NDIS Management System

1) Repository overview

- **frontend/** — React app (Vite). Handles UI, routing, forms, and calling backend APIs.
- **backend/** — FastAPI app. Exposes REST endpoints for Participants, Documents, and SIL Homes/Rooms.
- **.env.example** — Safe template for environment variables you copy to a local .env (not committed).
- **docker-compose.yml** *(optional)* — Compose services if you want to run DB + backend together.
- **README.md** — High-level overview.

---

2) Folder & file responsibilities

Root

- **.env.example**: list of required environment variables with placeholder values (no secrets).
- **docker-compose.yml** *(if present)*: spins up services (e.g., Postgres + backend).
- **README.md**: quick start notes for the whole repo.

backend/

- **backend/app/main.py**
  - Creates the FastAPI app (FastAPI()), sets title/description, mounts API routers, configures CORS, and may register middleware/events.
- **backend/app/api/v1/api.py**

    - Combines the versioned routers (e.g., participants, documents, sil_homes) under a base path such as /api/v1.

- **backend/app/api/v1/participants.py**
    - Endpoints for participants (CRUD, search, etc.).
- **backend/app/api/v1/documents.py**
    - Endpoints for document upload/listing/versioning/expiry checks.
- **backend/app/api/v1/sil_homes.py**
    - Endpoints for SIL homes and rooms (create home, manage rooms).
- **backend/app/schemas/**
    - Pydantic models defining request/response payloads (e.g., Participant, Document, SILHome).
- **backend/app/models/** *(if present)*
    - Database models (e.g., SQLAlchemy) & mappings.
- **backend/app/core/** *(if present)*
    - App settings, security deps, constants, logging config.
- **backend/app/services/** *(if present)*
    - Business logic (e.g., document expiry, quotation generation).
- **backend/app/utils/** *(if present)*
    - Helpers (formatting, validators, common utilities).
- **backend/tests/** *(if present)*
    - Unit/integration tests for backend.

- **backend/requirements.txt**
  - Python dependencies (fastapi, uvicorn, pydantic, sqlalchemy, psycopg2-binary, etc.).

frontend/

- **frontend/package.json**
  - Project metadata, scripts (dev, build, preview) and dependencies.
- **frontend/src/index.tsx** or **frontend/src/main.tsx**
  - React app bootstrap (creates root and renders <App />).
- **frontend/src/App.tsx**
  - Routes + high-level layout (sidebar/topbar), context providers.
- **frontend/src/pages/***
  - Page-level components (Participants, Documents, SIL Homes/Rooms, Dashboard, etc.).
- **frontend/src/components/***
  - Reusable UI (forms, tables, modals, inputs).
- **frontend/src/context/*** *(if present)*
  - Context providers (e.g., AuthContext).
- **frontend/.env.*** *(local only)*
  - Frontend env files for Vite (e.g., VITE_API_URL). Not committed.

---

3) Environment & configuration

Copy .env.example → .env and fill real values **locally** (do not commit .env).

Example (backend):

```
# backend/.env (not committed)

DATABASE_URL=postgresql+psycopg2://user:pass@localhost:5432/ndis_db

CORS_ORIGINS=http://localhost:5173

APP_ENV=development
```

Example (frontend):

```
# frontend/.env.local (not committed)

VITE_API_URL=http://127.0.0.1:8000
```

---

4) Running the backend (FastAPI)

Windows (PowerShell)

```
cd backend

python -m venv venv

.\venv\Scripts\Activate.ps1

pip install -r requirements.txt



# Start the API with auto-reload

uvicorn app.main:app --reload --port 8000
```

macOS/Linux

```
cd backend
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt


# Start the API with auto-reload
uvicorn app.main:app --reload --host 127.0.0.1 --port 8000
```

Health checks (in another terminal):

```
curl http://127.0.0.1:8000/
curl http://127.0.0.1:8000/api/v1/participants
```

---

5) Running the frontend (React/Vite)

```
cd frontend
npm install      # installs node_modules locally (not committed)
npm run dev      # Vite dev server (default http://localhost:5173)
```

If your API runs on http://127.0.0.1:8000, set VITE_API_URL accordingly in frontend/.env.local.

