



# Training Python-Django - 3

## Sessions

*Summary: Or how a simple Django project can turn you into a black belt in permission and user management!*

*Version: 1*

# Contents

<b>I</b>	<b>General rules</b>	<b>2</b>
<b>II</b>	<b>Today's specific rules</b>	<b>3</b>
<b>III</b>	<b>Exercise 00</b>	<b>4</b>
<b>IV</b>	<b>Exercise 01</b>	<b>5</b>
<b>V</b>	<b>Exercise 02</b>	<b>7</b>
<b>VI</b>	<b>Exercise 03</b>	<b>8</b>
<b>VII</b>	<b>Exercise 04</b>	<b>10</b>
<b>VIII</b>	<b>Exercise 05</b>	<b>11</b>
<b>IX</b>	<b>Exercise 06</b>	<b>12</b>
<b>X</b>	<b>Submission and peer-evaluation</b>	<b>13</b>

# Chapter I

## General rules

- Your project must be realized in a virtual machine.
- Your virtual machine must have all the necessary software to complete your project. These softwares must be configured and installed.
- You can choose the operating system to use for your virtual machine.
- You must be able to use your virtual machine from a cluster computer.
- You must use a shared folder between your virtual machine and your host machine.
- During your evaluations you will use this folder to share with your repository.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a 0 during the evaluation.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.


# Chapter II

## Today's specific rules

- You must use the LTS version of django.
- You can use the bootstrap version of your choice. This version must work with the LTS version of django.
- You must use Python3.

# Chapter III

## Exercise 00

	Exercise
Exercise 00: Anonymous sessions.	
Turn-in directory : <i>ex/</i>	
Files to turn in : Your project, your requirements.txt file	
Allowed functions : All the Django functionalities, random, django-bootstrap*	


You're starting a website project that will allow the users to post "Life Pro Tips".

After creating your project, an application and set a homepage at this URL /, implement the following anonymous session system:

- A web user visiting your site can get a user name. This name is randomly chosen in a list of 10 names you will define in the `settings.py` file of your project, which validity will only last 42 seconds after it's been defined.
- This name must be persistent and remain identical for 42 seconds. Once the 42 seconds have elapsed, the name must be redefined as previously explained.
- Highlight the implemented behavior adding a message in a `<nav>` element in the header of the homepage. This message will be a simple "Hello user!" replacing `user` by the name randomly given until the end of its validity.
- The name must instantly appear on the visited pages: if you need to reload the page to make it appear or change, the exercise is not completed!

# Chapter IV

## Exercise 01

	Exercise
Exercise 01: User creation.	
Turn-in directory : <i>ex/</i>	
Files to turn in : Your project, your requirements.txt file	
Allowed functions : All the Django functionalities, django-bootstrap*	

Now the anonymous session system has been implemented, it's time to add the user management with the creation of an authentication interface.

In this exercise, you will create this interface step by step:

- Start by adding a registration page and a log-in page. Add a link to each of these pages in your **nav** element. To this end, add all the necessary urls, templates, views etc...

You will also add the name of the site at the beginning of your **nav** and turn it into a link to your homepage.

- Create forms you will insert in these pages following these specifications:
  - Registration form:
    - \* You will insert 3 different fields, including:
      - A field for the user name.
      - A field for the password.
      - A field for the password confirmation.
    - \* The validation will be carried out only if each field is properly filled, that the given name is not used already, and both password fields are strictly identically filled.

- A log-in form:
  - \* You must insert 2 different fields, including:
    - A field for the user name.
    - A field for the password.
  - \* The validation will be carried out only if each field is filled and that the used username/password couple already exists in the database and can be identified.
- Now, implement your pages behavior:
  - In case of data validation:

**Registration:** Creation and activation of a new user with the provided informations. Then, connexion with this user and redirection towards the homepage.

**Connection:** Connection with the user matching the specified informations, and redirection towards the homepage.
  - In case of non-validation:

In any case, you will redisplay the page with the form containing the informations you've previously entered (minus the password).

**Registration:** Displaying one (or several) error messages explaining the error (required field, existing username, wrong password).

**Log-in:** Displaying an error message with the form (required field or wrong log-in informations).
- Finally, when the user is logged-in:
  - The links to the registration and log-in pages are replaced by a log-out (functional, duh) link that logs out the user and redirects them towards the homepage after it's been clicked.
  - The name of the anonymous session in the `nav` element is replaced by the username.
  - The visitor must not be able to access the registration and log-in pages. If they try to access them, they will be redirected towards the homepage.
- Of course, the passwords will be hidden when typed.




In another exercise, you will have to create your own user class to replace the provided default one.

It might be wise (but not necessary) to anticipate this change in your project so you don't have manually rewrite each call to your user class in your code.

# Chapter V

## Exercise 02

	Exercise
Exercise 02: Our Tips!	
Turn-in directory : <i>ex/</i>	
Files to turn in : Your project, your requirements.txt file	
Allowed functions : All the Django functionalities, django-bootstrap*	

Time has come to implement the main functionality of this project: a tip system!

