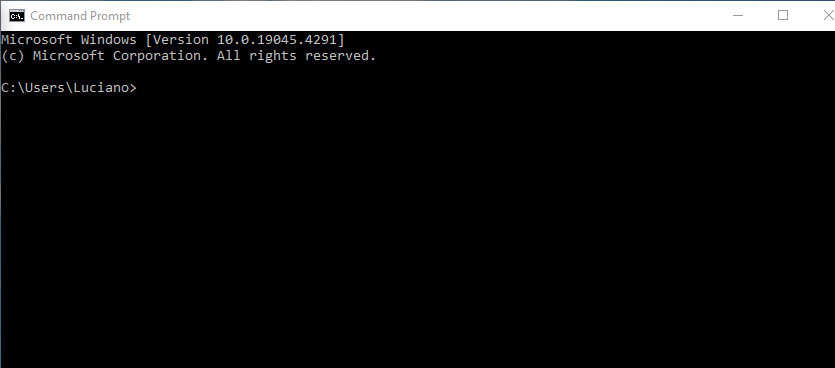
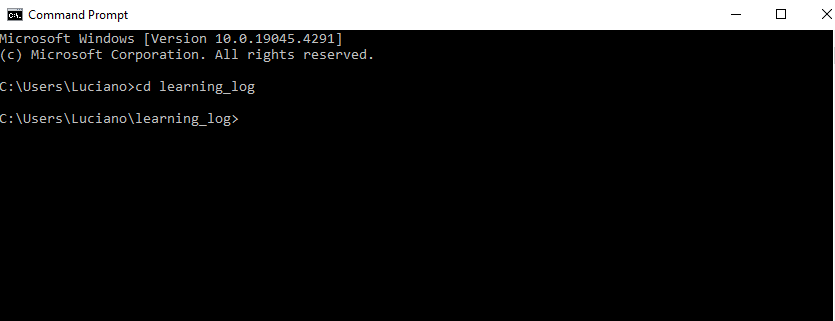
# Final Project – Web Application

Learning Log is an online journal system that lets you keep track of information you’ve   
learned about different topics.

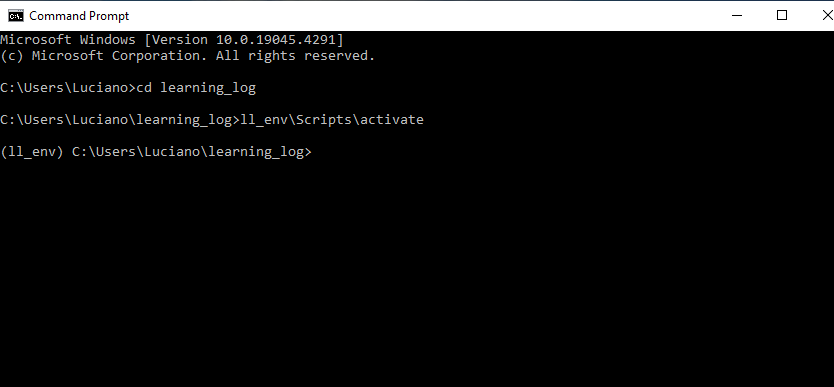
First we have to open a Command Prompt so we can run the program in our local server.



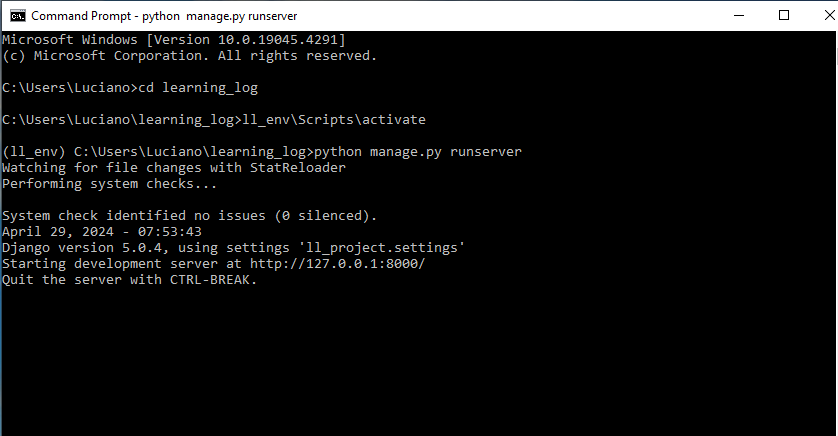
Type: **cd learning\_log** and press enter



