Development of Web Services

- Course Project Requirements-

September 2017

1 Introduction

This document describes the project work for the course Development of Web Applications and Web Services.

The objective of the course project is to develop the YAAS Web application and the YAAS Web service 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. All these requirements are compulsory in order to get the maximum grade. A table at the end of the document describes which requirements have to be fulfilled for different grades. At the same time, some important decisions about the project are left open and you have a lot of freedom to decide how to design and implement many aspects of your project. Be creative and implement the kind of system that you would like to use and show to others.

1.1 It is about the Brains, not the Looks

This course is about developing web applications 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.

1.3 Materials and Equipment

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

- Python 3.6.2 www.python.org
- Django 1.11.5 www.djangoproject.org
- Git - https://git-scm.com/

- Git client - many available, https://desktop.github.com/ recommended
- PyCharm, Professional Edition editor www.jetbrains.com/pycharm/ (licence in Moodle)

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.

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.

In the rest of this section, we describe the main requirements for the YAAS web application and web service.

3 The YAAS Application

You must develop the YAAS web application that implements an web-based auction site as described below. The YAAS application should implement the following use cases:

You are free to design how the user interacts with your application, as long as it follows the descriptions and rules below. You can add more features to your application if you wish, but you should make sure that all the required tasks are completed.

3.1 UC1 Create an user account

Any anonymous user can create a new regular user account.

We assume that administrator user 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

A logged in user can change his/her email address and password.

3.3 UC3 Create a new auction

Any registered user can create a new auction. 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 should be clearly specified in the above deadline input field.

There is a special requirement for UC3: the user must be asked for a confirmation before creating a new auction.

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

Optional feature (OP1): the email message includes a link which would allow the user to see the created auction details without logging.

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. While an auction is active:

A user (not the seller) can bid on the auction. An administrator can ban the auction. The seller can update the description of the auction.

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

An example of how the UC3 can be implemented 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.

3.4 UC4 Edit the description of an auction

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

3.5 UC5 Browse and search auctions

Both anonymous and registered users must be able to browse the list of auctions and search auctions by title.

3.6 UC6 Bid

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

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

• A seller cannot bid on own auctions.

• The YAAS Web application must show to the user the most recent description of the auction before accepting bids from the user.

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

• A bidder cannot bid on an auction that is already winning.

• The following users must be notified by email that a new bid has been registered: the seller the bidder that has placed the new bid the previous bidder (if any)

Optional feature (OP2): Soft deadlines. If a user bids at an auction within five minutes of the auction deadline, the auction deadline is extended automatically for an additional five minutes. This allows other users to place new bids.

3.7 UC7 Ban an auction

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

• A banned auction is not deleted from the system.

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

• It is not possible to bid on a banned auction.

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

• Banned auctions are not resolved.

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

3.8 UC8 Resolve auction

After the end date of an auction the system should automatically resolve the auction and elect the winner. The winning bid is the bid with the largest offer. Resolving auctions must be initiated automatically by the system and not by a user.

All the bidders and the seller are notified by email that the auction has been resolved.

3.9 UC9 Support for multiple languages

The web application must support multiple languages. This does not mean that you need to translate all the messages and templates to many languages, but you must implement:

• A mechanism for the user to choose the favorite language.

• The necessary application logic to find the right message or template for the user's language.

You need to create application messages and templates in English. It is sufficient that only a few words are translated in a different language when the user changes the language.

All users, including anonymous users, may be able to choose their favourite language and the application must remember the language preference during the user session.

Optional feature (OP3): Store language preference permanently for registered users.

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.

Still you need to study how the different users sessions can interact. 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 a new bid should be always greater than any previous bid in the same auction.

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

3.11 UC11 Support for currency exchange

All the auctions are created in EUROs. However the user can decide to visualize the prices of the currency in a different currency which is calculated every time based on the latest exchange rate. The

exchange rate for the currency should be fetched regularly (on each request or at interval) from sites like https://currencylayer.com (free accounts available) and http://fixer.io/ .

3.12.1 WS1 browse and search via API

You must develop a RESTful web service for the YAAS system implementing an API for the use cases UC5 - browsing and searching.

A user should be able to send a GET request for either browsing the auctions or for searching a specific auction. A auction or list of auctions should be sent back in the json format.

3.12.2 WS2 Bid via API

A web service which allows an authenticated user to 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. The web service should be able to verify the bidder's credentials and bid validity, and 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.

OBS: You do not need to implement a service for the other use cases.

3.13 TR1: Database Fixture for functional testing

You must create a **database fixture** that should be used when executing your functional tests. To populate the database you should write a **data generation program**, which should populate the database with at least 50 users, 50 auctions and bids for some of these auctions.

3.14 TR2 Automated Functional Tests

You must create automated functional tests for the use cases UC3 (TR2.1), UC6 (TR2.2) and UC10 (TR2.3). These tests should be implemented in Python using the *django.test* module.

4 Technical Requirements

4.1 Django

You must implement your application using the Python programming language and the Django framework. During the course lectures and laboratories we will discuss practical advice on how to implement the project tasks using Django.

4.2 Clear Separation of Presentation and Application Logic

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

5. Deliverables

The project should have always two artifacts: the **source code** and **a report**.

5.1 Source code

The project folder should include the following:
- source code of your django web application
- a **requirements** file to specify which version of different libraries you have used.
- the **email backend** in the project settings should be set to:
django.core.mail.backends.console.EmailBackend
- all **html pages should be properly connected** so one can navigate easily in the application

5.2 Report

You must create a report where you describe clearly and concisely the following information about your project (where applicable based on the implemented project requirements):

1. List of implemented requirements . Only the requirements in the list will be graded. Do not forget to include which optional features you implemented and how.

2. list of Python packages used beside django and their version (should be identical to the one in *requirement.txt* in the project source folder)

3. Mention the admin username and password for your DB

4. How does your application implement session management.

5. How do you implement the confirmation form in UC3.

6. UC8 Resolving bids must be initiated automatically (and not by a

user). How do you initiate this?

7. How do you avoid possible concurrency problems, specially in UC6?

8. Present the REST API for your service. Is your service RESTful? What resources do you expose over the

API and what is the URI to access them. Include concrete examples of HTTP requests and responses for bidding in an auction.

9. A description of your functional tests, what are you testing and why.

10. How have you implemented the language switching mechanism?

11. Specify the location data generation program and how it can be used (invoked)

NOTE!: clear and concise information helps the teachers to NOT overlook your contribution.