Malware*X*plore

Project Team

Minam Faisal 21I-1901

Momenah Saif 21I-1909

Session 2021-2025

Supervised by

Dr. Qaisar Shafi

Co-Supervised by

Mr. Muhammad Abdullah Abid



Department of Cyber Security

National University of Computer and Emerging Sciences Islamabad, Pakistan

June 2025

Table of Contents

[1. Introduction 6](#_Toc183859887)

[1.1 Existing Solutions 7](#_Toc183859888)

[1.2 Problem Statement 9](#_Toc183859889)

[1.3 Scope 9](#_Toc183859890)

[1.4 Modules 10](#_Toc183859891)

[**1.4.1** **Malware Analysis Module** 10](#_Toc183859892)

[**1.4.2** **Data management module** 10](#_Toc183859893)

[**1.4.3** **Visualization and Reporting Module** 11](#_Toc183859894)

[**1.4.4** **Federated Learning module** 11](#_Toc183859895)

[1.5 Work Division 11](#_Toc183859896)

[2. Project Requirements 12](#_Toc183859897)

[2.1 Event Response Table 12](#_Toc183859898)

[2.2 Functional Requirements 13](#_Toc183859899)

[**2.2.1** **Centralized and decentralized storage** 13](#_Toc183859900)

[**2.2.2** **Data privacy and security** 13](#_Toc183859901)

[**2.2.3** **Federated learning integration** 14](#_Toc183859902)

[**2.2.4** **AI-Driven learning** 14](#_Toc183859903)

[**2.2.5 Automated malware detection** 14](#_Toc183859904)

[**2.2.6 Visualization dashboard** 14](#_Toc183859905)

[**2.2.7 Web Portal for PDF Upload** 14](#_Toc183859906)

[2.3 Non-functional requirements 15](#_Toc183859907)

[**2.3.1** **Reliability** 15](#_Toc183859908)

[**2.3.2** **Usability** 15](#_Toc183859909)

[**2.3.3** **Performance** 16](#_Toc183859910)

[**2.3.4** **Security** 16](#_Toc183859911)

[3. System Overview 17](#_Toc183859912)

[3.1 Architectural Design 17](#_Toc183859913)

[3.2 Data design 20](#_Toc183859914)

[3.3 Domain Model 21](#_Toc183859915)

[3.4 Design Models 22](#_Toc183859916)

[3.4.1 Activity Diagrams 22](#_Toc183859917)

[3.4.2 Sequence Diagram 25](#_Toc183859918)

[3.4.3 State Diagram 27](#_Toc183859919)

[3.4.4 Data Flow Diagram 29](#_Toc183859920)

[4. Implementation and testing 33](#_Toc183859921)

[4.1 Algorithm Design 34](#_Toc183859922)

[4.2 External API 37](#_Toc183859923)

[4.3 Testing details 37](#_Toc183859924)

[4.3.1 Unit Testing 38](#_Toc183859925)

List of Figures

Figure 1.1 7

Figure 3.1.1 19

Figure 3.1.2 20

Figure 3.2.1 21

Figure 3.3.1 22

Figure 3.4.1 23

Figure 3.4.2 23

Figure 3.4.3 24

Figure 3.4.4 24

Figure 3.4.5 25

Figure 3.4.6 26

Figure 3.4.7 27

Figure 3.4.8 28

Figure 3.4.9 29

Figure 3.4.10 30

Figure 3.4.11 30

Figure 3.4.12 31

Figure 3.4.13 32

List of Tables

Table 1.1.1 8

Table 1.5.1 11

Table 2.1 12

Table 4.2.1 37

Table 4.3.1 38

Table 4.3.2 38

Table 4.3.3 39

Table 4.3.4 39

Table 4.3.5 40

Chapter 1

# Introduction

Malicious actors are progressively inserting malware into different file types as digital content grows in popularity. Because PDFs are so widely used and are thought to be safe, they are becoming a major channel. Traditional malware detection techniques frequently rely significantly on static detection techniques or manual analysis, which may not be able to detect threats that are hidden or growing. To overcome these challenges, this project develops an automated system for identifying malware based on PDFs. This solution combines static and dynamic analytic approaches to more precisely and effectively identify threats.

The recent studies and publications in the industry reveal the inadequacies of traditional methods of malware detection, mainly while dealing with the detection of complex attacks encoded in PDFs. The existing solutions, which are based on static signatures, fail to identify the recent adaptive threats and require much manual involvement by cybersecurity experts. While some of these have dynamic analytic tools, effective only up to a point in time, they are oftentimes independent and fail to leverage real-time collaborative models.

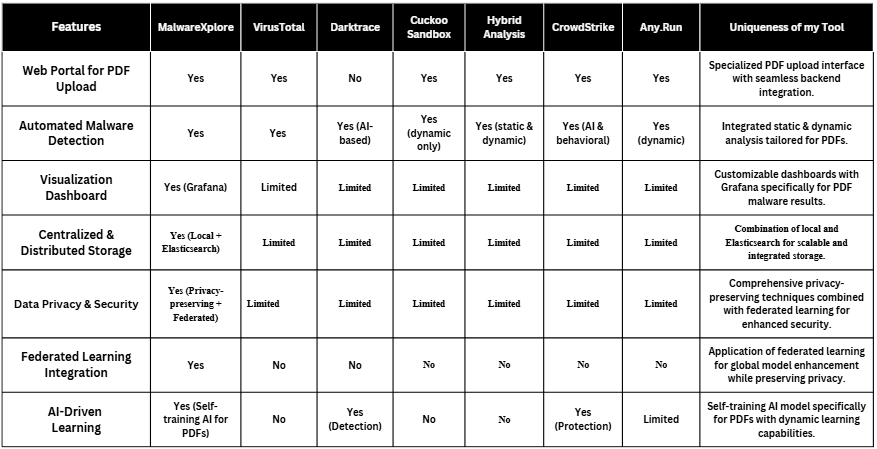
This study presents a new direction to filling in the gap by integrating both static and dynamic analytic methods into one single, automated system. That is, beyond the federated learning approach adopted by this system, instances can collaborate with each other and share ideas and improvements without exposing private information. More importantly, this method improves the detection accuracy beyond traditional static systems while allowing the model to learn from real world data in a continuous manner to better adapt to the emergence of new threats.

This system targets malware analysts, cybersecurity professionals, and system developers to reduce the level of manual labor required in the identification of malware by providing a more reliable and extendible solution for businesses working with thousands of PDF files. This has resulted in the outcome that is the powerful, flexible detection tool evolving according to the threat landscape and proving useful as a complement to extant cybersecurity tools.

## Existing Solutions

Current solutions, providing different means of automated malware detection, lack integration of static and dynamic analysis specifically tailored to PDF malware. Most tools lack federated learning capabilities, which would enable the collaboration of systems to learn from decentralized data sources while keeping privacy intact. Visualization options are quite limited and, in particular about storage of privacy-preserving support, most solutions lack sufficient support in cases of solutions that rely on centralized data storage. Besides, since the two existing systems are VirusTotal and Darktrace, which does not allow for customization of dashboards and does not focus specifically on PDFs, their capability to perform focused malware analysis in this area would be highly compromised.

MalwareXplore bridges the gaps between both methodologies to integrate static and dynamic analysis specifically for PDF files in a single platform. Also, it is offering federated learning for continuous improvement across the different systems, keeping data integrity and security completely intact. The introduction of central storage and distributed storage with privacy-preserving techniques along with AI-driven self-training specifically for PDFs gives a quality edge on this tool over the existing platforms. This focused and novel approach fills loopholes in contemporary solutions toward a more efficient, accurate, as well as privacy-focused malware detection approach.



