Assessment Brief

Module title	Web Services and Web Data
Module code	COMP3011/XJCO3011
Assignment title	News Aggregator
Assignment type and description	This is a programming coursework. You will use the Django web framework to implement a RESTful web API for a news agency. The API should adhere to exact specifications set out in this document. You will also write a simple news aggregator application for collecting news from all the APIs you and your colleagues in this module will have developed as part of this coursework.
Rationale	This coursework will give you the opportunity to acquire the knowledge and master the skills required to develop and implement RETSful web services.
Word limit and guidance	In addition to the repository of your server-side and aggregator code, you will submit a brief report, not exceeding 1000 words, that clearly describes your implementation.
Weighting	This coursework is worth 30% of the total module mark
Submission deadline	22 March 2024
Submission method	Electronic
Feedback provision	Electronic
Learning outcomes assessed	When you complete this coursework, you will: Understand the principles of RESTful APIs. Learn how RESTful API are specified. Acquire skills in implementing RESTful APIs. Acquire skills in writing client software that interacts with RESTful APIs. Acquire skills in deploying web services on the WWW. Acquire skills in implementing and programming APIs using a web development platform.

Module leader	M. A. ALSALKA
Other Staff contact	

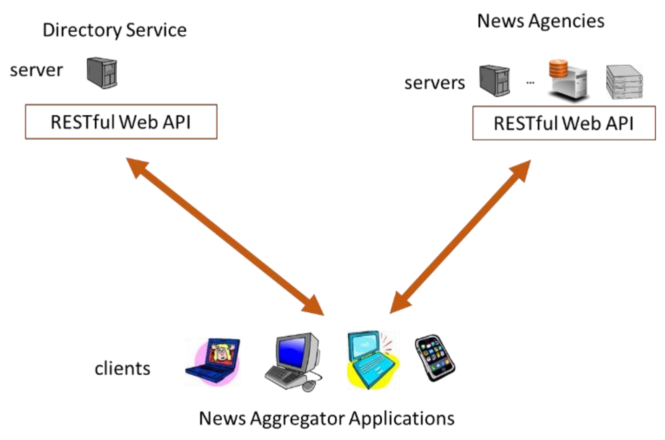
1. Assignment guidance

In this coursework, you will use the Django web framework to develop, implement, and deploy a RESTful web API for a news agency. The, you will host your API on www.pythonanywhere.com (for free). You will also write a simple news aggregator for collecting news from any API that adheres to the specifications set out in this brief. You will then test the aggregator by collecting news stories from all the web sites that your colleagues in this module have deployed.

2. Assessment tasks

News agencies often form alliances to promote cooperation and information exchange. For example, many European news agencies are members of the European Alliance of News Agencies https://www.newsalliance.org/. This coursework is based on role play where each student will create their own news agency and join an alliance with other news agencies (students). One of the purposes of this alliance is to allow client applications (news readers) to collect news from participating agencies using a standard API that all agencies can provide.

The overall architecture of a system that utilizes this alliance to collate news is shown in the following figure. Client applications sitting on users' devices communicate with the news agencies to collect stories. In addition to news agencies, a central directory service is needed to maintain a list of all participating agencies and their URLs. A news aggregator application will first contact the directory to obtain the list of participating agencies, then it will contact each news web site in turn to collect news stories.



The following sections define the exact specifications of a RESTful web interface that all news web site should offer.

What data is held?

Every news agency should maintain a table of authors that can post news to the website. For each author, the server must maintain a name (one string), a unique username, and a password. Only authors registered in this table are allowed to post news stories to the service. Authors are added to this table manually through the admin site of the web API. The admin site is automatically created by the Django framework. At least two authors must be added to this table, i.e. one for the student who developed the service, and one for the instructor. More details about the teacher's account are given at the end of this document.

Every server should also maintain a table for news stories containing the following data:

- A unique key (autogenerated)
- The story headline (limited to 64 characters)
- The story category, which can be one of the following: pol (for politics), art, tech (for technology new), or trivia (for trivial news, e.g. Tom has lost his red socks).
- The story region which can be one of the following: uk, eu (for European news), or w (for world news)
- The story's author
- The story date
- The story details (limited to 128 characters)

Specifications of the API

Except for two cases, we will use **application/json** objects to represent data in the payload of HTTP messages. When we specifically mention an HTTP response code in the API specification, the server should use this code, otherwise it should respond with one of the standard HTTP status codes as appropriate to the situation (e.g. 400 Bad Request). Here is a list of the services that every news API must provide:

Log In

Service Aim: to log in to an author's account in order to be able to post, or delete news stories.
Service Specifications:
The client sends a **POST** request to **/api/login** with the following data in an **application/x-www-form-urlencoded** payload with two items:

1. Username ("username", string)
2. Password ("password", string)

If the request is processed successfully, the server responds with **200 OK** and a **text/plain** payload giving some welcome message.
If login fails, the server should respond with the appropriate status code and an explanation (if any) in a **text/plain** payload.

