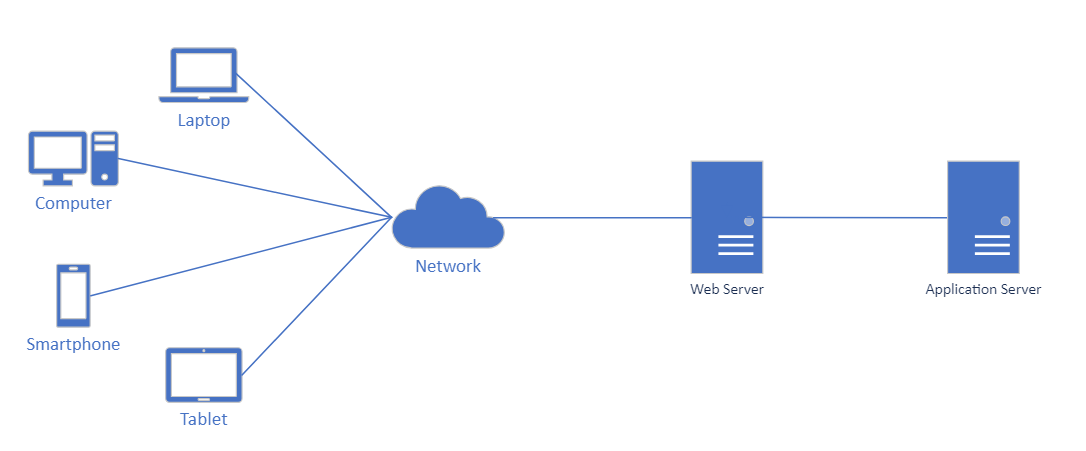
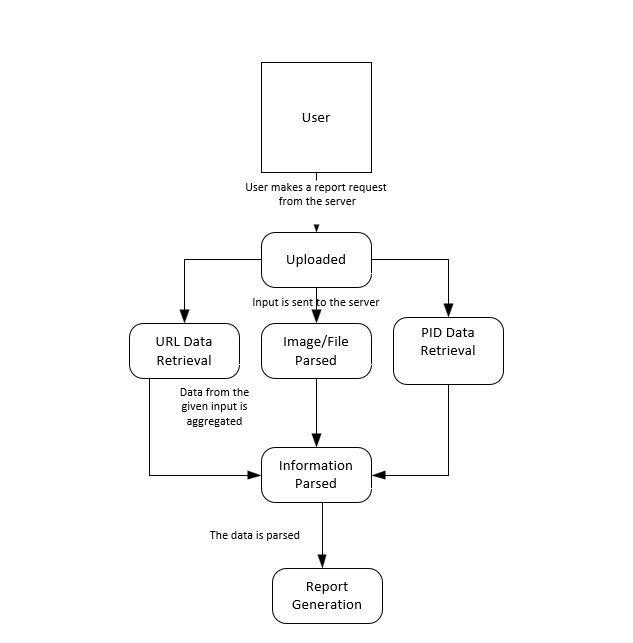
**Orthogonality - How Our Program Works**

The overall design of our system is quite orthogonal. Our OSINT Dashboard emphasizes the independence and modularity of the tools and components involved in the data retrieval and processing tasks. Below are some key aspects of our system:

1. **Independence of Data Retrieval Tools:**
   1. Within our system we have multiple tools for data retrieval given a valid url.
   2. The tools are responsible for fetching information such as IP information, DNS records, SSL certificates, and other relevant data.
   3. These tools operate independently, performing specific tasks without relying on the other tools.
   4. With this separation, our tools are able to focus on their designated purpose without being affected by changes or modifications made to the other tools.
      1. For example if there is an update or modification to the SSL certificate retrieval tool, it would not impact the performance or functionality of the DNS records retrieval tool, and vice versa.
2. **Independence of Image and File Analysis** 
   1. Similar to above, the tools involved in handling image and file analysis, such as the extraction of EXIF data, operate autonomously and independently
   2. This modularity ensures that the image/file analysis functionalities can be updated without causing disruption to other parts of the system
      1. For example, if enhancements are made to the EXIF data extraction component, it will not affect the functioning of other components responsible for file analysis, such as content scanning or metadata extraction.
3. **Flexibility and Scalability**
   1. By maintaining the orthogonality of these tools and components, your OSINT Dashboard gains flexibility and scalability. New functionalities and tools can be added or modified without affecting the existing components, allowing for the seamless expansion of the system's capabilities.
   2. This flexibility enables your system to adapt to evolving user requirements and technological advancements, ensuring that it remains relevant and effective in the rapidly changing landscape of OSINT analysis.
4. **Simplified Maintenance and Troubleshooting** 
   1. The modular and independent nature of the tools and components simplifies the maintenance and troubleshooting processes. It becomes easier to isolate and identify issues within specific components without the need to navigate complex interdependencies.
   2. This streamlines the debugging and maintenance tasks, reducing the overall complexity and time required to address any potential issues or updates within the system.

