Specifications of Acted Features

Profile page

The main purpose of this feature is to allow Acted customers to display an artist profile page and to allow the artist to modify and customize his profile page.
Consequently, this page will appear and behave differently, depending on the user who is loading the page.

The page URL will respect the following pattern:
http://www.acted.co/profile/[ARTIST_NAME]
Where [ARTIST_NAME] will be the value of ARTIST.name (the column "name" of the table "ARTIST"), cleaned up to be displayed in a URL (without special characters and spaces).

Scenario #1: Customer POV

If the user, who's loading the page, is not authentified (information saved in the Session variable) or if the user is authentified but is not the owner of the profile, the profile page will appear as a read-only page. The user won't be allowed to modify any information on this page.
In that case, the page will have to be loaded with all the artist information, retrieved from the database.

Scenario #2: Owner artist POV

If the user, who's loading the page, is authentified and is the owner of the profile, the profile page will appear as a configurable page. The user will be able to modify few information on his profile page, such as:

- his profile picture
- his background picture
- his caetgories and sub-categories
    - The categories and sub-categories (Sing Jazz, Vocalist,... in the sample page) of an artist should be editable. When the owner artist clicks on it, the step 2 of the registration popup (with the list of categories and sub-categories, in "index.html" file) should appear to allow the artist to select/unselect sub-categories.
- his performance blocks (add a new performance block, modify the performance items (name, price, pictures, videos, description or technical requirements) or delete an existing block)
    - When the owner artist clicks on a performance block price, to edit it, instead of making it editable, we would like to redirect the artist to another webpage http://www.acted.co/dashboard (to be created later), with a hidden (POST?) variable menu="price", which will allow us to load automatically the menu price of the artist dashboard page.
    - For the link "Technical requirements", we would like to modify it to make a "Check Tech Rider" button (same format as above button in the sample page). For the button action, same behavior, if the owner clicks on it in edit mode, redirect him to the dashboard page with a hidden menu="techRider" variable.
- his biography

- his media (photo, video, audio)
    - For the upload of photo, the max size of the picture should be configurable in a property file of the project (parameters.yml?).
    - Videos edition will be setup through YouTube or Vimeo links.
    - Audio edition will be setup through Soundcloud links.
- ~~his social network feeds (Facebook, Instagram, Twitter)~~ **DON'T DO IT FOR NOW**

Every item should appear normally while mouse is out and should appear with a pencil (or either other symbol you might think appropriate) to make the user understand he could click on it to modify its value.

Each item modification should be saved in the database as soon as the user validates the modification.

~~Finally, the "Follow me" button will have different behaviors depending on the status of the user.~~
~~If the user is not connected, the button will open a login popup (not designed yet, just open an empty popup for now).~~
~~If the user is authentified and is not the owner of the page, it will save in the database, in a new table (to be created, "follower", with 3 attributes "followerId", "artistId", "followDate"), the authentified userId, the artist id, and the current date.~~
~~If the user is authentified and is the owner of the profile page, the button should appear disabled.~~
DON'T DO IT FOR NOW

Input

- http://www.acted.co/profile/[ARTIST_NAME] URL.
- The design mockup of the profile page, under the "profile.html" file, in the Mockup repo.
- Acted database.

Expected output

A dynamically generated HTML document, respecting the "profile.html" file design, filled with the [ARTIST_NAME] artist information (stored in the database).

Core development

The "profile.html" file will have to be split into the Twig templates, using the 3 levels layout inheritance:

1. App layout: 1st level containing the shared HTML structure for all the Acted website pages: the <html> tags, the <head> tags, the Acted blue menu bar (<nav> tag in the <header> tag) and the "by default" white Acted footer (Facebook, Instagram and Twitter icons + footer links and copyright message).
2. Bundle layout: 2nd level, inheriting from the "App layout", containing the shared HTML structure for all the templates of the bundle (defining the few blocks architecture, composing the profile page).
3. Template layout: final structure of the template, inheriting from the "Bundle layout", containing the content to be displayed.

You could compare the structure of the “profile.html” file with the “index.html” file (still in draft version) to identify the common structures.

You will have to implement a Profile Bundle able to:

- Determine whether the user is authentified or not and whether the user is the owner of the profile to be displayed: [Id in the Session] = ARTIST.userId where ARTIST.name = cleanup([ARTIST_NAME]) => to be sent to the profile page to tell the scenario
- Load all the artist information from the database to be displayed in the profile page
- Save the modification of each item of the profile, achieved by the owner user.

Constraints

No commercial or licensed frameworks or API should be used to achieve this bundle.
Only open-source codes or frameworks could be integrated in the final developments.

Test materials

In order to test the development of the 2 scenarii, you could implement a very simple page with 2 forms:

- First form will load the profile page as an unauthentified user (no information in the Session)
- Second form will load the profile page as an authentified user, supplying an user Id (saved in the Session)

In the case the tester is loading the profile page, with the second form, supplying the Id of the artist (defined in the database), the profile page will have to behave as described in the Scenario #2.

Documentation

A Word document will be required to describe all the useful info about the features of this specification, to explain, if it applies:

- What is the main goal of the feature?
- What are the feature inputs?
- What are the feature outputs?
- What are the main business rules implemented?
- Define a quick “How it works” schema.
- Which bundle handles the feature?
- What are the main classes?
- What are the services?
- What are the Twig templates?
- What are the add-on bundle you installed?
- What are the commands to run to install the feature?
- What are the commands to run to test the feature?
- How to test the feature?