CSU2013/CSU3013: Group 10 System Requirements Document

Designing an intuitive interface for Knowledge Graph mappings

By Daniel Grace, David Green, Sanil Gupta, Ailbhe Merriman, Matthew Dowse and Nathan Doussot

Table of Contents

1. Introduction

1.1 Overview - Purpose of System

The purpose of the system is to allow users to easily create, edit and visualise mappings from heterogeneous sources into Knowledge Graphs from imported files. The editor will be built to allow for a more user-friendly and visually appealing interface than is currently available.

The tool will extract data from files imported by the user to populate mappings for a Knowledge Graph using an existing library. The system needs to balance the technical context of the classification with visual appeal that will make the results and use intuitive without cognitive overhead for end users.

1.2 Scope

The following item(s) are in scope for the project:

- Designing an intuitive interface for Knowledge Graph mappings which allows users to be able to export their Knowledge Graph which they created as a JSON file.

1.3 Objectives and success criteria

- The user can populate a Knowledge Graph mappings from existing JSON or CSV files and Ontologies.
- Users should be able to easily describe relationships between objects in the Knowledge Graph.
- The relationships between objects are displayed in the Knowledge Graph.
- The created mapping can be exported as JSON.
- The system is web-based and responsive.
- The product is well documented with the aims of maintenance and further development.

1.4 Definitions, abbreviations

- KG - Knowledge Graph represents a collection of interlinked descriptions of entities – real-world objects and events, or abstract concepts (e.g., documents) – where:
    - Descriptions have formal semantics that allow both people and computers to process them in an efficient and unambiguous manner;
    - Entity descriptions contribute to one another, forming a network, where each entity represents part of the description of the entities, related to it, and provides context for their interpretation.
- UI - User interface. The means by which the user and a computer system interact, in particular the use of input devices and software.
- GUI - Graphical user interface. A form of user interface that allows users to interact with electronic devices through graphical icons instead of text-based user interfaces, typed command labels or text navigation.
- JSON - is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types.

- CSV - Comma Separated Variables, A comma-separated values file is a delimited text file that uses a comma to separate values.

2. Current Systems Available on the Market

There exist certain systems that provide some aspects of functionality of the project. These systems however lack a GUI that is easy to learn and visually appealing.

Current Systems are unintuitive for non-tech literate people to use and require a strong background of understanding the systems and files in which they are trying to use to create KG's.

3. Proposed System

3.1 Overview

A portable web-based GUI that

- enables users to create KG mappings by:
  - selecting objects
  - adding relationships between objects
- stores the KG mappings with the relationships in a portable way, e.g. as JSON file
- enables users to upload existing schema of various formats, i.e. JSON, CSV.
- Responsive design for different screen sizes and resolutions

The GUI will consist of 4 components:

- a header with options for the mapping
- a view of the mapping
- a view of the ontology
- a view of the uploaded JSON file

Technologies:

- The system will be developed using Vue.js 2.0 as the projects front end javascript framework
- The system will use a library called Blockly which provides the draggable and droppable jigsaw pieces for the mapping
- The system will manage its dependencies with Node Packet Manager (NPM)
- The project will be deployed via GitHub Pages so our clients can view the current progress.

3.2 Functional Requirements

The system should be able to do the following:

- Allow users to upload a JSON file
- Allow users to upload an Ontology
- Scan the Ontology for the various concept names and connections
- Crawl through the JSON file to extract from there the field names and inter-connects the obtained elements

- Show the user relevant help and tips to guide the user
- Use “blocks/jigsaw pieces” to allow user to construct mappings
- Output the data in JSON format

The JSON should contain the fields and values which the user chooses and formats themselves.

