CS3219 S1-AY2021-22: OTOT - Own Time Own Target Tasks

This document describes OTOT tasks and respective submission instructions. There are two categories of tasks: Tasks A-G and Tasks X-Z. The last day of submission for all OTOT tasks is **November 12th, Friday, 2359.**

Read the document carefully before starting to do the tasks.

In general, each task will require a submission through LumiNUS which will be used to provide comments and grades while all communication for tasks will be conducted through Microsoft Teams.

Read the instructions, including the General FAQs, carefully. The FAQs will be updated through the FAQ files on MS-Teams, located under "General" -> "Files" -> "Class Materials" -> "OTOT_FAQs". Each respective OTOT FAQ document will contain the collated questions the teaching team has received and will be updated periodically. To submit a question, please post in the respective OTOT task channel on Teams. Further instructions on this can be found in the Teams channel for each task as a pinned post.

It is your responsibility to check the FAQ document regularly, and you should especially check the document:

1. Before attempting a task.
2. If you encounter any difficulties or queries while attempting a task.

Queries that have already been answered in the FAQ document previously or in this document will not be answered.

Given below is a description of the two ways you can accumulate a **maximum** of 35 marks for OTOT tasks, which includes an additional 5 marks for early submission or going beyond what is required.

OPTION 1

30 Marks: Accumulated by doing tasks listed in A-G.

5 Marks: Accumulated through options X-Z listed in the 'Additional Marks' section.

- **Important: These 5 marks cannot be earned without completing a minimum of 25 marks worth of tasks from A-G.**

OPTION 2

35 Marks: Accumulated by doing tasks listed in A-G.

Task A: Docker & Kubernetes Task

A1 – 2 Marks: Introduction to Docker.

Task

Deploy a simple web server using Nginx running in a Docker container.

Marking scheme

1. [1 mark] Demonstrate the ability to write a simple Dockerfile
2. [1 mark] Setup NGINX and run a reverse proxy

Resource Links

1. https://github.com/docker/labs/blob/master/beginner/chapters/setup.md
2. https://docs.docker.com/get-started/
3. https://docs.docker.com/docker-hub/
4. https://www.nginx.com/resources/wiki/start/
5. https://www.nginx.com/resources/library/complete-nginx-cookbook/

Instructions

1. Submit a PDF document as proof of completion into the OTOT task A1 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to Github repository.
    c. Instructions on how to run the Docker container.
    d. Any other relevant learnings.

    File format: <*Matriculation Number*>_A1.pdf

    Folder name: OTOT_Task_A1

FAQs

Question 1: Do we have to set up frontend and backend services?

Answer 1: No, you do not. While you are free to submit a more complex deliverable (and we encourage students to explore containerization further), for Task A we primarily expect that the Docker container be a simple web server serving a static HTML page using a reverse proxy. This will require some minimal configuration of the web server and Dockerfile.

Question 2: What files should be present in the GitHub repository?

Answer 2: Your repository should contain the files that were used to create the custom Docker image(s). Hence, we are primarily looking for the Nginx configuration file(s) (for the reverse proxy) and the Dockerfile/docker-compose file(s). If you would like to include any other custom files you used such as for the web server's HTML page, etc. you are free to do so.

A2 – 2 marks: Introduction to Kubernetes

Task

Deploy an image using Kubernetes and configure a service to access your application image.

Note that if you plan to attempt task A3, it would be helpful to deploy a suitable image that you can successfully send dummy load to in order to trigger the horizontal pod auto-scaler (refer to task A3 for more details). You need not use the same image as that used in task A1.

Learning Objective

1. To understand the basic architecture of Kubernetes and its terminologies.

Marking Scheme

1. [1 mark] Setup Kubernetes through Docker Desktop, understand its basic components (service, deployment, nodes, pods) and configure kubectl
2. [1 mark] Ability to connect to a deployed application image through a configured service, either by writing relevant service yml or through kubectl commands

Resource Links

1. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
2. https://kubernetes.io/docs/concepts/services-networking/connect-applications-service/

Instructions

1. Submit a PDF document as proof of completion into the OTOT task A2 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to Github repository.
    c. kubectl commands to deploy image, configure service and access deployed image through the service.
    d. Any other relevant learnings.

