Project Final Deliverables

BOUN-SWE 573

Spring

Anıl Mert Türemiş

2020719246

Git Repo: <https://github.com/AnilMertTuremis/bounswe573-2022.git>

Deployment URI: https://clownfish-app-fsimf.ondigitalocean.app/quora/

UserId: AnilMertTuremis

Date: 10.06.2022

gitTag: v0.9

**HONOR CODE**

Related to the submission of all the project deliverables for the Swe573 2022 Spring semester project reported in this report, **I ANIL MERT TÜREMİŞ declare that**:  
  
- I am a student in the Sofware Engineering MS program at Bogazici University and am registered for Swe573 course during the 2022 Spring semester.

- All the material that I am submitting related to my project (including but not limited to the project repository, the final project report, and supplementary documents)   
have been exclusively prepared by myself.

- I have prepared this material individually without the assistance of anyone else with the exception of permitted peer assistance which I have explicitly disclosed in this report.

TABLE OF CONTENT

ALL SOURCE COD AND RELATED CONTENTS.......................................................................... 3-34

PROJECT DETAILS................................................................................................................... 34-40

Overview.................................................................................................................. 35

Software Requirements Specification...................................................................... 36-37

Desig– UML diagrams and images........................................................................... 38

Status of your project...............................................................................................39

System manual.........................................................................................................39

User manual.............................................................................................................40

Docker-compose.ylm

version: "3.3"

services:

  app:

    build:

      context: .

    restart: always

    ports:

      - "8000:8000"

    command: >

      sh -c "python manage.py makemigrations &&

             python manage.py migrate &&

             python manage.py collectstatic --noinput &&

             python manage.py runserver 0.0.0.0:8000"

    environment:

      - DB\_HOST=db

      - DB\_NAME=app

      - DB\_USER=anil

      - DB\_PASS=locklock123

    depends\_on:

      - db

    volumes:

      - ./app:/app

      - media\_volume:/app/media

      - static\_volume:/app/static

  db:

    image: postgres:12-alpine

    restart: always

    environment:

      POSTGRES\_DB: app

      POSTGRES\_USER: anil

      POSTGRES\_PASSWORD: locklock123

volumes:

  my\_dbdata:

  media\_volume:

  static\_volume:

DockerFile

FROM python:3.7

ENV PYTHONUNBUFFERED 1

COPY ./requirements.txt /requirements.txt

RUN pip install -r /requirements.txt

EXPOSE 8000

ENV LC\_ALL=en\_US.UTF8

RUN mkdir /app

WORKDIR /app

COPY ./app /app

RUN mkdir -p /app/media

RUN mkdir -p /app/static

RUN mkdir -p /app/assets

Requirements.txt

Django==3.1.4

psycopg2-binary==2.7.7

asgi.py

"""

ASGI config for app project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/

"""

import os

from django.core.asgi import get\_asgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'app.settings')

application = get\_asgi\_application()

settings.py

"""

Django settings for app project.

Generated by 'django-admin startproject' using Django 3.1.1.

For more information on this file, see

https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/3.1/ref/settings/

"""

import os

from pathlib import Path

from django.utils.translation import gettext\_lazy as \_

# Build paths inside the project like this: BASE\_DIR / 'subdir'.

BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = "w28\_91ai\*r9-&d!2qg!1\_r7c4mvb+rtjwrogub@sb2d((ya49n"

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = ["127.0.0.1", "localhost", "0.0.0.0"]

USE\_X\_FORWARDED\_HOST = True

SECURE\_PROXY\_SSL\_HEADER = ("HTTP\_X\_FORWARDED\_PROTO", "https")

# Application definition

INSTALLED\_APPS = [

    "django.contrib.admin",

    "django.contrib.auth",

    "django.contrib.contenttypes",

    "django.contrib.sessions",

    "django.contrib.messages",

    "django.contrib.staticfiles",

    "quora.apps.QuoraConfig",

]

# AUTH\_USER\_MODEL = "user.User"

MIDDLEWARE = [

    "django.middleware.security.SecurityMiddleware",

    "django.contrib.sessions.middleware.SessionMiddleware",

    "django.middleware.locale.LocaleMiddleware",

    "django.middleware.common.CommonMiddleware",

    "django.middleware.csrf.CsrfViewMiddleware",

    "django.contrib.auth.middleware.AuthenticationMiddleware",

    "django.contrib.messages.middleware.MessageMiddleware",

    "django.middleware.clickjacking.XFrameOptionsMiddleware",

]

ROOT\_URLCONF = "app.urls"

TEMPLATES = [

    {

        "BACKEND": "django.template.backends.django.DjangoTemplates",

        "DIRS": [

            os.path.join(BASE\_DIR, "templates"),

        ],

        "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 = "app.wsgi.application"

SIGN\_UP\_FIELDS = ["first\_name", "last\_name", "email", "password1", "password2"]

# Database

# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.postgresql',

        # os.environ.get  => get variables from environment (of docker-compose)

        # all variables comes from docker-compose.yml

        'HOST': os.environ.get('DB\_HOST'),

        'NAME': os.environ.get('DB\_NAME'),

        'USER': os.environ.get('DB\_USER'),

        'PASSWORD': os.environ.get('DB\_PASS'),

    }

}

# DATABASES = {

#     "default": {

#         "ENGINE": "django.db.backends.sqlite3",

#         "NAME": BASE\_DIR / "db.sqlite3",

#     }

# }

# Password validation

# https://docs.djangoproject.com/en/3.1/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/3.1/topics/i18n/

def gettext(s):

    return s

LANGUAGES = (

    ("en", gettext("English")),

    ("tr", gettext("Turkish")),

)