3.3 Non-functional Requirements

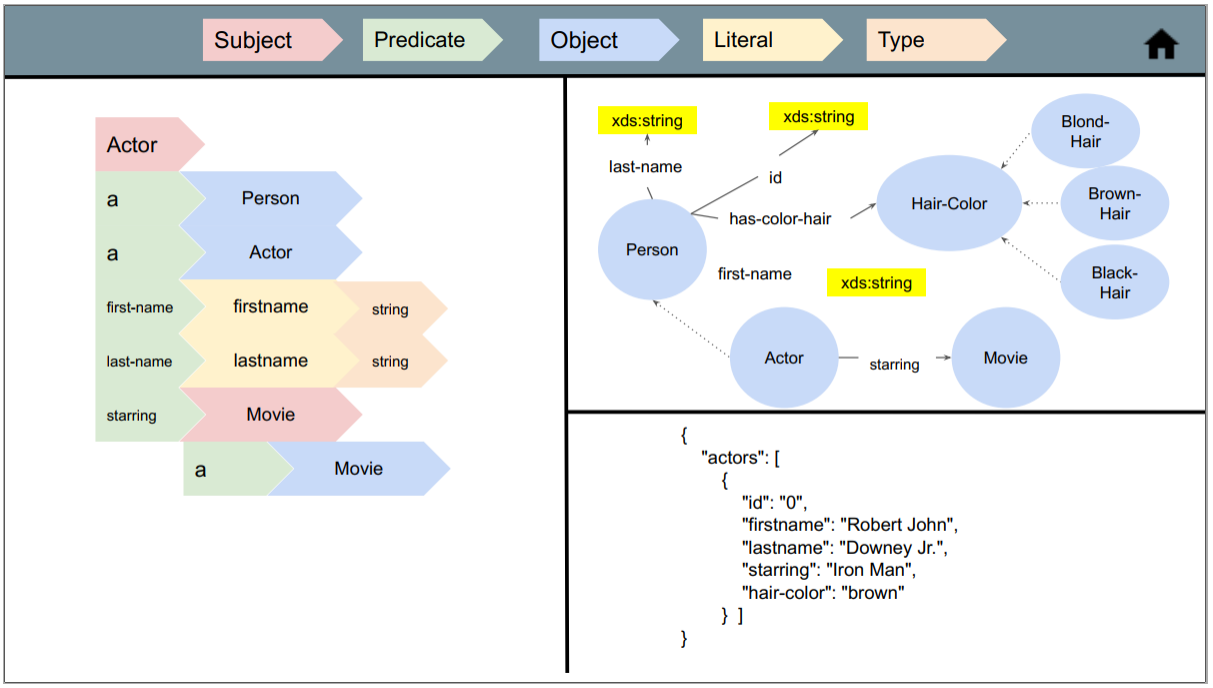
The system should be:

- Intuitive: The users should be able to easily apply their ideas. This should be aided with the use of directive tips and help within the UI
- Easy to use: Layout should be consistent and clear to prevent confusion and misuse
- Web-based: Portability is required to allow users to use on any device at any time
- Maintainability: The software must be well documented and commented to allow our clients to maintain the system after we hand it over.
- Modularity: System must be modular in design and allow component swapping in and out.
- Open Source: The software must only contain libraries that are open-source or public licenses.

3.4 System Prototype (Models)

3.4.1 User Interface Mockup

A mockup of the UI provided to us by the clients can be seen here



The elements shown in the mockup are as follows:

- An upload window to allow the user to upload a JSON schema and display the schema in a visually appealing way.
- An upload window to allow the user to upload an Ontology to view KG as they exist.
- A window that allows the user to edit the schema and relationships between objects within the existing schema.
- A window at the top of the system that contains the “jigsaw pieces” that allow the user to describe the objects and the relationships between them.

The following use cases may be seen in the prototype;

- Select the subject to define what object the relationship pertains to.
- View the JSON data.
- View the KG ontology.
- Define the relationship between multiple objects.
- Define the attributes pertaining to a specific object.

From any of these functions, the other functions may be invoked. To reset the prototype to screen 1, press the “home” icon below the prototype.

This is the initial prototype for the system. The overall design will be built on as the project progresses including the addition of design assets, colouring and theming.

