Research _ Kymbat Zaidi

Prompt 1: How do web APIs work? Why are they useful?

Web APIs, or Application Programming Interfaces, allow different software systems to communicate and interact with each other over the internet. They enable developers to access and utilize the functionality and data of another application or service without having to understand the underlying implementation details. Web APIs work through a request-response cycle. When a client application wants to access a specific resource or perform an action provided by an API, it sends an HTTP request to a specific URL, known as an endpoint, along with any necessary parameters or data. The API server receives the request, processes it, and sends back an HTTP response containing the requested data or information about the outcome of the requested action.

APIs are useful for several reasons:
1. Integration: APIs enable different systems, services, or applications to work together seamlessly, allowing data and functionality to be shared and utilized across platforms.
2. Extensibility: APIs provide a way to extend the functionality of an application by allowing developers to build on top of existing services or platforms.
3. Efficiency: Instead of reinventing the wheel, developers can leverage existing APIs to quickly access pre-built functionalities, such as payment processing, mapping services, social media integration, etc.
4. Collaboration: APIs facilitate collaboration between developers and teams by allowing them to work on different parts of a project independently while ensuring interoperability between components.
5. Innovation: APIs empower developers to create new applications or services by combining or integrating various APIs, unlocking new possibilities and driving innovation.

Prompt 3: What does REST stand for? What is REST? What are some of the key concepts that identify REST?

REST stands for Representational State Transfer. It is an architectural style for designing networked applications, particularly web services, that utilize the HTTP protocol. REST is based on a set of principles and constraints that promote scalability, simplicity, and interoperability between systems.

Key concepts that identify REST include:
1. Stateless: REST is stateless, meaning that the server does not store any client context between requests. Each request from the client must contain all the necessary information for the server to understand and process it.
2. Resource-Oriented: REST treats resources as the fundamental building blocks of the system. Each resource is uniquely identified by a URL (Uniform Resource Locator) or URI (Uniform Resource Identifier).
3. CRUD Operations: RESTful APIs support standard CRUD (Create, Read, Update, Delete) operations on resources using the HTTP methods: POST, GET, PUT, and DELETE, respectively.

4. Uniform Interface: REST emphasizes a uniform interface between clients and servers, promoting a clear separation of concerns. This includes using standard HTTP methods, following a consistent URL structure, and utilizing appropriate HTTP status codes and response formats.
5. Hypermedia as the Engine of Application State (HATEOAS): HATEOAS is a concept in REST that suggests including hypermedia links in responses, enabling clients to navigate the API's available actions dynamically.

RESTful APIs have become widely adopted due to their simplicity, scalability, and compatibility with the existing web infrastructure.

Sources:
- Roy T. Fielding's doctoral dissertation: "Architectural Styles and the Design of Network-based Software Architectures": [https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm]
- RESTful API Design - Best Practices in a Nutshell: [https://blog.mwaysolutions.com/2014
- MDN Web Docs: "Introduction to web APIs": [https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction]
- RapidAPI Blog: "What is an API and Why is it Important?": [https://rapidapi.com/blog/api-glossary/what-is-an-api/]