    File format: <*Matriculation Number*>_A2.pdf

    Folder name: OTOT_Task_A2

A3 – 4 marks: Advanced Kubernetes features

Task

Implement a load balancer and horizontal pod auto-scaler for your deployed image.

Learning Objective

1. To appreciate the benefits of an orchestration service over and above a containerization service.
2. To understand how deployment of containerized applications might look like through load-balancing.

Marking Scheme

1. [1 mark] Demonstrate the ability to write a simple ingress yml file.
2. [1 mark] Use an ingress controller (for example, ingress-nginx) to enable sticky session load-balancing between 2 replicas using cookies
3. [2 marks] Define a basic horizontal pod auto-scaler and demonstrates it works as expected for a scaling metric of your choosing

Resource Links

1. https://kubernetes.io/docs/concepts/services-networking/ingress/
2. https://kubernetes.github.io/ingress-nginx/
3. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/
4. https://kubernetes.github.io/ingress-nginx/examples/affinity/cookie/

Instructions

1. Grading for this task will be conducted through a demonstration. More instructions for the demonstration and slot booking will be released at a later date through the Teams channel for task A.
2. In addition, submit a PDF document as proof of completion into the OTOT task A3 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to Github repository.

    File format: <*Matriculation Number*>_A3.pdf

    Folder name: OTOT_Task_A3

FAQs

Question 1: Can I combine my submission for tasks A1, A2 and A3 somehow?

Answer 1: In order to assist the teaching team with marking, please submit all sub-tasks separately. This is because your mark for each sub-task will be announced to you through the PDF submission for each task. Please refer to the question 4 in the FAQs section for more details. Moreover, while all three tasks require a pdf submission on LumiNUS, please remember that only task A3 will be graded through a demonstration while task A1 and A2 will be graded solely on the work submitted on LumiNUS.

Question 2: Must I use the ingress-nginx ingress controller? Can I use another ingress controller instead?

Answer 2: Yes, you may use any ingress controller as long as you demonstrate the ability to enable sticky sessions using cookies.

Task B: CRUD Application Task

There are 4 sub-tasks under Task B. For these four tasks, you may choose to attempt any number of them (i.e. 1 - 4) but do note that each subsequent task builds on the previous one. We would like to emphasize that while doing different sub-tasks at different times is possible, each subsequent sub-task builds upon the previous one and hence would recommend completing and submitting them at one go.

B1 – 3 Marks: Implementing a simple backend.

Task

Implement a simple backend and REST API to GET, POST, PUT, DELETE (example: an API for a quotes). You may use any programming language of your choice.

Marking scheme

1. [1 mark] Successful GET, POST, PUT, DELETE API calls on localhost using Postman
2. [1 mark] Successful GET, POST, PUT, DELETE API calls to deployed endpoints using Postman
3. [1 mark] Some ability to handle common edge cases and error-resiliency. The deployed endpoint especially should not crash.

Resource Links

1. https://expressjs.com/en/starter/hello-world.html
2. https://medium.com/@dinyangetoh/how-to-build-simple-restful-api-with-nodejs-expressjs-and-mongodb-99348012925d

FAQs

Question 1: What components need to be configured in addition to the expected basic API? Do I need to provision a database?

Answer 1: There are no specific requirements to what components need be included for your API application, including the need to provision a database. At the end of the day, what components your API application should be made up of is dependent on the kind of functionality you are trying to deliver.

B2 – 3 Marks: Testing through Continuous Integration (CI).

Task

Write simple test cases for your designed API and use a CI tool (Travis, etc) to automate testing. You may use any programming language of your choice.

Learning Objectives

1. To understand the importance of testing and CI tools in simplifying the process of testing.

Marking scheme

1. [1.5 marks] Successful testing for API using Mocha/Chai, or any other testing framework
2. [1.5 marks] Ability to use Travis or any other CI tool to automate testing, including the necessary instructions to run these tests

Resource Links

1. https://mochajs.org/#getting-started
2. https://medium.com/@asciidev/testing-a-node-express-application-with-mocha-chai-9592d41c0083
3. https://dev.to/bushraalam/introduction-to-testing-with-mocha-and-chai-57po

FAQs

Question 1: How rigorous must the testing be?

Answer 1: At minimum, testing must cover the 4 REST API calls (i.e. the 4 HTTP methods: GET, POST, PUT, DELETE). There is no need to test the frontend if you are implementing task B4.

