A Project report on

PREDICTION OF TOP-N POPULAR VIDEOS VIA A CROSS DOMAIN HYBRID MODEL

Submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

Computer Science & Engineering

By

M.AMRUTHA	**164G1A0503**
N.KOUSHIKA	**164G1A0540**
P.MANISHA	**164G1A0553**
M.KAVYASREE	**164G1A0536**

Under the Guidance

of

Mr. K. Varun Kumar Reddy, M.Tech.,

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(B. Tech Program Accredited by NBA)

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY:ANANTAPUR

(Approved by AICTE, New Delhi, Affiliated to JNTUA, Accredited by NAAC with 'A'Grade)

2019-2020

SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY:ANANTAPUR

(Approved by AICTE, New Delhi, Affiliated to JNTUA, Accredited by NAAC with 'A'Grade)

Rotarypuram Village, B K SamudramMandal, Ananthapuramu – 515701

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(B.Tech Program Accredited by NBA)

Certificate

This is to certify that the Project Report entitled **TOP-N POPULAR VIDEOS VIA A CROSS DOMAIN HYBRID MODEL** is the bonafide work carried out by **M.AMRUTHA**bearing Roll Number **164G1A0503, N.KOUSHIKA**bearing Roll Number **164G1A0540, P.MANISHA**bearing Roll Number **164G1A0553, M.KAVYASREE** bearing Roll Number **164G1A0536**in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2018- 2019.

Signature oftheGuide	**Head of theDepartment**
Mr. K.Varun Kumar Reddy, M.Tech.,	Dr. B. Lakshmi Narayana Reddy,Ph.D.,
AssistantProfessor.	Professor

<<Dept. Round Seal>>

Date: **EXTERNALEXAMINER**

Place: Rotarypuram

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned my efforts withsuccess.

I wish to convey my special thanks **Dr. T. HitendraSarma,** **Ph.D**, **Principal of Srinivasa Ramanujan Institute of Technology** for giving the required information in doing my projectwork.

I am very much thankful to **Dr. B. Lakshmi Narayana Reddy,** **Ph.D.**, **Professor and Head of Department, Computer Science & Engineering,** for his kind support and for providing necessary facilities to carry out thework.

I express my deep felt gratitude to **Mr. P. Veera Prakash,** **M.Tech.**,project coordinator and my Guide **Mr. K. Varun Kumar Reddy,** **M.Tech.**,whose valuable guidance and unstinting encouragement enabled me to accomplish my project successfully in time.

I extend my sincere thanks to all other teaching and non-teaching staff, and my friends who had directly or indirectly helped and supported me in completing my project in time. I also express my sincere thanks to the Management for providing excellent facilities.

Finally, I wish to convey my gratitude to my family who fostered all the requirements and facilities that I need.

.

DECLARATION

We, Ms. **N.Koushika** with reg no:**164G1A0540 , Ms. M.Amrutha** with reg no: **164G1A0503**, Ms. **P.Manisha** with reg no: **164G1A0553** AND Ms. **M.Kavyasree** with reg no: **164G1A0540** student of **SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY, Rotarypuram,** I hereby declare that the dissertation entitled "**Prediction of Top N Popular Videos via a Cross Domain Hybrid Model**" embodies the report of my project work carried out by me during IV year **Bachelor of Technology** in **COMPUTER SCIENCE & ENGINEERING**, under the supervision of Mr. **K. Varun Kumar Reddy, M.Tech., Assistant Professor,** Department of CSE, **SRINIVASA RAMANUJAN INSTITUTE OF TECHNOLOGY** and this work has been submitted for the partial fulfillment of the requirements for the award of **Bachelor of Technology** degree.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

OVS	sOnline Video Services
DA-RNN	Dual –stage Attention Recurrent Neural Networks
RMSEs	Root Mean Square Errors
MLP	Multi-layer Perceptron
PERL	Practical Extraction and Reporting Language
PHP	Hypertext Pre-processor
GPL	General Public License
OOP	Object-Oriented Programming
XML	eXtensible Mark-up Language
HTML	Hyper Text Mark-up Language
SGML	Standard Generalized Mark-up Language
CSRF	Cross-Site Request Forgery
CGI	Common Gateway Interface
MFDI	Multi Factor Differential Influence

ABSTRACT

Predicting the top-N popular videos and their future views for a large batch of newly uploaded videos is of great commercial value to online video services (OVSs). Although many attempts have been made on video popularity prediction, the existing models has a much lower performance in predicting the top-N popular videos than that of the entire video set. The reason for this phenomenon is that most videos in an OVS system are unpopular, so models preferentially learn the popularity trends of unpopular videos to improve their performance on the entire video set. However, in most cases, it is critical to predict the performance on the top-N popular videos which is the focus of this study. The challenge for the task are as follows. First, popular and unpopular videos may have similar early view patterns. Second, prediction models that are overly dependent on early view patterns limit the effects of other features. To address these challenges, we propose a novel multifactor differential influence (MFDI) prediction model based on multivariate linear regression (MLR). The model is designed to improve the discovery of popular videos and their popularity trends are learnt by enhancing the discriminative power of early patterns for different popularity trends and by optimizing the utilization of multi-source data. We evaluate the proposed model using real-world YouTube data, and extensive experiments have demonstrated the effectiveness of our model.

CHAPTER: 1
INTRODUCTION

Since the popularity of online videos has been proven to be predictable through the statistical analysis of large-scale YouTube data, numerous related studies have been conducted. Szabo and Huberman (S-H) proposed a content-scaling (CS) model based on log-transformed relations between a video's long-term popularity and its early popularity. Their conclusion is one of the most important foundations of popularity prediction research and has been succeed by many related works.

All of the approaches cited above achieved initial success, but their shortcomings has been uncovered by subsequent research. Popularity prediction of online videos, especially the prediction of the top-N popular videos is of great importance to support the development of online video services (OVSs). From the perspective of better user experience, the ability to identify the top-N popular videos is beneficial to video services, such as caching and recommendation.

From the perspective of commercialization, identifying the top-N popular videos helps the video service providers to maximize their profits, as advertisers are more likely to pay more for popular videos. Although many attempts have been made on popularity prediction of online videos, because most of the videos in an OVS system are unpopular; consequently, models preferentially learn the popularity trends of these unpopular videos to achieve better performance on the video set as a whole.

Prediction of the top-N popular videos remains a challenging problem for the following reasons. First, popular and unpopular videos may have similar early view patterns, and this similarity limits the performance benefit of video classification based on early view patterns. We evaluate the proposed model using real-world YouTube data, and extensive experiments have demonstrated the effectiveness of our model.

1.1. Problem Definition

Prediction of the top-N popular videos remains a challenging problem for the following reasons. First, popular and unpopular videos may have similar early view patterns, and this similarity limits the performance benefit of video classification based on early view patterns. Second existing studies show that the strong correlation between early views and long-term popularity dominates the training of the prediction models. This overdependence on early view patterns prevents models from finding popular videos based on multisource data.

1.2. Objectives

The main objectives of the project work is as follows:

- Provides high performance in predicting top-n popular videos.
- Provides great commercial value to Online Video Services to make more profits.

CHAPTER: 2
LITERATURE SURVEY

2.1. Existing System:

A Novel Time Series Approach:

Since the rate at which new content is uploaded to the Internet has reached unprecedented marks, knowing the popularity of online content, especially video, is of importance for network management, recommendation schemes, service design, advertising planning, and so on. Despite the fact that various models have been developed, few of them address the short-term popularity prediction.

Toward this goal, we exploit the self-attention mechanism of the Transformer, a state-of-the-art model in neural machine translation, to forecast the values of multiple time series in the near future. Specifically, we propose an attention-based non-recursive neural network, a novel model that entirely dispenses with recurrence and convolutions, for time series prediction.

Since our model is the combination of the input attention mechanism in the dual-stage attention-based recurrent neural network (DA-RNN) and the self-attention of Transformer, it is able to adaptively select the most relevant input sequences as well as capture the long-term dependencies across previous time steps to make the prediction.

The experiments show the root mean square errors (RMSEs) achieved by our model are only 6.06 and 3.60 when testing on the NASDAQ 100 dataset and the views count of the top most popular videos on Youtube respectively, while the RMSEs of DA-RNN are 8.52 and 12.31. Hence, our model outperforms the baseline not only in time series prediction but also in contents popularity prediction aspect.

Predicting popularity of online videos using support vector regression:

In this work, we propose a regression method to predict the popularity of an online video measured by its number of views. Our method uses Support Vector Regression with Gaussian radial basis functions. We show that predicting popularity patterns with this approach provides more precise and more stable prediction results, mainly thanks to the nonlinear character of the proposed method as well as its robustness. We prove the superiority of our method against the state of the art using datasets containing almost 24 000 videos from YouTube and Facebook.

We also show that using visual features, such as the outputs of deep neural networks or scene dynamics' metrics, can be useful for popularity prediction before content publication. Furthermore, we show that popularity prediction accuracy can be improved by combining early distribution patterns with social and visual features and that social features represent a much stronger signal in terms of video popularity prediction than the visual ones.

Popularity prediction of online videos, especially the prediction of the top-N popular videos is of great importance to support the development of online video services (OVSs). From the perspective of better user experience, the ability to identify the top-N popular videos is beneficial to video services, such as caching and recommendation. From the perspective of commercialization, identifying the top-N popular videos helps the video service providers to maximize their profits, as advertisers are more likely to pay more for popular videos. Although many attempts have been made on popularity prediction of online videos2, because most of the videos in an OVS system are unpopular; consequently, models preferentially learn the popularity trends of these unpopular videos to achieve better performance on the video set as a whole.

Prediction of the top-N popular videos remains a challenging problem for the following reasons. First, popular and unpopular videos may have similar early view patterns, and this similarity limits the performance benefit of video classification based on early view patterns. Second, existing studies show that the strong correlation between early views and long-term popularity dominates the training of the prediction models. This overdependence on early view patterns prevents models from finding popular videos based on multisource data.

2.2. Proposed System:

We evaluate the proposed model using real-world data consisting of videos from YouTube and social network data from Twitter. Our experimental results show that the proposed model outperforms state-of-the-art models, thereby confirming the benefits of our efforts to improve the prediction performance for the top-N popular videos. The main contributions of this paper can be summarized as follows; we propose a model for predicting the top-N popular videos.

By enhancing the ability of early patterns to distinguish among popularity trends and optimizing the model's utilization of multi-source data, we develop a model that achieves the promised performance. By using the tags of videos as indicators of their content and jointly training a multi-layer perceptron (MLP) network on the popularity data of videos and their related social content, we estimate the contribution of the popularity of a video's content on a social network to the long-term popularity of the video.

Predicting the top-N popular videos and their future views for a large batch of newly uploaded videos is of great commercial value to online video services (OVSs). Although many attempts have been made on video popularity prediction, the existing models has a much lower performance in predicting the top-N popular videos than that of the entire video set.

