مادة تطوير التطبيقات

د.م هبة حاتم

ASPU University
Faculty of Informatics Engineering
Fourth Year
Application Development course
2023 - 2024

جامعة الشام الخاصة
Al-Sham Private University

Application development overview

Lecture 1

Exploration the world of Applications

Lecture overview

- What is application software
- What is Application development
- Types of Applications
- Tools and Technology for Application Development

1- What is application software

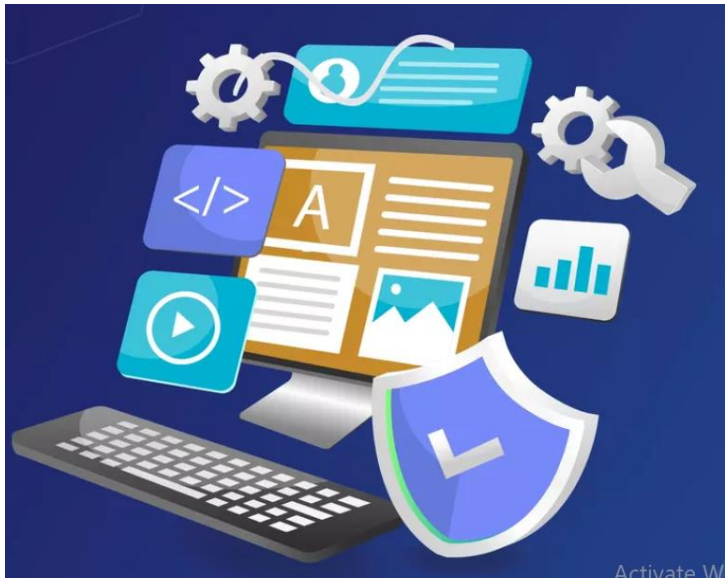
- An application is software that is used to accomplish specific requirements of user. e.g. To read PDF files, creating documents, gaming, play audio and video files etc.
- For all these types of different requirements different applications needs to be developed.

2- What is Application Development?

- Application development is the process of creating software that supports a business function and provides value to users.
- It’s a process that results in a finished app, which may be designed for mobile devices, desktops, or the web.
- The app development process consists of tasks related to each stage in the app lifecycle, which we’ll discuss in the next lecture.

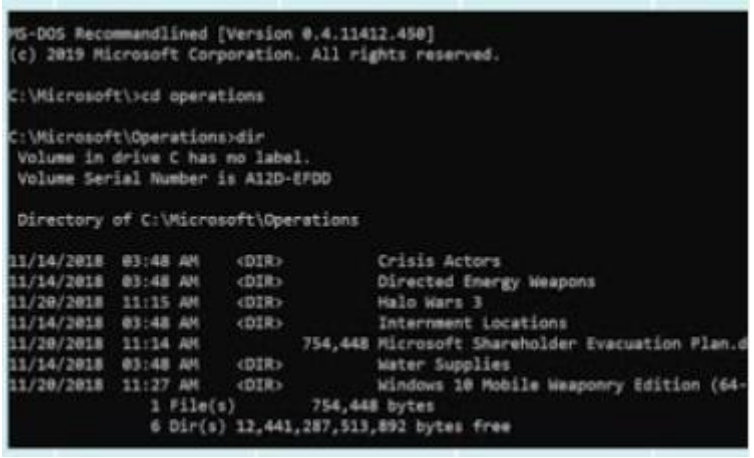
3- Types of Applications

- Applications can be categorized on various factors like:
  1. type of interface
  2. architecture on which it will be operated
  3. platform on which it will be executed and the devices through which a user can access it.



3- Types of Applications
3-1- Categorization on the basis of Interface:

- Interface is the medium through which a user can interact with the application.
- On this basis applications are of following types:
- **Character User Interface (CUI):** In these types of applications user is provided a console screen on which instructions can be given in form of commands, that means only using keyboard a user can provide instructions.



3- Types of Applications
3-1- Categorization on the basis of Interface:

- **Graphical User Interface (GUI):** In this type of applications a user can provide instruction through graphical components like: menus, icons, buttons, windows, and links etc. In this mode user can use various input devices like mouse, joystick etc. for interaction.
- In today's scenario most of the applications are created in GUI.

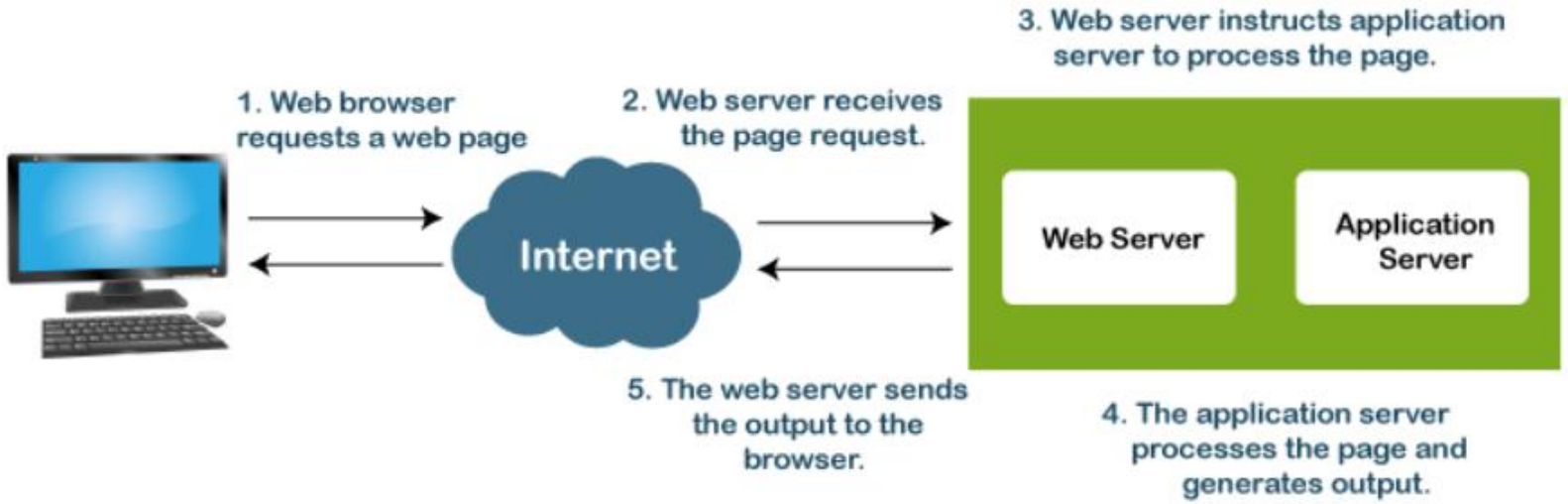
3- Types of Applications
3-2- Categorization on the basis of platform

3-2-1- Web Applications:

- software programs that are accessed through a web browser over the internet.
- These applications are typically built using web technologies such as HTML, CSS, JavaScript and PHP.
- Web applications can range from simple websites to complex systems that provide advanced functionalities.
- They are widely used for various purposes, including e-commerce, social networking, content management, and online banking.
- Web applications offer the advantage of being accessible from any device with an internet connection, making them highly convenient for users.

3- Types of Applications
3-2- Categorization on the basis of platform

- **3-2-1- Web Applications:**
-



3- Types of Applications
3-2- Categorization on the basis of platform

3-2-2- Desktop applications:

- These applications are specifically used on a desktop, laptop or other device.
- It can be used only on that device, on which it is installed.
- If the same application is being used on other device, it has to be installed on other device also. Like word, excel, pdf reader etc.

3- Types of Applications
3-2- Categorization on the basis of platform
Desktop apps vs. web apps

DESKTOP APPS	**WEB APPS**
They require installation on the computer to run.	They are accessible through web browsers and do not require installation.
Generally, desktop apps do not require an internet connection to run.	Web apps cannot run without an internet connection.
They take space on the hard drive of the local computer.	They take up space on the remote server.
Deployment and updating are to be done individually on each computer.	Deployment and updating are done only on the server.
They have strict hardware requirements for proper functionality.	Web apps are hardware-independent and just require a web browser and internet connection to function.
As they are confined to a device and single or limited users, they are highly secure.	As web apps are accessible to all through the internet, they are less secure than desktop apps.
Generally, they are faster than web applications.	Generally, they are slower than desktop applications.

3- Types of Applications
3-2- Categorization on the basis of platform

- **3-2-3**- **Mobile applications:** Mobile applications, commonly known as mobile apps, have become an integral part of our daily lives. With the widespread use of smartphones and tablets, mobile apps have revolutionized the way we communicate, work, and entertain ourselves.

3- Types of Applications
3-2- Categorization on the basis of platform

- **3-2-4- Enterprise applications**
- Enterprise applications are software solutions designed to meet the complex needs of large organizations.
- These applications are specifically developed to support and streamline various business processes, enhance productivity, and improve overall efficiency within an enterprise.
- Unlike consumer applications, which are primarily focused on individual users, enterprise applications are built to cater to the needs of multiple users within an organization.
- Some common types of enterprise applications include: **JIRA and Confluence , Zoom**

3- Types of Applications
3-2- Categorization on the basis of platform

- **3-2-5- cloud applications:**
- Known as **Software as a Service (SaaS),** are web-based applications that are hosted on remote servers and accessed through the internet.
- These applications leverage the power of cloud computing to provide scalable, flexible, and cost-effective solutions to businesses and individuals.
- Offer numerous advantages over traditional applications: They eliminate the need for expensive hardware infrastructure and reduce the burden of software maintenance and updates.
- Users can access their data and applications from anywhere, using any device with an internet connection. This flexibility and accessibility have made cloud applications increasingly popular in today's digital landscape.
- Example of cloud applications are Google docs, Google Colab and Kaggle

3- Types of Applications
3-2- Categorization on the basis of platform

- **3-2-6- gaming applications:**
- Game development is a dynamic and exciting field that combines creativity, technology, and storytelling.
- Game development is made easier and more efficient with the use of specialized tools and game engines. These tools provide developers with pre-built systems and libraries for common game development tasks, such as physics simulation, artificial intelligence, and rendering.
- Some popular game engines include Unity and Unreal Engine
- These engines provide a wide range of features and support multiple platforms.

Layers of application

- Provides Interface to user

Presentation layer:

- controls the flow of execution and communication between presentation and data layer.

Business logic layer(Application Layer)

- contains data storage.
- The data layer is the part of a software application or system that stores and manages the data used by that application or system.

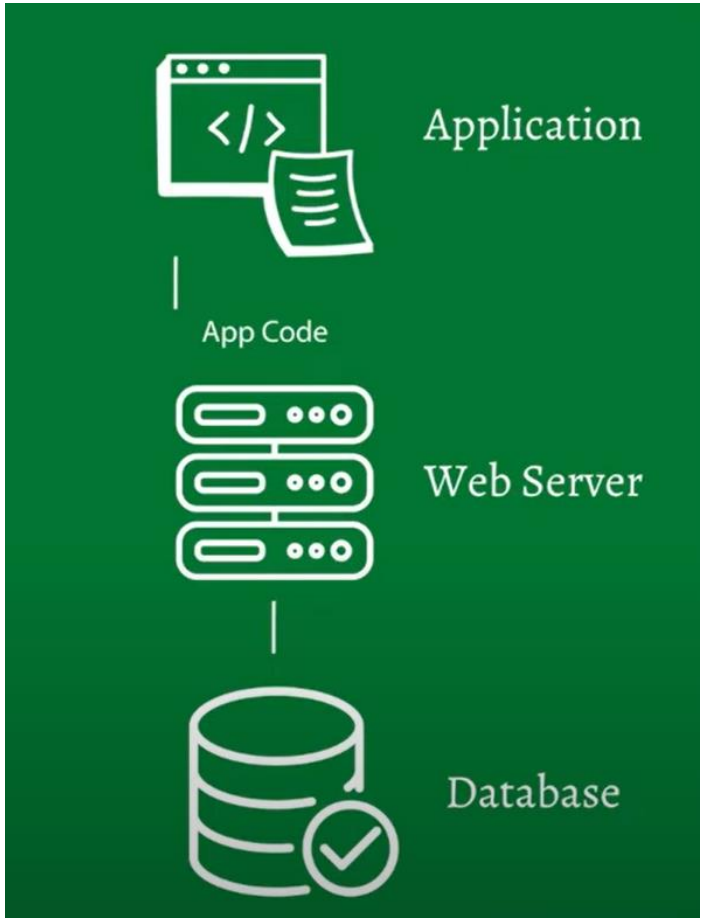
Data layer

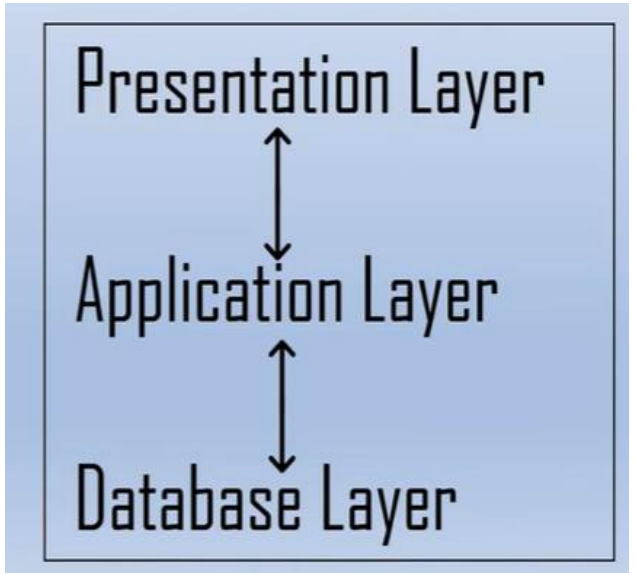
3- Types of Applications
3-3-3 based on type of architecture

- On the basis of all these layers, applications may contain one of the following architecture:
- Single Tier Architecture
- Two Tier Architecture
- Three Tier Architecture

3- Types of Applications
3-3-3 Based on Type of architecture

- **3-3-3-1- Single-tier architecture, also known as single-layer architecture, or client tier architecture** is an application architecture where all the components of an application are deployed on a single machine or server. In this architecture, the user interface, business logic, and data access layers are tightly coupled and run on the same platform.
- Single-tier architecture is typically used for small-scale applications or prototypes where simplicity and ease of development are prioritized over scalability and maintainability. It is not commonly used in production environments due to its limitations.





3- Types of Applications
3-3-3 Types of architecture

3-3-3-1- Single Tier Architecture

- Here is an example of a simple single-tier architecture code for a basic web application using HTML, CSS, and JavaScript
- other examples include media players

Example

single_tire_app_example.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Single Tier Architecture</title>
    <style>
        /* CSS styles */
    </style>
</head>
<body>
    <h1>Welcome to Single Tier Architecture!</h1>
    <div id="content">
        <!-- Application content -->
    </div>

