CSU2013/CSU3013: Group 10 Software Design Specification Outline

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 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.

1.4 References

2. System Design

2.1 Design Overview

2.1.1 High-level overview of how the system is implemented, what tools, frameworks and languages are used etc.

The development of the system will mainly be done with Vue.js, an open source model–view–viewmodel front end JavaScript framework for building user interfaces and single-page applications. We chose to use Vue.js due to its simple integration, flexibility, and thorough documentation. GitHub is used for version control and code sharing. We are using the Vuetify component library for Vue in order to allow us to easily create Material Design components to create visually easy to follow user interfaces. The fact we are using a Material Design library means our users will recognize and be familiar with the look and feel of the web application.

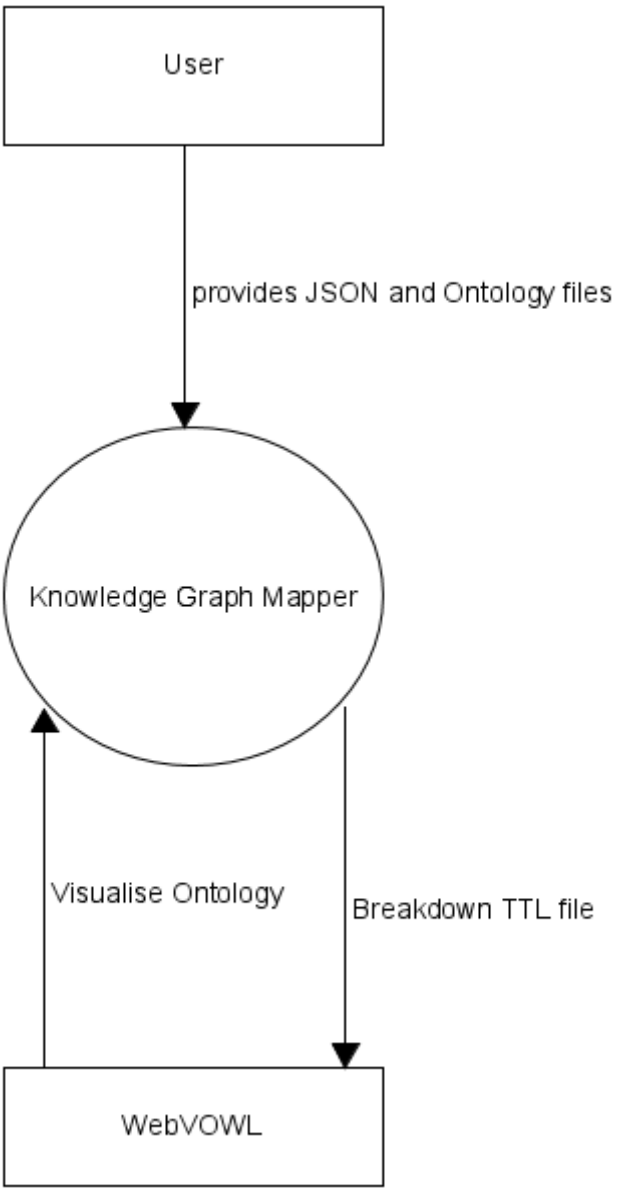
We will be allowing the user to upload local files to the application without having to upload the files to the server hence alleviating the potential security risk of uploading sensitive information. We are using a Google backed library called Blockly in order to allow the user to create their mappings in a “jigsaw” like manner. This allows us to create unique block names, titles and properties to each individual mapping.

The entire project is built on NodeJS which means it can be easily run and built for production through Node Package Manager commands. The coding is done in JavaScript, HTML and CSS within .vue files.

2.2 System Design Models

The following are different diagrammatic representations of the Knowledge Graph Mapping System. The Use Case, Object and Sequence Diagram are carried over from the requirements document which was reviewed and edited by our clients.

2.2.1.System Context


User
provides JSON and Ontology files
Knowledge Graph Mapper
Visualise Ontology
Breakdown TTL file
WebVOWL