The reason for this phenomenon is that most videos in an OVS system are unpopular, so models preferentially learn the popularity trends of unpopular videos to improve their performance on the entire video set. However, in most cases, it is critical to predict the performance on the top-N popular videos which is the focus of this study. The challenge for the task are as follows. First, popular and unpopular videos may have similar early view patterns. Second, prediction models that are overly dependent on early view patterns limit the effects of other features.

CHAPTER: 3

ANALYSIS

3.1. The Study of the System:

To conduct studies and analysis of an operation and technical nature. To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

3.1.1. Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER diagrams i.e. Entity Relation Diagrams.

3.1.2. Physical Design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified or authenticated, how it is processed, and how it is displayed as output. In physical design, following requirements about the system are decided.

1. Input requirements
2. Output requirements
3. Storage requirements
4. Processing requirements

The physical portion of systems designs can generally be broken down into three sub- tasks:

1. User interface design
2. Data flow

User Interface design is concerned with how users add information to the system and with how the system presents information back to them. Data design is concerned with how the data is represented and stored within the system. Finally, Process design is concerned with how data moves through the system, and how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the system design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, videos/graphic cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

3.2. Software Development Tool

3.2.1. Introduction to Python Framework

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.

3.2.2. Python

What Is A Script?

Up to this point, I have concentrated on the interactive programming capability of Python. This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

Scripts are r̄eusable

Basically, a script is a text file containing the statements that comprise a Python program. Once you have created the script, you can execute it over and over without having to retype it each time.

Scripts are editable

Perhaps, more importantly, you can make different versions of the script by modifying the statements from one file to the next using a text editor. Then you can execute each of the individual versions. In this way, it is easy to create different programs with a minimum amount of typing.

You will need a text editor

Just about any text editor will suffice for creating Python script files.You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word, or just about any word processor if you want to.

Difference between a script and a program

Script:

Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, whereas the applications they control are traditionally compiled to native machine code.

Program:

The program has an executable form that the computer can use directly to execute the instructions. The same program in its human-readable source code form, from which executable programs are derived (e.g., compiled)

Python:

What is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

Python concepts

If your not interested in the the how’s and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy to interface with C++ (via SWIG)

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

- Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and UNIX shell and other scripting languages.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features:

Python's features include –

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Dynamic vs. Static

- Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.
- For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.
- This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.
- If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

- Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.
- With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).
- If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.
- With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.
- For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

Variables

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric variables.

Python has various standard data types that are used to define the operations.
Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

Python Tuples:

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read- only lists.

Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object. Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Different modes in python

Python has two basic modes: normal and interactive. The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole.

20 Python libraries

1. Requests. The most famous http library written by Kenneth reitz. It’s a must have for every python developer.
2. Scrapy. If you are involved in web scraping then this is a must have library for you. After using this library you won’t use any other.
3. Wxpython. A GUI toolkit for python. I have primarily used it in place of tkinter. You will really love it.
4. Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.
5. SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.
6. Beautiful Soup. I know it’s slow but this xml and html parsing library is very useful for beginners.
7. Twisted. The most important tool for any network application developer. It has a very beautiful API and is used by a lot of famous python developers.
8. NumPy. How can we leave this very important library? It provides some advance math functionalities to python.
9. SciPy. When we talk about NumPy then we have to talk about SciPy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.
10. Matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
11. Pygame. Which developer does not like to play games and develop them? This library will help you achieve your goal of 2d game development.
12. Pyglet. A 3d animation and game creation engine. This is the engine in which the famous python port of minecraft was made
13. PyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI’s for my python scripts.
14. PyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.
15. Scapy. A packet sniffer and analyzer for python made in python.

16. Pywin32. A python library which provides some useful methods and classes for interacting with windows.

17. nltk. Natural Language Toolkit – I realize most people won't be using this one, but it's generic enough. It is a very useful library if you want to manipulate strings. But its capacity is beyond that. Do check it out.

18. Nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

19. SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

20. I Python. I just can't stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

NumPy

NumPy main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.

In NumPy dimensions are called axes. The number of axes is rank.

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

- Offers Mat lab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra etc.

Matplotlib

High quality plotting library.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something. Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me. Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code Objects are an encapsulation of variables and functions into a single entity.

Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here's a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new Namespace.
• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax**: ClassName.Attribute**.
• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.
• Calling a class (just like a function) creates a new instance of the class.This is where the multiple copies part comes in.
• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.
• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

Inheritance

First off, classes allow you to modify a program without really making changes to it. To elaborate, by sub classing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we've seen, an instance of a class inherits the attributes of that class. However, classes can also inherit attributes from other classes. Hence, a subclass allowing you to make a generic superclass that is specialized via subclasses.
The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads:

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more Consistent to Python’s built-in objects, hence overloading is easier to learn and use. User-made classes can override nearly all of Python’s built-in operation methods.

Exceptions

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors. They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python. Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a Key Error exception.

• Searching a list for a non−existent value will raise a Value Error exception

• Calling a non−existent method will raise an Attribute Error exception.

• Referencing a non−existent variable will raise a Name Error exception.

• Mixing data types without coercion will raise a Type Error exception.

One use of exceptions is to catch a fault and allow the program to continue working. We have seen this before when we talked about files. This is the most common way to use exceptions. When programming with the Python command line interpreter, you don't need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs. Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception. It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don't put any overhead on the code processing. Because exceptions aren't supposed to happen very often, they aren't processed until they occur.

Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions. However, as already mentioned, exceptions aren't processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program. The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions:

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions. You probably won’t have to do this very often but it’s nice to have the option when necessary .However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you. They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free. Making your own exceptions involves object-oriented programming, which will be covered in the next chapter.

To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class. To make a custom exception, simply inherit the base exception and define what it will do.

Python modules

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

Testing code

- As indicated above, code is usually developed in a file using an editor.
- To test the code, import it into a Python session and try to run it.

- Usually there is an error, so you go back to the file, make a correction, and test again.
- This process is repeated until you are satisfied that the code works.
- The entire process is known as the development cycle.
- There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.
- This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

Functions in Python

It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

To carry out that specific task, the function might or might not need multiple inputs. When the task is carried out, the function can or cannot return one or more values. There are three types of functions in python: help(), min(), print().

Python Namespace

Generally speaking, a namespace (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a name spacing system from daily life, i.e. the naming of people in first name and family name (surname).

An example is a network: each network device (workstation, server, printer,) needs a unique name and address. Yet another example is the directory structure of file systems. Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs. Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

- global names of a module
- local names in a function or method invocation
- built-in names: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

Garbage Collection

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

Python XML Parser

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML. This is recommended by the World Wide Web Consortium and available as an open standard. XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML. The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX: Here, you register callbacks for events of interest and then let the parser proceed through the document.This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

Python Web Frameworks

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

Why are web frameworks useful?

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality:

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

Comparing web frameworks:

There is also a repository called compare-python-web-frameworks where the same web application is being coded with varying Python web frameworks, templating engines and object.

Web framework resources:

- When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
- Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
- What is a web framework? Is an in-depth explanation of what web frameworks are and their relation to web servers?
- Django vs. Flash vs. Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
- This fascinating blog post takes a look at the code complexity of several Python web frameworks by providing visualizations based on their code bases.
- Python's web frameworks benchmarks is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
- What web frameworks do you use and why are they awesome? is a language agnostic Reedit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
- This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

Web frameworks learning checklist:

1. Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.

2. Work through a detailed tutorial found within the resources links on the framework's page.

3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.

4. Build the first simple iteration of your web application then go to the deployment section to make it accessible on the web.

<u>Python-Data Base Communication</u>

Connector/Python provides a connect() call used to establish connections to the MySQL server. The following sections describe the permitted arguments for connect() and describe how to use option files that supply additional arguments.

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring this information.

The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users database itself.

Databases are popular for many applications, especially for use with web applications or customer-oriented programs. Users don't have to be human users. They can be other programs and applications as well. We will learn how Python or better a Python program can interact as a user of SQLdatabase. This is an introduction into using SQLite and MySQL from Python.

The Python standard for database interfaces is the Python DB-API, which is used by Python's database interfaces. The DB-API has been defined as a common interface, which can be used to access relational databases.

In other words, the code in Python for communicating with a database should be the same, regardless of the database and the database module used. Even though we use lots of SQL examples, this is not an introduction into SQL but a tutorial on the Python interface.

SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM. It was developed for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.

SQLITE is "quite" fast, even though it uses a simple file. It can be used for large databases as well. If you want to use SQLite, you have to import the module sqlite3. To use a database, you have to create first a Connection object.

The connection object will represent the database. The argument of connection - in the following example "companys.db" - functions both as the name of the file, where the data will be stored, and as the name of the database. If a file with this name exists, it will be opened. It has to be a SQLite database file of course! In the following example, we will open a database called company.

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library. For notes detailing the changes in each release of Connector/Python, see MySQL Connector/Python Release Notes.

MySQL Connector/Python includes support for:

- Almost all features provided by MySQL Server up to and including MySQL Server version 5.7.
- Converting parameter values back and forth between Python and MySQL data types, for example Python date time and MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance.

- All MySQL extensions to standard SQL syntax.
- Protocol compression, which enables compressing the data stream between the client and server.
- Connections using TCP/IP sockets and on UNIX using UNIX sockets.
- Secure TCP/IP connections using SSL.
- Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library.

3.2.3. Django Framework

Introduction to Django This book is about Django, a Web development framework that saves you time and makes Web development a joy. Using Django, you can build and maintain high-quality Web applications with minimal fuss. At its best, Web development is an exciting, creative act; at its worst, it can be a repetitive, frustrating nuisance. Django lets you focus on the fun stuff — the crux of your Web application — while easing the pain of the repetitive bits. In doing so, it provides high-level abstractions of common Web development patterns, shortcuts for frequent programming tasks, and clear conventions for how to solve problems.

Django tries to stay out of your way, letting you work outside the scope of the framework as needed. The goal of this book is to make you a Django expert. The focus is twofold. First, we explain, in depth, what Django does and how to build Web applications with it. Second, we discuss higher-level concepts where appropriate, answering the question “How can I apply these tools effectively in my own projects?” By reading this book, you’ll learn the skills needed to develop powerful Web sites quickly, with code that is clean and easy to maintain.

What Is a Web Framework?

Django is a prominent member of a new generation of Web frameworks. So what exactly does that term mean? To answer that question, let’s consider the design of a Web application written using the Common Gateway Interface (CGI) standard, a popular way to write Web applications circa 1998. In those days, when you wrote a

CGI application, you did everything yourself — the equivalent of baking a cake from scratch. For example, here’s a simple CGI script, written in Python, that displays the ten most recently published books from a database:

```
import MySQLdb

print "Content-Type: text/html"
print
print "<html><head><title>Books</title></head>"
print "<body>"
print "<h1>Books</h1>"
print "<ul>"

connection = MySQLdb.connect(user='me', passwd='letmein', db='my_db')
cursor = connection.cursor()
cursor.execute("SELECT name FROM books ORDER BY pub_date DESC LIMIT 10")
for row in cursor.fetchall():
    print "<li>%s</li>" % row[0]

print "</ul>"
print "</body></html>"

connection.close()
```

This code is straightforward. First, it prints a “Content-Type” line, followed by a blank line, as required by CGI. It prints some introductory HTML, connects to a database and executes a query that retrieves the latest ten books. Looping over those books, it generates an HTML unordered list. Finally, it prints the closing HTML and closes the database connection.