TIME\_ZONE = "Europe/Istanbul"

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC\_URL = "/static/"

MEDIA\_URL = "/media/"

STATIC\_ROOT = os.path.join(BASE\_DIR, "static")

STATIC\_URL = "/static/"

MEDIA\_ROOT = os.path.join(BASE\_DIR, "media")

MEDIA\_URL = "/media/"

STATICFILES\_DIRS = [

    os.path.join(BASE\_DIR, "assets"),

]

LOGIN\_REDIRECT\_URL = "/"

url.py

"""quoraapp URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

    https://docs.djangoproject.com/en/2.2/topics/http/urls/

Examples:

Function views

    1. Add an import:  from my\_app import views

    2. Add a URL to urlpatterns:  path('', views.home, name='home')

Class-based views

    1. Add an import:  from other\_app.views import Home

    2. Add a URL to urlpatterns:  path('', Home.as\_view(), name='home')

Including another URLconf

    1. Import the include() function: from django.urls import include, path

    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))

"""

from django.conf import settings

from django.conf.urls.static import static

from django.contrib import admin

from django.urls import include, path

from django.views.generic import RedirectView

urlpatterns = [

    path("admin/", admin.site.urls),

    path("quora/", include("quora.urls")),

    path("accounts/", include("django.contrib.auth.urls")),

    path("", RedirectView.as\_view(url="/quora/")),

] + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

-----wsgi.py----

"""

WSGI config for app project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/

"""

import os

from django.core.wsgi import get\_wsgi\_application

os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'app.settings')

application = get\_wsgi\_application()

-----/migrations/0001\_initial.py-----------

# Generated by Django 3.1.4 on 2022-05-23 17:55

from django.conf import settings

from django.db import migrations, models

import django.db.models.deletion

import django.utils.timezone

class Migration(migrations.Migration):

    initial = True

    dependencies = [

        migrations.swappable\_dependency(settings.AUTH\_USER\_MODEL),

    ]

    operations = [

        migrations.CreateModel(

            name='Answer',

            fields=[

                ('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

                ('answer\_text', models.TextField(help\_text='Write your answer here...', max\_length=2000)),

                ('date\_created', models.DateField(default=django.utils.timezone.now, null=True)),

                ('date\_updated', models.DateField(default=django.utils.timezone.now, null=True)),

                ('upvote', models.IntegerField(default=0)),

                ('downvote', models.IntegerField(default=0)),

                ('views', models.IntegerField(default=0)),

            ],

        ),

        migrations.CreateModel(

            name='Author',

            fields=[

                ('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

                ('email', models.EmailField(blank=True, max\_length=70)),

                ('date\_created', models.DateField(default=django.utils.timezone.now, null=True)),

                ('credits', models.IntegerField(default=0)),

                ('user', models.ForeignKey(on\_delete=django.db.models.deletion.CASCADE, to=settings.AUTH\_USER\_MODEL)),

            ],

        ),

        migrations.CreateModel(

            name='Question',

            fields=[

                ('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

                ('question\_text', models.TextField(help\_text='Enter your question in brief', max\_length=1000)),

                ('date\_created', models.DateField(default=django.utils.timezone.now, null=True)),

                ('date\_updated', models.DateField(default=django.utils.timezone.now, null=True)),

                ('author', models.ForeignKey(null=True, on\_delete=django.db.models.deletion.SET\_NULL, to='quora.author')),

            ],

        ),

        migrations.CreateModel(

            name='Comment',

            fields=[

                ('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

                ('comment\_text', models.TextField(help\_text='Enter your comment...', max\_length=1000)),

                ('date\_created', models.DateField(default=django.utils.timezone.now)),

                ('date\_updated', models.DateField(default=django.utils.timezone.now)),

                ('answer', models.ForeignKey(null=True, on\_delete=django.db.models.deletion.SET\_NULL, to='quora.answer')),

                ('author', models.ForeignKey(null=True, on\_delete=django.db.models.deletion.SET\_NULL, to='quora.author')),

            ],

        ),

        migrations.AddField(

            model\_name='answer',

            name='author',

            field=models.ForeignKey(null=True, on\_delete=django.db.models.deletion.SET\_NULL, to='quora.author'),

        ),

        migrations.AddField(

            model\_name='answer',

            name='question',

            field=models.ForeignKey(null=True, on\_delete=django.db.models.deletion.SET\_NULL, to='quora.question'),

        ),

    ]

-----/migrations/0002\_initial.py-----------

# Generated by Django 3.1.4 on 2022-05-23 18:47

from django.db import migrations, models

import django.db.models.deletion

class Migration(migrations.Migration):

    dependencies = [

        ('quora', '0001\_initial'),

    ]

    operations = [

        migrations.CreateModel(

            name='VoteUser',

            fields=[

                ('id', models.AutoField(auto\_created=True, primary\_key=True, serialize=False, verbose\_name='ID')),

                ('option', models.CharField(choices=[('up', 'UP'), ('down', 'DOWN')], max\_length=4)),

                ('answer', models.ForeignKey(null=True, on\_delete=django.db.models.deletion.CASCADE, to='quora.answer')),

                ('author', models.ForeignKey(null=True, on\_delete=django.db.models.deletion.CASCADE, to='quora.author')),