    <script>
        // JavaScript code for application logic
    </script>
</body>
</html>
```

3- Types of Applications
3-3-3 based on type of architecture
3-3-3-1- Single Tier Architecture

pros

- **Simplicity**: One-tier architecture is the simplest form of architecture as it does not involve any distribution or separation of components. It is easy to develop and deploy applications in this architecture.
- **Performance**: Since the entire application runs on a single machine, there is no network overhead or latency, resulting in better performance and faster response times.
- **Data Integrity**: One-tier architecture allows for direct access to data without any intermediate layers, ensuring data integrity and consistency.

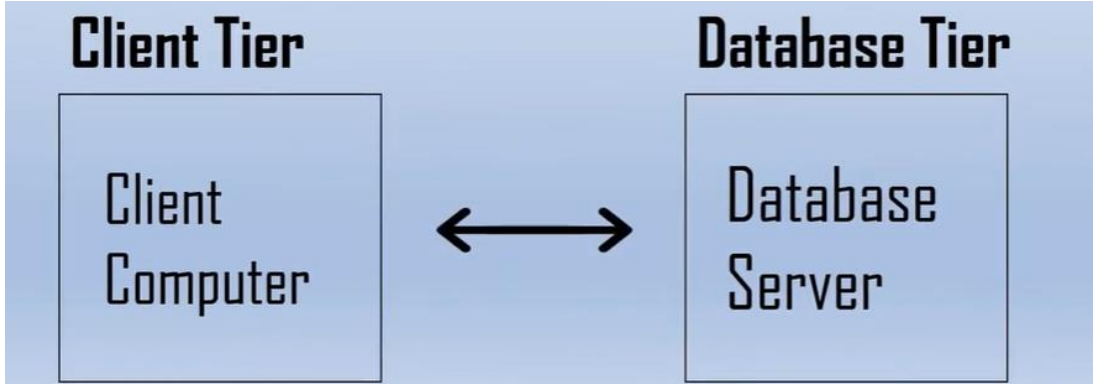
Cons

- **Limited Scalabilit**y: One-tier architecture lacks scalability as the entire application runs on a single machine. It becomes challenging to handle increased user loads or accommodate growing data volumes.
- **Maintenance Challenges**: With all components tightly coupled, making changes or updates to one part of the application may require modifications to the entire system. This can lead to maintenance challenges and increased effort.
- **Lack of Flexibility**: One-tier architecture does not provide flexibility in terms of using different technologies or platforms for different components. All parts of the application are tightly integrated, limiting the ability to adapt to changing requirements or leverage specialized technologies.

3- Types of Applications
3-3-3 Types of architecture

- **3-3-3-2- two Tier Architecture (client-server architecture)** : is a software design pattern that separates an application into two distinct tiers or layers: the client tier and the server tier. Each tier has its own specific responsibilities and functions within the overall system.

1. **Client Tier**: This tier represents the user interface or the front-end of the application. It is responsible for presenting the data to the user and capturing user input. The client tier interacts directly with the end-user and provides a user-friendly interface for accessing and interacting with the application.
2. **Server Tier**: This tier is responsible for processing the business logic, data storage, and managing the overall functionality of the application. It handles the core processing and data manipulation operations. The server tier receives requests from the client tier, processes them, and sends back the response.



3- Types of Applications
3-3-3 Types of architecture
3-3-3-2- Two tier architecture

Pros

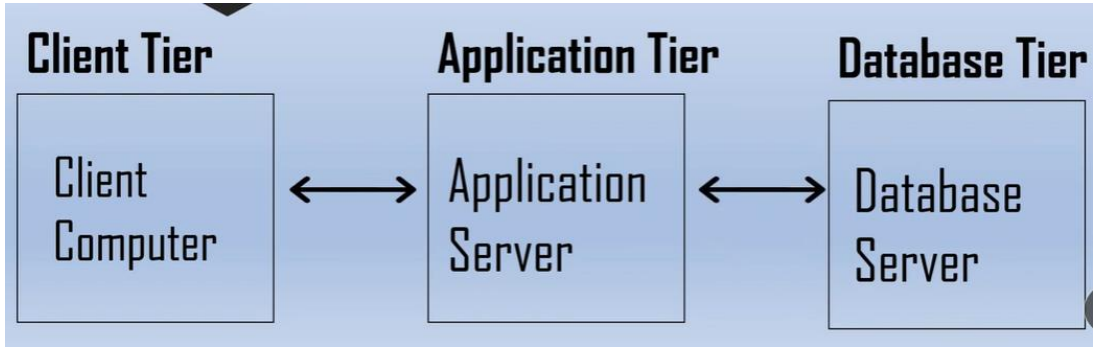
- **Simplicity**: Two-tier architecture is relatively simple compared to more complex architectures like three-tier or It has a straightforward structure with a clear separation between the client and server components.
- **Efficiency**: The direct communication between the client and server tiers enables faster processing and response times as there are no intermediate layers involved.
- **Data Integrity**: The server tier is responsible for data storage and management, ensuring data integrity and security. This centralization helps in maintaining consistency and reliability in data

Cons

- **limited scalability:** Two-tier architecture may have limitations in terms of scalability because both the client and server components are tightly coupled. Scaling the system may require scaling both tiers simultaneously.
- potential security risks if not properly implemented.
- Hence, more complex architectures like three-tier or n-tier may be preferred for larger and more scalable applications.

3- Types of Applications
3-3-3 Types of architecture

- **3-3-3-3- Three Tier Architecture: also known as multi-tier architecture**, is a software design pattern that divides an application into three distinct tiers or layers: the presentation tier, application tier, and data tier. Each tier has its own specific responsibilities and functions within the overall system.



3- Types of Applications
3-3-3 Types of architecture
3-3-3-3- Three Tier Architecture

- Presentation Tier (Client Tier): This tier represents the user interface or the front-end of the application. It is responsible for presenting the application's user interface to the end-user and capturing user input. The presentation tier handles user interactions, displays data, and provides a user-friendly interface.
- Application Tier (Middle Tier): This tier contains the application logic and business rules. It processes and manages the application's functionality, including data validation, business workflows, and other processing operations. The application tier acts as an intermediary between the presentation tier and the data tier.
- Data Tier (Backend Tier): This tier is responsible for data storage and retrieval. It manages the persistence and storage of data, typically in a database or other data storage systems. The data tier handles data access, manipulation, and ensures data integrity and security.

3- Types of Applications
3-3-3 Types of architecture
3-3-3-3- Three Tier Architecture

Pros

- **Modularity**: The separation of concerns into distinct tiers allows for better modularity and maintainability. Each tier can be developed, updated, or scaled independently, as long as the interfaces between the tiers remain consistent.
- **Scalability**: Three-tier architecture provides better scalability compared to two-tier architecture. Each tier can be scaled independently, allowing for efficient resource utilization and handling increased user loads.
- **Flexibility**: The separation of the application logic from the presentation and data layers allows for flexibility in choosing different technologies or platforms for each tier. This flexibility enables the use of different programming languages or frameworks that are best suited for each layer's requirements.

Cons

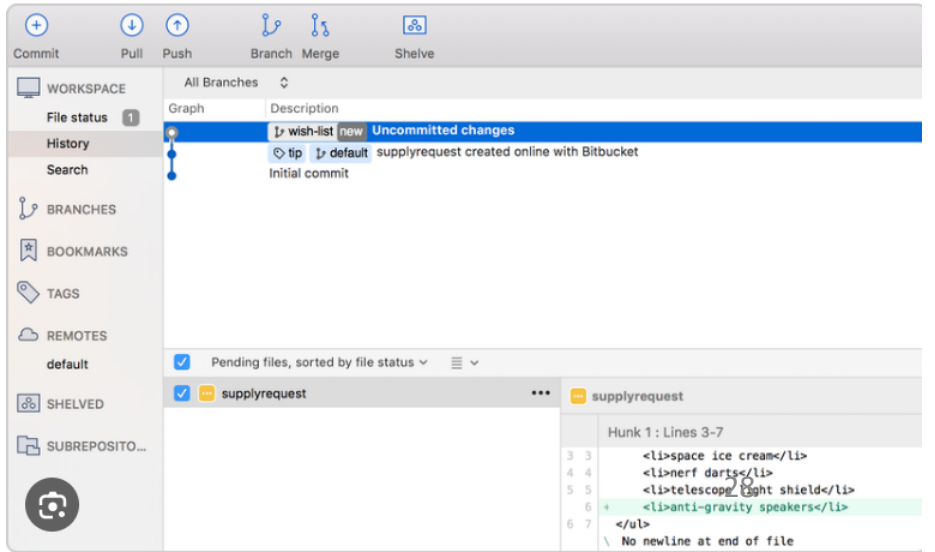
- May introduce additional complexity and overhead in terms of communication and coordination between the tiers.

4 - Tools and Technologies for Application Development

- **4-1- Integrated Development Environments (IDEs)**
- Integrated Development Environments, or IDEs, are software applications that provide a comprehensive set of tools and features to facilitate application development.
- IDEs typically include a code editor, a debugger, a compiler, and other tools that help developers write, test, and debug their code. Some popular IDEs used in application development include: **Eclipse, VS code, PyCharm , Spyder**

4 - Tools and Technologies for Application Development

- **4-2- Version Control Systems**
- Version control systems are essential tools for managing changes to source code and collaborating with other developers. These systems allow developers to track modifications, revert to previous versions, and merge changes made by multiple developers. Some popular version control systems used in application development include: **Git, SourceTree**



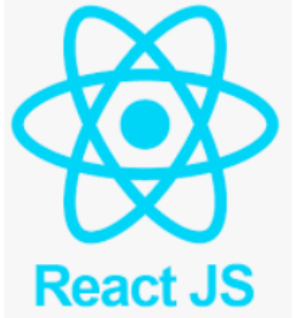
4 - Tools and Technologies for Application Development

- **4-3- Testing Frameworks:**
- Testing is a critical aspect of application development to ensure that the application functions as intended and is free from bugs and errors.
- Testing frameworks provide a structured approach to writing and executing tests, making it easier for developers to identify and fix issues.
- Some commonly used testing frameworks include: **Junit and Selenium**

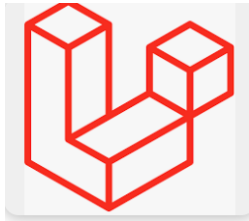
4 - Tools and Technologies for Application Development

- **4-4- Integrated Development and Continuous Integration (CI) Tools:**
- Integrated development and continuous integration tools help automate various aspects of the development process, such as building, testing, and deploying applications.
- These tools enable developers to streamline their workflow, improve code quality, and ensure that changes are integrated smoothly.
- Some commonly used integrated development and CI tools include: **Jenkins and Buildbot**

4 - Tools and Technologies for Application Development



- **4-5- Frameworks and Libraries:**
- Frameworks and libraries provide developers with pre-built components and functionalities that can be used to accelerate the development process.
- These provide a structured approach to building applications.
- Some widely used frameworks and libraries in application development include: **React** which is a JavaScript Library for building user interface **, Laravel** which is a PHP web application framework and **Django** which is a web application python framework



End of lecture_1

Thanks for listening

ASPU University
Faculty of Informatics Engineering
Fourth Year
Application Development course
2023 - 2024

جامعة الشام الخاصة
Al-Sham Private University

Requirements Gathering , Architectural and Object Oriented Design with examples

SDLC – Software Development Life Cycle – part1

Lecture 2

Lecture Overview

- Stages of App development SDLC
  1. Requirements Gathering
  2. Design
     a. Architectural Design
        - Use case Diagram
     b. Object Oriented Design
        - Class diagram
     c. real life examples : Banking system, order system

1 - The Stages of App Development/ SDLC Software Development Life Cycle

1 - The Stages of App Development/ SDLC Software Development Life Cycle

- Application development typically follows seven stages: planning, requirements gathering, design, development, testing, deployment and maintenance. Each of these stages plays an important role in creating a successful app.
- **Planning:** The first step is to create a plan for your app project. This should include defining the app's purpose and goals and creating timelines and budgets that will guide you through the process.
- **Requirements Gathering:** The next step is to gather the app's requirements. This includes identifying user needs and app features and creating technical specifications that will guide development.
- **Design:** Once you've gathered the app's requirements, you can start designing your app.

1- The Stages of App Development/ SDLC
Software Development Life Cycle .. Cont.

- **Development:** This includes coding the app, setting up databases and integrating APIs
- **Testing:** This includes running tests to make sure the app works as expected and meets all requirements, debugging errors, and testing app performance in various scenarios.
- **Deployment:** This involves setting up hosting environments, creating app store accounts, and submitting the app for review.
- **Maintenance:** Once the app is deployed, it's important to maintain it over time. This includes monitoring app performance, releasing updates, responding to user feedback, and adding new features as needed.

1-1-Gathering & Analysis of Requirement:

- This stage involves answering these 4 questions
  - Who is supposed to use this software?
  - How will the software be used upon completion?
  - What type of data should be added to the software?
  - What should be the data output by this software?
- Lastly, a document for requirement specification is prepared which serves as a guideline for the next level of the software development process.

1-1-Gathering & Analysis of Requirement
1-1-1- Functional Requirements vs. non functional requirements

- Requirements are generally split into two types: functional and non functional

Functional Requirements	Non Functional Requirements
A functional requirement defines a system or its component.	A non-functional requirement defines the quality attribute of a software system.
It specifies “What should the software system do?”	It places constraints on “How should the software system fulfill the functional requirements?”
Functional requirement is specified by User.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a system as a whole.
Helps you verify the functionality of the software.	Helps you to verify the performance of the software.
Functional Testing like System, Integration, End to End, API testing	Non-Functional Testing like Performance, Stress, Usability, Security testing
Usually easy to define.	Usually more difficult to define.

1-1-Gathering & Analysis of Requirement
1-1-1-1- Functional Requirements vs. non functional requirements / A real life example YouTube

YouTube

Functional requirements

1. Users should be able to create an account and log in.
2. Users should be able to upload videos.
3. Users should be able to view and search for videos.
4. Users should be able to like, comment on, and share videos.
5. Users should be able to subscribe to channels and receive updates on new videos.
6. Users should be able to create and manage playlists.

Non functional requirements

- The system should have a fast and responsive user interface to provide a seamless browsing experience.
- The system should be able to handle a large number of concurrent users without any performance degradation.
- The system should have a reliable and scalable infrastructure to handle high traffic and storage requirements.
- The system should have a robust video streaming capability to ensure smooth playback across different devices and network conditions.
- The system should have a secure authentication and authorization mechanism to protect user accounts and data.

1-2- Design

- The design step in the software life cycle is a crucial phase where the overall structure and architecture of the software system are planned and defined.
- It involves transforming the requirements gathered during the analysis phase into a detailed design that serves as a blueprint for the development team.
- During the design step, several key activities take place:

1-2- Design
1-2-1-Architectural Design

- This involves identifying the high-level structure of the software system, including the components, modules, and their interactions.
- It focuses on defining the overall system architecture and ensuring that it aligns with the requirements.
- For architectural design it is helpful to create a **use case diagram** which helps to identify the system's actors, their interactions, and the high-level functionality of the system.

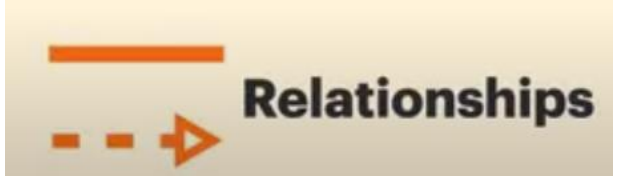
1-2- Design

1-2-1-1- Use Case Diagram

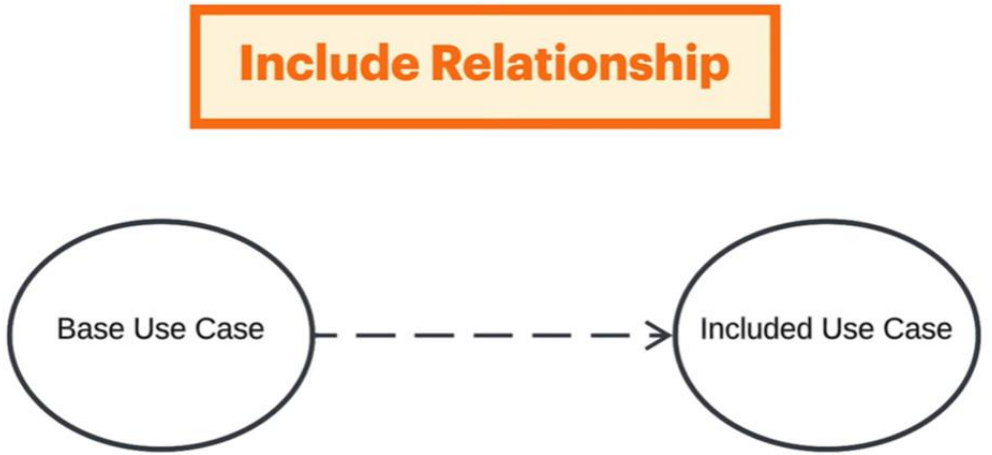
- In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.
- It is used for Modeling the basic flow of events in a use case
- **Use case diagram components:**

1. **Actors**: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
2. **System**: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
3. **Use cases**: Use cases represent the specific tasks or actions that the system performs to achieve a particular goal. They describe the interactions between actors and the system. Use cases are depicted as ovals and are connected to actors through associations. Each use case has a name that describes the action it represents.
4. **Relationships/Associations**: Associations are lines that connect actors and use cases in a use case diagram. They represent the relationship between an actor and a use case. Associations can be labeled to indicate the nature of the relationship, such as "performs," "uses," or "initiates."

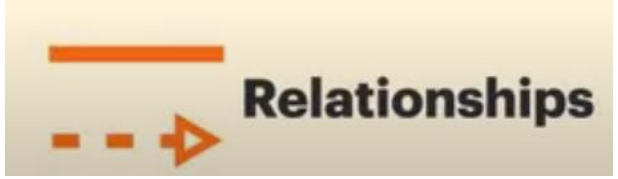
1-2- Design
1-2-1-1- Use Case Diagram



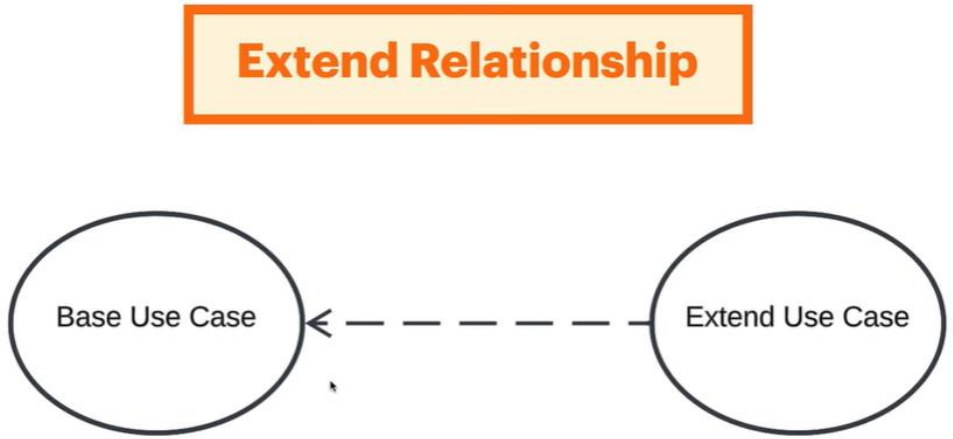
- **Types of relationships:**
- **Include Relationship**:
- is used to show that one use case includes another use case.
- It indicates that the behavior of the included use case is always performed when the base use case is executed.
- The included use case is a separate and independent use case that is called upon by the base use case.
- This relationship is represented by a dashed arrow with an open arrowhead pointing to the included use case.



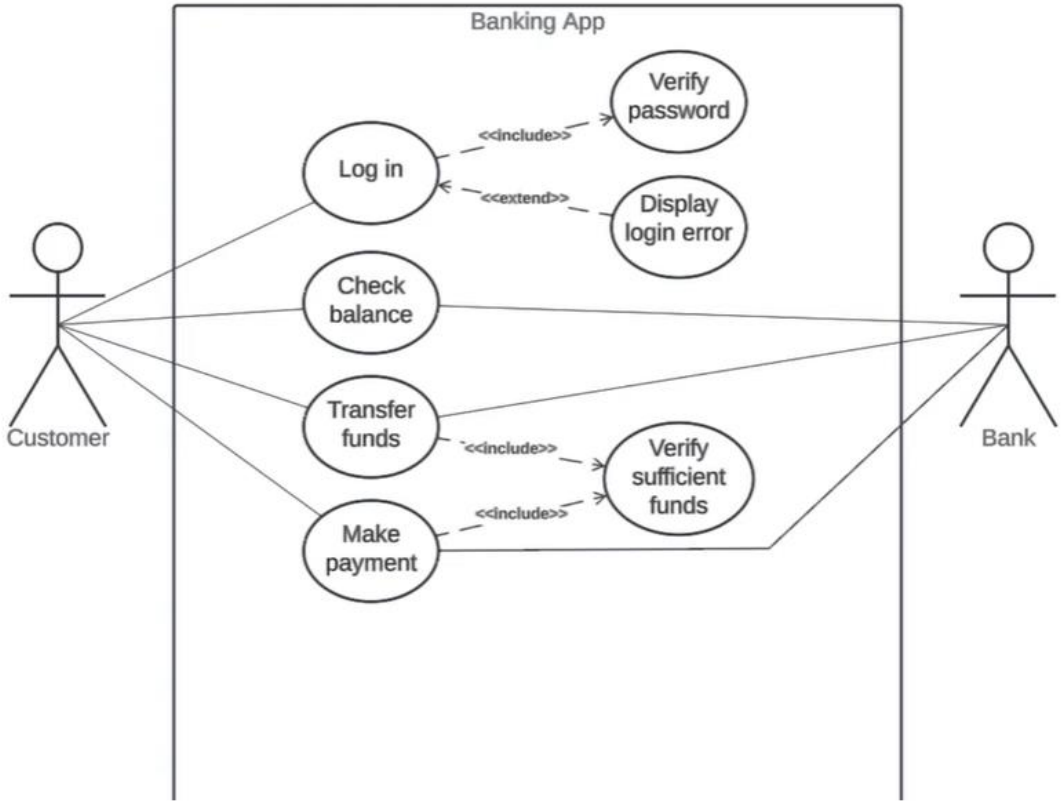
1-2- Design
1-2-1-1- Use Case Diagram



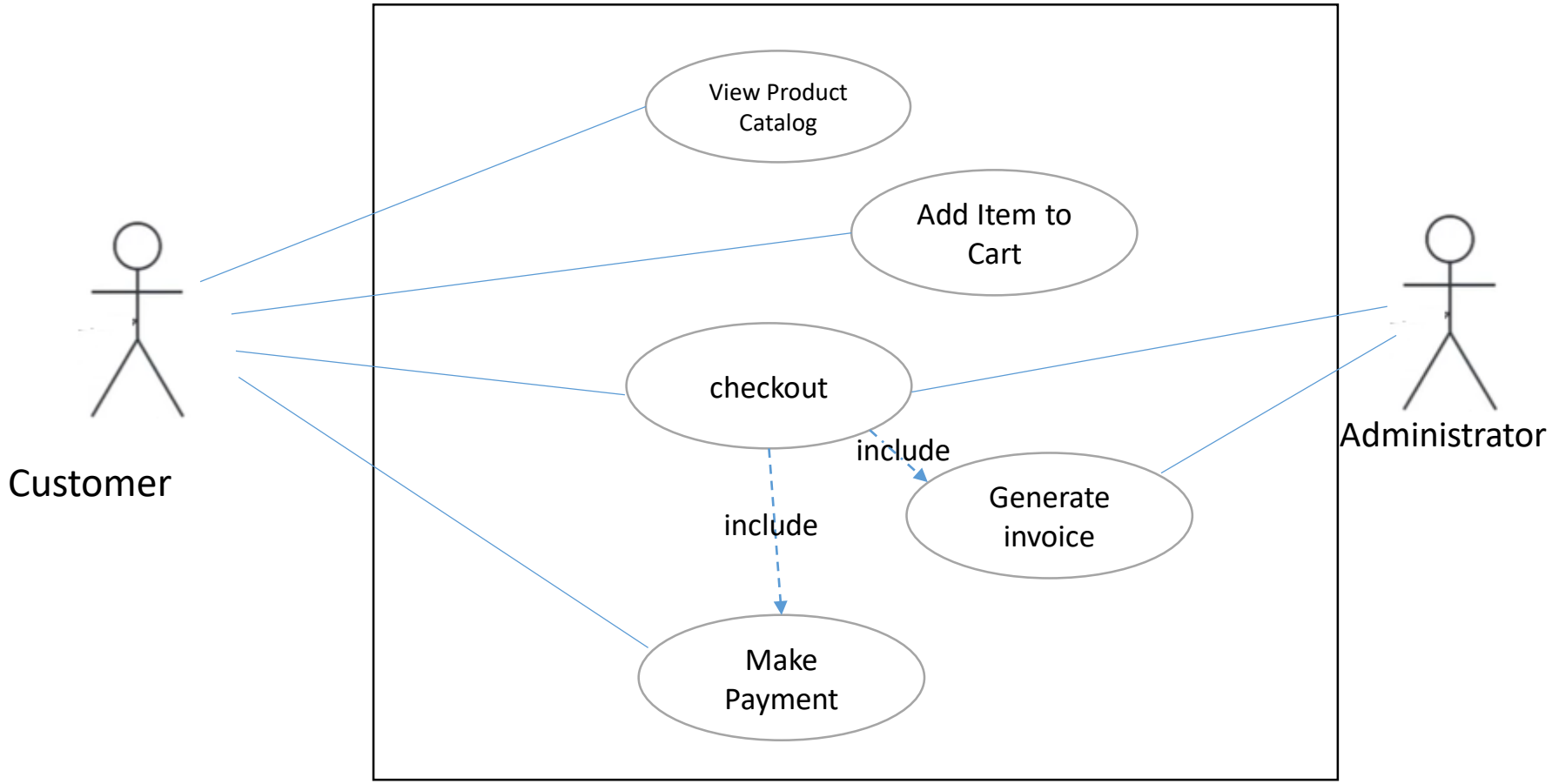
- **Types of relationships:**
- **Extend Relationship**: used to show that one use case can be extended by another use case.
- It indicates that the behavior of the extended use case may be optionally performed when certain conditions are met in the base use case.
- The extended use case adds additional functionality to the base use case.
- This relationship is represented by a dashed arrow with an open arrowhead pointing from the base use case to the extended use case.



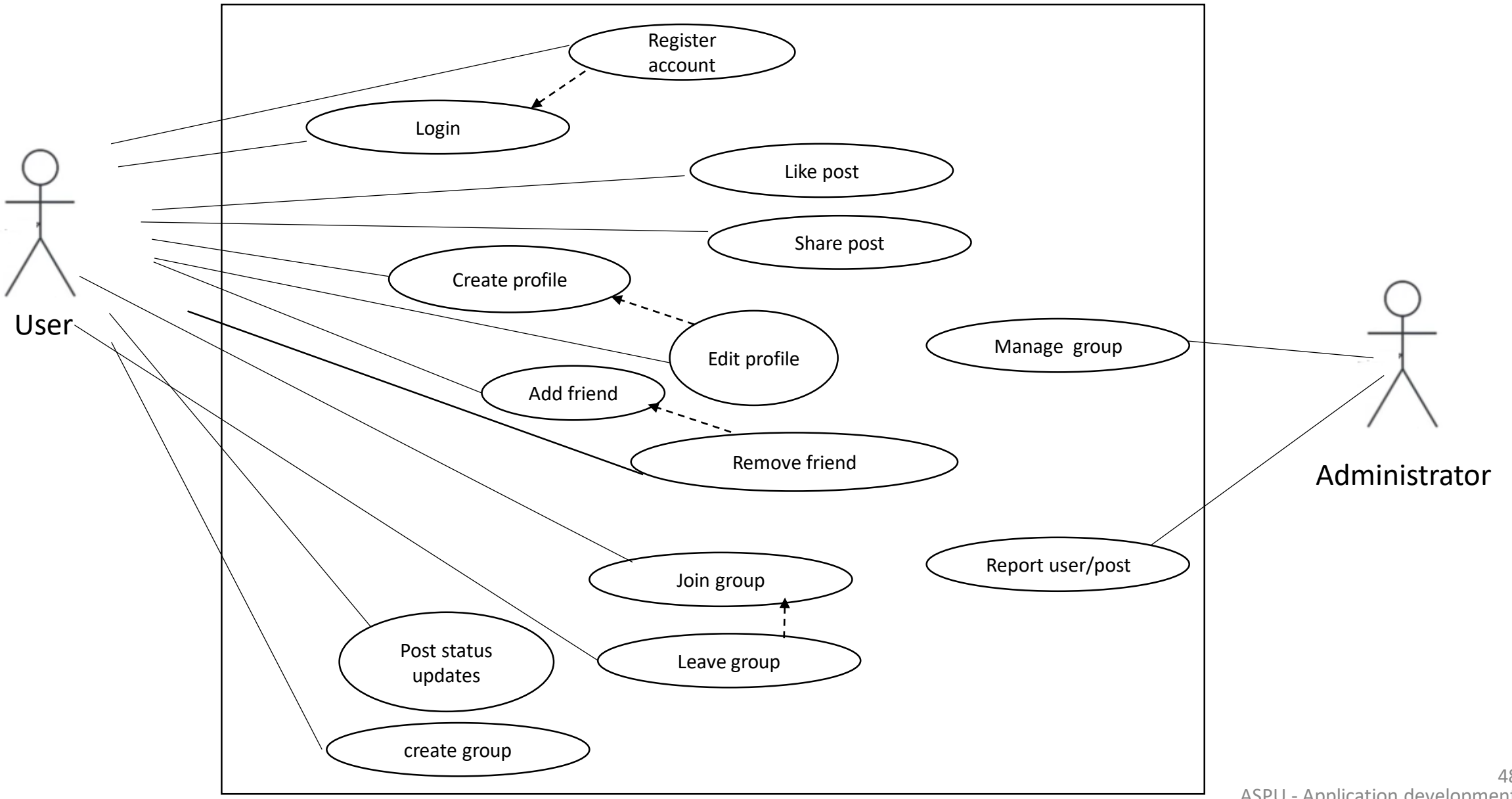
1-2- Design
1-2-1-Architectural Design/ Use Case Diagram / exercise 1 / example Banking system



1-2- Design
1-2-1-Architectural Design/ Use Case Diagram / exercise 2 / example simple online shopping system



1-2- Design
1-2-1-Architectural Design/ Use Case Diagram / exercise 3 / social network platform system



1-2- Design
1-2-2-Detailed Design (Object Oriented Design)

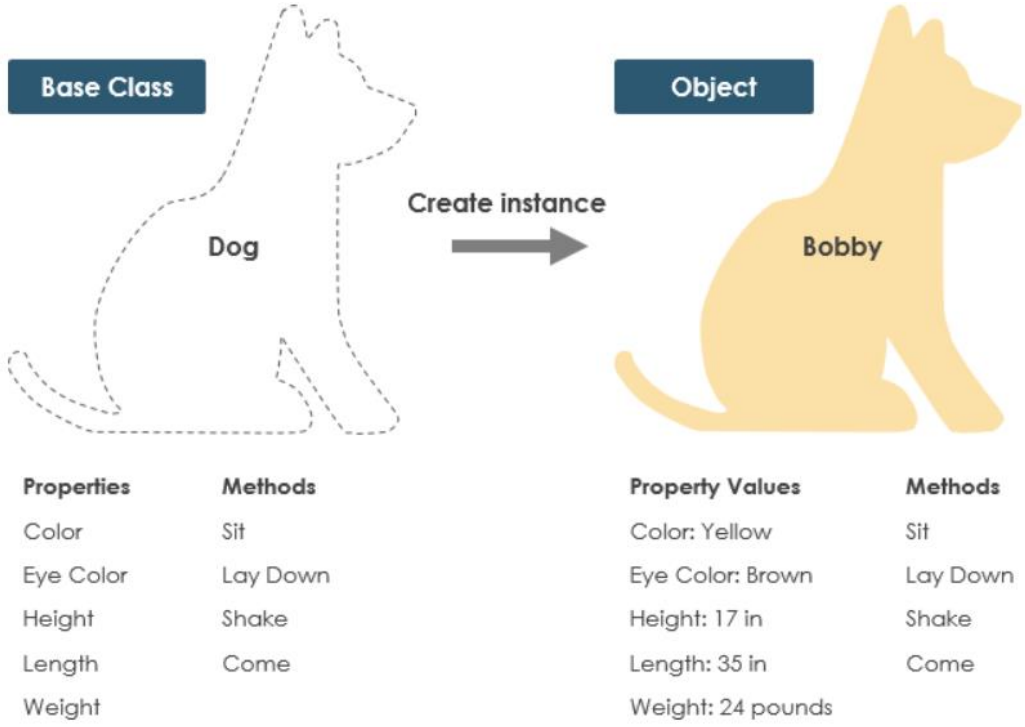
- Once the architecture is defined, the detailed design phase begins. This involves breaking down the system into smaller components and modules, specifying their interfaces, and defining their internal structure.
- It includes designing algorithms and data structures.

1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

- The UML Class diagram is a graphical notation used to construct and visualize object oriented systems.
- A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's:
- classes
- Attributes,
- Operations (or methods)
- Relationships among objects.

1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

- A Class is a blueprint for an object
- Classes describe the type of objects, while objects are usable instances of classes.
- Each Object was built from the same set of blueprints and therefore contains the same components (properties and methods).



1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

- **A class** represent a concept which encapsulates state (attributes) and behavior (operations).
- Each attribute has a type.
- Each operation has a signature.
- The class name is the only mandatory information.

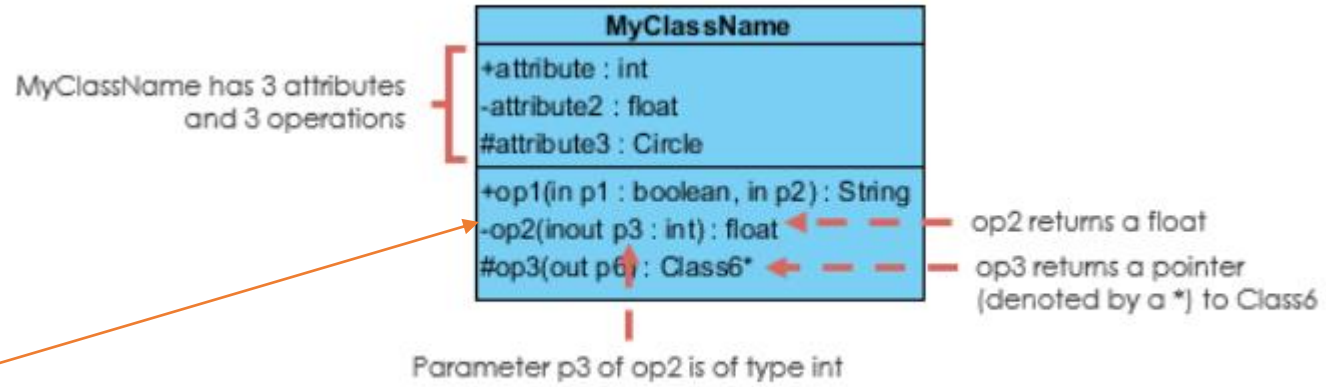
The name of the class appears in the first partition.

- **Attributes** are shown in the second partition.
- Attributes map onto member variables (data members) in code

1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

Class Operations (Methods):

- Operations are shown in the third partition.
- They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters are shown after the colon following the parameter name.
- Operations map onto class methods in code



Class Visibility

The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

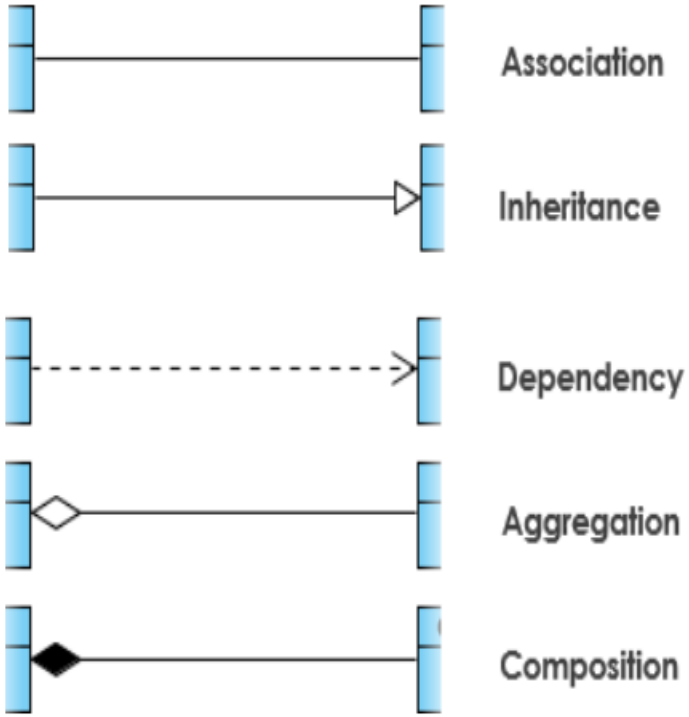
+ denotes public attributes or operations

- denotes private attributes or operations

# denotes protected attributes or operations

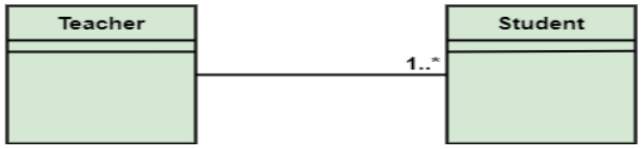
1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

Relationships between classes

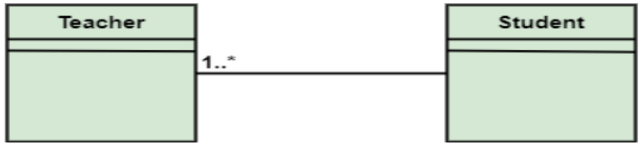


- **Association**
- A structural link between two peer classes.
- It depicts a binary relationship between the objects representing an activity.
- It is represented by a solid line between classes
- We can specify the multiplicity of an association by adding it on the line that will denote the association.
- A bilateral association is represented by a straight line connecting two classes. It simply demonstrates that the classes are aware of their relationship with each other.

1) A single teacher has multiple students.

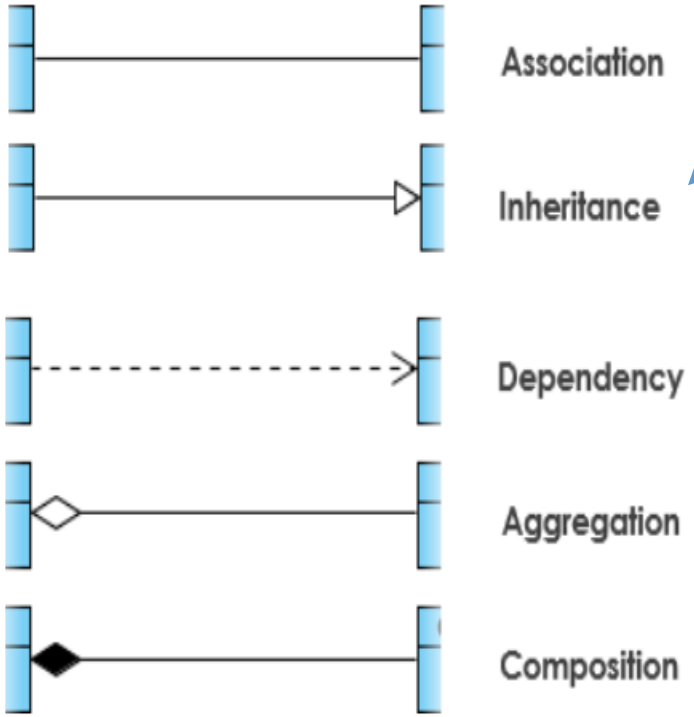


2) A single student can associate with many teachers.

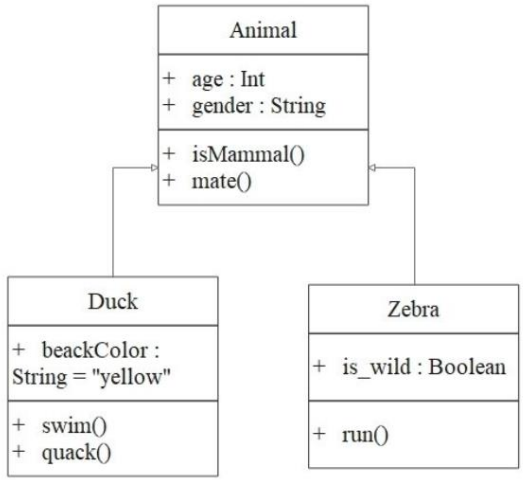


1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

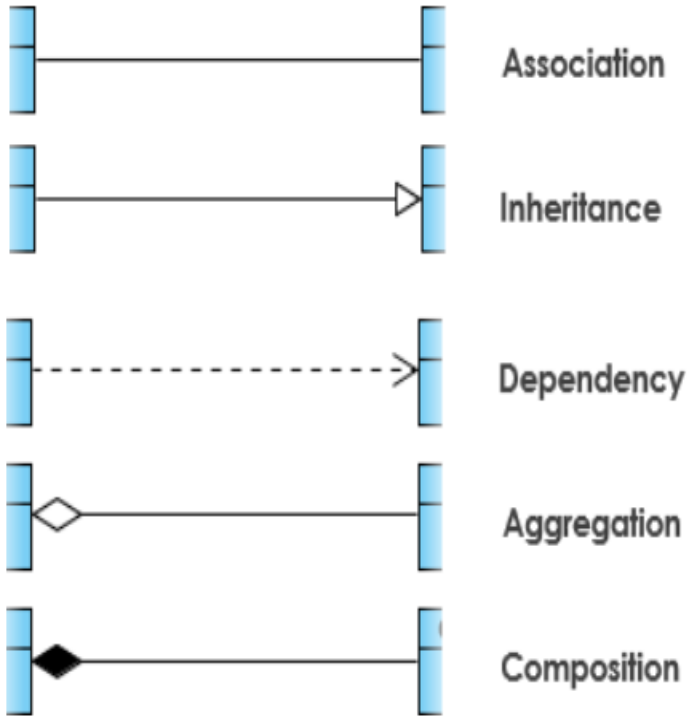
Relationships between classes



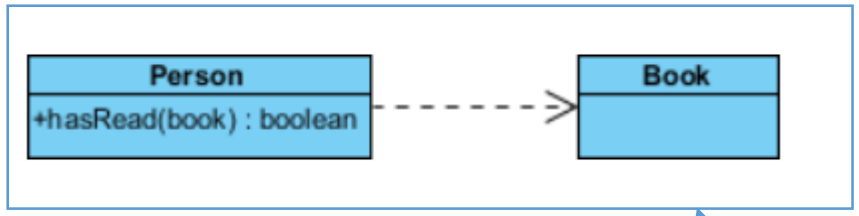
- **Inheritance**
- specific classifier inherits the features of the more general classifier.
- SubClass1 and SubClass2 are derived from SuperClass.
- The relationship is displayed as a solid line with a hollow arrowhead that points from the child element to the parent element.



1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

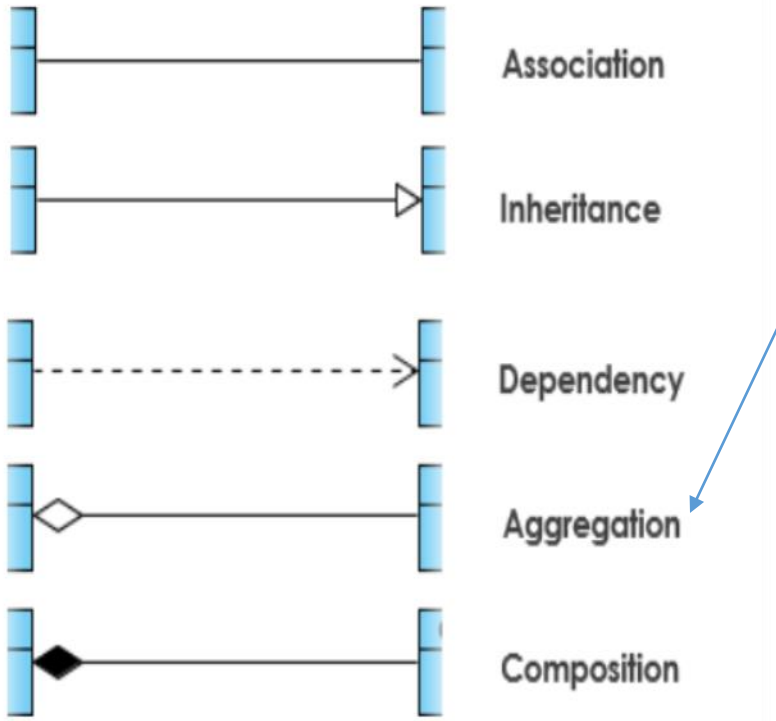


- **Dependency**
- A special type of association.
- Exists between two classes if changes to the definition of one may cause changes to the other (but not the other way around).
- Class1 depends on Class2

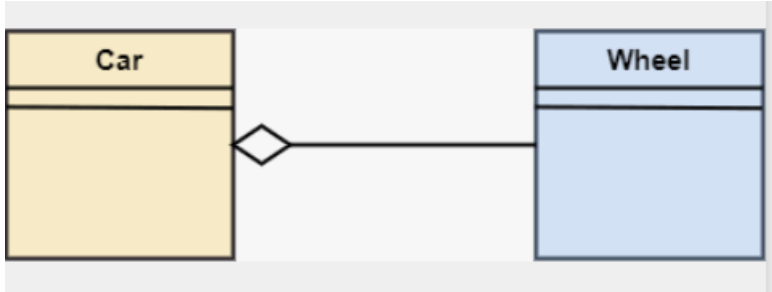


Example: The Person class might have a hasRead() method with a Book parameter that returns true if the person has read the book (perhaps by checking some database).

1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

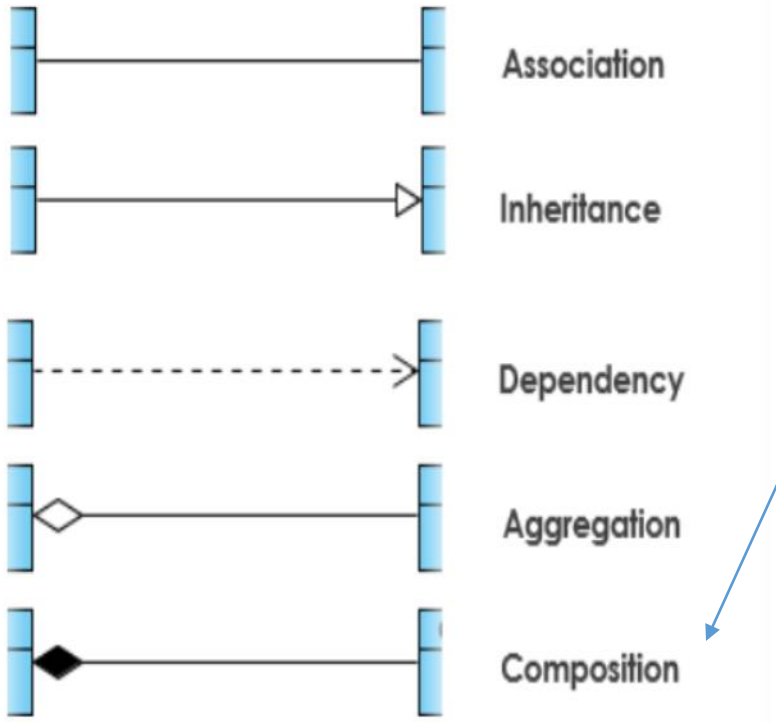


- **Aggregation**
- A special type of association.
- It represents a "part of" relationship. Class2 is part of Class1.
- The relationship is displayed as a solid line with a unfilled diamond at the association end, which is connected to the class that represents the aggregate
- It is a kind of relationship in which the child is independent of its parent.



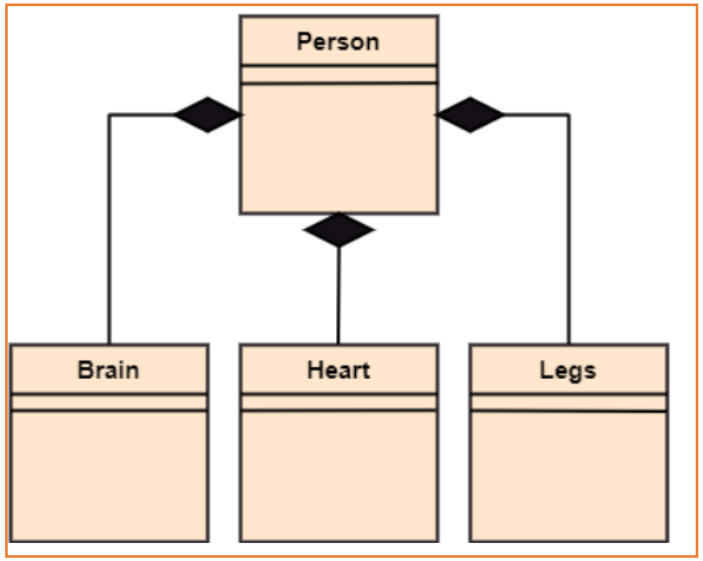
A car can have multiple wheels, and the wheels can exist independently even if the car is not present.

1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram

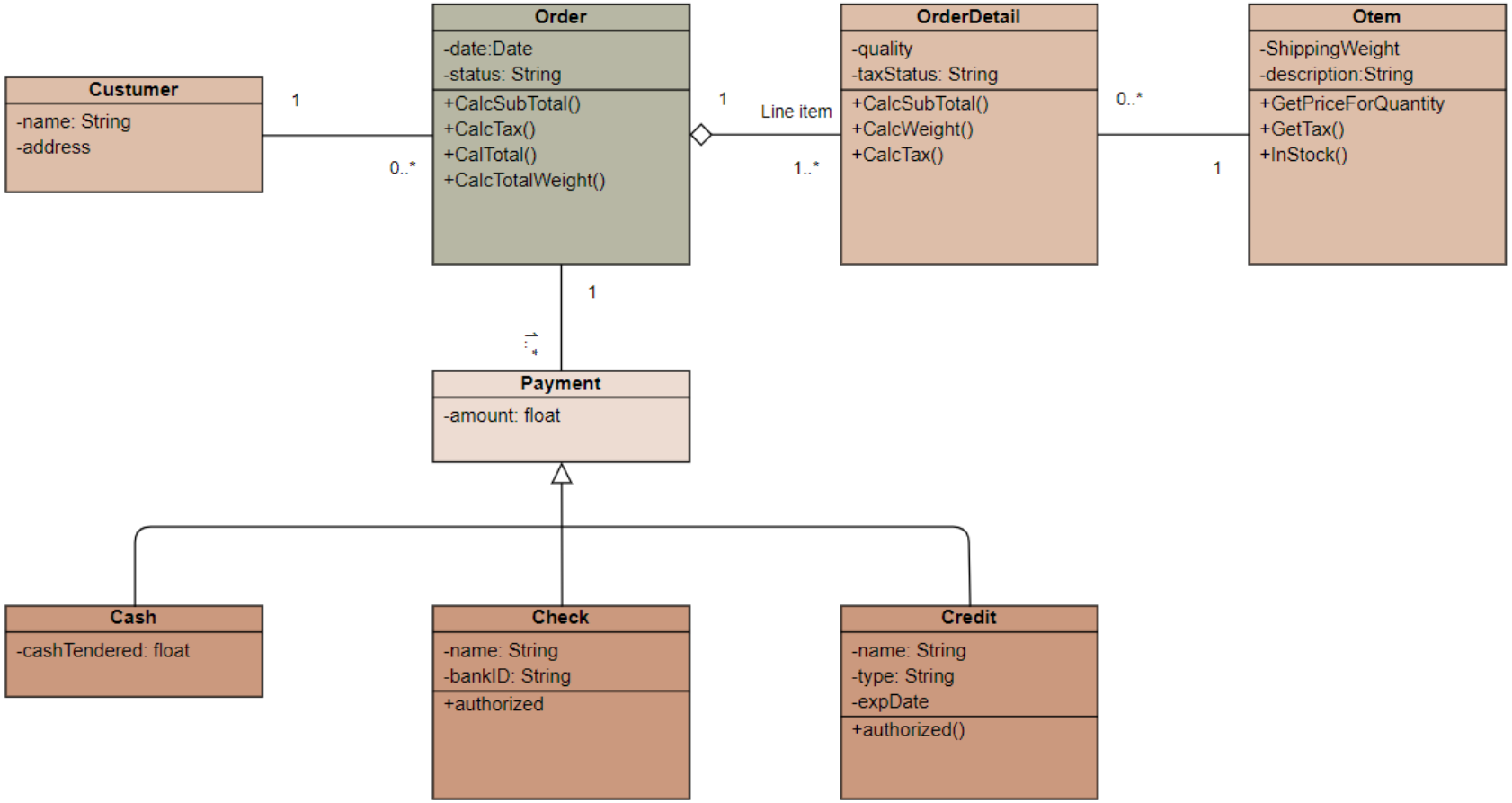


- **Composition**
- A special type of aggregation where parts are destroyed when the whole is destroyed.
- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- The relationship is displayed as a solid line with a filled diamond at the association end, which is connected to the class that represents the whole or composite.

example, the composition association relationship connects the Person class with Brain class, Heart class, and Legs class. If the person died, the brain, heart, and legs will also die.



1-2- Design
