CS3219 AY2526S1 Project

Maximum marks: 50

Expected Learning outcomes 2
Organising work within your Project Team 2
Reuse policy and declaration 2
Project context 3
Choice of architecture 3
PeerPrep Features 3
Project Milestones 7
Milestone D1: Requirements Specification 8
Tasks: 8
Instructions for Evaluation 8
Milestone D2: Progress Check 9
Decisions to be Checked (5 × 4 points each) 9
Rubric for Each Decision (4 points) 10
Example (Question Service) 10
Milestone D3: Final Submission & Presentation 11
Deliverables 11
Submission Instructions 11
Evaluation Criteria 11
Appendix 1: Setting up of team on GitHub Classroom 13
Appendix 2: Declaration 15
Appendix 3: AI Usage Policy for CS3219 15
Appendix 4: Sample data 18

Expected Learning outcomes

1. You shall apply the concepts learned in the lectures in a real-world context by designing a useful application that can be added to their portfolios.
2. You shall extend their existing software engineering knowledge by developing a requirements specification (containing both functional and non-functional requirements); developing an architectural design specification that satisfies the requirement specification; following design principles and patterns in designing the application.
3. You shall explore various design decisions in implementing the features of the product.
4. You shall get to practice and improve their communication skills (technical and non-technical) through collaborative group work, technical documentation writing, and presentations.

Organising work within your Project Team

1. Each project team will have 5 members (there will be no workload adjustments for. 4-member team; teams of 3 or less or a 4-member team; teams of 3 or fewer ally disallowed).
2. Allocate the responsibility of completing the product features and the remaining project tasks, (i.e., report, demo, and presentations) roughly equally among the team members.
3. Keep a note of the individual contributions. This will be factored in while grading the project components and marks for individual members may differ. We will use multiple peer reviews and mentor's feedback to gather information about the individual contributions and team dynamics. The teaching team will intervene if necessary.
4. A mentor will be allocated to each project team.

Reuse policy and declaration

Each group is expected to sign a declaration (see Appendix 2) that the project is bona fide work of the team. If any code is reused, or any reference is made to an existing design or documentation, acknowledge the source(s). Attach this declaration to your final submission (the presentation slides).

Additionally, the group is expected to read through the course's AI usage policy in Appendix 3.

Project context

This Project involves designing and developing a technical interview preparation platform and peer matching system called PeerPrep, where students can find peers to practice whiteboard-style interview questions together. A general description of the use of PeerPrep, also the context, is (**but not limited to**) as follows:

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

You are encouraged to explore the requirements for such an application without deviating from its overarching purpose.

Choice of architecture

To design/implement PeerPrep, we suggest you explore and use the microservices architecture. **If your team decides to use any other architecture, please contact lecturers and your mentor by the end of Week 6 at the latest and get their approval.** You are free to choose the technology stack to implement PeerPrep.

PeerPrep Features

The following are some of the features envisioned for PeerPrep. Its UI is expected to be designed for a 13-inch screen/device (i.e. basic laptop and above). It is not envisioned as a responsive design for various screen sizes. However, it could be designed to work with various browsers.

Here, we term them microservices due to our guidance based on microservices architecture.

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

- M1: User Service – responsible for user profile management.
- M2: Matching Service – responsible for matching users based on some reasonable criteria (e.g., topics and difficulty level of questions, proficiency level of the users, etc.) This service can potentially be developed by offering multiple matching criteria.
- M3: Question service – responsible for maintaining a reasonable size question repository indexed by difficulty level (and any other indexing criteria – e.g., specific topics, popularity, etc.,). Evolvable as needed, and provides good user experience on accessibility of the questions, for example, retrieving questions on the fly during a session initiation, supporting images within the question, etc.
- 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 containers.

The following are the **nice-to-have (N2H) categories** (labeled *Nx*) for PeerPrep. The items listed under each category represent a **wishlist from the teaching team**—they are **not exhaustive or prescriptive**. Your team has the freedom to design, adapt, and extend them based on your own interpretation.

Before implementing, you should **discuss your chosen direction with your mentor and lecturer**.

