

AUTONOMOUS MACHINE LEARNING MODELING USING A TASK ONTOLOGY

A Project Report

*submitted to Swarnandhra College of Engineering and Technology
in partial fulfilment of the requirements for the award of the degree*

MASTER OF COMPUTER APPLICATIONS

By

**CHAMAKURI PRASAD
(17A21F0004)**

Under the guidance of

**A.N.L. Kumar
Associate Professor, Department of MCA**



**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
SWARNANDHRA COLLEGE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)**

**(Approved by AICTE & Permanently Affiliated to JNTUK)
(Accredited by NAAC with “A” Grade)
SEETHARAMPURAM, NARASAPUR-534280**

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

**SWARNANDHRA COLLEGE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)**

Seetharampuram, Narsapur – 534 280.



CERTIFICATE

This is to certify that this project work entitled **“Autonomous Machine Learning Modelling Using a Task Ontology ”** is the bonafide work of **Mr. CHAMAKURI PRASAD (Regd.No.17A21F0004)** who carried out the work under my supervision, and submitted in partial fulfilment of the requirements for the award of the degree, **Master of Computer Applications**, during the academic year **2019-2020**.

Project Supervisor

**A.N.L. Kumar
Department of MCA**

Head of Department

**A.N.L. Kumar
Department of MCA**

Submitted for the project Thesis/Dissertation Viva-voice held on

External Examiner

DECLARATION

I certify that

- a. The work contained in the project work is original and has been done by me under the guidance of my supervisor
- b. The work has not been submitted to any other University for the award of any degree or diploma
- c. The guidelines of the University are followed in writing this report.

Date:

CHAMAKURI PRASAD

(17A21F0004)

ACKNOWLEDGEMENT

I extend my heartfelt gratitude to the Almighty for giving me strength in proceeding with this project title “**Autonomous Machine Learning Modelling Using a Task Ontology**”.

I express my heartfelt gratitude to my parents for supporting me in all the ways in every walk of my life.

I express my sincere thanks to Honorable **Dr. S. Ramesh Babu**, Secretary & Correspondent of our college for making necessary arrangement for doing the project

I wish to express my gratitude to **Dr. S. Suresh Kumar**, principal of our college, for giving us permission to carry out this project.

I express my sincere thanks to **Mr. A.N.L. Kumar**, Head of the Department of M.C.A. for his learned suggestions and encouragement which made this project a successful one.

I convey my sincere thanks to my project guide **Mr. A.N.L. Kumar**, Head of the Department of M.C.A. for his learned suggestions and encouragement which made this project a success.

I express my sincere thanks of all the faculty members of our Department of MCA for their valuable support throughout this project work.

I wish to express my thanks to my friends for their enthusiasm, support and encouragement for the completion of this project.

CHAMAKURI PRASAD

Reg NO:17A21F0004.

Autonomous Machine Learning Modeling using a Task Ontology

ABSTRACT

Now a days many researchers are used artificial intelligence technology for strongly engaged in investigation on that recognizes, learns and acts on external information in a wide range of fields those are technologies of computing big data and machine learning algorithms for combining, The artificial intelligence technology is currently used in almost all industries, and many machine learning experts are working on combining and standardizing different machine learning tools so that non-experts will also easily apply them to their particular domain. The researchers are also studying an autonomous machine learning as well as ontology construction for standardizing the machine learning concepts.

This project having two types of functionalities, first one is the take the image input from the user then processing that image it will find object names in the image. Second one is the take the text(sample.txt) file input from the user then processing that text file it will find the meaning to every word. The technical terminology can separate typical problem-solving steps for autonomous machine learning as tasks and present a problem-solving process. We propose the modeling method of an autonomous machine learning using a process of the task execution on machine learning such as workflow.

INDEX

S. NO	CONTENTS	PAGE NO
1	INTRODUCTION	1
2	PROBLEM DEFINITION 2.1 Existing System 2.2 Proposed System:	4
3	LITERATURE SURVEY	6
4	SYSTEM ANALYSIS 4.1 Feasibility Study 4.2 Software Requirement Specification 4.2.1 Functional Requirements 4.2.2 Non-functional Requirements: 4.3 Software Requirements 4.4 Hardware Requirements: 4.5 Functional model of the system	9
5	SYSTEM DESIGN 5.1 Algorithms: 5.2 System Architecture: 5.3 Data Flow Diagram 5.4 UML Diagrams 5.4.1 Class Diagram 5.4.2 Sequence Diagram 5.4.3 Activity Diagram 5.5 Database Design	16

6	SYSTEM IMPLEMENTATION 6.1 Input Design 6.2 Output Design 6.3 About software	29
7	CODE	36
8	SYSTEM TESTING 8.1 System Objective 8.2 Types of Tests	62
9	OUTPUTS	67
10	CONCLUSION	74

1.INTRODUCTION

Artificial intelligence technology has become one of the most essential tools in research and business context recently. Most of the machine learning frameworks are open-source, so the entry of barriers into machine learning are lowered. The typical machine learning frameworks include TensorFlow, Eras [Caffe, Scikit-learn, and Theano implemented in programming languages such as Python, Java, and R. In this respect, many machine learning experts are working on integrating and standardizing various tools so that machine learning nonexperts can easily apply them to their domains. On the other hand, an autonomous machine learning is still in its infancy, and some techniques provide the ability to reduce the unnecessary tasks that are progressively refined to prepare the model and improve its accuracy. The tools of autonomous machine learning provide an optimal algorithm for machine learning tasks and functions to determine the hyper-parameter setting through self-analysis. The typical tools include Auto sklearn, Auto-Weka, H2o Driverless AI and Google's Auto ML. In this paper, we describe a typical problem-solving process for the machine learning as tasks, present their procedure, and propose the modeling method of an autonomous machine learning for using task execution processes. The modeling method of autonomous machine learning based on the task ontology define a structure-based grouping method of the UML (Unified Modeling Language) activities and implement a function to automatically generate models based on common elements and structures. The purpose of the proposed autonomous machine learning model is to model autonomous machine learning by reusing existing resources and producing new knowledge through relearning it.

Task ontologies

Ontology is defined in various fields depending on the field of applications. In the field of artificial intelligence, it is an explicit and formal specification of how objects and concepts described in the field of interest. In the Semantic Web, an ontology plays a very important role in processing, sharing, and reusing the knowledge for exchanging information between different databases. An ontology is also defined as an explicit description of concepts, attributes, constraints, and relationships between them on the domains. On the other hand, domain ontology can be defined as an 'explicit protocol for conceptualization' of the problem. A task ontology is defined as 'extracting and organizing the concepts and relations existing in the problem-solving process domain-independent'. In particular, a task ontology is a specification of the concept structure for

the task execution process. Thus, the core concept is the subject of processing and the procedure of processing for a problem solving. In general, a person becomes a subject in a task ontology. However, in this paper, agents (programs) become subjects to perform the tasks. Expose ontology is an ontology for machine learning experiments. It is used in openML as a data structuring and data sharing(API) method. Machine Learning (ML) Schema is used to export all openML as linked open data. The DMOP ontology is explicitly designed to support data mining and machine learning. This covers the structure and parameters of predictive models, associated cost functions, and optimization strategies. Onto MD ontology provides a unique framework for data mining research.

Machine learning ontologies

ML schema is a top-level ontology that provides classes, properties, and constraints for machine learning algorithms, datasets and experimentation suggested by the W3C(ML Schema community group). It can be easily extended and refined and can be mapped to other domain ontologies developed in the field of machine learning and data mining. MEX vocabulary has been designed to solve the share of provenance information in a lightweight form. The extended PROV ontology provides a model for representing, capturing, and sharing provision information on the Web. This can enable the use of analytical data and code so that another person can reuse the results. The code and the markup language are written in a single file and processed to create a document. A provenance meta information was proposed as a standard model of data management by W3C. The provenance information is also “information about entities, activities, and people involved in producing a piece of data which can be used to form assessments about its quality, reliability or trustworthiness”. As a standard query language, SPARQL is a query language similar to SQL and stored in Resource Description Framework(RDF) for queries on data.

Autonomous machine learning modeling

Autonomous machine learning modeling is the work for standardization and abstraction to the core of the components base on the meta information of the machine learning. The model consists of the task and process and saves as the method library(API). It defines into small units for modular and systematization of its components. The defined components redefine as a UML-based metamodel for the consistency, traceability, reusability, and implementation-ready between tasks

and the results. So, the core class of the UML-based meta-model consist of tasks and processes. The Knowledge of the autonomous machine learning also describes a small task unit based on the MEXvocabulary. depicts a part of the knowledge of the object detection using the “YOLO” of the deep learning algorithm. The machine learning pipeline for object detection consists of data import, decision of attribute selection or schema, selection of learning model, construction of learning model, hyper-parameter setting, model training, measurement of model performance, and so on. In this way, the knowledge representation of the project unit is written as ‘.json’ files using the mapping rules based on the machine learning schema and the vocabulary and convert it into a UML-based meta-model. This model makes that objectives, optimizers, metrics, and layers in the Keras API are meta-model for deep learning.

2.PROBLEM DEFINITION

2.1 Existing System

The domain ontology can be defined as an 'explicit protocol for conceptualization' of the problem. A task ontology is defined as 'extracting and organizing the concepts and relations existing in the problem-solving process domain-independent'. In particular, a task ontology is a specification of the concept structure for the task execution process. Thus, the core concept is the subject of processing and the procedure of processing for a problem solving. In general, a person becomes a subject in a task ontology. However, in this paper, agents (programs) become subjects to perform the tasks.

Disadvantages of Existing System:

- An autonomous machine learning is still in its infancy, and some techniques provide the ability to reduce the unnecessary tasks that are progressively refined to prepare the model and improve its accuracy.

2.2 Proposed System:

In the proposed system, we describe a typical problem-solving process for the machine learning as tasks, present their procedure, and propose the modeling method of an autonomous machine learning for using task execution processes. The modeling method of autonomous machine learning based on the task ontology define a structure-based grouping method of the UML (Unified Modeling Language) activities and implement a function to automatically generate models based on common elements and structures. The purpose of the proposed autonomous machine learning model is to model autonomous machine learning by reusing existing resources and producing new knowledge through relearning it.

Advantages of Proposed System:

- ML schema is a top-level ontology that provides classes, properties, and constraints for machine learning algorithms, datasets and experimentation suggested by the W3C (ML Schema community group).

- It can be easily extended and refined and can be mapped to other domain ontologies developed in the field of machine learning and data mining.

3.LITERATURE SURVEY

1) TensorFlow: A system for large-scale machine learning

AUTHORS: Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

TensorFlow is a machine learning system that operates at large scale and in heterogeneous environments. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. It maps the nodes of a dataflow graph across many machines in a cluster, and within a machine across multiple computational devices, including multicore CPUs, general purpose GPUs, and custom-designed ASICs known as Tensor Processing Units (TPUs). This architecture gives flexibility to the application developer: whereas in previous “parameter server” designs the management of shared state is built into the system, TensorFlow enables developers to experiment with novel optimizations and training algorithms. TensorFlow supports a variety of applications, with a focus on training and inference on deep neural networks. Several Google services use TensorFlow in production, we have released it as an open-source project, and it has become widely used for machine learning research. In this paper, we describe the TensorFlow dataflow model and demonstrate the compelling performance that TensorFlow achieves for several real-world applications.

2) Caffe: Convolutional Architecture for Fast Feature Embedding

AUTHORS: Yangqing Jia*, Evan Shelhamer*, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell SUBMITTED to ACM MULTIMEDIA 2014 OPEN SOURCE SOFTWARE COMPETITION UC Berkeley EECS, Berkeley, CA 94702

Caffe provides multimedia scientists and practitioners with a clean and modifiable framework for state-of-the-art deep learning algorithms and a collection of reference models. The framework is a BSD-licensed C++ library with Python and MATLAB bindings for training and deploying general purpose convolutional neural networks and other deep models efficiently on commodity architectures. Caffe fits industry and internet-scale media needs by CUDA GPU computation, processing over 40 million images a day on a single K40 or Titan GPU (≈ 2.5 ms per image). By

separating model representation from actual implementation, Caffe allows experimentation and seamless switching among platforms for ease of development and deployment from prototyping machines to cloud environments. Caffe is maintained and developed by the Berkeley Vision and Learning Center (BVLC) with the help of an active community of contributors on GitHub. It powers ongoing research projects, large-scale industrial applications, and startup prototypes in vision, speech, and multimedia.

3. Exposé: An ontology for data mining experiments

AUTHORS: V. Joaquin, S. Larisa

Research in machine learning and data mining can be speeded up tremendously by moving empirical research results out of people's heads and labs, onto the network and into tools that help us structure and filter the information. This paper presents Exposé, an ontology to describe machine learning experiments in a standardized fashion and support a collaborative approach to the analysis of learning algorithms. Using a common vocabulary, data mining experiments and details of the used algorithms and datasets can be shared between individual re-searchers, software agents, and the community at large. It enables open repositories that collect and organize experiments by many researchers. As can be learned from recent developments in other sciences, such a free exchange and reuse of experiments requires a clear representation. We therefore focus on the design of an ontology to express and share experiment meta-data with the world.

