CS3219 AY2323 S1 Project – PeerPrep

Maximum marks – 27

Final deliverable(s) due - 15 Nov Wednesday, 5pm

Intermediate milestones in Week 6/7 and Week 9/10

1. Introduction

This Project involves creating a technical interview preparation platform and peer matching system called PeerPrep, where students can find peers to practice whiteboard-style interview questions together. Following is a scenario to describe the context. Note that the features of application are not limited to this scenario.

> *A student who is keen to prepare for their technical interviews visits the site. They create an account and then log in. After logging in, the student selects the question difficulty level (easy, medium, or hard) and a topic they want to attempt today. The student then waits until they are matched with another online student who has selected the same difficulty level as them. If they are not successfully matched after a specific duration, they time out. If they are successfully matched, the student is provided with the question and a collaborative space to develop their solution in real-time, allowing both the student and their matched peer to collaborate on the provided question. The application should allow the users to terminate the collaborative session gracefully.*

You have the liberty to choose the implementation specifics as long as the application fits the overarching purpose.

2. Choice of architecture

To implement PeerPrep, we suggest you explore and use the microservices architecture. If your team decides to use any other architecture, discuss it with your mentor before or by the end of week 6 and get their approval. You are free to choose the technology stack to implement PeerPrep.

3. Organising work within your Project Team

Each project team will have 4 or 5 members. There will not be any workload adjustment based on team size. Form sub-groups of 2 or 3 members within each team, considering their skills and strengths. Allocate each sub-group the responsibility of completing 3 “nice-to-have" features aimed at enhancing the project. The remaining project tasks, including must-have features, reports, demos, and presentations will be managed collectively by the entire team.

4. PeerPrep Features

The following are some of the features envisioned for PeerPrep. Here we term them microservices due to our guidance based on microservices architecture. .

The following are the must-have features (labelled *Mx*) of PeerPrep.

- M1: User Service – responsible for user profile management.
- M2: Matching Service – responsible for matching users based on some criteria (e.g., difficulty level of questions, topics, proficiency level of the users, etc.) This service can potentially be developed by offering multiple matching criteria.
- M3: Question service – responsible for maintaining a question repository indexed by difficulty level (and any other indexing criteria – e.g., specific topics).
- M4: Collaboration service – provides the mechanism for real-time collaboration (e.g., concurrent code editing) between the authenticated and matched users in the collaborative space.
- M5: Basic UI for user interaction – to access the app that you develop.
- M6: Deploying the application on your local machine (e.g., laptop) using native technology stack. OR Deploy the app on a (local) staging environment (e.g., Docker-based, Docker + Kubernetes).

The following are nice-to-have features (labelled *Nx*) of PeerPrep.
Note that the list is not exhaustive. You are welcome to conceptualize and implement other nice-to-have features/functionalities. Discuss your ideas with your mentor and get his approval before implementing them.

- N1: Communication: Implement a mechanism to facilitate communication among the participants in the collaborative space (other than the shared workspace) e.g., text-based chat service and/or video (+voice) calling service.
- N2: History: Maintain a record of the questions attempted by the user e.g., maintain a list of questions attempted along with the date-time of attempt, the attempt itself and/or suggested solutions.
- N3: Code execution: Implement a mechanism to execute attempted solution/code in a sandboxed environment, and retrieve+present the results in the collaborative workspace.
- N4: Enhance question service to enable managing questions, for example, tagging (by topic, popularity, etc.,), retrieving questions on the fly during a session initiation.
- N5: Enhance collaboration service by providing an improved code editor with code formatting, syntax highlighting for one language, syntax highlighting for multiple languages.
- N6: Enhance collaboration service by providing code translation from one language to another (e.g., JS to Python) depending on the preference of the users. (I.e., user 1 prefers python, user 2 prefers JS, provide a mechanism to translate code to the preferred language of each of the users)
- N7: Incorporate generative AI to assist during the preparation. For example, users can seek help from ChatGPT, Copilot (or such other services) to seek explanation of code written by the other participant. Users can seek help from gen AI to solve the puzzle by providing prompts from within the interface.
- N8: Extensive (and automated) unit, integration, and system testing using CI. Teams can also use various test frameworks or demonstrate effective usage of CI/CD in the project.