1-2-2-1-Object Oriented Design – UML class diagram – example / order and pay system



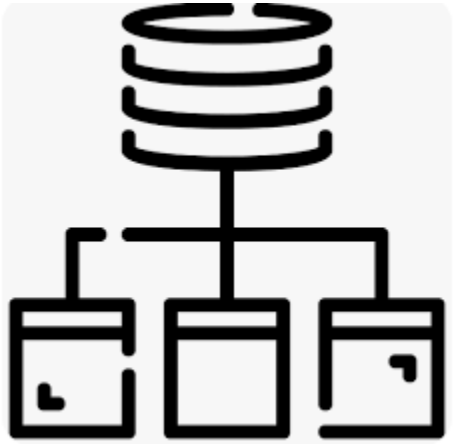
End of lecture 2

ASPU University
Faculty of Informatics Engineering
Fourth Year
Application Development course
2023 - 2024

جامعة الشام الخاصة
Al-Sham Private University

Data Base Design / Data Models and Data Modelling

Lecture_3



Lecture Overview

- Database design
  - Data Modelling and Data Models
    I. hierarchical data model
    II. networked data model
    III. Relational data model
    IV. Object oriented data model
    V. Entity Relationship model
  - Types of relational data model
    I. Conceptual data model
    II. Logical Data model
    III. Physical data model

Database Design

- Database is a collection of information that is organized so data can be easily stored, managed, updated and retrieved
- If the software system requires a database, this step involves designing the structure, tables, relationships, and constraints of the database. It ensures that the data is organized efficiently and can be accessed and manipulated effectively.

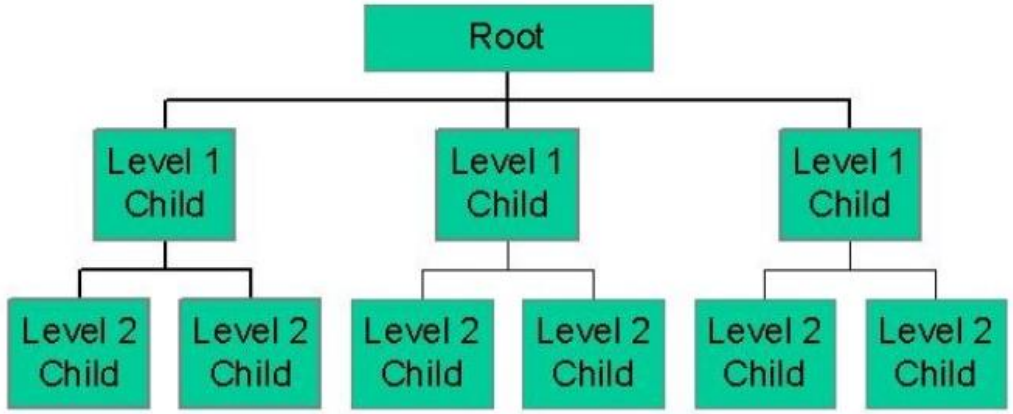
Types of database models

This categorization focuses on the specific data model used to represent and organize data within a database system.

- Hierarchical database model
- Network model
- Relational model
- Object-oriented database model

Hierarchical database model

- In this model, data is organized in a tree-like structure with parent-child relationships.
- It allows one-to-many relationships between entities.
- Data access is typically top-down, starting from the root and moving downwards.
- Each node has exactly one parent.
- One parent may have many child.
- Thus Hierarchical Database System can have one-to-one and one-to-many relations, but not many-to-many
- It is efficient for simple and well-defined data structures but can become complex for more complex relationships.



Examples
- A website sitemap is example of a hierarchy.
- the folder structure in your machine can also be represented hierarchically.
- XML data storage

Hierarchical database model

- In a Hierarchical Database System, the data is stored as records.
- each record has a parent ID that lends it to the tree structure.
- The first record is known as the root record.

Example : a database containing 3 tables: Company, Departments, and Employees.

- each record in each table has a parent ID that links to a record in its parent table.
- The Company table has only one record and it is the root table.
- The Departments table has departments belonging to the company,
- and the employees table has employees belonging to each department.
- each record (except the root node) has only one parent.

Company table(root)

ID	Company Name
1	ABC Ltd.

Department table(level1 child)

ID	Department Name	Department Description	Parent ID
10	R&D	Innovate new products	1
11	Finance	Manage Working Capital	1
12	HR	Define and maintain org. values	1
13	Sales and Marketing	Create the brand and sell the products	1

Employee table (level2 child)

ID	Employee Name	Age	Parent ID
100	Yash	26	10
101	Isha	26	10
102	Siddhi	24	13
103	Hemal	24	11
104	Ayush	28	12

Advantages of Hierarchical database model

- Simplicity is the greatest advantage of Hierarchical Databases.
- Since each record has only one parent, traversing through the database is very fast.
- integrity: a record in a child table must be linked to a record in the parent table. Consequently, if a record is deleted in the parent table, corresponding child records also get deleted.
- Several programming languages provide support for dealing with tree-type data structures, making dealing with Hierarchical Databases example very simple. Examples:
  - **Python** provides various libraries and modules that make dealing with tree-type data structures easy. For example, the **xml.etree.ElementTree module** allows parsing and manipulating XML data, which can be represented as a tree structure.
  - **Java**: provides built-in classes and interfaces to work with tree-type data structures. The **java.util.TreeMap** and **java.util.TreeSet** classes are examples of tree-based data structures that can be used to store and manipulate hierarchical data.

Disadvantages of Hierarchical Database Systems

1. **The rigidity of this database** is a big disadvantage. Since a child can have only one parent, often redundant records have to be introduced to properly explain some relationships.

Company table(root)

ID	Company Name
1	ABC Ltd.

Department table(level1 child)

ID	Department Name	Department Description	Parent ID
10	R&D	Innovate new products	1
11	Finance	Manage Working Capital	1
12	HR	Define and maintain org. values	1
13	Sales and Marketing	Create the brand and sell the products	1

Employee table (level2 child)

ID	Employee Name	Age	Parent ID
100	Yash	26	10
101	Isha	26	10
102	Siddhi	24	13
103	Hemal	24	11
104	Ayush	28	12

2**. Hard to expand** if a child has no parent assigned, then it becomes difficult to add the child record to this database. A temporary dummy parent record would have to be created

Example of difficulty to exapand consider a fresher joining the company and going through the training. His/her department would be allotted only after training. In such a case, it is difficult to add the record of that employee to the hierarchical database unless a dummy department like ‘Training’ is created.

Example of rigidity if Siddhi was also taking part in some work in the Finance department, a new entry for her would’ve had to be created, with the Finance department being the parent. Thus, there would be two Siddhis with different IDs in the database, although they represent the same person.

Disadvantages of Hierarchical Database Systems

3. **difficult to Reorganize data** because the parent-child relationships should not be disturbed

Company table(root)

ID	Company Name	Parent ID
1	ABC Ltd.	NA

Department table(level1 child)

ID	Department Name	Department Description	Parent ID
10	R&D	Innovate new products	1
11	Finance	Manage Working Capital	1
12	HR	Define and maintain org. values	1
13	Sales and Marketing	Create the brand and sell the products	1

Employee table (level2 child)

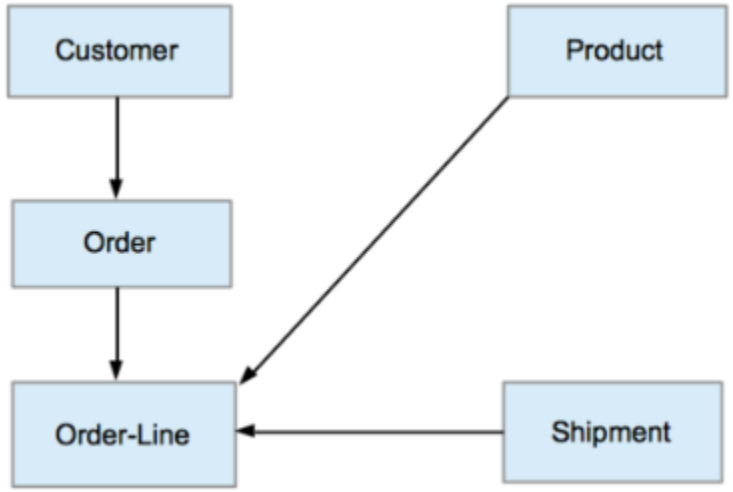
ID	Employee Name	Age	Parent ID
100	Yash	26	10
101	Isha	26	10
102	Siddhi	24	13
103	Hemal	24	11
104	Ayush	28	12

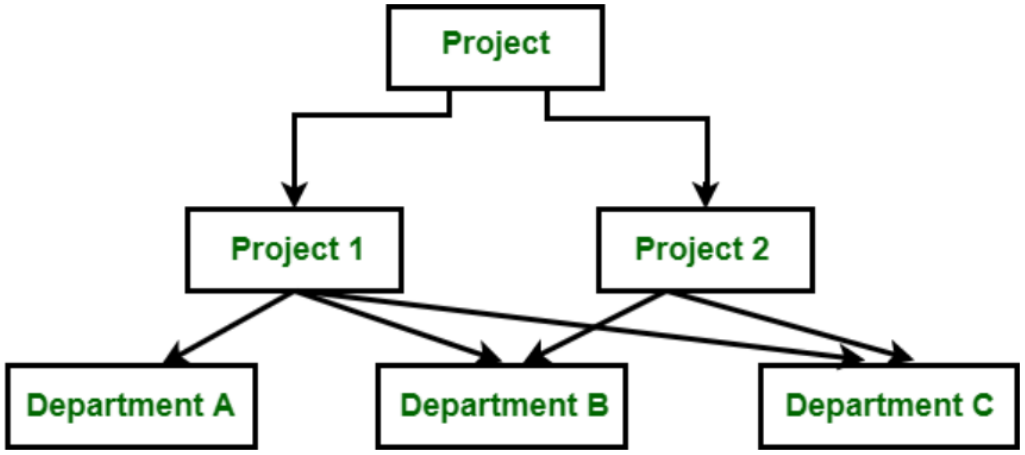
Example of difficulty to reorganize If someone wants to introduce a new relationship in the future, say ‘Category’ between ‘Departments’ and ‘Employees’, then reorganizing the entire database would be hard. Even if you do add the new relationship, all programs and applications accessing this database need to be modified to incorporate this new relationship.

Network model:

- In this model, data is organized in a network-like structure with interconnected records.
- It allows many-to-many relationships between entities.
- It uses pointers or links to establish relationships between records.
- It is more flexible than the hierarchical model but can be complex to implement and maintain.

Simple example





Network model – pros and cons

Pros

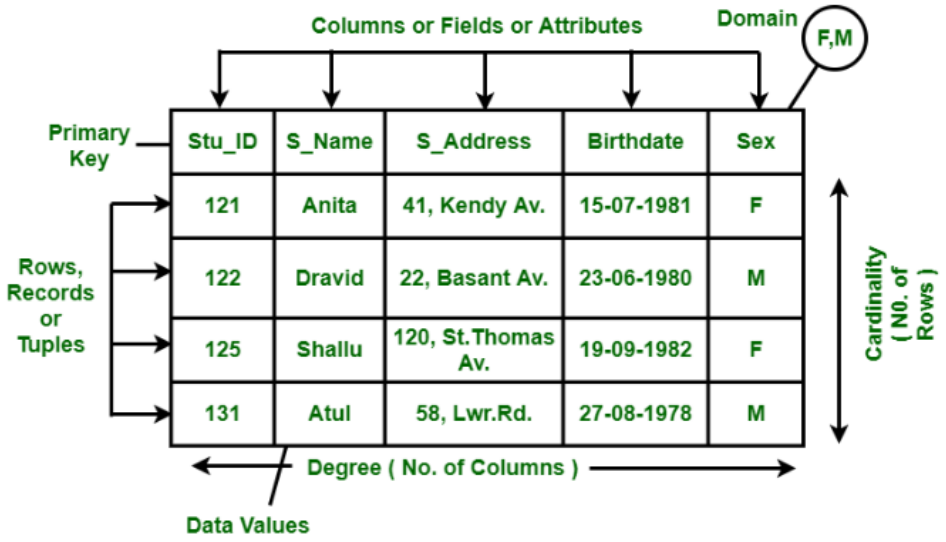
- The network model is conceptually simple to implement.
- The network model can better represent data redundancy than the hierarchical Model.
- The network model can handle one-to-many and many-to-many relationships, which is extremely useful in simulating real-world scenarios

Cons

- Because all records are maintained using pointers, the Database structure becomes extremely complicated.
- Any record’s insertion, deletion, and updating operations needs numerous pointer adjustments.
- Changing the Database’s structure is extremely difficult.

Relational model

- In this model, data is organized in tables with rows and columns.
- It uses primary and foreign keys to establish relationships between tables.
- It allows for flexible querying and manipulation of data using SQL.
- It provides a clear separation between data and its representation.
- It is widely used and has good scalability and performance.
- Examples: MySQL , SQLlite



Relational model DBMS

- **MySQL**: MySQL is relational DBMS that is widely used for web applications and other types of applications. It is known for its scalability, reliability, and ease of use.
- **Oracle Database**: Oracle Database is a relational DBMS that is widely used in enterprise applications. It is known for its high performance, scalability, and security features.
- **Microsoft SQL Server**: Microsoft SQL Server is a relational DBMS that is widely used in enterprise applications. It is known for its integration with Microsoft's other products, such as Visual Studio and Excel.
- **PostgreSQL:** PostgreSQL is relational DBMS that is known for its scalability, reliability, and support for advanced features such as JSON data types and geospatial data.
- **SQLite**: SQLite is a lightweight relational DBMS that is often used in embedded systems and mobile applications. It is known for its small size, simplicity, and ease of use.

Relational model pros and cons

Pros

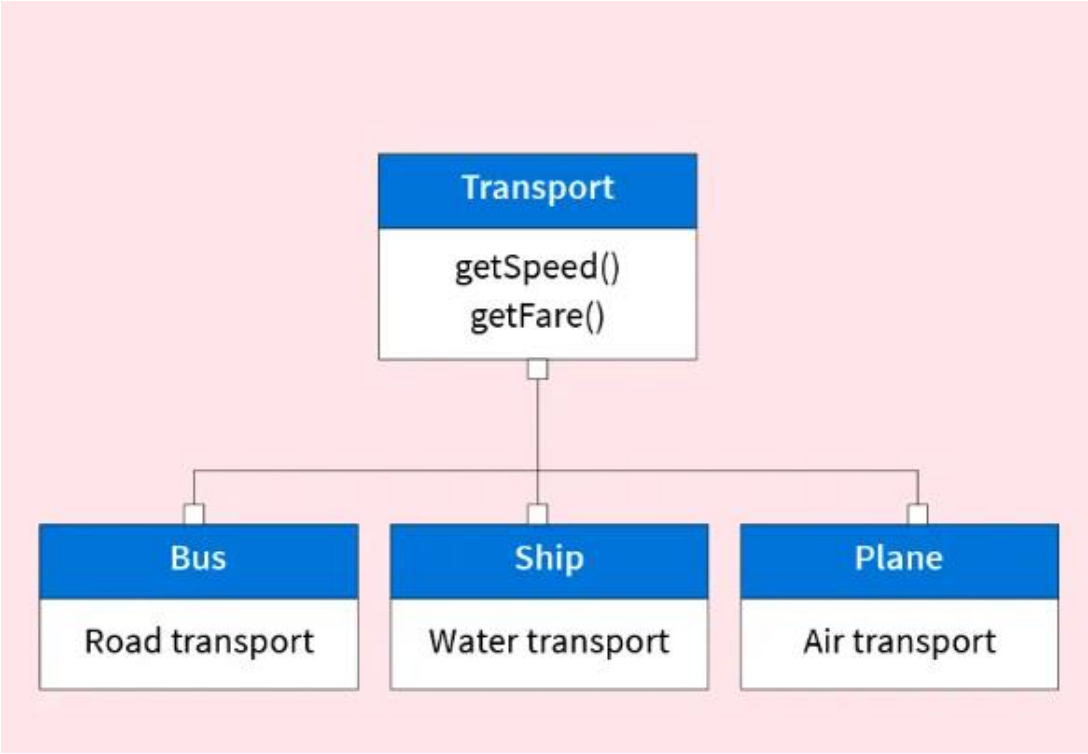
- easy data retrieval, manipulation, and analysis.
- data integrity through constraints such as primary keys, foreign keys, and unique constraints
- Powerful querying capabilities: Relational databases through support of Structured Query Language (SQL)
- Data security: Relational databases offer robust security features to protect sensitive data. Access control mechanisms, user authentication, and encryption
- Wide industry support

Cons

- Lack of flexibility: Relational databases require a predefined schema, meaning the structure of the data must be determined in advance. This can limit flexibility when dealing with evolving or unstructured data
- Scalability challenges: Scaling relational databases horizontally (across multiple servers) can be complex and challenging. As the data and workload grow, it can become difficult to maintain performance and handle large-scale data processing.
- Data redundancy: In a relational database, data is often duplicated across multiple tables to establish relationships.
- Limited support for unstructured data: Relational databases are not well-suited for handling unstructured or semi-structured data such as images, videos, or free-form text.

Object-based database model:

- In this model, data is organized as objects, which encapsulate both data and behavior.
- It allows for complex relationships and inheritance between objects.
- It supports encapsulation, polymorphism, and other object-oriented principles.
- It is well-suited for handling complex data structures and for applications with complex business logic.
- It is less widely used compared to the relational model but has gained popularity in certain domains such as multimedia and web applications.



Object-based database model Management System example

Db4objects: is an open-source object database for Java and .NET developers.

ObjectStore is an OODBMS developed by Progress Software Corporation. It provides a persistent object storage mechanism and supports object-oriented features like inheritance and polymorphism

Object-based database model pros and cons

Pros

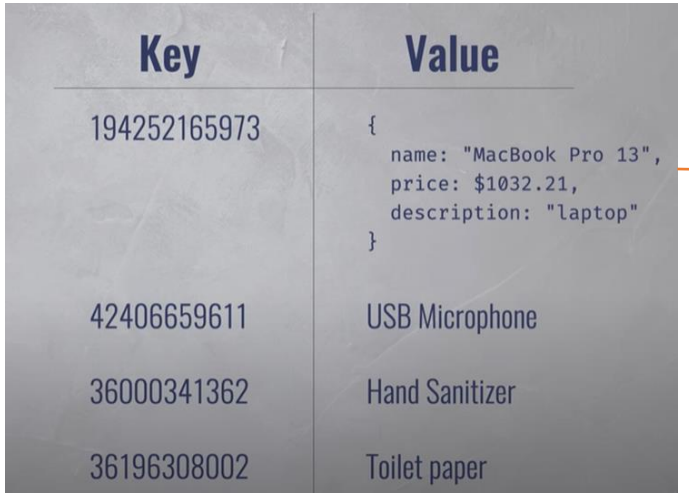
- Object-oriented databases (OODB) provide a natural and direct way to store and retrieve complex data structures, including objects, classes, and inheritance hierarchies. This allows for more intuitive and efficient handling of complex data models.
- OODB models allow for flexible schema evolution, enabling the addition or modification of attributes and behaviors without requiring alterations to the entire database structure. This makes it easier to adapt to changing requirements.
- Retrieving and manipulating objects can be faster compared to relational databases, as there is no need for complex joins or mapping between objects and relational tables.

Cons

- Limited industry adoption: This can result in a less frameworks and community support compared to relational databases.
- Lack of standardization
- Scalability limitations

NoSQL database

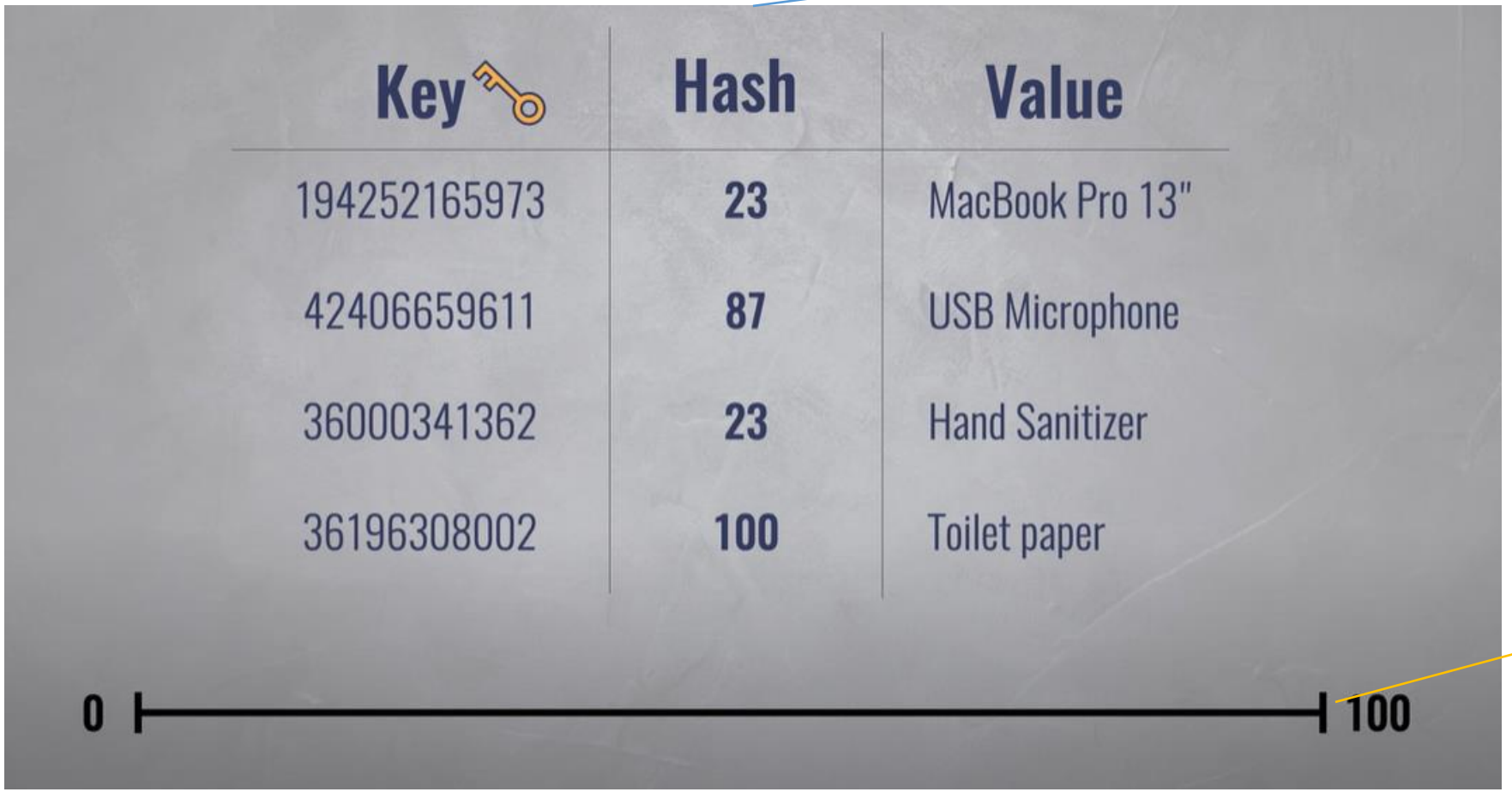
- NoSQL databases store every item in the database as a key-value pair, eliminating the need for complex relationships.
- Each item in a NoSQL database has a unique key and a value, making it a simple and efficient storage structure.



The value in a key-value pair can be a JSON document containing more data.

NoSQL database

NoSQL databases use partitions to split the workload across servers.



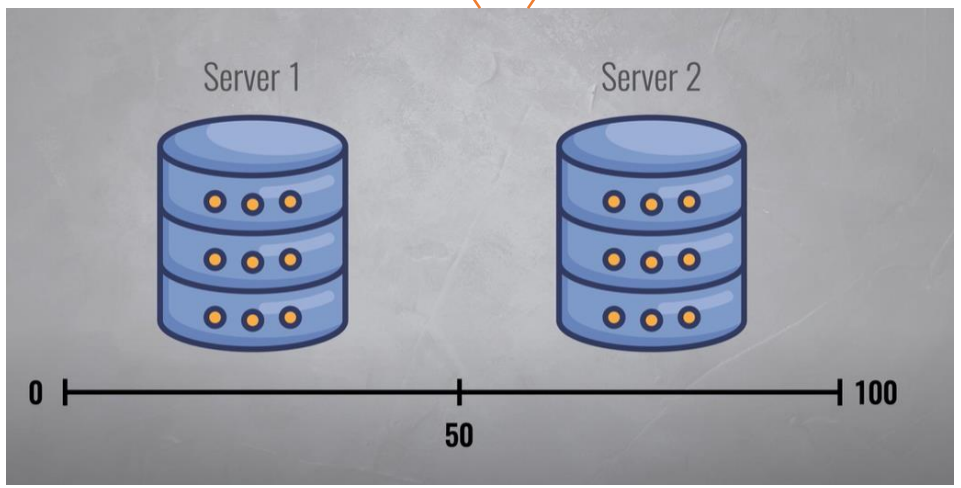
The primary key determines where an item is stored by using a hash function to convert it into a number that falls into a fixed range.

A keyspace is the range of values that a server is responsible for storing.

NoSQL database

Horizontal scaling

Doubled database capacity



Items in the NoSQL database are schemaless

NoSQL database

Pros

- NoSQL databases can scale almost without restrictions due to their large key spaces.
- NoSQL databases are schemaless, allowing items in the database to have different structures.

Cons

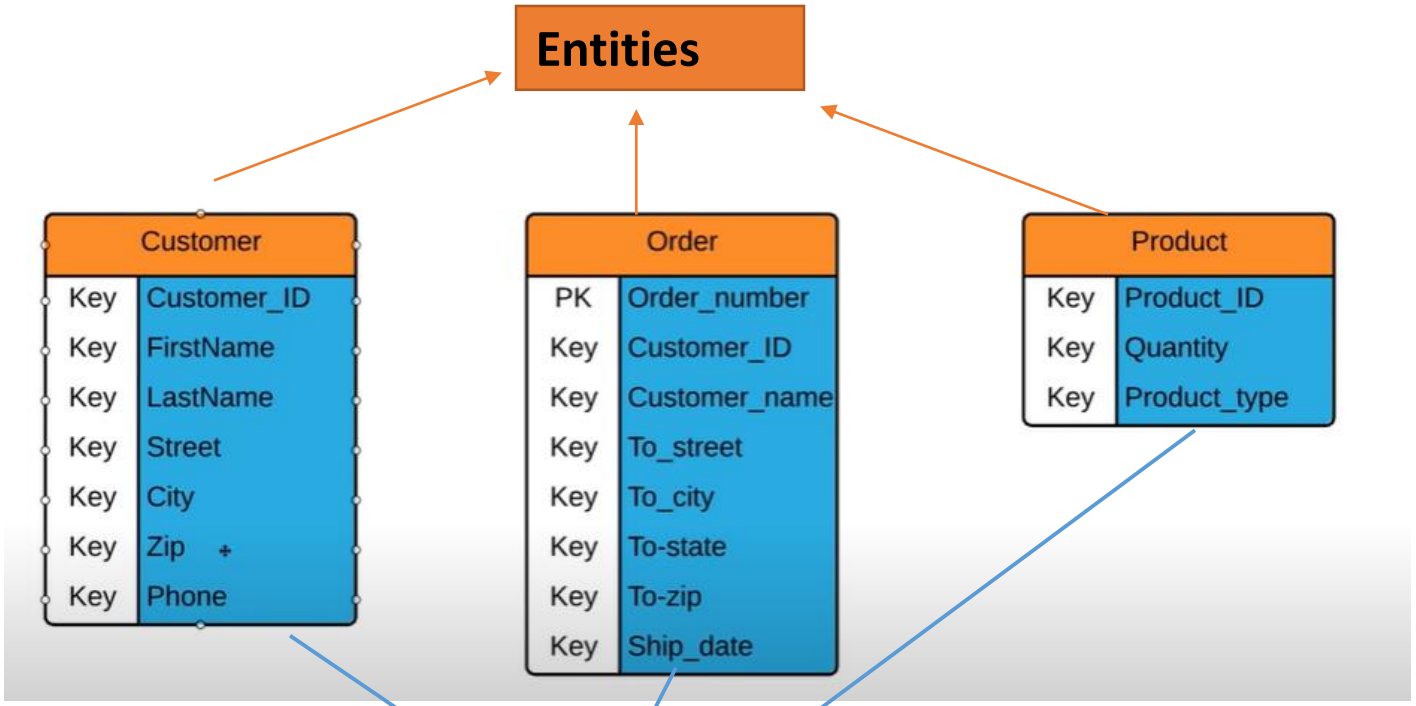
- NoSQL databases have limitations in data retrieval, only allowing retrieval by primary key.

NoSQL DBMS

Entity Relationship diagram ERD

(ERD): is a visual representation that illustrates the relationships between entities in a database.

- ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. commonly used in database design to model the structure and organization of data.
- It depicts the entities, their attributes, and the relationships between them in a database.

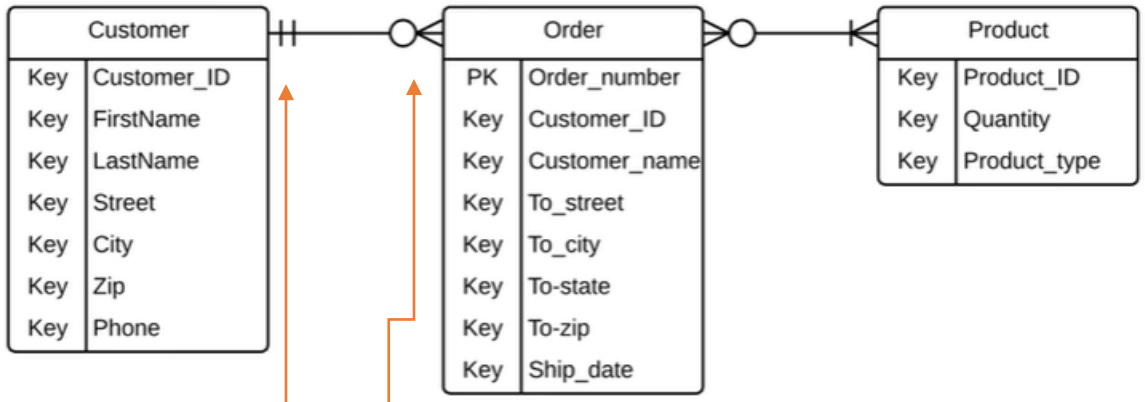


In the actual database the Entities will be the rows while the attributes will be the column

ure 3

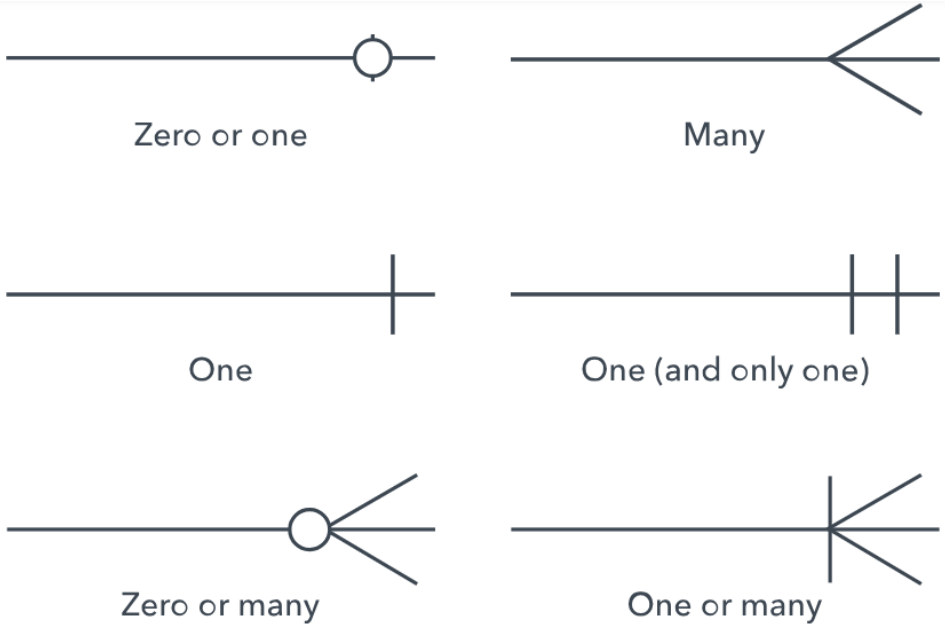
Data Models / ER model

Cardinalitv:



An order has one and only one customer

A customer may has no or many orders



Types of data modelling

This categorization focuses on the level of abstraction and the purpose of the modeling process

Conceptual data model

Describe what the system contains

Logical data model

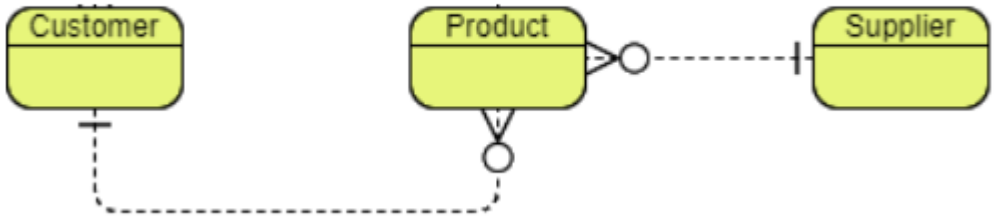
Describe how the system will be implemented

Physical data model

Describe how the system will be implemented using specific DBMS

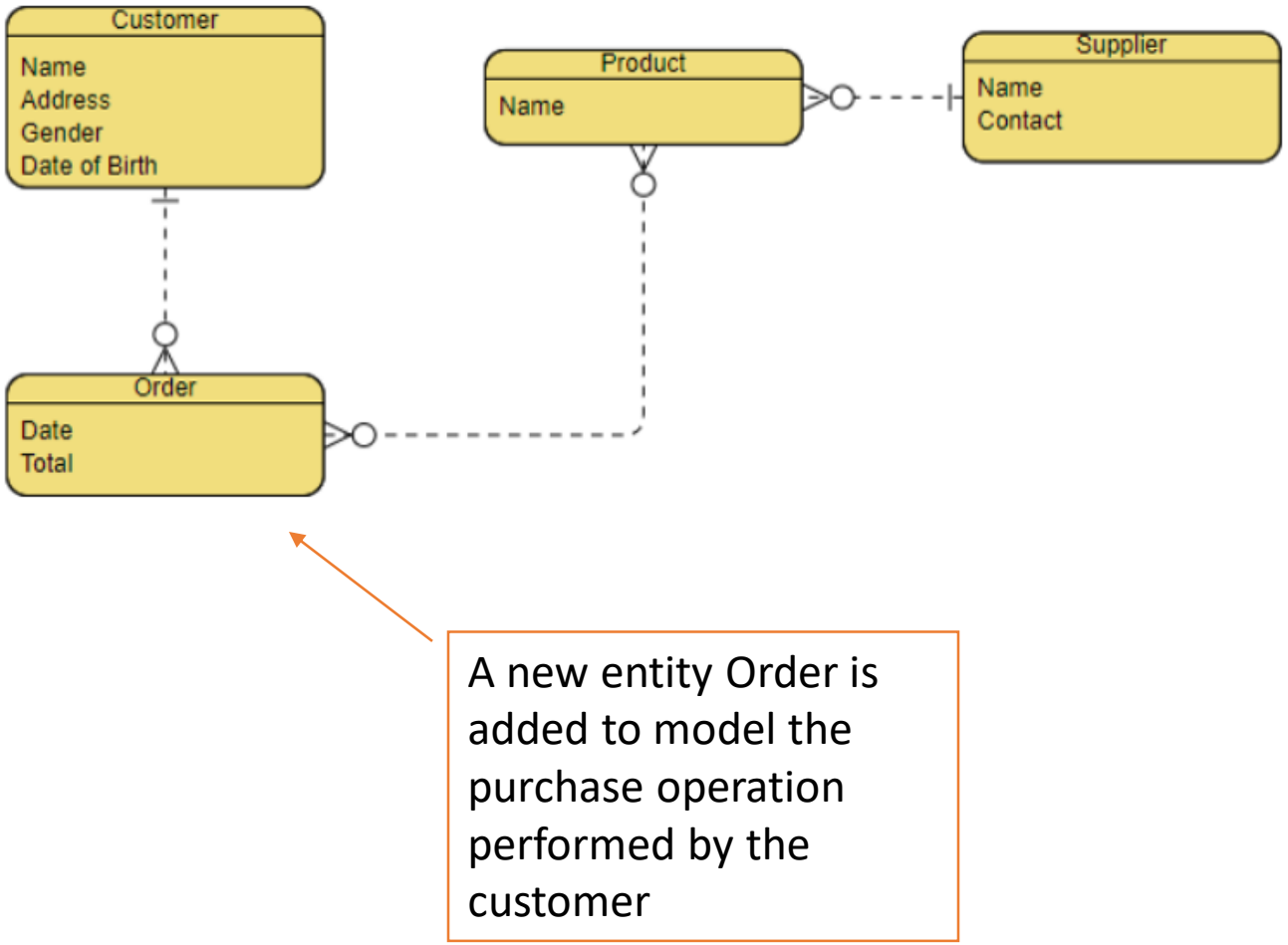
Conceptual data model

- A **Conceptual Data Model** is an organized view of database concepts and their relationships.
- The purpose of creating a conceptual data model is to establish entities, their attributes, and relationships.
  - **Entity**: A real-world thing
  - **Attribute**: Characteristics or properties of an entity
  - **Relationship**: Dependency or association between two entities
- In this data modeling level, there is hardly any detail available on the actual database structure.
- Business stakeholders and data architects typically create a conceptual data model.



Logical Data model

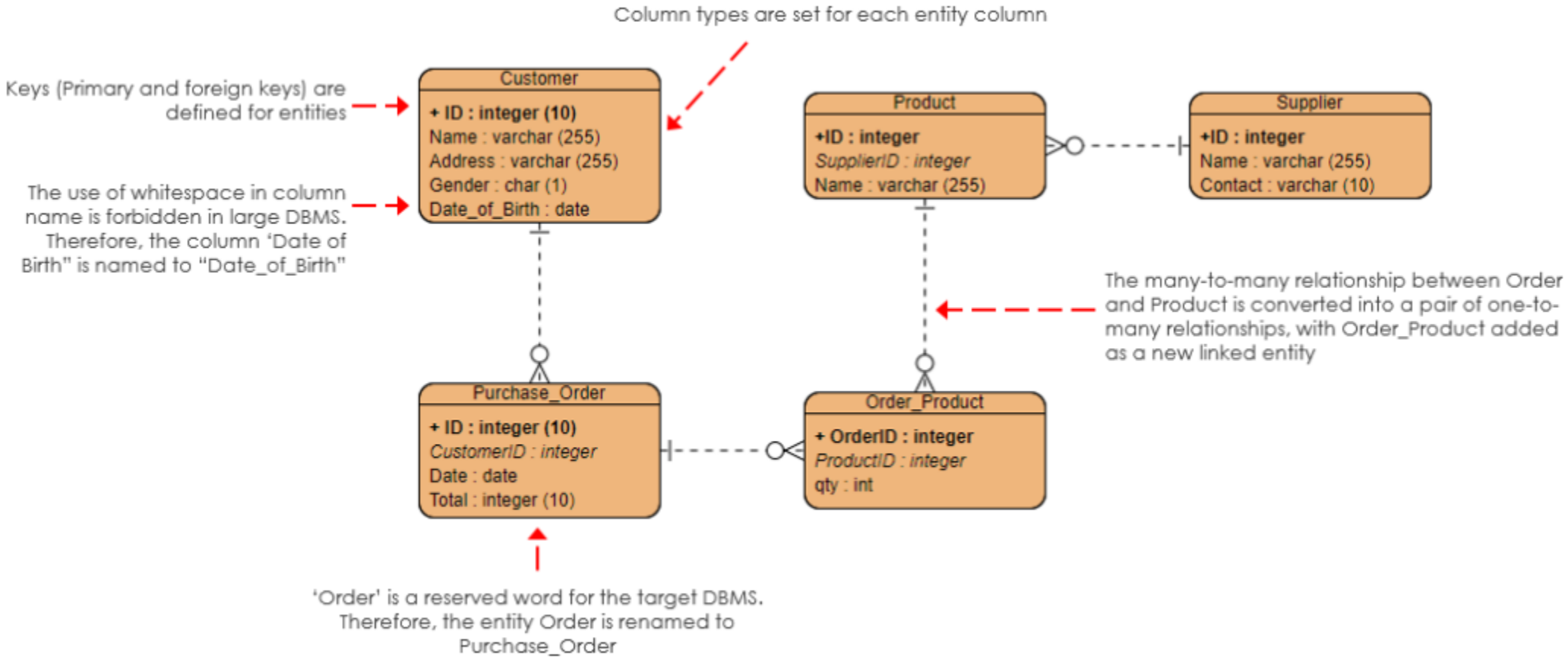
- The logical data model adds further information to the conceptual data model elements. So it is a detailed version of a Conceptual ERD
- The advantage of using a Logical data model is to provide a foundation to form the base for the Physical model.
- However, the modeling structure remains generic.



Physical Data Model

- A physical data model represents how the data model will be implemented in the specific database.
- physical implementation requires defining low-level detail that may be specific for the certain database provider.
- A physical data model elaborates on the logical data model by assigning each column with type, length… etc.
- Transitioning from a logical data model to a physical data model requires more iterations and fine-tuning of the entities and relationships defined in the logical data model.

Physical Data Model



Exercise

- A university wants to design a database system to manage student information.
- The university has multiple departments, each offering various courses.
- Each student is enrolled in one or more courses. The database should store information about students, courses, departments, and the enrollment status of students in courses.
- Each student has a unique student ID, name, and date of birth. Each course has a unique course ID, title, and credit hours. Each department has a unique department ID and name.
- Draw an ER diagram that represents this explained scenario

End of lecture 3

جامعة الشام الخاصة
Al-Sham Private University

Development – testing and maintenance

Lecture_4

SDLC - Software Development Life Cycle – part2

1- Development

1-1- Choosing the Programming Language:

- In this stage of SDLC the actual development starts and the product is built. There are important concepts that should be taken into account:
- **1-1-1- Project Requirements:** Different programming languages excel in different areas, such as
- web development: HTML/CSS , JavaScript, PHP,

- mobile app development:
  - Swift: Developed by Apple, it is used for creating iOS and macOS applications.
  - React Native: It is a framework that allows developers to create cross-platform mobile apps using JavaScript.

- data analysis:
  - Python: Widely used for data analysis and scientific computing due to its extensive libraries such as NumPy, Pandas and Matplotlib.
  - SQL: A language used for managing and querying databases.

1- Development
1-1- Choosing the Programming Language:

- **1-1-2- Performance and Efficiency**
- Some programming languages are known for their speed and efficiency, making them suitable for high-performance applications. On the other hand, certain languages prioritize ease of development and may sacrifice some performance. Evaluate the trade-offs between performance and development speed to make an informed decision.

1- Development

- **1-1- Choosing the Programming Language:**
- **1-1-1- Performance and Efficiency**

C/C++:

- high performance and efficiency.
- provide low-level control over hardware resources.
- allow developers to optimize code for speed
- can be more complex and time-consuming to develop in compared to other languages.

Python

- prioritizes ease of development and readability,
- offers a wide range of libraries and frameworks that accelerate development speed.
- Python is an interpreted language, which can result in slower execution compared to compiled languages like C++.

Java

- balance between performance and development speed.
- provides a wide range of libraries and tools that simplify development.
- Java's virtual machine (JVM) allows for platform independence but can introduce some overhead.

JavaScript

- used for web development
- offers fast development speed due to its simplicity and extensive libraries.
- it may not be as performant as lower-level languages like C++ due to its interpreted nature and reliance on browser engines.

1- Development

- **1-1- Choosing the Programming Language:**
- **1-1-2- Scalability and Flexibility**
- Consider the scalability and flexibility requirements of your application. Some programming languages are better suited for small-scale projects, while others excel in handling large-scale enterprise applications. Additionally, evaluate the language's ability to integrate with other technologies and frameworks, as this can impact the future growth and expansion of your application.

1- Development

- **1-1- Choosing the Programming Language:**
- **1-1-2- Scalability , Flexibility and integration**

1.Scalability: Suppose you are developing a web application that you expect to attract a large number of users. In this case, you might consider using a language like **Java** or **Python**, which are known for their ability to handle high traffic and scale up easily. These languages have robust frameworks and libraries that can help you build and maintain complex, high-performance applications.

2.Flexibility: Let's say you are working on a project that requires frequent updates and modifications. In this scenario, you might choose a language like **Ruby** or **JavaScript**, which are known for their flexibility and ease of use. These languages have concise syntax and dynamic typing, which make them ideal for rapid prototyping.


3.Integration: Suppose you are building an e-commerce platform that needs to integrate with various payment gateways and shipping providers. In this case, you might consider using a language like **PHP**, which have extensive libraries and modules for handling web services and APIs. This languag can help you seamlessly connect your application to external systems and services, enabling you to offer a more comprehensive solution to your customers.

2- Testing

- Testing in software refers to the process of evaluating a software system or application to identify any defects, errors, or bugs. It involves running the software under controlled conditions to verify that it meets the specified requirements and functions as intended.
- The main objectives of software testing are to ensure the quality, reliability, and functionality of the software, as well as to uncover any issues that may impact the user experience or cause system failures.
-

2- Testing
2-1- Levels of software testing

- **1-2-1- Unit Testing:** Unit testing is the process of testing individual components or units of code to verify their correctness and functionality.
- It focuses on testing small, isolated parts of the application, such as functions, methods, or classes, to ensure they work as expected.
- Unit tests are typically written by developers themselves and are executed frequently during the development process.
- The primary goal of unit testing is to identify and fix bugs early in the development cycle.

2- Testing
2-1- Levels of software testing
2-1-1- Unit testing