4. Task ontology: Ontology for building conceptual problem-solving models

AUTHORS: I. Mitsuru, S. Kazuhisa, K. Osamu, M. Riichiro

We have investigated the property of problem-solving knowledge and tried to design its ontology, that is, Task ontology. The main purpose of this paper is to illustrate a Conceptual Level Programming Environment (named CLEPE) as an implemented system based on Task ontology. CLEPE provides three major advantages as follows. (A) It provides human-friendly primitives in terms of which users can easily describe their own problem-solving process (descriptiveness, readability). (B) The systems with task ontology can simulate the problem-solving process at an abstract level in terms of conceptual level primitives (conceptual level operationality). (C) It provides ontology author with an environment for building task ontology so that he/she can build a consistent and useful ontology. In this paper, firstly we briefly introduce the concept of task

ontology. Secondly, CLEPE and its design principle is described. In CLEPE, one can represent his/her own problem-solving knowledge and realize the conceptual-level execution.

5. Models for Representing Task Ontologies

AUTHORS: A. F. Martins, R. A. F. De

Knowledge is of general utility and should be captured thinking in reuse. A key idea underlining knowledge capturing for reuse is to consider that there are two major kinds of knowledge: domain and task knowledge. Ontologies can be used for representing both kinds of knowledge. However, while domain ontologies are broadly used and there are many proposals of models for representing them, the same does not occur for task ontologies. In this paper we propose the use of UML activity diagrams for capturing task control-flow, and UML class diagrams for capturing the knowledge roles involved in its activities. We also discuss the interrelationship between these two models and how they can be combined with domain ontologies in order to describe the knowledge involved in a class of applications.

4.SYSTEM ANALYSIS

4.1 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ◆ **ECONOMICAL FEASIBILITY**
- ◆ **TECHNICAL FEASIBILITY**
- ◆ **SOCIAL FEASIBILITY**

4.1.2 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.1.3 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.1.4 Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel

threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.2 Software Requirement Specification

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

Purpose:

This document is to provide more information Autonomous Machine Learning. The Autonomous Machine Learning is using for processing images and text files.

Scope:

The scope of this document we can separate typical problem-solving steps for autonomous machine learning as tasks and present a problem-solving process. We propose the modeling method of an autonomous machine learning using a process of the task execution on machine learning such as workflow.

4.2.1 Functional Requirements:

Functional requirements are important to construction your requirement creation. A functional requirement document helps you to define the functionality of a system or one of its subsystems.

➤ Admin

- Login
- View User Data
- User Activation
- Logout

➤ **User**

- Registration(Login)
- Upload Text
- Upload Images
- Find Vocabulary
- Detecting Objects
- View Reviews
- Logout

The algorithm that are used in these projects are

- **Convolutional Neural networks:** Convolutional Neural networks are designed to process data through multiple layers of arrays. This type of neural networks is used in applications like image recognition or face recognition. The primary difference between CNN and any other ordinary neural network is that CNN takes input as a two-dimensional array and operates directly on the images rather than focusing on feature extraction which other neural networks focus on.
- **k nearest neighbors:** K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition.

4.2.2 Non-functional Requirements:

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. The following are the important non-functional requirements including our system.

- Performance
- Safety
- Security
- Internet

4.3 Software Requirements

- ❖ **Operating system** : Windows 10
- ❖ **Coding Language** : Python.
- ❖ **Front-End** : Python,Django.
- ❖ **Designing** : Html, CSS, JavaScript.
- ❖ **Data Base** : MySQL

Debugger and Emulator

- Any Browser (Particularly Chrome)

4.4 Hardware Requirements:

- ❖ **System** : intel core i3 7th generation.
- ❖ **Hard Disk** : 1 TB.
- ❖ **Monitor** : 15.6' Colour Monitor.
- ❖ **Mouse** : Optical Mouse.
- ❖ **Ram** : 4 GB.

4.5 Functional model of the system

Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

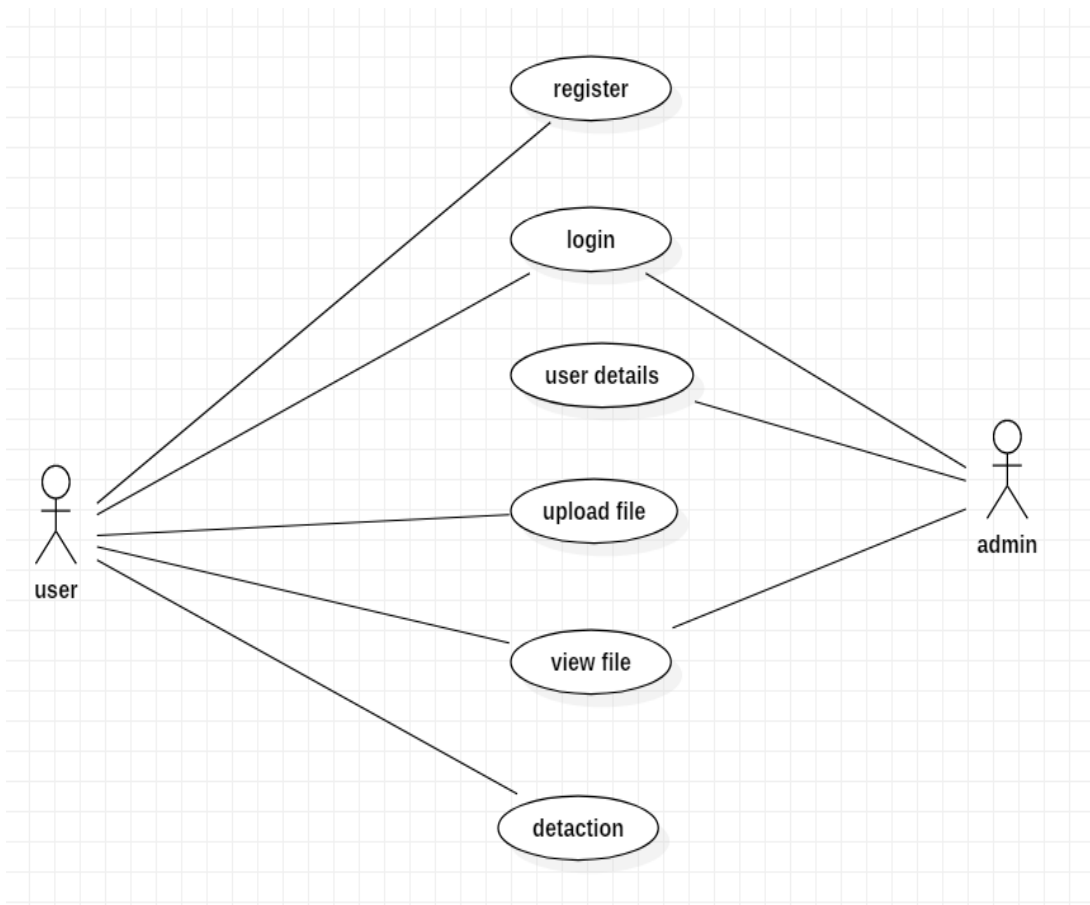


Fig: use case

Use Case Descriptions:

1. User Register Use Case:

Use case name: User Register

Participating Actors: User

Pre-condition: User must have the basic details to register in this page.

Flow of Events:

- Enter all the required details.
- Enter into Horne Page.

Post Condition: If the user is success fully registered by giving the required details then he/she enter into the login page.

2. Admin Login Use Case:

Use case name: Admin Login

Participating Actors: Admin

Pre-Condition: Admin must have the valid username, password.

Flow of Events:

- Verify the username.
- Verify the password.
- Login into the Admin page.

Post Condition: If the name, password is valid then the Admin enter into the Admin page, otherwise it will ask for the valid username and password.

3. Activate User Use Case:

Use case name: Activate User

Participating Actors: Admin

Pre-Condition: Admin can activate the registered user.

Flow of Events:

- Admin enter into Admin page by entering Admin id and password.
- Admin activate the registered user into admin page and click to view user data.

Post Condition: If the Admin activate the registered user can upload the files.

4. User Login Use Case:

Use case name: User Login

Participating Actors: User

Pre-Condition: User must have the valid username, password to enter into the system.

Flow of Events:

- Verify the username.
- Verify the password.
- Login into the User page.

Post Condition: If the name, password is valid then the User enter into the User page, otherwise it will ask for the valid username and password.

5. Upload File use case:

Use case name: Upload File

Participating Actors: User

Pre-Condition: The user can upload the file.

Flow of Events:

- User must select the file.
- Upload the file.

Post Condition: Successfully the file will be uploaded.

6. View file data use case:

Use case name: View file data

Participating Actors: User

Pre-Condition: The user can click the file data and select view and find the file.

Flow of Events:

- User must select the file data.
- User must select either View and find
- User view the result of file.

5.SYSTEM DESIGN

5.1 Algorithms:

They are two algorithms used in this project that are Convolutional Neural networks (CNN) and K-nearest neighbors (KNN). The detail explanation of two algorithms is given below.

Convolutional Neural networks : image processing

K-nearest neighbors : identifying words in the paragraph

1.Convolutional Neural networks

Convolutional Neural networks are designed to process data through multiple layers of arrays. This type of neural networks is used in applications like image recognition or face recognition. The primary difference between CNN and any other ordinary neural network is that CNN takes input as a two-dimensional array and operates directly on the images rather than focusing on feature extraction which other neural networks focus on.

The dominant approach of CNN includes solutions for problems of recognition. Top companies like Google and Facebook have invested in research and development towards recognition projects to get activities done with greater speed.

Let us understand these ideas in detail.

CNN utilizes spatial correlations that exist within the input data. Each concurrent layer of a neural network connects some input neurons. This specific region is called local receptive field. Local receptive field focusses on the hidden neurons. The hidden neurons process the input data inside the mentioned field not realizing the changes outside the specific boundary.

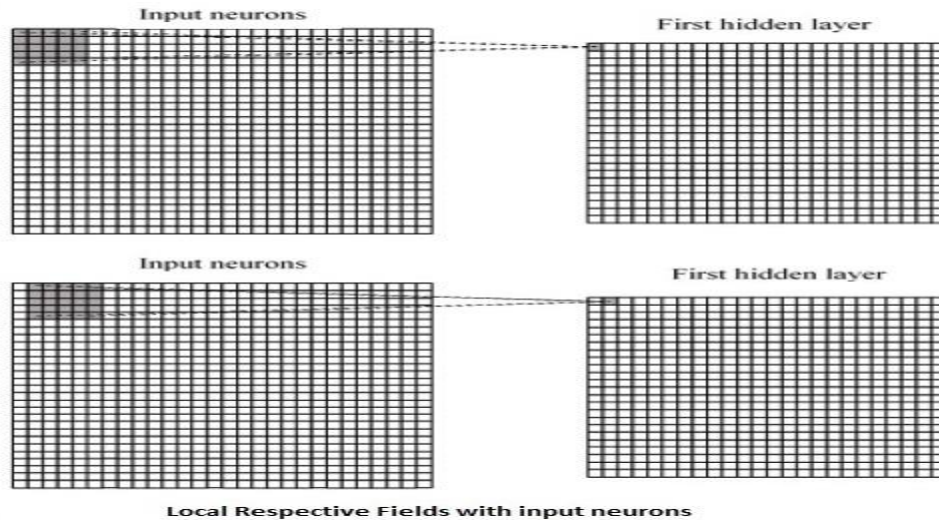


Fig: CNN image processing

If we observe the above representation, each connection learns a weight of the hidden neuron with an associated connection with movement from one layer to another. Here, individual neurons perform a shift from time to time. This process is called “convolution”.

The mapping of connections from the input layer to the hidden feature map is defined as “shared weights” and bias included is called “shared bias”.

CNN or convolutional neural networks use pooling layers, which are the layers, positioned immediately after CNN declaration. It takes the input from the user as a feature map that comes out of convolutional networks and prepares a condensed feature map. Pooling layers helps in creating layers with neurons of previous layers.

2.K-nearest neighbors (KNN) algorithm

K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new data points which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps –

Step 1 – For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 – Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 – For each point in the test data do the following –

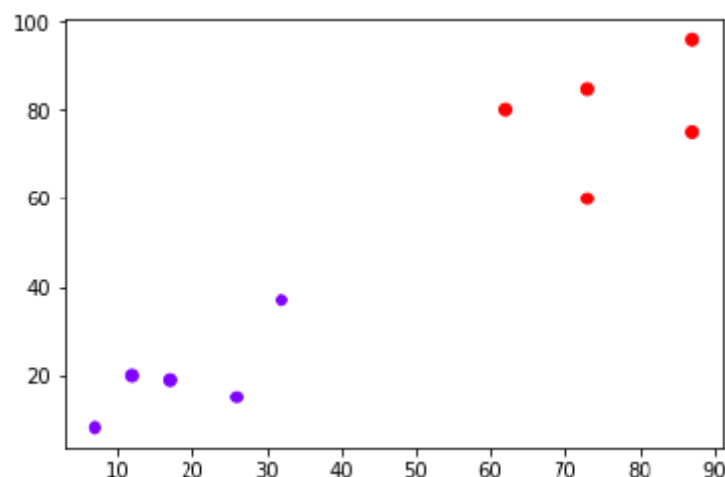
- **3.1** – Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- **3.2** – Now, based on the distance value, sort them in ascending order.
- **3.3** – Next, it will choose the top K rows from the sorted array.
- **3.4** – Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 – End

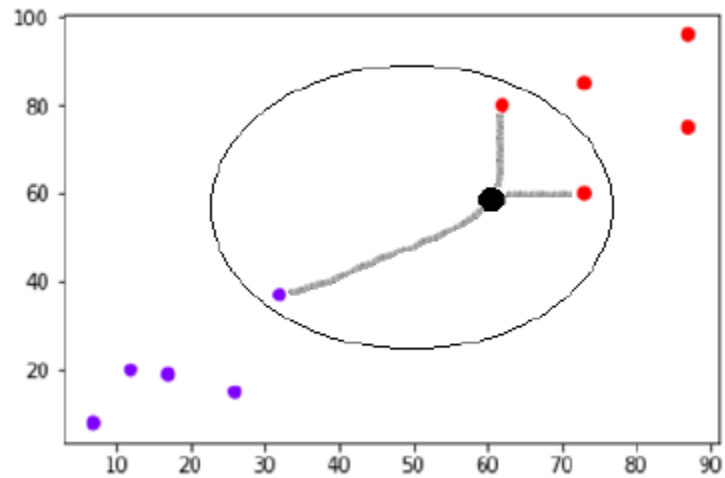
Example

The following is an example to understand the concept of K and working of KNN algorithm