With a one-off dynamic page such as this one, the write-it-from-scratch approach isn’t necessarily bad. For one thing, this code is simple to comprehend — even a novice developer can read these 16 lines of Python and understand all it does, from start to finish. There’s nothing else to learn; no other code to read. It’s also simple to deploy: just save this code in a file called latestbooks.cgi, upload that file to a Web server, and visit that page with a browser. But as a Web application grows beyond the trivial, this approach breaks down, and you face a number of problems.

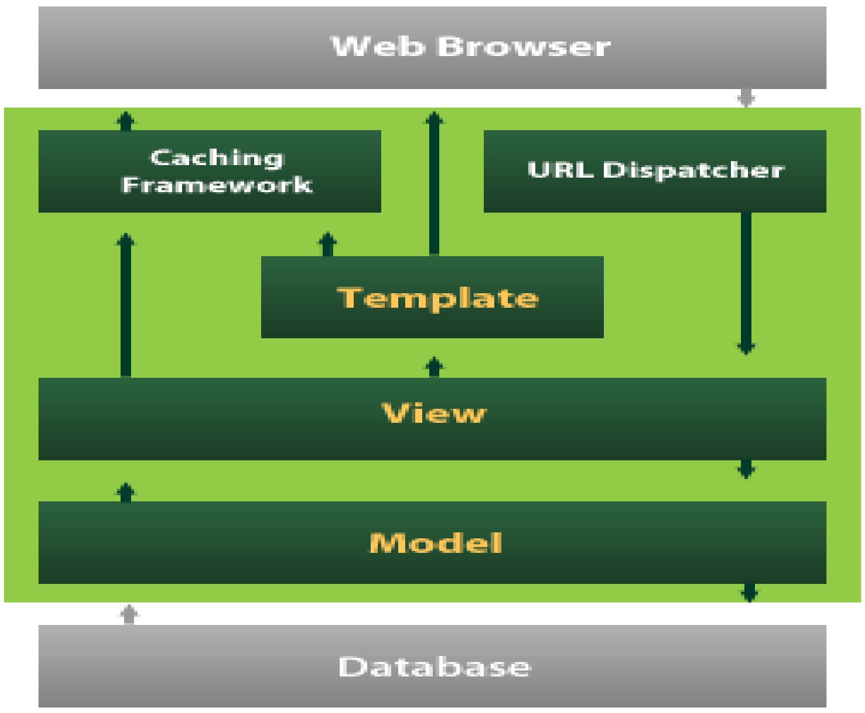


Figure 3.2.1 Python Framework

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 "plug ability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

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

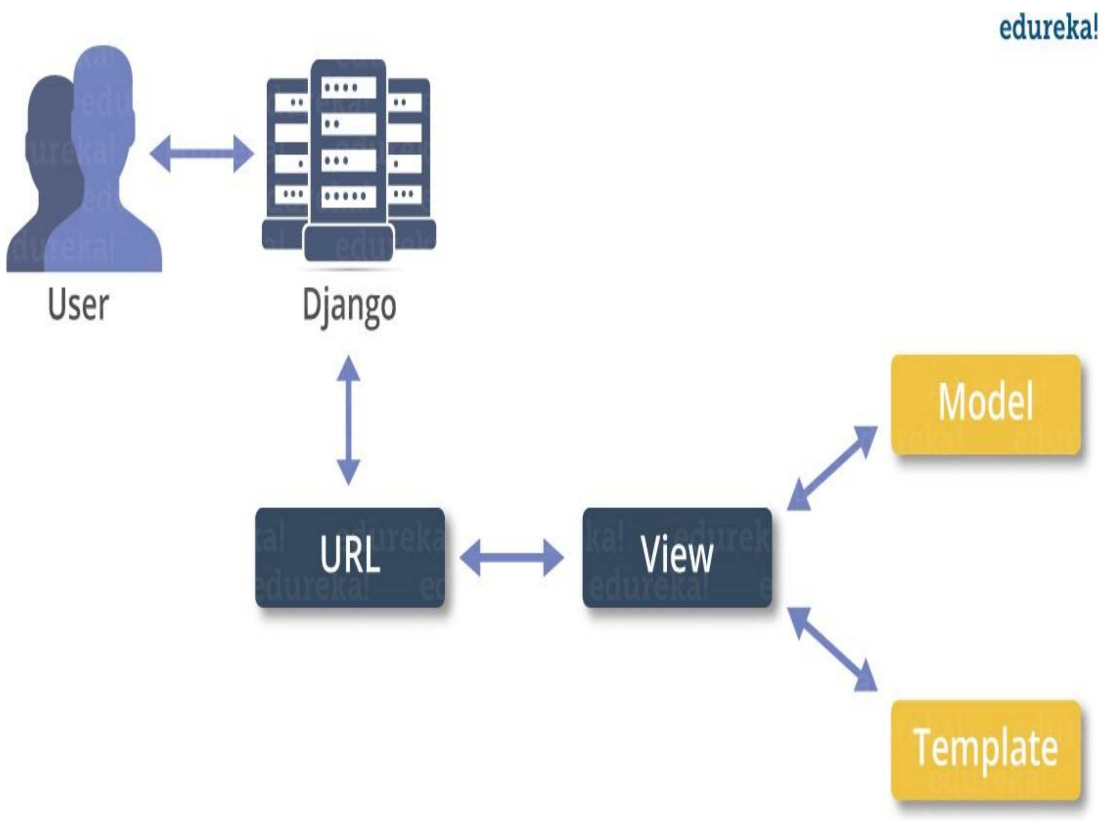


Figure 3.2.1 Admin Models of Django

Should a developer really have to worry about printing the "Content-Type" line and remembering to close the database connection? This sort of boilerplate reduces programmer productivity and introduces opportunities for mistakes. These setup- and teardown-related tasks would best be handled by some common infrastructure.

- ✓ What happens when this code is reused in multiple environments, each with a separate database and password? At this point, some environment-specific configuration becomes essential.
- ✓ What happens when a Web designer who has no experience coding Python wishes to redesign the page? Ideally, the logic of the page — the retrieval of books from the database — would be separate from the HTML display of the page, so that a designer could edit the latter without affecting the former.

- ✓ These problems are precisely what a Web framework intends to solve. A Web framework provides a programming infrastructure for your applications, so that you can focus on writing clean, maintainable code without having to reinvent the wheel. In a nutshell, that's what Django does.

These problems are precisely what a Web framework intends to solve. A Web framework provides a programming infrastructure for your applications, so that you can focus on writing clean, maintainable code without having to reinvent the wheel. In a nutshell, that's what Django does.

3.3. Requirement Specification

Functional Requirements

- Graphical User interface with the User.

Software Requirements

For developing the application the following are the Software Requirements:

1. Python
2. Django
3. MySQL
4. Wampserver

Operating Systems supported

1. Windows 7
2. Windows XP
3. Windows 8

Technologies and Languages used to Develop

1. Python

Debugger and Emulator

- Any Browser (Particularly Chrome)

Hardware Requirements

For developing the application the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

3.4. Installation of PyCharm

PyCharm is a cross-platform editor developed by Jet Brains. PyCharm provides all the tools you need for productive Python development. Below are the detailed steps for installing Python and PyCharm:

Step 1:

To download and install Python visit the official website of Python https://www.python.org/downloads/ and choose your version. We have chosen Python version 3.6.3.

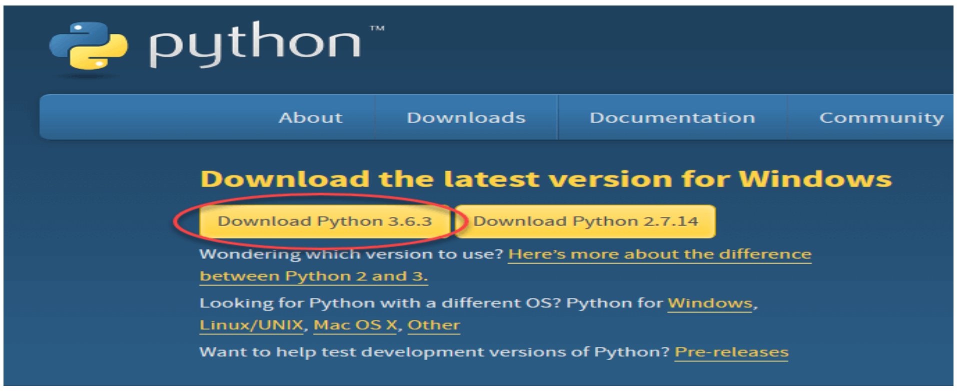
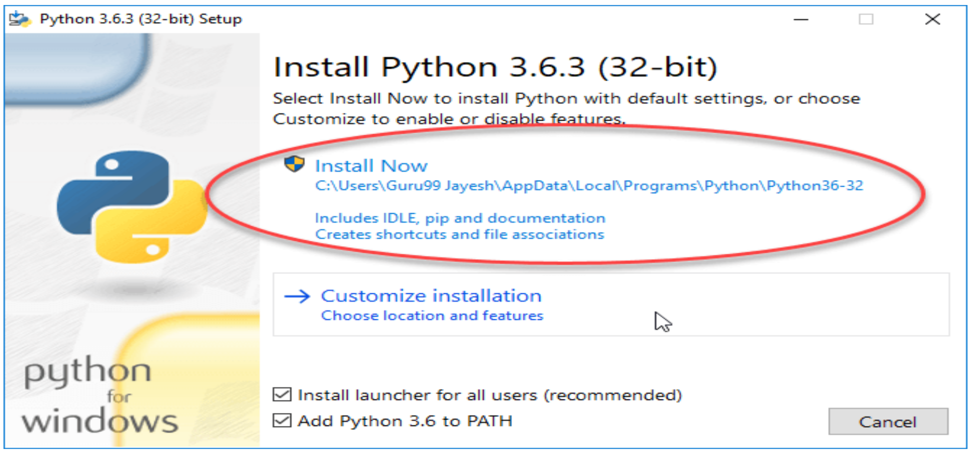

python™
About
Downloads
Documentation
Community
Download the latest version for Windows
Download Python 3.6.3
Download Python 2.7.14
Wondering which version to use? Here's more about the difference between Python 2 and 3.
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other
Want to help test development versions of Python? Pre-releases

Figure: 3.4.1 Downloading Python 3.6.3

Step 2:

Once the download is complete, run the exe for install Python. Now click on Install Now.


