CSIS 4490 - CityScope (Sahan Nonis 300389470)

Progress Report Number: 5

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.

Oct 12, 2025	3	Initialized Next.js frontend app under `apps/web` inside monorepo. Configured TailwindCSS, TypeScript, and React libraries. Verified connection to backend API endpoints.
Oct 14, 2025	3	Built initial UI with neighborhood table and comparison chart. Integrated `/neighborhoods` API to dynamically fetch and display metrics (average rent, transit count, mall count).
Oct 16, 2025	3	Implemented Mapbox GL map visualization to show neighborhood boundaries and markers. Displayed data from backend on the map.
Oct 18, 2025	2.5	Added support for transit stops and shopping mall markers (loaded from CSV → database → API). Debugged display alignment between geographic coordinates and Mapbox rendering.
Oct 20, 2025	3	Integrated Overpass (OSM) JSON export for malls and TransLink GTFS stops.txt for transit data. Converted formats into CSV for ingestion and linked to Prisma ETL seed script.
Oct 22, 2025	3	Enhanced charts to reflect live metrics from database. Created counter widgets for “Transit Stops” and “Malls” per neighborhood, ensuring values match CSV imports.
Oct 24, 2025	2.5	Completed Phase 2 deliverables: API + frontend integration, working map with neighborhood overlays, dynamic analytics.
Oct 26, 2025	2.5	Began transition from monorepo architecture to a simpler Python-based analytics setup. Removed NestJS and Next.js dependencies to focus on data processing and visualization.
Oct 28, 2025	3	Set up new project folder structure (`data/`, `scripts/`, `frontend/`). Installed Python dependencies (osmnx, geopandas, folium, streamlit).
Oct 30, 2025	3	Imported CMHC 2024 BC rental dataset and explored structure. Cleaned and prepared columns for analysis (city, rent, property type, bedroom count).
Nov 1, 2025	2.5	Integrated OpenStreetMap (OSM) API using osmnx to extract amenities (transit, schools, malls). Defined scoring logic based on amenity density per km².
Nov 3, 2025	3	Implemented Python scripts for automated data processing (`01_build_osm_data.py`, `02_process_cmhc_data.py`). Debugged geometry NaN errors in OSM data.

Nov 5, 2025	3	Configured GitHub repository for the new version of CityScope. Committed refactored structure and initial working code. Verified environment setup.
Nov 6, 2025	2.5	Built Streamlit dashboard layout with interactive map and filtering controls for amenities. Connected CMHC and OSM data for preliminary visualizations.
Nov 8, 2025	2	Finalized scoring algorithm combining CMHC rent metrics and amenity density. Tested comparison feature between BC neighbourhoods (e.g., Downtown vs Kitsilano).
Nov 9, 2025	3	Rebuilt project from scratch using Streamlit after evaluating the limitations of the earlier Python analytics-only interface. Defined a new architecture centered on interactive mapping and scoring.
Nov 11, 2025	2	Created complete new data model: neighbourhood CSV, rent CSV, amenities CSV, OSM POI CSV. Cleaned and normalized coordinates, population, amenity counts, and rent values.
Nov 12, 2025	2.5	Implemented Streamlit layout with a sidebar filtering system. Added sliders for rent, amenities, transit, population, and score thresholds.
Nov 14, 2025	2	Added neighbourhood scoring algorithm using weighted components (rent_score, transit_score, amenities_score, size_score). Tuned weight interactions to produce meaningful ranking behaviour.
Nov 15, 2025	2	Integrated PyDeck map with neighbourhood circles colored by score. Added radius scaling based on population and hover tooltip logic. Fixed multiple rendering and file path errors.
Nov 16, 2025	3	Added OpenStreetMap (OSM) amenities display (schools, restaurants, transit stops, parks, grocery). Implemented separate tooltips for POI vs neighbourhoods. Troubleshot missing files, formatting errors, and hover issues.
Nov 17, 2025	2	Implemented the neighbourhood comparison module. Added bar charts, raw metrics table, and side-by-side comparisons of selected neighbourhoods.
Nov 18, 2025	2.5	Added dynamic scoring engine updates based on filters. Ensured all metrics (amenities, rent, population, transit) re-compute in real-time. Added global and per-city filters.

Nov 19, 2025	2.5	Built the “AI Assistant” tab. Implemented NLP-style keyword interpretation (budget, student, transit, quiet, parks, restaurants, family). Created recommendation scoring (ai_score) and explanation generator.
Nov 20, 2025	2	Conducted extensive debugging and UI refinement: fixed missing tooltip values, handled empty POI categories, polished map styling, improved consistency between tabs.
Nov 21, 2025	2.5	Added final enhancements: custom radius scaling, color interpolation logic for score visualization, improved sidebar filters, and polished layout/headers.

Description of Work Done

During this reporting period, CityScope evolved into a complete **interactive neighbourhood intelligence platform** built entirely with Streamlit and Python. This phase focused on integrating OSM data, creating a dynamic scoring engine, building the comparison tools, and introducing an AI-driven recommendation assistant.

1. Complete Rebuild in Streamlit

Following the transition documented in Report 4, the project was fully reimplemented in Streamlit for rapid prototyping, smooth UI flow, and built-in interactivity. The entire architecture was redesigned to eliminate backend complexity and instead focus on analytics, visualization, and user experience.