You may approach the **N2H categories** in one of two ways:

1. **Deep Dive into a Single Category**
    - Select one category and explore it thoroughly.
    - Demonstrate both its **relevance to PeerPrep** and its **technical implementation**, showcasing your team's strengths in that specific area.
2. **Distributed Exploration Across Categories**
    - Select features from different categories, assigning each team member one feature to investigate in depth.
    - The team then presents an **aggregated showcase**, demonstrating how these diverse features complement and strengthen PeerPrep as a whole.

Please note that these concepts will not be explicitly covered in lectures or tutorials, so your team is encouraged to conduct independent research and design.

Beyond the Suggested Categories

We do **not require** you to limit yourselves strictly to the listed N2H categories. If your team conceives of features that are more relevant to the **PeerPrep project context** and better aligned with its objectives, you are welcome to propose and implement those instead. Be sure to **discuss your ideas with your mentor and lecturer** before moving forward, to ensure they fit within the scope of the project.

N1.: Service Enhancements
This category focuses on improving the collaboration and learning experience in PeerPrep by extending the capabilities of the platform's core services. Possible directions include (but are not limited to): 1. **Enhanced Code Editor** a. Add support for code formatting. b. Provide syntax highlighting for a single language or multiple languages. c. Improve the editor to better support real-time collaborative coding. 2. **Code Translation Between Languages** a. Enable automatic translation of code from one language to another (e.g., JavaScript ⇄ Python). b. Allow each participant to view the shared solution in their **preferred programming language** (e.g., user A in Python, user B in JavaScript). 3. **Improved Communication Tools** a. Introduce additional communication channels beyond the shared coding workspace.

b. Options include text-based chat, voice calls, or integrated video calling to support richer collaboration.

4. **Question Attempt History**
   a. Maintain a record of all questions attempted by each user.
   b. Store metadata such as attempt timestamps, submitted solutions, and suggested solutions.
   c. Provide a way for users to review and reflect on their past attempts.
5. **Code Execution Environment**
   a. Implement sandboxed code execution for attempted solutions.
   b. Capture and present the output directly within the collaborative workspace.
   c. Ensure safe execution with resource limits and security isolation.

** **Note:** some of the features in this category could potentially use AI tools or can be enhanced with AI capabilities. But these are not necessarily designed as AI features.

N2.: AI Features

This category explores the integration of **generative AI** into PeerPrep to provide intelligent assistance during preparation. Potential directions include (but are not limited to):

1. **AI-Assisted Explanations**
   - Allow users to request explanations of code written by their peers.
   - Integrate with services such as ChatGPT, Copilot, or similar to generate contextual, human-readable explanations.
2. **AI-Assisted Problem Solving**
   - Enable users to prompt generative AI directly within the PeerPrep interface.
   - Use AI to suggest possible solutions, explain problem statements, or provide hints while solving coding questions.
3. **Conceptual Expansion**
   - Extend AI support to tasks such as test-case generation, debugging suggestions, or refactoring recommendations.
   - Enable customizable AI modes (e.g., “explain like I’m a beginner” vs. “give me an optimization hint”).
4. **Open-Ended Innovations**
   - You are encouraged to conceptualize and prototype additional AI-driven features that could meaningfully enhance the PeerPrep experience.
   - Any such ideas should be discussed with your mentor and lecturer before implementation.

N3.: Integration, Testing and Deployment

1. Extensive automated unit/integration/system testing, and browser compatibility, demonstrating the effective use of CI in the development process; for example:
    a. **Automated tests:** unit, integration, end-to-end/system.
    b. **Coverage targets:** e.g., ≥80% lines for core services; critical paths at 100% branch coverage.
    c. **Cross-browser/app testing:** define supported browsers/devices and pass criteria.
    d. **Non-functional tests:** load, soak, and failure-injection scenarios with pass/fail thresholds.
    e. **CI:** build → test → security scan → artifact versioning.
    f. **CD:** automated deploy to staging, manual/auto-gated promote to production.
    g. **Infra as Code:** reproducible envs (e.g., Terraform/CloudFormation), immutable images.
    h. **Rollbacks:** documented strategy (blue/green or canary) with clear MTTR goals.
    i. **Possible Deliverables:** CI logs, coverage reports, test plan + results, defect list with resolutions, pipeline config, IaC repo path, promotion policy, rollback playbook.