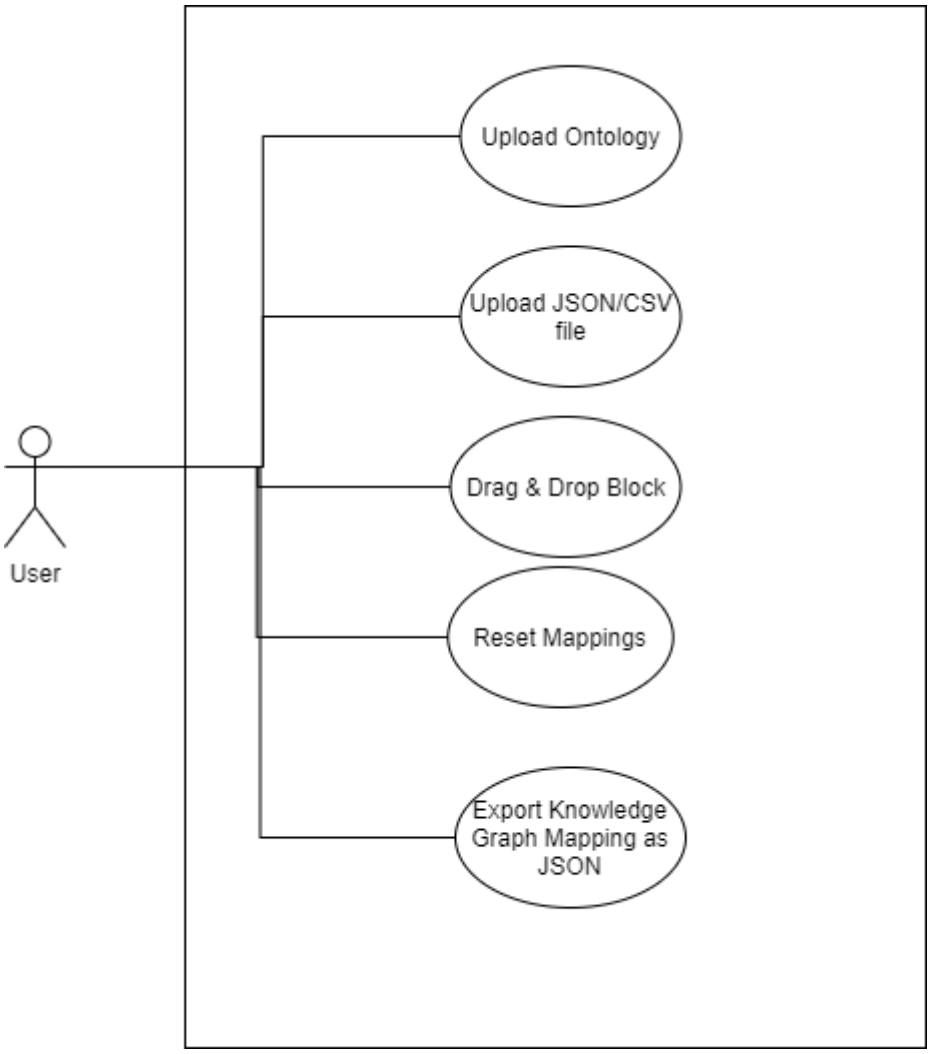
2.2.2.Use cases (from Requirements)

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

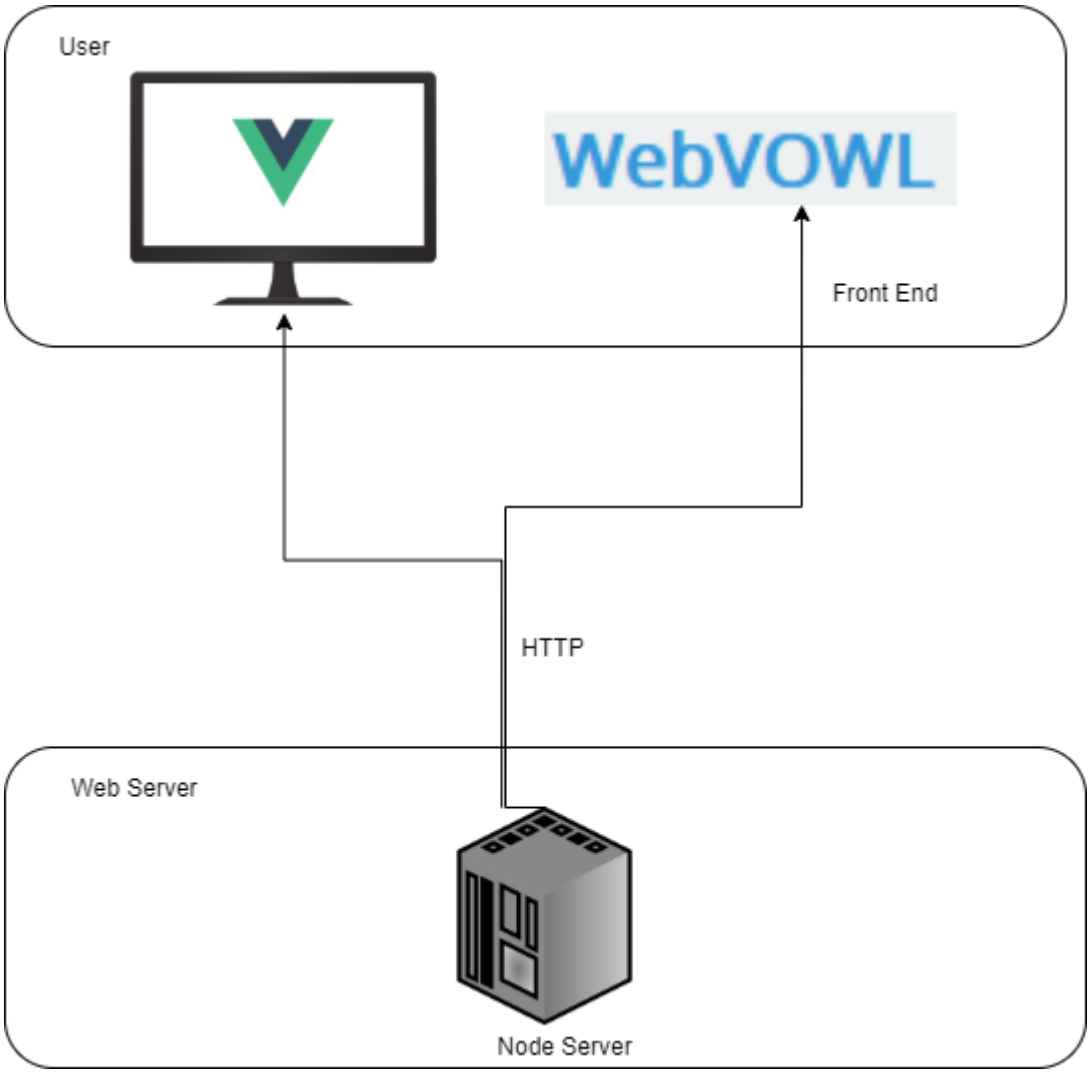
The user can upload both a JSON/CSV file as well as an KG ontology.

