Data Generation Guide

Complete guide for generating and maintaining reference data for the hiking trail recommendation system.

Table of Contents

1. Overview
2. Prerequisites
3. Initial Setup
4. Generating Reference Data
5. Using Reference Data
6. Updating Reference Data
7. Troubleshooting
8. Best Practices

Overview

This system uses **reference data files** (JSON) to ensure that all developers can generate exactly the same database content. This solves the problem of non-deterministic data generation caused by:

- **External API calls** (Open Elevation API)
- **Processing order variations** (shapefile reading order)
- **Timing differences** (dates, IDs)

Two Modes of Operation

1. **Reference Mode** (`--use-reference`): Uses pre-exported JSON files for exact reproducibility
2. **Fresh Mode** (default): Generates data from shapefile (may vary slightly between runs)

Prerequisites

Before generating reference data, ensure you have:

1. **Python virtual environment activated** `bash source .venv/bin/activate`
2. **Shapefile downloaded** `bash python download_trails_shapefile.py` This creates files in `data/source/`:
3. `hiking_foot_routes_lineLine.shp`
4. `hiking_foot_routes_lineLine.dbf`
5. `hiking_foot_routes_lineLine.shx`
6. `hiking_foot_routes_lineLine.prj`
7. **Required Python packages installed** `bash pip install -r requirements.txt`

Initial Setup

Step 1: Download the Shapefile

The shapefile contains the source trail data from OpenStreetMap.

```
cd adaptive_quiz_system
python download_trails_shapefile.py
```

Expected output:

```
======================================================================
DOWNLOADING HIKING TRAILS SHAPEFILE
======================================================================

This script will download the shapefile from data.gouv.fr
and extract it to data/source/ to be used by init_db.py

Fetching dataset information from data.gouv.fr...
✓ Found download URL: https://...
Downloading from: https://...
Progress: 100.0%
Downloaded 125.00 MB

Extracting hiking_trails.zip...
  Extracted: hiking_foot_routes_lineLine.shp
  Extracted: hiking_foot_routes_lineLine.dbf
  Extracted: hiking_foot_routes_lineLine.shx
  Extracted: hiking_foot_routes_lineLine.prj

✓ SUCCESS: Shapefile downloaded and extracted successfully!
```

Step 2: Generate Reference Data

Export the trails and completed trail assignments to JSON files.

2.1 Export Trails Reference

```
python data/seed/export_trails_reference.py
```

What this does: - Loads trails from the shapefile (calls elevation API) - Exports all trail characteristics to `trails_reference.json` - Includes elevation profiles (so API isn't needed later)

Expected output:

```
======================================================================
EXPORTING TRAILS REFERENCE DATA
======================================================================

Loading trails from shapefile...
Loaded 100 trails

✓ Exported 100 trails to:
  data/seed/trails_reference.json
  File size: 485.2 KB

This file can be versioned in Git and used for reproducible data generation.
```

2.2 Export Completed Trails Reference

```
python data/seed/export_completed_trails_reference.py
```

What this does: - Uses trails from `trails_reference.json` (or loads from shapefile if not found) - Generates completed trail assignments for all demo users - Exports assignments to `completed_trails_reference.json`

Expected output:

```
======================================================================
EXPORTING COMPLETED TRAILS REFERENCE DATA
======================================================================

Loading trails from reference JSON...
Loaded 100 trails

Generating completed trails assignments...

✓ Exported 45 completed trail assignments to:
  data/seed/completed_trails_reference.json
  File size: 8.5 KB
  Assigned to 15 users

This file ensures reproducible user profile generation.
```

Step 3: Verify Reference Data

Check that the files were created:

```
ls -lh data/seed/*.json
```

You should see: - `trails_reference.json` (~500 KB) - `completed_trails_reference.json` (~10 KB)

Step 4: Commit Reference Data

```
git add data/seed/*.json
git commit -m "Add reference data for reproducible database generation"
git push
```

Using Reference Data

For Developers (Consuming Reference Data)

Once reference data is committed to the repository, other developers can use it:

Option 1: Using Reference Data (Recommended)

```
# Activate virtual environment
source .venv/bin/activate

# Initialize database with reference data
cd adaptive_quiz_system
python backend/init_db.py --use-reference
```

Advantages: - ✅ Fast (no API calls, no shapefile processing) - ✅ Exact reproducibility (same data for everyone) - ✅ No internet required

Output:

```
Initializing adaptive hiking datasets...
Mode: Using reference data for reproducibility
Loading trails from reference JSON for reproducibility...
Loaded 100 trails from reference
rules.db initialized with adaptive ruleset
Loading completed trails from reference JSON for reproducibility...
Calculating user profiles...
users.db initialized with demo hikers and contextual history
Initialization complete.
```

Option 2: Generating Fresh Data

```
python backend/init_db.py
```

Advantages: - ✅ Always up-to-date with shapefile - ✅ Tests the full pipeline

Disadvantages: - ⚠️ Slower (API calls, shapefile processing) - ⚠️ May vary slightly between runs

