MVGR BLOG BUNDLE

A Mini Project Report

B Tech – VI Semester

D. HUSSAIN	**19331A1216**
A. RAJA	**19331A1203**
M. AKSHAYA	**19331A1235**
CH. SRI VAISHNAVI	**19331A1210**

At

DEPARTMENT OF INFORMATION TECHNOLOGY

MVGR COLLEGE OF ENGINEERING (A) VIZIANAGARAM.

June 2022

Project Coordinator	**Academic Coordinator**	**HOD - IT**
Dr. B. ANJANADEVI	**Dr. T. PAVAN KUMAR**	**Dr. V. NAGESH**

Abstract:

A blog is a frequently updated online personal journal or diary. It is a place to express yourself to the world. A place to share your thoughts and your passions. Really, it’s anything you want it to be. For our purposes we’ll say that a blog is your own website that you are going to updateon an ongoing basis. Blog is a short form for the word weblog and the two words are used interchangeably.

Blogs range from the personal to the political and can focus on one narrow subject or a whole range of subjects. It can also play an important role in student’s life. It can help in the promotion of critical and analytical thinking, increased access and exposure to quality content and acombination of solitary and social interactions with peers.

Currently students in schools or colleges are unable to express their ideas, their talent or anything that can expressed for some benefits for everyone. The reason is, schools or colleges don’t have any proper medium to accomplish it. But proposed online blogging system can help in accomplish these things and even much more. We willcover the objectives of this online blogging system in next section.

Internet has become reality and usage of internet become very much popular and there is tremendous increase of internet in all over the world for educational purpose. The Blog Web Application is easy to use, full-featured and much more.

Introduction:

A blog (a blend of the term web log) is a type of website or part of a website, which is the publicationof regular articles over time in some area of personal or professional interest. Blogs are usually maintained by an individual with regular entries of commentary, descriptions of events, or other material such as graphics or video. Blog can also be used as a verb, meaning to maintain or add content to a blog.

There are many different types of blogs, differing not only in the type of content, but also in the waythat content is delivered or written.

The software for writing blogs can be divided into **user hosted** (software application installed by weblog authors to run on their own system) and **developer hosted** (software services operated by the developer, requiring no software installation for the blog author.)

Problem statement:

Currently students in schools or colleges are unable to express their ideas, their talent or anything that can expressed for some benefits for everyone. The reason is, schools or colleges don't have any proper medium to accomplish it.

The project **"MVGR BLOG BUNDLE"** is a blogging application or website related to MVGR college of engineering, only the students and staff of the college will have the access to the website. It allows us to write blogs on different topics by different people. The website provides some categories, the blogs which are written by the users are categorized into these categories. This website also has a community environment between the users.

Software & Hardware Requirements

PYCHARM:

PyCharm is one of the most popular Python IDEs. There is a multitude of reasons for this, including the fact that it is developed by JetBrains, the developer behind the popular IntelliJ IDEA IDE that is one of the big 3 of Java IDEs and the "smartest JavaScript IDE" WebStorm. Having the support for web development by leveraging Django is yet another credible reason. There are a galore of factors that make PyCharm one of the most complete and comprehensive integrated development environments for working with the Python programming language. Before proceeding further into exploring the know-how of PyCharm i.e., features, installation, and pros & cons, let's first get a brief introduction to PyCharm.

PyCharm comes with a plethora of modules, packages, and tools to hasten Python development while cutting-down the effort required to do the same to a great extent, simultaneously. Further, PyCharm can be customized as per the development requirements, and personal preferences call for. It was released to the public for the very first time back in February of 2010. In addition to offering code analysis, PyCharm features:

- A graphical debugger
- An integrated unit tester
- Integration support for version control systems (VCSs)
- Support for data science with Anaconda

Features of PyCharm:

1. Intelligent Code Editor

PyCharm comes with a smart code editor that facilitates writing high-quality Python code. It offers an enhanced level of code comprehension and readability by means of distinct colour schemes for keywords, classes, and functions, i.e., syntax and error highlighting.

2. Availability of Integration Tools

PyCharm provides support for integrating a range of tools. These tools vary from helping in enhancing the code productivity to facilitate dealing with data science projects.

3. Data Science and Machine Learning [Professional Edition Only]

PyCharm comes with support for scientific libraries, such as Matplotlib and SciPy, to help Python developers accomplish data science and machine learning projects.

4. Google App Engine [Professional Edition Only]

Google App Engine, or directly App Engine, is a PaaS and cloud computing platform targeted towards developing and hosting web applications. It offers automatic scaling for web applications. The professional edition of PyCharm provides support for Google App Engine.

5. Integrated Debugging and Testing

An IDE comes with support for debugging and testing programs. To accomplish the same, PyCharm features an integrated Python debugger and integrated unit testing with line-by-line code coverage.

FRAMEWORK(DJANGO):

Django is an MVT web framework that is used to build web applications. The huge Django web-framework comes with so many "batteries included" that developers often get amazed as to how everything manages to work together. The principle behind adding so many batteries is to have common web functionalities in the framework itself instead of adding latter as a separate library. One of the main reasons behind the popularity of Django framework is the huge Django community. The community is so huge that a separate website was devoted to it where developers from all corners developed third-party packages including authentication, authorization, full-fledged Django powered CMS systems, e-commerce add-ons and so on. There is a high probability that what you are trying to develop is already developed by somebody and you just need to pull that into your project.

Reasons to choose Django for web development:

1.Python

Python is arguably one of the easiest programming languages to learn because of its simple language constructs, flow structure and easy syntax. It is versatile and runs websites, desktop applications and mobile applications embedded in many devices and is used in other applications as a popular scripting language.

2.Batteries Included

Django comes with common libraries which are essential to build common functionalities like URL routing, authentication, an object-relational mapper (ORM), a templating system and db-schema migrations.

3.Built-in admin

Django has an in-built administration interface which lets you handle your models, user/ group permissions and to manage users. With model interface in place, there is no need for a separate database administration program for all but advanced database functions.

4.Doesn’t get in your way

Creating a Django application adds no boilerplate and no unnecessary functions. There’s no mandatory imports, third-party libraries and no XML configuration files.

5.Scalable

Django is based on MVC design pattern. It means that all the entities like db (database), back-end and front-end code are individual entity. Django allows us to separate code from the static media including pictures, files, CSS and JavaScript that make up your site.

TECHNOLOGIES USED (HTML, CSS):

HTML:

HTML is used to create electronic documents (called pages) that are displayed on the World Wide Web. Each page contains a series of connections to other pages called hyperlinks. Every web page you see on the Internet is written using one version of HTML code or another. HTML code ensures the proper formatting of text and images so that your Internet browser may display them as they are intended to look. Without HTML, a browser would not know how to display text as elements or load images or other elements.

HTML5 is the update made to HTML from HTML4 (XHTML follows a different version numbering scheme). It uses the same basic rules as HTML4, but adds some new tags and attributes which allow for better semantics and for dynamic elements that are activated using JavaScript. New elements include section. There are also new input types for forms, which include tel, search, url, email, datetime, date, month, week, time, number, range and colour.

CSS:

Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Plus, CSS makes it easy to change styles across several pages at once. If the pages all reference the same style sheet, the text size only needs to be changed on the style sheet and all the pages will show the larger text.While CSS is great for creating text styles, it is helpful for formatting other aspects of Web page layout as well.