2. Deploy the app on a production system (AWS/GCP cloud platform).
3. 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.
    Show Scalability plan
    a. Horizontal/vertical scaling triggers (CPU, latency, Q depth).
    b. Stateful vs. stateless components; session affinity strategy if needed.
    c. Async patterns: queues/streams, outbox, sagas, exactly-once/at-least-once handling.
4. If you're deploying multiple services to demonstrate scalability, your deep dive could include, for example:
    Service registry / discovery
    a. Chosen mechanism (e.g., Consul, Eureka, AWS Cloud Map, Kubernetes DNS).
    b. Registration/health-check protocol, TTLs, and how clients resolve endpoints.
    c. Failure handling (stale instances, partial partitions).
    API gateway
    d. Gateway selection (managed vs. self-hosted), routing rules, auth, rate limits.
    e. Request/response transformation, header policies, CORS, and version routing.
    f. Canary/shadow traffic strategy and circuit breaking at the edge.

Project Milestones

To facilitate incremental work on the project, we have 3 graded milestones during the semester. A summary is provided in the table below.

Milestone	Deliverable	Weightage (percentage)	Submission Format and Deadline
D1	Requirements specification.	5	Meeting + presenting to mentor and lecturers **before end of Week 5 (Fri, 12 Sep 2025)**
D2	Progress Check	20	Meeting(s) + presenting to mentor **before end of Week 11 (Fri, 31 Oct 2025)**
D3	Full project demonstration with slides.	25	Slides to be submitted and product final commit by **Thu 13 Nov 2025 09:00hrs.** 30 min presentation with demo to graders **before the end of Week 13.** Demo slot selection will open a week before.

For any queries, post to MS-Teams channel “Clarifications [Project]” or reach out to your project mentor.

Milestone D1: Requirements Specification

Weightage: 5 points
When: by Week 5

In this milestone, you will develop:

- The project requirements, in the form of a product backlog.
- Wireframe or prototype user interface of the various product features

Tasks:

1. Refer to the must-have features of PeerPrep, M1 to M4. Treat the feature description as the high-level capabilities that the customer expects from the product.
    a. Analyze the feature capabilities, think through them, and write the requirements for each of these features; refine each high-level requirement[1] at least two levels of refinement. This milestone expects you to apply requirements analysis and requirements specification concepts.
    b. The requirements should be categorized (e.g. FR/NFR), indexed, and prioritized. You can refer to the example screenshot in the lecture notes (L1-SoftwareApplicationsDeploymentsProcesses.pdf slide number 31)
2. Refer to the must-have feature M5 of PeerPrep. Envision what the user interface would look like when you access M1 to M4 features. You can code OR draw legible wireframe diagrams OR make a UI prototype[2] the UI.
3. Refer to the nice-to-have categories of PeerPrep, N1 to N3. Decide on the category(ies) your team wishes to explore and develop in the project. Capture the key requirements for these identified nice-to-haves and document them briefly (in about 2-3 lines each) at the end of the product backlog. You can revise, refine, and elaborate on them during Milestone D2 or before Milestone D3.

Instructions for Evaluation

- Arrange a meeting with your group's project mentor in **Week 5** and present your product backlog to them. You are encouraged to prepare a slide deck for your presentation. Your mentor can ask you submit your presentation slides to them for a later reference. Slides style or polish do not carry any marks.
- Ensure all team members are present. Avoid reading from a script while presenting your work.

For clarifications on this milestone:

Use the MSTeams channel "Clarifications [Project milestones]" to seek clarifications or ask the teaching team if you have any questions about this milestone

[1] Note: Avoid rewording the given feature description e.g., "M1: User Service – responsible for user profile management" should not be reworded to "The system should provide a user service to manage user profiles". You may not get any marks for simply rewording the given features.
[2] Note: The wireframe can be drawn using any simple diagram software like Balsamiq or Powerpoint. You are NOT required to use any sophisticated software for this task.

Milestone D2: Progress Check

Weightage: 20 points (5 decisions × 4 points each)
When: Weeks 7 – 11

Milestone 2 is a **mid-project progress check**. Its goal is to ensure your team has made concrete design and implementation decisions across the core services, and that you are on track to deliver a complete system.

