# ST. XAVIER'S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu

**FINAL YEAR PROJECT REPORT**

**ON**

# "ML Store: Decentralized Machine Learning Platform"

**(CSC - 404)**

A Final Year Project Report submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

**Under the supervision of**

**Er. Rajan Karmacharya**

**Department of Computer Science**

**Submitted By:**

**Bibhuti Poudyal (7127/072)**

**Rohan Shrestha (7148/072)**

**Roshan Chapagain (7149/072)**

**Submitted To:**

**ST. XAVIER'S COLLEGE**

**Department of Computer Science**

**Maitighar, Kathmandu, Nepal**

**June 25, 2019**

# ML Store: Decentralized Machine Learning Platform

**(CSC - 404)**

A Final Year Project Report submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

**Submitted By:**

Bibhuti Poudyal (T.U. Exam Roll no.: 7127/072)
Rohan Shrestha (T.U. Exam Roll no.: 7148/072)
Roshan Chapagain (T.U. Exam Roll no.: 7149/072)

**Submitted To:**

**ST. XAVIER'S COLLEGE**
**Department of Computer Science**
**Maitighar, Kathmandu, Nepal**

**June 25, 2019**

**ST. XAVIER'S COLLEGE**
MAITIGHAR, KATHMANDU, NEPAL
Post Box :7437
Contact: 4221365,4244636
Email: ktm@sxc.edu.np

**सेन्ट जेभियर्स कलेज**
माईतीघर, काठमाडौं, नेपाल
पो.ब.नं. : ७४३७
फोन : ४२२१३६५,४२४४६३६
ईमेल : ktm@sxc.edu.np

# CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the Department of Computer Science for acceptance, a project proposal entitled "ML Store: Decentralized Machine Learning Platform" submitted by **Bibhuti Poudyal (7127/072), Rohan Shrestha (7148/072), Roshan Chapagain (7149/072)** For the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University.

…………………………..

**Er. Rajan Karmacharya**
Supervisor /Lecturer
St. Xavier's College

…………………………….

**External Examiner**
Tribhuvan University

……………………………..

**Mr. Jeetendra Manandhar**
Head of the Department
Department of Computer Science
St. Xavier's College

# ACKNOWLEDGEMENT

This project could not have been successful if it were not for all the people who provide us with their inspiration and support. We are indebted to all their help and wish to thank all of them.

Firstly, we would like to thank Er. Rajan Karmacharya, Department of Computer Science, for his guidance and supervision throughout the course of this project, without which we could not have imagined to complete this project to fruition.

We would also like to thank Mr. Jeetendra Manandhar, Head of Department of Department of Computer Science, for his insightful suggestions and support for this project as well as for providing us with the necessary logistics.

Our gratitude also extends towards our friends, seniors and families for their invaluable encouragement and feedback. We also want to give thanks to Github, Stack Overflow and Tensorflow community for their support throughout the experimental development and research work. Similarly, we extend our sincere thanks to all other seen and unseen personalities involved in the preparation of the project work.

We look forward to any further feedback or comments regarding this work. Suggestions for further improvement will be much appreciated.

Bibhuti Poudyal (T.U. Exam Roll no.: 7127/072)
Rohan Shrestha (T.U. Exam Roll no.: 7148/072)
Roshan Chapagain (T.U. Exam Roll no.: 7149/072)

# ABSTRACT

This project is an attempt at implementing the concept of decentralization in the context of machine learning. Considering how security and sensitivity of private user data has become a prime concern, the project aimed to utilize the concepts of federated learning and decentralized machine learning (DML) to find a solution to train machine learning models that would respect those needs. To that end, this project involved making a web platform to share and train machine learning models with on-device machine learning without the need to upload private user data to a central server.

In this project, a machine learning engineer would upload their machine learning models they would like to have trained. Instead of the data being provided to the model in the server from various sources, the model itself is distributed among various participating users in the platform. The users would voluntarily train the model in their own devices using the data within their devices; the user in return gets incentivized for their participation. The updates to the model is sent back to the server as new weight values for the model. Updates from all the different users are averaged to generate a newly trained version of the model which then can further be distributed for further training or be used by the engineers for deployment in their own applications.

This report is the culmination of the research and development that was involved for the completion of this project.

**Keywords:** Machine Learning, Decentralization, Data Privacy, Data Security, Decentralized Machine Learning, Federated Learning, On-device Machine Learning

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| DML | Decentralized Machine Learning |
| IOT | Internet Of Things |
| JS | JavaScript |
| JSON | JavaScript Object Notation |
| ML | Machine Learning |
| SOA | Service Oriented Architecture |
| TF | TensorFlow |

# CHAPTER 1: INTRODUCTION

## 1.1 Overview

Artificial intelligence has been rapidly moving forward to shape the future of the world we live in. The main driving force behind this revolution has been the increasing adoption of machine learning, which leverages the extensive amount of data generated by users to improve the software without need of them being explicitly programmed. The quality and relevance of the data are essential for generating machine learning models of high quality and accuracy. Although most of the user data is publicly available, a lot of the data specific to the user remains untapped [1].

The workflow suggested in this proposal is designed to expand the reach of machine learning models to the untapped private data and unleash their potential to better adapt such models for every user while respecting data privacy and also providing economic incentives to the developers or businesses. Machine learning algorithm will operate within the users' devices without their data being sent to a server or any outside entities. Only the results from the computation based on their data is sent to the server. From there, the machine learning model will be aggregated with outcomes generated from every other device to form an unbiased, comprehensive and accurate analysis and prediction, to inevitably generate a better model. Through this approach, both the private data and processing power for machine learning are decentralized as algorithms are run directly on individual devices by utilizing their idle processing power.

This project idea is for such machine learning models in which the training data is too large to be feasibly collected centrally or the data is too sensitive to be shared by the users. A  marketplace is used for crowdsourcing various machine learning models bundled with a script and a user interface that allows users to train the model via a web browser within their own devices, incorporating the use of federated learning [2]. The trained model is then sent from the users' devices back to the server to be aggregated along with trained models uploaded by other users. The main model is then improved accordingly.

## 1.2 Background

### 1.2.1 Machine Learning

Machine learning is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task.

Machine learning has been rapidly moving forward to shape the future of the world we live in. The main driving force behind this revolution has been the increasing adoption of machine learning, which leverages the extensive amount of data generated by users to improve the software without need of them being explicitly programmed. The quality and relevance of the data are essential for generating machine learning models of high quality and accuracy. Although most of the user data is publicly available, a lot of the data specific to the user remains untapped [1].

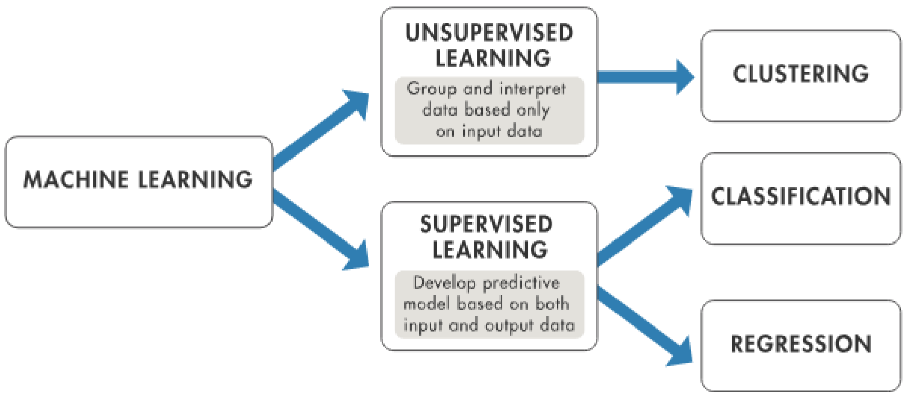

MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

**Figure 1.1: Different Machine Learning approaches [3]**

Training a machine learning model with terabytes to petabytes of data using very deep neural networks doesn’t scale well within a single machine. A significant amount of work in recent years has gone into distributing the training of such neural networks across a cluster of machines, by partitioning on both the data and the model itself. The most well-established form of distributed training uses a centralized parameter server to manage the shared state of neural network weights used across all partitions of the data, but this introduces a bottleneck and single-point of failure during training [4].

### 1.2.2 Federated Learning

Federated Learning is a machine learning approach which enables remote devices to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices (like the Mobile Vision API and On-Device Smart Reply) by bringing model training to the device as well [2].

It works like this: the device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on the device, and no individual updates are stored in the cloud [2].

### 1.2.3 Decentralized Machine Learning

Decentralized machine learning refers to an approach of machine learning in which the training process is carried out in several smaller units with their own data, rather than a single server holding all data and carrying out the sole task to training and testing. With such approach load on a single server can be reduced and the privacy of data can be maintained. Rather than taking the data to models, it brings model to data. It aims to connect and leverage the idle processing power of individual devices for machine learning [5].

### 1.2.4 Machine Learning Frameworks

Working on building Machine Learning models from scratch can be very difficult and tedious. However, we have in our disposal myriad of frameworks that allow us to develop tools that can offer a better level of abstraction along with simplification of difficult programming challenges.

Some of the popular machine learning frameworks are listed below:

**TensorFlow:** Perhaps the most popular of Machine Learning Frameworks, TensorFlow is available on both desktop and mobile and also supports languages such as Python, C++ and R to create deep learning models.

**Caffe:** Caffe is a machine learning framework that is supported with interfaces like C,C++, Python, MATLAB as well as CLI. It is well known for its speed and transportability in modelling CNN. The biggest benefit however of using Caffe is getting access to great numbers of pre trained and can be used immediately.

**Microsoft Cognitive Toolkit (CNTK):** It is, popularly known for easy training and combination of popular model types across servers. It is very similar to Caffe in terms of supported interfaces.

**Keras:** Keras framework is well known for being minimalist and user friendly. The Keras neural networks library supports both convolutional and recurrent networks that are capable of running on either TensorFlow or Theano. The library is written in Python and was developed specially for quick experimentation.

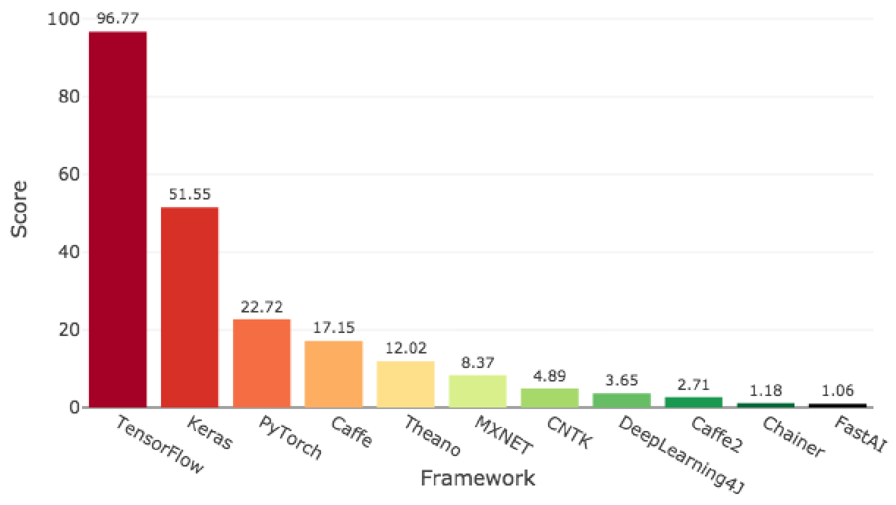


**Figure 1.2: Deep Learning Framework Power Scores 2018 [6]**

**Choice of Machine Learning Framework**

Through the research and experimentation, TensorFlow framework was found to meet all the constraints and be the perfect candidate for the purpose of the project. [7].

Apart from this, TensorFlow is based on Python, which is very popular for its great learnability and community, supported by Google and comes with clear and descriptive documentation and walkthroughs.

But the main reason for choosing TensorFlow was because it was the only framework that provided ways to implement federated learning. In fact, most of the recent previous research on federated learning had been done by the people behind TensorFlow. TensorFlow Federated was announced officially at TensorFlow Dev Summit 2019, which made the libraries necessary for implementing federated learning public. [8], [9].

### 1.2.5 TensorFlow

TensorFlow is a flexible end-to-end open source platform for machine learning developed by the Google Brain team. It is equipped with comprehensive ecosystem of tools, libraries and community resources for both researchers and developers. It runs across all devices like PCs, servers, mobile, web browser and even in IOT devices or microcontrollers and provides API in major popular languages . [10]

TensorFlow makes use of various machine learning tools along with Artificial Neural Networks to solve many real world problems. TensorFlow helps its users solve complicated problems like translating medieval text as well as smaller simpler problems like aiding small companies with inventory management [11].

#### 1.2.3.1 TensorFlow Federated

TensorFlow announced TensorFlow Federated at TensorFlow Dev Summit 2019, which is an open-source framework for machine learning and decentralized data. TensorFlow Federated is especially developed to facilitate open research and experimentation with Federated Learning. “TensorFlow Federated” not only enables machine learning but also provides means for implementing computations, such as aggregated analytics over decentralized data [9],[12].

#### 1.2.3.2 TensorFlow Federated Architecture

TensorFlow Federated is basically divided into two layers:

**Federated Learning (FL) API:** This layer offers a set of high-level interfaces that allow developers to apply included implementations of federated training and evaluation to their existing TensorFlow models.

**Federated Core (FC) API:** The core layer consists of a set of lower-level interfaces for concisely expressing novel federated algorithms by combining TensorFlow with distributed communication operators within a strongly-typed functional programming environment [13].