            ],

        ),

    ]

--------/static/css/styles.css-----

.sidebar-nav {

    margin-top: 20px;

    padding: 0;

    list-style: none;

}

-------/templates/quora/answer\_form.html----

{% extends "base\_generic.html" %}

{% block content %}

<form action='/quora/question/{{q\_id}}/answer/' method="POST">

    {% csrf\_token %}

    <label for="answer\_text">Write your answer</label>

    <br>

    <br>

    <textarea name="answer\_text" id="answer\_text" cols="80" rows="10"></textarea>

    <br><br>

    <button type="submit" class="btn btn-primary">Answer</button>

</form>

{% endblock %}

-------/templates/quora/answer\_list.html----

{% extends "base\_generic.html" %}

{% block content %}

  <h1>answer List</h1>

  {% if answer\_list %}

  <ul>

    {% for answer in answer\_list %}

      <div class="row text-info">

          &nbsp; {{ answer.question.question\_text }}

      </div>

          <hr></hr>

        <div class="row text-dark">

          {{ answer.answer\_text }}</a>

        </div>

        <br>

        <br>

        <div class="row text-success">

            <div class="col"></div>

            <div class="col-6"></div>

            <div class="col">

                by <a href="{{ answer.author.get\_absolute\_url }}">{{answer.author}} </a>

                (on {{answer.date\_updated}})

            </div>

        </div>

    {% endfor %}

  </ul>

  {% else %}

    <p>There are no answers in the library.</p>

  {% endif %}

{% endblock %}

----answer\_update\_form.html---

{% extends "base\_generic.html" %}

{% block content %}

    <form method="POST">

        <div class="form-group">

            {% csrf\_token %}

        </div>

        <div class="form-group">

            <label for="id\_answer\_text">answer text:</label>

            <textarea class="form-control" name="answer\_text" cols="40" rows="10" maxlength="1000" required id="id\_answer\_text">

                    {{answer\_text}}

            </textarea>

            <br>

            <span class="helptext">Enter your answer in brief</span>

        </div>

        <button type="submit" class="btn btn-primary">Update</button>

    </form>

{% endblock %}

-------author\_confirm\_delete.html----

<form action='/quora/author/add/' method='post'>

    {% csrf\_token %}

    {{ form.as\_p }}

    <!-- <label for="email">Email</label> -->

    <!-- <input type="email" name="email" id="email"> -->

    <br><br>

    <button type="submit" class="btn btn-primary">Create</button>

</form>

------author\_detail.html----

{% extends "base\_generic.html" %}

{% block content %}

<hr></hr>

<h2>Author Details</h2>

<p>App has the following record counts:</p>

<ul>

        <li><strong>Authors:</strong> {{ author.user.username }}</li>

        <li><strong>Email:</strong> {{ author.email }}</li>

        <li><strong>Created on:</strong> {{ author.date\_created }}</li>

        <li><strong>Credits:</strong> {{ author.credits }}</li>

</ul>

{% endblock %}

------author\_form.html------

{% extends "base\_generic.html" %}

{% block content %}

<br>

<form action='/quora/author/add/' method='post'>

    {% csrf\_token %}

    <div class="form-group">

        <label for="id\_username">Username:</label>

        <input type="text" name="username" class="form-control" autofocus required id="id\_username">

        <span class="helptext">Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.</span>

    </div>

    <div class="form-group"></div>

    <label for="email">Your Email:</label>

    <input type="email" name="email" id="email" class="form-control" aria-describedby="emailHelp"

        placeholder="Enter email" required>

    <div class="form-group">

        <label for="id\_password1">Password:</label>

        <input type="password" name="password1" required id="id\_password1" class="form-control"

            placeholder="Enter Password">

        <span class="helptext">

            <ul>

                <li>Your password can&#39;t be too similar to your other personal information.</li>

                <li>Your password must contain at least 8 characters.</li>

                <li>Your password can&#39;t be a commonly used password.</li>

                <li>Your password can&#39;t be entirely numeric.</li>

            </ul>

        </span>

    </div>

    <div class="form-group">

        <label for="id\_password2">Password confirmation:</label>

        <input type="password" name="password2" required id="id\_password2" class="form-control"

            placeholder="Confirm Password">

        <span class="helptext">Enter the same password as before, for verification.</span>

    </div>

    <button type="submit" class="btn btn-primary">Register</button>

</form>

{% endblock %}

----comment\_form.html----

{% extends "base\_generic.html" %}

{% block content %}

<form action='/quora/answer/{{a\_id}}/comment/' method="post">

    {% csrf\_token %}

    <label for="comment\_text">Write your comment: </label>

    <br>

    <br>

    <textarea name="comment\_text" id="comment\_text" cols="80" rows="8" required></textarea>

    <br><br>

    <button type="submit" class="btn btn-primary">Comment</button>

</form>

{% endblock %}

----question\_detail.html----

{% extends "base\_generic.html" %}

{% block content %}

<div style="margin-left: 5px;" class="row text-info">

  <h3>Que: {{ question.question\_text }} </h3>

</div>

<div class="text-success">

  <div class="row">

    <div class="col">

      <form action="{{answer\_url}}" method="GET">

        <input type="hidden" name="q\_id" value={{q\_id}} />

        <input type="submit" class="btn btn-success" role="button" value="Answer this question">

      </form>

    </div>

    <div class="col-5"></div>

    <div class="col">

      <strong>Author:</strong> <a href="{{ question.author.get\_absolute\_url }}">{{ question.author }}</a>

