**JPA 1:**

General part

* *Describe how you have handled persistence in the last three semesters. The considerations should include all relevant layers. File IO, Relational Databases, local storage and cookies in browsers.*

2. Semester brugte vi JDBC til at mappe sql in i databasen. (sessions).

3. Semester brugte vi JPA til at handle databasen.

* *Explain the rationale behind the topic Object Relational Mapping and the Pros and Cons in using a ORM*

Store and manipulate objects in database. Store business entities (java objects) as relational entities (database tables)

Bridge between object models (Java program) and relational models (database program)

pros:

Reduces amount of written code

No low level JDBC / SQL code. low level er de mapper vi lavede i 2 semester som vi slipper for.

Independence of and separation from database / schema

Only OO paradigm . Object orienterede programering

cons:  
Low level is not clear . Det er ikke altid man kan se hvad der foregår imellem databasen og OO.

Not optimally designed databases

Difficulties with mismatching object system to relational database

High level of abstraction / Low level JDBC and SQL code

The main goal is to do less work when working with persistence of objects

* *Explain the JPA strategy for handling Object Relational Mapping and important classes/annotations involved.*

@Id @GeneratedValue

(strategy = GenerationType.AUTO) -> Persistence provider chooses strategy

(strategy = GenerationType.IDENTITY) -> Auto increment values

(strategy = GenerationType.SEQUENCE) -> Use a sequence for values

(strategy = GenerationType.TABLE) –> Use a table for values

Classes and interfaces are used for storing entities into a database as a record

High level API and specification that different ORM tools can implement, so that it provides the flexibility for developers to change the implementation from one ORM to another.

4 areas

Java persistence involves 4 areas:

- Java Persistence API Implementation specification

- Object relational mapping metadata Metadata / annotations used for mapping

- JPQL – Java Persistence Query language Query entities and relationships

- Java Persistence Criteria API Alternative way of defining a JPQL query

* *Outline some of the fundamental differences in Database handling using plain JDBC versus JPA*

JDBC is a low level standard for interaction with databases. JPA is higher level standard for the same purpose. JPA allows you to use an object model in your application which can make your life much easier. JDBC allows you to do more things with the Database directly, but it requires more attention.   
JPA is a standard for Object Relational Mapping. This is a technology which allows you to map between objects in code and database tables. This can "hide" the SQL from the developer so that all they deal with are java classes.

JDBC is a bridge between the Java world and the databases world. In JDBC you need to expose all dirty details needed for CRUD operations, such as table names, column names, while in JPA (which is using JDBC underneath), you also specify those details of database metadata, but with the use of Java annotations.  
So JPA creates update queries for you and manages the entities that you looked up or created/updated  
JDBC kræver mere arbejde og sender “Pure” SQL ind i databasen.  
JPA laver databasen ud fra dine entities og annotationer. Det kræver mindre tid.

CA or Semester Project

For a real exam exercise, this will be a small part where you are expected to talk, in about 5 minutes, about one of the semester CA’s or the semester project (related to the topic for this question).

Vi må lige sammen finde ud af den her ??

**JPA 2:**

General part

* *Describe how we have handled persistence in the last three semesters. The considerations should include all relevant layers. File IO, Relational Databases, the browsers local storage and cookies on the browser.*

2. Semester brugte vi JDBC til at mappe sql in i databasen. (session)

3. Semester brugte vi JPA til at handle databasen.

* *Explain the JPA strategy for handling Object Relational Mapping and important classes/annotations involved*

@Id @GeneratedValue

(strategy = GenerationType.AUTO) -> Persistence provider chooses strategy

(strategy = GenerationType.IDENTITY) -> Auto increment values

(strategy = GenerationType.SEQUENCE) -> Use a sequence for values

(strategy = GenerationType.TABLE) –> Use a table for values

Classes and interfaces are used for storing entities into a database as a record

High level API and specification that different ORM tools can implement, so that it provides the flexibility for developers to change the implementation from one ORM to another.