```python
import unittest

# Assume we have a simple function that adds two numbers
def add_numbers(a, b):
    return a + b

# Define a test case class that inherits from unittest.TestCase
class TestAddNumbers(unittest.TestCase):
    # Define a test method that tests the add_numbers function
    def test_add_numbers(self):
        # Test the function with two integers
        result = add_numbers(2, 3)

        # Verify that the result is as expected
        self.assertEqual(result, 5)

        # Test the function with two floats
        result = add_numbers(2.5, 3.5)

        # Verify that the result is as expected
        self.assertEqual(result, 6.0)

if __name__ == '__main__':
    unittest.main()
```

We then define a test case class called TestAddNumbers that inherits from unittest.TestCase.

we define a test method called test_add_numbers that tests the add_numbers function.

Finally, we use the unittest.main() method to run the test case. This will execute the test_add_numbers method and report any failures or errors.

```
In [2]: runfile('C:/Users/MA-Laptop/.spyder-py3/unittest.py',
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

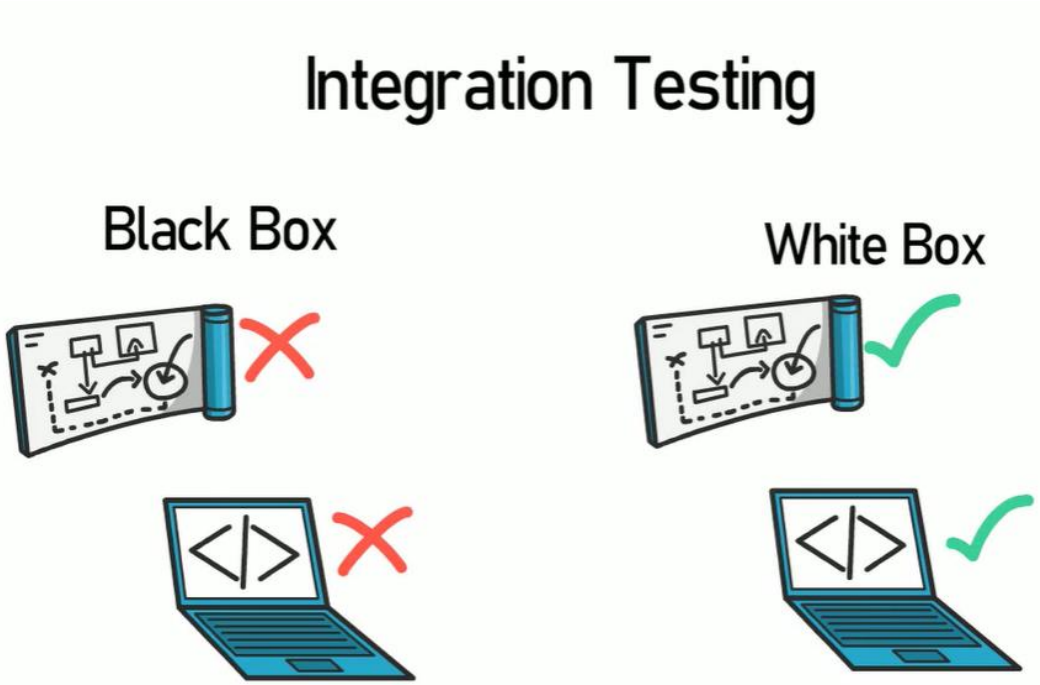
2- Testing
2-1- Levels of software testing

- **2- 1-2- Integration testing:** Integration testing is the process of testing the interaction between different components or modules of an application.
- It aims to identify any issues that may arise when these components are integrated and working together.
- Integration testing ensures that the individual units of code function correctly when combined and that they communicate and exchange data as expected.
- During integration testing, developers test:
  - the interfaces and interactions between various modules,
  - subsystems, or services.
- This type of testing helps uncover issues such as
  - incompatible data formats,
  - incorrect data transfers, or
  - communication failures.

2- Testing
2-1- Levels of software testing
2-1-2-Integration testing

Black box testing

- focuses on testing the functionality and behavior of a system without having knowledge of its internal implementation details.
- Testers treat the system as a "black box" and only interact with its inputs and outputs.
- Testers are not concerned with how the system processes the inputs or produces the outputs.



white box testing

- involves having knowledge of the internal structure, design, and implementation of the system.
- Testers use this knowledge to design test cases that exercise specific conditions within the system.

2- Testing
2-1- Levels of software testing
2-1-2- Integration test

```python
import unittest

# Assume we have two components: Calculator and Database
class Calculator:
    def __init__(self):
        self.database = Database()

    def add(self, a, b):
        result = a + b
        self.database.save_operation("add", a, b, result)
        return result

class Database:
    def __init__(self):
        self.operations = []

    def save_operation(self, operation, operand1, operand2, result):
        self.operations.append({
            "operation": operation,
            "operand1": operand1,
            "operand2": operand2,
            "result": result
        })
```

```python
class TestCalculatorIntegration(unittest.TestCase):
    def test_add_integration(self):
        # Create an instance of the Calculator
        calculator = Calculator()

        # Call the add method
        result = calculator.add(2, 3)

        # Verify that the result is correct
        self.assertEqual(result, 5)

        # Verify that the operation was saved in the database
        database = calculator.database
        saved_operations = database.operations
        self.assertEqual(len(saved_operations), 1)
        self.assertEqual(saved_operations[0]["operation"], "add")
        self.assertEqual(saved_operations[0]["operand1"], 2)
        self.assertEqual(saved_operations[0]["operand2"], 3)
        self.assertEqual(saved_operations[0]["result"], 5)

if __name__ == '__main__':
    unittest.main()
```

```
In [5]: runfile('C:/Users/MA-Laptop/.spyder-py3/integrationTestASPU.py
Laptop/.spyder-py3')
.
----------------------------------------------------------------------
Ran 1 test in 0.002s

OK
```

2- Testing - Unit test vs. Integration test

Unit Testing	Integration Testing
It tests small modules or a piece of code of an application or a product	Two or more units of a program are combined and tested as a group
It's a quick write-and-run test	It is slower to run
Typically performed by a software developer	It is traditionally carried out by a separate team of testers
It can be performed at any time	It is usually carried out after Unit Testing but before the overall system testing
It is very limited in scope, as it only covers a piece of code	It has wider scope as it covers broader part of the application or the product
It focus on one single module	It pays attention to integration among two or more modules
Finding errors is relatively easy	Finding errors is more difficult
Test executor knows the internal design of the software	Testers don't necessary know the internal design of the software

2- Testing
2-1- Levels of software testing

- **2-1-3: System testing**: is a comprehensive testing approach that evaluates the entire application as a whole.
- It focuses on verifying that the integrated system meets the specified requirements and functions correctly in its intended environment.
- System testing is typically performed after integration testing and ensures that all components work together seamlessly.
- During system testing, testers simulate real-world scenarios and test the application's functionality, performance, security, and usability.
- This type of testing helps identify any issues that may arise due to the interaction between different components or external systems.
- System testing also ensures that the application meets the end-users' expectations and performs as intended in different environments.

2- Testing
2-1- Levels of software testing
2-1-3- System testing

```python
class CalculatorSystemTest(unittest.TestCase):

    def setUp(self):
        self.calculator = Calculator()
        self.db = Database()

    def tearDown(self):
        self.db.clear()

    def test_addition(self):
        # Perform addition using the calculator
        # Verify the result is correct
        # Verify the calculation is stored in the database
        pass


    def test_subtraction(self):
        # Perform subtraction using the calculator
        # Verify the result is correct
        # Verify the calculation is stored in the database
        pass

    def test_multiply(self):
        pass

    def test_division(self):
        pass
```

- The setUp method sets up the test environment by creating instances of the Calculator and Database classes. The tearDown method clears the database after each test.
- This simple code is used as a system test code for a simple calculator App

2- Testing
2-1- Levels of software testing
Integration test vs. System testing

	Integration test	System testing
scope	focuses on testing the interactions and interfaces between individual components or modules.	verifies the behavior and functionality of the entire system as a whole
Timing	performed during the integration phase when individual modules are being combined and tested together.	performed after integration testing and once all the individual components have been integrated into a complete system
Approach	more technical and focused on ensuring that individual components or modules work together correctly and produce the expected results.	performed from a user's perspective and focuses on testing the system as a whole to ensure that it meets the specified requirements and objectives.
Test environment	performed in a controlled environment with simulated components	performed in an environment that closely resembles the production environment, including hardware, software, and other systems that interact with the system being tested.

2- Testing
2-1- Levels of software testing
2-1-4: Acceptance Testing

- Acceptance Testing is the last phase of software testing performed after System Testing and before making the system available for actual use.
- It focuses on validating the system from a user's perspective, ensuring that it meets the intended business requirements and user expectations.
- Acceptance testing is a quality assurance process that determines to what degree an application meets end users' approval.
- An acceptance test returns either a pass or fail result. A fail suggests that there is a flaw present, and the software should not go into production.

2- Testing
2-1- Levels of software testing
System Testing vs. Acceptance testing

	System testing	**Acceptance testing**
Scope	verifies the system as a whole, including its components, interactions, and interfaces. It aims to ensure that the system functions correctly and meets its technical specifications	focuses on validating the system from a user's perspective, ensuring that it meets the intended business requirements and user expectations.
Testers	developers , Quality Assurance QA engineers	end-users or business stakeholders

3- Deployment

- deployment refers to the process of making a software application available for use by end-users.
- It involves the installation and configuration of the software on the target environment, which could be a physical server, a virtual machine, or a cloud-based platform.
- It involves tasks such as
  - packaging the application
  - configuring databases and servers,
  - setting up security measures
  - and ensuring compatibility with the target environment.

3- Deployment
3-1- Deployment Methods
3-1-1- Manual Deployment

- This involves manually installing and configuring the software on the target environment. It may require running installation scripts, copying files, and configuring settings by hand.
- **Example**: A small business creates a web application for managing customer orders. They manually deploy the application by copying the necessary files to a physical server located in their office.
- They configure the server settings manually, including installing and configuring the required software dependencies. They also manually test the application to ensure that it is working correctly.

3- Deployment
3-1- Deployment Methods
3-1-2- Automated Deployment

- This involves using deployment tools or frameworks to automate the installation and configuration process. These tools can streamline the deployment process, ensure consistency, and allow for easier scaling and updates.
- **Example**: A software development team creates a mobile application for their clients. They use a continuous integration/continuous deployment (CI/CD) tool like Jenkins to automate the deployment process. Whenever a new code change is committed to the repository, Jenkins automatically builds the application, runs automated tests, and deploys it to the app stores.

3- Deployment
3-1- Deployment Methods
3-1-3 Continuous Deployment

- This is an advanced deployment method where software changes are automatically deployed to production environments as soon as they pass automated tests and quality checks. It enables rapid and frequent releases.
- **Example**: A software company uses a microservices architecture and deploys their services to a cloud-based platform like Amazon Web Services (AWS). They use a continuous deployment approach, where code changes are automatically deployed to production environments as soon as they pass automated tests and quality checks. The deployment process is fully automated, and the services are deployed to multiple regions for high availability.

3- Deployment
3-2- Software Types and Target Environments:

- Web Applications: Web applications are deployed on web servers such as Apache HTTP Server.
- They can be hosted on physical servers, virtual machines, or cloud platforms like AWS(Amazon Web Services), or Google Cloud Platform.

3- Deployment
3-2- Software Types and Target Environments:

- Mobile Applications: Mobile applications can be deployed through app stores like Apple App Store or Google Play Store.
- Developers typically package the application into an installer file (APK for Android or IPA for iOS) and submit it to the respective app store.

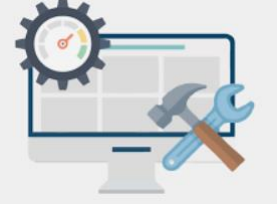
3- Deployment
3-2- Software Types and Target Environments:

- Desktop Applications: Desktop applications are typically deployed by creating installer packages for different operating systems (e.g., Windows Installer for Windows or DMG for macOS).
- Users can then download and install the application on their local machines.

3- Deployment
3-2- Software Types and Target Environments:

- Database Systems: Database systems like MySQL or MongoDB are deployed on dedicated servers or cloud-based platforms.
- They require installation and configuration of the database software on the target environment, along with setting up databases and user access controls.

4- Maintenance



- Maintenance refers to the process of managing and enhancing a software system after it has been deployed. It involves making necessary changes, fixing bugs, and adding new features to ensure the software remains functional and up-to-date.
- There are generally three types of maintenance in SDLC:

1. **Corrective Maintenance**: This type of maintenance focuses on fixing defects or bugs in the software. It involves identifying and rectifying errors that may have been discovered by users or during testing.
2. **Adaptive Maintenance**: Adaptive maintenance involves making changes to the software to keep it compatible with changes in the environment, such as operating system upgrades or hardware changes. It ensures that the software remains functional and usable in different contexts.
3. **Perfective Maintenance**: Perfective maintenance aims to improve the software by enhancing its performance, reliability, or usability. It involves adding new features or optimizing existing ones to meet changing user requirements or expectations.