For example, CSS can be used to define the cell padding of table cells, the style, thickness, and color of a table's border, and the padding around images or other objects. CSS gives Web developers more exact control over how Web pages will look than HTML does. Therefore,most Web pages today incorporate cascading style sheets

DATABASE DESIGN:

PostgreSQL:
PostgreSQL is a powerful, open-source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments, and help you manage your data no matter how big or small the dataset. In addition to being free and open source, PostgreSQL is highly extensible. For example, you can define your own data types, build outcustom functions, even write code from different programming languages without recompiling yourdatabase.

PostgreSQL tries to conform with the SQL standard where such conformance does not contradict traditional features or could lead to poor architectural decisions. Many of the features required by the SQL standard are supported, though sometimes with slightly differing syntax or function. Further moves towards conformance can be expected over time.

- Data Types
    - Primitives: Integer, Numeric, String, Boolean
    - Structured: Date/Time, Array, Range / Multirange, UUID
    - Document: JSON/JSONB, XML, Key-value (Hstore)
    - Geometry: Point, Line, Circle, Polygon
    - Customizations: Composite, Custom Types
- Data Integrity
    - UNIQUE, NOT NULL
    - Primary Keys
    - Foreign Keys
    - Exclusion Constraints
    - Explicit Locks, Advisory Locks
- Concurrency, Performance
    - Indexing: B-tree, Multicolumn, Expressions, Partial
    - Advanced Indexing: GiST, SP-Gist, KNN Gist, GIN, BRIN, Covering indexes,
      Bloom filters
    - Sophisticated query planner / optimizer, index-only scans, multicolumn statistics
    - Transactions, Nested Transactions (via savepoints)
    - Multi-Version concurrency Control (MVCC)
    - Parallelization of read queries and building B-tree indexes
    - Table partitioning
    - All transaction isolation levels defined in the SQL standard, including Serializable
    - Just-in-time (JIT) compilation of expressions
- Reliability, Disaster Recovery
    - Write-ahead Logging (WAL)
    - Replication: Asynchronous, Synchronous, Logical
    - Point-in-time-recovery (PITR), active standbys
    - Tablespaces

- Security
    - Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate, and more
    - Robust access-control system
    - Column and row-level security
    - Multi-factor authentication with certificates and an additional method
- Extensibility
    - Stored functions and procedures
    - Procedural Languages: PL/PGSQL, Perl, Python (and many more)
    - SQL/JSON path expressions
    - Foreign data wrappers: connect to other databases or streams with a standard SQLinterface
    - Customizable storage interface for tables
    - Many extensions that provide additional functionality, including PostGIS
- Internationalisation, Text Search
    - Support for international character sets, e.g. through ICU collations
    - Case-insensitive and accent-insensitive collations
    - Full-text search

Existing System:

A blog is a discussion or informational website published on the World Wide Web consisting of discrete, often informal diary-style text entries (posts). Posts are typically displayed in reverse chronological order, so that the most recent post appears first, at the top of the web page. Until 2009, blogs were usually the work of a single individual occasionally of a small group, and often covered a single subject or topic. In the 2010s, "multi-author blogs" (MABs) emerged, featuring the writing of multiple authors and sometimes professionally edited. MABs from newspapers, other media outlets, universities, think tanks, advocacy groups, and similar institutions account for an increasing quantity of blog traffic. The rise of Twitter and other "microblogging" systems helps integrate MABs and single-author blogs into the news media. Blog can also be used as a verb, meaning to maintain or add content to a blog. 'Blog' and 'blogging' are now loosely used for content creation and sharing on social media, especially when the content is long-form and one creates and shares content on regular basis. So, one could be maintaining a blog on Facebook or blogging on Instagram.

Early blogs were simply manually updated components of common Websites. In 1995, the "Online Diary" on the Ty, Inc. Web site was produced and updated manually before any blogging programs were available. Posts were made to appear in reverse chronological order by manually updating text-based HTML code using FTP software in real time several times a day. To users, this offered the appearance of a live diary that contained multiple new entries per day. At the beginning of each new day, new diary entries were manually coded into a new HTML file, and at the start of each month, diary entries were archived into their own folder which contained a separate HTML page for every day of the month. Then menus that contained links to the most recent diary entry were updated manually throughout the site. This text-based method of organizing thousands of files served as a springboard to define future blogging styles that were captured by blogging software developed years later.[16]

The evolution of electronic and software tools to facilitate the production and maintenance of Web articles posted in reverse chronological order made the publishing process feasible to a much larger and less technically-inclined population. Ultimately, this resulted in the distinct class of online publishing that produces blogs we recognize today. For instance, the use of some sort of browser-based software is now a typical aspect of "blogging". Blogs can be hosted by dedicated blog hosting services, on regular web hosting services, or run using blog software.

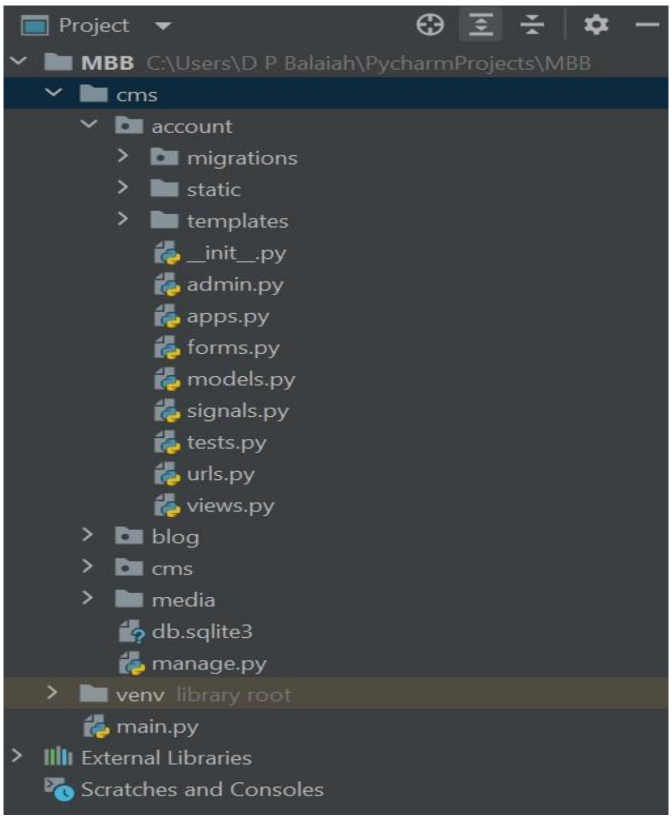
Proposed system:

A student as a part of our college shows interests in various things going on in that college. What would a student do if he/she requires-information of that college? Will they be going to every corner of college asking random people in and around to gather fact, information, knowledge? No, this is a hectic and burden load and leads to the depreciation of the innocent student's interests about the college his entire stay study period.

As time changes, interests and, requirements and wants change too. For example, the same student was interested in a sport and joined a sports club in his initial year of study. Then by the time he goes to the third year of study, he decides to join some other club, let it be “Entrepreneurship” club to develop his business skills. What would the student do to know about this club and how would he join without any prior information of that club? He/she needs a platform where they he/she can collect the details and proceed as needed.

