INDUSTRIAL ORIENTED MINI PROJECT

ON

STATIC KEYSTROKE DYNAMIC AUTHENTICATION MODEL TO AUTHENTICATE USER DURING PASSWORD CHANGE

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology.

In

Computer Science & Engineering

Submitted By

NUKALAGUDAM HIMA BINDU	**21RP1A0587**
SHAGA KAVYANJALI	**21RP1A05A4**
PINGILI SHRUTHI	**21RP1A0596**
VADLAKONDA SAISNEHA	**21RP1A05B8**

Under Guidance of

Mr. P. Venu Madhav

Assistant Professor,
CSE Dept.

Department of Computer Science & Engineering,

MEGHA INSTITUTE OF ENGINEERING &TECHNOLOGY FOR WOMEN

(Affiliated to JNTU-HYD, Approved by AICTE) Edulabad (V), Ghatkesar (M), Medchal (Dist.)-501301

(2021-2025)

CERTIFICATE

This is to certify that the Project entitled "STATIC KEYSTROKE DYNAMIC AUTHENTICATION MODEL TO AUTHENTICATE USER DURING PASSWORD CHANGE." is being submitted by **NUKALAGUDAM HIMABINDU 21RP1A0587, SHAGA KAVYANJALI 21RP1A05A4 , PINGILI SHRUTHI 21RP1A0596, VADLAKONDA SAISNEHA 21RP1A05B8** in partial fulfillment of the requirements for the award of the degree of B. Tech in Computer Science & Engineering, Affiliated to JNTU Hyderabad is a record of Bonafide work carried out by them under my guidance and supervision.

The result presented in this project work have been verified and found to be satisfactory. The results embodies in this report have not been submitted to any other University for the award of any other degree or diploma.

INTERNAL GUIDE
Mr. P. VENU MADHAV

Assistant Professor

HOD-CSE
Mr. S. VENKATESWARA RAO

Head of the Department,

Assistant Professor

EXTERNAL EXAMINAR

PRINCIPAL

DECLARATION

We hereby declare that the project entitled **"Static keystroke dynamic authentication model to authenticate user during password change."** submitted to the JNTUH in partial fulfillment of the requirements for the award of the degree of **B. Tech in Computer Science & Engineering** is a record of an original work done by us under the guidance of **Mr. P.VENU MADHAV, Assistant professor** and this project work have not been submitted to any other university for award of any degree or diploma.

NUKALAGUDAM HIMA BINDU - **21RP1A0587**

SHAGA KAVYANJALI - **21RP1A05A4**

PINGILI SHRUTHI - **21RP1A0596**

VADLAKONDA SAISNEHA - **21RP1A05B8**

ACKNOWLEDGEMENT

It is our privilege and pleasure to express our profound sense of respect, gratitude, and indebtedness to my project Guide **Mr. P. Venu Madhav**, **Assistant Professor**, Department of Computer Science and Engineering, Megha Institute of Engineering and Technology for Women, for their suggestions, motivation, indefatigable inspiration, guidance, cogent discussion, constructive criticisms, and encouragement throughout the project work.

We express our sincere gratitude to **Mr. S. Venkateshwara Rao**, **Head of The Department, Assistant Professor**, Department of Computer Science and Engineering, Megha Institute of Engineering and Technology for Women, for their suggestions, motivation indefatigable inspiration, guidance, cogent discussion, constructive criticisms, and encouragement throughout the project.

We extend our sincere thanks to **Mr. N. Mohan Reddy**, **Chairman**, **Dr. P. Ramasubramanian, Principal** of Megha Institute of Engineering and Technology for Women, Edulabad (V), Ghatkesar (M), Medchal Dist. for their encouragement and constant help. Last but not least, we wish to acknowledge my friends, family members and colleagues for giving moral strength and hoping us to complete this project.

Submitted by:

NUKALAGUDAM HIMA BINDU	**- 21RP1A0587**
SHAGA KAVYANJALI	**- 21RP1A05A4**
PINGILI SHRUTHI	**- 21RP1A0596**
VADLAKONDA SAISNEHA	**- 21RP1A05B8**

ABSTRACT

Keystroke dynamics is considered as a supporting factor of authentication. Especially in the static keystroke dynamics, the user is identified by using the timing featured, which is captured while the user enters the login ID and password. To achieve this, the user profile needs to be created with timing features. However, in a scenario like a change of password where nearly no keystroke timing data is available, non-conventional features may be helpful. This article focuses on using non- conventional features such as Num lock key, shift key, Caps Lock key, etc. for identifying users during a change of password. The paper also details how to capture the non-conventional features in static keystroke dynamics and build a model that can be used in the change of password.

INDEX

1. INTRODUCTION

Password based authentication is the most frequently used authentication method for providing access to personal computers, online services, internet access, etc. As password-based authentication has many limitations many times to strengthen the authentication multi-factor authentication is used. Which results in user grievance. To overcome this concern, use of support factor authentication is recommended. One such support factor of authentication is keystroke dynamics which is based on behavioral characteristics of a user, in this case the typing behavior. Thus, along with the correctness of the password, the keystroke analysis is used to verify if the user is the same as he claims. This behavioral biometric technique is considered to be unique per user.

There are two types of keystroke dynamics: static and continuous. In the static keystroke dynamics, the typing rhythm of a user is captured only at login time whereas in continuous keystroke dynamics keystrokes are captured continuously when the user is using his/her personal computer. For an identity management system which is checking the authenticity of the user only during the login process, static keystroke dynamics technique is used, as keystrokes cannot be captured by the system continuously. A lot of research has taken place in the use of static keystroke dynamics using timing parameters referred to as conventional features for authenticating users. The use of timing features requires capturing timing of every key press and release for the given characters. It means that when the password is decided for understanding the timings of the user it needs to be captured at least a few times until then it cannot be used as a support factor as well. Use of various keys such as Caps Lock key, shift key and Backspace Key like information is captured in continuous keystroke dynamics to improve the performance of the decision. As these do not capture timing such features are called as Non-Conventional features. In our earlier work, we have experimented and demonstrated that non-conventional features when combined with timing features can become a better supportive factor for user authentication and system performance is improved in static keystroke dynamics.

Since in static keystroke dynamics, a model is built for the user through training of timing features as well, whenever the user changes the password, the training cycle is again required to build the model for the user for a new password. It means that we will not be able to make the user of this supporting factor until the system gets trained. In this paper we propose that the use of non-conventional features in static keystroke dynamics can be used during change of password. It will help to predict the user during the training cycle as the model based on the non-conventional features for the user will be available. This paper explores and discusses this possibility and proposes a model that uses non-conventional features as a support factor when the user wants to change the password until the model gets created through timing features for the new password. The paper is organized as follows: Section II of the paper gives the literature study and research done in the area of keystroke dynamics, non-conventional features and the existing dataset. Section III describes the Keystroke Dynamics System and its features, also explains the password change scenario and the challenges in using only the timing features.

Section IV gives details about the keystroke dynamics non- conventional features, how to capture them, challenges in using them in change of password scenario, need of policy, describes the policies that are used for framing passwords and text to capture non-conventional features in static keystroke dynamics along with the test cases. Section V explains the proposed approach to authenticate users during change of password through the use of static keystroke dynamics with non-conventional as well as timing features, along with experimentation which was carried out on the proposed approach to validate the model. Section VI focuses on the experimental results of the research work. Section VII concludes the paper with concluding remarks.

2. LITERATURE SURVEY

1) Continuous Authentication in a real-world settings.

AUTHORS: Soumik Mondal, Patrick A. H. Bours.

Continuous Authentication by analyzing the user's behavior profile on the computer input devices is challenging due to limited information, variability of data and the sparse nature of the information. As a result, most of the previous research was done as a periodic authentication, where the analysis was made based on a fixed number of actions or fixed time period. Also, the experimental data was obtained for most of the previous research in a very controlled condition, where the task and environment were fixed. In this paper, we will focus on actual continuous authentication that reacts on every single action performed by the user. The experimental data was collected in a complete uncontrolled condition from 52 users by using our data collection software. In our analysis, we have considered both keystroke and mouse usages behavior pattern to avoid a situation where an attacker avoids detection by restricting to one input device because the continuous authentication system only checks the other input device. The result we have obtained from this research is satisfactory enough for further investigation on this domain.