### 1.2.3.3 TensorFlow Federated Uses

TensorFlow Federated has found its use in character recognition. The most famous image dataset: NIST contains images of 810,000 handwritten digits, collected from over 3,600 volunteers, say our goal is to build an ML model that will recognize the digits [14]. One method (also the traditional one) would be to apply an ML algorithm to the entire dataset at once. However, for some reason the data could not be combined because the volunteers did not agree to upload their raw data to a central server. TensorFlow Federated would be best suited for this goal.

With TensorFlow Federated, the ML model architecture of choice can be represented easily, and then trained across the data provided by all the volunteers, while successfully keeping each writer's data separate, local and private [15].

TensorFlow Federated is also helpful with is performing various computations over a decentralized dataset. For instance, computing average temperature reading from array of sensors capturing temperature readings, without uploading their data to a central location.

This might be compared to a function with has inputs and outputs that reside in different places.

### 1.2.4 Service-oriented architecture

Service-oriented architecture has been adopted to run complex machine learning on the web. It is a style of software design where services are provided to the other components by application components, through a communication protocol over a network. The basic principles of service-oriented architectures are independent of vendors, products and technologies. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently, such as retrieving a credit card statement online [16].

The workflow suggested in this proposal is designed to expand the reach of machine learning models to the untapped private data and unleash their potential to better adapt such models for every user while respecting data privacy and also providing economic incentives to the developers or businesses. Machine learning algorithm will operate within the users' devices without their data being sent to a server or any outside entities. Only the results from the computation based on their data is sent to the server. From there, the machine learning model will be aggregated with outcomes generated from every other device to form an unbiased, comprehensive and accurate analysis and prediction, to inevitably generate a better model. Through this approach, both the private data and processing power for machine learning are decentralized as algorithms are run directly on individual devices by utilizing their idle processing power.

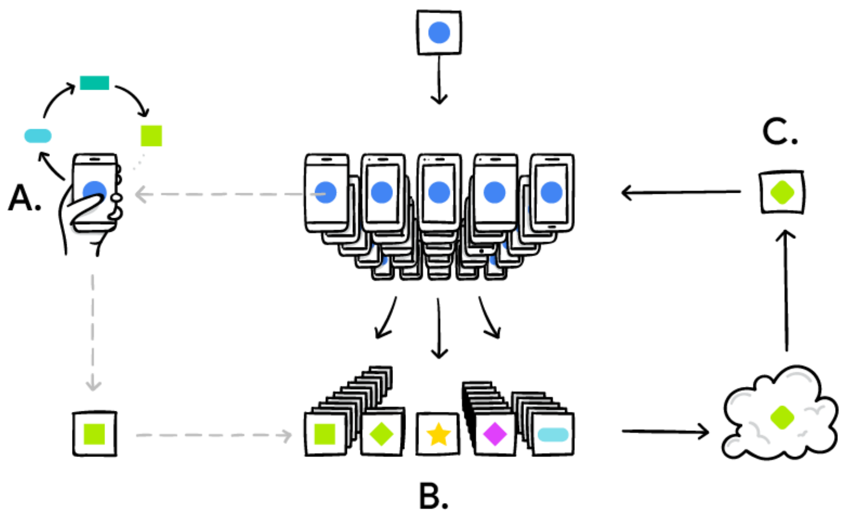


**Figure 1.3: Federated Learning working mechanism [2]**

### 1.2.4 Plug-in System

In computing, a plug-in is a software component that adds a specific feature to an existing computer program. When a program supports plugins, it enables customization. The architecture of the application is designed to work like a plugin system. While training a model, HTML, CSS and JS are loaded from server and gets embedded in the application. The Html provides basic layout for user interface through which users would input training data. Css applies styling to the layout and makes it more user friendly. Javascript is used for making the interface interactive. And the training and validation logic are also written in javascript so that they can be run on web browsers [17].

This project idea is for such machine learning models in which the training data is too large to be feasibly collected centrally or the data is too sensitive to be shared by the users. A  marketplace is used for crowdsourcing various machine learning models bundled with a script and a user interface that allows users to train the model via a web browser within their own devices, incorporating the use of federated learning [2]. The trained model is then sent from the users' devices back to the server to be aggregated along with trained models uploaded by other users. The main model is then improved accordingly.

## 1.3 Problem Statement

Private data held by users are more relevant and accurate as compared to publicly accessible data which are collected and processed for specific needs. It is because such data is more specific to the user and not a generalization observed in most publicly available data. Such private data are never exposed to the machine learning model as they are stored in individual electronic devices such as smartphones, tablets and computers. A main reason for such isolation is the privacy concerns surrounding the access of such data. Large technology firms have come under scrutiny over the years for accessing users' personal data without their awareness or consent [18]. Incidents such as these have further complicated the issues surrounding the collection of such data, even if the methods involved respect the privacy of the users. As a result, most of these private data have mostly been cut off from use in any kind of data analytics or computation. If machine learning models were provided access to such data, it would be possible to tailor such models to suit each user, resulting in outputs that will be more relevant to the user .

## 1.4 Project Objectives and Scope

The main objectives of the project include:

- Develop a common platform where developers can share their machine learning models and users or businesses can utilize those models for their own purposes while also improving the models in the process.
- Utilize untapped private data of devices for on-device machine learning whilst protecting the privacy of users through the use of federated learning.

## 1.5 Significance of the Project

Through the implementation of this project, machine learning engineers and researchers can easily train their models without having to spend time and resources on expensive central server workstations. This project can also prove to be helpful for a wider adoption of machine learning in businesses which depend on users' data. Modern personal computers and mobile devices are now powerful enough to train several machine learning models on their own. The idle processing power of these devices collectively can help to provide an overall greater processing power to improve the algorithms. The implementation of on-device machine learning on the users' private data within their devices also means that their data privacy is not compromised. All of this may contribute to develop better machine learning algorithms which help to solve any number of complex or unknown problems.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Machine Learning

A group of researchers at IBM led by Arthur Samuel coined the term  “Machine learning” in 1959. As a scientific endeavour, machine learning is found to have grown out of the quest for artificial intelligence. Since then, researchers have attempted to approach the problem with various symbolic methods, as well as what were then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis. Different techniques and models like CNN, LSTM, Random Forest etc. have been used to approach various problems in different ways. Such algorithms are found to have been used in a wide variety of applications, such as email filtering, and computer vision, recommendation system, where it is infeasible to develop an algorithm of specific instructions for performing the task. But all there computations have been done at a central device or server where  both the data and control are centralized [19].

For training such models researchers have been found to gather huge amount of data from users, process them and train the models on their device. This approach is termed as centralized Machine Learning. It is called so because the data are gathered at a central location and that central body takes care of training the model. The trained model is then accessed by users via a web interface or APIs. Although a lot of research has gone into making Machine learning process more accurate, the problem of data privacy, value of users data, the load on the server etc. has been rarely addressed. The data collected for training a specific model is often found to be used for other models too, as data collection is a very time consuming job. The data processed for one model may not fulfill the requirements of other models. Moreover, the processed data is never as qualitative as the raw data collected directly from user.

### 2.1.1 Data Privacy in Machine Learning

The rise in the use of machine learning has largely been contributed by the wealth of data being generated by users around the world. Most of these data come from the users' online activity in their digital devices and also from various home automation devices which have seen a growing increase over the years. These data are utilized by different machine learning algorithms to generate a data analysis or assist decision-making, while also improving the algorithms themselves. For users, this means that their devices get more adapted based on their usage behaviour and they get results that are more relevant to them.

Although machine learning algorithms can provide users with results and predictions pertaining to them, it comes at the cost of having to give up their data privacy. Papernot et al. showed that there is usually a trade-off between the privacy or security and the precision of ML predictions in machine learning systems with finite capacity [20]. This may be due to the fact that machine learning algorithms require more data from the users, which are often privacy sensitive, to make more relevant decisions and precise predictions. When such data are used as training data in machine learning models, they can be extracted in a number of ways, which if successful means that the users' privacy is compromised. Ateniese et al. used statistical inference on the training dataset from a trained model to infer the actual training datasets [21]. Fredrikson et al. presented a model inversion attack which they showed could recover genomic information of a patient given that the attacker had access to the patient's stable medicine dosage record and the model used to generate it [22]. Vulnerabilities like these have induced a growing concern regarding the future of data privacy. Countermeasures for such vulnerabilities or the introduction of new techniques to facilitate machine learning whilst preserving user privacy maybe some of the options to deal with the problem.

### 2.1.2 Data Communication in Machine Learning

With the increase in the size of datasets, training the machine learning models requires more computational power. Most of the available machine learning models rely on a centralized training data on a single machine or a datacenter. When the clients providing the training data are limited to a slow or unreliable network, communication of such training data to the centralized model becomes a bottleneck for the overall efficiency of the model. In their survey of traditional machine learning techniques, Qiu et al. (2016) have found that most of them are inefficient and are not scalable enough for big data processing [23]. In order to cope with large volume of data comprising a multitude of data types with optimal speed and accuracy, they have suggested reinventing these techniques. More specifically, a more efficient way to deal with the communication of training data to the centralized models running on a server may be a good first step.

A number of studies have advocated the use of distributed or decentralized approaches to improve the overall efficiency of the machine learning algorithms. Lian et al. (2017) have shown that decentralized machine learning algorithm can outperform centralized algorithms in their comparative study of decentralized and centralized version of Parallel Stochastic Gradient Descent [24]. However, Khan et al. (2018) argue that such decentralized models only work well when the number of agents in consideration is small . They instead propose a centralized model with distributed optimization setup for multi-agent reinforcement learning [25].

### 2.1.3 Federated Learning for Communication Efficiency and preserving privacy

Federated Learning has been proposed as a solution to deal with the issues of training data communication and data privacy where the training data is distributed over a large number of clients while the model is centralized in the servers. McMahan et al. (2016) and Konečný et al. (2016) in their separate works on federated learning have shown that federated learning can help to reduce the rounds of communication in a variety of model architecture to improve the communication of training data and also preserve the privacy of user data [26], [27] . They have also suggested using differential privacy to guarantee stronger privacy.

### 2.1.4 Decentralized Systems and Decentralized Machine Learning

A decentralised system have been found to be a working mechanism in which lower level components operate on local information to accomplish global goals. The global pattern of behaviour is an emergent property of dynamical mechanisms that act upon local components, such as mutual communication, rather than the sole component giving orders in a centralized way. In such a decentralized system complex behaviour are found to emerge through the work of lower level components operating on local data i.e. data of users' devices, not the instructions of any centralized server that would demand the data and control. This was the conclusion by Resnick (1999) [28].

**Limitations of centralized system**

The International Data Corporation has foreseen that the industry's revenue will reach over USD 210 billion by 2020. Tech giants like Google, Apple, Facebook etc.are found to have invested heavily and gained remarkable achievements in machine learning. However, the current approach to machine learning development is seen to face many challenges, as mentioned by decentralized ML team on their blog [1]. Traditional machine learning techniques have been found to require datasets to be uploaded to a dedicated server. Due to privacy concerns, massive amount of private data stored in individual devices is found to be untapped.

Machine learning is mainly conducted through a centralized computer, which its processing power is usually limited or confined to the processors of a single machine. Only large corporations have been found to be able to afford huge initial capital and resources to build in-house machine learning models and algorithms, or acquire tailor-made ones from consultancy firms to apply machine learning in their own business, as mentioned in the DML whitepaper [29].

According to Addair (2015), training a machine learning model with terabytes to petabytes of data using very deep neural networks doesn't seem to scale well within a single machine. A significant amount of work in recent years has gone into distributing the training of such neural networks across a cluster of machines, by partitioning on both the data and the model itself. The most well-established form of distributed training have been found to use a centralized parameter server to manage the shared state of neural network weights used across all partitions of the data, but this introduces a bottleneck and single-point of failure during training

**Decentralized Machine Learning**

The DML white paper (2017) explores a more experimental form of decentralized training that removes this bottleneck. Finally, it concludes that by taking advantage of idle resources and data of users devices, decentralized training can be made more accessible, private and monetized by giving value to users data. What they propose is a community of machine learning enthusiasts who want to train their data with real world data i.e. such data that is not tampered and processes as usually found over the internet. And the users who want to get monetary value out of their daily collected data and idle CPU and memory. As the DML white paper concludes data privatization is found to be another major issue regarding machine learning. In a centralized model, users have to send their data to the central server to train the model.  How decentralized technique differs is: it trains the model on users devices using the private data and send the model updates to the server. The server then aggregates the updates from different users and updates the global model. Even if machine learning is decentralized, only the experts in this domain are found to get access the service. Lack of technical knowledge in business owners and data holders make them unable to access the service. This problem seems to emerge due to lack of a platform that brings machine learning engineers and data holders together [29].

A lot of research has been done to make the machine learning process decentralized. What these research lack is a way to make the process simpler and easily accessible. How can the process be made more lighter and portable. If someone is to train the same model across different devices like PC, mobile, IOT devices; what could be the best possible way?

According to McMahan et al. (2016), the federated learning algorithm powering the decentralized machine learning process also faces a lot of issues. The training data on a given client is typically based on the usage of the mobile device by a particular user, and hence any particular user's local dataset will not be representative of the population distribution. Similarly, some users will make much heavier use of the service or app than others, leading to varying amounts of local training data. They expect that the number of clients participating in an optimization to be much larger than the average number of examples per client. Mobile devices are frequently offline or on slow or expensive connections. Their emphasis is on the non-IID and unbalanced properties of the optimization, as well as the critical nature of the communication constraints. A deployed federated optimization system must also address a myriad of practical issues: client datasets that change as data is added and deleted; client availability that correlates with the local data distribution in complex ways (e.g., phones from speakers of American English will likely be plugged in at different times than speakers of British English); and clients that never respond or send corrupted updates [26].