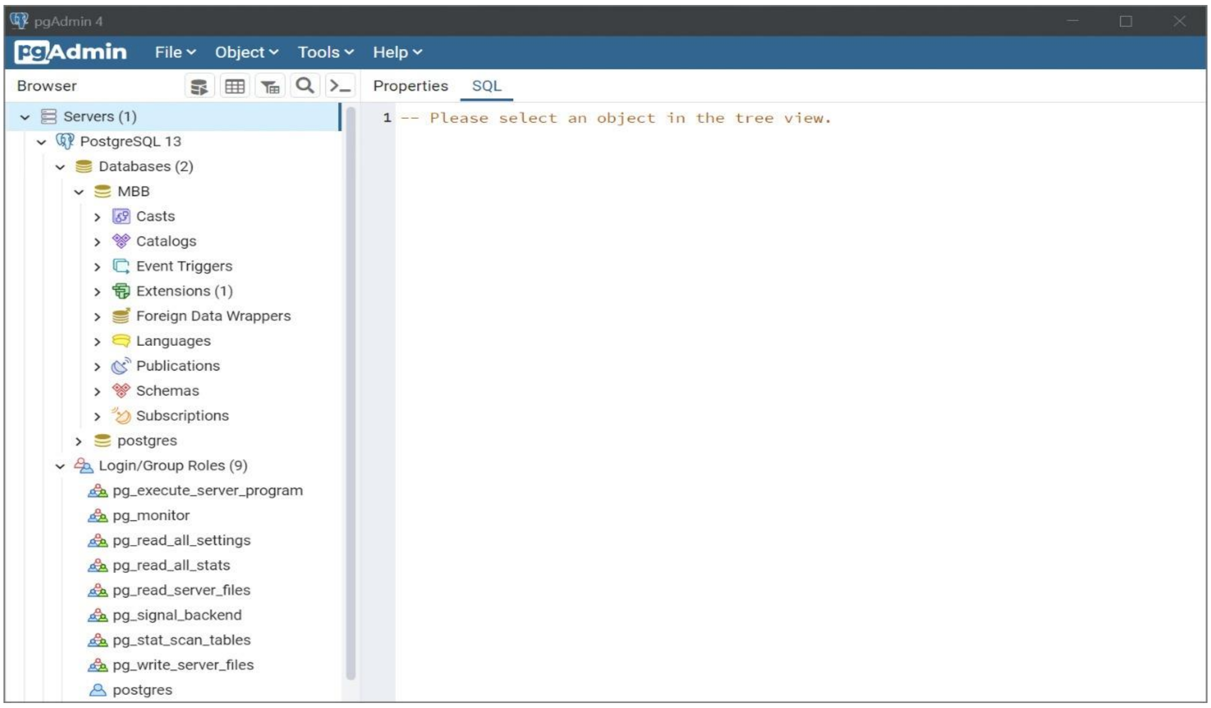
Similarly, there are a huge number of students and staff present in the college or university who might be facing similar problem in search of their interests regarding college. Few of the topics that are majorly be chosen as the interests are college events & activities, Achievements of college and its part, Facts & myths of college, Library details, inspiring and influencing and popular people of college, groups & clubs, flora and fauna in college, other students’ or staffs’ stories or poems, or any donations & contributions and many. The people who are interested in writing would be highly beneficial using this platform as this platform also promotes students to express their writing skills in a highly collaborative and creative way.

PROJECT STRUCTURE:



```
Project
MBB  C:\Users\D P Balaiah\PycharmProjects\MBB
  cms
    account
      migrations
      static
      templates
      __init__.py
      admin.py
      apps.py
      forms.py
      models.py
      signals.py
      tests.py
      urls.py
      views.py
    blog
    cms
    media
    db.sqlite3
    manage.py
  venv  library root
  main.py
External Libraries
Scratches and Consoles
```

DATABASE STRUCTURE:



```
Servers (1)
  PostgreSQL 13
    Databases (2)
      MBB
        Casts
        Catalogs
        Event Triggers
        Extensions (1)
        Foreign Data Wrappers
        Languages
        Publications
        Schemas
        Subscriptions
      postgres
    Login/Group Roles (9)
      pg_execute_server_program
      pg_monitor
      pg_read_all_settings
      pg_read_all_stats
      pg_read_server_files
      pg_signal_backend
      pg_stat_scan_tables
      pg_write_server_files
      postgres
```

```
-- Please select an object in the tree view.
```

DATA SET USED WITH SAMPLE CODE:

Sample main code:

initial.py:

```
import django.contrib.auth.models
import django.contrib.auth.validators
from django.db import migrations, models
import django.utils.timezone


class Migration(migrations.Migration):


    initial = True


    dependencies = [
        ('auth', '0011_update_proxy_permissions'),
    ]


    operations = [
        migrations.CreateModel(
            name='MyUser',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('password', models.CharField(max_length=128, verbose_name='password')),
                ('last_login', models.DateTimeField(blank=True, null=True, verbose_name='last login')),
                ('is_superuser', models.BooleanField(default=False, help_text='Designates that this user has all permissions without explicitly assigning them.', verbose_name='superuser status')),
                ('username', models.CharField(error_messages={'unique': 'A user with that username already exists.'}, help_text='Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.', max_length=150, unique=True,
validators=[django.contrib.auth.validators.UnicodeUsernameValidator()],
verbose_name='username')),

            ('first_name', models.CharField(blank=True, max_length=30,
verbose_name='first name')),

            ('last_name', models.CharField(blank=True, max_length=150,
verbose_name='last name')),

            ('is_staff', models.BooleanField(default=False, help_text='Designates whether
the user can log into this admin site.', verbose_name='staff status')),

            ('is_active', models.BooleanField(default=True, help_text='Designates
whether this user should be treated as active. Unselect this instead of deleting accounts.',
verbose_name='active')),

            ('date_joined', models.DateTimeField(default=django.utils.timezone.now,
verbose_name='date joined')),

            ('email', models.EmailField(max_length=254, unique=True)),

            ('bio', models.CharField(blank=True, max_length=500)),

            ('profile_image', models.ImageField(blank=True,
upload_to='account/profile_pic')),

            ('is_author', models.BooleanField(default=False)),

            ('groups', models.ManyToManyField(blank=True, help_text='The groups this
user belongs to. A user will get all permissions granted to each of their groups.',
related_name='user_set', related_query_name='user', to='auth.Group',
verbose_name='groups')),

            ('user_permissions', models.ManyToManyField(blank=True,
help_text='Specific permissions for this user.', related_name='user_set',
related_query_name='user', to='auth.Permission', verbose_name='user permissions')),

        ],

        options={

            'verbose_name': 'user',

            'verbose_name_plural': 'users',

            'abstract': False,

        },

        managers=[

            ('objects', django.contrib.auth.models.UserManager()),

        ],
        ),
    ]
myuser_slug.py:
from django.db import migrations, models
class Migration(migrations.Migration):
    dependencies = [
        ('account', '0001_initial'),
    ]
    operations = [
        migrations.AddField(
            model_name='myuser',
            name='slug',
            field=models.CharField(blank=True, max_length=60, unique=True),
        ),
    ]
views.py:
from django.shortcuts import render
from .models import MyUser
from django.views.generic import ListView, DetailView, CreateView, UpdateView
# from django.contrib.auth.forms import UserCreationForm
from .forms import SignUpForm, ProfileUpdateForm, AuthorRequestForm
from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin
from django.urls import reverse_lazy, reverse

# Create your views here.
class UserCreateView(CreateView):
    form_class = SignUpForm
    template_name = "account/signup.html"
    success_url = reverse_lazy("login")
```

```
class ProfileView(DetailView):
    model = MyUser
    content_object_name = "user"
    template_name = "account/profile.html"

class ProfileUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):
    login_url = "/accounts/login"
    model = MyUser
    form_class = ProfileUpdateForm
    template_name = "account/profile_update.html"

    def test_func(self, *args, **kwargs):
        current_user = self.request.user
        profile_obj = MyUser.objects.get(slug=self.kwargs.get('slug'))
        if current_user == profile_obj:
            return True
        else:
            return False

class AuthorRequestView(LoginRequiredMixin, CreateView):
    login_url = "/accounts/login"
    form_class = AuthorRequestForm
    template_name = "account/author_request.html"

    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        user = MyUser.objects.get(username= self.request.user)
        kwargs.update({'initial':{'sender': user}})
        return kwargs
    def form_valid(self, form):
        form.instance.sender = self.request.user
        return super().form_valid(form)


class MyPostsView(LoginRequiredMixin, ListView):
    login_url = "/accounts/login"
    model = MyUser
    context_object_name = "user"
    template_name = "account/my_posts.html"


    def get_context_data(self, *args, **kwargs):
        context = super().get_context_data()
        user = MyUser.objects.get(username = self.request.user)
        context['posts'] = user.post_set.all()
        return context
```