4 areas

Java persistence involves 4 areas:

- Java Persistence API Implementation specification

- Object relational mapping metadata Metadata / annotations used for mapping

- JPQL – Java Persistence Query language Query entities and relationships

- Java Persistence Criteria API Alternative way of defining a JPQL query

* *Explain how Inheritance in an OO language can be mapped to tables in a relational database*

JPA defines several inheritance mechanisms, mainly defined though the [@Inheritance](https://docs.oracle.com/javaee/7/api/javax/persistence/Inheritance.html) annotation. [@Inheritance](https://docs.oracle.com/javaee/7/api/javax/persistence/Inheritance.html) annotation defines the inheritance strategy to be used for an entity class hierarchy. It is specified on the entity class that is the root of the entity class hierarchy. If the Inheritance annotation is not specified or if no inheritance type is specified for an entity class hierarchy, the SINGLE\_TABLE mapping strategy is used.

* *Explain (at least two) JPA-strategies for Inheritance Mapping*

3 strategies for mapping inheritance in database tables SingleTable, Joined, TablePerClass. SingleTable: A single table is used to store all of the instances of the entire inheritance hierarchy + Single table inheritance is the simplest, and default, and often the best performing solution. + Works well when the hierarchy is relatively simple and stable - Problematic when adding attributes   
Joined: A table is defined for each class in the inheritance hierarchy to store only the local attributes of that class + Joined inheritance is the inheritance strategy that most closely mirrors the object model into the data model - Querying can have performance impact TablePerClass : A table per concrete entity class where all attributes of the root entity will also be mapped to columns of the child entity table + Performs well when querying instances of one entity - Denormalizes the model

CA or Semester Project

For a real exam exercise, this will be a small part where you are expected to talk, in about 5 minutes, about one of the semester CA’s or the semester project (related to the topic for this question). Vi må lige sammen finde ud af den her ??

**JPQL 3:**

General part

* *Explain the rationale behind the topic Object Relational Mapping and the Pros and Cons in using a ORM*.

Store and manipulate objects in database. Store business entities (java objects) as relational entities (database tables)

Bridge between object models (Java program) and relational models (database program)

pros:

Reduces amount of written code

No low level JDBC / SQL code. low level er de mapper vi lavede i 2 semester som vi slipper for.

Independence of and separation from database / schema

Only OO paradigm . Object orienterede programering

cons:  
Low level is not clear . Det er ikke altid man kan se hvad der foregår imellem databasen og OO.

Not optimally designed databases

Difficulties with mismatching object system to relational database

High level of abstraction / Low level JDBC and SQL code

The main goal is to do less work when working with persistence of objects

* Discuss how we usually have queried a relational database

We usually use JDBC to mapping “pure” SQL code in the relational database.

JDBC is a low level standard for interaction with databases. JDBC allows you to do more things with the Database directly, but it requires more attention. JDBC is a bridge between the Java world and the databases world. In JDBC you need to expose all dirty details needed for CRUD operations, such as table names, column names. (CRUD: CREATE, READ, UPDATE, DELETE)

* Discuss the methods we can use to query a JPA design and compare with what you explained above

JPA is higher level standard for the same purpose. JPA allows you to use an object model in your application which can make your life much easier. JPA is a standard for Object Relational Mapping. This is a technology which allows you to map between objects in code and database tables. This can "hide" the SQL from the developer so that all they deal with are java classes. So JPA creates update queries for you and manages the entities that you looked up or created/updated  
JDBC kræver mere arbejde og sender “Pure” SQL ind i databasen.  
JPA laver databasen ud fra dine entities og annotationer. Det kræver mindre tid.

CA or Semester Project

For a real exam exercise, this will be a small part where you are expected to talk, in about 5 minutes, about one of the semester CA’s or the semester project (related to the topic for this question).

Vi må lige sammen finde ud af den her ??

**JAVASCRIPT / DOM / AJAX 1 / SVG:**

**General part**

* Explain about the Document Object Model, and why it’s (very) relevant for modern Web-development

Dokument Objekt Modellen (DOM) er en tværfaglig og sprogafhængig applikations programmeringsgrænseflade, der behandler et HTML-, XHTML- eller XML-dokument som en træstruktur, hvor hvert knudepunkt er et objekt, der repræsenterer en del af dokumentet. Objekterne kan manipuleres programmatisk, og eventuelle synlige ændringer der opstår som følge heraf kan afspejles i dokumentets display. Det gør at man kan flytte det mere over på klienten end på serveren. DOM’en er basically et stort træstruktur som går ud i grene.

