CHAPTER 1

INTRODUCTION

The CLASSIC BLOG is Personal Blog Management System. A blog is an online journal or informational website displaying information in the reverse chronological order, with latest posts appearing first. It is a platform where a writer or even a group of writers share their views on an individual subject. This project is designed for successful completion of project on personal blog management system. We intend to share short, summarized content relevant to Hardware, Software, Web technologies, Algorithms, Neural Networks, Competitive programming, programming language heads-up etc. The basic building aim is to provide a proper way to write a blog which will help their users to look upon. CLASSIC BLOG is a browser-based system which contains a separate admin panel where admin can manage to add, edit, delete posts along with images. Each user will have an option to post as many comments as he wants and each blog post will have a view-count too that will show the number of post visits.

1.1Need of Personal Blog

Importance of blogging is huge on the Internet. A blog is useful for many reasons. It can become a source of leads, it can be the place where, in the future, you might sell your products if you want to become an independent developer, or it can simply be the place where you have your audience and express your ideas. Blogging helps programmers to connect with students. The more frequent and better your blog posts are, the higher the chances for your website to get discovered and visited by your target audience. Which means, a blog is an effective lead generation tool.

A CLASSIC BLOG can help them in following ways:

- Users will be able to view relevant contents.
- Admin can easily manage the posts from dashboard section.
- Users can share their views through comments on each post.
- Users can contact Admin and give their valuable feedback.

CHAPTER 2

REQUIREMENT SPECIFICATION

A high-level requirements specification is required. The purpose of the requirements analysis is to identify requirements for the proposed system. The emphasis is on the discovery of user requirements.

2.1 SOFTWARE REQUIREMENTS

Operating System : Windows 10 or any compatible operating system.

Database : MySQL

Tools : WAMP Server, Visual Studio Code

2.2 HARDWARE REQUIREMENTS

Processor : Any Processor above 500 MHz

RAM : 4GB

Hard Disk : 2 GB free space

Input device : Keyboard, Mouse

Output device : Monitor

System type : 32-bit or 64-bit operating system

2.3 FUNCTIONAL REQUIREMENTS

Home page: Home page is the main page of the project. It contains various posts along with their titles and cover images.

Admin-Module: In this page, admin can add, edit or delete posts according to his needs.

.

2.4 NON-FUNCTIONAL REQUIREMENTS:

PERFORMANCE:

Performance requirements define acceptable response times for system functionality.

- The load time for user interface screens shall take no longer than five seconds.
- The log in information shall be verified within five seconds.
- Queries shall return results within five seconds.

RELIABILITY:

- Good validations for user inputs will be done.
- Avoid incorrect storage of records.

FLEXIBILITY:

- The system keeps on updating the data according to the new articles that takes place.

MAINTAINABILITY:

- During maintenance stage, the SRS can be referred for the validation.

TIMELINESS:

- The system carries out all the operations with consumptions of very less time.

CHAPTER 3

OBJECTIVE OF THE PROJECT

The main objective of this proposed system is to make a platform mainly for coders where one can find articles which can be read by the user and based upon their understanding of the article, they can comment on them as they like. There is a separate section for feedback where the users can post their ideas and reviews or complaints. The articles can be posted only by the admin and not by the users. The users can recommend any article of their interest to the admin. The admin can then take those topics in consideration and can later post the detailed article on the topic requested by the users.

We want to build relationships with new online connections and we want to strengthen relationships with existing users. Our ultimate goal is to become more and more successful through our blog and how effectively it touches the users.

CHAPTER 4

SYSTEM DESIGN

4.1 FLOW OF WEB PAGES

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. Figure 4.1 below shows the use case diagram for this website.

User and Administrator are the two actors included in the Classic Blog.

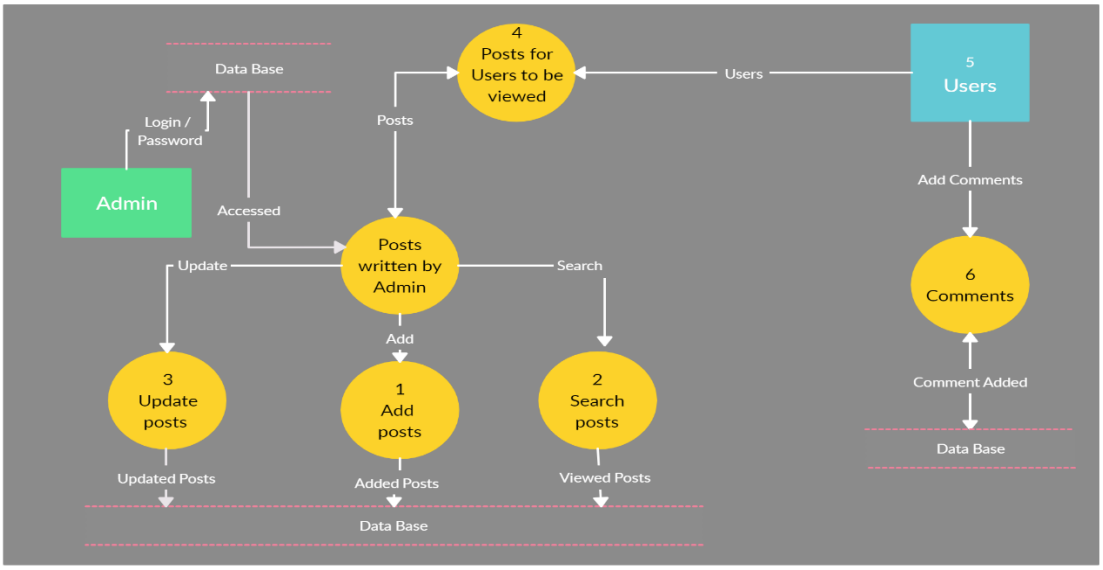

Figure 4.1: Flow of Web Pages

4.2 ENTITY RELATIONSHIP DIAGRAM

The entity-relationship data model is based on a perception of a real world that consists of a collection of basic objects called entities and of relationships among these objects. An entity is an "object" in the real world that is distinguishable from other objects. For e.g. each customer is an entity and rooms can be considered to be entities. Entities are described by a set of attributes. Figure 4.2 Shows the Entity Relationship between the tables.

