Exercise 1. REST API

We will be using the Django Rest Framework (henceforth called DRF) in this session.

Ex 1a. IDEAS app, server-side API endpoint

Put together an extremely simple project to collect ideas. Every time you have an idea, you enter it, where it joins the collection of ideas. Eventually, we want to be able to share ideas between users, and to make it interactive, but for now, we'll keep it brain-dead simple. This is just to get a feel for what Django can do automatically.

1. Create a project called ideas_project, with an app called "ideas"
2. There is only one model, called "Idea", with the following trivial schema.
   Idea:
      content : varchar(1000) // Contains the text of the idea
   Use the default sqlite3 database; no need to configure postgres.
3. Migrate, to init the database
4. Follow the quickstart tutorial on the DRF site:
   https://www.django-rest-framework.org/tutorial/quickstart/
   You should have models.{Idea, IdeaSerializer} and views.IdeaViewSet.
5. Configure urls.py to have url 'ideas' point to idea.IdeaViewSet.
6. Check localhost:8000/api/ideas. See the OPTIONS menu to see a self-describing, hyper-linked API.

Ex 1b. Accessing API endpoint from client-side javascript

1. Start 'node', a shell for javascript
2. We will use axios from https://github.com/axios/axios.. This has already been installed.
   Start with:
   > axios = require('axios')
   Then follow the instructions on the axios page to access the ideas api
3. Use axios.get to get the entire list, and to get specific items
4. Use axios.post to create a new instance.

Ex 1c. Accessing API endpoint from a client UI

1. Create a directory static at the top-level (where manage.py lives)
2. Create 'static/ideas.html' with the following elements:
   a. input text (id = "ideatxt")

b. "add idea" button (id = "addidea")
c. an unnumbered list (id="idealist').

When an idea is entered in the text box, and the 'add idea' button is pressed, the text should get appended to the list. Try it out.

3. **ideas.html should load axios directly from a CDN, so it doesn't require the node_modules or any support from the server side. That is, have the following line in the header of your html:**

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

4. Serve ideas.html from a view function as follows:

```
def home(request);
    if request.user.is_authenticated:
        return render(None, 'ideas.html')
```

5. Now we will change the scheme of step 2. When the add idea button is pressed, don't update the list right away instead, use axios to initiate a POST (ex 1b, step 4), and update the list only when the server reports success. Also, blank out the input text box at that point, so another idea can be entered.
6. Have a global windows.onload event handler that uses axios.get('/api/ideas') to retrieve the entire list

Exercise 2. Authentication and authorization using OAuth2

In this exercise, you will learn how to use an external social media account and their user database, instead of having our own (or django's built-in) user scheme. We will use Github as our source of authentication.

Authentication: Verifying that the user is authentic, that he is who he says he is.
Authorization: Verifying that the user, authentic as he may be, is authorized to do something. Different users have different levels of authorization.

OAuth is meant for both. We can log into github (get authenticated) and github decides what we can do and not do (authorization).

Exercise 2a. OAuth2 DIY

In this exercise, we will authenticate using Github's oauth interface, and retrieve some user-level info such as email address, photo etc. We will follow
https://developer.github.com/apps/building-oauth-apps/authorizing-oauth-apps/
In this exercise we will ignore the built-in User models, and not worry about authorization. We are simply using Django as a dumb server to serve html.

1. Signup with github if you don't already have an account.
2. Register a new application with Github. https://github.com/settings/applications/new

   Application name: ideas, home page URL http://localhost:8000/, and the callback URL http://localhost:8000/ghauthenticated/ (Don't forget trailing slash) The callback URL is crucial; your app needs to handle it.
   From the point of view of Github, your app is the client (for its services). Github issues a "client_id" and a "client_secret" which your app will use later in its communication with github.

3. Create a new app in the ideas project called gh_auth (and add to INSTALLED_APPS). FCreate a view function that shows the following page to the user;
   "Please login using
   GitHub

   Link it up with a url to 'ghlogin/' in urls.py.
4. When the user clicks on the link and gets taken to github and ok's it all, github redirects the request with the following URL: localhost:3000/ghauthenticated?code=9823984. Recall this URL? It is the callback you configured earlier
5. Create a function of the following form and link it to the URL "/ghauthenticated"
   def github_authenticated(request):
   if request.method = GET:
   auth_code = *extract code from request*
   access_token = get_access_token(code) *// step 2 in documentation.*
   user_info = get_user_info(access_token) // *step 3 in documentation*
   return HtmlResponse("<html><body>" + str(user_info) + </body></html>")
   You have to fill in get_access_token and get_user_info yourself by following the instructions in the link outlined earlier.
6. Test it by going to localhost:8000/ghlogin. The second time you do it, you will not be taken to github at all, since the session has been cached.

   note: You will get an email from Github saying that this particular way of authorizing is going to be deprecated by the end of the year. Ignore it!

Exercise 2b. OAuth and DJango REST Framework.

Ex 2a was to give you a familiarity with the terminology. We will only use the knowledge, but none of the code! We will mosty follow the tutorial in https://www.digitalocean.com/community/tutorials/django-authentication-with-facebook-instagram-and-linkedin, but switch it for github. I will refer to this document as 'tutorial'. This is a long document, but only because it deals with facebook, instagram and linkedin accounts, each of which have small differences. We will use the `social-auth-app-django` library for our purpose.

1. Continuing in the ideas_project, remove gh_auth from INSTALLED_APPS, but let the code remain.
2. Wherever you see "facebook" in the tutorial code, replace it with 'github', in the corresponding upper or lower case.
3. Once you are able to get the entire login process working, go to localhost:8000/api/ideas and see your login name show up in the top right.

Submit this project as your_roll_number-feb18.zip