2) Identity authentication based on keystroke latencies

AUTHORS: R. Joyce, G. Gupta

The variables that help make a handwritten signature a unique human identifier also provide a unique digital signature in the form of a stream of latency periods between keystrokes. This article describes a method of verifying the identity of a user based on such a digital signature, and reports results from trial usage of the system.

3) Keystroke dynamics as a biometric for authentication

AUTHORS: Fabian Monorose, Aviel D. Rubin

More than ever before the Internet is changing computing as we know it. Global access to information and resources is becoming an integral part of nearly every aspect of our lives. Unfortunately, with this global network access comes increased chances of malicious attack and intrusion. In an effort to confront the new threats unveiled by the networking revolution of the past few years reliable, rapid, and unintrusive means for automatically recognizing the identity of individuals are now being sought. In this paper we examine an emerging non-static biometric technique that aims to identify users based on analyzing habitual rhythm patterns in the way they type

4). Biometric Identification of a Genuine User/Imposter from Keystroke Dynamics Dataset.

AUTHORS: R.Abhinaya, An.Sigappi

Keystroke dynamics has been used to strengthen password-based user authentication systems by considering the typing characteristics of legitimate users. Dependence on computers to store and process sensitive information has made it necessary to secure them from intruders a behavioral biometric, keystroke dynamics flow which makes utilization of the typing style of an individual can be utilized to reinforce existing security systems adequately and inexpensively and the examination of keystroke validation, to use the Discrete Cosine Transform (DCT) to describe the keystroke progression, and gives Bei Hang keystroke dataset comes about are one of the well-known classifier random forest classifier it best results achieved were respectively 90% accuracy when compared with other classifier results such as Support Vector Machine and Random tree classifier.

3. SYSTEM SPECFICATIONS

3.1 HARDWARE REQUIREMENTS:

- **System** : Intel i5
- **Hard Disk** : 1 TB.
- **Monitor** : 14’ Colour Monitor.
- **Mouse** : Optical Mouse.
- **Ram** : 8GB.

3.2 SOFTWARE REQUIREMENTS:

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

4. SYSTEM ANALYSIS

EXISTING SYSTEM:

The current system employs static keystroke dynamics and dynamic authentication for user verification during password changes. During enrollment, users establish a baseline profile through static keystroke data. When changing passwords, users provide dynamic keystroke data by typing a specific prompt. The system compares this dynamic data to the baseline for user authentication. Security measures, user experience, continuous improvement, and adherence to privacy regulations with user consent are integrated into the existing system. The overall aim is to provide a secure, user-friendly, and privacy-compliant solution for enhanced user authentication during password changes.

DISADVANTAGES OF EXISTING SYSTEM:

The existing keystroke dynamics system, while innovative, has several disadvantages. Firstly, keystroke dynamics can be affected by variations in user behavior due to stress, fatigue, or changes in typing patterns, leading to false rejections or acceptances. Secondly, the system may struggle with accommodating users who have inconsistent typing habits or disabilities, reducing accessibility. Additionally, enrollment requires a significant amount of static data, which may lead to user frustration and hinder adoption. The reliance on dynamic keystroke data for password changes could also introduce delays, impacting user experience. Moreover, this system can be vulnerable to sophisticated attacks, such as replay or mimicry attacks, where attackers imitate a user's typing style. Lastly, continuous monitoring and data collection could raise privacy concerns if not handled transparently, despite user consent.

PROPOSED SYSTEM:

The proposed SKDA system combines static keystroke dynamics and dynamic authentication for enhanced user verification during password changes. Users create a baseline profile during enrollment, and during a password change, dynamic keystroke data is collected through a specific typing prompt. The system compares this dynamic data with the baseline to authenticate the user, allowing for a secure password update. Emphasis is placed on security measures, positive user experience, continuous improvement, and adherence to privacy regulations with user consent for data collection. The goal is to provide a robust, user-friendly, and privacy-compliant solution for improved authentication.

ADVANTAGES OF PROPOSED SYSTEM:

The proposed SKDA system offers several advantages for enhancing user authentication during password changes. By combining static and dynamic keystroke dynamics, it strengthens security through multi-layered verification, reducing the risk of unauthorized access. The use of a baseline profile ensures that authentication remains accurate over time, even if minor variations in typing occur. This approach enhances user convenience, as it eliminates the need for additional hardware or complex authentication steps. The system also supports continuous improvement through adaptive learning, making it more effective with frequent use. Furthermore, strict adherence to privacy regulations and user consent ensures that data is handled securely and transparently, fostering trust and compliance. Overall, it provides a robust, user-friendly, and secure solution for password.

5.SYSTEM STUDY

FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are,

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

ECONOMICAL FEASIBILITY:

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

TECHNICAL FEASIBILITY:

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

SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

6. SOFTWARE ENVIRONMENT

6.1 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. C Python, which is the reference implementation of Python, is considered as an open source software and has a community based development model, as do nearly all of its variant implementations. C Python is managed by the nonprofit 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.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

Hello, Python!

SCRIPT MODE PROGRAMMING

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

```
Hello, Python!
```

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python

print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

```
$ chmod +x test.py     # This is to make file executable
$./test.py
```

This produces the following result −

```
Hello, Python!
```

PYTHON IDENTIFIERS

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

1. Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

2. Starting an identifier with a single leading underscore indicates that the identifier is private.

3. Starting an identifier with two leading underscores indicates a strongly private identifier.

4. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and, exec, not,
assert, finally, or,
break, for, pass,
class, from, print,
continue, global,
raise, def, if,
return, del,
import, try, else,
if, in, while, else,
is, with, expect,
lambda, yield.

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python

import sys

try:
   # open file stream
   file = open(file_name, "w")
```

```
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
if len(file_name) == 0:
   print "Next time please enter something"
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
    'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python

# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
This is a multiline
comment.
'''
```

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python

raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class

require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concept.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Updating Tuples

Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Zara
dict['Age']:  7
```

If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
```

When the above code is executed, it produces the following result −

```
dict['Alice']:
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result −

```
dict['Age']:  8
dict['School']:  DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note − del() method is discussed in subsequent section

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

```
 Live Demo
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

```
 Live Demo
#!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

```
 Live Demo
#!/usr/bin/python

tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;

# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);
print tup;
del tup;
print "After deleting tup : ";
print tup;
```

This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
Name Error: name 'tup' is not defined.
```