      <br>

      <strong>Created On:</strong> {{ question.date\_created }}

    </div>

  </div>

  <div style="margin-left:20px;margin-top:10px; margin-right: 10px;">

    {% if answer\_list %}

    <ul>

      <h4>All Answers</h4>

      {% for answer in answer\_list %}

      <br>

      <div class="card">

        <div class="card-body">

          {% comment %} <h5 class="card-title">Special title treatment</h5> {% endcomment %}

          <p class="card-text">

          <div class="row text-dark">

            {{ answer.answer\_text }}

          </div>

          <br>

          <br>

          <div class="row text-success">

            <!-- upvote form -->

            <div class="col">

              <form action="{{upvote\_url}}{{answer.id}}" method="POST">

                {% csrf\_token %}

                <input type="submit" value="Upvote" class="btn btn-primary">

                <span class="text-dark">{{ answer.upvote }}</span>

              </form>

            </div>

            <!-- downvote form -->

            <div class="col">

              <form action="{{downvote\_url}}{{answer.id}}" method="POST">

                {% csrf\_token %}

                <input type="submit" value="Downvote" class="btn btn-primary">

                <span class="text-dark">{{ answer.downvote }}</span>

              </form>

            </div>

            <!-- comment form -->

            <div class="col">

              <form action="/quora/answer/{{answer.id}}/comment/" method="GET">

                {% csrf\_token %}

                <input type="submit" value="Comment" class="btn btn-primary">

              </form>

            </div>

            {% comment %} update answer {% endcomment %}

            <div class="col-4">

              {% if user.is\_authenticated %}

              {% if user.get\_username == answer.author.user.username %}

              <a href="/quora/answer/{{answer.id}}/update/" class="btn btn-primary">Update Answer</a>

              {% endif %}

              {% endif %}

            </div>

            <div class="col-3">

              by <a href="{{ answer.author.get\_absolute\_url }}">{{answer.author}} </a>

              (on {{answer.date\_updated}})

            </div>

            </p>

            {% comment %} <a style="padding: 3px;" class="btn-success"

              href="/quora/question/{{question.id}}/update/">Update Question</a> {% endcomment %}

          </div>

        </div>

      </div>

      <div style="margin-left: 20px;">

        <br>

        <ul class="list-group">

          {% for key, comment in comment\_dictionary.items %}

          {% if key == answer.id %}

          {% if comment %}

          {% for comm in comment %}

          <li class="list-group-item disabled">{{ comm }}</li>

          {% endfor %}

          {% endif %}

          {% endif %}

          {% endfor %}

          <ul>

      </div>

      {% endfor %}

    </ul>

    {% else %}

    <div class="text-danger">

      There are no answers for this question.

    </div>

    {% endif %}

  </div>

  {% endblock %}

----question\_form.html-----

{% extends "base\_generic.html" %}

{% block content %}

<form action='/quora/question/add/' method="post">

    {% csrf\_token %}

    <label for="question\_text">Question Title: </label>

    <br>

    <br>

    <textarea name="question\_text" id="question\_text" cols="80" rows="10" required></textarea>

    <br><br>

    <button type="submit" class="btn btn-primary">Ask</button>

</form>

{% endblock %}

----question\_list.html-----

{% extends "base\_generic.html" %}

{% block content %}

<h1>Question List</h1>

{% if question\_list %}

<ul class="list-group">

  {% for question in question\_list %}

  <li style="padding: 4px; margin: 3px; list-style: none; text-decoration: none;"

    class="list-group-item list-group-item-action list-group-item-dark">

    <a style="text-decoration: none; color: black" href="{{ question.get\_absolute\_url }}">{{ question.question\_text }}

      </a>

    &nbsp; &nbsp; &nbsp;

    by

    <a style="text-decoration: none; color: black;" href="{{ question.author.get\_absolute\_url }}"> {{question.author}}

    </a>

    (on {{question.date\_updated}})

    {% if user.is\_authenticated %}

    {% if user.get\_username == question.author.user.username %}

    <a style="padding: 3px; text-decoration: none;" class="btn btn-primary"

      href="/quora/question/{{question.id}}/update/">Update Question</a>

    {% endif %}

    {% endif %}

  </li>

  {% endfor %}

</ul>

{% else %}

<p>There are no questions in the library.</p>

{% endif %}

{% endblock %}

----question\_update\_form.html---

{% extends "base\_generic.html" %}

{% block content %}

<form method="POST">

    <div class="form-group">

        {% csrf\_token %}

    </div>

    <div class="form-group">

        <label for="id\_question\_text">Question text:</label>

        <textarea class="form-control" name="question\_text" cols="10" rows="10" maxlength="1000" required id="id\_question\_text">

                {{question\_text}}

        </textarea>

        <br>

        <span class="helptext">Enter your question in brief</span>

    </div>

    <button type="submit" class="btn btn-primary">Update</button>

</form>

{% endblock %}

-----base\_generic.html---

<!DOCTYPE html>

<html lang="en">

<head>

  {% block title %}<title>My Platform web app</title>{% endblock %}

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

  <!-- Add additional CSS in static file -->

  {% load static %}

  <link rel="stylesheet" href="{% static 'css/styles.css' %}">

</head>

<body>

  <div class="container-fluid">

    <div class="row">

      <div style="background: gray; height: 800px;" class="col-sm-2">

      {% block sidebar %}

        <ul class="sidebar-nav">

          {% if user.is\_authenticated %}

            <li  style="text-decoration: none; color: rgb(255, 255, 255);"  >User: {{ user.get\_username }}</li>

