Web Engineering 2023-24 Project Description

V. Andrikopoulos

November 17, 2023
Version: 1.0*

1 Introduction

The Web Engineering project allows you to put into practice the material discussed during the lectures and tutorials in groups of 2 or 3. The following sections discuss the application to be developed by all group projects for the course, the requirements for this application, the expected deliverables and their deadlines, and their assessment points. The use of the verbs must, can, etc. is canonical.

2 Case Study Description

The Web app to be developed builds on the "Data on CO2 and Greenhouse Gas Emissions by Our World in Data" dataset available on GitHub. The dataset is available in CSV and JSON formats, together with a short description of the fields in the form of a codebook. The dataset is relatively large and should be processed further before being incorporated into the application, i.e. *it should not be used as-is*. In order to keep it to a manageable size you could first remove all fields that will not used by the API below; beyond that you can choose to reduce its size further by only keeping data from the last 50 years. In any case, the original CSV or JSON file **must not** be in your group project repository. Consider adding it in the `.gitignore` file to avoid accidentally including it in the repository.

The goal of this project is to deliver to users both basic and advanced features based on this dataset as a Web app. It comes with a minimum set of requirements on what features are to be delivered. The following section discusses these requirements.

Note: This document is a "living" one, meant to be updated in the following weeks if necessary. Use the version identifier for an indication of changes to it. Refer to Section 7 for changes between versions.

3 Requirements

The text below is meant to be read as essential requirements on the project; this means that your projects must *treat all statements as conditions on your solution and convincingly address them.* The use of verbs in this section is compliant with RFC 2119[1].

REQ$_1$: Design and document a RESTful API that must provide **all** of the following functionalities:

1. to retrieve, create, update, or delete the population and GDP[2] data for a country identified by its name or ISO code for a given year;
2. to retrieve all available emissions related data (at minimum everything under the `co2`, `methane`, `nitrous_oxide`, and `total_ghg`[3] columns, where available) for a country identified by its name or ISO code, optionally filtered from a provided year and later;
3. to retrieve all temperature change data (columns `share_of_temperature_change_from` and `temperature_change_from_*`) per continent, optionally filtered from a provided year and later;
4. to retrieve the energy per capita and per GDP data for all countries in a given year, if available, sorted per population size and returned in batches of $M = \{10, 20, 50, 100\}$;
5. to retrieve the top or bottom $N, N \geq 1$ countries based on their share of contribution to climate change (column `share_of_temperature_change_from_ghg`) either for a specific year or for the previous $M, M \geq 1$ years.

Note: There does not have to be an 1-to-1 mapping between entries in the list above and distinct endpoints in your API. It is allowed to separate functionalities into multiple endpoints, or create one endpoint that satisfies multiple requirements, as long as your API as a whole provides all required functionalities.

REQ$_2$: Each API endpoint you design should support both JSON and CSV representations of the resources (i.e. Content-Type is application/json or text/csv). JSON should be the default option if none is specified. The endpoints **must not** return HTML representations under any circumstances, including error messages.

REQ$_3$: The documentation of the API must cover at minimum for each endpoint:

- its name together with a short description of its functionality,

[1] https://www.ietf.org/rfc/rfc2119.txt
[2] Gross Domestic Product https://en.wikipedia.org/wiki/Gross_domestic_product
[3] GHG: GreenHouse Gases

- how to access it,
- what representations it accepts and returns, and
- what errors are returned together with their meaning.

Providing a specification for the API out of which the documentation is generated is *strongly recommended* but not enforced. *Note:* consider using an appropriate tool for the API specification/documentation; see the REST slideset for more information.

REQ$_4$: Implement the API as the back-end of the app using the technologies and programming language(s) of your choice, and document the implementation code appropriately. You must update the REST API documentation and specification if necessary in case of deviations from the initial API design.
Note: The use of a database for persistence in the back-end of the app is optional but highly recommended. The design of the database, if any, will **not** be evaluated.

REQ$_5$: Implement a front-end for the Web app which builds on the implemented API to offer a User Interface (UI). All functionalities designed and implemented for the API as per the previous requirements must be easily accessible through it.
Note: The goal of this task is to demonstrate the developed features as a complete Web app; graphic and UI design skills will **not** be evaluated. However, your front-end should allow users to access **all** the functionality provided by the API through its UI in an appropriate manner. The use of Web technologies discussed during lectures and tutorials is expected and desired, respectively.

REQ$_6$: In order to allow us to independently test and run your project you must provide a `docker-compose.yml` file that will start your complete application after just running `docker compose up` in the root of your repository, using ".env" files to configure the ports on which the application will be made available on the Docker host where necessary. You are allowed to build custom images using one or more Dockerfiles. The assumption is that you are using Linux as an operating system for this purpose. Use deliverable D4 as discussed below to make sure that your application can be successfully deployed and ran by your TA before the final deadline for the project. The submission for D4 does not have to be complete in terms of functionality.

REQ$_7$: In addition, provide all the necessary documentation for your application. This consists of two parts. The first part is a short report documenting:

1. the technologies used for the application and the basis on which they were selected,
2. how the delivered application fulfills all the requirements above,

3. the level of maturity of your RESTful API and any potential issues with it, and

4. the distribution of work among group members and the role(s) that each member performed.

All documentation artifacts must be navigable and visible by browser alone (i.e. no PDF, Word, etc.), preferably but not necessarily as markdown documents. Your source code should have enough comments so as to be easily understandable in combination with your project report.

