Yet Another Auction System (YaaS)

- Project specification -

1 Introduction

The objective of the course project is to develop the YAAS Web application as described below. If you have any questions about the interpretation of this document please contact the course lecturers.

This document includes many requirements for your project. The grade for the project is based on how many of the requirements below you implement successfully. The design decision and how business logic is implemented is for you to decide.

1.1 It is about the Brains, not the Looks

The focus of the project is to develop the business logic (server-side/backend part) and not about designing web pages. Your application should be usable, but you should not allocate too much time to make it look nice. As an example of a functional but simple looking site, check craigslist.org. Your web application does not need to look better than that.

1.2 Individual Work

You should work on your project alone. Discussing the course tasks and specific issues with other participants is allowed and even encouraged. Teamwork, pair programming or borrowing code from other persons it is not allowed. Such things can be easily spotted when checking your projects.

2 YAAS: Yet Another Auction Site

YAAS is a web application and a web service to create and participate in auctions. Examples of popular auction sites include ebay.com and huuto.net.

An auction site is a good example of a service offered as a web application. The organization owning an auction site does not buy or sell anything. Instead, it creates a community of users interested in buying or selling the most diverse items and provides its members with the tools to communicate and interact in a convenient, fast and easy way.

The YAAS application has the following types of users: *anonymous*, *registered*, and *admin*.

Constraint: For convenience, always set the admin password as **admin.**

The YAAS application should implement the following use cases:

3.1 UC1 Create a user account

REQ1.1 Any anonymous user can create a new user account using username, email and password.

Constraint: We assume that administrator accounts are created using the Django admin interface. For this, you must enable the Django admin interface in your project.

3.2 UC2 Edit account information

REQ2.2 A logged in user can change his/her email address

REQ2.3 A logged in user can change his/her password.

3.3 UC3 Create a new auction

REQ3.1: Any registered user can create a new auction.

REQ3.2: When creating an auction, the system registers the following information:

• The user that creates the auction (this user is also called the seller).

• The title of the auction.

• The description of the item(s) to sale.

• The minimum price. (Other users should bid an amount higher than the minimum price.)

• The deadline: the end date and time for the auction. The minimum duration of an auction is 72 hours from the moment it is created. **The expected format for the date and time must be set to `%d.%m.%Y %H:%M:%S` (e.g., 22.12.2017 13:42:00).**

REQ3.3: the user must be asked for a confirmation before creating a new auction.

REQ3.3.1: a confirmed auction will be stored in the system

REQ3.3.1: an unconfirmed auction will NOT be stored in the system

REQ3.4: The system must confirm by email to the seller that an auction has been created.

REQ3.5 The email message includes a special link which would allow the seller to modify the description of the created auction without logging in.

REQ3.6 A user (other than the seller) can bid on an active auction.

The lifecycle of an auction is as follows:

• Active: Once an action has been created, it becomes active until its deadline or until it is

banned.

• Banned: An active auction can be banned by an administrator as described by UC7.

• Due: After the auction end date, the auction is due for adjudication.

• Adjudicated (see UC8): A due action has been processed by the system and the winner has been selected from all the bidders.

REQ3.7 The seller can update the description of the auction.

An example of how the UC3 can be implemented is as follows:

1. The user selects the create new auction link (or similar)

2. A form with all the required auction fields is shown. The user fills in the form and submits it. If the user enters incorrect information in the form the system should show a proper error message and show the form again.

3. A confirmation message is shown, asking if the user is sure to create a new auction. The user can select Yes or No.

• if the user selects Yes in the confirmation form: The new auction is recorded in the system.

• if the user selects No in the confirmation form: The new auction is not recorded in the system.

• The user never answers the confirmation form: The new auction is never recorded in the system.

Technical Requirements for Sending emails: console backend must be used for sending emails. https://docs.djangoproject.com/en/2.2/topics/email/#console-backend

3.4 UC4 Edit the description of an auction

REQ4.1 The seller of an active auction can change the description of the item(s) for sale.

3.5 UC5 Browse and search auctions

REQ5.1 anonymous users must be able to browse the list of active auctions

REQ5.2 anonymous users must be able to search auctions by *title*.