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.

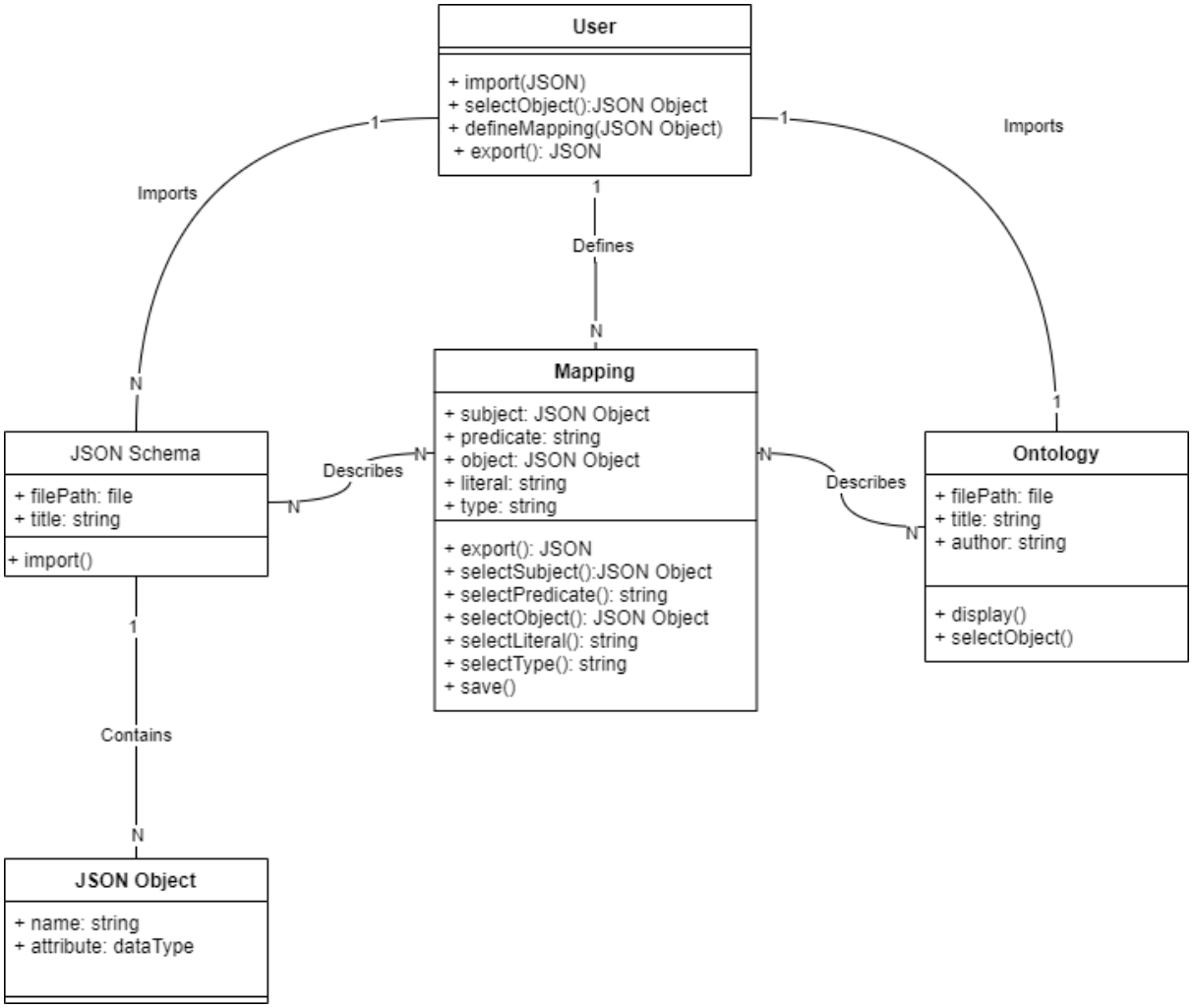
The user can then export the mappings.



