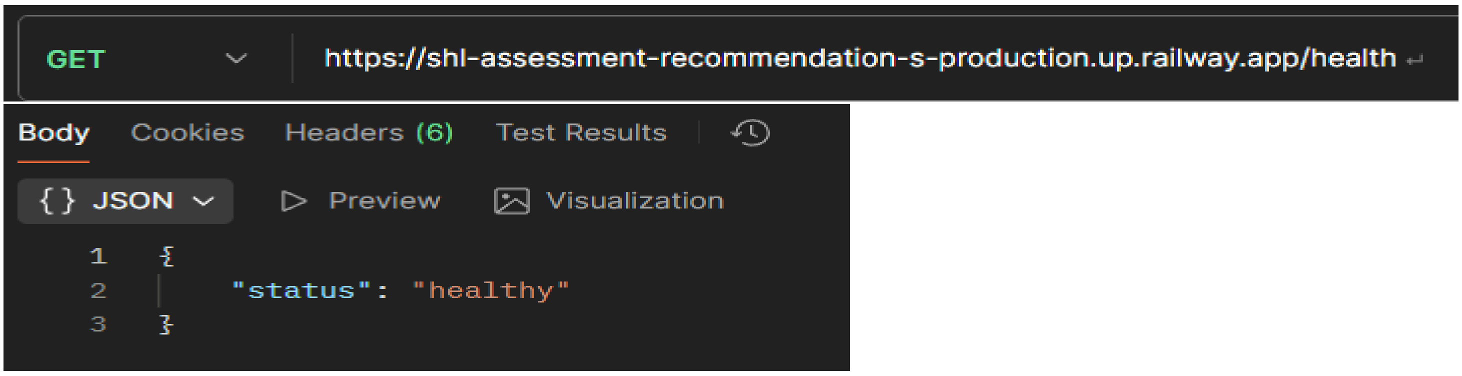
SHL Assessment Recommendation System - Final Project Documentation

Overview:
This project provides a web-based intelligent system that takes a natural language job description or query and recommends the most relevant SHL individual assessments. It aims to simplify the hiring process by aligning job requirements with SHL's assessment catalog using semantic similarity.

Tech Stack:
- **Frontend**: React.js (deployed on Vercel)
- **Backend**: FastAPI (deployed on Railway)
- **ML Model**: Sentence Transformers (for embedding and semantic similarity)
- **Data**: Manually curated SHL assessment metadata (with descriptions, duration, type, etc.)

Key Features:
- Semantic search for SHL tests using job descriptions
- Up to 10 relevant assessments returned
- Health check endpoint for service availability
- Fully compliant API structure as per assignment specs

API Endpoints:
1. /health



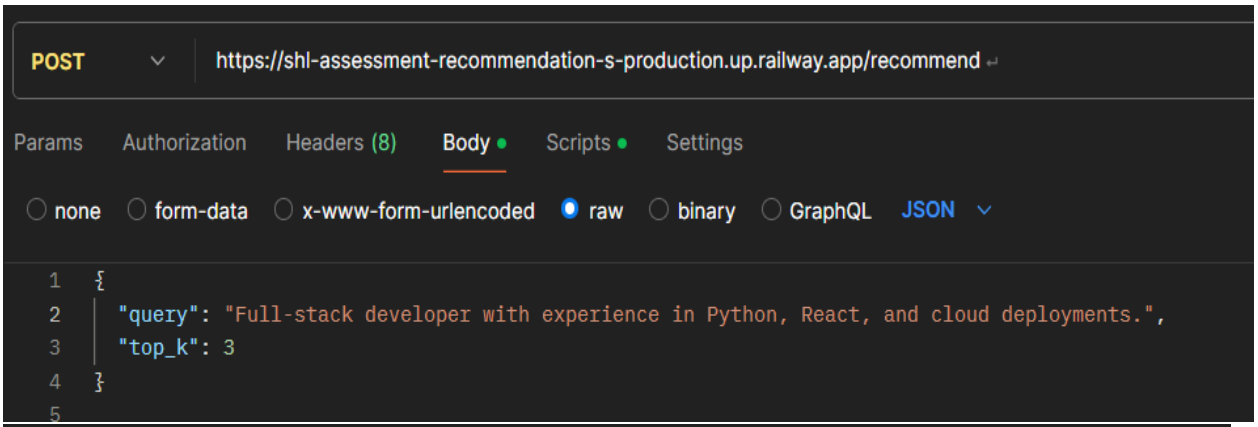
```
GET  https://shl-assessment-recommendation-s-production.up.railway.app/health
```

