Report for assignment 4

Note: stepan.pythonanywhere.com posted but it does not work (Connection error at HTTPSConnectionPool(host='openexchangerates.org', port=443))

1. Srboljub Stepanovic 39349

2. Requirements

Part 1

UC3: create auction
UC4: edit auction description
UC5: Browse & Search
UC11: currency exchange with https://currencylayer.com

Part 2

UC1: create user
UC2: edit user
UC6: bid
UC7: ban auction
WS1: Browse & Search API
WS2: bid API
UC8: Resolve auction
OP2: Soft deadlines for bidding

OP2 was implemented as a method in the Auction class and is invoked when bidding on an auction.

Part 3

UC9: language switching
OP3: store language preference
OP1: send seller auction link
UC10: concurrency
TR1: Database fixture and data generation program
TR2.1: Functional tests for UC3
TR2.2: Functional tests for UC6
TR2.3: Functional tests for UC10

OP1: the email message includes a link which would allow the user to see the created auction details without logging. Email message body implemented with template confirmemail.html which is populated with username, auction id and site domain in saveauction view invoked at confirming auction creation.

3. Python packages

Python 3.6.2
PyJWT 1.5.3
certifi 2017.7.27.1
chardet 3.0.4
django-common-helpers 0.9.1
django-cron 0.5.0
django-discover-runner 1.0

django-extensions 1.9.6
django-sslserver 0.20
djangorestframework 3.7.0
djangorestframework-jwt 1.11.0
idna 2.6
pip 9.0.1
pytz 2017.2
requests 2.18.4
rest-framework-auth0 0.4.5
setuptools 28.8.0
six 1.11.0
urllib3 1.22

4. Username and password

admin admin123

5. Session management

I implemented session management to keep track of the current user when trying to create a new auction. This is used when moving from the auction creation template to the confirmation template to be able to set the seller attribute of the new auction.
I also implemented session management to keep track of the current user when trying to bid on an auction. This is used when moving from the home template to the bid template to be able to set the users attribute of the new bid.
I implemented session management to save a user's language preference under LANGUAGE_SESSION_KEY when the user:

1. changes the language, from the request.POST parameter
2. logins, from the database
3. first time ever runs the application, to set English.

Locale Middleware then uses this LANGUAGE_SESSION_KEY to automatically activate language.

8. Concurrency problems in UC6

My web application does not allow any user to bid on an auction if the description of the auction has been changed by the seller. I implemented this using pessimistic locking with the flag lockedby in the Auction model. Every time a seller tries to edit the description in GET request, this flag is set to some value to indicate that bidding on the auction is locked. I used session key to set the flag but also a boolean value would get the job done because only the seller can change the description. When a bidder tries to bid on the auction, the logic will check the flag and refuse the bidder. When the seller confirms the update of the description in POST request, the lock is released by resetting the flag and bidding is allowed.
My web application also prevents that two bids on the same auction that have both prices higher then maximum current price can be saved in the database at the same time. I implemented this using pessimistic locking with the flag lockedbiddingby in the Auction model. Every time a bidder having a bid price higher than maximum current price tries to save the bid in POST request, this flag is set to some value to indicate that bidding on the auction is locked. I used session key to set the flag but also a boolean value would get the job done because only to this bidder is allowed to save bid at that moment. When another bidder tries to save bid on the same auction that has price also higher than maximum current price, the logic will check the flag and refuse the bidder. When the first bid is saved in the database, the lock is released by resetting the flag and bidding is allowed.

10. Functional tests

Tests for UC3(TR2.1)

#1 Number of auctions after and before creating a new auction. Difference of these numbers should be 1.

#2 Get request for creating an auction. Creationauction.html page should be rendered in response.

#3 Several Post requests for creating and saving an auction

-Post request for creating an auction by an anonymous user should be redirected to the login view.

-Post request for creating an auction by an authenticated user with proper data. Confirmauction.html page should be rendered in response.

-Post request for saving the auction (option=No) by the same user should be redirected to the home view without sending a email to the user
-Post request for saving the auction (option=Yes) by the same user should be redirected to the home view and a email to the user should be sent.

#4 Post request for creating an auction by an authenticated user with zero minimum price. Createauction.html page should be rendered in response with a message.

#5 Post request for creating an auction by an authenticated user with expired deadline. Createauction.html page should be rendered in response with a message.

#6 Post request for saving an auction (option=Yes) by a user whose id is not stored in the current session. It should be redirected to the createauction view.

Tests for UC6(TR2.2)

#7 Number of bids after and before creating a new bid. Difference of these numbers should be 1.

#8 Tests for creating and saving a bid

- Get request for creating a new bid. Bid.html page should be rendered in response.
- Post request for saving the bid with wrong auction id by the same user should be redirected to the home view without sending a email to the users
- Post request for saving the bid with proper price by the same user should be redirected to the home view with sending a email to the users
- Post request for saving the bid with price less than the maximum price by the same user should be redirected to the bid view
- Post request for saving the bid with price that is string by the same user should be redirected to the bid view

#9 Get request for creating a new bid by the seller of an auction. It should be redirected to the home view.

#10 Get request for creating a new bid for an auction that is not active. It should be redirected to the home view.

#11 Post request for saving a bid by a user whose id is not stored in the current session. It should be redirected to the showauction view.

Tests for UC10(TR2.3)

#12 The seller changes description of an auction. Another user tries to bid on the same auction. The bidder should be redirected to the showauction view. The seller then saves new description and is redirected to the home view. The bidder tries to bid again and bid.html is rendered in the response enabling him to bid.

#13 Two bidders try to save their bids on the same auction at the same time. The first higher bid will pass and the first bidder will be redirected to the home view. The second lower bid will not pass and the second bidder will be redirected to the bid view. The concurrency scenario will not occur.

11. Language switching mechanism

Language switching mechanism consists of two parts: template and view.
I added in each template a form with a html select element *language* from where a user will select desired language and a hidden input field *next* that has the value of the redirect_to context variable. This variable tells Django the URL of the page to which the user will be redirected.
I used existing view django.views.i18n.set_language(), that sets a user's language preference and redirects to a given URL or, by default, back to the previous page. The view is called via the POST method with a *language* parameter set in request. The view saves the language choice in the user's session under LANGUAGE_SESSION_KEY. The view also takes the *next* parameter from the POST data and redirects the user to that URL.
The Locale Middleware uses this LANGUAGE_SESSION_KEY to automatically set the language without a need to use *activate* function. But because this session key will be established only after the user first time ever changes the language, I added in the home *auction* view the code that establishes a new session language key and activates English.

OP3: store language preference

First, I extended User class with my Profile class that have one additional field for storing the language. Along to this class I defined two signals so my Profile model will be automatically created/updated when users create/update User instances.
I also defined another signal that will activate language according to the Profile field language from database whenever user logins.
Finally, I added a piece of code in django.views.i18n.set_language() that saves user's language preference to the database whenever a logged in user changes it.

12. Location of my data generation program is:
assign4_stepanovicApp/management/commands/data_generation_program.py

It can be invoked from command terminal by the command:
python manage.py data_generation_program