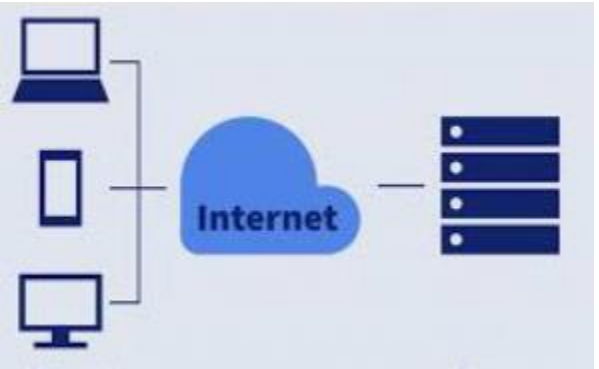
End of lecture 4

ASPU University
Faculty of Informatics Engineering
Fourth Year
Application development Course
2023 - 2024

جامعة الشام الخاصة
Al-Sham Private University

Introduction to Building a Client – Server Application

Lecture 5

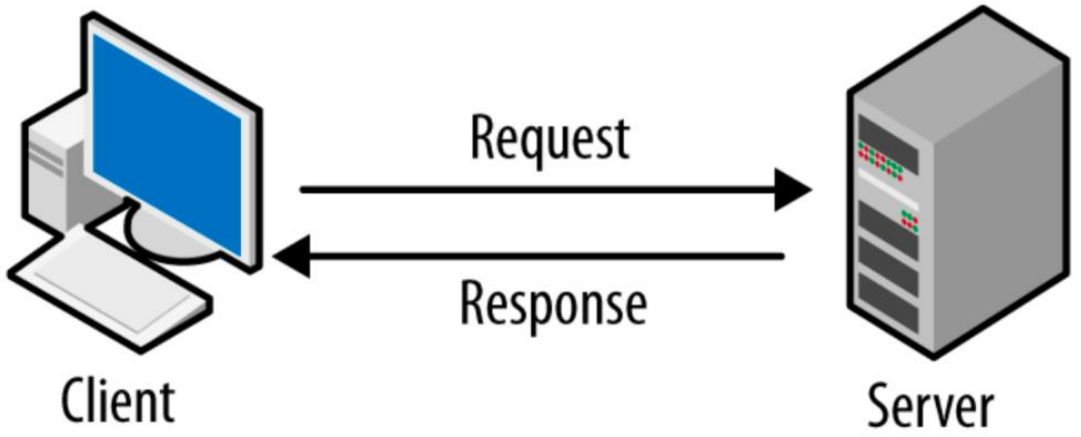


Lecture overview

- Client – server Model
- Basics of PHP
- Handling Form request
- Get vs Post
- Server side validation
- File uploads
- Handling file uploads

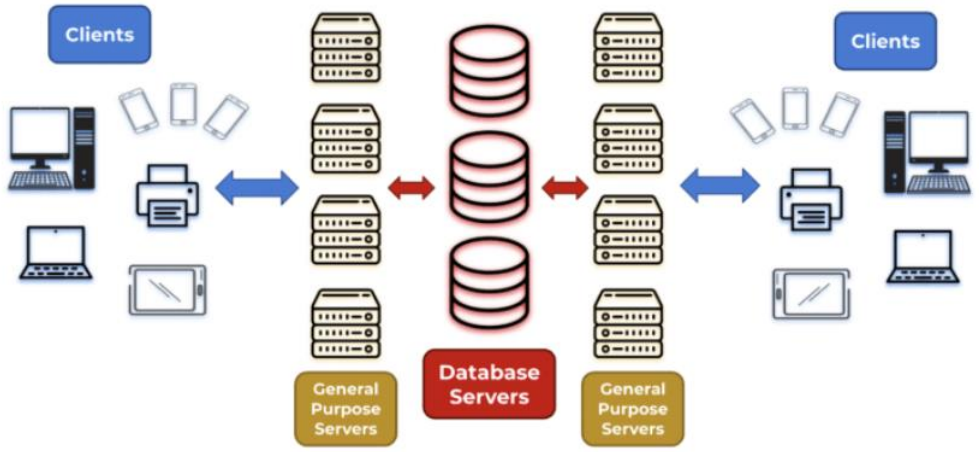
Client – server Model

- A client-server application is a software architecture where a client program or device interacts with a server to request services or resources using HTTP protocol.
- In this model, the client is responsible for initiating communication and making requests, while the server handles these requests and provides the requested services or resources.
- The client and server can be separate machines or software components connected over a network.



Client – server Model

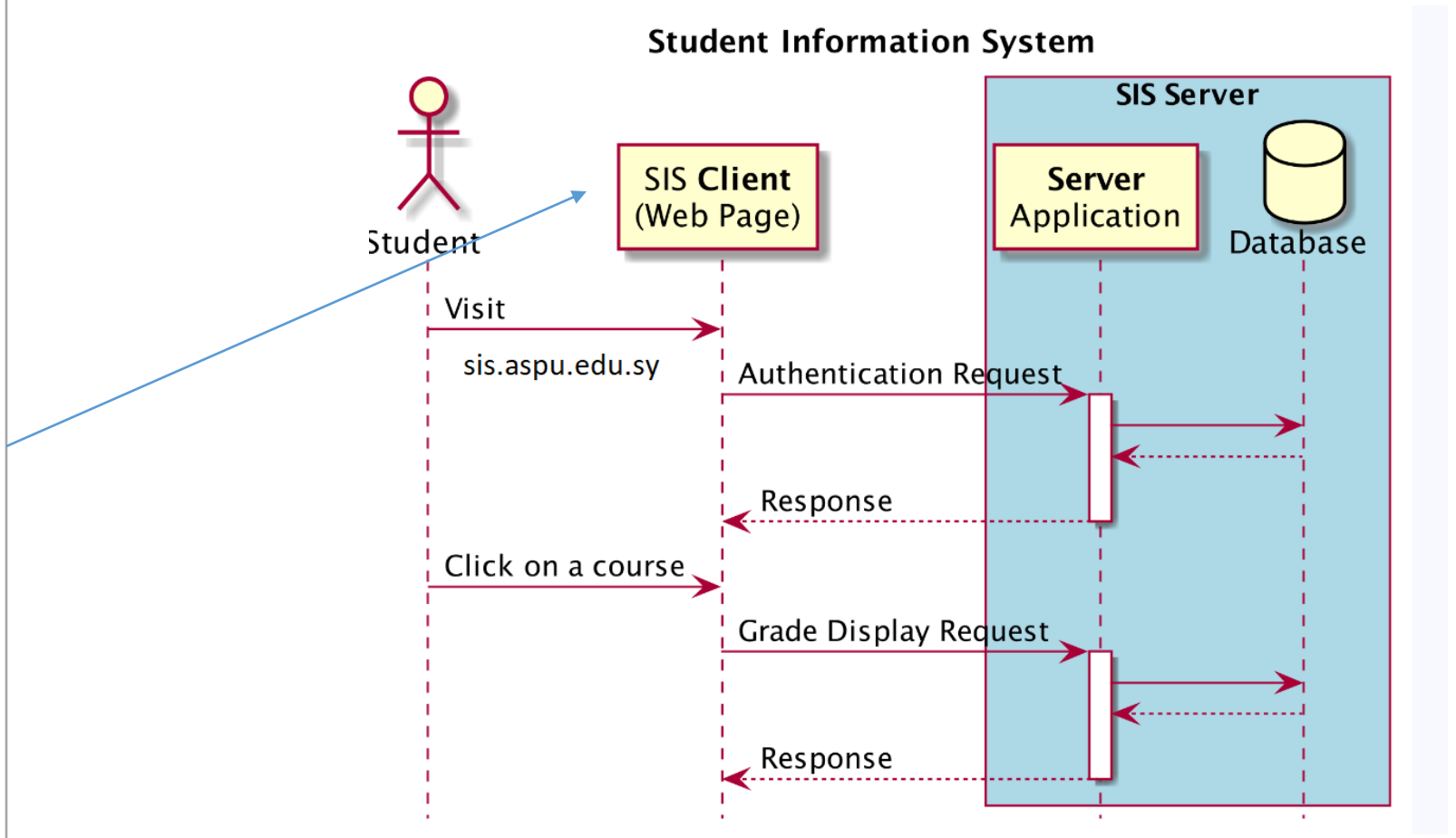
- The client-server architecture allows for distributed processing, where the workload can be divided between the client and server, enabling efficient resource utilization and scalability.
- This approach enables applications like email clients, file transfer programs, and database systems to function seamlessly by leveraging the strengths of both the client and server components.



An example of Client-Server Application

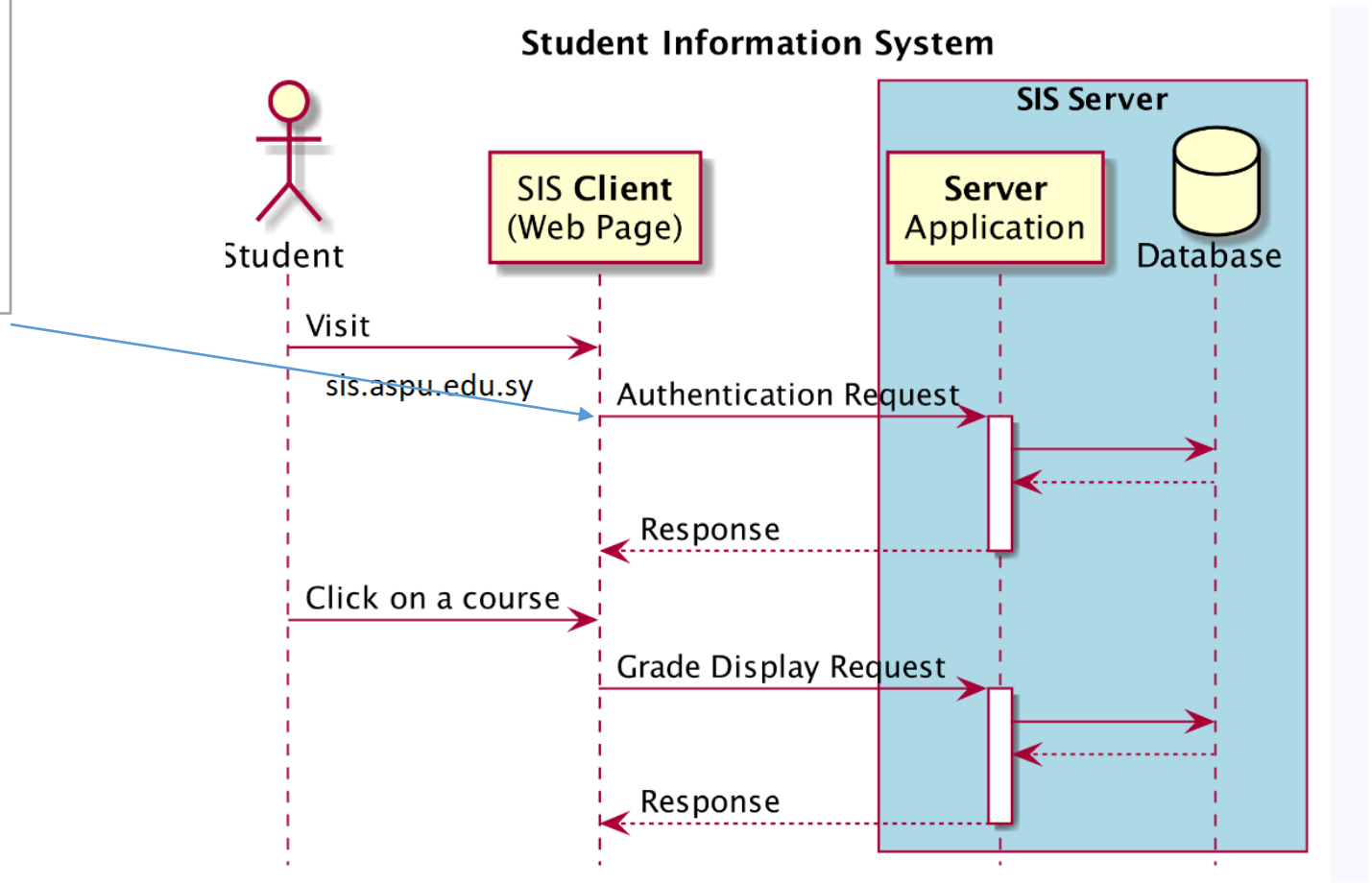
You visit sis.aspu.edu.sy using an internet browser like Chrome on any device that provides internet browsing.

Following your visit, a web-page will be displayed in your browser.
This web-page is the user-interface of SIS and it constitutes the "client" side of the SIS application.
The client application allows you to interact with the SIS server (which, for the intent of this example, may be a physical computer at one of ASPU's campuses).



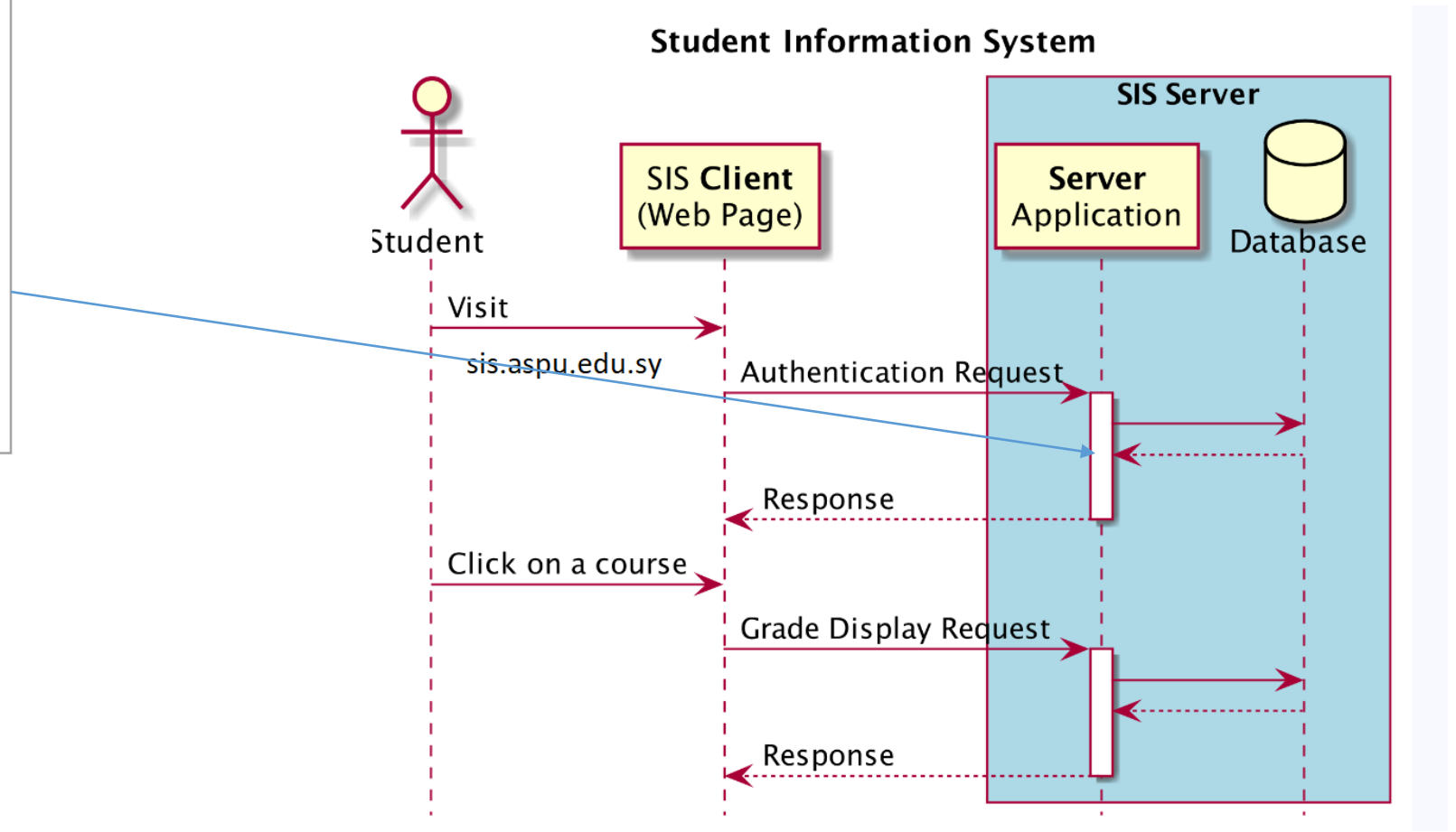
An example of Client-Server Application

On the client-side, you enter your username and password to log into SIS. As this stage, the client application (web-page where you put your username and password) will send (authentication) **request** to the (SIS) server.

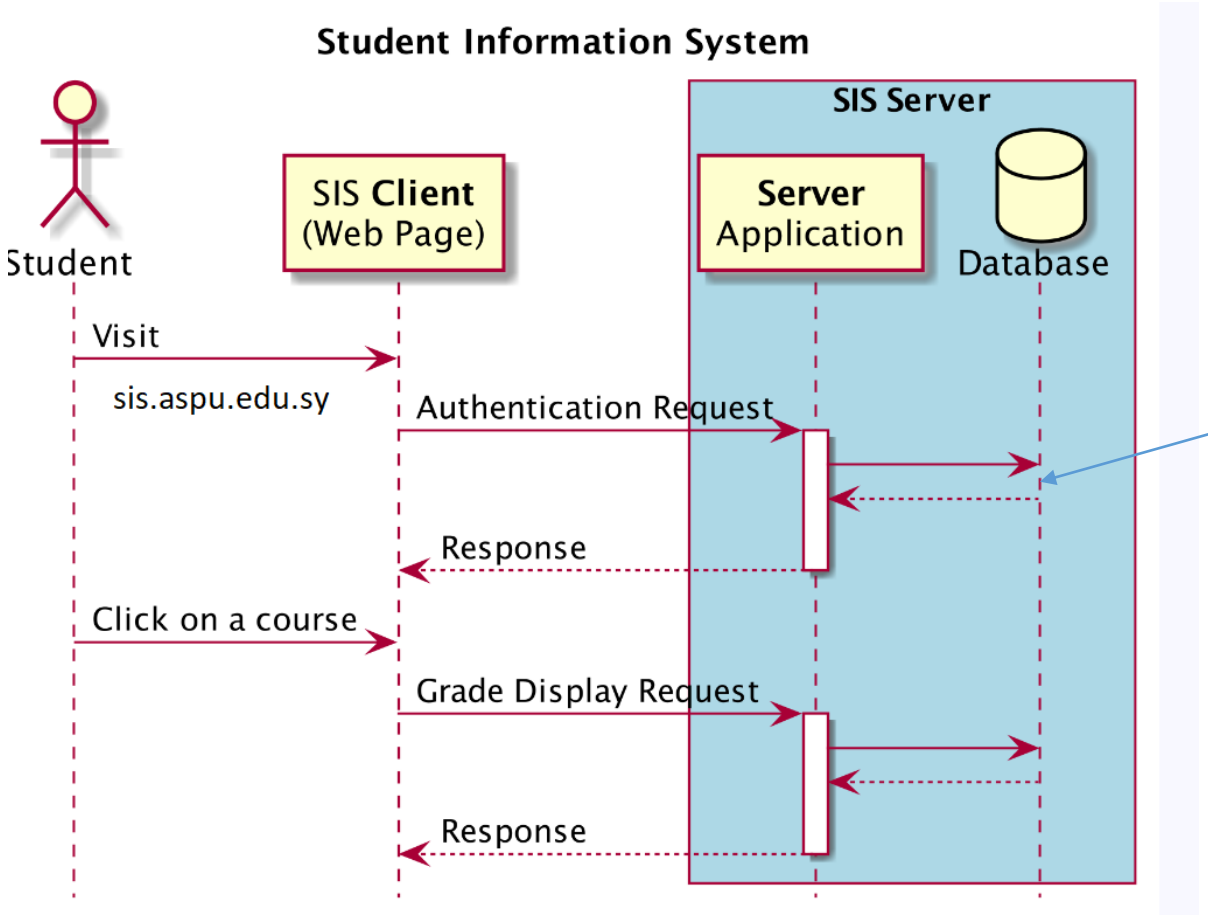


An example of Client-Server Application

Your (authentication) request travels across the internet to the SIS server. The server, which is actively *listening* for requests from **all users**, receives your request and triggers a **response**.

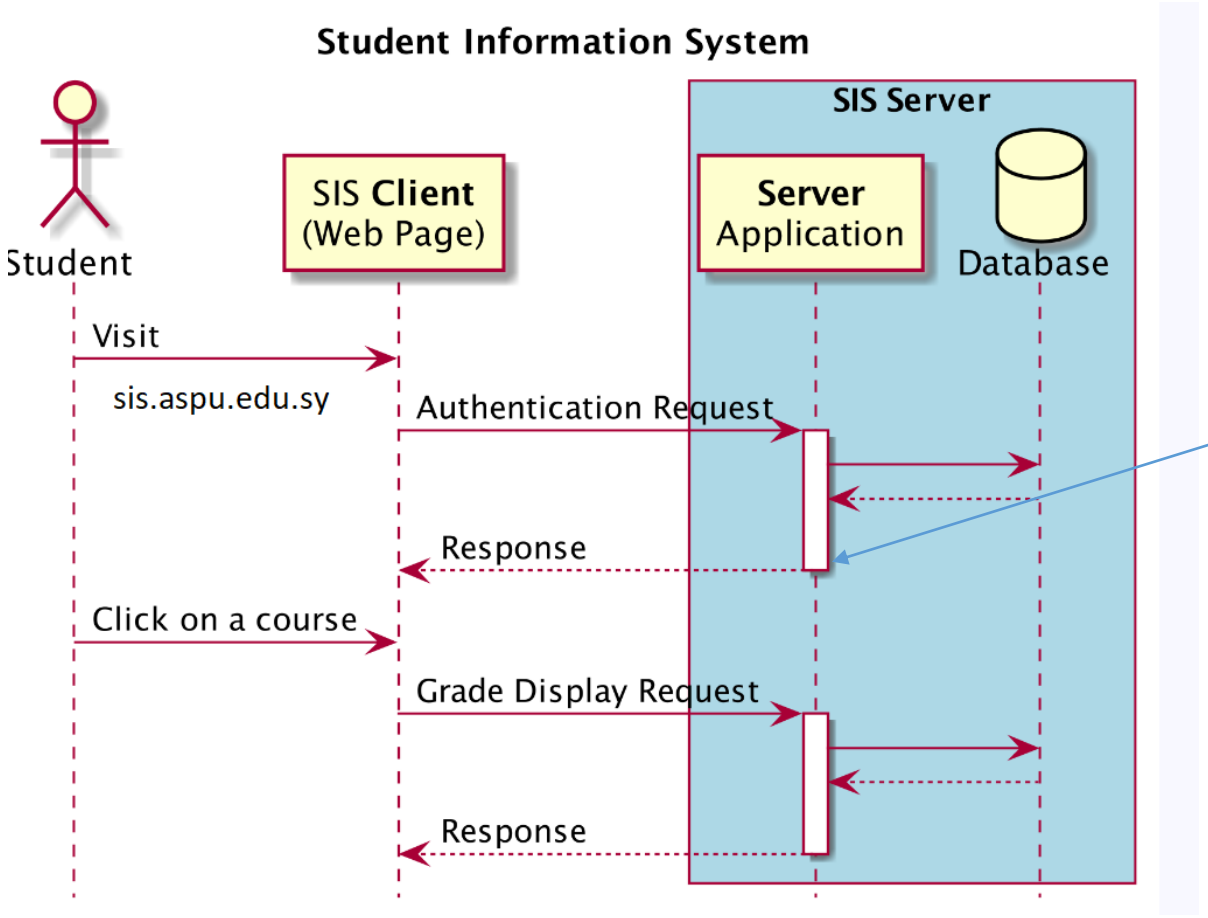


An example of Client-Server Application



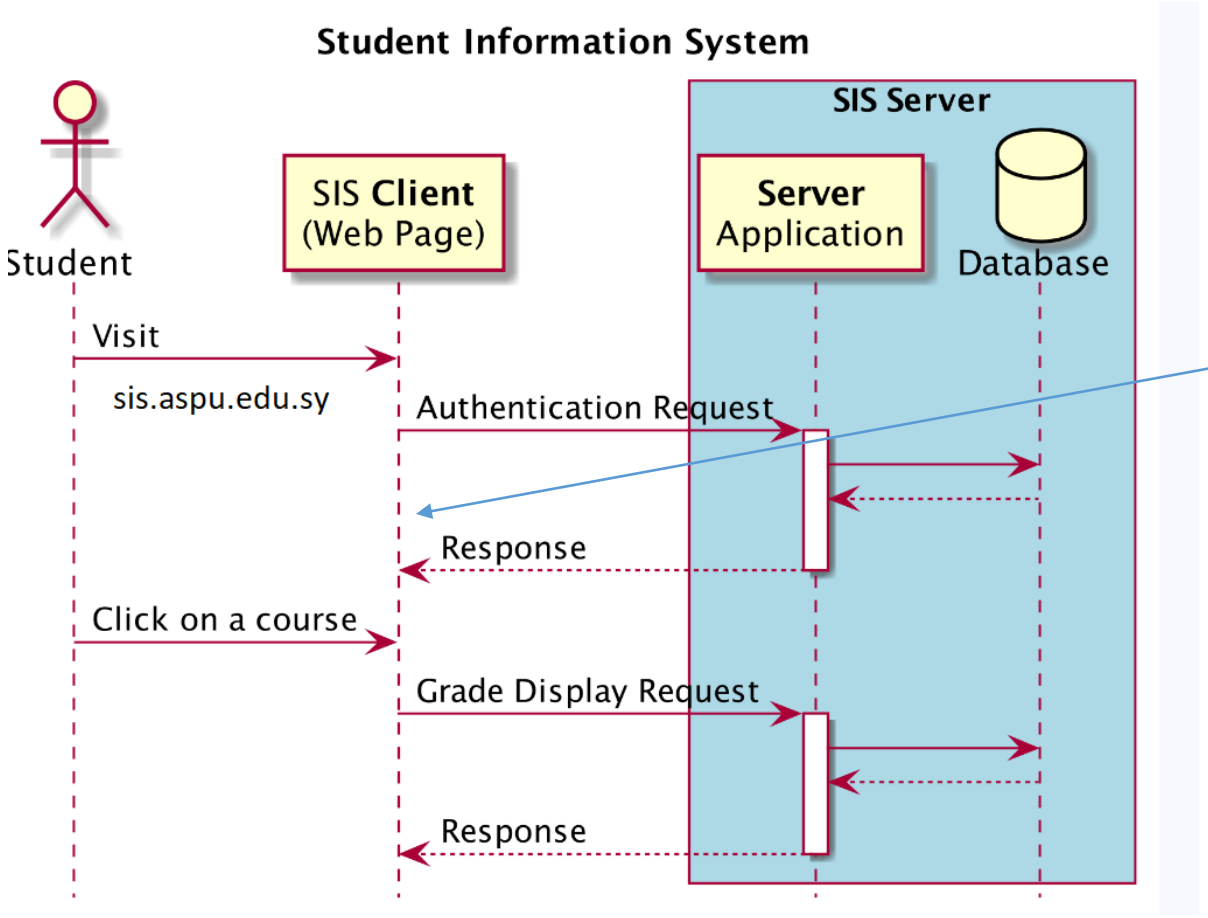
The response process on the server makes a **database query** to check your login credentials. This database may contain other information about you such as your grades. The database query is executed, and the database sends the requested data back to the server.

An example of Client-Server Application



The server receives the data that it needs from the database, and it is now ready to construct and send its response back to the client (you). In this case, the response would be the *privilege* to access SIS (assuming your login credential were accredited).

An example of Client-Server Application



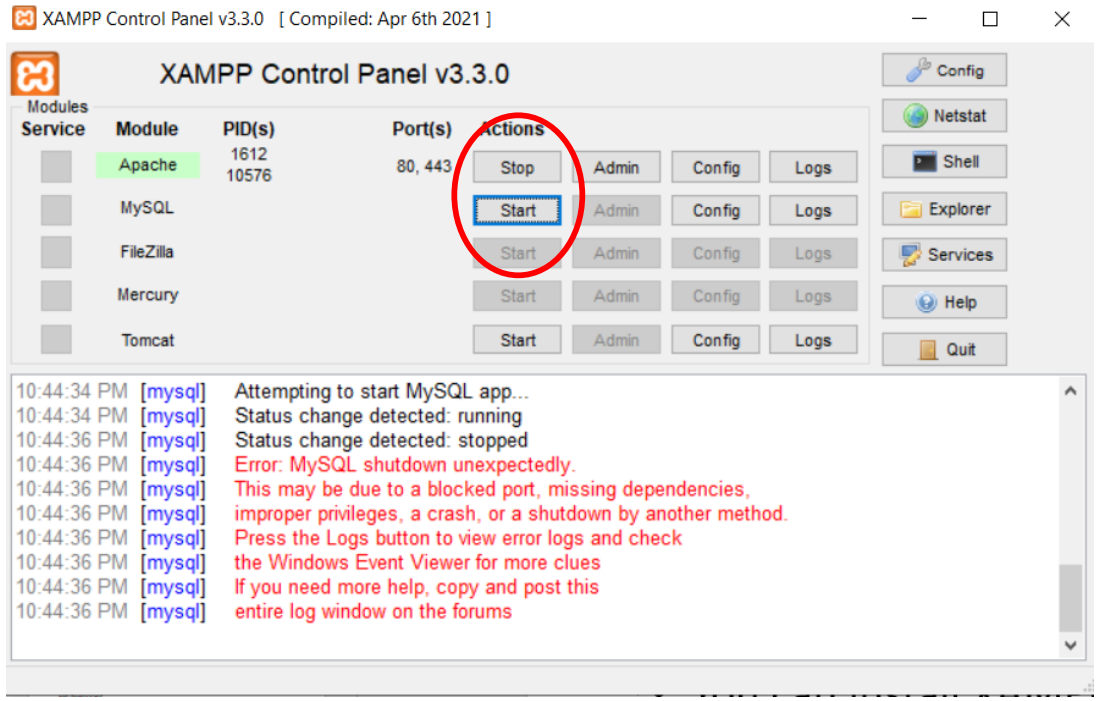
The response travels across the internet, back to your computer. Your browser receives the response and uses that information to create and render the **view** that you ultimately see after successfully logging in.

What is PHP

- PHP, which stands for Hypertext Preprocessor, is a popular server-side scripting language used for web development.
- PHP is open-source and free to use, making it accessible to developers of all levels.
- PHP is primarily used for creating dynamic web pages and web applications. Unlike static HTML pages, which remain the same for every user, PHP allows developers to generate dynamic content that can change based on user input, database queries, or other conditions. This makes PHP an essential tool for building interactive and personalized websites.
- PHP code can be embedded directly into HTML files, allowing developers to mix PHP and HTML code within the same document. This combination of PHP and HTML, known as PHP embedded code, enables developers to create dynamic web pages by executing PHP code and generating HTML output.
- PHP is also compatible with various databases, making it easy to interact with and manipulate data. It supports popular database management systems like MySQL, PostgreSQL, and Oracle, allowing developers to store and retrieve information from databases using PHP functions and SQL queries.