Question 2: The submission instructions state to contain the instructions for how to run tests via Travis. Does this mean that I would have to give the teaching team the access rights to my GitHub repository so that they can trigger the builds?

Answer 2: No, that is not necessary. What we are looking out for is the travis.yml file (in the repository) and a brief description in the report on which part of the config initiates testing. This would sufficiently demonstrate how you went about triggering the tests. Please also include screenshots showing the output of the tests from one of your Travis builds.

B3 – 3 Marks: Deployment through Continuous Deployment (CD).

Task

Integrate a CD tool for automated deployment to a serverless service.

Learning Objectives

1. To understand the power of automated deployment, and serverless tools in the creation of APIs or microservices.

Marking scheme

1. [3 marks] Ability to use automatic deployment tools through serverless services (e.g. AWS Lambda or Google Cloud Functions)

Resource Links

1. https://docs.travis-ci.com/user/deployment/lambda/
2. https://www.cloudflare.com/learning/serverless/glossary/serverless-microservice/
3. https://microservices.io/patterns/deployment/serverless-deployment.html
4. https://martinfowler.com/articles/serverless.html

B4 – 3 Marks: Implement a frontend.

Task

Build a frontend Single Page Application (SPA) using React, Vue or any other library/framework.

Learning Objectives

1. To understand Model-View-Controller in the context of real-life frameworks.

Marking scheme

1. [2 marks] Ability to interact with the API using the frontend
2. [1 mark] Implementation of style e.g. using Bootstrap

Resource Links

React.js

1. Official React Docs - https://reactjs.org/
2. Introduction to React by Harish - https://www.youtube.com/playlist?list=PL2egE6KxHa-tdkFMsxvNeig9KjAJTvW_3
3. Building an All-Star React Stack by Harish - https://www.youtube.com/watch?v=FA3pAXKz2fo&t=278s

Vue.js

1. https://vuejs.org/v2/guide/
2. https://vuejs.org/v2/cookbook/using-axios-to-consume-apis.html

Instructions

If you are submitting the entirety of task B:

1. Submit a PDF document as proof of completion into the OTOT task B folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to the GitHub repository.
    c. Instructions on how to:
        i. Run the API locally, including Postman calls used to demonstrate a working API. Be sure to include the edge cases tested or proof of error-resiliency as per the marking scheme. (B1)
        ii. Access the deployed API (B1/B3)
        iii. Run tests locally and via Travis (B2)
        iv. Set up frontend (B4)
    d. Supporting screenshots wherever relevant

    File format: <*Matriculation Number*>_B.pdf

    Folder name: OTOT_Task_B

If you are submitting only parts of task B:

In your filename and within the PDF document, please state clearly which sub-tasks of B your submission is for and include the relevant instructions as described above. For example, when submitting for sub-

tasks B1 and B2, your filename can be <*Matriculation Number*>_B1_B2.pdf and at the top of your PDF you can state clearly "Submission for Task B1 and B2".

If you partially submit and then wish to submit more after:

When/if you submit subsequent sub-tasks of B, please do not delete your previous submission for the other previously completed sub-tasks from the folder, else we will not be able to track early submission records. Simply add a new file with the information as specified above. Continuing from the above example, the subsequent submission of sub-tasks B3 and B4, your filename can be <*Matriculation Number*>_B3_B4.pdf and at the top of your PDF you can state clearly "Submission for Task B3 and B4".

Note: For submissions without these pieces of information, we will simply assume that you are submitting the entirety of Task B (i.e. Tasks B1 - B4).

Task C: Authentication & Authorization Task

Task – 4 marks

Implement authentication and authorization features for a simple backend through a REST API. You may build upon the code submitted for task B. You may also feel free to use any frameworks that support these features out of the box, provided you demonstrate the ability to meet the marking criteria.

Learning Objectives

1. Understand the difference between authentication and authorization, as well as the importance of both with regards to security practices.
2. Encourage students to consider security concerns during development and take a more proactive role in implementing security best practices.

