# CITB679
# Практика по бизнес информационни системи

## Study assistant

### Изготвили:

Десислава Петрова – f95361

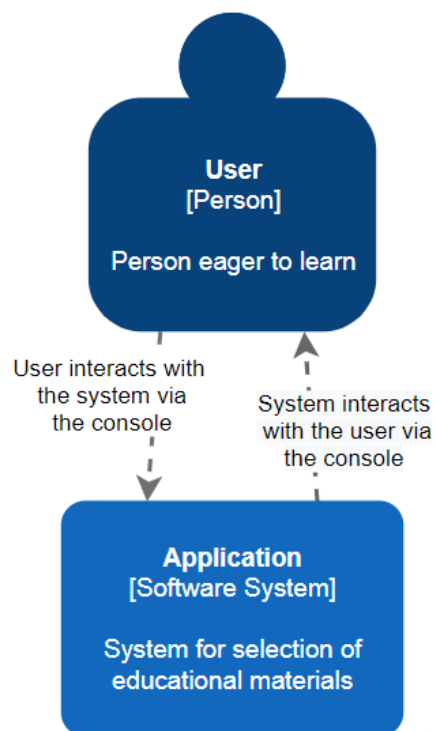Лъчезар Захариев – f103149

Асен Леков – f101445

# Table of Contents

# 1. Project description

All educational courses require some prerequisites. Also it is hard to introduce materials to different people. Different approaches must be taken, but most of the time students cannot have personal education just for them. All classes are composed from a group of people with different knowledge levels, interests, background etc. Teachers try to take different approaches to present and provide all learning materials to be accessible for all but this is not the case all the time. Given a structured data of learning materials and their dependencies between, tagged with tags, background concepts topics and more, the goal of this project is to provide an easy way to generate a list of material based on a criteria and the student personality or background.
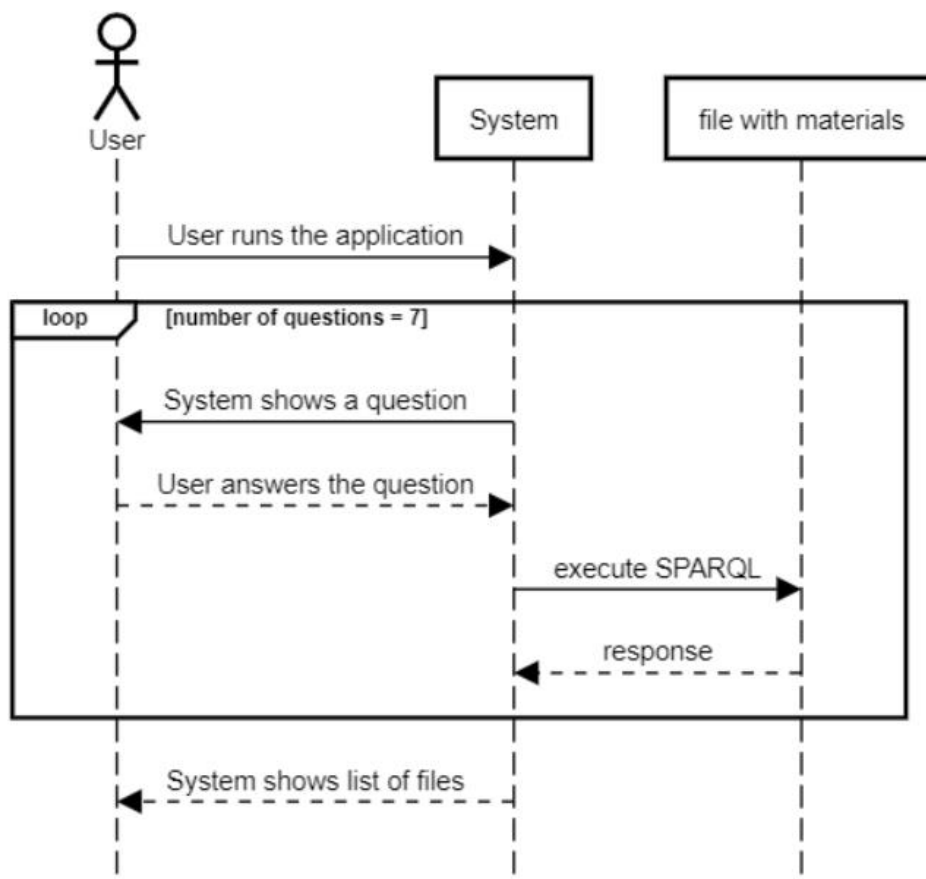
# 2. Use case

