COMP3227 REST Coursework 2023/24

Key Information

Deadline: 06/12/23
https://handin.ecs.soton.ac.uk/handin/2324/COMP3227/1
The usual late penalties apply for late handins.
Lecturers: Heather Packer, Richard Gomer, Nicholas Gibbins
Value: 50% of the marks available for this module.

This coursework is to be completed individually.

Setting the Scene

You've just started your new job as a Web Architect at the IT department in the university. Your predecessor started to build a REST API to allow people to view details about the university's faculties, schools and departments, and the degrees and modules that they run, however it was left unfinished and did not start any API documentation.

Before they left for a meeting, your boss hurriedly scribbled this URI on a piece of paper:

http://comp3227.ecs.soton.ac.uk/modules/15

Try to GET this, as a starting point. (You must connect to the ECS Virtual Private Network to access it.)

Part 1

Your first task is to browse and document the existing University Faculty API. You have one starting URI, but use what you have learned about HTTP and REST to explore the API and make note of what you find. Your colleagues recommend that you use an HTTP tool like Postman, RESTClient or cURL to do this.

Some questions that might help guide your exploration. You should include answers to these in your documentation:

- What methods can you use on each URI?
- What format can URIs take?
- What other URIs are possible for this service?

You must:

1. Add your choice of a new module to the course of your choice to the University Faculty API (figured out from the URI above), so that it appears in the list. Pick a module that does not already exist.

2. Write an HTML5 page to document how the API works. You should write about each endpoint in the API that you find. You must have one page for the whole API, ordered and split into sections as appropriate. It is expected that the documentation for an API endpoint will include:

a. The URI template.
b. What each parameter in the URI means, and what types of value it takes.
c. Which HTTP methods can be used on the resource.
d. Provide example output for each type of resource.
e. Provide example input for each interaction that modifies a resource.

3. Write a critique of the REST API you've documented as a RESTful interface

Note: We are assessing your ability to suitably structure an HTML5 document to represent the data, and to use some basic CSS to make the document readable and navigable, if you wish you may also use JavaScript but it is not required. However, **you are NOT allowed to use any frameworks to write your HTML5 or CSS, this means that you can not use frameworks like Bootstrap, Angular, or Vue**. And your artistic ability is not being assessed!

You will then **write a critique** of the existing API as a **RESTful interface**. Alongside the critique you may include potential changes to the existing endpoints and/or data structures to support the RESTful extension detailed in Part 2.

Part 2

Your boss now wants you to **design** a RESTful extension to the API which enables students and staff to make announcements relating to faculties, courses, courses by year group, and module, each announcement can be for either (i) staff and students, (ii) students, and (iii) staff. And staff can manage any announcements made by staff and students. **Please note that you will NOT build your designed API**. The system should handle:

- Adding an announcement to the relevant resource
- Editing and deleting announcements
- Reporting of misuse in the system

It is required that the following will be supported:

Staff can:

- Edit or remove announcements;
- Adding an announcement to a relevant resource.

Students can:

- If they are reported of misuse, the student's right to post an announcement will be revoked permanently (unless successfully refuted in an appeals process);
- Adding an announcement to a relevant resource maximum number of 10 per month to prevent spamming;
- Start an appeals process.

Staff and Students can:

- Adding an announcement to a relevant resource;

- Report misuse in the system.

While the above restrictions describe the possible actions, consider what will be handled at the API level, what will be represented in the data structures, and what is appropriate to happen behind the scenes on a server. You may include any additional support in your design for the needs of students and staff, which you see fit.

You must:

1. Plan each of the steps that go into the announcement process. Define URIs to identify the required additional resources and the methods that interact with them to perform the process and move through the states.
2. Draw a single state diagram that shows a user's (both staff and students) interactions with the API. (like the one in Figure 4.1 of REST in Practice).
3. You must provide a structured representation of all your new API endpoints.

Please document your extension to the API using the Swagger framework (https://swagger.io/), making sure you document:

a. The URI template.
b. What each parameter in the URI means, and what types of value it takes.
c. Which HTTP methods can be used on the resource?
d. Provide example output for each type of resource.
e. Provide example input for each interaction that modifies a resource.

Submission

You will be expected to submit the following:

- For Part 1:
  - Evidence of submitting a new module (screenshots or a text log of request and response).
  - 1 html file named ***index.html*** and 1 accompanying css file named ***style.css***, documenting the API in the existing system.
  - 1 txt file named ***critique.txt*** containing a critique of existing API as a RESTful interface

- For Part 2:
  - A state diagram named ***state.pdf***
  - 1 yaml file named ***extension.yaml*** containing the Swagger documentation,

detailing your designed extensions to the API to add the ability to make announcements.

Mark Scheme

Section	Notes	Marks

Presentation of documentation LO: Use common Web technologies	Use of appropriate HTML5 and CSS to present the documentation professionally	12%
API discovery and documentation LO: Identify the key characteristics of the Web Architecture	Coverage of API endpoints and documentation of endpoints with demonstration of use (sample output) Critique of existing API as a RESTful interface	23% 10%
Design of extension to the API LO: Design RESTful Web applications, Apply the Representational State Transfer (REST) architectural style to Web application design	State diagram Appropriate design of URIs Appropriate use of HTTP methods Application of HATEOAS Richardsons maturity level of API design	55%
	Total:	100%