Suppose we have a dataset which can be plotted as follows –



Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming $K = 3$ i.e. it would find three nearest data points. It is shown in the next diagram –



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

5.2 System Architecture:

A System architecture is the conceptual model, the conceptual model is used for defines the structure, behaviour, and more views of a system. An architecture description is a formal description and the formal description is used for shows the performance of a System behaviour. A System architecture is containing the system components and the sub-systems developments

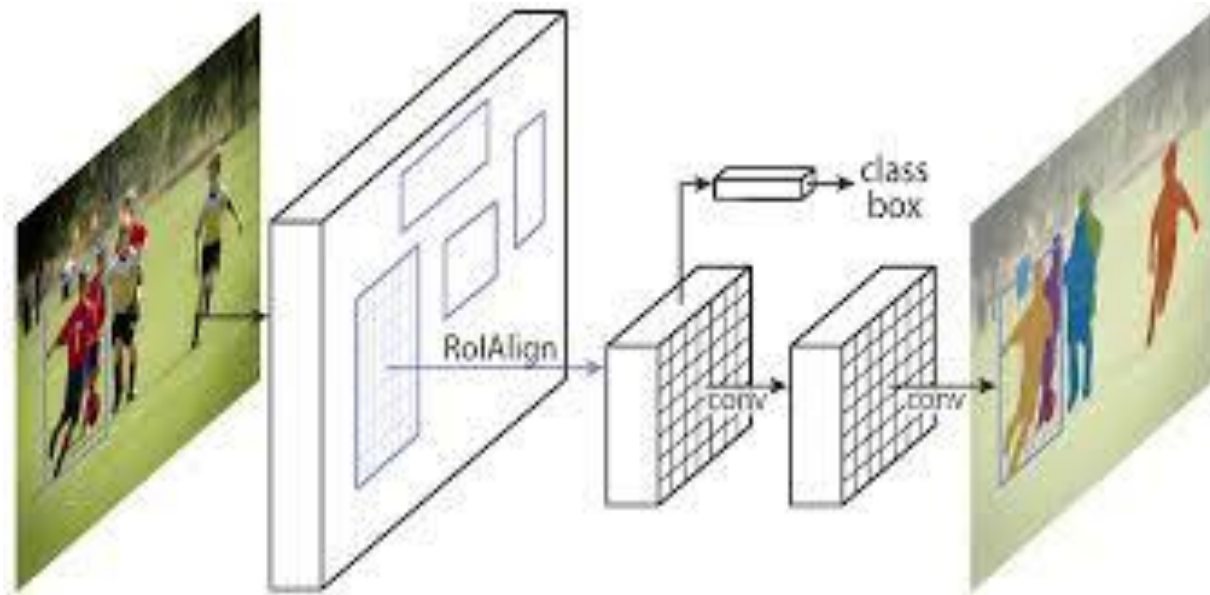


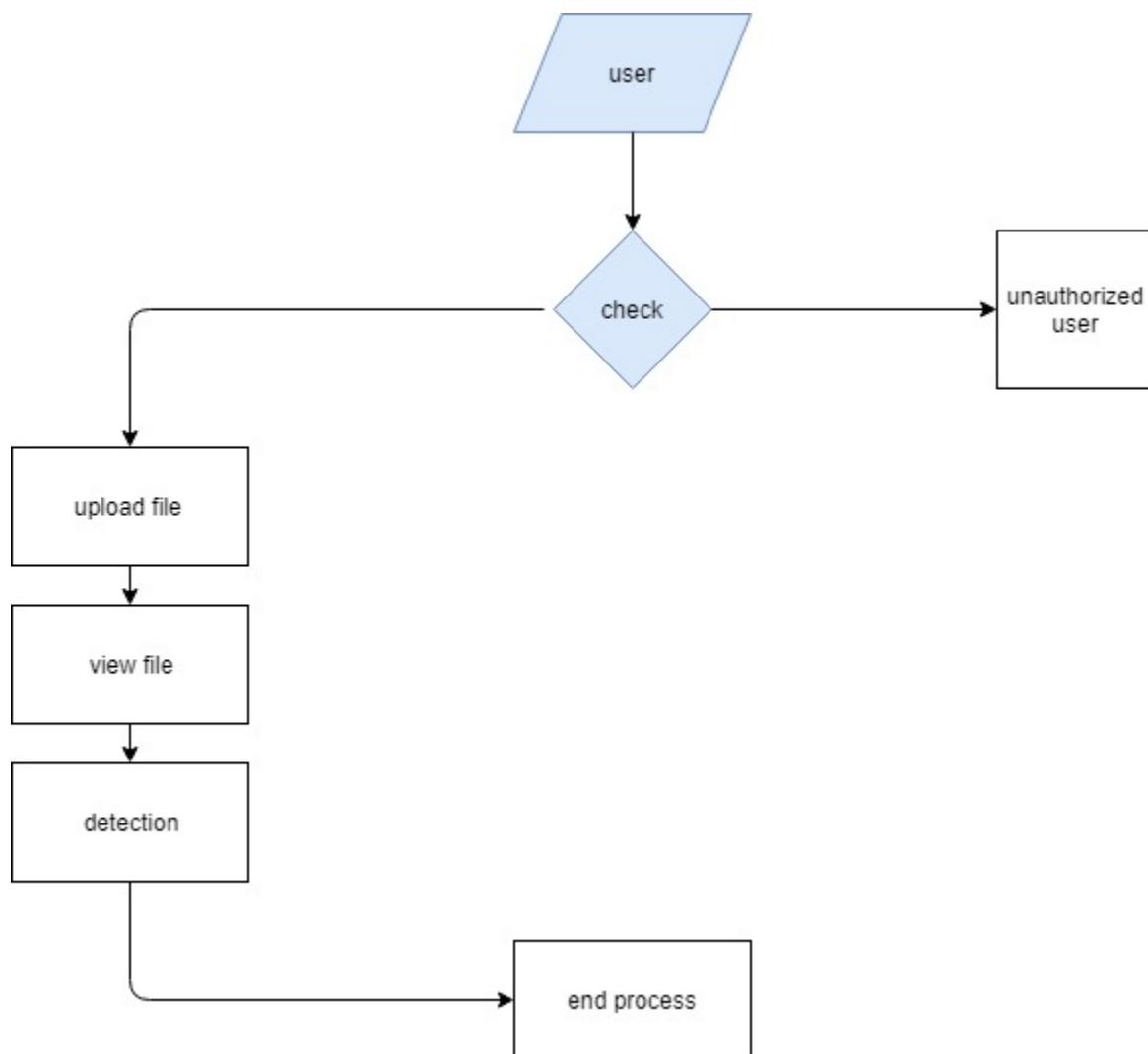
Fig: image processing

5.3 Data Flow Diagram:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by

the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



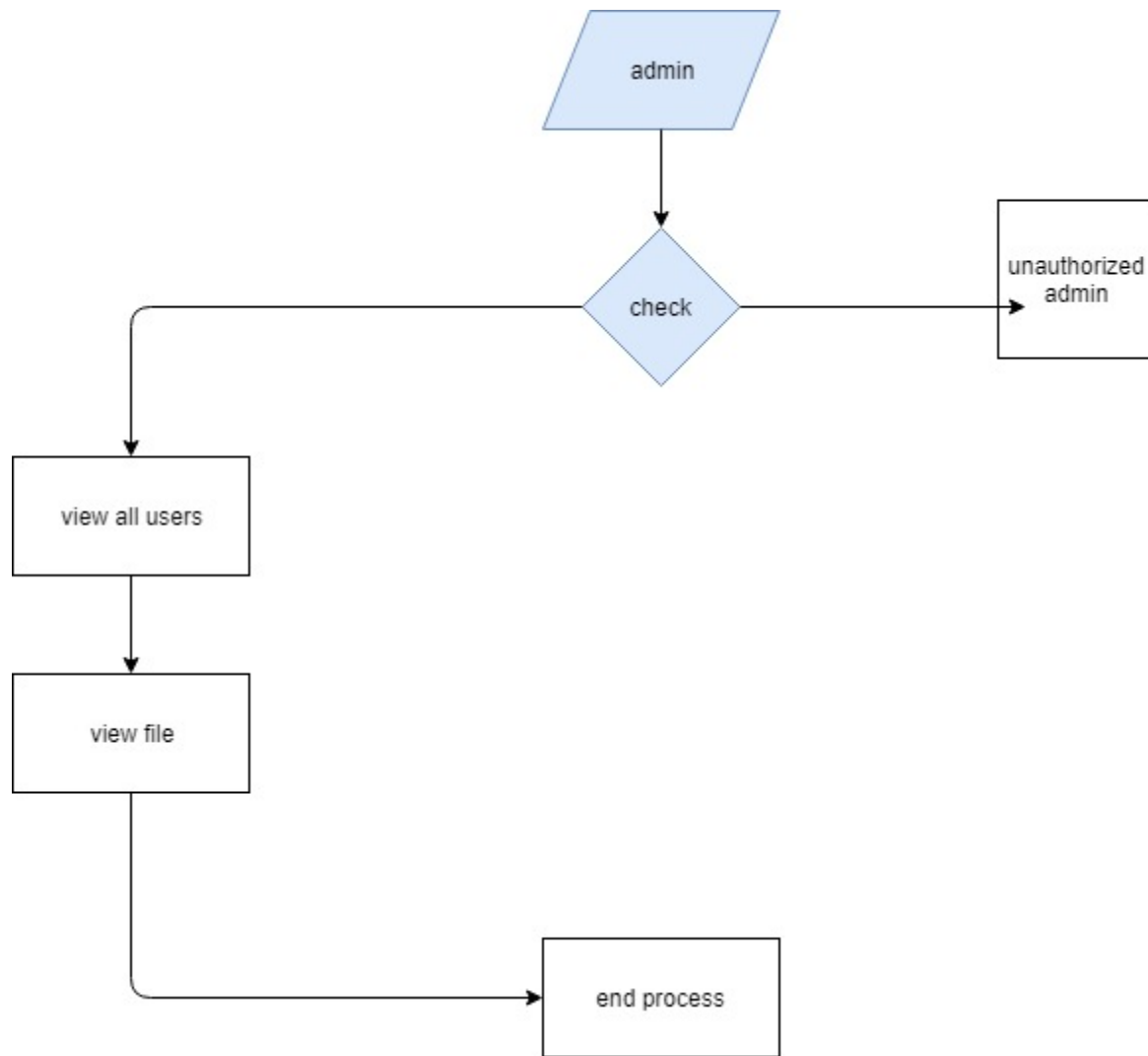


Fig: Data Flow Diagram

5.4 UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

5.4.1 Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



Fig: Class Diagram

5.4.2 Sequence Diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

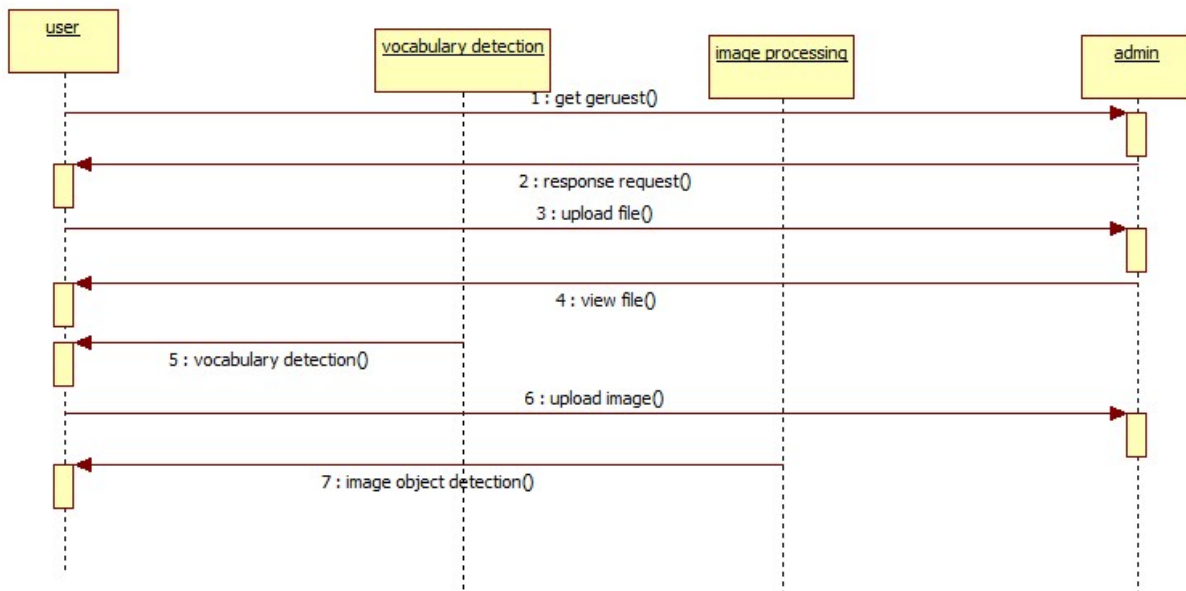


Fig: Sequence Diagram:

5.4.3 Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

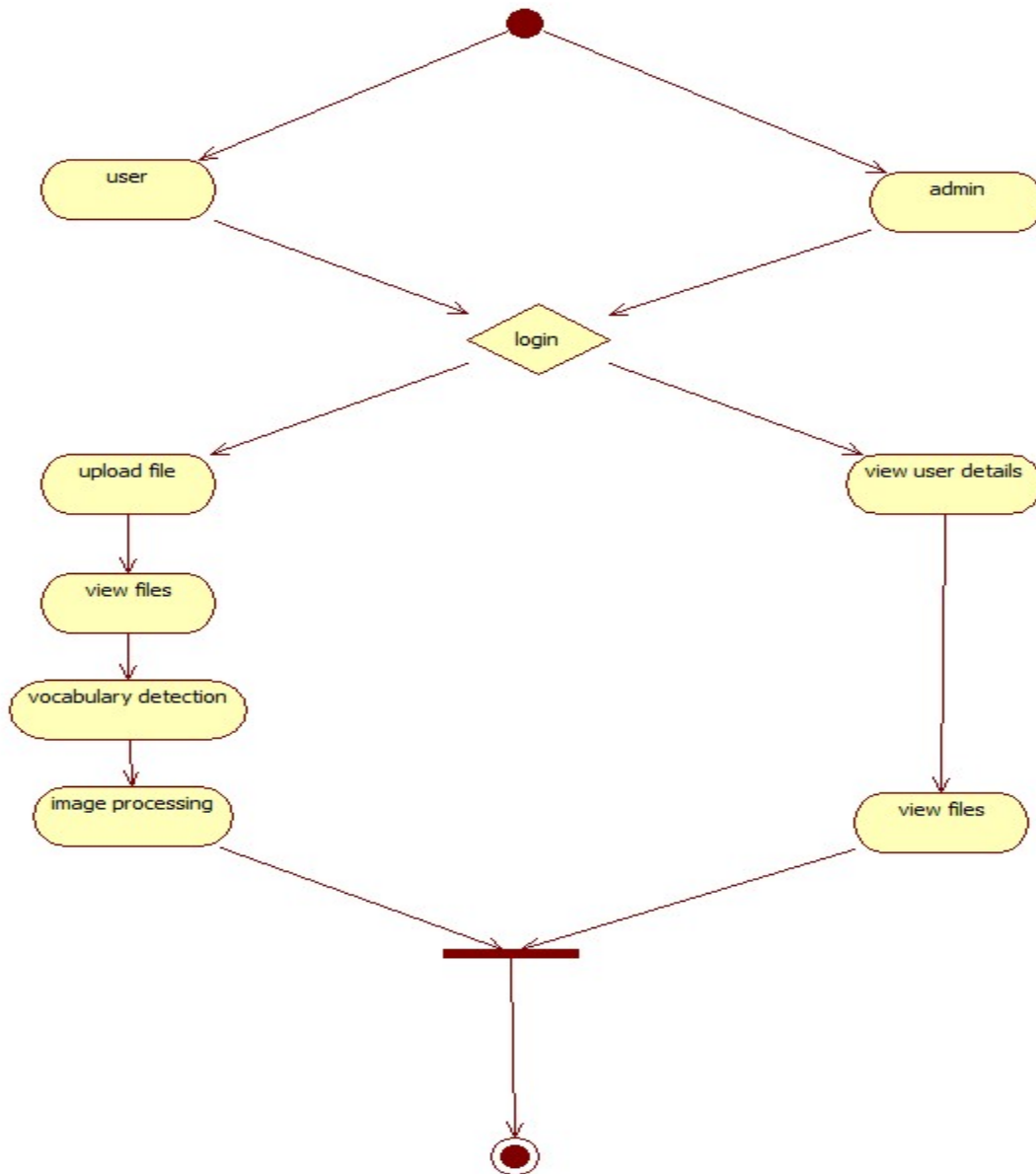


Fig: Activity Diagram

5.5 Database Design:

Normal Forms:

First Normal Form (1NF)

First normal form (1NF) sets the fundamental rules for database normalization and relates to a single table within a relational database system. Normalization follows three basic steps, each building on the last. The first of these is the first normal form.

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single (atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

Second Normal Form (2NF)

Second normal form (2NF) is the second step in normalizing a database. 2NF builds on the first normal form (1NF).

Normalization is the process of organizing data in a database so that it meets two basic requirements:

- There is no redundancy of data (all data is stored in only one place).
- Data dependencies are logical (all related data items are stored together).

A 1NF table is in 2NF form if and only if all of its non-prime attributes are functionally dependent on the whole of every candidate key

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

To understand what is Partial Dependency and how to normalize a table to 2nd normal form, jump to the Second Normal Form .

Third Normal Form (3NF)

Third normal form (3NF) is the third step in normalizing a database and it builds on the first and second normal forms, 1NF and 2NF.

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

But we suggest you to first study about the second normal form and then head over to the third normal form.

User Registration

S. No	Field	Data Type	Size	Constraint
1	Login ID	Varchar	30	Not Null
2	Mail	Varchar	30	Not Null
3	Password	Varchar	20	Not Null
4	Mobile	Varchar	10	Not Null
5	Place	Varchar	30	Not Null
6	City	Varchar	30	Not Null

User Login

S. No	Field	Data Type	Size	Constraint
1	Login ID	Varchar	30	Not Null
2	Password	Varchar	30	Not Null
3	image	LONGBLOB	Below 1MB	Not Null
4	Text	Varchar	150	Not Null

6.SYSTEM IMPLEMENTATION

6.1 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Objectives

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

6.2 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

6.3 About software

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating

systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

6.3.1 Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

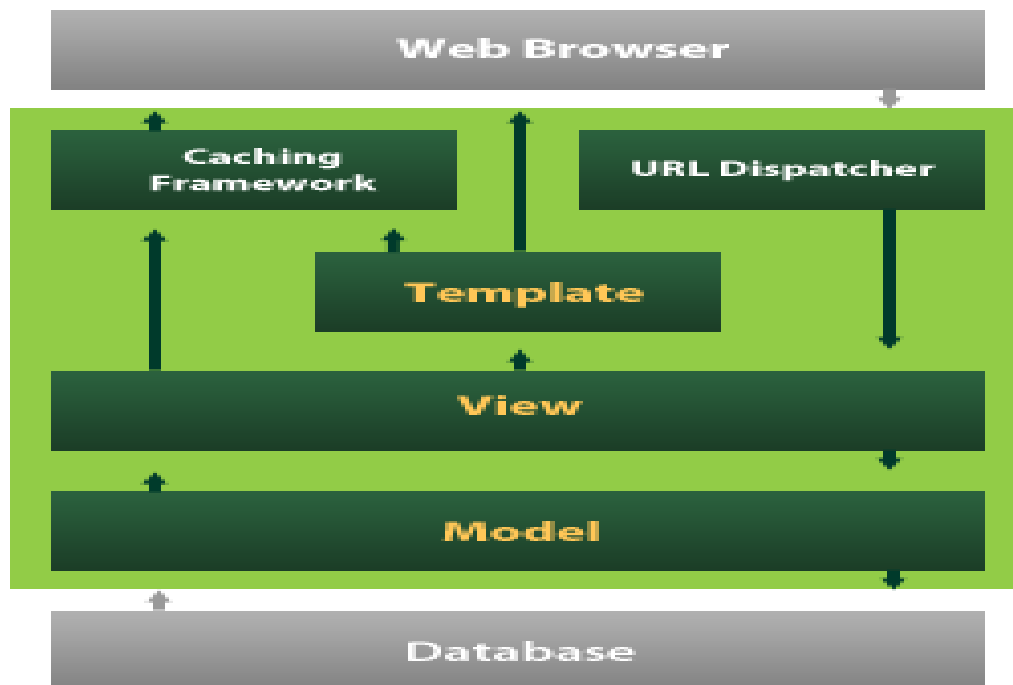
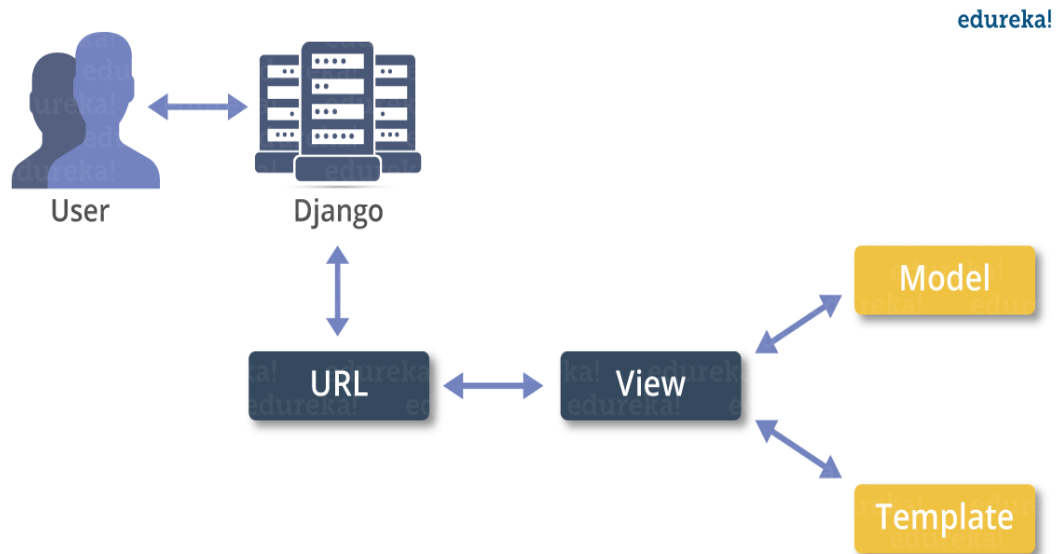


Fig: Application Architecture

Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models



Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code –

```
$ django-admin startproject myproject
```

This will create a "myproject" folder with the following structure –

```
myproject/
```

```
    manage.py
```

```
    myproject/
```

```
        __init__.py
```

settings.py

urls.py

wsgi.py

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements –

manage.py – This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code –

```
$ python manage.py help
```

The “myproject” subfolder – This folder is the actual python package of your project. It contains four files –

__init__.py – Just for python, treat this folder as package.

settings.py – As the name indicates, your project settings.

urls.py – All links of your project and the function to call. A kind of ToC of your project.

wsgi.py – If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set –

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to 'True' for a live project. However, this has to be set to 'True' if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.sqlite3',  
        'NAME': 'database.sql',  
        'USER': '',  
        'PASSWORD': '',  
        'HOST': '',  
        'PORT': '',  
    }  
}
```

Database is set in the 'Database' dictionary. The example above is for SQLite engine. As stated earlier, Django also supports –

MySQL (django.db.backends.mysql)

PostgreSQL (django.db.backends.postgresql_psycopg2)

Oracle (django.db.backends.oracle) and NoSQL DB

MongoDB (django_mongodb_engine)

Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE...

```
$ python manage.py runserver
```

You will get something like the following on running the above code –

Validating models...

0 errors found

September 03, 2015 - 11:41:50

Django version 1.6.11, using settings 'myproject.settings'

Starting development server at http://127.0.0.1:8000/

Quit the server with CONTROL-C.

7. CODE

Urls.py

```
"""Taskontology URL Configuration
```

The `urlpatterns` list routes URLs to views. For more information please see:

<https://docs.djangoproject.com/en/2.2/topics/http/urls/>

Examples:

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `path("", views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `path("", Home.as_view(), name='home')`

Including another `URLconf`

1. Import the `include()` function: `from django.urls import include, path`
2. Add a URL to `urlpatterns`: `path('blog/', include('blog.urls'))`

```
"""
```

```
from django.conf.urls import url
```

```
from django.contrib import admin
```

```
from django.urls import path
```

```
from django.conf import settings
```

```

from django.conf.urls.static import static

from Taskontology.views import adminlogin, adminloginaction, logout, userdetails, activateuser,
userfiles, adminhome

from user.views import index, base, registration, user, userlogincheck, userhome, uploadfile,
viewuserfiles, \

    findvocabulary, detection, imagedetect


urlpatterns = [

    url(r'^admin/', admin.site.urls),

    url(r'^$', index, name="index"),

    url(r'^index/', index, name="index"),

    url(r'^base/', base, name="base"),

    url(r'^registration/', registration, name="registration"),

    url(r'^user/', user, name="user"),

    url(r'^adminhome/', adminhome, name="adminhome"),

    url(r'^adminlogin/', adminlogin, name="adminlogin"),

    url(r'^adminloginaction/', adminloginaction, name="adminloginaction"),

    url(r'^logout/', logout, name="logout"),

    url(r'^userdetails/', userdetails, name="userdetails"),

    url(r'^userfiles/', userfiles, name="userfiles"),

    url(r'^activateuser/', activateuser, name="activateuser"),

    url(r'^userlogincheck/', userlogincheck, name="userlogincheck"),

```

```
url(r'^userhome/', userhome, name="userhome"),

url(r'^uploadfile/', uploadfile, name="uploadfile"),

url(r'^viewuserfiles/', viewuserfiles, name="viewuserfiles"),

url(r'^findvocabulary/', findvocabulary, name="findvocabulary"),

url(r'^detection/', detection, name="detection"),

url(r'^imagedetect', imagedetect, name="imagedetect")

]

if settings.DEBUG:

    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Main Views.py

```
from random import randint

from django.shortcuts import render

from django.contrib import messages

from user.models import registrationmodel, uploadmodel

def adminhome(request):

    return render(request, 'admin/adminhome.html')
```

```
def adminlogin(request):

    return render(request,'admin/adminlogin.html')


def adminloginaction(request):

    if request.method == "POST":

        if request.method == "POST":

            login = request.POST.get('username')

            print(login)

            pswd = request.POST.get('password')

            if login == 'admin' and pswd == 'admin':

                return render(request,'admin/adminhome.html')

            else:

                messages.success(request, 'Invalid user id and password')

                #messages.success(request, 'Invalid user id and password')

            return render(request,'admin/adminlogin.html')

def logout(request):

    return render(request,'index.html')

def userdetails(request):

    userdata = registrationmodel.objects.all()

    return render(request,'admin/viewuserdetails.html', {'object': userdata})
```

```

def userfiles(request):

    userfile = uploadmodel.objects.all()

    return render(request, 'admin/viewfiles.html',{'object': userfile})

def activateuser(request):

    if request.method=='GET':

        usid = request.GET.get('usid')

        authkey = random_with_N_digits(8)

        status = 'activated'

        print("USID = ",usid,authkey,status)

        registrationmodel.objects.filter(id=usid).update(authkey=authkey , status=status)

        userdata = registrationmodel.objects.all()

        return render(request,'admin/viewuserdetails.html',{'object':userdata})

def random_with_N_digits(n):

    range_start = 10**(n-1)

    range_end = (10**n)-1

    return randint(range_start, range_end)

```

User **views.py**

```

import nltk

from PIL import Image

from django.conf import settings

from django.shortcuts import render, HttpResponseRedirect, redirect

```

```
from django.contrib import messages

# Create your views here.

from nltk import word_tokenize

from nltk.corpus import wordnet as wn

from user.forms import registrationform, UploadfileForm

from user.models import registrationmodel, uploadmodel

from msilib.schema import File

from nltk.corpus import wordnet


import numpy as np

import argparse

import time

import cv2

import os


def index(request):

    return render(request, "index.html")


def base(request):

    return render(request, "base.html")
```



```
def registration(request):

    if request.method == 'POST':

        form = registrationform(request.POST)

        if form.is_valid():

            # print("Hai Meghana")

            form.save()

            messages.success(request, 'you are successfully registred')

            return HttpResponseRedirect('trainer')

        else:

            print('Invalid')

    else:

        form = registrationform()

    return render(request, "user/registration.html", {'form': form})


def userhome(request):

    return render(request, "user/userhome.html")


def user(request):

    return render(request, "user/user.html")


def userlogincheck(request):
```

```

if request.method == 'POST':

    usid = request.POST.get('loginid')

    print(usid)

    pswd = request.POST.get('password')

    print(pswd)

    try:

        check = registrationmodel.objects.get(loginid=usid, password=pswd)

        # print('usid',usid,'pswd',pswd)

        request.session['userid'] = check.loginid

        status = check.status

        if status == "activated":

            request.session['email'] = check.email

            #auth.login(request, usid)

            return render(request,'user/userpage.html')

        else:

            messages.success(request, 'user is not activated')

            return render(request,'user/user.html')

    except Exception as e:

        print('Exception is ', str(e))

        messages.success(request,'Invalid user id and password')

```

```

    return render(request, 'user/user.html')

def uploadfile(request):
    if request.method == 'POST':
        form = UploadfileForm(request.POST, request.FILES)

        if form.is_valid():
            form.save()

            return redirect('user/upload_list.html')

    else:
        form = UploadfileForm()

    return render(request, 'user/uploadfile.html', {'form': form})

def upload_list(request):
    files = File.objects.all()

    return render(request, 'upload_list.html', {'files': files})

def viewuserfiles(request):
    filedata = uploadmodel.objects.all()

    return render(request, 'user/viewuserdata.html', {'object': filedata})

def findvocabulary(request):

```

```

if request.method == "GET":

    file = request.GET.get('id')

    try:

        #check = uploadmodel.objects.get(id=usid)

        #file = check.file

        print("Path is ", settings.MEDIA_ROOT+'/'+file)

        raw = open(settings.MEDIA_ROOT+'/'+file).read()

        #print(raw)

        tokens = word_tokenize(raw)

        #print(tokens)

        words = set(w.lower() for w in nltk.corpus.words.words())

        # tokens1 = word_tokenize(words)

        tokens1 = list(words)

        # print(tokens1)

        voc = set(tokens) & set(tokens1)

        meg = str(voc)

        #print('Word type ',meg)

        word = meg.split(",")

        for x in word:

            #print('X = ',x)

            line = nltk.re.sub('[^ a-zA-Z0-9]', " ", x)

```

```

    #print("Line ", line)

for x in line:

    sysns = wn.synsets(x)

    #print('Rslt ',sysns)

#texts = [[word.lower() for word in text.split()] for text in voc]

#syms = wn.synsets(meg)

#print("synsets:", syms)


dict = {

    "file": file,

    "voc": voc,

    "sysns": sysns,

}

#print(dict)

katti = { }

vcData = dict['voc']

#print(vcData)


try:

    for xword in vcData:

```

```

        #print('for NLTK =',xword)

        syn = wordnet.synsets(xword)

        if len(syn) !=0:

            description = syn[0].definition()

            katti.update({xword:description})

        else:

            pass

    except Exception as e:

        print(e)

        pass


    #print('katti type ',katti)

    dict1 = {

        "katti": katti,

        'dict':dict

    }

    #print("dict1:",dict1)

    #return render(request, "user/vocabulary.html",{ 'dict':dict,'katti':katti })

    #return render(request, "user/vocabulary.html", katti)

    return render(request, "user/vocabulary.html", dict1)

except Exception as e:

```

```

        print('Exception is ', str(e))

        messages.success(request, 'Invalid Details')

    return render(request, 'user/viewuserdata.html')


def detection(request):

    if request.method == 'POST':

        images = request.FILES.get('imgfile')

        print("image:", images)

        img = Image.open(images)

        #print("meghana:", img)

        image = img.save(settings.MEDIA_ROOT + "/cropped_picture.jpg")

        args = {'yolo': 'yolo-coco', 'confidence': 0.5, 'threshold': 0.3} # vars(ap.parse_args())

        print("Volvorine Args ", type(args))

        args.update({'image': image})

        print("Dict Data ", args)

        # load the COCO class labels our YOLO model was trained on

        labelsPath = os.path.sep.join(["yolo-coco/coco.names"])

        LABELS = open(labelsPath).read().strip().split("\n")

        # initialize a list of colors to represent each possible class label

        np.random.seed(42)

```

```

COLORS = np.random.randint(0, 255, size=(len(LABELS), 3), dtype="uint8")

# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join(["yolo-coco/yolov3.weights"])
configPath = os.path.sep.join(["yolo-coco/yolov3.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes)
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# load our input image and grab its spatial dimensions
# image = "F:/Python/Alex Codes/yolo-object-detection/images/soccer.jpg"
# image = cv2.imread(args["image"])
image = cv2.imread(settings.MEDIA_ROOT+"/cropped_picture.jpg")
print("images:",image)

(H, W) = image.shape[:2]

# determine only the *output* layer names that we need from YOLO
ln = net.getLayerNames()

ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]

```



```

# construct a blob from the input image and then perform a forward

# pass of the YOLO object detector, giving us our bounding boxes and

# associated probabilities

blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),

                               swapRB=True, crop=False)

net.setInput(blob)

start = time.time()

layerOutputs = net.forward(ln)

end = time.time()


# show timing information on YOLO

print("[INFO] YOLO took {:.6f} seconds".format(end - start))


# initialize our lists of detected bounding boxes, confidences, and

# class IDs, respectively

boxes = []

confidences = []

classIDs = []


# loop over each of the layer outputs

for output in layerOutputs:

```

```

# loop over each of the detections

for detection in output:

    # extract the class ID and confidence (i.e., probability) of
    # the current object detection

    scores = detection[5:]

    classID = np.argmax(scores)

    confidence = scores[classID]

    # filter out weak predictions by ensuring the detected
    # probability is greater than the minimum probability

    if confidence > args["confidence"]:

        # scale the bounding box coordinates back relative to the
        # size of the image, keeping in mind that YOLO actually
        # returns the center (x, y)-coordinates of the bounding
        # box followed by the boxes' width and height

        box = detection[0:4] * np.array([W, H, W, H])

        (centerX, centerY, width, height) = box.astype("int")

        # use the center (x, y)-coordinates to derive the top and
        # and left corner of the bounding box

        x = int(centerX - (width / 2))

```

```

y = int(centerY - (height / 2))

# update our list of bounding box coordinates, confidences,
# and class IDs

boxes.append([x, y, int(width), int(height)])

confidences.append(float(confidence))

classIDs.append(classID)


# apply non-maxima suppression to suppress weak, overlapping bounding
# boxes

idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],
                        args["threshold"])


# ensure at least one detection exists

if len(idxs) > 0:

    # loop over the indexes we are keeping

    for i in idxs.flatten():

        # extract the bounding box coordinates

        (x, y) = (boxes[i][0], boxes[i][1])

        (w, h) = (boxes[i][2], boxes[i][3])

```

```

        # draw a bounding box rectangle and label on the image

        color = [int(c) for c in COLORS[classIDs[i]]]

        cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)

        text = "{: {:.4f}".format(LABELS[classIDs[i]], confidences[i])

        cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,

                    0.5, color, 2)


    # show the output image

    cv2.imshow("Image", image)

    cv2.waitKey(0)

    return render(request, "user/objectdetect.html")

def imagedetect(request):

    return render(request, "user/imagedetect.html")

```

User forms.py

```

from django import forms

from user.models import registrationmodel, uploadmodel


class registrationform(forms.ModelForm):

```

```
loginid = forms.CharField(widget=forms.TextInput(), required=True, max_length=100)

password = forms.CharField(widget=forms.PasswordInput(), required=True,
max_length=100)

email = forms.EmailField(widget=forms.TextInput(),required=True)

mobile = forms.CharField(widget=forms.TextInput(),required=True,max_length=100)

place = forms.CharField(widget=forms.TextInput(),required=True,max_length=100)

city = forms.CharField(widget=forms.TextInput(),required=True,max_length=100)

authkey = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max_length=100)

status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max_length=100)
```

```
class Meta:
```

```
    model = registrationmodel
```

```
    fields = ['loginid','password','email','mobile','place','city','authkey','status' ]
```

```
class UploadfileForm(forms.ModelForm):
```

```
    class Meta:
```

```
        model = uploadmodel
```

```
        fields = ('filename','file')
```

```
user Models.py
```

```
from django.db import models
```

Create your models here.

```
class registrationmodel(models.Model):
```

```
    loginid = models.CharField(max_length=100)
```

```
    password = models.CharField(max_length=100)
```

```
    email = models.EmailField()
```

```
    mobile = models.CharField(max_length=100)
```

```
    place = models.CharField(max_length=100)
```

```
    city = models.CharField(max_length=100)
```

```
    authkey = models.CharField(max_length=100)
```

```
    status = models.CharField(max_length=100)
```

```
def __str__(self):
```

```
    return self.email
```

```
class uploadmodel(models.Model):
```

```
    filename = models.CharField(max_length=100)
```

```
    file = models.FileField(upload_to='files/pdfs/')
```

```
def __str__(self):
```

```
    return self.filename
```

Adminbase.html

```
{% load static %}  
  
<!DOCTYPE html>  
  
<html lang="en">  
  
<head>  
  
<title>Task Ontology</title>  
  
<meta charset="utf-8">  
  
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  
  
< link href="https://fonts.googleapis.com/css?family=Muli:300,400,700,900" rel="stylesheet">  
  
<link rel="stylesheet" href="{% static 'fonts/icomoon/style.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/jquery-ui.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/owl.carousel.min.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/jquery.fancybox.min.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/bootstrap-datepicker.css' %}">  
  
<link rel="stylesheet" href="{% static 'fonts/flaticon/font/flaticon.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/aos.css' %}">  
  
<link rel="stylesheet" href="{% static 'css/style.css' %}">
```

```

</head>

<body data-spy="scroll" data-target=".site-navbar-target" data-offset="300">

<div class="site-wrap">

<div class="site-mobile-menu site-navbar-target">

  <div class="site-mobile-menu-header">

    <div class="site-mobile-menu-close mt-3">

      <span class="icon-close2 js-menu-toggle"></span>

    </div>

  </div>

  <div class="site-mobile-menu-body"></div>

</div>

<header class="site-navbar py-4 js-sticky-header site-navbar-target" role="banner">

<div class="container-fluid">

  <div class="d-flex align-items-center">

<div class="site-logo mr-auto w-25"><a href="{ % url 'index' % }">Task Ontologys</a></div>

<div class="mx-auto text-center">

  <nav class="site-navigation position-relative text-right" role="navigation">

    <ul class="site-menu main-menu js-clone-nav mx-auto d-none d-lg-block m-0 p-0">

      <li><a href="{ % url 'adminhome' % }" class="nav-link">AdminHome</a></li>

      <li><a href="{ % url 'userdetails' % }" class="nav-link">UserDetails</a></li>

      <li><a href="{ % url 'userfiles' % }" class="nav-link">Viewfiles</a></li>

```


</nav>

</div>

<div class="ml-auto w-25">

<nav class="site-navigation position-relative text-right" role="navigation">

<ul class="site-menu main-menu site-menu-dark js-clone-nav mr-auto d-none d-lg-block m-0 p-0">

