# Colorado Workforce Intelligence Platform

Comprehensive technical and user documentation for the GW Hackathon NLx project.

---

## 1) Project Summary

Colorado Workforce Intelligence is a Streamlit-based decision-support app built on National Labor Exchange (NLx) Colorado job posting data. It helps three audiences:

- Job seekers find roles aligned with free-text skills.
- Students / career changers discover in-demand skills and field-specific opportunities.
- Veterans translate MOS/MOC military experience into civilian job pathways.

The solution uses NLP and vector similarity (TF-IDF + cosine similarity) to transform job/skill text into searchable, rankable recommendations.

---

## 2) Core Capabilities

### Job Seeker

- Free-text skill matching against inferred job skill profiles.
- City filtering.
- Match score ranking.
- Education and experience display with source transparency (dataset vs inferred).
- Skill-gap view (matched vs potential missing skills).
- Optional external job posting link when available.

### Student / Career Changer

- Top-20 in-demand skill analysis.
- Demand tiering (High / Medium / Low) for quick scanning.
- Field Explorer with curated field prompts.
- Field-specific skills and sample jobs.
- View toggle for Table View vs Card View in field insights.

### Veteran Translator

- Direct MOC code matching against employer-tagged `moc_codes`.
- Skill-based fallback/recommendation using expanded MOC dictionary.
- Dual output: direct tagged matches + skill-based matches.

**Usage Insights**

- Persistent analytics (cross-session) stored in SQLite and mirrored to CSV.
- Visit, search, and recommendation counters and trend charts.

---

## 3) Technical Architecture

###[Application Layer] - Streamlit UI and interaction flows: hackathon/app.py

###[Domain / Core Layer] - Data preparation and artifact lifecycle: hackathon/core/data.py - NLP feature extraction and requirement inference: hackathon/core/nlp_pipeline.py - Matching and skill-gap logic: hackathon/core/matching.py - Student skill trend helpers: hackathon/core/student.py - Veteran mapping and matching: hackathon/core/veterans.py - Persistent analytics logger (SQLite + CSV): hackathon/core/analytics_logger.py

###[Execution Layer] - Prepare data artifacts: hackathon/scripts/prepare_data.py - Run all local steps: hackathon/scripts/run_all.py - Run Streamlit locally: hackathon/scripts/run_local.py - Optional ngrok run: hackathon/scripts/run_colab.py

---

## 4) Data Pipeline and Artifact Flow

1. Raw data availability
   - Expects zipped NLx files under `data/Colorado-Hackathon-Dataset-selected/`.
2. Raw extraction
   - `prepare_raw_data()` extracts to:
     - `data/raw/colorado.csv`
     - `data/raw/colorado_processed.csv`
3. Job cleaning and schema enforcement
   - `REQUIRED_JOB_COLUMNS` are enforced in hackathon/core/data.py.
4. Requirements preprocessing (education/experience)
   - Inferred with regex and source tagging in hackathon/core/nlp_pipeline.py.
   - Persisted to `data/processed/nlp_requirements_profile.csv`.
5. NLP skill mention extraction
   - Job text and taxonomy skills are vectorized and matched.
   - Mentions and profiles are persisted to:
     - `data/processed/nlp_skill_mentions.csv`
     - `data/processed/nlp_skill_profiles.csv`
6. Runtime consumption
   - App loads cleaned jobs + skill profiles + mention structure.
7. Analytics persistence
   - App logs usage events to:

    - `data/processed/analytics/usage_analytics.db`
    - `data/processed/analytics/usage_analytics_events.csv`

---

## 5) NLP and Matching Approach

### Skill Profile Construction

- Skill mentions are extracted from job text using a TF-IDF-based relevance process.
- Mentions are deduplicated and aggregated by `system_job_id` into a `skill_text` profile.

### Job Matching

- User input is transformed by the same TF-IDF vectorizer.
- Cosine similarity scores compare user vectors vs job skill profile vectors.
- Top-N results are ranked by `match_score`.

### Skill Gap Analysis

- For each recommended job, top weighted/informative skills are compared to user text.
- Output groups skills into:
    - matched
    - potential gaps

### Requirements Inference

- Education and experience are either:
    - taken from source dataset if present, or
    - inferred from title/description text by regex patterns.
- Source labels: `dataset`, `nlp_inferred`, `not_specified`.

