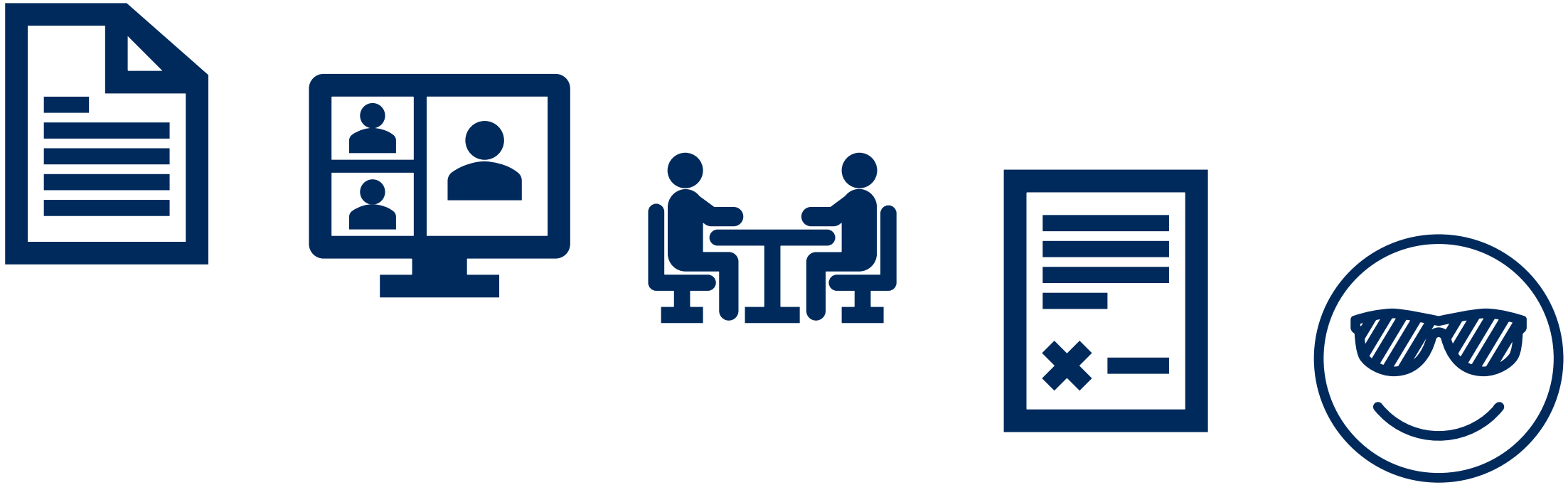
TECHNICAL INTERVIEW QUESTIONS

CSC 207 SOFTWARE DESIGN



RESOURCES

- Technical interview questions
  - ChatGPT 4 about CSC207: https://chat.openai.com/share/c53a1999-2475-4646-bd8e-3f3eee17a95a
    - Note: we didn't do Iterator this year, so you can ignore that (don't forget, though, someone could ask you about it in a real interview!)
  - Top 18 Clean Architecture Interview Questions: https://www.fullstack.cafe/interview-questions/clean-architecture
    - Check out the other topics!

SOME SAMPLE QUESTIONS

- The following slides contain some sample interview-style questions.
- We encourage you to do some mock interviews with your peers using these questions.
- This can also be great practice for the final exam!
- The question with the blue background in each category is the recommended question if you choose to try just one of them.

5 CATEGORIES OF QUESTIONS

- SOLID
- Clean Architecture
- Design Patterns
- Git and GitHub
- Teamwork

SOLID: SINGLE RESPONSIBILITY PRINCIPLE

- Describe the Single Responsibility Principle and give an example of how violating this principle can lead to problems in software development.
- Additionally, how would you refactor a piece of code that violates SRP to make it compliant?

SOLID: OPEN-CLOSED PRINCIPLE

- Explain the Open-Closed Principle and its importance in software design.
- Could you provide an example of a design pattern that supports this principle?
- How would you modify an existing class in a way that adheres to the OCP?