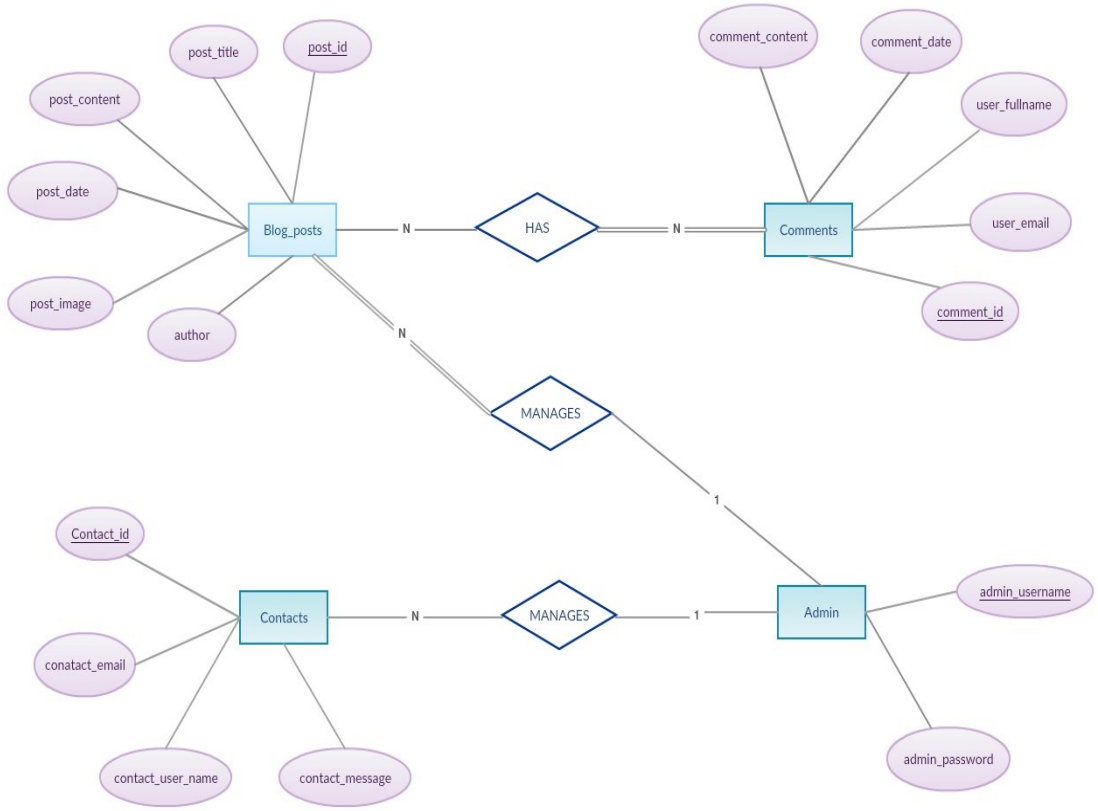

Figure 4.2: Entity Relationship Diagram

CHAPTER 5

IMPLEMENTATION

5.1 SOURCE CODE

main.py (Flask integration)

```
from flask import Flask, render_template, request, session, redirect, flash
from flask_sqlalchemy import SQLAlchemy
from werkzeug import secure_filename
import json
import os
import math
from datetime import datetime


with open('config.json', 'r') as c:
    params = json.load(c)["params"]

app = Flask(__name__)
app.secret_key = 'dont-tell-anyone'
app.config['UPLOAD_FOLDER'] = params['upload_location']
ALLOWED_EXTENSIONS = set(['jpg', 'jpeg'])
local_server = True
if(local_server):
    app.config['SQLALCHEMY_DATABASE_URI'] = params['local_uri']
else:
    app.config['SQLALCHEMY_DATABASE_URI'] = params['prod_uri']
db = SQLAlchemy(app)

class Contacts(db.Model):
    sno = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)
    phone_num = db.Column(db.String(12), nullable=False)
    msg = db.Column(db.String(120), nullable=False)
    date = db.Column(db.String(12), nullable=True)
    email = db.Column(db.String(20), nullable=False)

class Posts(db.Model):
    sno = db.Column(db.Integer, primary_key=True)
```

```
    title = db.Column(db.String(80), nullable=False)
    slug = db.Column(db.String(40), nullable=False)
    content = db.Column(db.Text, nullable=False)
    author = db.Column(db.String(120), nullable=False)
    date = db.Column(db.String(12), nullable=True)
    img_file = db.Column(db.String(12), nullable=True)
    countcomm=db.Column(db.Integer, default=0)
    views=db.Column(db.Integer, default=0)

class Comments(db.Model):
    cid = db.Column(db.Integer, primary_key=True)
    postid = db.Column(db.Integer, db.ForeignKey('posts.sno'), nullable=False)
    commentdate = db.Column(db.DateTime, nullable=True, default=datetime.now)
    name = db.Column(db.String(50), nullable=False, unique=True)
    emailid = db.Column(db.String(65), nullable=False, unique=False)
    message = db.Column(db.String(550), nullable=False)


@app.route("/")
def home():
    posts = Posts.query.filter_by().order_by(Posts.sno.desc()).all()
    last = math.ceil(len(posts)/int(params['no_of_posts']))
    page = request.args.get('page')

    if(not str(page).isnumeric()):
        page = 1

    page= int(page)
    posts = posts[(page-1)*int(params['no_of_posts']):
                (page-1)*int(params['no_of_posts'])+ int(params['no_of_posts'])]

    if (page==1):
        prev = "/?page="+ str(page+1)
        next = "#"
    elif(page==last):
        prev = "#"
        next = "/?page=" + str(page - 1)
    else:
        prev = "/?page=" + str(page + 1)
        next = "/?page=" + str(page - 1)
    return render_template('index.html', params=params, posts=posts, prev=prev,
next=next)
```

```
@app.route("/post/<string:post_slug>", methods=['GET','POST'])
def post_route(post_slug):
    post = Posts.query.filter_by(slug=post_slug).first()
    cocomment=Comments.query.filter_by(postid=post.sno).all()
    post.views += 1
    db.session.commit()
    if request.method == 'POST':
        name=request.form.get('name')
        emailid=request.form.get('emailid')
        message=request.form.get('message')
        comments=Comments(name=name, emailid=emailid, message=message,
postid=post.sno)
        db.session.add(comments)
        post.countcomm += 1
        db.session.commit()
    return render_template('posts.html', params=params, post=post,
cocomment=cocomment)

@app.route("/about")
def about():
    return render_template('about.html', params=params)

@app.route("/dashboard", methods=['GET', 'POST'])
def dashboard():
    if ('user' in session and session['user'] == params['admin_user']):
        posts = Posts.query.all()
        return render_template('dashboard.html', params=params, posts = posts)

    if request.method=='POST':
        username = request.form.get('uname')
        userpass = request.form.get('pass')
        if (username == params['admin_user'] and userpass == params['admin_password']):
            #set the session variable
            session['user'] = username
            posts = Posts.query.all()
            flash("LogIn Successful")
            return render_template('dashboard.html', params=params, posts = posts)

    return render_template('login.html', params=params)

@app.route("/edit/<string:sno>", methods = ['GET', 'POST'])
def edit(sno):
    if ('user' in session and session['user'] == params['admin_user']):
```

```
        if request.method == 'POST':
            box_title = request.form.get('title')
            postedby = request.form.get('postedby')
            slug = request.form.get('slug')
            content = request.form.get('content')
            img_file = request.form.get('img_file')
            date = datetime.now()

            if sno=='0':
                post = Posts(title=box_title, slug=slug, content=content, author=postedby,
img_file=img_file, date=date)
                db.session.add(post)
                db.session.commit()
            else:
                post = Posts.query.filter_by(sno=sno).first()
                post.title = box_title
                post.slug = slug
                post.content = content
                post.author = postedby
                post.img_file = img_file
                post.date = date
                db.session.commit()
                return redirect('/edit/'+sno)

        post = Posts.query.filter_by(sno=sno).first()
        return render_template('edit.html', params=params, post=post, sno=sno)

@app.route("/uploader", methods = ['GET', 'POST'])
def uploader():
    if ('user' in session and session['user'] == params['admin_user']):
        if (request.method == 'POST'):
            f= request.files['file1']
            f.save(os.path.join(app.config['UPLOAD_FOLDER'], secure_filename(f.filename)
))
            return redirect('/index')
        else:
            return redirect('/dashboard')
    return redirect('/dashboard')

@app.route("/logout")
def logout():
    session.pop('user')
    return redirect('/dashboard')
```

```
@app.route("/delete/<string:sno>", methods = ['GET', 'POST'])
def delete(sno):
    if ('user' in session and session['user'] == params['admin_user']):
        post = Posts.query.filter_by(sno=sno).first()
        db.session.delete(post)
        db.session.commit()
    return redirect('/dashboard')

@app.route("/contact", methods = ['GET', 'POST'])
def contact():
    if(request.method=='POST'):
        name = request.form.get('name')
        email = request.form.get('email')
        phone = request.form.get('phone')
        message = request.form.get('message')
        entry = Contacts(name=name, phone_num = phone, msg = message, date=
datetime.now(),email = email )
        db.session.add(entry)
        db.session.commit()
        flash('Contact Submitted Successfully')
    return render_template('contact.html', params=params)

app.run(debug=True)
```