---

6) API docs (“fancy URLs”)

FastAPI generates these automatically:

- **Swagger UI:** http://127.0.0.1:8000/docs
- **ReDoc:** http://127.0.0.1:8000/redoc
- **OpenAPI JSON:** http://127.0.0.1:8000/openapi.json

You can customize the paths by editing backend/app/main.py:

```
from fastapi import FastAPI


app = FastAPI(
    title="NDIS Platform API",
    description="APIs for Participants, Documents, SIL Homes/Rooms",
    version="1.0.0",
    docs_url="/api/docs",
    redoc_url="/api/redoc",
    openapi_url="/api/openapi.json",
)
```

Quick API tests (examples):

```
# Get participants
curl http://127.0.0.1:8000/api/v1/participants


# Create a participant (adjust fields to your schema)
curl -X POST http://127.0.0.1:8000/api/v1/participants \
  -H "Content-Type: application/json" \
  -d '{"name":"Alice Example","ndisNumber":"1234567890"}'
```

---

7) What .gitignore does (and template)

What it is: A list of patterns that tell Git which files/folders **not** to track (e.g., builds, caches, virtualenvs, local env files, node_modules).
Important: It only affects **untracked** files. If you’ve already committed something, untrack it with git rm -r --cached.

Common commands

```
# Why is this file ignored?
git check-ignore -v path/to/file


# Stop tracking a file/folder that should be ignored
git rm -r --cached path/to/file_or_dir
git commit -m "chore: untrack generated files"
```

Template for this repo (drop at repo root):

```
# --- OS / Editor ---
.DS_Store
Thumbs.db
*.swp
.idea/
```

```
.vscode/


# --- Env & secrets ---
.env
.env.*
!.env.example
*.pem
*.key


# --- Node / React ---
**/node_modules/
**/dist/
**/build/
**/.vite/
**/.cache/
npm-debug.log*
yarn-error.log*
pnpm-debug.log*
**/.eslintcache
**/.parcel-cache/
```

```
# --- Python / FastAPI ---
**/__pycache__/
**/*.py[cod]
**/.venv/
**/venv/
**/.mypy_cache/
**/.pytest_cache/
**/.ruff_cache/
**/.coverage
**/htmlcov/

# --- Docker / containers ---
docker-volumes/
**/.env.docker

# --- Local data / artifacts ---
logs/
*.log
tmp/
```

```
data/
```

---

8) Git: create a branch and push to main (no remote yet)

From your repo root:

```
# Ensure branch is 'main'

git branch -M main



# First commit (if you haven't)

git add .

git commit -m "chore: initial project"



# Create an empty GitHub repo (via web), copy its URL, then:

git remote add origin <YOUR_REPO_URL>



# Push main and set upstream

git push -u origin main



# Create and push a working branch (don’t work on main)

git switch -c develop

git push -u origin develop
```

Feature flow (daily use):

```
git switch develop

git pull

git switch -c feature/participants-onboarding

# ...work...

git add .

git commit -m "feat(participants): onboarding form + validations"

git push -u origin feature/participants-onboarding
```

---

9) Troubleshooting quick notes

- **node_modules is trying to commit**
  Add it to .gitignore (see template), then:
- git rm -r --cached frontend/node_modules
- git commit -m "chore: untrack node_modules"
- **__pycache__/.pyc files are tracked**
  Add patterns to .gitignore, then:
- git rm -r --cached backend/app/__pycache__
- git commit -m "chore: drop python caches"
- **frontend accidentally became a submodule (mode 160000)**
  Remove nested Git metadata and the .gitmodules entry:
- if (Test-Path .\frontend\.git) { Remove-Item -Recurse -Force .\frontend\.git }

- if (Test-Path .\.gitmodules) { git rm .gitmodules && git commit -m "chore: remove submodule file" }
- git add .
- git commit -m "chore: normalize frontend as regular folder"
- **CRLF/LF line-end warnings on Windows**
  Safe to ignore. To normalize:
- git config core.autocrlf true
- **CORS errors in the browser**
  Ensure backend CORS allows your frontend origin (e.g., http://localhost:5173).