# 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
3. Create a new auction UC3
4. Pid UC6
5. Ban auction UC7
6. Browse & Search API WS1
7. Pid API WS2 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 formis 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.

### UC2 Edit account information

Link to Edit User Data can be found at home page. User needs to login before link is visible. User can change password and email address from page. a Edit User Data link on home page leads to edituser method on views. Edituser method checks if user is authenticated and renders form on edituserdata.html page with password and email data, if request method is GET. User can change password and email on webpage. Changed password and email are sent back with POST method to same edituser URL. Sent form is handled and changes are saved to database by edituser method.

### UC3 Create a new auction

User can create a new auction from home page by pressing Create User link. Link is only visible if user is logged in. This login check has been done in template level. Link leads to AddAuction class, which renders createauction html page as request is a GET method.

Createauction page shows form, which checks that user fills required data to a new auction correctly. Filled form is checked that auction has more than 72 hours run and minimum price is at least 0.01. Also, Title and description are required to proceed. Submitted form is received in AddAuction class, which checks validity of POST method request (checks are already described). Form with all already filled data and additional auction status data and seller data and confirmation form are rendered as confirmation html page. This page shows user given data and asks for user confirmation to save auction. If user selects yes from radio button and submits data, then form is handled by saveauction method. If user selected NO then user is redirected to home page. If user selected YES, then data from html form is saved to database as a new auction. Each auction has fields for seller, title, description, start price, latest pid, auction end time and auction status. After the auction has been saved, confirmation email is sent to auction creator (seller) with html link to homepage. Html parts of message are not rendered as email is sent to console, but link is shown on console.

### UC4 Edit the description of an auction

Seller can edit auction description by pressing Edit Auction link from main page. a Link leads to editauction method, which checks if user is authenticated and if user is seller of item. GET-method is sent with id number of selected auction. User is redirected to login page or home page, if user is not authenticated or seller. Correct auction is found by help of id number in request parameter. If user is seller then auction lockedby status is checked and if auction is not locked by someone else, then auction is locked for edit and editauction.html page is rendered with auction data. A Seller can modify description of auction on html page and submit the page for saving to savechanges URL which leads to savechanges method. Correct auction is again filtered with help of offset parameter. Modified auction description from POST message is saved over older version in database. Also, auction is released from locked status before auction is saved. User is redirected to home page with message enabled for translation.

### UC5 Browse and search auctions

Auction browsing is implemented in browseauctions method, which lists all auction objects in order by title. Listed auctions are rendered with many other things to auctionslist.html page. Auctionlists.html page is shown as block on base.html template and together they form home page. Auctionlists.html page renders and shows each auction as a list which can be browsed by scrolling home page up and down.

Auctions on home page are shown if auction status is not B, which means that auction is banned.

Auction can be searched by its title by pressing Search Auction link on home page. The Link leads to search method. Search method checks if request has a query parameter or not. If parameter doesn’t exist, then filtering return empty list. Search method renders searchauction.html page with empty auctions list. Rendered page has input box for searching by title. User writes searched title and accepts by search button. Form is sent and received again by search method. Now query parameter exists and filters shows 10 closest matching auctions. User can end searching by clicking home page link.

### 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 to seller (console) and all the pidders.

### UC7 Ban

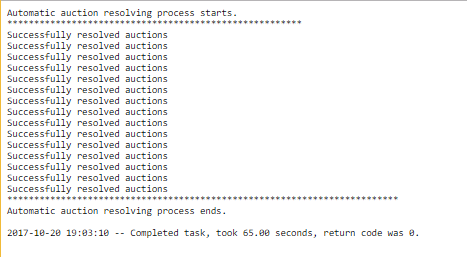
Banned auction is selected at home page in same manner as in pid. Auction to be banned is parameter of GET request. User who bans the auction needs to be authenticated administrator. This is first checked and if authentication is ok, confirmban.html page is rendered with auction table. At confirmation 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.