Marking scheme

1. [1 mark] Unsuccessful GET request to API endpoint when a user is not authenticated, returns HTTP 401 response. Successful GET request to API endpoint after user is authenticated, returns HTTP 200.
2. [1 mark] Unsuccessful GET request to API endpoint when a user is not authorized, returns HTTP 403 response. Successful GET request to API endpoint if user is authorized, returns HTTP 200.
3. [2 marks] Implementation details. 1 mark will be awarded to students for a basic implementation while 2 marks will be awarded for an advanced implementation.
    a. Basic implementation examples: HTTP basic authorization, hardcoded checks for roles and permissions that is not easily extendable.
    b. Advanced implementation example: Usage of JWT, usage of framework with role and permission support, implementation of a role and permission system that can be easily extended. Essentially, anything that shows consideration of how authentication and authorization may actually be implemented in production.

Resource Links

1. https://auth0.com/docs/get-started/authentication-and-authorization - Difference between authorization and authentication.
2. https://www.youtube.com/watch?v=7Q17ubqLfaM - An introduction to JWT. Note that at 0:43 in the video he wrongly mentions that JWT is used for **authorization**, when it is actually used for **authentication.** Refer to https://stackoverflow.com/questions/31687442/where-do-i-need-to-use-jwt for more clarification. This channel also provides many tutorials and explanations on authorization and authentication that could be a useful starting point.
3. https://leastprivilege.com/2014/10/02/401-vs-403/ - Difference between HTTP 401 vs HTTP 403 and when each should be used.

Instructions

1. Grading for this task will be conducted through a demonstration. More instructions for the demonstration and slot booking will be released at a later date through the Teams channel for task C.

2. In addition, submit a PDF document as proof of completion into the OTOT task C folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to Github repository.

    File format: <*Matriculation Number*>_C.pdf

    Folder name: OTOT_Task_C

Task D: Pub-Sub Messaging

Task – 6 marks

Create a 3 node Apache Kafka cluster using Docker that demonstrates pub-sub messaging, with a Zookeeper ensemble created to manage the Kafka cluster.

Learning Objectives

1. To have hands-on experience implementing a messaging service and managing failover in containers.

Marking scheme

1. [3 marks] Successful implementation of Pub-Sub messaging system using Apache Kafka
2. [3 marks] Successful management of the failure of the master node in the cluster, i.e. another node takes over as master node

Resource Links

1. https://kafka.apache.org/intro
2. Visit this video link or find it at LumiNUS -> Multimedia -> "Video-Resources_For_OTOT Tasks" -> "OTOT_Task_D_Overview of Kafka and Demo on macOS"
3. https://hackernoon.com/thorough-introduction-to-apache-kafka-6fbf2989bbc1
4. https://medium.com/rahasak/kafka-and-zookeeper-with-docker-65cff2c2c34f

Instructions

1. Submit a PDF document as proof of completion into the OTOT task D folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to Github repository.
    c. Report on how to set up and run the Kafka cluster.
    d. Supporting screenshots where relevant

    File format: <*Matriculation Number*>_D.pdf

    Folder name: OTOT_Task_D

FAQs

Question 1: (Clarification on submission report)

Answer 1: Your report should provide detail on how to set up the Kafka cluster and how to publish a message that will be received by a subscriber. (This publishing of a message does not have to be automated and can be done manually over the command line.)

To demonstrate successful management of master node failure, you will need to kill the master node and then show how the Zookeeper elects a new master to continue the Pub-Sub messaging functionality as described above. You can show this in your report either by specifying the instructions to kill the master node, publish a message and receive the message or simply provide a screenshot that clearly indicates that this is achieved (adding annotations to the screenshot will assist the teaching team in marking)

Task E: Blogpost Task

E1 – 3 marks: Writing and publishing a technical blogpost.

Task

Write and publish a technical blog post on any software engineering principle, pattern or concept from within or outside the module scope.

Learning Objectives

1. Reinforce concepts learnt in the module by making students explain it in their own words.
2. Encourage students to consider how concepts learnt in class can be used in other situations.
3. Encourage students to look for other SE principles, patterns and concepts beyond those taught in the module.

Marking scheme

You can pick and choose any marking criteria (up to a maximum of 3 marks).

1. [1 mark] Clear, precise language used to explain the principle, pattern or concept.
2. [1 mark] Effective use of diagrams.
3. [1 mark] Providing a concrete, hypothetical example where such a pattern, principle or concept is utilized.
4. [1 mark] Unique topic chosen that is relevant to the module but not covered in module syllabus.
5. [2 marks] Provide a concrete, real-life example (based on student's past or existing project or work) where such a pattern, principle or concept is utilized.