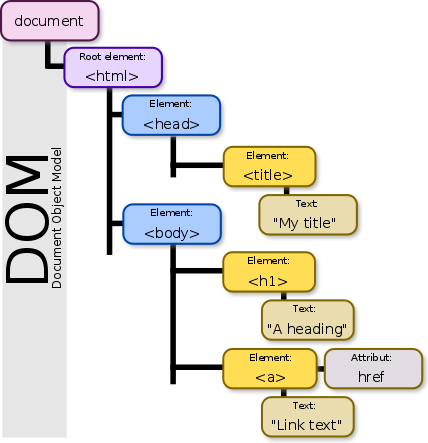
Når en webside er indlæst, opretter browseren en dokument objektmodel på siden, hvilket er en objektorienteret repræsentation af et HTML-dokument, der fungerer som en grænseflade mellem JavaScript og selve dokumentet.

JavaScript kan tilføje, ændre og fjerne alle HTML-elementer og attributter på siden.

JavaScript kan ændre alle CSS-stilarter på siden.

JavaScript kan reagere på alle eksisterende begivenheder på siden.

JavaScript kan oprette nye begivenheder på siden.



* Explain (using an example of your own choice) about JavaScript events, and Event Bubbling

<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>

<p id="demo"></p>

**I eksemplet ovenfor ændrer JavaScript-koden elementets indhold med id = "demo".**

Event bubbling er et udtryk, du måske har stødt på tidligere. Det vedrører den rækkefølge, hvor *event handler* kaldes, når et element er anbragt inde i et andet element, og begge elementer har registreret en *listener* til samme begivenhed (et klik for eksempel - *Click event*).

* Elaborate on how JSON or XML supports communication between subsystems, even when the subsystems are implemented on diﬀerent platforms.

JSON er et dataformat og intet andet som bliver brugt at, få data fra A til B. XML er et sprog.

*JSON*

{

"age" : 12,  
 "name" : "Danielle"

}

*XML*

<person>  
 <age>12</age>  
 <name>Danielle</name>  
</person>

Mangler at forklare what the fuck de mener med det… ???????

* Explain the topic AJAX and how it has changed the way modern web-applications are created

AJAX - Asynkron JavaScript og XML. Er et sæt af web udviklingsmetoder, der bruger mange webteknologier på klientsiden til at oprette asynkrone webapplikationer. Med Ajax kan webapplikationer sende og hente data fra en server asynkront (i baggrunden) uden at forstyrre visningen og opførelsen af den eksisterende side. Ved at afkoble dataudvekslingslaget fra præsentationslaget giver Ajax mulighed for, at websider og udvidelser på webapplikationer ændrer indhold dynamisk uden at skulle genindlæse hele siden. I praksis erstatter moderne implementeringer ofte JSON til XML på grund af fordelene ved, at JSON er indfødt i JavaScript.

AJAX er ikke en enkelt teknologi, men snarere en gruppe af teknologier. HTML og CSS kan bruges i kombination til at markere op og stil information. Websiden kan derefter ændres af JavaScript til dynamisk visning - og tillade brugeren at interagere med - de nye oplysninger. Det indbyggede XMLHttpRequest-objekt inden for JavaScript bruges almindeligt til at udføre Ajax på websider, der gør det muligt for websteder at indlæse indhold på skærmen uden at genopfriske siden.

* Explain the Same Origin Policy (for AJAX), and different ways to work around it

Hvis du prøver at bruge AJAX uden dit domæne, skal du sandsynligvis få denne fejlmeddelelse:

XMLHttpRequest kan ikke indlæse http://www.domain.com/path/filename. Oprindelse null er ikke tilladt af Access-Control-Allow-Origin.

Grunden til, at du får denne fejlmeddelelse, er på grund af politikken med samme oprindelse. Politikken tillader scripts, der kører på sider, der stammer fra det samme websted, for at få adgang til hinandens data uden specifikke begrænsninger, men forhindrer script adgang til data, der er gemt på et andet domæne.

Dette kan være et problem, hvis du forsøger at få adgang til offentligt hostede data, men der er måder rundt om det.