layout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>{{params.blog_name}}</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
    <link
href="https://fonts.googleapis.com/css?family=Aclonica|Do+Hyeon|Secular+One|Open+S
```

```
ans|Berkshire+Swash|Dosis|Fjalla+One|Varela|Varela+Round&display=swap"
rel="stylesheet">
    <script src="https://kit.fontawesome.com/d1e4d7425f.js"
crossorigin="anonymous"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
</head>
<body>
 <div class="wrapper">
   <header>
    <nav>
      <div class="logo"><a href="/">Classic <span>Blog</span></a></div>
     <div class="menu">
         <ul>
           <li><a href="/">Home</a></li>
           <li><a href="/about">About</a></li>
           <li><a href="/contact">Contact</a></li>
         </ul>
       </div>
     </nav>
   </header>
 </div>

         {% block body %}

  {% endblock %}
<script>
     $(window).on("scroll", function() {
         if($(window).scrollTop()) {
             $('nav').addClass('black');
         }
         else {
             $('nav').removeClass('black');
         }
     })
 </script>
 <div id="footer">
   <div>Coded by <a href="https://www.linkedin.com/in/i-ashu" target="_blank"
style="color: #e4cb58;text-decoration: none;">Aashu | Amrit</a></div>
   <div>Inspired from <a href="#" target="_blank" style="color: #e4cb58;text-decoration:
none;">Medium</a></div>
   <div>Made For <a href="https://www.gat.ac.in" target="_blank" style="color: #e4cb58;
text-decoration: none;"> GAT</a></div>
```

```
  </div>
</body>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script src="https://code.jquery.com/jquery-3.3.1.js"></script>
</html>
```

Index.html

```
{% extends "layout.html" %}
{% block body %}
<br>
<aside>
        <br><br>
        <form class="searchme" method="Post" action="/search">
    <div class="search">
     <h4>Search</h4>
     <input type="text" id="searchbox" name="searchbox" id="searchbox"
placeholder="Search here..."><br>
     <button class="btn btn-primary" value="submit"><i class="fas fa-search"></i>
Search</button>
     </div>
  </form>
   </aside>
        <div class="cool-card">
                        <div class="card-column-next">
            {%for post in posts%}
                    {% set fname = 'posts_img/' + post.img_file %}
                            <article class="card">
                          <a href="/post/{{post.slug}}" class="card-image" style="
background-image: url('{{ url_for('static', filename=fname) }}')"></a>
                             <section class="card-content1">

                               <h2 class="card-body" style="font-family: sans-serif;">
                                 <a href="/post/{{post.slug}}" style="text-decoration:
none">{{post.title}}</a>
                             </h2>

                             <p class="card-text"><small class="text">
                                                    <span style="color:
black;">Written by</span>
                                                                <a style="color:
black;font-size:12px;font-weight: bold;">{{post.author}}</a>  <i>|| </i><span
style="color: black;"> {{post.date.strftime('%b %d, %Y')}}</span>
                                                                <span
```

```
style="color: black;display: flex;float: right;"><i class="far fa-eye"> {{post.views}}</i>
     <i class="far fa-comment">
{{post.countcomm}}</i></span>
                                    </small></p>
                                   <section class="card-text1">
                                    <p>{{post.content[0:300] | safe}}...
                                    </p>
                                    <div class="post-footer">
                                     <!-- <p class="read-more"><a
href="https://www.aventr.com/blog">Read more...</a></p> -->
                                      <p><a href="/post/{{post.slug}}" class="btn btn-primary p-
3 px-xl-4 py-xl-2">Read More</a></p>
                                    </div>
                                   </section>
                                  </section>
                                 </article>
                                  {%endfor%}
          </div>
         </div>
         <div class="pagibuttons">
      <a href="{{prev}}"><button class="pagebutton">&lt;Older</button></a>
      <a href="{{next}}"><button class="pagebutton">Newer&gt;</button></a>
   </div>