REQ$_8$ (in case of Generative AI adoption): Depending on the type of Generative AI approach you used to help with the project, the project report also has to contain a section describing:

- *In case of adopting LLMs* such as ChatGPT: the name and version of the LLM, together with a clarification if a paid version is used; **all prompts** resulting in code or documentation that (eventually) ended up in your project; a reflection on the amount of effort required for identifying the appropriate prompts, and for incorporating the LLM-generated outcomes to your project.

- *In case of adopting AI assistants* such as Copilot: the name (and version if possible) of the assistant and the IDE in which it was integrated, together with a reflection on how frequently and on which parts of the codebase the assistant's suggestions were more (or less) useful. Specific examples would be appreciated but not mandatory. In addition to this, **all parts of the codebase** that are accepted suggestions from the assistant should be marked accordingly in the codebase using comments.

4 Deliverables

The following deliverables are expected for the project:

D0 Self-enrolment into a group and submission of GitLab handle through brightspace.

D1 Project plan outlining the deadlines for the rest of the deliverables as identified below, with the exception of the final (D5) deliverable which has a fixed deadline.

D2 API design and, optionally, specification.

D3 Back-end of the application.

D4 Functional but not necessarily complete deployable application to be checked.

D5 Complete application, including its documentation.

All reports and software are to be delivered by doing a Merge Request (MR) to the main branch of the repository that you were provided with as a group, tagging your assigned TA to the request. All development **must** therefore take place in separate branch(es). Use the deliverable identifier to label your MR accordingly. All feedback to the deliverables will be provided through comments on closing the respective MRs, and by opening a separate issue where necessary.

Deliverables D0, D1, and D5 have fixed deadlines: 22/11, 29/11, and 17/1 at 23:59 CET, respectively. Deliverable D1 is to provide a timetable for delivering deliverables D2 to D4, at the discretion of the group. No two deliverables can be scheduled within 6 days (inclusive) of each other, i.e. there has to be a calendar week distance between them, including their distance from fixed deliverables D1 and D5. No deliverable can be scheduled on a non-work day; this includes both all weekends and the two weeks in December/January where there are no teaching activities. A suggested plan for the deliverables is as follows:

D2 At the latest by 6/12.

D3 At the latest by 13/12.

D4 At the latest by 10/1.

Deadlines can be amended after D1 is submitted through an MR to the same deliverable, but only if this MR is done at least a week in advance from the next scheduled deadline.

In addition, the project plan can (optionally) include an indication of the programming language and technology stack that is going to be used for the development of the Web application. Some suggestions/indicative choices in this case are:

1. Java using Spring or Quarkus for the back-end, and TypeScript using Vue for the front-end;
2. Python using Flask or FastAPI for the back-end, and the same as above for the front-end;
3. C# using the stack provided by the course tutorials.

Groups are allowed to request an in-person or online meeting (30' max) with the responsible TA per deliverable before its deadline by opening an issue in their repository and tagging the TA, but only if they have respected the deadlines for **all** previous deliverables.

5 Assessment

As defined in the introductory slideset for the course, the capstone project consists of three components: the software (Web app) itself, the documentation for the software, and the demonstrable functionality of the app. These components are assessed in turn as follows:

Software The endpoints designed for the API satisfy all relevant requirements, they are consistent in their style and interaction with the API consumer, and they follow good design principles. Their implementation in the back-end is complete and is based on sound design patterns supported by the adopted supporting framework, allowing for easy programmatic interaction with the API for all delivered functionality. There is minimum coupling between the back-end and the front-end, and the front-end relies only on the endpoints offered by the back-end and/or by third parties (if necessary) in order to operate. The front-end is sufficient for delivering all functionality delivered by the back-end, with extra points for value added functionality delivered on top of this (see below).

Documentation The documentation of the Web app, in terms of code comments, API documentation, and project report is sufficient for any developer with a Web Engineering background to easily understand how the delivered software works, which architectural decisions were involved, and what are potential issues that they should be aware of when adopting the delivered API.

Demonstrable functionality The Web app can be independently deployed using only the provided documentation and scripts, without interventions to its code or configurations beyond what is absolutely essential. The app is intuitive to navigate, and using the report as a guide only if necessary, the user is able to access all implemented functionalities.

6 Grading

In order for a project to even be eligible to receive a grade then deliverables **D1 and D5** have to be delivered within the defined deadlines. Projects that fail to meet the deadlines in this case will receive a failing grade (0.0) in the assignment for all group members and are eligible for the resit. Failing to meet deadlines D2 to D4 results in the group losing access to the coaching by the TAs as per the introductory slideset; in this case the group will only receive feedback on their project after the final deliverable. Grades are given per project since they reflect the team effort, except where and when it can be established clearly that there has actually been none.

Projects that ensure that all requirements discussed in Section 3 are fulfilled receive *a maximum of 7.0*. If and only if **all requirements** are met, then the remaining 3.0 points will be given to projects providing the users with added value by:

1. implementing additional endpoints to the API of sufficient complexity and utility, and offering them as features in the UI front-end (0.5 point max),

2. using at least one 3rd party API to deliver advanced features to the Web app users (0.5 point max),

3. deploying the tiers (front- and back-end, plus databse if available) of the application across multiple containers (0.5 point max),
4. following appropriate design and implementation principles and patterns discussed during the course, and the report rationalizing the developers' decisions effectively and efficiently (0.5 point max), or
5. implementing a CI/CD pipeline to automatically build, deploy, and run their Web app (1.0 point max).

7 Change Log

Version 1.0 Initial version of the project description.