2. Data Model Finalization

Custom CSV datasets were produced and cleaned for:

- Neighbourhoods (ID, name, city, population, coordinates)
- CMHC-style average rent
- OSM amenity counts (schools, restaurants, transit stops, parks, grocery)
- Raw OSM POIs with lat/lon for map visualization

All datasets were standardized and merged during load-time using Streamlit caching.

3. Scoring Engine Implementation

A **weighted neighbourhood scoring model** was created using:

- **rent_score** → lower rent = higher score
- **transit_score** → based on transit_stops count
- **amenities_score** → normalized counts of schools, restaurants, parks, grocery
- **size_score** → derived from population

Users can dynamically adjust weights, triggering instant recalculation.

4. Interactive Map with PyDeck

A multi-layer interactive map was built featuring:

- Neighbourhood circles colored by score
- Circle radius scaled by neighbourhood population
- Full OSM amenity layers with toggles
- Dual tooltip system
    - neighbourhoods → score & rent
    - amenities → type & name

This required significant debugging around tooltips, scaling, and OSM alignment.

5. Comparison Engine

Implemented an analytics section where users can:

- Select 2–5 neighbourhoods
- View raw metrics
- Compare through bar chart visualizations
- Analyze differences in rent, transit, amenities, size, and score

6. AI Assistant Tab

A second tab was created that acts like a lightweight LLM-powered advisor:

- User inputs free-text needs
- Text is parsed using keyword-based NLP
- System generates **ai_score** to re-rank neighbourhoods
- Provides narrative explanation of reasoning
- Displays top matches

This acted as the "AI layer" of the platform without external APIs.

7. UI, Debugging & Stability Improvements

Throughout Nov 15–22, multiple refinements were made:

- Tooltip logic fixes
- Error handling for missing files and empty categories
- More intuitive filters
- Visual consistency across the UI
- Better header structure and user instructions

Technical Stack

Language & Framework

- Python 3.12
- Streamlit (primary application framework)
- PyDeck (map visualization)
- Pandas (data processing)
- NumPy (scoring calculations)

Geospatial Data

- OSM POIs via Overpass API
- Custom amenity datasets
- CMHC-style rental data

Visualization

- Streamlit charts
- PyDeck multi-layer map
- Interactive comparison charts

Version Control

- GitHub repository active and updated

Features Implemented

Complete Interactive Map

- Neighbourhood scoring circles
- OSM amenities
- Hover-based tooltips
- Dynamic filtering

Data-driven Scoring Model

- Weight-adjustable
- Normalization applied to all categories
- Composite final score

Neighbourhood Comparison Engine

- Raw metrics
- Bar chart visuals
- Multi-selection

AI Assistant

- Keyword-based intent detection
- Personalized neighbourhood ranking
- Narrative explanation

Robust Dataset Handling

- Custom cleaned CSVs
- Automatic merging & normalization
- Cache optimized loading

Challenges Encountered

1. Missing/Empty Files

    - The system initially crashed due to missing or empty CSVs (e.g., `poi_counts.csv`). This required implementing checks, default fallbacks, and consistent file creation.

2. PyDeck Tooltip Limitations

    - PyDeck only allows **one tooltip template** across all layers. A workaround was created using a shared `tooltip_html` field, enabling differentiated tooltip content.

3. Data Integrity Issues

    - OSM extraction produced duplicate or unnamed POIs; name cleaning and category normalization were implemented.

4. Real-time Rank Recalculation

    - Weight sliders required live recalculation of all metrics. Efficient caching and selective recomputation were essential.

5. AI Assistant Logic Tuning

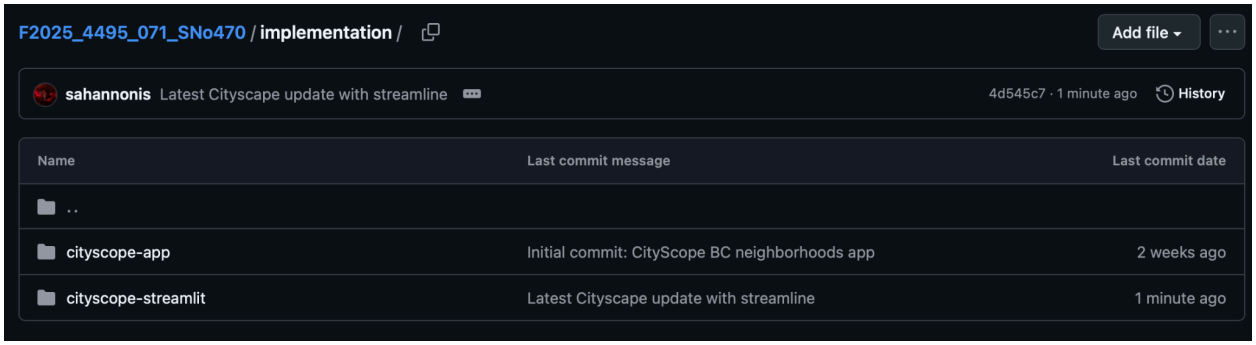
    - Balancing narrative explanation, intent detection, and ranking required iterative adjustments for natural output.

Repo Check-In of Implementation Completed

Since Report 4, the following major additions were committed:

- Full Streamlit-based front-end
- Map + filters + scoring engine
- OSM amenities integration
- CSV data processing scripts

- AI assistant logic
- Comparison analytics module
- Complete debugging & UI polish



Created the project brandnew using streamlight