   {% endblock %}
```

dashboard.html

```
{% extends "layout.html" %}
{% block body %}
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
   <!-- Page Header -->
    <div class="overlay">
    </div>
     <div class="container">
      <div class="row">
       <div class="col-lg-8 col-md-10 mx-auto">
        <div class="site-heading">
         <br><br><br>
                    {% with messages = get_flashed_messages() %}
          {% if messages %}
             {% for message in messages %}
               <div class="alert alert-success alert-dismissible show" role="alert">
                <strong>{{message}}</strong>
```

```
                <button type="button" class="close" data-dismiss="alert" aria-
label="Close">
                  <span aria-hidden="true">&times;</span>
                </button>
              </div>
            {% endfor %}
          {% endif %}
{% endwith %}
          <h1 style="text-align: center; font-size: 42px; font-weight: bold; ">Admin
Panel</h1>
          <hr><hr>
          <h1 class="subheading" style="text-align: center;">Manage Posts</h1>
          <hr>
         </div>
        </div>
      </div>
    </div>
  </header>
  <!-- Main Content -->
  <div class="container mx-auto">
    <div class="row">
      <div class="col-lg-10 col-md-10 mx-auto">
        <h1>Dashboard Actions</h1>

         <a href="/edit/0">  <button class="btn btn-primary"> Add a new post</button></a>
         <a href="/logout">  <button class="btn btn-primary"> Logout</button></a>
<hr>
        <h2>Upload Image</h2>

         <form action="/uploader" method="post" enctype="multipart/form-data">

<input type="file" accept="image/*" name="file1">
<button type="submit" class="btn btn-primary">Submit</button>

         </form>
        <hr>
<h2>Edit Posts</h2>
<table class="table">
  <thead>
    <tr>
      <th>Sno</th>
      <th>Title</th>
      <th>Author</th>
```

```
      <th>Date</th>
      <th>Image</th>
      <th>Edit</th>
      <th>Delete</th>
    </tr>
  </thead>
  <tbody>


    {% for post in posts %}
    {% set fname = 'posts_img/' + post.img_file %}
    <tr>
      <td>{{post.sno}}</td>
      <td>{{post.title}}</td>
      <td>{{post.author}}</td>
      <td>{{post.date}}</td>
      <td><img class="table" src="{{ url_for('static', filename=fname) }}" alt="Image
placeholder" style="width: 120px; height: 75px;"></td>
      <td><a href="/edit/{{post.sno}}"><button class="btn btn-primary"
>Edit</button></a></td>
      <td><a href="/delete/{{post.sno}}"><button class="btn btn-
primary">Delete</button></a></td>
    </tr>


    {% endfor %}

  </tbody>
 </table>
      </div>
    </div>
  </div>
{% endblock %}
```

Posts.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>{{params.blog_name}}</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
```

```
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
    <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
</head>
<body>
 <div class="wrapper">
   <header>
    <nav>
      <div class="logo"><a href="/">Classic <span>Blog</span></a></div>
     <div class="menu">
         <ul>
           <li><a href="/">Home</a></li>
           <li><a href="/about">About</a></li>
           <li><a href="/contact">Contact</a></li>
         </ul>
       </div>
     </nav>
   </header>
 </div>
<body>
  <header>
        <br><br>
  </header>
  <main class="theme-standard">
    <article class="blog-page">
      <h2 class="h2">{{post.title}}</h2>
      <p class="blog-content-date" style="font-size: 15px;">Posted by <span style="font-
weight: 500; color: black;">{{post.author}}</span> On <i style="color:
black;">{{post.date.strftime('%b %d, %Y')}}</i>
       <span style="display: flex; float: right; color: rgb(15, 11, 19); font-size: 17px; font-
weight: 600;"><i class="far fa-eye"> {{post.views}}</i>      <i
class="far fa-comment-dots"> {{post.countcomm}}</i></span></p>
      <hr>
      <section class="blog-content">
        {% set fname = 'posts_img/' + post.img_file %}
        <img class="content-img" src="{{ url_for('static', filename=fname) }}" alt="" >
        <p class="blog-content-p">{{post.content | safe}}
        </p>
```

```
        </section>
      </article>
      <hr>
      <!-- Comment view -->
    <div class="card card-comments mb-3 wow fadeIn">
    <div class="card-header font-weight-bold">{{post.countcomm}} Comments</div>
    {%for Comments in cocomment%}
    <div class="card-body">
            <div class="media d-block d-md-flex mt-3">
               <div class="media-body text-center text-md-left ml-md-3 ml-0">
                  <h5 class="mt-0 font-weight-bold">{{Comments.name}}</h5>
                  <p class="date" style="font-size:
12px;">{{Comments.commentdate.strftime('%b %d, %Y %I:%M %p')}}</p>
                    <p style="font-size: 16px;">
                      {{Comments.message}}
                    </p>
               </div>
            </div>

          </div>   {%endfor%}
     </div>
     <!-- Comment view end -->
       <!--  General Comment reply -->
      <form action="" method="POST">
  <div class="form-group">
    <h2 class="heading">Leave a Comment</h2>
    <div class="controls">
      <input type="text" id="name" class="floatLabel" placeholder="Name" name="name">

    </div>
    <div class="controls">
      <input type="email" id="email" class="floatLabel" name="emailid"
placeholder="Email">
    </div>
  </div>
  <div class="form-group">
    <div class="controls">
      <textarea name="message" class="floatLabel" placeholder="Comment"
id="comments"></textarea>
    </div>
    <br>
    <div class="controls">
          <button class="btn btn-primary"  type="submit">Send It</button>
```

```
   </div>
  </div>
</form>
<!-- Comment reply end -->
   </main>
  <div id="footer">
    <div>Coded by <a href="https://www.linkedin.com/in/i-ashu" target="_blank"
style="color: #e4cb58;text-decoration: none;">Aashu | Amrit</a></div>
    <div>Inspired from <a href="https://www.getbootstrap.com" target="_blank"
style="color: #e4cb58;text-decoration: none;">Bootstrap 4</a></div>
    <div>Made For <a href="https://www.gat.ac.in" target="_blank" style="color: #e4cb58;
text-decoration: none;"> GAT</a></div>
  </div>
</body>
<script>
    if ( window.history.replaceState ) {
        window.history.replaceState( null, null, window.location.href );
    }
</script>
</html>
```