- N9: Deployment of the app on the production system (AWS/GCP cloud platform).
- N10: Scalability – the deployed application should demonstrate easy scalability of some form. An example would be using a Kubernetes horizontal pod auto-scaler to scale up the number of application pods when there is a high load.
- N11: The application should have an API gateway of some kind that redirects requests to the relevant microservices. An example would be using an ingress controller such as NGINX ingress controller if using Kubernetes (https://kubernetes.GitHub.io/ingress-nginx/).
- N12: The application should demonstrate service discovery or implement a service registry of some kind. For example, Kubernetes has built-in DNS features and service networking to pods (https://kubernetes.io/docs/concepts/services-networking/dns-podservice/).

5. Milestones

There are three milestones that we define to implement the features of PeerPrep. You can have your own internal milestones to help you plan better and deliver the product effectively and efficiently. In each milestone, you are required to demonstrate application to your mentors. The demonstrations Milestones 1 and 2 are meant for feedback, and not graded.

- Milestone 1: You will deliver the minimum viable product (MVP or proof of concept) containing the User service, Matching service, and question service. The suggested due date for milestone 1 is the end of Week 7.
- Milestone 2: You are expected to complete developing the collaboration service and one nice-to-have per sub-group by milestone 2. The suggested due date for milestone 2 is the end of Week 9. There will be a mandatory project update meeting in Week 9 or 10, where you can demonstrate your progress to the mentor.
- Milestone 3: This milestone refers to the final implementation along with the documentation and demonstration. The deliverables include all the must-have and nice-to-have features implemented, integrated, tested and deployed following Mx and selected Nx requirements.

6. Tasks and timeline

Week(s)	Tasks
Week 3	i) Group formation through Canvas. ii) Individual reading and understanding of the project document.
Week 4	i) Establish common understanding of the requirements and deliverables, roles, communication and project management tools, project strategy. ii) Setup the GitHub classroom repository (common repository for the Assignment and, subsequently, Project).
Weeks 5-7	i) Establish the scope of the product you will develop. ii) Develop the requirements/product backlog for your project. Think of the requirements in detail (both functional and non-functional) that you propose to deliver. iii) Prioritize the requirements and develop a provisional design (include design diagram(s)). You can use notations that you are familiar with. Remember: You will be assessed against the requirements specification

	and design in your final project report and presentation at the end of the semester. Share and discuss the product backlog with your mentor. iv) Research technologies you may need to use and gain working exposure. v) Milestone 1: Complete the project MVP. vi) Consult with your mentor if you have any questions.
Weeks 8-10	i) Continue refining the requirements and design. ii) Integrate the must-have microservices iii) Continue developing the product iv) Milestone 2: Complete developing the collaboration service and one nice-to-have feature per sub-group. v) Mandatory project update meeting with your mentor a. Arrange a meeting with your mentor to show your project progress at the end of Milestone 2. This is a mandatory deliverable. You will not be given an explicit grade for this deliverable, but the review of your mentor on your progress (Satisfactory/Slow/Unsatisfactory) may impact your final project grade.
Weeks 11-13	i) Milestone 3 SUBMISSION DATE – 15 Nov Wednesday 5pm a. Code: complete product implementation, integration, testing and deployment b. Report: A technical document that includes the requirement specification (product backlog), design and implementation details c. Presentation: Present your work to the lecturers (and other teaching team members) ii) Code deliverable: a. README document detailing how to run or access the project. Marks may be deducted if the README is unclear, and the project is unable to be run or accessed by the teaching team. b. A copy of your project report in your group's repository. iii) Report a. Background and purpose of the project. b. Individual/ Sub-group contributions to the project summarized in a tabular form clearly listing technical and non-technical contributions. Sub-groups will be graded separately for the successful implementation of the selected nice-to-have features. c. Clearly specified and prioritized requirements of the application. d. Developer documentation. This can include the development process, various design diagrams, design decisions, use of design principles and patterns, and any other relevant development artifacts. You may also include screenshots of the application here to illustrate features, but this section should remain a technical documentation. e. Suggestions for improvements and enhancements to the delivered application. f. Reflections and learning points from the project process and the group work.

	iv) Presentation a. Showcase/Demo the working application with sample input(s) and output(s). b. Briefly explain the software development process, design decisions and member roles. c. Short Q&A with the teaching team where you may be asked to demonstrate certain features, elaborate on certain points, etc. d. Don't forget to submit your presentation slides immediately after the presentation.
Submission Instructions: i) Code: a. Merge all your code to the master branch of your GitHub Classroom repo that is created for you on signing up. b. Tag the commit on the master branch that marks the end of your Project as "Project" and push the tags. c. Submission deadline: **15 November 2023 (Wednesday) 5pm** ii) Report: a. Label the report file <your project group number>_Report (e.g., G04_Report.pdf). b. You may submit a PDF, DOCX or ZIP file. Strictly one file per team. c. Submission deadline: **15 November 2023 (Wednesday) 5pm** iii) Presentation: a. Label the presentation file <your project group number>_Presentation (e.g., G04_Presentation.pptx). b. You may submit a PPTX file only. c. Submission deadline: Soon after the presentation, latest by **19 November 2023 (Sunday) 5pm**	

7. Code hosting and submission

You will be provided with a GitHub classroom template repository that you will use for your project. This repository is to be maintained as a public repository.