Installing PHP on Windows

- Download the PHP installer from the official PHP website https://www.php.net/manual/en/install.windows.php
- Install a web server like Apache You can install XAMPP to have Apache server and then use XAMPP control panel to start Apache server



PHP Basic Syntax and Variables

- **PHP Syntax** PHP code is typically embedded within HTML code, allowing you to mix dynamic server-side scripting with static client-side content. To indicate that a block of code contains PHP, you need to enclose it within **<?php and ?>**
- **Comments** are essential for documenting your code and making it more readable. In PHP, you can add single-line comments using // or multiline comments using /* */
- **Variables** are used to store and manipulate data in PHP. A variable is declared using the $ symbol followed by the variable name. PHP is a loosely typed language, meaning you don't need to specify the data type when declaring a variable
- **Variable Output** To display the value of a variable, you can use the echo or print statement
- **Constants** are similar to variables, but their values cannot be changed once defined. Constants are declared using the define() function
- **Concatenation** is the process of combining strings. In PHP, you can concatenate strings using the dot (.) operator

tryPHP.php > ...

```php
<?php
    // PHP code goes here
    $name = "Heba Hatem";
    $age = 25;
    $isLoggedIn = true;
    $numbers = [1, 2, 3, 4, 5];
    echo $name . "\n"; // Output: Heba Hatem
    define("PI", 3.14);
    echo PI. "\n"; // Output: 3.14
    $universityName = "ASPU";
    $departmentName = "Informatics";
    $student_in = $universityName . " " . $departmentName;
    echo $student_in; // Output: ASPU Informatics

?>
```

```
[Running] C:\xampp\php\php.exe "c:\Users\MA-Laptop\Desktop\مقررات الجامعة ومراجع عامة\Tishreen_University\First_Semester\web_design\lectures_practice\lecture_6\tryPHP.php"
Heba Hatem
3.14
ASPU Informatics
[Done] exited with code=0 in 0.214 seconds
```

PHP Associative arrays

- PHP associative arrays are arrays that use named keys instead of numeric indices to store and access values.
- Each key-value pair in an associative array is separated by => symbol.

```php
<?php
$person = array(
    "name" => "Sami",
    "age" => 22,
    "city" => "Damascus",
    "University" => "ASPU"
);

echo $person["name"]; // Output: Sami
?>
```

PHP Global Variables - Superglobals

- Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- The PHP superglobal variables are:
  - $GLOBALS
  - $_SERVER
  - $_REQUEST
  - **$_POST**
  - $_GET
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

PHP **$_POST** is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $_POST is also widely used to pass variables.

PHP **$_GET** is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$_GET can also collect data sent in the URL.

Processing Form Data with PHP

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form action="process.php" method="get">
        <label for="name">Name</label>
        <input type="text" id="name" name="name" required>
        <br><br>
        <label for="Email">Email</label>
        <input type="email" id="email" name="email" required>
        <br><br>
        <input type="submit" value="submit">

    </form>
</body>
</html>
```

testingPHP.html | process.php

process.php > ...

```php
<?php
    $name = $_POST['name'];
    $email = $_POST['email'];

    // Process the form data
    // ...

    // Redirect the user to a thank you page
    header("Location: thank-you.php");
    exit;
?>
```

To process the form data, you need to create a PHP script that will handle the form submission

Once the user submits the form, the form data is sent to the server for processing. In PHP, you can access the form data using the **$_POST superglobal variable.** The $_POST variable is an associative array where the keys are the names of the input fields defined in the HTML form.

After processing the form data, you can perform any necessary actions, such as storing the data in a database or sending an email. In this example, we simply redirect the user to a thank you page using the header() function.

Processing Form Data with PHP

testingPHP.html | process.php

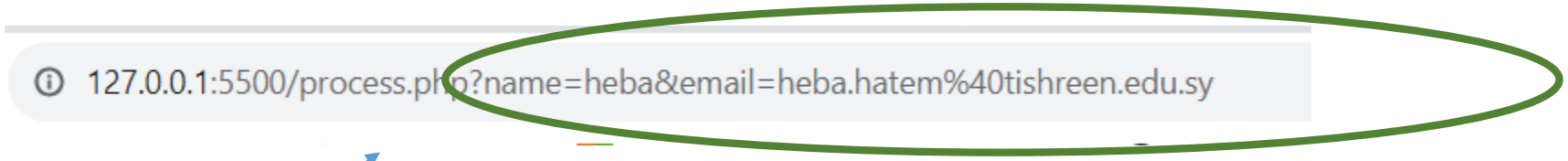
process.php > ...

```php
<?php
    $name = $_POST['name'];
    $email = $_POST['email'];

    // Process the form data
    // ...

    // Redirect the user to a thank you page
    header("Location: thank-you.php");
    exit;
?>
```

- The header() function sends a raw HTTP header to a client.
- When this line is executed in a PHP script, it sends an HTTP response header with the "Location" field set to "thank_you.php". This instructs the client (usually a web browser) to redirect to the specified URL.

- In practical terms, when a user's browser receives this header, it will automatically make a new request to the "thank_you.php" page, effectively redirecting the user to that page. This is commonly used after processing a form submission or completing a certain action to redirect the user to a different page for further instructions or confirmation.

Processing Form Data with PHP



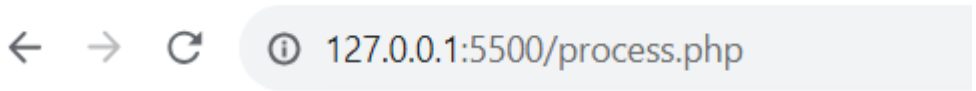
With get method the form data is appended to the URL, in name/value pairs

Name heba

Email heba.hatem@tishreen.edu.s

submit

With post method the submitted form data is not shown in the URL



HTML form – method attribute

- The method attribute specifies the HTTP method to be used when submitting the form data.
- The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").
- The default HTTP method when submitting form data is GET.
- Always use POST if the form data contains sensitive or personal information!

Post	GET
• Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL) • POST has no size limitations, and can be used to send large amounts of data. • Form submissions with POST cannot be bookmarked	• Appends the form data to the URL, in name/value pairs • NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!) • The length of a URL is limited (2048 characters) • Useful for form submissions where a user wants to bookmark the result

Server-Side Validation

- When it comes to form validation, it is crucial to perform validation on the server-side rather than relying solely on client-side validation.
- Client-side validation, which is performed using JavaScript, can be easily bypassed by malicious users or users with disabled JavaScript.
- Therefore, server-side validation acts as a safety net, ensuring that all submitted data is validated and processed correctly.
- To perform server-side validation in PHP, you can use a combination of built-in functions, regular expressions, and conditional statements.
- PHP provides various functions specifically designed for form validation,

Common Validation Techniques

Required Fields: Ensure that all required fields are filled out before allowing the form submission. You can check if a field is empty using the empty() function or by checking the length of the input string.

```php
<?
$name = $_POST['name'];
$email = $_POST['email'];

if (empty($name)) {
    // Name field is empty
    $errorMessage = "Name is required";
    echo $errorMessage;
} elseif (empty($email)) {
    // Email field is empty
    $errorMessage = "Email is required";
    echo $errorMessage;
} else {
    // All required fields are filled, proceed with form submission
    // ...
}
?>
```

Numeric Validation: Validate numeric input to ensure it is a valid number. You can use functions like is_numeric() or filter_var() with the FILTER_VALIDATE_INT or FILTER_VALIDATE_FLOAT filters.

```php
<?
$number = $_POST['number'];

if (!is_numeric($number)) {
    // Invalid number format
    echo "Please enter a valid number";
} else {
    // Valid number format, proceed with form submission
    // ...
}
?>
```

Common Validation Techniques

```php
<?
$username = $_POST['username'];

$minLength = 3;
$maxLength = 10;

if (strlen($username) < $minLength || strlen($username) > $maxLength) {
    // Invalid length
    echo "Username must be between $minLength and $maxLength characters long";
} else {
    // Valid length, proceed with form submission
    // ...
}
?>
```

Length Validation: Check if the length of a string falls within a specified range. You can use the strlen() function to get the length of the input and

```php
<?php
    $name = $_POST['name'];
    $email = $_POST['email'];

    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      // Invalid email format
      $errorMessage = "Invalid email format";
      echo $errorMessage;
    }
    ?>
```

Email Validation: In the code above, we use the filter_var() function with the FILTER_VALIDATE_EMAIL filter to validate the email address. If the email address is not in the correct format, we can display an error message to the user.

Common Validation Techniques

```php
<?
$password = $_POST['password'];

$minLength = 8;
$uppercaseRequired = true;
$lowercaseRequired = true;
$specialCharRequired = true;

$errors = [];

if (strlen($password) < $minLength) {
    $errors[] = "Password must be at least $minLength characters long";
}

if ($uppercaseRequired && !preg_match('/[A-Z]/', $password)) {
    $errors[] = "Password must contain at least one uppercase letter";
}

if ($lowercaseRequired && !preg_match('/[a-z]/', $password)) {
    $errors[] = "Password must contain at least one lowercase letter";
}

if ($specialCharRequired && !preg_match('/[!@#$%^&*]/', $password)) {
    $errors[] = "Password must contain at least one special character (!@#$%^&*)";
}

if (!empty($errors)) {
    foreach ($errors as $error) {
        echo $error . "<br>";
    }
} else {
    // Password is valid, proceed with form submission
    // ...
}

?>
```

In PHP, the preg_match() function is a built-in function used for pattern matching with regular expressions. It is used to search a string for a pattern and returns a boolean value indicating whether the pattern was found or not.

Password Validation: When dealing with password fields, it is essential to enforce certain criteria, such as minimum length, presence of uppercase and lowercase letters, and special characters. You can use a combination of string functions and regular expressions to validate passwords.

File Uploads

- In web development, file uploads are a common requirement when building applications that allow users to upload files to the server. PHP provides built-in functions and features to handle file uploads efficiently.
- When a file is uploaded, it is stored in a temporary location on the server. To access the uploaded file, PHP provides a superglobal variable called $_FILES. This variable is an associative array that contains information about the uploaded file, such as its name, type, size, and temporary location.
- To upload a file, you need to create an HTML form with the enctype attribute set to "multipart/form-data". This attribute tells the browser that the form contains binary data, such as files, that need to be uploaded.

upload_file.html | upload.php

upload_file.html > html

```
1   <html lang="en">
2   <head>
3       <title>Upload file example</title>
4   </head>
5   <body>
6
7       <form action="upload.php" method="POST" enctype="multipart/form-data">
8           <input type="file" name="file">
9           <input type="submit" value="Upload">
10        </form>
11
12  </body>
13  </html>
```

In the above example, the form has an input field of type "file" which allows the user to select a file from their local machine. The name attribute of the input field is important as it determines the key under which the file will be available in the $_FILES array.

Handling File Uploads

When the form is submitted, the file is sent to the server along with the other form data. In the PHP script that handles the form submission, you can access the uploaded file using the **$_FILES superglobal.**
The **$_FILES['file']** variable contains an array with several keys that provide information about the uploaded file. The most commonly used keys are:

- **name**: The original name of the file.
- **type**: The MIME type of the file.
- **size**: The size of the file in bytes.
- **tmp_name**: The temporary location of the uploaded file on the server.
- **error**: An error code if any error occurred during the upload process.

To move the uploaded file from the temporary location to a permanent location on the server, you can use the **move_uploaded_file()** function. This function takes two parameters: the temporary file path and the destination file path.

upload_file.html | upload.php

upload.php > ...

```php
<?
$file = $_FILES['file'];

// move the uploaded file to a permenent dir
$uploadedFilePath = 'uploads/' . $file['name'];
move_uploaded_file($file['tmp_name'], $uploadedFilePath);

//validate uploaded file
$allowedTypes = ['image/jpeg', 'image/png', 'application/pdf'];
if (!in_array($file['type'], $allowedTypes)) {
    // Invalid file type
}

// validate uploaded file size
$maxFileSize = 1048576; // 1MB
if ($file['size'] > $maxFileSize) {
    // File size exceeds the limit
}

// validate file extension
$allowedExtensions = ['jpg', 'png', 'pdf'];
$extension = pathinfo($file['name'], PATHINFO_EXTENSION);
if (!in_array($extension, $allowedExtensions)) {
    // Invalid file extension
}
?>
```

Validating File Uploads

To validate the file type, you can use the **$_FILES['file']['type']** variable, which contains the MIME type of the uploaded file. You can compare this value against a list of allowed MIME types to restrict the types of files that can be uploaded.

To validate the file size, you can use **the $_FILES['file']['size']** variable, which contains the size of the uploaded file in bytes. You can compare this value against a maximum file size limit to prevent users from uploading excessively large files.

To validate the file extension, you can use the **pathinfo()** function to extract the file extension from the uploaded file name. You can then compare the file extension against a list of allowed extensions.

upload_file.html | upload.php

upload.php > ...

```php
<?
$file = $_FILES['file'];

// move the uploaded file to a permenent dir
$uploadedFilePath = 'uploads/' . $file['name'];
move_uploaded_file($file['tmp_name'], $uploadedFilePath);

//validate uploaded file
$allowedTypes = ['image/jpeg', 'image/png', 'application/pdf'];
if (!in_array($file['type'], $allowedTypes)) {
    // Invalid file type
}

// validate uploaded file size
$maxFileSize = 1048576; // 1MB
if ($file['size'] > $maxFileSize) {
    // File size exceeds the limit
}

// validate file extension
$allowedExtensions = ['jpg', 'png', 'pdf'];
$extension = pathinfo($file['name'], PATHINFO_EXTENSION);
if (!in_array($extension, $allowedExtensions)) {
    // Invalid file extension
}
?>
```

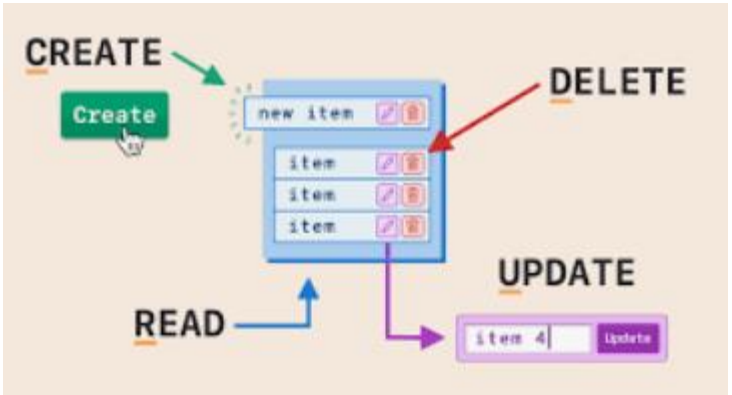
End of lecture 5

ASPU University
Faculty of Informatics Engineering
Fourth Year
Application development Course
2023 - 2024

جامعة الشام الخاصة
Al-Sham Private University

Building CRUD app using PHP

Lecture 6



Lecture overview

- CRUD
- Database in web application
- PHP MySQL database
- PHP different ways to interact with MySQL databases
- Open a connection to MySQL
- Close the connection
- Create a database
- Read from the database
- Update the database
- Delete from the database
- Prepared statements

1- CRUD Definition

CRUD stands for **Create**, **Read**, **Update**, and **Delete**. It is a set of basic operations that are commonly used in database systems to manage data.

- **Create**: This operation involves creating new records or entities in the database. It typically involves inserting new data into a table.
- **Read**: This operation involves retrieving or reading existing records or entities from the database. It typically involves querying the database to fetch specific data based on certain criteria.
- **Update**: This operation involves modifying or updating existing records or entities in the database. It typically involves changing the values of specific fields in a table.
- **Delete**: This operation involves removing or deleting existing records or entities from the database. It typically involves removing specific data from a table based on certain criteria.

2- Databases in web application development

- data is at the heart of every application. Whether it's a simple blog or a complex e-commerce platform.
- A database is a structured collection of data that allows efficient storage, retrieval, and management of information.
- In the context of web development, databases are used to store and retrieve data for dynamic websites and web applications.
- PHP, being a versatile and powerful programming language, provides various tools and libraries to interact with databases seamlessly.

3- PHP MySQL database

- PHP MySQL refers to the combination of the PHP programming language and the MySQL database management system
- MySQL Defenition
  - MySQL is a database system used on the web, runs on a server
  - MySQL is ideal for both small and large applications
  - MySQL is very fast, reliable, and easy to use
  - MySQL uses standard SQL
  - The data in a MySQL database are stored in tables.
  - PHP combined with MySQL are cross-platform
- PHP 5 and later can work with a MySQL database using:
  - **MySQLi extension** (the "i" stands for improved)
  - **PDO (PHP Data Objects)**
- The PHP MySQL combination is commonly used to build interactive and data-driven websites and web applications.

3-1- PHP different ways to interact with MySQL databases

Procedural MySQLi

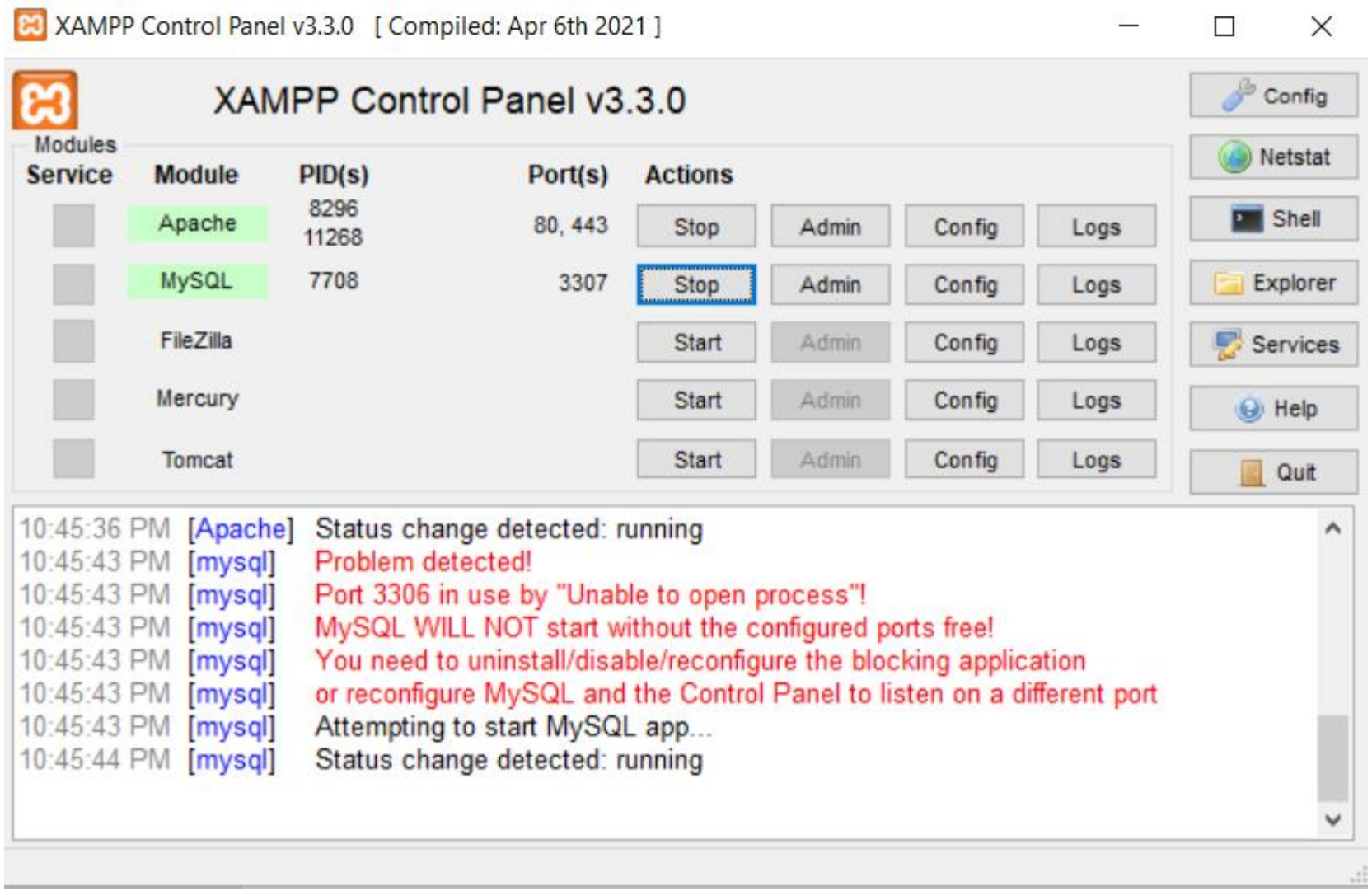
- is the oldest method, and it involves using a series of MySQL functions to interact with the database. This method can be simple and straightforward for small applications, but it can become difficult to manage as the application grows.

Object-oriented MySQLi

- is a newer method that involves creating objects that represent database connections, queries, and results. This method can be more organized and easier to manage than procedural MySQL, but it requires more code to set up.

PDO PHP Data Objects

- PHP extension that provides a consistent interface for working with databases, including MySQL. It supports both procedural and object-oriented programming styles, and it provides features like prepared statements. PDO can be a good choice for applications that need to work with multiple databases or that need to be easily portable to different database systems.



1- Start Apache server and MySQL server

2- Create your project in XAMPP\htdocs folder.

3-1-MySQLi vs PDO

MySQLi

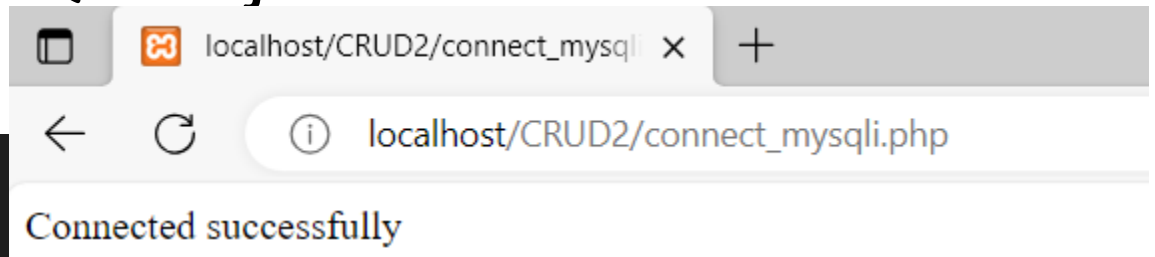
- MySQLi will only work with MySQL databases.
- object-oriented and also offers a procedural API.
- support **Prepared Statements**

Prepared Statements protect from SQL injection, and are very important for web application

PDO

- PDO will work on 12 different database systems(e.g. MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle)
- For future project development if you have to switch your project to use another database, **PDO makes the process easy.** You only have to change the connection string and a few queries.
- object-oriented
- support **Prepared Statements**

3-2- Open a Connection to MySQL
3-2-1- Connect to the server using MySQLi Object-Oriented



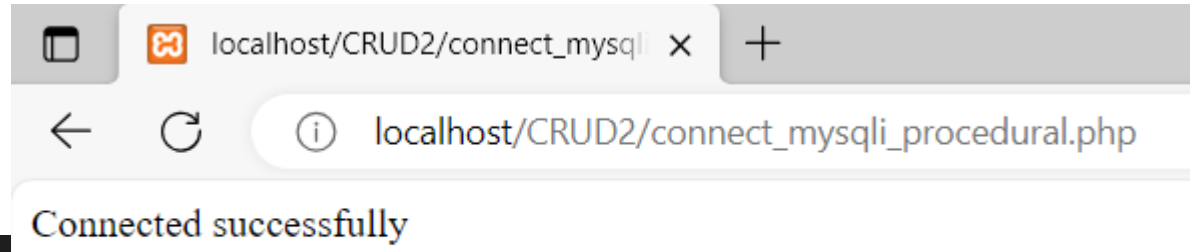
connect_mysqli.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

In PHP, the arrow symbol (->) is used as the object operator. It is used to access methods and properties of an object.

use the die() function to terminate the script and output an error message if the connection to the MySQL database fails.

3-2- Open a Connection to MySQL
3-2-2- Connect to the server using MySQLi Procedural



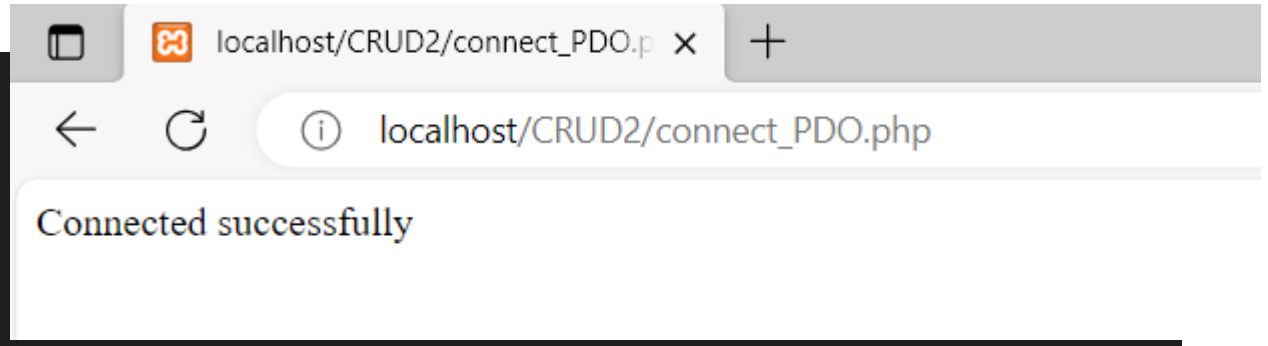
connect_mysqli_procedural.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

establishes a connection with the specified database.

This condition checks if the connection to the MySQL database was unsuccessful. If the connection fails (i.e., $conn evaluates to false), the code within the curly braces will be executed.

3-2- Open a Connection to MySQL
3-2-3- Connect to the server using PDO



connect_PDO.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

try {
  $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

3-2- Open a Connection to MySQL
3-2-3- Connect to the server using PDO

connect_PDO.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

try {
  $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

Note: In the PDO example above we have also specified a database (myDB). PDO require a valid database to connect to. If no database is specified, an exception is thrown.

Tip: A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

If an exception of type PDOException is thrown during the execution of the try block (i.e., if there is an error in establishing the connection), this block catches the exception and outputs an error message that includes the specific error message obtained from $e->getMessage()

3-2- Open a Connection to MySQL
3-2-3- Connect to the server using PDO

- **The setAttribute method is used to set attributes on a PDO instance. It allows you to configure various options and behaviors related to the PDO connection and its error handling.**
- **Syntax: $pdo->setAttribute($attribute, $value).**
- **Example: $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION) sets the error mode attribute to throw exceptions when errors occur, enabling more effective error handling.**
- **The first parameter specifies the attribute to be set, and the second parameter specifies the value of the attribute.**

connect_PDO.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

try {
  $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

3-2- Open a Connection to MySQL
3-2-3- Connect to the server using PDO

- **The getMessage method is used to retrieve the error message associated with an exception object. It is commonly used in exception handling to obtain a human-readable description of the error that occurred.**
- **Syntax: $exception->getMessage().**
- **Example: In the context of exception handling, $e->getMessage() retrieves the error message from the exception object $e.**
- **This method returns a string containing the error message that provides information about the specific error that occurred.**

connect_PDO.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

try {
  $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```

4- close the connection

- The connection will be closed automatically when the script ends. To close the connection before, use the following:

MySQLi Object-Oriented:

```
$conn->close();
```

MySQLi Procedural:

```
mysqli_close($conn);
```

PDO:

```
$conn = null;
```

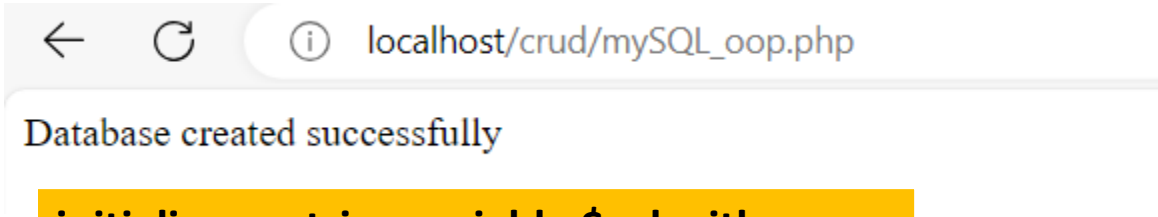
5- Create a MySQL Database

5- 1- Create a MySQL Database Using MySQLi object oriented

- The CREATE DATABASE statement is used to create a database in MySQL.

mySQL_oop.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```



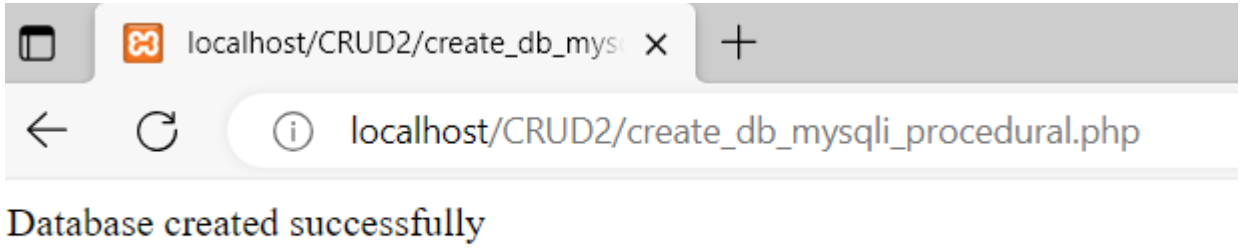
initializes a string variable $sql with an SQL query to create a new database named "myDB".

- **query method of the $conn object is used to execute the SQL query stored in the $sql variable.**
- **If the query is executed successfully, the condition ($conn->query($sql) === TRUE) evaluates to true, and the message "Database created successfully" is echoed to indicate that the database creation was successful.**
- **If the query execution fails, the else block is executed, and the message "Error creating database: " concatenated with the specific error message obtained from $conn->error is echoed. This provides information about the reason for the database creation failure.**

5- Create a MySQL Database
5- 2- Create a MySQL Database Using MySQLi procedural

create_db_mysqli_procedural.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

// Create database
$sql = "CREATE DATABASE myDB_procedural";
if (mysqli_query($conn, $sql)) {
  echo "Database created successfully";
} else {
  echo "Error creating database: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

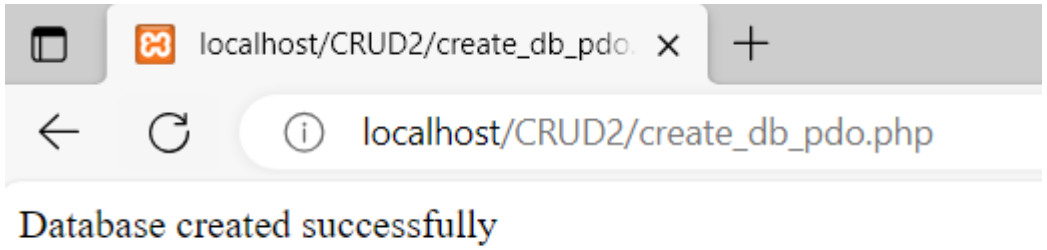


- **The mysqli_query function is used to execute the SQL query stored in the $sql variable using the connection object $conn to the MySQL server.**
- **If the query is executed successfully, the condition mysqli_query($conn, $sql) evaluates to true, and the message "Database created successfully" is echoed to indicate that the database creation was successful.**
- **If the query execution fails, the else block is executed, and the message "Error creating database: " concatenated with the specific error message obtained from mysqli_error($conn) is echoed. This provides information about the reason for the database creation failure.**

5- Create a MySQL Database
5- 3- Create a MySQL Database Using MySQLi PDO

create_db_pdo.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

try {
  $conn = new PDO("mysql:host=$servername", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "CREATE DATABASE myDBPDO";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Database created successfully<br>";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```



$conn: This is the PDO object representing the connection to the MySQL server.

- **exec($sql): This is the exec method called on the $conn object.**
- **It takes an SQL query as a parameter and executes the query on the database server. The $sql variable contains the SQL query to create a new database named "myDBPDO".**

5- Create a MySQL Database
5- 3- Create a MySQL Database Using MySQLi PDO

create_db_pdo.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";

try {
  $conn = new PDO("mysql:host=$servername", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "CREATE DATABASE myDBPDO";
  // use exec() because no results are returned
  $conn->exec($sql);
  echo "Database created successfully<br>";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

- **When exec is used to execute an SQL query, it returns the number of rows affected by the query, or false if the query fails.**
- **However, for SQL statements that do not return a result set, the returned value is not useful, so it is often not captured or used in the code.**

- **In this case, the exec method is used to execute the SQL query to create a new database without expecting a result set in return. If the query is successful, it will create the database "myDBPDO". If there is an error during execution, an exception will be thrown, and it will be caught and handled in the catch block.**

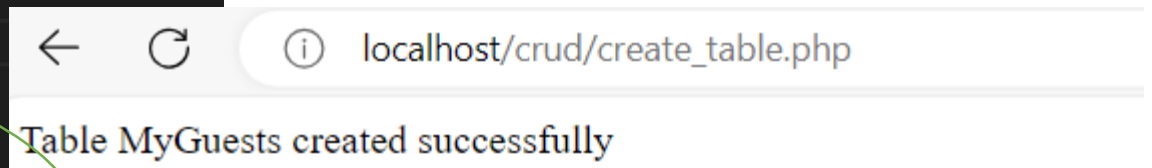
6- -Create a MySQL Table

6-1-Create a MySQL Table Using MySQLi object oriented

A database table has its own unique name and consists of columns and rows.

create_table.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

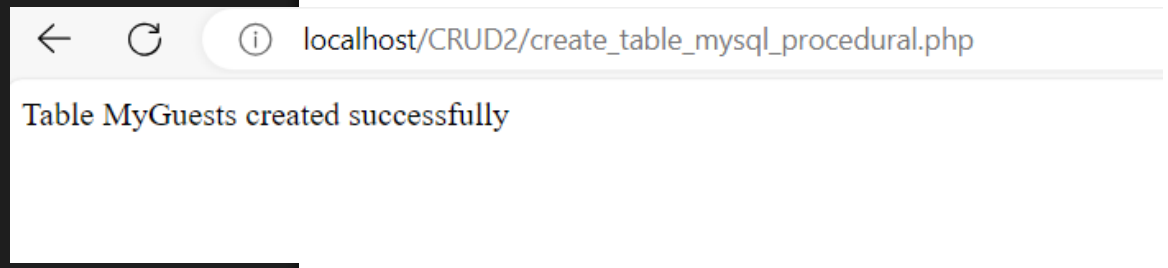


6- -Create a MySQL Table
6-2-Create a MySQL Table Using MySQLi PDO

```php
$conn->exec($sql);
echo "Table MyGuests created successfully";
} catch(PDOException $e) {
echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

create_table_mysql_procedural.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";
$dbname = "myDBPDO";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // sql to create table
  $sql = "CREATE TABLE MyGuests (
  id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
  firstname VARCHAR(30) NOT NULL,
  lastname VARCHAR(30) NOT NULL,
  email VARCHAR(50),
  reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
  )";

```



6-Create a MySQL Table
6-3-Create a MySQL Table Using Procedural

```php
// sql to create table
$sql = "CREATE TABLE MyGuests2 (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
  echo "Table MyGuests created successfully";
} else {
  echo "Error creating table: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

7- PHP MySQL Insert Data
Insert Data Into MySQL Using MySQLi and PDO

- After a database and a table have been created, we can start adding data in them.
- Here are some syntax rules to follow:

1. The SQL query must be quoted in PHP
2. String values inside the SQL query must be quoted
3. Numeric values must not be quoted
4. The word NULL must not be quoted

The *INSERT INTO* statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

If a column is AUTO_INCREMENT (like the "id" column) or TIMESTAMP with default update of current_timestamp (like the "reg_date" column), it is no need to be specified in the SQL query; MySQL will automatically add the value.

