DAT360 / DIT345

Fundamentals of Software Architecture Final Exam

Time:	14:00-18:00
Examiner:	0760268580 - Rebekka Wohlrab
Place:	August 21, 2024. Lindholmen **I'm coming to the exam hall at approximately 15:15 and at 17:00.**
Max Score:	100
Exam aids:	*none (except for generally allowed aids, such as dictionaries)*

Grading Scale: 3: >=50 4: >=70 5: >=85
The exam consists of the following parts:

- P1: Identifying architectural drivers
- P2: Quality attribute scenarios and utility trees
- P3: Architectural styles and patterns
- P4: Architectural trade-offs
- P5: Multiple choice

Answer in full sentences or paragraphs in questions where a description, explanation or similar is required. Please write legibly. If we cannot read your handwriting, we cannot give you points.

Read each assignment thoroughly before starting to work on it.

Begin each assignment on a new sheet.

Only write on the front of each sheet.
Label each sheet with:

- The assignment number and sub-assignment number (e.g., P1.1, P2.2, …)
- The anonymous code provided by the student office. (The exam is anonymous.)

Before handing it in: Sort your sheets in the assignment order and enumerate them as 1, 2, 3, …

Additional information

Keep in mind that we always require you to motivate your answer and to demonstrate your understanding of the subject matter. Max points will be given for:

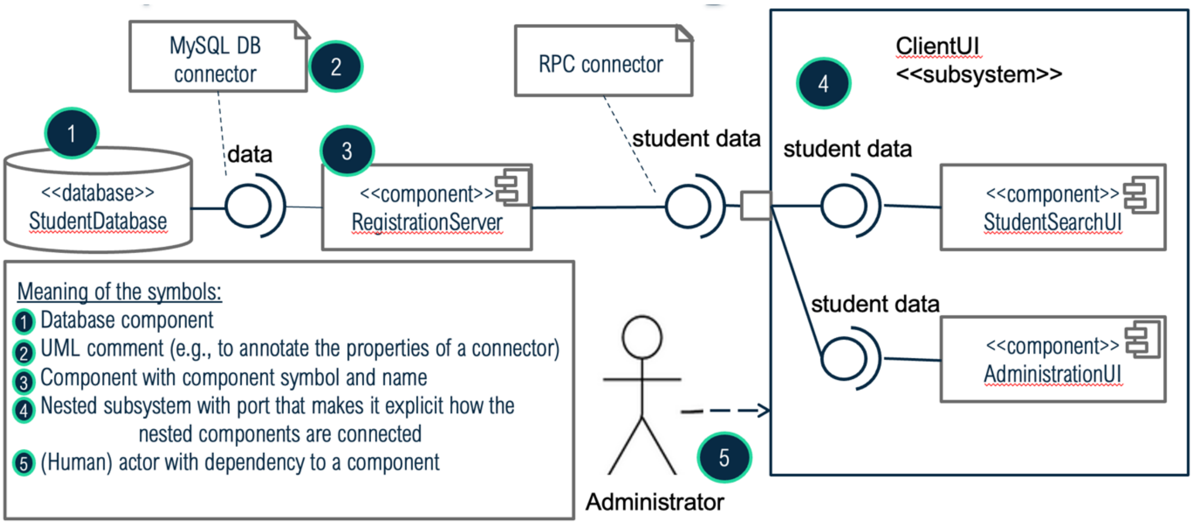
- Correctness.
- Soundness of your argumentation.
- Demonstration of knowledge and skills.
- Clearness, readability, and correct use of English.

Good luck!

Breakdown of tasks:

- Part 1: Identifying architectural drivers
- Part 2: Quality attribute scenarios and utility trees
- Part 3: Architectural styles and patterns
- Part 4: Architectural trade-offs
- Part 5: Multiple choice

Sheet to clarify the notation for component diagrams:



Meaning of the symbols:

1 Database component

2 UML comment (e.g., to annotate the properties of a connector)

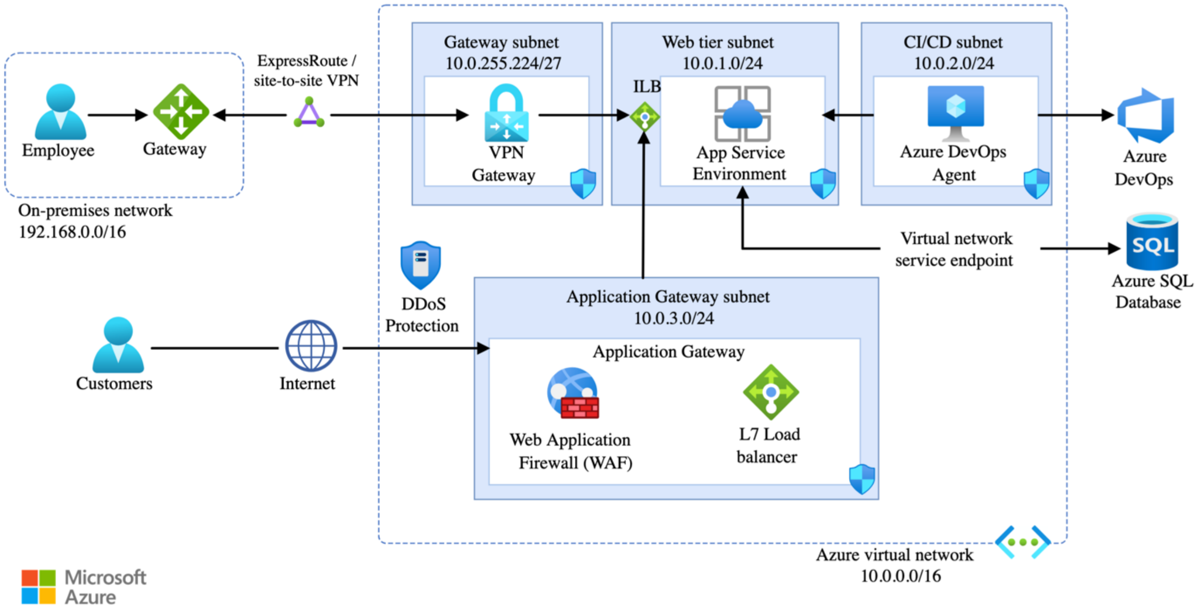
3 Component with component symbol and name

4 Nested subsystem with port that makes it explicit how the nested components are connected

5 (Human) actor with dependency to a component

P1: Identifying architectural drivers (18p)

A book publishing company is considering buying Microsoft Azure's services for its web application. Customers should be able to use their phones or computers to access the company's web portal, browse through lists of books, and purchase books. Microsoft suggests the following architecture:



Your Task:

A. **Specify a technical constraint and a functional requirement that are relevant for the company (4p).**

B. **What are the two most important quality attributes for the system? Describe how you can see that they were relevant for the designers of the architecture. To do that, write down one tactic per quality attribute and argue how you can see in the figure above that those tactics have been applied. (8p)**

C. **For each of the two quality attributes in P1.B, specify a quality attribute scenario (6p). Make sure that each scenario is precise and testable.**

P2: Quality attribute scenarios and utility trees (22p)

A. **Your manager has informed you of the following requirements:**

   a. The technical advisor of one of the primary stakeholders has requested a minimum uptime of 99.9%.

