

# **PASSWORD MANAGER WITH MULTI- FACTOR AUTHENTICATION**

A Major Project thesis submitted to the **JAWAHARLAL NEHRU  
TECHNOLOGICAL UNIVERSITY** in partial fulfilment of the  
requirement for the award of the degree of  
**BACHELOR OF TECHNOLOGY**

**In  
Cyber Security**

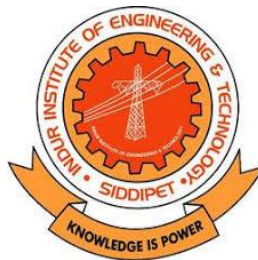
Submitted by

<b>B. Vamshi</b>	<b>(20D31A6202)</b>
<b>B. Sandeep</b>	<b>(20D31A6201)</b>
<b>P. Dhanesh</b>	<b>(20D31A6210)</b>
<b>M. A Kabir</b>	<b>(20D31A6207)</b>

Under the Guidance of

**Mr. RITESH THAKUR**

**Assoc. Professor, Dept. of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
INDUR INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(Affiliated to J.N.T.U.H, Hyderabad)**

**Ponnala (Vil), Siddipet (Dist.), Telangana State – 502 277.**

**December, 2023**

Date:     /     /2023

**CERTIFICATE**

This is to certify that the thesis ‘**A COMPARATIVE STUDY ON  
PASSWORD MANAGER WITH MULTI-FACTOR  
AUTHENTICATION**’ being submitted by

**B. Vamshi**                      **(20D31A6202)**

**B. Sandeep**                    **(20D31A6201)**

**P. Dhanesh**                   **(20D31A6210)**

**M. A Kabir**                   **(20D31A6207)**

In partial fulfilment for the award of “**BACHELOR OF TECHNOLOGY**” in the Department of “**COMPUTER SCIENCE & ENGINEERING**”. (**B. Tech**) to the “**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY**” is a record of bonafide Major Project Work carried out by them under our guidance and supervision.

**The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.**

**Mr. RITESH THAKUR**

**PROJECT GUIDE**

**Assoc. Professor, Dept. of CSE**

**DR. T. BENARJI**

**HOD**

**Assoc. Prof &**

**Head Dept. of CSE**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We are thankful to **Mr. Ritesh Thakur, Project Guide, Assoc. Prof., Dept. of CSE** who guided us a lot by his favourable suggestions to complete our project. He is the research-oriented personality with higher end technical exposure.

We are thankful to **Dr. T. Benarji, Head, Dept. of CSE, Indur Institute of Engineering & Technology**, for extending his help in the department academic activities in the course duration. He the personality of dynamic, enthusiastic in the academic activities.

We extend our thanks to **Dr. V.P. Raju Principal, Indur Institute of Engineering & Technology, Siddipet** for extending his help throughout the duration of this project.

We sincerely acknowledge to all the faculty of the Dept. of CSE for their motivation during my B. Tech course.

We would like to say thanks to all my friends and family members for their timely help and encouragement.

<b>B. Vamshi</b>	<b>(20D31A6202)</b>
<b>B. Sandeep</b>	<b>(20D31A6201)</b>
<b>P. Dhanesh</b>	<b>(20D31A6210)</b>
<b>M. A Kabir</b>	<b>(20D31A6207)</b>

## **DECLARATION**

We hereby declare that the project entitled ‘**A COMPARATIVE STUDY ON PASSWORD MANAGER WITH MULTI-FACTOR AUTHENTICATION**’ submitted in the partial fulfilment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering. This dissertation is our original work, and the project has not formed the basis for the award of any degree, associate, fellowship or any other similar titles and no part of it has been published or sent for the publication at the time of submission.

BY

<b>B. Vamshi</b>	<b>(20D31A6202)</b>
<b>B. Sandeep</b>	<b>(20D31A6201)</b>
<b>P. Dhanesh</b>	<b>(20D31A6210)</b>
<b>M. A Kabir</b>	<b>(20D31A6207)</b>

## **ABSTRACT**

Data breach is a serious issue as it leaks the personal information of more than billions of users and their privacy is compromised. More than 77% of organizations do not have a Cyber Security Incident Response plan. So, it is necessary to be informed of network security and ways to store and generate secured passwords. Having different and random passwords for ones digital accounts can exponentially increase the security of user's data. The goal of this project is to build a password manager which can securely store and encrypt passwords and other data. The multifactor authentication will provide increased security to validate the user into password management system. Multifactor authentication system includes physical security key and graphical password authentication.

# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Literature Survey	2
<b>2</b>	<b>SYSTEM ANALYSIS</b>	<b>4</b>
2.1	Existing System	4
2.1.1	Disadvantages of Existing System	4
2.2	Proposed System	4
2.2.1	Advantages of Proposed System	5
2.3	Feasibility Study	5
2.3.1	Economical Feasibility	5
2.3.2	Technical Feasibility	6
2.3.3	Social Feasibility	6
2.4	Requirement Analysis	6
2.5	Requirement Specification	6
2.6	System Specification	7
2.6.1	Hardware Requirement	7
2.6.2	Software Requirement	7
2.7	System Environment	8
2.7.1	Python	8
2.7.2	Django	19
<b>3</b>	<b>SYSTEM DESIGN</b>	<b>28</b>
3.1	Data Flow Diagram	28
3.2	UML Diagrams	28
3.2.1	Use case Diagram	29

3.3.2	Class Diagram	30
3.3.3	Sequence Diagram	31
3.3.4	Activity Diagram	32
<b>4</b>	<b>IMPLEMENTATION</b>	<b>33</b>
<b>4.1</b>	<b>Modules</b>	<b>33</b>
4.1.1	User	33
4.1.2	Admin	33
<b>4.2</b>	<b>Input and Output Design</b>	<b>33</b>
4.2.1	Input Design	33
4.2.2	Output Design	34
<b>5</b>	<b>RESULTS</b>	<b>36</b>
<b>6</b>	<b>SYSTEM TESTING</b>	<b>41</b>
<b>6.1</b>	<b>Unit Testing</b>	<b>41</b>
<b>6.2</b>	<b>Integration Testing</b>	<b>41</b>
<b>6.3</b>	<b>Functional Testing</b>	<b>42</b>
<b>6.4</b>	<b>System Testing</b>	<b>42</b>
<b>6.5</b>	<b>White Box Testing</b>	<b>42</b>
<b>6.6</b>	<b>Black Box Testing</b>	<b>42</b>
<b>6.7</b>	<b>Test Strategy and Approaches</b>	<b>43</b>
6.7.1	Unit Testing	43
6.7.2	Integration Testing	43
6.7.3	Acceptance Testing	44
<b>7</b>	<b>CONCLUSIONS</b>	<b>45</b>
<b>7.1</b>	<b>Conclusion</b>	<b>45</b>
<b>7.2</b>	<b>Future Improvement</b>	<b>45</b>