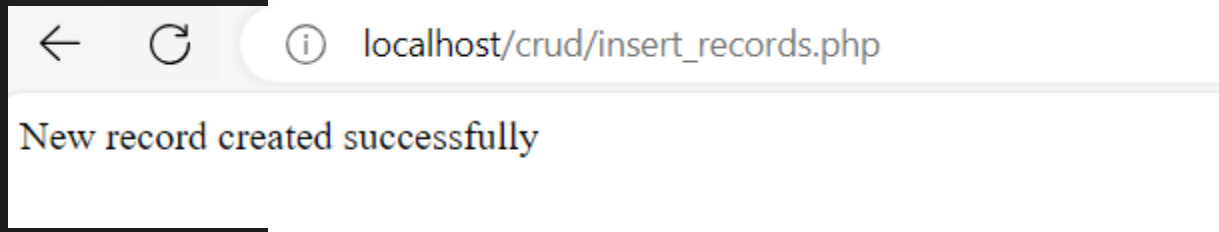
7-PHP MySQL Insert Data
7-1- Insert Data Into MySQL Using MySQLi object oriented

insert_records.php > ...

```php
<?php
$servername = "localhost";
$username = "root";
$password = "12345";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Ahmad', 'Ahmad', 'john@aspu.edu.sy')";

if ($conn->query($sql) === TRUE) {
  echo "New record created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
```

The following examples add a new record to the "MyGuests" table:

Inserted first record



8-PHP MySQL Get Last Inserted ID

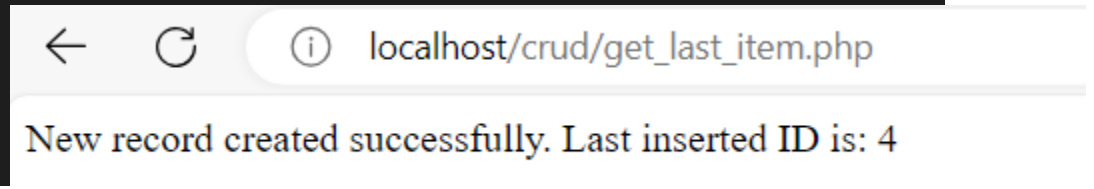
8-1-Get ID of The Last Inserted Record/ (MySQLi Object-oriented)

If we perform an INSERT or UPDATE on a table with an AUTO_INCREMENT field, we can get the ID of the last inserted/updated record immediately.

In the table "MyGuests", the "id" column is an AUTO_INCREMENT field:

Inserted second, third and fourth

```php
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Heba', 'Hatem', 'heba.hatem@tishreen.edu.sy')";
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mohammad', 'Mohammad', 'moh@gamil.com')";
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('sami', 'Mohammad', 'sam@gamil.com')";
if ($conn->query($sql) === TRUE) {
  $last_id = $conn->insert_id;
  echo "New record created successfully. Last inserted ID is: " . $last_id;
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

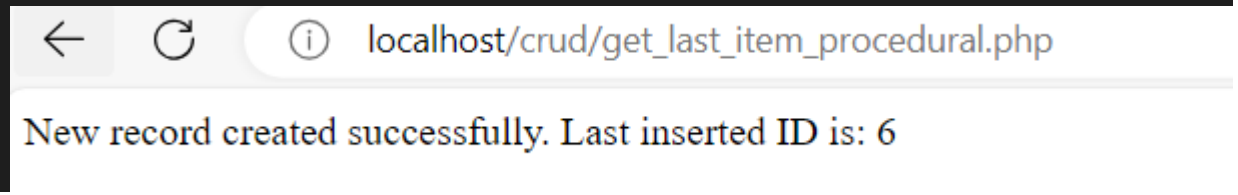


8-PHP MySQL Get Last Inserted ID

8-2-Get ID of The Last Inserted Record/ (MySQLi Procedural)

Inserted the fifth and the sixth

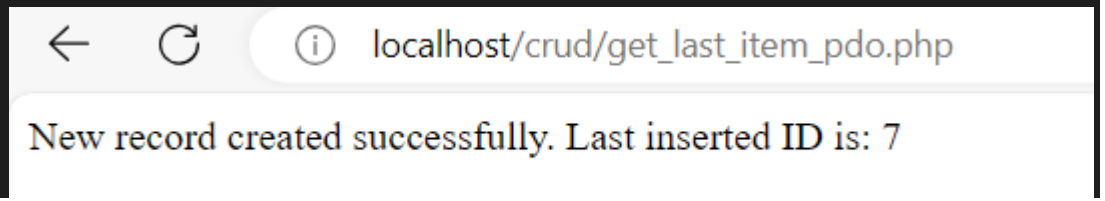
```php
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Maya', 'Mohammad', 'maya@tishreen.edu.sy')";
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mira', 'Mohammad', 'mira@tishreen.edu.sy')";

if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully. Last inserted ID is: " . $last_id;
  } else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
  }

mysqli_close($conn);
?>
```



8-PHP MySQL Get Last Inserted ID
8-3-Get ID of The Last Inserted Record/ (PDO)

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('John', 'Doe', 'john@example.com')";
  // use exec() because no results are returned
  $conn->exec($sql);
  $last_id = $conn->lastInsertId();
  echo "New record created successfully. Last inserted ID is: " . $last_id;
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

Inserted one more record

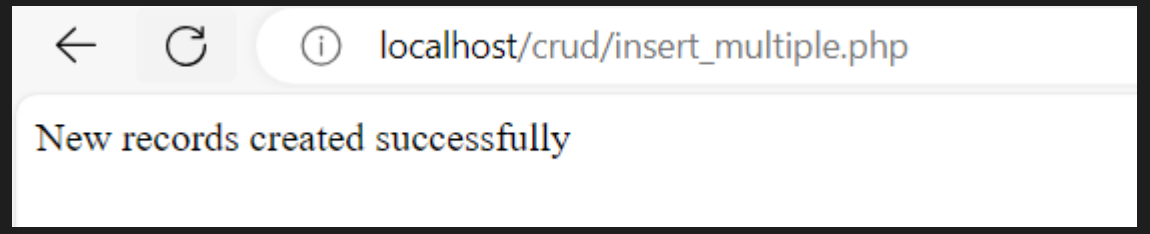


9-Insert Multiple Records Into MySQL

9-1- Using MySQLi Object-oriented

- Multiple SQL statements must be executed with the **multi_query() function.**
- The following examples add three new records to the "MyGuests" table:

```php
$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
  echo "New records created successfully";
} else {
  echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```



The multi_query function in PHP is used to execute multiple SQL queries in a single call to the database server.
It takes as a parameter : string containing multiple SQL queries separated by semicolons. These queries are then sent to the MySQL database server in a single call.

9-2-Insert Multiple Records Into MySQL Using MySQLi Procedural

```php
// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Mary', 'Moe', 'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('Julie', 'Dooley', 'julie@example.com')";

if (mysqli_multi_query($conn, $sql)) {
  echo "New records created successfully";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

mysqli_close($conn);
?>
```

9-3-Insert Multiple Records Into MySQL Using PDO

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  // begin the transaction
  $conn->beginTransaction();
  // our SQL statements
  $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('John', 'Doe', 'john@example.com')");
  $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('Mary', 'Moe', 'mary@example.com')");
  $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('Julie', 'Dooley', 'julie@example.com')");
  // commit the transaction
  $conn->commit();
  echo "New records created successfully";
} catch(PDOException $e) {
  // roll back the transaction if something failed
  $conn->rollback();
  echo "Error: " . $e->getMessage();
}
$conn = null;
?>
```

- the commit function is used in the context of database transactions.
- It is used to permanently save the changes made during a transaction to the database.

- When you use the commit function, you are indicating that you want to permanently save the changes made during the transaction to the database.
- This means that the changes become visible to other users and applications accessing the database.

. Once the commit function is called, the changes are made permanent.

9-3-Insert Multiple Records Into MySQL Using PDO

```php
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  // begin the transaction
  $conn->beginTransaction();
  // our SQL statements
  $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('John', 'Doe', 'john@example.com')");
  $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('Mary', 'Moe', 'mary@example.com')");
  $conn->exec("INSERT INTO MyGuests (firstname, lastname, email)
  VALUES ('Julie', 'Dooley', 'julie@example.com')");
  // commit the transaction
  $conn->commit();
  echo "New records created successfully";
} catch(PDOException $e) {
  // roll back the transaction if something failed
  $conn->rollback();
  echo "Error: " . $e->getMessage();
}
$conn = null;
?>
```

A database transaction is a sequence of one or more SQL operations that are treated as a single unit of work.
These operations are either all completed successfully (committed) or all rolled back (undone) if an error occurs.

- **It's important to handle errors and exceptions that may occur during the transaction.**
- **If an error occurs, you can choose to roll back the transaction using the rollback function to undo the changes made so far.**

10- Update Data In a MySQL Table

The UPDATE statement is used to update existing records in a table:

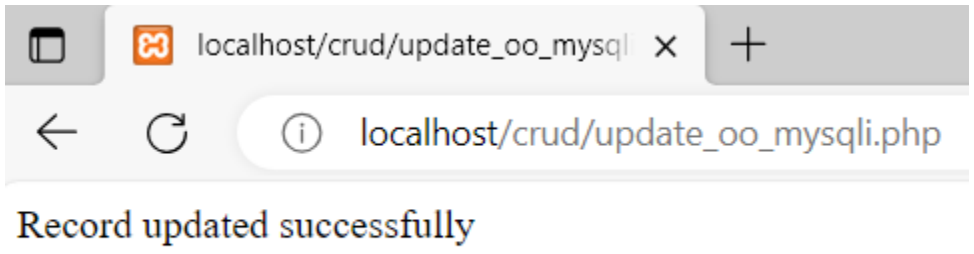
```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

- **The WHERE clause specifies which record or records that should be updated.**
- **If you omit the WHERE clause, all records will be updated!**

10-1- Update Data In a MySQL Table using MySQLi Object-oriented and MySQLi Procedural

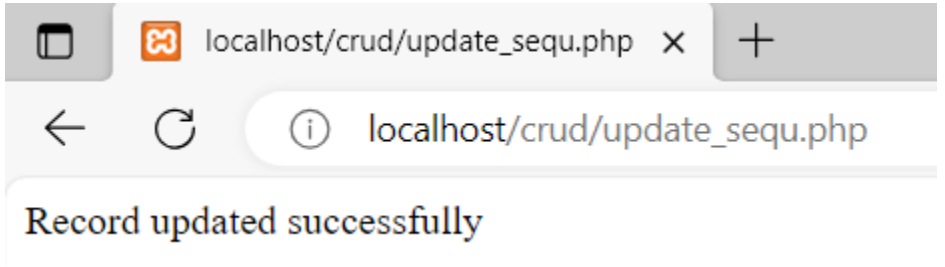
MySQLi Object-oriented

```php
$sql = "UPDATE MyGuests SET lastname='Mohammad' WHERE id=2";

if ($conn->query($sql) === TRUE) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```



MySQLi Procedural

```php
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id=2";

if (mysqli_query($conn, $sql)) {
  echo "Record updated successfully";
} else {
  echo "Error updating record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```



10-2- Update Data In a MySQL Table using PDO

```php
    $sql = "UPDATE MyGuests SET lastname='Hatem' WHERE id=1";

    // Prepare statement
    $stmt = $conn->prepare($sql);

    // execute the query
    $stmt->execute();

    // echo a message to say the UPDATE succeeded
    echo $stmt->rowCount() . " records UPDATED successfully";
  } catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
  }

  $conn = null;
  ?>
```

- The **prepare()** function is a method provided by database connection objects (such as $conn) in PHP. It is used to create a prepared statement from an SQL query.
- Once the statement has been prepared, it can be executed multiple times with different parameter values without needing to re-prepare the query each time.
- This provides security benefits by helping to prevent SQL injection attacks and can also improve performance by allowing the database to optimize query execution.

- **The rowCount() function is used to retrieve the number of rows affected by the most recent SQL statement executed on a database handle.**
- **This function is commonly used in PHP with the PDO (PHP Data Objects) and MySQLi extensions to obtain the number of rows affected by an INSERT, UPDATE, or DELETE operation.**

11- PHP MySQL Delete Data

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

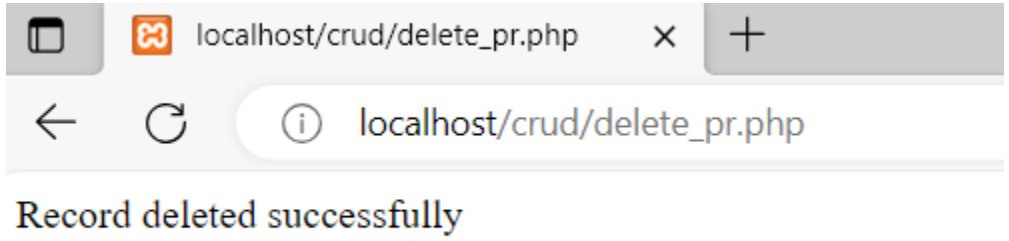
11-1- PHP MySQL Delete Data using MySQLi object oriented and Procedural

MySQLi Object-oriented

```php
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

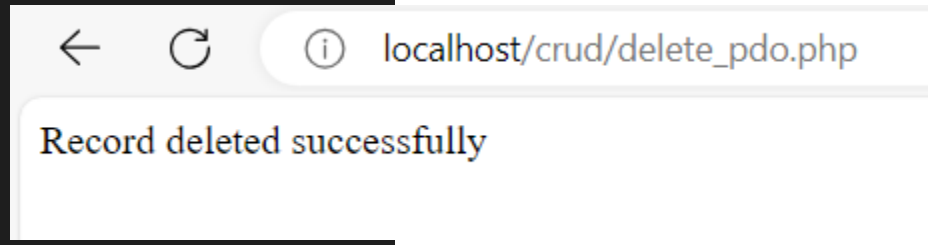
MySQLi Procedural

```php
// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=4";

if (mysqli_query($conn, $sql)) {
  echo "Record deleted successfully";
} else {
  echo "Error deleting record: " . mysqli_error($conn);
}

mysqli_close($conn);
?>
```



11-2- PHP MySQL Delete Data using PDO

```php
    // sql to delete a record
    $sql = "DELETE FROM MyGuests WHERE id=1";

    // use exec() because no results are returned
    $conn->exec($sql);
    echo "Record deleted successfully";
  } catch(PDOException $e) {
    echo $sql . "<br>" . $e->getMessage();
  }

$conn = null;
?>
```

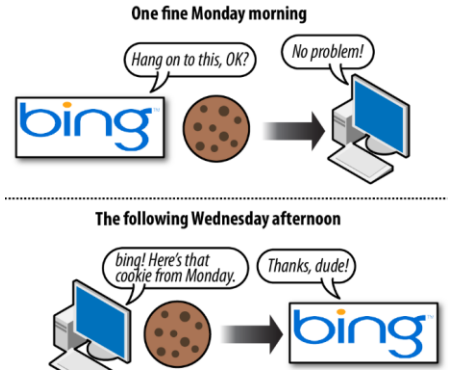


End of lecture 6

ASPU University
Faculty of Informatics Engineering
fourth Year
Internet technology Course
2023 - 2024

جامعة الشام الخاصة
Al-Sham Private University

Sessions and Cookies

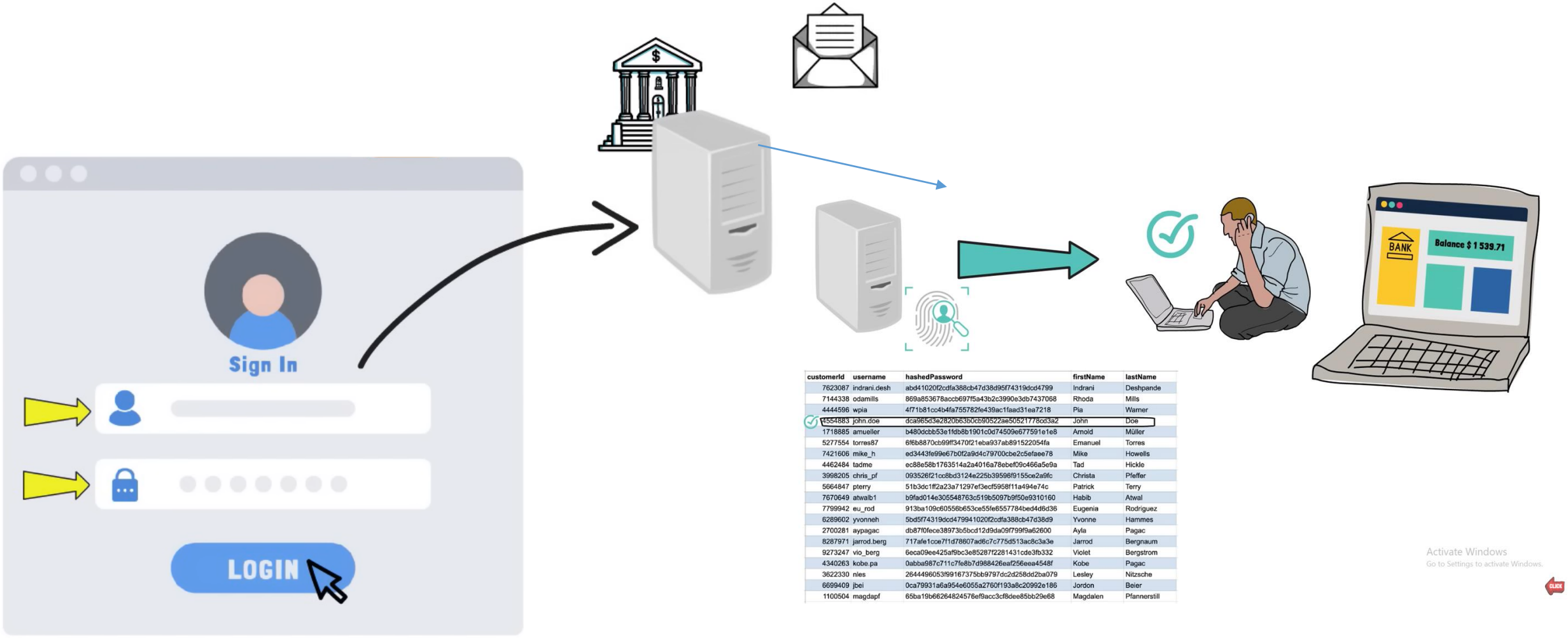


Lecture 7

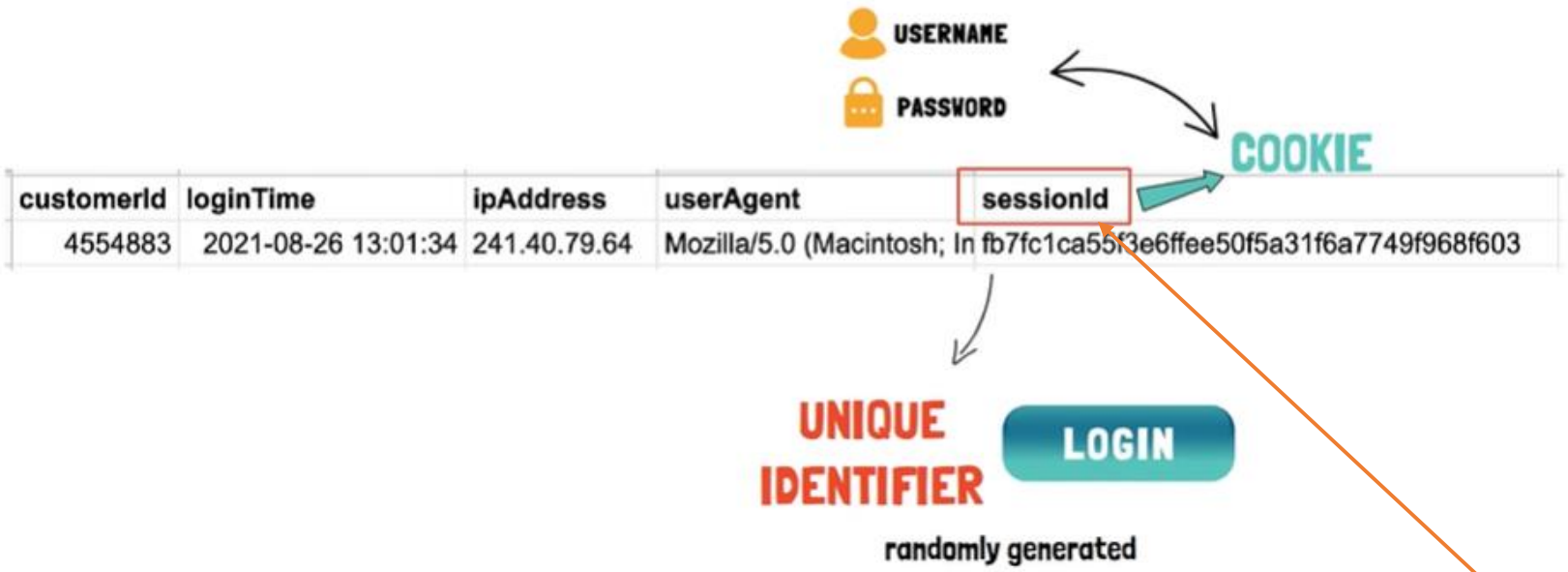
Lecture Overview

- Understand cookies and sessions from example
- What is cookie
- Create , modify and delete a cookie using PHP
- What is PHP session
- Start a PHP session
- Get PHP Session Variable Values
- Modify a PHP Session Variable
- Destroy a PHP Session

1- Understanding cookies and sessions from example



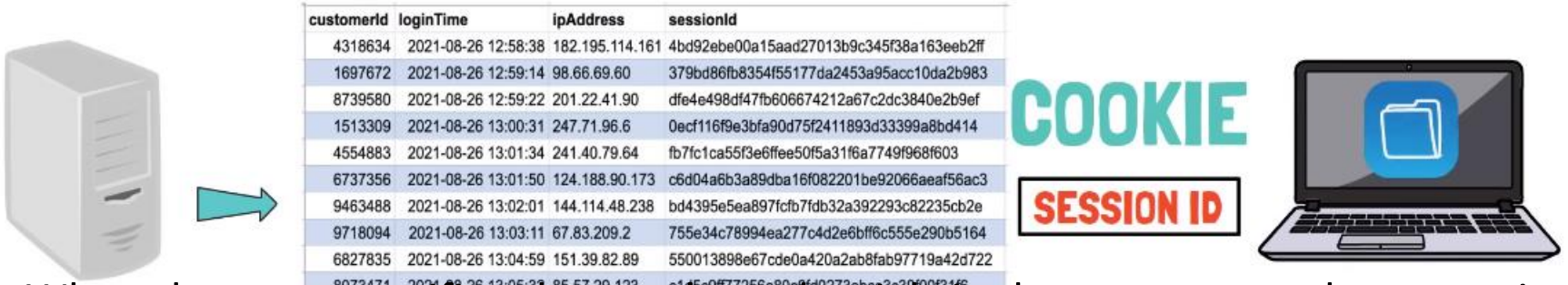
customerId	username	hashedPassword	firstName	lastName
7623087	indrani.desh	abd41020f2cdfa388cb47d38d95f74319dcd4799	Indrani	Deshpande
7144338	odamills	869a853678accb697f5a43b2c3990e3db7437068	Rhoda	Mills
4444596	wpia	4f71b81cc4b4fa755782fe439ac1faad31ea7218	Pia	Warner
4554883	john.doe	dca965d3e2820b63b0cb90522ae50521778cd3a2	John	Doe
1718885	amueller	b480dcbb53e1fdb8b1901c0d74509e677591e1e8	Arnold	Müller
5277554	torres87	6f6b8870cb99ff3470f21eba937ab891522054fa	Emanuel	Torres
7421606	mike_h	ed3443fe99e67b0f2a9d4c79700cbe2c5efaee78	Mike	Howells
4462484	tadme	ec88e58b1763514a2a4016a78ebef09c466a5e9a	Tad	Hickle
3998205	chris_pf	093526f21cc8bd3124e225b39596f9155ce2a9fc	Christa	Pfeffer
5664847	pterry	51b3dc1ff2a23a71297ef3ecf5958f11a494e74c	Patrick	Terry
7670649	atwalb1	b9fad014e305548763c519b5097b9f50e9310160	Habib	Atwal
7799942	eu_rod	913ba109c60556b653ce55fe6557784bed4d6d36	Eugenia	Rodriguez
6289602	yvonneh	5bd5f74319dcd479941020f2cdfa388cb47d38d9	Yvonne	Hammes
2700281	aypagac	db87f0fece38973b5bcd12d9da09f799f9a62600	Ayla	Pagac
8287971	jarrod.berg	717afe1cce7f1d78607ad6c7c775d513ac8c3a3e	Jarrod	Bergnaum
9273247	vio_berg	6eca09ee425af9bc3e85287f2281431cde3fb332	Violet	Bergstrom
4340263	kobe.pa	0abba987c711c7fe8b7d988426eaf256eea4548f	Kobe	Pagac
3622330	nles	2644496053f99167375bb9797dc2d258dd2ba079	Lesley	Nitzsche
6699409	jbei	0ca79931a6a954e6055a2760f193a8c20992e186	Jordon	Beier
1100504	magdapf	65ba19b66264824576ef9acc3cf8dee85bb29e68	Magdalen	Pfannerstill



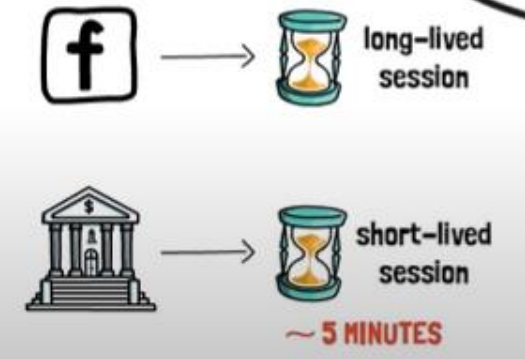
1.After entering the username and password, the server verifies the credentials and creates an entry in the database with the session ID.

2. server gives you a session ID in the form of a cookie

- **The session ID acts as a unique identifier for the logging session and is randomly generated**



- When the server verifies the user's credentials it also creates another entry in the database.
- So the server will store the session information on a database but the user will only have the session ID in the form of a cookie which is sorted in the computer user's file system
- Next time the user requests another page , the browser will automatically send a cookie which the server then with check to see if it is valid and then the username and password are no longer required
- When you log out, your login session is invalidated and the cookie containing the session ID is deleted.
- Session expires if you are inactive for some time, prompting you to provide your username and password again for security.
- Some websites have long-lived sessions, while others, like banks, have short-lived sessions.
- Cookies are sent using HTTP headers in the messages exchanged between the browser and the server.



2- What is a cookie

- A cookie is often used to identify a user.
- A cookie is a small file that the server embeds on the user's computer.
- Each time the same computer requests a page with a browser, it will send the cookie too.
- With PHP, you can both create and retrieve cookie values.

2-1-Create Cookies With PHP

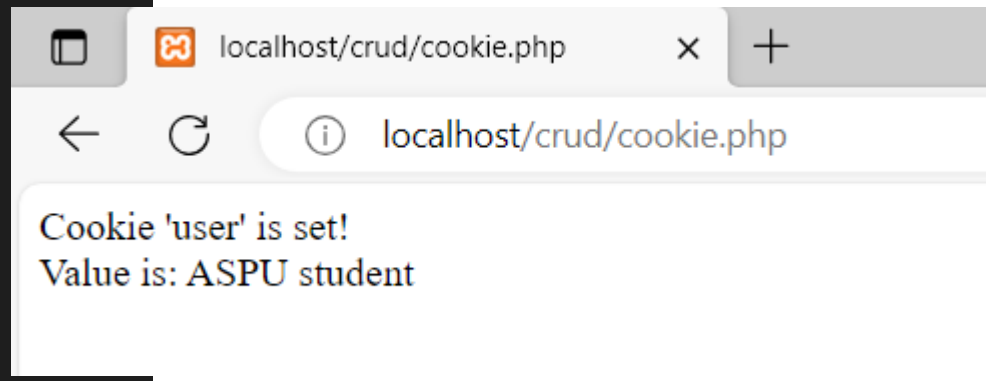
- A cookie is created with the setcookie() function.
- Syntax setcookie(*name, value, expire, path)*

The cookie will expire after 30 days (86400 * 30)

The setcookie() function must appear BEFORE the <html> tag.

cookie.php > html > body

```php
<?php
$cookie_name = "user";
$cookie_value = "ASPU student";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```