Log Out

Service Aim: to log out from an author's account
Service Specifications:
The client sends a **POST** request to **/api/logout** with no payload.
If the request is processed successfully, the server responds with **200 OK** and a **text/plain** payload with some goodbye message.
If logout fails, the server should respond with the appropriate status code and an explanation (if any) in a **text/plain** payload.

Post a Story

Service Aim: to post a news story
Pre-condition: the user must be logged in before posting stories
Service Specifications:
The client sends a **POST** request to **/api/stories** with the following data in a JSON payload. JSON 'key' names and value types are given in parenthesis:

1. Story headline ("headline", string)
2. Story category ("category", string)
3. Story region ("region", string)
4. Story details ("details", string)

Upon receipt of this request the server checks that the user is logged in, and if they are the story is added to the stories table (with the name of the logged in user) and a time stamp. The server responds with **201 CREATED**.
If the story cannot be added for any reason (e.g. unauthenticated author), the server should respond with **503 Service Unavailable** with **text/plain** payload giving reason.

Get Stories
Service Aim: to get news stories
Service Specifications:
The client sends a **GET** request to **/api/stories** with the following data in an **application/x-www-form-urlencoded** payload with 3 items:

1. Story category ("story_cat", string). For any category this should be "*".
2. Story region ("story_region", string). For any region this should be "*".
3. Story date ("story_date", string). For any date this should be "*".

When the request is received the server retrieves all stories, having the given category and region published *at or after the given date*. If the request is processed successfully, the server responds with **200 OK** and a list of stories in a JSON payload ("stories", array). For each story in the list, the following data must be provided:

1. The story's unique key ("key", string)
2. The story headline ("headline", string)
3. The story category ("story_cat" , string)
4. The story region ("story_region", string)
5. The story author name ("author", string)
6. The story date ("story_date", string)
7. The story details ("story_details", string)

If no stories are found, the server should respond with **404** status code with **text/plain** payload giving more information.

Delete Story
Service Aim: delete a story
Pre-condition: user must be logged in
Service Specifications:
The client sends a **DELETE** request to **/api/stories/key**, where /**key** is the story key.
If the request is processed successfully, the server responds with **200 OK.**
If the server is unable to process the request for any reason, the server should respond with a **503 Service Unavailable** with **text/plain** payload giving reason.

Specifications of the directory service

The module staff will implement the directory service. It will be hosted at directory.pythonanywhere.com. Once you have implemented your news service, you should register yourself to this directory using the directory's /api/register service.

The directory service is very simple and maintains a list of all participating agencies. For each agency the following data is maintained:

- The name of the agency. This should be derived from the name of the student, for example 'Jane Doe News Agency'.
- The address of the company's website. Since you will be hosting your websites at www.pythonanywhere.com, this will be something like 'xyz.pyhtonanywhere.com' where xyz is your university username.
- A unique three-letter code to uniquely identify this agency. This code should be derived from the initials of the student and their year of birth, e.g. for Jane Adams Doe born in 2005, it would be JAD05.

The directory's API
The following services are provided by the directory:

Register
Service Aim: to register a news agency in the directory
Service Specifications:

The client sends a **POST** request to **/api/directory/** with the following data in a JSON payload:

1. The name of the news agency (“agency_name”, string)
2. The URL of the agency’s website (“url” , string)
3. The unique code of the agency (“agency_code”, string)

If the request is processed successfully, the server responds with **201 CREATED** with no payload.
If the server is unable to process the request for any reason, the server should respond with a **503 Service Unavailable** with **text/plain** payload giving reason.

List
Service Aim: to get a list of all agencies in the directory
Service Details:
The client sends a **GET** request to **/api/directory/** with an empty payload.
If the request is processed successfully, the server responds with **200 OK** and the list of agencies ("agency_list", array) in a JSON payload. For each agency in the list, the following data must be given:

1. The name of the agency (“agency_name”, string)
2. The URL of the agency’s website (“url” , string)
3. The agency’s unique code (“agency_code”, string)

If the server is unable to process the request for any reason, the server will respond with the appropriate error code.

Unregister
Service Aim: unregister (delete) a company from the directory
This service is not available. To unregister a company, you should contact the directory admin to get it removed manually.

The Client Application

You will write a command line (console) application to interact with the APIs. The application is able to send requests to and process responses from service providers. The application should be able to display a list of all news agencies listed in the directory. It should also be able to aggregate news stories from all agencies registered in the directory. The basic commands accepted by the application interface are as follows:

login
This is used to log in to a news service. The syntax for this command is:

login *url* (where *url* is the address of the service, for example janedoe.pyhtonanywhere.com)

this will prompt the user to enter a user name and password which are then sent to the service for authentication.

logout
This causes the user to log out from the current session. The syntax for this command is:

logout

post
This is used to post a news story. When this command is invoked the application prompts the user to enter the story’s headline, category, region, and details. No need to enter the date or author name as these are automatically added by the server when the story is logged into the table. The syntax for this command is:
post

news
This command is used to request news stories from a web service. It has the following syntax:

news [-id=] [-cat=] [-reg=] [-date=]

where [-id], [cat], [reg], and [date] are optional switches that have the following effects:

-id: the id of the news service, for example -id="JAD05". If this switch is left out, the application should collect news from all services.