<b>8</b>	<b>BIBLIOGRAPHY</b>	<b>46</b>
<b>9</b>	<b>APPENDICES</b>	<b>49</b>



## **LIST OF FIGURES**

2.7.2(a)	Django Architecture	20
2.7.2(b)	Model View Template Architecture	20
3.1	Data Flow Diagram	28
3.2.1	Use Case Diagram	29
3.2.2	Class Diagram	30
3.2.3	Sequence Diagram	31
3.2.4	Activity Diagram	32
5.1	Home Page	36
5.2	User register page	36
5.3	Admin login page	50
5.4	Admin home page	50
5.5	View registered page	51
5.6	User Login Page	51
5.7	User home page	52
5.8	Upload page	52
5.9	Download page	53

## **LIST OF TABLES**

6.8	Sample Test Cases	44
-----	-------------------	----

# CHAPTER 1

## INTRODUCTION

Passwords act as the foremost security for one's digital information against unauthorized access. When the password is stronger and secured, it becomes a herculean task for malicious software and hackers to brute force attack. It is mandatory to have strong passwords for all digital accounts [5, 6].

Characteristics of a strong password: It should be more than eight characters. One should not use personal information. Use unique and random passwords. Avoid consecutive keyboard combinations. Include special characters and numbers. Use mixture of uppercase and lowercase letters. Change passwords regularly.

When the user signs up with various digital systems using several different passwords, it becomes very hard to remember them. To overcome this difficulty password manager is used. With password manager one no need to remember all the passwords. The user can generate and store random passwords. [8]

This doesn't mean the password manager is flawless. There are two most common ways to store the password. The first way is to store the passwords in the internal storage of the application. Here the password is not synchronized. So, if the user changes the device, one should manually port the passwords to the new device. All passwords stored on the device are permanently inaccessible and cannot be recovered in the event of theft or loss. The second way is to store the password in an online database like the cloud. Here the passwords are perfectly synchronized and available on multiple devices simultaneously. But the credentials for every device will be exposed if a data breach happens in the online database. With the leaked password, cyber criminals can hack into the users account and steal ones sensitive information, bank account details and digital identity. Having multi-factor authentication to login in, to password manager will provide additional security and makes to difficult for hackers to access the sensitive information.

## 1.1 LITERATURE SURVEY

The users need to enter three criteria first like a phrase, clue, and username. In this, two of the criteria remain the same and the user can change any one of them. From this method, the user needs to remember only two of the criteria, instead of all three of them. Creating distinct passwords completely based on the clue. Whenever the user wants to access the information in the application where one previously used the password by this approach, one should enter the clue one used earlier to identify that application and the system will recreate the password for them. [9] By this approach even if the user selects the exact password once again, it will be different from the previously hashed one. So, it can weaken hash table attacks. For this reason, salt can be used in hashing instead of these different criteria and the user no need to remember additional information other than the master password. [1]

A password manager that ensures the merging of key values of password managers and provides graphical passwords to make a solution for the existing problems of storing passwords and relating passwords with accounts. It uses image cues to remember graphical passwords Instead of remembering actual passwords one by one. These cues help users to remember the passwords in an easier way and to link passwords with different accounts in a better way. By enabling the user to use the same visual cue for several accounts, it also offers password reuse in a secure manner. The password is scrambled differently for each account when a user appends the relevant password cue to several accounts, which encourages secure reuse. It gives better memory, while better managing user's passwords and the link between passwords and accounts. It also improves security by using user habits and they also assist users with current habits like password reuse. [14]

According to several high-level password management results, it is not necessary to store passwords with a high level of entropy in order to retrieve them [4], [10], and [15]. By hashing the (password, domain) pair and registering it as a strong password with the server, PwdHash [10], for instance, converts low entropy-master passwords

to high entropy-store passwords and retrieves passwords. PwdHash converts a user's real password deterministically into a more challenging password, however this change does not provide protection against offline dictionary attacks. However, if any user makes use of easy to recollect password with PwdHash for numerous digital accounts, the data breach of even one web server succeeds in the leak of a password by a dictionary attack and then calculates users all actual passwords which are derived from a password[2, 6, 11, 13, 18].

An MBCCO scheme password manager, the use of a mobile device as a security token and an additional device for decryption is one of the essential aspects of this strategy that makes it vital to note the difficulty in extracting the encrypted data and the key used for decryption. Thus, a secure secret key in the system encrypts the registered login information before sending and storing the encrypted data on the connected device to the PC and erasing it [17, 19, 20]. That is, other from the login credentials, the PC is where the single secret key is kept. The PC does not save the login details. The user chooses the web service from the register on the device each time they attempt to log into a registered web service. The user-selected encrypted login information is then transmitted to the PC, where it is further decrypted using a secret key. The user can then log in the web service. Has an additional reliability security token is used here.[1,2,3]

## CHAPTER 2

# SYSTEM ANALYSIS

### 2.1 EXISTING SYSTEM

Existing Password Managers Typically use a combination of one or more of the following authentication methods:

**Master Password:** This is a strong password that the user must create and remember in order to access their password vault.

**Two-Factor Authentication (2fa):** This adds an additional layer of security by requiring a second factor, such as a code from a mobile app or a fingerprint scan, in addition to the master password.

**Biometric Authentication:** This uses the user's Fingerprints, Facial Recognition, Or Other Biometric data to authenticate them.

Some popular existing password managers That Support MFA Include:

1password

Bitwarden

Dashlane

Keeper

Lastpass

#### 2.1.1 DISADVANTAGES OF EXISTING SYSTEM

1. While some password managers support biometric authentication, it's not universally available across all devices and platforms, limiting its effectiveness.
2. While 2FA adds an extra layer of security, it's not foolproof. Methods like SMS-based authentication can be vulnerable to SIM swapping attacks, and even authenticator apps can be susceptible to phishing or malware.

### 2.2 PROPOSED SYSTEM

There are a number of proposed systems for password management with MFA that aim to improve security and usability. Some of these proposals include:

**FIDO2:** This is a new standard for authentication that is designed to be more secure and easier to use than traditional MFA methods.

**Password less authentication:** This is a system that does not require users to remember passwords at all. Instead, users are authenticated using other methods, such as biometrics or facial recognition.

**Distributed password managers:** These are password managers that store passwords on multiple devices or cloud servers. This can make them more secure and easier to access from anywhere.

### **2.2.1 ADVANTAGES OF PROPOSED SYSTEM:**

1. FIDO2 and password less authentication reduce the reliance on passwords, which are prone to theft and phishing attacks. By leveraging biometrics or other authentication factors, these systems offer stronger protection against unauthorized access.
2. Password less authentication eliminates the need for users to remember complex passwords, enhancing usability and reducing the risk of forgotten passwords or password reuse.

