CSIS 4490 - CityScope (Sahan Nonis 300389470)

Progress Report Number: 2

Date	Hours	Description of Work Done
Sep 20, 2025	2	Initial brainstorming of project ideas, reviewed proposal requirements, and finalized CityScope as the main project concept.
Sep 23, 2025	3	Drafted project proposal document with sections on introduction, methodology, and timeline. Added references.
Sep 27, 2025	2	Created project GitHub repository and organized folder structure (Implementation, Documents, etc.).
Sep 29, 2025	3	Set up NestJS API service inside the monorepo (`apps/api`). Installed Prisma, TypeScript, and supporting dependencies. Initialized Prisma schema with SQLite for simplicity.
Oct 1, 2025	2.5	Designed database schema (Neighborhood, MetricSnapshot) in Prisma. Created .env for database configuration and tested connection.
Oct 3, 2025	3	Created CSV datasets for neighborhoods, rental listings, transit points, and shopping malls. Prepared initial ETL (Extract-Transform-Load) pipeline.
Oct 5, 2025	3	Wrote `prisma/seed.ts` to parse CSV data, compute aggregated metrics (average rent, transit count, mall count), normalize values, and generate a composite livability score.
Oct 7, 2025	2.5	Debugged Prisma authentication and environment setup issues. Reconfigured `.env` for SQLite and re-generated Prisma client. Successfully seeded database with computed metrics.
Oct 9, 2025	3	Implemented API endpoints (`/neighborhoods`, `/neighborhoods/:id/summary`, `/compare`) in NestJS controller. Verified JSON responses in browser and ensured data flows correctly.
Oct 10, 2025	3	Final validation of MVP backend: tested full ETL → database → API cycle. Prepared Git commit and documented Phase 1 completion.

Description of Work Done

Since the last report, the focus has been on implementing the **backend of the CityScope MVP**, enabling the application to take raw data (listings, transit, malls, and neighborhoods) and transform it into **usable, comparable metrics** for end-users.

1. **API Service Setup**
    - Created a new **NestJS application** under `apps/api` within the monorepo.
    - Installed and configured dependencies including **Prisma ORM, TypeScript, ts-node, and CSV parsing libraries**.
    - Established project structure for future scalability (controllers, services, and Prisma integration).

2. **Database Schema & Configuration**
    - Designed the schema with two core models:
        - **Neighborhood**: stores static metadata such as name, city, coordinates.
        - **MetricSnapshot**: captures calculated metrics (average rent, transit count, mall count, livability score) tied to a neighborhood.
    - Initially attempted PostgreSQL via Docker, but due to environment issues, switched to **SQLite** for a lighter, portable MVP database.
    - Configured `.env` file for Prisma connection (`DATABASE_URL="file:./dev.db"`).

3. **Dataset Creation**
    - Populated four CSV files inside `data/`:
        - `neighborhoods.csv` (IDs, names, coordinates)
        - `listings.csv` (rental prices)
        - `transit.csv` (bus/rail stops per neighborhood)
        - `malls.csv` (shopping centers per neighborhood)
    - This dataset acts as a mock of real-world open data, simulating what the final product will ingest.

4. **ETL & Seed Script Development**
    - Implemented `prisma/seed.ts` to:
        - **Extract**: read CSV files with `csv-parse`.
        - **Transform**: compute average rent, count transit and malls, normalize values (min-max scaling), and invert affordability (lower rent = higher score).
        - **Load**: upsert neighborhoods and insert computed metric snapshots into the database.

Defined livability scoring formula:

```
Score = 0.55 * Affordability + 0.35 * Transit + 0.10 * Amenities
```

- Successfully seeded three neighborhoods with metrics.

5. **API Development**
    - Built and tested three REST endpoints:
        - `/neighborhoods`: returns all neighborhoods with latest metric snapshot.
        - `/neighborhoods/:id/summary`: returns summary metrics for one neighborhood.
        - `/compare?ids=1,2,3`: returns side-by-side comparison for multiple neighborhoods.
    - Verified endpoints via browser and confirmed JSON structure.

6. **Debugging & Fixes**
    - Resolved issues including:
        - Prisma schema misconfiguration (moved generator out of datasource).
        - Database authentication errors (caused by leftover PostgreSQL configs).
        - Docker permission issues on Windows (bypassed by using SQLite).
    - Final system now runs **without Docker**, making it portable and reliable.

7. **Outcome of Phase 1**
    - The backend is fully functional: raw CSV data → ETL pipeline → SQLite DB → API endpoints.
    - This sets the foundation for Phase 2 (frontend integration with map + charts).

Repo Check-In of Implementation Completed

New files/folders checked into `Implementation/`:

- `apps/api/`
    - `src/`
        - `app.controller.ts` – contains `/neighborhoods`, `/summary`, `/compare` endpoints.
        - `app.module.ts` – updated to register controller.
    - `prisma/`
        - `schema.prisma` – defines Neighborhood + MetricSnapshot models.
        - `seed.ts` – ETL + seeding script.
    - `package.json` – with added seed script.
    - `.env` – configured for SQLite database.

- `data/`
    - `neighborhoods.csv`
    - `listings.csv`
    - `transit.csv`

    - `malls.csv`

- `node_modules/` updates from new dependencies.
- `dev.db` – generated SQLite database.