Python 3.6.3 (32-bit) Setup

Install Python 3.6.3 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Guru99 Jayesh\AppData\Local\Programs\Python\Python36-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

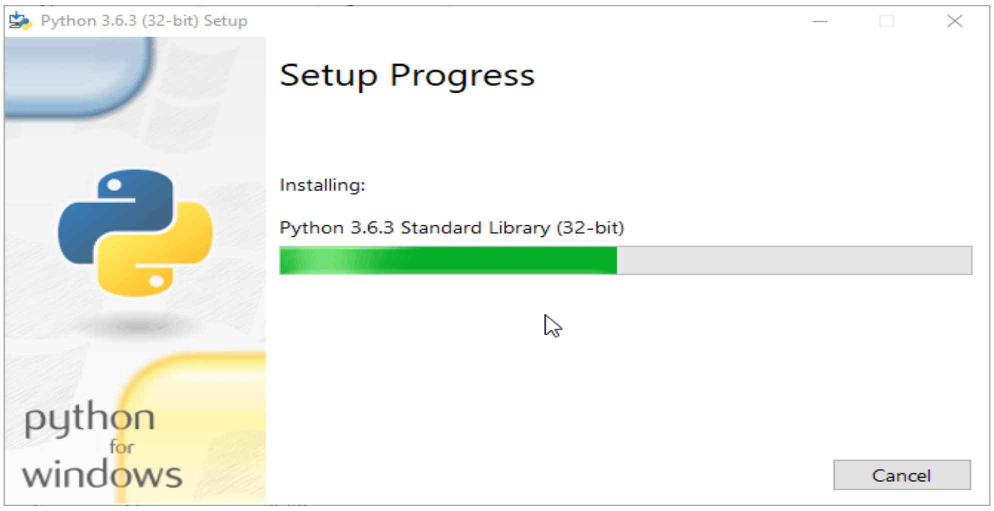
Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

Figure 3.4.2 Installation of Python

Step 3:

You can see Python installing at this point.


Python 3.6.3 (32-bit) Setup

Setup Progress

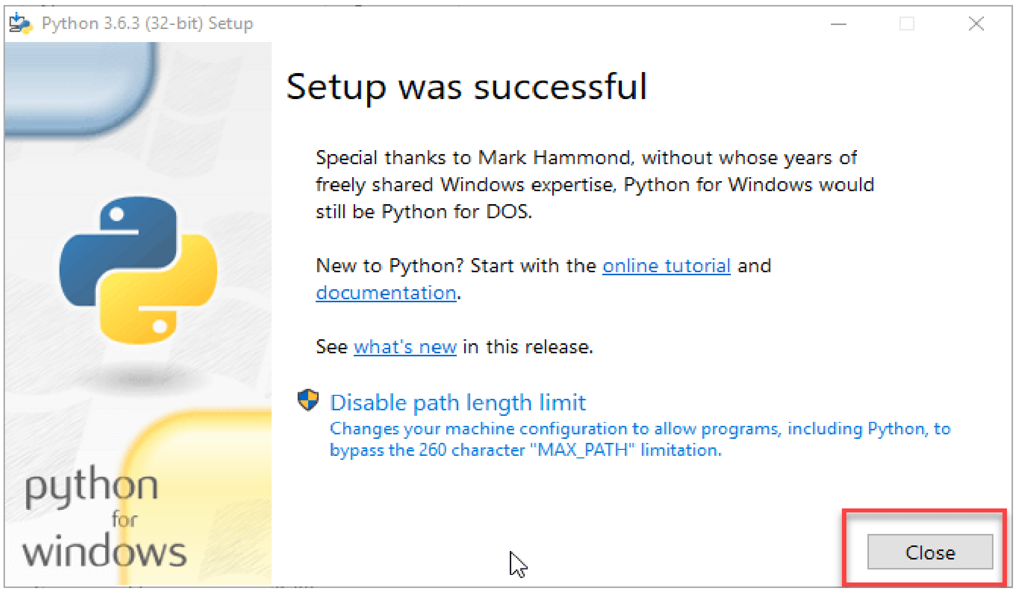
Installing:

Python 3.6.3 Standard Library (32-bit)

Figure 3.4.3 setting up Python

Step 4:

When it finishes, you can see a screen that says the Setup was successful. Now click on "Close".



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Figure 3.4.4 Installation Successful

CHAPTER: 4
DESIGN

4.1 Introduction

System design is the solution to the creation of a new system. This phase is composed of several systems. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design specifications to performance specification is system design. System design has two phases of development logical and physical design.

During logical design phase the analyst describes inputs (sources), outputs (destinations), databases (data stores) and procedures (data flows) all in a format that meats the uses requirements. The analyst also specifies the user needs and at a level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the candidate system must do.

The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

4.1.1. Input and Output Design:

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.

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.

4.2. System Architecture

4.2.1. Architectural Design

3-Tier architecture is also called layered architecture. Some people called it n-tier architecture. Layer architectures are essentially objects and work in object oriented environment. 3-tier architecture is a very well-known architecture in the world of software development, it doesn't matter whether you are developing web based application or desktop based, it is the best architecture to use.

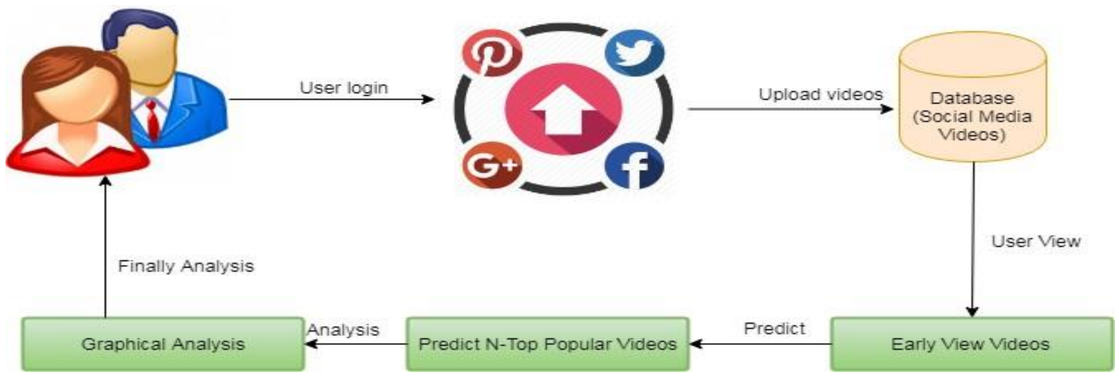


Figure 4.2.1 Architecture Design

3- Tier architecture consists of

1. UI or Presentation Layer
2. Business Access Layer or Business Logic Layer
3. Data Access Layer

Presentation Layer:

Presentation layer consists of pages like .java or desktop based form where data is presented to users or getting input from users.

Business Logic layer or Business Access Layer:

Business logic layer contains all of the business logic. Its responsibility is to validate the business rules of the component and communicating with the Data Access Layer. It is the class in which we write functions that get data from Presentation Layer and send that data to database through Data Access Layer.

Data Access Layer:

Data Access Layer is also the class that contains methods to enable business logic layer to connect the data and perform desired actions. These desired actions can be selecting, inserting, updating and deleting the data.

4.3. 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 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 Unified Modeling Language represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The Unified Modeling Language 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:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.

- Encourage the growth of the Object Oriented tools market.
- Support higher level development concepts such as collaborations, framew

4.3.1. Use Case Diagram

Use cases are used during the requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from the external point of view. A use case is a graph of actors, a set of use cases enclosed by the system boundary, communication association between actors and the use cases, and generalization among the use cases.

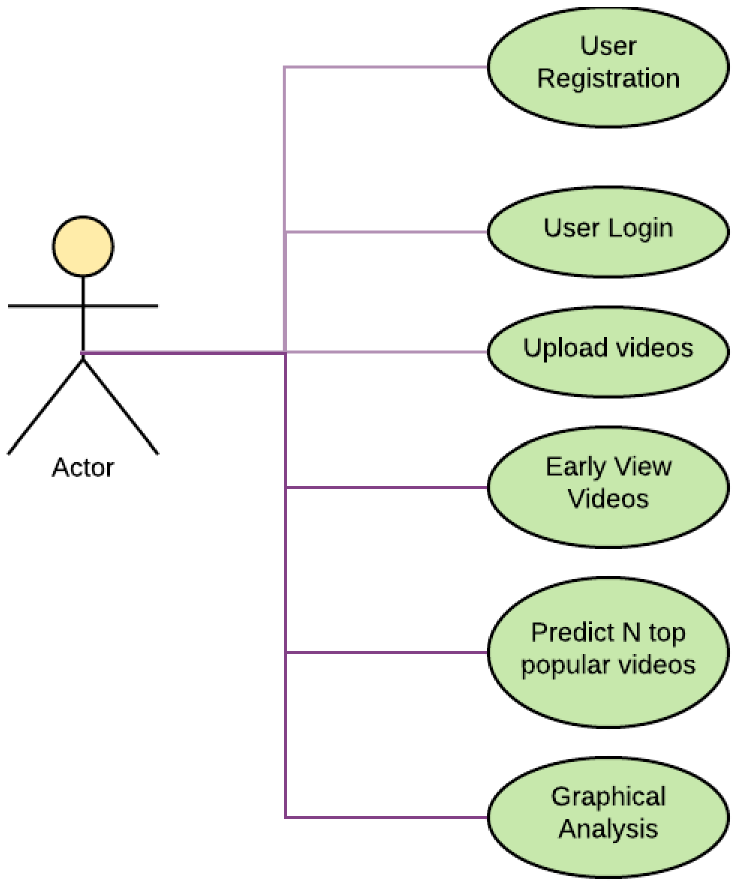


Figure 4.3.1.1 Use Case Diagram

4.3.2. Class Diagram

A class diagram is a type of static structural diagram which describes the structure of a system by showing the system's classes, their attributes, operations and the relationships among objects.

No of classes is based on No of Users.

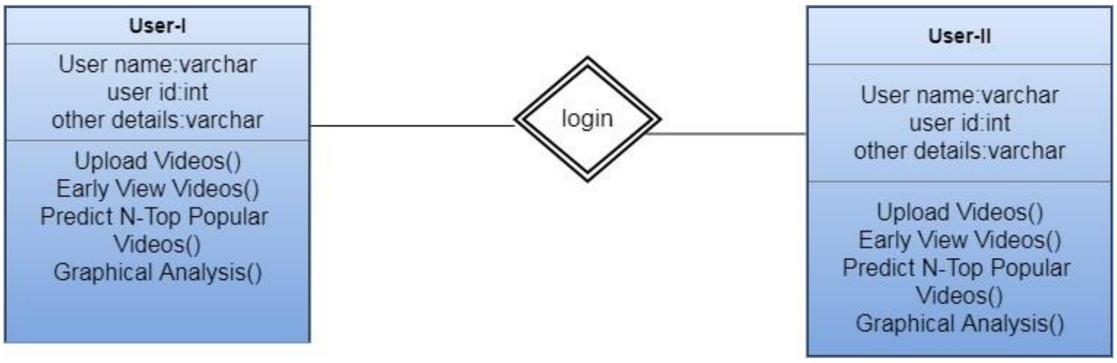


Figure 4.3.2 Class Diagram

4.3.3. Sequence Diagram:

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It has two dimensions, Vertical dimension represent time and Horizontal dimension represent different objects. The vertical line called Object Life Line that represents the objects existing during the interaction. The order in which messages appear is shown top to bottom. Each message is labeled with message name.