Style.css

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans:300,400,500,700);
@import url(https://fonts.googleapis.com/css?family=Montserrat:500);
@import url("https://use.fontawesome.com/releases/v5.8.2/css/all.css");
@import
url("https://fonts.googleapis.com/css?family=Fjalla+One|Lobster|PT+Sans:400,400i,700,7
00i&display=swap");
@import url('https://fonts.googleapis.com/css?family=Lato&display=swap');
@import url('https://fonts.googleapis.com/css?family=Roboto&display=swap');

body {
  font-family: "Montserrat", sans-serif;
  background: #f2f2f2;
  margin: 0;
}
/*Nav bar Styling start*/
.wrapper{width: 100%; }
logo {
    line-height: 60px; position: fixed;
    float: left; margin: 0px 46px;
    color: #fff; font-weight: bold;
    font-size: 29px; font-family: 'Do Hyeon', sans-serif;
```

```
   letter-spacing: 2px;}
span{color: #e4cb58;}
nav {position: fixed; width: 100%;
   z-index: 100; background-color: #2A242E;}

nav ul {
   line-height: 60px; list-style: none; background: rgba(0, 0, 0, 0);
   overflow: hidden; color: #fff; text-align: right; margin: 0;
   padding-right: 40px; transition: 1s;}

nav.black ul { background: #000; }

nav ul li { display: inline-block; padding: 0px 36px;}

nav ul li a {
   text-decoration: none; color: #fff;
   font-size: 18px; font-family: 'Fjalla One', sans-serif;}
nav a:hover{
  background-color: white !important; color: black;
  padding: 2px 12px; transition: ease-in-out 700ms all;
  text-decoration: none;}

.menu-icon {
   line-height: 60px; width: 100%;
   background: #000; text-align: right;
   box-sizing: border-box; padding: 15px 24px;
   cursor: pointer; color: #fff; display: none;
}
/* Navbar Styling end */

/* Blog Content Styling start */
.cool-card{
        width: 990px; margin: 0 auto;
        padding: 20px 0; }

p{
  font-family: 'Roboto',sans-serif;
  font-weight: 200; color: #596256;}

article {margin-top: 1.5em;}

.card {
  display: -webkit-flex; display: -ms-flexbox;
```

```
  display: flex; color: #333;
  box-shadow: 0 1px 2px rgba(0, 0, 0, 0.7);}

.card-image {
  display: block; -webkit-flex: 1 0 50%;
  -ms-flex: 1 0 50%; flex: 1 0 50%;
  min-height: 260px; background-position: center;
  background-size: cover;  background-color: #ccc;
}

.card-body{ font-weight: 600;}

.card-content1 {
  display: block; -webkit-flex: 1 0 50%;
  -ms-flex: 1 0 50%; flex: 1 0 47%;
  margin: 0; padding: 0; background: #fff;}

.card-content1 {
  padding: 1.5em 1em 1.5em 1.5em;
  margin: 0; padding-bottom: 0;}

.card-subtitle-new {
  font-family:'Open Sans', sans-serif;
  font-size: 12px; color: #66CCCC;
  margin: 0;margin-bottom: 2px;
  padding: 0;font-weight: 200;}

.card-header {
  font-family: 'Secular One', sans-serif;
  font-weight: 900; font-size: 27px;
  color: #333;}

.card-header a {
  color: #333; text-decoration: none;
  transition: ease 500ms all;}

.card-header a:hover {
  color: #666;text-decoration: none;}

.card-body {
  font-family:'Varela', sans-serif;
  display: block;}
```

```
.read-more {
  margin-top: 1.5rem;
  text-align: right;
  color: #333;
  padding-right: 30px;
}

.read-more a {
  color: #333;
  opacity: .6;
  text-decoration: none;
  font-size: 14px;
}
/* Blog Content Styling End */