## 2.2 Plugin System

Triglianos and Pautasso (2015) studied that JavaScript has become a language for programming complex web applications, whose logic is deployed across both Web browsers and Web servers. Current software packaging mechanisms for JavaScript enable a basic level of modularity and reuse. However, they have not yet reached full maturity in terms of enabling system extensions with features contributed as third-party plugins, while encapsulating them adequately.They present a novel plugin system for JavaScript applications, which integrate Node.js modules with HTML5 Web Components. It provides abstractions for: real time and loosely coupled communication between front-end and back-end components, persistent state storage, and isomorphic usage of JavaScript. Plugins can use hooks and events to contribute functionality and embed it into the main application flow, while respecting the common asynchronous non-blocking programming paradigm of JavaScript [30].

Any models and scripts before uploading to the server, need to be reviewed for vulnerability check. The plugin system have been found to have many security risks. Plugin-related vulnerabilities tend to be recurrent, exploitable and hard to be detected and may lead to severe consequences for the product, as studied by Mesa (2018) [31].

## 2.3 Application Architecture

Dustdar and Thanh (2005) studied Service-oriented architecture and found it to be based on the request/reply design paradigm for synchronous and asynchronous applications. An application's business logic or individual functions are modularized and presented as services for consumer/client applications. The key to these services is found to be their loosely coupled nature; i.e., the service interface is independent of the implementation. This architecture seems to be the best option for performing decentralized machine learning as all the user devices need to collaborate to operation the operation. The approach  tries to keep the architecture more modular and scalable. To balance the load complex operation is done at the separate server and authentication and data management at another one [32].

# CHAPTER 3: PROJECT DESIGN AND DEVELOPMENT

## 3.1 Project Management

Considering the research-oriented nature of the project, a strategy was employed which allowed for design and development of the project alongside the research that was required for it.

### 3.1.1 Project Team

**Table 3.1: Project Member and Roles**

| Team Member | Role |
|---|---|
| Er. Rajan Karmacharya | Supervisor |
| Bibhuti Poudyal | Developer/Team Manager |
| Rohan Shrestha | Developer/Designer |
| Roshan Chapagain | Developer/Designer |

The table above shows all the members associated with the design and development of the project. Project supervisor is responsible for the guidance and observance of the working methods and procedures of the project members from the start of the project. The project managers are required to prepare their design and develop the project according the necessary criteria and under the direct supervision of the supervisor.

### 3.1.2 Work Breakdown

The overall work of the project was broken down into several components which provided a more organized approach to managing the project. However, there was not a clear distinction between several work components because of the non-linear approach.

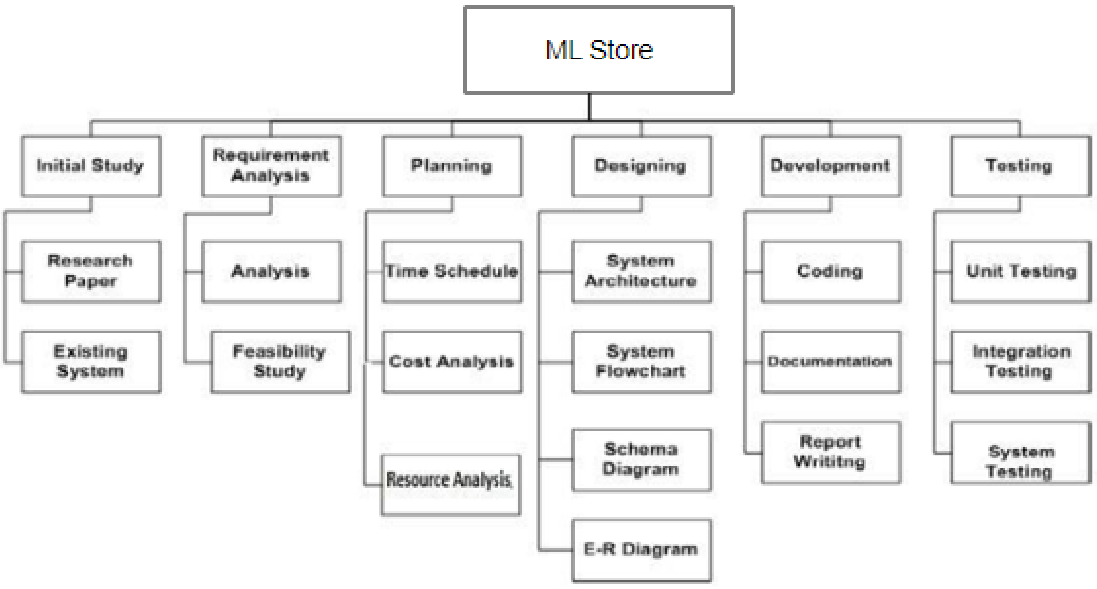


**Figure 3.1: Work Breakdown Structure**

### 3.1.3 Tools used for project management

The project relied on a number of tools that allowed the authors to collaborate on the various tasks online. A list of the tools are listed below:

**3.1.3.1 Trello:** Online scrum board; used to keep track of the list of tasks where each task were assigned for different members and separated on different lists based on its progress

**3.1.3.2 GitHub:** Online version control and code management; used for hosting the source code of the project for version control and code management

**3.1.3.3 G Suite:** For file sharing and documentation

**3.1.3.4 Slack:** For team communication and collaboration

## 3.2 Development tools and technology description

The project was divided into two components: the online store website and the federated learning component.

The tools used for the store are described below:

### 3.2.1 Vue.js

Vue.js is an open-source JavaScript framework used for web user interface design. It was used to design the user interface for the store platform.

The project uses Vue.js to construct the front end of the store. In addition, any user interaction with the project is also handled by Vue.js. Vue.js and Laravel interact closely and communicate back and forth to  handle all user interactions.

### 3.2.2 Laravel

Laravel is a PHP based web-framework for building high-end web applications using its significant and graceful syntaxes. It comes with a strong collection of tools and provides application architecture. Moreover, it includes various characteristics of technologies like ASP.NET MVC, CodeIgniter, Ruby on Rails and lot more. This framework is open source framework.

Laravel, in these project, has direct interaction with the database and thus is used to write Application Program Interfaces(APIs). Vue.js generally requests or stores data using these APIs written using Laravel.

The tools used for the federated learning component are described below:

### 3.2.3 TensorFlow

TensorFlow is an open-source library for machine learning research and production. The TensorFlow library along with its federated learning component, TensorFlow Federated, were used for implementing federated learning for model training.

TensorFlow in the project is used for Federated Learning. It has certain built-in functions which makes performing operations required for Federated Learning, like getting the weights of the model, setting the weights to the model, a lot easier than it is.

#### 3.2.4 Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is primarily used in this project for writing federated learning algorithm. It uses tensorflow-node to run averaging process over GPU. The primary reason to use Node.js is because it makes it possible to use same API throughout frontend training and averaging process.

## 3.3 System Analysis

System analysis is a process of collecting factual data, understanding the processes involved, identifying problems and recommending feasible suggestions for improving the system functioning. The major objectives of the system analysis phase is to find the answers for each business process: What is being done? How is it being done? Who is it being done by? When will it be done? Why is it being done and how can it be further improved?

A good analysis is immensely important for the development of an improved system.

### 3.3.1 Feasibility Study

Feasibility analysis is a thorough process of analysis and evaluation of a proposed project to ensure its, technical, economic, legal, operational feasibility. As the name suggests, a feasibility analysis is a study of the viability of an idea.

A feasibility study is performed before committing to a project and leads to a concrete decision whether the proposed project is fit to be initiated and completed. The feasibility study included following aspects:

#### 3.3.1.2 Technical Feasibility

Technical feasibility determines if the company or a group has technical expertise to complete the project. It also evaluates the hardware and software currently available to the group and verifies if they are sufficient for the completion of the project.

The project is divided into two chunks, one deals with the store where ML Engineers upload their ML models and the store users train the model, the other chunk is where the real magic happens, federated averaging server resides there, averaging the models weights and updating the weights of the models.

The first chunk (store) requires adequate knowledge in building web applications, with expertise in both front-end and back-end development. The second part required thorough understanding of Machine Learning, Federated Learning principles and TensorFlow.

The group was not new to building web apps and we managed quickly, to be acquainted with Machine Learning and TensorFlow. Also armed with modern computing devices, the project was found to be technically feasible.

### 3.3.1.3 Economic Feasibility

Economic feasibility analyzes the project’s cost and revenues in an effort to determine whether or not it is logical possible to complete.

The only resources spent during the development of the project were time, energy and knowledge. The only monetary cost would be on hosting the database and the Federated Averaging Server. On the client side, the store can be accessed and models trained on almost any device that can run a browser and store data, mobile phones, computers etc. Thus, the project’s cost is very low, which led to the conclusion that the project is economically feasible.

### 3.3.1.4 Legal Feasibility

Legal feasibility is the study to know if the proposed project conform the legal and ethical requirements.

The project does not violate any local or international laws. The project in no way misuses users data. Infact, one of the reasons the project was born was to ensure data privacy for training of machine learning models. Thus, the project is legally feasible.

### 3.3.1.5 Operational Feasibility

Operational feasibility refers to the measure of solving problems with the help of a new proposed system.

The project’s main goal is to make use of data stored on the user devices for training of machine learning models, which helps in creating a more powerful and capable machine learning models while also preserving user’s data privacy. The project effectively manages to do so by training the model locally on the user’s device using the available data and uploading the weights for the particular modal without transferring any other types of data from the user's device.

#### 3.3.1.6 Schedule Feasibility

Schedule feasibility is the degree to which a deadline for a strategy, plan, project or process is realistic and achievable.

With modern web app building methodologies and frameworks, along with increasing support from TensorFlow for federated learning, the project passes the test of "Schedule Feasibility".

## 3.4 System Design

The thorough design of the project addresses the requirements set in the initial stages of the application development.

Diagrams such as system architecture, data flow diagrams, use case diagrams, sequence diagrams, entity-relationship diagram and system flowchart were created in the design phase to make the development process a lot easier. An additional benefit of using this graphical representation is that it gives an overview of the system and its various structural and functional aspects.

However, the design part acts independently removing any association with the code that builds the project.

### 3.4.1 Overall System Workflow

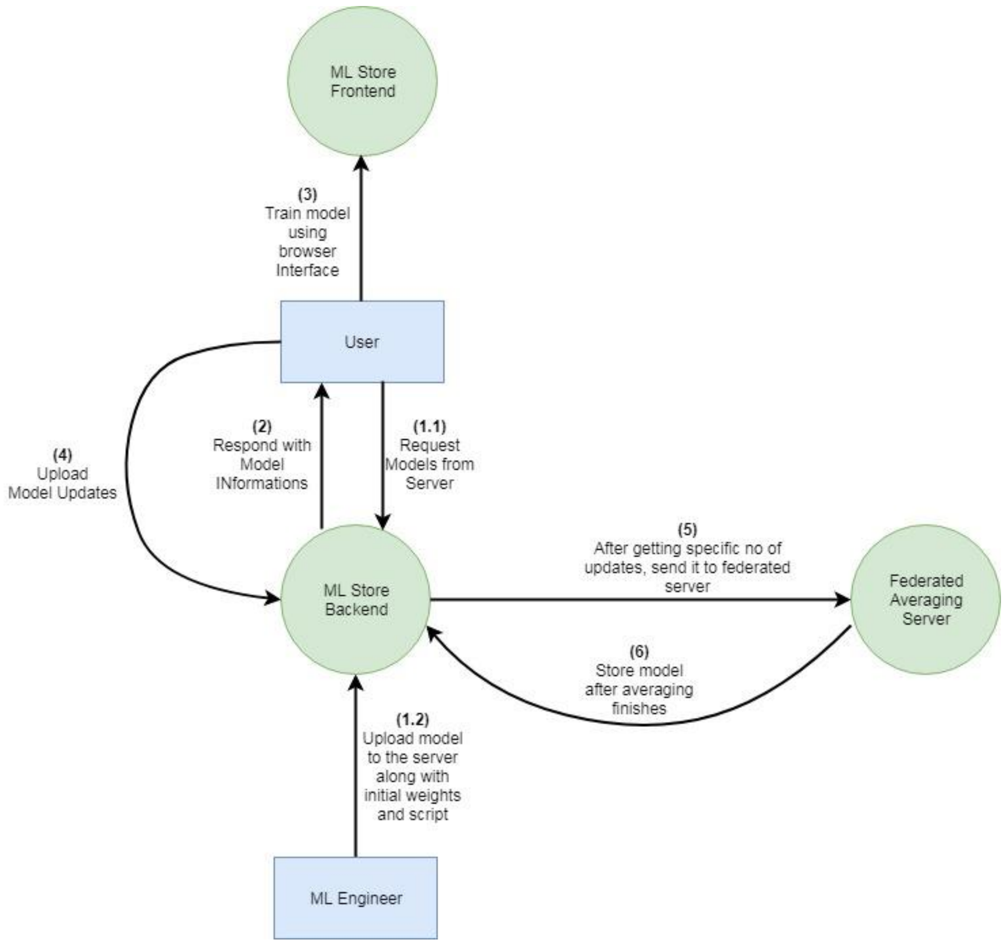

ML Store Frontend
(3) Train model using browser Interface
User
(4) Upload Model Updates
(2) Respond with Model INformations
(1.1) Request Models from Server
(5) After getting specific no of updates, send it to federated server
ML Store Backend
Federated Averaging Server
(6) Store model after averaging finishes
(1.2) Upload model to the server along with initial weights and script
ML Engineer

**Figure 3.2: Overall System workflow**

The figure shows the overall workflow of the system. The ML engineer uploads model, scripts and initial weights to the system. The user, via the ML store downloads and trains the model. After a certain amount of training, the updated weights are sent to the server. Each model has a threshold for performing averaging. When the threshold is met, the updates are averaged and it replaces the original weight. From then on, whenever the user downloads a model for training, the model with latest weight updates is downloaded.