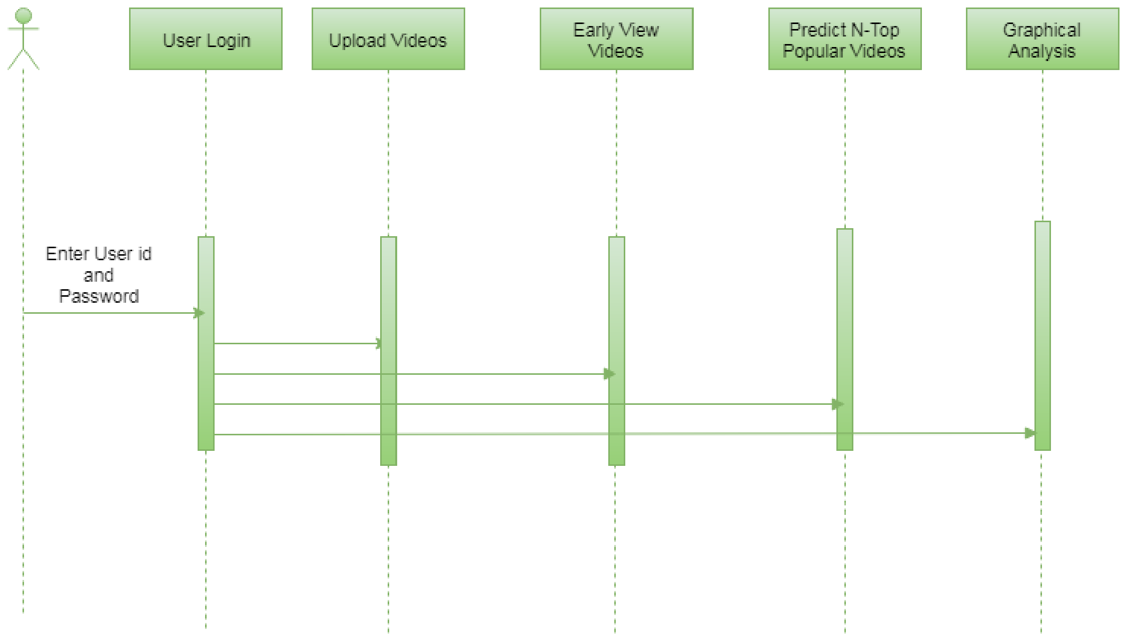


Figure 4.3.3 Sequence Diagram

4.3.4. Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of system. The Control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagram deal with al type of flow control by using different elements such as fork, join etc.

The basic Purpose of activity diagram is similar to other four diagrams. It captures the dynamic behavior of system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity another. Activity is a particular operation of the system.

Activity diagram are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. Activity diagram is sometimes considered as the flowchart. Although the diagram look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as Draw the activity flow of a system.

- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

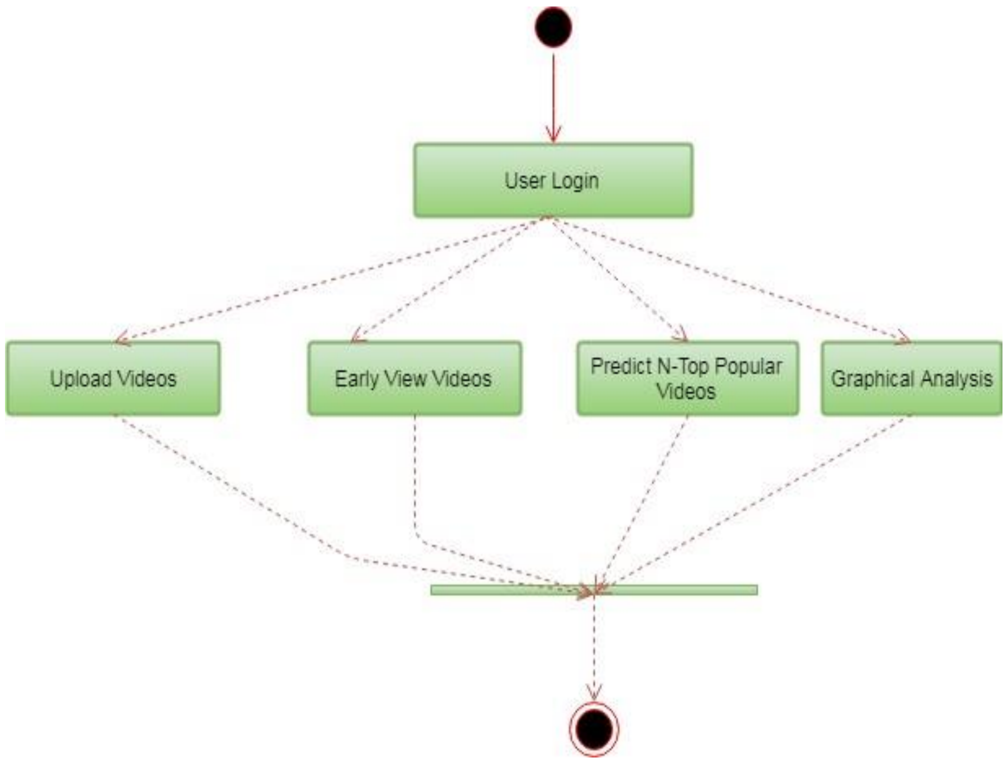


Figure 4.3.4 Activity Diagram

4.3.5. Component Diagram

The purpose of a component diagram is to show the relationship between different components in a system. For the purpose of UML 2.0, the term "component" refers to a module of classes that represent independent systems or subsystems with the ability to interface with the rest of the system.

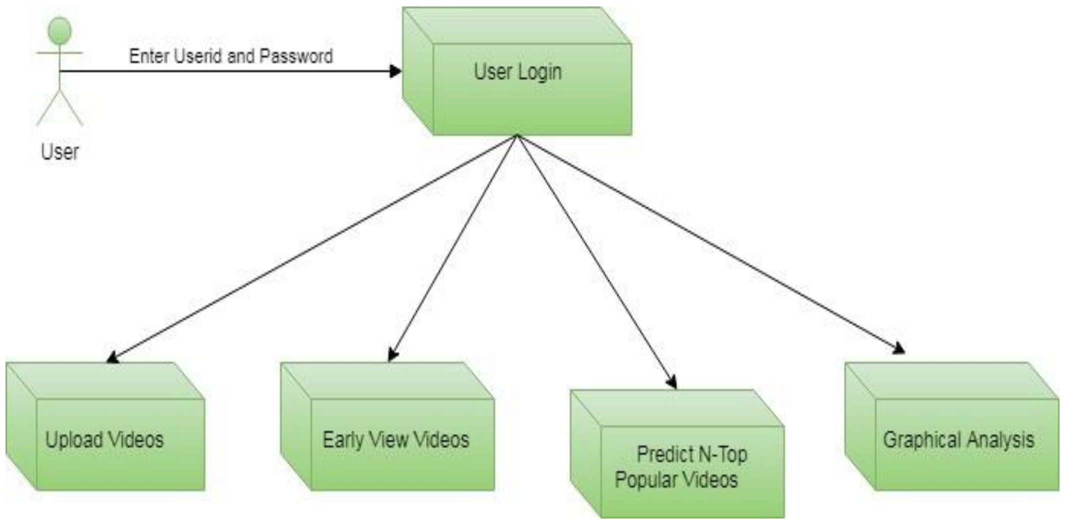


Figure 4.3.5 Component Diagram

CHAPTER: 5
IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

5.1. Modules Description

5.1.1. User Module

Register

The user registers into the app with the credentials (Name of the User, Phone number, Email-id, Password) so as to avoid registration next time.

User Sign-In

The user will login to the app using Gmail, in which he/she has/had to login using his Gmail it will be stored for later use. When a user already signed in using his Gmail he is going to use the app with the same Gmail until he removes that account from his mobile.

Get all All Videos

This module will uniquely identify and displays all the videos to the users.

5.1.2. Upload Videos

Predicting the top-N popular videos and their future views for a large batch of newly uploaded videos is of great commercial value to online video services (OVSs). Although many attempts have been made on video popularity prediction,the ability to identify the top-N popular videos is beneficial to video services, such as caching and recommendation. From the perspective of commercialization, identifying the top-N popular videos helps the video service providers to maximize their profits, as advertisers are more likely to pay more for popular videos.

5.1.3. Early View Videos

Similar early view patterns could lead to different popularity dynamics. For n

This problem is caused by the Pareto distribution of videos' popularity, as most of the views received by a video set are associated with only a few popular videos. Therefore, to reduce the prediction error over the entire video set, models will preferentially learn the popularity trends of the unpopular videos, hence sacrificing prediction performance on popular videos. Some recent studies have attempted to more deeply analyze the dynamics of video popularity and have related the popularity dynamics to various factors.

5.1.5. Graphical Analysis

The analysis of the system is done in this module. The proposed algorithm's efficiency is calculated here. The comparison of various factors can be handy to calculate and visualize in the graphs such as pie chart, bar chart, line chart. The data to plot the graph is taken from the system which is done.

5.2. Software Description

5.2.1. Python

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

5.2.2. 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 "plug ability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.

5.2.3. MySQL

Out of the box Django comes with SQLite database embedded within the project. Minimum amount of configurations and the ease of data storage and retrieval from SQLite is suitable for web apps on small scale, however for large scale web apps SQLite is going to cause performance issues. In order to avoid performance issues the first step is to select a suitable database able to cater hundreds or even thousands of simultaneous hits. MySQL database can be used for this purpose.

In this article I’ll follow you though all the necessary configurations required to connect Django with MySQL database.

Open the MySQL Workbench and create a new schema by clicking the create new schema in the connected server option.

Select a suitable schema/database name and press apply. This will create the schema and it will appear in the SCHEMAS section of the MySQL workbench.

5.2.4. Wamp Server

A Windows web admin's first instinct may be to install WampServer on a trusty IIS web server. For a number of reasons, this is not advisable, especially for a first-time installation. You may encounter port conflicts or other configuration problems that could thwart your efforts to get WampServer up and running smoothly.