            <li><a  style="text-decoration: none; color: rgb(255, 255, 255);"  href="{% url 'logout'%}?next={{request.path}}">Logout</a></li>

          {% else %}

          <li><a style="text-decoration: none; color: rgb(255, 255, 255);"   href="{% url 'login'%}?next={{request.path}}">Login</a></li>

          <li><a style="text-decoration: none; color: rgb(255, 255, 255);"  href="{% url 'author-add'%}?next={{request.path}}">Register</a></li>

          {% endif %}

          <hr>

          <li><a style="text-decoration: none; color: rgb(65, 218, 218);"  href="{% url 'index' %}">Home</a></li>

          <li><a  style="text-decoration: none; color: rgb(46, 193, 204);"  href="{% url 'questions' %}">All Questions</a></li>

          <li><a style="text-decoration: none; color: rgb(25, 198, 211);"   href="{% url 'question-add' %}">Ask Question</a></li>

        </ul>

     {% endblock %}

      </div>

    <br>

    <br>

    <br>

    <div class="col-sm-10 ">

      {% block content %}

      {% endblock %}

      {% block pagination %}

      {% if is\_paginated %}

          <div class="pagination">

              <span class="page-links">

                  {% if page\_obj.has\_previous %}

                      <a href="{{ request.path }}?page={{ page\_obj.previous\_page\_number }}">previous</a>

                  {% endif %}

                  <span class="page-current">

                      Page {{ page\_obj.number }} of {{ page\_obj.paginator.num\_pages }}.

                  </span>

                  {% if page\_obj.has\_next %}

                      <a href="{{ request.path }}?page={{ page\_obj.next\_page\_number }}">next</a>

                  {% endif %}

              </span>

          </div>

      {% endif %}

    {% endblock %}

    </div>

  </div>

</body>

</html>

------index.html---

{% extends "base\_generic.html" %}

{% block content %}

<div class="jumbotron jumbotron-fluid">

    <div class="container">

        <h1>My Learning App</h1>

        <hr></hr>

        <h2>App Data</h2>

        <p>App has the following record counts:</p>

        <ul>

          <li><strong>Authors:</strong> {{ num\_authors }}</li>

          <li><strong>Questions:</strong> {{ num\_questions }}</li>

          <li><strong>Answers:</strong> {{ num\_answers }}</li>

          <li><strong>Comments:</strong> {{ num\_comments }}</li>

        </ul>

     </div>

    </div>

{% endblock %}

--------admin.py------

from django.contrib import admin

# Register your models here.

from .models import (

    Answer,

    AnswerAdmin,

    Author,

    AuthorAdmin,

    Comment,

    CommentAdmin,

    Question,

    QuestionAdmin,

)

admin.site.register(Author, AuthorAdmin)

admin.site.register(Question, QuestionAdmin)

admin.site.register(Answer, AnswerAdmin)

admin.site.register(Comment, CommentAdmin)

----apps.py----

from django.apps import AppConfig

class QuoraConfig(AppConfig):

    name = "quora"

------models.py------

from django.contrib import admin

from django.contrib.auth.models import User

from django.db import models

from django.urls import reverse

from django.utils import timezone

class Author(models.Model):

    """Model representing an author."""

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

    email = models.EmailField(max\_length=70, blank=True)

    date\_created = models.DateField(null=True, default=timezone.now)

    credits = models.IntegerField(default=0)

    def get\_absolute\_url(self):

        """Returns the url to access a particular author instance."""

        return reverse("author-detail", args=[str(self.id)])

        # return reverse('author-detail', kwargs={'pk': self.pk})

        # another way of doing this

    def \_\_str\_\_(self):

        """String for representing the Model object."""

        return f"{self.user.username}"

class AuthorAdmin(admin.ModelAdmin):

    list\_display = ("user", "email", "date\_created", "credits")

class Comment(models.Model):

    """Model representing a question."""

    author = models.ForeignKey("Author", on\_delete=models.SET\_NULL, null=True)

    answer = models.ForeignKey("Answer", on\_delete=models.SET\_NULL, null=True)

    comment\_text = models.TextField(max\_length=1000, help\_text="Enter your comment...")

    date\_created = models.DateField(default=timezone.now)

    date\_updated = models.DateField(default=timezone.now)

    def \_\_str\_\_(self):

        """String for representing the Model object."""

        return f"{self.comment\_text}"

class CommentAdmin(admin.ModelAdmin):

    list\_display = ("comment\_text", "author", "date\_created")

    list\_filter = ("date\_created", "author")

class Answer(models.Model):

    """Model representing a question."""

    author = models.ForeignKey(Author, on\_delete=models.SET\_NULL, null=True)

    question = models.ForeignKey("Question", on\_delete=models.SET\_NULL, null=True)

    answer\_text = models.TextField(

        max\_length=2000, help\_text="Write your answer here..."

    )

    date\_created = models.DateField(null=True, default=timezone.now)

    date\_updated = models.DateField(null=True, default=timezone.now)

    upvote = models.IntegerField(default=0)

    downvote = models.IntegerField(default=0)

    views = models.IntegerField(default=0)

    def \_\_str\_\_(self):

        """String for representing the Model object."""

        return (

            f"Que: {self.question.question\_text[:50]}.. Ans: {self.answer\_text[:50]}.."