settings.py:

```
import os
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
TEMPLATES_DIR = os.path.join(BASE_DIR, 'templates')
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '-c_m6!nr-(hr5it9jg1ihj4s)^a)w+xi8xcxq&p*074hq0ln&4'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog.apps.BlogConfig',
    'account.apps.AccountConfig',
    'tinymce',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'cms.urls'
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [TEMPLATES_DIR],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'cms.wsgi.application'
# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'MBB',
        'USER':'postgres',
        'PASSWORD':'mbb123',
        'HOST':'localhost',
    }
}
# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS=[os.path.join(BASE_DIR,'static')]
# Media files
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, "media")

# Login/Logout Redirect urls
LOGIN_REDIRECT_URL = '/blogs'
LOGOUT_REDIRECT_URL = '/blogs'

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'

AUTH_USER_MODEL = 'account.MyUser'

TINYMCE_JS_URL=os.path.join(MEDIA_URL,
"tinymce/js/tinymce/tinymce.min.js")
```

```
TINYMCE_JS_ROOT = os.path.join(MEDIA_ROOT, "tinymce")
```

Sample CSS codes:

bases1.css:

```
body {
    background-image: url("/cms/blog/static/blog/img/violet.png");


  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}


#top-bar {
    background-color: #98FB98;
    position: fixed;
    top: 0;
    left: 0;
    align-items: center;
    box-sizing: border-box;
    box-shadow: 0 1px 1px #ccc;
    width: 100%;
    height: 80px;
    z-index: 90;
    padding: 1em 4.875em;
    display: flex;
    justify-content: space-between;
}


.logo-bar {
    cursor: pointer;
}
```

```
.logo {
   background-image:              linear-gradient(269deg,#0048ff,#0096ff),              linear-
gradient(#000,#000);
   -webkit-background-clip: text;
   -webkit-text-fill-color: transparent;
   display: block;
   font-family: "Raleway";
   font-size: 32px;
   font-weight: 900;
   height: 34px;
   letter-spacing: -0.9px;
   color: rgb(0, 132, 255);
}

.logo-bar a {
   text-decoration: none;
}

.logo-bar p {
   font-family: Raleway,sans-serif;
   font-size: 1em;
   font-weight: 600;
   font-style: italic;
   font-stretch: normal;
   line-height: .8;
   letter-spacing: .9px;
   margin: 0;
}
```

```
.search {
   width: 300px;
   position: relative;
   display: flex;
   align-items: center;
 }

.searchTerm {
   width: 100%;
   border: 2px solid #882341;
   border-right: none;
   padding: 5px;
   height: 30px;
   border-radius: 5px 0 0 5px;
   outline: none;
   color: #9DBFAF;
 }
.searchTerm:focus{
   color: #910a2a;
}
.searchButton {
   width: 38px;
   height: 30px;
   border: 1px solid #0044F0;
   background: #0044F0;
   text-align: center;
   color: #fff;
   border-radius: 0 5px 5px 0;
   cursor: pointer;
   font-size: 20px;
```

```
  }
.fa-search {
    color: white;
    font-size: 18px;
}
#menu-items-container {
    display: flex;
    justify-content: flex-end;
    align-items: center;
}
#menu-items-container a {
    text-decoration: none;
    color: #000;
}
.menu-item {
    font-family: "Raleway";
    font-size: 16px;
    font-weight: 600;
    letter-spacing: -0.9px;
    line-height: 69.84px;
    margin: 0px;
    margin-right: 12px;
}
#signup-btn {
    width: auto;
    height: 40px;
    min-width: 100px;
    overflow-wrap: break-word;
    word-wrap: break-word;
    border-radius: 20px;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
    border: none;
    cursor: pointer;
    padding: 8px 32px;
    /* margin-right: 16px; */
    color: #fff;
    outline: none;
    font-family: Raleway,sans-serif;
    font-size: .9em;
    font-weight: 600;
    font-style: normal;
    font-stretch: normal;
    letter-spacing: .03em;
    background: #0084ff;
    background: -webkit-linear-gradient(269deg,#0048ff,#0096ff);
    background: -webkit-linear-gradient(181deg,#0048ff,#0096ff);
    background: -o-linear-gradient(181deg,#0048ff,#0096ff);
    background: linear-gradient(269deg,#0048ff,#0096ff);
    filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr="#0084ff",endColorstr="#
0052ff",GradientType=1);
}


#bottom-bar {
    background-color: #172146;
    width: 100%;
    height: 80px;
    /* position: fixed; */
    /* bottom: 0; */
```

```
}
```