While you should not be too verbose, your post should have a minimum length of 400 words.

Resource Links

1. Visit this video link or find it at LumiNUS -> Multimedia -> "Video-Resources_For_OTOT Tasks" -> "OTOT_Task_E_How to Write a Great Technical Blog"

Instructions

1. Submit a PDF document as proof of completion into the OTOT task E1 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Marking criteria numbers you wish to be marked against.
    c. Link to your published blog post (you can consider using Medium).

    File format: <*Matriculation Number*>_E1.pdf

    Folder name: OTOT_Task_E1

2. Add the link to your published blog post in the document **"Collated_Blogpost_Links".** This can be found under Teams channel "OTOT_Task_E" -> "Files". This will be used to collate all the technical blogposts as a resource for students and will be used by your peers to find a suitable blogpost to review for task E2.

E2 – 3 marks: Reading, reflecting and critically responding to peers' blogposts.

Task

Pick any blog post written by your peer to learn from and to provide feedback. Please refer to the document **"Collated_Blogpost_Links"** under Teams channel "OTOT_Task_E" -> "Files" for links to the various blogposts written by your peers. If you wish to reserve a blogpost to comment on (for example, if you need more time to formulate your comment but are keen to comment on a specific blogpost), you can add your name to the "Commentor 1" or "Commentor 2" column as needed. However, please reserve only **one** blogpost to comment on and **ensure you submit the entirety of OTOT task E2 within a week after reservation**. Furthermore, you should not comment on a blogpost that already has 2 commentors as indicated by the table, although you are more than welcome to read through your peers' blogposts.

Learning Objectives

1. Encourage students to browse through their peers' writings to learn from each other.
2. Encourage students to think critically about the information and teachings provided in the module, as opposed to taking them at face value.

Marking scheme

1. [1 mark] Insightful feedback provided. While there is no strict definition of this, avoid nitpicking on minor issues such as grammar. Instead, examples of appropriate feedback include but are not limited to:
    a. Feedback on structure or flow of ideas.
    b. Wrong definition of a key term.
    c. Highlight unmentioned assumptions.
2. [1 mark] Demonstrate understanding of original content. Examples of this include but are not limited to:
    a. Summarizing the principle succinctly.
    b. Providing an additional example that demonstrates the original principle, pattern or concept.
3. [1 mark] Build upon the original post. Examples of this include but are not limited to:
    a. Making links to other relevant principles, patterns or concepts.
    b. Highlighting tools that could be useful in applying principles, patterns or concepts mentioned in original post.
    c. Highlight limitations or assumptions the original post might have missed.

Note that there is no need to be critical without reason. You can (and should!) also highlight what the original author did well and frame your response as an actual response rather than just attempting to hit all the points in the marking scheme.

Instructions

1. Submit a PDF document as proof of completion into the OTOT task E2 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to the blog post you commented on, and where your comment should be publicly visible.

    File format: <*Matriculation Number*>_E2.pdf

Folder name: OTOT_Task_E2

2. If you have not already done so, please add your name to the "Commentor 1" or "Commentor 2" column in the table in the **"Collated_Blogpost_Links"** document under Teams channel "OTOT_Task_E" -> "Files" based on the blogpost you commented on.

FAQs

Question 1: What if someone reserves a blogpost and does not comment within a week?

Answer 1: Upon reserving a blogpost to comment on, you have one week to post your comment and complete the submission for OTOT task E2. The teaching team will be regularly checking the reservation table and students who abuse the reservation system may be penalized.

Task F: Caching Task

Task – 3 marks

Implement a Redis cache to improve application performance. **Note that it may be helpful to build upon your code from Task B should you have attempted it.**

Learning Objectives

1. Appreciate the use of a caching middleware to help improve application performance.

Marking scheme

1. [1 mark] Successful GET request that retrieves a large amount of data from a local database.
2. [2 marks] Subsequent successful GET request to the same endpoint that demonstrates substantial performance improvement due to Redis caching.