Type: **ll\_env\Scripts\activate** and press enter

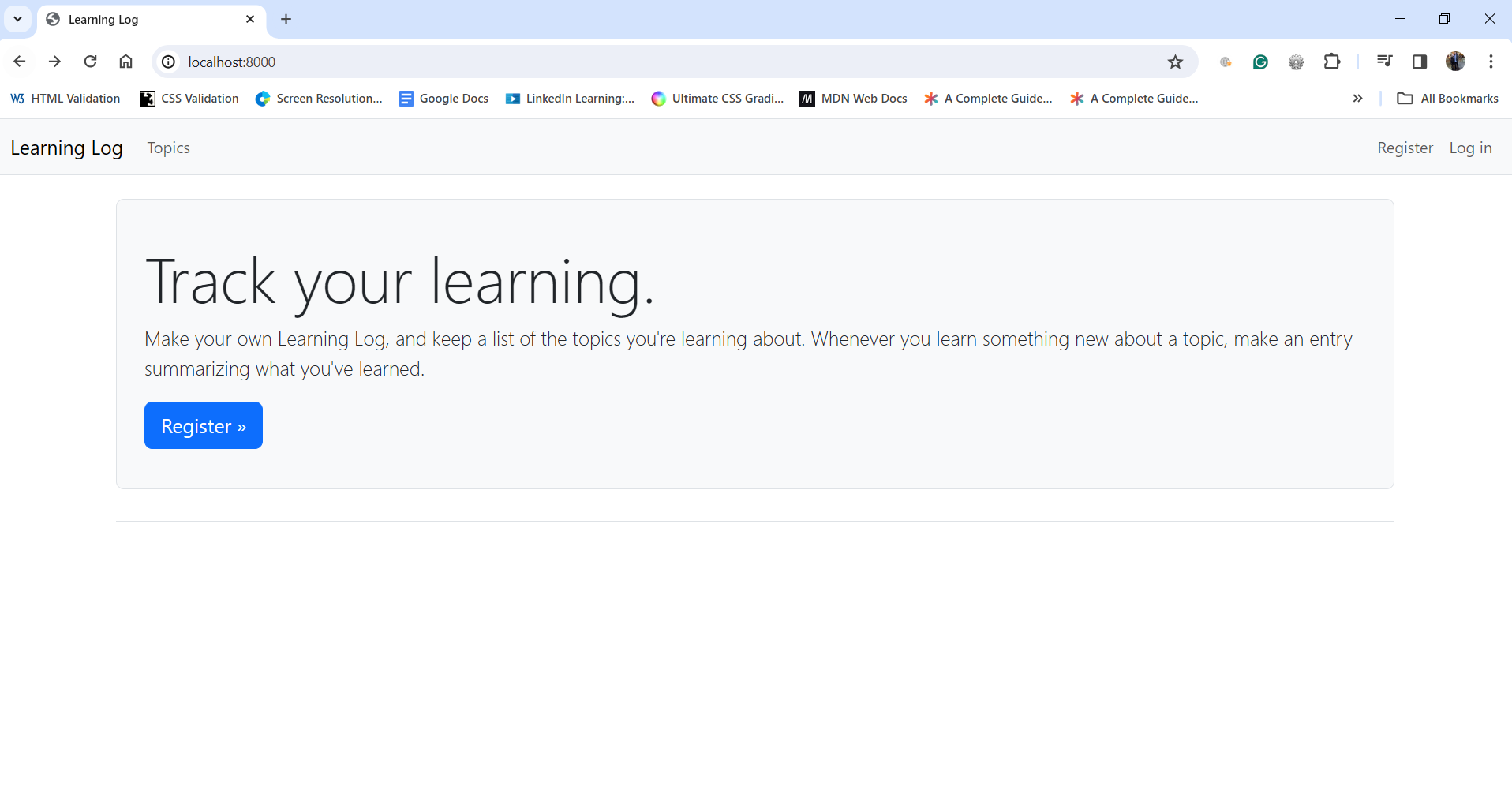


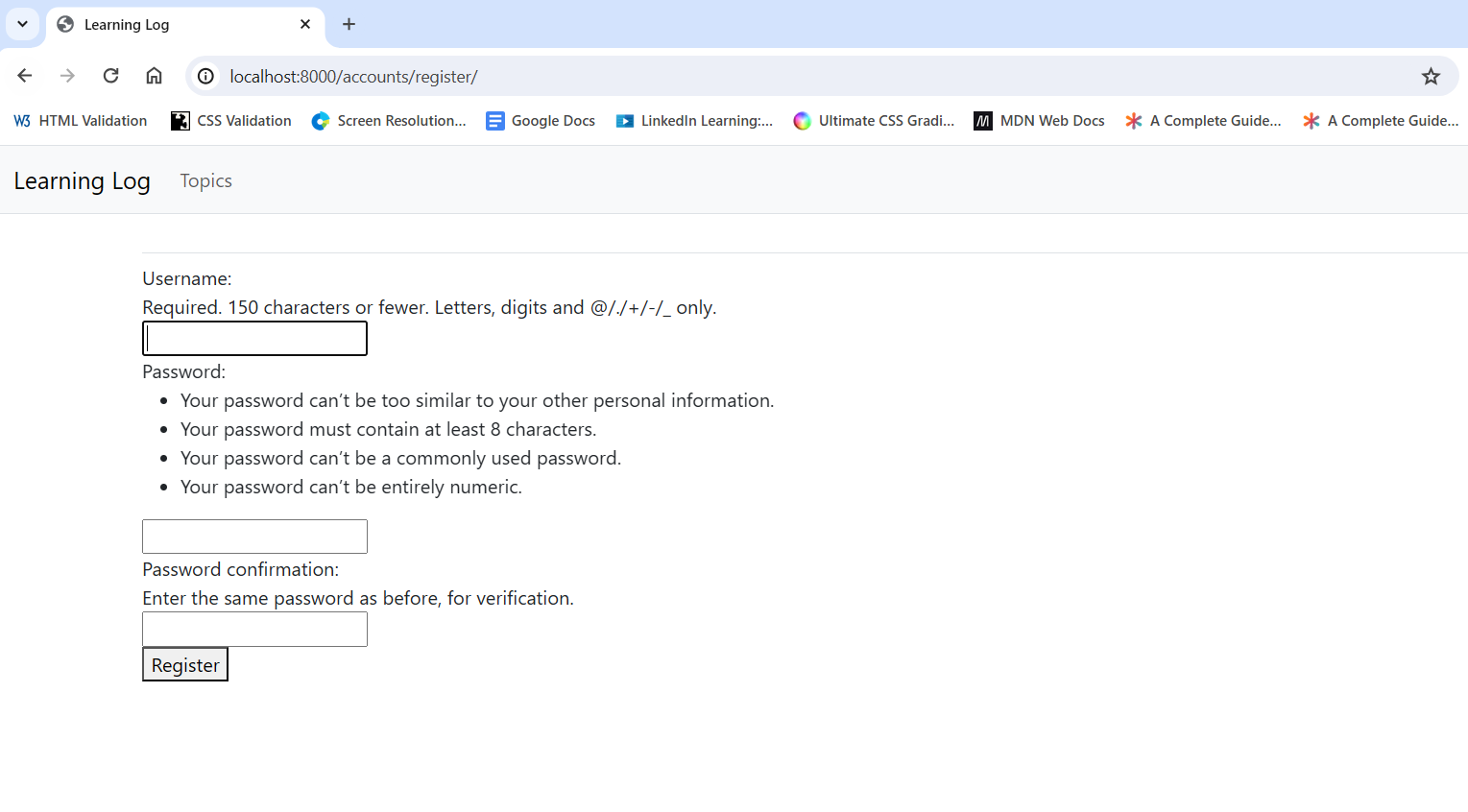
Type: **python manage.py runserver** and press enter