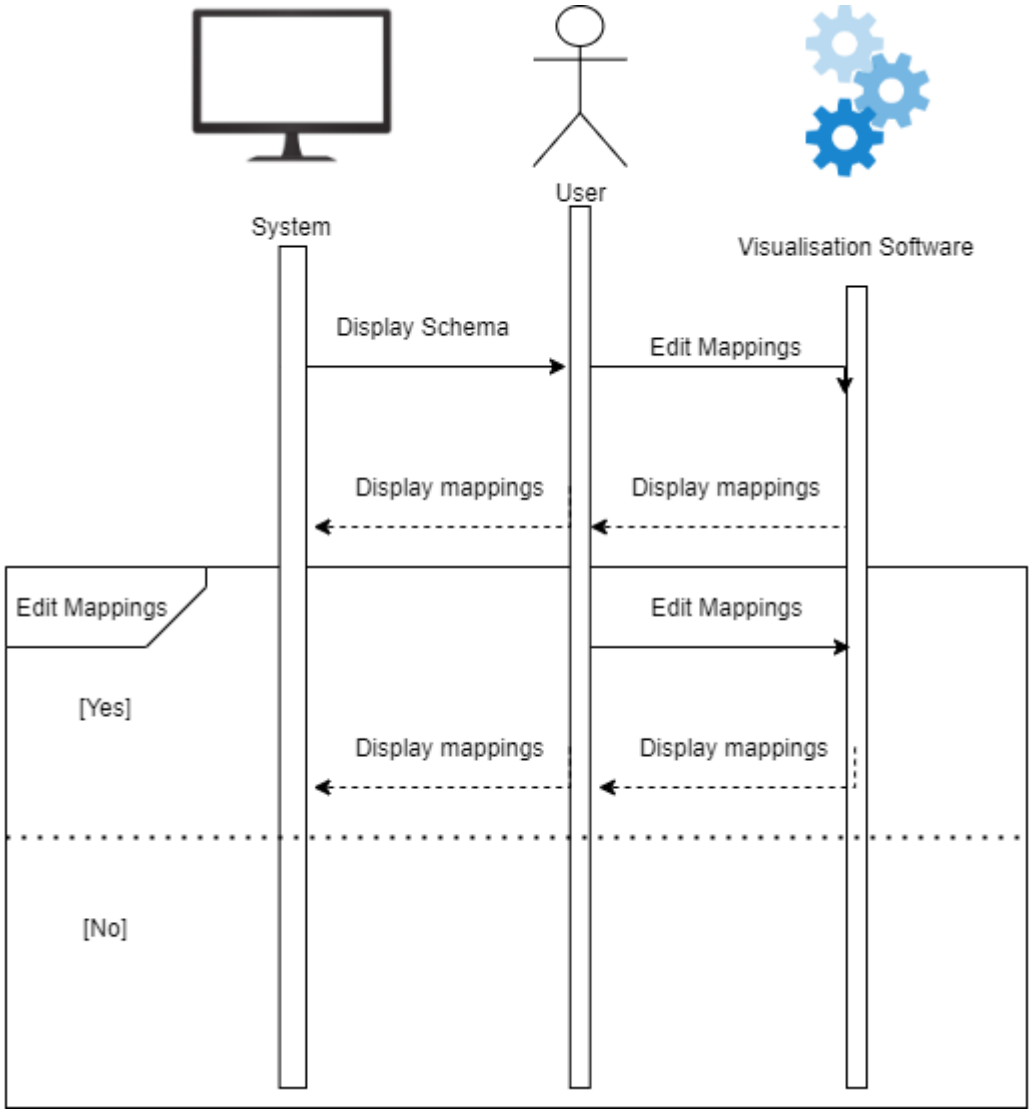
2.2.3.System Architecture



2.2.4.Class Diagrams



2.2.5.Sequence Diagram



2.2.6.State Diagrams