Body Cookies Headers (6) Test Results

{} JSON Preview Visualization

```
{
    "status": "healthy"
}
```

2. /recommend



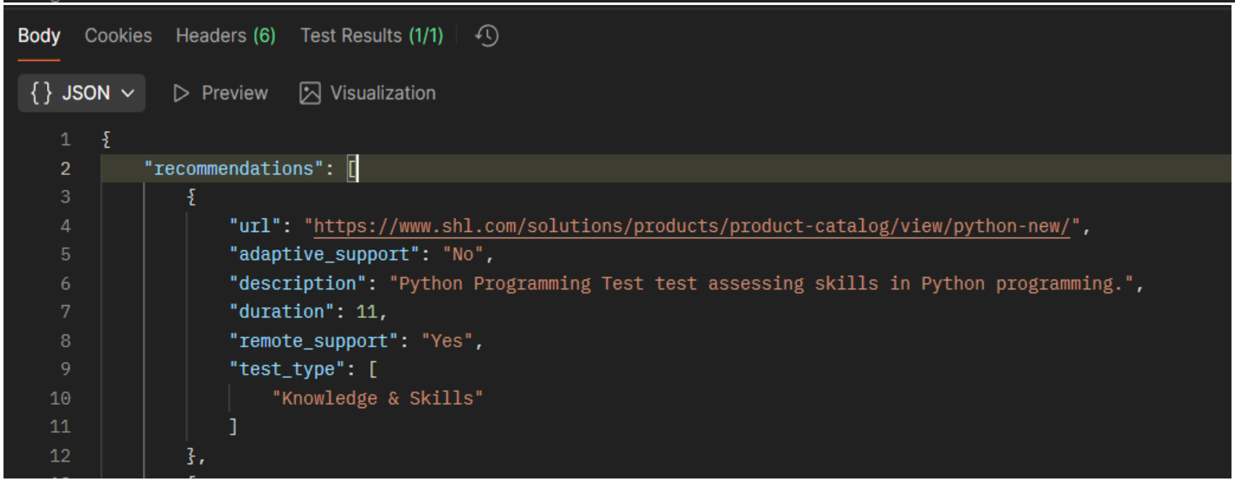
```
POST  https://shl-assessment-recommendation-s-production.up.railway.app/recommend
```

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
  "query": "Full-stack developer with experience in Python, React, and cloud deployments.",
  "top_k": 3
}

```



Body Cookies Headers (6) Test Results (1/1)

{} JSON Preview Visualization

```
{
    "recommendations": [
        {
            "url": "https://www.shl.com/solutions/products/product-catalog/view/python-new/",
            "adaptive_support": "No",
            "description": "Python Programming Test test assessing skills in Python programming.",
            "duration": 11,
            "remote_support": "Yes",
            "test_type": [
                "Knowledge & Skills"
            ]
        },
```

Getting Started

1. **Backend Setup (FastAPI)**
   - cd backend
   - python -m venv env
   - source env/bin/activate # On Windows: env\Scripts\activate
   - pip install -r requirements.txt
   - uvicorn main:app --reload

   Make sure the backend is running at http://127.0.0.1:8000.
2. **Frontend Setup (React)**
   - cd frontend
   - npm install
   - npm run dev

   Access the app at http://localhost:5173

System Architecture:

1. **User Input**: Via a React.js form on the frontend
2. **Request Sent**: POST to Railway-deployed FastAPI backend
3. **Processing**: Input embedded using Sentence Transformers
4. **Matching**: Similarity scored with SHL assessment descriptions
5. **Response**: Top 1-10 assessments returned as JSON
6. **Frontend Display**: UI showing recommendations with links and metadata

---

Deployment:

- **Frontend (React.js)**: Hosted on Vercel
- **Backend (FastAPI)**: Hosted on Railway

Challenges Faced:

- Ensuring **semantic relevance** with various job inputs
- Maintaining strict **response format** for evaluation
- **Latency** during real-time embedding (solved with preloading optimizations)
- **CORS issues** during frontend-backend local integration

Status: Completed and deployed. Ready for evaluation!