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

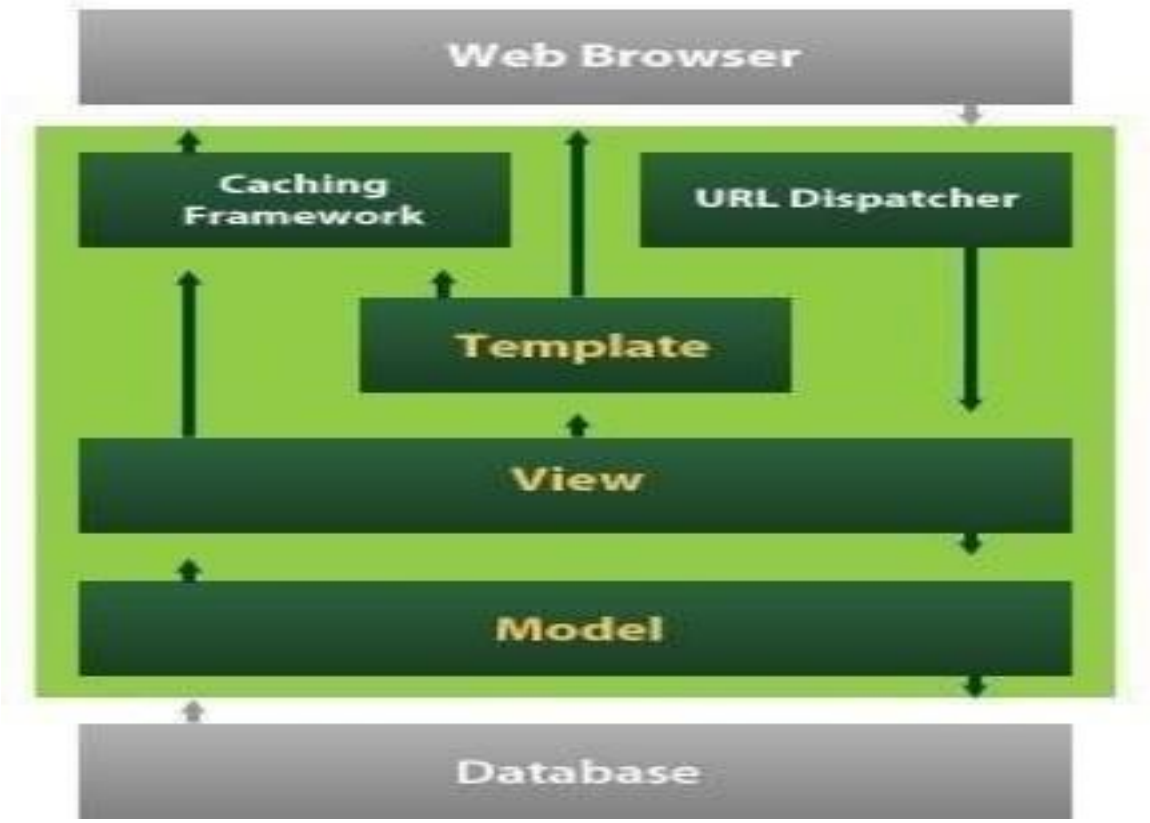


Fig.a

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

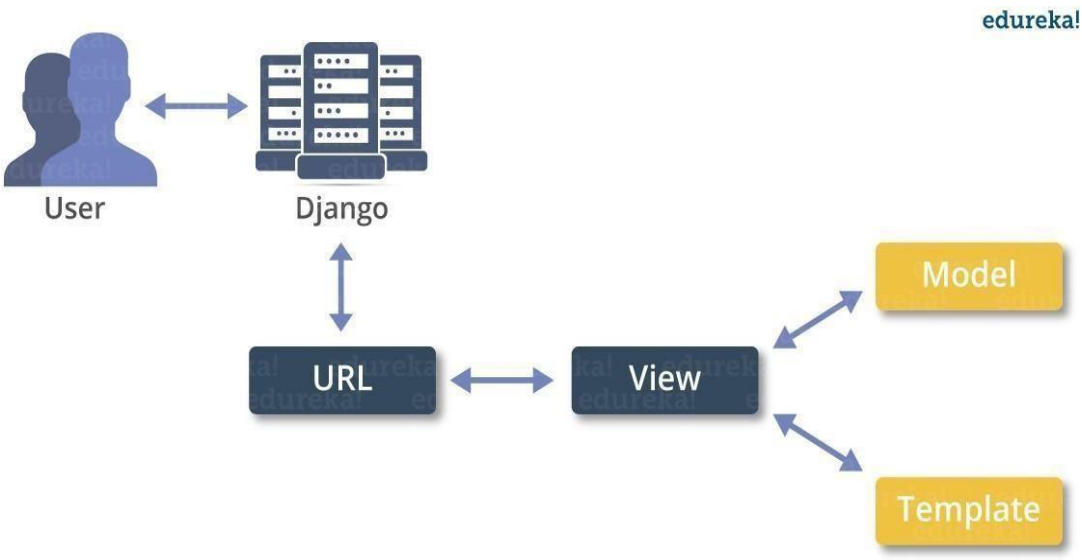


Fig.b

Create a Project

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

$ django-admin startproject myproject
This will create a "myproject" folder with the following structure −

```
myproject/
  manage.py
  myproject/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

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

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

$ python manage.py help
The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

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

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

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

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

Setting Up Your Project

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

DEBUG = True

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

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

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

MySQL (django.db.backends.mysql)
PostGreSQL (django.db.backends.postgresql_psycopg2)
Oracle (django.db.backends.oracle) and NoSQL DB
MongoDB (django_mongodb_engine)

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

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

Now that your project is created and configured make sure it's working −

$ python manage.py runserver
You will get something like the following on running the above code −

Validating models...

0 errors found
September 03, 2015 - 11:41:50
Django version 1.6.11, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

$ python manage.py startapp myapp

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

myapp/

__init__.py

admin.py

models.py

tests.py

views.py

__init__.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

```
INSTALLED_APPS = (
  'django.contrib.admin',
  'django.contrib.auth',
  'django.contrib.contenttypes',
  'django.contrib.sessions',
  'django.contrib.messages',
  'django.contrib.staticfiles',
  'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from django import forms
class LoginForm(forms.Form):
   user = forms.CharField(max_length = 100)
   password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from myapp.forms import LoginForm
def login(request):
   username = "not logged in"
    if request.method == "POST":
      #Get the posted form
      MyLoginForm = LoginForm(request.POST)
```

```
    if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
  else:
    MyLoginForm = Loginform()
        return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
  <body>
    <form name = "form" action = "{% url "myapp.views.login" %}"
     method = "POST" >{% csrf_token %}
      <div style = "max-width:470px;">
        <center>
          <input type = "text" style = "margin-left:20%;"
            placeholder = "Identifiant" name = "username" />
        </center>
      </div>
        <br>
      <div style = "max-width:470px;">
        <center>
          <input type = "password" style = "margin-left:20%;"
            placeholder = "password" name = "password" />
        </center>
      </div>
```

```
<br>
 <div style = "max-width:470px;">
  <center>
   <button style = "border:0px; background-color:#4285F4; margin-
top:8%;
      height:35px; width:80%;margin-left:19%;" type = "submit"
      value = "Login" >
      <strong>Login</strong>
    </button>
  </center>
 </div>
 </form>
</body>
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

```
{% csrf_token %}
```

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>
  <body>
   You are : <strong>{{username}}</strong>
  </body>
```

```
</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
urlpatterns = patterns('myapp.views',
  url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),
  url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered −

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have −

```
'django.contrib.sessions.middleware.SessionMiddleware'
```

And INSTALLED_APPS should have −

```
'django.contrib.sessions'
```

By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side –

```
def login(request):
   username = 'not logged in'
    if request.method == 'POST':
      MyLoginForm = LoginForm(request.POST)
      if MyLoginForm.is_valid():
         username = MyLoginForm.cleaned_data['username']
         request.session['username'] = username
      else:
         MyLoginForm = LoginForm()

   return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won't display the form if cookie is set −

```
def formView(request):
   if request.session.has_key('username'):
```

```
    username = request.session['username']
    return render(request, 'loggedin.html', {"username" : username})
  else:
    return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view

