**UNITY UNIVERSITY**

**INTERNET PROGRAMMING ONE ASSIGNMENT**

**EYUEL ABIYOT**

**ID-NO: 04570/14**

**Section-2**

**Differentiate between websites, web applications and mobile apps?**

**Websites:**

* Websites are collections of web pages that are accessible through a web browser.
* They are primarily informational or content-driven, providing information, text, images, and multimedia to users.
* Websites are designed to be responsive and accessible on various devices, including desktops, tablets, and mobile phones.
* Interactivity is limited to links and forms, and they do not typically offer complex functionality or user-specific experiences.
* They do not need to be installed and are accessible via a URL.
* Example includes

1. **News Websites:** Websites like BBC, CNN, and The New York Times are classic examples of information-driven websites.
2. **Blogs:** personal or professional blogs, such as Medium or WordPress-hosted blogs.
3. **Portfolio Websites:** Individual artists, photographers, software developers and designers often create websites to showcase their work in a relatively static manner.

**Web Apps:**

* Web apps are interactive applications that are accessed through a web browser, just like websites.
* They provide more advanced functionality and interactivity, allowing users to perform specific tasks or actions online.
* Web apps can have user accounts and store data, often using databases.
* They may offer features like online shopping, social media platforms, email services, and productivity tools.
* Users can access web apps by entering a URL or bookmarking a specific web address.
* Example includes:
  + - * 1. **Social medias:** like YouTube, Facebook, Twitter, Instagram, Reddit, Discord and many more.
        2. **Online shopping’s:** likes of Amazon, eBay, Alibaba, Ali-express, Etsy to name a few.
        3. **Online banking’s:** they allow users to manage and do different operation on their accounts.

**Mobile Apps:**

* Mobile apps are software applications designed specifically for mobile devices (smartphones and tablets).
* They are downloaded and installed from app stores like the Apple App Store, Google Play Store and other stores as well.
* Mobile apps are optimized for the smaller screens and touch interfaces of mobile devices.
* They can work both online and offline, often providing a seamless user experience regardless of internet connectivity.
* Mobile apps can access device-specific features like GPS, camera, and push notifications.
* Examples:

**Social media apps:** most of the web apps mentioned above have mobile apps because there are features that can be implemented better on mobile apps than the web version one. For instance if we take the mobile app version of Instagram is a lot different that the web one like you can take pictures while you are on the web can’t add filters and effects and many other stuffs.

In summary, websites are primarily for information and content, web apps provide more advanced functionality through web browsers, and mobile apps are tailored for specific tasks on mobile devices and offer a more integrated experience. Each has its unique use case and advantages, and businesses often use a combination of these platforms to reach their audience effectively.

**What’s API? How to build it and why we need API.**

An API, or Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate with each other. APIs define the methods and data formats that applications can use to request and exchange information. They act as intermediaries, enabling the integration of various systems and services. APIs are essential in modern software development for several reasons:

* Data Sharing: APIs allow applications to share and access data from other sources. For example, a weather application can use an API to fetch real-time weather data from a weather service provider like openweathermap.org.
* Functionality Integration: APIs enable the integration of external functionality into an application. For instance, a payment gateway API like **stripe, Paypal, chapa, telebirr** can be used to process online payments within an e-commerce website.
* Modular Development: APIs facilitate modular development, where different components of a software system are built separately but can interact through APIs. This promotes code reusability and simplifies development.
* Cross-Platform Compatibility: APIs make it possible for applications to work across different platforms and devices. For example, social media APIs enable users to share content from one app to another. For instance Instagram provide users with the ability to share the same post to their Facebook profiles this is possible with use of api’s.
* Third-Party Services: Developers often use APIs from third-party services to leverage existing solutions and reduce development time and effort. This is common in areas like mapping, social media integration, and cloud services.

To build an API, we typically follow these steps:

* Design: Define the purpose and functionality of the API. Decide what data or operations it will expose to other applications.
* Choose a Protocol: Select a communication protocol for your API, such as HTTP/HTTPS, REST, GraphQL, or SOAP. The choice of API protocol depends on factors like the use case, programming language, performance requirements, and existing infrastructure.
* Development: Implement the API by creating endpoints like GET, POST or functions that handle incoming requests and return appropriate responses.
* Testing: Thoroughly test the API to ensure it functions as intended, handling various input scenarios and error conditions.
* Documentation: Create comprehensive documentation that describes how to use the API, including available endpoints, request parameters, and response formats. These allow other developers to work on the api’s with ease.
* Security: Implement security measures like authentication and authorization to control who can access your API and what actions they can perform.
* Deployment: Deploy the API on a server or cloud platform, making it accessible over the internet.
* Maintenance: Continuously monitor and maintain the API, addressing bugs, adding features, and ensuring it remains compatible with evolving technologies.