### 3.4.2 Context Diagram

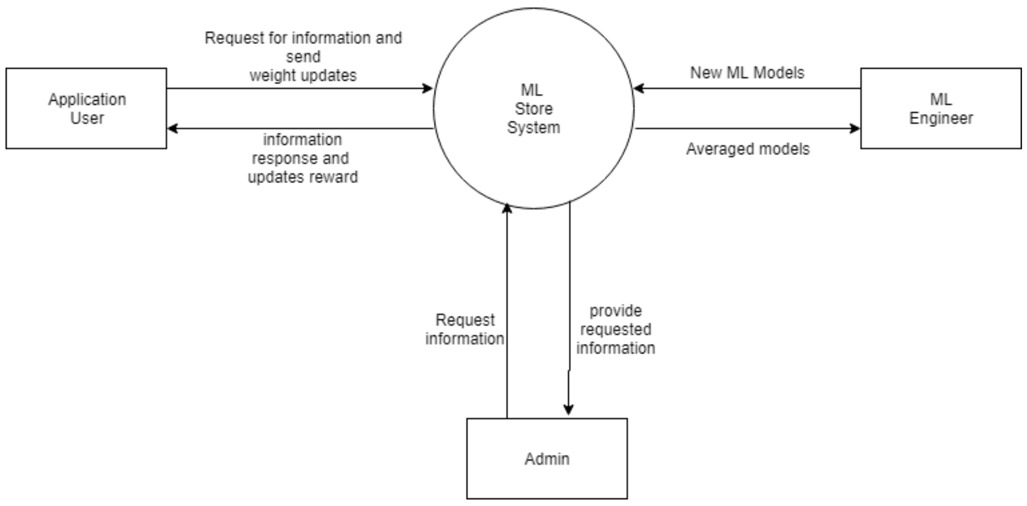

Request for information and send weight updates
Application User
information response and updates reward
ML Store System
New ML Models
ML Engineer
Averaged models
Request information
provide requested information
Admin

**Figure 3.3: Context diagram**

The users have to login to enter the system and system allows them to request information regarding models, submit review/ratings and post model updates. All that information is stored on the database and the models are stored on a shared storage which can be accessed easily by both the information server and the federated averaging server. A guest user can only view the models and their description and reviews. To get access to other features the user has to log in.

The ML engineer uploads their ML models to train on the store platform. They then get the trained models after the model has had enough data to train on.

The administrator is responsible for managing the overall platform.

### 3.4.3 DFD Level 1

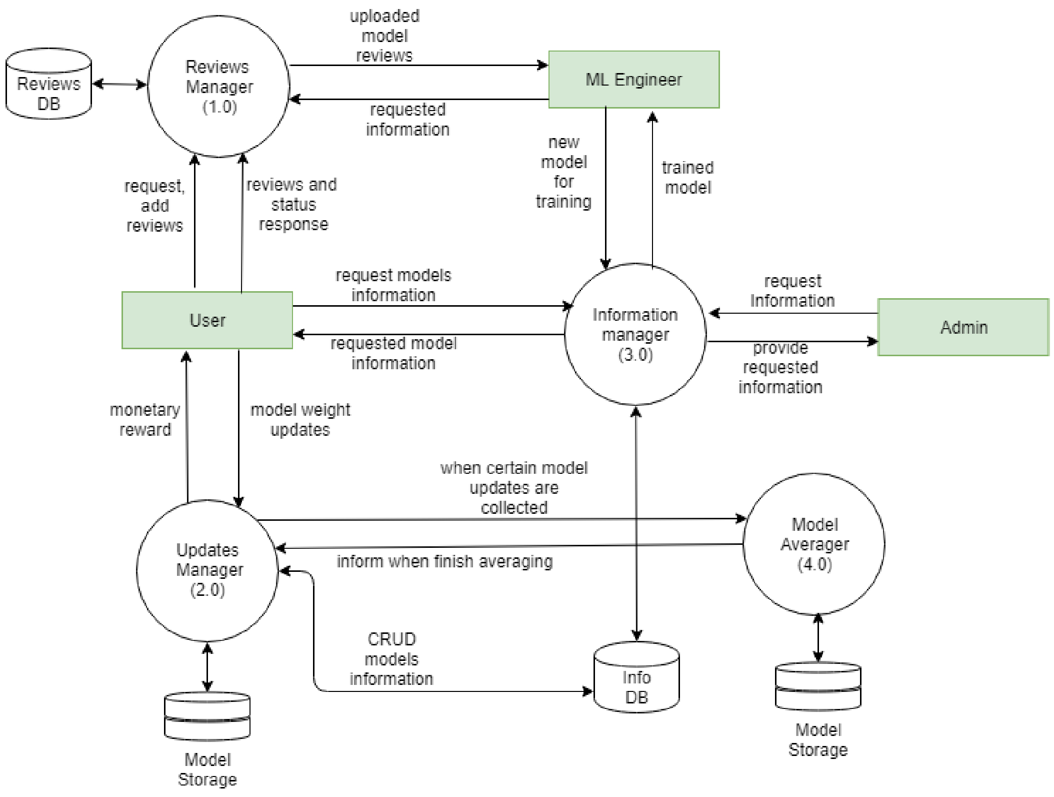


**Figure 3.4: DFD Level 1**

The figure represents the flow of information between the three entities-clients, ML engineers' application itself and the admin. The data that are passed are various model related information, details about the user, ML models, model updates and user reviews and ratings. The user gets information about models from the information manager. The information is then used to train the model. After training the updates goes to the update manager which saves the data in database. After collecting specified amount of data, it is forwarded to the Model Averager which calculates the average and stores it in shared storage.

### 3.4.4 DFD Level 2

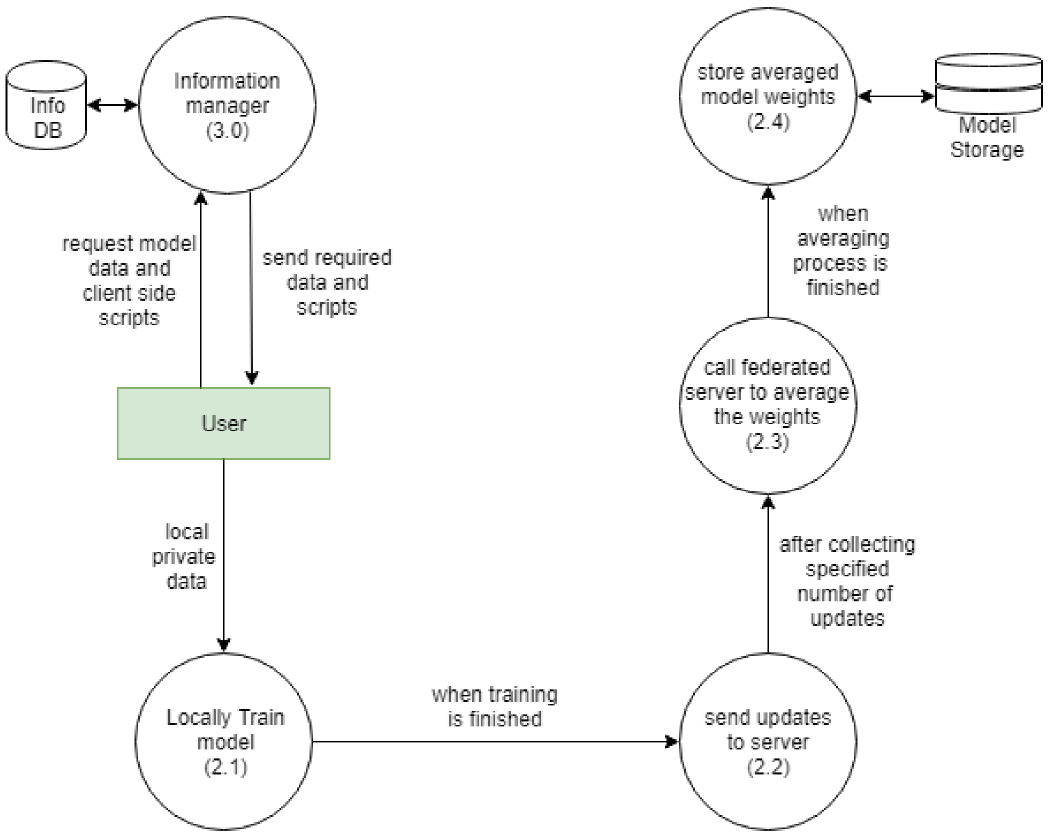

Info DB
Information manager (3.0)
request model data and client side scripts
send required data and scripts
User
local private data
Locally Train model (2.1)
when training is finished
send updates to server (2.2)
after collecting specified number of updates
call federated server to average the weights (2.3)
when averaging process is finished
store averaged model weights (2.4)
Model Storage

**Figure 3.5: DFD Level 2**

The figure represents the flow of information beginning from user getting the model from server and training it locally and ends at the multiple weight updates being averaged by the federated averaging server.

### 3.4.5 Schema Diagram

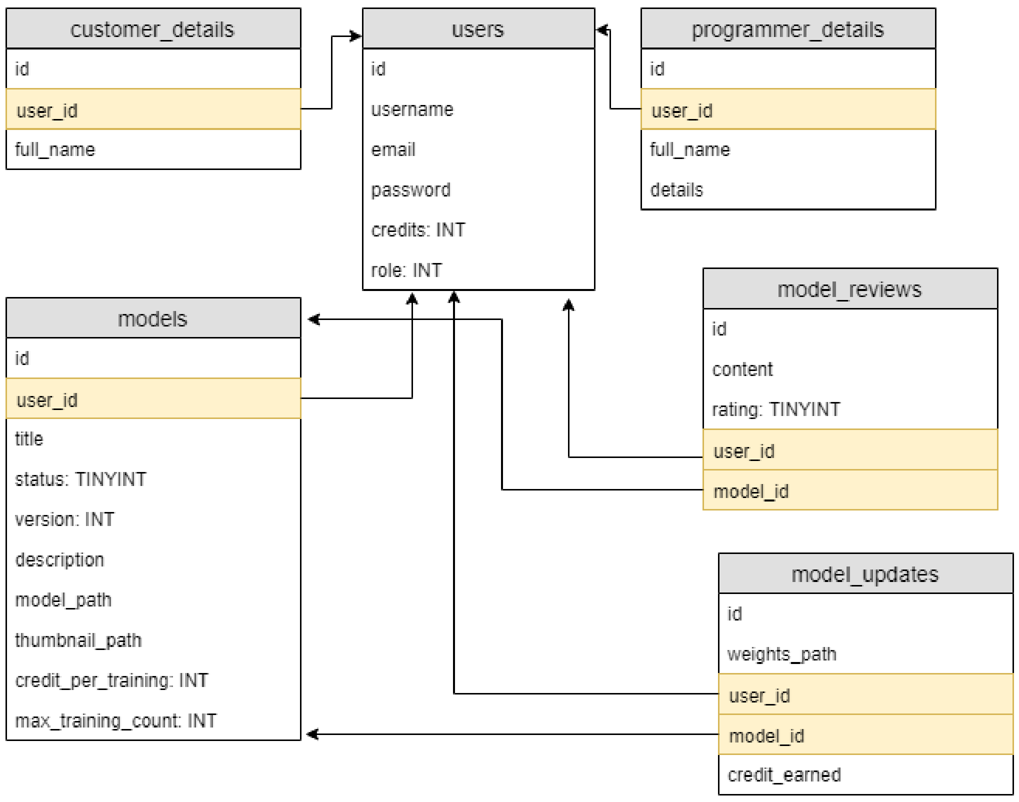


**Figure 3.6: Schema diagram**

The above diagram in figure shows the schema diagram of the application. The diagram consists of 6 different tables. The shaded rows are the foreign keys. In each table the “id” field is the primary key. Various different types of relationship can be seen among different tables as represented by the arrows.

