PROJECT ID:- 288703

An Internship Report on

"Web Development With Python"

Submitted under the Subject

Internship/Project (3180701)

By

Mahida Chintan Jayendrasinh
(180760107023)

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING
In
COMPUTER ENGINEERING

Guided By	Head of the Department
PROF. DRASTI CHAUHAN	**PROF. CHIRAG PATEL**
Assistant Professor	**Assistant Professor**
CoED, SSASIT, Surat	**CoED, SSASIT, Surat**

Shree Swami Atmanand Saraswati Institute of Technology, Surat

GUJARAT TECHNOLOGICAL UNIVERSITY

[MAY 2023]

PROJECT ID:- 288703

CERTIFICATE

This is to certify that an internship work embodied in this report entitled **“WEB DEVELOPMENT WITH PYTHON”** was carried out by **MAHIDA CHINTAN JAYENDRASINH. (180760107023)** studying at **SHREE SWAMI ATMANAND SARASWATI INSTITUTE OF TECHNOLOGY, SURAT (INSTITUTE CODE: 076)** for partial fulfilment of Bachelor of Engineering Degree in **COMPUTER ENGINEERING** to be awarded by Gujarat Technological University. This internship work has been carried out under my guidance and supervision and it is up to my satisfaction.

Date:

Place: SSASIT, Surat

Prof. Drasti Patel
Assistant Professor
Faculty Mentor
CoED, SSASIT, SURAT

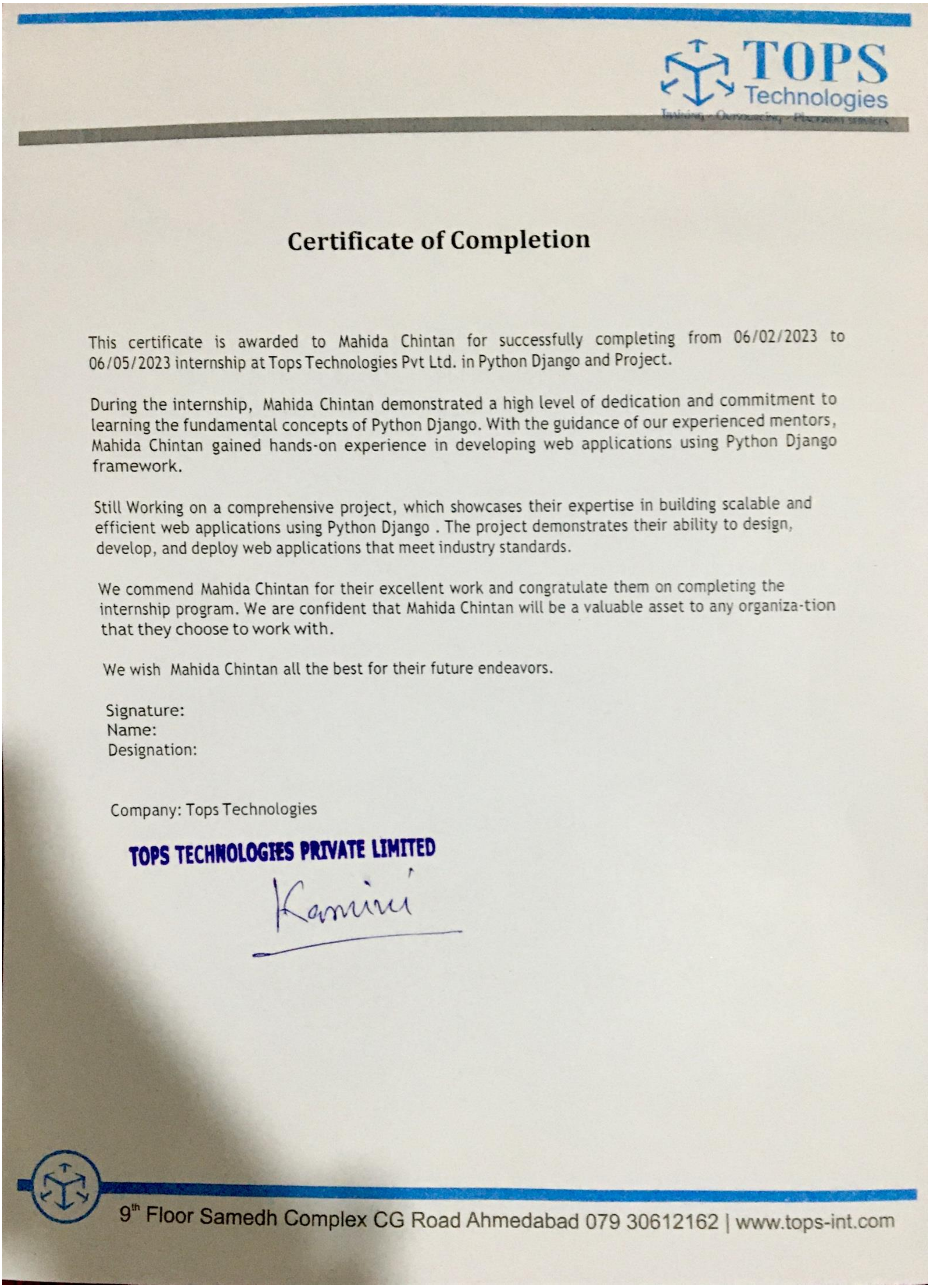
Prof. Chirag Patel
Assistant Professor
Head of the Department
CoED, SSASIT, SURAT

External Examiner Sign

PROJECT ID:- 288703

CERTIFICATE FOR COMPLETION OF ALL ACTIVITIES AT ONLINE PROJECT PORTAL

B.E. SEMESTER VIII, ACADEMIC YEAR 2022-2023



TOPS Technologies

Training - Outsourcing - Placement services

Certificate of Completion

This certificate is awarded to Mahida Chintan for successfully completing from 06/02/2023 to 06/05/2023 internship at Tops Technologies Pvt Ltd. in Python Django and Project.

During the internship, Mahida Chintan demonstrated a high level of dedication and commitment to learning the fundamental concepts of Python Django. With the guidance of our experienced mentors, Mahida Chintan gained hands-on experience in developing web applications using Python Django framework.

Still Working on a comprehensive project, which showcases their expertise in building scalable and efficient web applications using Python Django . The project demonstrates their ability to design, develop, and deploy web applications that meet industry standards.

We commend Mahida Chintan for their excellent work and congratulate them on completing the internship program. We are confident that Mahida Chintan will be a valuable asset to any organiza-tion that they choose to work with.

We wish Mahida Chintan all the best for their future endeavors.

Signature:
Name:
Designation:

Company: Tops Technologies

TOPS TECHNOLOGIES PRIVATE LIMITED

Kamini

9th Floor Samedh Complex CG Road Ahmedabad 079 30612162 | www.tops-int.com

DECLARATION

I hereby certify that I am the sole author of this internship report and that neitherany part nor the whole of the report has been submitted for a degree to any other University or Institution.

I certify that, to the best of my knowledge, the current internship report does notinfringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations or any other material from the work of other people included inmy internship report, published or otherwise, are fully acknowledged in accordance with the standard referencing practices.

Furthermore, to the extent that, I have included copyrighted material that surpasses the boundary of fair dealing within the meaning of the Indian Copyright (Amendment) Act 2012, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in the current report and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of an internship report, including any final revisions, as approved by internship review committee.

I have checked write up of the present report using anti-plagiarism database andit is in allowable limit. Even though later on in case of any complaint pertaining of plagiarism, I am sole responsible for the same and I understand that as per UGC- norms,University can even revoke Bachelor of Engineering Degree conferred to the students submitting this internship report.

Date:

Place: SSASIT, Surat

MAHIDA CHINTAN
(180760107023)

ACKNOWLEDGEMENT

No work is ever the outcome of efforts or ability of single person. Numerous instructors', companions, well-wishers have helped this work specifically or in around out way and made it workable to present it in its present shape. Despite the factthat it is impractical name and thank all separately, I must make unique specify of some of the personalities andrecognize true obligation to them and save appreciationto numerous others in the heart. For that I express my profound and earnest appreciationto my University and my SSASIT, Surat. I express a deep sense of gratitude to **Prof. Drasti J. Chauhan,** my Internship Mentor from **SSASIT, Surat** for his wisdom, knowledge and commitment to the highest standards inspired and motivated me. Without his encouragement and guidance my internshipwork would not have materialized.

I express my sincere thanks to **Dr. Jignesh G. Vaghsia**, ***Principal of*** **SSASIT-*Surat,*** and **Prof. Chirag Patel HOD**, **HOD of Computer engineering, division of SSASIT- Surat** and all the staff Members.

I take the opportunity here to express my sincere gratitude to **"My Gardian, My Parents and Friends"** for their support and guidance during my internship work.

MAHIDA CHINTAN
(180760107023)

Table Of Content

LIST OF FIGURES

PROJECT ID:- 288703

"INTERNSHIP AS ANDROID DEVELOPER"

Submitted by

MAHIDA CHINTAN JAYENDRASINH

(180760107023)

Guided By

Prof. Drasti J Chauhan

Assistant Professor, COED, SSASIT-Surat

ABSTRACT

The thesis researches the ability of use of Python programming language in the field of high school education and uses qualitative and quantitative methods of research and finds Python suitable. The thesis also brings theoretical analysis of the Python language, including practical exams in the form of sample programs. It research even other programming languages used in education and their suitability for this purpose and compares them with Python programming language. It also finds two most common used programming languages at high schools, which are Pascal / Object Pascal and Java.

COMPANY OVERVIEW

Chapter – 1: Overview of Company

Company Name: Tops Technologies
Address: 407 Dhara Arcade Opp Swaminarayan temple, Mahadev Chowk MOTA Varachha Surat ,394101 Gujarat,
Contact No: +91 73830 23957

1.1 About Us

Tops Technologies is a Company Offering Services in The Field of Software Development, Website Development, Mobile App Development, Graphic Designing, Digital Marketing, Testing & QA, Bulk SMS & Hosting Provider and IT Consultancy. In Terms of Services, Designing and Coding. Services Include Customized Software Development, Website Development and Programming, Mobile App Development, Graphic Designing, Digital Marketing, Testing & QA, Bulk SMS & Hosting Provider and IT Consultancy, And Maintenance. Company Has a Vast Experience in Development for Software/Web and Mobile Application. Our Processes Are Highly Streamlined and Proven to Ensure Fastest Delivery of a high quality Solution At A Reasonable Cost. We Understand Our Client's Need and Changes Over Period of Time. We Have Expertise to Accommodate Changes at Any Stage of Software Lifecycle

1.2 Our Work Is Our Identity

We Provide a Wide Range of Web Solutions Such as Professional Website Design and Development, Customized Software Development, Website Development and Programming, Mobile App Development, Graphic Designing, Digital Marketing, Testing & QA, Bulk SMS & Hosting Provider And IT Consultancy To A Wide Range Of Clients. We Give Our First Priority To Our Client Satisfaction. We Solve Any Query of Our Customer with Appropriate Explainations and Also Adopt Their Ideas After Proper Discussions. We Are Available at Any Time According to Our Clients Requirement. We Do Our Best to Fulfill The Requirement Of Our Clients. We Deliver Our Product on Time And To Specification. We Accurately Follow the Best and Usual Web Developing Life Cycle Which Is: Planning, Analysis, Design, Implementing And Review.

Why Choose Us?

Tops Technologies is a company in offering services in the field of Software development, Website Development, Mobile Application Development and Customized ERP Solutions.

Chapter – 2: Introduction

2.1 Project Summary

The health management system project would involve designing and developing a comprehensive software platform that can be used by healthcare providers to manage and store patient data securely. The system should be able to handle a wide range of medical data, including patient demographics, medical histories, diagnoses, treatments, lab results, and medication records.

2.2 Purpose

The purpose of the health management system project is to develop a comprehensive software platform that can improve the quality of healthcare by providing healthcare providers with easy and secure access to electronic health records (EHRs) and other medical data.
The health management system project aims to address some of the challenges that healthcare providers face when managing patient data, including:

Inefficient data management: Healthcare providers often struggle to manage the large volumes of patient data generated in the course of providing care. The health management system can help to streamline data management by providing a centralized platform for storing and accessing patient records.

Medical errors: Medical errors can occur when healthcare providers do not have access to complete and accurate patient information. The health management system can help to reduce the risk of medical errors by providing up-to-date and accurate information on patient histories, diagnoses, treatments, and medications.

Data security and privacy: Patient data must be stored and managed securely to comply with regulations and protect patient privacy. The health management system can help to ensure data security and privacy by implementing robust security protocols and access controls.

Fragmented care: When patient data is stored in disparate systems or paper records, it can be difficult to provide coordinated care across different healthcare providers and settings.The health management system can help to facilitate coordinated care by providing a centralized platform for sharing patient data among healthcare providers.

2.3 Scope

Electronic health records (EHRs) management: The system should allow healthcare providers to create, store, and manage patient health records securely. It should be able to capture patient demographics, medical histories, diagnoses, treatments, lab results, and medication records. Patient scheduling and appointment management: The system should allow healthcare providers to schedule appointments, manage patient waitlists, and send appointment reminders.

Clinical decision support tools: The system should provide healthcare providers with access to clinical decision support tools to help them make informed decisions about patient care. Secure messaging and communication tools: The system should provide secure messaging and communication tools to facilitate collaboration among healthcare providers and patients.

Prescription management and drug interaction checking: The system should allow healthcare providers to manage prescriptions and check for potential drug interactions.
Lab test result management: The system should allow healthcare providers to manage and store lab test results and share them with patients and other healthcare providers.

Billing and payment processing: The system should allow healthcare providers to manage billing and payment processing efficiently.Reporting and analytics tools: The system should provide healthcare providers with reporting and analytics tools to help them track patient outcomes and improve care delivery.

2.4 Objective

Improve patient care: The health management system aims to improve patient care by providing healthcare providers with easy and secure access to electronic health records (EHRs) and other medical data. This will help healthcare providers to make informed decisions about patient care, reduce the risk of medical errors, and improve patient outcomes.

Streamline healthcare operations: The health management system aims to streamline healthcare operations by providing a centralized platform for storing and managing patient data. This will help to reduce inefficiencies in data management, improve communication and collaboration among healthcare providers, and reduce administrative burdens.

Enhance data security and privacy: The health management system aims to enhance data security and privacy by implementing robust security protocols and access controls to protect patient information. This will help to comply with regulations and protect patient privacy.

Increase accessibility: The health management system aims to increase accessibility to patient data by providing healthcare providers with easy and secure access to patient records. This will help to improve care coordination across different healthcare providers and settings.

CHAPTER 3

3.1 Introduction with Python

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development.
- Python supports modules and packages, which encourages program modularity and code reuse.
- The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

3.1 Why Python?

- Designed to be easy to learn and master
- Clean, clear syntax
- Very few keywords
- Highly portable
- Runs almost anywhere - high end servers and workstations, down to windows CE
- Uses machine independent byte-code
- Extensible
- Designed to be extensible using C/C++,
- allowing access to many external libraries

Features of Python

- Clean syntax plus high-level data types
- Leads to fast coding (First language in many universities abroad!)
- Uses white-space to delimit blocks
- Humans generally do, so why not the language?
- Try it, you will end up liking it
- Variables do not need declaration

Python Productivity

- Reduced development time
- code is 2-10x shorter than C, C++, Java
- Improved program maintenance
- code is extremely readable
- Less training
- language is very easy to learn

OOPS in Python

- Python has been an object-oriented language since it existed. Because of this, creating and using classes and objects are downright easy.
- Overview of OOP Terminology
- **Class:** A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.

- **Class variable:** A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- **Data member:** A class variable or instance variable that holds data associated with a class and its objects.
- **Function overloading:** The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.
- **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class.
- **Inheritance:** The transfer of the characteristics of a class to other classes that are derived from it.
- **Instance:** An individual object of a certain class. An object obj that belongs to a class Circle, for example, is an instance of the class Circle.
- **Instantiation:** The creation of an instance of a class.
- **Method:** A special kind of function that is defined in a class definition.
- **Object:** A unique instance of a data structure that's defined by its class. An object comprises both data members (class variables and instance variables) and methods.
- **Operator overloading:** The assignment of more than one function to a particular operator.

Programming Style

- Python programs/modules are written as text files with traditionally a .py extension.
- Each Python module has its own discrete namespace.
- Name space within a Python module is a global one.
- Python modules and programs are differentiated only by the way they are called. .py files executed directly are programs (often referred to as scripts) .py files referenced via the import statement are modules.
- Thus, the same .py file can be a program/script, or a module.

Core python concepts

Conditional Statements

o If o If- else o Nested if-else

- Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.
- Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome.
- **If statement** o It is similar to that of other languages.

o The if statement contains a logical expression using which data is compared and a decision is made based on the result of the comparison. Syntax: if expression:
Statement

- **If Example:**

```
a=10 if a>0:
print(“A Is Positive Number”)
```

If…else Statement

- It is frequently the case that you want one thing to happen when a condition it true, and something else to happen when it is false.
- For that we have the if else statement.
- **Syntax :** if expression:
  Statement
  else: Statement
- **If….else Example**

```
a=10
if
a>0:
    print(“A Is Positive Number”) else:
   print(“A Is Negative Number”)
```

Nested if...else Statement

- There may be a situation when you want to check for another condition after a condition resolves to true.
- In such a situation, you can use the nested if construct
- **Syntax :** if expression1:

```
                Statement
    if expression2:
                Statement
  else:
  Statement
```

- **Nested if example:**

```
a=10
b=20
c=30
if a>b:

        if a>c:
           print(“A Is Greater”) else:
           print(“C Is Greater”)
else:

        if b>c:
           print(“B Is Greater”) else:
           print(“C Is Greater”)
```

Looping

- A loop statement allows us to execute a statement or group of statements multiple times.
- Python programming language provides following types of loops to handle looping requirements.
- While Loop
- For Loop

For loop has the ability to iterate over the items of any sequence, such as a list or a string.

- Next, the statements block is executed.
- Each item in the list is assigned to *iterating_var*, and the statement(s) block is executed until the entire sequence is exhausted

While

- A **while** loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true. Syntax : while expression: Statement
- Here, statement(s) may be a single statement or a block of statements. The condition may be any expression, and true is any non-zero value. The loop iterates while the condition is true.

```
number=23
running=True

while running:
   guess=int (input ("Enter An Integer : "))     if
guess==number:                print  ("Success")
running=False     elif guess<number:
      print ("Lower Than Original")       elif
guess>number:
      print  ("Higher  Than  Original")
else:                print  ("Sorry")  else:
print("The While Loop Is Over") print
("Done")
```

Nested loops

- Python programming language allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.
- Syntax :For for iterating_var in sequence: for iterating_var in sequence: Statements(s) statements(s)
  Note: The range () Function
- If you do need to iterate over a sequence of numbers, the built-in function range() comes in handy.
- It generates Arithmetic progressions.

Control Statements

- Loop control statements change execution from its normal sequence.
- When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
- Python supports the following control statements.
- Break
- Continue
- P

```
i=0;j=0      a
while i<7:
                 s
j=0          while
j<i:s
                  print(i, end=' ')
                   j=j+1
     i=i+1
      print()
```

Break statement:

- It brings control out of the loop and transfers execution to the statement immediately following the loop.

```
for l in "Tops Technology":     if l=='p' or l=='s':        break
print ("Current Letter: ", l)
```

Continue Statement :

- It continues with the next iteration of the loop

```
for l in "Tops Technology":     if l=='p' or l=='s':
continue print ("Current Letter: ", l)
```

Pass Statement:

- The pass statement does nothing.
- It can be used when a statement is required syntactically but the program requires no action. For example:

```
for l in "Tops Technology": pass
print("Current Letter: ", l)
```

String Manipulation:

- Textual data in Python is handled with ”str” objects, or *strings*. Strings are immutable(fixed/rigid) sequences of Unicode code points.
- String literals are written in a variety of ways:
- Single quotes: 'allows embedded "double" quotes' Double quotes: "allows embedded 'single' quotes".
- Triple quoted: '''Three single quotes''', """Three double quotes"""
- Triple quoted strings may span multiple lines - all associated whitespace will be included in the string literal.

String functions

- **str.capitalize()**
- Return a copy of the string with its first character capitalized and the rest lowercased.
- **str.casefold()**
- Return a case folded copy of the string. Case folded strings may be used for caseless matching.
- **str.center(*width*[, *fillchar*])**
- Return centered in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if*width* is less than or equal to len(s).
- **str.count(*sub*[, *start*[, *end*]])**
- Return the number of non-overlapping occurrences of substring *sub* in the range [*start*, *end*].
- Optional arguments *start* and *end* are interpreted as in slice notation.
- **str.endswith(*suffix*[, *start*[, *end*]])**
- Return True if the string ends with the specified *suffix*, otherwise return False.
- *Suffix* can also be a tuple of suffixes to look for.
- With optional *start*, test beginning at that position.
- With optional *end*, stop comparing at that position.
- **str.find(*sub*[, *start*[, *end*]])**
- Return the lowest index in the string where substring *sub* is found within the slice s [start: end].

Note:

- The find () method should be used only if you need to know the position of *sub*. To check if *sub* is a substring or not, use the ”in”operator:
- >>>>>> 'Py' **in** 'Python' True
- **str.format(**args*, ***kwargs*)**
- Perform a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces{}.
- Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.
- str.index (*sub* [, *start* [, *end*]]) Like find (), but raise Value Error when the substring is not found.
- **str.isalnum()**
- Return true if all characters in the string are alphanumeric and there is at least one character, false otherwise.
- A character c is alphanumeric if one of the following returns True:
- **c.isalpha(), c.isdecimal(), c.isdigit(), or c.isnumeric().**
- **str.isidentifier()**

- Return true if the string is a valid identifier according to the language definition **str.islower()**
- Return true if all cased characters in the string are lowercase and there is at least one cased character, false otherwise.
- **str.istitle()**
- Return true if the string is a titlecased string and there is at least one character, for example uppercase characters may only follow uncased characters and lowercase characters only cased ones. Return false otherwise.
- **str.isupper()**
- Return true if all cased characters in the string are uppercase and there is at least one cased character, false otherwise.

- **str.ljust(*width*[, *fillchar*])**
- Return the string left justified in a string of length *width*. Padding is done using the specified *fillchar* (default is an ASCII space). The original string is returned if *width* is less than or equal to len(s).
- **str.lower()**
- Return a copy of the string with all the cased characters converted to lowercase.
- **str.partition(*sep*)**
- Split the string at the first occurrence of *sep*, and return a 3-tuple containing the part before the separator, the separator itself, and the part after the separator.
- If the separator is not found, return a 3-tuple containing the string itself, followed by two empty strings.
- **str.replace(*old*, *new*[, *count*])**
- Return a copy of the string with all occurrences of substring *old* replaced by *new*.
- If the optional argument *count* is given, only the first countoccurrences are replaced.
- **str.split(*sep=None*, *maxsplit=-1*)**

  Return a list of the words in the string, using *sep* as the delimiter string. If *maxsplit* is given, at most *maxsplit* splits are done (thus, the list will have at most maxsplit+1 elements).
- If *maxsplit* is not specified or -1, then there is no limit on the number of splits **str.swapcase()**
- Return a copy of the string with uppercase characters converted to lowercase and vice versa.
- **str.title()**
- Return a titlecased version of the string where words start with an uppercase character and the remaining characters are lowercase.

Functions

Name	Description
len(list)	Gives the total length of the list.
max(list)	Returns item from the list with max value.
min(list)	Returns item from the list with min value.
list(seq)	Converts a tuple into list.

Methods	Description
list.append(x)	Add an item to the end of the list. Equivalent to a [len(a):]=[x].
list.extend(L)	Appends the contents of L to list
list.insert(I,x)	Insert an item at a given position. The first argument is the index of the element before which to insert, so a.insert(0,x) inserts at the front of the list,and a.insert(len(a),x) is equivalent to a.append(x).
list.count(obj)	Returns count of how many times obj occurs in list
list.index(obj)	Returns the lowest index in list that obj appears
List.pop(obj=list[-1]	Removes and returns last object or obj from list.
List.reverse()	Reverses Objects of list in place
List.sort([fun])	Sorts objects of list, use compare function If given
List.remove(obj)	Removes object obj from list

Dictionaries

- Introduction
- Accessing values in dictionaries
- Working with dictionaries
- Properties
- Functions

1 . Introduction

- Dictionaries are sometimes found in other languages as “associative memories” or “associative arrays”.
- Unlike sequences, which are indexed by a range of numbers, dictionaries are indexed by *keys*, which can be any immutable type; strings and numbers can always be keys.
- Tuples can be used as keys if they contain only strings, numbers, or tuples; if a tuple contains any mutable object either directly or indirectly, it cannot be used as a key.
- You can’t use lists as keys, since lists can be modified in place using index assignments, slice assignments, or methods like append() and extend().
- The main operations on a dictionary are storing a value with some key and extracting the value given the key.
- Like lists they can be easily changed, can be shrunk and grown ad libitum at run time. They shrink and grow without the necessity of making copies. Dictionaries can be contained in lists and vice versa.
- A list is an ordered sequence of objects, whereas dictionaries are unordered sets.
- But the main difference is that items in dictionaries are accessed via keys and not via their position.

2 Methods and Functions Methods

dist.copy()	A dictionary can be copied with the method copy():.
dist.update()	It merges the keys and values of one dictionary into another, overwriting values of the same key.
dist.values()	It returns the list of dictionary dict’s values.
dist.keys()	It returns the list of dictionary dict’s keys.
dist.items()	It returns the list of dictionary dict’s keys,values in tuple pairs.
dist.clear()	Removes all elements of dictionary dict.

Defining a function

- Python gives us many built-in functions like print(), etc. but we can also create our own functions.
- A function in Python is defined by a def statement. The general syntax looks like this:
- The keyword ”def” introduces a function *definition.*
- It must be followed by the function name and the parenthesized list of formal parameters.
- The statements that form the body of the function start at the next line, and must be indented.
- The ”return” statement returns with a value from a function.”return” without an expression argument returns None.
- Falling off the end of a function also returns None.
- Example:

  -
    ```
    def checkNoEvenOdd(x):
    if x % 2 == 0:
    print(x, "is even")
    else:
            print(x, "is odd")
    ```
- Calling a function
- Once function define, we can call it directly or in any other function also.
- Eg. checkNoEvenOdd(20)
- Output: 20 is even

Types of functions

Functions can be of

Built-in functions	Functions that come built into the Python language itself are called built-in functions and are readily available to us. Eg: input(),eval(),print() etc…
User defined functions	Functions that we define ourselves to do certain specific task are referred as user-defined functions. Eg:checkNoEvenOdd(20)

- Function Arguments
- It is possible to define functions with a variable number of arguments.
- The function arguments can be
- Default arguments values
- Keyword arguments
- ArbitraryArgumentLists
- Default arguments values
- The most useful form is to specify a default value for one or more arguments.
- This creates a function that can be called with fewer arguments than it is defined to allow. Eg. def employeeDetails(name,gender='male',age=35)
- This function can be called in several ways:
- called using keyword arguments of the form kwarg=value.
- For instance, the following function:
- def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):

- **Modifiers :**
- Regular expression literals may include an optional modifier to control various aspects of matching. The modifiers are specified as an optional flag. You can provide multiple modifiers using exclusive OR (|).

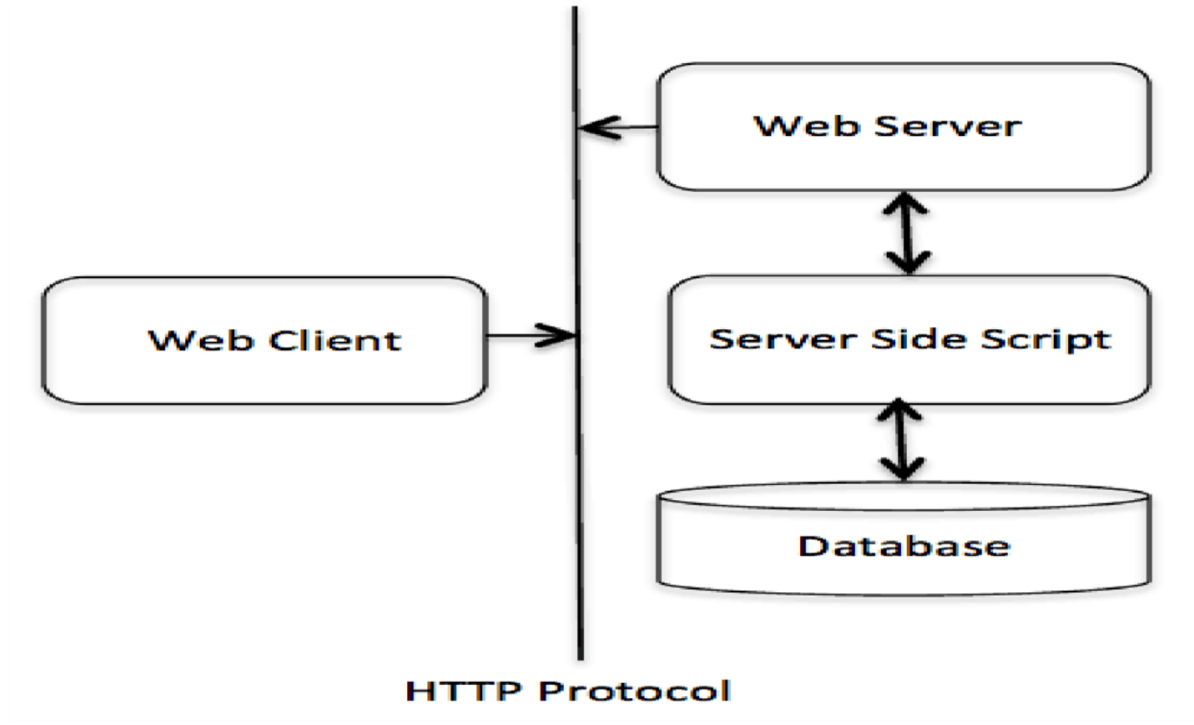
Modifier	**Description**
re.I :IGNORECASE,	Perform case-insensitive matching; character class and literal strings will match letters by ignoring case. For example, [A-Z]will match lowercase letters, too, and Spam will match Spam, spam, or spAM.
re.L LOCALE	
re.M :MULTILINE,	Multi-line matching, affecting ^ and $. When this flag is specified, ^ matches at the beginning of the string and at the beginning of each line within the string, immediately following each newline. Similarly, the $metacharacter matches either at the end of the string and at the end of each line
re.A ASCII	Make \w, \W, \b, \B, \s and \S perform ASCII-only matching instead of full Unicode matching. This is only meaningful for Unicode patterns, and is ignored for byte patterns.
re.S DOTALL	Makes the '.' special character match any character at all, including a newline; without this flag, '.' will match anything except a newline.
re.XVERBOSE	When this flag has been specified, whitespace within the RE string is ignored, except when the whitespace is in a character class or preceded by an unescaped backslash; this lets you organize and indent the RE more clearly.

Patterns

 Except for control characters, **(+ ? . * ^ $ ( ) [ ] { } |** \), all characters match themselves. You can escape a control character by preceding it with a backslash.

Description	
|	Alternation, or the “or” operator.
^	Matches at the beginning of lines.
$	Matches at the end of a line,
\A	Matches only at the start of the string.
\Z	Matches only at the end of the string.
\b	Word boundary. This is a zero-width assertion that matches only at the beginning or end of a word.
\w	Matches the word characters.
\W	Matches the nonword characters.
\d	Matches digits.
\D	Matches non digits.
\s	Matches whitespaces.
\S	Matches nonwhitespaces.
\B	Another zero-width assertion, this is the opposite of \b, only matching when the current position is not at a word boundary.
re{n,m}	Matches at least n and at most m occurrences of preceding expression.

Architecture



Variables	
CONTENT_TYPE	The data type of the content. Used when the client is sending attached content to the server. For example, file upload.
CONTENT_LENGTH	The length of the query information. It is available only for POST requests.
HTTP_COOKIE	Returns the set cookies in the form of key & value pair.
HTTP_USER_AGENT	The User-Agent request-header field contains information about the user agent originating the request. It is name of the web browser.
SERVER_NAME	The server's hostname or IP Address
SERVER_SOFTWARE	The name and version of the software the server is running.

Get And Post Methods

- o Browser uses two methods two pass this information to web server. These methods are GET Method and POST Method.
- o The GET method sends the encoded user information appended to the page request.
  The page and the encoded information are separated by the ? Character like o http://www.xyz.com/cgi-bin/hello.py?key1=value1&key2=value2 o The GET method is the default method to pass information from browser to web server.
- o Never use GET method if you have password or other sensitive information to pass to the server.
- o The GET method has size limtation: only 1024 characters can be sent in a request string.
- o The GET method sends information using QUERY_STRING header and will be accessible in your CGI Program through QUERY_STRING environment variable.

Post method

- o A generally more reliable method of passing information to a CGI program is the POST method.
- o This packages the information in exactly the same way as GET methods, but instead of sending it as a text string after a ? in the URL it sends it as a separate message.
- o This message comes into the CGI script in the form of the standard input. o

Cookies

- o HTTP protocol is a stateless protocol.
- o For a commercial website, it is required to maintain session information among different pages. For example, one user registration ends after completing many pages.
- o In many situations, using cookies is the most efficient method of remembering and tracking preferences, purchases, commissions, and other information required for better visitor experience or site statistics.

 How It Works?

- o **Your server sends some data to the visitor's browser in the form of a cookie.** o **The browser may accept the cookie.** o If it does, it is stored as a plain text record on the visitor's hard drive.
- o Now, when the visitor arrives at another page on your site, the cookie is available for retrieval.
- o Cookies are a plain text data record of 5 variable-length fields:
- o Expires: The date the cookie will expire. If this is blank, the cookie will expire when the visitor quits the browser.
- o Domain: The domain name of your site.
- o Path: The path to the directory or web page that sets the cookie. This may be blank if you want to retrieve the cookie from any directory or page.

File Upload

 To upload a file, the HTML form must have the enctype attribute set to multipart/form-data. The input tag with the file type creates a "Browse" button.

```
<html>
<body>
  <form enctype="multipart/form-data"
              action="save_file.py"   method="post">         <p>File:
<input type="file" name="filename" /></p>
  <p><input type="submit" value="Upload" /></p>
  </form>
</body>
```

</html>

MultiThreading

- Thread
    - A Thread or a Thread of Execution is defined in computer science as the smallest unit that can be scheduled in an operating system.
    - Threads are usually contained in processes.
    - More than one thread can exist within the same process.
    - Every process has at least one thread, i.e. the process itself.
    - A process can start multiple threads.
- Starting A Thread
    - There are two modules which support the usage of threads in Python:
    - thread
    - &
    - threading
    - It's possible to execute functions in a separate thread with the module Thread.
    - To do this, we can use the function thread.start_new_thread:
    - thread.start_new_thread(function, args[, kwargs]) This method starts a new thread and return its identifier.
    - The method call returns immediately and the child thread starts and calls function with the passed list of *agrs*. When function returns, the thread terminates.
    - Here, *args* is a tuple of arguments; use an empty tuple to call function without passing any arguments. *kwargs* is an optional dictionary of keyword arguments.

Example

- Threading Module
    - The **threading** module constructs higher-level threading interfaces on top of the lower level **_thread** module.
    - Creating Thread using threading module.
    - Define a new subclass of the *Thread* class.
    - Override the *__init__(self [,args])* method to add additional arguments.
    - Then, override the run(self [,args]) method to implement what the thread should do when started.
    - Once you have created the new *Thread* subclass, you can create an instance of it and then start a new thread by invoking the *start()*, which in turn calls *run()*method.
    - The *threading* module exposes all the methods of the *thread* module and provides some additional methods:

Method	Description
run():	The run() method is the entry point for a thread.
start():	The start() method starts a thread by calling the run method.
join([time]):	The join() waits for threads to terminate.
isAlive():	The isAlive() method checks whether a thread is still executing.
getName():	The getName() method returns the name of a thread.
setName():	The setName() method sets the name of a thread.

CHAPTER 4 Django

4.1 What is Django?

- Django is a free and open source web application framework, written in Python. A web framework is a set of components that helps you to develop websites faster and easier. o When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc.
- To understand what Django is actually for, we need to take a closer look at the servers. o The first thing is that the server needs to know that you want it to serve you a web page.
- Imagine a mailbox (port) which is monitored for incoming letters (requests). This is done by a web server.
- The web server reads the letter and then sends a response with a webpage. o But when you want to send something, you need to have some content. And Django is something that helps you create the content.

4.2 How It Works?

- When a request comes to a web server, it's passed to Django which tries to figure out what is actually requested.
- It takes a web page address first and tries to figure out what to do. This part is done by Django's urlresolver (note that a website address is called a URL – Uniform Resource Locator – so the name *urlresolver* makes sense).
- It is not very smart – it takes a list of patterns and tries to match the URL. Django checks patterns from top to bottom and if something is matched, then Django passes the request to the associated function (which is called *view*).

4.3 Django Installation

- **Virtual environment**
- Before we install Django we will get you to install an extremely useful tool to help keep your coding environment tidy on your computer.
- It's possible to skip this step, but it's highly recommended.
- So, let's create a virtual environment (also called a *virtualenv*). Virtualenv will isolate your Python/Django setup on a per-project basis.
- This means that any changes you make to one website won't affect any others you're also developing
- All you need to do is find a directory in which you want to create the virtualenv;
- For windows ,
- To create a new virtualenv,you need to open the console and run C:\Python35\python –m venv myvenv.

- It will look like this:
- Command-line

- C:\Users\Name\djangogirls> C:\Python35\python –m venv myvenv where C:\Python35\python is the directory in which python is installed and ‘myvenv’ is the name of your virtualenv.

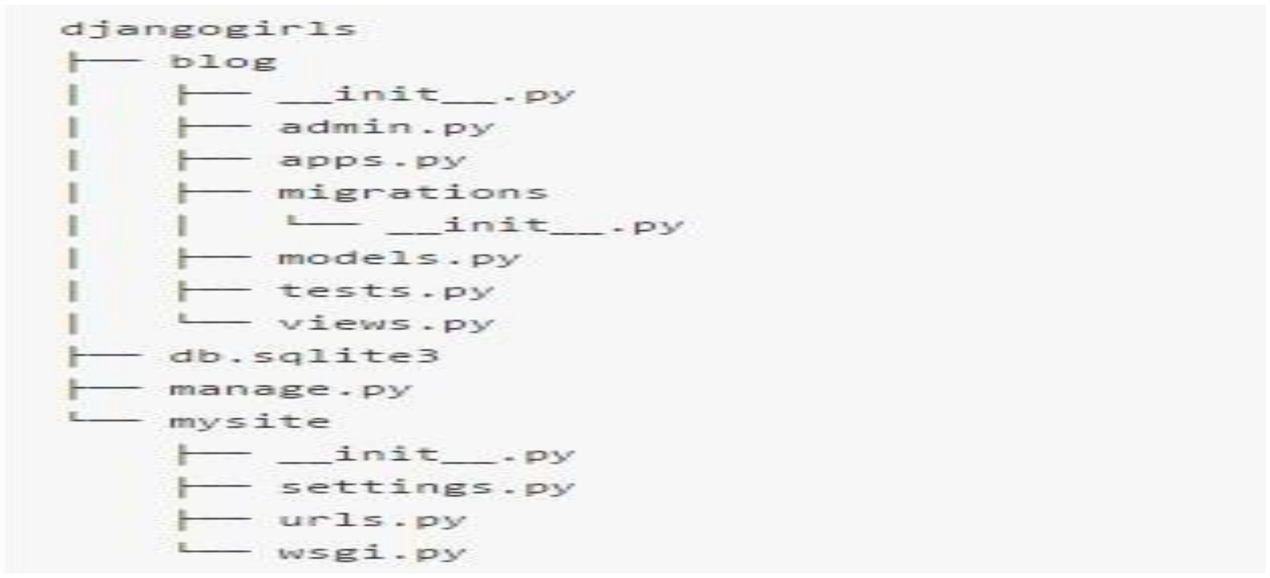
- **Database Setup**
  - There's a lot of different database software that can store data for your site. We'll use the default one, sqlite3.
  - This is already set up in this part of your mysite/settings.py file.
  - mysite/settings.py

```
DATABASES = {
        'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'mydatabase',
            }
          }
```

- **For Postgres database :**

```
DATABASES = {
        'default': {
    'ENGINE': 'django.db.backends.postgresql_psycopg2',
     'NAME': 'todo',
    'USER': 'postgres',
    'PASSWORD': 'admin',
    'HOST': 'localhost',
     'PORT': '5432',
            }
          }
```

  - = Model in database as a spreadsheet with columns(fields) and rows (data).
  - Creating an application
  - To create an application we need to run the following command in the console
  - (myvenv)~/djangogirls$ python manage.py startapp blog
  - You will notice that a new 'blog' directory is created and it contains a



```
djangogirls
├── blog
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── db.sqlite3
├── manage.py
└── mysite
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

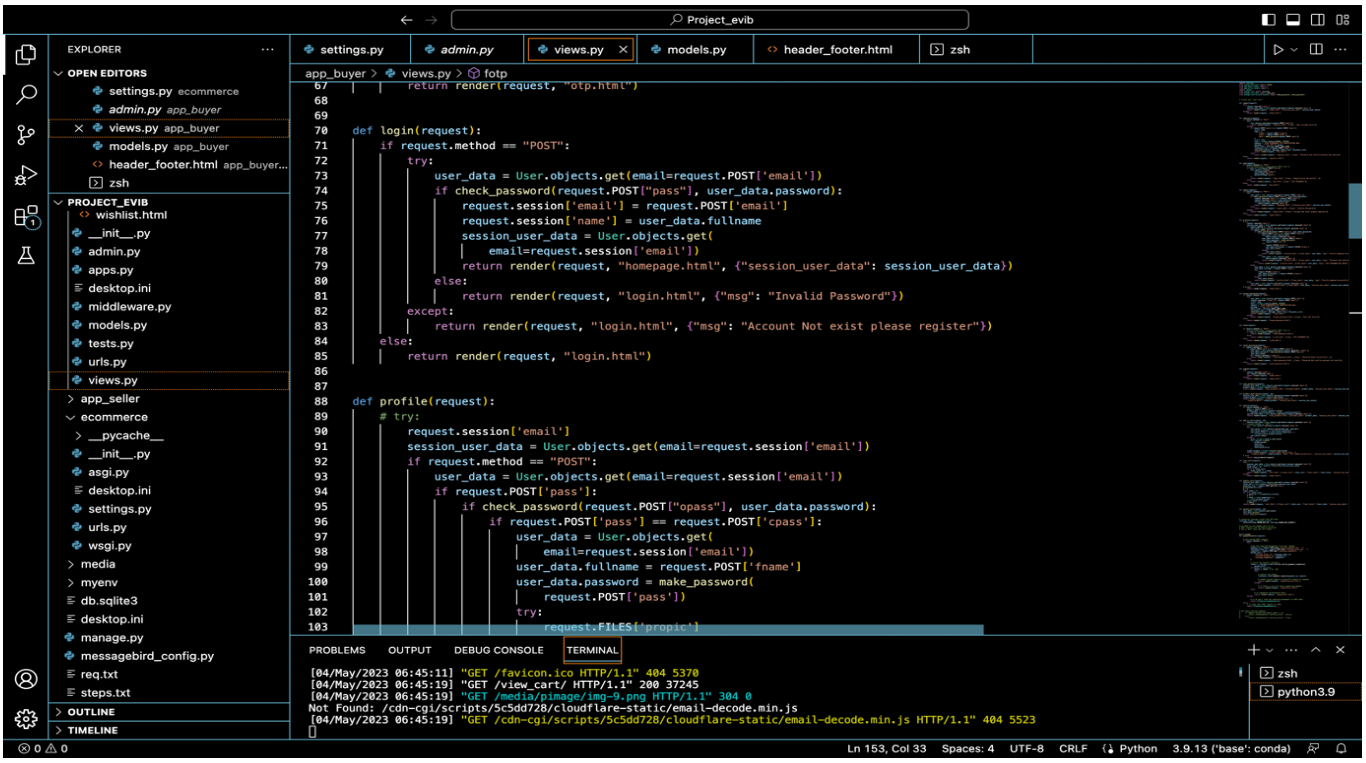
- After creating an application, we also need to tell Django that it should use it. We do that in the file mysite/settings.py.
- We need to find INSTALLED_APPS and add a line containing 'blog' just above].
- EG:
- INSTALLED_APPS = [
- 'django.contrib.admin',
- 'django.contrib.auth',
- 'django.contrib.contenttypes',
- 'django.contrib.sessions',
- 'django.contrib.messages',
- 'django.contrib.staticfiles',
- 'blog'
-]

- **Creating Model**
  - Creating a blog post model:
  - In the blog/models.py file we define all objects called Models – this is a place in which we will define our blog post. □ blog/models.py
  - class Post(models.Model):
  - ……….
  - ……….
  - Create tables for models in your database
  - The last step here is to add our new model to our database.
  - First we have to make Django know that we have some changes in our model. (We have just created it!)
  - Go to your console window and type
  - python manage.py makemigrations blog
  - Django prepared a migration file for us that we now have to apply to our database. Type
  - python manage.py migrate blog

Django Admin

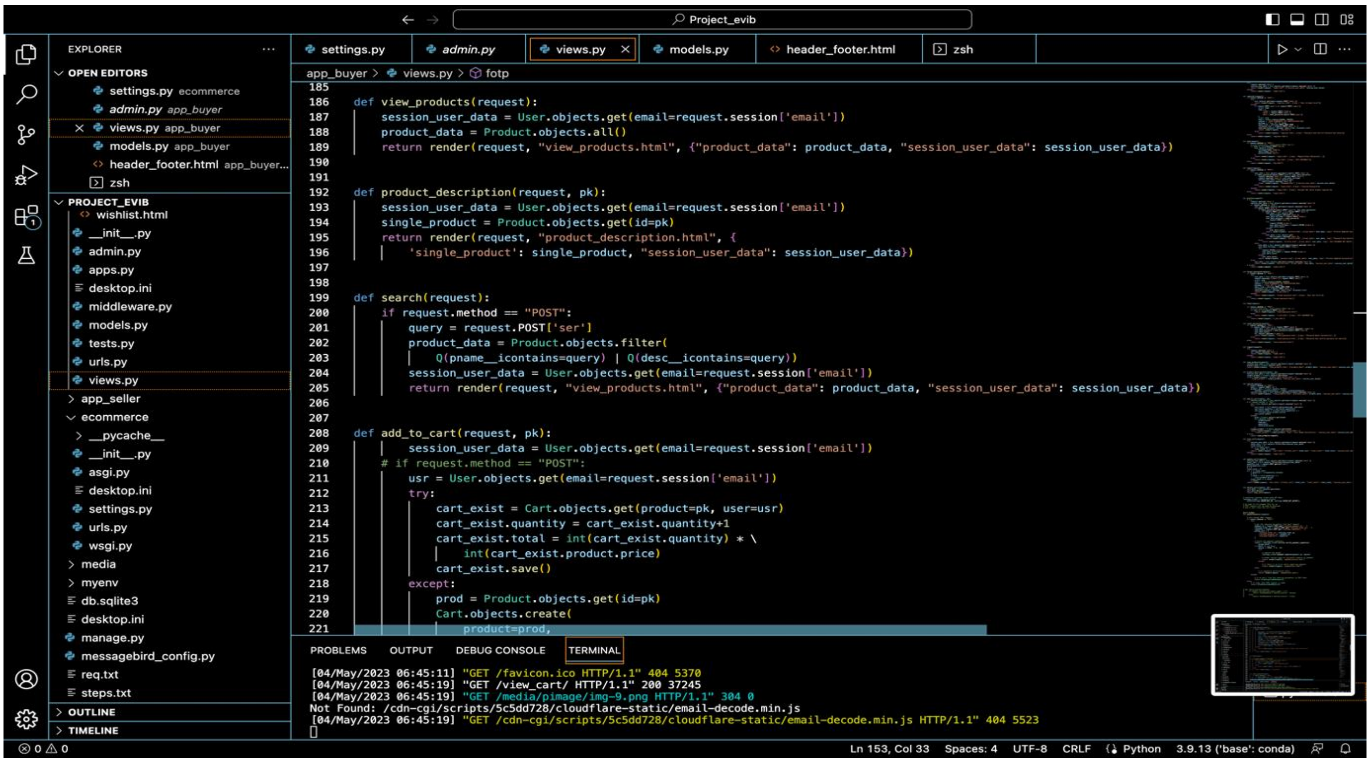
- To add, edit and delete the posts we've just modeled, we will use Django admin.
- Let's open the blog/admin.py file and replace its contents with this:
- blog/admin.py
- from django.contrib import admin
- from .models import Post
- admin.site.register(Post)
- To log in, you need to create a *superuser* - a user account that has control over everything on the site. Go back to the command line, type python manage.py createsuperuser and press enter.
- (myvenv) ~/djangogirls$ python manage.py createsuperuser
- Username: admin
- Email address: admin@admin.com Password:
- Password (again):
- Superuser created successfully.

CHAPTER 5 CODING



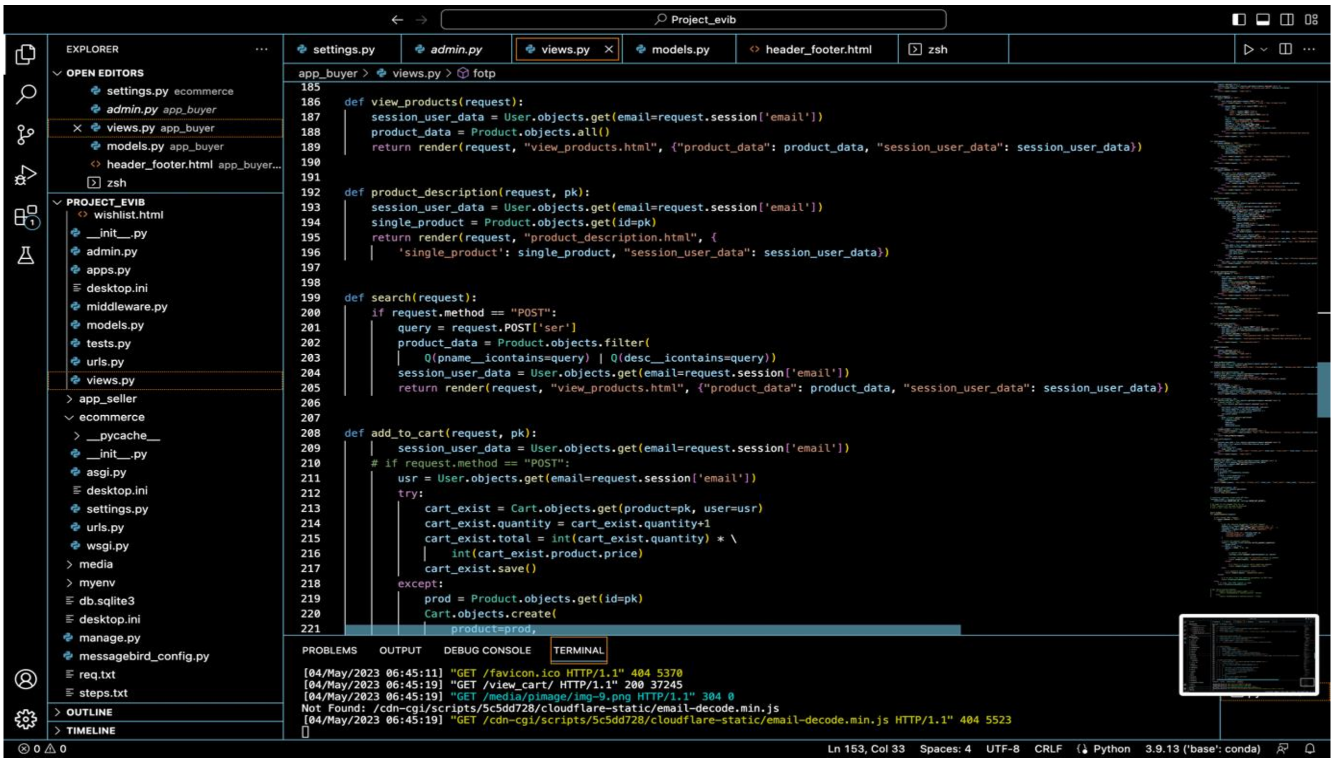
```python
        return render(request, "otp.html")


def login(request):
    if request.method == "POST":
        try:
            user_data = User.objects.get(email=request.POST['email'])
            if check_password(request.POST["pass"], user_data.password):
                request.session['email'] = request.POST['email']
                request.session['name'] = user_data.fullname
                session_user_data = User.objects.get(
                    email=request.session['email'])
                return render(request, "homepage.html", {"session_user_data": session_user_data})
            else:
                return render(request, "login.html", {"msg": "Invalid Password"})
        except:
            return render(request, "login.html", {"msg": "Account Not exist please register"})
    else:
        return render(request, "login.html")


def profile(request):
    # try:
        request.session['email']
        session_user_data = User.objects.get(email=request.session['email'])
        if request.method == "POST":
            user_data = User.objects.get(email=request.session['email'])
            if request.POST['pass']:
                if check_password(request.POST["opass"], user_data.password):
                    if request.POST['pass'] == request.POST['cpass']:
                        user_data = User.objects.get(
                            email=request.session['email'])
                        user_data.fullname = request.POST['fname']
                        user_data.password = make_password(
                            request.POST['pass'])
                        try:
                            request.FILES['propic']
```

```
[04/May/2023 06:45:11] "GET /favicon.ico HTTP/1.1" 404 5370
[04/May/2023 06:45:19] "GET /view_cart/ HTTP/1.1" 200 37245
[04/May/2023 06:45:19] "GET /media/pimage/img-9.png HTTP/1.1" 304 0
Not Found: /cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js
[04/May/2023 06:45:19] "GET /cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js HTTP/1.1" 404 5523
```



```python

def view_products(request):
    session_user_data = User.objects.get(email=request.session['email'])
    product_data = Product.objects.all()
    return render(request, "view_products.html", {"product_data": product_data, "session_user_data": session_user_data})


def product_description(request, pk):
    session_user_data = User.objects.get(email=request.session['email'])
    single_product = Product.objects.get(id=pk)
    return render(request, "product_description.html", {
        'single_product': single_product, "session_user_data": session_user_data})


def search(request):
    if request.method == "POST":
        query = request.POST['ser']
        product_data = Product.objects.filter(
            Q(pname__icontains=query) | Q(desc__icontains=query))
        session_user_data = User.objects.get(email=request.session['email'])
        return render(request, "view_products.html", {"product_data": product_data, "session_user_data": session_user_data})


def add_to_cart(request, pk):
        session_user_data = User.objects.get(email=request.session['email'])
    # if request.method == "POST":
        usr = User.objects.get(email=request.session['email'])
        try:
            cart_exist = Cart.objects.get(product=pk, user=usr)
            cart_exist.quantity = cart_exist.quantity+1
            cart_exist.total = int(cart_exist.quantity) * \
                int(cart_exist.product.price)
            cart_exist.save()
        except:
            prod = Product.objects.get(id=pk)
            Cart.objects.create(
                product=prod,
```

```
[04/May/2023 06:45:11] "GET /favicon.ico HTTP/1.1" 404 5370
[04/May/2023 06:45:19] "GET /view_cart/ HTTP/1.1" 200 37245
[04/May/2023 06:45:19] "GET /media/pimage/img-9.png HTTP/1.1" 304 0
Not Found: /cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js
[04/May/2023 06:45:19] "GET /cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js HTTP/1.1" 404 5523
```

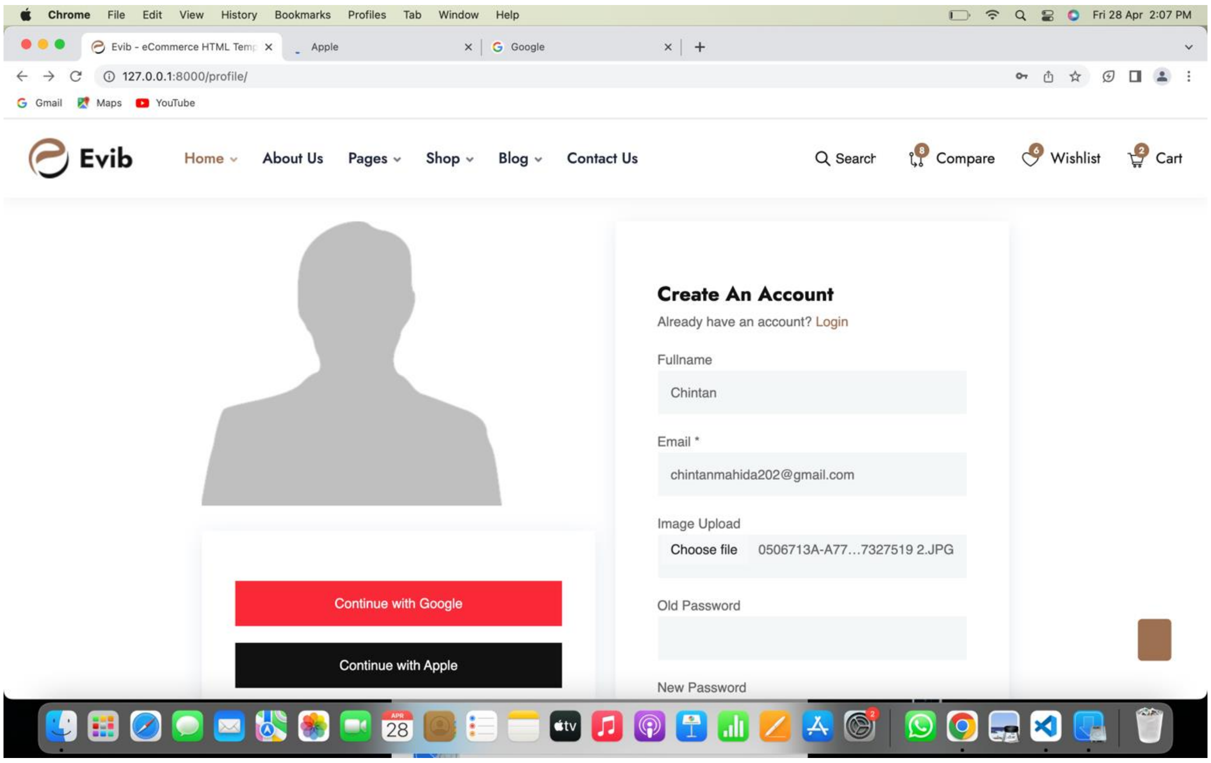


```python

def view_products(request):
    session_user_data = User.objects.get(email=request.session['email'])
    product_data = Product.objects.all()
    return render(request, "view_products.html", {"product_data": product_data, "session_user_data": session_user_data})


def product_description(request, pk):
    session_user_data = User.objects.get(email=request.session['email'])
    single_product = Product.objects.get(id=pk)
    return render(request, "product_description.html", {
        'single_product': single_product, "session_user_data": session_user_data})


def search(request):
    if request.method == "POST":
        query = request.POST['ser']
        product_data = Product.objects.filter(
            Q(pname__icontains=query) | Q(desc__icontains=query))
        session_user_data = User.objects.get(email=request.session['email'])
        return render(request, "view_products.html", {"product_data": product_data, "session_user_data": session_user_data})


def add_to_cart(request, pk):
        session_user_data = User.objects.get(email=request.session['email'])
    # if request.method == "POST":
        usr = User.objects.get(email=request.session['email'])
        try:
            cart_exist = Cart.objects.get(product=pk, user=usr)
            cart_exist.quantity = cart_exist.quantity+1
            cart_exist.total = int(cart_exist.quantity) * \
                int(cart_exist.product.price)
            cart_exist.save()
        except:
            prod = Product.objects.get(id=pk)
            Cart.objects.create(
                product=prod,
```

```
[04/May/2023 06:45:11] "GET /favicon.ico HTTP/1.1" 404 5370
[04/May/2023 06:45:19] "GET /view_cart/ HTTP/1.1" 200 37245
[04/May/2023 06:45:19] "GET /media/pimage/img-9.png HTTP/1.1" 304 0
Not Found: /cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js
[04/May/2023 06:45:19] "GET /cdn-cgi/scripts/5c5dd728/cloudflare-static/email-decode.min.js HTTP/1.1" 404 5523
```

PROJECT ID:- 288703

5.1 SCREENSHOT OF WEBSITE



Create An Account

Already have an account? Login

Fullname: Chintan

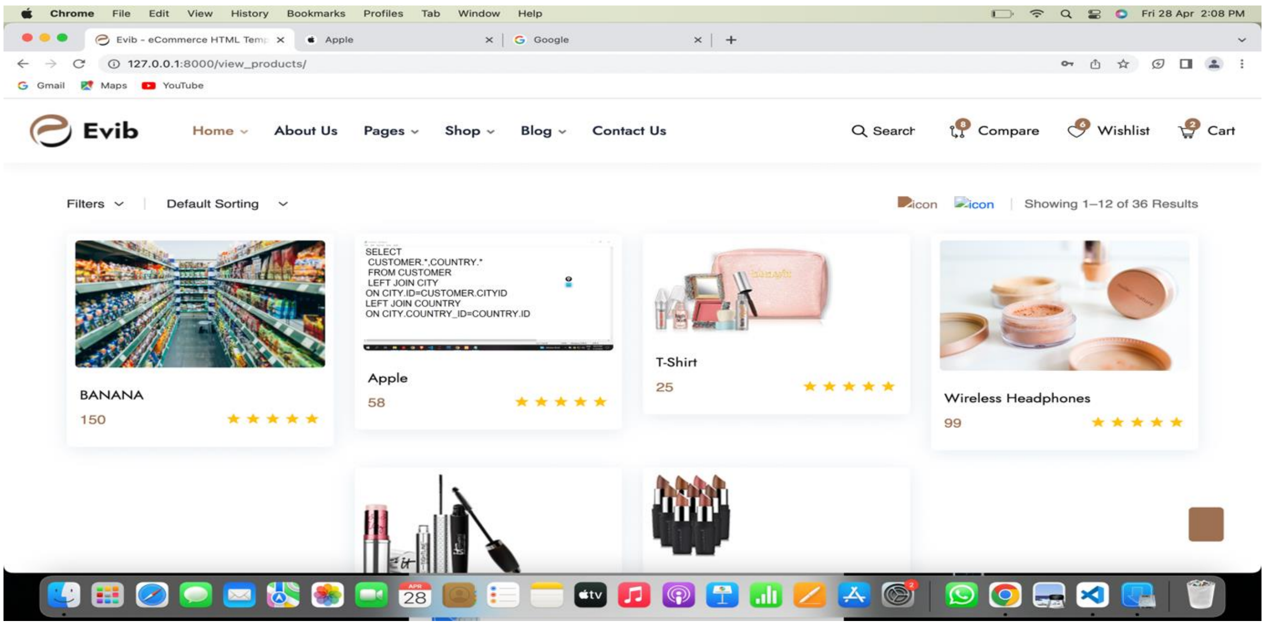
Email *: chintanmahida202@gmail.com

Image Upload: Choose file 0506713A-A77...7327519 2.JPG

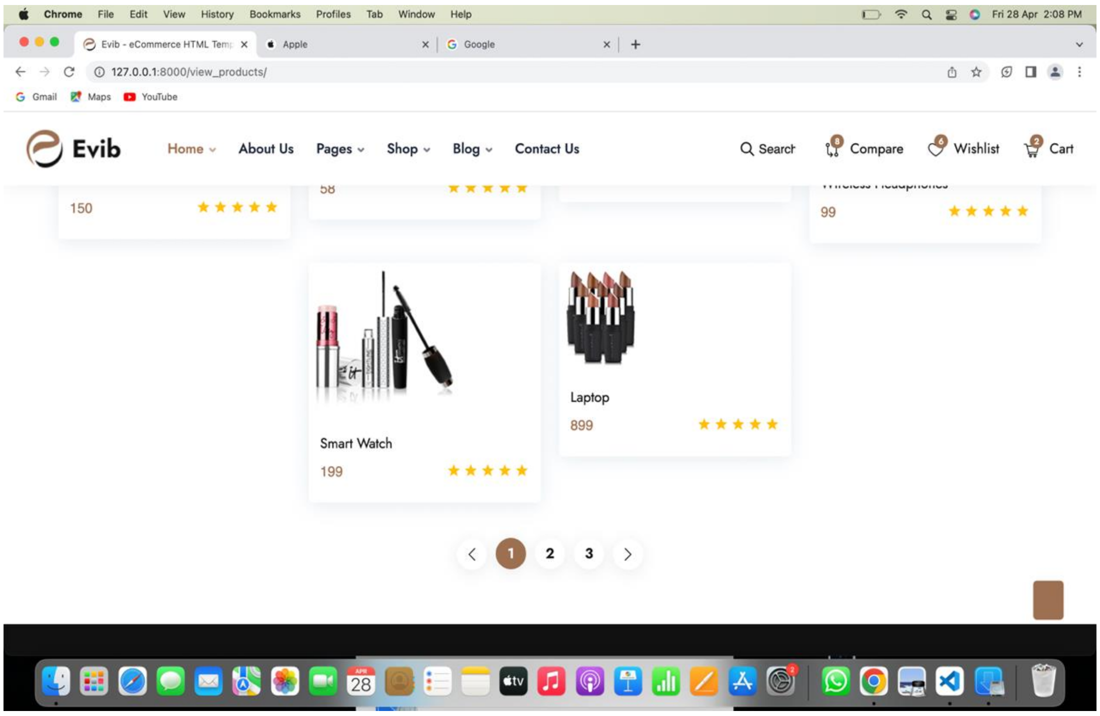
Old Password

New Password

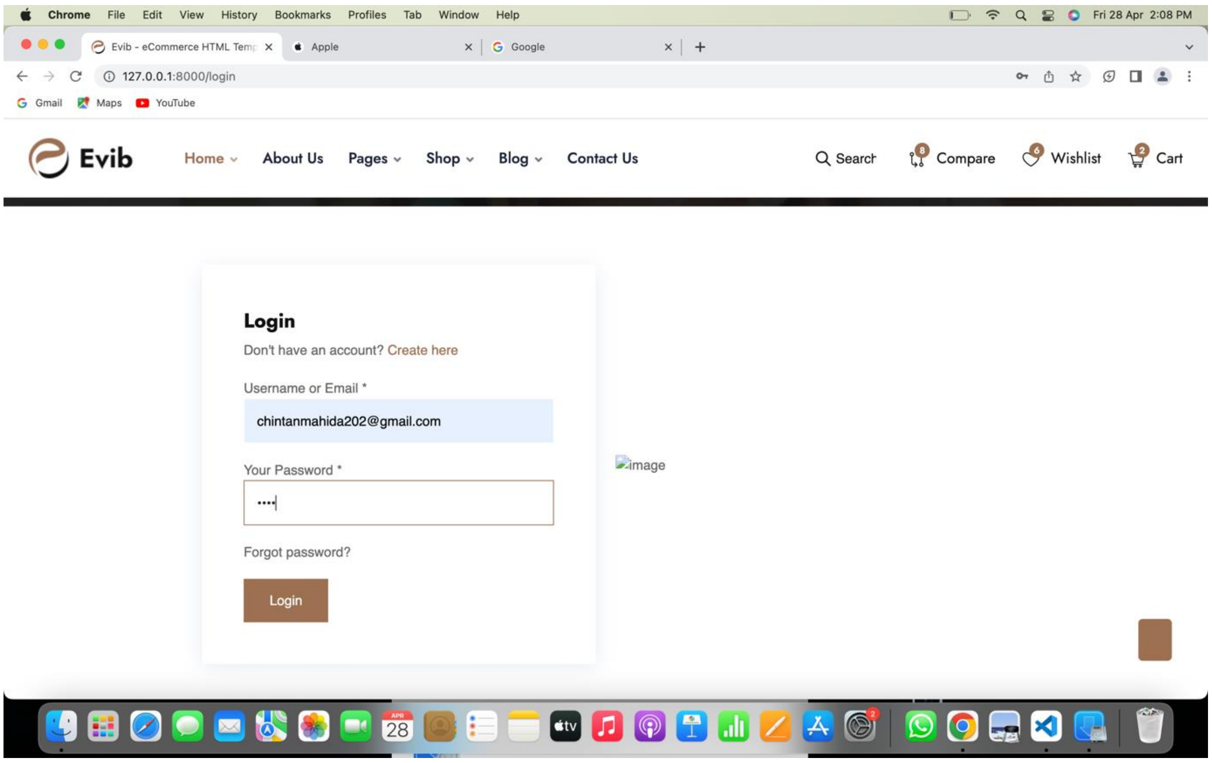
(Figure 5.1) Profile page



```
SELECT
 CUSTOMER.*,COUNTRY.*
 FROM CUSTOMER
 LEFT JOIN CITY
 ON CITY.ID=CUSTOMER.CITYID
 LEFT JOIN COUNTRY
 ON CITY.COUNTRY_ID=COUNTRY.ID
```



(Figure 5.2) Products page



Login

Don't have an account? Create here

Username or Email *

chintanmahida202@gmail.com

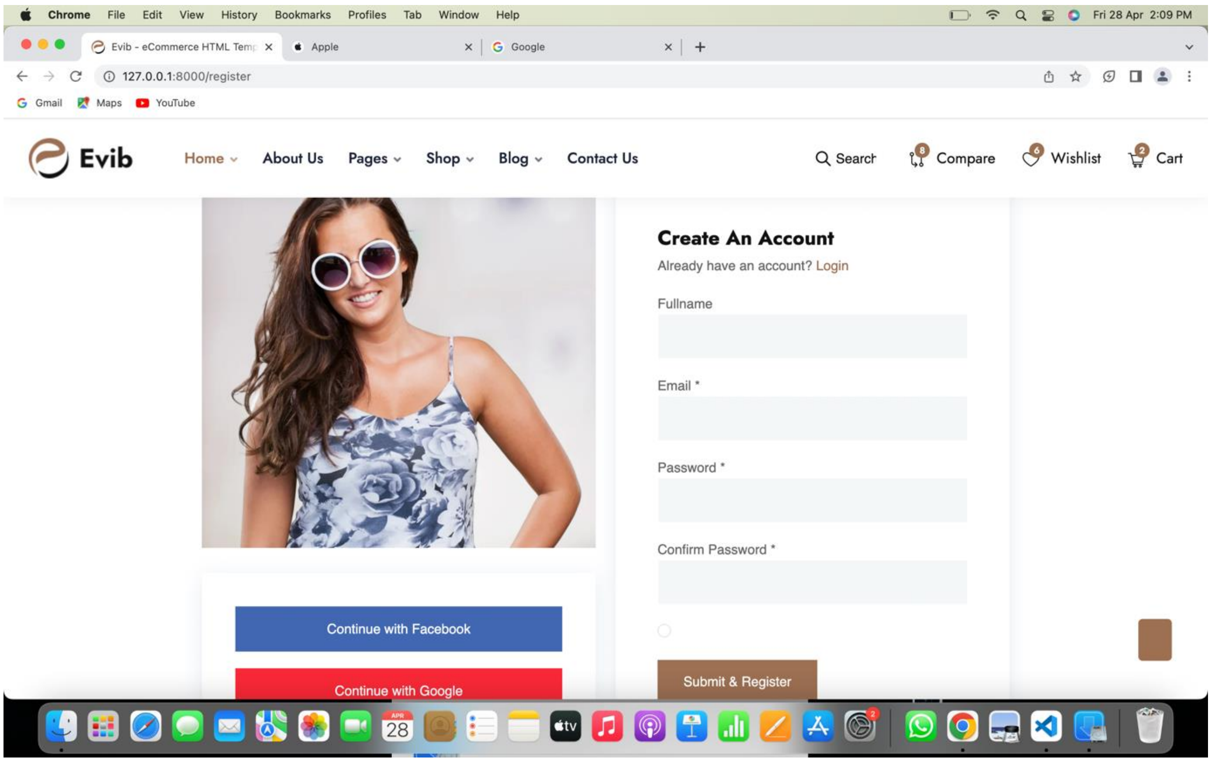
Your Password *

••••

Forgot password?