*Fig 1.1: Comparison of Existing Solutions with MalwareXplore*

Table 1.1.1: Comparison of Existing Solutions

|  |  |  |
| --- | --- | --- |
| System Name | System Overview | System Limitations |
| VirusTotal | VirusTotal provides a web portal for file uploads and automated malware detection. It supports limited visualization and does not offer federated learning or privacy-preserving techniques. | Limited data privacy measures, restricted to basic storage, and lacks federated learning capabilities, making it less adaptable for collaborative learning. |
| Darktrace | Darktrace offers AI-based automated malware detection, primarily for enterprise networks, with advanced threat detection algorithms. | Limited visualization and storage option, lacks support for federated learning, and does not focus on specific file types like PDFs. |
| Cuckoo Sandbox | Cuckoo Sandbox provides dynamic analysis capabilities to detect malicious behavior in files. It is widely used for behavior-based malware analysis. | Focused only on dynamic analysis without static analysis, limited visualization and storage capabilities, and lacks federated learning or AI-driven self-training. |
| Hybrid Analysis | Hybrid Analysis combines static and dynamic analysis, providing a broader detection scope and some behavioral insights. | Limited visualization and distributed storage option, lacks federated learning and strong data privacy features. |
| CrowdStrike | CrowdStrike offers AI-driven automated malware detection with behavioral analysis, targeted at large organizations for network security. | Limited focus on PDFs, lacks federated learning, and does not prioritize data privacy or decentralized storage solutions. |
| Any.Run | Any.Run provides an interactive web-based platform for dynamic malware analysis with real-time feedback and visualization. | Limited automated static analysis, restricted storage options, and lacks federated learning or advanced privacy measures. |

## Problem Statement

Current malware analysis is severely fragmented because it relies on manual investigation and has minimal collaborative capabilities. Many cases of static and dynamic analysis have to take place independently; Isolated results are then produced, and there is a good chance that these may fail to detect multi-layered threats with multiple layers. Analysts normally have to manually extract information from logs, investigate suspicious activities, and then gather results from multiple unconnected tools-a situation that is both inefficient as well as prone to human error. As malware threats accelerate, with evolving sophistication, a purely human-based approach often results in delayed detection with threats continuing to flourish and spread.

These traditional systems, meanwhile, are not coordinated with one another. When individual systems learn to recognize new types of malwares, these updates are kept encapsulated, limiting overall detection efficiency and endangering the development of detection models within the field. This further limit innovation in detection accuracy and makes it difficult for analysts and systems to maintain parallelism with new threats in circulation.

This solution unifies the static and dynamic analysis in one system, giving a much more wholesome approach towards malware detection. The system reduces manual input with AI-driven automation, accelerates threat detection, and boosts accuracy. The decentralized federated model also considers that different instances can share their techniques on detection without exposing sensitive information. This collaborative framework allows for detection accuracy to be improved, plus the system constantly updates itself with new threats, providing an active and resilient malware analysis approach. This solution is important to the field because it creates an efficient, precise, and more collaborative global malware detection system that may evolve with the progression of the threat landscape.

## Scope

The malware detection system is built to assist malware analysts by automating both static and dynamic analysis, specifically targeting PDF files, which are often used to deliver hidden threats. The tool’s main goal is to bring both types of analysis together in one platform, making it easier for analysts to identify and examine potential malware without switching between multiple tools. The results will be displayed on a clear, easy-to-use dashboard, with the option to download comprehensive reports for deeper analysis.

One of its core feature mechanisms of federated learning enables it to continually enhance detection accuracy by learning continuously from data across different systems without exposing the underlying sensitive information. This decentralized collaborative approach lets the tool quickly adapt to new and emerging threats, ensuring it remains effective against the ever-changing malware landscape. Privacy concerns are addressed because there is no central storage of data, which would be attractive for organizations with very strict data security requirements.

However, exclusions have been given to the project to keep the focus. For instance, advanced post-deployment support like system maintenance or version updates, after it is initially developed, would not fall under the scope of work for the project. Moreover, the tool specializes particularly in identifying malware based on PDFs. Other file types, such as Word files and Excel files, Exe files are out of the scope of the tool since it needs different techniques to carry out the tasks.

## Modules

Here is a breakdown of each module along with its explanation:

### **Malware Analysis Module**

This module automates the analysis of malware in PDFs. It includes a web portal where user can easily upload the file for analysis.

Feature 1: Automated Malware Analysis

Feature 2: Web Portal for PDF Upload

### **Data management module**

Storing and managing the analysis data will be done by this module. By using both centralized and distributed storage it ensures data security and availability.

Feature 1: Centralized and Distributed Storage

Feature 2: Data Privacy and Security

### **Visualization and Reporting Module**

This module includes a dashboard that displays the analysis results through charts and graphs. It also provides report of analysis results which the user can download.

Feature 1: Visualization Dashboard

Feature 2: Downloadable Reports

### **Federated Learning module**

This module focuses on the malware detection part. It ensures that AI models are trained on decentralized sources and enhanced detection by continuously improving its detection capabilities and preserving privacy.

Feature 1: Federated Learning Integration

Feature 2: AI-Driven Learning

## Work Division

For each module and respective feature, assign responsibility to a team member:

Table 1.5.1 Work Division

|  |  |  |
| --- | --- | --- |
| Name | Registration | Responsibility/ Module / Feature |
| Minam Faisal | 21i-1901 | * UI/UX design, Documentation * Frontend and Dashboard Development * Automation   Federated Learning Setup |
| Momenah Saif | 21i-1909 | •Static Analysis Tool Integration  •Dynamic Analysis Tool Integration  •Automation  •Elastic Search and Grafana Setup |

Chapter 2

# Project Requirements

This chapter describes the functional and non-functional requirements of our project:

## Event Response Table

We used Event- response table as requirement gathering technique. The event- response table is for a real-time system in which most of the functionalities are performed at backend.

Table 2.1 Event Response Table

|  |  |  |
| --- | --- | --- |
| Event | System State | Response |
| User Create account | |  | | --- | | No active account for the user | | System validates input, creates account, and notifies user of successful registration |
| User login attempt | Account information is saved in database | System checks credentials; if valid, user logs in; if invalid, error message is shown |
| User signup attempt | Account already exists with matching details | System notifies user that account exists, and lets the user access the website |
| User goes to About Us page | Website content is loaded and an option of about us is displayed | Displays the About Us page content to the user |
| User goes to History | Website content is loaded and option to show history of user is displayed | Displays the previously uploaded pdfs to the user |
|  |  |  |
| User edits profile | User profile data is retrieved | Updates profile with new data; saves changes in the database and confirms update to the user |
| PDF file uploaded | Valid PDF file detected | File is accepted and submitted to Flask backend for further processing |
| Invalid file uploaded | |  | | --- | | File does not meet format |  |  | | --- | |  | | File is rejected, and error message is shown to user |
| Static analysis of PDF | Valid PDF file uploaded by user and static analysis is selected | Analyze file structure and metadata without executing the file |
| Dynamic analysis of PDF initiated | Valid PDF file uploaded by user and dynamic analysis is selected | Execute PDF in a virtual Environment of sandbox. |

## Functional Requirements

The following are the Functional Requirements of our Project:

### **Centralized and decentralized storage**

The following are the requirements:

* FR1: The system will store the malware analysis results in both centralized (Elasticsearch) and decentralized (Federated local models) ways.
* FR2: The system will make it easier to find and use information from different locations, making it less likely for data to be lost or stolen.

### **Data privacy and security**

The following are the requirements for this module:

* FR1: All PDF files, analysis results and user information will be encrypted in transit and rest states.
* FR2: The system will ensure that the analysis of the PDFs the user has uploaded are only shown to that specific user.

### **Federated learning integration**

* FR1: The system will enable the local models to share the data of malicious pdf to global model for training without the need to share the malicious pdf or sensitive data.
* FR2: The system will regularly update its local model which will be sent to global model and all the other participating nodes will be refined.

### **AI-Driven learning**

* FR1: The system will automatically update its machine learning model on new features for more accurate detection of malwares.
* FR2: The system will support continuous model improvement by adding new analysis data into the training dataset.

### **2.2.5 Automated malware detection**

* + FR1: The system shall do static analysis on uploaded files to know its metadata and much more.
  + FR2: The system shall do dynamic analysis on uploaded files by running them in virtual environment of cuckoo sandbox.