### 3.4.5 ER Diagram

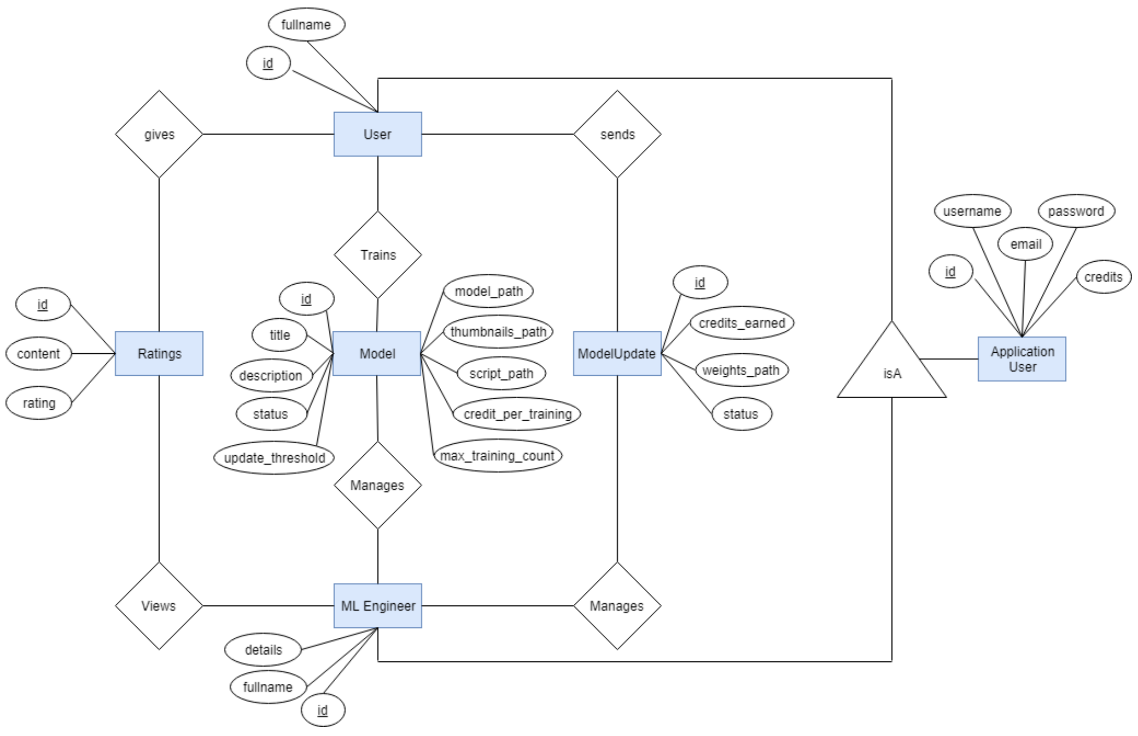


**Figure 3.7: ER diagram**

The above diagram in figure shows the entity relationship diagram of the application. The diagram shows 5 different entities with generalization and specialization in mind. Various different types of relationship can be seen among different entities.

### 3.4.6 Use Case Diagram

The below diagram shows the use case diagram of the system. It consists of 3 actors which include user, ML engineer and system administrator. The user is involved in searching, viewing and training the ML models. The users can also view their model updates along with credits earned and rate the models. ML engineers can create new models, update and delete it. They can also view the recent updates, model details and reviews and ratings on each model. System administrator has direct access to both database and filesystem. He/she is involved in privileged task such as adding, editing and deleting user, model, updates and also view/edit the model ratings.

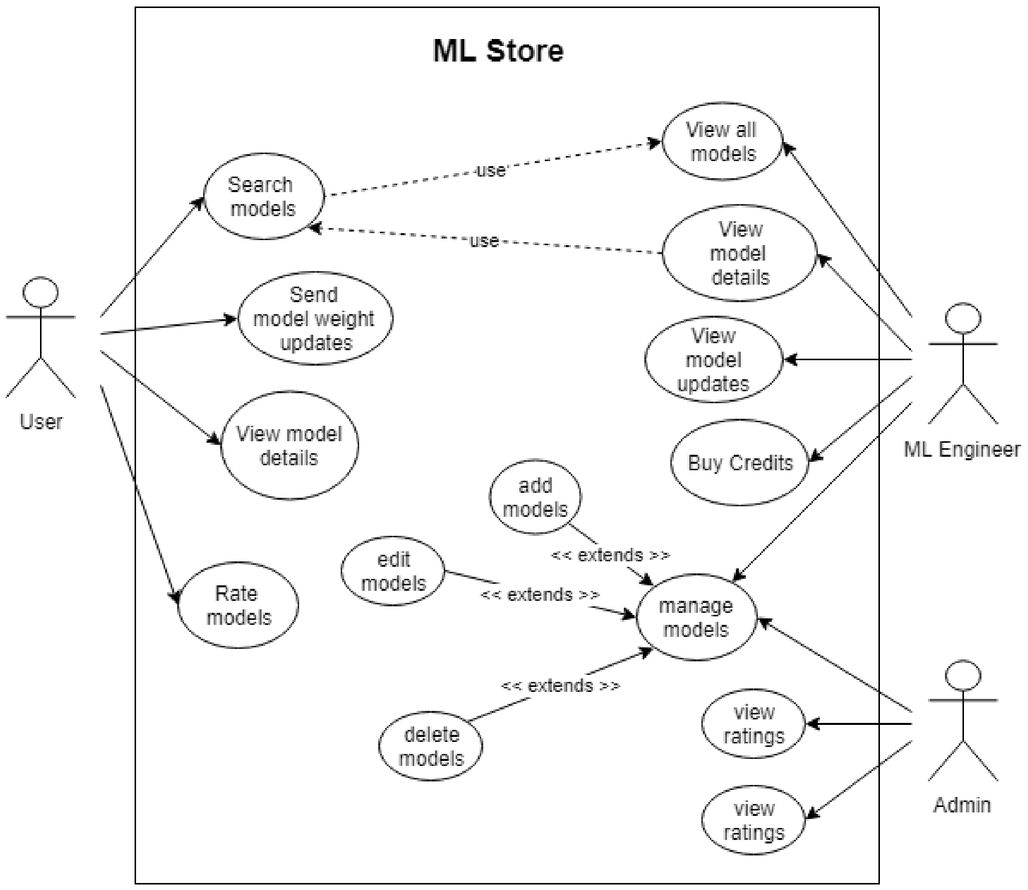


**Figure 3.8: Use Case Diagram**

## 3.5 Project Schedule

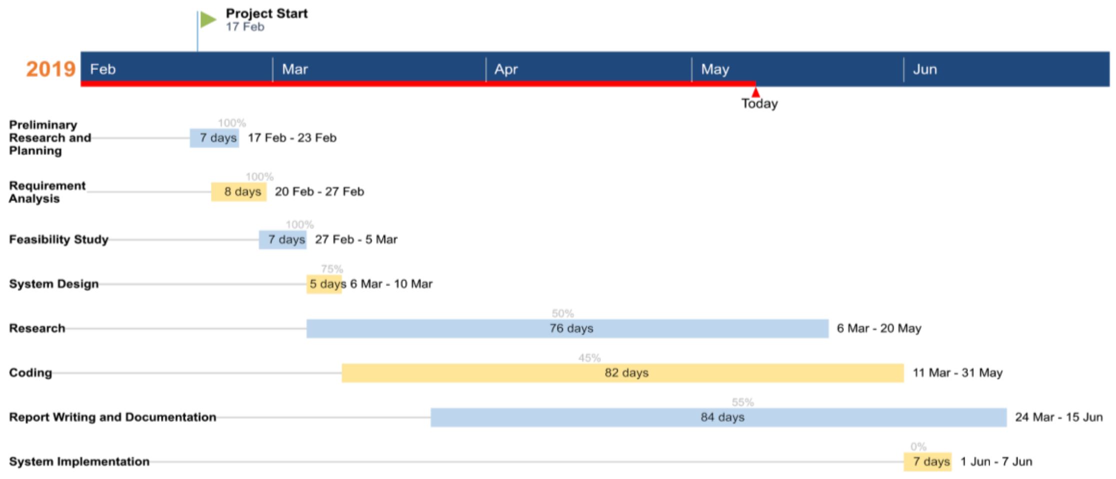


**Figure 3.9: Schedule Gantt Chart**

## 3.6 Testing Strategies

Testing is a vital part of application development which is conducted at various stages of application development with the intention of finding errors, loopholes and bugs. It is also the process of verifying and validating your application against the user requirement. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Testing is done in different forms

1. During the requirement gathering phase, the analysis and verification of requirements are also considered as testing.

2. Reviewing the design in the design phase with the intent to improve the design is also considered as testing.

3. Testing performed by a developer on completion of the code is also categorized as testing.

### 3.6.1 Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

**Federated averaging Module:**

The Federated averaging module was tested for the validation of federated averaging algorithm. In case of errors, exceptions were raised. The module was tested against wide range of datasets, trained and untrained weight updates. Initially the module was tested using hard coded data from console. Later Postman was used to test the method via HTTP post method.

**Model Upload module:**

The model upload module was also tested for data validation error. In case of such errors, exceptions were raised. The module was also tested for the CRUD operations. The operations were tested within different devices ranging from android phones to PCs, with internet connection. Testing was done with different set of input data, models, initial model weights and script files.

**Payment module:**

Payment module was also tested for the data validation error. In case such errors were to be found, an exception was raised. In order to maintain the security aspect at present the payment module operates only in presence of internet connection. The test public keys provided by Khalti API was used to perform the job.

**Search module:**

The search feature to search for ML models by name was tested. Different types of search keywords were used to test the outcomes. In case of errors or no results, respective message were shown.

**Rating module:**

The rating module used for rating ML models was tested for validation error. In case of such errors, exceptions were raised. Various rage of comment data with very long and short length and various characters were tested.

**Table 3.2: Test cases used in testing**

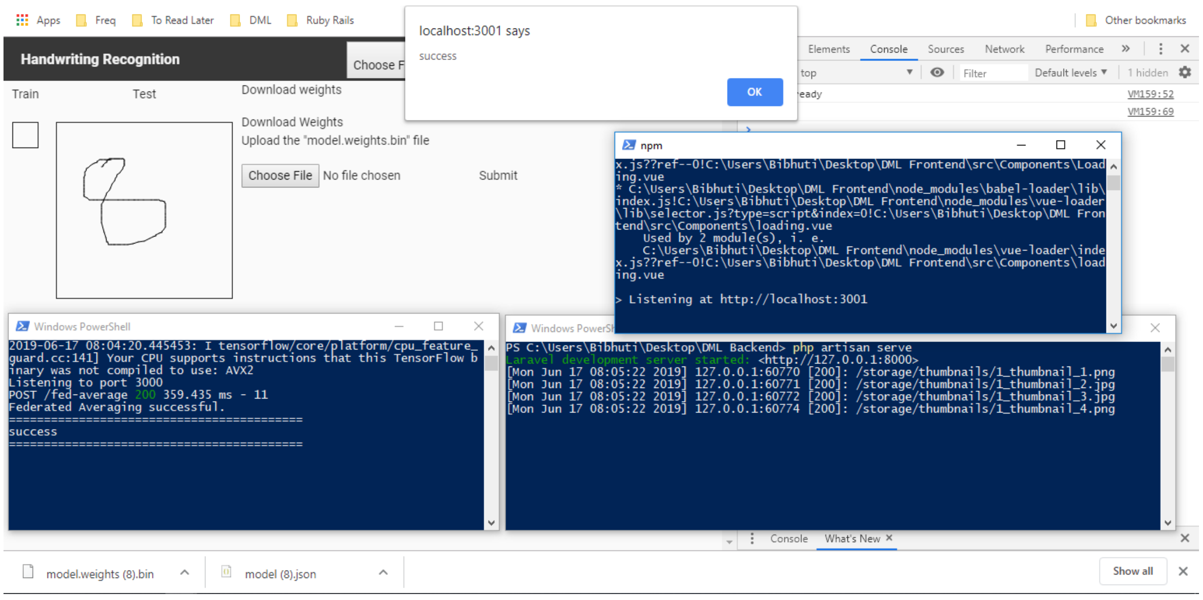
| **S. No.** | **Test Cases** | **Test Procedure** | **Expected Result** | **Status** |
|---|---|---|---|---|
| 1. | Login Form | Enter email address and password | Display home screen | Success |
| 3. | Registration Form | Enter full name, phone number, password and confirmation password | Display the dashboard screen | Success |
| 4. | Search a model | Enter initial character of the particular model | Display all the models related to the input initials character | Success |
| 5. | Select a model | Click one of the models from a list | Display model details screen | Success |
| 6 | Train model | Click train model button at model details page | Display the page where respective script is rendered. | Success |
| 7 | Send model updates | Clicked submit button from Run screen | Received success message on response | Success |
| 8 | Send model updates more than threshold | Clicked submit button from Run screen | Send the request further to federating averaging server to average the models. | Success |

| | | | | |
|---|---|---|---|---|
| 9 | Create ML model | Fill out the model create form | Display success message and redirect to model list screen. | Success |
| 10 | Delete model | Press delete button in the models list screen | Delete the model and redirect to model list screen. | Success |
| 11 | Rate a model | Press star rating icons at the bottom of model details page an d write comment and give a rating in the input dialog. | Create a new rating and display it immediately on the model details page. | Success |
| 12 | View added models | Press my models in navigation bar of ML engineer dashboard. | Display the models list | Success |
| 13 | Buy credits | Fill the credits field and click buy credits button. | Add the respective amount to users account. | Success |
| 14 | Logout | Click Logout button | Display the home screen as guest | Success |

### 3.6.2 Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. In integration testing various modules or components are integrated together and tested for the proper interface among those modules. The sub systems must work coherently with other sub system so that they perform and yield the result as expected. Integration testing was done by checking whether the values were being passed correctly from one page to another page of Vue.js components. The values of the variables were monitored through browser console, Vue.js devtools and terminal. In integration testing, the following test was carried out.