–

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView
  urlpatterns = patterns('myapp.views',
  url(r'^connection/','formView', name = 'loginform'),
  url(r'^login/', 'login', name = 'login'))
```

When accessing /myapp/connection, you will get to see the following page

7. SYSTEM DESIGN

SYSTEM ARCHITECTURE:

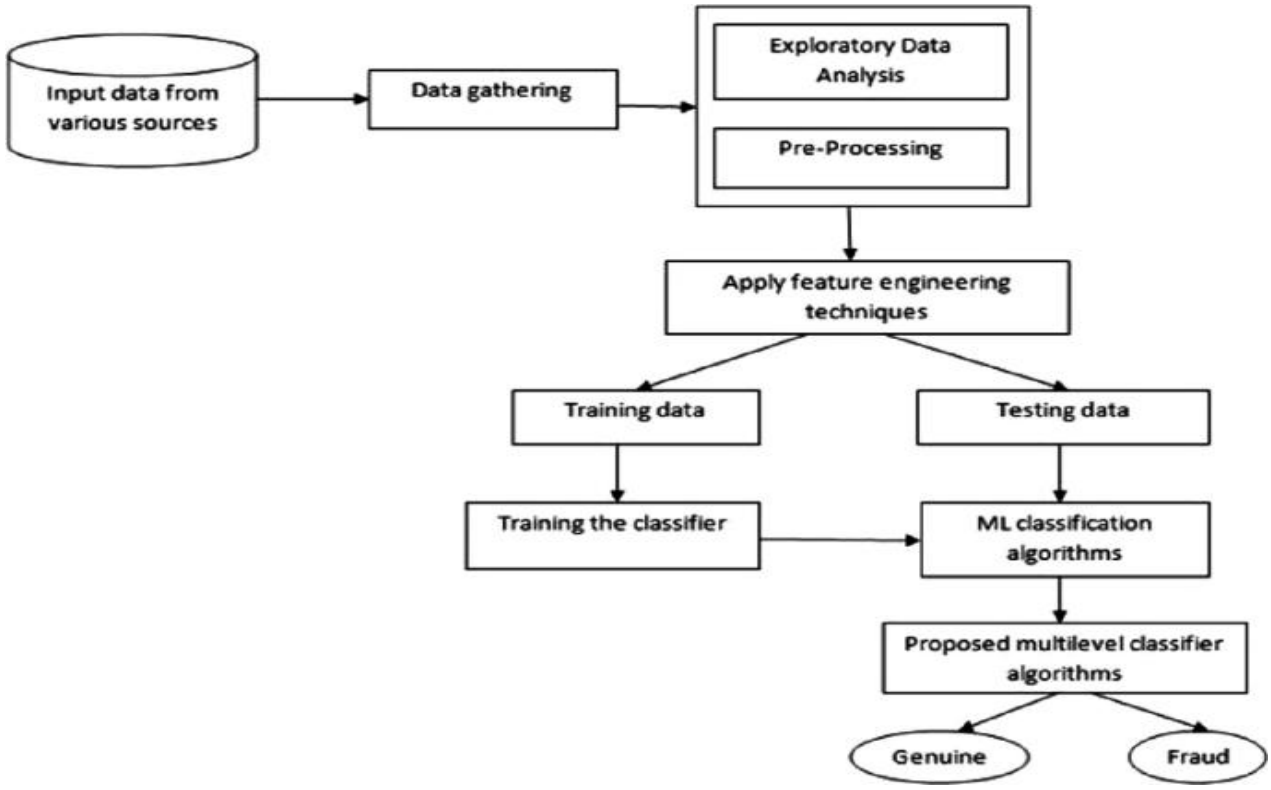


Fig. system architecture

DATA FLOW DIAGRAM:

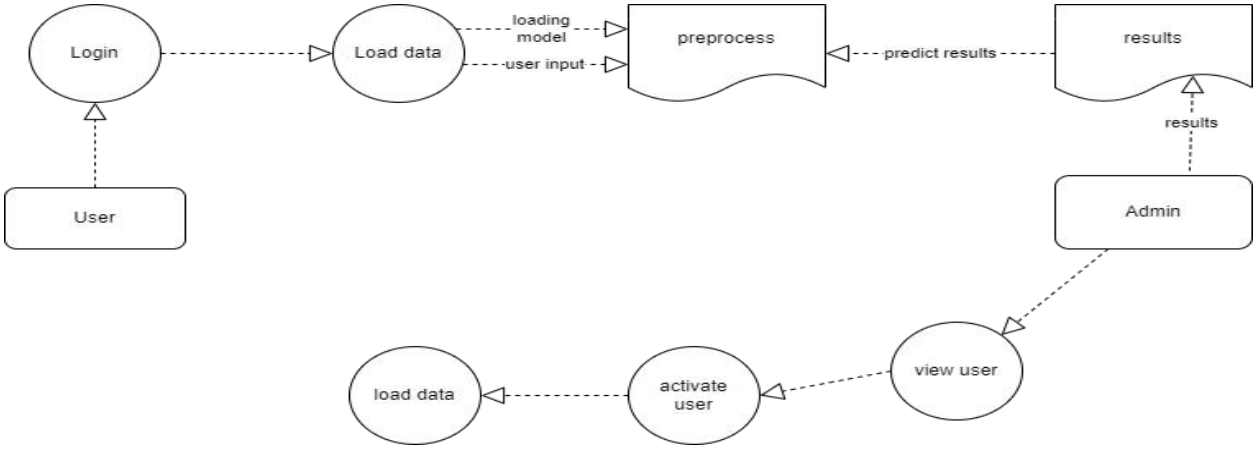


Fig. Data flow

1. The is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

7.1 UML DIAGRAMS

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

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

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

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

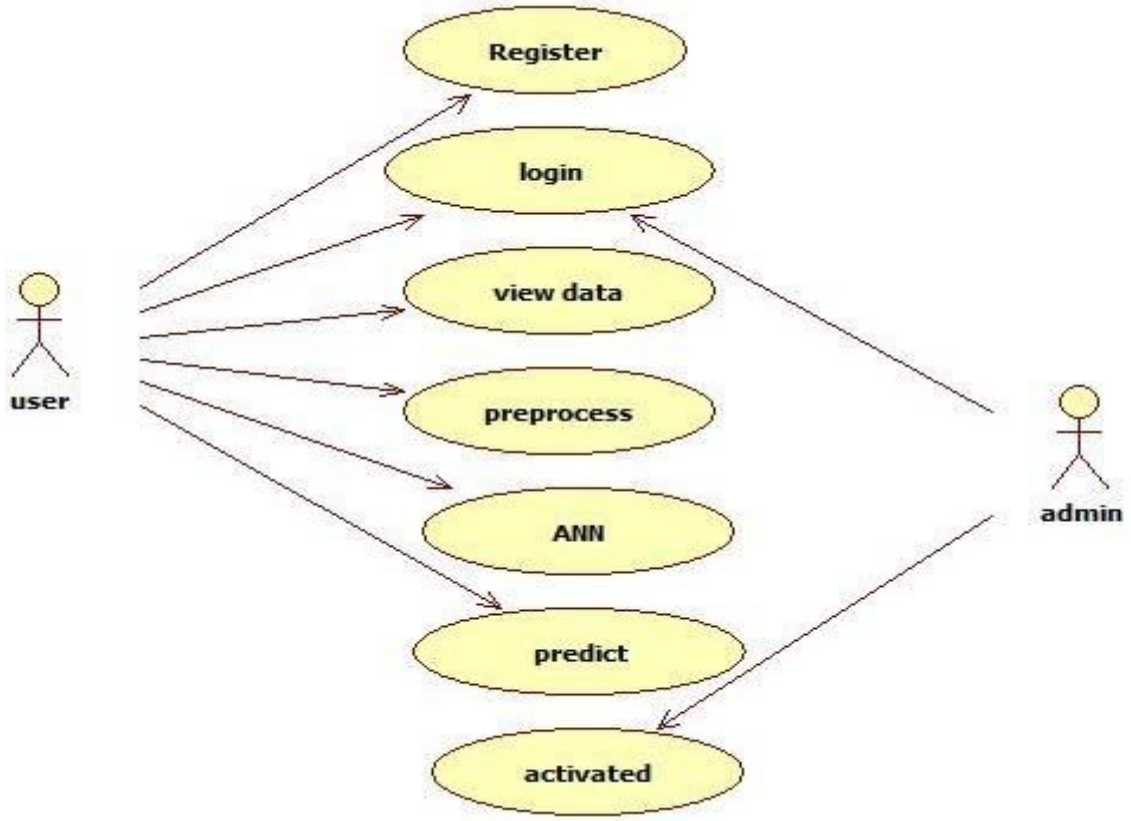
The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

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

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

USE CASE DIAGRAM:



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

CLASS DIAGRAM:

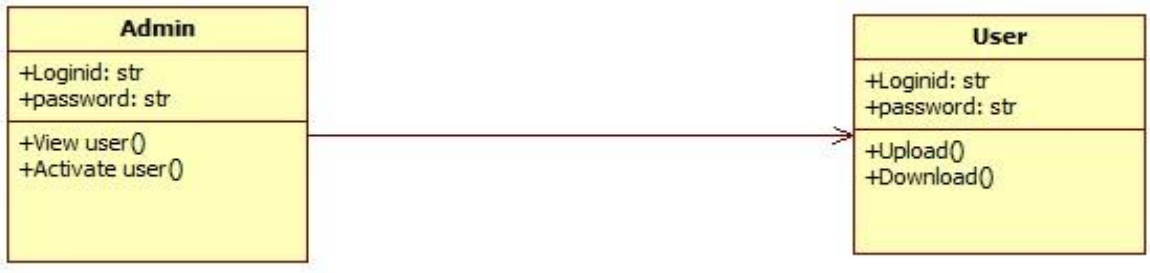


Fig. Class diagram

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

SEQUENCE DIAGRAM:

Fig. Sequence diagram

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

ACTIVITY DIAGRAM:

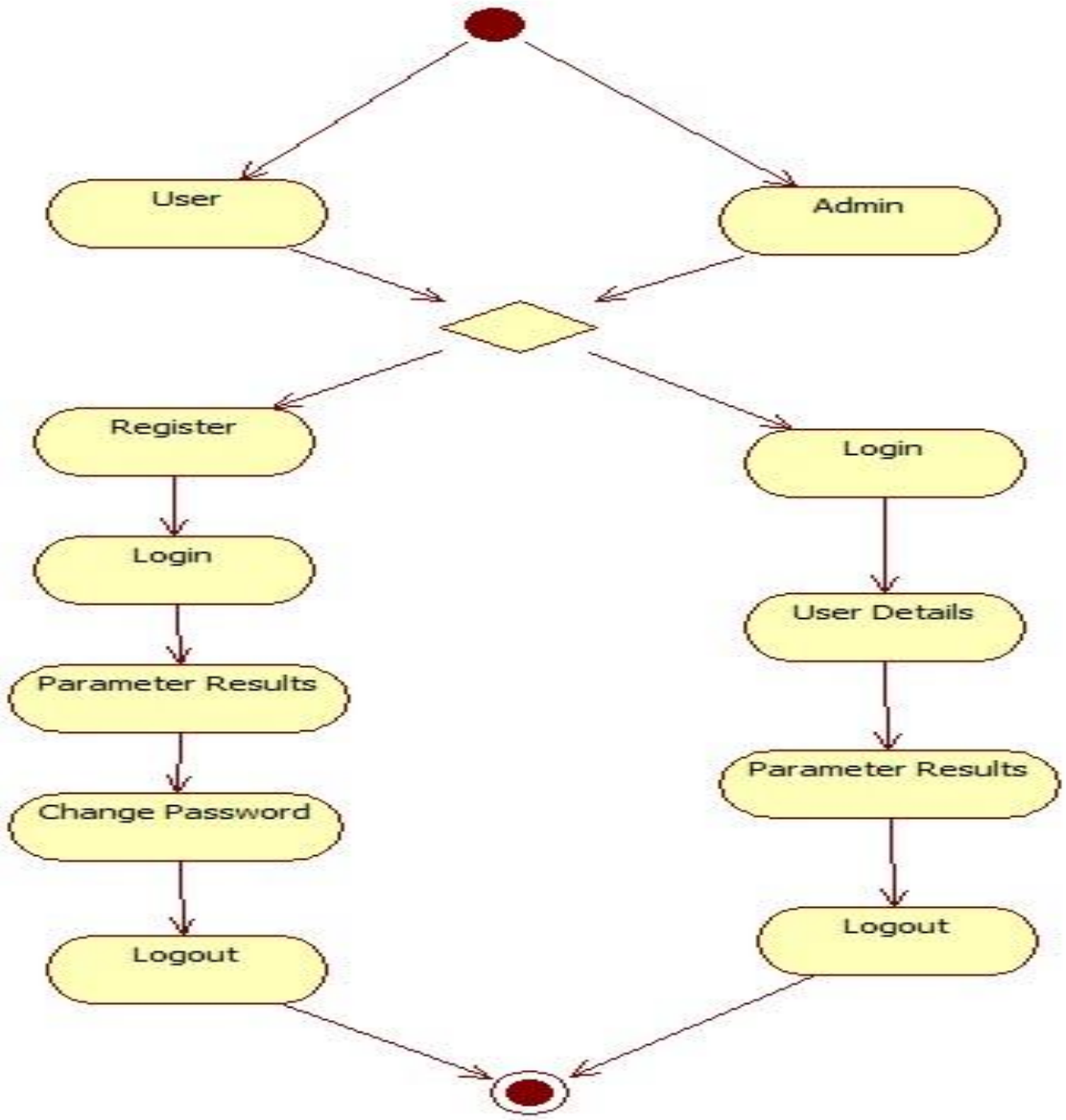


Fig. Activity diagram

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

8. IMPLEMENTATION

8.1 MODULES:

- User
- Admin

8.2 MODULES DESCRIPTION:

User:

The User can register the first. While registering he required a valid user email and mobile for further communications. Once the user register then admin can activate the user. Once admin activated the user then user can login into our system. User can upload the dataset based on our dataset column matched. For algorithm execution data must be in float format. Here we took Employment Scam Aegean Dataset (EMSCAD) containing 18000 sample datasets. User can also add the new data for existing dataset based on our Django application. User can click the Classification in the web page so that the data calculated Accuracy and macro avg, weighted avg based on the algorithms. User can display the ml results. user can also display the prediction results.

Admin:

Admin can login with his login details. Admin can activate the registered users. Once he activates then only the user can login into our system. Admin can view the overall data in the browser. After that admin can logout.

9.SOURCE CODE:

SAMPLE CODE:

User side views:

```
from django.shortcuts import render, HttpResponse
from django.contrib import messages
from .forms import UserRegistrationForm
from .models import UserRegistrationModel
from django.conf import settings
import numpy as np
import pandas as pd
import os
# Create your views here.
def UserRegisterActions(request):
    if request.method == 'POST':
        # UserRegistrationModel.objects.all().delete()
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            print('Data is Valid')
            skda = form.cleaned_data['skda']
            print("Milli Seconds:", skda)
            form.save()
            messages.success(request, 'You have been successfully registered')
            form = UserRegistrationForm()
            return render(request, 'UserRegistrations.html', {'form': form, 'skda': skda})
        else:
            messages.success(request, 'Email or Mobile Already Existed')
            print("Invalid form")
    else:
        form = UserRegistrationForm()
    return render(request, 'UserRegistrations.html', {'form': form})
def UserLoginCheck(request):
    if request.method == "POST":
        loginid = request.POST.get('loginid')
        pswd = request.POST.get('pswd')
        print("Login ID = ", loginid, ' Password = ', pswd)
        try:
            check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)
            status = check.status
            print('Status is = ', status)
            if status == "activated":
                request.session['id'] = check.id
                request.session['loggeduser'] = check.name
                request.session['loginid'] = loginid
                request.session['email'] = check.email
                print("User id At", check.id, status)
                return render(request, 'users/UserHomePage.html', {})
            else:
```

```
                messages.success(request, 'Your Account Not at activated')
                return render(request, 'UserLogin.html')
        except Exception as e:
            print('Exception is ', str(e))
            pass
        messages.success(request, 'Invalid Login id and password')
    return render(request, 'UserLogin.html', {})
def UserHome(request):
    return render(request, 'users/UserHomePage.html', {})
def DatasetView(request):
    path = os.path.join(settings.MEDIA_ROOT, 'data.csv')
    df = pd.read_csv(path)
    df = df.to_html(index=False)
    return render(request, 'users/viewdataset.html', {'data': df})
def UserClassification(request):
    import pandas as pd
    from .utility import FARFRR_Calc
    rf_report,FAR,FRR,ERR = FARFRR_Calc.process_randomForest()
    rf_report = pd.DataFrame(rf_report).transpose()
    rf_report = pd.DataFrame(rf_report)
    return render(request, 'users/results.html',
                  {'rf': rf_report.to_html,'far': FAR,'frr':FRR,'err': ERR})
def UserChangePassword(request):
    if request.method == 'POST':
        cpassword = request.POST.get('cpassword')
        npassword = request.POST.get('npassword')
        cskda = int(request.POST.get('cskda'))
        nskda = int(request.POST.get('nskda'))
        loginid = request.session['loginid']
        data = UserRegistrationModel.objects.get(loginid=loginid, password=cpassword)
        dbMillis = int(data.skda)
        print(f"{cpassword}: {npassword}: {cskda}: {nskda} DB Millis: {dbMillis}")
        if dbMillis - 300 <= cskda <= dbMillis + 300:
            print("True Condition")
            UserRegistrationModel.objects.filter(loginid=loginid,
password=cpassword).update(password=npassword, skda=nskda)
            return render(request, "users/UserChangePassword.html", {'msg': 'Your Password
Successful Update'})
        else:
            print("False Condition")
            return render(request, "users/UserChangePassword.html", {'msg': 'Your Keystroke doesnot
Match'})
    else:
        return render(request, "users/UserChangePassword.html", {})
```

base.html:

```
{%load static%}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
```

```
    <meta name="author" content="webthemez">
    <title>Static Keystroke Dynamic Authentication (SKDA) Model to Authenticate User during
Password Change</title>
    <!-- core CSS -->
    <link href="{%static 'css/bootstrap.min.css'%}" rel="stylesheet">
    <link href="{%static 'css/font-awesome.min.css'%}" rel="stylesheet">
    <link href="{%static 'css/animate.min.css'%}" rel="stylesheet">
    <link href="{%static 'css/prettyPhoto.css'%}" rel="stylesheet">
    <link href="{%static 'css/styles.css'%}" rel="stylesheet">
    <link rel="stylesheet" type="text/css" href="{%static 'css/slider-style.css'%}"/>
    <link rel="stylesheet" type="text/css" href="{%static 'css/slider-custom.css'%}"/>
    <script type="text/javascript" src="{%static 'js/modernizr.custom.79639.js'%}"></script>
    <!--[if lt IE 9]>
    <script src="{%static 'js/html5shiv.js'%}"></script>
    <script src="{%static 'js/respond.min.js'%}"></script>
    <![endif]-->
    <link rel="shortcut icon" href="{%static 'images/ico/favicon.ico'%}">
</head>
<body id="home">
<header id="header">
    <nav id="main-nav" class="navbar navbar-default navbar-fixed-top" role="banner">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" href="{% url 'index' %}">
                    <h3>Keystroke Dynamic Authentication</h3>
                </a>
            </div>
            <div class="collapse navbar-collapse navbar-right">
                <ul class="nav navbar-nav">
                    <li class="scroll"><a href="{% url 'index' %}">Home</a></li>
                    <li class="scroll"><a href="{% url 'UserLogin' %}">User</a></li>
                    <li class="scroll"><a href="{% url 'AdminLogin' %}">Admin</a></li>
                    <li class="scroll"><a href="{% url 'UserRegister' %}">Register</a></li>
                </ul>
            </div>
        </div><!--/.container-->
    </nav><!--/nav-->
</header><!--/header-->
<section id="about">
    <div class="container">
        <div class="section-header">
            <h2 class="section-title wow fadeInDown">
                Static Keystroke Dynamic Authentication (SKDA) Model to Authenticate User during
Password Change
            </h2>
```

```
            <p class="wow fadeInDown">
               Keystroke dynamics is considered as a supporting
               factor of authentication. Especially in the static keystroke dynamics,
               the user is identified by using the timing featured, which
               is captured while the user enters the login ID and password.
               To achieve this, the user profile needs to be created with timing
               features. However, in a scenario like a change of password where
               nearly no keystroke timing data is available, non-conventional
               features may be helpful. This article focuses on using nonconventional
               features such as NumLock key, Shift key, CapsLock
               key, etc for identifying users during a change of password. The
               paper also details how to capture the non-conventional features
               in static keystroke dynamics and build a model that can be used
               in the change of password.
            </p>
         </div>
         <div class="row">
            <div class="col-sm-6 wow fadeInLeft">
               <img class="img-responsive" src="{%static 'images/about.png'%}" alt="">
            </div>
            {%block contents%}
            {%endblock%}
         </div>
      </div>
   </section><!--/#about-->
   <footer id="footer">
      <div class="container">
         <div class="row">
            <div class="col-sm-6">
               &copy; 2024 Alex Corporations. Template by <a target="_blank" href="#"
                                                             title="Free Bootstrap Themes and HTML
   Templates">Alex</a>
            </div>
         </div>
      </div>
   </footer><!--/#footer-->
   <script src="{%static 'js/jquery.js'%}"></script>
   <script src="{%static 'js/bootstrap.min.js'%}"></script>
   <script src="{%static 'js/mousescroll.js'%}"></script>
   <script src="{%static 'js/smoothscroll.js'%}"></script>
   <script src="{%static 'js/jquery.prettyPhoto.js'%}"></script>
   <script src="{%static 'js/jquery.isotope.min.js'%}"></script>
   <script src="{%static 'js/jquery.inview.min.js'%}"></script>
   <script src="{%static 'js/wow.min.js'%}"></script>
   <script type="text/javascript" src="{%static 'js/jquery.ba-cond.min.js'%}"></script>
   <script type="text/javascript" src="{%static 'js/jquery.slitslider.js'%}"></script>
   <script type="text/javascript" src="{%static 'js/slitslider-custom.js'%}"></script>
   <script src="{%static 'js/custom-scripts.js'%}"></script>
   </body>
   </html>
```

Index.html:

```
{%extends 'base.html'%}
{% load static %}
{%block contents%}
<div class="col-sm-6 wow fadeInRight">
        <h3 class="column-title">KEYSTROKE DYNAMICS SYSTEM AND
FEATURES</h3>
        <p>The study of a number of key events and the intervals
          between them is known as the keystroke dynamics. In order to
          capture timing features of keystroke, KeyPress and KeyRelease
          are two most important extracted features.</p>
        <p>In a keystroke dynamics system, users' keystroke features
          are first captured to create user profile for him/her. The
          keystroke dynamic system then analyses users' typing patterns
          and creates user profile. From the user's typing rhythm different
          timing features are collected.</p>
        <p>The system described in the earlier section in useful while
          verifying the password. But when the user changes the
          password, the model derived from the training of different
          password may not work until a new model is developed for the
          new password. The trained timing features will be helpful only
          in verifying the user through their typing behaviour of specific
          words. Entering login and password being Static keystroke
          dynamics additional typing information cannot be captured
          and their is a dependency on the model developed for the
          specific password.</p>
        <ul class="listarrow">
          <li><i class="fa fa-angle-double-right"></i>Hold time is a distance between KeyPress
and KeyRelease
            timing
          </li>
          <li><i class="fa fa-angle-double-right"></i>UP-Up time is the distance between two
consecutive
            KeyReleases
          </li>
          <li><i class="fa fa-angle-double-right"></i>Up-Down time is the distance between a
KeyRelease and
            the next KeyPress
          </li>
          <li><i class="fa fa-angle-double-right"></i>Down- Down is the time between two
consecutive Key-
            Presses
          </li>
          <li><i class="fa fa-angle-double-right"></i>Down-Up time The time difference
between a KeyPress
            and the next KeyRelease is known as down-up time
          </li>
        </ul>
      </div>
{%endblock%}
```

Admin side views:

```
from django.shortcuts import render, HttpResponse
from django.contrib import messages
from users.models import UserRegistrationModel