### **2.2.6 Visualization dashboard**

* FR1: The system will allow the user to access the results on analysis on Grafana dashboard in form of charts.
* FR2: The system will allow the user to download the analysis results in form of report from Grafana dashboard.

### **2.2.7 Web Portal for PDF Upload**

* FR1: The system will allow user to create accounts.
* FR2: The system will allow user to upload PDF files and will reject any other format files.

## Non-functional requirements

This section outlines the non-functional requirements that define the system’s quality attributes such as reliability, usability, performance, and security.

### **Reliability**

The system must provide high reliability for malware analysis, data management and federated learning operations so that failures are not repeated frequently within the system.  
  
**RELI-1** If a system failure occurs during the uploading of files or during the analysis, the system will alert the user, then automatically continue from the last known state.

**RELI-2:** Mechanism for automated failure detection must be in place so that there is detection and notification of failure during malware detection or update in federated learning.

### **Usability**

The usability of the system will ensure user-friendly experience, especially in case of malware analysts and security personnel, who use a dashboard and reporting tools.

**USE-1:** The web portal must assure that the interface is user friendly and that can be achieved with obvious visible indicators of upload progress and analysis status as well as reports are available.

**USE-2:** The dashboard user interface must enable users to select charts and graphs for graphical display of analysis output, to their taste.

**USE-3:** There must be a help section, on the web portal, enabling the users to guide them through the upload and analysis and reporting processes.

### **Performance**

Performance requirements will make the system functional at different times, allowing many user requests and other processes of malware analysis to be performed within a timely manner.  
  
**PER-1:** At normal load, the system shall be able to complete its static and dynamic analysis on a 10MB PDF quickly.

**PER-2:** The model updates of the federated learning should be processed and integrated into the system within a few minutes of receiving new data from decentralized sources.

### **Security**

The security aspect is what ensures the integrity of the system and the privacy of user information.

**SEC-1:** All data uploads, downloads, and any communications between the web portal and back-end services should be encrypted end-to-end.

**SEC-2:** All data of analysis should be encrypted with strong algorithm, including PDF uploads and results.

**SEC-3:** Backend services and databases used by the system shall be accessible only by authorized personnel through authentication.

**SEC-4:** Federated learning updates and data exchanges should be performed over a secure environment that does not allow the sharing of sensitive data between organizations.

Chapter 3

# System Overview

The PDF Malware Detection System automates the analysis of malicious PDF files using both static and dynamic techniques. It allows users to upload files via a web portal, with results securely stored in a centralized and distributed database. The system includes an interactive dashboard for visualization and provides downloadable reports. Additionally, the system integrates a Federated Learning module, enhancing detection capabilities by continuously training AI models on decentralized data.

## Architectural Design

This malware detection architecture is divided into four major modules: the Malware Analysis Module, Data Management Module, Visualization and Reporting Module, and Federated Learning Module. Each module satisfies distinct functionalities; however, the modules work together to achieve seamless analysis and reporting experiences: -

1. **Malware Analysis Module**

**Description:**

This module automatically analyses uploaded PDF files for malware threats. Users upload the files through a web interface, and then all those files are statically and dynamically analyzed by the module for the presence of malicious behavior.

**Interactions:**

Requests the Data Management Module to store and retrieve files uploaded by users as well as the results of analysis performed by the module.

1. **Data Management Module**

**Description:**

It makes sure that the uploaded files, the results of analysis, and all related data are stored, organized, and secured. It maintains data integrity based on availability, security, and privacy through a centralized and decentralized storage mechanism.

**Interactions:**

Received results of analysis and file uploads from the Malware Analysis Module.

Serves data appropriate for the visualization reporting module to display results of analysis.

1. **Visualization and Reporting Module**

**Description:**

This module is supposed to provide the user with an easy user interface to view the results of the malware analysis graphically, say, in charts and graphs. Thus, the detailed reports of the malware analysis may be downloaded thus allowing the ultimate user to have a comprehensive view of the results of the analysis.

**Interactions:**

Retrieve data from the Data Management Module and Federated learning for the creation of graphical representations of analysis results.

1. **Federated Learning Module**

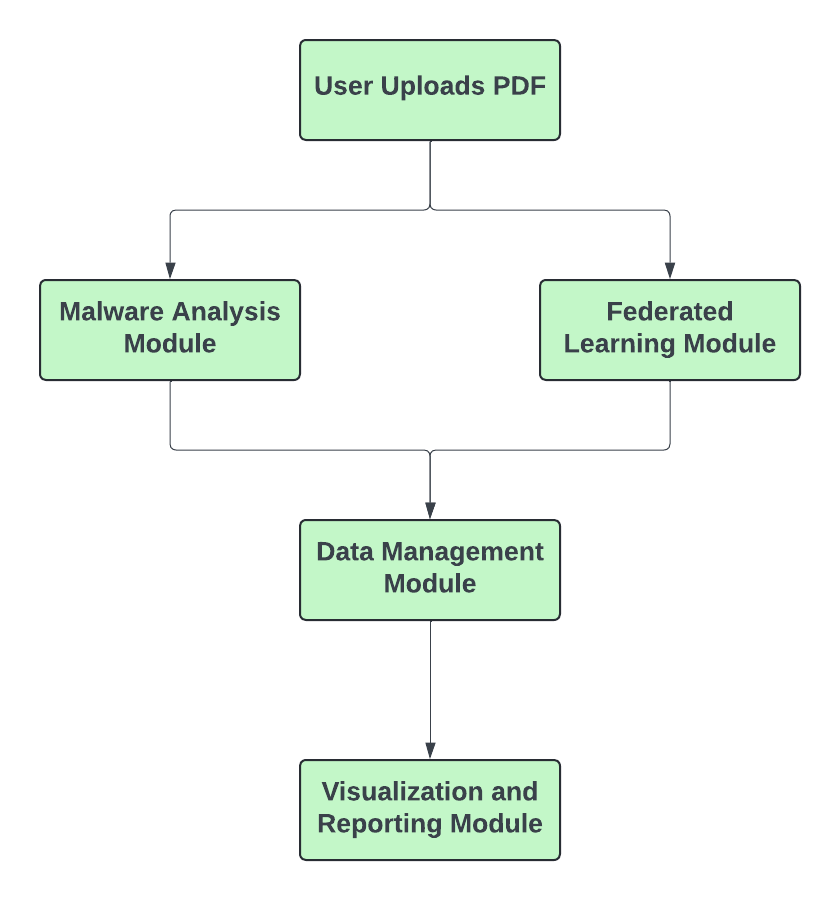
**Description:**

In the Federated Learning Module, machine learning improves malware detection capabilities. Models are trained in decentralized data, such that user privacy is preserved, and detection algorithms constantly improve because of learning from rich historical data from multiple sources.

**Interactions:**

It works to draw improvements in its algorithms through learned patterns from past data analyzed. It provides access to relevant datasets. It interacts with visualization module to display the results.