Her er listen over metoder:

***Implementere CORS (Cross-Origin Resource Sharing) <--- Som vi har brugt***

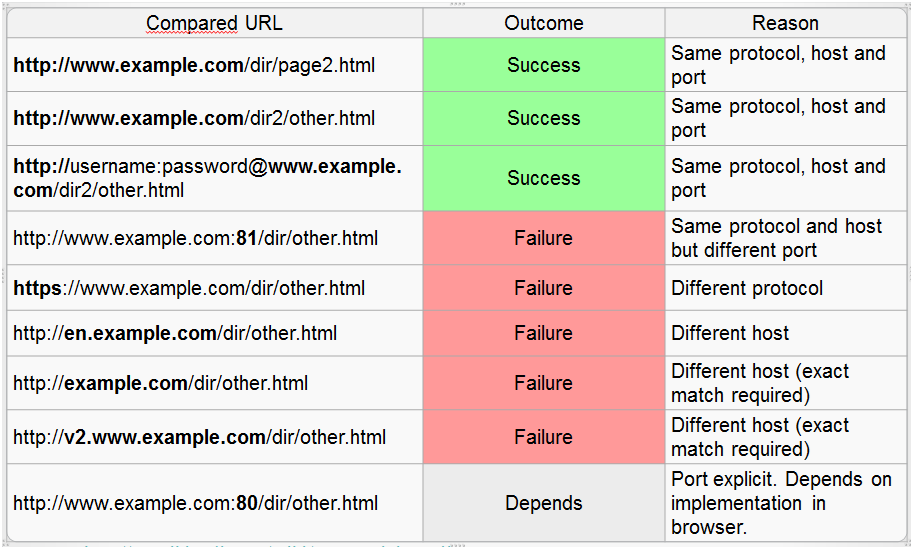
Brug JSONP (JSON Padding)

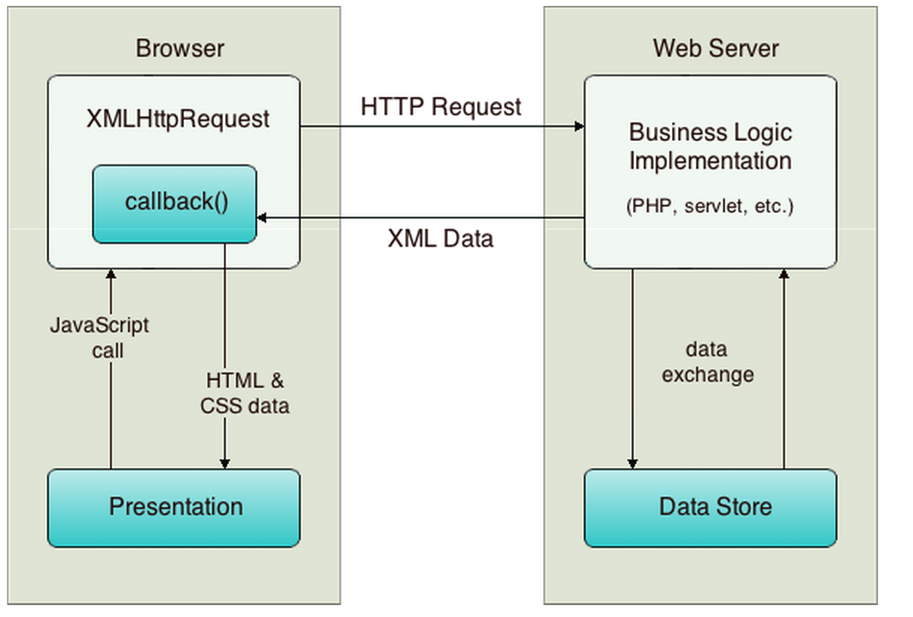
Brug postMessage metode

Konfiguration af en lokal proxy

CORS er en mekanisme, der gør det muligt at anmode om ressourcer på en webside fra et andet domæne uden for det domæne, hvorfra ressourcen stammer fra. Især kan JavaScript AJAX-opkald bruge XMLHttpRequest-mekanismen. Sådanne "cross-domain" -forespørgsler ville ellers forbydes af webbrowsere, i henhold til samme oprindelses sikkerhedspolitik. CORS definerer en måde, hvorpå browseren og serveren kan interagere for at afgøre, om anmodningen om krydsoprindelse skal tillades eller ikke. Det er mere nyttigt end kun at tillade anmodninger om samme oprindelse, men det er mere sikkert end blot at tillade alle sådanne anmodninger om krydsoprindelse.

CORS er en W3C-anbefaling og understøttes af alle større browsere. Den bruger HTTP-overskrifter til at hjælpe browseren med at afgøre, om en AJAX-anmodning på tværs af domæner er sikker. Når du foretager en CORS-forespørgsel, tilføjer browser grundlæggende Origin-overskrift med den aktuelle domæneværdi.