form.css:

```
#outer-wrapper {
    width: 400px;
    height: 700px;
    margin:3% auto;
    border-radius: 25px;
    background-color: rgba(0,0,0,0.1);
    box-shadow: 0 0 50px #fcfbf9;
}

#inner-wrapper {
    padding: 5% 8%;
}

h3 {
    text-align: center;
    margin-top: 16px;
    margin-bottom: 44px;
    font-size: 1.8em;
    -o-object-fit: contain;
    object-fit: contain;
    font-family: Raleway,sans-serif;
    font-weight: 600;
    font-style: normal;
    font-stretch: normal;
    letter-spacing: -.9px;

    color: #000;
    /* margin-top: 0;
    margin-bottom: 0; */
}

.field-container {
    margin-top: 1.5em;
    width: 100%;
    height: auto;
    font-size: 16px;
    /* -webkit-box-sizing: border-box; */
    box-sizing: border-box;
    /* display: inline-block; */
    position: relative;
    /* display: -ms-flexbox; */
    display: flex;
    /* -ms-flex-direction: column; */
    flex-direction: column;
    /* -ms-flex-pack: end; */
    justify-content: flex-end;
    /* margin-top: .825em; */
}

.field-container label {
    display: block;
    /* -webkit-transition: all .5s; */
    /* -o-transition: all .5s; */
    transition: all .5s;
    font-family: Raleway,sans-serif;
    font-weight: 400;
    font-style: normal;
    font-stretch: normal;
    text-align: left;
    color: #ffc400;
}

.field-container input {
    padding: .675em 0;
    font-size: 1em;
    outline: none;
    border-top: none;
    border-left: none;
    border-right: none;
    border-bottom: 2px solid #ccc;
    /* padding: .375em 0; */
    width: 100%;
    background-color: transparent;
    position: relative;
    font-family: Roboto,sans-serif;
    /* font-size: .875em; */
    font-weight: 500;
    font-style: normal;
    font-stretch: normal;
    letter-spacing: .5px;
    text-align: left;
    color: #fcfbf9;
}

#button-container {
    margin-top: 44px;
    margin-bottom: 0;
}
```

```
#submit-btn {
    width: auto;
    height: 40px;
    min-width: 100px;
    overflow-wrap: break-word;
    word-wrap: break-word;
    border-radius: 20px;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
    border: none;
    cursor: pointer;
    padding: 8px 32px;
    margin-right: 16px;
    color: #fff;
    outline: none;
    font-family: Raleway,sans-serif;
    font-size: .9em;
    font-weight: 600;
    font-style: normal;
    font-stretch: normal;
    letter-spacing: .03em;
    background: #0084ff;
    background: -webkit-linear-gradient(269deg,#0048ff,#0096ff);
    background: -webkit-linear-gradient(181deg,#0048ff,#0096ff);
    background: -o-linear-gradient(181deg,#0048ff,#0096ff);
    background: linear-gradient(269deg,#0048ff,#0096ff);
    filter:
progid:DXImageTransform.Microsoft.gradient(startColorstr="#0084ff",endColorstr="#
0052ff",GradientType=1);
}

#button-container a {
    text-decoration: none;
    font-family: Roboto;
    font-weight: 600;
    color: black;
}
```

Sample template codes:

login.html:

```
{% extends "blog/base.html" %}
{% load static %}

{% block extra_head %}
    <link href="{% static 'account/css/lform.css' %}" rel="stylesheet" type="text/css">
{% endblock %}

{% block content %}
<br>
<br>
<br>
<br>
    <div id="outer-wrapper">
        <div id="inner-wrapper">
            <h3>Login</h3>
            <form method="POST" enctype="multipart/form-data">
                {% csrf_token %}
                {% for field in form %}
                <div class="field-container">
                    {{ field.label_tag }}
```

```
          {{ field }}
          {{ field.errors }}
        </div>
        {% endfor %}
        <div id="button-container">
          <input type="submit" value="LOGIN" id="submit-btn">
          <a href="{% url 'password-reset' %}" id="forgot-password">Forgot Password?</a>
        </div>
      </form>
    </div>
  </div>
{% endblock %}
```

my_posts.html:

```
{% extends "blog/base.html" %}
{% load static %}

{% block extra_head %}
  <link href="{% static 'account/css/mypost1.css' %}" rel="stylesheet" type="text/css">
  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400&display=swap" rel="stylesheet">
  <script src="https://kit.fontawesome.com/f9c7a50154.js" crossorigin="anonymous"></script>
{% endblock %}

{% block content %}
<br>
<br>
<br>
<br>
```

```
  <div id="outer-wrapper">
    <div id="inner-wrapper">

      <div id="create-post">
        <a href="{% url 'post-create' %}"><h2>Create New Post</h2></a>
        <i class="fas fa-edit"></i>
      </div>

      <div id="cards-container">
        {% for post in posts %}
        <div id="card">
          <div id="card-inner-upper">
            <div id="post-image">
              <img src="{{ post.image.url }}" alt="Post Cover">
            </div>
            <div id="post-content">
              <h3>{{ post.title }}</h3>
              <!-- <p>{{ post.content }}</p> -->
            </div>
          </div>

          <div id="card-inner-lower">
            <a href="{% url 'post-detail' post.slug %}">VIEW</a>
            <a href="{% url 'post-update' post.slug %}">EDIT</a>
            <a href="{% url 'post-delete' post.slug %}" style="color:
red;">DELETE</a>
          </div>
        </div>
        {% endfor %}
      </div>
```

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

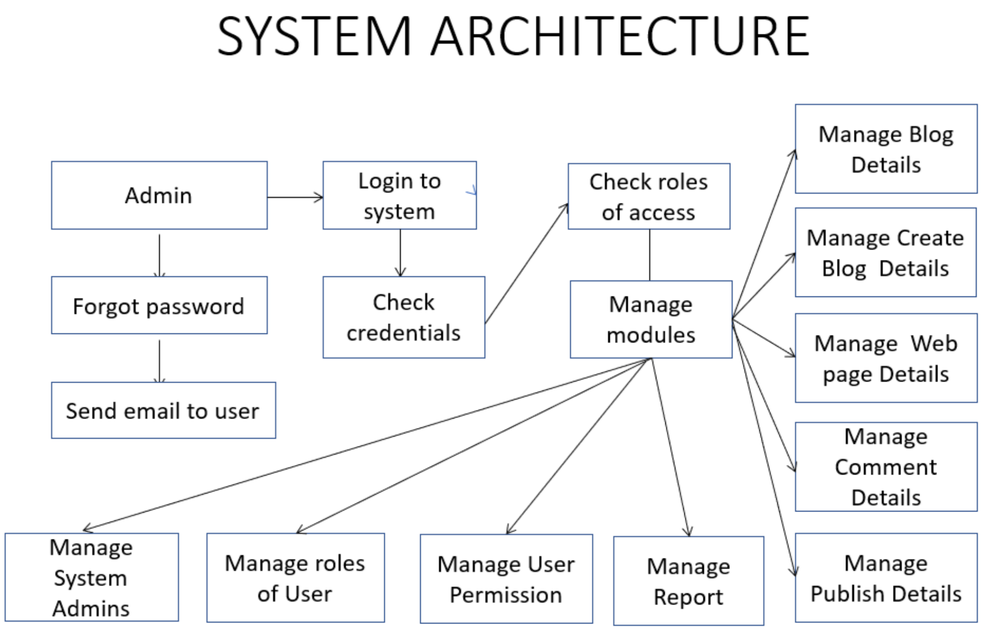
signup.html:

```
{% extends "blog/base.html" %}
{% load static %}
{% block extra_head %}
    <link rel="stylesheet" type="text/css" href="{% static 'account/css/form.css' %}" >
{% endblock %}
{% block content %}
<br><br><br><br>
    <div id="outer-wrapper">
        <div id="inner-wrapper">
            <h3>Signup</h3>
            <form method="POST" enctype="multipart/form-data">
                {% csrf_token %}
                {% for field in form %}
                <div class="field-container">
                    {{ field.label_tag }}
                    {{ field }}
                    {{ field.errors }}
                </div>
                {% endfor %}
                <div id="button-container">
                    <input type="submit" value="SIGNUP" id="submit-btn">
                </div>
            </form>
        </div>
    </div>
{% endblock %}
```

profile.html:

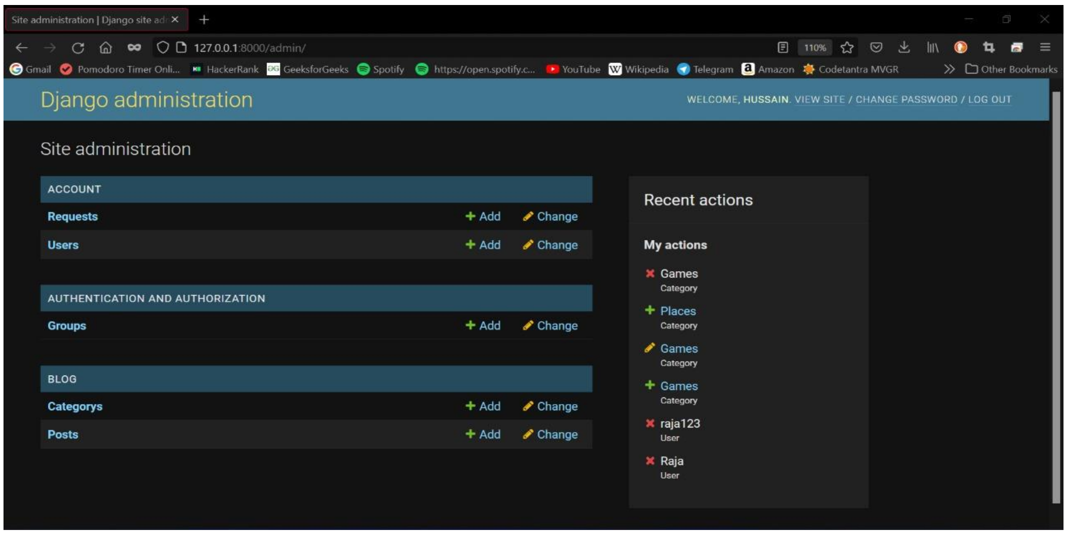
```
{% extends "blog/base.html" %}

{% load static %}


{% block extra_head %}

    <link href="{% static 'account/css/profile1.css' %}" rel="stylesheet" type="text/css">

    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400&display=swa
p" rel="stylesheet">

    <script                    src="https://kit.fontawesome.com/f9c7a50154.js"
crossorigin="anonymous"></script>

{% endblock %}

{% block content %}

<br><br><br><br>

    <div id="outer-wrapper">

        <div id="inner-wrapper">

            <div>

                <div id="basicinfosection">

                    <img   src="{{   request.user.profile_image.url   }}"   alt="Profile   Picture"
id="profile-picture">

                    <h1 id="username">{{ request.user.username }}</h1>

                </div>

            </div>

            <div class="section-container">

                <h3>Bio</h3>

                <p>{{ user.bio }}</p>

            </div>

        </div>

    </div>

{% endblock %}
```

SYSTEM ARCHITECTURE AND DESCRIPTION:



Output Screenshots

Django administration (admin page)



Django administration

WELCOME, **HUSSAIN**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

ACCOUNT		
Requests	+ Add	Change
Users	+ Add	Change

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change

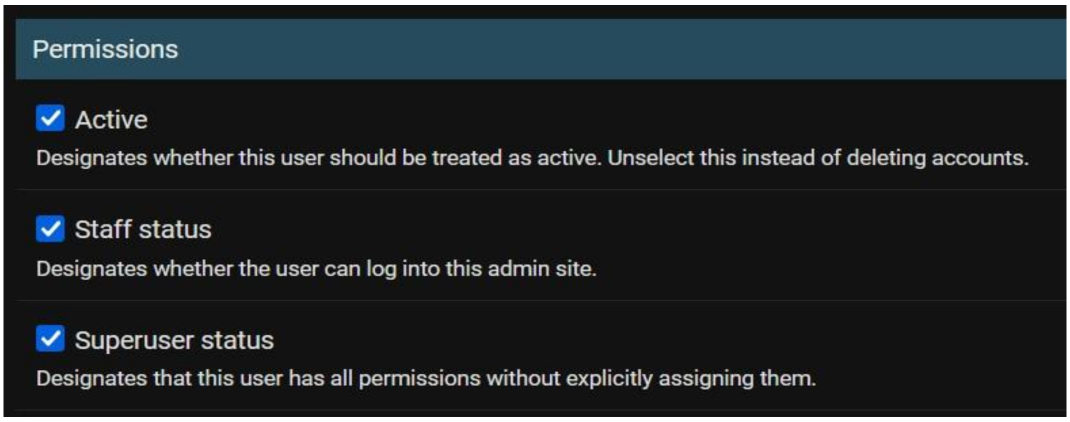
BLOG		
Categorys	+ Add	Change
Posts	+ Add	Change

Recent actions

My actions

- ✗ Games
  Category
- + Places
  Category
- ✎ Games
  Category
- + Games
  Category
- ✗ raja123
  User
- ✗ Raja
  User

Permissions regards the status of user:


Permissions

✓ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

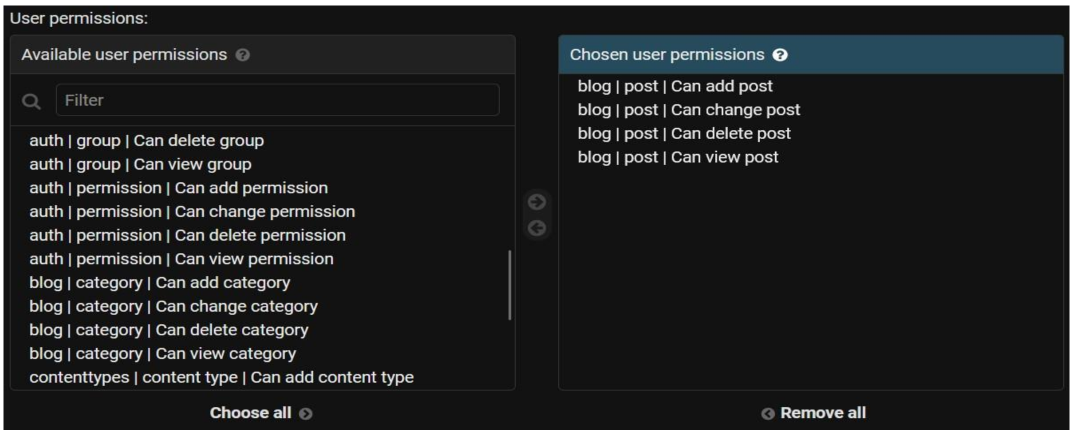
✓ Staff status

Designates whether the user can log into this admin site.

