List of implemented features

- UC1 create new account
- UC3 create new auction
- TR2.1 automated test for UC3
- TR1 DB fixtures and data generation program
- UC5 browse and search
- WS1 browse and search API for web service
- UC7 ban auction
- UC4 edit auction
- UC2 edit user info
- UC8 resolve auction
- UC9 support multiple language
- UC6 bid
- WS2 bidding API for web server
- TR2.2 automated test for bidding
- UC10 support multiple concurrent sessions
- TR2.3 automated test for concurrency when bidding

Optional features:

- Store user language in the database

Python and Django version

- Python version 2.7.5
- Django version 1.5.4

Admin user for the DB

There is no default admin user, “python manage.py createsuperuser” must be run in order to populate the admin user.

Session management implementation

The session is implemented using the default mechanism provided by Django. This mechanism stores the session data in django_session table and then set a cookie named “sessionid” to identify each session. Inside django_session table, the data is encrypted using SECRET_KEY setting for security purpose.

Auction confirmation form implementation

The confirmation form in UC3 is implemented using 2 different views. Firstly, when the user enters information into auction.views.CreateView and submit, a confirmation view is display

(auction.views.ConfirmView) instead of directly creating the auction. This confirmation view contains another form holding the information entered by the user previously in the create view. The user is now presented with 2 choices, yes or no. If they choose yes, the confirmation view creates the new auction. Otherwise, no new record is created.

Resolving bids

In UC8, resolving auctions are initiated automatically using a library called celery. This library basically provides a task queue mechanism. When an auction is created, a scheduled background task is created and will be exactly executed at the specific deadline provided by the user. Celery has many backend options, one of them is to use the database using by django. Task data is stored in the database and celery will execute them.

Concurrency control

In order to avoid conflicts between sessions when bidding, the auction table has 2 columns called version and bid_version. The version column keeps track of the current data state of the auction itself. Whenever the description of auction is changed the version field will be updated (increased by one). The bid_version column tracks the current state of bidding, whenever a new bid is placed successfully, the bid_version will be increased by one. In order to verify both auction state and bidding state, the auction view will include 2 hidden fields (version and bid_version) and send them to the server whenever a bid is placed. The server then checks the sent versions against the current versions of the auction to prevent the user from placing bid based on an out-of-date version of the auction

REST API

The API does not fully follow RESTful style. It makes use of the HTTP methods, but it does not follow the URI convention of REST. There are 2 APIs for others to consume.

API for searching auction
URI: /api/search
Sample request represented by cURL
curl http://localhost/api/search\?query\=test
Sample response
{"data": [{"description": "Auction number 1 description", "min_price": 100.0, "deadline": "2013-10-31 21:26:53", "id": 1, "title": "Auction number1", “version”:1, “bid_version”:1}]}

API for placing new bid
URI: /api/bid
Sample request represented by cURL
curl -X POST -H "Api-User: username" -H "Api-User-Pass: password" http://localhost/api/bid -d "auction_id=10&amount=150&version=2&bid_version=5"

Sample successful response

{"data": {"description": "Auction number 1 description", "min_price": 10.0, "deadline": "2013-10-31 15:39:02", "id": 10, "title": "Auction number 1", “version”:2, “bid_version”:6}}

Function tests

The functional tests cover create new auction and place new bids. Those 2 actions are covered because they are the fundamental features.

In create new auction feature, the test covers the following cases
- Accessing the create auction page without authenticating results in a redirect to the login page
- Creating new auction with all valid details shows the confirmation form
- Creating new auction with some valid data such as negative min price or deadline in the past displays the form for refilling and showing errors
- Confirming new auction creates new record in the database and redirect to the auction page
- Failing to confirm new auction does not create any new record

In bid auction feature, the following cases are covered
- Successfully placing a bid creates new record in the database and updates the bid_version
- Using invalid bid amount (less than the current highest bid or less than 0.01 from the highest bid) shows error and does not create new bid record in the database
- Seller can not bid
- Finished auctions can not receive new bids
- version and bid_version are always checked to avoid conflicts between concurrent sessions

Multi-language implementation

Multi-language is implemented using Django internationalization and localization library. In order to switch languages, the application provide an URI in form of /set-language/en or /set-language/vn. Behind the scene, the language code is stored into the session with the key django_language because django will look into session for the key django_language for the language setting