## **2.3 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

### **2.3.1 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.

### **2.3.2 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.

### **2.3.3 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.

## **2.4 REQUIREMENT ANALYSIS**

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

## **2.5 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

### **Operating Systems supported**

1. Windows 10 64 bit OS

### **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: Intel i9
- RAM: 32 GB
- Space on Hard Disk: minimum 1 TB

## **2.6 SYSTEM SPECIFICATION**

System specifications are classified into two types.

### **2.6.1 HARDWARE REQUIREMENTS**

- ❖ **System** : Intel i7
- ❖ **Hard Disk** : 1 TB
- ❖ **Monitor** : 14' Colour Monitor.
- ❖ **Mouse** : Optical Mouse.
- ❖ **Ram** : 8GB.

### **2.6.2 SOFTWARE REQUIREMENTS**

- ❖ **Operating system** : Windows 7.
- ❖ **Coding Language** : Python.
- ❖ **Front-End** : Html. CSS
- ❖ **Designing** : Html, CSS, JavaScript.
- ❖ **Data Base** : MYSQL.



## 2.7 SYSTEM ENVIRONMENTS

### 2.7.1 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](#).

#### 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 –

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

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

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

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
elif    in      while
else    is      with
except 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.

```

#!/user/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 concepts.

### **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>
```

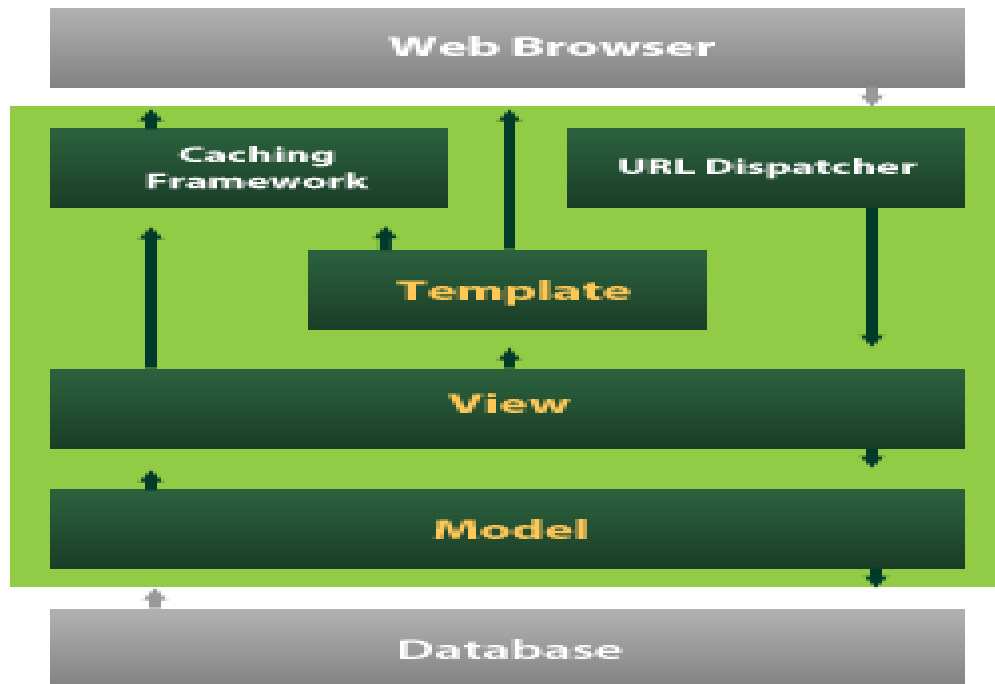
```
    print tup;
```

NameError: name 'tup' is not defined

### 2.7.2 DJANGO

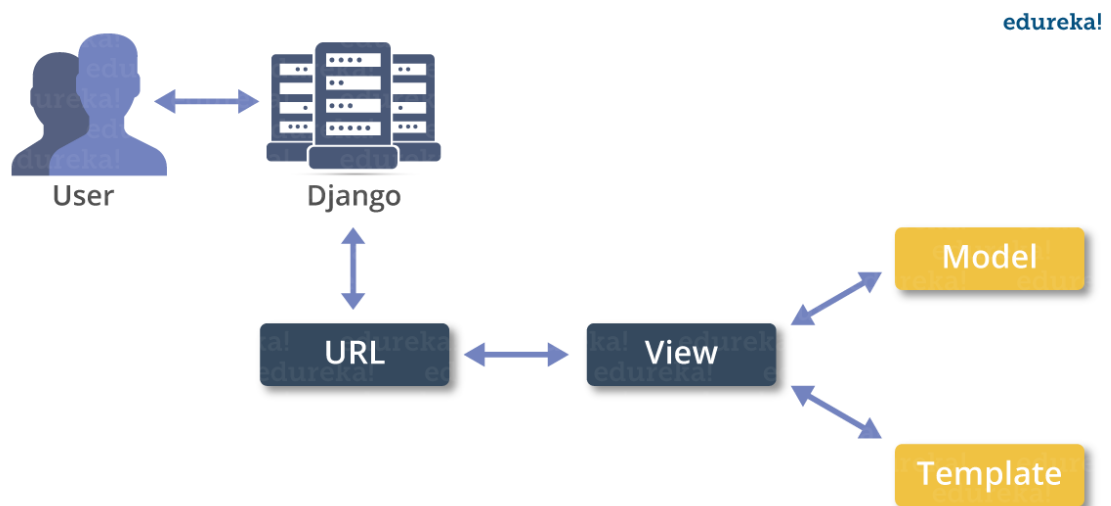
Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

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 2.7.2(a): Django Architecture**