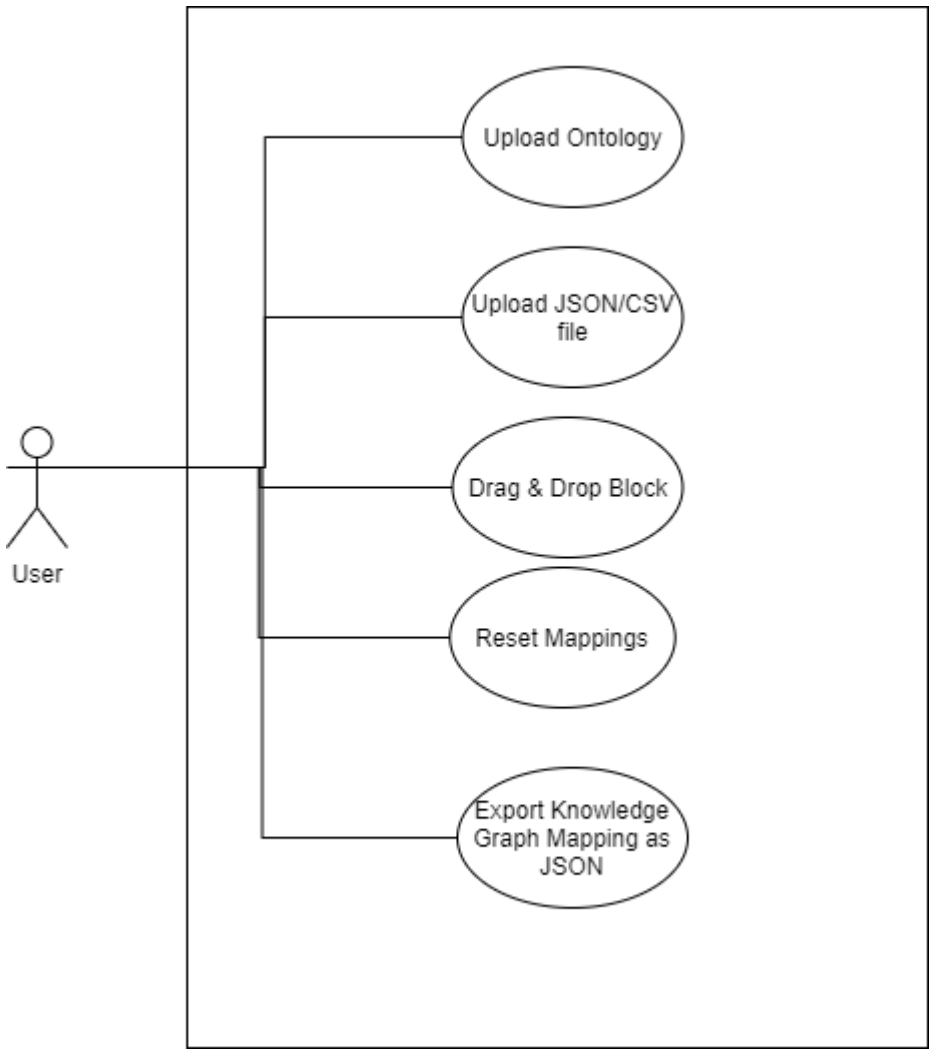
3.4.2 Use Case Diagram

There is one actor for the system we are building. This actor is the intended user of the system.

The user can upload both a JSON/CSV file as well as an KG ontology.

The user will then be able to view the different sections available to them to build their mapping. The user will then be able to drag and drop these sections denoted as “jigsaw pieces” in order to let them experience the process of building their own mapping.

