
POWER
CAPTAIN CONSOLE
Nintendo
B
A
SELECT START

Captain Console

A new retro gaming shop has emerged called **Captain Console**. The shop has one goal to become the greatest, biggest and most recognized retro gaming shop out there! They sell everything from ZX Spectrum to Gameboy but there is one problem.. they only have brick and mortar stores and no online presence whatsoever. This is where you and your group come in! Your job is to provide **Captain Console** a robust web application using state-of-the-art technologies such as **Django**, **Python** and **PostgreSQL**. They have already sent you the fax containing the assignment description - read carefully!

Assignment description

The demands for the assignment are categorized in two: **structural demands** and **featural demands**. Structural demands focus on how the application is structured, e.g. *code structure, database setup, etc.* Featural demands focus on what a user can do within the application, e.g. *a user can view a gaming console, etc.*

Here below are lists of featural- and structural demands and if those are implemented fully you and your group can receive a grade of 8,0. If you want to go the extra mile and receive a full score of 10,0, you and your group will need to add some features which are not specified below.

Featural demands

Here below is a list of featural demands:

- Layout site
    - Navigation bar
    - Footer
- Edit profile
    - Name
    - Image
- Catalogue site - lists available products in the store, e.g. *consoles, games, etc.*
    - Filter by type, e.g. *NES, Sega, Gameboy*
    - Search (*based on name*)
    - Order by
        - Price
        - Name
- Search history
- Product detail site - *shows more information about a certain product*
    - Name of product
    - Images
    - Long description

- Buy a product - *should work similar to a checkout phase in most online stores*
    - Contact information
        - Full name
        - Street name
        - House number
        - City
        - Country (*displayed as a <select> HTML element*)
        - Postal code
    - Payment step
        - Name of cardholder
        - Card number
        - Expiration date
        - CVC
    - Review step - *this is a read-only site where a user can review what he is buying and what information he has already typed in*
    - Confirmation - *this is the final step where a user gets a confirmation that everything is successful*
    - Easy navigation between steps - *it should be easy to navigate between the steps in the checkout phase*

Structural demands

Here below is a list of structural demands:

- The application must use a database to store the data
- Django Model API must be used
- The MTV pattern must be used
- Git is mandatory for version control and GitHub for repository
- All exceptions should be handled in a proper manner, which means that the application should not crash when an exception is thrown