Resource Links

1. https://www.mockaroo.com/, http://jsonplaceholder.typicode.com/ - Generate large amounts of mock data.
2. https://medium.datadriveninvestor.com/all-things-caching-use-cases-benefits-strategies-choosing-a-caching-technology-exploring-fa6c1f2e93aa - Very thorough article explaining all facets of caching.
3. https://www.youtube.com/watch?v=jgpVdJB2sKQ

Instructions

1. Grading for this task will be conducted through a demonstration. More instructions for the demonstration and slot booking will be released at a later date through the Teams channel for task F.
2. In addition, submit a PDF document as proof of completion into the OTOT task F folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to Github repository. Note that this can link to your Task B repository if you implemented this task on top of your Task B's code.

    File format: <*Matriculation Number*>_F.pdf

    Folder name: OTOT_Task_F

Task G: Module Content Enhancement Task

Task – 3 marks

Note: This task can only be submitted between weeks 10-13 to allow for enough exposure to course contents and reflections. Tasks submitted before week 10 will not be graded.

Suggest an enhancement to the module content e.g. addition to the lecture or tutorial material. For this task you may consider (but not limited to) the following:

1. Propose an existing topic you would like covered with greater depth
2. Propose a new topic to be covered in the module
3. Propose the replacement of an existing topic

Learning Objectives

1. To encourage students to critically examine the existing course curriculum and explore content outside of what is taught in the classroom.

Marking scheme

1. [1 mark] Well-elaborated rationale for the proposal given
2. [2 marks] Brief explanation of the pedagogy, i.e. what you would teach about this new or more in-depth topic and how you would teach and assess it.

Instructions

1. Submit a PDF document as proof of completion into the OTOT task G folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Write-up on the change or addition to the material that covers the points in the marking scheme.

    File format: <*Matriculation Number*>_G.pdf

    Folder name: OTOT_Task_G

Please ensure that you are submitting this task between weeks 10-13.

Tasks X-Z – Maximum 5 marks

Task X: Early Submission

Task – 3 marks maximum

Either:

- 2 marks for completing 15 marks worth of tasks A-G by the **end of week 7 (Sunday, 3rd October 2021, 2359H).**
- 1 mark for completing additional 15 marks worth of tasks A-G by the **end of week 10 (Sunday, 24th October 2021, 2359H).**

Or:

- 2 marks for completing 30 marks worth of tasks A-G by the **end of week 10 (Sunday, 24th October 2021, 2359H).**

Please note that the early submission marks are awarded not for scoring 15 marks but rather for attempting tasks that total up to 15 marks. The teaching team expects demonstrable effort on your part before awarding these early submission marks but will not penalize you on this front for not scoring the full 15 marks.

For example, attempting 15 marks worth of tasks by Week 7 could mean attempting Tasks B and F. Thus, even if you end up scoring only 7 marks for these tasks, you will still get the 2 bonus marks for early submission.

Furthermore, if you have completed 15 marks worth of tasks A-G by the end of week 7 and obtain 2 marks, but fail to complete an additional 15 marks worth of tasks A-G by the end of week 10 to get the remaining 1 mark, you will not lose the earlier 2 marks.

Finally, there is no limit to how many tasks you can do. So long as you are under the maximum of 35 marks that have been allocated for the OTOT tasks you can continue attempting tasks to accrue as many marks as possible.

Task Y: Noteworthy Submissions

Task – 3 marks maximum

Lecturer and tutors will be on lookout for notable submissions that are worthy of mention. This will be announced during lectures or through LumiNUS announcements. You score an additional 1 mark every time your submission gets featured by the tutor or lecturer, up to a maximum of 3 times (for a maximum of 3 marks).

Task Z: Pursuits outside class

Task – 5 marks maximum

Up to 5 marks for contributions or pursuits outside of class (must be dated between 9th August 2021 and 12th November 2021). For each of tasks in this category, one mark will be awarded up to a maximum of 5 marks (e.g. 1 merged open-source PRs, 2 software engineering certifications and 2 out of 3 marks for the mock interview will get you 5 marks).

Tasks in this category include:

Task Z1 - 1 mark each, maximum 2 marks

- Merged open-source PRs for non-school related projects or modules

Instructions

1. Submit a PDF document as proof of completion into the OTOT task Z1 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to merged open-source PR you worked on.

    File format: <*Matriculation Number*>_Z1.pdf

    Folder name: OTOT_Task_Z1

When/if you submit subsequent sub-tasks of Z1, please do not delete your previous submission for the other previously completed sub-tasks from the folder, else we will not be able to track early submission records. Simply add a new file with the information as specified above. For example, if you later merged another open-source PR, your subsequent submission filename should be <*Matriculation Number*>_Z1.pdf as well.