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



**Fig 2.7.2(b): Model View Template Architecture**

### 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 let's 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.

## CHAPTER 3

### SYSTEM DESIGN

#### 3.1 DATA FLOW DIAGRAM

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

#### 3.2 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.

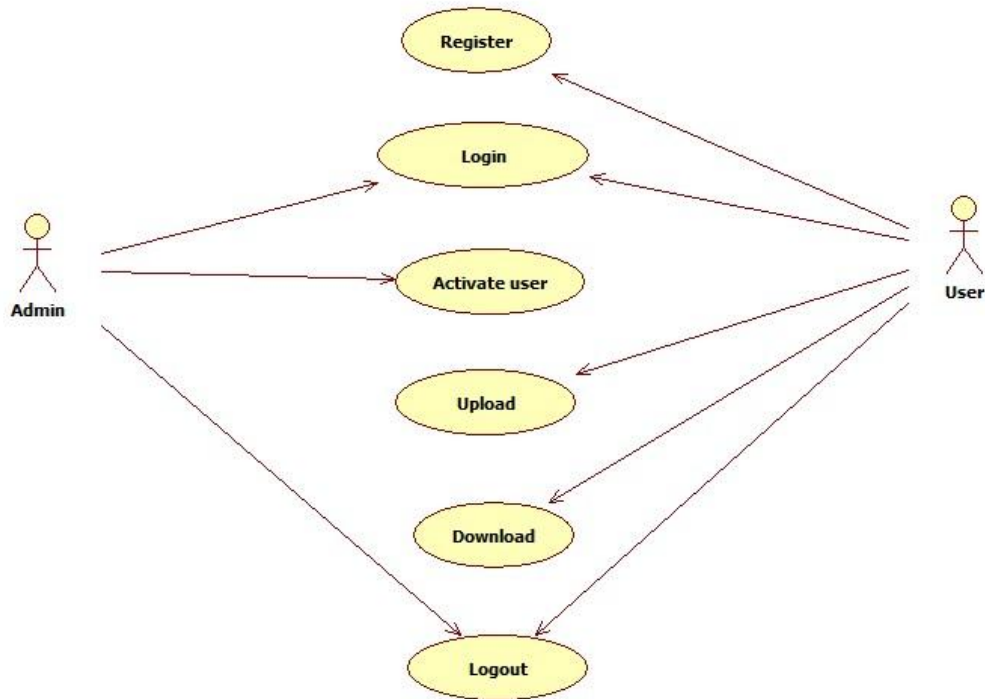
#### **GOALS:**

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.

#### **3.2.1 USE CASE DIAGRAM**

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



**Fig 3.2.1** Use Case Diagram of User and Admin

### 3.2.2 CLASS DIAGRAM

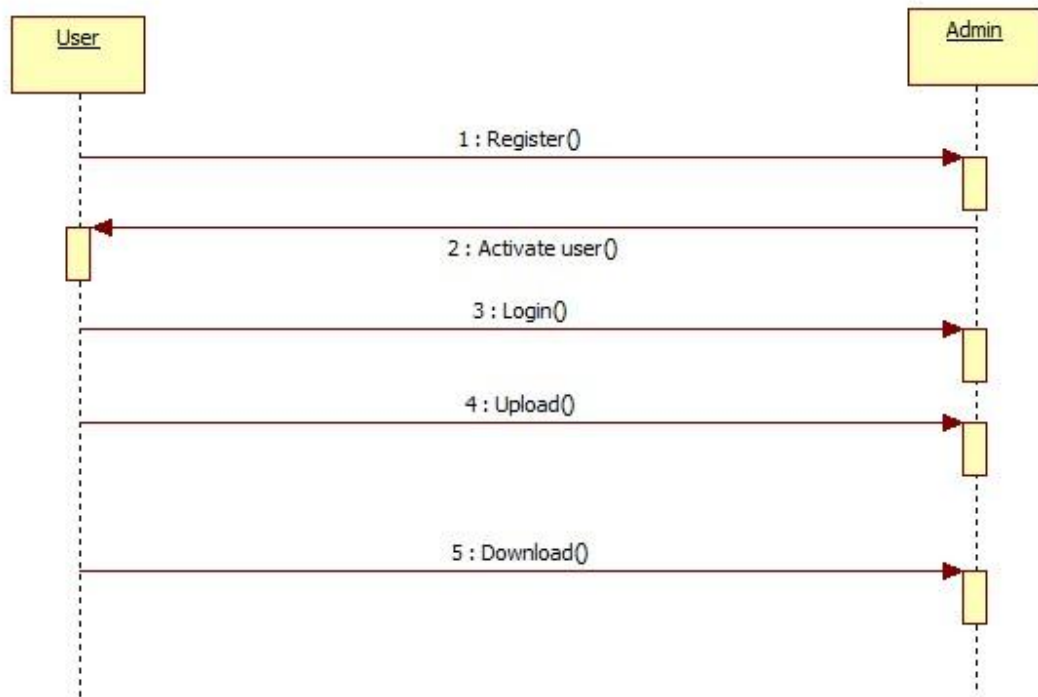
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains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