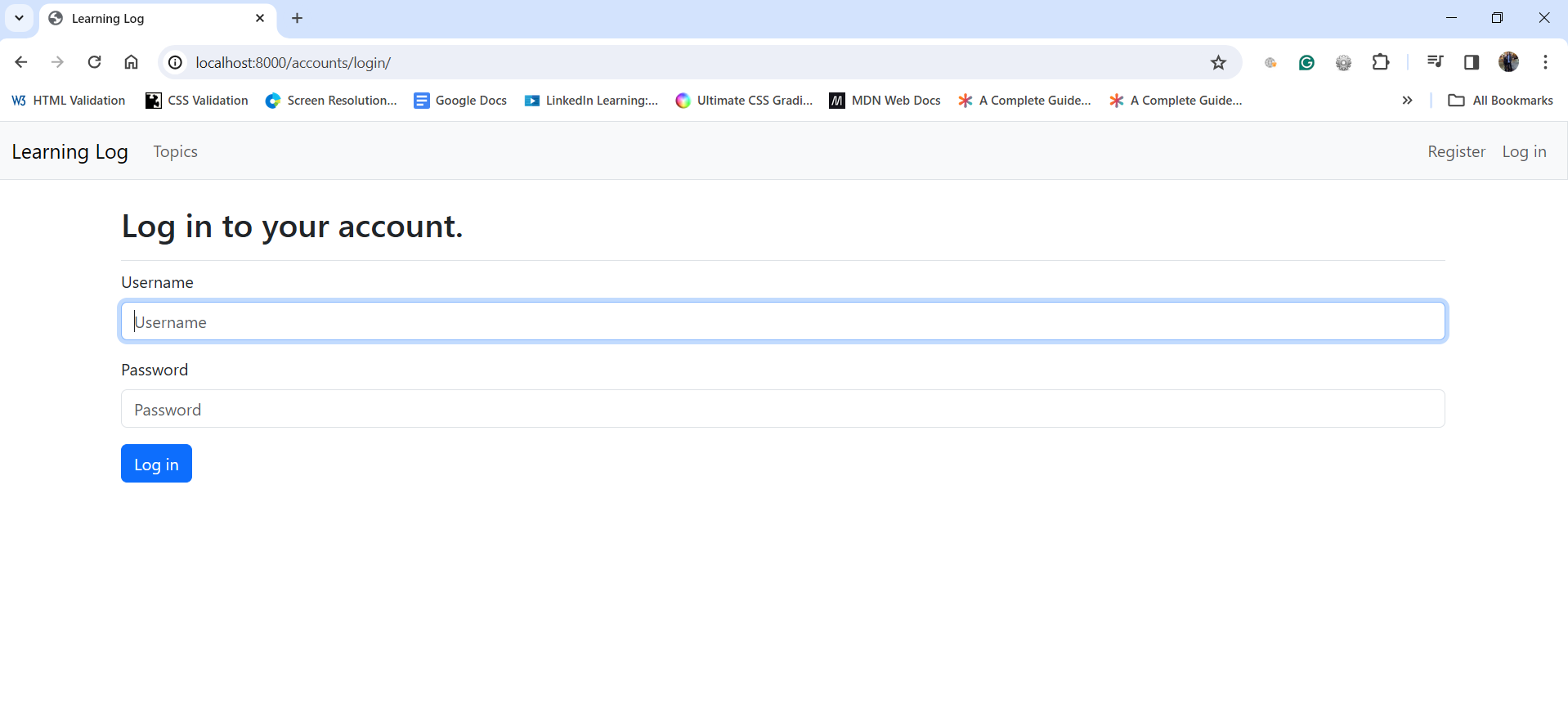


Now open a browser and type: <http://localhost:8000/>

You’ll be at the register/login page. You can create an account by typing a username and password or login if you already have an account.