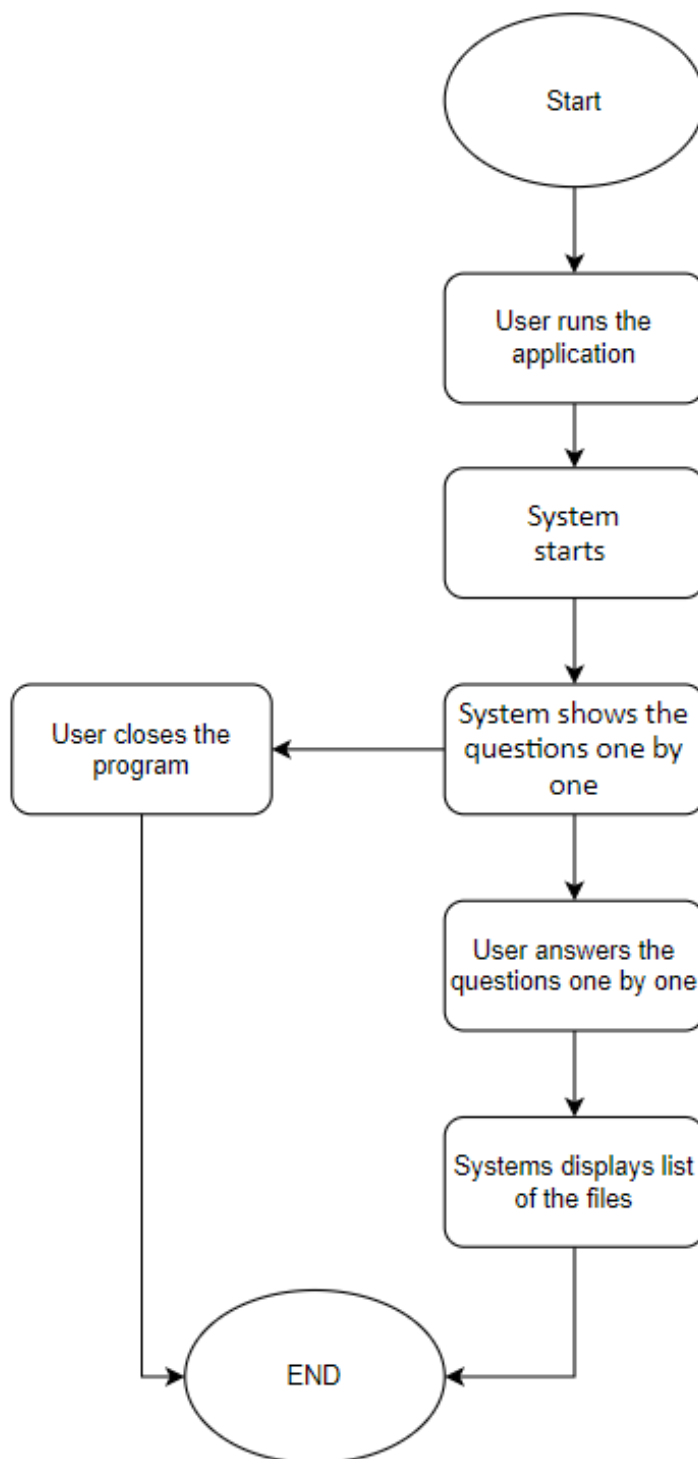| Use case name | Search for materials for studying |
|---|---|
| Description | Program extracting the necessary training materials according to the information entered by the user |
| Actor | User/ Person |
| Triggers | |
| Preconditions | User has the desire to learn and has previously downloaded the application. |
| Post conditions | |
| Main course | 1. User runs the program.<br>2. System starts and shows the first question about the age group. (EX1/EX2)<br>3. User selects an age group. (EX1/EX2)<br>4. System displays a question about the topic. (EX1/EX2)<br>5. User chooses a topic. (EX1/EX2)<br>6. System displays a question about the programming language. (EX1/EX2)<br>7. User selects a language. (EX1/EX2)<br>8. Systems shows a question about the concept. (EX1/EX2)<br>9. User chooses a concept. (EX1/EX2)<br>10. System asks the user what is his education level. (EX1/EX2)<br>11. User selects a level of education. (EX1/EX2)<br>12. System asks the user to choose educational materials. (EX1/EX2)<br>13. User selects educational materials. (EX1/EX2)<br>14. System asks the user to describe what he knows from the materials. (EX1/EX2)<br>15. User describes his knowledge. (EX1/EX2)<br>16. System displays a learning path of materials for studying. (EX1/EX2) |
| Alternate course | |
| Exceptions | EX1. User decides to close the program.<br>1. Program closes.<br>2. Go to preconditions.<br><br>EX2. The system interrupts.<br>1. Program stops and closes.<br>2. Go to Main course step 1. |