<li class="cta">Logout

</nav>

</div>

</div>

</div>

</header>

<!--<div class="intro-section" id="home-section">

```
<div class="slide-1" style="background-image: url({% static 'images/hero_1.jpg' % });" data-
stellar-background-ratio="0.5">
```

```
<div class="container">
```

```
<div class="row align-items-center">
```

```
<div class="col-12">
```

```
<div class="row align-items-center">
```

```
<div class="col-lg-6 mb-4">
```

```
<h1 data-aos="fade-up" data-aos-delay="100">Autonomous Machine Learning
Modeling using a Task Ontology</h1>
```

```
<!--<p class="mb-4" data-aos="fade-up" data-aos-delay="200">Lorem ipsum dolor
sit amet consectetur adipisicing elit. Maxime ipsa nulla sed quis rerum amet natus quas
necessitatibus.</p>
```

```
<p data-aos="fade-up" data-aos-delay="300"><a href="#" class="btn btn-primary py-
3 px-5 btn-pill">Admission Now</a></p>
```

```
</div> <!-- .site-wrap -->
```

```
<script src="{% static 'js/jquery-3.3.1.min.js' % }"></script>
```

```
<script src="{% static 'js/jquery-migrate-3.0.1.min.js' % }"></script>
```

```
<script src="{% static 'js/jquery-ui.js' % }"></script>
```

```
<script src="{% static 'js/popper.min.js' % }"></script>
```

```
<script src="{% static 'js/bootstrap.min.js' % }"></script>
```

```
<script src="{% static 'js/owl.carousel.min.js' % }"></script>
```

```
<script src="{% static 'js/jquery.stellar.min.js'></script>
```

```
<script src="{% static 'js/jquery.countdown.min.js' % }"></script>
```

```

<script src="{% static 'js/bootstrap-datepicker.min.js' %}"></script>

<script src="{% static 'js/jquery.easing.1.3.js' %}"></script>

<script src="{% static 'js/aos.js' %}"></script>

<script src="{% static 'js/jquery.fancybox.min.js' %}"></script>

<script src="{% static 'js/jquery.sticky.js' %}"></script>

<script src="{% static 'js/main.js' %}"></script>

</body>

</html>

```

Vocabulary.html

```

{% extends 'userbase.html' %}

{% load static %}

{% block contents %}

<div class="intro-section" id="home-section">

    <div class="slide-1" style="background-image: url({% static 'images/hero_1.jpg' %});" data-
stellar-background-ratio="0.5">

        <div class="container">

            <div class="row align-items-center">

                <div class="col-12">

                    <div class="row align-items-center">

                        <div class="col-lg-6 mb-4">

```

```

    <h1 data-aos="fade-up" data-aos-delay="100">&nbsp;</h1>

    <!--<p class="mb-4" data-aos="fade-up" data-aos-delay="200">Lorem ipsum dolor
    sit amet consectetur adipisicing elit. Maxime ipsa nulla sed quis rerum amet natus quas
    necessitatibus.</p>

    <p data-aos="fade-up" data-aos-delay="300"><a href="#" class="btn btn-primary py-
    3 px-5 btn-pill">Admission Now</a></p>-->

    <table border="2px">

        <tr style="color:RED"><th>S.No</th><th>Word</th><th>Defination</th></tr>

        {% for key,value in katti.items% }

            <tr style="color:orange">

                <td>{{ forloop.counter }}</td>

                <td>{{ key }}</td>

                <td>{{ value }}</td>

            </tr>

        {% endfor %}

    </table>

</div>

<div class="col-lg-5 ml-auto" data-aos="fade-up" data-aos-delay="500">

    <form action="" method="POST" class="form-box">

<h3 class="h4 text-black mb-4">Vocabulary Words</h3>

    <table >

        Results = {{ dict }}</table> </div>{% endblock %}

```

8.SYSTEM TESTING

8.1 System Objective

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.2 Types of Tests

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

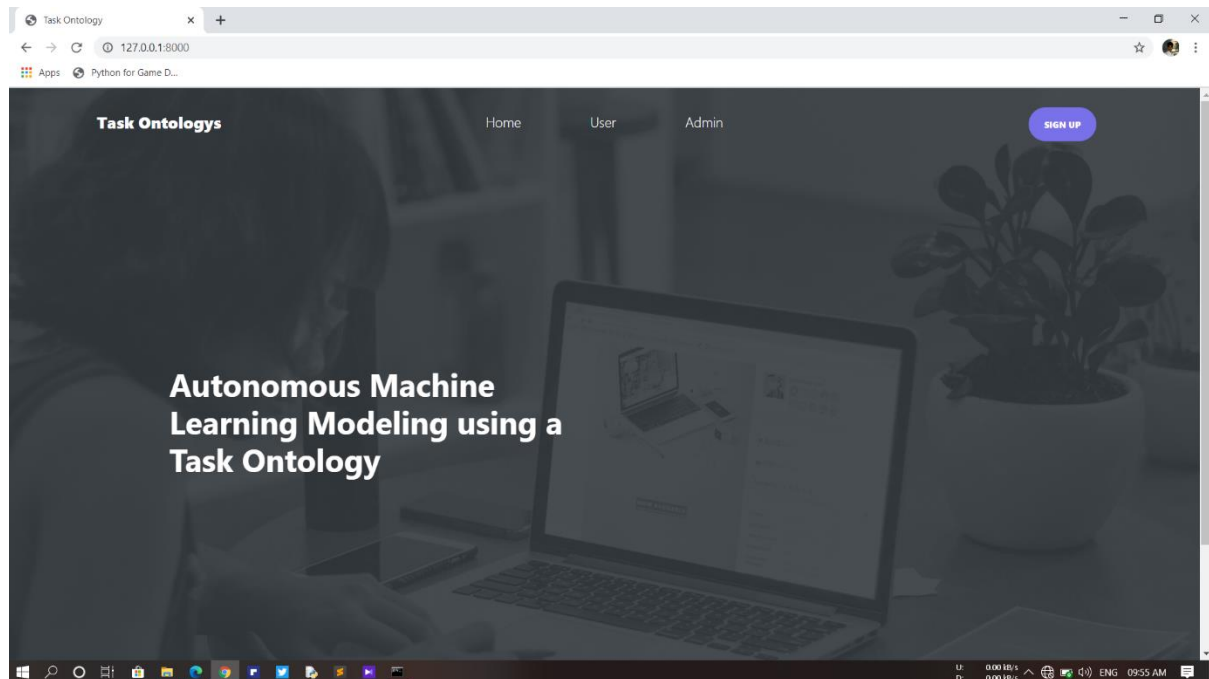
8.2 Test Cases

| S.no | Test Case | Excepted Result | Result | Remarks (IF Fails) |
|------|----------------------|--|--------|---|
| 1 | USER REGISTERED | If USER registration successfully. | Pass | If USER is not registered. |
| 2 | USER LOGIN | If USER name and password is correct then it will be getting valid page. | Pass | If USER name or password is not correct. |
| 4 | ADMIN | USER rights will be accepted here. | Pass | If USER is not registered. |
| 5 | USER upload files | Choose or select USER files | Pass | If USER is not select or SEND MESSAGES |
| 6 | Vocabulary detection | All Vocabulary verification | Pass | If Vocabulary detection is not available. |
| 7 | Image processing | Image processing verification | Pass | If Image processing is not available. |

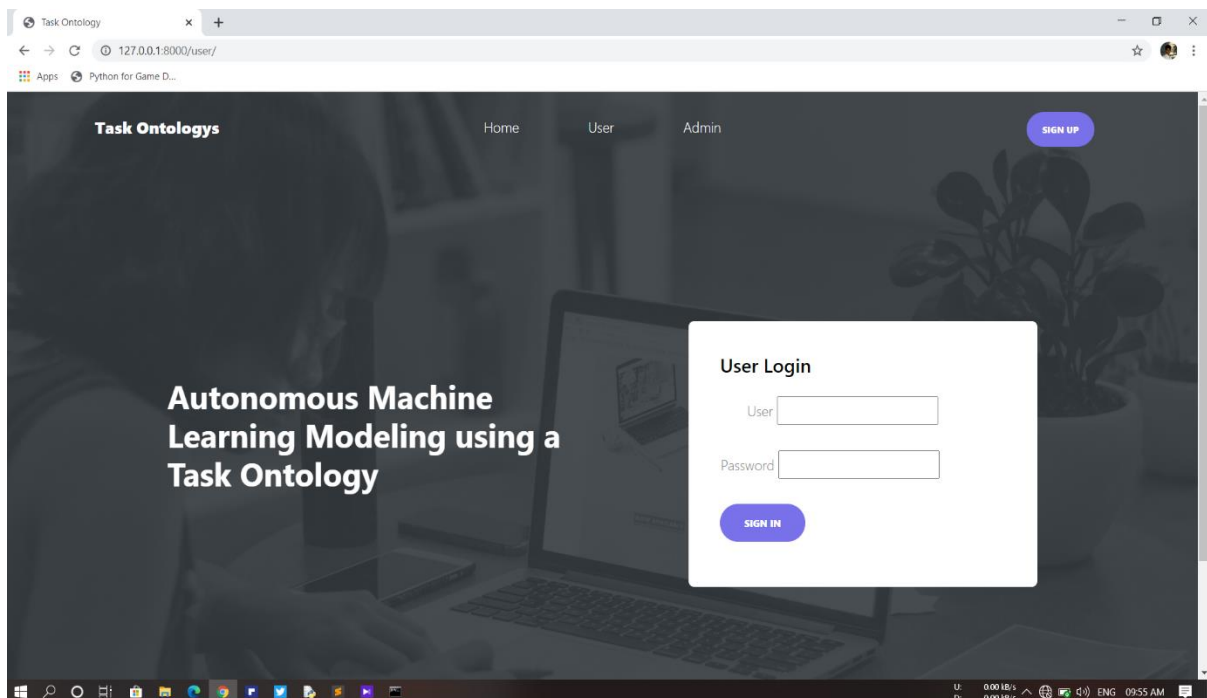
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9.OUTPUTS

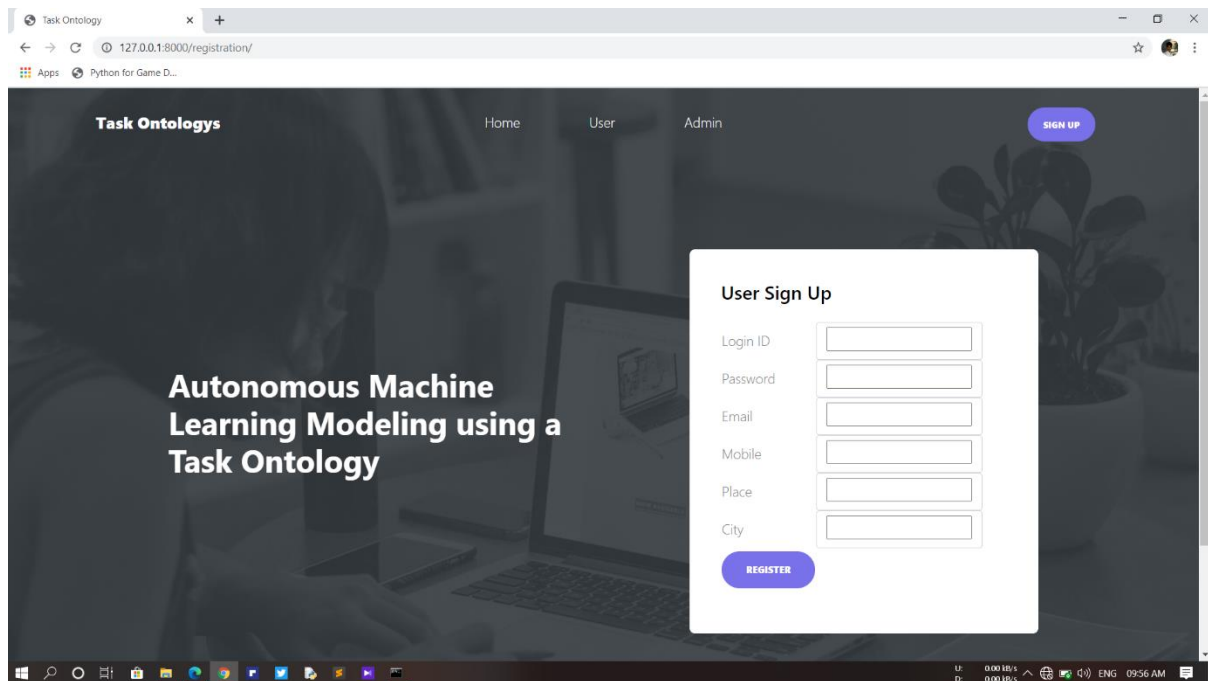
Home Page



User login



User register



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/registration/". The page title is "Task Ontology". The main header includes "Task Ontology", "Home", "User", "Admin", and a "SIGN UP" button. The background features a laptop and a potted plant. A large text overlay reads "Autonomous Machine Learning Modeling using a Task Ontology". A "User Sign Up" modal is open, containing the following fields: Login ID, Password, Email, Mobile, Place, and City. A "REGISTER" button is at the bottom of the modal. The Windows taskbar at the bottom shows the time as 09:56 AM.

