YAAS Django project YAA_App application

Application URL

Application can be found at http://mikak.pythonanywhere.com/

Implemented requirements

List of tasks for this assignment round. Those tasks, which will be implemented later on so called leftover assignment are also in this list, but they are marked (implemented later).

List of implemented features:

Part 1

Part 2

1. Create user UC1
2. Edit user UC2 (**will be implemented later**)
3. Pid UC6
4. Ban auction UC7
5. Browse & Search API WS1
6. Pid API WS2 (**will be implemented later**)
7. Resolve auction UC 8
8. Soft deadlines for bidding OP2

Administrator credentials

Administrator username to database is **mkkvjk7**. Administrator password is **salasana**.

List of python packages

Django==1.11.6

requests==2.18.4

django-cron==0.5.0

django-mathfilters==0.4.0

djangorestframework==3.7.0

Database

YAAS_App application is using Sqlite data base. There are Auction, Pid and User model. User model is using Django's build in implementation.

Implementation descriptions

This implementation descriptions relates to implemented features.

UC1 Create user

From main page link create user leads to URL **url(r'^createuser/$',register_user**). View function register_user will first render user form as it is GET request and not POST request. User form is

named Registrationform and it will be rendered to template registration.html. User form is extended from Django's default user creation form by adding email field to form and template. After user has filled the form and template sends form with POST request to same url. Filled user form is validated and if validation is ok. User data will be saved to data base.

UC6 Pid

User activated add pid from main page by selecting auction and clicking add pid. This will send GET request with auction number as request parameter. User authentication status is first checked and redirected to login page if not authenticated. Next user is tested if he is administrator or seller and redirected to home with appropriate message if he/she is. As only normal authenticated users are allowed to pid. If user is ok form is rendered. In this case there is data from pid and auction tables so data from both database tables is rendered to form and to template. Data from auction table is presented to pidder. Pidder can change only pid value. Pid validity is checked by default validator. Time, when pid is placed will be checked. If pidding time exceeds the pid's end time, pid is not accepted. If pid is placed 5 minutes before pid's closing time the pid time will be extended by 5 minutes. All additional pids will extend this time by another 5 minutes until nobody adds pids within 5 minutes of closing time. If pidd is placed after the end time the pid is cancelled and auction status changed to D due to wait for resolve process. Form is send for further checking and saving POST request. Savepid view fuction will checked if form is valid. Then it will check if placed pid was incremented more than minimum 0.01 euros. If pid is not valid, user is redirected back to pidding page.

If pid is accepted, email is send(console) to seller and all the pidders.

UC7 Ban

Banned auction is selected at home page in same manner as in pid. Auction to be banned is parameter of GET request. User who bans the auction needs to be authenticated administrator. This is first checked and if authentication is ok, confirmban.html page is rendered with auction table. At confirmation page there is auction data and simple yes and no selection to confirmation. form is send as POST request to another banauction URL. POST request is processed by ban function. Selected auction's auction status value will be changed to 'B', means banned. When auction status is banned it is not shown in main page. It is not possible to pid banned auction. Also, automatic resolve process will not process it.

After auction status is changed the seller and all pidders are notified about ban by email.

UC8 Resolve auction

Auctions are resolved once a day by Django's management command. Management command needs to be fired by external scheduler in order to do tasks in automatically. This process is handled by task scheduler provided by Pythonanywhere site. It runs once a day resolveauction.py file in YAAS_App management directory. More detailed management command from task scheduler is following **/home/MikaK/.virtualenvs/YAAS-virtualenv/bin/python /home/MikaK/web-services-2017-project-part2-xzerkses/manage.py resolveauctions.**

Actual auction resolve is handled by Command class in resolveauction.py file, which is under YAAS_App/management/commands folder. Handle function reads periodically all auctions saved

to system and check auction status and auction ending time. If auction status is still Active and auctions ending time has passed, the auction will be resolved. Also, if auction's status is 'C',it will be resolved. In all these mentioned cases status of auction will be changed to D, which is for adjudicated. All involved pidders and seller is notified by email that auction is resolved.

```
Automatic auction resolving process starts.
**********************************************************
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
Successfully resolved auctions
**********************************************************************
Automatic auction resolving process ends.

2017-10-20 19:03:10 -- Completed task, took 65.00 seconds, return code was 0.
```

Picture 1 Pythonanywhere task scheluder log. All auctions have already been resolved.

WS1 Browse & Search API

Django's Rest framework was installed for this task. URL for browsing is api/auctions/$, which uses AuctionList class, which prosses GET method by simply selecting all auctions and then serializing them. Serialized auctions are, then send to client in json format or auctions are not found then 404 response is sent. Search is implemented by AuctionSearch class. Selected auction parameter transferred in GET request's URL, which is used to select correct auction. Selected auction is serialized and response to user sent back in json format.
Related to this Rest API implementation is serializer found in serializer.py, which serializes the auction model. Rest functions can be found in RestfulAPI.py and serializer is found in serializers.py file.

OP2: Soft deadlines for bidding

Soft deadlines are implemented by savepid function in views.py. As mentioned earlier before pid is saved the auction's end time is checked and if pid is placed closer than 5 min of closing time the auction end time will be extended by 5 minutes. This is done simply by adding 5 minutes to auction table's auction endtime value. Resolve process and other checks for ending time will rely on this value and therefore it will extent the pidding time.