---

Updating Reference Data

Reference data should be updated when:

1. **New regions are added** to the trail loading configuration
2. **Trail selection criteria change** significantly
3. **User profile assignments need adjustment**
4. **Bug fixes** affect data generation

Update Process

```
# 1. Make code changes (if needed)

# 2. Regenerate reference data
python data/seed/export_trails_reference.py
python data/seed/export_completed_trails_reference.py

# 3. Test that reference mode works
python backend/init_db.py --use-reference

# 4. Verify profiles are correct
python check_user_profiles.py

# 5. Commit updated reference files
git add data/seed/*.json
git commit -m "Update reference data: [describe changes]"
git push
```

When to Update

Update immediately if: - Trail count changes (e.g., limit increased from 100 to 150) - New regions added (e.g., adding "brittany" region) - User assignment logic changes

May not need update if: - Only scoring weights change (profiles recalculated anyway) - UI changes only - Documentation updates

---

Troubleshooting

Problem: "trails_reference.json not found"

Solution:

```
# Option 1: Generate reference data
python data/seed/export_trails_reference.py

# Option 2: Use fresh mode instead
python backend/init_db.py  # Without --use-reference
```

Problem: "No trails loaded"

Check: 1. Shapefile exists: `ls data/source/*.shp` 2. Shapefile is complete (all 4 files: .shp, .dbf, .shx, .prj) 3. Run download script if missing: `python download_trails_shapefile.py`

Problem: "API errors during export"

Causes: - Internet connection issues - Open Elevation API temporarily unavailable - Rate limiting

Solutions:

```
# Retry the export
python data/seed/export_trails_reference.py

# Or use existing reference if available
python backend/init_db.py --use-reference
```

Problem: "Different profiles detected"

Check: 1. Using `--use-reference` flag? 2. Reference JSON files are up to date? 3. Same code version as when reference was generated?

Solution:

```
# Regenerate reference data
python data/seed/export_trails_reference.py
python data/seed/export_completed_trails_reference.py

# Re-initialize
python backend/init_db.py --use-reference
```

Problem: "Git conflicts on .json files"

Cause: Multiple developers updated reference data simultaneously

Solution:

```
# Pull latest changes
git pull

# If conflicts, regenerate reference data
python data/seed/export_trails_reference.py
python data/seed/export_completed_trails_reference.py

# Resolve conflicts and commit
git add data/seed/*.json
git commit -m "Resolve reference data conflicts"
```

Best Practices

For Maintainers

1. **Update reference data after significant changes**
2. When trail loading logic changes
3. When user assignment logic changes
4. When new regions are added
5. **Test before committing** ```bash # Test reference mode python backend/init_db.py --use-reference python check_user_profiles.py

# Verify all profiles are detected ```

1. **Document changes in commit message** `bash git commit -m "Update reference data: Added Pyrenees region, increased limit to 120"`
2. **Keep reference files small**
3. Only include necessary fields
4. Compress if files get too large (>1 MB)

For Developers

1. **Use reference mode for consistency** `bash python backend/init_db.py --use-reference`
2. **Use fresh mode for testing** `bash python backend/init_db.py # Test full pipeline`
3. **Don't commit generated .db files**
4. They're in `.gitignore` for good reason
5. Each developer generates their own
6. **Pull reference data updates regularly** `bash git pull # Get latest reference JSON files python backend/init_db.py --use-reference # Regenerate with new data`

File Structure

```
adaptive_quiz_system/
├── data/
│   ├── seed/                          # Reference data directory
│   │   ├── trails_reference.json      # Trail data (versioned in Git)
│   │   ├── completed_trails_reference.json  # User assignments (versioned)
│   │   ├── export_trails_reference.py      # Export script
│   │   ├── export_completed_trails_reference.py  # Export script
│   │   ├── README.md                  # Quick reference
│   │   └── DATA_GENERATION_GUIDE.md   # This file
│   └── source/                        # Shapefile (not versioned)
│       ├── hiking_foot_routes_lineLine.shp
│       ├── hiking_foot_routes_lineLine.dbf
│       ├── hiking_foot_routes_lineLine.shx
│       └── hiking_foot_routes_lineLine.prj
├── backend/
│   ├── init_db.py                     # Main initialization script
│   └── ...
└── ...
```

Summary

Quick Reference

Generate reference data (maintainer):

```
python data/seed/export_trails_reference.py
python data/seed/export_completed_trails_reference.py
git add data/seed/*.json && git commit -m "Update reference data"
```

Use reference data (developer):

```
python backend/init_db.py --use-reference
```

Generate fresh data (developer):

```
python backend/init_db.py
```

Questions?

If you encounter issues or have questions about data generation:

1. Check this guide's troubleshooting section
2. Review `data/seed/README.md` for quick reference
3. Check Git history for recent reference data updates
4. Contact the maintainer who last updated the reference data