REQ5.3 registered users must be able to browse the list of active auctions

REQ5.4 registered users must be able to search auctions by *title*.

3.6 UC6 Bid

REQ6.1 Any registered user can bid for an active auction, except the seller.

REQ6.2 The minimum bid increment is 0.01. Only two decimal places are considered when bidding.

REQ6.3 A seller cannot bid on own auctions.

REQ6.4 The application must show to the user the most recent description of the auction before accepting bids from the user.

REQ6.5 A new bid should be greater than any previous bid and the minimum price.

REQ6.6 A bidder cannot bid on an inactive auctions.

REQ6.7 the seller must be notified by email that a new bid has been registered

REQ6.8 the latest bidder must be notified by email when a new bid has been placed

3.7 UC7 Ban an auction

REQ 7.1 An administrator user can ban an active auction that does not comply with the usage terms of the site.

REQ 7.2 A banned auction is not deleted from the system.

REQ 7.3 The seller and all bidders are notified by email that the auction has been banned.

REQ 7.4 It is not possible to bid on a banned auction.

REQ 7.5 Banned auctions are not shown in the list of auctions neither in search results.

REQ 7.6 Banned auctions are not resolved.

REQ 7.7 Admin should be able to see the list of banned auctions**.**

Constraint1: You are not allowed to use the Django Admin Interface for banning.

Constraint2: the user account for administrator and its password should be **'admin'** and **'admin'**

3.8 UC8 Resolve auction

After the end date of the auction, the system should resolve the auction by electing the winner. The winning bid is the bid with the largest offer.

REQ8.1 This resolving of auctions is triggered via a GET request to /auction/resolve, which can be invoked manually by a user or by an external scheduler.

REQ8.2 All the bidders are notified by email that the auction has been resolved.

REQ8.3 The seller notified by email that the auction has been resolved.

3.9 UC9 Support for multiple languages

The web application must support multiple languages.

REQ9.1 all users, including anonymous users, must be able to choose their favorite language. For practical purposes having two languages available is enough.

REQ9.2 the application must remember the language preference during the user session.

REQ9.3 the application must store the language preference permanently for registered users.

OBS: This does not mean that you need to translate all the messages and templates to many languages, but you need to create application messages and templates in English. It is sufficient that only a few words are translated into a different language when the user changes the language. For this you need to implement the necessary application logic to find the right message or template for the user's language.

3.10 UC10 Support Multiple Concurrent Sessions

The YAAS Web Application must support multiple concurrent users but you can assume that there is only one application server process.

REQ10.1 The web application should be able to handle multiple concurrent user sessions.

For example, the seller can change the description of an auction (UC4) but there is a requirement stating that the YAAS Web application must show to the user the most recent description of the auction before accepting bids from the user. Another example is that there can be several concurrent users bidding on the same auction, the web application must make sure that a new bid should be always greater than any previous bids on the same auction.

You need to ensure that the application behaves as expected even when it is serving multiple concurrent user sessions.

Hint: You can use optimistic locking mechanism to handle concurrency.

3.11 UC11 Support for currency exchange

REQ11.1 All the auctions are created by default in EUROs.

REQ11.2 A user should be able to visualize the prices of the auctions and bids in a different currency based on the latest exchange rate.

REQ11.3 The exchange rate for the currency should be fetched when needed from sites like https://currencylayer.com (free accounts available) and http://fixer.io/ .

3.12. RESTFul web service

You must develop a RESTful web service for the YAAS system implementing an API for browsing, searching, and bidding on an auction.

REQ12.1 - A user should be able to send a GET request for browsing the auctions and fetching a specific auction by ID.

REQ12.2 - A user should be able to send a GET request along with a search string parameter to search auctions by their titles.

Constraint: An auction or a list of auctions should be sent back in the JSON format.

REQ12.3 a registered user can bid on a given auction by sending a POST request with the necessary information attached in the json format. (similar to UC6 - bidding on an auction).

REQ12.3.1 The web service should be able to verify the bidder's credentials

REQ12.3.2 The web service should be able to verify bid validity

REQ12.3.3 The web service should respond depending on the result of the bidding with a message containing the description of the new bid or a description of the error also in json format.