You are expected to arrange 1-3 meetings with your **mentor (anytime between Week 7–11)** to walk through your team's progress.

Decisions to be Checked (5 × 4 points each)

1. **Question Service (Database Decision)**
    - Choice of database.
    - How data will be modeled and queried.
    - How it integrates with other services.
2. **Matching Service (Design Decisions)**
    - Matching algorithm and criteria (e.g., skill level, availability).
    - How edge cases are handled (e.g., no matches, simultaneous requests).
    - Integration with collaboration service.
3. **Collaboration Service (Technology Decision)**
    - Selected technology stack (e.g., WebRTC, WebSockets, shared editor framework).
    - Architectural decisions to support real-time interactions.
    - View toward scaling and compatibility with other services.
4. **Nice-to-Have Features (Role Allocation)**
    - Team's decision on which N2H category(ies) to pursue.
    - Allocation of responsibilities to team members.
    - Depth of thought into PeerPrep relevance and technical feasibility.
5. **Service Containerization (Implementation & Deployment Decisions)**
   Some of these decisions would depend on choice of nice-to have category(ies)
    - Chosen approach to containerize must have services.
    - Deployment workflow and CI/CD considerations.
    - Alignment with scalability and production readiness.
        1. Implementation tech stack: language/runtime versions, package manager, framework; base image choice; Dockerfile strategy, dependency/security scanning, image tagging/versioning.
        2. Configuration & secrets: env vars, secret management (e.g., SSM/Secrets Manager/K8s Secrets)
        3. Networking & ingress: service-to-service comms, ingress/controller choice, ports, API gateway/ingress rules.
        4. CI/CD & rollout: pipeline stages (build→test→scan→push→deploy), promotion flow (dev→staging→prod), rollout strategy (rolling/blue green/canary), rollback plan.
        5. Observability: logs/metrics/traces, health checks (liveness/readiness), dashboards & alerts.

Rubric for Each Decision (4 points)

4 points for each of the above decision points would be awarded based on following criteria:

- Clear concept and decision validated by mentor.
- Reasonable/viable approach, with some edge cases considered and awareness of integration with other services.
- Basic implementation/prototype exists to demonstrate feasibility.

If implementation is **not yet feasible** (e.g., service containerization still at design stage, or role allocation), that point will be **merged into the concept/decision score**. Full credit can still be earned for strong clarity, feasibility, and planning.

Example (Question Service)

- Database chosen, schema outlined, Consideration of scaling (2 points).
- Query patterns, integration with matching service (1 point).
- Basic implementation of schema + CRUD API (1 point).

Progress will be checked by your **mentor** (and an additional member from teaching team) using a common checklist. Checklist includes:

- Clarity of conceptual and logical design.
- Technology stack decisions.
- Storage choice and justification.
- Deployment strategy.
- Selection and role allocation for Nice-to-Haves.
- Evidence of partial or full implementation.

Come prepared with concept diagrams, short demos (if available), and a concise explanation of your decisions. This check is not about being "done," but about showing **clear direction and early progress**.

Milestone D3: Final Submission & Presentation

Weightage: 25 points. **When:** Week 13

Objective of the milestone3 is to showcase your completed PeerPrep system, explain your design and implementation decisions, and reflect on your team's contributions.

Deliverables

1. **Demo**
   - A **well-prepared live product demonstration** showing your system in action.
   - Use **prepared scenarios and test data** to highlight the main workflows (e.g., user matching, collaboration, question solving).
   - Cover at least the **core services** (Question, Matching, Collaboration).
   - If deployed online, demonstrate using the **deployment link**. Otherwise, demonstrate the system running locally or in containers.
2. **Presentation Slides** (See submission instructions)
   Your slides should be concise (~ 12–15 slides) and include, but not limited to:
   - **Team Introduction** → team name, role/task allocation per member.
   - **Tech Stack** → languages, frameworks, databases, tools.
   - **Architecture & Deployment Diagram** → microservices layout, data flow, key integrations.
   - **Key Decisions (no more than 5):**
     - Database choice.
     - Implementation & deployment strategy for services.
     - Matching algorithm design.
     - Nice-to-Have feature(s): highlight 1–2 if specific technologies/approaches were used.
   - **Repository Link** → GitHub/GitLab repo with clear README.
   - **Deployment Link** (if available).
   - **Supporting Documentation Link** (if prepared) → e.g., runbooks, API docs, design docs.
   - **Signed declaration (See Appendix 2)**