# 3. Project architecture



# 4. Sequence diagram

# 5. Process diagram

```
                    ┌─────────┐
                    │  Start  │
                    └────┬────┘
                         │
                         ▼
                 ┌───────────────┐
                 │ User runs the │
                 │  application  │
                 └───────┬───────┘
                         │
                         ▼
                 ┌───────────────┐
                 │    System     │
                 │    starts     │
                 └───────┬───────┘
                         │
                         ▼
┌───────────────┐ ┌───────────────┐
│ User closes the│◄│System shows the│
│    program     │ │questions one by│
└───────┬────────┘ │      one       │
        │          └───────┬────────┘
        │                  │
        │                  ▼
        │          ┌───────────────┐
        │          │ User answers the│
        │          │questions one by one│
        │          └───────┬────────┘
        │                  │
        │                  ▼
        │          ┌───────────────┐
        │          │Systems displays list│
        │          │  of the files  │
        │          └───────┬────────┘
        │                  │
        ▼                  ▼
       ┌─────────────────────┐
       │        END          │
       └─────────────────────┘
```

# 6. Used technologies – RDF and OWL

At the rise of the internet a computer scientist brings an idea to build a semantic web and connect all the possible data inside the internet. Because of the heterogeneous data which must be linked together they came up with a Resource Definition Framework and more specific for the case of the Internet - OWL or Web Ontology Language. The RDF framework defines a different schema for different called ontologies. Each ontology provides a set of instructions on how to define and describe a piece of data into a given ontology and something more interested - RDF is not scoped to a single field of ontology, it's a framework and the user can link together different data from a different ontologies. The idea behind was, all webpages to define such data in a given schema with their own or a common ontologies and web crawlers to build a massive graph of resources connected between. Sounds similar to the purpose of linking learning documents together.

# 7. Classes

```
┌─────────────────┐        ┌─────────────────┐
│  KnowledgeApi   │        │ KnowledgeLibrary│
│   <<Class>>     │        │   <<Class>>     │
├─────────────────┤        ├─────────────────┤
│                 │        │                 │
│                 │        │                 │
│                 │        │                 │
└─────────────────┘        └─────────────────┘
```

| Class name | KnowledgeApi | KnowledgeLibrary |
|---|---|---|
| Attributes and methods | + __init__(self, library)<br>+ __language_where_clause(self)<br>+ __concept_where_clause(self)<br>+ __edu_level_where_clause(self)<br>+ __ask_for_age(self)<br>+ __ask_for_topic(self)<br>+ __ask_for_lang(self)<br>+ __ask_for_concept(self)<br>+ __ask_for_level(self)<br>+ __ask_for_material(self)<br>+ __ask_for_already_known(self)<br>+ __build_query(self)<br>+ prompt_for_user_profile(self)<br>+ get_learning_path_by_criteria(self)<br>+ set_age(self)<br>+ set_topic(self)<br>+ set_lang(self)<br>+ set_concept(self)<br>+ set_education(self)<br>+ search(self)<br>+ search_deps(self)<br>+ list_ages(self)<br>+ list_topics(self)<br>+ list_languages(self)<br>+ list_concepts(self)<br>+ list_educations(self) | - __init__(self)<br>- generate(self)<br>- __populateUnique(self)<br>- __populateCourseDependencies(self)<br>- __spreadRanges(self)<br>- __populateCourseAgeRanges(self)<br>- __populateCourseTopics(self)<br>- __populateCourseLanguages(self)<br>- __populateCourseEduLevels(self)<br>- __populateCourseConcepts(self)<br>- __populateCourseTags(self)<br>- export(self)<br>- load(self) |

# 8. Wireframe



# 9. Project structure

Following the good practices this project defines a Python module library with few classes and an embed module for simple web UI built on Flask. Three main parts of the project source code can be addressed:

**KnowledgeLibrary**

This is a class which provides a way to import, generate and export RDF graphs. Generation is done via parsing of excel sheets with a given format into a RDF graph using all necessary schemas. The library also supports defining dependencies of the learning materials. Each excel row have an ID in a format T{course}.{material} eg. T8.12

The sheet row also contains a column with a comma separated or a range of prerequisites. The library take care of parsing the ranges in create the links between the nodes into the graph

**KnowledgeApi**

The KnowledgeApi takes a library and using SPARQL queries define different methods to fetch available topics, background, age ranges etc. Also the API class gives the ability for searching based on a list of criteria which are called user_data.