To do so, you will need to conceive a **Tip** model with:

- A **content** field with... the tip content (in text form of course).
- An **author** field (the user who created the tip).
- A **date** field (tip's date creation).

The tips will have to be listed on the homepage and all the attributes stated in this section will have to be displayed.


Once logged-in, the user must be able to create a **Tip** with the form set on the homepage. (use a **modelform!**)

If the user is not logged-in, the form must not appear and the visitor must not be able to create a tip.



# Chapter VI

## Exercise 03

	Exercise
Exercise 03: Votes.	
Turn-in directory : <i>ex/</i>	
Files to turn in : Your project, your requirements.txt file	
Allowed functions : All the Django functionalities, django-bootstrap*	

Now that your Tips system is ready, it would be nice to be able to gather the best ones. Yes, it's time to set up an upvote and downvote system!

In order to do so, you must modify your `Tip` model so you can implement these functionalities.

You also must implement a way to delete the tips that were deemed uninteresting.

You must add the following elements to the tips listed on your homepage:

- A delete button
- An upvote button
- A downvote button
- The number of upvotes
- The number of downvotes

For the moment, the only restriction you will implement will be the necessity to be logged in in order to vote or delete. If the user is not logged in, none of these actions will be possible.

Each of these actions will reload the page.

Obviously, a `Tip` that's just been created will have 0 upvotes and 0 downvotes.

**Warning:**

- Upvoting and downvoting the same tip at once must be impossible.
- A user can only upvote or downvote for the same tip once. Clicking twice on the same button cancels the upvote or the downvote.
- If a user downvotes a tip they had previously upvoted, the upvote is replaced by the down vote (and vice-versa).




Carefully watch your upvote system implementation. An inconsistency, an imperfection (a number of votes not decreasing after the cancelling of a vote) will invalidate the exercise. You should consider using manytomany fields rather than "counters" to implement your vote system.

At this stage, every action should only work if the user is logged-in. You must be able to display or disable the element the user will not be able to use.

# Chapter VII

## Exercise 04

	Exercise
Exercise 04: Primary use of authorizations.	
Turn-in directory : <i>ex/</i>	
Files to turn in : Your project, your requirements.txt file	
Allowed functions : All the Django functionalities, django-bootstrap5	

Now our project is a little more substantial, we realize anyone can do anything as long as they're logged-in. Time might have come to add some tiny restrictions.

The user now needs an authorization to delete Tips if they want to... delete tips.


Activate the administration interface that comes with Django by default and use a superuser to modify the permissions and highlight the behavior of your project.

Of course, it takes more than just delete the "delete" button of the page (even though you could do this). Find the right way to restrain this functionality properly checking the permissions.

NB: Of course, the author of a Tip can delete their tip without any authorization. This will be the only exception to the rule.

# Chapter VIII

## Exercise 05

	Exercise
Exercise 05: Personalized authorization.	
Turn-in directory : <i>ex/</i>	
Files to turn in : Your project, your requirements.txt file	
Allowed functions : All the Django functionalities.	

Given the negative character of the downvote, its use should be reserved to specific users.


Implement the same control as the previous exercise to the downvotes. Since this is not a default permission and this would be a shame if you used another permission that would not be fit, create your own permission to restrain the use of the downvote!

Just like the previous exercise, it's not just about deleting the downvote button. You can try, right? Success is not guaranteed, though.

NB: like with a tip suppression, the author of a tip can always downvote their own tips if they believe it's, like, dumb or something.

# Chapter IX

## Exercise 06

	Exercise
Exercise 06: Automation and reputation	
Turn-in directory : <i>ex/</i>	
Files to turn in : <b>Your project, your requirements.txt file</b>	
Allowed functions : <b>All the Django functionalities, django-bootstrap*</b>	

Now we have most of the necessary tools to run this project, it would be a good thing not to have to manually manage those permissions. Some sites have started awarding privileges with a reputation scale. It gives the worthy users the rights to use new functions! This is what this exercise is all about. You will have to establish a reputation system depending on the votes the Tips posted by a user have received.

Replace the standard user class by your own user class. In this class, you will implement a way for the downvote and delete tips authorizations to depend on a user's reputation. This user's reputation will depend of the numbers of upvotes and downvotes their tips have received.

Each new user starts with 0 rep point. Each upvote on their tips will award them with 5 rep points. A downvote will take off 2. If a tip is deleted, the influence of its votes on its user's rep disappears.

A user will unlock their downvote authorization (on other tips than their own) with 15 rep points. It will take 30 rep points to be granted the authorization to delete tips. If the reputation of a user decreases, they can lose the authorizations their reputation had granted them.

You will also modify the appearance of the user's name, displaying their reputation on its side (between brackets) when it appears on your pages.

# Chapter X

## Submission and peer-evaluation

Turn in your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double check the names of your folders and files to ensure they are correct.



The evaluation process will happen on the computer of the evaluated group.