**Fig 3.2.2** Class Diagram

### 3.2.3 SEQUENCE DIAGRAM

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

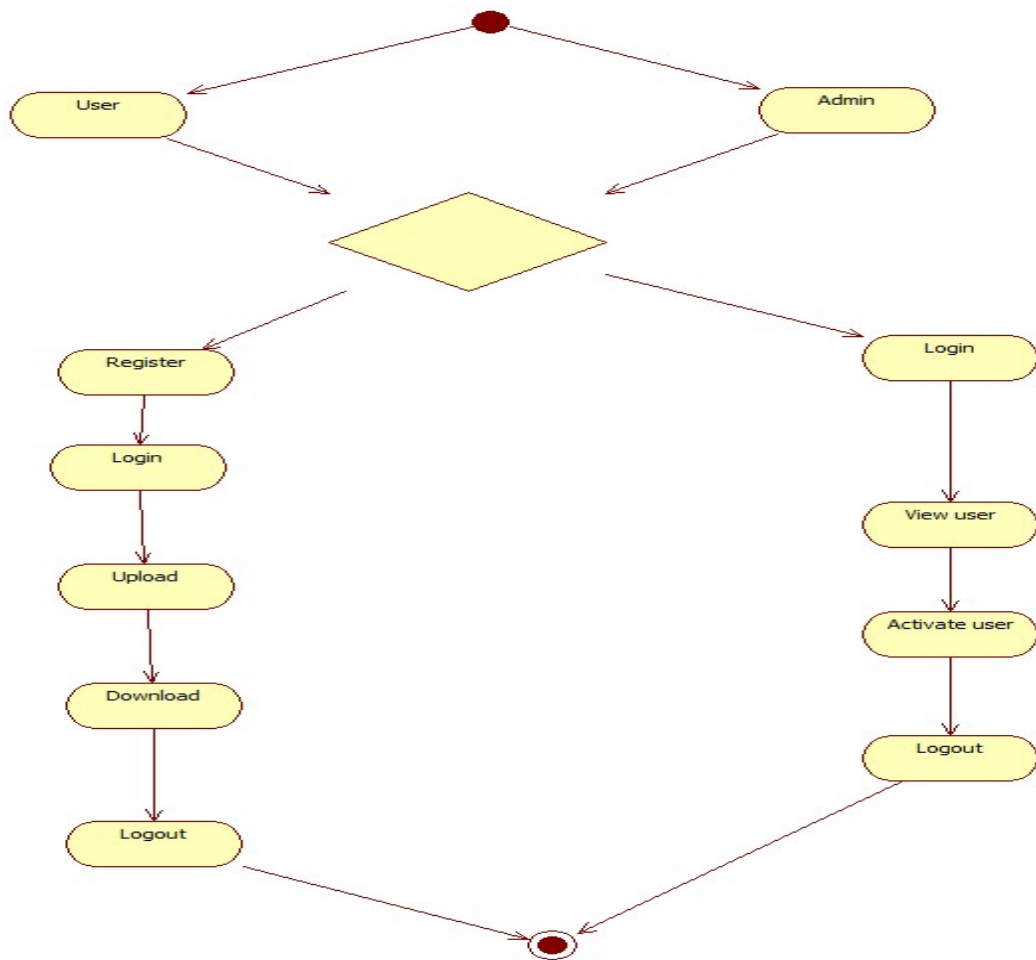


**Fig 3.2.3** Sequence Diagram



### 3.2.4 ACTIVITY DIAGRAM

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



**Fig 3.2.4** Activity Diagram

## **CHAPTER 4**

### **IMPLEMENTATION**

#### **4.1MODULES**

- User
- Admin

##### **4.1.1 User**

The User can register 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 int or float format. Here we took Adacel Technologies Limited dataset for testing purpose. User can also add the new data for existing dataset based on our Django application. User can click the Data Preparations in the web page so that the data cleaning process will be starts. The cleaned data and its required graph will be displayed.

##### **4.1.2 Admin**

Admin can login with his login details. Admin can activate the registered users. Once he activate then only the user can login into our system. Admin can view Users and he can view overall data in the browser and he load the data. Admin can view the training data list and test data list. Admin can load the data and view forecast results.

#### **4.2 INPUT AND OUTPUT DESIGN**

##### **4.2.1 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.

### **4.2.2 OUTPUT DESIGN**

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

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

2. Select methods for presenting information.

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

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

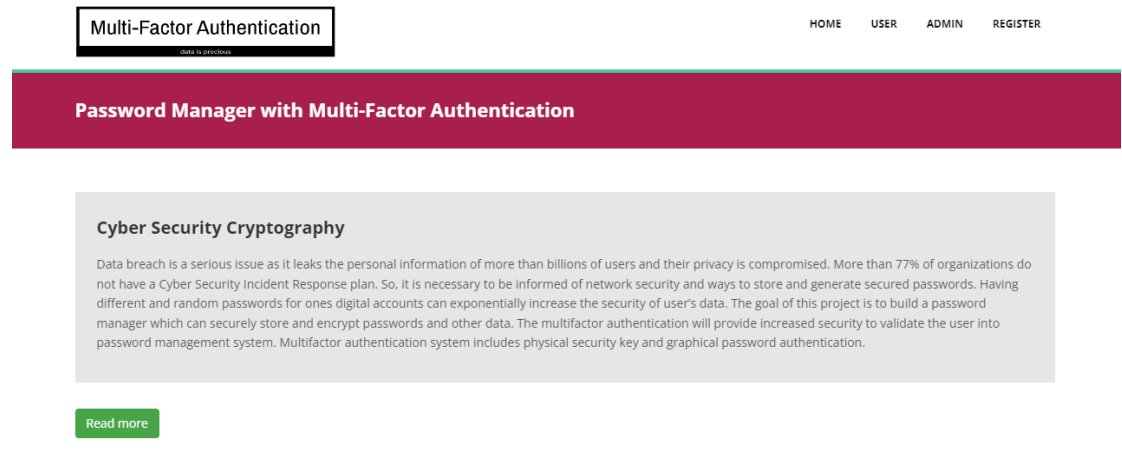
- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

# CHAPTER 5

## RESULTS

### SCREENS

#### 5.1 Home Page



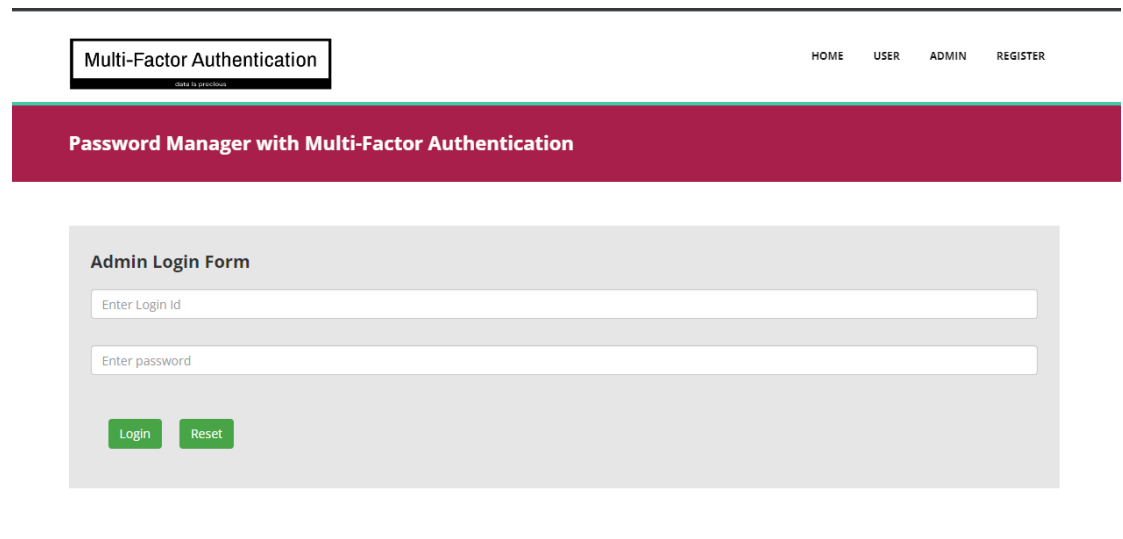
**Fig 5.1** Home Page

#### 5.2 User register page

The screenshot shows the user registration page of the same web application. The top navigation bar and banner are identical to the home page. The main content area features a "User Registration Form" with the heading "Your Master Password is:". The form includes input fields for User Name, Login ID, Password, Mobile, email, Locality, Address, City, and State. A green "Register" button is located at the bottom right of the form.