For the 32-bit installation, we installed WampServer on a machine running a fresh install of Windows Server 2008 with Service Pack 2 (patched), with no server roles and no web services running. If you don't have a dedicated box for the install, you can test on a virtual machine. The latest version of WampServer is compatible with Windows 7 and Windows Server 2008. Previously released versions can operate on older Windows platforms going all the way back to Windows NT.

The WampServer installation on both our 32- and 64-bit Windows servers was surprisingly straightforward with just a few prompts from the Windows executable file we downloaded from WampServer.com (there are separate files for 32 and 64-bit architectures).

First, to make it easier to clearly identify and work with the newly installed WampServer files, we selected an empty, newly-formatted NTFS extended partition.

Next the WampServer installer prompted for a choice of a website browser. It defaulted to Internet Explorer, and we accepted the default, although we also later installed and tested WampServer with Google's Chrome browser.

5.3. ALGORITHM:

Novel Multifactor Differential Influence (MFDI):

To address the above problem, we present a novel popularity prediction model named multi-factor differential influence (MFDI) based on multivariate linear regression (MLR). We first enhance the ability of early view patterns to identify different popularity trends. We conduct a large-scale analysis of statistical data of early viewers' attitude-related behavior and the long-term popularity of videos. We find that the increase in the future popularity of videos follows an approximate Rayleigh distribution with respect to the degree of contradiction between early viewers with different attitudes. Based on this discovery, by combining early views with knowledge of early viewers' attitudes, we construct early rating patterns that offer better discriminative power for identifying popularity trend and use these rating patterns to replace early view patterns as the input to the proposed model. Furthermore, we incorporate the popularity of the videos' content on a social network to help the proposed model to discover popular videos and to learn their popularity trends.

CHAPTER: 6

TESTING AND VALIDATION

6.1. Introduction

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 requirement 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.

6.2. Design of Test cases and Scenarios

6.2.1. Espresso Testing:

Espresso is a tool used for doing unit testing. Espresso is used to write concise, beautiful and reliable Android User Interface test. The core API is small, predictable, and easy to learn and yet remains open for customization. Espresso tests state expectations, interactions, and assertions clearly without the distraction of boilerplate content, custom infrastructure, or messy implementation details getting in the way. Espresso tests run optimally fast! It lets you leave your waits, syncs, sleeps, and polls behind while it manipulates and asserts on the application UI when it is at rest.

Espresso is targeted at developers, who believe that automated testing is an integral part of the development lifecycle. While it can be used for black-box testing, Espresso’s full power is unlocked by those who are familiar with the code base under test. Espresso tests run optimally fast.

6.2.2. System Testing

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.

6.2.3. White Box Testing

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

6.2.4. 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. 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.

Procedure of running the application on a system using two virtual machines

1. First we create and install two android virtual machines.
2. Then we interface them with each other using their respective numbers.
3. The application which we have running will be automatically saved into the virtual machine.
4. Then when we open the application, it will display the speaker and user press speaker button record the voice it is converted into text messages

5. Then these text messages stores to external storage and finally it will send text messages to receiver.

How to run the application on an android phone

1. After running the application on the system once, an .apk file is generated automatically in the workplace.
2. We now copy the .apk file from /bin on the computer to the phone and install it.
3. After installing we run the application in the same way as we run it on a virtual machine.

6.3. Validation of Test Cases:

Test Cases	**Test case description**	**Expected result**	**Actual Result**	**Status**
Test Case 1	User Registration and Login	The user should login after registration is done	The user should login after registration is done	pass
Test Case 2	Displaying Videos	The videos present in the data set are displayed	The videos present in the data set are displayed	pass
Test Case 3	Uploading Videos	The user can upload the videos	The user can upload the videos	pass
Test Case 4	Checking the Functionality of buttons	Functionality of buttons is successful	Functionality of buttons is successful	pass
Test Case 5	Checking the functionality of making call	Call has been made successfully	Call has been made successfully	pass

Test Case 6	Displaying the top-N popular videos	The Top-N Popular videos are displayed	The Top-N Popular videos are displayed	pass
Test Case 7	Analyzing trending topic	Trending topic is analyzed by a bar char	Trending topic is analyzed by a bar char	pass

Table No 6.3 Validating Test Cases

At the culmination of integration testing the software is complete as a package and the interfacing errors have been uncovered and fixed, final tests -validation testing may begin. Validation tests succeed when the software performs exactly in the manner as expected by the user.

Software validation is done by a series of black box tests that demonstrate the conformance with requirements. Alpha and beta testing fall in this category. We will not do beta testing but alpha testing will certainly be done.

CHAPTER: 7
EXECUTION AND RESULT

7.1. User Screens

7.1.1. User Login Page

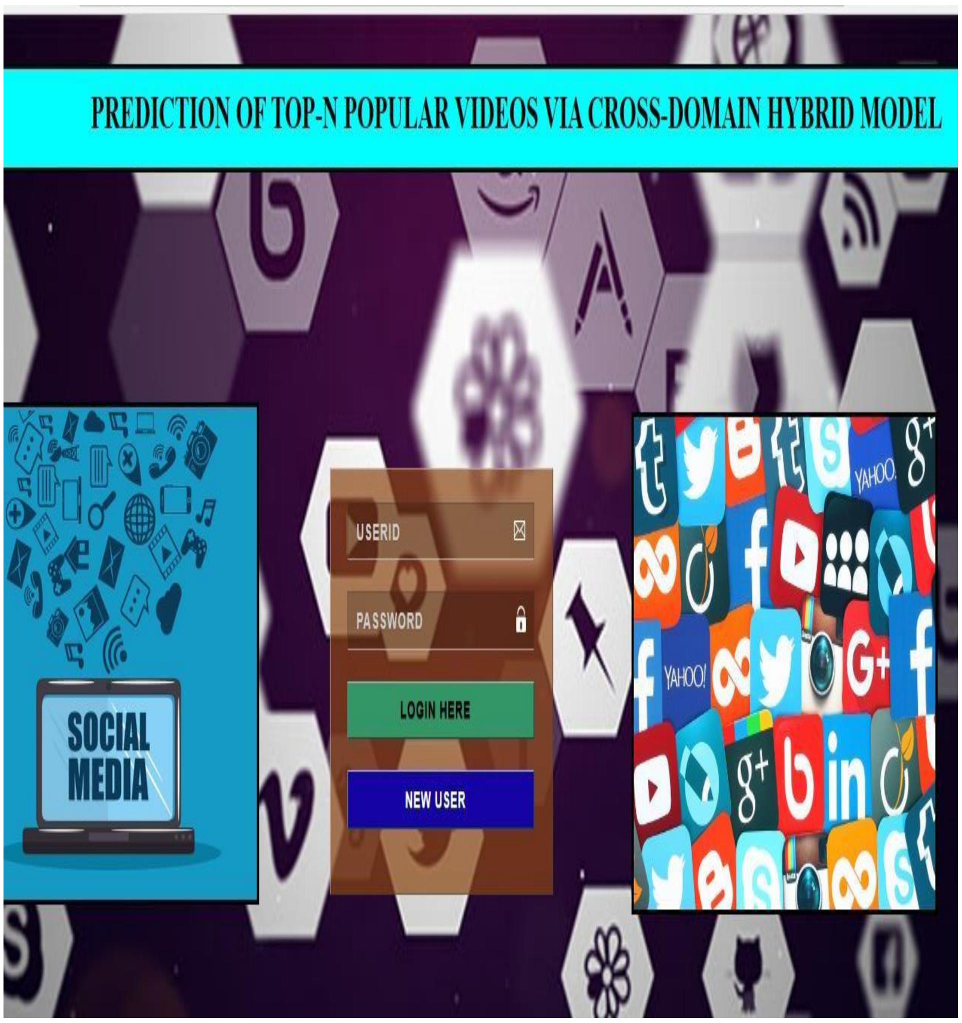
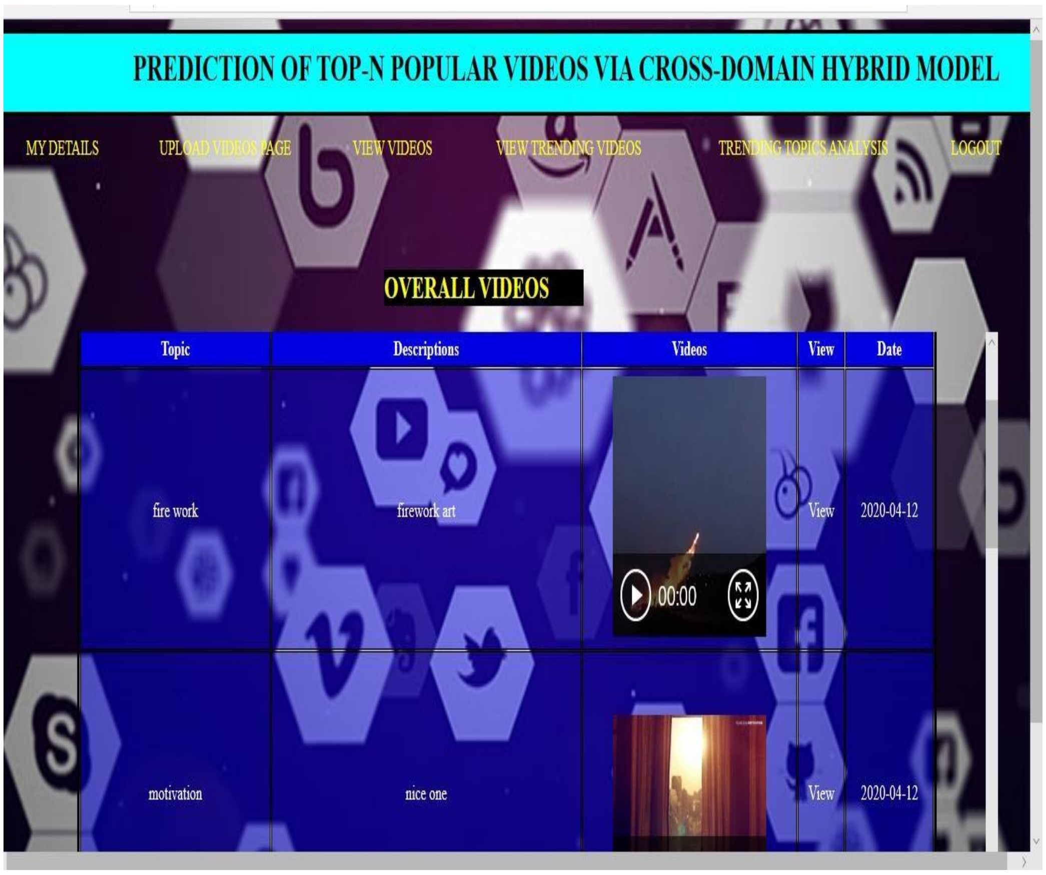

PREDICTION OF TOP-N POPULAR VIDEOS VIA CROSS-DOMAIN HYBRID MODEL
SOCIAL MEDIA
USERID
PASSWORD
LOGIN HERE
NEW USER

Figure 7.1.1 User Login Page

When a user opens the website the login page is displayed. The user after registering in the app, user logins by entering his User-ID and password. After login into the website the user navigates into view videos page.