8. Rubrics

The project deliverables will be graded for a total of 54 points according to the rubric below. Sections in the rubric would be adjusted to accommodate relevant effort that spans outside the expected deliverables. For example, it is not explicitly listed in the rubric below, but the overall evaluation will consider how software engineering principles, patterns, and practices are followed. In particular, we would look for evidence of understanding of specifying requirements and architectural design of the application that is built.

Section [weight]	In the end, you will be given a grade (A-D), for each of the sections mentioned in the left column. If there is any specific case where we need to differentiate with rubric, it will be communicated to the team. Grade A (~8-10 points on a 10-point scale) would be awarded to students who fulfil the criteria as mentioned under each of the following sections. Grade B (~6-8) C (~5-6) and D (~1-5) will be awarded if the expected criteria are not met and deductions are applied. Overall expect a ~B for demonstrating and reporting only the Must-have features.
Report [10/54]	Expected: Excellent understanding and specification of the requirements, in the project report, with most/all cases considered. Concise and unambiguous language in specifying the requirements, and that requirements are identified/traceable, categorized or prioritized. Cross-referencing, labelling, linking parts of the document is expected throughout the report. Well specified Development process and Design of the application. Design specification would expect project-specific diagrams and description of architectural design and that of detailed design/choices of key components, technical choices and tool choices, code related descriptions e.g. how code is organized, or any standards are used, project management descriptions. Cross-referencing, labelling, linking parts of the document, and an effective use of diagrams and tables, is expected throughout the report. ~ 30% marks of this section will be kept aside for design and or demonstration of: Architecture specific design/development decisions e.g., front-end/back-end design, and how the key component(s) of the app are handled or designed for in meeting the specified non-functional requirements e.g., Usability, Scalability etc. For example, for scalability, you can demonstrate it by implementing API gateway and service discovery/registry requirements as mentioned in nice-to have requirements or you could demonstrate it by simulating a stress test or any other relevant development or deployment decision you might have taken. Consistent progress in the design, development, and project management is expected. Project mentor's feedback will be considered. Deductions: generic (not project-specific) diagrams, ambiguous notations, shallow or incomplete requirements/design descriptions, missing sections e.g., missing development process, missing or incomplete requirements; requirements not identified/traceable, categorized or prioritized or language is vague.

Delivered Must have features [10/54]	Expected: consistent progress in the development and has sufficiently met or gone beyond the expectations set from the original project scope. Project mentor's feedback will be considered. deliver a well-demonstrated/deployed application with useful features. Your features belong to end users. This means that your code will primarily solve a problem for someone else. Your main purpose is to provide users with a solution that adds value and ease to their life. Further, each software requirement is not an island; individual features function in unison with other components being developed. It is your responsibility to ensure that each separate feature works as part of the whole. Marks here will overlap with the Demo section of the rubric. Deductions: superficial features/application, or the size of the application is too small to be useful, or the application is not fully functional or demonstrated or code structures have issues e.g., it does not match with design provided in the report or does not follow any standard.
Delivered 3 (or more) nice-to-have features [24/54]	These marks will be awarded to sub-groups on successful implementation and demo of the 3 selected nice-to-have features. (These should not overlap with the nice-to-have features by the other sub-group of the same project team) Sub-groups can make a claim for their selected nice-to-have features in the report as well as in the presentation. Typically 8 points are reserved for each of well defined, well implemented and integrated feature with the must-have and rest of the nice-to have features. The points overlap with the sections corresponding to requirements, design and technicalities of these nice-to-have features in the report as well as during demonstration. Deductions: superficial/not much useful implementation or not fully functional or demonstrated or code structures have issues e.g., it does not match with design provided in the report or does not follow any standard.
Presentation & Demo [10/54]	Expected: A comprehensive presentation of your project including plan, process, design, and application demonstration. Clear, concise communication is fundamental to the success of software engineering teams. It is important that your evaluators understand what you've done, how it is meant to function, how it should be deployed, what problem it solves, and what value it adds. The presentation should be highly technical in nature and not marketing the product. Deductions: not well-prepared team, does not well-communicate one or more of the expected e.g., the project plan, process, design, and application features.