During the model training process the frontend application, the backend application and the federated averaging server must work together to update the model weights at the server. The interaction between these were tested by sending the data from browser, testing manually whether the data was stored on database or not, then after reaching certain threshold the federated averaging process was validated via Node.js console.



localhost:3001 says

success

OK

```
eady
```

npm

```
x.js??ref--0!C:\Users\Bibhuti\Desktop\DML Frontend\src\Components\Load
ing.vue
* C:\Users\Bibhuti\Desktop\DML Frontend\node_modules\babel-loader\lib\
index.js!C:\Users\Bibhuti\Desktop\DML Frontend\node_modules\vue-loader
\lib\selector.js?type=script&index=0!C:\Users\Bibhuti\Desktop\DML Fron
tend\src\Components\loading.vue
    Used by 2 module(s), i. e.
    C:\Users\Bibhuti\Desktop\DML Frontend\node_modules\vue-loader\inde
x.js??ref--0!C:\Users\Bibhuti\Desktop\DML Frontend\src\Components\load
ing.vue

> Listening at http://localhost:3001
```

Windows PowerShell

```
2019-06-17 08:04:20.445453: I tensorflow/core/platform/cpu_feature_
guard.cc:141] Your CPU supports instructions that this TensorFlow b
inary was not compiled to use: AVX2
Listening to port 3000
POST /fed-average 200 359.435 ms - 11
Federated Averaging successful.
==========================================
success
==========================================
```

Windows PowerSh

```
PS C:\Users\Bibhuti\Desktop\DML Backend> php artisan serve
Laravel development server started: <http://127.0.0.1:8000>
[Mon Jun 17 08:05:22 2019] 127.0.0.1:60770 [200]: /storage/thumbnails/1_thumbnail_1.png
[Mon Jun 17 08:05:22 2019] 127.0.0.1:60771 [200]: /storage/thumbnails/1_thumbnail_2.jpg
[Mon Jun 17 08:05:22 2019] 127.0.0.1:60772 [200]: /storage/thumbnails/1_thumbnail_3.jpg
[Mon Jun 17 08:05:22 2019] 127.0.0.1:60774 [200]: /storage/thumbnails/1_thumbnail_4.png
```

**Fig 3.10: Integration testing**

The screenshot shows integration testing process. The model is trained from the browser. The three consoles show running of Vue.js server, Laravel server and the federated averaging server. The success message in the console shows successful completion of federated averaging.

## 3.7 System Testing

System testing process involved testing the application after development using the following techniques:

1. **Usability testing**

   This was done to determine the usability of the application. To carry out this exercise, a total number of 10 respondents were sampled to test and give their feedback with regard to the application. This feedback was useful as it was used to refine the application and in validating the system.

2. **Functional Testing**

   This was done to test the systems' functional and nonfunctional requirements.

3. **Compatibility Testing**

   This involved testing the web application on different web browsers and the mobile application on different android mobile to ensure compatibility.

4. **Performance Testing**

   This was done for checking the amount of time the application would take to process a request or perform a certain functionality by performing queries and observing how long it took to execute and accomplish them.

5. **Validation**

   To validate if the web application streamlines the process of training a model via federated averaging, a sample 10 users were selected from the target population and put through the process of testing. The entire process of searching and submitting model updates was tested and the analysis of the feedback collected led to this validation. Validation was also done by analysis of the feedback received from data collection process.

## 3.8 Implementation

Implementation of a system deals with all the processes involved in delivering the system into the real world. This may involve processes such as analyzing requirements, installation, configuration, customization, running, testing, systems integration, user training, delivery and making necessary changes.

Implementation which is also sometimes referred to as Deployment is an important stage which requires utmost attention. A number of systems may be underutilized, don't meet their potential or fail totally due to improper implementation. The objective of the system implementation phase is to deliver a fully operational and documented system.

# CHAPTER 4: RESULT ANALYSIS

The application was developed from scratch which can be used for decentralized machine learning training of private data and earn incentives for the task. The screenshots of "ML Store: Platform for decentralized Machine Learning" are shown in figures below. Brief descriptions of each screenshot follow.

## 4.1 Results

The Screenshots of some of the main components of the system is shown below:

### 4.1.1 Client Homepage

**Main Function prototype:**

```
 public function index()
  {
    $models = MLModel::all();
    return response()->json($models, 200);
  }

  public function search(Request $request)
  {
    $models =  MLModel::query()
      ->where('title',     'LIKE',     "%{$request->query-
>keyword}%")
      ->get();
    return response()->json($models, 200);
  }

  public function popular()
  {
    $models    =    MLModel::orderBy('version',    'DESC')-
>limit(5)->get();
    return response()->json($models, 200);
  }
```

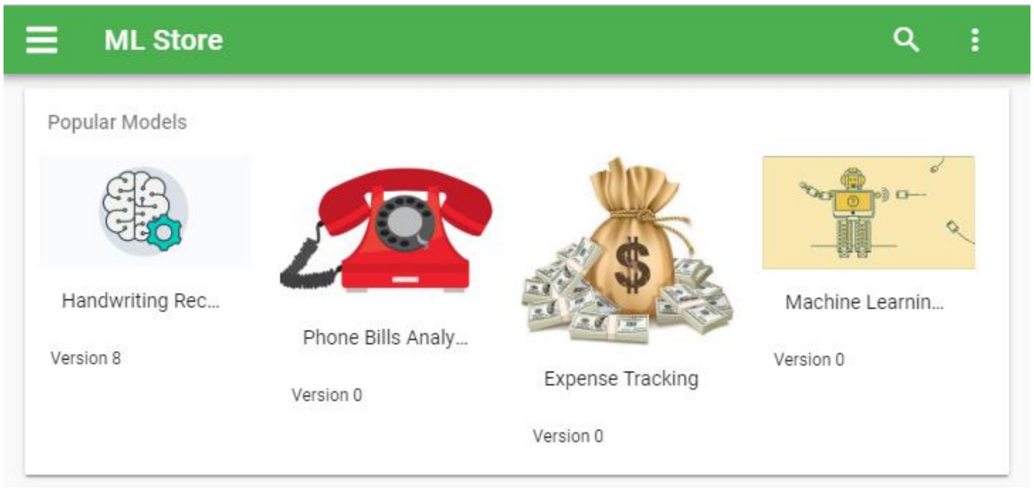


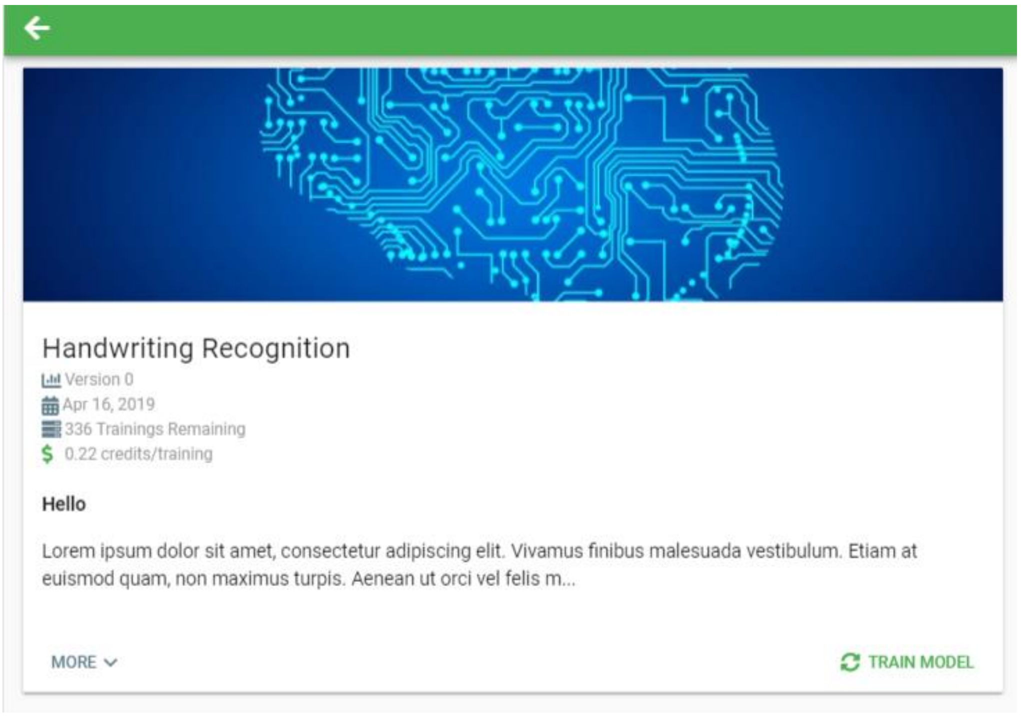
**Figure 4.1: Client homepage**

Above figure shows the main page of the clients/customers. The users can view and search various Machine Learning models. Popular models, top trained models etc. can also be viewed. Each of the models contains little information about it

### 4.1.2 Model Details

#### 4.1.2.1 Main Function Prototype:

```
public function show($id)
  {
    $model = MLModel::findOrFail($id);
    return response()->json($model, 200);
  }
```


Handwriting Recognition

Version 0

Apr 16, 2019

336 Trainings Remaining

$ 0.22 credits/training

Hello

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus finibus malesuada vestibulum. Etiam at euismod quam, non maximus turpis. Aenean ut orci vel felis m...

**Figure 4.2: Model Details**

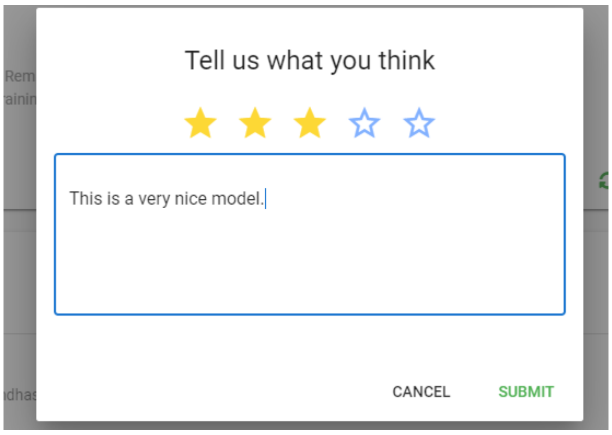
The Model details page shows description of the model, credits(money) per training i.e. received by the user for training, model version (version gets upgraded on each iteration of federated averaging), remaining number of trainings and other details of the model.

### 4.1.3 Comments and Reviews

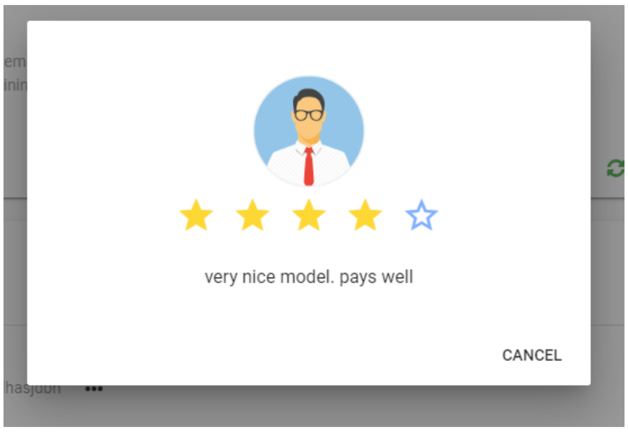
#### 4.1.3.1 Main Function Prototype:

```
 public function show($id)
  {
    $modelReview = ModelReview::findOrFail($id);
    return response()->json($modelReview, 200);
  }

  public function store(Request $request, $model_id)
  {
    $request->merge(['user_id'         =>  $request->auth-
>id]);
    $request->merge(['model_id'     => $model_id]);
    $created = ModelReview::create($request->all());

    return ($created)
      ? response()->json(['message' => 'success'], 201)
```

```
        : response()->json(['message' => 'fail'], 400);
}
```



This is a very nice model.



very nice model. pays well

**Figure 4.3: Comments and reviews**

The users can view and submit comments and submit their reviews and ratings. According to the rating and reviews top, popular models are determined.

### 4.1.4 Model Training

#### 4.1.4.1 Main Function Prototype

```
module.exports = async function(model_path, update_paths,
callback) {
    var weightsList = [];

    // make sandbox
    if (!fs.existsSync("sandbox")) {
        fs.mkdirSync("sandbox");
    } else {
        deleteFolderR("sandbox");
        fs.mkdirSync("sandbox");
    }

    // copy model and initial weight to sandbox
    fs.copyFileSync(

`../public/storage/models/${model_path}/${model_path}.jso
n`,
        `./sandbox/${model_path}.json`
    );
    fs.copyFileSync(

`../public/storage/models/${model_path}/weights.bin`,
        `./sandbox/weights.bin`
    );

    // load initial weights
    model = await tf.loadLayersModel(
        "file://./sandbox/" + model_path + ".json"
    );
    weightsList.push(model.getWeights());
    fs.unlinkSync("./sandbox/weights.bin");

    // load each weight updates
    for (let i = 0; i < update_paths.length; i++) {
        const update = update_paths[i];
        fs.copyFileSync(
            `../public/storage/updates/${update}`,
            `./sandbox/weights.bin`
        );
        model = await tf.loadLayersModel(
            "file://./sandbox/" + model_path + ".json"
        );
        weightsList.push(model.getWeights());
        fs.unlinkSync("./sandbox/weights.bin");
    }
    model = await tf.loadLayersModel(
        "file://./sandbox/" + model_path + ".json"
    );
    fs.unlinkSync("./sandbox/weights.bin");

    // average the weights
    var length = weightsList[0].length;
    var meanWeights = [];
    for (var j = 0; j < length; j++) {
        var collection = [];
        for (var i = 0; i < weightsList.length; i++) {
            collection.push(weightsList[i][j]);
        }
        meanWeights.push(tf.stack(collection).mean((axis
= 0)));
    }

    // set averaged weights
    model.setWeights(meanWeights);

    // get the model
    fs.unlinkSync(`./sandbox/${model_path}.json`);
    await model.save(`file://./sandbox`);

    // save updated weight to public directory
    fs.copyFileSync(
        `./sandbox/weights.bin`,

`../public/storage/models/${model_path}/weights.bin`
    );

    console.log("Federated Averaging successful.");
    callback();
};
```