        )

class VoteOption(models.TextChoices):

    up = "up", "UP"

    down = "down", "DOWN"

class VoteUser(models.Model):

    author = models.ForeignKey(Author, on\_delete=models.CASCADE, null=True)

    answer = models.ForeignKey(Answer, on\_delete=models.CASCADE, null=True)

    option = models.CharField(max\_length=4, choices=VoteOption.choices)

class AnswerAdmin(admin.ModelAdmin):

    list\_display = (

        "question",

        "answer\_text",

        "author",

        "date\_created",

        "upvote",

        "downvote",

        "views",

    )

    list\_filter = ("date\_created", "author", "upvote", "downvote", "views")

    fieldsets = (

        (None, {"fields": ("author", "question", "answer\_text")}),

        ("Dates", {"fields": ("date\_created", "date\_updated")}),

        ("Actions", {"fields": ("upvote", "downvote", "views")}),

    )

class Question(models.Model):

    """Model representing a question."""

    author = models.ForeignKey(Author, on\_delete=models.SET\_NULL, null=True)

    question\_text = models.TextField(

        max\_length=1000, help\_text="Enter your question in brief"

    )

    date\_created = models.DateField(null=True, default=timezone.now)

    date\_updated = models.DateField(null=True, default=timezone.now)

    def get\_absolute\_url(self):

        """Returns the url to access a particular question and its answer."""

        return reverse("question-detail", args=[str(self.id)])

    def \_\_str\_\_(self):

        """String for representing the Model object."""

        return f"{self.question\_text}"

class QuestionAdmin(admin.ModelAdmin):

    list\_display = ("question\_text", "author", "date\_created")

    list\_filter = ("date\_created", "author")

----urls.py----

from django.urls import path

from . import views

urlpatterns = [

    path('', views.index, name='index'),

    path('question/add/', views.QuestionCreate.as\_view(), name='question-add'),

    path('questions/', views.QuestionListView.as\_view(), name='questions'),

    path('answers/', views.AnswerListView.as\_view(), name='answers'),

    path('question/<int:pk>', views.QuestionDetailView.as\_view(), name='question-detail'),

    path('question/<int:pk>/answer/', views.AnswerCreate.as\_view(), name='answer-add'),

    path('question/<int:pk>/update/', views.UpdateQuestion.as\_view(), name='question-update'),

    path('answer/<int:pk>/update/', views.UpdateAnswer.as\_view(), name='answer-update'),

    path('answer/<int:pk>/comment/', views.CommentCreate.as\_view(), name='comment-add'),

    path('answer/upvote/<int:pk>', views.UpvoteCreate.as\_view(), name='answer-upvote'),

    path('answer/downvote/<int:pk>', views.DownvoteCreate.as\_view(), name='answer-downvote'),

    path('author/<int:pk>', views.AuthorDetailView.as\_view(), name='author-detail'),

    path('author/add/', views.AuthorCreate.as\_view(), name='author-add'),

    path('author/<int:pk>/', views.AuthorUpdate.as\_view(), name='author-update')

]

------views.py----

from django.contrib.auth.hashers import PBKDF2PasswordHasher

from django.contrib.auth.mixins import LoginRequiredMixin

from django.contrib.auth.models import User

from django.shortcuts import redirect, render

from django.urls import reverse, reverse\_lazy

from django.views import generic

from django.views.generic.edit import CreateView, DeleteView, UpdateView

from .models import Answer, Author, Comment, Question, VoteOption, VoteUser

class UpdateAnswer(LoginRequiredMixin, UpdateView):

    model = Answer

    fields = ["answer\_text"]

    template\_name = "quora/answer\_update\_form.html"

    def get\_context\_data(self, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        answer\_id = self.kwargs["pk"]

        answer = Answer.objects.get(id=answer\_id)

        self.pk = answer.question.id

        context["answer\_text"] = answer.answer\_text

        return context

    def get\_success\_url(self):

        return reverse("question-detail", kwargs={"pk": self.object.question.id})

class UpdateQuestion(LoginRequiredMixin, UpdateView):

    model = Question

    fields = ["question\_text"]

    success\_url = reverse\_lazy("questions")

    template\_name = "quora/question\_update\_form.html"

    def get\_context\_data(self, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        question\_id = self.kwargs["pk"]

        question = Question.objects.get(id=question\_id)

        context["question\_text"] = question.question\_text

        return context

class CommentCreate(LoginRequiredMixin, CreateView):

    model = Comment

    fields = ["author", "answer", "comment\_text"]

    def get(self, request, \*args, \*\*kwargs):

        a\_id = self.kwargs["pk"]

        return render(request, "quora/comment\_form.html", {"a\_id": a\_id})

    def post(self, request, \*args, \*\*kwargs):

        comment\_text = request.POST.get("comment\_text")

        params = self.kwargs["pk"]

        author = Author.objects.get(user=self.request.user)

        answer = Answer.objects.get(id=params)

        Comment.objects.create(author=author, answer=answer, comment\_text=comment\_text)

        response = redirect(

            reverse("question-detail", kwargs={"pk": answer.question.id})