**Box and Line Diagram:**



*Figure 3.1.1*

**Final Architecture:**

**A diagram of a computer

Description automatically generated**

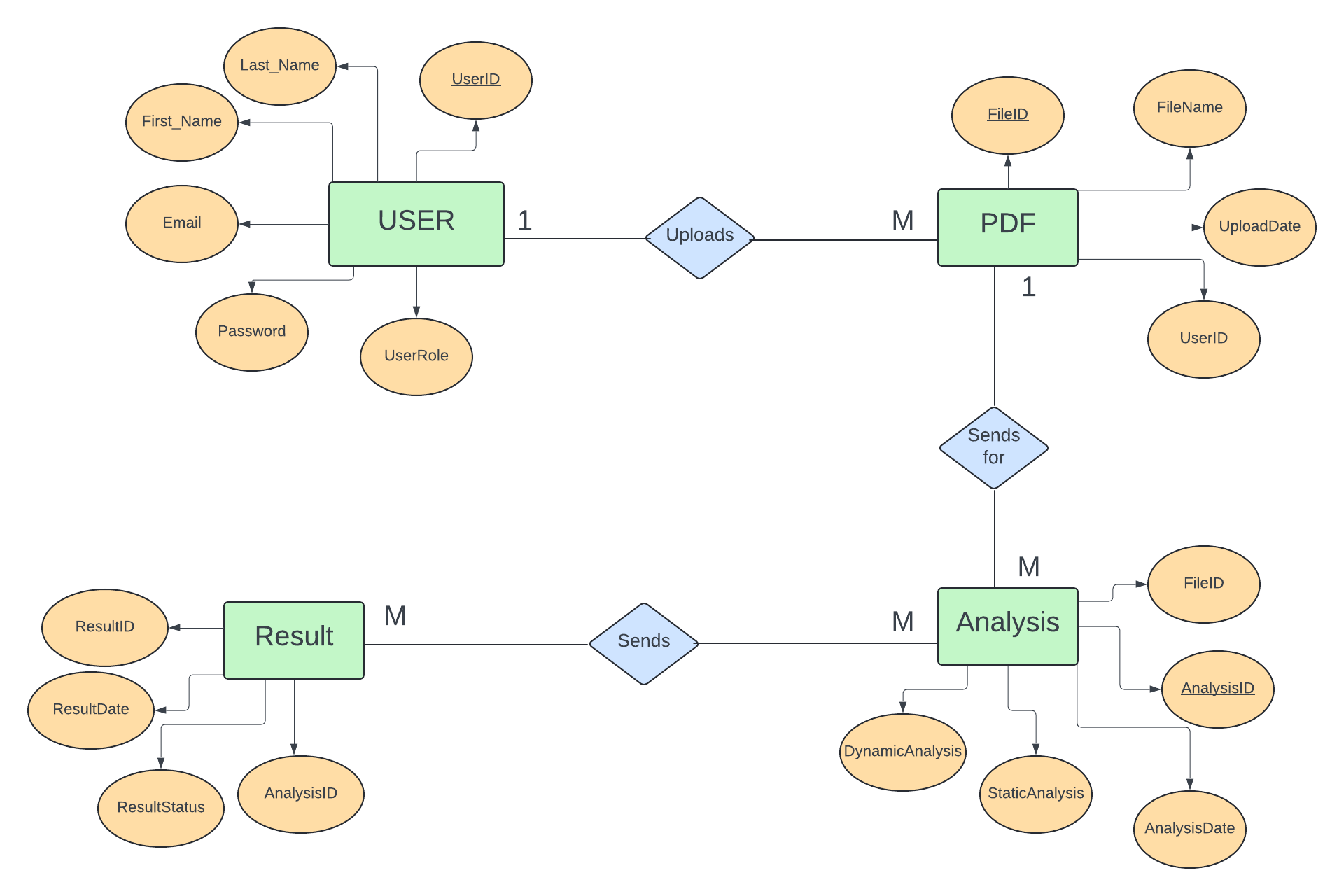
*Figure 3.1.2*

## Data design

In our system, the information domain is transformed into data structures by organizing the main entities (USER, PDF, Result, Analysis, and Federated Learning) into tables. Each entity has associated attributes, such as UserID, FileID, and ResultID, which are stored as fields in these tables. The relationships between entities are represented as foreign keys, such as users uploading multiple PDFs.

Data is stored in relational databases, where USER, PDF, Result, Analysis are structured as tables. Data processing involves users uploading PDFs, which are analyzed, and the results are then sent to the Federated Learning module for further analysis. The system uses a relational database (e.g., MySQL) and Elasticsearch as a database to manage and organize this information.

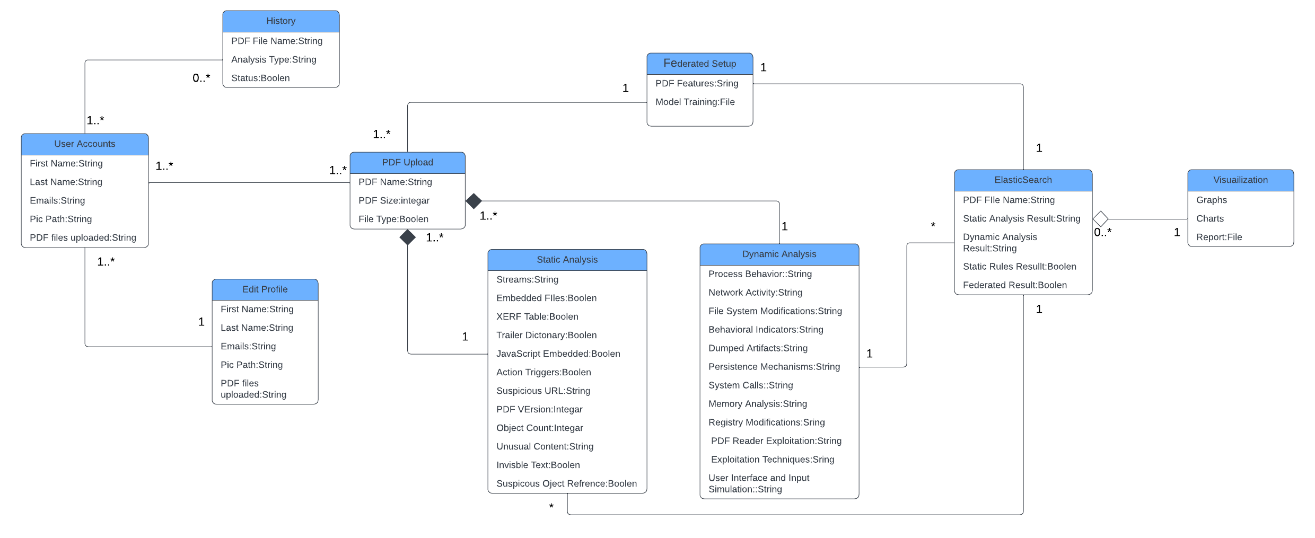
**ERD:**

****

*Figure 3.2.1*

## Domain Model

For the system, this domain model is uploading and analyzing PDFs with static and dynamic analysis and a federated learning model. Users are represented with attributes like userID, name, email, password. The PDF analysis produces static and dynamic results, time-stamped with start and end times. The result from analysis is stores with associated data display.



*Figure 3.3.1*

## Design Models

The applicable models for our project using ***a procedural approach*** are:

### Activity Diagrams

Activity Diagrams are based on events we found in event response table:

***Login/Sign up***

A diagram of a process

Description automatically generated

Figure 3.4.1

***Edit Profile***

***A diagram of a process

Description automatically generated***

Figure 3.4.2

***Upload PDF***

***A diagram of a correct format

Description automatically generated***

Figure 3.4.3

***Malware Analysis***

A diagram of a software analysis

Description automatically generated

Figure 3.4.4

***Overall Activity Diagram (up to Current iteration):***

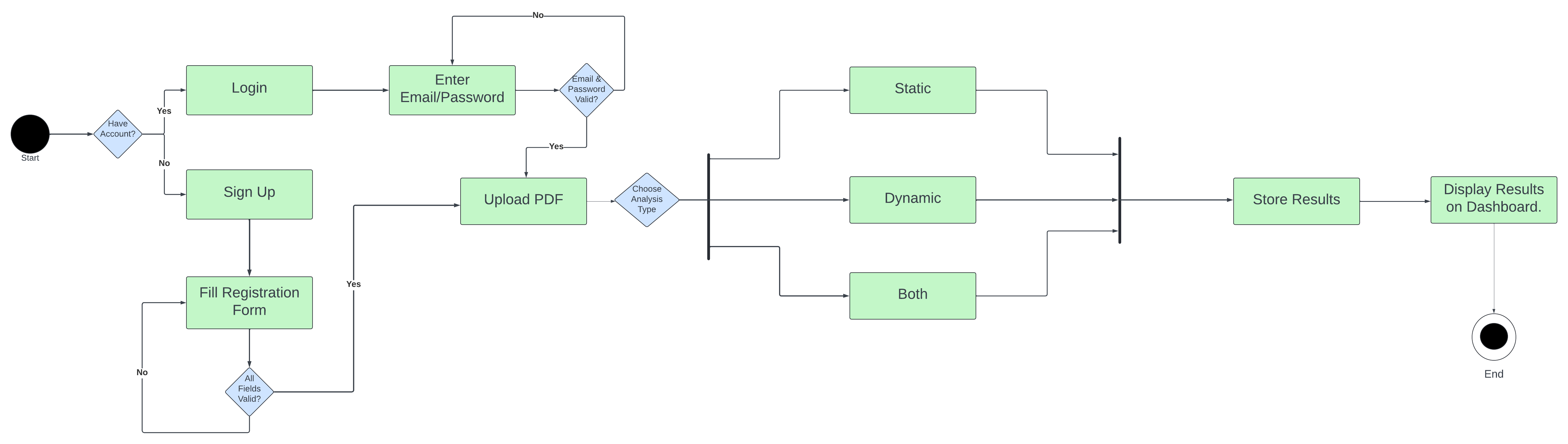


Figure 3.4.5

### Sequence Diagram

The sequence diagram is based on the communications and interactions between the different system components and user.