/*Comment section*/
.controls{
  text-align: left;
  position: relative;
}
  input[type="text"],
  input[type="email"],
  textarea,
  select{
    padding: 12px;
    border: 1px solid rgb(31, 30, 30);
    width: 100%;
    margin-bottom: 18px;
    color: #888;
    font-family: 'Lato', 'sans-serif';
    font-size: 12px;
    font-weight: 300;
    background-color: #fff;
    border-radius:4px;
    transition:all .3s;
    border-radius: 9px;
}
```

CHAPTER 6

TESTING

This chapter gives the outline of the testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. 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 component sub-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 test. Each test type addresses a specific testing requirement.

6.1 TESTING PROCESS

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

6.2 TESTING OBJECTIVES

The main objectives of testing process are as follows.

- Testing is a process of executing a program with the intent of finding an error.
- A good test case is one that has high probability of finding undiscovered error.
- A successful test is one that uncovers the undiscovered error.

Table 5.1: Test cases

S.NO	CASE	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT
1	Homepage	Posts from database	Article Pages	Article Pages
2	Article Page	Database Contents	Article contents and images	Article contents and images
3	Comment Section	Name, Email, Message	Comment posted with Name and Message	Comment posted with Name and Message
4	Comment Section	Blank Fields	Comment cannot be posted	Please fill out the fields
5	Contact Us Page	Name, Email, Phone, Message	Submitted Successfully	Submitted Successfully
6	Contact Us Page	Blank Fields	Form cannot be submitted	Form cannot be submitted
7	Admin Login	Username and Password	Redirect to dashboard page	Redirect to dashboard page
8	Admin Login	Wrong Username or Password	Redirect to Login Page	Redirect to Login Page

CHAPTER 7

RESULTS

This section describes the screens of the “Classic Blog”.

The snapshots are shown below for each module.

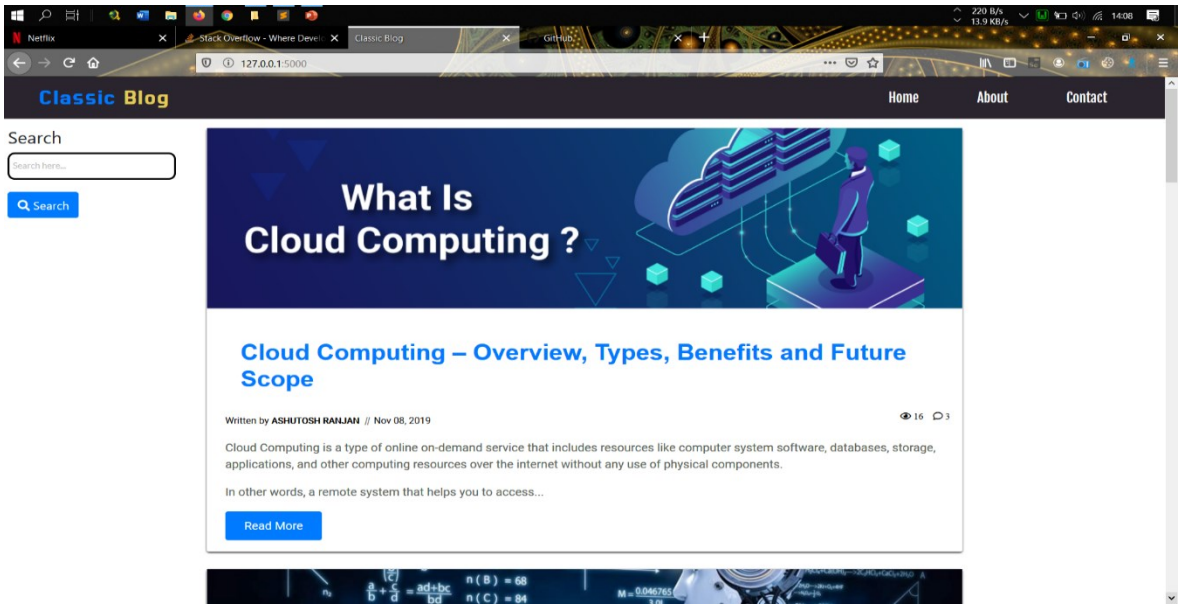

Figure 7.1: Front Page Header

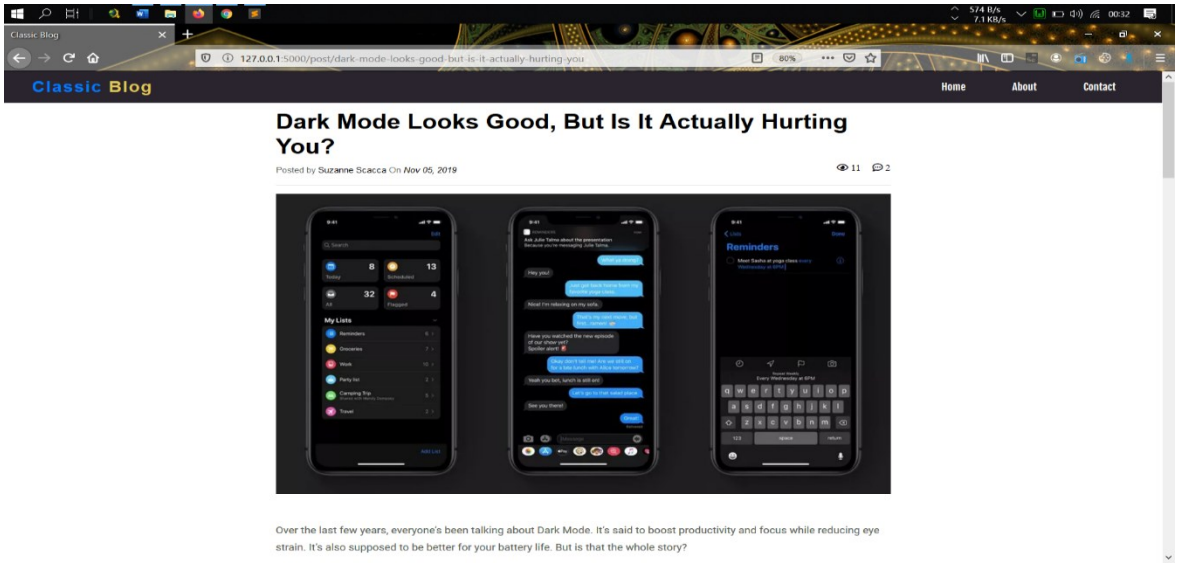

Figure 7.2: Article Page

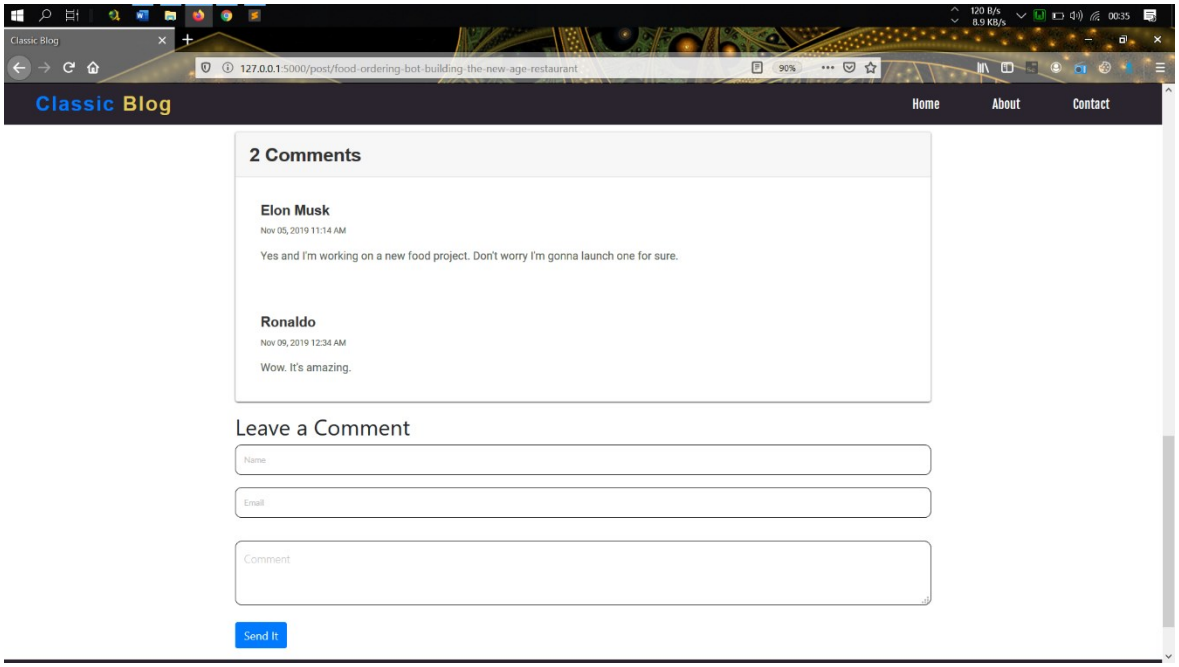

Figure 7.3: Comment Page

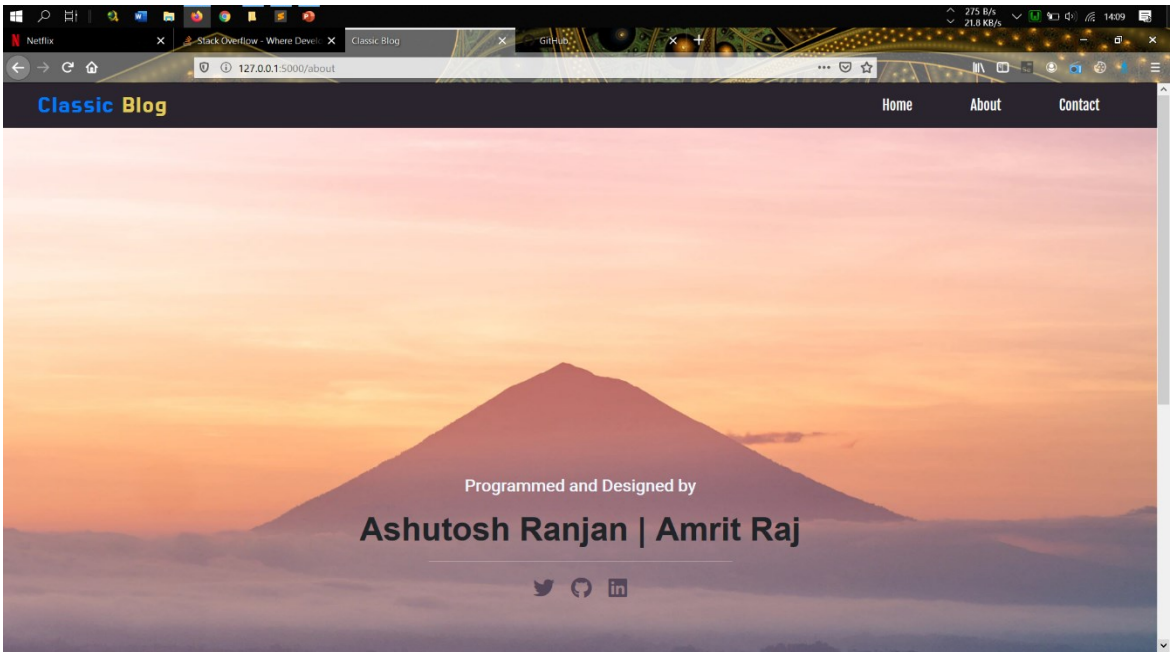

Figure 7.4: About Page

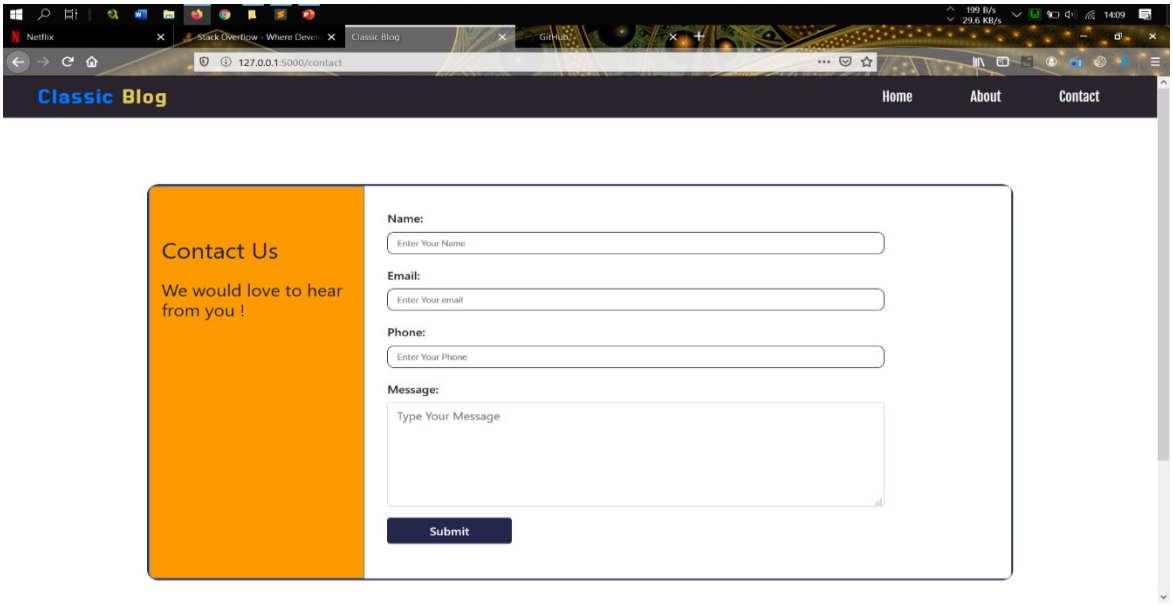

Figure 7.5: Contact Us Page

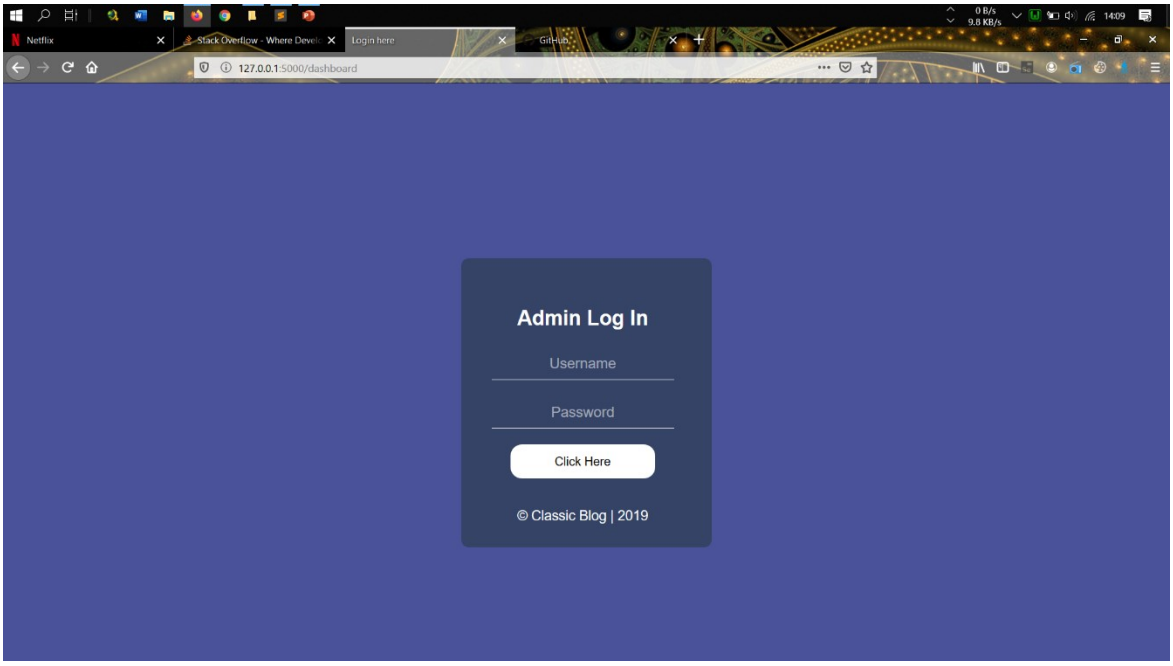

Figure 7.6: Admin Login Page

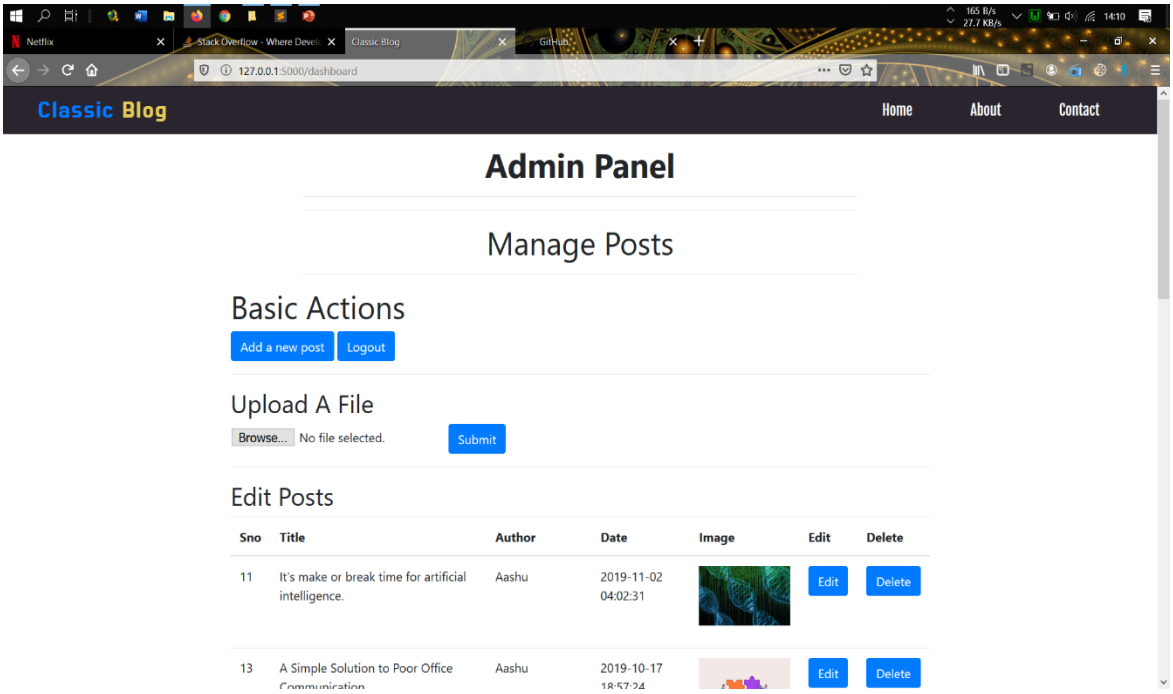

Figure 7.7: Dashboard

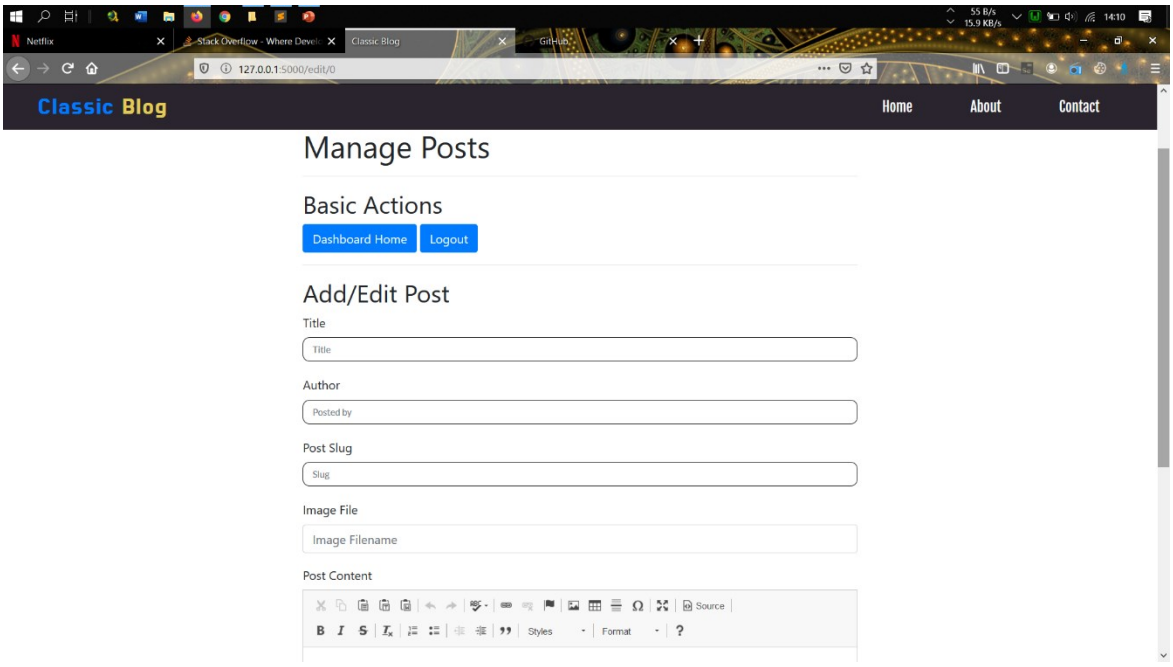

Figure 7.8: Add Post Page

CONCLUSION

With the theoretical inclination of our syllabus it becomes very essential to take at most advantage of any opportunity of gaining practical experience that comes along. The building blocks of this Major Project "CLASSIC BLOG" was one of these opportunities. It gave us the requisite practical knowledge to supplement the already taught theoretical concepts thus making us more competent as a computer engineer. The project from a personal point of view also helped us in understanding the following aspects of project development:

- The planning that goes into implementing a project.
- The importance of proper planning and an organized methodology.
- The key element of team spirit and co-ordination in a successful project.

BIBILIOGRAPHY

[1]. https://www.w3schools.com

[2]. https://flask.palletsprojects.com/en/0.12.x/

[3]. Randy Connolly, Ricardo Hoar, **"Fundamentals of Web Development"**, 1st Edition, Pearson Education India.

[4]. Robin Nixon, **"Learning PHP, MySQL & JavaScript with jQuery, CSS and HTML5"**, 4th Edition, O'Reilly Publications, 2015.