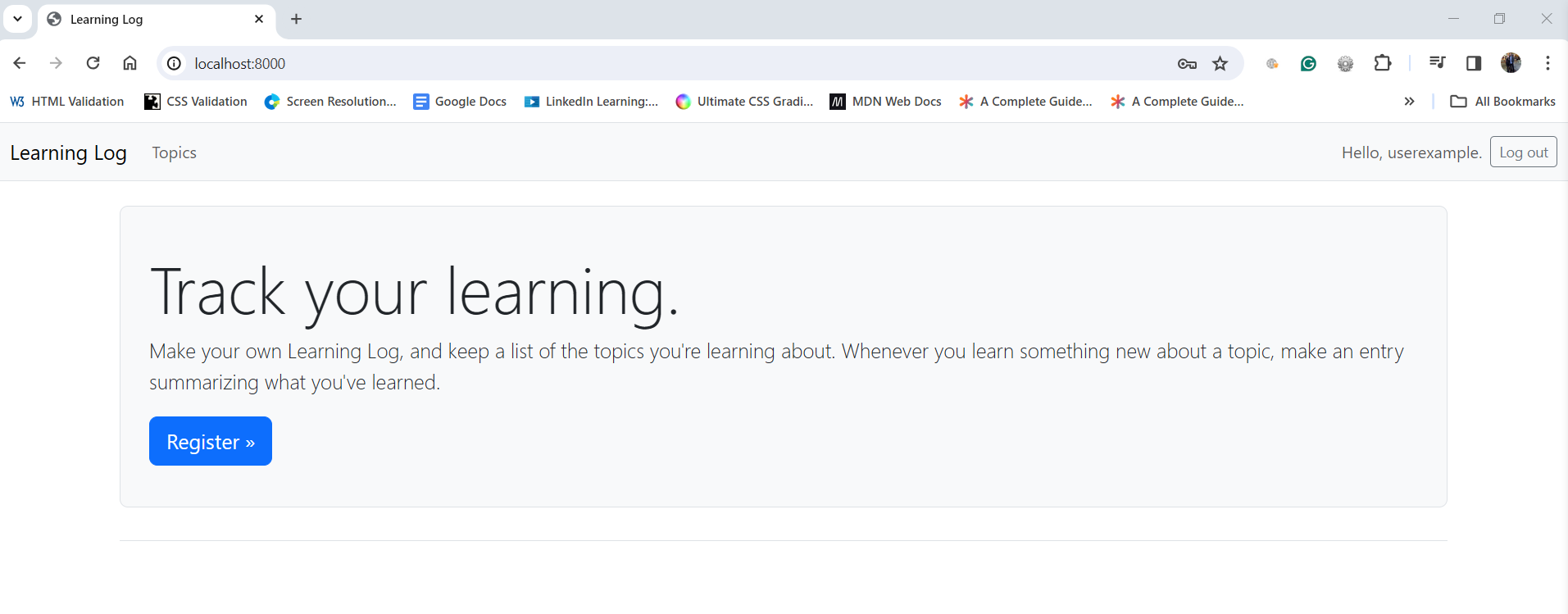




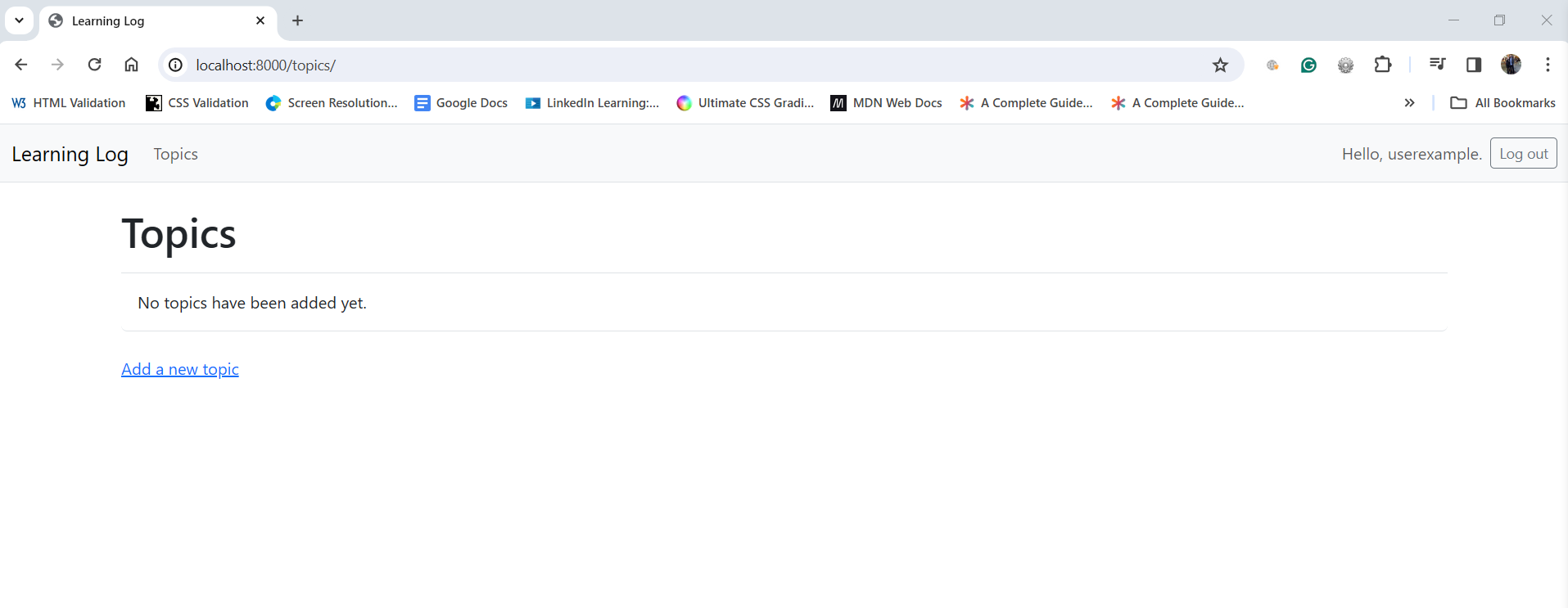


For this tutorial I created a username: **userexample** and password: **thefinalproject**

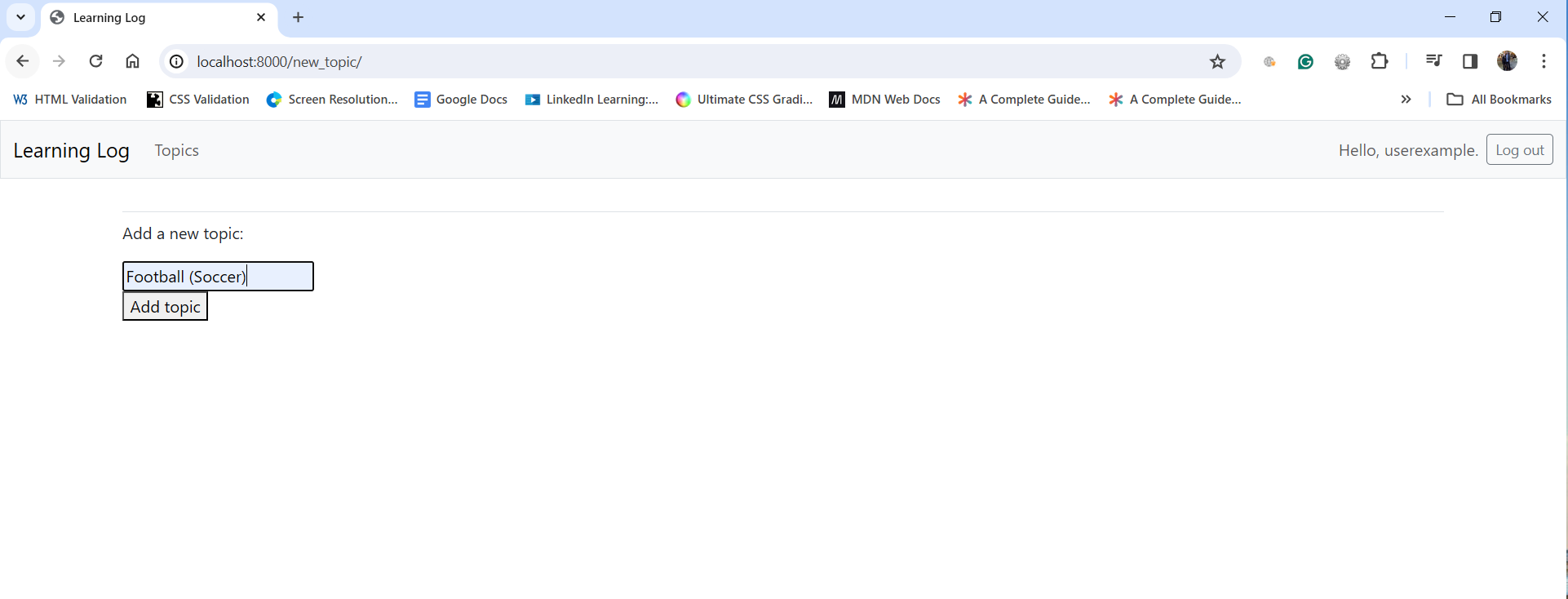
Once you’re logged in, click **Topics**.