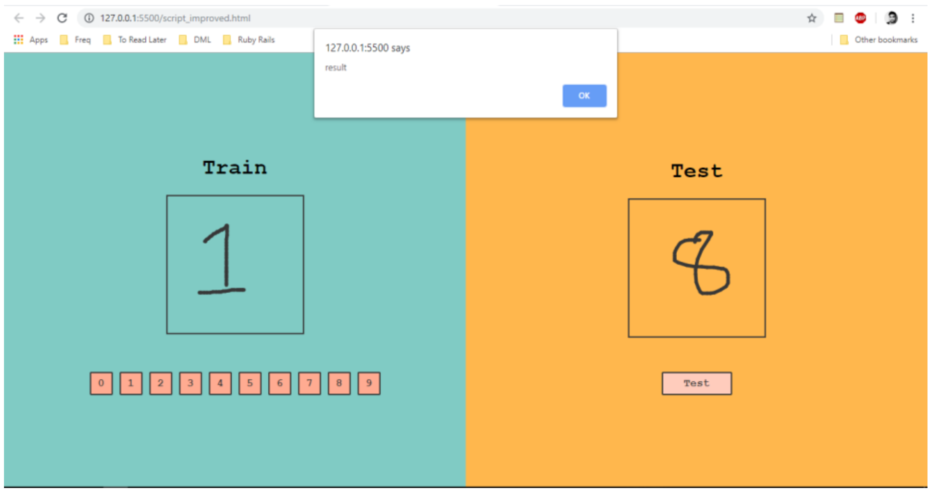


**Figure 4.4: Model Training**

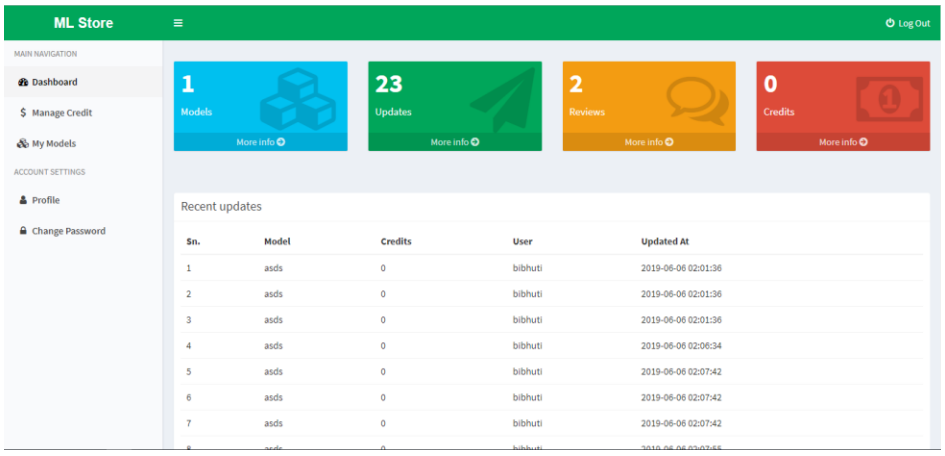
The picture shows training of handwriting recognition model in a web browser. The images that are drawn on the canvas is used for training on the client side and the updates are sent to the server.

### 4.1.5 ML Engineer dashboard

#### 4.1.5.1 Main Function Proptype

```
    public function index()
    {
      $models_count = Auth::user()->models()->count();
      $updates_count =Auth::user()->updates()->count();
      $reviews_count = Auth::user()->reviews()->count();
      $credits = Auth::user()->credits;
      $updates   =   ModelUpdate::orderBy('created_at')-
>limit(15)->get();
```

```
      return   view('dashboard',   compact('models_count',
'updates_count', 'reviews_count', 'credits', 'updates'));
    }
```



Recent updates

| Sn. | Model | Credits | User | Updated At |
|---|---|---|---|---|
| 1 | asds | 0 | bibhuti | 2019-06-06 02:01:36 |
| 2 | asds | 0 | bibhuti | 2019-06-06 02:01:36 |
| 3 | asds | 0 | bibhuti | 2019-06-06 02:01:36 |
| 4 | asds | 0 | bibhuti | 2019-06-06 02:06:34 |
| 5 | asds | 0 | bibhuti | 2019-06-06 02:07:42 |
| 6 | asds | 0 | bibhuti | 2019-06-06 02:07:42 |
| 7 | asds | 0 | bibhuti | 2019-06-06 02:07:42 |

**Figure 4.5: ML Engineer dashboard**

The ML engineer’s dashboard shows the overall information of user’s models, updates, reviews and remaining credits. The table below shows the brief information on recent model updates received from the users.

### 4.1.6 Manage credits

#### 4.1.6.1 Main Function Prototype:

```
   public function verifyPayment(Request $request){

      $args = http_build_query(array(
        'token' => $request->input('token'),
        'amount'  => $request->input('amount')
      ));

      $url = "https://khalti.com/api/v2/payment/verify/";

      # Make the call using API.
      $ch = curl_init();
      curl_setopt($ch, CURLOPT_URL, $url);
      curl_setopt($ch, CURLOPT_POST, 1);
      curl_setopt($ch, CURLOPT_POSTFIELDS,$args);
      curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

$headers          =          ['Authorization:          Key
test_secret_key_f59e8b7d18b4499ca40f68195a846e9b'];
```

```
        curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);

        // Response
        $response = curl_exec($ch);
        $status_code          =          curl_getinfo($ch,
CURLINFO_HTTP_CODE);
        curl_close($ch);

        Auth::user()->update([
          'credits'  =>  Auth::user()->credits  +  ($request-
>input('amount')/100)
        ]);

        return $response . '<br>' . $status_code;
      }
```

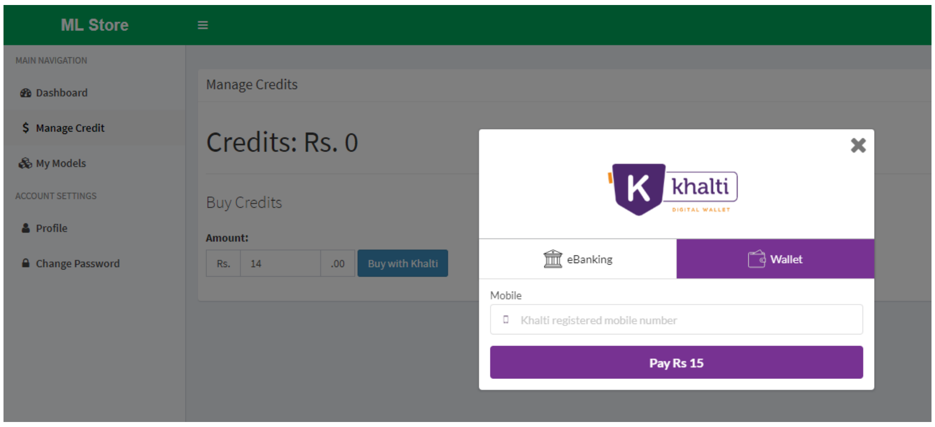


**Figure 4.6: Manage credits**

The ML Engineer can view and manage its credits. For now the application has been integrated with Khalti API through which engineers can buy credits.

### 4.1.7 Create and manage Models

#### 4.1.7.1 Main Function Prototype

```
   public function store(Request $request)
     {
        $request->validate([
          'title' => 'required|unique:ml_models|max:255'
        ]);
        $request->merge(['user_id' => Auth::id()]);
        $model = MLModel::create($request->all());
   if($request->hasFile('thumbnail'))             $model-
>thumbnail_path = $this->storeFile('thumbnail', $request,
$model->id);
      if($request->hasFile('script')) $model->script_path
= $this->storeFile('script', $request, $model->id);
      if($request->hasFile('model')) $model->model_path =
$this->storeFile('model', $request, $model->id);
      $created = $model->save();

      return ($created)
        ? back()->with('message', 'Model created')
        : back()->with('error', 'Error creating model');
    }

   public function storeFile($name, $request, $model_id){
      $ext            =            $request->file($name)-
>getClientOriginalExtension();
      $foldername = 'public/' . $name . 's';
      $filename = Auth::id().'_'.$name.'_'. $model_id. '.'
. $ext;

      // json and bin file should be in one folder
      if($name === 'model') {
        $localFolderName    =    Auth::id().'_'.$name.'_'.
$model_id;
        $foldername =  $foldername . '/'. $localFolderName
;
        Storage::makeDirectory($foldername);
        $request->file($name)->storeAs($foldername,
$filename);
        $request->file('weights')->storeAs($foldername,
'weights.bin');
        return $filename;
      }

      $request->file($name)->storeAs($foldername,
$filename);
      return $filename;
    }
```

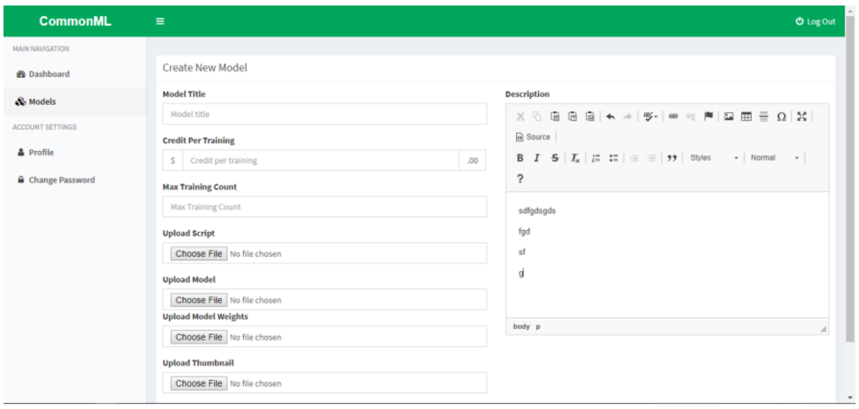


Figure 4.7: Create and manage models

From the dashboard, the ML engineer can view, create, edit and delete the created models. At any time the model and its scripts, weights can be modified.

## 4.2 Critical Analysis

### 4.2.1 Data Communication

Training machine learning models on a central server machine or cloud computing platforms has been the de facto method for most applications that utilize machine learning. Given how a server machine often has a large computing resource at its disposal compared to an average client machine, it seems the most feasible option. But even a wealth of computing resource does not change the fact that these models require huge amounts of data, more than anything else, to function successfully. And because these training data have to be sent to the server for each round of training, data communication can be a constraining factor, especially when the data sets are large. By using federated learning as a method to train the models, the communication cost is reduced substantially as the data do not have to be sent to the server each time a model is to be trained. Only the model updates are communicated between the server and client machines which when compared to the large datasets is a small amount. This may come at a tradeoff of the computational efficiency that is provided by the server or cloud platforms as users' devices may not have sufficient computational resources required to train the models.

### 4.2.2 Common Model Training Platform

In traditional training methods, models are deployed into applications after an initial training on the server after which the users would provide the necessary data to further train the model as they use the application. The approach in this project provides a common platform to share the models and crowdsource their training before they are deployed. The models are open for users to train on their own devices. This means that the models upon deployment are more accurate and because of federated learning more relevant for every user. An obstacle in this approach may be having to deal with data poisoning where an adversary may knowingly provide a carefully crafted false data to tamper with the model. However, if there are a large number of users training a model and unless the adversary does not have a majority share of the devices training the model, the effects of the data poisoning may be minimal as the final trained model in the server is a culmination of the training data from all the users.

### 4.2.3 Data Security and Privacy

In the light of Facebook's recent data scandal and the newly formed GDPR regulations, data security and privacy has become a pressing concern for developers and engineers. As the traditional models running on the server require data to be sent to the server machine, it becomes problematic when the training data consists of private or sensitive information. The use of federated learning provides a way to train the models within the users' devices, which means that no private data is ever sent out of the realms of their devices. Only the updates to the model are sent back to the server. The updates are also generated in such a way that it cannot be used to determine what data had been used to update the model. Furthermore, encryption techniques can be added on top of the existing communication techniques for ensuring better data security.

### 4.2.4 Performance

As a core part of the application is model training, the performance of the application is a crucial factor. The performance depends heavily on the hardware devices. Since GPU is highly optimized for working with numerical data and matrices, devices with GPU can train the models faster than the devices with CPU only. In comparison with other machine learning applications like Google's Gboard and Mozilla's URL bar suggestion, the approach in this project is comparatively slower. The reason behind this is the training of ML models in browsers via JavaScript. But the good part is that the models are openly accessible for all users to train. The users don't have to worry about any kind of setup and configurations. Anyone with a web browser can easily use this application.

### 4.2.5 Comparison with online machine learning platforms

Although the use of federated learning for training models is a relatively new concept, the use of online platforms for training and sharing machine learning models has long been in practice. The closest example is OpenML. It is a comprehensive platform which allows everything from upload models and data, have them trained, analyzed and run, and also share the results [33]. Then there are also the big-name cloud platforms like Amazon Web Services, Google Cloud Platform and Microsoft Azure, all of which provide an extensive amount of computational resource to both train and deploy the models. But in all these platforms, the data stays with the platforms themselves, meaning that it requires data to be sent to these platforms in order to train them. The difference in this project is that the users' data stays with them and does not get uploaded to the platform.

This project was inspired the most by an example prototype that was discussed in the DML website [33], [34]. However, in the prototype as well as in the general concept of DML, blockchain and smart contracts have been used as a means of wider decentralization, automation of model updates and also tokenization for the purpose of incentivizing the participants. Because of the imminent complexity that would be caused by the use of blockchain technology in this time constrained project, it was omitted from the project. While the decentralization provided by blockchain may help in more thorough and diverse training, an iterative training method was chosen instead for the project. For the incentivization, a local online payment system was used instead.

