James Moreby (210344149)

LearnOuts.com Logout

My modules

Individual Coursework - Part 2/2

ECS639U Web Programming 2023/24

As with the first individual coursework, ask a demonstrator to check your solution during labs (until given deadline). This coursework must be checked and submitted on QM+ by the above deadline. You must use this Github repository as a starting point for your coursework. You'll be mainly modifying the Django files: models.py, urls.py and api.py, and the Vue files: App.vue and the components folder.

Develop a basic Vue/Ajax frontend and Django backend for a simple web API. Your app must have **one model** with **four fields of different types** (on top of Django's auto-increment id field), for instance: CharField, DateField, EmailField and IntegerField.

On the Django backend your application should respond to the four Ajax request methods (GET, POST, PUT and DELETE). You can use the "method" attribute of the request object (`request.method`) to find out which type of request is being made (or use class-based views). For each request type you should respond appropriately according to the HTTP semantics, e.g. a POST request should be used to create a new instance of your model and save it on the database. All data being returned to the client by the web API should be done using Django's `JsonReponse` object, which takes a Python dictionary as input. You should NOT use the django REST framework library for this coursework. Your frontend must use:

- **Vue** to "reactively" modify the page based on current data (this is already setup in the template repository). You should use components to organise your frontend code, and make use of props and emit to pass data back and forth between parent and child components;
- There is no need to use a global storage (e.g. Pinia or Vuex) for this coursework;
- You must use Vue's options API (you should not use the composition API);

- There is no need to use Typescript for this coursework, you can write your frontend components in "vanilla" JavaScript;
- Bootstrap to style the page (this is already setup in the template repository). Most elements of the page should have a Bootstrap style, and forms should be included in Bootstrap modals;
- and the fetch API to make GET/POST/PUT and DELETE requests from the client to the server

On the **frontend** one should be able to:

1. see the list of all elements in your model (the list should be fetched via a GET request as soon as the page loads)
2. add a new element to the list (POST)
3. update the details of one of the elements (PUT)
4. delete an element (DELETE)

It is up to you to decide how these will be laid out on the page (but format it nicely with Bootstrap). Any request from the client to the server should be done via Ajax requests using the **fetch API**. **So after the first page load, there should be no further page refreshes**.

Your Python backend code should also conform to the PEP 8 style guide, in particular: all classes and functions must have a docstring, use 4 consecutive spaces for indentation, and follow PEP 8 variable naming conversions.

Submissions that do not use the starter template https://github.qmul.ac.uk/eew252/cwindividual will not be accepted.

Marking criteria:

Criteria	Max. mark	Considered?
One model with at least four fields of distinct types	15	
Good use of Vue components, props and emit to pass data between parent and child	30	
Ajax methods using fetch API fully working for all four HTTP	25	

methods		
Good use of Bootstrap styles and modals	20	
Python code conforms to PEP 8 style guide (classes and functions have a docstring)	10	
Total:	**100**	