Login

(Figure 5.3) Login page



Create An Account

Already have an account? Login

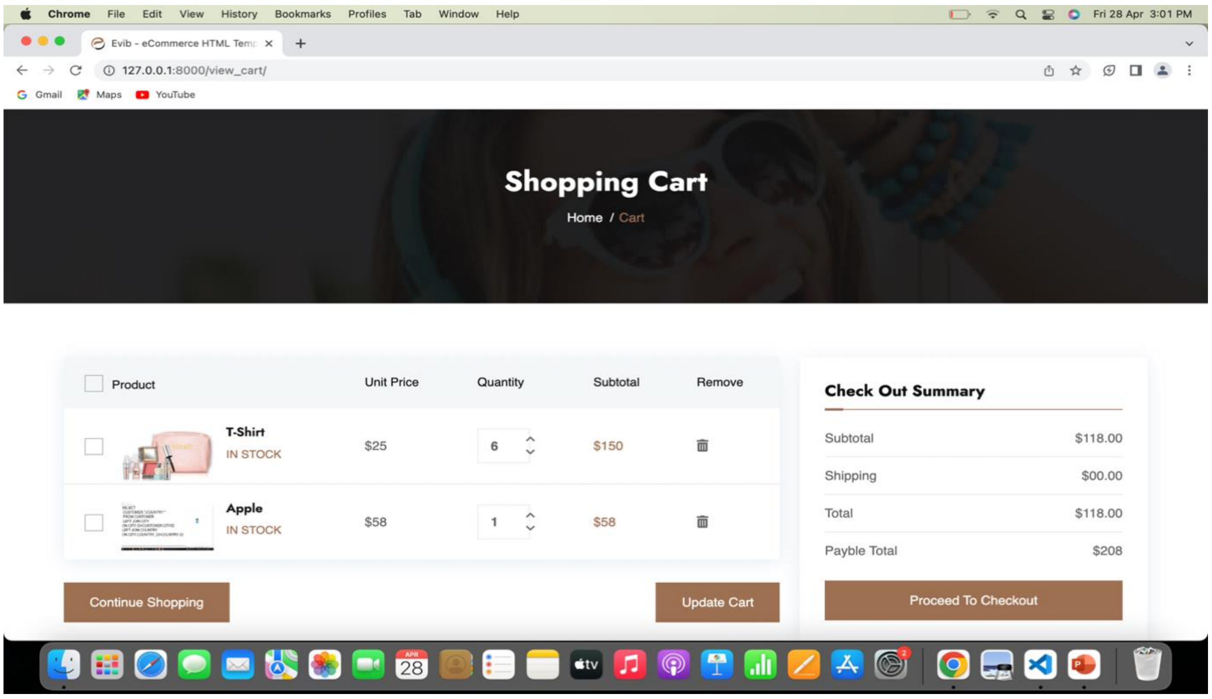
Fullname

Email *

Password *

Confirm Password *

(Figure 5.4) Register page


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 28 Apr 3:01 PM
Evib - eCommerce HTML Temp
127.0.0.1:8000/view_cart/
Gmail Maps YouTube

Shopping Cart

Home / Cart

Product	Unit Price	Quantity	Subtotal	Remove
T-Shirt IN STOCK	$25	6	$150	
Apple IN STOCK	$58	1	$58	

Continue Shopping

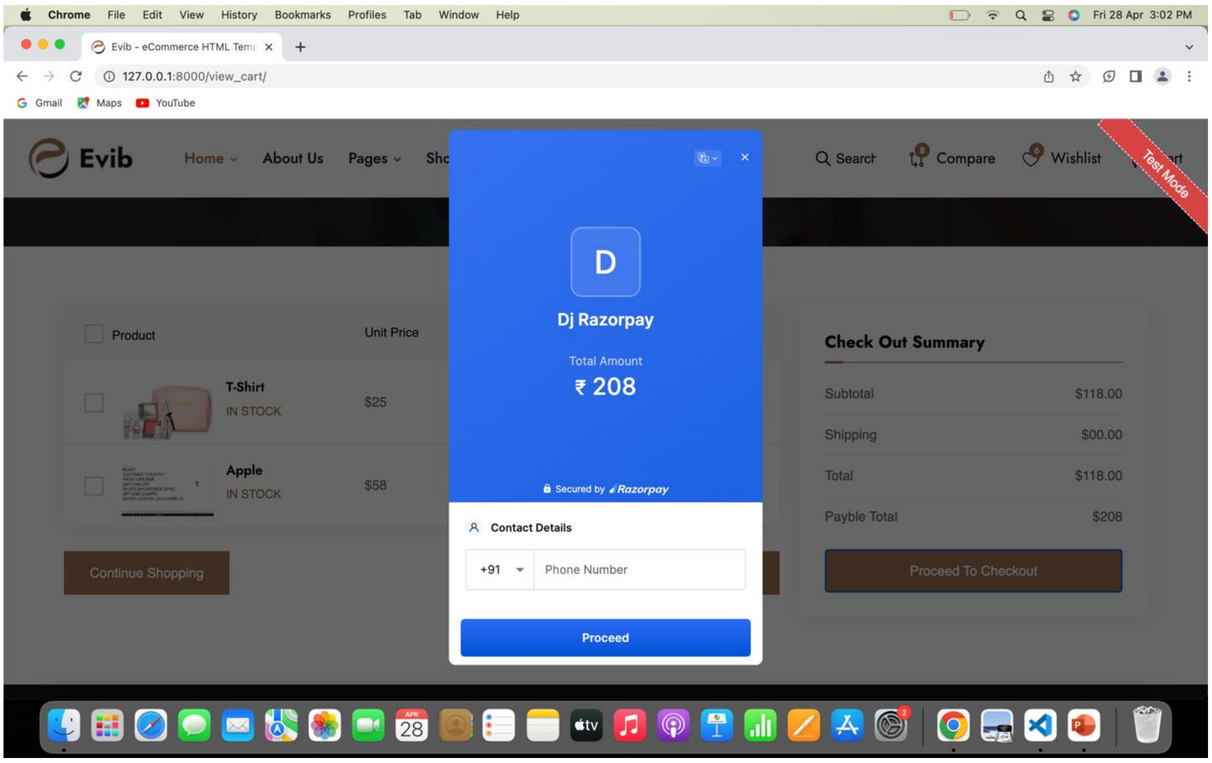
Update Cart

Check Out Summary

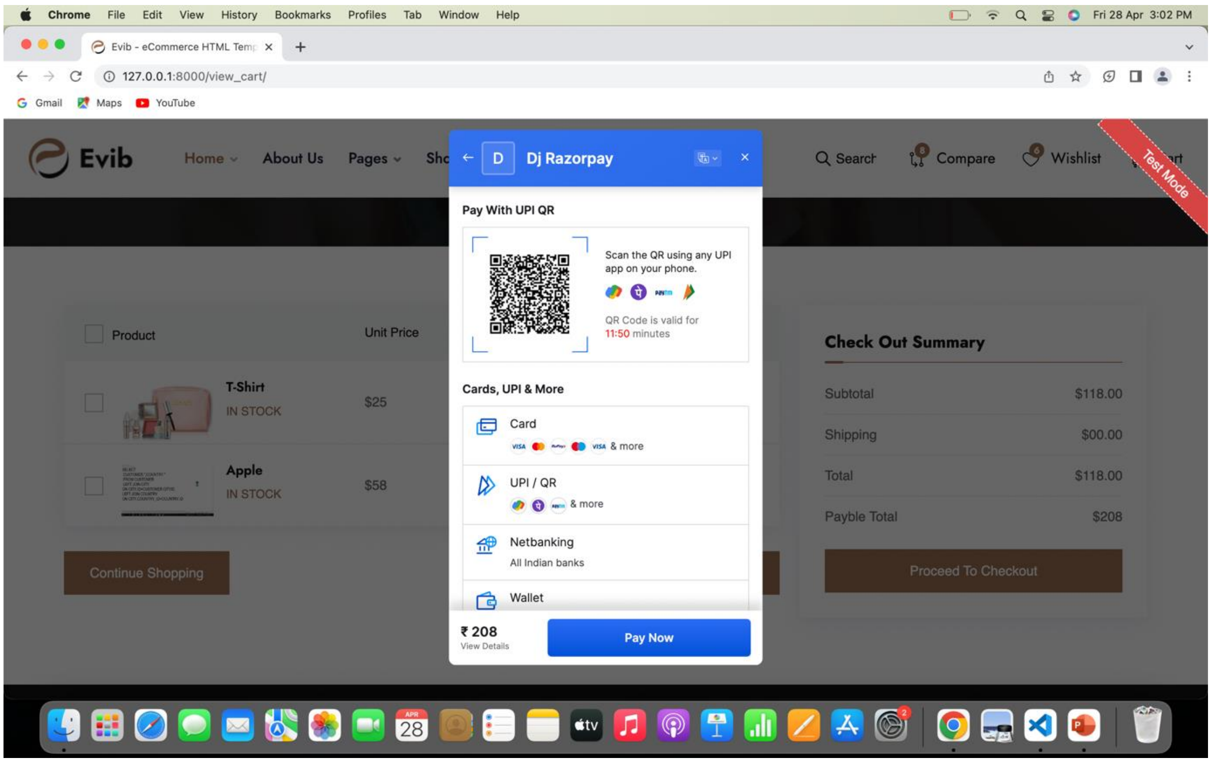
Subtotal	$118.00
Shipping	$00.00
Total	$118.00
Payble Total	$208

Proceed To Checkout

(Figure 5.5) Cart Page



(Figure 5.6)



(Figure 5.6) Payment Page

CHAPTER 6 CONCLUSION

I wanted to take a moment to express my sincere appreciation for the opportunity to work as a Web Developer Internship at Tops technologies. It has been an enriching and rewarding experience to be a part of the team and learn from some of the best professionals in the industry.

During my internship, I had the chance to work on several projects that have allowed me to hone my skills and knowledge of Web development. I also had the chance to work with a team of talented developers and designers, who have been instrumental in my growth as a developer.

I am grateful for the support, mentorship, and guidance that you and the team provided me throughout my internship. Your expertise and insights have been invaluable in helping me become a better developer. I will carry these learnings with me as I move forward in my career.

- **REFERANCE**

https://www.python.org

https://www.geeksforgeeks.org/python-oops-concepts/

https://www.djangoproject.com

https://github.com/PipedreamHQ/pipedream/tree/master/components/github?gclid=EAIaIQobChMIvuzCnZjl_gIVB38rCh3htws7EAAYASAAEgInZ_D_BwE#github-api-integration-platform

https://www.geeksforgeeks.org/e-commerce-website-using-django/

https://www.geeksforgeeks.org/django-project-creating-a-basic-e-commerce-website-for-displaying-products/