        )

        return response

class UpvoteCreate(LoginRequiredMixin, CreateView):

    model = Answer

    fields = ["answer\_text", "id", "upvote"]

    def post(self, request, \*args, \*\*kwargs):

        answer\_id = self.kwargs["pk"]

        answer = Answer.objects.get(id=answer\_id)

        author = Author.objects.get(user=self.request.user)

        is\_voted = VoteUser.objects.filter(answer=answer, author=author).first()

        if is\_voted:

            if is\_voted.option != VoteOption.up:

                answer.downvote -= 1

                answer.upvote += 1

                is\_voted.option = VoteOption.up

                is\_voted.save()

        else:

            VoteUser.objects.create(answer=answer, author=author, option=VoteOption.up)

            answer.upvote += 1

        answer.save()

        response = redirect(

            reverse("question-detail", kwargs={"pk": answer.question.id})

        )

        return response

class DownvoteCreate(LoginRequiredMixin, CreateView):

    model = Answer

    fields = ["answer\_text", "id", "downvote"]

    def post(self, request, \*args, \*\*kwargs):

        answer\_id = self.kwargs["pk"]

        answer = Answer.objects.get(id=answer\_id)

        author = Author.objects.get(user=self.request.user)

        is\_voted = VoteUser.objects.filter(answer=answer, author=author).first()

        print(is\_voted.option)

        if is\_voted:

            if is\_voted.option != VoteOption.down:

                answer.upvote -= 1

                answer.downvote += 1

                is\_voted.option = VoteOption.down

                is\_voted.save()

        else:

            VoteUser.objects.create(answer=answer, author=author, option=VoteOption.down)

            answer.upvote += 1

        answer.save()

        response = redirect(

            reverse("question-detail", kwargs={"pk": answer.question.id})

        )

        return response

class QuestionCreate(LoginRequiredMixin, CreateView):

    model = Question

    fields = ["question\_text", "credits"]

    def get(self, request, \*args, \*\*kwargs):

        return render(request, "quora/question\_form.html")

    def post(self, request, \*args, \*\*kwargs):

        question\_text = request.POST.get("question\_text")

        Question.objects.create(

            author=Author.objects.get(user=self.request.user),

            question\_text=question\_text,

        )

        response = redirect("/quora/questions")

        return response

class AnswerCreate(LoginRequiredMixin, CreateView):

    model = Answer

    fields = ["answer\_text"]

    def get(self, request, \*args, \*\*kwargs):

        q\_id = self.kwargs["pk"]

        return render(request, "quora/answer\_form.html", {"q\_id": q\_id})

    def post(self, request, \*args, \*\*kwargs):

        answer\_text = request.POST.get("answer\_text")

        answer = Answer.objects.create(

            author=Author.objects.get(user=self.request.user),

            question=Question.objects.get(id=self.kwargs["pk"]),

            answer\_text=answer\_text,

        )

        response = redirect(

            reverse("question-detail", kwargs={"pk": answer.question.id})

        )

        return response

class AuthorCreate(CreateView):

    model = Author

    fields = ["email", "credits"]

    def get(self, request, \*args, \*\*kwargs):

        return render(request, "quora/author\_form.html")

    def post(self, request, \*args, \*\*kwargs):

        username = request.POST.get("username")

        password1 = request.POST.get("password1")

        password2 = request.POST.get("password2")

        email = request.POST.get("email")

        if password1 == password2:

            hasher = PBKDF2PasswordHasher()

            password = hasher.encode(password=password1, salt="salt", iterations=150000)

            user = User.objects.create(username=username, password=password)

            Author.objects.create(

                user=user,

                email=email,

            )

            response = redirect("/accounts/login/")

            return response

        else:

            return render(request, "quora/author\_form.html")

class AuthorUpdate(UpdateView):

    model = Author

    fields = ["email"]

class AuthorDelete(DeleteView):

    model = Author

    fields = ["user"]

    fields = ["email"]

    fields = ["credits"]

    success\_url = reverse\_lazy("questions")

class AuthorDetailView(generic.DetailView):

    """Generic class-based detail view for an Author."""

    model = Author

class QuestionDetailView(generic.DetailView):

    """Generic class-based detail view for a Question."""

    model = Question

    def get\_context\_data(self, \*\*kwargs):

        context = super().get\_context\_data(\*\*kwargs)

        question\_id = self.kwargs["pk"]

        question = Question.objects.get(id=question\_id)

        answer\_list = Answer.objects.filter(question=question)

        comment\_dictionary = {

            ans.id: Comment.objects.filter(answer=ans) for ans in answer\_list

        }

        context["answer\_list"] = answer\_list

        context["answer\_url"] = "/quora/question/" + str(question\_id) + "/answer/"

        context["upvote\_url"] = "/quora/answer/upvote/"

        context["downvote\_url"] = "/quora/answer/downvote/"

        context["comment\_dictionary"] = comment\_dictionary

        return context

class AnswerListView(generic.ListView):

    model = Answer

    paginate\_by = 3

class QuestionListView(generic.ListView):

    model = Question

    paginate\_by = 10

def index(request):

    """View function for home page of site."""

    num\_questions = Question.objects.all().count()

    num\_answers = Answer.objects.all().count()

    num\_authors = Author.objects.count()

    num\_comments = Comment.objects.count()

    context = {

        "num\_questions": num\_questions,

        "num\_answers": num\_answers,

        "num\_authors": num\_authors,

        "num\_comments": num\_comments,

    }

    return render(request, "index.html", context=context)

-----manage.py---

#!/usr/bin/env python

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main():

    """Run administrative tasks."""

    os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'app.settings')

    try:

        from django.core.management import execute\_from\_command\_line

    except ImportError as exc:

        raise ImportError(

            "Couldn't import Django. Are you sure it's installed and "

            "available on your PYTHONPATH environment variable? Did you "

            "forget to activate a virtual environment?"

