Module: **Web Application Development**

Assignment No. = **3**

Total Marks: **30**

In this assignment, you are required to extend the client-side HTML & Javascript interface for your Web application submitted as Assignment 1 & 2 and convert it into a Python Django server-side web application. It should run from a Docker container and connect to a specified SQL database.

To get started, you can use the provided "Django Example Project" template provided in class, which contains the standard Django Tutorial project, along with configuration for Visual Studio Code, Docker and DATABASE_URL environment variable handling. This is optional but recommended.

The functionality requested below should fit into your overall application design.

Q1: Django Models / SQL Database Tables (5 marks)

- 4 or more SQL database tables managed by Django ORM Model classes. These are tables you add in "models.py", not including any existing built-in Django tables.
- You must have at least one Model/Table with a ForeignKey relation to another, for example a table "Book" can have a ForeignKey relation to a table "Author".
- You source code should include the “migrations/*.py” output files automatically produced by Django “manage.py migrations", such that when we run your project as described in Q5, the SQL database will have the proper schema matching your models.

Q2: Forms for typical CRUD (Create, Read, Update, Delete) operations (7 marks)

- 4 or more POST request forms. These can be regular HTML POST forms or AJAX POST forms. You must have at least one or more forms for each of the following:
  - (A) Creating a new entry in the database
  - (B) Updating an existing entry in the database
  - (C) Deleting an existing entry in the database
- After using these forms, the changes should be visible somewhere on the website.
- All POST request forms should include server-side validation of every uploaded field.
- At least 2 POST request forms should also include client-side Javascript validation.

Q3: Client-side AJAX Functionality (3 marks)

- 1 or more JSON-based AJAX GET Request. This can initiated from any kind of event, e.g. a button press or on page load. If the call fails, it should show an error but the rest of the page should continue working. If the call succeeds, it should update part of the page.
- 1 or more JSON-based AJAX POST Request that perform some action to an entry in the database, i.e. Create, Update, or Delete. If the request fails, an error should be shown on the page. If the request succeeds, it should result in some part of the page being updated, either using the JSON results returned directly from the AJAX POST request, or by triggering a seperate AJAX GET Request.

Q4: User registration & login functionality (5 marks)

- User registration. There should be a sign-up or registration form somewhere on your site that allows unauthenticated users to create new User accounts.
- User login. There should be a simple login page somewhere on your site that takes a username and password and authenticates the current user.
- User logout. There should be a way for a user to log out of the site

To implement this section, you are recommended to take advantage of the Django authentication system built-in “User” objects. For additional user profile information, you can create an associated UserProfile model in your models.py which has a ForeignKey relation to the built-in User model you just created, and which also contains additional fields relevant to your application, for example Address, Telephone Number, any user preferences etc.

- “staff” users which can perform additional functionality on the site, e.g. the ability to add products to the catalogue, or make new posts.

For project demonstration purposes, these staff users can be created as normal users using your existing registration functionality, then manually converted into staff accounts using the built-in Django Admin UI by setting the user’s “is_staff” field to true. Then in your project HTML templates, you can use “{% if request.user.is_staff %}Something only staff should see{% endif %}” and in your View functions check for “request.user.is_staff == True” before allowing staff restricted form operations to continue.

Q5: Docker container, SQL Database connectivity & migrations (5 marks)

(This section is optional, i.e. you can submit a Django project without a Dockerfile and that just uses the built-in sqlite3 database without DATABASE_URL + MySQL. In that case, you must add 2 additional Database Models in Q1 and 2 additional HTML forms in Q2. If you include both extra models + forms, and a Dockerfile + working DATABASE_URL functionality, you will get the higher marks of the 2).

- It should be possible to build your source code into a Docker container and run the resulting image with the following commands:

```
docker build -t myapp
docker volume create myapp-storage
docker run -ti -e
DATABASE_URL="mysql://myappdbuser:myappdbpass@host.docker.internal:3306/
myappdb" -v myapp-storage:/app/storage -p 8000:8000 myapp
```

- The Web Server should be accessible at "http://localhost:8000"
- The project should be able to connect to the database specified in DATABASE_URL environment variable (i.e. using the “dj-database-url” plugin). In the commands above, it will connect to a running MySQL server running on your local computer, to the database "myappdb", with credentials "myappdbuser", "myappdbpass".
- The SQL migrations should automatically run on project startup against a clean fresh database.

The "Django Example Project" already implements all the required Dockerfile and dj-database-url configuration for this section, but you are required to test and make sure it is still working before submitting your project, as this is how we will test your project.

Q6: Overall Summary (5 marks)

You are required to prepare a demonstration of your individual assignments 1 to 3. You are required to create a video demonstration where you will explain the domain of your application and your approach to build your web application. Consider this assignment as a brief introductory video of your web application and its functionalities. You should ideally record a zoom-style video (similar to lecture recordings and demos) where you showcase your web application on screen and then talk over it showcasing alternatively the front-end webbrowser and your implemented code in Visual Studio code as required.

The overall video length should be between 10 to 15 minutes with 2 to 3 minutes per section for each of the five sections listed below. Please try to adhere to the guidelines for demo duration and avoid uploading too short or too lengthy videos. The maximum size for a file upload on Loop is 250MB but if your submission file size is too large, you can split it into multiple files and then upload multiple files on the loop.

The demo is required to discuss/present the following aspects of your web application:

1) Introduction to your Web Application
   a) What is domain of your web application and why there is a need to have web application for this domain?
   b) Business requirements in terms of web application contents, target audience and how user will interact with web application?

2) Client-Side Programming
   a) User interface design (CSS, colouring scheme, screen area etc.)
   b) Technology selection and any limitations
   c) How can the user interact (including JS interaction and event handling) with the application and what actions can be performed while visiting the application and how user will navigate (showcase all front-end web pages through navigation)?

3) Server-Side Programming
   a) Requirements for server-side code in your web application
   b) Technology selection and any limitations
   c) How client-side code interacts with server-side code and how responses are generated (showcase all server-side program functionality through navigation)

4) Database Structure & Connectivity
   a) What information is stored in your database and what is your database structure?
   b) How are you connecting to the database through server-side programming?
   c) Showcase how user input can be processed to select or update records in the database?

5) Building and Running in a Docker container
   a) How this application can be built and run as a Docker container
   b) Showcase your application running from Docker and connecting to your local MySQL.
   c) (Optional, not required for marks) Show your Docker container running on a hosting service such as Heroku or Google Cloud Run.

Preparation/Submission Instructions:

For Q1-Q5, you will be submitting.

- 1x Zip file of project folders copied from Visual Studio Code with all relevant code files from the project directory
    - If you are using git for source control, you can use the following command to export your folder, which will exclude any unnecessary binaries.
        - `git archive --format=zip --output project-export.zip HEAD`
- You are required to add comments in your code wherever possible, as you would explain to someone new to the project what a particular section does and how it works and why you did it that way. You don't have to comment every line, just the major sections of work that you have done.
- 1x ReadMe file containing installation and configuration instructions
- 1x Briefing document (word doc or pdf)

For Q6, you will be submitting:

- A single or multiple video file/files containing your screencast video/videos.

Students must ensure that their web application must not have any broken links when building and running as a Docker container. Students should also ensure that they have tested their application for a variety of browsers ensuring their compatibility.

Please note, your submission will be assessed qualitatively e.g. marks for completion of a required task and overall solution quality.

Please also note, in case of suspected plagiarism (whenever a template with all functionalities is downloaded from the Web with a possibility that the student has no understanding of the underlying code), you could be asked to be available for viva and present/explain your work.