

Knowledge Graph Application - Setup & Deployment Instructions

For Third-Party Users

This document provides step-by-step instructions to set up and run the Knowledge Graph web application from the provided zip file.

Table of Contents

1. [Project Overview](#)
 2. [System Requirements](#)
 3. [Folder & File Structure](#)
 4. [Installation Steps](#)
 5. [Running the Application](#)
-

Project Overview

The **Knowledge Graph Application** is a web-based tool for creating, visualizing, and querying relationship networks between entities. Users can:

- Add relationships manually (Entity1 → Relationship → Entity2)
- Bulk import data via CSV files
- Query the graph (find neighbors, shortest paths, relationship search)
- Visualize graphs interactively using D3.js
- Export graph data as JSON
- View graph statistics

Tech Stack:

- Backend: Python (Flask)
 - Frontend: HTML5, CSS3, JavaScript, D3.js
 - Graph Engine: NetworkX
 - Data Format: CSV → In-Memory Graph
-

System Requirements

Minimum Requirements

- **Python:** 3.7 or higher
- **OS:** macOS, Linux, or Windows
- **RAM:** 2GB
- **Disk Space:** 500MB
- **Browser:** Chrome, Firefox, Safari, or Edge (modern versions)

Check Your Python Version

```
python3 --version
```

Should output: **Python 3.7.x** or higher

Folder & File Structure

After extracting the zip file, you'll see:

```
Assignment/
├── knowledge_graph/                # Main application folder
│   ├── app/
│   │   ├── __init__.py            # Flask app initialization & API
│   │   └── endpoints
│   │       ├── knowledge_graph.py # Graph logic using NetworkX
│   │       ├── templates/
│   │       │   ├── index.html     # Main web interface
│   │       │   └── static/
│   │       │       ├── css/
│   │       │       └── style.css   # Styling (dark theme, Gemini
│   │       └── design)
│   │           ├── js/
│   │           └── app.js          # Frontend logic & D3.js
│   └── visualization
│       ├── data/
│       │   └── sample_ecommerce.csv # Sample dataset for demo (25+
│       └── relationships)
│           ├── docs/
│           └── ENHANCEMENT_PLAN_TASK_B.md # Documentation for scalability
│   └── improvements
│       ├── imgs/                  # Screenshots for README
│       └── documentation
│           ├── requirements.txt    # Python dependencies (Flask,
│           └── NetworkX, Pandas)
│               ├── run.sh          # One-command startup script
│               └── (Linux/macOS)
│                   ├── README.md   # Full documentation with API
│                   └── endpoints
│                       ├── verify.sh # Script to verify Python
│                       └── environment
│                           ├── Literature_Survey_RAG.md # Academic literature survey (5-
│                           └── mark submission)
│                               ├── Assignment-1_PS10.pdf # Assignment PDF (requirements &
```

```
rubric)
└─ SETUP_INSTRUCTIONS.md          # This file
```

Key Files Explained

File/Folder	Purpose	Importance
app/__init__.py	Flask app setup, all API endpoints	Critical
app/knowledge_graph.py	Graph algorithms & operations	Critical
templates/index.html	Web interface	Critical
static/css/style.css	Styling & theming	Important
static/js/app.js	Frontend interactivity & visualization	Critical
requirements.txt	Python package dependencies	Critical
data/sample_ecommerce.csv	Sample data for testing	Optional (for demo)
README.md	Full documentation	Reference

Installation Steps

Step 1: Extract the Zip File

```
# Navigate to where you downloaded the zip
cd ~/Downloads # or your download location

# Unzip the file
unzip PartA_KnowledgeGraphApp.zip

# Navigate to the project folder
cd PartA_KnowledgeGraphApp/knowledge_graph
```

Step 2: Verify Python Installation

```
bash verify.sh
```

Expected output: Python 3.x.x and confirmation that Python is properly installed.

If this fails, install Python 3.7+ from python.org

Step 3: Create a Virtual Environment

A virtual environment isolates project dependencies from your system Python.

On macOS/Linux:

```
python3 -m venv venv  
source venv/bin/activate
```

On Windows (Command Prompt):

```
python -m venv venv  
venv\Scripts\activate
```

On Windows (PowerShell):

```
python -m venv venv  
venv\Scripts\Activate.ps1
```

After activation, your terminal should show **(venv)** prefix.

Step 4: Install Dependencies

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

This installs:

- **Flask** (web framework)
- **NetworkX** (graph algorithms)
- **Pandas** (CSV data processing)
- **Werkzeug** (file handling)

Expected output: Multiple packages installing successfully. If errors occur, see [Troubleshooting](#).

Running the Application

Option A: Quick Start (Recommended for macOS/Linux)

```
bash run.sh
```

The script will:

1. Create/activate the virtual environment
2. Install dependencies
3. Start the Flask server

4. Print the URL (usually `http://localhost:8080`)
5. Automatically open the browser (if available)

Option B: Manual Start (All Platforms)

```
# Activate virtual environment (if not already active)
source venv/bin/activate # macOS/Linux
# or
venv\Scripts\activate # Windows

# Start the Flask application
python3 -m app.__init__
```

Expected output:

```
* Serving Flask app 'app'
* Debug mode: on
* Running on http://localhost:8080
```

Option C: Access the Application

1. **If browser didn't open automatically:** Open your browser and visit:

```
http://localhost:8080
```

2. **You should see:**

- Header with "Knowledge Graph Application" title
- Left sidebar with controls (Add Relationship, Upload CSV, Query options)
- Center area with empty graph visualization
- Right section showing graph statistics

3. **To stop the server:** Press `Ctrl+C` in the terminal

Troubleshooting

Issue 1: "Python 3.7+ not found"

Solution:

- Install Python 3.7+ from python.org
- Restart your terminal after installation
- Verify: `python3 --version`

Issue 2: "ModuleNotFoundError: No module named 'flask'"

Solution:

```
# Ensure virtual environment is activated
source venv/bin/activate # macOS/Linux

# Reinstall dependencies
pip install --upgrade pip
pip install -r requirements.txt
```

Issue 3: "Port 8080 already in use"**Solution:**

```
# Find process using port 8080
lsof -i :8080 # macOS/Linux
netstat -ano | findstr :8080 # Windows

# Kill the process (or use a different port in app/__init__.py)
kill -9 <PID> # macOS/Linux
```

Issue 4: "Permission denied" on run.sh**Solution:**

```
chmod +x run.sh
bash run.sh
```

Issue 5: Virtual environment activation not working**Solution:**

```
# Delete and recreate virtual environment
rm -rf venv
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Feature Quick Start

1. Add a Single Relationship

1. Fill in the form:

- **Entity 1:** Laptop
- **Relationship:** belongs_to
- **Entity 2:** Electronics

2. Click "Add Relationship"

3. Graph updates with new node and edge

2. Bulk Import Sample Data

1. Click "Upload CSV File"
2. Select `data/sample_ecommerce.csv`
3. Click "Upload"
4. Graph populates with 25+ e-commerce relationships

3. Query the Graph

Find Neighbors:

- Entity: Amazon
- Direction: both
- Shows all connected entities and relationships

Find Path:

- Source: Customer1
- Target: Laptop
- Shows shortest paths connecting these entities

Search by Relationship:

- Relationship: belongs_to
- Shows all instances of this relationship type

4. Export Graph

- Click "Export as JSON"
- Downloads `graph_export.json` with nodes, edges, and statistics

5. View Statistics

- Graph statistics tab shows:
 - Total entities count
 - Total relationships count
 - Breakdown by relationship type
-