✓ Superuser status

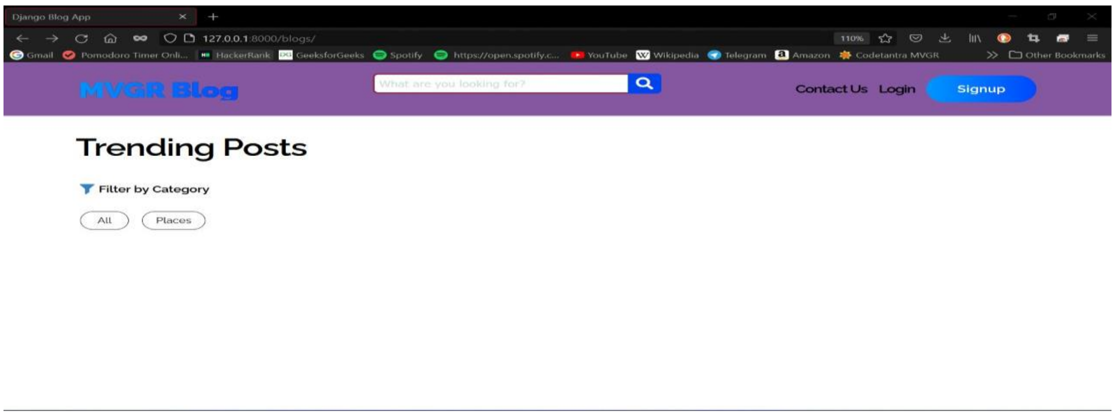
Designates that this user has all permissions without explicitly assigning them.

List of permissions:

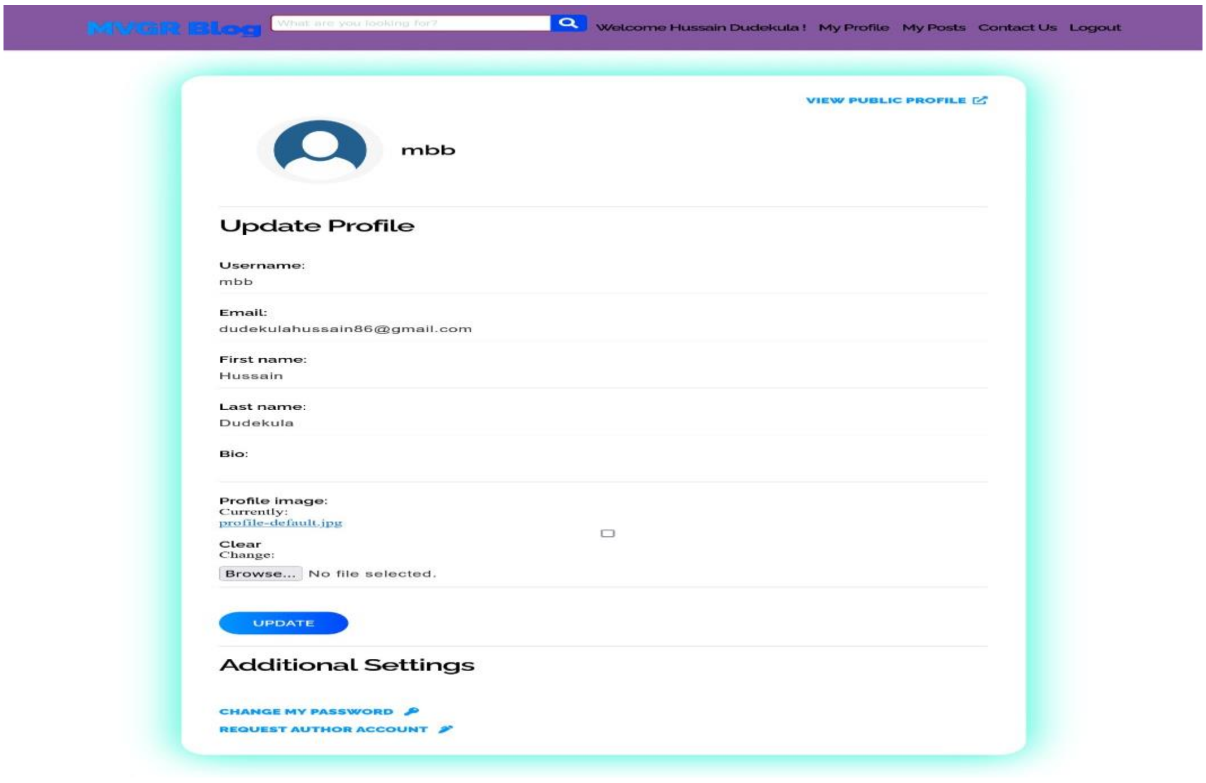

User permissions:

Available user permissions	Chosen user permissions
auth | group | Can delete group	blog | post | Can add post
auth | group | Can view group	blog | post | Can change post
auth | permission | Can add permission	blog | post | Can delete post
auth | permission | Can change permission	blog | post | Can view post
auth | permission | Can delete permission	
auth | permission | Can view permission	
blog | category | Can add category	
blog | category | Can change category	
blog | category | Can delete category	
blog | category | Can view category	
contenttypes | content type | Can add content type	

Home page:



Profile page:



MVGR Blog | What are you looking for? | Welcome Hussain Dudekula ! | My Profile | My Posts | Contact Us | Logout

VIEW PUBLIC PROFILE

mbb

Update Profile

Username:
mbb

Email:
dudekulahussain86@gmail.com

First name:
Hussain

Last name:
Dudekula

Bio:

Profile image:
Currently:
profile-default.jpg

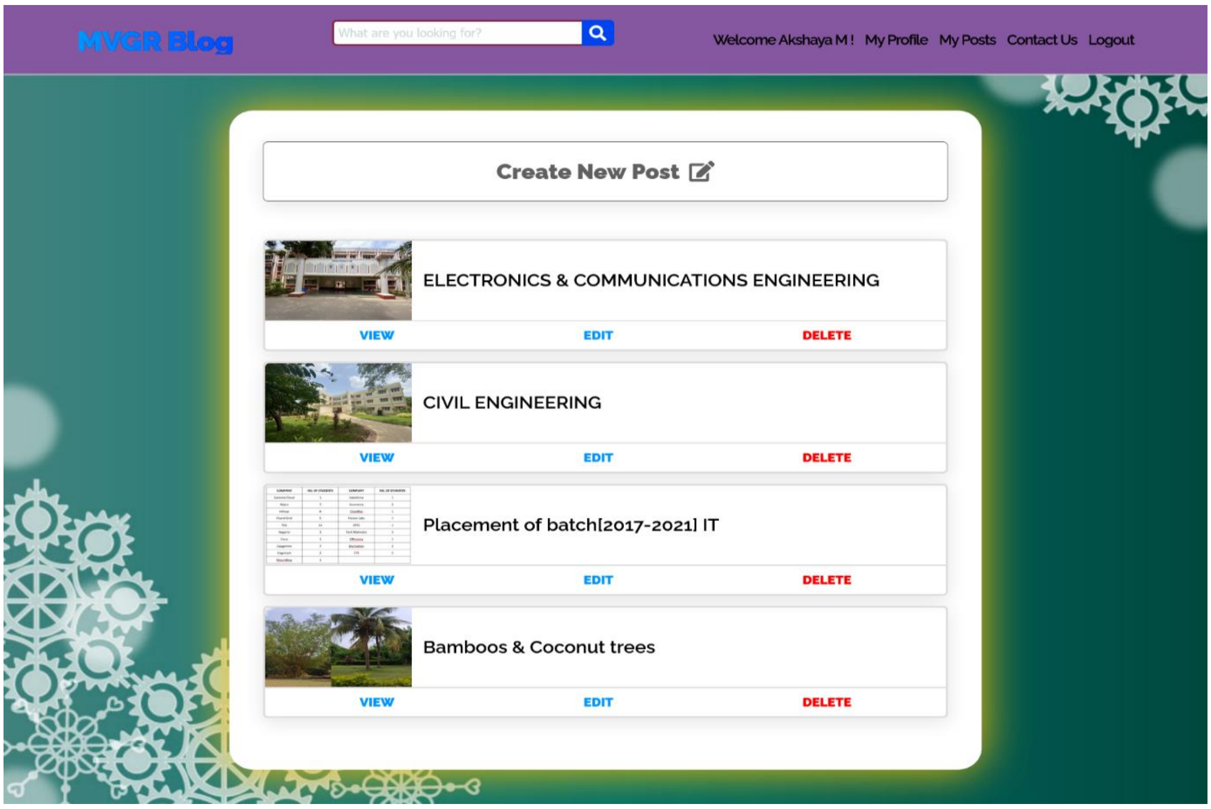
Clear
Change:
Browse... No file selected.