9. FAQ:

1. What is Collaborative space?
   You define according to the solution you are developing.
2. What should happen when user terminates the session?
   It depends on how you want to handle the scenario. E.g., session can be terminated for both, one continues to work on the questions etc. The user can exit the application, can go back to the question selection part of the application etc.,
3. I do not have credits in my AWS/Google cloud/<Cloud provider> account. What should I do?
   Deploying on the cloud is a "nice to have". It is for your learning purpose. If you do not have credits, you can develop the project on your own machine and show the demo in the local (test/staging) environment as well. We are sourcing S$100-200 free cloud credits for each team. We shall let you know via a Canvas announcement by week 4.
   Alternatively, most cloud providers offer some sort of free trial or credits. For example, Google Cloud Platform and AWS provides credit for student sign up for limited duration. You can do a new sign up and enjoy all the benefits of the cloud platform for free. Please be aware that there could be other constraints.
4. What are some general credit management tips?
   Make sure you disable the application and pause the use of resources when it is not in use. This will save your resource usage to a large extent. Also, make sure that you do not do your testing solely on the deployed application. This increases the traffic on the app engine and hence incurs more cost.

Appendix A: Project Progress check at Week 9/10

This section details what is expected in the Project Progress check at the end of week 9 (actual schedule to be decided nearer to the time).

1. This is a mandatory meeting. All team members are expected to attend the meeting.
2. There is no graded component. However, your effort will be noted.
3. **Purpose**: Project mentors to meet the project team and verify the implementation progress by your team, and also understand the scope of the project and preliminary design of the project you are undertaking.
4. Project teams will share scope, design, implementation, and plan for the remaining weeks**.** Any member of the team can lead the session.
5. This session is expected to be technical. Most important is that you demo the product you are building.
6. Lecturers and other mentors could join your session. In total, there are 58 teams. We expect the sessions to spread over 3-5 days at the end of Week 9 and early Week 10. It will be a challenge to keep to schedule. Your cooperation and understanding in this matter would be important. Keep your presentation within 15-30 mins. If you see a time issue, speed up & keep to most essential points. Be on time. Wait patiently for the previous team to conclude if you notice a delay.

This meeting does not intend to be a Q&A session. If your mentor asks you some questions, it would be mostly to understand your presentation and thought process. Your mentor will give you feedback during or after the session. If you have any questions, arrange a separate meeting with your mentor.

Appendix B: Project Presentation Details

This section details what is expected in the final project presentation during week 13. Tentative dates are 16-18 November, 2023. Actual sign up slots (20 mins each project group) shall be announced one week before.

Tip 1: Avoid spending time on what is already provided in report

Cover very briefly if you discuss roles, process, details of SRS, design details, challenges. All these are supposed to be in the report. Therefore, avoid spending much time on the above. You may have only 15-20 mins for the presentation and expect **DEMONSTRATION OF FEATURES** followed by **KEY DESIGN DECISIONS**.

Tip 2: Let evaluators know what you have built first

Suggestion: Make a ppt slide deck with few slides (we will ask you to submit it; slides are not graded) **For example** :

1. Intro slide --- Project title, team number, names/matric (simply flash the team slide without spending additional time on it)
2. Functionality/ features implemented and if deployed.
3. Claim slide --- It should clearly state what do you think is built in terms of features each sub-group and the project team want to claim. You can also indicate a complexity rating for each feature you claim e.g. High, Medium, Low. Consider using a tabular format.
4. Tech Stack slide --- Include various tools/libraries/technologies you used in product development as well in process management

The above suggested format is meant to ensure that you let the evaluators know what you have done in the first 1-2 minutes of your presentation. **Please do not wait till the end to tell evaluators about your solution.** Briefly talk about the functionality so that the evaluator knows what you have created, so that they know what to expect in the demo and how to better evaluate the project. Please note above is just a suggested format and is not meant to say that there should be only these 4 points covered.

Tip 3: Demonstration

We are interested in knowing the important details (not every detail) of your implementation. Prepare the demonstration sequence well so that we get to see all the features you wish to demonstrate. You may mention if you had any design considerations, or if you took any crucial decisions.

1. **Do** mention if you did something special
    a. Testing
    b. Used any specific technology for the first time and how it helped you
2. **Don't** tell us
    a. Your process of doing it, e.g. you followed weekly sprints

Tip 4: Additional guidelines

1. Not every member needs to present/speak, so define roles and strategize your demonstration to make it a more coherent experience
2. Control your timing