**Sansiro Pool Club - Web Application Documentation**

**GROUP MEMBERS**

* PRUDENCE NJOGU SCT212-0052/2023
* MERCY NZAMBI SCT212-0058/2023
* KERRY NKIRIA SCT212-0327/2023
* ADRIAN KAGAMBI-SCT212-0305/2023

**Distinctiveness and Complexity**

The Sansiro Pool Club web application is a distinctive project that serves as a membership management system for a pool billiards club. The uniqueness of this project lies in its specific focus on a niche recreational business with carefully implemented membership tiers. Unlike typical e-commerce platforms or social media applications commonly developed for web development courses, this project addresses the practical needs of a small business in the sports and entertainment sector.

**Key Distinctive Features**:

* Specialized Business Focus: The application specifically caters to a pool club business model with membership tiers and pricing.
* Tiered Membership System: Implementation of different membership options (monthly, quarterly, annual) with distinct pricing.
* Mobile-Responsive Design: The interface adapts seamlessly to various screen sizes while maintaining functionality and aesthetic appeal.

**Complexity Elements:**

1. Backend Data Management: The application uses Django models to store and manage membership data with appropriate field validation.
2. Form Validation: Client-side JavaScript validation complements server-side Django validation to ensure data integrity.
3. Responsive Design Implementation: CSS media queries and flexible layouts ensure proper rendering across devices.
4. Django Integration: Proper utilization of Django's template system, URL routing, and form handling.
5. Interactive UI Elements: JavaScript-powered interactions enhance user experience and form submission process.

**Design Approach**

**Backend Design**

The backend was developed using Django, following the MVT (Model-View-Template) architecture. This approach was chosen for:

* Rapid Development: Django's built-in features accelerate development.
* Security: Django's protection against common vulnerabilities like CSRF, SQL injection, etc.
* Scalability: The structure allows for easy addition of new features and models.

The Membership model was created to store user information and their selected membership types. The model includes validation for email fields and uses choices for membership types to ensure data consistency.

**Frontend Design**

The frontend design prioritizes both aesthetics and usability with:

* Clean Interface: Minimalist design with clear sections and intuitive navigation.
* Responsive Layout: Flexible grid system and media queries to ensure compatibility with all devices.
* Visual Hierarchy: Important elements like the membership form and benefits are prominently displayed.
* Consistent Styling: Cohesive color scheme and typography throughout the application.

JavaScript was used for client-side form validation to improve user experience by providing immediate feedback on form errors before submission.

**Integration Approach**

The integration between frontend and backend was achieved through:

* Django templates with static files management for CSS and JavaScript
* Form handling through Django views with POST request processing
* Clear separation of concerns between presentation logic and business logic

**File Contents and Structure**

**Django Files**

\_\_init\_\_.py

* An empty file that indicates to Python that the directory should be treated as a Python package.

admin.py

* Registers the Membership model with the Django admin site.
* Allows administrators to view and manage membership records through Django's admin interface.

apps.py

* Contains the MyappConfig class for the application configuration.
* Specifies the default auto field for model primary keys.

models.py

* Defines the Membership data model with the following fields:
  + full\_name: CharField for storing member's name
  + email: EmailField with validation for proper email format
  + membership type: CharField with choices for monthly, quarterly, and annual options
  + message: Optional TextField for additional information

tests.py

* Currently empty, prepared for future unit tests.

urls.py

* Defines URL patterns for the application:
  + Root URL path mapped to home view
  + '/about/' URL path mapped to about view

views.py

* Contains view functions that handle HTTP requests and return appropriate responses:
  + home (): Renders the home page and processes form submissions for new memberships
  + about (): Renders the about page with information about the club

**Templates**

home.html

* Main landing page template with:
  + Club introduction and branding
  + Membership benefits and perks section
  + Image gallery of available games
  + Membership signup form
  + Navigation to about page
  + Footer with copyright information
* Integration with CSS and JavaScript through static files

about.html

* Secondary page with detailed information about the club:
  + Club history and mission
  + Description of facilities and community
  + Simple, focused design without distractions

**Static Files (referenced in templates)**

* CSS files for styling (home-style.css and about-style.css)
* JavaScript file (home-js.js) for form validation
* Images of pool games and facilities

**How to Run the Application**

**Prerequisites (what is needed?)**

* Python 3.x
* Django (latest stable version)
* Web browser

**Setup Instructions**

1. Clone or download the project files to your local machine
2. Install required dependencies:
3. pip install django
4. Navigate to the project directory containing manage.py
5. Run migrations to create the database:
6. python manage.py makemigrations
7. python manage.py migrate
8. Create a superuser for accessing the admin panel (optional but recommended):
9. python manage.py createsuperuser
   1. Follow the prompts to set up an admin username, email, and password.
10. Start the development server:
11. python manage.py runserver
12. Access the application by opening a web browser and navigating to:
    1. Main site: http://127.0.0.1:8000/
    2. About page: http://127.0.0.1:8000/about/
    3. Admin panel: http://127.0.0.1:8000/admin/ (login with superuser credentials)

**Using the website**

1. The home page displays information about the pool club and its membership options
2. Users can fill out the membership form to join the club
3. Form validation ensures all required fields are properly completed
4. Upon successful submission, the user's membership information is saved to the database
5. Administrators can view and manage all memberships through the Django admin panel

**Additional Information**

**Technical Considerations**

* The application follows Django best practices for security and performance
* Static files are properly configured for a production environment
* The model design allows for easy expansion of membership features

**Responsive Design Testing**

The application has been tested on various devices and screen sizes to ensure compatibility:

* Desktop (1920x1080)
* Mobile (375x667)

All features remain functional and aesthetically pleasing across these form factors.

**Conclusion**

The Sansiro Pool Club web application successfully meets all project requirements while delivering a useful and distinctive product. The combination of Django's powerful backend and JavaScript-enhanced frontend creates a robust platform for membership management with room for future expansion and improvement.