SHL Assessment Recommendation System

A system that recommends the most relevant SHL assessments based on job requirements, skills needed, and other criteria.

Features

- **Web Scraping**: Extracts assessment data from SHL's product catalog
- **Semantic Search**: Uses embeddings to find the most relevant assessments for a query
- **Constraint-Based Filtering**: Filters results based on time limits, skills, and test types
- **API Backend**: FastAPI service for serving recommendations
- **User Interface**: Streamlit frontend for user interaction
- **Evaluation**: Metrics to evaluate recommendation quality

Architecture

The system consists of the following components:

1. **Data Collection**:
    - Web scraper to extract assessment data
    - Data storage in JSON format
2. **Recommendation Engine**:
    - Embedding generation using sentence-transformers
    - Vector database (ChromaDB) for semantic search
    - Query processor for constraint extraction and filtering
3. **API and Interface**:
    - FastAPI backend with endpoints for recommendations
    - Streamlit frontend for user interaction
    - Docker containerization for deployment

Installation

Requirements

- Python 3.9+
- Libraries listed in requirements.txt

Setup

1. Clone the repository:

```bash
git clone https:// https://github.com/Vikrant1507/SHL-Assignment.git
cd shl-recommendation-system
```

2. Install dependencies:

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

3. Run the system:

```bash
python main.py
```

Usage

Command Line Interface

Run the interactive CLI:

```bash
python cli.py
```

Available commands:

- `exit` - Exit the program
- `help` - Show help message
- `eval` - Run evaluation metrics
- `list` - List all available assessments

API Endpoints

- GET /health - Health check
- POST /recommend - Get assessment recommendation

Example request:

```json
{
  "query": "Looking for a Java coding assessment that takes less than 45 minutes",
  "url": "any URL" (optional)
}
```

Web Interface

Run the Streamlit interface:

```bash
streamlit run app.py
```

Evaluation

The system includes evaluation metrics:

- Recall@K
- Precision@K
- NDCG@K

Run evaluation:

```bash
python cli.py --eval
```

Future Improvements

- Advanced NLP techniques for job description parsing
- User feedback integration for recommendation improvement
- Additional filtering criteria (industry, job level, etc.)
- Expanded assessment database with more detailed attributes
- Integration with ATS (Applicant Tracking Systems)