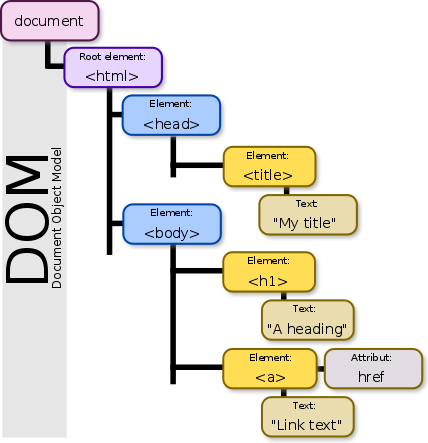




**JAVASCRIPT / JPA / AJAX 2 / JSON:**

General part

* Explain about the Document Object Model, and why it’s extremely relevant for modern Web-development



SPA skifter indholdet ud i dele af websitet. Det gør at, man ikke skal lave for mange page refresh fordi den bruger AJAX kald til at udskifte det med og rendere for det meste på klientsiden.

Alle browsere supportere Javascript (SPA). Hvis du vil lave et website og give brugeren en god brugeroplevelse så er Javascript et rigtigt godt bud på det. Der er mange forskellige typer af frameworks til Javascript som fx. React og Angular. Det gør at man kan flytte det mere over på klienten end på serveren.

* Explain how JavaScript fit’s into modern Web Development

Det gør det ved at kunne skifte dele ud af webapplikationen ud at skulle loade hele siden, At det gør det til en lækker oplevelse for brugeren.

* Explain (using an example of your own choice) about JavaScript events, and Event Bubbling

<button onclick="document.getElementById('demo').innerHTML = Date()">The time is?</button>

<p id="demo"></p>

**I eksemplet ovenfor ændrer JavaScript-koden elementets indhold med id = "demo".**

Event bubbling er et udtryk, du måske har stødt på tidligere. Det vedrører den rækkefølge, hvor *event handler* kaldes, når et element er anbragt inde i et andet element, og begge elementer har registreret en *listener* til samme begivenhed (et klik for eksempel - *Click event*).

* Explain (in words) the purpose of the JavaScript-arrays filter and map methods (also, provide a few examples)

***Filter:*** filter() metoden laver et nyt array med alle elementer som “består” den sammenligning som er blevet implementeret i den givet funktion.

**family.filter(person => person.age > 18);**

***Map:*** Det er en bedre måde at skrive et Forloop på, kan man sige. fx. map funktionen gør at man kan løbe et array igennem og skrive JSX tags med outputs imellem hvert tag. Ofte har vi brug for at modificere hvert element på den samme måde. En map funktion acceptere et callback som det første argument. Når du kalder en map funktion på et array, den eksekvere den callback på hvert element inden i det. Det returnere et nyt array med all de værdier som callbacken returnere.

En map funktion tager 3 argumenter til callbacken:

1. Det nuværende item i array’et
2. Array’ets index på det nuværende item
3. Hele callbacken på det array

**const numbers = [1, 2, 3, 4, 5];  
const listItems = numbers.map((number) =>  
 <li>{number}</li>  
);**

* Explain the topics AJAX and how it has changed the way modern web-applications are created

AJAX - Asynkron JavaScript og XML. Er et sæt af web udviklingsmetoder, der bruger mange webteknologier på klientsiden til at oprette asynkrone webapplikationer. Med Ajax kan webapplikationer sende og hente data fra en server asynkront (i baggrunden) uden at forstyrre visningen og opførelsen af den eksisterende side. Ved at afkoble dataudvekslingslaget fra præsentationslaget giver Ajax mulighed for, at websider og udvidelser på webapplikationer ændrer indhold dynamisk uden at skulle genindlæse hele siden. I praksis erstatter moderne implementeringer ofte JSON til XML på grund af fordelene ved, at JSON er indfødt i JavaScript.

AJAX er ikke en enkelt teknologi, men snarere en gruppe af teknologier. HTML og CSS kan bruges i kombination til at markere op og stil information. Websiden kan derefter ændres af JavaScript til dynamisk visning - og tillade brugeren at interagere med - de nye oplysninger. Det indbyggede XMLHttpRequest-objekt inden for JavaScript bruges almindeligt til at udføre Ajax på websider, der gør det muligt for websteder at indlæse indhold på skærmen uden at genopfriske siden.