7.1.2. View Videos Page


PREDICTION OF TOP-N POPULAR VIDEOS VIA CROSS-DOMAIN HYBRID MODEL

OVERALL VIDEOS

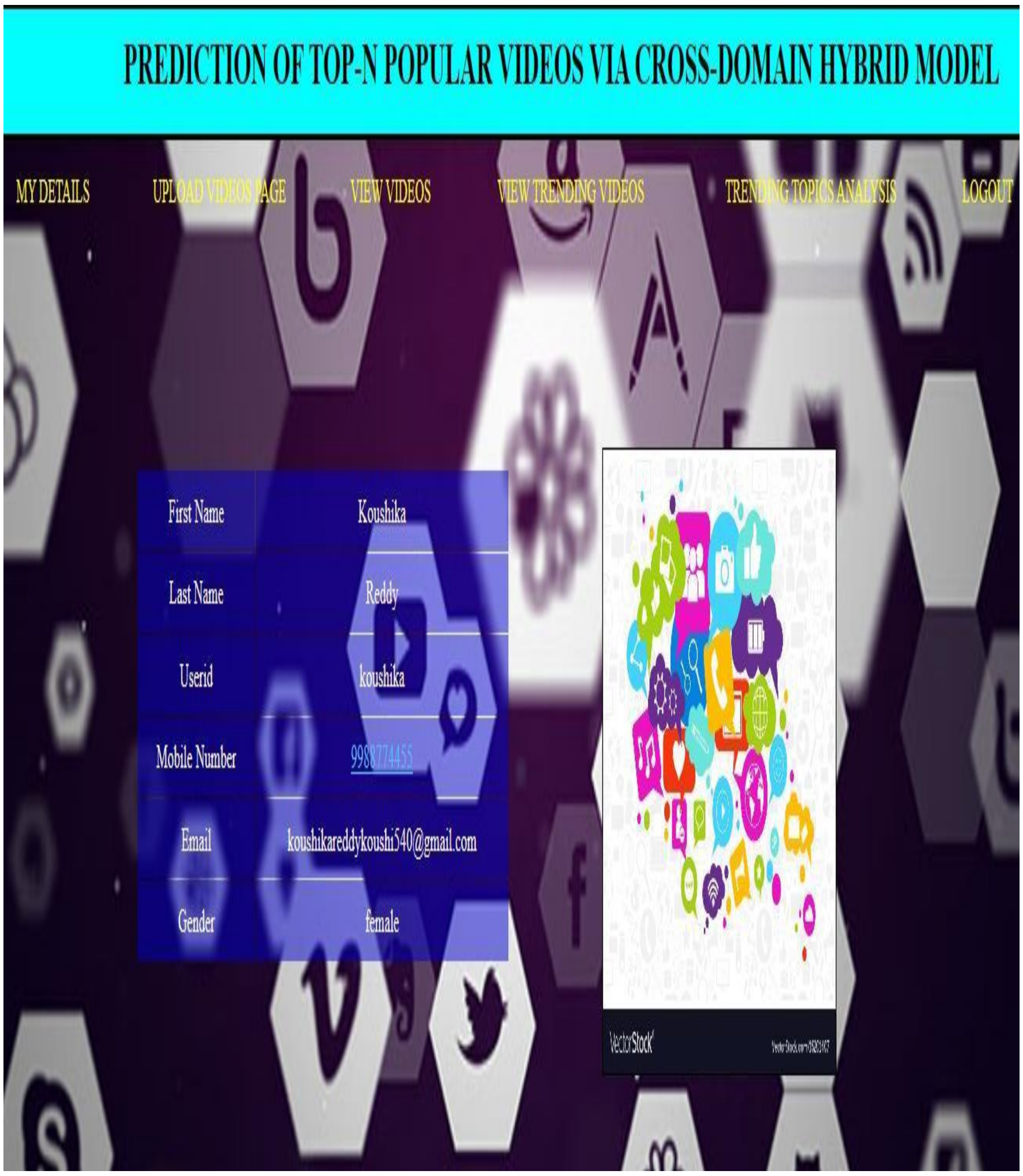
Topic	Descriptions	Videos	View	Date
fire work	firework art		View	2020-04-12
motivation	nice one		View	2020-04-12

Figure 7.1.2 View Videos Page

After login to the website, the user is able to see the overall videos available in the data set. We can know the topic to which the video belongs to and the description of the video. Here we can also know when the video was uploaded. We can watch the video by playing it, if the user likes the video he/she can like the video.

7.1.3. My Details Page

After login to the website if the user forgets his login details, he/she can know their login details by clicking on the login page which contains First Name, Last Name, User ID, Password, Gmail, Phone No and Gender. These details are stored in the data base.



First Name	Koushika
Last Name	Reddy
Userid	koushika
Mobile Number	9980774455
Email	koushikareddykoushi540@gmail.com
Gender	female

Figure 7.1.3 My Details Page

7.1.4. Upload Video Page:

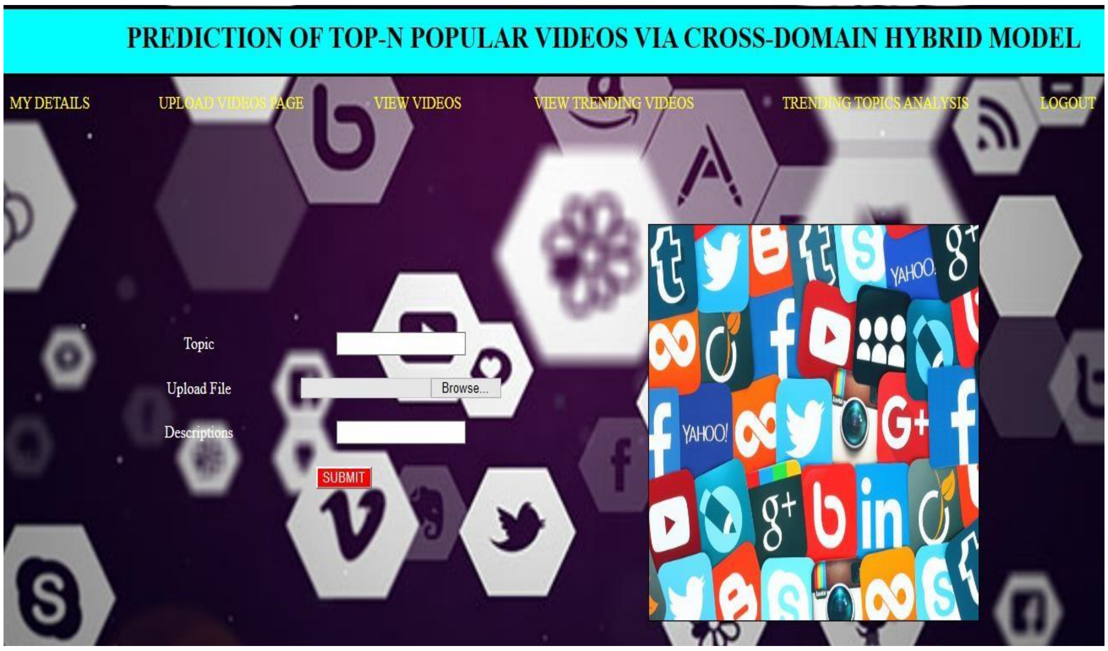
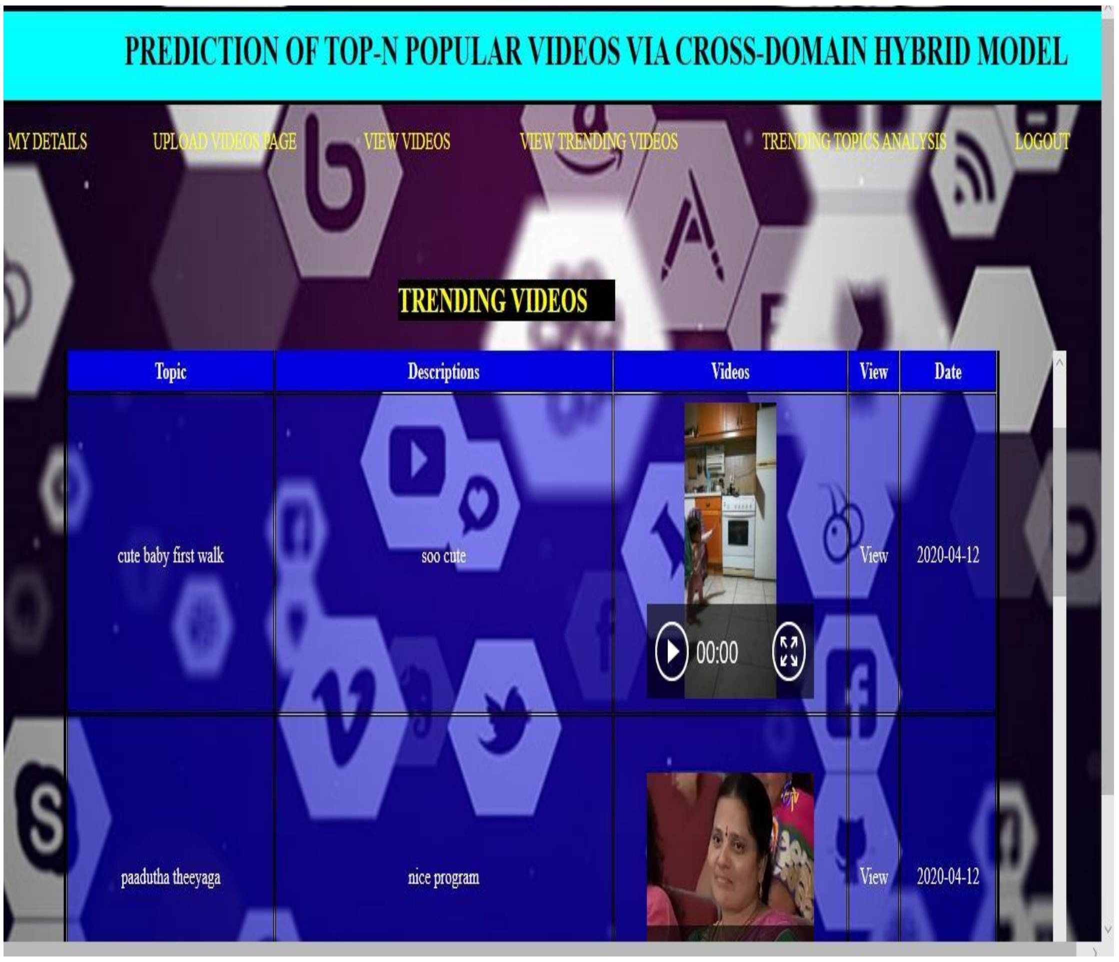


Figure: 7.1.4 Upload Video Page

In this page the user can upload a video into the dataset and this video is stored into the database. The user can upload a video by choosing the file from where he want to upload. He should give the topic and description of the videos. After uploading the video it is directly stored in the database.