2-1-Create Cookies With PHP

cookie.php > html > body

```php
<?php
$cookie_name = "user";
$cookie_value = "ASPU student";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
// 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

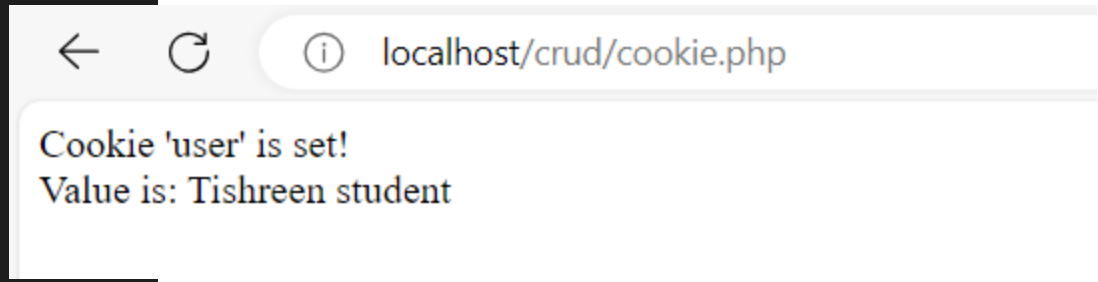
- **The last parameter "/" in the setcookie function call represents the path on the server where the cookie will be available. When a cookie is set with a specific path, it will only be sent to pages within that path or its subdirectories.**
- **In this case, setting the path to "/" means that the cookie will be available to the entire domain. It will be sent to all pages on the domain, regardless of their directory structure. This is often used to make the cookie accessible across the entire website.**
- **If you were to set the path to a specific directory, the cookie would only be sent to pages within that directory and its subdirectories. For example, setting the path to "/example" would restrict the cookie to pages within the "/example" directory and its subdirectories.**
- **By setting the path to "/", you are making the cookie available site-wide.**

2-2-Modify a Cookie Value

- To modify a cookie, just set (again) the cookie using the **setcookie()** function:

cookie.php

```php
<?php
$cookie_name = "user";
$cookie_value = "Tishreen student";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
 // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

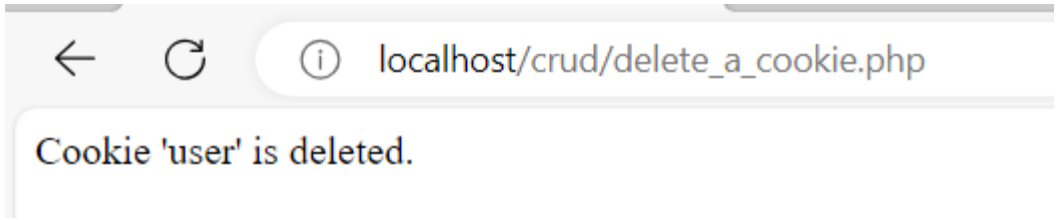


2-3-Delete a Cookie

- To delete a cookie, use the **setcookie()** function with an expiration date in the past:

delete_a_cookie.php > html

```php
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

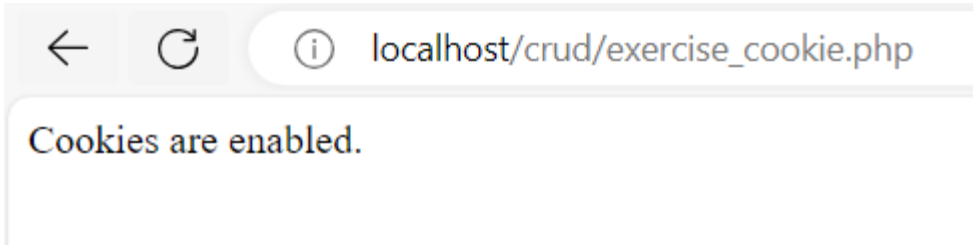


2-4-Check if Cookies are Enabled

- The following example creates a small script that checks whether cookies are enabled.
- First, try to create a test cookie with the setcookie() function, then count the $_COOKIE array variable:

exercise_cookie.php > html

```php
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
  echo "Cookies are enabled.";
} else {
  echo "Cookies are disabled.";
}
?>

</body>
</html>
```



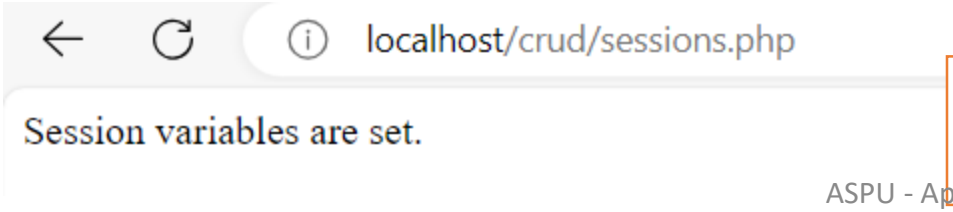
3-What is a PHP Session

- A session is a way to store information (in variables) to be used across multiple pages.
- Unlike a cookie, the information is not stored on the users computer.
- When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc).
- By default, session variables last until the user closes the browser.
- So; Session variables hold information about one single user, and are available to all pages in one application.

3-1-Start a PHP Session

sessions.php > html > body

```php
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```



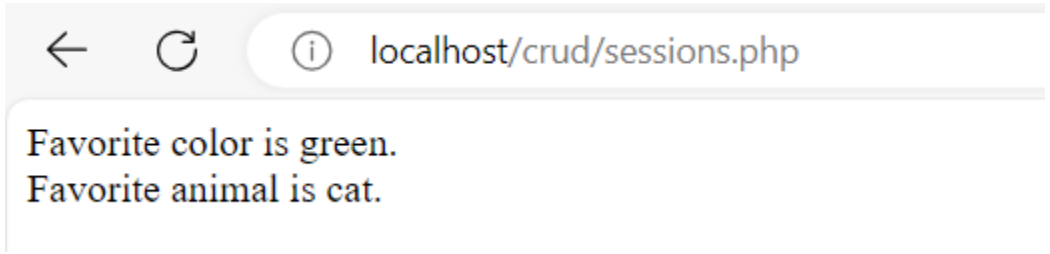
- A session is started with the **session_start()** function.
- Session variables are set with the PHP global variable: $_SESSION.
- In this example, we start a new PHP session and set some session variables:

- **The session_start() function must be the very first thing in your document. Before any HTML tags.**

3-2-Get PHP Session Variable Values

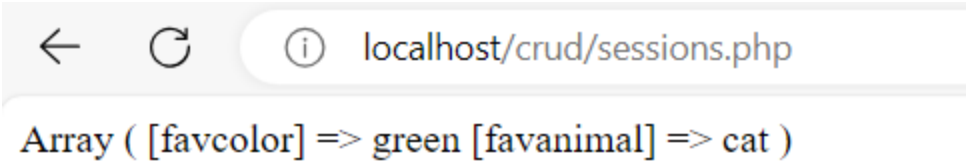
```php
14    // Echo session variables that were set on previous page
15    echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
16    echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
17
```

This line helps accessing the session information that were set



3-3-Get PHP Session Variable Values

```
print_r($_SESSION);
```



Another way to show all the session variable values for a user session is

3-4-Modify a PHP Session Variable

```
14    // to change a session variable, just overwrite it
15    $_SESSION["favcolor"] = "yellow";
16    $_SESSION["favanimal"] = "dog";
17    print_r($_SESSION);
```

To change a session variable, just overwrite it

3-5-Destroy a PHP Session

- **To remove all global session variables and destroy the session, use session_unset() and session_destroy():**

destroy_session.php > html

```php
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

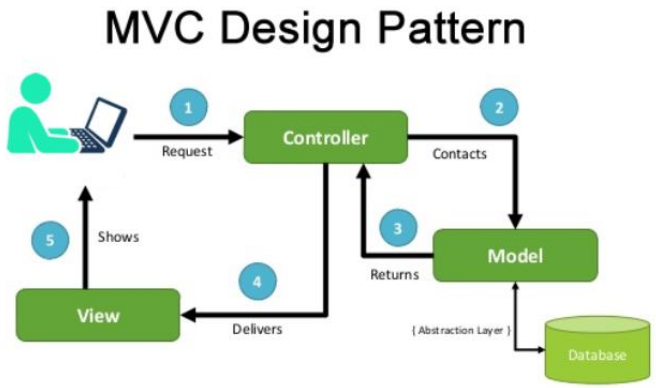
End of Lecture 7

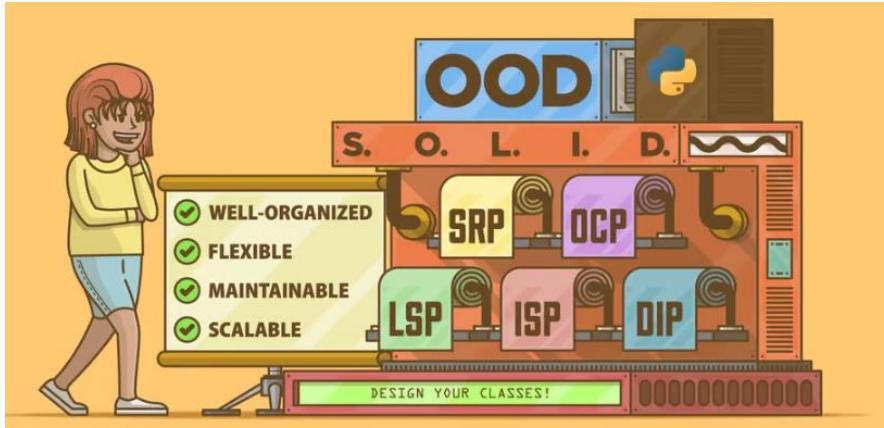
ASPU University
Faculty of Informatics Engineering
fourth Year
Internet technology Course
2023 - 2024

جامعة الشام الخاصة
Al-Sham Private University

MVC Design Pattern

Lecture 8





What is a design pattern?

A practical proven solution to a reoccurring design problem.

Designed solutions that are often used by experts.

They are not theoretical proposals for academic interests → USED IN INDUSTRIAL SOFTWARE

Think of them as RECIPES but don’t hesitate to try cooking your self.

Design patterns help to create a design vocabulary that every good developer should speak fluently.

Design Patterns Objectives

Extend Your Knowledge in OOAD (Object-Oriented Analysis and Design).

Learning How to apply design patterns to address design issues.

Gain a foundation to design and implement complex applications.

Learn design principles to make your software more REUSABLE, FLEXIBLE, and MAINTAINABLE.

IDENTIFY THE CODE SMELLS

- God Class, God Method, and Feature Envy.

Design Patterns Objectives / Identify The Code Smells

God Class

- is a class that has too many responsibilities and is overly complex. It violates the Single Responsibility Principle and tends to become a difficult-to-maintain class.

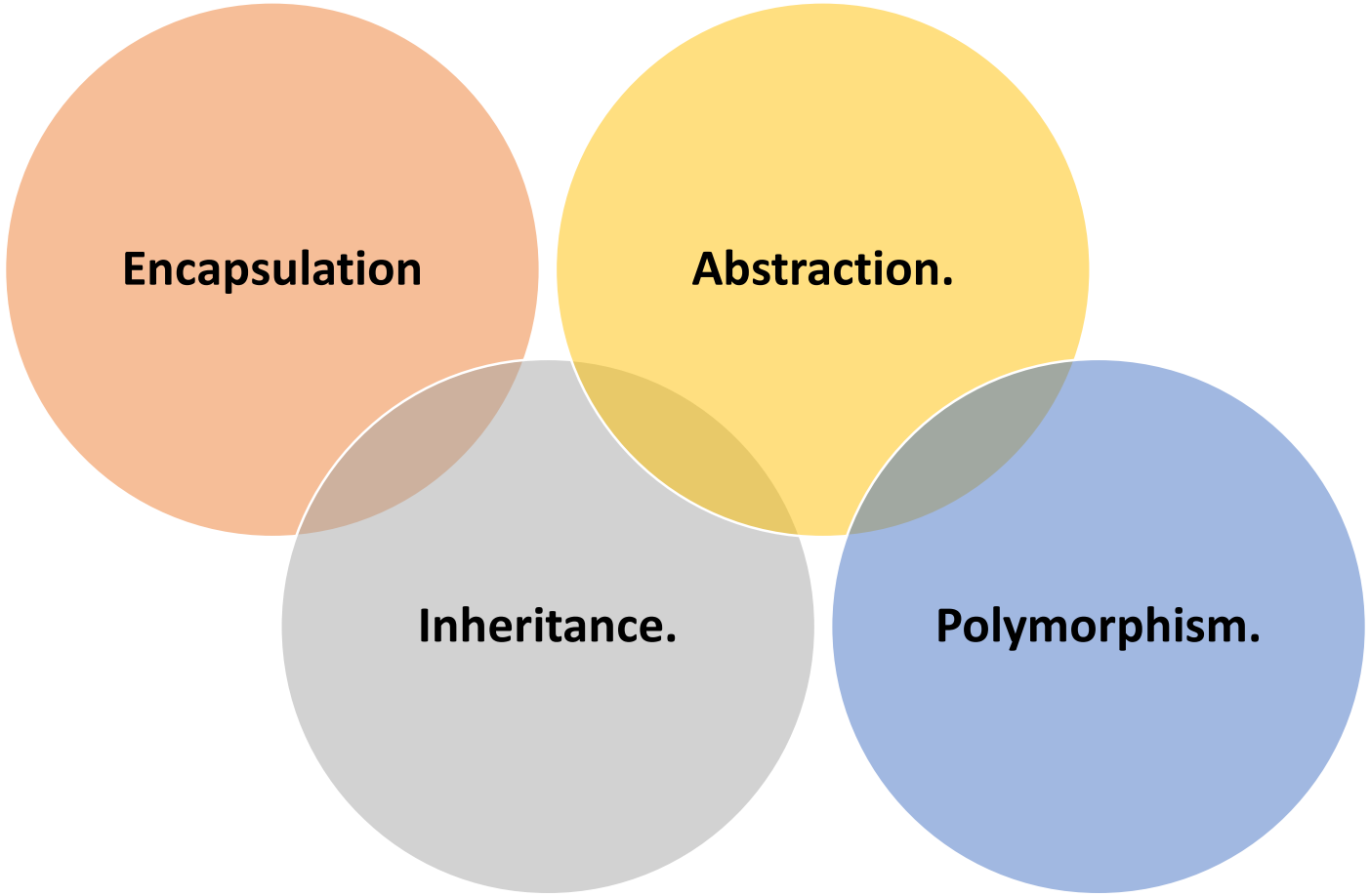
God Method

- is a method that does too much work, has too many lines of code, and handles too many responsibilities. It violates the Single Responsibility Principle at the method level

Feature Envy

- occurs when one class uses methods and properties of another class excessively. It often indicates that the responsibilities of the other class should be moved to the class that is using its features.

OOP-principles



Symptoms of a bad design

- **Rigidity**: It is hard to change because every change affects too many other parts of the system.
- **Fragility:** When you make a change, unexpected parts of the system break.
- **Immobility:** It is hard to reuse in another application.

Design principles – SOLID principles

- **Single Responsibility Principle:** A class should have only one reason to change
- **Open Close Principle:** Software entities like classes, modules and functions should be **open for extension** but **closed for modifications**
- **Liskov Substitution Principle (LSP**): It was introduced by Barbara Liskov and states that objects of a superclass should be replaceable with objects of its subclasses without affecting the correctness of the program. In other words, if S is a subtype of T, then objects of type T may be replaced with objects of type S without altering any of the desirable properties of the program. This principle ensures that inheritance is used correctly and that subclasses do not violate the expected behavior of the superclass.

Design principles – SOLID principles

- **Interface Segregation Principle:** Clients should not be forced to depend upon interfaces that they don't use.
- **Dependency Inversion Principle:** High-level modules should not depend on low-level modules. Both should depend on abstractions.

Design Patterns Categories

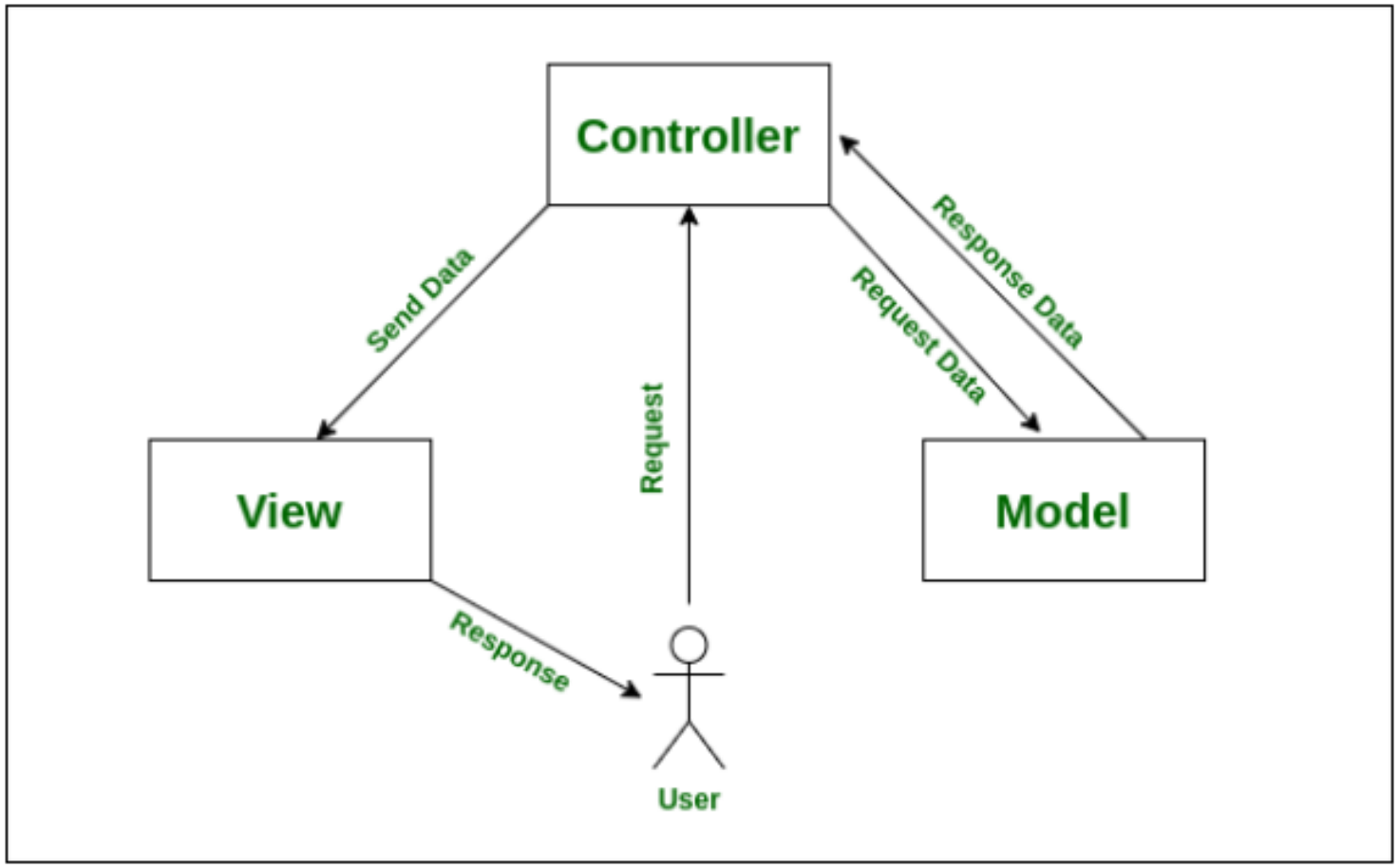
1. **Creational**: these patterns are focused on object creation mechanisms. They deal with object creation in a way that is more flexible and suitable to the situation at hand.
    - These patterns abstract the instantiation process, making the system independent of how its objects are created, composed, and represented.
    - Examples of creational design patterns include ***Singleton, Factory Method, Abstract Factory, Builder, Prototype, and Object Pool.***

Design Patterns Categories

2. **Structural**: are focused on the composition of classes or objects to form larger structures.
    - They help in identifying simple ways to realize relationships between entities.
    - Examples of structural design patterns include Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy and MVC

3. **Behavioral**: are focused on communication between objects, defining how objects interact and fulfill a given task.
    - They facilitate the assignment of responsibilities between objects and help them communicate in a loosely coupled manner.
    - Examples of behavioral design patterns include Observer, Strategy, Command, Iterator, State, Template Method, and Chain of Responsibility.

MVC - Model View Controller



MVC Implementation in native PHP
Model and Classes

- The Model is responsible for the data, and the logic of an application or the business logic.
- Model's responsibilities include; data store, retrieve, delete, and update. Generally, it includes database operations and the implementation of the operations that invoke external web services or APIs.
- The Model class doesn't need a presentation.
- In a proper implementation of the MVC pattern, we only expose entity classes(objects) by the Model, and these objects cannot encapsulate any business logic.
- Their sole purpose in an entity class is to keep data. Depending on the implementation, an Entity object can be in XML or JSON format.

MVC Implementation in native PHP
View

- The View (presentation layer)is responsible for formatting the data received from the Model.
- The data can come in different formats from the Model, including, XML structures, JSON, HTML,…
- The *view* (presents the model's data to the user.
- The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.

MVC Implementation in native PHP Controller

- The Controller receives the request, parses it, and then it initializes and invokes the Model.
- The Controller, then, takes the model response and sends it to the View layer.
- It’s practically the bridge between the Model and the View.
- in other words: the *controller* exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events.
- In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a notification mechanism, the result of this action is then automatically reflected in the view.

Example1 – login system example in MVC – the model

UserModel.php > UserModel > getUser

```php
<?php
class UserModel {
    public function getUser($username, $password) {
        // Replace this with your actual user authentication logic (e.g., database query)
        $validUsernames = array("user1", "user2");
        $validPasswords = array("password1", "password2");

        $index = array_search($username, $validUsernames);
        if ($index !== false && $validPasswords[$index] === $password) {
            return true;
        } else {
            return false;
        }
    }
}
?>
```

In PHP, the *array_search* function returns the key of the element if it is found in the array, and false otherwise.

Example1 – login system example in MVC – The view

loginView.php > html > body

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
</head>
<body>
    <h2>Login</h2>
    <form action="loginController.php" method="post">
        <input type="text" name="username" placeholder="Username" required>
        <input type="password" name="password" placeholder="Password" required>
        <button type="submit">Login</button>
    </form>
</body>
</html>

```

Example1 – login system example in MVC - the controller

loginController.php > ...

```php
<?php
require_once 'UserModel.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];

    $model = new UserModel();
    $isValidUser = $model->getUser($username, $password);

    if ($isValidUser) {
        echo "Login successful";
        // Add code to redirect to the user dashboard or home pag
    } else {
        echo "Invalid username or password";
    }
}
?>
```

- require_once statement is used to include and evaluate a specified file during the execution of a script.
- The "once" part ensures that the file is only included once, and if it has already been included, it will not be included again.
- This helps prevent issues related to redeclaring functions, redefining classes, and so on.

Example1 – login system example in MVC - the controller

loginController.php > ...

```php
<?php
require_once 'UserModel.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];

    $model = new UserModel();
    $isValidUser = $model->getUser($username, $password);

    if ($isValidUser) {
        echo "Login successful";
        // Add code to redirect to the user dashboard or ho
    } else {
        echo "Invalid username or password";
    }
}
?>
```

- **is used in PHP to check if the current request was made using the HTTP POST method.**
- **In the context of web development, when an HTML form is submitted using the POST method, the form data is sent in the body of the HTTP request.**
- **By using $_SERVER["REQUEST_METHOD"], you can check if the current request is of type POST.**

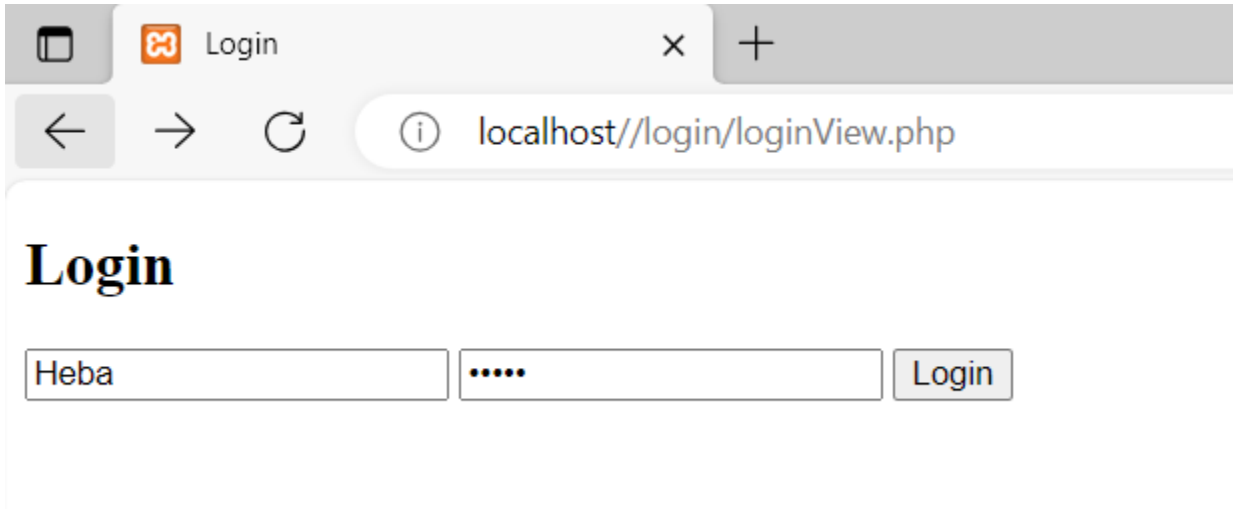
Example1 – login system example in MVC - the controller

loginController.php > ...

```php
<?php
require_once 'UserModel.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $username = $_POST["username"];
    $password = $_POST["password"];

    $model = new UserModel();
    $isValidUser = $model->getUser($username, $passw

    if ($isValidUser) {
        echo "Login successful";
        // Add code to redirect to the user dashboar
    } else {
        echo "Invalid username or password";
    }
}
?>
```

- $_SERVER superglobal in PHP is an array that contains information about the server and the current request.
- It includes details such as headers, paths, and script locations.
- When a PHP script is executed on a web server, the $_SERVER superglobal provides access to this server and request information.
- It can be used to obtain details about the current request, server environment, and other useful information related to the execution of the PHP script.

login system example in MVC



Example 2 – Calculator - The model

E: > XAMPP > htdocs > calc > CalculatorModel.php > CalculatorModel > calculate

```php
<?php
class CalculatorModel {
    public function calculate($num1, $num2, $operator) {
        switch ($operator) {
            case "add":
                return $num1 + $num2;
            case "subtract":
                return $num1 - $num2;
            case "multiply":
                return $num1 * $num2;
            case "divide":
                if ($num2 == 0) {
                    return "Cannot divide by zero";
                } else {
                    return $num1 / $num2;
                }
            default:
                return "Invalid operator";
        }
    }
}
?>
```

Example 2 – Calculator – the view

calculatorView.php

calculatorView.php > html > body > form > input

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Calculator</title>
</head>
<body>
    <h2>Simple Calculator</h2>
    <form action="calculatorController.php" method="post">
        <input type="number" name="num1" placeholder="Enter first number" required>
        <select name="operator">
            <option value="add">+</option>
            <option value="subtract">-</option>
            <option value="multiply">*</option>
            <option value="divide">/</option>
        </select>
        <input type="number" name="num2" placeholder="Enter second number" required>
        <button type="submit">Calculate</button>
    </form>
</body>
</html>
```

Example 2 – Calculator – The controller

E: > XAMPP > htdocs > calc > calculatorController.php > ...

```php
<?php
require_once 'CalculatorModel.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $num1 = $_POST["num1"];
    $num2 = $_POST["num2"];
    $operator = $_POST["operator"];

    $model = new CalculatorModel();
    $result = $model->calculate($num1, $num2, $operator);

    include 'calculatorView.php';
}
?>
```

In the context of the MVC design pattern, the require_once 'CalculatorModel.php'; statement is used in the controller file to include the model file ('CalculatorModel.php'), which contains the definition of the CalculatorModel class and its methods. This allows the controller to create an instance of the model and use its methods to perform calculations.

Example 2 – the controller

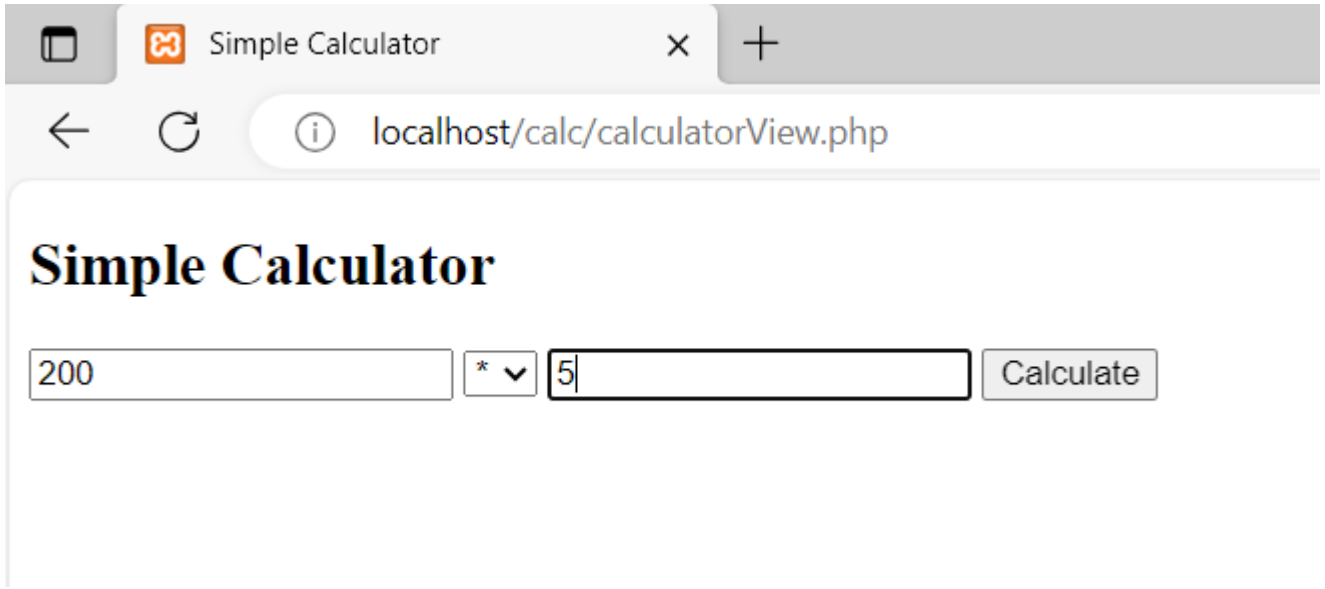
E: > XAMPP > htdocs > calc > calculatorController.php > ...

```php
<?php
require_once 'CalculatorModel.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $num1 = $_POST["num1"];
    $num2 = $_POST["num2"];
    $operator = $_POST["operator"];

    $model = new CalculatorModel();
    $result = $model->calculate($num1, $num2, $operator);

    include 'calculatorView.php';
}
?>
```

The include 'calculatorView.php'; statement in PHP is used to include and execute the content of the specified file (in this case, 'calculatorView.php') in the current script. The include statement is used for including files, and it differs from require in that it generates a warning if the file cannot be included rather than stopping the script execution with a fatal error.

When the include statement is used, if the specified file cannot be included (e.g., if the file does not exist or cannot be accessed), it will result in a warning, but the script will continue executing. If the file is successfully included, its content becomes part of the current script, and any HTML, PHP, or other code defined in the included file will be executed and displayed as part of the output.

In the context of the MVC design pattern, the include 'calculatorView.php'; statement is used in the controller file to include the view file ('calculatorView.php'), which contains the HTML code for the calculator interface. This allows the controller to generate and display the calculator interface when the corresponding action is triggered.


Simple Calculator
localhost/calc/calculatorView.php
Simple Calculator
200
*
5
Calculate

End of lecture 8