***Full Sequence Diagram:***

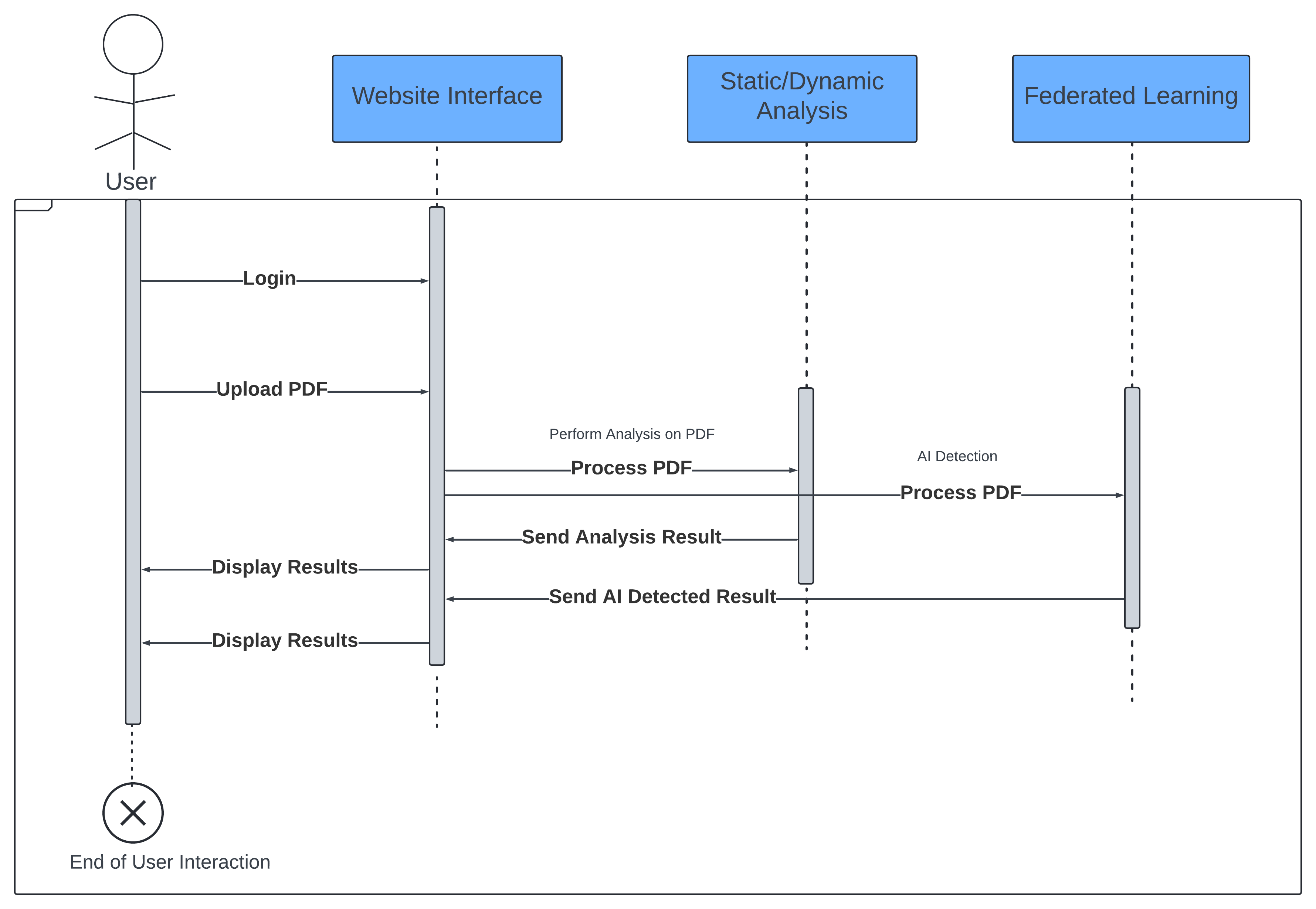


Figure 3.4.6

***Upto Current Iteration:***

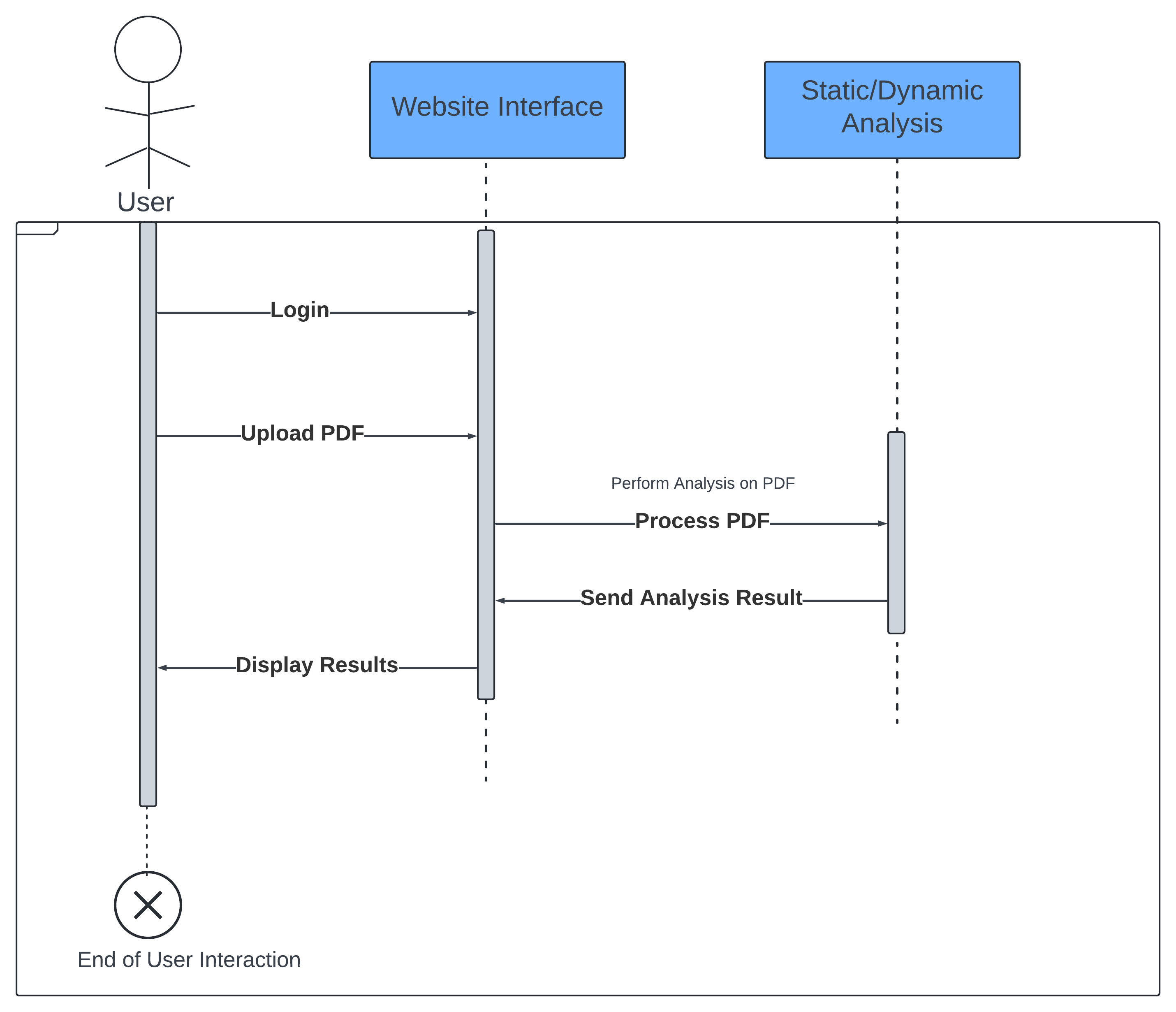


Figure 3.4.7

### State Diagram

A state transition diagram is a graphical representation that shows the different states of a system and the transitions between them. In our malware analysis system, there are various states the system transitions through, such as file upload, malware analysis, and result visualization. Therefore, we have chosen a state transition diagram to represent these processes. The diagram shows how the system moves from the "PDF Upload" state to "Malware Analysis," and depending on success or failure, transitions to either storing results or prompting the user to re-upload. Successful analysis also triggers the training of local and global models through federated learning, enhancing overall system detection capabilities.