-cat: the news category, for example -cat="tech". If this switch is left out the application should assume that cat="*".

-reg: the region of the required stories, for example -reg="uk". If this switch is left out the application should assume that cat="*".

- **date:** the date at or after which a story has happened, for example, -date="12/2/2019". The date should be in following format dd/mm/yyyy. If this switch is left out the application should assume that cat="*".

list
This command is used to list all news services in the directory. It has the following syntax:
list

delete
used to delete a news story. It has the following syntax:
delete story_key

Implementation Guidelines

To simplify system development, you can divide your work into individual Work Packages (WP) as follows:
WP1: implement the database model in Django, activate the admin site, test your model manually and refine it as needed.
WP2: implement the web API of your agency, one service at time, and in parallel write the client application code to request services, and test each service as you go.
WP3: Complete and finalize the client application, adding news aggregation to it.
WP4: Test and debug your entire system.

3. General guidance and study support

Detailed information about the Django framework and links to external sources are available on the Minerva page of the module.

4. Assessment criteria and marking process

Your implementation will be assessed according to both functional and quality requirements. As a minimum, your implementation should provide all the functionality mentioned above for both the API and the client application. In addition, the quality of the implementation particularly the data presentation and interface robustness of the application will affect the overall mark awarded for the application.

5. Presentation and referencing

For the written report, the quality of written English will be assessed in this work. As a minimum, you must ensure:

- Paragraphs are used
- There are links between and within paragraphs although these may be ineffective at times

- There are (at least) attempts at referencing
- Word choice and grammar do not seriously undermine the meaning and comprehensibility of the argument
- Word choice and grammar are generally appropriate to an academic text

These are pass/ fail criteria. So irrespective of marks awarded elsewhere, if you do not meet these criteria, you will fail overall.

6. Submission requirements

1. Once you are satisfied that your API is working properly on the local host, open a free account and upload the server (API) code to pythonanywhere.com. Detailed instructions on how to upload a Django project to pythonanywhere.com are available here https://help.pythonanywhere.com/pages/DeployExistingDjangoProject
2. When you register for a free account at www.pythonanywhere.com, please use your university username when you are prompted for an account name. Hence, if your university username is sc15xyz, your root domain address would be sc15xyz.pythonanywhere.com. This will make it much easier for module instructors to know who owns an account during coursework assessment.
3. Make sure that you have properly uploaded your server code to pythonanywhere.com and tested it thoroughly.
4. Make sure that your client was written in Python 3 and fully tested.
5. Nominate the module leader as your teacher on pythonanywhere.com. This will enable the module leader to access and view your API code directly on the website. You can do this from the 'teacher' tab in your 'account' page. The username you have to use is 'ammarsalka'.
6. Using your admin site, add the module leader as another admin user to your web service, use ‘ammar’ for username and invent a simple password for this account. Also, add ‘ammar’ as an author to the service and use the same password for this author as for the admin account.
7. Prepare a plain text file called readme.txt containing:
   I. instructions on using the client (like a quick help page with a list of all commands and how to use them).
   II. the name of your pythonanywhere domain.
   III. the password to use to login to the module leader’s admin account on your service.
   IV. any other information needed in order to use your client.
8. Put the readme.txt file and your client program in one directory called 'myclient'
9. Bundle your Django project directory (the outer one) and the 'myclient' directory into a single directory. The name of this directory should be your university email address (without the @leeds.ac.uk)
10. Compress the directory with Zip, and upload to Minerva.

As part of your submission, you should also submit a brief report (around 3 pages excluding the title page) that clearly, yet briefly, describes how you implemented the database, the APIs, and the client application. Please do NOT fill your report by copying text from online resources as we are only interested to know what you have done yourself. A template for the report can be found on Minerva with this brief. Upload the report to Minerva as part of your submission.

7. Academic misconduct and plagiarism

Leeds students are part of an academic community that shares ideas and develops new ones.

You need to learn how to work with others, how to interpret and present other people's ideas, and how to produce your own independent academic work. It is essential that you can distinguish between other people's work and your own, and correctly acknowledge other people's work.

All students new to the University are expected to complete an online Academic Integrity tutorial and test, and all Leeds students should ensure that they are aware of the principles of Academic integrity.

When you submit work for assessment it is expected that it will meet the University’s academic integrity standards.

If you do not understand what these standards are, or how they apply to your work, then please ask the module teaching staff for further guidance.

By submitting this assignment, you are confirming that the work is a true expression of your own work and ideas and that you have given credit to others where their work has contributed to yours.

8. Assessment/ marking criteria grid

The web service database is correctly designed and properly implemented in Django	(3 marks)
The web service is deployed on pythonanywhere.com	(4 marks)
All APIs are correctly implemented and meet the specifications	(10 marks)
The client application works correctly with the student’s own web service	(3 marks)
The client application can aggregate news from all other web services	(4 marks)
The client application clearly and nicely presents all data	(3 marks)
Quality and completeness of the report	(3 marks)