**Flask app**

Is a simple module following the Flask standards and implementing a simple web app with a page where a RDF KnowledgeApi is consumed. Because of the modularity of Python this is easily implemented.

**edu_graph**

This is a python module containing all above described classes and files. Using the default __init__.py class for a Python module this project implements an interactive CLI tool. When imported as a module inside another module, the API can be used as a standalone class with public methods.

## 10. Project decisions

Learning materials are composed via different characteristics like topic, field, concept complexity and more. Each material can be prerequisite to one or more others or can relate to others. Because of this the project targets to have all the data structured in a way to be easily queryable, persist and update. After a research was made few interesting models appeared - RDF and OWL, SPARQL matching all the project requirements which can help to build a model of the data which can be searched within. For a programming language Python was set

as a requirement and is a good candidate for such a project as it provides a powerful analytics library for parsing and extracting data from documents. This is used to prepare a preset document data into a RDF structure. Python has in addition different modules for building simple web interfaces, building interactive CLI tools and more.

Because of the mature nature of the RDF and all available schemas, there are plenty of them available to be used and a custom can be defined. This project uses few standard schemas and one focused on education - Open Educational Resource Schema http://oerschema.org.

Once a graph is built using all required RDF ontology schemas SPARQL will be used as a most common and widely spread used RDF query language. This will open the field to provide powerful APIs and accomplish the goals of the project.

## 11.   User guide

### Installation

The project readme file describes in detail how the project can be set up and get running. It provides three ways of usage - API classes, CLI tool and a Flask web app. A common virtual env approach is taken to create an isolated env where the project can be installed and executed. All dependencies are located into requirements.txt and can be easily imported via console. **The only requirement is the machine to have a Python 3 version installed.**

## 12.  Testing

Testing is done the old-fashioned way for now, through pure trial and error following use cases. No unit tests have been introduced yet. Creating unit tests should be strongly emphasized in the future.

Tests undertaken to this point are checking the correct behavior of the following features:

- Ability to correctly query non-specific/missing languages, concepts and educational levels.
- Ability to dynamically display a set of possible answers based on the user's input.
- Ability to select many options in multiple choice questions and correctly query based on all of them.
- Ability to correctly display learning path after supplying full set of answers
- Ability to correctly parse data from the learning materials file
- Ability to correctly display the full set of answers that are available in the learning data.

## 13.  Features to be developed

- In the future, the scientific fields covered by the application may be extended

- Ability to create an improved interface
- Ability to download the listed files/learning path from the application
- Unit tests

## 14.  Documents

| File | Description |
| --- | --- |
| Example Learning Data.xlsx | Learning Data |
| Meeting`s documentation | Minutes of meetings |
| https://github.com/L3K0V/nbu-citb679 | Git repository of the project |
|  |  |
|  |  |
|  |  |

|  |  |
|---|---|

## 15. Used materials

| Link/File | Description |
|---|---|
| **https://en.wikipedia.org/wiki/Resource_Description_Framework** | What is RDF |
| **https://www.orpha.net/sparql?nsdecl** | RDF namespaces |
| **https://en.wikipedia.org/wiki/RDF_Schema** | RDF schemas |
| **https://rdflib.readthedocs.io/en/stable/intro_to_parsing.html** | Introduction to loading/saving a RDF |
| **https://rdflib.readthedocs.io/en/stable/gettingstarted.html** | Introduction to RDFLib |
| **https://rdflib.readthedocs.io/en/stable/intro_to_graphs.html** | Introduction to Navigating Graph |
| **https://rdflib.readthedocs.io/en/stable/intro_to_sparql.html** | Introduction to SPARQL |
| **https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#section-1** | DCMI metadata |
| **https://www.w3.org/2000/01/rdf-schema#** | Sample RDF schema |
| **https://www.w3.org/2009/Talks/0615-qbe/** | SPARQL examples |
| **https://www.w3.org/TR/sparql11-query/#pp-language** | SPARQL Property paths |