***Full State Transition Diagram:***

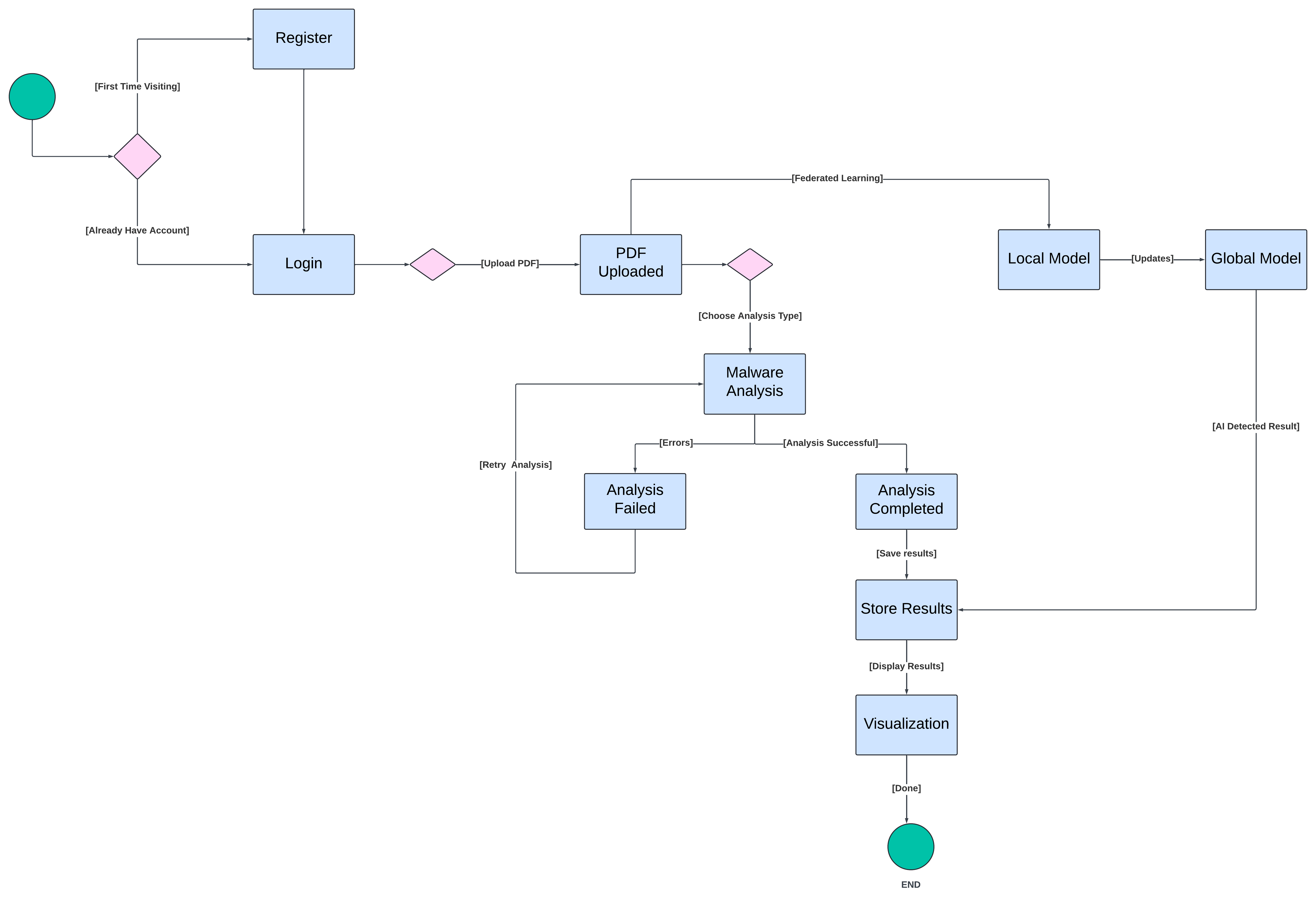


Figure 3.4.8

***Upto Current Iteration:***

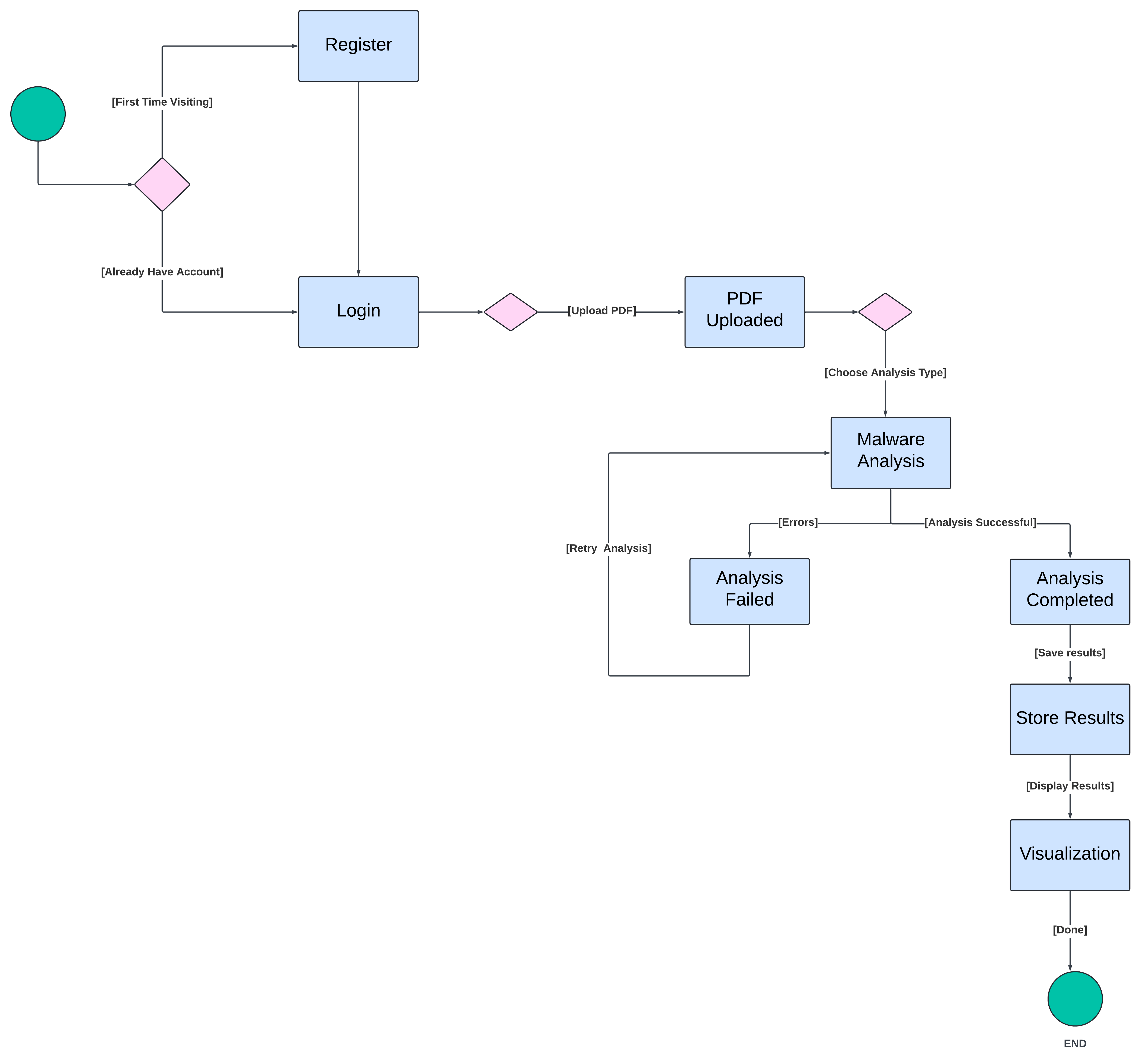


Figure 3.4.9

### Data Flow Diagram

In our malware analysis system, the focus is on the flow of data and the interactions between different components, such as the user, Sandbox API, Elasticsearch, and the admin. Therefore, we have chosen a DFD (Data Flow Diagram) to visualize this aspect of our system. The DFD shows the flow of data in the malware analysis process, where a user uploads a PDF file, the analysis is done via the Sandbox API, and the results are stored in Elasticsearch, with the admin managing system updates and requests.

***Context Diagram:***

A diagram of a computer

Description automatically generated

Figure 3.4.10

***DFD Level 0:***

A diagram of a computer system

Description automatically generated

Figure 3.4.11

***DFD level 1:***

The Level 1 DFD illustrates the detailed interactions between the user, admin, and various system components like the Sandbox API and Elasticsearch. It shows how a user uploads a PDF for malware analysis, the results are processed, stored, and displayed, while the admin manages system access and updates.