UPDATE

Additional Settings

CHANGE MY PASSWORD
REQUEST AUTHOR ACCOUNT

Create a now post:



MVGR Blog | What are you looking for? | Welcome Akshaya M ! | My Profile | My Posts | Contact Us | Logout

Create New Post

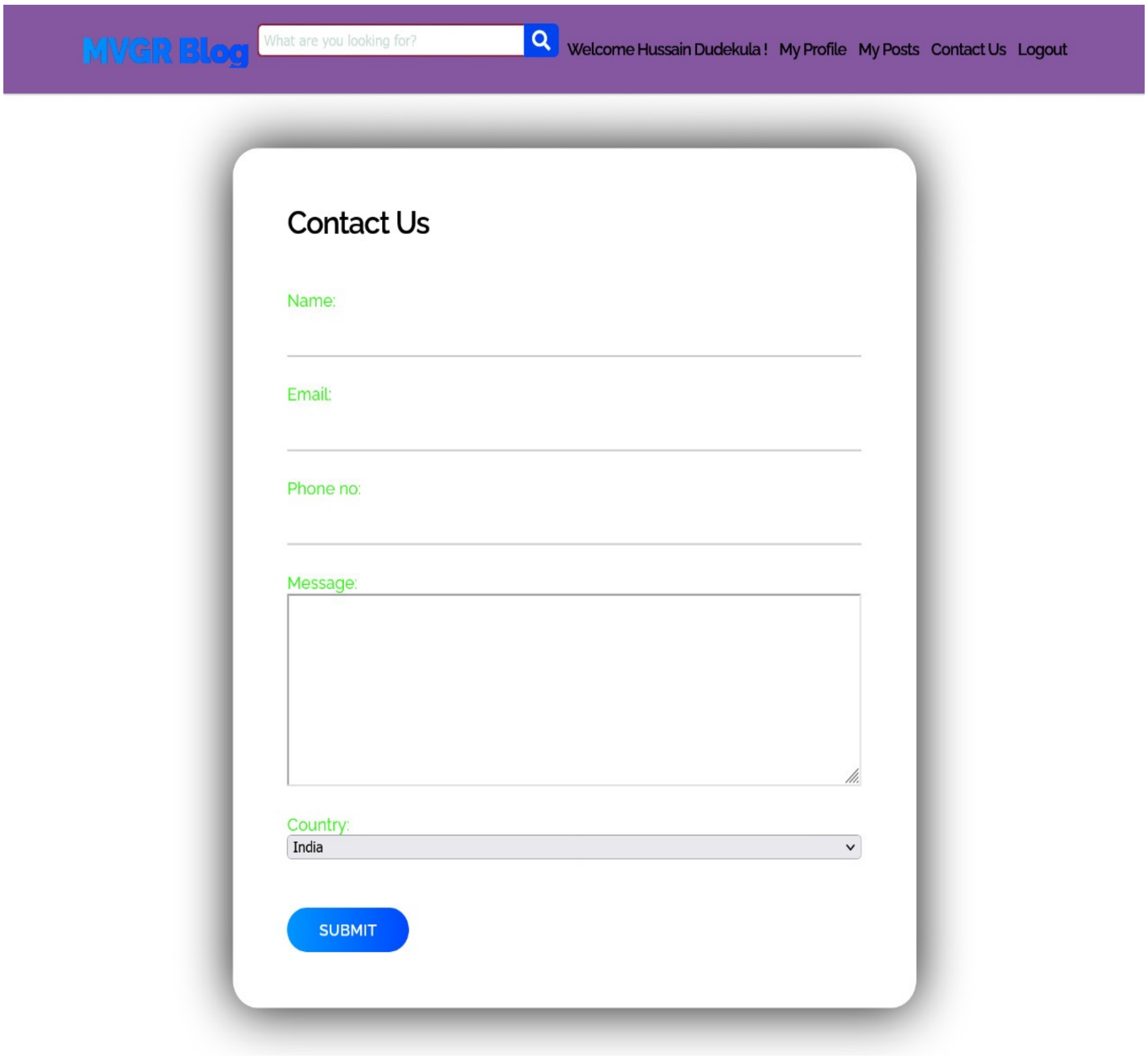
ELECTRONICS & COMMUNICATIONS ENGINEERING
VIEW EDIT DELETE

CIVIL ENGINEERING
VIEW EDIT DELETE

Placement of batch[2017-2021] IT
VIEW EDIT DELETE

Bamboos & Coconut trees
VIEW EDIT DELETE

Contact us


MVGR Blog
What are you looking for?
Welcome Hussain Dudekula ! My Profile My Posts Contact Us Logout

Contact Us

Name:

Email:

Phone no:

Message:

Country:
India

Conclusion

MVGR Blog Bundle resolves almost all such encounters of such situations upto a high extent. MVGR blog bundle is not just a website it encounters and resolves all the information regarding issues up to high extent. Users can login into website and fetch the required data and use the beneficiary programs available in the college. The people who are interested in writing would be highly beneficial using this platform as this platform also promotes students to express their writing skills in a highly collaborative and creative way. Few of the topics that are majority be chosen as the interests are college events & activities, achievements of college and its part, facts & myths of college, library details, inspiring and influencing and popular people of college, groups & clocks, flora & fauna in college, other students' or staff stories for poems, or any data donations & contributions and many. The people who are interested in writing would be highly beneficial using this platform as this platform also promotes students to express collaborate and creative ways.

References:

- Django tutorial - https://docs.djangoproject.com/en/4.0/intro/tutorial01/
- Django Settings - https://docs.djangoproject.com/en/2.2/topics/settings/
- migrate, makemigrations -https://docs.djangoproject.com/en/4.0/intro/overview/
- WSGI vs ASGI - https://medium.com/analytics-vidhya/difference-between-wsgi-and-asgi-807158ed1d4c
- Database - https://docs.djangoproject.com/en/2.2/ref/settings/#databases
- Django: https://www.djangoproject.com/
- W3 Schools – Django: https://www.w3schools.com/django/index.php
- MVGR College of Engineering: https://www.mvgrce.com/
- PostgreSQL: https://www.postgresql.org/

About PyCharm: https://hackr.io/blog/what-is-pycharm