Internet:

The Internet is a vast network that connects computers all over the world. Through the Internet, people can share information and communicate from anywhere with an Internet connection.

URIs and URLs:

URI is a string of characters used to identify a resource on the internet either by location or by name, or both.

URIs (Uniform Resource Identifiers) identify resources on the Internet. URIs that start with http:// are called URLs (Uniform Resource Locators). Common URLs refer to files, directories or server-side code that performs tasks such as database lookups, Internet searches and business-application processing. If you know the URL of a publicly available resource anywhere on the web, you can enter that URL into a web browser's address field and the browser can access that resource.

Parts of a URL

A URL contains information that directs a browser to the resource that the user wishes to access. Web servers make such resources available to web clients. Popular web servers include Apache's HTTP Server and Microsoft's Internet Information Services (IIS).

Let's examine the components of the URL. The text http:// indicates that the HyperText Transfer Protocol (HTTP) should be used to obtain the resource. Next in the URL is the server's fully qualified hostname (for exam-ple, www.deitel.com)—the name of the web-server computer on which the resource re-sides. This computer is referred to as the host, because it houses and maintains resources.

The hostname www.deitel.com is translated into an IP (Internet Protocol) address—a numerical value that uniquely identifies the server on the Internet. An Internet Domain Name System (DNS) server maintains a database of hostnames and their corresponding IP addresses and performs the translations automatically.

Web Browser:

A web browser (commonly referred to as a browser) is application software for accessing the World Wide Web. When a user requests a web page from a particular website, the web browser retrieves the necessary content from a web server and then displays the page on the user's device.

Application:

An application program is a computer program designed to carry out a specific task other than one relating to the operation of the computer itself, typically to be used by end-users. Word processors, media players, and accounting software are examples

Internet Application:

An Internet application is a client/server application that uses standard Internet protocols for connecting the client to the server.

Web Application: Web application is a piece of software that can be accessed by the browser. A Browser is an application that is used to browse the internet. Web application needs authentication. The web application uses a combination of server-side scripts and client-side scripts to present information. It requires a server to manage requests from the users.

SDK:

A **software development kit** (**SDK**) is a collection of software development tools in one installable package. They facilitate the creation of applications by having a compiler, debugger and perhaps a software framework. They are normally specific to a hardware platform and operating system combination.[1][2][3] To create applications with advanced functionalities such as advertisements,[4] push notifications,[5] etc; most application software developers use specific software development kits.[6]

Web Services:

-Web services are services spread over the web.

-Enables the communication between application over the web and provides

a standart protocol or format for communication.

-Using webservices two different applications designed in different languages

can talk to each other share information and exchange data

-Platform independent.

Platform:

A platform is a group of technologies that are used as a base upon which other applications, processes or technologies are developed.

In personal computing, a platform is the basic hardware (computer) and software (operating system) on which software applications can be run. This environment constitutes the basic foundation upon which any application or software is supported and/or developed.

Framework:

A framework is a structure that you can build software on. It serves as a foundation, so you're not starting entirely from scratch. Frameworks are typically associated with a specific programming language and are suited to different types of tasks.

IDE:

An IDE, or Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program.

IDEs increase programmer productivity by combining common activities of writing software into a single application: editing source code, building executables, and debugging.

An integrated development environment (IDE)

is software for building applications that combines

common developer tools into a single graphical user interface (GUI).

An IDE typically consists of:

Source code EDITOR:

- syntax highlighting

- language specific auto-completion

- checking for bugs as code is being written.

AUTOMTION:

simple, repeatable tasks as

creating a local build

compiling, packaging, running automated tests.

DEBUGGER:

testing programs

display the location of a bug in the original code.

API:

An application programming interface, or API, enables companies to open up their applications' data and functionality to external third-party developers, business partners, and internal departments within their companies. This allows services and products to communicate with each other and leverage each other's data and functionality through a documented interface. Developers don't need to know how an API is implemented; they simply use the interface to communicate with other products and services. API use has surged over the past decade, to the degree that many of the most popular web applications today would not be possible without APIs.

Microservices:

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

The microservice architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

```
Micro Service is an architecture that
allows the developers to develop and
deploy services independently. Each service
running has its own process and this
achieves the lightweight model to
support business applications.
```

Module:

A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program. An enterprise-level software application may contain several different modules, and each module serves unique and separate business operations.

Modules make a programmer's job easy by allowing the programmer to focus on

only one area of the functionality of the software application. Modules are typically incorporated into the program (software) through interfaces.