**Fig 5.2** User register page

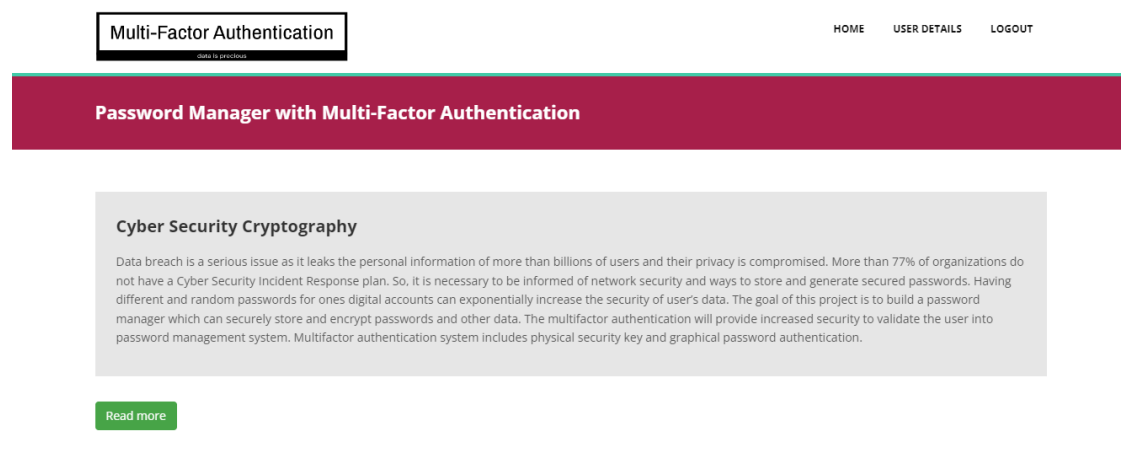
## 5.3 Admin Login Page



The screenshot shows the Admin Login Page. At the top, there is a navigation bar with a logo "Multi-Factor Authentication" and links for HOME, USER, ADMIN, and REGISTER. Below the navigation bar is a red banner with the text "Password Manager with Multi-Factor Authentication". The main content area is a light gray box titled "Admin Login Form". It contains two input fields: "Enter Login Id" and "Enter password". Below the input fields are two green buttons: "Login" and "Reset".

**Fig 5.3** Admin login page

## 5.4 Admin home page



The screenshot shows the Admin home page. At the top, there is a navigation bar with a logo "Multi-Factor Authentication" and links for HOME, USER DETAILS, and LOGOUT. Below the navigation bar is a red banner with the text "Password Manager with Multi-Factor Authentication". The main content area is a light gray box titled "Cyber Security Cryptography". It contains a paragraph of text about data breaches and the need for secure password management. Below the paragraph is a green button labeled "Read more".

**Fig 5.4** Admin home page

## 5.5 View registered page

Multi-Factor Authentication

HOMEUSER DETAILSLOGOUT

Password Manager with Multi-Factor Authentication

View Registered Users

S.No	Name	Login ID	Mobile	Email	Locality	Status	Activate
1	alex	alex	9849098490	lx160cm@gmail.com	Hyderabad	activated	Activated
2	Teja	teja	9878012345	teja@aol.com	Hyderabad	activated	Activated

© Alex Corporations 2024 All right reserved. By Alex Hales

**Fig 5.5** View registered page

## 5.6 User login page

Multi-Factor Authentication

HOMEUSERADMINREGISTER

Password Manager with Multi-Factor Authentication

User Login Form

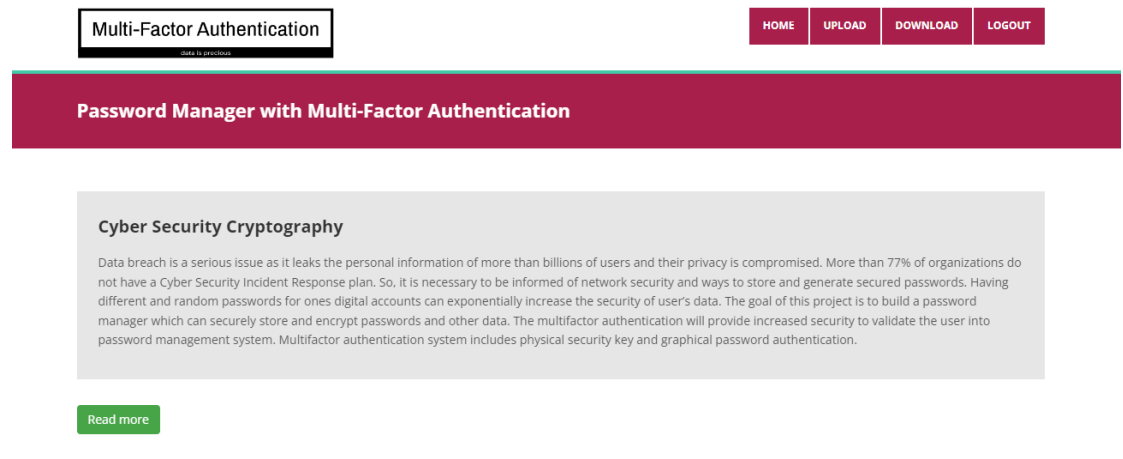
Enter Login Id

Enter password

LoginReset

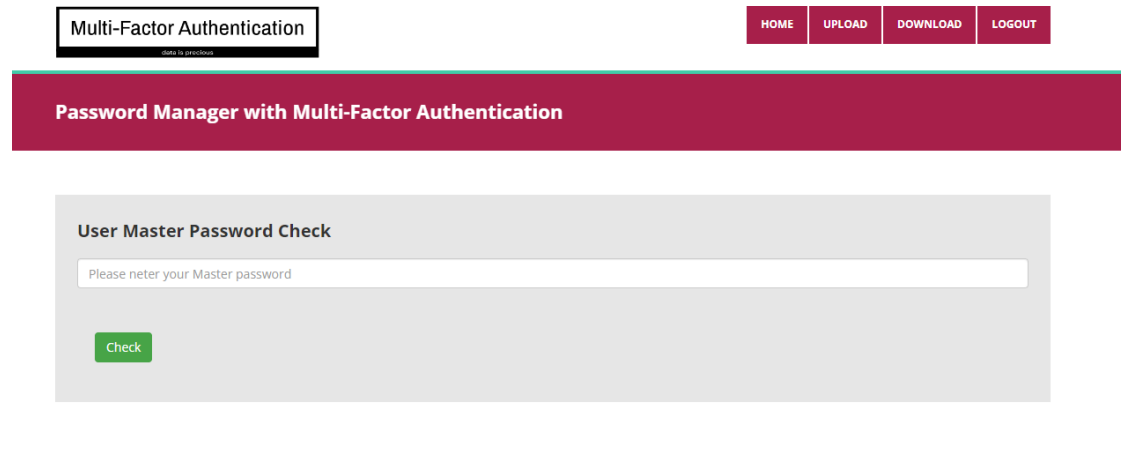
**Fig 5.6** User Login Page

## 5.7 User home page



**Fig 5.7** User home page

## 5.8 Upload page



**Fig 5.8** Upload page



## 5.9 Download page

Multi-Factor Authentication

HOMEUPLOADDOWNLOADLOGOUT

Password Manager with Multi-Factor Authentication

User Master Password Check

Please enter your Master password

Check

© Alex Corporations 2024 All right reserved. By Alex Hales

**Fig 5.9** Download page

## **CHAPTER 6**

### **SYSTEM TESTING**

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 TESTING**

##### **6.1 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.

##### **6.2 Integration testing**

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

## 6.3 Functional testing

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 : 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.