Task Ontology Home User Admin [SIGN UP](#)

Autonomous Machine Learning Modeling using a Task Ontology

User Sign Up

Login ID

Password

Email

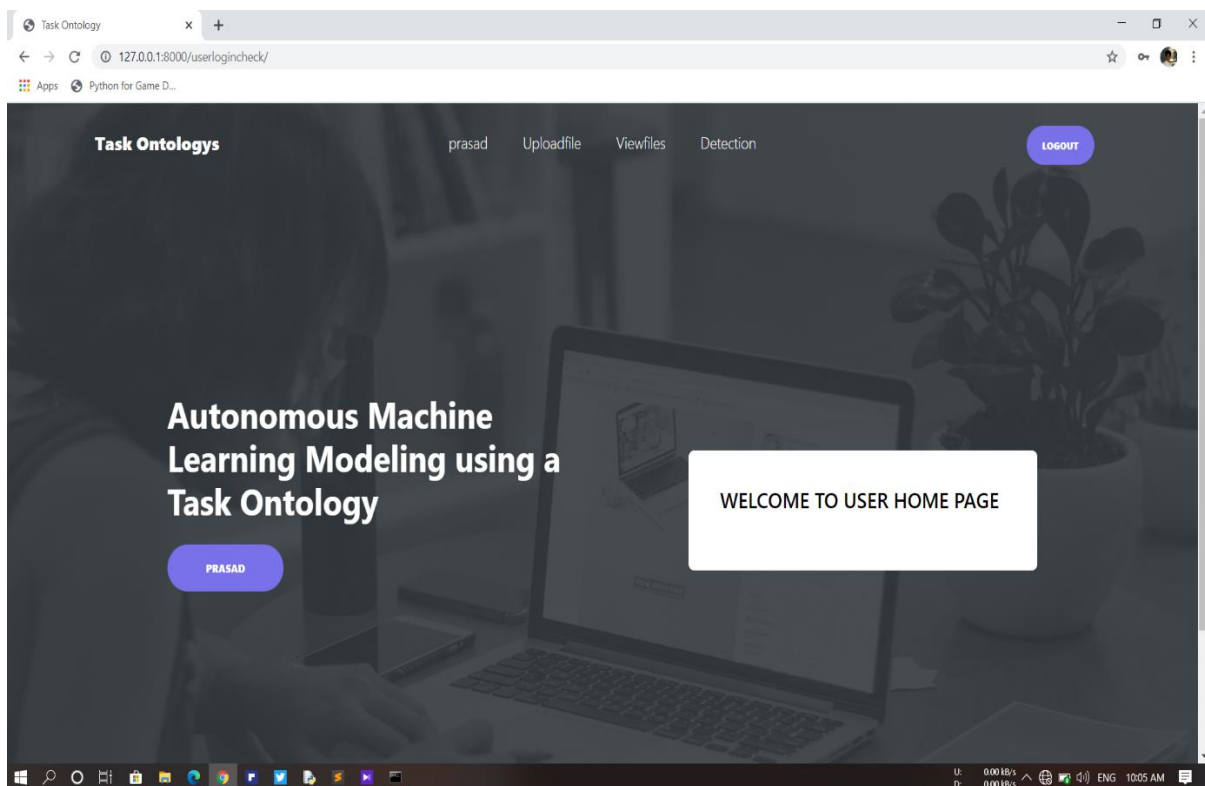
Mobile

Place

City

[REGISTER](#)

User home page



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/userlogincheck/". The page title is "Task Ontology". The main header includes "Task Ontology", "prasad", "Uploadfile", "Viewfiles", "Detection", and a "LOGOUT" button. The background features a laptop and a potted plant. A large text overlay reads "Autonomous Machine Learning Modeling using a Task Ontology". A "WELCOME TO USER HOME PAGE" modal is open. A "PRASAD" button is visible in the bottom left. The Windows taskbar at the bottom shows the time as 10:05 AM.

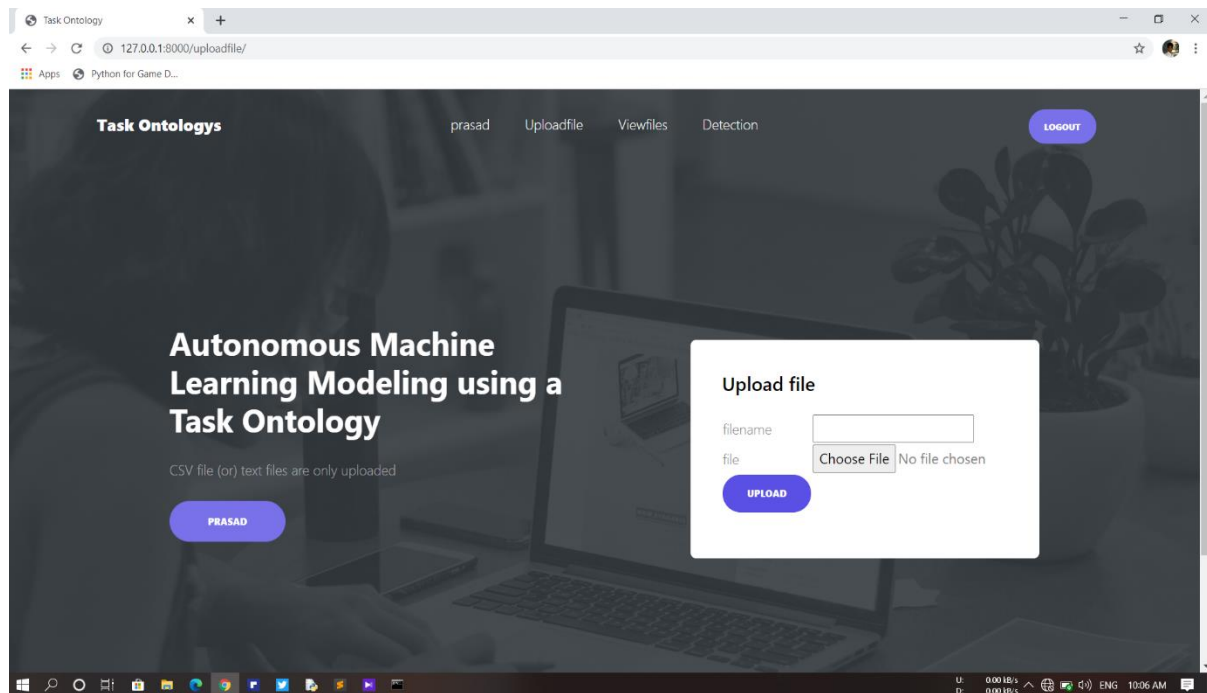
Task Ontology prasad Uploadfile Viewfiles Detection [LOGOUT](#)

Autonomous Machine Learning Modeling using a Task Ontology

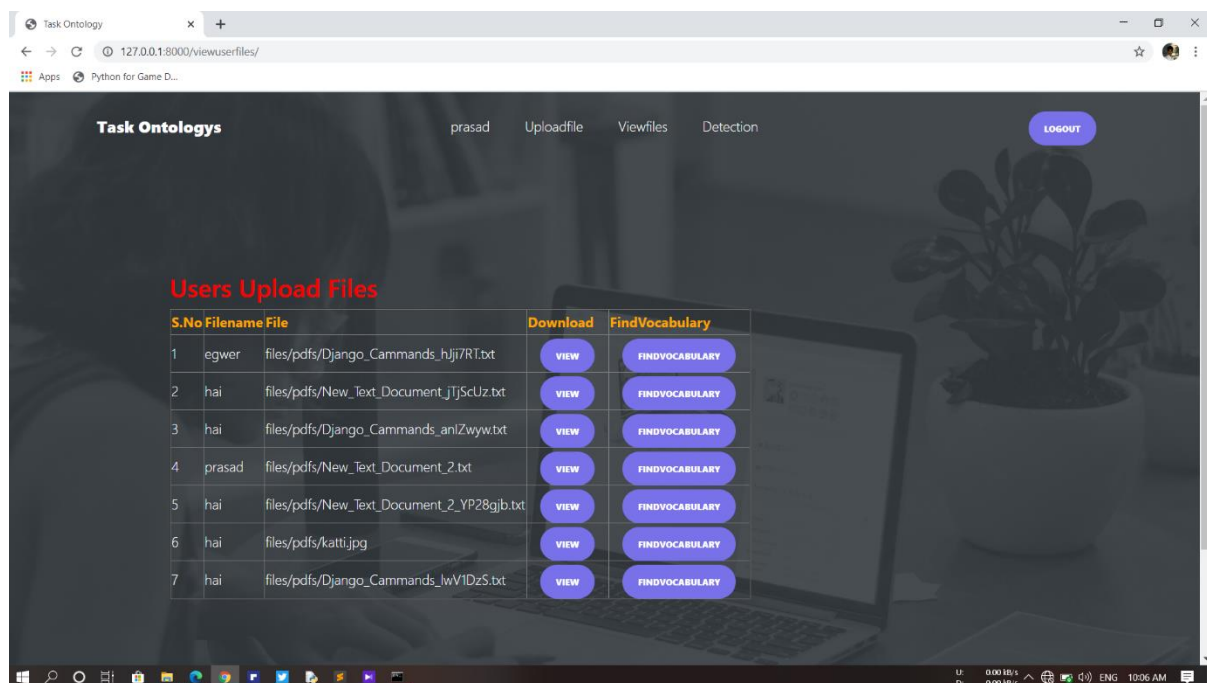
WELCOME TO USER HOME PAGE

[PRASAD](#)

User upload file



User view file



Detection of images

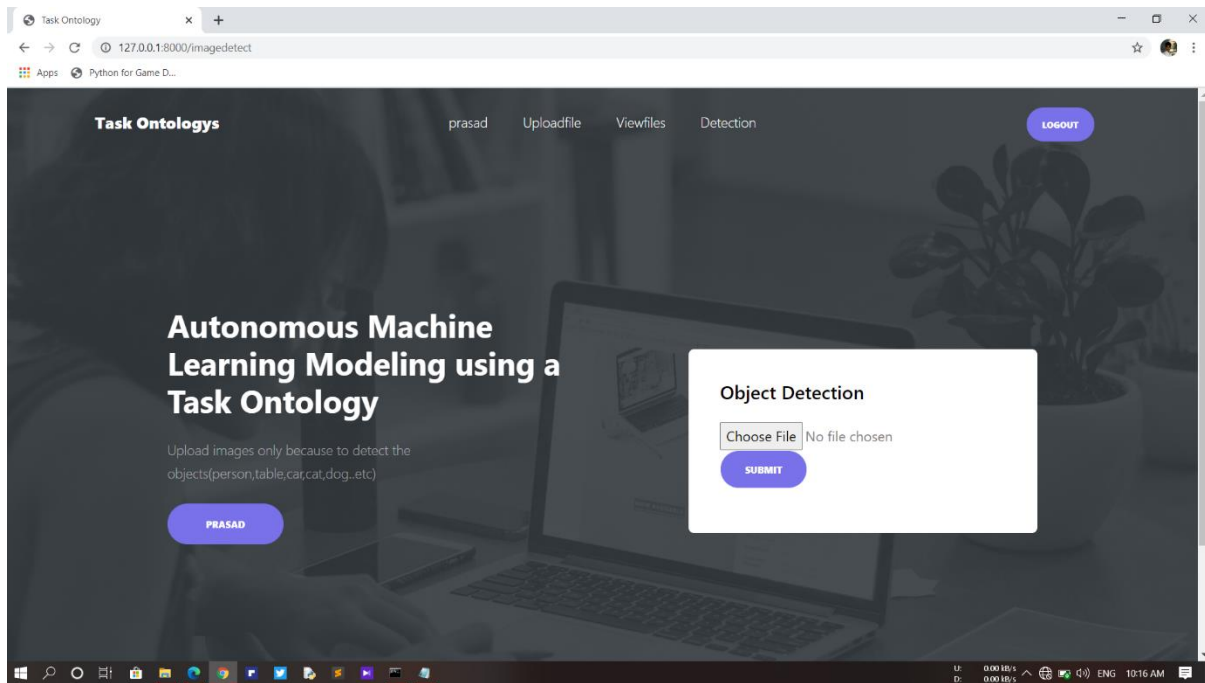
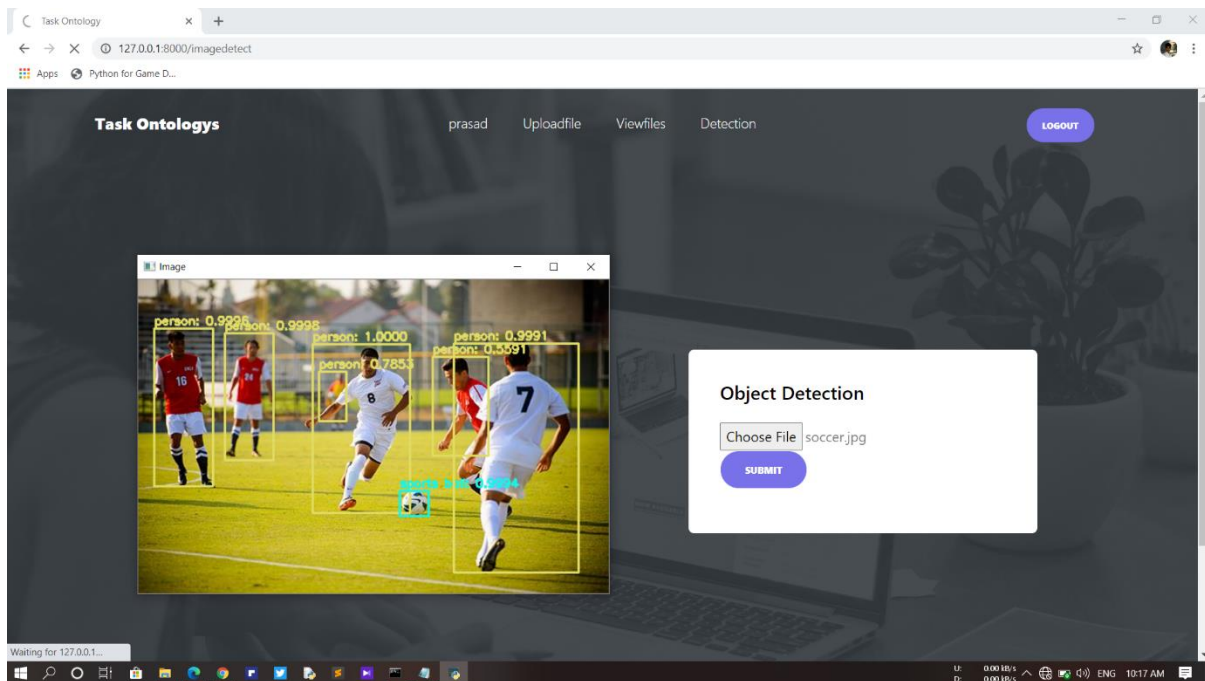
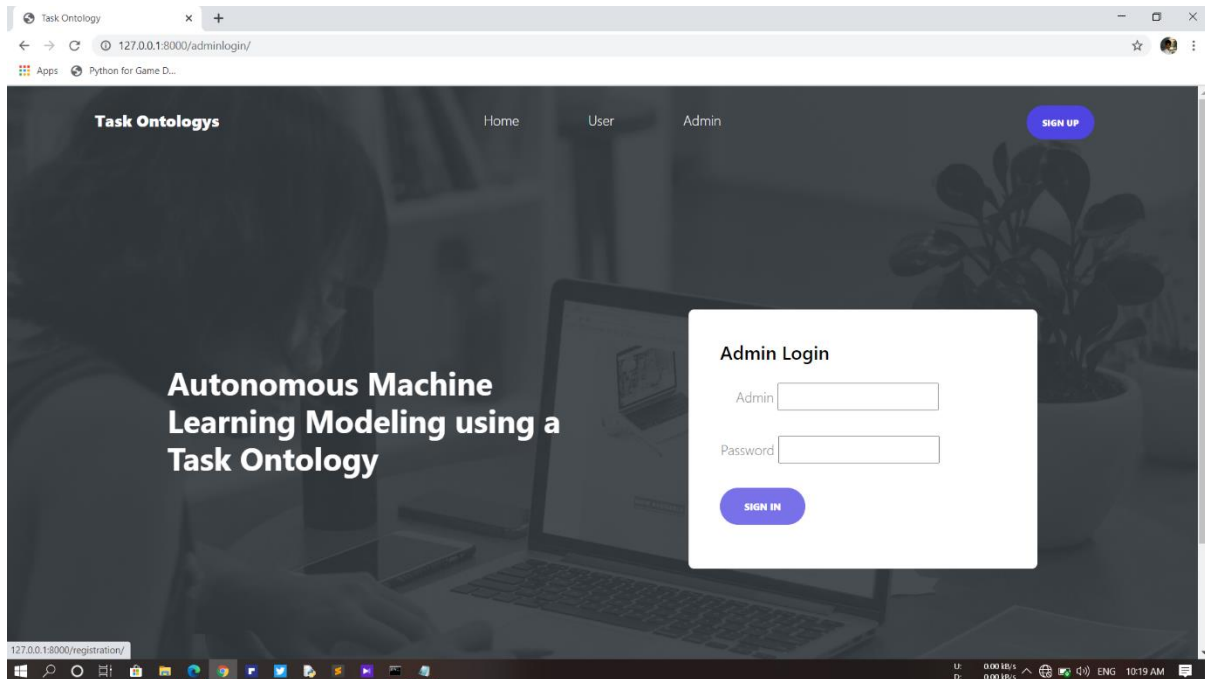