**User/Client Architecture**

For our system’s architecture is a basic web application hosted via the user/client model. Users access the web and application servers tool through the internet.

**Data Flow Diagram**

**Hardware Our System Will Run On**

Web Server Specifications:

* CPU: Dual quad-core processors (Intel® Xeon® Bronze 3408U Processor (22.5M Cache, 1.80 GHz))
* RAM: 32 GB
* Storage: 128 GB SSD
* Network: Ethernet
* Operating System: Windows Server 2012

Application Server Specifications:

* CPU: Quad-core processor (Intel® Xeon® Bronze 3408U Processor (22.5M Cache, 1.80 GHz))
* RAM: 16 GB
* Storage: 128 GB SSD
* Network: Ethernet
* Operating System: Windows Server 2012

Logging:  
Grafana Cloud for monitoring of the system's performance. The free plan includes 50 GB logs, 14-day retention, and 3 monthly active users. We can upgrade to a premium plan if necessary.

Scalability:  
We will migrate to Microsoft Azure if the demand rises to a point that more web and application servers become unmanageable due to limited staff, location and power restrictions, financial incentives, etc.

Topics not covered:  
Load Balancers, Redundancy, Disaster Recovery, CDN Services, Power & Cooling, DDoS protection mechanism (CloudFlare)

**Security**

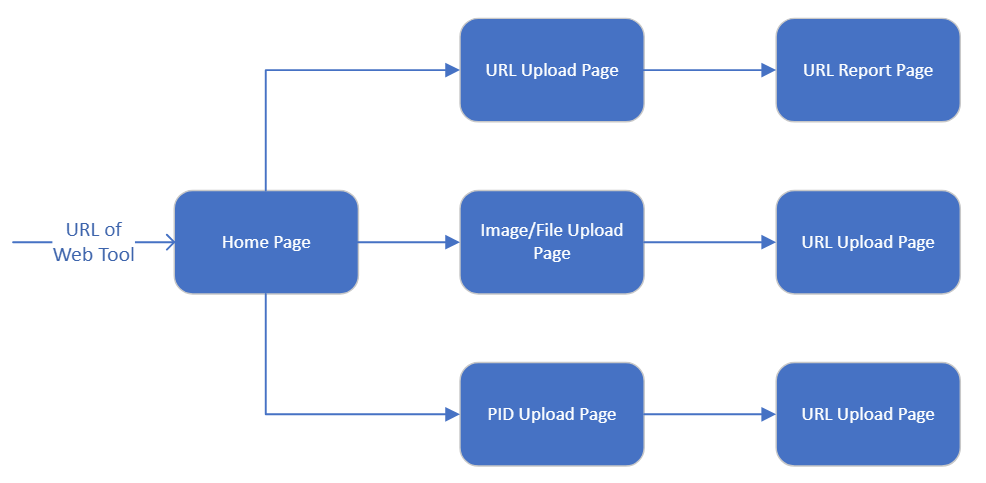
* Input Validation: Implement input validation mechanisms to ensure that the data entered by the users is in the expected format and does not contain any malicious code or scripts.
* Rate Limiting: Implement rate limiting to prevent potential misuse or abuse of the system. This will help in controlling the number of requests that can be made from a particular IP address within a specific time frame.

Security Audits and Updates: Regularly perform security audits to identify and address any vulnerabilities or weaknesses in the system. Keep the system updated with the latest security patches and fixes.

**User Interface**

Our user interface design revolves around the idea of report content generation, with three different paths the user can take depending on the type of information they request. Depending on the type, a report page will be generated with the included content.

The Home page will explain how the tool works and various applications of the tool. There will also be three buttons that will redirect the user to the three different user input pages (URL, File, and PID). Each of the upload pages are more then less the same, which offers text boxes collecting information. Once information is uploaded, a report page is created specifically for the type of content submitted. On the report page for each type of content, there will be information about all the data that was collected, its origination, and its use cases. Users will be able to save the generated report, as well as traverse back to the home or other upload pages.

UI Diagram  
****

**Reports**

* Report of data generated by user input. Will be used in the creation of the report pages.
* Report what information is being uploaded into the system (URLS and filenames)
  + Files will not be saved by the system.
* Report of individuals connecting to the service, as well as which tools are being utilized by who.
  + Help identify the popularity of each tool.