        ) from exc

    execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

----------/templates/rregistration/logged\_out.html---

{% extends "base\_generic.html" %}

{% block content %}

  <p>Logged out!</p>

  <a href="{% url 'login'%}">Click here to login again.</a>

{% endblock %}

------login.html-----

{% extends "base\_generic.html" %}

{% block content %}

{% if form.errors %}

  <p>Your username and password didn't match. Please try again.</p>

{% endif %}

{% if next %}

  {% if user.is\_authenticated %}

    <p>Your account doesn't have access to this page. To proceed,

    please login with an account that has access.</p>

  {% else %}

    <p>Please login to see this page.</p>

  {% endif %}

{% endif %}

<form method="post" action="{% url 'login' %}">

{% csrf\_token %}

<table>

<tr>

  <td>{{ form.username.label\_tag }}</td>

  <td>{{ form.username }}</td>

</tr>

<tr>

  <td>{{ form.password.label\_tag }}</td>

  <td>{{ form.password }}</td>

</tr>

</table>

<input type="submit" value="login" />

<input type="hidden" name="next" value="{{ next }}" />

</form>

{# Assumes you setup the password\_reset view in your URLconf #}

<p><a href="{% url 'password\_reset' %}">Lost password?</a></p>

{% endblock %}

-----passaport\_reset\_complete.html-----

{% extends "base\_generic.html" %}

{% block content %}

  <h1>The password has been changed!</h1>

  <p><a href="{% url 'login' %}">log in again?</a></p>

{% endblock %}

------passaport\_resert\_confirm.html---

{% extends "base\_generic.html" %}

{% block content %}

    {% if validlink %}

        <p>Please enter (and confirm) your new password.</p>

        <form action="" method="post">

        {% csrf\_token %}

            <table>

                <tr>

                    <td>{{ form.new\_password1.errors }}

                        <label for="id\_new\_password1">New password:</label></td>

                    <td>{{ form.new\_password1 }}</td>

                </tr>

                <tr>

                    <td>{{ form.new\_password2.errors }}

                        <label for="id\_new\_password2">Confirm password:</label></td>

                    <td>{{ form.new\_password2 }}</td>

                </tr>

                <tr>

                    <td></td>

                    <td><input type="submit" value="Change my password" /></td>

                </tr>

            </table>

        </form>

    {% else %}

        <h1>Password reset failed</h1>

        <p>The password reset link was invalid, possibly because it has already been used. Please request a new password reset.</p>

    {% endif %}

{% endblock %}

------passaport\_reset\_form.html---

{% extends "base\_generic.html" %}

{% block content %}

  <form action="" method="post">

  {% csrf\_token %}

  {% if form.email.errors %}

    {{ form.email.errors }}

  {% endif %}

      <p>{{ form.email }}</p>

    <input type="submit" class="btn btn-default btn-lg" value="Reset password">

  </form>

{% endblock %}

OVERVIEW OF PROJECT

Software Requirements Specification

-Users can create analytics and reporting

-Users can create new contents and customize their own page

-System has core learning tools for new users

-Users can create new course or blog and manage the contents

-System allows users to integrate their courses in a single page

-Users should sign-in with e mail and password

-System provides file share module to transfer files from a user to another

-System has chat box and mail box options for users

-System allows users to live video share system for social learning

-System stores the users data in temporary databases

-System recommends some course or event for users in terms of their interests

-Systems reminds user by e-mail for closing deadlines

-As a user,I want to ask questions on my home page to all my connections

-As a user,I want to create quiz tests and games for competition among my connections

-As a user,I want to send reward to my quiz winners

-As a user,I want to block some users who bothers me

-As a user,I want to access all my learning courses on my home page

-As a user,I want to communicate all users who are taking the same course

-As a user,I want to filter all courses in the platform by my specifications

-As a user,I want to see my learning scores in the table that shows all my co-learners score

Design– UML diagrams and images

Diagram

Description automatically generated

Diagram

Description automatically generated

Mock-Up Screens

Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

Status of project

Most of the requirements are satisfied but project is not deployed and test cases are not applied yet.

Status of Deployment

Project is dockerized but did not deployed to any web server yet.

System Manual

Because the project is dockerized, all dependecies and required applications will be build when dockerized project file is run.

To run dockerized project follow the instractions:

1. Get the clone of the project repository in local machine

“git clone <https://github.com/AnilMertTuremis/bounswe573-2022.git>”

2)Now docker container can be started and all required apps and dependencies will be build up

“docker-compose build”

1. The following command is run server in detached mod.

“docker-compose up -d”

User Manual

The project is a co-learning platform that users can post a question and can answer to posted questions.Users can also rate the answers that is correct and suitable to related post.

1)Create a new user:

Two types of user profile can be created.First one is standard user.This type of user profile can access all posted questios and can answer all of them.Standard user can be created to follow “New User” button on the home page.

Admin user can be created on the application docker terminal via using following commands:

“docker-compose run --rm app python manage.py createsuperuser”

2)Create New Post:

After creating a new user, on left column of the main page “ask question” button directs to new post creation page. First the title of the question is asked then the main part of question is asked.

3)Commend or answer the questions:

On the main page , “Answer the post” button is placed.After this link user can see all of the post that are asked so far.By clicking the post title user can give command or vote the answers.