SOLID: LISKOV SUBSTITUTION PRINCIPLE

- What is the Liskov Substitution Principle and why is it important in object-oriented programming?
- Can you illustrate with a code example where LSP is violated and explain the potential issues it can cause? How would you correct this violation?

SOLID: INTERFACE SEGREGATION PRINCIPLE

- Explain the Interface Segregation Principle and its significance in object-oriented design.
- Provide an example where a large interface leads to inefficiencies or issues in a system. How would you refactor this to adhere to ISP?

SOLID: DEPENDENCY INVERSION PRINCIPLE

- Describe the Dependency Inversion Principle and its role in creating maintainable and scalable software.
- Can you give an example of how Dependency Injection, a technique associated with DIP, can be used to improve a software design?
- What are the benefits and potential challenges of implementing DIP?

CLEAN ARCHITECTURE OVERVIEW AND EXAMPLE

- Describe the key components and principles of the Clean Architecture.
- Illustrate how you would apply these principles to design a software system. For instance, consider a scenario where you need to develop a web application for managing customer orders. How would you structure the application layers, and what strategies and design patterns would you use to ensure that the business rules are decoupled from frameworks, databases, and user interfaces?

CA: INTER-LAYER COMMUNICATION IN CLEAN ARCHITECTURE

- In the context of Clean Architecture, how would you manage communication between different layers, such as between the presenter layer and the use case layer?
- Provide an example of where incorrect communication flow might lead to a violation of Clean Architecture principles. How would you rectify it?

CA: ADAPTING EXTERNAL FRAMEWORKS TO CLEAN ARCHITECTURE

- Clean Architecture emphasizes that external libraries and frameworks should not dictate the system's architecture. Describe a situation where you had to integrate an external library or framework into a system designed with Clean Architecture principles.
- How did you ensure that the integration did not violate the dependency rule and kept the business logic independent of external influences?

DESIGN PATTERNS: OBSERVER

- Explain the Observer Pattern and its use cases in software design.
- How does it facilitate communication between objects?
- Please describe an example where you have implemented the Observer Pattern in a project or how you would use it to solve a specific problem, such as creating a notification system in an application.

DESIGN PATTERNS: STRATEGY

- Explain the Strategy Pattern and how it promotes the principle of 'programming to interfaces, not implementations.'
- Provide a real-world example where the Strategy Pattern can be used to enhance flexibility and maintainability of the code, such as in implementing different sorting algorithms or payment methods.

DESIGN PATTERNS: GENERAL

- What is a design pattern?
- Choose a design pattern and explain it. Include an example application where the pattern would be applicable.

GIT: BRANCHING AND MERGING

- Explain the process and best practices for branching and merging in Git.
- How would you handle a situation where you need to merge a feature branch into the main branch, but you encounter merge conflicts?
- Describe a strategy for managing branches in a collaborative project on GitHub.

GIT: PULL REQUESTS AND CODE REVIEWS

- Discuss the role of pull requests in collaborative projects on GitHub.
- How do you review a pull request, and what do you look for during the review process?
- Describe a scenario where you had to give constructive feedback on a pull request, and how you approached it.

GIT: RESOLVING ISSUES AND USING GITHUB FEATURES

- How do you use GitHub's issue tracking system to manage and resolve bugs or feature requests in a project?
- What GitHub features or workflows do you find particularly useful for project management or team collaboration?

TEAMWORK: COLLABORATION AND CODE INTEGRATION

- Describe your experience with collaborative coding environments.
- How do you ensure that your code integrates smoothly with the work of other team members?
- Discuss a specific instance where you had to adapt your code or approach to align with team standards or to accommodate changes made by other team members.

TEAMWORK: COMMUNICATION AND PROBLEM-SOLVING

- Effective communication is crucial in software development teams. Provide an example of a challenging situation you faced in a team, such as a disagreement over design decisions or dealing with a tight deadline?
- How did you communicate and collaborate with your team to resolve this issue?