COMP 019 - Applications Development and Emerging Technologies

Full-Stack Python Development

SESSION 3: DJANGO ADVANCED FEATURES

ACTIVITY 3.1: DJANGO FORMS AND USER INPUT

Topic: Handling user input with Django forms

Description: Create forms to accept and validate user input

INSTRUCTIONS:

Create forms.py in your app:

• PostForm - Form for creating/editing posts

• CommentForm - Form for adding comments

• Use ModelForm for automatic form generation

Implement form views:

• post_create - Display and process new post form

• post_edit - Edit existing post

• comment_add - Add comment to post

• Handle GET (display) and POST (process) requests

Create form templates:

• post_form.html with {{ form.as_p }}

• Add CSRF token: {% csrf_token %}

• Style forms with CSS

Implement validation:

• Built-in field validation

• Custom validation in clean() method

• Display error messages in template

Deliverables: Submit forms.py, views, templates, and working form screenshots	**30 Points**

ACTIVITY 3.2: USER AUTHENTICATION

Topic: Implementing user registration, login, and logout

Description: Add user authentication to your Django application

INSTRUCTIONS:

Configure authentication:

• Use Django's built-in User model

• Configure LOGIN_URL, LOGIN_REDIRECT_URL in settings.py

Create authentication views:

• Registration view with UserCreationForm

• Login view using Django's LoginView

• Logout view using Django's LogoutView
• Profile view showing user info

Create authentication templates:
• registration/login.html
• registration/register.html
• registration/logout.html

Protect views:
• Use @login_required decorator
• Only allow post owners to edit/delete
• Show different content for logged-in users

Update models:
• Link Post author to User model (ForeignKey)

Deliverables: Submit authentication code and login/logout demo screenshots	**35 Points**

ACTIVITY 3.3: DJANGO ADMIN PANEL

Topic: Customizing Django admin for data management

Description: Configure and customize Django's powerful admin interface

INSTRUCTIONS:

Create superuser:
• python manage.py createsuperuser
• Access /admin in browser

Register models in admin.py:
• admin.site.register(Post)
• admin.site.register(Comment)

Customize admin display:
• Create PostAdmin class
• list_display - columns to show
• list_filter - sidebar filters
• search_fields - searchable fields
• ordering - default sort order
• date_hierarchy - date-based navigation

Advanced customization:
• Inline editing (CommentInline for Post)
• Custom actions (mark posts as featured)
• Fieldsets for organized form layout

Test CRUD operations in admin panel

Deliverables: Submit admin.py and customized admin panel screenshots	**25 Points**

TOTAL POINTS: 90