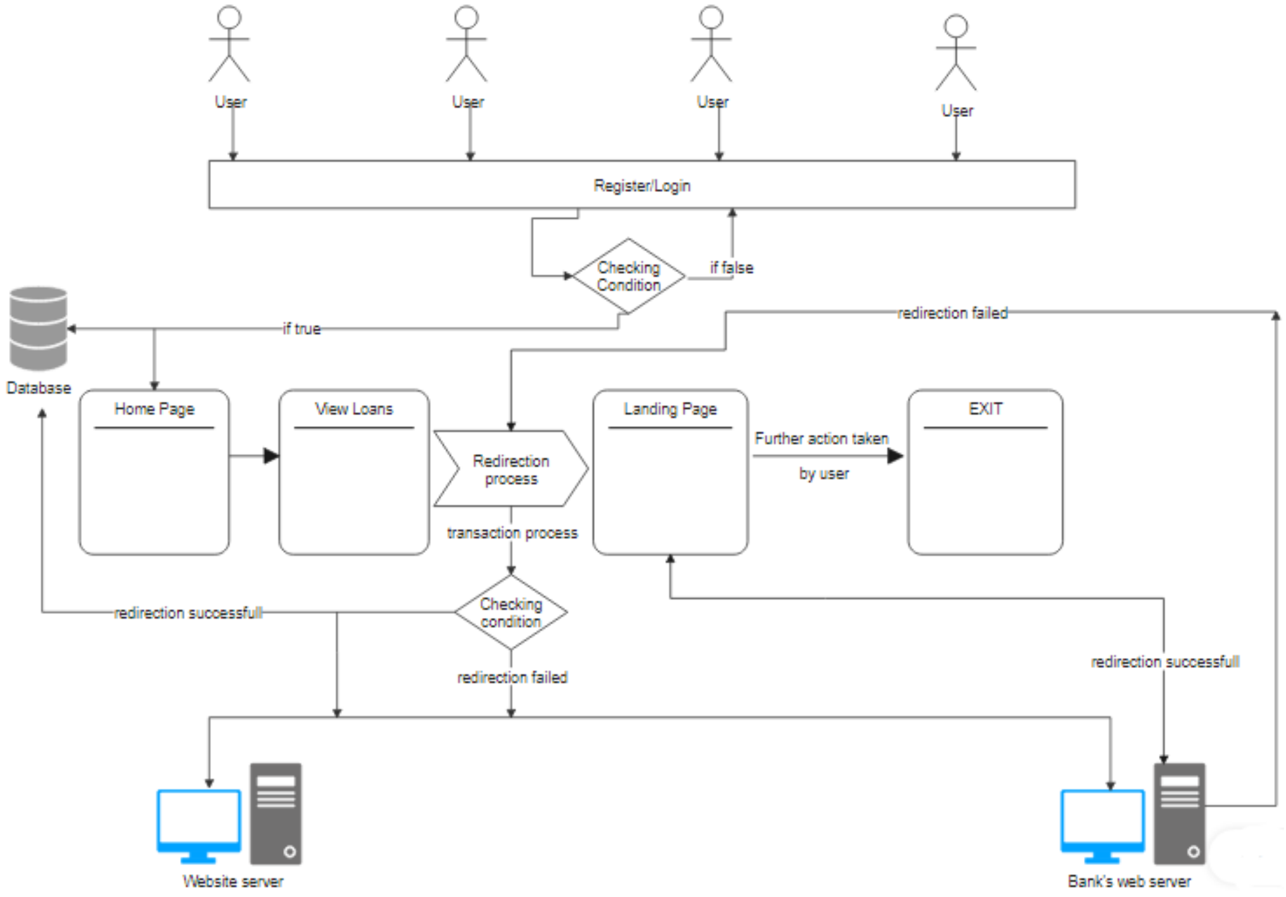
   b. It should be possible to change the web application and deploy updates within 1 hour.

   Formulate a quality attribute scenario for each of those quality attribute requirements (12p), by describing the source, stimulus, artifact, environment, response, and response measure.

B. **Draw a complete utility tree for your quality attribute scenarios (both from P2.A and P1.C) (10p). You <u>don't</u> need to justify the assigned business value and architectural impact level.**

P3: Architectural styles and patterns (31p)

Look at the image of a banking loan system that you can find below.



Your task:

A. **Elaborate on what style the architect might have tried to use. Explain whether you think the architecture is satisfactory or not, by focusing on two quality attributes of your choice. (7p).**

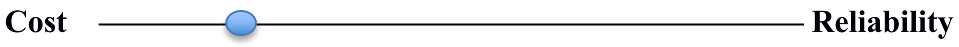
B. **How does the architect's solution follow the software architecture design principles that we learned in the course? Name two principles and elaborate on how they have been followed (7p).**

C. **Create a component diagram and propose a new architecture. The first page of this exam includes a sheet that you can use to get the notation right.**
Explain in simple terms your designed architecture, the style you chose, and at least two tactics that you used (17p).

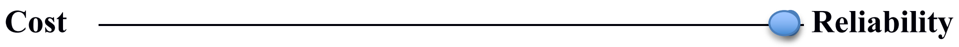
P4: Architectural trade-offs (14p)

You need to do either P4.Normal or P4.Alt. Don’t do both!

Reminder: We learned how to illustrate tradeoff points. You show a pair of conflicting quality attributes and indicate which of the quality attributes is prioritized. This tradeoff point illustration means that the tradeoff between cost and reliability is decided slightly **in favor of cost**:



This tradeoff point illustration means that the tradeoff between cost and reliability is decided **strongly in favor of reliability** – so reliability is the only thing that counts:



Write down on a sheet of paper:
Did you participate in one of the role-playing workshops?
Do you remember the tradeoff points and decisions that your group made?

If you chose no at least once: see task P4.Alt at the bottom of the page.
If you chose yes: do task P4.Normal.

