



**Tribhuwan University**

**Institute of Engineering**

**Thapathali Campus**

**Subject: Web Application Programming**

**LAB #3**

**Lab Report on:** Server Side Programming and Django

**Submitted By**

Name: Srijal Basnet

Roll No: THA080BCT044

**Submitted To**

Department Of Electronics and Computer Engineering

5<sup>th</sup> February 2026

## **OBJECTIVES**

To understand the concept and workflow of server-side programming.

To learn the fundamental structure and architecture of the Django framework.

To build a basic CRUD web application using Django.

## **THEORY**

Server-side programming refers to backend development where application logic runs on a web server instead of the user's browser. When a user sends a request through HTTP methods such as GET or POST, the server processes the request, interacts with the database if required, and sends back a response like HTML or JSON. This allows web applications to generate dynamic content based on user input or stored data.

Server-side systems also provide improved security because sensitive logic and database operations remain hidden from the client side. Backend frameworks handle data processing, authentication, and CRUD operations, ensuring structured data management and persistence.

Django is a high-level Python web framework designed to simplify web development by providing built-in tools for routing, database management, authentication, and templating. It follows a structured architecture that separates data models, business logic, and presentation layers, allowing developers to build scalable applications quickly.

In this lab project, I created a simple CRUD Django application called 'My Game List'. The application allows users to add, edit, view, and delete games. Each game entry includes fields such as rating, review, and play status. The application integrates with the RAWG API to retrieve game database information, demonstrating how external APIs can be used alongside Django to enrich application data while maintaining a local database.

## Project Folder Structure

```
└─ CRUD DJANGO
    └─ GameLibrary
        └─ GameLibrary
            > __pycache__
            ⚡ __init__.py
            ⚡ asgi.py
            ⚡ settings.py
            ⚡ urls.py
            ⚡ wsgi.py
        └─ MyGameList
            > __pycache__
            > migrations
            └─ static
                # style.css
            └─ templates
                └─ registration
                    <> login.html
                    <> register.html
                    <> add_game.html
                    <> base.html
                    <> edit_game.html
                    <> game_list.html
                ⚡ __init__.py
                ⚡ admin.py
                ⚡ apps.py
                ⚡ models.py
                ⚡ tests.py
                ⚡ urls.py
                ⚡ views.py
            └─ db.sqlite3
            ⚡ manage.py
```

## SOURCE CODE

<https://github.com/Srijal-Basnet/My-Game-List-App.git>

## OUTPUT

### 1. Login page

The screenshot shows a web browser window with a dark header bar labeled "GameList". On the right side of the header are the user handle "@Sreezl" and a red "Logout" button. The main content area has a white background and features a "Login" heading. Below it are two input fields: one for "Username" containing "Sreezl" and another for "Password" containing "\*\*\*\*\*". A blue "Login" button is positioned below the password field. At the bottom left, there is a link "Don't have an account? [Register](#)".

### 2. Register page

The screenshot shows a web browser window with a dark header bar labeled "GameList". On the right side of the header are the user handle "@Sreezl" and a blue "Logout" button. The main content area has a white background and features a "Register" heading. Below it are three input fields: one for "Username" containing "Sreezl", one for "Password" containing "\*\*\*\*\*", and one for "Confirm Password" also containing "\*\*\*\*\*". A blue "Register" button is positioned below the confirm password field. At the bottom left, there is a link "Already have an account? [Login](#)".

### 3. Home Page/ Game list

GameList

@Sreezl Logout

#### My Game List

[+ Add Game](#)

					
<b>Grand Theft Auto V</b> Action ★ 8/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	<b>God of War (2018)</b> Action ★ 9/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	<b>God of War: Ragnarök</b> Action ★ 10/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	<b>Clair Obscur: Expedition 33</b> RPG ★ 10/10 Playing <a href="#">Edit</a>   <a href="#">Delete</a>	<b>Marvel's Spider-Man Remastered</b> Casual ★ 10/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	<b>Marvel Rivals</b> Multiplayer ★ 8/10 Playing <a href="#">Edit</a>   <a href="#">Delete</a>
					
Action ★ 8/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	Action ★ 9/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	Action ★ 10/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	Clair Obscur: Expedition 33 RPG ★ 10/10 Playing <a href="#">Edit</a>   <a href="#">Delete</a>	Casual ★ 10/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	Multiplayer ★ 8/10 Playing <a href="#">Edit</a>   <a href="#">Delete</a>
					
<b>PEAK.</b> Multiplayer ★ 7/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	<b>The Last of Us Part I</b> Story ★ 9/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>	<b>Ghost of Tsushima Director's Cut</b> Adventure ★ 8/10 On Hold <a href="#">Edit</a>   <a href="#">Delete</a>	<b>Red Dead Redemption 2</b> Action ★ 0/10 Planned <a href="#">Edit</a>   <a href="#">Delete</a>	<b>Minecraft</b> Multiplayer ★ 10/10 Playing <a href="#">Edit</a>   <a href="#">Delete</a>	<b>DOOM (2016)</b> Shooter ★ 8/10 Completed <a href="#">Edit</a>   <a href="#">Delete</a>

### 4. Edit Game Page

GameList

@Sreezl Logout

#### Edit Game

Marvel's Spider-Man Remastered

Casual

Playing

Rating (0–10)  
9

Status  
Playing

[Update](#)

## 5. Add Game Page

The screenshot shows a web application interface titled "Add Game". At the top, there is a search bar containing the text "GTA V" and a "Search" button. Below the search bar is a list of five game entries, each with a title, genre, and rating: "Grand Theft Auto V (Action) ★ 4", "Gears 5 (Shooter) ★ 3", "Tekken 5 (Fighting) ★ 4", "DIRT 5 (Racing) ★ 3", and "Persona 5 (Adventure) ★ 4". Below this list are several input fields: "Title" (empty), "Genre" (empty), "Review" (empty), "Rating (0-10)" (empty), and "Status" (a dropdown menu set to "Playing"). At the bottom right is a prominent blue "Add Game" button.

## DISCUSSION AND CONCLUSION

The development of this app demonstrated how server-side programming enables dynamic and interactive web applications. Using Django, the application successfully implemented CRUD functionality for managing game entries while maintaining a clear separation between backend logic and frontend templates. The integration of the RAWG API showed how external services can be used to fetch structured data and enhance user experience without manually maintaining a large dataset.

Through this lab, key concepts such as request-response cycles, database interaction using Django ORM, and API integration were explored. Django's built-in features simplified authentication, routing, and data handling, making development faster and more organized. Overall, the project demonstrates that Django provides a reliable and efficient framework for building secure server-side applications with clean architecture and scalable design.