## 6.4 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.

## 6.5 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.

## 6.6 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 test 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.

## **6.7 Test strategy and Approaches**

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

### **6.7.1 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 objectives**

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

#### **Features to be tested**

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

### **6.7.2 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.

### 6.7.3 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.

### 6.8 Sample Test Cases

S.no	Test Case	Excepted Result	Result	Remarks(IF Fails)
1	User Register	If User registration successfully.	Pass	If already user email exist then it fails.
2	User Login	If Username and password is correct then it will getting valid page.	Pass	Un Register Users will not logged in.
3	Upload	Used to upload the password or private key	Pass	Results not true failed
4	Download	Based on the uploaded key used to download	Pass	Result not true if key is wrong
4	User classification	Display reviews with true results	Pass	Results not true failed
5	Admin login	Admin can login with his login credential. If success he get his home page	Pass	Invalid login details will not allowed here
6	Admin can activate the register users	Admin can activate the register user id	Pass	If user id not found then it won't login.

## **CHAPTER 7**

### **CONCLUSION**

#### **7.1 CONCLUSION**

The password manager is the necessary application to protect one's privacy and secure passwords. This approach is suggested to strengthen authentication security in password managers. The drawbacks of this authentication system are there is a possibility of many security bugs as the system is new. In further improvements these problems will be solved.

#### **7.2 FUTURE IMPROVEMENT**

The application design should be user-friendly and simple to use. This system will be available on all major platforms. Since all the passwords are available in a single system it is at most important to penetration test it with different techniques. Additional hashing techniques like Scrypt will be added. The application will be open-source. The security analysts and coders can test the system and find vulnerabilities. By fixing these security bugs, the security of the methodology can be increased.

## CHAPTER 8

### BIBLIOGRAPHY

1. Billa, J. B., Nawar, A., Shakil, M. M. H., & Das, A. K. (2019, June). PassMan: A New Approach of Password Generation and Management without Storing. In 2019 7th International Conference on Smart Computing & Communications (ICSCC) (pp. 1-5). IEEE.
2. Dhanalakshmi. R., & Chellappan. C. (2012). Fraud and Identity Theft Issues. In Strategic and Practical Approaches for Information Security Governance: Technologies and Applied Solutions (pp. 245-260). IGI Global.
3. Fukumitsu, M., Hasegawa, S., Iwazaki, J. Y., Sakai, M., & Takahashi, D. (2016, March). A proposal of a password manager satisfying security and usability by using the secret sharing and a personal server. In 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA) (pp. 661-668). IEEE.
4. Halderman, J. A., Waters, B., & Felten, E. W. (2005, May). A convenient method for securely managing passwords. In Proceedings of the 14th international conference on World Wide Web (pp. 471- 479).
5. Li, Z., He, W., Akhawe, D., & Song, D. (2014). The emperor's new password manager: Security analysis of web-based password managers. In 23rd {USENIX} Security Symposium ({USENIX} Security 14) (pp. 465-479).
6. Mannan, M., & Van Oorschot, P. C. (2007, February). Using a personal device to strengthen password authentication from an untrusted computer. In

International Conference on Financial Cryptography and Data Security (pp. 88-103). Springer, Berlin, Heidelberg.

7. Pearman, S., Zhang, S. A., Bauer, L., Christin, N., & Cranor, L. F. (2019, August). Why people (don't) use password managers effectively. In Fifteenth Symposium On Usable Privacy and Security (SOUPS 2019). USENIX Association, Santa Clara, CA (pp. 319-338).
8. Prakash, A., & Dhanalakshmi. R. (2016). Stride Towards Proposing Multi-Modal Biometric Authentication for Online Exam. *IJ Network Security*, 18(4), 678-687
9. Prakash, A., Krishnaveni. R., & Dhanalakshmi. R. (2020). Continuous user authentication using multimodal biometric traits with optimal feature level fusion. *International Journal of Biomedical Engineering and Technology*, 34(1), 1-19.
10. Ross, B., Jackson, C., Miyake, N., Boneh, D., & Mitchell, J. C. (2005, August). Stronger Password Authentication Using Browser Extensions. In *USENIX Security Symposium* (pp. 17-32).
11. Schaub, F., Walch, M., Könings, B., & Weber, M. (2013, July). Exploring the design space of graphical passwords on smartphones. In *Proceedings of the Ninth Symposium on Usable Privacy and security* (pp. 1-14).
12. Shirvanian, M., Jareckiy, S., Krawczyk, H., & Saxena, N. (2017, June). Sphinx: A password store that perfectly hides passwords from itself. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)* (pp. 1094-



1104). IEEE.

13. Silver, D., Jana, S., Boneh, D., Chen, E., & Jackson, C. (2014). Password managers: Attacks and defenses. In 23rd {USENIX} Security Symposium ({USENIX} Security 14) (pp. 449-464).
14. Stobert, E., & Biddle, R. (2014, September). A password manager that doesn't remember passwords. In Proceedings of the 2014 New Security Paradigms Workshop (pp. 39-52).
15. Yee, K. P., & Sitaker, K. (2006, July). Passpet: convenient password management and phishing protection. In Proceedings of the second symposium on Usable privacy and security (pp. 32-43).\

## CHAPTER 9

### APPENDICES

#### SAMPLE CODE

##### User side view

```
# Create your views here.

from django.shortcuts import render, HttpResponseRedirect, redirect
from django.contrib import messages
from .forms import UserRegistrationForm
from .models import UserRegistrationModel, UserFilesModel
from django.conf import settings
from django.core.files.storage import FileSystemStorage
import numpy as np
import os

# Create your views here.
def UserRegisterActions(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        # UserRegistrationModel.objects.all().delete()
        newPass = "
        print(form.is_valid())
        # if form.is_valid():
        #     obj = form.save(commit=False)
        password = form.cleaned_data['password']
        name = form.cleaned_data['name']
        loginid = form.cleaned_data['loginid']
        mobile = form.cleaned_data['mobile']
        email = form.cleaned_data['email']
        locality = form.cleaned_data['locality']
        address = form.cleaned_data['address']
```

```

city = form.cleaned_data['city']
state = form.cleaned_data['state']
status = form.cleaned_data['status']

mspswd = list(password)
mspswd.reverse()
for idx, x in enumerate(mspswd):
    print(idx + 1, x)
    newPass += f" {(idx + 1) * x}"
print(f'Data is Valid password: {password} Master Password is: {newPass}')
import os, binascii
from backports.pbkdf2 import pbkdf2_hmac
salt = binascii.unhexlify('aef2d3f4d77ac66e9c5a6c3d8f921d1')
passwd = newPass.encode("utf8")
key = pbkdf2_hmac("sha256", passwd, salt, 50000, 32)
key_is = binascii.hexlify(key)
# obj.cleaned_data['masterPassword'] = newPass
# obj.cleaned_data['hashedPass'] = key_is
# obj.save()
masterPassword = newPass
hashedPass = key_is
UserRegistrationModel.objects.create(
    name=name, loginid=loginid, password=password, mobile=mobile,
email=email, locality=locality,
    address=address, city=city, state=state, status=status,
masterPassword=masterPassword,
    hashedPass=hashedPass
)
messages.success(request, 'You have been successfully registered')
form = UserRegistrationForm()
return render(request, 'UserRegistrations.html', {'form': form, 'masterPassword':
newPass})

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 UploadFormData(request):
    if request.method == 'POST':
        mspwd = request.POST.get('pswd')
        loginUser = request.session['loginid']
        try:
            data = UserRegistrationModel.objects.get(loginid=loginUser,
masterPassword=mspwd)
            if data:
                return redirect("UserUploadDataCloud")
            except:
                return render(request, 'users/checkmsrpasword.html', {"msg": 'wrong master
password'})
        else:
            return render(request, 'users/checkmsrpasword.html')