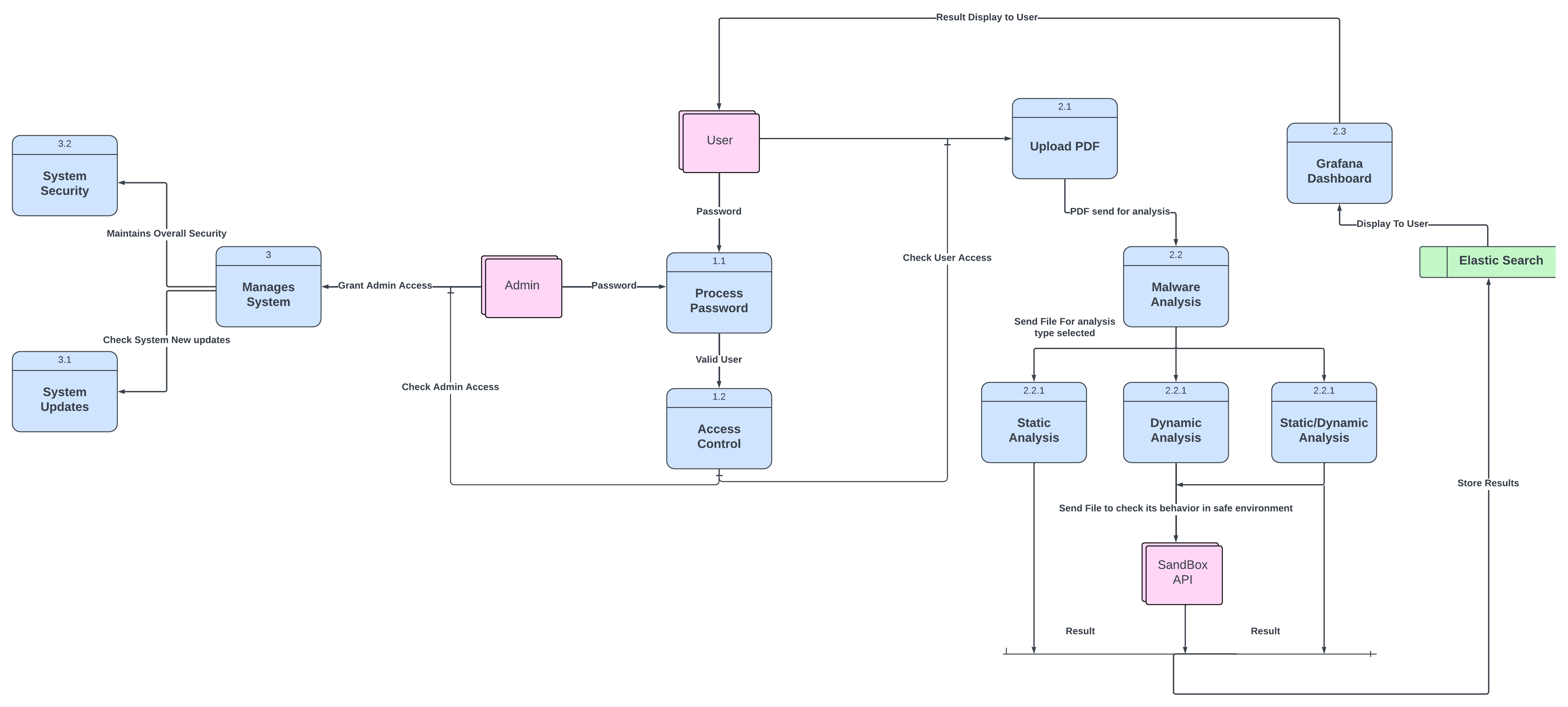


Figure 3.4.12

***DFD Level 2:***

The **Level 2 DFD** provides a more detailed breakdown of system processes. It focuses on internal operations, such as **password processing**, **access control**, and how the **admin manages the system**. It also shows interactions like **checking for system updates**, monitoring the **performance of the tool**, and ensuring **system security**. Additionally, there are processes for handling **privacy settings for sharing models** and managing updates related to the **AI model training** and **new PDFs added for analysis**.

Further details are the user's interactions within the malware detection system. It shows how the **user uploads a PDF** for analysis, which then goes through both **static** and **dynamic analysis**. The **Sandbox API** is utilized for dynamic analysis, while static analysis is conducted internally. The results from both analyses are sent to the **Elastic Search** component for processing, and then displayed on the **Grafana Dashboard** for the user. The diagram also includes processes like **federated learning**, **local model training**, and **global model training**, which ensure the system continually improves its detection capabilities.

