Assignment

1) There are 4 OOP concepts.

1. Polymorphism
2. Inheritance
3. Encapsulation
4. Abstraction

Polymorphism

Ability to assign different meaning or usage to something in different contexts.

Ability of an object to take many forms.

Ex: Animal sounds. Different animals make different sounds.

Inheritance

It is a mechanism in which one object acquires all the properties and behaviors from parent object.

Here child class inherit the attributes and methods define in the parent class.

Inheritance promotes code reusability.

Ex: Taxonomy of animals that shows their ancestry. (Origin of dogs from Gray Wolf)

Encapsulation

Encapsulation is the process of combining data and code into a single unit. Here, encapsulation hides the details of the implementation of an object.

Ex: The medical capsule is the most common example. This capsule combines several types of medications and stores them in a single capsule.

Abstraction

Abstraction is the process of removing characteristics and behaviors from something in order to reduce the set of essential characteristics and behaviors that is needed for the particular system.

Ex: ATM Machine; We can all perform operations on the ATM machine such as cash withdrawal, money transfer, retrieving mini-statements, and so on, but we don't know the internal details of the ATM.

2) The SOLID design principles useful to create flexible, maintainable, reusable and flexible software designs.

This implies,

S – Single Responsibility Principle

O – Open – Closed Principle

L – Liskov Substitution Principle

I – Interface Segregation Principle

D – Dependency Inversion Principle

Applying SOLID design principles throughout development will result in more maintainable, expandable, testable, and reusable systems.

The SOLID principle supports to reduce tight coupling. Tight coupling describes a group of classes that are highly dependent on one another and should be avoided in your code. The inverse of tight coupling is loose coupling, and your code is considered good if it has loosely coupled classes. Loosely coupled classes help to reduce changes in your code, making it more reusable, maintainable, flexible, and stable.

So, the SOLID principles' goal is to reduce dependencies so that engineers can change one area of software without affecting others. They're also meant to make designs simpler to understand, manage, and extend. Finally, applying these design principles helps software engineers avoid problems and create adaptive, efficient, and agile software.

3) RESTful API – Architectural style for API(Application Program Interface) which is uses HTTP request to access and use data. This data cam be used to read, update, create, and delete actions involving resources. It can also be used to GET, PUT, POST, and DELETE different data types.

A RESTful API is based on representational state transfer (REST), an architectural style and approach to communications that is frequently used in the development of web services. It is also known as a RESTful web service or REST API.

Commands are used by a RESTful API to access resources. A resource representation is the status of a resource at any timestamp. A RESTful API makes use of existing HTTP techniques, such as:

GET - to retrieve a resource.

PUT - used to update or change the state of resources, which can be objects, files, or blocks.

POST -resource can be created.

DELETE – delete the resource.

4)

Numerous factors affect how well web apps run. You may speed up your website in a variety of ways using a different tools and methods. You must apply methods in your web application's client-side and server-side code.

Techniques used to improve the speed and performance of the webpage.

1. Optimizing image size

Graphics are widely used on many websites. The performance of your website will be slowed down if your images are not compressed or if you choose an excessive resolution.

Compressing photos utilizing programs like ImageOptim, JPEGmini, or Kraken is the best technique to minimize the size of the image without sacrificing its quality. The process could take some time, but it's worthwhile. Using the HTML responsive images secret> and size> attributes, which modify image size according on user display parameters, is another technique to decrease the picture size.

2. Reduce number of JS and CSS files

When visitors to your website wish to access specific files, a lot of HTTP requests are generated if your website has a lot of JavaScript and CSS files. These requests cause the website to load more slowly because each visitor's browser handles them separately. JavaScript and CSS files should be kept to a minimum to speed up your website. Try to combine all CSS files and JavaScript into a single file. The total number of HTTP requests will decline as a result. There are many tools available to fast minify HTML, CSS, and JavaScript files.

3. Reduce the usage of web fonts

The use of web fonts in website design has grown significantly. Unfortunately, using web fonts slows down the speed at which pages are rendered. Web fonts increase the number of HTTP queries to outside resources.

4. Reduce redirects

Redirections on websites result in more HTTP requests, which reduces performance. We suggest minimizing them or doing away with them altogether. By performing a site scan, you should first find all redirects on your page. Then you must determine whether they are necessary and just keep the essential ones.

5)

Modern web development faces numerous challenges, one of which is security, which is both critical and frequently overlooked. While techniques like threat analysis are increasingly acknowledged as essential to any major development, there are some basic practices that every developer can and should do as a safety precaution.

When you assess the security of your application, you should be aware of these three key concepts:

Threats are situations that could endanger your application. Consider them as external processes that your application needs to protect itself from.

Vulnerabilities are openings for attackers to take advantage of in your application. They may be caused by bugs or flawed designs in both your code and the dependencies on it. Infrastructure-level flaws can also exist, such as network problems or insecure protocols.

Risks are the possible harm that could befall your application if an attacker takes advantage of a vulnerability. Risks can be thought of as the point where threats and vulnerabilities meet.

Understanding these ideas is essential to safeguarding your applications against hackers.