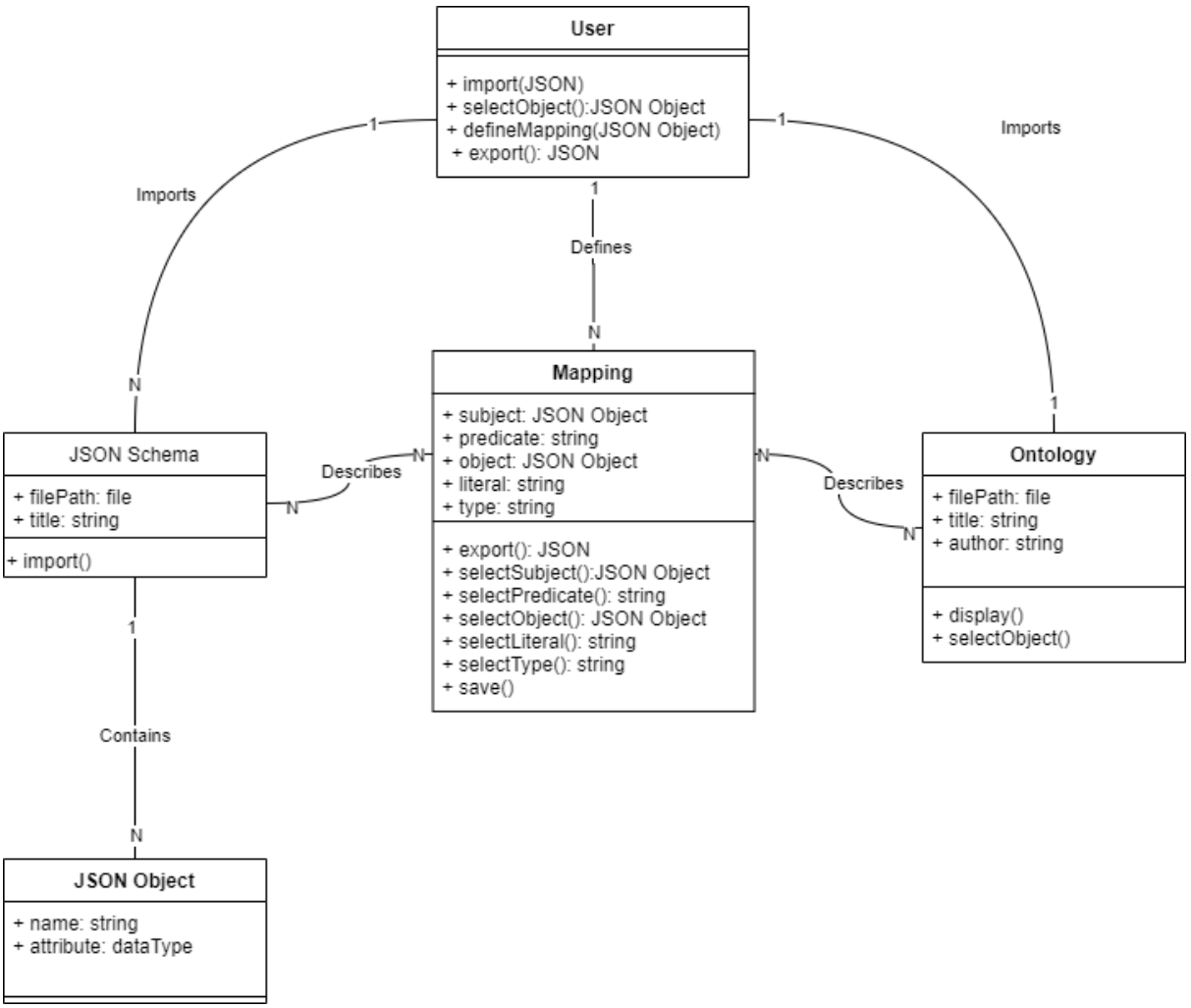
The user can then export the mappings.



3.4.3 Object Model

The planned objects are reflected in the object model below. The Userobject contains the information and the actions about the primary actor of the system. The Mappings will be the primary object on the screen at any time reflected in the UI mockup in Section 3.4.1. Secondary objects of the JSON Schema and the Ontology files that the User might upload are secondary objects that persist on the screen as per the UI mockup referred to earlier. The Mapping object will be defined by the User and can be represented by metadata such as the JSON object that is the “subject” of the mapping. The JSON Schema and Ontology objects can contain multiple Mapping objects.

Note that this is the team’s current view of the object model of the system, which may evolve as the project progresses. The primary focus will be on following the UI mockup and fulfilling the use cases, and additional objects may be added to the system to achieve these objectives.


User
+ import(JSON)
+ selectObject():JSON Object
+ defineMapping(JSON Object)
+ export(): JSON
1
Imports
1
Imports
1
Defines
N
N
Mapping
+ subject: JSON Object
+ predicate: string
+ object: JSON Object
+ literal: string
+ type: string
+ export(): JSON
+ selectSubject():JSON Object
+ selectPredicate(): string
+ selectObject(): JSON Object
+ selectLiteral(): string
+ selectType(): string
+ save()
1
JSON Schema
+ filePath: file
+ title: string
+ import()
N
Describes
N
N
Describes
N
Ontology
+ filePath: file
+ title: string
+ author: string
+ display()
+ selectObject()
1
Contains
N
JSON Object
+ name: string
+ attribute: dataType

3.4.4 Dynamic Model

The dynamic model for this system. The actions in the sequence are the same actions provided in the use case. This particular sequence is a sequence where the user uploads a Schema or Ontology. The system then displays the KG. The user then selects an object from the ontology and defines any relationship that object has to another or defines an attribute of the selected object. The system updates the attributes and relationships of the object, the system can export the updated schema as JSON.