Within the domain of Data Analytics (DA), there is an important field known as Machine Learning (ML), which is embedded in everyday people's lives. A significant topic within this field is Federated Learning (FL). FL occurs when different devices collaborate to build a common model without exchanging their data; instead, the data remains on the devices, and only model updates are sent to a central server where aggregation occurs.

A good example for FL in everyday people’s lives is when phone users are typing a message, and the keyboard predicts words to complete the sentence, a model developed by Google engineers.

A different example where FL is helping society move forward is in the healthcare sector. Patients are using wearable devices to track their movements and help doctors monitor their health conditions.

FL is growing rapidly and is helping the technological and medical sectors build robust machine learning models. This significant development motivates the focus of this thesis.

The primary purpose of this project is to experiment with FL frameworks to evaluate their implementability and develop a functional FL server. Therefore, the research objectives are:

* **To evaluate the implementability of existing FL frameworks.** This section experiments with popular FL frameworks, such as *TensorFlow Federated* (TFF)*, Flower, EasyFL, IBM Federated* (IBMF)and *FLGo*, by examining their architecture and their applicability to real-world FL scenarios. This evaluation serves as the starting point for the primary research.
* **To develop a FL server**. A practical example of a web Flask FL server will be built, featuring two scenarios: technological and pharmaceutical. Each scenario will run separately, connecting five clients. Each scenario will have its own data, synthetic data for the technological scenario and images for the pharmaceutical scenario. These configurations will be trained, validated, and tested using Neural Networks (NN) that will classify binary outputs. This experiment aims to bridge the gap between popular FL frameworks and real-world FL applications.
* **To compare FL frameworks and the FL server.** This includes discussing the pros and cons each method brings to key industries such as the technological and pharmaceutical sectors. The comparison will address aspects like ease of implementation, scalability, data privacy, and model performance.

This concept was introduced in 2016 by Google engineers (McMahan et al., 2016). FL is a shared model that is trained across multiple devices, often referred to as clients. Each client trains its own local model and sends the updates to a central server, where the updates are aggregated to improve the global model. It is important to note that clients do not exchange their data; the data remains private for each client. This principle drove the design of FL, following the concepts of focused collection or data minimization, which were introduced by the White House in 2013. The intent is to prevent personal data from being sent over the network and potentially being stolen or manipulated by malicious third parties.

Depending on the nature of client FL can be classified in two types cross-device and cross-silo

The clients for cross-device can be mobile devices, edge devices, Internet of Things (IoT) devices, smartphones, tablets, wearables, etc. Figure 2.2.1. illustrates this scenario.  
The characteristics are, the high number of participants it can be thousands to millions of devices, it may have limited processing power and battery life, datasets tend to be small and network bandwidth may be limited. Devices may also connect and disconnect intermittently.

In this scenario, clients can be organizations or institutions such as hospitals, banks, and companies using large data centers. Figure 2.2.2 illustrates a cross-silo ecosystem. Some differences compared to the cross-device scenario include, clients are no longer small devices, there are fewer clients, clients have high computational power and large datasets, and the network is reliable with stable communication.

FL can be categorized according to the distribution of the data held by the clients participating in the modelling. These categories help to understand the different methodologies and use cases for FL.

Horizontal Federated Learning (HFL) or sample-based FL, occurs when different clients have datasets that share the same feature space but differ in the samples they hold (see Figure 2.3.1). A practical example of HFL is when two hospitals in different regions each have patient records with the same features (e.g., age, height, weight, diagnosis) but for different patients. These hospitals can collaborate to train a model to predict disease outcomes without sharing patient data.

Vertical Federated Learning (VFL), or feature-based FL, occurs when different clients have datasets that share the same sample IDs but differ in the feature space. An example of this is a bank and an e-commerce company that have data on the same set of customers. The bank has financial information such as credit scores and loan histories, while the e-commerce company has purchase behaviours and browsing records. By combining their data, they can build a model to predict customer credit without sharing raw data.

Federated Transfer Learning (FTL) is applicable when the datasets of different clients differ in both samples and features, and the overlap between both is minimal. To clarify, a good example is a European pharmaceutical company and a healthcare research institution in China collaborating using FTL. The pharmaceutical company has drug efficacy data, while the healthcare institution has patient health records. Even though they have different types of data, it is possible to train a global model to predict drug effectiveness on certain health conditions.

Terms like FL and Distributed Machine Learning (DML) can create confusion due to their similarities. The main difference lies in the training process: in FL, there is a central server that aggregates updates sent by the clients, whereas in DML, there is no central server; instead, data is spread across different nodes and computations are shared among these nodes. Table 2.4 will help clarify the intricacies of each concept.