Database: mydatabase

## File Uploads

SET THE ENV STORAGE\_DRIVER TO "local" when running on local dev envirnment and to "s3" when running multiple instance. The application supports file uploads with a maximum size of 2GB. Uploaded files are stored in the configured upload directory.

# 🐛 Troubleshooting

## Common Issues

1. Port conflicts

# Check if ports are in use  
netstat -an | findstr :80  
netstat -an | findstr :3000  
netstat -an | findstr :5432

2. Docker not running

- Ensure Docker Desktop is started

- Check Docker service status

3. Build failures

# Clean and rebuild  
docker-compose down  
docker system prune -f  
docker-compose up --build

4. Database connection issues

# Check database logs  
docker-compose logs postgres  
  
# Restart database  
docker-compose restart postgres