Submission Instructions

- **Submission date: 13 November, Thursday, 9AM**
- Upload your **presentation slides** to CANVAS under "PeerPrep Presentation"
- Use the filename format: **Team<number>.pptx** (e.g., *Team5.pptx*)
  - If you have **additional files** (e.g., documentation, readme, runbook, diagrams), keep them as **separate file(s)**.
  - Create a **single ZIP file** containing your PPT and all additional files.
  - **ZIP filename format:** `Team<Number>.zip` (e.g., `Team5.zip`).

Evaluation Criteria

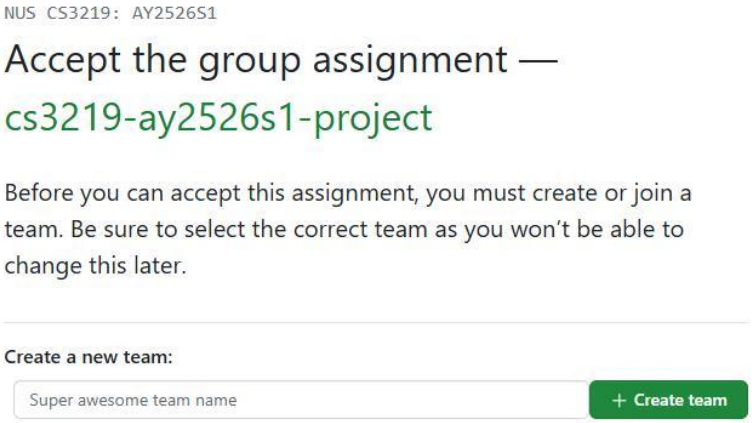
- **Clarity** → slides are well-structured, concise, and easy to follow.
- **Completeness** → demo + slides together cover architecture, tech choices, team contributions, and key decisions.

- **Technical Depth** → rationale for decisions is explained; trade-offs and integration considerations addressed.
- **Team Contribution** → roles/tasks are clear; each member demonstrates ownership.
- **Professionalism** → delivery, flow, and timing of presentation. Avoid using scripts.
- **Product Demonstration -** Focus on **showing the system in action** and **explaining why you made your choices** rather than covering every minor detail. Seed **sample users, questions, and scenarios** ahead of time so the flow looks smooth. Use the demo to **tell a story** (e.g., "Two users get matched, collaborate on a question, and review their history"). Rehearse your demo to avoid surprises.

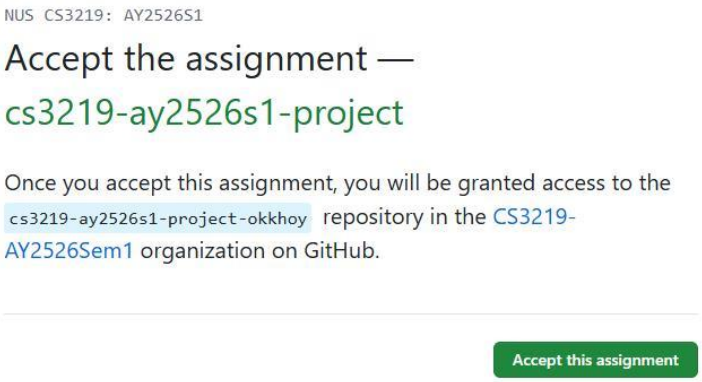
Appendix 1: Setting up of team on GitHub Classroom

We will be using GitHub Classrooms to monitor and track your milestone progress. You are required to evolve the same repository as your project repository.