```

```

def UserUploadDataCloud(request):
    if request.method == 'POST':
        image_file = request.FILES['file']
        fs = FileSystemStorage(location="media/files/")
        filename = fs.save(image_file.name, image_file)
        uploaded_file_url = "/media/files/" + filename # fs.url(filename)
        print("Image path ", uploaded_file_url)
        from .utility.Cloud_upload_test import user_input
        # user_input(filename)
        loc = settings.MEDIA_ROOT + '\\' + 'files'
        filepath = os.path.join(loc, filename)
        loginid = request.session['loginid']
        email = request.session['email']
        UserFilesModel.objects.create(username=loginid, email=email,
filename=filename, file=filepath)
        return render(request, 'users/checkmsrpasword.html', {"msg": 'File Uploaded
Success'})
    else:

```

```

return render(request, "users/UploadForm.html", {})

def DownloadFiles(request):
    if request.method == 'POST':
        mspwd = request.POST.get('pswd')
        loginUser = request.session['loginid']
        try:
            data = UserRegistrationModel.objects.get(loginid=loginUser,
masterPassword=mspwd)
            if data:
                return redirect("UserDownloads")
        except:
            return render(request, 'users/checkmsrpasword1.html', {"msg": 'wrong master
password'})
        else:
            return render(request, 'users/checkmsrpasword1.html')

def UserDownloads(request):
    loginid = request.session['loginid']
    data = UserFilesModel.objects.filter(username=loginid)
    response = HttpResponse('text/csv')
    return render(request, "users/viewUploadedFiles.html", {"data": data})

```

### **Base.html:**

```

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Password Manager with Multi-Factor Authentication </title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <meta name="description" content=""/>

```

```

<meta name="author" content="http://webthemez.com"/>
<!-- css -->
<link href="{ %static 'css/bootstrap.min.css'% }" rel="stylesheet"/>
<link href="{ %static 'css/fancybox/jquery.fancybox.css'% }" rel="stylesheet">
<link href="{ %static 'css/jcarousel.css'% }" rel="stylesheet"/>
<link href="{ %static 'css/flexslider.css'% }" rel="stylesheet"/>
<link href="{ %static 'css/style.css'% }" rel="stylesheet"/>

<!-- HTML5 shim, for IE6-8 support of HTML5 elements -->
<!--[if lt IE 9]>
<script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->

</head>
<body>
<div id="wrapper">

<!-- start header -->
<header>
<div class="navbar navbar-default navbar-static-top">
<div class="container">
<div class="navbar-header">
<button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
<span class="icon-bar"></span>
<span class="icon-bar"></span>
<span class="icon-bar"></span>
</button>
<a class="navbar-brand" href="index.html"></a>
</div>
<div class="navbar-collapse collapse">
<ul class="nav navbar-nav">

```

```

<li class=""><a href="{ % url 'index' % }">Home</a></li>
<li class=""><a href="{ % url 'UserLogin' % }">User</a></li>
<li class=""><a href="{ % url 'AdminLogin' % }">Admin</a></li>
<li class=""><a href="{ % url 'UserRegister' % }">Register</a></li>
</ul>
</div>
</div>
</div>
</div>
</header><!-- end header -->
<section id="inner-headline">
<div class="container">
<div class="row">
<div class="col-lg-12">
<h2 class="pageTitle">Password Manager with Multi-Factor
Authentication</h2>
</div>
</div>
</div>
</div>
</section>
{ %block contents% }
{ %endblock% }
<footer>

<div id="sub-footer">
<div class="container">
<div class="row">
<div class="col-lg-6">
<div class="copyright">
<p>
<span>&copy; Alex Corporations 2024 All right reserved. By
</span><a href="#"
target=" blank">Alex
Hales</a>
</p>

```



```

_____/div>
_____/div>

_____/div>
_____/div>
_____/div>
_____/footer>
_____/div>
<a href="#" class="scrollup"><i class="fa fa-angle-up active"></i></a>
<!-- javascript
===== -->
<!-- Placed at the end of the document so the pages load faster -->
<script src="{ %static 'js/jquery.js'% }"></script>
<script src="{ %static 'js/jquery.easing.1.3.js'% }"></script>
<script src="{ %static 'js/bootstrap.min.js'% }"></script>
<script src="{ %static 'js/jquery.fancybox.pack.js'% }"></script>
<script src="{ %static 'js/jquery.fancybox-media.js'% }"></script>
<script src="{ %static 'js/portfolio/jquery.quicksand.js'% }"></script>
<script src="{ %static 'js/portfolio/setting.js'% }"></script>
<script src="{ %static 'js/jquery.flexslider.js'% }"></script>
<script src="{ %static 'js/animate.js'% }"></script>
<script src="{ %static 'js/custom.js'% }"></script>
</body>
</html>

```

### **Admin side views:**

```

from django.shortcuts import render, HttpResponseRedirect
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 RiceLeaf_Classification
    rf_report = RiceLeaf_Classification.process_randomForest()

```

```

dt_report = RiceLeaf_Classification.process_decisionTree()
nb_report = RiceLeaf_Classification.process_naiveBayes()
gb_report = RiceLeaf_Classification.process_gradientBoosting()
rf_report = pd.DataFrame(rf_report).transpose()
rf_report = pd.DataFrame(rf_report)
dt_report = pd.DataFrame(dt_report).transpose()
dt_report = pd.DataFrame(dt_report)
nb_report = pd.DataFrame(nb_report).transpose()
nb_report = pd.DataFrame(nb_report)
gb_report = pd.DataFrame(gb_report).transpose()
gb_report = pd.DataFrame(gb_report)
# report_df.to_csv("rf_report.csv")
return render(request,'admins/ml_reports.html',{'rf': rf_report.to_html,'dt':
dt_report.to_html,'nb':nb_report.to_html,'gb': gb_report.to_html})

```