UNIVERSITY OF LONDON

Appendix B – Coursework brief template

MSc Computer Science

Module: CSM020 - Cloud Computing

Coursework: January to March 2024 study session

Submission Deadline: Monday 1 April 2024 13.00 BST

- **Please Note**: You are permitted to upload your Coursework in the final submission area as many times as you like before the deadline. You will receive a similarity/originality score which represents what the Turnitin system identifies as work similar to another source. The originality score can take over 24 hours to generate, especially at busy times e.g. submission deadline.
- If you upload the wrong version of your Coursework, you are able to upload the correct version of your Coursework via the same submission area. You simply need to click on the 'submit paper' button again and submit your new version before the deadline.
- In doing so, this will delete the previous version which you submitted and your new updated version will replace it. Therefore, your Turnitin similarity score should not be affected. If there is a change in your Turnitin similarity score, it will be due to any changes you may have made to your Coursework.
- Please note, when the due date is reached, the version you have submitted last, will be considered as your final submission and it will be the version that is marked.
- **Once the due date has passed, it will not be possible for you to upload a different version of your assessment. Therefore, you must ensure you have submitted the correct version of your assessment which you wish to be marked, by the due date.**

Coursework is weighted at 75% of final mark for the module.

Coursework Description

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

- The coursework requires you to install, develop and test a Cloud SaaS.

- Upload your final scripts and a technical report to describe the functionality of your solution. Claim your Github classroom repository here: **https://classroom.github.com/a/KmFOd0b1**

- Follow the specifications and guidelines described in this document.

1. MiniWall RESTFul API

MiniWall is a SaaS API where users can post their thoughts, views, or criticisms for everyone to see while other users can browse and interact by providing comments and likes.

You should install, test, and document your developments as described in Section 3. Your software should support the following:

- Authorise your users to access the MiniWall API using the oAuth protocol.
- Authorised users could post a text in the MiniWall system.
- Authorised users could browse all posts.
- Authorised users could comment or like a post.

Use the NodeJS 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 code). You are encouraged to improvise giving clear descriptions of your functionality and your implementation decisions.

2. Coursework phases

Phase A: Set up your MiniWall project and install necessary libraries

[5 marks]

- Install all the required packages in your local machine.
- Your REST API endpoints should be available under localhost. Provide screenshots of your URL endpoints in the report.
- Provide a short description of your setup in the report, including the node.js libraries used. You do not need to analyse libraries or configurations.
- Briefly discuss your installation and the structure of your folders.

Phase B: Enforcing authentication/verification functionalities

[15 marks]

- Your service should include user management and JWT functionality using NodeJS.
    - The JWT will authorise the wall RESTful API to store data on behalf of authorised users in your database, you should use a MongoDB server for this coursework, as shown in labs.

- Your code should authenticate users each time you perform any interaction with the MiniWall, for example, whenever users post, comment or like posts.
    - Unauthorised users are not allowed to access any resources and perform database requests.
    - Any user input should follow a verification process for validation purposes. You are encouraged to improvise and apply your own validations.

Phase C: Development of the MiniWall RESTful API

[30 marks]

- Your MiniWall APIs should allow the basic CRUD functionalities for posting or commenting a text such as Create, Read, Update, Delete. You are encouraged to improvise and create your custom data models.
- Each post should include the following data:
    - A post identifier.
    - A post title.
    - A timestamp of the post registration in the system.
    - A post owner.
    - A post description.
    - Any other information you might need to store and is essential for your project.
- Each user could comment or like a post including:
    - Comment on a post. This should include a timestamp and the commented text.
    - Like a post. **Note**: In the MiniWall a post owner cannot comment or like her/his post. The MiniWall always show posts according to the number of likes. Popular posts, with high number of likes are on the top. Posts with the same number of likes are shown in chronological order of the date of post.
    - Any other information you might need to store and is essential for your project.
- You are encouraged to develop your user management system or duplicate the user data model and implementation of lab four.
- You are encouraged to develop your MiniWall database tables and application logic for the service developments 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 post a text.

- You should use validations to improve the software's functionality, e.g. email validation, string validation, min and max characters allowed in a field. Feel free to include more validations as needed and improvise to demonstrate the validations.
- You should only store hashed user passwords in your database.
- You are encouraged to improvise and develop functionality as needed.

Phase D: Development of the MiniWall testing cases

[10 marks]

Develop testing cases for users posting text in the MiniWall. Feel free to use Postman or Python or any other programming language that you prefer to implement and demonstrate the calls.