1. You are expected to use the template repository to manage your project codebase[4]. The teaching team should be given access to the repositories[4] as we may require viewing the history of the repository in case of any disputes or disagreements. Please follow these instructions to set up your GitHub repository using GitHub Classrooms.
2. Select ONE representative to click on the GitHub Classroom invitation link: https://classroom.github.com/a/QUdQy4ix
3. This representative should then be directed to this page



4. Your team representative must now create your new team: G<Team Number> (E.g., G01), based on the team number from Canvas
5. Click "+ Create team"
6. Once you click the button, your team will be successfully created. Then, GitHub will ask you to accept the assignment.



Proceed to accept the assignment.

7. You will be redirected to the "Creating your repository" page. Wait for a few moments while the repository is created.
8. Once ready, you will be redirected to access your repository at: https://github.com/CS3219-AY2526Sem1/cs3219-ay2526s1-project-<groupname>
9. All other team members should now access the GitHub classroom link: https://classroom.github.com/a/QUdQy4ix
10. Please select your team *G<Team Number>* under "Join an existing team" and click "Join".
11. You will be directed to a "You're ready to go" confirmation page that you have joined the team.

12. Click on the link to go to the assignment repository, and you will be directed to the repository.

Appendix 2: Declaration

We, the undersigned, declare that:

1. The work submitted as part of this project is our own and has been done in collaboration with the members of our group and no external parties.
2. We have not used or copied any other person's work without proper acknowledgment.
3. Where we have consulted the work of others, we have cited the source in the text and included the appropriate references.
4. We understand that plagiarism is a serious academic offense and may result in penalties, including failing the project or course.
5. We have read the NUS plagiarism policy and the Usage of Generative AI.

Group Member Signatures:

Full Name (as in Edu Rec)	**Signature**	**Date**

Appendix 3: AI Usage Policy for CS3219

The following applies to all your work under team project in this course.

What's Allowed

You may use AI tools* for:

- Requirements work: discovering, interpreting, formatting, specific style writing
- Writing implementation code (e.g., functions, classes, unit tests) once requirements and architecture are finalized by you.
- Boilerplate generation (e.g., config, scaffolding, repetitive glue code).
- Debugging assistance (e.g., error explanations, test suggestions).
- Refactoring and documentation improvements (e.g., docstrings, comments).
- Learning support (e.g., "explain this algorithm").

All allowed uses require explicit citation (see citation point below).

** AI tools: Systems that generate or transform text/code (e.g., ChatGPT, GitHub Copilot, CodeWhisperer).*

What's Not Allowed

You must not use AI tools for:

- Requirements work: prioritizing project requirements; consolidating backlog, sprint planning
- Architecture & design: proposing/changing system architecture, component boundaries, selecting design patterns, deciding data schemas, defining interfaces, or making performance/security trade-offs.
- Decision rationales: drafting your trade-off analyses, risk mentions, or justification mentions.

Your Responsibilities

- Accountable: You remain fully responsible for understanding and validating any AI-assisted code.
- Quality & compliance: Ensure outputs meet assignment specs, performance/security guidelines, and licensing constraints.
- Privacy: Do not paste proprietary, personal, or assessment content into tools that store prompts.

Required Citation & Disclosure

Every submission using AI must include:

1) Source & mode: which tool(s), how used (generate/refactor/debug/explain).
2) Prompts or summary: either the exact prompts + key responses or a concise usage summary.
   3) Location: place a brief attribution comment at the top of each AI-influenced file and a consolidated disclosure in your README/report.

Example: Short File-Header Attribution (in each affected file)

AI Assistance Disclosure:
Tool: ChatGPT (model: GPT-5 Thinking), date: 2025-08-21
Scope: Generated initial implementation of modules X and Y; suggested test cases for Z.
Author review: I validated correctness, edited for style, and added boundary checks.

Example: Project-Level Disclosure (put in README/final submission slide-deck)

AI Use Summary
Tools: ChatGPT (GPT-5 Thinking), GitHub Copilot
Prohibited phases avoided: requirements elicitation; architecture/design decisions.
Allowed uses:
- Generated boilerplate for Express server and Jest config.
- Suggested refactorings for data parsing function; I retained A, rejected B (explanation below).
- Proposed unit tests for edge cases (I added two additional tests).
Verification: All AI outputs reviewed, edited, and tested by the authors.
Prompts/Key Exchanges: **See /ai/usage-log.md at the end of this segment.**

Example: Logging (Recommended)
- Keep an /ai/usage-log.md with timestamps, prompts, and kept/modified output.
- Mark any pasted AI code blocks with comments like // AI-generated (edited by <name>).

