CSC 560 Unit One Notes

9 January 2023

API

* API 🡪 Application Programming Interface, a software intermediary that allows two applications to talk to each other.
  + The application connects to the internet and sends data to a server, the server then retrieves that data, interprets it, performs the necessary actions and sends it back to the application, where the application interprets that data and presents you with the information you wanted in a readable way
  + Provides security as your application is never completely connected to the server, you are using an interface to interact with the server
* Modern APIs
  + Adhere to standards (HTTP and REST)
  + Treated more like a product than like code 🡪 they follow the SDLC for being a product itself rather than just a codebase
  + Stronger discipline for security and governance & are monitored and managed for performance and scale
* REST (REpresentational State Transfer) Constraints
  + Client-Server
    - Separating the UI from the data, the portability of UI across platforms improves and simplifies the server components allowing scalability
  + Stateless
    - Each request from the client to server must contain all the necessary information to understand the request, aka there is no stored context on the server & only kept on the client
  + Cacheable
    - Data within a response must be labeled as cacheable or non-cacheable, aka whether or not you are able to use response data for later, equivalent requests
  + Uniform interface
    - Identification of resources
    - Manipulation of resources through representations
    - Self-descriptive messages
    - Hypermedia as the engine of application state
  + Layered system
    - Allows an architecture to be composed of hierarchical layers by constraining component behavior such that each component cannot “see” beyond the immediate layer with which they are interacting
  + Code on demand (optional)
    - Allows client functionality to be extended by downloading and executing code in the form of applets or scripts, simplifying clients by reducing the number of features required to be pre-implemented
* HTTP
  + A protocol that allows the fetching of resources, such as HTML documents
  + The foundation of any data exchange on the web & is client-server protocol
  + An application layer protocol that is sent over TCP (or TLS-encrypted TCP) through any reliable transport protocol
  + Used to fetch HTML, images, videos, or post content to servers
  + HTTP methods used in REST based architecture
    - GET 🡪 provide read only access to a resource
    - PUT 🡪 used to create a new resource
    - DELETE 🡪 used to remove a resource
    - POST 🡪 used to update an existing resource or create a new resource
* Data representation languages
  + XML (eXtensible Markup Language)
    - Similar to HTML
      * HTML was designed to display data, XML was designed to carry data
      * XML tags are not predefined like HTML
    - Designed to store and transport data
    - Designed to be self-descriptive
    - Simplifies data sharing, transport & availability and platform changes
  + JSON (Java Script Object Notation)
    - Similarities to XML
      * Both are self describing
      * Both are hierarchical
      * Both can be parsed and used by a lot of programming languages
      * Can be fetched with an XMLHttpRequest
    - Differences to XML
      * Doesn’t use end tag
      * Shorter
      * Quicker to read and write
      * Can use array
      * Can be parsed by a standard JavaScript function
    - Easier to use than XML because it is parsed into a ready-to-use Javascript object (aka you can fetch and parse a JSON string)
    - Data types
      * String
      * Number
      * JSON object
      * Array
      * Boolean
      * Null
    - Why use JSON?
      * Standard Structure 🡪 easily human readable because developers know what to expect from JSON
      * Light weight 🡪 easier to get and load the requested data quickly
      * Scalable 🡪 language independent (aka it works with most modern languages), you can easily change server or client side languages if needed
* RESTful Web Services
  + Web service 🡪 a collection of open protocols and standards used for exchanging data between applications or systems
  + RESTful Web Services 🡪 uses HTTP methods to implement the concept of REST architecture
    - Usually defines a URI (Uniform Resource Identifier) 🡪 provides resource representation like JSON and HTTP methods
    - URI vs. URL
      * URI
        + Provides a technique for defining the identity of an item
        + Used to distinguish one resource from other regardless of method used
        + Doesn’t contain the protocol specification
        + Superset of URL
      * URL
        + Used to describe the identity of an item
        + Links a web page, a component of a web page, or a program on a web page with the help of accessing methods like protocols
        + Provides the details about what type of protocol is used
        + A type of URI
* Node.js
  + Runtime environment for executing JavaScript code outside of a browser
  + Ideal for building highly-scalable, data-intensive backend services (APIs) that power your client’s apps
  + Good for prototyping and agile development
* Application Architecture

A picture containing diagram

Description automatically generated

January 12, 2023

(ACH is trying to combine structural analytical techniques & HIM) 🡪 Prototype this as much as possible

* Structure analytical techniques
  + Mission that has a framework that allows the user to determine what is going on
  + Generate a dataset in order to put into the human intelligence networks
* Human Intelligence Networks
  + Ability to make ad hoc human intelligence networks that exist

ACH (need competing hypotheses)

* User input
* Need at least two possible root causes from the analyst
* Only a human can create the competing hypotheses