Internet app architecture, Client-Server architecture, Distributed architecture, p2p architecture:

http://www.cs.sjsu.edu/~pearce/modules/patterns/distArch/index.htm

https://www.tutorialspoint.com/software_architecture_design/distributed_architecture.htm

https://www.techopedia.com/definition/454/peer-to-peer-architecture-p2p-architecture

https://www.techopedia.com/definition/454/peer-to-peer-architecture-p2p-architecture

Development Architecture: 1,2,3 and tier arch

Applications, advantages challenges and issues of each arch:

https://www.softwaretestingmaterial.com/software-architecture

http://clientserverarch.blogspot.com/2013/03/advantages-and-disadvantages-of.html

Deployment models:

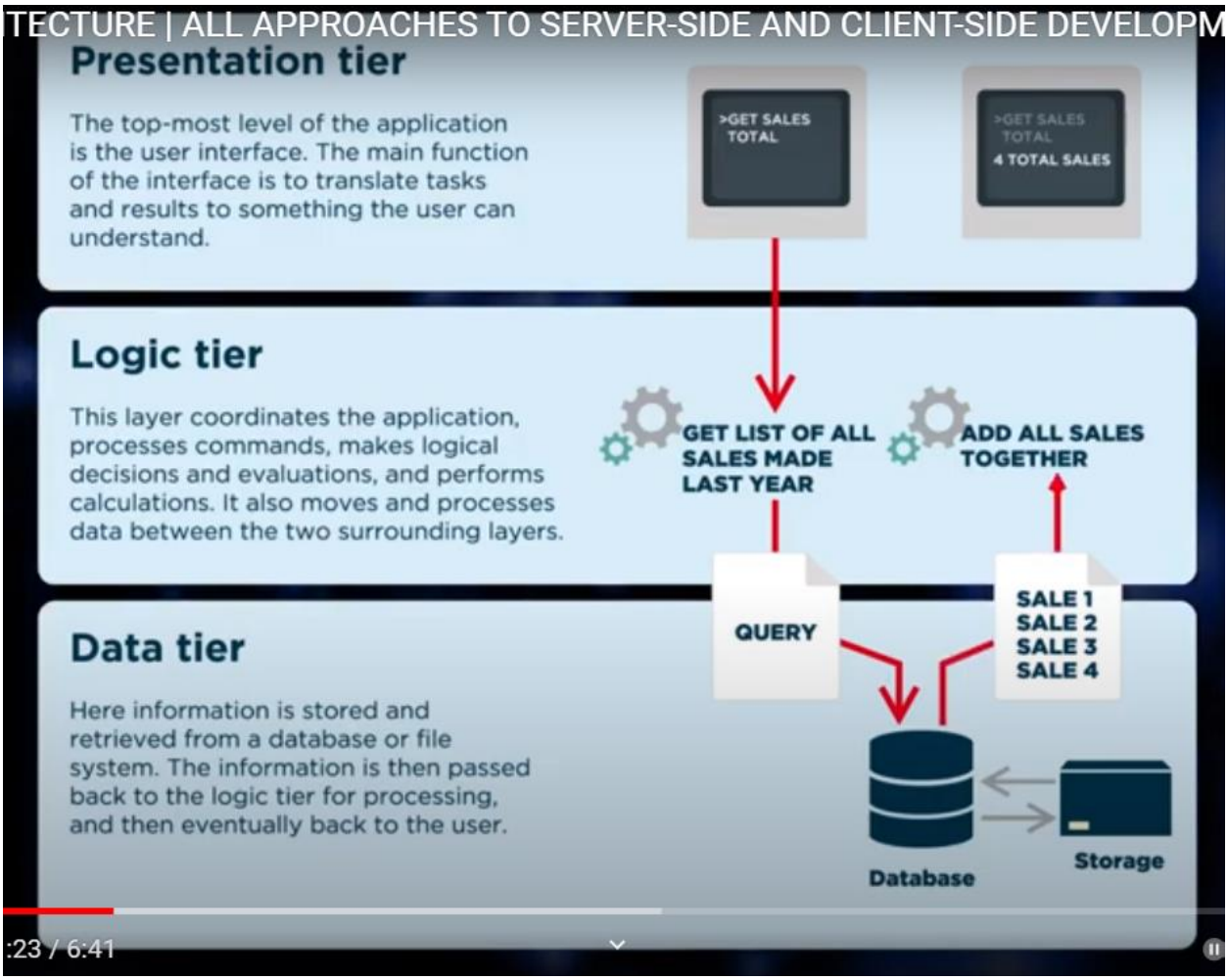
https://www.digitalocean.com/community/tutorials/5-common-server-setups-for-your-web-application

Other links:

https://www.intellectsoft.net/blog/web-application-architecture/

https://datarob.com/what-is-web-application-architecture/#:~:text=Web%20Application%20Architecture%20is%20a,%2C%20user%20interfaces%2C%20and%20databases.

Monolithic Arch:



It is recommended for a small project.

Microservices:

For large projects, each microservice is responsible for single operation.

Isolation: services can develop independently

Scalability: a new service added at any stage of time

Flexible, Simplified development

Serverless Arch: