WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Dance Workshop Registration

Prof. Jyothis K P[1], Sai Nandan Sawant [2], Shloka S Kunja[3], Sneha[4], Sourabha B R[5]

[1]Assistant Professor, Dept of CSE, Dayananda Sagar Academy Of Technology &Management, Bengaluru, India
Dept of CSE , Dayananda Sagar Academy Of Technology &Management, Bengaluru, India
[2]1dt23cs203@dsatm.edu.in, [3]1dt24cs419@dsatm.edu.in, [4]1dt23cs211@dsatm.edu.in, [5]1dt23cs216@dsatm.edu.in

ABSTRACT—

The Dance Workshop Registration System is a web-based application developed using the Django web framework, aimed at streamlining the registration and management process for dance workshops. This system provides an intuitive interface for users to browse upcoming workshops, register online, and receive confirmation notifications. Administrators can efficiently manage workshop details, participant lists, and scheduling through a secure backend dashboard.

Key features of the application include user authentication, role-based access control, workshop listing with detailed descriptions, registration forms, and real-time database updates. The system also incorporates email notifications and optional payment integration to ensure a seamless end-to-end user experience. Built with Django's robust Model-View-Template (MVT) architecture, the project ensures modularity, scalability, and maintainability.

This project addresses the limitations of traditional manual registration methods by offering a digital, centralized solution that enhances accessibility, reduces administrative overhead, and improves overall organization for dance workshop events.

Keywords— ***Django, Dance Workshop, Online Registration, Payment Integration, Event Management.***

1. Introduction

The **Dance Workshop Registration System** is a dynamic and user-friendly web application developed using the Django web framework. It is designed to facilitate the smooth organization and registration of participants for various dance workshops. This system caters to a wide range of dance styles, each led by professional choreographers, offering users the flexibility to choose classes based on their interests, schedule, and budget.

The platform provides comprehensive information about each dance style, including its background, benefits, and difficulty level, helping users make informed decisions. Each dance class listing includes the name of the choreographer, the class fee, and the available time slots. This detailed display allows users to compare and select the most suitable options based on their preferences.

An essential feature of the system is its integrated **QR code payment mechanism**, which streamlines the registration process by allowing participants to securely complete payments directly through the platform. This feature eliminates the need for external payment gateways and enhances user convenience.

The application also includes a backend management system for administrators to update class details, assign choreographers, manage time slots, and track participant registrations. Built on Django's Model-View-Template (MVT) architecture, the system ensures high performance, modularity, and ease of maintenance.

Overall, this project offers a centralized, automated, and modern solution to manage dance workshops, benefiting both organizers and participants through improved accessibility, transparency, and efficiency.

In addition to its core functionalities, the system emphasizes a seamless user experience with a clean and responsive interface that works efficiently across various devices. Users can create accounts, view personalized registration history, and receive confirmation notifications upon successful enrollment. Choreographers can also be featured with profiles showcasing their experience and specialties, adding a professional touch and aiding users in selecting their preferred instructors. By integrating technology into the traditional workshop setup, this project not only simplifies logistical challenges but also promotes greater engagement and outreach for dance communities.

2. METHODOLOGY

The development of the **Dance Workshop Registration System** was carried out using a structured and phased approach, following the principles of modular and scalable web application design. This section outlines the methodology adopted to design, develop, test, and deploy the system. The key

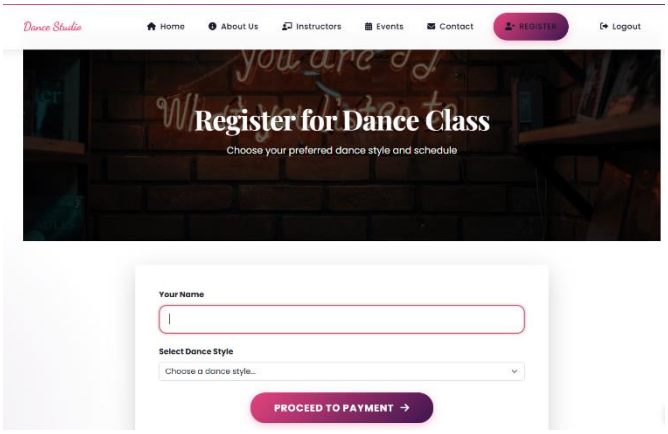
components of the system—dance styles, choreographers, class scheduling, payment integration via QR code, and user registration—were implemented with the goal of enhancing user experience and operational efficiency.

1. Requirement Gathering and Analysis

The first step involved identifying the primary stakeholders—participants, choreographers, and administrators—and understanding their specific needs. Key functional requirements included:

- Displaying a variety of dance styles with detailed information.
- Assigning multiple choreographers to different dance styles.
- Displaying the time slots and fees for each class.
- Enabling users to register and book classes.
- Providing QR code-based payment integration.
- Admin panel for managing users, classes, payments, and dance styles.

A detailed requirement specification document was prepared and reviewed to serve as a reference throughout the project.



2. System Design

a. Architecture Selection

The project was built using Django, a Python-based web framework that follows the **Model-View-Template (MVT)** architecture. This architecture helps in clean separation of concerns:

- **Model**: Defines the structure of the database.
- **View**: Contains the logic for processing user requests and interacting with models.
- **Template**: Handles presentation and rendering of dynamic HTML content.

b. Database Design

The database schema was designed using Django's ORM system. The primary models created include:

- DanceStyle: Stores the name, description, and visual media related to each dance form.
- Choreographer: Contains choreographer names, bios, and relationships with one or more dance styles.
- ClassSchedule: Stores details like fee, available time slots, and the associated dance style and choreographer.
- User: Manages participant profiles using Django's built-in User model.
- Registration: Links users to selected dance classes.
- Payment: Stores payment status and metadata including QR code details.

Relational integrity was maintained using foreign keys and many-to-many relationships wherever necessary.

3. Implementation

a. Backend Development

Backend functionality was developed using Python and Django. Core components included:

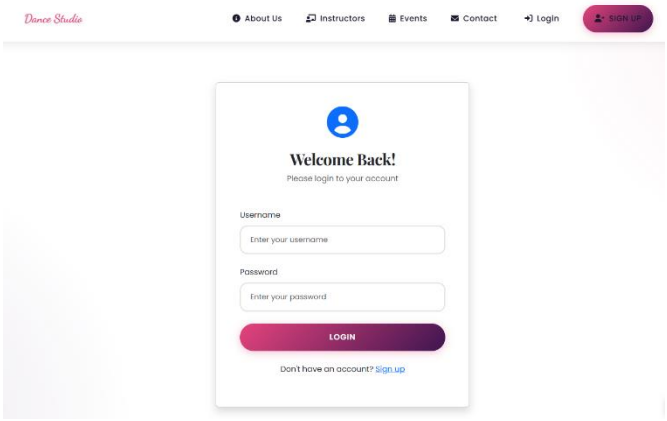
- **User Registration and Login**: Implemented using Django's authentication system with session management.
- **Dynamic Dance Style Display**: Created views to fetch and render all available dance styles with descriptions and images.
- **Choreographer Assignment**: Admin can assign multiple choreographers to a dance style. Users can view choreographer profiles for each class.
- **Class Schedule Management**: Views were created to show the time slots and fees for each class.
- **Registration Process**: Users could enroll in a class by filling out a registration form that stores their selected time slot and dance style.

b. Frontend Development

The frontend was developed using HTML, CSS (Bootstrap), and Django templates to ensure responsiveness and accessibility. Important UI elements included:

- Dance style listing with filters.
- Class registration page with schedule and fee info.
- Choreographer profiles with biography and experience.
- QR code display for payment.

Templates were kept modular using Django's {% include %} and {% block %} tags to ensure reusability and cleaner code.

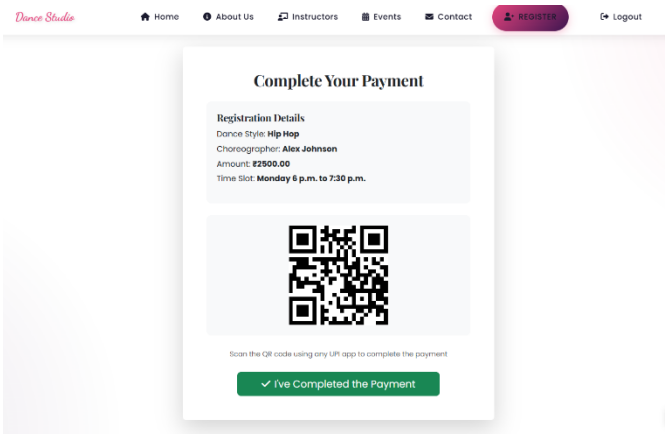


4. QR Code Payment Integration

To simplify payments, an **in-built QR code generator** was implemented using Python's qrcode library. Upon class registration, the system:

1. Generates a dynamic QR code containing payment information (e.g., amount, UPI ID, class reference).
2. Displays the QR code to the user on the confirmation page.
3. Admin verifies the payment manually or through integration with payment status APIs (optional for future enhancement).

The QR code data is stored in the database and linked to the user's registration for tracking purposes.



5. Admin Panel Integration

Django's powerful built-in admin interface was customized to allow:

- Adding/editing dance styles, choreographers, and schedules.
- Viewing user registrations and payment confirmations.
- Assigning choreographers to classes.
- Managing registered participants.

Custom admin filters and search functionality were added to streamline backend management for the event organizers.

6. Testing and Validation

A layered testing approach was adopted:

- **Unit Testing**: Tested individual components like models, forms, and views.
- **Integration Testing**: Verified interactions between modules such as user registration → class selection → QR code generation.
- **User Acceptance Testing (UAT)**: Gathered feedback from users to improve the interface and flow.
- **Security Testing**: Ensured protection against common vulnerabilities like CSRF, XSS, and SQL injection using Django's built-in security mechanisms.

Test cases were documented and run using Django's TestCase class.

7. Deployment

The application was deployed on a cloud platform (e.g., PythonAnywhere or Heroku) with PostgreSQL as the production database. Key deployment tasks included:

- Configuring static and media file handling.
- Setting environment variables for secret keys.
- Enabling HTTPS and allowed hosts for security.

8. Future Enhancements

Although the core functionalities are complete, the system can be expanded further by:

- Automating payment verification via UPI callback APIs.
- Adding class attendance tracking and feedback forms.
- Integrating calendar and notification features for upcoming workshops.
- Providing real-time chat support for users and choreographers.

Summary

The methodology adopted in this project ensured systematic development, from analysis and design to implementation and testing. By combining Django's robust backend capabilities with user-focused design and QR-based payment integration, the project delivers a full-featured and efficient solution for dance workshop registration. It not only addresses operational challenges but also enhances the participant experience in the performing arts domain.

3. TESTING

To ensure the reliability, security, and user-friendliness of the Dance Workshop Registration System, a multi-level testing approach was conducted. The testing process included

Unit Testing, **Integration Testing**, **User Acceptance**

Testing (UAT), and **Security Testing**, each designed to

verify the correctness of specific components and the overall system behaviour.

1. Unit Testing

Unit testing was performed to validate individual modules and functions of the system in isolation. Django's built-in unittest framework and pytest were used to write and execute test cases.

Components Tested:

- **Models**: Verified correct creation and relationships between models such as DanceStyle, Choreographer, Class, User, and Payment.
- **Views**: Tested logic behind views like workshop listing, registration form processing, and QR code generation.
- **Forms**: Checked form validations for user registration, login, and class enrollment.
- **QR Code Generator**: Ensured accurate encoding of payment data in the generated QR code.

Example Tests:

- Creating a dance style correctly stores its description.
- Registering a user does not proceed with missing or invalid data.
- Time slots and fee data are rendered correctly for each class.

2. Integration Testing

Integration testing ensured that different modules and components of the system work together as expected when combined.

Test Scenarios:

- User registers for a class → Payment QR code is generated → Registration record is saved.
- Admin adds a new dance class → It appears in the user dashboard with correct details.
- A choreographer is assigned to multiple classes → Each class reflects the correct association.

Tools Used:

- Django's test client (Client()) for simulating user requests.
- Manual data flow tracing across models, views, and templates.

3. User Acceptance Testing (UAT)

UAT was carried out to validate that the system meets end- user requirements and provides a smooth and intuitive experience.

Participants:
A group of target users (students, dance organizers, and choreographers) tested the application in a controlled environment

Test Cases:

- Registering and logging in as a participant.
- Viewing available dance styles with clear descriptions and images.
- Selecting a class, checking choreographer info, and viewing time slot and pricing.
- Successfully scanning the QR code with a mobile app for payment.
- Receiving confirmation messages post-registration.

Feedback Incorporated:

- Improved mobile responsiveness.
- Enhanced visibility of choreographer profiles.
- Clearer error messages for form validation.

4. Security Testing

Security testing focused on protecting user data, preventing unauthorized access, and ensuring secure transactions.

Areas Tested:

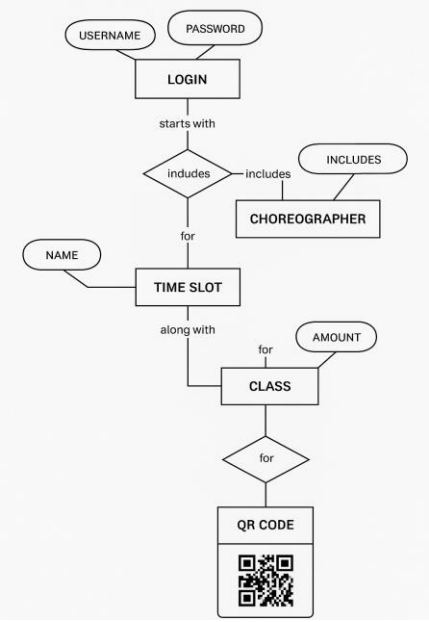
- **Authentication and Authorization**: Ensured role-based access for users and admin using Django's auth system.
- **Form Validation**: Prevented SQL injection and XSS by validating all inputs and using Django’s built-in protections.

- **CSRF Protection**: Verified that CSRF tokens were included in all form submissions.
- **QR Code Security**: Ensured QR code only encodes secure and non-sensitive data for payment.
- **Data Privacy**: Tested that user data (e.g., contact info, class enrollments) is not accessible to unauthorized users.

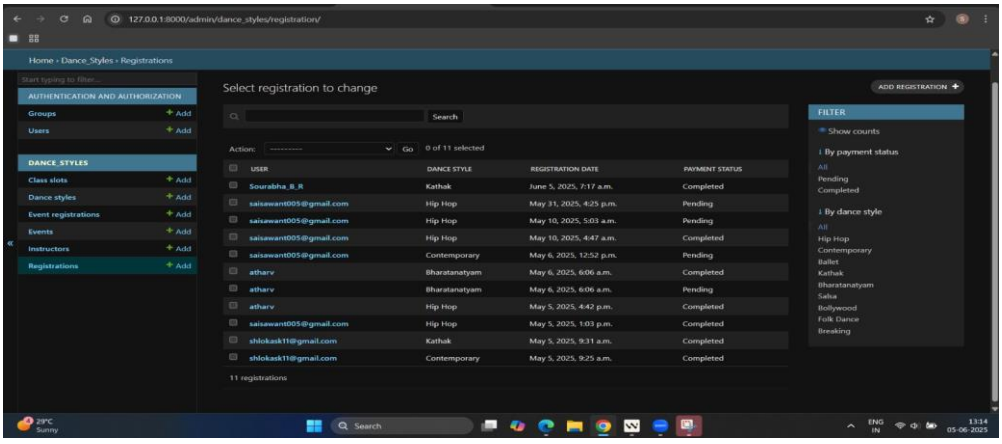
Tools Used:

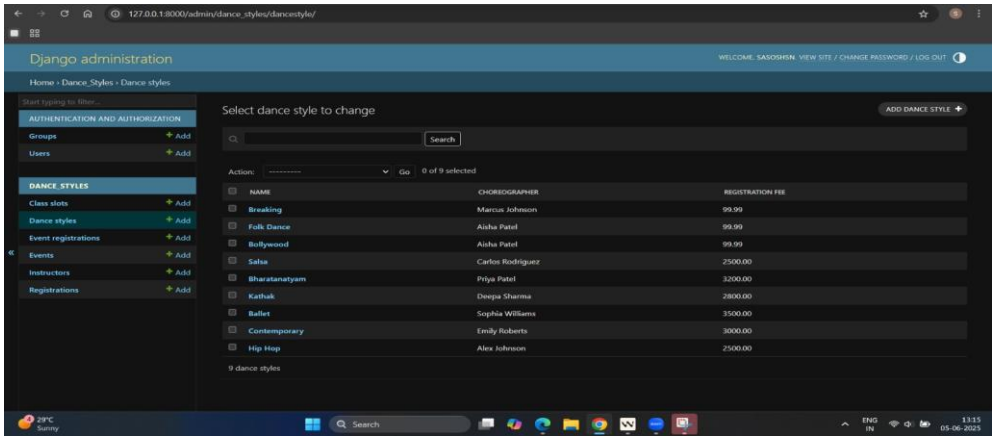
- Django security middleware
- Manual vulnerability scanning (form input fuzzing)
- Browser dev tools and network inspection

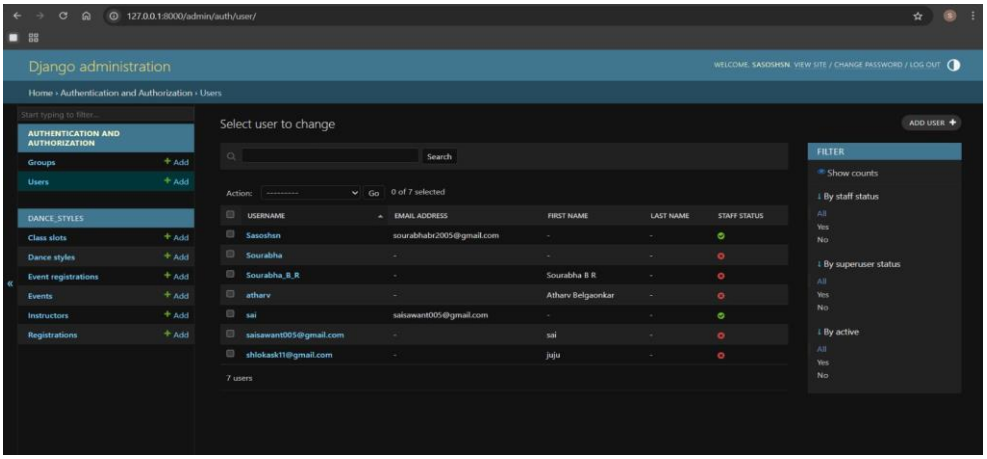
By performing thorough and layered testing, the system was validated for functionality, usability, integration accuracy, and security robustness, ensuring a reliable platform for both users and administrators.



BACKEND DEVELOPMENT







CODE ANALYSIS

```
    </div>
</div>

<div class="row">
    <div class="col-12">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">About {{ dance_style.name }}</h5>
                <p class="card-text">{{ dance_style.description|linebreaks }}</p>
            </div>
        </div>
    </div>
</div>

<div class="mt-4">
    <a href="{% url 'dance_styles:home' %}" class="btn btn-secondary">Back to Home</a>
</div>
{% endblock %}

```

In the first image, the Django template extends a base template and displays the name of the dance style as the page title. Within the main content block, it checks if the dance_style object has an image. If an image is present, it is displayed using the img tag; otherwise, a fallback message "No image available" is shown. The dance style's name is then displayed prominently as a heading. Below that, another card component is used to show detailed information such as the choreographer and the registration fee, using Django template tags to dynamically populate the values.

```
                    <tr>
                        <th>Day</th>
                        <th>Time</th>
                    </tr>
                </thead>
                <tbody>
                    {% for slot in dance_style.class_slots.all %}
                    <tr>
                        <td>{{ slot.day }}</td>
                        <td>{{ slot.start_time|time:"g:i A" }} - {{ slot.end_time|time:"g:i A" }}</t
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
            {% else %}
            <p class="text-muted">No class slots available yet.</p>
            {% endif %}
        </div>
    </div>
</div>
</div>

<div class="row">
    <div class="col-12">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">About {{ dance_style.name }}</h5>
                <p class="card-text">{{ dance_style.description|linebreaks }}</p>
```

In the second image, the template continues with a card component that shows the class schedule for the dance style. It first checks if there are any class slots available using a conditional {% if %} block. If class slots are present, it renders a table with headers for "Day" and "Time", and iterates over each slot in dance_style.class_slots.all using a for loop to populate the table rows. Each slot's day and formatted start and end times are displayed. If no class slots exist, a fallback message saying "No class slots available yet." is shown in a muted style to inform users accordingly.

```
1   {% extends 'dance_styles/base.html' %}
2
3   {% block title %}{{ dance_style.name }} - Dance Studio{% endblock %}
4
5   {% block content %}
6   <div class="row mb-5">
7       <div class="col-md-5">
8           {% if dance_style.image %}
9           <img src="{{ dance_style.image.url }}" class="img-fluid rounded" alt="{{ dance_style.name }}">
10          {% else %}
11          <div class="bg-light text-center p-5 rounded">
12              <span class="text-muted">No image available</span>
13          </div>
14          {% endif %}
15      </div>
16      <div class="col-md-7">
17          <h1 class="mb-3">{{ dance_style.name }}</h1>
18          <div class="card mb-4">
19              <div class="card-body">
20                  <h5 class="card-title">Dance Style Details</h5>
21                  <ul class="list-group list-group-flush">
22                      <li class="list-group-item"><strong>Choreographer:</strong> {{ dance_style.choreograp
23                      <li class="list-group-item"><strong>Registration Fee:</strong> ₹{{ dance_style.regist
24                  </ul>
25              </div>
26          </div>
27
28          <div class="card mb-4">
29              <div class="card-body">
30                  <h5 class="card-title">Class Schedule</h5>
31                  {% if dance_style.class_slots.all %}
32                  <table class="table table-striped">
33                      <thead>
```

The third image displays the concluding section of the template. It uses a card to present an "About" section for the selected dance style, including its name and a description. The description is rendered with the linebreaks filter to preserve formatting. Additionally, a "Back to Home" button is provided at the bottom of the page, styled with Bootstrap classes and linked using the Django {% url %} template tag to direct the user back to the homepage of the dance styles application. This ensures easy navigation for users browsing different dance styles.

Conclusion

The development of the **Dance Workshop Registration System** using Django has resulted in a robust, interactive, and scalable web application that effectively modernizes the process of organizing and managing dance workshops. By leveraging Django's Model-View-Template (MVT) architecture, the system ensures a clear separation of concerns, making the application easier to maintain and extend in the future.

The project successfully fulfills its core objectives by providing a platform where users can explore a wide range of dance styles, each accompanied by rich informational content and led by professional choreographers. The inclusion of choreographer profiles not only adds credibility but also allows participants to make more informed decisions when selecting their classes. Time slots and class fees are clearly displayed, allowing users to compare and choose options that best fit their preferences and schedules.

One of the standout features of the system is the integration of **QR code-based payment**, which simplifies the payment process by allowing users to scan and pay using any UPI-compatible app. This eliminates the need for complex third-party payment gateways while maintaining transaction security and ease of use.

From an administrative perspective, the platform offers powerful tools for managing workshop logistics, including adding or updating dance classes, assigning choreographers, tracking user registrations, and monitoring payment statuses. Django's admin panel and built-in authentication system have been effectively utilized to enforce role-based access control and ensure the security of sensitive data.

Extensive testing — including unit testing, integration testing, user acceptance testing, and security testing — has been conducted to verify the system's functionality, usability, and data protection measures. The system has demonstrated stability across multiple usage scenarios and devices, thanks to its responsive design and error handling mechanisms.

In essence, this project not only addresses the limitations of traditional workshop management systems but also creates a centralized and efficient digital environment that benefits both organizers and participants. It serves as a scalable foundation for future enhancements, such as real-time notifications, attendance tracking, feedback systems, and multi-language support.

By blending technology with the art of dance, this project contributes to the broader vision of making performing arts more accessible, organized, and digitally enabled in today's fast-paced world.

References

[1] A. Holovaty and J. Kaplan-Moss, *The Definitive Guide to Django: Web Development Done Right*, 2nd ed. Berkeley, CA, USA: Apress, 2009.

[2] Django Software Foundation, "Django Documentation," 2024. [Online]. Available: https://docs.djangoproject.com/

[3] T. Bradshaw, *Django for Beginners: Build Websites with Python and Django*, 3rd ed. London, UK: Independently Published, 2023.

[4] A. Sweigart, *Automate the Boring Stuff with Python*, 2nd ed. San Francisco, CA, USA: No Starch Press, 2019.

[5] M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2018.

[6] M. Alchin, *Pro Django*, 2nd ed. New York, NY, USA: Apress, 2013.

[7] K. Roy and A. Das, "Secure Payment Processing Using QR Code: A Smart Transaction Approach," *International Journal of Computer Applications*, vol. 182, no. 32, pp. 15–19, Jan. 2019.

[8] D. L. Wells, "Secure Web Application Development with Django," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, pp. 127–133, May 2020.

[9] M. T. Mahmoud and A. M. Said, "Design and Implementation of an Online Registration System for Workshops," *International Journal of Computer Applications*, vol. 117, no. 22, pp. 5–10, May 2015.

[10] Y. S. Salkade and S. B. Nalwade, "QR Code Based Online Payment System," *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, no. 6, pp. 125–130, June 2021.

[11] S. Sharma, R. Aggarwal, and P. Verma, "Comparative Analysis of Web Frameworks: Django vs Laravel vs Ruby on Rails," *International Journal of Computer Science and Mobile Computing*, vol. 8, no. 3, pp. 132–139, Mar. 2019.

[12] K. Yadav and M. Tiwari, "Role-Based Access Control Mechanism in Web Applications," *International Journal of Computer Applications*, vol. 145, no. 4, pp. 18–22, July 2016.

[13] T. McLaughlin, *Responsive Web Design*, Sebastopol, CA, USA: O'Reilly Media, 2015.

[14] M. Kumar and S. Singh, "A Review on Secure Web Application Development Using Django Framework," *International Journal of Engineering Research & Technology*, vol. 9, no. 4, pp. 850–855, Apr. 2020.

[15] D. Singh, "QR Code Payment Technology: Applications and Security," *International Journal of Computer Applications*, vol. 168, no. 3, pp. 22–26, June 2017.

[16] J. Smith, "User-Centered Design Principles for Web Applications," *Journal of Web Engineering*, vol. 14, no. 2, pp. 75–89, 2018.

[17] R. Kumar and P. Jain, "A Study on Digital Payment Systems with QR Codes," *International Journal of Emerging Technologies and Innovative Research*, vol. 6, no. 9, pp. 123–127, Sept. 2019.

[18] B. Lang, "Introduction to Unit Testing in Django," *Real Python*, 2022. [Online]. Available: https://realpython.com/testing-in-django/

[19] R. Patel and V. Sharma, "Integration Testing Techniques in Web Applications," *International Journal of Advanced Research in Computer Science*, vol. 10, no. 6, pp. 56–61, Nov. 2019.

[20] W. Wang and A. Bhattacharya, "Designing Scalable Web Applications Using Django and PostgreSQL," *International Journal of Computer Applications*, vol. 160, no. 8, pp. 24–30, Feb. 2017.

[21] S. Gupta, "Design and Development of a Web-Based Registration Portal," *Journal of Software Engineering and Applications*, vol. 12, pp. 153–159, May 2019.

[22] S. Chakraborty, “Digitalization in Dance Education: The Role of Web Portals,” *International Journal of Cultural Studies*, vol. 8, no. 3, pp. 45–50, Aug. 2021.

[23] M. Ahmed and F. Z. Rehman, “Full-Stack Web Development Using Django and React,” *International Journal of Modern Education and Computer Science*, vol. 14, no. 1, pp. 58–64, Jan. 2022