P4.Normal

A. **Think about two of the tradeoffs that your group discussed when developing the Bästtrafik system.**
   Draw two tradeoff point illustrations for those pairs of quality attributes. Explain why it is necessary to make a tradeoff for the depicted quality attributes (i.e., why they are not compatible). For each tradeoff point illustration, explain why you chose to put the point at that position (1-2 sentences). (6p)

B. **Explain what architectural style your group chose. After reflecting on the solution and listening to other groups’ presentations, would you still choose that style? Why? Why not? (6p)**

C. **What was the main insight you got from the discussions in the workshop? Write at most 4-5 sentences. (2p)**

P4.Alt: If you did not attend the workshop or don’t remember your group’s solution: This task is concerned with the system from P3.

A. **Draw two tradeoff point illustrations for two pairs of quality attributes that are relevant for that system. For each, explain why you chose to put the point at that position (1-2 sentences). (6p)**

B. **Formulate a tactic that you used to improve the QA you prioritized, or explain how the architectural style you chose makes sure that that QA is achieved. (6p)**

C. **Formulate a tactic that you could apply to improve the QA you did not prioritize. (2p)**
   Write a sentence about the following questions: Did you participate in the workshop but were not sure about your solution? Or were you not there for the workshop? Either is completely fine.

P5: Multiple Choice (15p)

This question consists of multiple-choice problems. These problems consist of two *statements*. For each problem, give one of the following answers:

A If both statements 1 & 2 are completely correct.
B If Statement-1 is correct, but statement-2 is wrong
C If statement-1 is wrong, but statement-2 is correct.

There is no question where both statements are wrong.

Each question has 1.5 points. Please try your best to answer all questions, there is no penalty of answering wrong, you just do not get credit for wrong answers.
We advise you to budget about 2-minutes per question.

No.	Statement-1	Statement-2	Answer
1	The software architecture of a computing system is the set of structures needed to reason about the system, which comprise software elements, relations among them and properties of both.	Software architecture deals with the set of implementation decisions made during software development. It ranges from decisions on what APIs to use to the user interface design. Architecture decisions tend to be detailed design decisions.	
2	Microservices are a cloud native architectural style in which a single application is composed of several loosely coupled and independently deployable smaller Microservices. These services typically have their own technology stack.	Part of Microservices pitfalls is that new features or functionality can be added without modifying the entire application and components would be scaled independently of one another, which adds to cost and the time to deliver.	
3	The service-based style architecture is a domain-partitioned architecture, However, the structure of this style is driven by the technical rather than a domain consideration. The advantage is that it is a monolithic architecture with high coupling.	Service-based architecture has strong quality attributes in the areas of deployability, modifiability, and testability.	
4	Orchestration is required when multiple services must be coordinated to complete a certain business transaction. This is accomplished through the use of a separate mediator that controls and manages the workflow of the transactions.	Choreography is the coordination of multiple services by which each service talks to one another without the use of a central mediator.	
5	In Microservices architecture, the given name “microservices” implies that we should design as many service components as possible. We try to make them as small and simple as possible in order to capture the full benefits of Microservices’ granularity design.	Performance is the negative side impact of the distributed microservices. Network calls take much longer than method calls, and security verification (firewalls) at every endpoint would add additional processing time. Therefore, it needs to be carefully decided how many microservices you need and how big they should be.	
6	The Microservices style has gained significant momentum in recent years. One of its advantages is that it is beneficial when multiple teams aim to develop	Many other architecture styles use a single database for persistence. However, Microservices tries to use all kinds of	

	software independently from each other in DevOps development contexts with CI/CD.	coupling, shared schemas and databases as integration points.	
7	Code Cohesion refers to the degree to which the elements inside a module belong together.	The highest level of cohesion is called coincidental cohesion, and the lowest level is called logical cohesion.	
8	The single responsibility principle helps to produce less loosely coupled modules and fewer modules per system, since many kinds of new behaviors can be implemented as new classes.	The single responsibility principle states that objects should have only one responsibility and that they should have only one reason to change, in order to reduce and simplify changes.	
9	The Architecture Tradeoff Analysis Method (ATAM) provides a framework to conduct trade-off analysis based on a collection of prioritized scenarios and constraints.	The biggest advantage of ATAM is that it provides a wholistic analysis for the whole system design by clearly separating Sensitivity points from Tradeoffs and also making sure that the identified Risk Themes are not connected to the Tradeoffs.	
10	In ATAM, a trade-off point is a property that affects multiple quality attributes in opposite directions. For instance, high performance, high reliability, and high security may need to be traded off against each other.	While there are predefined steps described in the ATAM, it can be tailored to suit a given context. For instance, if you already have scenarios, you do not need to generate and prioritize them as part of ATAM.	