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

### UC9 Support for multiple languages

### UC10 Support for multiple concurrent sessions

Support for concurrent sessions has been implemented by using pessimistic locking. When user is placing a pid to an auction, the lockedby status of auction will be checked in addpid method. If auction is not locked by anybody, then user can place a pid and auction is locked for any edit during that time. Locking is done by writing user session key to lockedby field in auction model. Auction model lock will be released when a pid value is saved to database. This is done by writing lockedby field back to “” and saving auction model. Auction is locked in similar fashion, when seller edits auction description. Lockedby status will be checked in Editauction method before seller can modify auction description. If lockedby status is empty, a seller sessions key is saved to lockedby field in auction model and that auction is locked for editing. When seller has made changes to auction description and submitted form, the lockedby status will be again written to “” and saved to auction model and thus releasing a lock.

If user chooses to cancel placin a pid or cancel auction editing he/she must use application home link. Home link will handle that auction lock is released properly. This is done in clearhome method again by writing lockedby status to “” and saving model, just before redirecting back to home page.

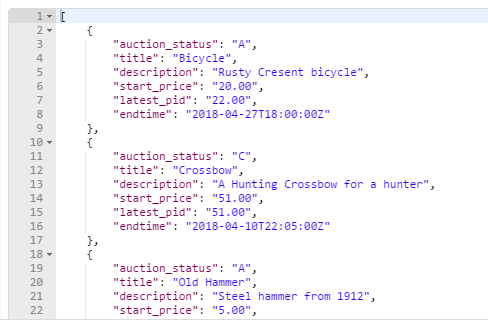
### UC11 Support for currency exchange

### WS1 Browse & Search API

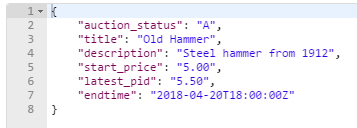
Django’s Rest framework was installed for this task. URL for browsing is api/auctions/$, which uses AuctionList class. AuctionList class handles GET method by simply selecting all auctions and then serializing them. Serialized auctions are, then send to client in json format or if auctions are not found then 404 response is sent. Search is implemented by AuctionSearchAndPid class. Selected auction parameter is transferred in GET request’s URL, which is then 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.

Settings.py contains global setting for default authentication classes and default permission classes, which have been set to TokenAuhentication and IsAuhenticatedOrReadOnly. This allows to use browse and search functions without authenticating the user.



Picture 2 Response to GET message (Browse) sent with postman. Sent message URI GET /api/auctions/



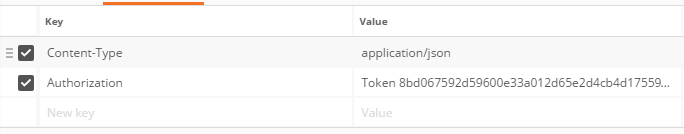
Picture 3 Response to GET message (Search) sent with postman. Sent message URI GET /api/auctions/2/

### WS2 Bid via API

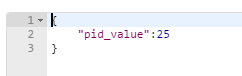
PID is added via API by using PUT method. Implementation can be found in AuctionSearchAndPid class in RestfulApi.py file. Parsing method uses additional permission and authentication class function decorators in order to restrict access for authenticated users. Authentication is done with token authentication. Method parse PUT message’s payload and makes same checks that was done with main pid function. After checks have been successfully verified pid is saved to database.



Picture 4 URI of PUT message (PID fuction)



Picture 5 PUT method Header content.



Picture 6 body of PUT method (pid value)

### 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.

### TR1 Database fixture and data generation program

Data generation program is populatedateabase.py under management/commands directory. program extends BaseCommand class and it is basically custom-made management command. It picks random username from 9 available and adds randomized numbers to picked names and uses 5 different email suffixes to create 50 users with emails and passwords. Passwords are 8 characters long randomized passwords. Sellers are picked from 50 users randomly, starting prices and items are selected randomly. There are 7 items where program can choose.

Program creates from 6 to 15 pids on some of the auctions and using available pidders.

Program can be started by running YAAS directory command **python manage.py populatedatabase**