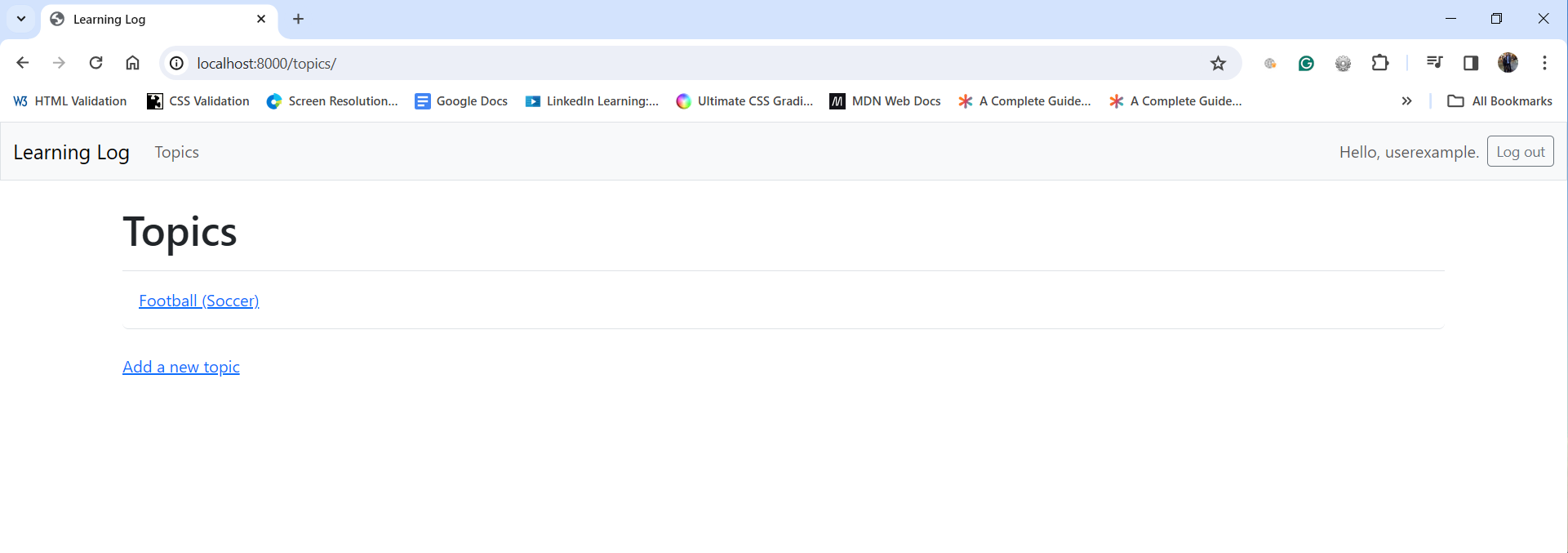


You’ll be redirected to the topic’s page where you can now add a topic.