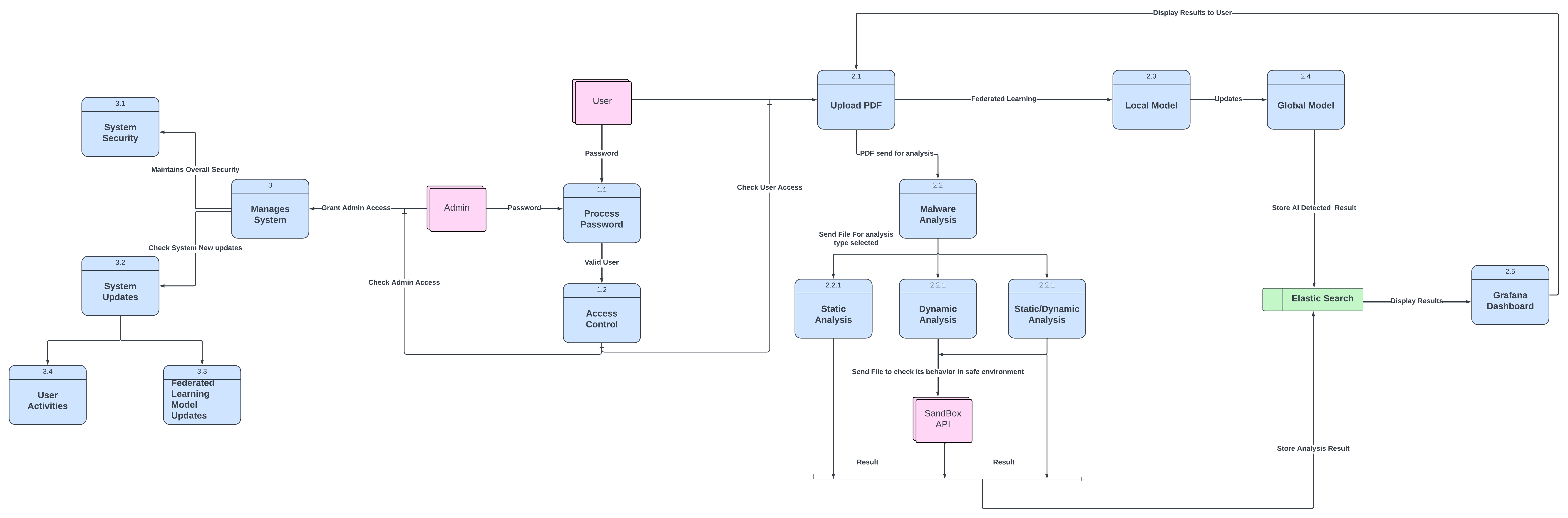


Figure 3.4.13

Chapter 4

# Implementation and testing

**General Description:**

The current version of the project focuses on the creation of an integrated malware detection system designed to scan PDF files with static and dynamic analysis approaches within a unified framework. It reduces the manual effort that a malware analyst expends on detection. The analysis results are presented through a friendly visualization dashboard, providing rich insights into suspicious PDFs. Another aspect of the system is ensuring the centralised and distributed storage of data, thus allowing for scalable and efficient management of data. Although federated learning and administrative functionalities are yet planned for the future, the implemented components form the strong foundation for an efficient and accurate detection solution of PDF malware.

**Context and Background:**

PDFs have become one of the common channels for the propagation of malicious payloads because they are widely in use and support the embedding of scripts and media. Tools such as static analysis or dynamic analysis, if done in isolation, are incomplete in their results because of which the proposed system incorporates both in one automated framework to merge the two analysis. The AI component further increases detection capabilities by adapting it to emerging threats.

**Design of Project:**

* Frontend: A React-based web portal allows users to upload PDF files for analysis. The interface is dark-themed and intuitive, providing easy navigation and access to functionalities.
* Backend: A Flask-based backend handles file uploads, analysis, and interaction with external tools such as Sandbox. It processes files sequentially, ensuring robust and error-free feature extraction.
* Data Storage: Analysis results and extracted features are stored in Elasticsearch, ensuring scalability and high-speed querying.
* Visualization: Grafana dashboards are customized to display results in a detailed yet comprehensible format.
* AI Model: Machine Learning Model has been trained using extracted features from malicious and benign PDFs to classify files with high accuracy.

## Algorithm Design

The below written algorithms represent the functionality completed up to the current iteration, covering the entire flow from static and dynamic analysis to data visualization and user interaction through the front end. The remaining work includes federated learning and admin functionalities.

**1. Static Analysis**

The static analysis algorithm extracts key features from PDF files without executing them, analyzing metadata, structure, and potential malicious indicators.

**Pseudocode: Static Analysis**

A screenshot of a computer program

Description automatically generated

**2. Dynamic Analysis**

Dynamic analysis involves executing the PDF in a sandbox environment to observe its behavior, such as API calls and network activity.

**Pseudocode: Dynamic Analysis**

A computer screen shot of white text

Description automatically generated

**3. Visualization**

The visualization module uses Grafana to display static and dynamic analysis results, providing a user-friendly dashboard for malware analysts.

**Pseudocode: Visualization**

A screen shot of a computer

Description automatically generated

**4. Front-End Functionalities**

The front end allows users to sign up, log in, edit/view their profiles, and upload PDFs for analysis.

**Pseudocode: Front-End User Interaction**

A screenshot of a computer

Description automatically generated

## External API

Here’s a table describing the third-party APIs and SDKs used in our project upto current iteration:

|  |  |  |  |
| --- | --- | --- | --- |
| **API and version** | **Description** | **Purpose of usage** | **API endpoint/function/class used** |
| PyPDF2 (v3.0.1) | Python library for PDF file handling | Parsing PDF files and extracting Features | PdfFileReader |
| Elasticsearch (v8.15.3) | Search and analytics engine | Storing and querying analysis results | index, search, update |
| Flacon Sandbox API | Malware analysis sandbox | Performing dynamic analysis of PDFs | /tasks/create/file, /tasks/report |
| Grafana API (v11.1.4) | Visualization and monitoring tool | Creating real-time dashboards for analytics | /dashboards/db, /datasources |

Table 4.2.1

## Testing details

Testing ensures that each component of the system functions as intended, identifying issues and verifying the system's functionality at a granular level. Below is a breakdown of the unit testing conducted so far.

### Unit Testing

Unit testing isolates individual functions or methods to verify their functionality, ensuring that they work as intended. Each test checks a specific behavior and compares the actual output to the expected results.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Test Objective** | **Precondition** | **Steps** | **Test Data** | **Expected Result** | **Postcondition** | **Actual Result** | |  | | --- | |  |  |  | | --- | | **Pass/Fail** | |
| TC001 | Verify static feature extraction | A valid PDF file is uploaded. | 1. Upload a sample PDF file.  2. Execute extract\_static\_features(file). 3. Observe output. | Sample PDF file | Static features are extracted successfully as a dictionary. | Features are stored in the database. | |  | | --- | | As Expected, |  |  | | --- | |  | | Pass |

Table 4.3.1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Test Objective** | **Precondition** | **Steps** | **Test Data** | **Expected Result** | **Postcondition** | **Actual Result** | **Pass/Fail** |
| TC002 | Verify dynamic analysis function | A valid malicious or benign PDF file is uploaded | 1. Upload a sample PDF file.  2. Execute perform\_dynamic\_analysis(file). 3. Observe sandbox behavior and output. | Sample PDF file | Sandbox behavior logs and analysis report are generated | Logs are stored in the database. | As Expected.   |  | | --- | |  | | Pass |

Table 4.3.2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Test Objective** | **Precondition** | **Steps** | **Test Data** | **Expected Result** | **Postcondition** | **Actual Result** | **Pass/Fail** |
| |  | | --- | | TC003 |  |  | | --- | |  | | Verify login functionality | User is registered in the system | 1. Open login page.  2. Enter valid credentials.  3. Submit login form. 4. Verify authentication message and redirection. | Email: user@domain.com Password: password123 | User is logged in and redirected to the main page. | User session is maintained. | |  | | --- | | As Expected. |  |  | | --- | |  | | Pass |

Table 4.3.3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Test Objective** | **Precondition** | **Steps** | **Test Data** | **Expected Result** | **Postcondition** | **Actual Result** | **Pass/Fail** |
| TC004   |  | | --- | |  | | Verify file upload functionality | User is logged in. | 1. Log in as user. 2. Navigate to the file upload page. 3. Upload a valid/invalid PDF file. 4. Observe validation results and status message. | Valid/invalid file format | Valid files are accepted; invalid files raise errors. | Uploaded files are processed or rejected. | |  | | --- | | As Expected |  |  | | --- | |  | | Pass |

Table 4.3.4

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **Test Objective** | **Precondition** | **Steps** | **Test Data** | **Expected Result** | **Postcondition** | **Actual Result** | |  | | --- | |  |   **Pass/Fail**   |  | | --- | |  | |
| TC005 | Verify visualization rendering | Analysis results are stored in the database | 1. Perform a static or dynamic analysis. 2. Execute generate\_visualization(data). 3. Observe the graphs displayed on the screen. | Sample JSON data | Accurate Analysis are displayed. | Visualization updates dynamically on the screen. | As Expected | Pass |

Table 4.3.5

Bibliography

### 