---

## 6) User Interface Overview

### Global UI

- Dark, data-themed background with layered gradients and subtle grid motif.
- KPI strip for jobs indexed, cities, skill profiles, and employers.

### Tab 1 — Job Seeker

- Input text + city filter.

- Ranked recommendations with score, salary, requirements, skill gap, and posting link.

**Tab 2 — Student / Career Changer**

- Top-20 demand dashboard:
  - chart + tiered table + top-skill snapshot cards.
- Career Field Explorer with field descriptions.
- Field results in Table/Card modes.

**Tab 3 — Veteran**

- MOS/MOC input.
- Direct tagged matches and skill-based matches.
- Expanded MOC dictionary for richer civilian translation cues.

**Tab 4 — Usage Insights**

- Visit/search/recommendation totals.
- Visit-by-hour chart.
- Search volume by workflow and over time.
- Top recommended roles/cities.

---

## 7) Persistent Analytics Specification

Analytics table schema (`analytics_events`) includes:

- `timestamp`
- `event_type` (`visit`, `search`, `recommendation`)
- `channel`
- `city_filter`
- `field`
- `moc`
- `title`
- `city`
- `results_count`
- `direct_count`
- `skill_count`
- `match_score`

Storage behavior: - SQLite is the primary source for analytics visualizations. - CSV is an append mirror for easy export/review.

---

## 8) Setup and Execution

### Prerequisites

- Python 3.10+
- Dependencies from requirements.txt

### Install

```
pip install -r requirements.txt
```

### Data Preparation

```
python -m hackathon.scripts.prepare_data
```

### Run App

```
python -m hackathon.scripts.run_local
```

or

```
python -m streamlit run hackathon/app.py
```

### One-command local sequence

```
python -m hackathon.scripts.run_all
```

### Optional public tunnel

```
python -m hackathon.scripts.run_colab
```

---

## 9) Directory Map (Key Paths)

- App entrypoint: hackathon/app.py
- Data pipeline: hackathon/core/data.py
- NLP extraction and requirement inference: hackathon/core/nlp_pipeline.py
- Matching logic: hackathon/core/matching.py
- Student logic: hackathon/core/student.py
- Veteran logic: hackathon/core/veterans.py
- Analytics logger: hackathon/core/analytics_logger.py
- Prepare script: hackathon/scripts/prepare_data.py

---

## 10) Data Contracts (Important Columns)

**Jobs (`jobs_clean`)**

- `system_job_id`, `title`, `description`, `city`, `zipcode`
- `parameters_salary_min`, `parameters_salary_max`
- `requirements_min_education`, `requirements_experience`
- `classifications_onet_code`, `moc_codes`, `cip_codes`
- `application_company`, `link`, `created_date`, `classifications_naics_code`
- plus merged requirement outputs:
  - `education_display`, `education_source`
  - `experience_display`, `experience_source`

**Skill profiles (`skill_profiles`)**

- `system_job_id`, `skill_text`

**Skill mentions (`structured` / mention dataframe)**

- `Research ID`, `Taxonomy Skill`, `NLP Score`

---

## 11) Known Limitations

- Job postings are demand signals, not hiring outcomes.
- Source coverage may vary by employer and region.
- MOC translation is heuristic and approximate.
- Salary and requirement completeness depends on source records.
- Regex-based requirement inference may miss edge-case phrasing.

---

## 12) Suggested Next Improvements

- Add confidence intervals/explanations for recommendation quality.
- Improve skill-gap matching with lemmatization and synonym expansion.
- Add date-window filtering using `created_date`.
- Add NAICS and O*NET drilldowns in UI filters.
- Add optional export button for analytics CSV directly in UI.

---

## 13) Quick Troubleshooting

**App does not start**

- Confirm environment and dependencies installed.
- Run `python -m streamlit run hackathon/app.py` from repository root.

### Missing data files

- Ensure zipped source files exist in `data/Colorado-Hackathon-Dataset-selected/`.
- Re-run `python -m hackathon.scripts.prepare_data`.

### Empty recommendations

- Check whether skill profiles were generated in `data/processed/`.
- Try broader input text terms.

### Analytics charts blank

- Perform at least one search action in any tab.
- Confirm files exist under `data/processed/analytics/`.

---

## 14) License and Ownership

Refer to LICENSE for licensing terms and repository ownership metadata.