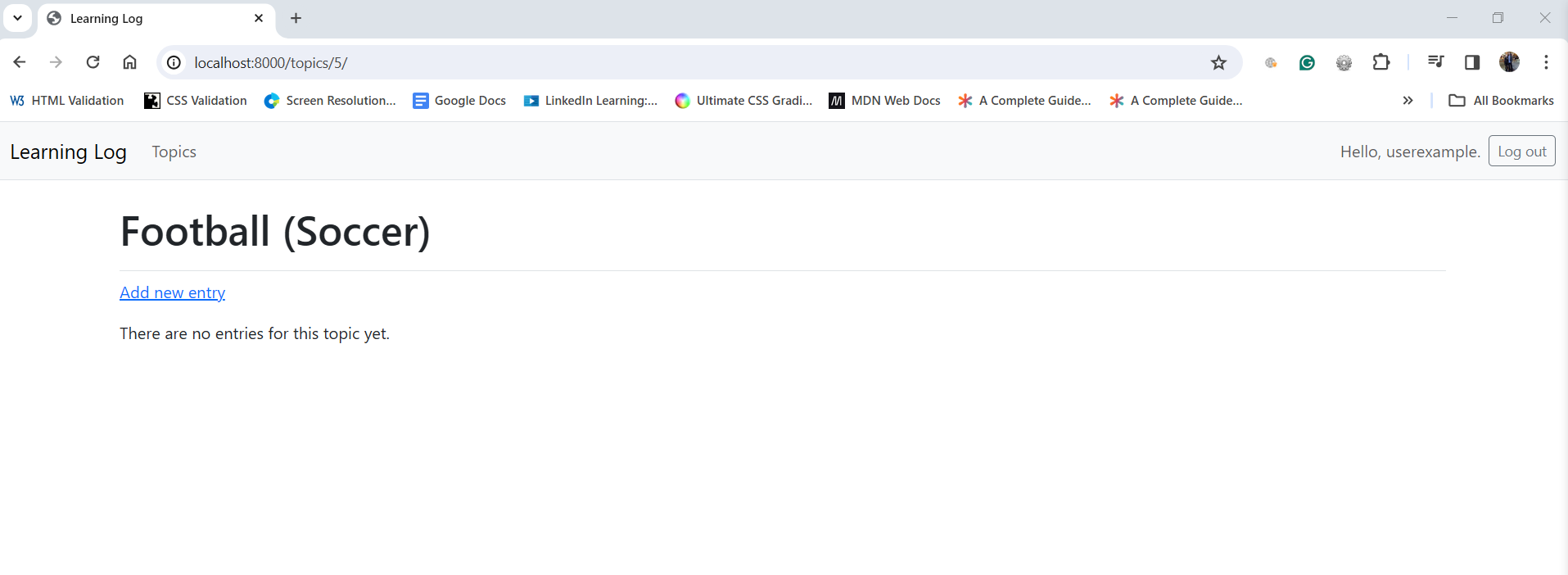


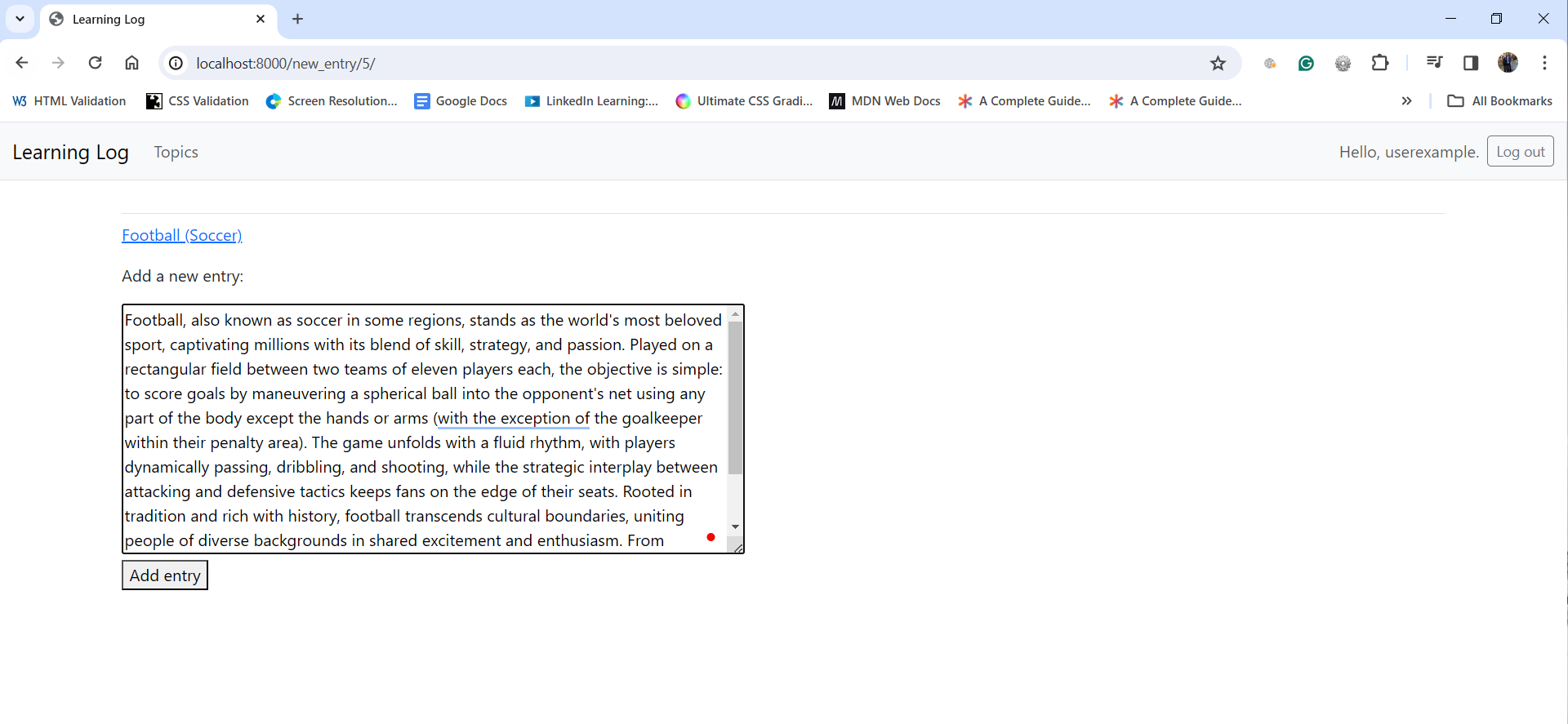
Choose a topic that you’d like to create and click the Add topic button.