Your test cases should connect to the API endpoints and perform the necessary HTTP calls to manipulate data as required.

The test cases should demonstrate posting text, as described in Section 2, to verify your system functionality.

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

TC 1. Olga, Nick and Mary register in the application and access the API.

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

TC 3. Olga calls the API (any endpoint) without using a token. This call should be unsuccessful as the useris unauthorised.

TC 4. Olga posts a text using her token.

TC 5. Nick posts a text using his token.

TC 6. Mary posts a text using her token.

TC 7. Nick and Olga browse available posts in chronological order in the MiniWall; there should be three posts available. **Note:** that we do not have any likes yet.

TC 8. Nick and Olga comment Mary's post in a round-robin fashion (one after the other).

TC 9. Mary comments her post. This call should be unsuccessful; an owner cannot comment owned posts.

TC 10. Mary can see posts in a chronological order (newest posts are on the top as there are no likes yet).

TC 11. Mary can see the comments for her posts.

TC 12. Nick and Olga like Mary's posts.

TC 13. Mary likes her posts. This call should be unsuccessful; an owner cannot like their posts.

TC 14. Mary can see that there are two likes in her posts.

TC 15. Nick can see the list of posts, since Mary's post has two likes it is shown at the top.

Note: To extract comments and likes per post you can use the post id, or any strategy that you feel fits to your application.

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

Phase E: Deploy your MiniWall project into a GCP VM using Docker

[5 marks]

Upload your code to a GitHub repo and then deploy it in a GCP VM. Provide a list of commands in the report and screenshots to demonstrate your deployment.

Phase F: Document your MiniWall solution in a technical report

[15 marks]

Provide details on your implementations, your database design, descriptions of your services, API resources, and other required information. Provide references to any online resources, including source code used from online tutorials or repositories. You should provide your report in a PDF file and submit it via TurnitIn in Moodle.

Phase G: Submit quality scripts

[20 marks]

Your codes should be written in such a way that makes them highly readable. Some factors are a good use of programming concepts, comments, proper database models, clear notations, and simplicity in the flow. You are advised to follow the implementation methods seen in labs 3.

General coursework guidelines

Consider the following when developing your software.

- You are advised to provide your solutions using NodeJS for the cloud service and Postman to test your endpoints.
- You do not need to develop a front end.
- You are encouraged to reuse the lab tutorials to deploy and develop software, as seen in class.
- Follow the instructions of the coursework brief and coursework description on the Moodle page for the time and mode of submission of your software.
- Provide comments to explain key functionalities and implementations.
- Whenever necessary, provide screenshots of your API calls to demonstrate functionality.
- Provide a clear description and example of your API endpoints in the report.
- The report should be no more than 3000 words in length. Supplementary material such as tables can be attached as an appendix.
- Provide clear explanations of the test cases and future work.
- The coursework requires you to create database tables in MongoDB to save your data, so you can improvise as needed.
- Consider that there could be multiple ways to implement database tables to manage the data of your developments.

How to submit:

Work on your coursework using the repository allocated to you in the module github classroom. To claim your repository, please follow the submission link from the assessment section of the VLE or follow this link: **https://classroom.github.com/a/KmFOd0b1**

Make sure you use the repository provided in the classroom throughout development, so that the examiners can track your progress and process cf. Development Style section on the marking rubric below.

Note: Make sure that in addition to your source code you also include any other files or directories that are needed for your program to run.

To submit your work for marking, clone the version of the github repository that you wish to be considered by cloning it to Codio.

Go to the coursework assignment in Codio. Open the terminal (using Tools-> Terminal) and type:

>> git clone YOUR_GITHUBCLASSROOM_REPO_LINK

where YOUR_GITHUBCLASSROOM_REPO_LINK is the URL of your assigned GitHub project repository.

Enter the credentials of your GitHub account. Remember that you need to use a personal access token as your password.

Note: You must provide:

• the GitHub repo with the final version of your coursework

(and history).

• The written report (3000 words).

Both entries contribute to your mark for the implementation.

Assessment Criteria:

Please refer to Appendix C of the Programme Regulations for detailed Assessment Criteria.

Plagiarism:

This is cheating. Do not be tempted and certainly do not succumb to temptation. Plagiarised copies are invariably rooted out and severe penalties apply. All assignment submissions are electronically tested for plagiarism. More information may be accessed via: https://learn.london.ac.uk/mod/page/view.php?id=3214