When possible you should share program code with UC6. Also, you can assume that users will create user accounts and new auctions using the YAAS application.

4. Additional requirements

4.1 Testing

TREQ4.1.1 You must create automated functional tests for REQ9.3 - store language permanently

TREQ4.1.2 You must create automated functional tests for REQ3.5 - auction link

Constraint: These tests should be implemented in Python using the *django.test* module.

4.2 Test data generation program

TREQ4.2 write a **data generation program**, which should populate the database with at least 50 users, 50 auctions and bids for some of these auctions. The data generation programme should be accessible via URL in the format ***myapp.com/generatedata***

5 Technical Requirements

All the software necessary to carry out the project is distributed as open source and available for the most popular operating systems.

- Python 3 www.python.org
- Django 2.2 www.djangoproject.org (this is a strong requirement, older versions not allowed.!!!!)
- Git - https://git-scm.com/
- Git client - many available, https://desktop.github.com/ recommended

*IMPORTANT !!!: Both the admin username and password for your web application must be set to "**admin**". You can disable the password validators if django does not allow to create accounts with simple passwords. Read more here.*

We recommend PyCharm, Professional Edition editor www.jetbrains.com/pycharm/. You can obtain free license at https://www.jetbrains.com/student/.

You can use your own computer or the computers in the K126 computer class in the basement of the Agora building. Login with your Åbo Akademi username and password. The computers have all the necessary software tools already installed.

Your implementation must follow the separation of the application logic from its presentation by using consistently a template mechanism (the Django's template mechanism).

6. Deliverables

Your project should be versioned and delivered using github.com using the following link https://classroom.github.com/a/Dq4lg42o**. The repository is private, but lecturers have access.**

A report has to be included in the root folder as readme file. The report must be a text (e.g., a Markdown) or PDF document. A Word document is **NOT** allowed.

The report should contain:

1. Your full name and student number at Åbo Akademi.
2. List of implemented requirements (that successfully passed the tests and do not crash in the browser)
3. Which browser you used to test the application
4. List of Python packages used beside django and their version (should be identical to the ones in *requirements.txt* in the project source folder)

6.1 Submission guidelines

- All the code must be submitted via github. However you should submit the link to your github repository in Moodle.
- Automated tests will be used to check your project. The tests are provided to you from the beginning. For the requirements for which the tests pass, the lecturers will also inspect the code and run the application in the browser.

6.2 Using git

1. Download and install Git from https://git-scm.com/downloads
2. Create a user account on Github
3. Use the Repository Invitation link (which will be provided by the teachers) to create a private repository for the project on Github
4. You can create a new repository on your local computer by entering the following commands in a command line:

   $ git init
   $ git remote add origin

   https://github.com/it-teaching-abo-akademi/yaas-project-2019-<yourusername>.git

   $ git pull origin master
   $ git add .
   $ git commit -m "some message"
   $ git push

   Or push an existing repository from your local computer

   $ git remote add origin
   https://github.com/it-teaching-abo-akademi/yaas-project-2019-<yourusername>.git
   $ git pull origin master
   $ git push

7. Grading

Each use case gives a number of points based on its complexity. The maximum number of points is 30.

- UC1: create user 1
- UC2: edit user 1
- UC3: create auction 3
- UC4: edit auction description 1
- UC5: Browse & Search 1
- UC6: bid 3
- UC7: ban auction 1
- UC8: resolve auction 2
- UC9: language switching 2
- UC10: concurrency 2
- UC11: currency exchange 2
- WS1: Browse & Search API 2
- WS2: Bid api 2

- REQ9.3: store language preference (no test provided) 1
- REQ3.5: send seller auction link (no test provided) 1
- TREQ4.1.1 test REQ9.3 verified manually 2
- TREQ4.1.2 test for REQ3.5 verified manually 2
- TREQ4.2: implement data generation program (no test provided) 2

Total points: 31
Max points 30
Min points to pass 15.

Final grade:
0 - 14 - grade 0
15-17 - grade 1
18-20 - grade 2
21-23 - grade 3
24-26 - grade 4
27-30 - grade 5