FAQs

Question 1: Is there a minimum degree of difficulty/complexity that a merged open-source PR must have for it to be accepted?

Answer 1: The key requirement for merged open-source PRs is that they be feature implementation oriented. While contributions such as documentation and information updating are important parts of building and maintaining software, the focus of the module is on design principles and patterns hence the aforementioned requirement. This feature implementation can be for either the frontend or the backend. Evaluation of the PRs and awarding of marks is left to the discretion of the CS3219 Teaching Team.

Question 2: General clarifications on open-source PRs

Answer 2:

- PRs will only be considered if you are not getting paid for working on them and is not getting credit for them as part of another module (e.g. CS3281/2).
- PRs must be merged after 9th August 2021 and before 12th November 2021. It does not matter when you started work on these PRs.

- All PRs that you would like considered must be submitted (as per the instructions listed in the OTOT tasks document) to LumiNUS and not directly to the CS3219 email. Any submissions to the CS3219 email will not be evaluated.

Task Z2 - 1 mark each, maximum 2 marks

- Certifications related to Software Engineering

Instructions

1. Submit a PDF document as proof of completion into the OTOT task Z2 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. Link to software engineering certification you have achieved, or similar.

    File format: <*Matriculation Number*>_Z2.pdf

    Folder name: OTOT_Task_Z2

When/if you submit subsequent sub-tasks of Z2, please do not delete your previous submission for the other previously completed sub-tasks from the folder, else we will not be able to track early submission records. Simply add a new file with the information as specified above. For example, if you later obtained another software certification, your subsequent submission filename should be <*Matriculation Number*>_Z2.pdf as well.

FAQs

Question 1: What kind of software engineering certifications will be accepted? Is there a list of certifications we can reference?

Answer 1: We do not have a list of certifications at the moment and so we encourage students to write in to us the details of the certification they are pursuing for us to review before they embark on it. Broadly, we will approve certifications that are related to software development and deployment, and its related verticals such as the cloud. They must also be from reputable companies or platforms such as Amazon or edX respectively.

For example, you may look at certifications such as the AWS Certified Developer Associate/Professional or a course that teaches service-oriented architecture and subsequently certifies you. The courses and certifications must not be a rehash of what you may have learnt in other modules (e.g. Programming in Java, which would be very similar to CS2030 and CS2103).

Task Z3 - Maximum 3 marks

- Participate in mock interview; Limited to 20 students on a **first-come-first-serve basis**, but more slots will be made available if possible and there is demand.
    - Duration: Approximately 40 mins
    - Grading: Marks are given based on active participation and not performance. This includes but is not limited to:
        - Demonstrate effort to prepare for the interview through engagement during the interview.
        - Tailoring CV and cover letter to the job description provided.
    - Step 1: Apply for Software Engineer/DevOps/Full Stack Engineer/Quality Assurance positions based on the job description in Appendix A. Additional job descriptions, if made available, will be announced through the OTOT_Task_Z channel on Teams. Submit your CV and a short cover letter including which position is applied for to the OTOT task Z3 folder on LumiNUS (refer to the instructions below). **Submissions will be evaluated on a first-come, first-serve basis, and we will let you know through your LumiNUS submission if you did not make the cut-off.** Refer to question 4 of the general FAQs for more information.
    - Step 2: Wait for the interview call and date/time set-up + the topic to be prepared. You will receive a response within a week from your submission. Interview will be on weekend evenings. **The interview will be conducted by working professionals who are current Software Engineers.**
    - Step 3: Prepare for the interview.
    - Step 4: Take the interview.
    - Step 5: Receive feedback and participation marks 😊

Instructions

1. Submit a PDF document into the OTOT task Z3 folder on LumiNUS. The document should contain the following information:
    a. Student Name and Matriculation Number.
    b. CV and cover letter submitted.

    File format: <*Matriculation Number*>_Z3.pdf

    Folder name: OTOT_Task_Z3

General Frequently Asked Questions (FAQs)

Question 1: Are we allowed to resubmit any task for re-grading to improve our score?

Answer 1: No, you are not allowed to resubmit any task to improve your score. Once graded, your task score is final. Nonetheless, you are allowed to do more OTOT tasks to make up for any lost marks.