..

7.1.5. View Trending Videos Page:


PREDICTION OF TOP-N POPULAR VIDEOS VIA CROSS-DOMAIN HYBRID MODEL

TRENDING VIDEOS

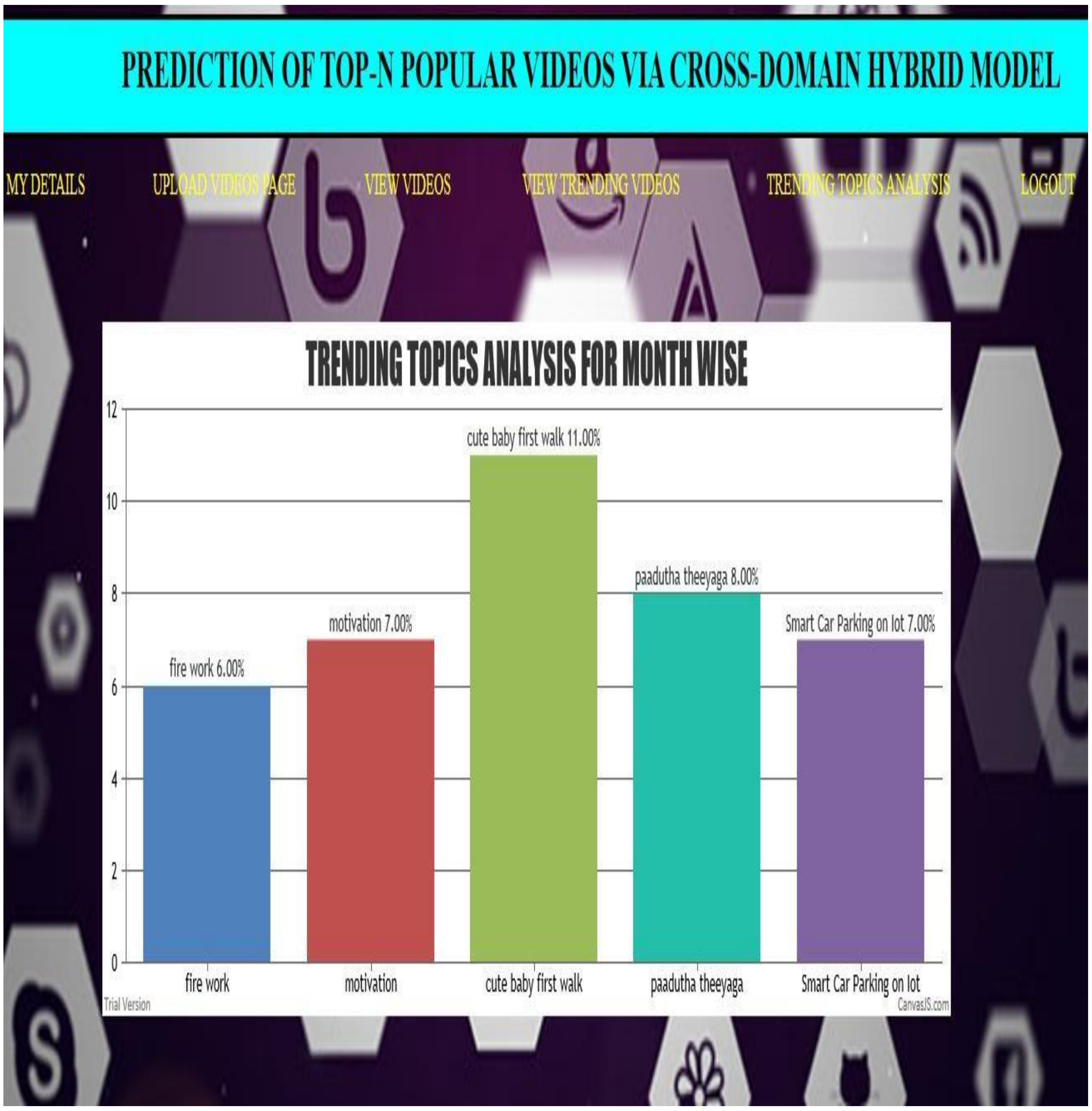
Topic	Descriptions	Videos	View	Date
cute baby first walk	soo cute	00:00	View	2020-04-12
paadutha theeyaga	nice program		View	2020-04-12

Figure 7.1.5 Trending Videos Page

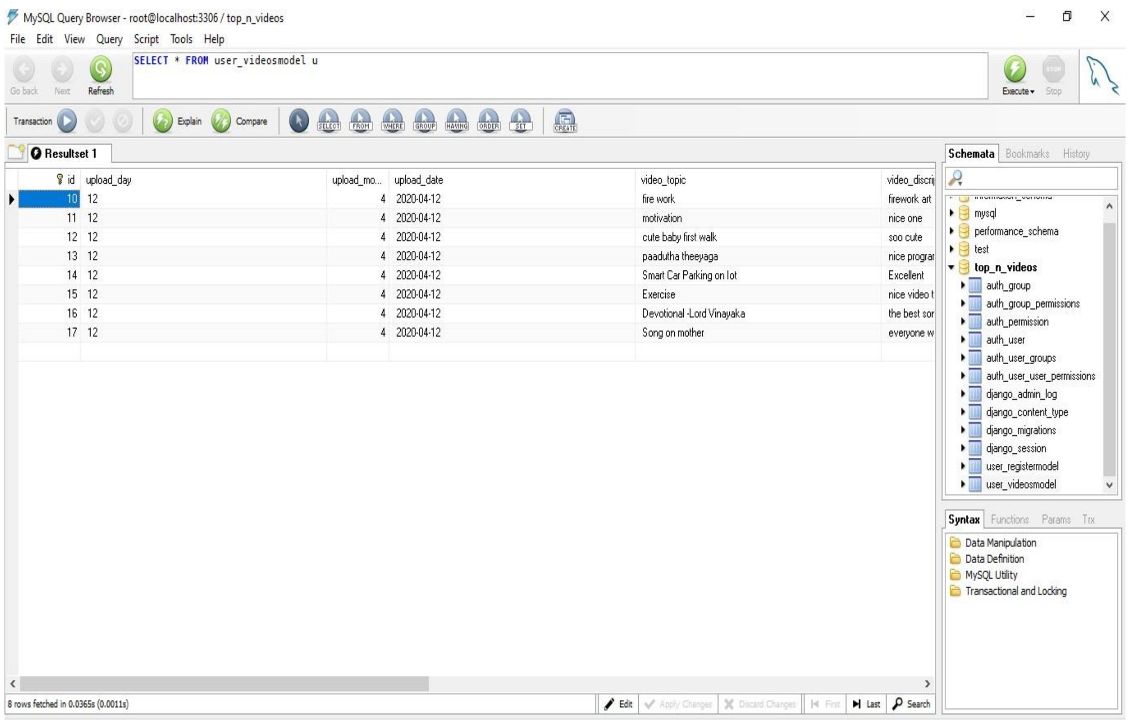
Trending videos page displays the popular videos which were uploaded into the database. The videos which have more views and likes comes under trending or popular videos list. Here we have two options to view the videos. First, we can directly watch the video by clicking on the play button. Second, by clicking on the view button. To like the video one must watch the video by clicking on the view button.

7.1.6. Trending topics Analysis

This page provides the entire information i.e., it represents the likes and views of the video in the form of bar chart with different colors for easy identification. The video which has more likes is considered as popular video and video which has less likes is considered as an unpopular video.



7.2. Database


MySQL Query Browser - root@localhost:3306 / top_n_videos
File Edit View Query Script Tools Help
Go back Next Refresh
Execute Stop
Transaction Explain Compare SELECT FROM WHERE GROUP HAVING ORDER SET CREATE
Resultset 1
Schemata Bookmarks History
mysql
performance_schema
test
top_n_videos
auth_group
auth_group_permissions
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session
user_registermodel
user_videosmodel
Syntax Functions Params Trx
Data Manipulation
Data Definition
MySQL Utility
Transactional and Locking
Edit Apply Changes Discard Changes First Last Search

```
SELECT * FROM user_videosmodel u
```

id	upload_day	upload_mo...	upload_date	video_topic	video_discri
10	12	4	2020-04-12	fire work	firework art
11	12	4	2020-04-12	motivation	nice one
12	12	4	2020-04-12	cute baby first walk	soo cute
13	12	4	2020-04-12	paadutha theeyaga	nice prograr
14	12	4	2020-04-12	Smart Car Parking on lot	Excellent
15	12	4	2020-04-12	Exercise	nice video t
16	12	4	2020-04-12	Devotional -Lord Vinayaka	the best sor
17	12	4	2020-04-12	Song on mother	everyone w

```
8 rows fetched in 0.0365s (0.0011s)
```

Figure 7.2 SQL DATABAS

CHAPTER: 8
CONCLUSION

In this article, we have investigated the problem of top-N popular video prediction and have proposed a novel MFDI prediction model. The proposed model predicts the top-N popular videos by enhancing the ability of early patterns to identify different popularity trends and by optimizing the model's utilization of multi-source data. Experimental results obtained using real-world data demonstrate that the proposed model outperforms other models, including the state-of-the-art model. This article is our initial study on popularity prediction for Top-N popular videos. To the best of our knowledge, this study is the first popularity prediction research to focus on top-N popular videos. Our study still has room for improvement. Possible improvements include leveraging additional related early features and discovering more precise mathematical correlations between the attitudes of early viewers and future popularity trends. For example, in this study, the early viewers' attitudes are inferred from only the three explicit behavior factors; however, early viewers' attitudes may also be reflected in many implicit ways. If more data related to early viewers' attitudes or similar features could be well modeled, they would be helpful for further improving the model's prediction performance, especially on the top-N popular videos.

CHAPTER: 9

REFERENCES

Journals:

[1] W. Hoiles, A. Aprem, and V. Krishnamurthy. Engagement and popularity Dynamics of youtube videos and sensitivity to meta-data. IEEE Transactions on Knowledge and Data Engineering, 29(7):1426–1437, July 2017.

[2] D. K. Krishnappa, M. Zink, C. Griwodz, and P. Halvorsen. Cachecentric video recommendation: An approach to improve the efficiency of youtube caches. ACM Trans. Multimedia Comput. Commun. Appl., 11(4):48:1–48:20, June 2015.

[3] C. Li, J. Liu, and S. Ouyang. Characterizing and predicting the popularity of online videos. IEEE Access, 4:1630–1641, 2016.

Web References:

5. Introduction to Python
Url: **https://www.geeksforgeeks.org/python-language-introduction/**

6. Django framework
Url: **https://django-book.readthedocs.io/en/latest/chapter01.html**

7. MySQL and Wampserver
Url: **https://forums.mysql.com/read.php?20,628837**