Fire sale!

In this assignment we are going to create a mini-replica of the giant platform Craigslist. The main focus is on building a website which allows a user to sell items, and allow other users to send offers in order to buy the item

Tech stack

Our tech stack will consist of the following programming languages / frameworks / tools.

- Django (**Python** / **HTML** / **CSS** / **JS**) - Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design
- PostgreSQL - PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance
- PyCharm (https://www.jetbrains.com/pycharm/) - The Python IDE for Professional Developers
- DataGrip (https://www.jetbrains.com/datagrip/) - Meet DataGrip, our new database IDE that is tailored to suit the specific needs of professional SQL developers
  OR
  DBeaver (https://dbeaver.io/) - A free multi-platform database tool
- Git - A free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency

Grading

In order to receive a grade of up to 9,0 you are required to implement the following requirements. If the goal is to receive a higher grade than 9.0, you are going to need to be creative and implement your own requirements, which are not enlisted in the requirements below. All extra requirements are graded based on the following things:

- Time consumption
- Complexity
- Quality

Featural demands

Here below is a list of all featural demands:

- Authentication
  - Login
  - Register

- Layout site
  - Navigation bar
    - Profile image
    - Average rating
    - Logout
  - Footer
- Edit profile
  - Name
  - Bio
  - Image
- Catalog site - lists available items for auction
  - Search (*based on name*)
  - Order by
    - Price
    - Name
  - Each item should be clickable, which navigates to the item detail site
- Create an item
  - Name
  - Images
  - Condition
  - Long description
- Item detail site - *shows more information about a certain item*
  - Name
  - Condition
  - Images
  - Long description
  - Current highest offer
  - A list of similar items
  - A button which allows you to place an offer
- Placing an offer
  - All items which are available (they don't have an accepted offer) can receive an offer
  - Users can send multiple offers, until the owner of the item accepts an offer
  - All users who placed an offer on an item should be notified when an offer has been accepted. The one that made the accepting offer should be notified that his offer has been accepted, but all others should be notified that their offer was declined. Preference of notifications could be:
    - Email
    - Push notification
    - etc

- Checkout
    - The user with the accepting offer must checkout in order to finalize the sale of the item
    - There should be three stages of the checkout phase
        - Contact information
            - Full name
            - Street name
            - House number
            - City
            - Country (displayed as a <select> HTML element)
            - Postal code
        - Payment
            - Name of cardholder
            - Card number
            - Expiration date
            - CVC
        - (Optionally) Rate the seller
        - Review (a read-only site where a user can review what he is buying and what information he has already typed in)
            - Confirm
    - Easy navigation between steps - *it should be easy to navigate between the steps in the checkout*

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