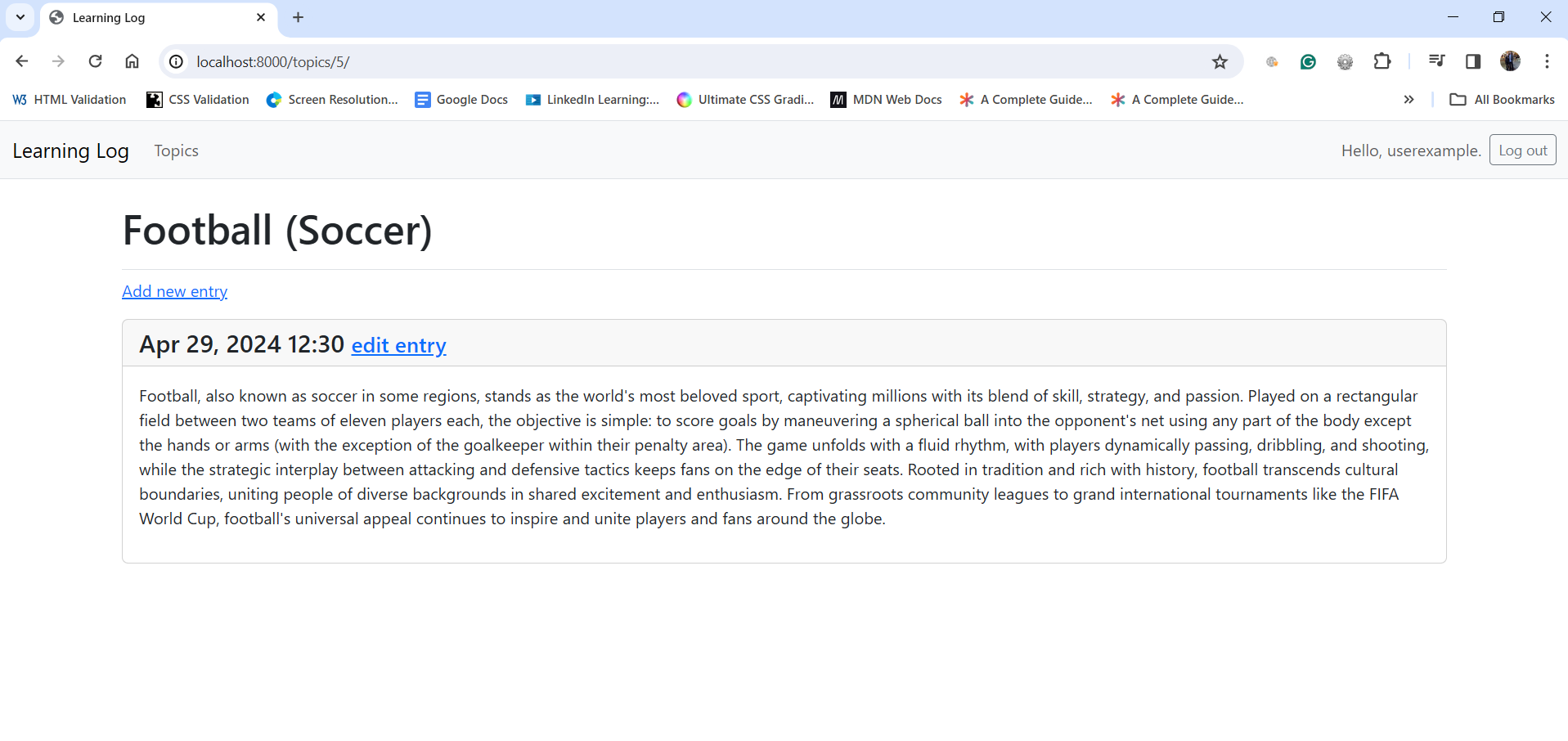




Now that you created a topic, you can click on it and add an entry, you can say what would like to talk about the topic. After that, click the Add entry button.







After your topic discussion is created, you have the option to edit, and to add a new entry about the topic created.

**Resources used to create this project:**

* Python Crash Course Third Edition
* GitHub
* Google and YouTube – random articles and videos to figure things out in case I was stuck on something.

## Code Copy

**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', 'll\_project.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()

## accounts folder

## apps.py

from django.apps import AppConfig

class AccountsConfig(AppConfig):

    default\_auto\_field = 'django.db.models.BigAutoField'

    name = 'accounts'

## urls.py

"""Defines URL patterns for accounts."""

from django.urls import path, include

from . import views

app\_name = 'accounts'

urlpatterns = [

 # Include default auth urls.

 path('', include('django.contrib.auth.urls')),

 # Registration page.

 path('register/', views.register, name='register'),

]

## views.py

from django.shortcuts import render

# Create your views here.

from django.shortcuts import render, redirect

from django.contrib.auth import login

from django.contrib.auth.forms import UserCreationForm

def register(request):

    """Register a new user."""

    if request.method != 'POST':

        # Display blank registration form.

        form = UserCreationForm()

    else:

        # Process completed form.

        form = UserCreationForm(data=request.POST)

        if form.is\_valid():

            new\_user = form.save()

            # Log the user in and then redirect to home page.

            login(request, new\_user)

            return redirect('learning\_logs:index')

    # Display a blank or invalid form.

    context = {'form': form}

    return render(request, 'registration/register.html', context)

## learning\_logs folder

## admin.py

from django.contrib import admin

# Register your models here.

from .models import Topic

from .models import Entry

admin.site.register(Topic)

admin.site.register(Entry)

## apps.py

from django.apps import AppConfig

class LearningLogsConfig(AppConfig):

    default\_auto\_field = 'django.db.models.BigAutoField'

    name = 'learning\_logs'

## forms.py

from django import forms

from .models import Topic, Entry

class TopicForm(forms.ModelForm):

    class Meta:

        model = Topic

        fields = ['text']

        labels = {'text': ''}

class EntryForm(forms.ModelForm):

    class Meta:

        model = Entry

        fields = ['text']

        labels = {'text': ''}

        widgets = {'text': forms.Textarea(attrs={'cols': 80})}

## models.py

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

class Topic(models.Model):

  """A topic the user is learning about."""

  text = models.CharField(max\_length=200)

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

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

  def \_\_str\_\_(self):

    """Return a string representation of the model."""

    return self.text

class Entry(models.Model):

    """Something specific learned about a topic."""

    topic = models.ForeignKey(Topic, on\_delete=models.CASCADE)

    text = models.TextField()

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

    class Meta:

        verbose\_name\_plural = 'entries'

    def \_\_str\_\_(self):

        """Return a simple string representing the entry."""

        return f"{self.text[:50]}..."