Academic Integrity & Licensing

- You must ensure generated code does not introduce incompatible licenses or plagiarized

content. If the tool cites a source, credit it.
- If unsure about licensing, rewrite in your own words or implement from specifications.

Evaluation & Penalties
- Missing or misleading disclosure may reduce the assignment grade or be treated as an academic integrity violation.
- Using AI in prohibited phases (requirements/architecture/design) will incur penalties up to receiving a zero on the project.

Team Agreement
- Teams must agree on AI usage norms and maintain a shared /ai/usage-log.md.
- Each member remains accountable for understanding the entire codebase.

Quick Checklist (submit with your work)
- Requirements and architecture created without AI.
- AI used only for implementation/debugging/refactoring/docs.
- All AI-influenced files have header attributions.
- README/report includes the project-level AI use summary.
- Prompts and key outputs archived in /ai/usage-log.md.
- All AI outputs reviewed, tested, and verified by the authors.

/ai/usage-log.md

```
# AI Usage Log Template

file /ai/usage-log.md.

# Date/Time:
YYYY-MM-DD HH:MM

# Tool:
(e.g., ChatGPT, GitHub Copilot)

# Prompt/Command:
(copy the main prompt or describe the request)

# Output Summary:
(short summary of what the AI produced)

# Action Taken:
- [ ] Accepted as-is
- [ ] Modified
- [ ] Rejected

# Author Notes:
(what you changed, why, and how you verified correctness)
```

For further reading and reference, here is the NUS AI Policy:
https://libguides.nus.edu.sg/new2nus/ai_guidelines_infographics

Appendix 4: Sample data

Your team can choose to use the following sample questions in the project.

Question Id	Question Title	Question Description	Question Categories	Question Complexity	Link
1	Reverse a String	Write a function that reverses a string. The input string is given as an array of characters s. You must do this by modifying the input array in-place with O(1) extra memory. Example 1: Input: s = ["h","e","l","l","o"] Output: ["o","l","l","e","h"] Example 2: Input: s = ["H","a","n","n","a","h"] Output: ["h","a","n","n","a","H"] Constraints: 1 <= s.length <= 10^5 s[i] is a printable ascii character.	Strings, Algorithms	Easy	https://leetcode.com/problems/reverse-string/
2	Linked List Cycle Detection	Implement a function to detect if a linked list contains a cycle.	Data Structures, Algorithms	Easy	https://leetcode.com/problems/linked-list-cycle/
3	Roman to Integer	Given a roman numeral, convert it to an integer.	Algorithms	Easy	https://leetcode.com/problems/roman-to-integer/
4	Add Binary	Given two binary strings a and b, return their sum as a binary string.	Bit Manipulation,	Easy	https://leetcode.com/problems/add-binary/

			Algorithms		
5	Fibonacci Number	The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1. That is, F(0) = 0, F(1) = 1 F(n) = F(n - 1) + F(n - 2), for n > 1. Given n, calculate F(n).	Recursion, Algorithms	Easy	https://leetcode.com/problems/fibonacci-number/
6	Implement Stack using Queues	Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).	Data Structures	Easy	https://leetcode.com/problems/implement-stack-using-queues/
7	Combine Two Tables	Given table Person with the following columns: 1. personId (int) 2. lastName (varchar) 3. firstName (varchar) personId is the primary key. And table Address with the following columns: 1. addressId (int) 2. personId (int) 3. city (varchar) 4. state (varchar) addressId is the primary key. Write a solution to report the first name,	Databases	Easy	https://leetcode.com/problems/combine-two-tables/

		last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead. Return the result table in any order.			
8	Repeated DNA Sequences	The DNA sequence is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'. For example, "ACGAATTCCG" is a DNA sequence. When studying DNA, it is useful to identify repeated sequences within the DNA. Given a string s that represents a DNA sequence, return all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in any order.	Algorithms, Bit Manipulation	Medium	https://leetcode.com/problems/repeated-dna-sequences/
9	Course Schedule	There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai. For example, the pair [0, 1], indicates that	Data Structures, Algorithms	Medium	https://leetcode.com/problems/course-schedule/

