# P. E. S. COLLEGE OF ENGINEERING, MANDYA - 571 401

**(An Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)**

****



**Mini Project Report On**

## “INSTAGRAM CLONE MASTER”

Submitted in partial fulfillment for the award of Degree of

## BACHELOR OF ENGINEERING IN

**INFORMATION SCIENCE AND ENGINEERING**

Submitted by

CHINMAYI K S 4PS22IS012 JYOTHSNA SHIVANYA K S 4PS22IS024

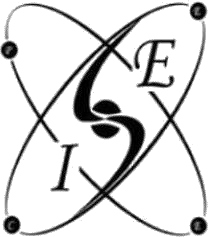
M J REETHU 4PS22IS032

MONISHA D R 4PS22IS036

VARSHINI M R 4PS22IS036

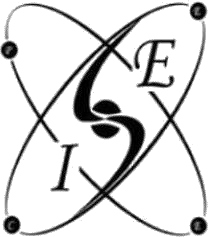
**Under the guidance of Dr. Bramesh S M** Associate Professor Dept. of IS&E

PESCE, Mandya



## DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

#### P. E. S. College of Engineering, Mandya – 571 401 2024 – 2025



**P.E.S. COLLEGE OF ENGINEERING, MANDYA - 571 401**

**(An Autonomous Institution Affiliated to VTU, Belagavi) DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

**CERTIFICATE**

This is to certify that**,**

**Chinmayi K S [4PS22IS012], Jyothsna Shivanya K S [4PS22IS024], M J Reethu [4PS22IS032], Monisha D R [4PS23IS036] ,Varshini M R [4PS22IS056]** has satisfactorily completed the Mini Project work **[P22ISMP607]** entitled “**INSTAGRAM CLONE MASTER”** carried out at **“P. E. S. College of Engineering, Mandya”** in partial fulfillment for the award of degree of Bachelor of Engineering in Information Science and Engineering of P. E. S. College of Engineering (An Autonomous Institution Affiliated to VTU, Belagavi). It is certified that all corrections indicated in the internal assessment have been incorporated in the report deposited in the library. The Mini Project work has been approved as it satisfies the academic requirements in respect of Mini Project work **[P22ISMP607]** prescribed for the degree in Bachelor of Engineering.

.

**Signature of the Guide Head of the Department**

Dr. Bramesh S M Dr. Minavathi

Associate Professor Professor & Head

Dept. of IS&E Dept. of IS&E

**Dr. N L Murali Krishna**

Principal, PESCE, Mandya

|  |  |  |  |
| --- | --- | --- | --- |
| **Details of Mini Project [P22ISMP607] Viva Voce Examination held** | | | |
| **Sl.**  **No.** | **Examiners** | | **Date** |
| **Name** | **Signature** |
| **1.** |  |  |  |
| **2.** |  |  |

## ACKNOWLEDGMENT

We dedicate this page to acknowledge and thank those who have always supported this work, without their guidance and help, the experience would not have been so smooth and efficient.

We are extremely thankful to our Principal **Dr. N L Murali Krishna,** for his support and encouragement.

We sincerely thank **Dr. Minavathi,** Head of the Information Science and Engineering Department, for her guidance and valuable suggestions which helped us to complete our mini project work.

We extend heartfelt gratitude to **Mr. Shankar,** Web Developer, Zestech Global Pvt. Ltd, for his guidance, constant supervision as well as providing us with the necessary information regarding the project, without his willing disposition, the spirit of accommodation, frankness, timely clarification and above all faith in us, this project could not have been completed in due time. His readiness to discuss all important matters at work deserves special attention.

We also extend heartful gratitude to our Guide **Dr. Bramesh S M,** Associate Professor, Department of Information Science and Engineering, for his coordination and helpful suggestions. We also like to extend heartily gratitude and thanks to teaching and non-teaching staff of the Information Science and Engineering Department for their continuous support.

We also like to extend heartily gratitude and thanks to my friends and family members for their continuous support.

## ABSTRACT

In today’s digital age, an effective online presence is vital for companies to communicate with stakeholders, manage internal operations, and streamline recruitment processes. This project presents the design and development of a **corporate web portal for Zentrixon Technologies,** built using the Django framework along with HTML and CSS for frontend development.

The portal is designed to serve two main user groups: **public visitors** and **company employees**. Visitors can explore available job roles, learn about the organization’s leadership team, and get in touch through a dedicated contact section. The **Careers module** displays current openings like Frontend Developer and AI/ML Engineer with detailed role descriptions, encouraging skilled professionals to apply. The **Our Team module** showcases key personnel with their roles and photos, adding credibility and a human touch to the organization’s online image.

This mini project not only enhanced our practical skills in web development but also fostered creative thinking and teamwork. This project emphasizes **user-friendly interface design**, **functional clarity,** and **modular development practices**, making it a potential base for a real-time company portal. It can further be integrated with authentication, real-time notifications, and admin dashboards for more advanced enterprise-level applications.

## TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **SL.NO.** | **TITLE** | **PAGE NO.** |
| 1 | INTRODUCTION | 1 |
| 2 | SYSTEM ANALYSIS | 2-3 |
| 3 | REQUIREMENTS SPECIFICATION | 4-5 |
| 4 | SYSTEM DESIGN | 6-8 |
| 5 | SYSTEM IMPLEMENTATION | 9-13 |
| 6 | SAMPLE OUTPUT | 14 |
| 7 | SYSTEM TESTING | 15 |
| 8 | CONCLUSION | 16 |
| 9 | BIBLIOGRAPHY | 17 |

CHAPTER 1

# INTRODUCTION

In the digital era, web applications play a crucial role in enabling companies to showcase their brand, manage internal processes, and engage with stakeholders. With increasing demand for efficient communication and automation within organizations, having a centralized and functional web portal has become a necessity. This project is a simulated corporate web portal developed for **Zentrixon Technologies**, a fictional tech company, using the **Django web framework**, supported by **HTML** and **CSS** for the front-end design.

The web application provides a seamless interface where visitors can explore career opportunities, learn about the company’s leadership, and get in touch through an integrated contact section. At the same time, it serves as a centralized hub for employees to manage essential work-related activities such as tracking leave status, accessing payroll information, and viewing performance reviews. By incorporating these features, the project mirrors the structure of a typical enterprise-level platform..

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It follows the Model-View-Template (MVT) architecture and comes with built-in features such as user authentication, ORM-based database interaction, URL routing, and admin interface. Its robustness and scalability make it a preferred choice for developing secure and maintainable web applications.

Overall this project not only demonstrates technically but also it serves as a strong foundation for further expansion into a fully functional organizational portal, capable of supporting real-time user interactions and administrative control.

CHAPTER 2

# SYSTEM ANALYSIS

### Proposed System

The proposed system is a dynamic, database-driven web portal for Zentrixon Technologies*,* developed using Django, HTML, and CSS. It serves both external users—by providing access to career opportunities, team information, and contact details—and internal employees, who can manage leaves, view payroll, and check performance reviews.

#### Scope of the Project

 To develop a web portal using Django, HTML, and CSS.

 To display company details like careers, team members, and contact info.

 To create a functional employee portal for internal use.

 To include features like leave management, payroll, and performance review.

#### 2.1.2 Aim of the Project

 To design a corporate web portal for Zentrixon Technologies*.*

 To implement the project using Django, HTML, and CSS.

 To showcase company information like careers and team profiles.

 To build an internal system for employees to manage work-related tasks.

 To create a responsive, user-friendly, and interactive web application.

#### 2.1.1.3 Project Modules

The project consists of the following main modules:

### 1. ****Homepage Module****

* Introduces the company with a professional layout.
* Welcomes users and links to key sections of the website.

### 2. ****Careers Module****

* Displays current job openings with roles and descriptions.
* Encourages candidates to apply or learn more about the company.

### 3. ****Our Team Module****

* Showcases team members with their names, roles, and images.
* Enhances the company’s credibility and human connection.

### 4. ****Contact Module****

* Provides company contact information including email and phone.
* Serves as a communication bridge between users and the organization.

### 5. ****Employee Portal Module****

* Accessible to employees for internal use.
* Contains sub-modules:
  + **Leave Management**: Employees can view or manage leave records.
  + **Payroll View**: Allows employees to check their salary or payment details.
  + **Performance Review**: Displays employee evaluations and feedback.

.

CHAPTER 3

# Requirement Specifications

This chapter describes the software tools and system requirements needed to develop and run the Zentrixon Technologies Corporate Portal mini project.

### Details of Software

#### Frontend Technologies

* + - 1. **HTML** – is used to create the structure and layout of each webpage. It defines the elements such as headings, paragraphs, buttons, forms, and navigation menus across different pages like Home, Careers, Team, and the Employee Portal.
      2. **CSS** – CSS is used to control the visual appearance and formatting of HTML elements. In this project, CSS was applied to enhance the look and feel of the application, It controls the color schemes, fonts, spacing, layout, and responsiveness of the portal formatting, and overall page responsiveness. By using CSS, the user interface was made more attractive and user-friendly without relying on external frameworks like Bootstrap.

#### Backend Technologies

* + - 1. **Python** – Main programming language used.
      2. **Django** – Web framework for building the application.
      3. **SQLite** – Default database used in Django for storing data (like users, posts, comments).

### System Requirements

#### Hardware Configuration

* + - 1. **Processor:** Intel Core i3 or above
      2. **RAM:** Minimum 4 GB
      3. **Storage:** At least 500 MB free space

#### Software Configuration

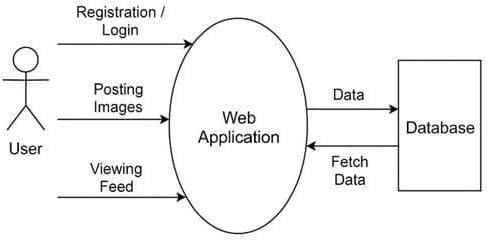
* + - 1. **Operating System:** Windows 10 / 11, Linux, or macOS
      2. **Python Version:** Python 3.8 or higher
      3. **Django Version:** Django 3.x or above
      4. **Text Editor/IDE:** VS Code
      5. **Browser:** Google Chrome / Firefox

CHAPTER 4

# SYSTEM DESIGN

System design helps to visualize how data flows and how different components of the system interact with each other. It provides a blueprint for building the application.

#### Data Flow Diagram

****

**Fig.4.1:Data Flow Diagram**

The **Data Flow Diagram** shows how data moves through the Instagram Clone system.

**Users** interact with the system in the following ways:

#### Registration/Login:

* + User provides credentials.
  + System checks in the database.

1. **Career Module**

* Input: User requests job openings
* Process: System retrieves job listings
* Output: List of available roles

**iii)** **Team Module**

* Input: Request for team info
* Process: Fetch team members from DB
* Output: Display team details

**iv)** **Leave Management**

* Input: Employee submits/view leave
* Process: Update or fetch leave records
* Output: Confirmation or status

**v)** **Payroll View**

* Input: Employee requests payroll info
* Process: Retrieve payroll data
* Output: Display salary details

**vi) Performance Review**

* Input: Request for feedback
* Process: Fetch reviews from DB
* Output: Show performance data

**viii) Contact Module**

* Input: Request contact details
* Output: Display email, phone

#### Sequence Diagram

The Sequence Diagram shows the step-by-step interaction between the user and the system over time for common actions.

#### User Login Sequence

**User → Browser → Login Page → Django View → Model → Database → Template**

 **User** sends a request for the Careers page.

 **Browser** forwards the request to the Django server.

 **Django View** processes the request and queries the Model.

 **Model** interacts with the **Database** to retrieve job listings.

 Retrieved data is returned from **Database → Model → View**.

 The **View** renders the data using an HTML **Template**.

 **Template** is sent back to the **Browser**, and the user sees the result.

CHAPTER 5

# SYSTEM IMPLEMENTATION

System implementation is the process of converting the design into a working system. The Instagram Clone Master project was implemented using the Django framework. Each part of the system was developed as a module to handle specific tasks like user login, posting images, liking posts, and commenting.

#### 5.1 Modular Description

1. **User Authentication Module**

This module is responsible for handling user registration, login, and logout functionalities. It ensures that only authenticated users can access core features such as posting images, liking content, and commenting. Django's built-in authentication system is used to securely manage user credentials, hash passwords, handle session tracking, and enforce access control. This provides a secure and efficient method for user management and safeguards the platform from unauthorized access.

Code for this module is given below:

class Profile(models.Model):

user= models.OneToOneField(User, related\_name='profile', on\_delete=models.CASCADE) image = models.ImageField(upload\_to="profile\_pciture", null=True, default="default.jpg") first\_name = models.CharField(max\_length=200, null=True, blank=True)

last\_name = models.CharField(max\_length=200, null=True, blank=True) bio = models.CharField(max\_length=200, null=True, blank=True) location = models.CharField(max\_length=200, null=True, blank=True) url = models.URLField(max\_length=200, null=True, blank=True) favourite = models.ManyToManyField(Post, blank=True)

#### Post Upload Module

This module allows logged-in users to upload image-based posts along with optional captions. The uploaded media is stored securely and associated with the user who submitted it. This module includes backend validation for image formats and manages file storage paths using Django’s media handling capabilities. Once uploaded, the post becomes publicly visible to all users and is displayed within the main content area of the application and the user’s profile page.

Code for this module is given below:

from django import forms from post.models import Post

class NewPostform(forms.ModelForm):

# content = forms.FileField(widget=forms.ClearableFileInput(attrs={'multiple': True}), required=True)

picture = forms.ImageField(required=True)

caption = forms.CharField(widget=forms.TextInput(attrs={'class': 'input', 'placeholder': 'Caption'}), required=True)

tags = forms.CharField(widget=forms.TextInput(attrs={'class': 'input', 'placeholder': 'Tags | Seperate with comma'}), required=True)

class Meta:

model = Post

fields = ['picture', 'caption', 'tags']

#### Like Module

The Like Module enables interactive engagement by allowing users to like posts shared by others. It ensures that each user can like a post only once, preventing repeated actions that could distort the like count. The system updates the total likes dynamically and provides real-time visual feedback, such as toggling the like icon. This feature encourages content appreciation and increases user interaction within the platform.

Code for this module is given below: class Likes(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

post=\models.ForeignKey(Post,\on\_delete=models.CASCADE, related\_name="post\_likes")

def user\_liked\_post(sender, instance, \*args, \*\*kwargs):

like = instance post = like.post sender = like.user

notify = Notification(post=post, sender=sender, user=post.user) notify.save()

def user\_unliked\_post(sender, instance, \*args, \*\*kwargs):

like = instance post = like.post sender = like.user

notify = Notification.objects.filter(post=post, sender=sender, notification\_types=1) notify.delete()

#### Comment Module

This module supports textual interaction by allowing users to add comments to individual posts. Each comment is linked to both the post and the user who created it. It includes input validation, stores the comment in the database, and displays all associated comments below each post. The comment module enhances communication among users and adds a layer of social interactivity to the platform.

Code for this module is given below:

class Comment(models.Model):

post= models.ForeignKey(Post, on\_delete=models.CASCADE, related\_name="comment") user = models.ForeignKey(User, on\_delete=models.CASCADE)

body = models.TextField()

date = models.DateTimeField(auto\_now\_add=True, null=True)

def user\_comment\_post(sender, instance, \*args, \*\*kwargs):

comment = instance post = comment.post

text\_preview = comment.body[:90] sender = comment.user

notify = Notification(post=post, sender=sender, user=post.user, text\_preview=text\_preview, notification\_types=2)

notify.save()

def user\_del\_comment\_post(sender, instance, \*args, \*\*kwargs): comment = instance

post = comment.post sender = comment.user

notify=Notification.objects.filter(post=post,sender=sender,user=post.user, notification\_types=2)

notify.delete()

post\_save.connect(Comment.user\_comment\_post, sender=Comment) post\_delete.connect(Comment.user\_del\_comment\_post, sender=Comment)

#### Profile Module

The Profile Module provides a personalized space for each user, showcasing their uploaded posts, username, and other optional details like bio and profile picture. It allows users to view their own activity and lets others explore public profiles. This module fetches user-specific content based on session data and offers a structured layout for displaying personal content, helping users manage and present their identity on the platform.

Code for this module is given below:

def UserProfile(request, username): Profile.objects.get\_or\_create(user=request.user)

user = get\_object\_or\_404(User, username=username) profile = Profile.objects.get(user=user)

url\_name = resolve(request.path).url\_name

posts = Post.objects.filter(user=user).order\_by('-posted')

if url\_name == 'profile':

posts = Post.objects.filter(user=user).order\_by('-posted') else:

posts = profile.favourite.all() # Profile Stats

posts\_count = Post.objects.filter(user=user).count() following\_count = Follow.objects.filter(follower=user).count() followers\_count = Follow.objects.filter(following=user).count() # count\_comment = Comment.objects.filter(post=posts).count()

follow\_status = Follow.objects.filter(following=user, follower=request.user).exists() # pagination

paginator = Paginator(posts, 8) page\_number = request.GET.get('page')

posts\_paginator = paginator.get\_page(page\_number) context = {

'posts': posts, 'profile':profile, 'posts\_count':posts\_count,

'following\_count':following\_count, 'followers\_count':followers\_count, 'posts\_paginator':posts\_paginator, 'follow\_status':follow\_status,

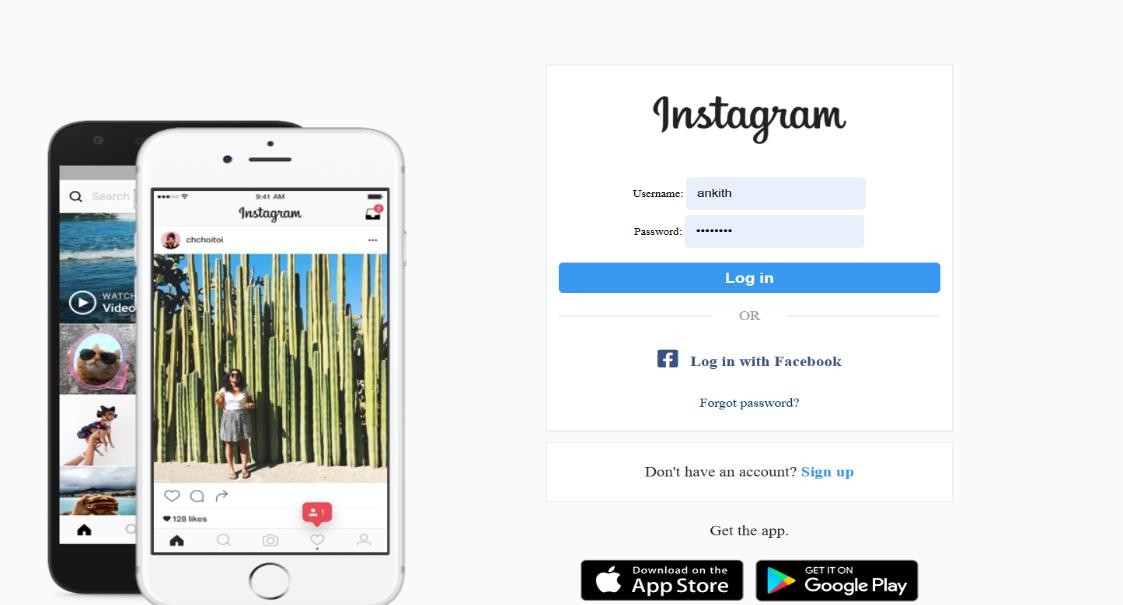
# 'count\_comment':count\_comment,

}

return render(request, 'profile.html', context)

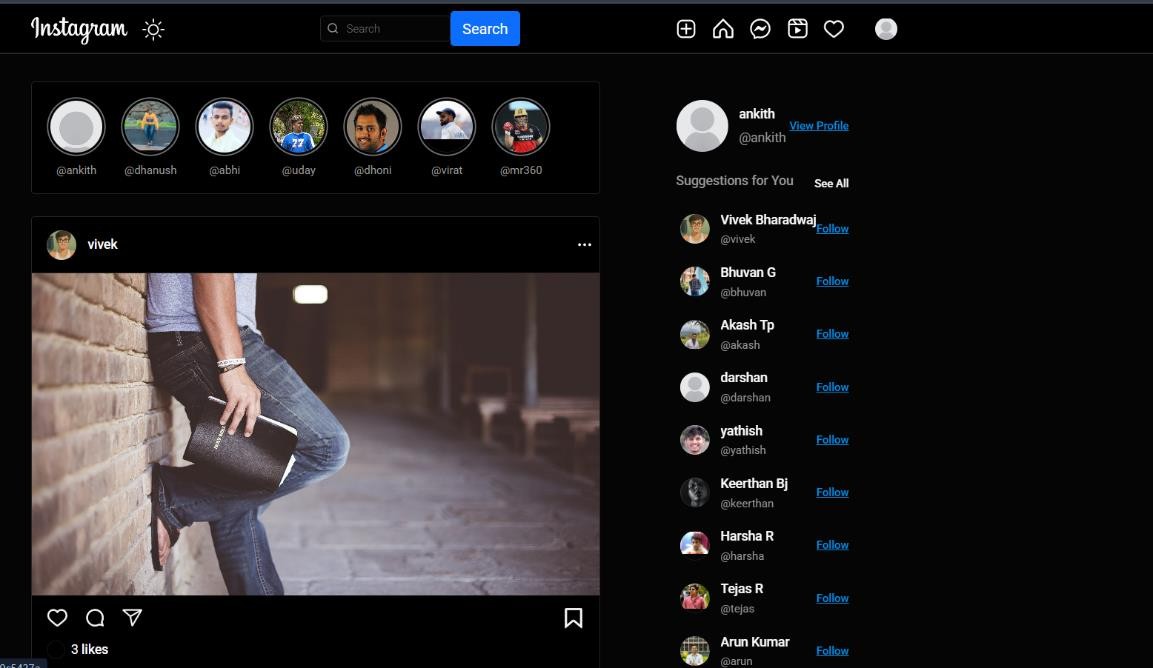
CHAPTER 6

# SAMPLE OUTPUT

****

#### Fig.6.1 Login page

This page allows users to securely log in to their accounts using their registered username and password.



#### Fig.6.2: Home Page

This page displays the news feed with posts from all users, including images, captions, likes, and comments.

CHAPTER 7

# SYSTEM TESTING

System testing is done to check whether the complete project works as expected. It helps in finding and fixing any errors before the project is finalized. The Instagram Clone Master project was tested module by module to ensure all functions perform correctly.

#### Types of Testing Performed:

#### ****1. Functional Testing****

* Checked if all pages (Home, Careers, Team) load without errors.
* Verified that job roles (e.g., Frontend Developer, AI/ML Engineer) appear correctly on /careers/.
* Confirmed team member details are displayed on /team/.
* Tested Logout functionality – redirects user and clears session.
* Validated theme switch (dark/light mode) changes UI accordingly.

#### ****2. Integration Testing****

* Ensured views (views.py) pass correct data to templates.
* Confirmed proper routing using urls.py for all pages.
* Checked template rendering using Django's {{ variable }}.
* Verified static files (CSS/images) load without issues.

.

#### ****3. Authentication Testing****

* Logout button successfully ends user session.
* Restricted access (like Employee Portal) should redirect unauthenticated users (if implemented).
* Session-based login/logout flow works correctly.

#### Result:

All modules were manually tested successfully. The system worked as expected without any major errors. Minor layout issues were corrected during testing.

# CONCLUSION

The **Zentrixon Technologies Careers and Team Portal** project is a successful implementation of a dynamic web-based platform built using the Django framework. The main objective of the project — to create a structured and interactive portal for career opportunities and team introductions — has been effectively achieved. This application serves as both a promotional and functional tool for the organization, enhancing its online presence and recruitment process.

Through this project, we developed a responsive and user-friendly interface that allows potential candidates to explore available job roles in various domains such as **Frontend Development** and **AI/ML Engineering**. The **Careers** page is designed to be informative and easily navigable, displaying available positions in a clean card layout with brief role descriptions. This enables users to quickly understand what roles are open and which ones suit their interests..

The system has undergone **extensive system and unit testing** to ensure functionality, performance, and reliability. Features such as **navigation, page rendering,** and **logout functionality** were tested and verified. The result is a robust and error-free application ready for deployment or expansion.

Furthermore, this project has given valuable insights into the practical implementation of web development concepts, including user interface design, backend integration, routing mechanisms, and modular code practices in Django. It demonstrates the capability to build real-world applications that are scalable, maintainable, and user-centric.

.

BIBLIOGRAPHY

1. <https://www.gti.bh/Library/assets/djangobookwzy482.pdf>
2. <https://www.gti.bh/Library/assets/djangobookwzy482.pdf>
3. <https://youtu.be/qw5ZEvylQBA?si=4Vz8Hjp8wbNAEJ56>
4. <https://www.youtube.com/watch?v=2702kd5aSRQ>
5. <https://www.w3schools.com/django/>
6. <https://www.geeksforgeeks.org/django-tutorial/>