Question 2: General clarification on penalties for submissions that do not meet requirements as specified in OTOT tasks document

Answer 2: Students who do not submit their OTOT tasks as specified in instructions will be penalized as follows:

- Up to 20% deduction of max score for wrong file format, missing name or matriculation number, lack of detailed instructions regarding setup/access, etc.
- 0 marks awarded if task cannot be tested (as a result of missing GitHub repository link to clone from, missing deployment link, etc.)

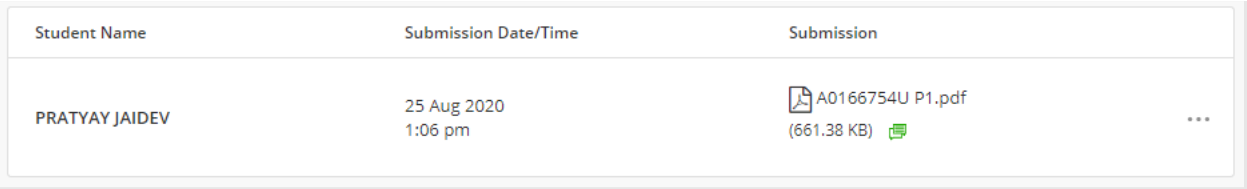
We encourage all students to double-check their work before submission to ensure that it has all the required components so that the teaching team can smoothly proceed with testing and grading. Please feel free to get in touch with the teaching team for any clarifications on submission instructions, so long as the question has not already been answered in the instructions or FAQs.

Question 3: General clarification on penalties for mistyped commands or missing instructions

Answer 3: The teaching team can only grade based on what is provided in the submissions that we receive. However, please rest assured that penalties for such errors will be minimal. An example of this would be missing a step in the instructions submitted from Task B that is necessary to run the API locally.

Question 4: Where can I check my scores?

Answer 4: Scores for each task will be released on LumiNUS in the respective submission folder within 2 weeks of submission (see the image below). We seek your patience on this matter as the teaching team has a high volume of tasks to grade.



Appendix A – CS3219 Job Description. (for OTOT Task Z3)

Company

SoC QuickTalk is an upcoming messaging platform used by institutions to power daily communications at work. It also involves a whole suite of various tools and services that helps make sharing and communication a breeze. This involves platforms for social workplace sharing. QuickTalk is used by millions to drive their businesses forward and we are excited to be in a phase of rapid expansion and scaling.

Description

We are looking for a passionate Software Engineer to design and develop software solutions with a focus on scalability.

Note: You can apply for a generic Software Engineer role or choose among one of these specific roles: Full-Stack, DevOps, QA

Software Engineer responsibilities include gathering user requirements, defining system functionality and writing code in various languages, like Java, Node.js, Ruby on Rails. Our ideal candidates are familiar with the software development life cycle (SDLC) from preliminary system analysis to tests and deployment.

Ultimately, the role of the Software Engineer is to build high-quality, innovative and fully performing software that complies with coding standards and technical design.

Responsibilities

- Execute full software development life cycle (SDLC)
- Develop flowcharts, layouts and documentation to identify requirements and solutions
- Write well-designed, testable code
- Produce specifications and determine operational feasibility
- Engineering large-scale systems
- Integrate software components into a fully functional software system
- Develop software verification plans and quality assurance procedures
- Document and maintain software functionality
- Troubleshoot, debug and upgrade existing systems
- Deploy programs and evaluate user feedback
- Comply with project plans and industry standards
- Ensure software is updated with latest features

Requirements

- Ability to develop software in Java, Python, JavaScript, Golang, C++ or other programming languages
- Understanding of HTML5, CSS3, and JavaScript
- Knowledge of relational or non-relational databases, SQL and ORM technologies (JPA2, Hibernate), MongoDB etc.

- Experience developing web applications using at least one popular web framework (Node.js, Ruby on Rails, Django etc.)
- Experience with test-driven development
- Familiarity with project management tools like JIRA or Trello
- Proficiency in software engineering tools
- Ability to document requirements and specifications
- BSc degree in Computer Science, Engineering or relevant field

Available Positions

1. Software Engineer (Generic)
2. Full-Stack Web
3. DevOps
4. QA

Does the above excite you? Apply now with your CV and a short cover letter!