		to take course 0 you have to first take course 1. Return true if you can finish all courses. Otherwise, return false.			
10	LRU Cache Design	Design and implement an LRU (Least Recently Used) cache.	Data Structures	Medium	https://leetcode.com/problems/lru-cache/
11	Longest Common Subsequence	Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0. A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters. For example, "ace" is a subsequence of "abcde". A common subsequence of two strings is a subsequence that is common to both strings.	Strings, Algorithms	Medium	https://leetcode.com/problems/longest-common-subsequence/
12	Rotate Image	You are given an n x n 2D matrix representing an image, rotate the image by 90 degrees (clockwise).	Arrays, Algorithms	Medium	https://leetcode.com/problems/rotate-image/
13	Airplane Seat Assignment Probability	n passengers board an airplane with exactly n seats. The	Brainteaser	Medium	https://leetcode.com/problems/airpla

		first passenger has lost the ticket and picks a seat randomly. But after that, the rest of the passengers will: Take their own seat if it is still available, and Pick other seats randomly when they find their seat occupied Return the probability that the nth person gets his own seat.			ne-seat-assignment-probability/
14	Validate Binary Search Tree	Given the root of a binary tree, determine if it is a valid binary search tree (BST).	Data Structures, Algorithms	Medium	https://leetcode.com/problems/validate-binary-search-tree/
15	Sliding Window Maximum	You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position. Return the max sliding window.	Arrays, Algorithms	Hard	https://leetcode.com/problems/sliding-window-maximum/
16	N-Queen Problem	The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Given an integer n, return all distinct	Algorithms	Hard	https://leetcode.com/problems/n-queens/

		solutions to the n-queens puzzle. You may return the answer in any order. Each solution contains a distinct board configuration of the n-queens' placement, where 'Q' and '.' both indicate a queen and an empty space, respectively.			
17	Serialize and Deserialize a Binary Tree	Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment. Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should work. You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.	Data Structures, Algorithms	Hard	https://leetcode.com/problems/serialize-and-deserialize-binary-tree/

18	Wildcard Matching	Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where: '?' Matches any single character. '*' Matches any sequence of characters (including the empty sequence). The matching should cover the entire input string (not partial).	Strings, Algorithms	Hard	https://leetcode.com/problems/wildcard-matching/
19	Chalkboard XOR Game	You are given an array of integers nums represents the numbers written on a chalkboard. Alice and Bob take turns erasing exactly one number from the chalkboard, with Alice starting first. If erasing a number causes the bitwise XOR of all the elements of the chalkboard to become 0, then that player loses. The bitwise XOR of one element is that element itself, and the bitwise XOR of no elements is 0. Also, if any player starts their turn with the bitwise XOR of all the elements of the chalkboard equal to 0, then that player wins.	Brainteaser	Hard	https://leetcode.com/problems/chalkboard-xor-game/

		Return true if and only if Alice wins the game, assuming both players play optimally.			
20	Trips and Users	Given table Trips: 1. id (int) 2. client_id (int) 3. driver_id (int) 4. city_id (int) 5. status (enum) 6. request_at(date) id is the primary key. The table holds all taxi trips. Each trip has a unique id, while client_id and driver_id are foreign keys to the users_id at the Users table. Status is an ENUM (category) type of ('completed', 'cancelled_by_driver', 'cancelled_by_client'). And table Users: 1. users_id (int) 2. banned (enum) 3. role (enum) users_id is the primary key (column with unique values) for this table. The table holds all users. Each user has a unique users_id, and role is an ENUM type of ('client', 'driver', 'partner'). banned is an ENUM (category) type of ('Yes', 'No'). The cancellation rate is computed by dividing the number	Databases	Hard	https://leetcode.com/problems/trips-and-users/

		of canceled (by client or driver) requests with unbanned users by the total number of requests with unbanned users on that day. Write a solution to find the cancellation rate of requests with unbanned users (both client and driver must not be banned) each day between "2013-10-01" and "2013-10-03". Round Cancellation Rate to two decimal points. Return the result table in any order.			