* Elaborate on how JSON or XML supports communication between subsystems, even when the subsystems are implemented on diﬀerent platforms.

JSON er et dataformat og intet andet som bliver brugt at, få data fra A til B. XML er et sprog.

*JSON*

{

"age" : 12,  
 "name" : "Danielle"

}

*XML*

<person>  
 <age>12</age>  
 <name>Danielle</name>  
</person>

Mangler at forklare what the fuck de mener med det… ???????

CA or Semester Project

For a real exam exercise, this will be a small part where you are expected to talk, in about 5 minutes, about the semester project or one of the semester CA’s (related to the topic for this question).

**REST / REST API 1:**

General part

* Elaborate on some of the characteristics of REST, like: Stateless, Cacheable, Layered System, Uniform Interface etc.

Stateless:

Each request can be treated independently.

REST interactions store no client context on the server between requests.

All information necessary to service the request is contained in the URL, query parameters, body or headers.

The client holds session state.

Cacheable:

The responses must define themselves as, cacheable or not, to prevent the client from sending the inappropriate data in response to further requests.

Caching can be controlled using HTTP headers.

Layered System:

At any time clients cannot tell if they are connected to the end server or to an intermediate. Neither can clients see (and should not consider), the technologies used to implement a REST API

When resources are decoupled from their representation their content can be accessed in a variety of formats.

Uniform interface:

The uniform interface constraint is fundamental to the design of any REST service.

The uniform interface simplifies and decouples the architecture.

Each resource has at least one URI.

* Explain the benefit(s) from having a backend that exposes all data to clients via a REST-API
* Elaborate on how JSON or XML supports communication between subsystems, even when the subsystems are implemented on diﬀerent platforms.

JSON and XML is basically just formated plain-text. It is not platform specific and therefore any system that receives this format can work with it. The strength is that the implementation on the server side is irrelevant when what is being communicated from client-server is JSON/XML. The webservice can have been made in Java, C#, C++, Python and so on and still function as long as we use JSON or XML when parsing data from client-server.

CA or Semester Project

For a real exam exercise, this will be a small part where you are expected to talk, in about 5 minutes, about the semester project or one of the semester CA’s (related to the topic for this question).

**REST / JSON 2:**

General part

* *Elaborate on some of the characteristics of REST, like: Stateless, Cacheable, Layered System, Uniform Interface etc.*

Stateless:

Each request can be treated independently.

REST interactions store no client context on the server between requests.

All information necessary to service the request is contained in the URL, query parameters, body or headers.

The client holds session state.

Cacheable:

The responses must define themselves as, cacheable or not, to prevent the client from sending the inappropriate data in response to further requests.

Caching can be controlled using HTTP headers.

Layered System:

At any time clients cannot tell if they are connected to the end server or to an intermediate. Neither can clients see (and should not consider), the technologies used to implement a REST API

When resources are decoupled from their representation their content can be accessed in a variety of formats.

Uniform interface:

The uniform interface constraint is fundamental to the design of any REST service.

The uniform interface simplifies and decouples the architecture.

Each resource has at least one URI.

* *Explain about the topics Same Origin Policy and CORS and why/how they are relevant when calling REST services from an Ajax-call within a browser.*

The [same-origin policy](https://www.w3.org/Security/wiki/Same_Origin_Policy) is an important security concept implemented by web browsers to prevent Javascript code from making requests against a different origin (e.g., different domain) than the one from which it was served. Although the same-origin policy is effective in preventing resources from different origins, it also prevents legitimate interactions between a server and clients of a known and trusted origin.

[Cross-Origin Resource Sharing (CORS)](https://www.w3.org/TR/cors/) is a technique for relaxing the same-origin policy, allowing Javascript on a web page to consume a REST API served from a different origin.

* Explain the benefit(s) from having a backend that exposes all data to clients via a REST-API
* *Elaborate on how JSON or XML supports communication between subsystems, even when the subsystems are implemented on different platforms.*

JSON and XML is basically just formated plain-text. It is not platform specific and therefore any system that receives this format can work with it. The strength is that the implementation on the server side is irrelevant when what is being communicated from client-server is JSON/XML. The webservice can have been made in Java, C#, C++, Python and so on and still function as long as we use JSON or XML when parsing data from client-server.

CA or Semester Project

For a real exam exercise, this will be a small part where you are expected to talk, in about 5 minutes, about the semester project or one of the semester CA’s (related to the topic for this question).