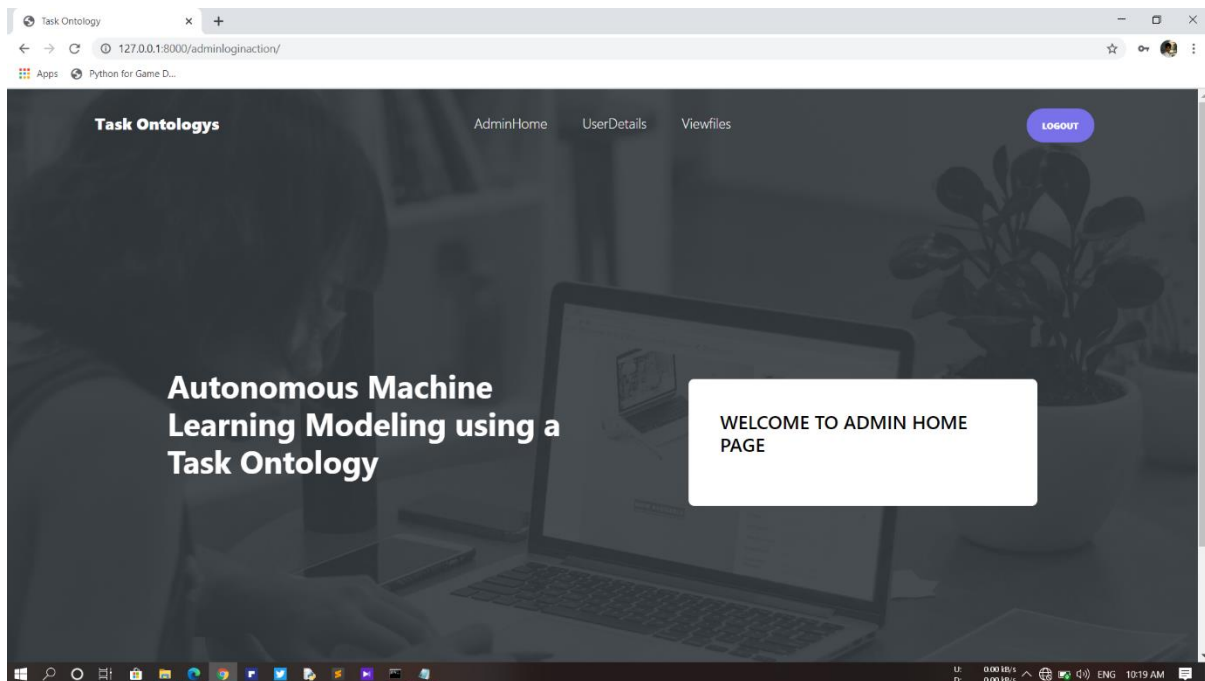
Image detection



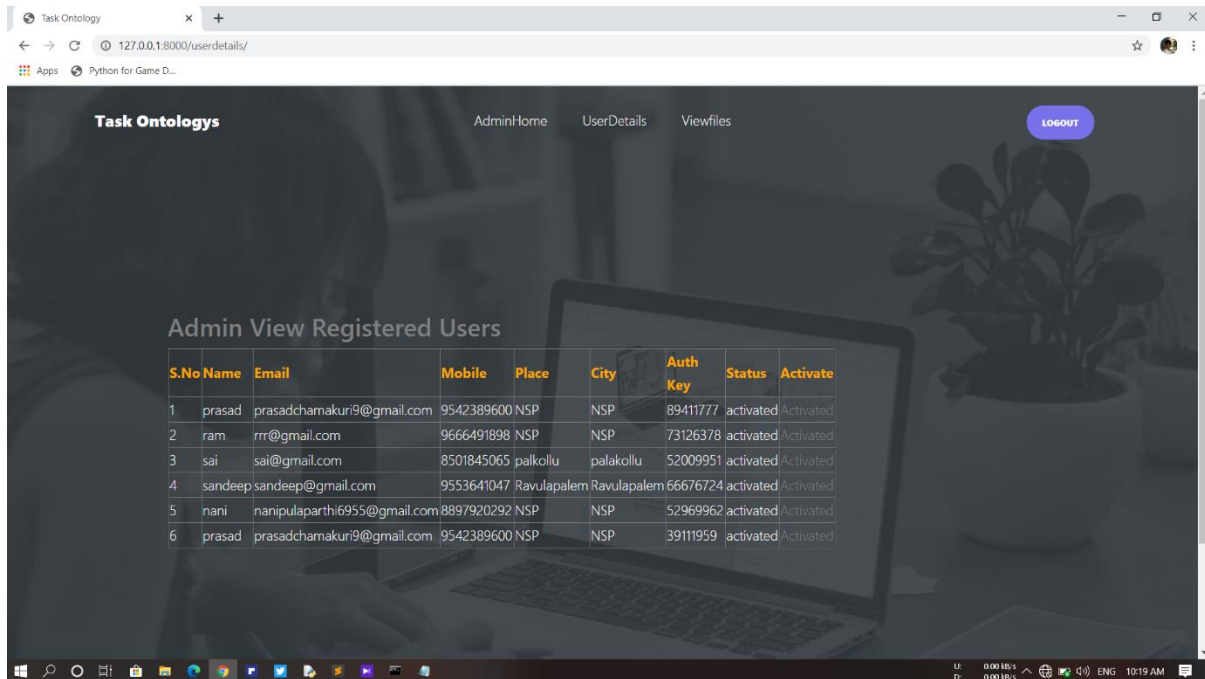
Admin login



Admin home page



User details



Task Ontology

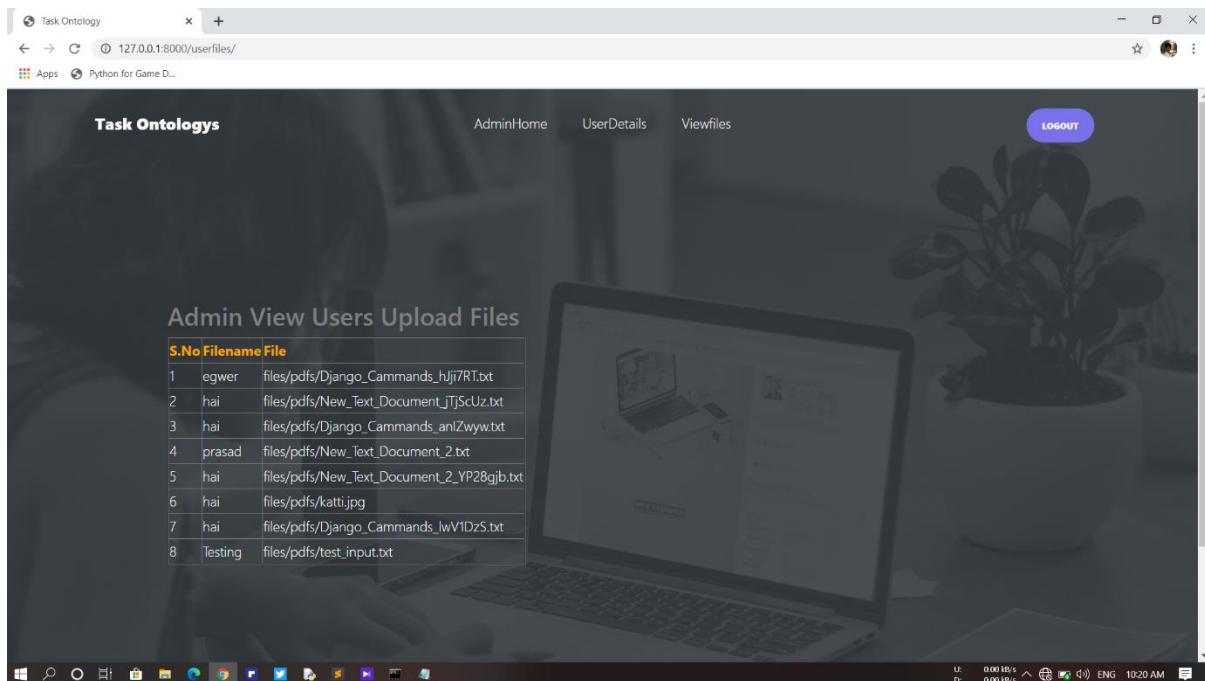
AdminHome UserDetails Viewfiles

LOGOUT

Admin View Registered Users

| S.No | Name | Email | Mobile | Place | City | Auth Key | Status | Activate |
|------|---------|-----------------------------|------------|-------------|-------------|----------|-----------|-----------|
| 1 | prasad | prasadchamkur9@gmail.com | 9542389600 | NSP | NSP | 89411777 | activated | Activated |
| 2 | ram | rr@gmail.com | 9666491898 | NSP | NSP | 73126378 | activated | Activated |
| 3 | sai | sai@gmail.com | 8501845065 | palkollu | palakollu | 52009951 | activated | Activated |
| 4 | sandeep | sandeep@gmail.com | 9553641047 | Ravulapalem | Ravulapalem | 66676724 | activated | Activated |
| 5 | nani | nanipulparthi6955@gmail.com | 8897920292 | NSP | NSP | 52969962 | activated | Activated |
| 6 | prasad | prasadchamkur9@gmail.com | 9542389600 | NSP | NSP | 39111959 | activated | Activated |

View files



Task Ontology

AdminHome UserDetails Viewfiles

LOGOUT

Admin View Users Upload Files

| S.No | Filename | File |
|------|----------|--|
| 1 | egwer | files/pdfs/Django_Cammands_hjji7RT.txt |
| 2 | hai | files/pdfs/New_Text_Document_LjTjScUz.txt |
| 3 | hai | files/pdfs/Django_Cammands_anIZwyw.txt |
| 4 | prasad | files/pdfs/New_Text_Document_2.txt |
| 5 | hai | files/pdfs/New_Text_Document_2_YP28gjb.txt |
| 6 | hai | files/pdfs/kattl.jpg |
| 7 | hai | files/pdfs/Django_Cammands_lwV1DzS.txt |
| 8 | Testing | files/pdfs/test_input.txt |

10.CONCLUSION

In this paper, we extracted important keywords for constructing an ontology from papers and textbooks about machine learning. Moreover, we designed a task ontology based on the MEX vocabulary. We also studied workflow for the autonomous machine learning model. The proposed method is applicable for automatic workflow according to designated autonomous level. Therefore, the non-experts are capable of doing complex tasks using the proposed method and can easily implement the machine learning model in a specific application