# Create your views here.
def AdminLoginCheck(request):
    if request.method == 'POST':
        usrid = request.POST.get('loginid')
        pswd = request.POST.get('pswd')
        print("User ID is = ", usrid)
        if usrid == 'admin' and pswd == 'admin':
            return render(request, 'admins/AdminHome.html')
        else:
            messages.success(request, 'Please Check Your Login Details')
    return render(request, 'AdminLogin.html', {})
def AdminHome(request):
    return render(request, 'admins/AdminHome.html')
def RegisterUsersView(request):
    data = UserRegistrationModel.objects.all()
    return render(request,'admins/viewregisterusers.html',{'data':data})
def ActivaUsers(request):
    if request.method == 'GET':
        id = request.GET.get('uid')
        status = 'activated'
        print("PID = ", id, status)
        UserRegistrationModel.objects.filter(id=id).update(status=status)
        data = UserRegistrationModel.objects.all()
        return render(request,'admins/viewregisterusers.html',{'data':data})
def AdminViewResults(request):
    import pandas as pd
    from users.utility import FARFRR_Calc
    rf_report, FAR, FRR, ERR = FARFRR_Calc.process_randomForest()
    rf_report = pd.DataFrame(rf_report).transpose()
    rf_report = pd.DataFrame(rf_report)
    return render(request, 'admins/results.html',
                  {'rf': rf_report.to_html, 'far': FAR, 'frr': FRR, 'err': ERR})
```

10. SYSTEM TEST

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

TYPES OF TESTS

Unit testing

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

Integration testing

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

Functional test

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

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

System Test

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

White Box Testing

White Box Testing is a testing in which 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.

Black Box Testing

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

Unit Testing

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

Test strategy and approach

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

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.
- The entry screen, messages and responses must not be delayed.

Integration Testing

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

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

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

Acceptance Testing

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

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

11.SCREENSHOTS

Home page:

Fig.1

User register:

Fig.2

Admin login:

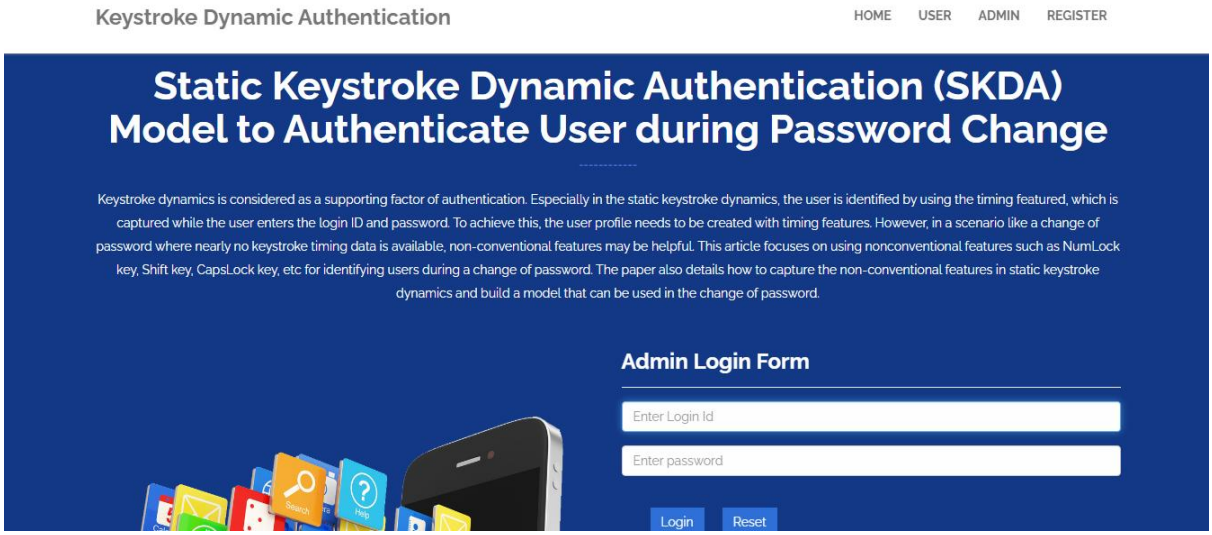


Fig.3

Admin home page:

Fig.4

User details:

Fig.5

Parameters:

Fig.6

User login:

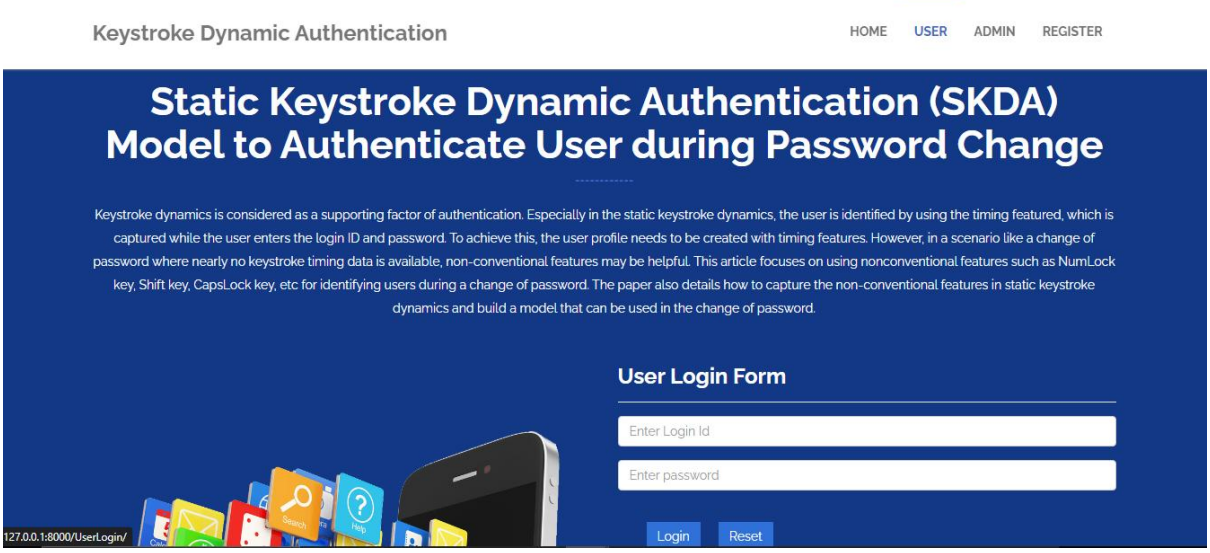
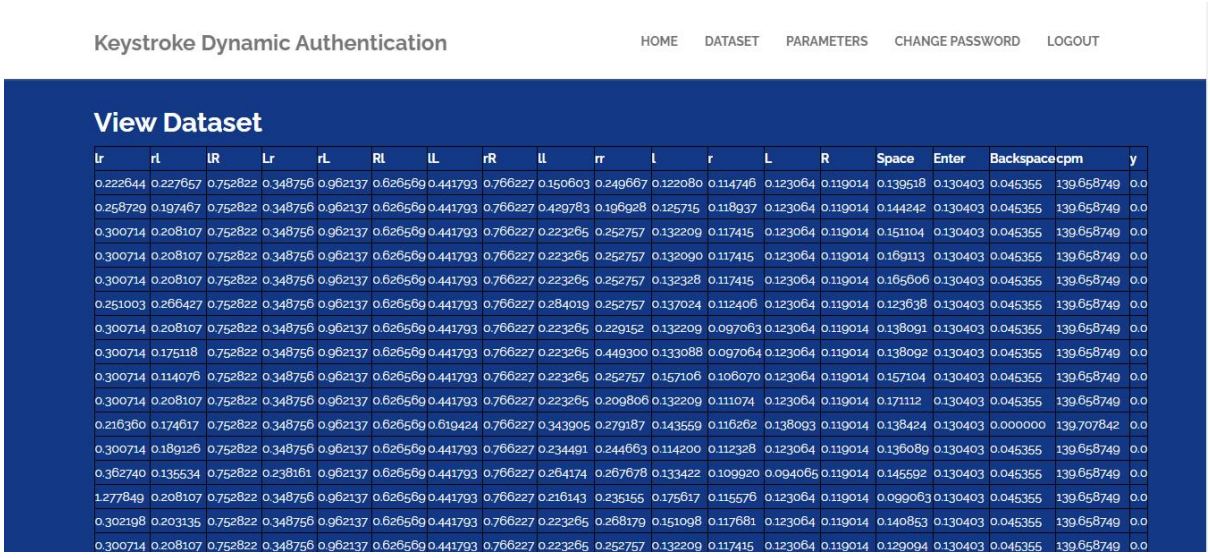


Fig.7

User home:

Fig.8

Dataset:



lr	rl	lR	Lr	rL	RL	LL	rR	ll	rr	l	r	L	R	Space	Enter	Backspace	cpm	y
0.222644	0.227657	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.150603	0.249667	0.122080	0.114746	0.123064	0.119014	0.139518	0.130403	0.045355	139.658749	0.0
0.258729	0.197467	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.429783	0.196928	0.125715	0.118937	0.123064	0.119014	0.144242	0.130403	0.045355	139.658749	0.0
0.300714	0.208107	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.252757	0.132209	0.117415	0.123064	0.119014	0.151104	0.130403	0.045355	139.658749	0.0
0.300714	0.208107	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.252757	0.132090	0.117415	0.123064	0.119014	0.169113	0.130403	0.045355	139.658749	0.0
0.300714	0.208107	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.252757	0.132328	0.117415	0.123064	0.119014	0.165606	0.130403	0.045355	139.658749	0.0
0.251003	0.266427	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.284019	0.252757	0.137024	0.112406	0.123064	0.119014	0.123638	0.130403	0.045355	139.658749	0.0
0.300714	0.208107	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.229152	0.132209	0.097063	0.123064	0.119014	0.138091	0.130403	0.045355	139.658749	0.0
0.300714	0.175118	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.449300	0.133088	0.097064	0.123064	0.119014	0.138092	0.130403	0.045355	139.658749	0.0
0.300714	0.114076	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.252757	0.157106	0.106070	0.123064	0.119014	0.157104	0.130403	0.045355	139.658749	0.0
0.300714	0.208107	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.209806	0.132209	0.111074	0.123064	0.119014	0.171112	0.130403	0.045355	139.658749	0.0
0.216360	0.174617	0.752822	0.348756	0.962137	0.626569	0.619424	0.766227	0.343905	0.279187	0.143559	0.116262	0.138093	0.119014	0.138424	0.130403	0.000000	139.707842	0.0
0.300714	0.189126	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.234491	0.244663	0.114200	0.112328	0.123064	0.119014	0.136089	0.130403	0.045355	139.658749	0.0
0.362740	0.135534	0.752822	0.238161	0.962137	0.626569	0.441793	0.766227	0.264174	0.267678	0.133422	0.109920	0.094065	0.119014	0.145592	0.130403	0.045355	139.658749	0.0