## 4.3 Limitation and Future Enhancements

### 4.3.1 Limitations

- Running machine learning models on web browsers can be resource intensive
- Insufficient computing resources on users' device may create a bottleneck in the training process
- Web browser used for the purpose need to support Tensorflow.js
- Incentives for users may be low for each training
- Use of online payment may prevent the anonymization of users contributing for training the model
- Large number of updates are required for the model to converge towards its goal
- As the version of models increase, updates become useless if training is continued on older version of the model

### 4.3.2 Future Enhancements:

- Implement a CPU usage customization settings for users so they can decide how much CPU power to spend for the training
- Update the system so that it can make use of idle processing power
- Find a more efficient communication mechanism to reduce bandwidth while sending models and their updates
- Strict accuracy measurement mechanism so that it's easier to incentivise according to the quality of the data
- Make all transactions completely anonymous by deploying the whole application in a blockchain network
- Make the models available to users only if the users' devices meet the hardware and software requirements
- Include new features and exhaustive tools on the platform for more advanced analytics, organization, testing, and deployment of the models

## 4.4 Conclusion

This project extends the idea of machine learning models to find a sound way to use the untapped private data and unleash their potential to better adapt such models for every user while respecting data privacy. Through this approach, both the private data and processing power for machine learning are decentralized as algorithms are run directly on individual devices by utilizing their idle processing power.

This project would not have been accomplished if it were not for the series of research that went on in the field of federated learning, many of which had only been completed during the course of this project. In this early stage of federated learning, it would have been difficult to move forward for the project if these researches and their findings had not been made publicly available.

Furthermore, a lot of time and effort was used to research about the tools to be used for the development of the system. As per the ML Store is much concerned with machine learning and federated averaging. Tensorflow.js was used as the machine learning framework. For data management and content management, the PHP framework Laravel and MySQL were used. The core federated averaging algorithm in written in Node.js. To support different platforms the web browser has been used as the main platform and the application is developed as a Single Page Application in Vue.js using the Vuetify UI library.

The work done on this project is far from comprehensive and is bound to have errors and fallacies which may have been overlooked. Through this project, the authors hope to lay a foundation for the future projects which may expand upon the concepts and ideas shown in this project. As such the authors welcome any kind of queries or suggestions regarding the project or the ideas outlined within it.

# BIBLIOGRAPHY

1. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.
2. H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, ‘Communication-Efficient Learning of Deep Networks from Decentralized Data’, arXiv preprint, 2016.
3. J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, ‘Federated learning: Strategies for improving communication efficiency’, arXiv preprint.
4. “Decentralized Machine Learning White Paper.” [Online]. Available: https://decentralizedml.com/DML_whitepaper_31Dec_17.pdf.

# REFERENCES

[1] Decentralized Machine Learning, “Introducing DML — Decentralized Machine Learning Protocol,” *Medium*, 07-Jan-2018. [Online]. Available: https://medium.com/decentralized-machine-learning/introducing-dml-decentralized-machine-learning-protocol-f954ccd9f90d. [Accessed: 25-Mar-2019].

[2] “Federated Learning: Collaborative Machine Learning without Centralized Training Data,” *Google AI Blog*. [Online]. Available: http://ai.googleblog.com/2017/04/federated-learning-collaborative.html. [Accessed: 25-Mar-2019].

[3] D. Swanson, “Machine Learning Algorithms.” [Online]. Available: http://icl.cs.utk.edu/classes/cosc462/2017/pdf/project/swanson_ML.pdf.

[4] “Decentralized and Distributed Machine Learning Model Training with Actors.” [Online]. Available: http://www.scs.stanford.edu/17au-cs244b/labs/projects/addair.pdf. [Accessed: 29-Mar-2019].

[5] “Decentralized Machine Learning (DML) - ICO rating and details,” *ICObench*. [Online]. Available: https://icobench.com/ico/decentralized-machine-learning. [Accessed: 22-Jun-2019].

[6] J. Hale, “Deep Learning Framework Power Scores 2018,” *Towards Data Science*, 20-Sep-2018. [Online]. Available: https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a. [Accessed: 22-Jun-2019].

[7] “How to find the best machine learning frameworks for you,” *SearchEnterpriseAI*. [Online]. Available: https://searchenterpriseai.techtarget.com/feature/How-to-find-the-best-machine-learning-frameworks-for-you. [Accessed: 29-Mar-2019].

[8] A. D. Rayome, “Why Python is so popular with developers: 3 reasons the language has exploded,” *TechRepublic*. [Online]. Available: https://www.techrepublic.com/article/why-python-is-so-popular-with-developers-

3-reasons-the-language-has-exploded/. [Accessed: 29-Mar-2019].

[9] *TensorFlow Federated (TFF): Machine Learning on Decentralized Data (TF Dev Summit '19)*. 2019.

[10] "TensorFlow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/. [Accessed: 29-Mar-2019].

[11] "What are the types of problems TensorFlow can help solve?," *Stack Overflow*. [Online]. Available: https://stackoverflow.com/questions/37295295/what-are-the-types-of-problems-tensorflow-can-help-solve. [Accessed: 29-Mar-2019].

[12] "TensorFlow Federated | TensorFlow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/federated. [Accessed: 29-Mar-2019].

[13] "TensorFlow Federated | TensorFlow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/federated. [Accessed: 29-Mar-2019].

[14] S. G. Johnson, "NIST Special Database 19," *NIST*. [Online]. Available: https://www.nist.gov/srd/nist-special-database-19. [Accessed: 29-Mar-2019].

[15] TensorFlow, "Introducing TensorFlow Federated," *Medium*, 06-Mar-2019. [Online]. Available: https://medium.com/tensorflow/introducing-tensorflow-federated-a4147aa20041. [Accessed: 29-Mar-2019].

[16] "IBM Knowledge Center." [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSMQ79_9.5.1/com.ibm.egl.pg.doc/topics/pegl_serv_overview.html. [Accessed: 22-Jun-2019].

[17] "What is a Plug-In? - Definition from Techopedia," *Techopedia.com*. [Online]. Available: https://www.techopedia.com/definition/4324/plug-in. [Accessed: 22-Jun-2019].

[18] "As Facebook Raised a Privacy Wall, It Carved an Opening for Tech Giants," 19-Dec-2018. [Online]. Available: https://www.nytimes.com/2018/12/18/technology/facebook-privacy.html. [Accessed: 25-Mar-2019].

[19] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM J. Res. Dev.*, vol. 44, no. 1.2, pp. 206–226, Jan. 2000.

[20] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, “SoK: Security and Privacy in Machine Learning,” *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2018.

[21] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, “Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers,” *International Journal of Security and Networks*, vol. 10, no. 3. p. 137, 2015.

[22] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart, “Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing,” *Proc USENIX Secur Symp*, vol. 2014, pp. 17–32, Aug. 2014.

[23] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, “A survey of machine learning for big data processing,” *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1. 2016.

[24] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, “Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5330–5340.

[25] A. Khan, C. Zhang, D. D. Lee, V. Kumar, and A. Ribeiro, “Scalable Centralized Deep Multi-Agent Reinforcement Learning via Policy Gradients,” *arXiv preprint*, 2018.

[26] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” *arXiv preprint*, 2016.

[27] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” *arXiv preprint*, 2016.

[28] M. Resnick, “Decentralized Modeling and Decentralized Thinking,” *Modeling and Simulation in Science and Mathematics Education*. pp. 114–137, 1999.

[29] “Decentralized Machine Learnng White Paper.” [Online]. Available: https://decentralizedml.com/DML_whitepaper_31Dec_17.pdf. [Accessed: 29-Mar-2019].

[30] V. Triglianos and C. Pautasso, “Asqium: A JavaScript Plugin Framework for Extensible Client and Server-Side Components,” *Engineering the Web in the Big Data Era*. pp. 81–98, 2015.

[31] O. Mesa *et al.*, “Understanding vulnerabilities in plugin-based web systems,” *Proceeedings of the 22nd International Conference on Systems and Software Product Line - SPLC ’18*. 2018.

[32] I. Jorstad, S. Dustdar, and D. Van Thanh, “A Service Oriented Architecture Framework for Collaborative Services,” *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE’05)*. .

[33] J. Vanschoren, “OpenML,” *OpenML: exploring machine learning better, together*. [Online]. Available: https://www.openml.org. [Accessed: 24-Jun-2019].

[34] “Decentralized Machine Learning - DML.” [Online]. Available: https://decentralizedml.com/. [Accessed: 24-Jun-2019].

# APPENDIX

## Appendix A: Federated Learning

### A.1 Federated Learning Process

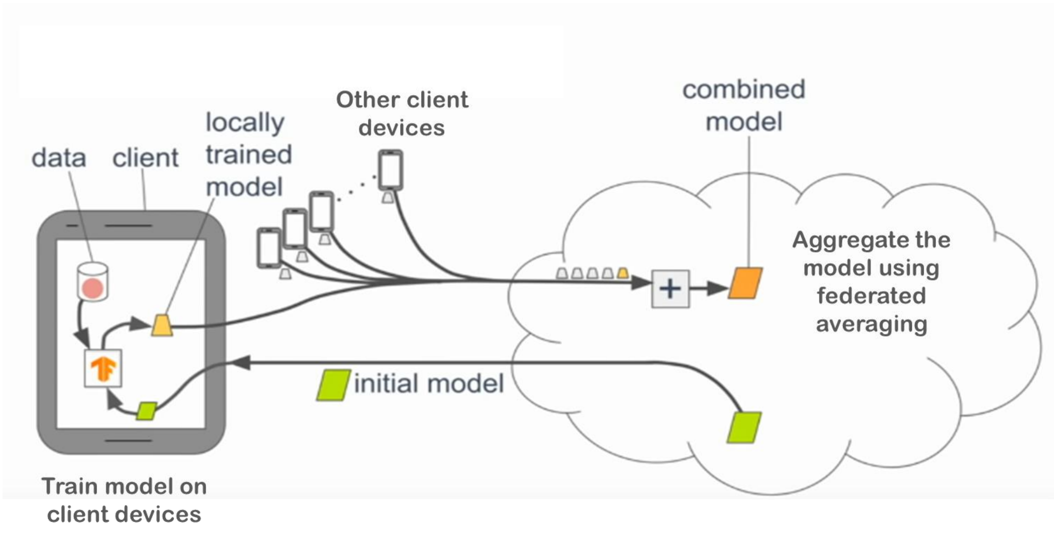
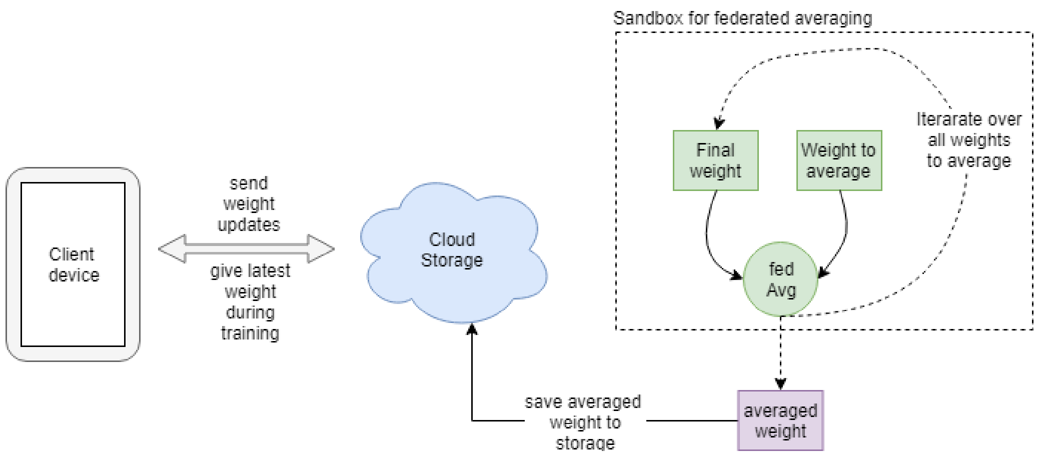


**Figure A.1: Federated Learning process [9]**

The figure explains the overall mechanism of federated learning. At the beginning an initial model is sent to the user from the server. The user trains the model, using its local private data and sends the updates to the server. Similarly, other users training the model with their own data also send the updates to the server. At the server the updates are averaged. In the next iteration the updates model is sent to the users.

## A.2 Federated Averaging



Sandbox for federated averaging
Client device
send weight updates
give latest weight during training
Cloud Storage
Final weight
Weight to average
Iterarate over all weights to average
fed Avg
averaged weight
save averaged weight to storage

**Figure A.2: Federated averaging workflow**

The figure shows the federated averaging algorithm, that we have implemented for this project. The updates sent by the users are stored on the server. When specified number of updates are collected averaging is performed. Every operation happens inside a sandbox. Its a folder where all the ways to be averaged are performed. In this way, the original files are untouched and operation happens with the copy of original files. If anything goes wrong, it doesn't affect the original files.

## A.3 Training of handwriting recognition model via HTML5 canvas

```
async function doTrain(digit){

  let tensor = tf.browser.fromPixels(canvasTrain)
    .resizeNearestNeighbor([28, 28])
    .mean(2)
    .expandDims(2)
    .expandDims()
    .toFloat().div(255.0);

  let labels = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
  labels[digit] = 1;
  let label = tf.tensor2d(labels, [1, 10]);
  await model.fit(tensor, label, {batchSize: 1, epochs: 1});
}
```

The function performs the job of training a handwriting recognition model by taking input from HTML5 canvas. It takes a digit as input and asynchronously trains the model. As a part of data preparation, it creates two tensors as training data and label. The training data represents the pixels of the canvas and the label refers to the digit for which it is trained.