Why do we need APIs?

* Interoperability: APIs allow different software systems to work together, regardless of their underlying technologies or platforms. This interoperability is crucial for modern software ecosystems.
* Efficiency: APIs save developers time and effort by enabling them to build on existing functionalities rather than reinventing the wheel. This accelerates development and reduces costs.
* Scalability: APIs facilitate the modular architecture of applications, making it easier to scale and add new features without disrupting the entire system.
* Ecosystem Growth: APIs enable businesses to create platforms and ecosystems that encourage third-party developers to build applications and services around their core offerings.
* Specialization: APIs empower developers to specialize in specific areas of software development without needing expertise in every aspect. For instance, a mobile app developer can use APIs to add advanced features like geolocation or social media sharing.

In summary, APIs are the backbone of modern software development, allowing applications to interact, share data, and access external services, ultimately enhancing functionality and efficiency in the digital world.

**What’s library and why we need libraries?**

In the context of programming languages, a library is a collection of pre-written code and functions that can be used by developers to simplify and expedite the software development process. Libraries serve several crucial purposes in programming:

* Reusability: Libraries encapsulate common and reusable code, such as data structures, algorithms, and utility functions. This allows developers to avoid reinventing the wheel and instead utilize existing solutions to solve typical programming problems. This reusability saves time, effort, and reduces the risk of introducing bugs.
* Abstraction: Libraries provide a level of abstraction by hiding complex implementation details. Developers can use library functions without needing to understand the internal workings of the code, which makes it easier to work with and simplifies the development process.
* Efficiency: Libraries are typically well-optimized and thoroughly tested, making them efficient and reliable. This can lead to improved performance and fewer programming errors.
* Interoperability: Libraries facilitate interoperability by providing a common interface to interact with external systems, services, and hardware components. This is essential for integrating software with various technologies and devices.
* Extensibility: Developers can extend and enhance the functionality of their programs by incorporating third-party libraries. This allows them to add new features and capabilities without having to build everything from scratch.
* Domain-Specific Solutions: Libraries are often tailored to specific domains, such as web development, data analysis, machine learning, or graphics. They provide specialized tools and functions that are essential for those particular fields.

Examples of libraries in programming include the Standard Template Library (STL) in C++, the Python Library like NumPy, and numerous JavaScript libraries like React and jQuery. These libraries cover a wide range of functionality, from handling data structures to implementing user interfaces, making them indispensable tools for developers working with specific programming languages

**Website development process**

* **Requirements gathering and analysis**: The first step in any website development project is to gather and analyze the requirements. This involves meeting with the client to understand their goals for the website, their target audience, and their budget. The development team will also need to understand the technical requirements of the website, such as the hosting platform and the content management system (CMS).
* **Information architecture and design:** Once the requirements have been gathered and analyzed, the development team can begin to create the information architecture and design of the website. The information architecture is the structure of the website, and it defines how the different pages will be organized and linked together. The design of the website includes the layout, typography, color scheme, and overall visual style.
* **Content development:** Once the information architecture and design are complete, the development team can begin to develop the content for the website. This includes writing the text for the pages, as well as adding images, videos, and other multimedia content. The content should be well-written, informative, and engaging.
* **Development:** Once the content has been developed, the development team can begin to develop the website. This involves writing the code for the website and implementing the features and functionality. The development team will also need to make sure that the website is responsive and mobile-friendly.
* **Testing:** Once the website has been developed, it is important to test it thoroughly to make sure that it is working properly. This involves testing all of the links, forms, and other functionality. The development team will also need to test the website on different browsers and devices to make sure that it is compatible.
* **Deployment:** Once the website has been tested and approved by the client, it is ready to be deployed. This involves uploading the website files to a web server and making it live to the public.
* **Maintenance and support:** Once the website is deployed, it is important to maintain and support it. This includes fixing any bugs that are found, updating the content, and adding new features and functionality. The development team may also need to provide technical support to the client.