1.277849	0.208107	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.216143	0.235155	0.175617	0.115576	0.123064	0.119014	0.099063	0.130403	0.045355	139.658749	0.0
0.302198	0.203135	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.268179	0.151098	0.117681	0.123064	0.119014	0.140853	0.130403	0.045355	139.658749	0.0
0.300714	0.208107	0.752822	0.348756	0.962137	0.626569	0.441793	0.766227	0.223265	0.252757	0.132209	0.117415	0.123064	0.119014	0.129094	0.130403	0.045355	139.658749	0.0

Fig.9

Change Password:

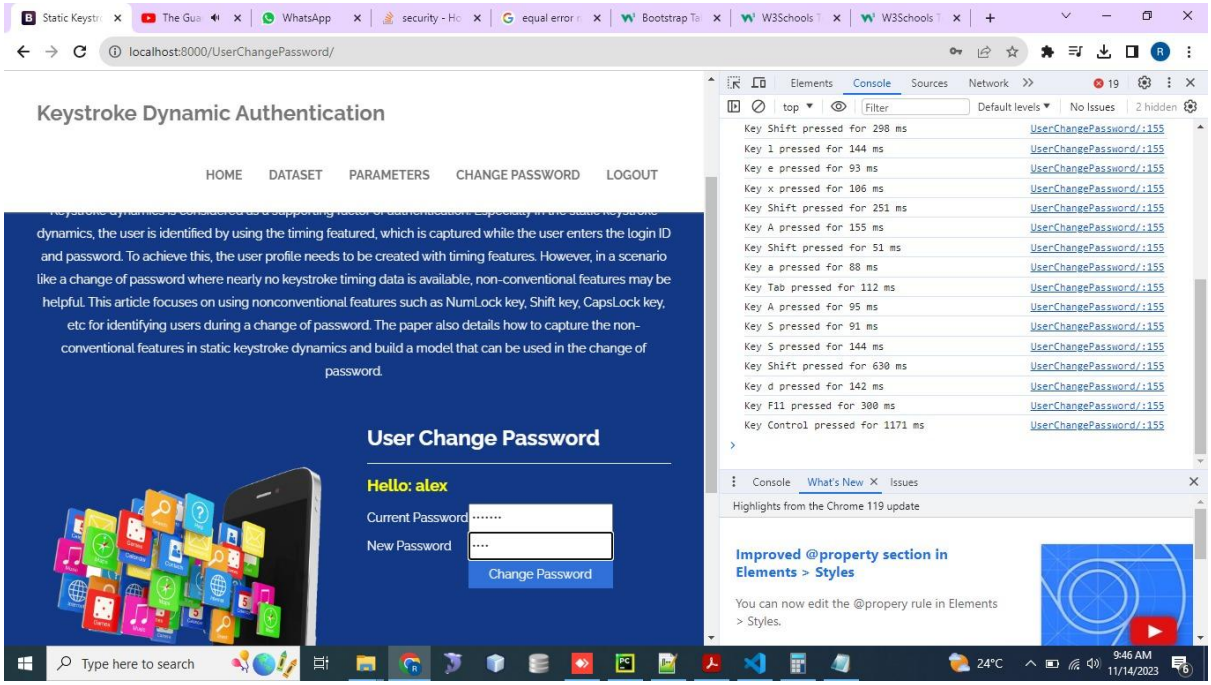


Fig.10

12. INPUT DESIGN AND OUTPUT DESIGN

12.1 INPUT DESIGN

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

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

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

12.2 OUTPUT DESIGN

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

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

13. CONCLUSION

In static keystroke dynamics, the user is authenticated based on the model generated from the timing features captured while entering the login and password. However, during the change of password, no data related to key press and release for the new password is available. This results in, the model available for static keystroke dynamics for the new password of the user. In this paper the use of non-conventional features for change of password is proposed and demonstrated. The paper also discusses the policies which can be used to capture the non-conventional features such as usage of Caps Lock key, Num Lock key, Shift keys and key release pattern during use of shift keys. The SKDA model uses non-conventional features with timing features for user authentication. The notion was tested through an experiment which was carried out and was tested on 155 participants, and have achieved FAR of about 0.02% and FRR of about 0.3%.

Future work includes working on various types of keyboards and identifying keystroke dynamics with respect to different devices.

14. FUTURE ENHANCEMENT

The future enhancements for the Static Keystroke Dynamic Authentication (SKDA) model involve a comprehensive strategy to elevate security, usability, and adaptability. By integrating additional behavioral biometrics and leveraging machine learning for continuous refinement, the model aims to keep pace with evolving user behaviors. Adaptive authentication policies, exploration of multi-factor authentication, and real-time anomaly detection contribute to a dynamic and responsive authentication system.

Enhancements also include user-focused initiatives such as improved education on keystroke dynamics, customization options for users, and considerations for accessibility. Technical advancements involve extending compatibility to mobile devices, exploring blockchain for heightened data security, and integrating threat intelligence feeds for real-time threat awareness.

Furthermore, the model anticipates future regulatory landscapes by incorporating features for monitoring and ensuring compliance with data privacy and security regulations. Collectively, these enhancements position the SKDA model as a robust, user-friendly, and forward-looking solution, addressing both current and emerging challenges in user authentication during password changes.

15. BIBLOGRAPHY

[1] R. Joyce and G. Gupta, “Identity authentication based on keystroke latencies,” in Communications of the ACM, 33(2), 168-176, 1990.

[2] S. Mondal and P. Bours, “Continuous authentication in a real world settings,” in Advances in Pattern Recognition, Eighth International Conference, pp. 1-6, 2015.

[3] F. Monrose and A. D. Rubin, “Keystroke dynamics as a biometric for authentication,” in Future Generation computer systems, 16(4), 351-359, Elsevier Science, 2000.

[4] R. Abinaya and A. Sigappi, “Biometric identification of a genuine user/imposter from keystroke dynamics dataset,” in International Journal of Chemtech Research, Vol.11 No.08, pp 147-160, 2018.

[5] A. Andrey, Vyazigin, Y. Nadezhda, Tupikina, and V. E. Sypin, “Software tool for determining of the keystroke dynamics parameters of personal computer user,” in International Conference on Micro/Nanotechnologies and electron devices EDM,978-1-7281-1753-9/19/$31.00, IEEE, 2019.

[6] P. H. Pisani and A. C. Lorena, “A systematic review on keystroke dynamics,” in Journal of the Brazilian Computer Society, 2013.

[7] M. L. Ali, J. V. Monaco, C. C. Tappert, and M. Qiu, “Keystroke bio- metric systems for user authentication,” in Journal of Signal Processing Systems, DOI: 10.1007/s11265-016-1114-9, Springer, 2016.

[8] K. Shekhawat and D. P. Bhatt, “Recent advances and applications of keystroke dynamics,” in International Conference on Computational Intelligence and Knowledge Economy (ICCIKE),978-1-7281- 3778-0/19/$31.00, IEEE, 2019.

[9] D. R. Gentner et al., “A glossary of terms including a classification of typing errors,” in in Cognitive aspects of skilled typewriting, ed: Springer, pp. 39-43, 1983.

[10] J. Roth, X. Liu, A. Ross, and D. Metaxas, “Investigating the discriminative power of keystroke sound,” in IEEE Transactions on Information Forensics and Security, vol. 10, pp. 333-345, 2015.