Cloud Computing & Cloud Computing Concepts Coursework

1. Development of a Cloud Software as a Service

The goal of the coursework is to make you apply the concepts and software development methods and frameworks seen in class to develop a Cloud Software as a Service (SaaS).

- The coursework requires you to install, develop and test a Cloud SaaS in a virtualised environment according to the guidelines described in this document.
- Upload your final scripts and a technical report to describe the functionality of your solution following the guidelines of the coursework brief document.

2. The Piazza system

Your task is to develop a RESTful SaaS for a Twitter-like system called **Piazza**. In **Piazza**, users post messages for a particular topic while other users browse posts per topic and perform basic interactions, including like, dislike, or add a comment. You should install, test, and document your developments as described in Section 3.

Piazza should support the following actions.

Action 1: Authorised users access the **Piazza** API using the oAuth v2 protocol.

Action 2: Authorised users post a message for a particular topic in the **Piazza** API.

Action 3: Registered users browse messages per topic using the **Piazza** API.

Action 4: Registered users perform basic operations, including "like", "dislike" or "comment" a message posted for a topic.

Action 5: Authorised users could browse for the most active post per topic that has the highest total number of likes and dislikes.

Action 6: Authorised users could browse the history data of expired posts per topic.

Use the Django RESTful API framework to develop your software. You are advised to use code samples from the lab tutorials and online sources by providing the appropriate references (in text and in code). You are encouraged to improvise providing clear descriptions of your functionality and your implementation decisions.

3. Coursework phases

Phase A: Install and deploy software in virtualised environments [10 marks]

- Install all the necessary packages in your virtual machine.
- Deploy your code in the virtual machine.
- Your REST API endpoints should be available under your virtual machine IP address based on examples and guidelines seen in class.
- Provide a short description of your setup in the report.
- Briefly discuss your installation and the structure of your folders.
    - Provide screenshots if necessary.
- Do not include Linux commands in the report.

Phase B: Development of authorisation and authentication services [15 marks]

- Create an authorisation server using Django frameworks as seen in the labs.
    - The server will authorise the **Piazza** RESTful API to store data in a database, on behalf of authorised users.
- Register and authenticate users each time you perform an Action of Section 2, e.g. to post, like, dislike or browse messages.
    - Unauthorised users are not allowed to access the resources and perform database queries.
- The authorisation service should implement the oAuth v2 protocol as in Class 4.
- You should provide a brief description of the functionality of the oAuth v2 protocol and how you integrate it in your software to perform authorisation in the report.

Phase C: Development of Piazza RESTful APIs [35 marks]

- Your APIs should allow the basic functionalities as provided in Action points of Section 2.
- Each post in the **Piazza** wall should include the following data:
  - The post identifier.
  - The title of a post.
  - The topic of a post, from one of the following four categories: Politics, Health, Sport or Tech. Each post could belong to one or more topics.
  - A timestamp of the post registration in the **Piazza** API.
  - The message body of a post.
  - The post expiration time. After the end of this time, the message will remain in the **Piazza** wall, but it will not accept any further actions e.g. (likes, dislikes, or comments).
  - The status of a post, this could be "Live" or "Expired".
  - Information about the post owner (e.g. a name).
  - The number of likes, dislikes, or a list of comments if any.
  - Any other information you might need to store essential for your project.
- Each user interaction with a post of a topic (like, dislike or comment) should include the following data:
  - Information about the user interacting with the post of a topic (e.g. a name).
  - The interaction value (including a like or a dislike, or a comment made).
  - The time left for a post to expire.
  - Any other information you might want to store, essential for your project.
- You are encouraged to develop your database tables and application logic and expand Phase C as required.
  - Provide a brief description of your database models in the report.
  - Use screenshots if necessary.

- You should provide a list of the RESTful API endpoints in your report with a simple example of how to use your API.
- Consider the following.
    - Only authorised users should access the API.
    - The API should allow any registered user to perform posts.
    - You are encouraged to set any other constraints to improve functionality of the software.
    - You are encouraged to improvise and develop any functionality you like.

Phase D: Development of a testing application **[15 marks]**

Develop a testing application using Python to automate the test cases of users posting messages. This application should be developed outside of your virtual machine, e.g. in your desktop or laptop. The application should connect to the API endpoints and perform the necessary HTTP calls to manipulate data as required. The test cases should demonstrate Action points, as described in Section 2. The test cases will verify your system functionality.

Let us assume a use case scenario of four users such as Olga, Nick, Mary, and Nestor that access the Piazza API. Provide the following test cases (TCs).

TC 1. Olga, Nick, Mary, and Nestor register and are ready to access the **Piazza** API.

TC 2. Olga, Nick, Mary, and Nestor use the oAuth v2 authorisation service to register and get their tokens.

TC 3. Olga makes a call to the API without using her token. This call should be unsuccessful as the user is unauthorised.

TC 4. Olga posts a message in the Tech topic with an expiration time (e.g. 5 minutes) using her token. After the end of the expiration time, the message will not accept any further user interactions (likes, dislikes, or comments).

TC 5. Nick posts a message in the Tech topic with an expiration time using his token.

TC 6. Mary posts a message in the Tech topic with an expiration time using her token.

TC 7. Nick and Olga browse all the available posts in the Tech topic, there should be three posts available with zero likes, zero dislikes and without any comments.

TC 8. Nick and Olga "like" Mary's post in the Tech topic.

TC 9. Nestor "likes" Nick's post, and "dislikes" Mary's post in the Tech topic.

TC 10. Nick browses all the available posts in the Tech topic; at this stage he can see the number of likes and dislikes for each post (Mary has 2 likes and 1 dislike and Nick has 1 like). There are no comments made yet.

TC 11. Mary likes her post in the Tech topic. This call should be unsuccessful, as in **Piazza** a post owner cannot like their own messages.

TC 12. Nick and Olga comment for Mary's post in the Tech topic in a round-robin fashion (one after the other adding at least 2 comments each).

TC 13. Nick browses all the available posts in the Tech topic; at this stage he can see the number of likes and dislikes of each post and the comments made.

TC 14. Nestor posts a message in the Health topic with an expiration time using her token.

TC 15. Mary browses all the available posts in the Health topic; at this stage she can see only Nestor's post.

TC 16. Mary posts a comment in the Nestor's message in the Health topic.

TC 17. Mary dislikes Nestor's message in the Health topic after the end of post expiration time. This should fail.

TC 18. Nestor browses all the messages in the Health topic. There should be only post (his own) with one comment (Mary's).

TC 19. Nick browses all the expired messages in the Sport topic. These should be empty.

TC 20. Nestor queries for an active post having the highest interest (maximum sum of likes and dislikes) in the Tech topic. This should be Mary' s post.

Feel free to develop more test cases to test your implementations. You should provide appropriate screenshots or descriptions of each test case in the report.

Phase E: Report your solution in a technical annex **[15 marks]**

Provide details on your implementations, your database design, descriptions of your services, API resources and any other information required. You do not need to include a list of the Linux commands in the report. Provide references to online resources, including source codes used from online tutorials or online repositories. There is no report size, but you are advised to keep the report short and clear in terms of the development phases.

Phase F: Submit quality scripts **[10 marks]**

Your codes should be written in such a way that makes them highly readable and easy to follow. Good use of object-oriented programming concepts, use of REST concepts, understanding of computational complexity in software development, use of comments in code, proper database models, proper indentation, clear notations, and simplicity in the flow are factors to consider.

4. General coursework guidelines

Consider the following when developing your software.

- You are recommended to provide your solutions using Python programming and the Django REST and oAuth frameworks.
- You are encouraged to use the material of the lab tutorials to deploy and develop software.
- Follow the instructions of the coursework brief and coursework description on the Moodle page for time and mode of submission of your software.

- Provide comments in your code to explain the key functionalities and implementation aspects.
- Whenever necessary, provide screenshots of your API calls to demonstrate functionality.
- Avoid adding large code samples in the report.
- Provide a clear description and example of your API endpoints in the report.
- There is no limit for the report size.
- Provide clear explanations of the test cases and of work limitations for incomplete tasks.
- The coursework requires you to create database tables to save your data.
- You are recommended to follow the Django object relational mapping as seen in class.
- There could be multiple ways to implement tables to manage the data of your developments, feel free to select the one you prefer.
- Best coursework(s) will win a special prize.