## urls.py

"""Defines URL patterns for learning\_logs."""

from django.urls import path

from . import views

app\_name = 'learning\_logs'

urlpatterns = [

     # Home page

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

     # Page that shows all topics.

 path('topics/', views.topics, name='topics'),

     # Detail page for a single topic.

 path('topics/<int:topic\_id>/', views.topic, name='topic'),

     # Page for adding a new topic.

 path('new\_topic/', views.new\_topic, name='new\_topic'),

     # Page for adding a new entry.

 path('new\_entry/<int:topic\_id>/', views.new\_entry, name='new\_entry'),

     # Page for editing an entry.

 path('edit\_entry/<int:entry\_id>/', views.edit\_entry, name='edit\_entry'),

]

## views.py

from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login\_required

from django.http import Http404

from .models import Topic, Entry

from .forms import TopicForm, EntryForm

def index(request):

    """The home page for Learning Log."""

    return render(request, 'learning\_logs/index.html')

@login\_required

def topics(request):

    """Show all topics."""

    topics = Topic.objects.filter(owner=request.user).order\_by('date\_added')

    context = {'topics': topics}

    return render(request, 'learning\_logs/topics.html', context)

@login\_required

def topic(request, topic\_id):

    """Show a single topic and all its entries."""

    topic = Topic.objects.get(id=topic\_id)

    # Make sure the topic belongs to the current user.

    if topic.owner != request.user:

        raise Http404

    entries = topic.entry\_set.order\_by('-date\_added')

    context = {'topic': topic, 'entries': entries}

    return render(request, 'learning\_logs/topic.html', context)

@login\_required

def new\_topic(request):

    """Add a new topic."""

    if request.method != 'POST':

        # No data submitted; create a blank form.

        form = TopicForm()

    else:

        # POST data submitted; process data.

        form = TopicForm(data=request.POST)

        if form.is\_valid():

            new\_topic = form.save(commit=False)

            new\_topic.owner = request.user

            new\_topic.save()

            return redirect('learning\_logs:topics')

    # Display a blank or invalid form.

    context = {'form': form}

    return render(request, 'learning\_logs/new\_topic.html', context)

@login\_required

def new\_entry(request, topic\_id):

    """Add a new entry for a particular topic."""

    topic = Topic.objects.get(id=topic\_id)

    if request.method != 'POST':

        # No data submitted; create a blank form.

        form = EntryForm()

    else:

        # POST data submitted; process data.

        form = EntryForm(data=request.POST)

        if form.is\_valid():

            new\_entry = form.save(commit=False)

            new\_entry.topic = topic

            new\_entry.save()

            return redirect('learning\_logs:topic', topic\_id=topic\_id)

    # Display a blank or invalid form.

    context = {'topic': topic, 'form': form}

    return render(request, 'learning\_logs/new\_entry.html', context)

@login\_required

def edit\_entry(request, entry\_id):

    """Edit an existing entry."""

    entry = Entry.objects.get(id=entry\_id)

    topic = entry.topic

    if topic.owner != request.user:

        raise Http404

    if request.method != 'POST':

        # Initial request; pre-fill form with the current entry.

        form = EntryForm(instance=entry)

    else:

        # POST data submitted; process data.

        form = EntryForm(instance=entry, data=request.POST)

        if form.is\_valid():

            form.save()

            return redirect('learning\_logs:topic', topic\_id=topic.id)

    context = {'entry': entry, 'topic': topic, 'form': form}

    return render(request, 'learning\_logs/edit\_entry.html', context)

## ll\_project folder

## asgi.py

"""

ASGI config for ll\_project 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/5.0/howto/deployment/asgi/

"""

import os

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

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

application = get\_asgi\_application()

## settings.py

"""

Django settings for ll\_project project.

Generated by 'django-admin startproject' using Django 5.0.4.

For more information on this file, see

https://docs.djangoproject.com/en/5.0/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/5.0/ref/settings/

"""

from pathlib import Path

# 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/5.0/howto/deployment/checklist/

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

SECRET\_KEY = 'django-insecure-6@#tco1grgw+$xhvi@2r4or\*!x\*uz2zciyg$o%fibyu@sy7hi#'

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

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

    # My apps.

    'learning\_logs',

    'accounts',

    # Third party apps.

    'django\_bootstrap5',

    # Default django apps.

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

]

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 = 'll\_project.urls'

TEMPLATES = [

    {

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

        'DIRS': [],

        '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 = 'll\_project.wsgi.application'

# Database

# https://docs.djangoproject.com/en/5.0/ref/settings/#databases

DATABASES = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE\_DIR / 'db.sqlite3',

    }

}

# Password validation

# https://docs.djangoproject.com/en/5.0/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/5.0/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/5.0/howto/static-files/

STATIC\_URL = 'static/'

# Default primary key field type

# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field

DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

# My settings.

LOGIN\_REDIRECT\_URL = 'learning\_logs:index'

LOGOUT\_REDIRECT\_URL = 'learning\_logs:index'

LOGIN\_URL = 'accounts:login'

# Platform.sh settings.

from platformshconfig import Config

config = Config()

if config.is\_valid\_platform():

    ALLOWED\_HOSTS.append('.platformsh.site')

    DEBUG = False

    if config.appDir:

        STATIC\_ROOT = Path(config.appDir) / 'static'

    if config.projectEntropy:

        SECRET\_KEY = config.projectEntropy

    if not config.in\_build():

        db\_settings = config.credentials('database')

        DATABASES = {

            'default': {

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

                'NAME': db\_settings['path'],

                'USER': db\_settings['username'],

                'PASSWORD': db\_settings['password'],

                'HOST': db\_settings['host'],

                'PORT': db\_settings['port'],

    },

}

## urls.py

"""

URL configuration for ll\_project project.

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

    https://docs.djangoproject.com/en/5.0/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.contrib import admin

from django.urls import path, include

"""from django.urls import path""" #possible mistake and have to delete this line

urlpatterns = [

    path('admin/', admin.site.urls),

    path('accounts/', include('accounts.urls')),

    path('', include('learning\_logs.urls')),

]

## wsgi.py

"""

WSGI config for ll\_project 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/5.0/howto/deployment/wsgi/

"""

import os

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

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

application = get\_wsgi\_application()