A
Project Synopsis

On

Django-Based Blog Web Application

Submitted to
Savitribai Phule Pune University

**In Partial Fulfillment of the requirement of the award of the degree
of
Master of Computer Application
FYMCA, Sem I
Academic Year 2025-26**

Submitted by
Mr. Prathamesh Vijay Pavale

Department of Computer Application

RAJMATA JIJAU SHIKSHAN PRASARAK MANDAL's

INSTITUTE OF COMPUTER & MANAGEMENT RESEARCH

1.Title:

Django-Based Blog Web Application

2. Introduction:

Blogging has become a fundamental tool for content creation, knowledge sharing, and personal expression on the internet. However, many existing blog platforms lack flexibility, customization options, or user-friendly interfaces for both administrators and content creators. Traditional blog systems often rely on outdated content editors or lack seamless image management capabilities.

Problem Statement:

Users need a modern, scalable blog platform that:

- Provides an intuitive rich-text editing experience for creating and formatting posts
- Supports seamless image uploading and management directly within the editor
- Ensures secure user authentication and role-based access control
- Offers easy post management (create, read, update, delete) with minimal technical overhead
- Provides a responsive, user-friendly interface accessible across devices

Solution Overview

This project develops a Django-based blog web application leveraging the Froala WYSIWYG editor for rich-text content creation with integrated image upload functionality. The application provides:

- User registration, authentication, and profile management
- Full-featured post lifecycle management (create, edit, publish, delete)
- Direct image uploads within the editor with organized file storage (`media/blog_pics/<username>/post_<id>/images/`)
- Role-based permissions (author ownership, admin moderation)
- Responsive UI built with Bootstrap 5
- Structured database schema with Django ORM

3. Objectives:

Primary Objectives

1. Design and implement a secure user authentication system with registration, login/logout, and profile management capabilities.

2. Develop a rich-text content editor interface using Froala WYSIWYG editor for intuitive post creation and editing.
3. Build an integrated image upload management system with organized folder structure based on user and post identifiers.
4. Implement a complete post management system supporting create, read, update, and delete (CRUD) operations with author-based access control.
5. Create a responsive user interface using Bootstrap 5 that works seamlessly on desktop and mobile devices.
6. Establish role-based access control differentiating permissions for regular users and administrators.

Secondary Objectives

7. Implement error handling, validation, and security measures (CSRF protection, input sanitization).
8. Design a scalable database schema with proper relationships and constraints.
9. Provide comprehensive API/URL routing for frontend-backend communication (REST-like endpoints).
10. Create detailed system documentation including UML diagrams (DFD, Class, Use Case, Sequence, Component, Deployment, Activity).

4. Scope:

Included Features:

User Management

- User registration with email validation
- Secure login/logout using Django's built-in authentication
- User profile with profile picture upload (stored in `media/profile_pics/`)
- Password management (set/change password)

Post Management

- Create new blog posts with title, rich-text content (Froala editor), and optional cover image
- Edit existing posts (author-only by default)
- Delete posts (author or admin)
- List all posts with pagination and filtering
- View detailed post information
- Author attribution and date-posted metadata

Image Upload & Management

- Direct image upload within Froala editor via AJAX
- Custom upload endpoint (`/froala_editor/upload_image/`) handling multipart requests

- Organized storage structure: `media/blog_pics/<username>/post_<id>/images/<uuid>.<ext>`
- CSRF token validation for secure uploads
- File type and size validation (configurable limits)
- JSON response with media URLs for editor embedding

Content Display

- Homepage listing all posts with pagination (3 posts per page)
- User-specific posts view (filter by author)
- Post detail view with full content rendering and media display
- Search and filter capabilities (by author)

Admin Features

- Django admin panel for site management
- User and post moderation capabilities
- Content override/deletion by admin
- System configuration and settings management

Security Features

- CSRF protection on all POST requests
- SQL injection prevention via Django ORM
- Input sanitization and validation
- Session-based authentication
- Role-based access control (LoginRequiredMixin, UserPassesTestMixin)

Excluded Features

Out of Scope (Future Enhancements)

- Comment system (not included in current version)
- User-to-user messaging or notifications
- Email confirmations for registration (optional in code, not enforced)
- Social media sharing or OAuth integrations
- Full-text search engine (basic filtering only)
- API authentication (JWT/OAuth) for third-party apps
- Advanced analytics and statistics dashboards
- Automated backup and disaster recovery
- Multi-language support or localization
- Video embedding or advanced media types (audio, documents)
- Real-time collaboration or drafts system
- CDN integration for media delivery (deployment-level)
- Auto-scaling and load balancing (infrastructure-level)
- Two-factor authentication (2FA)

5. Methodology:

Development Approach

- Phase 1: Planning & Design (Weeks 1-2)
- Phase 2: Backend Development (Weeks 3-5)
- Phase 3: Frontend Development (Weeks 6-7)
- Phase 4: Integration & Testing (Weeks 8-9)
- Phase 5: Deployment & Documentation (Week 10)

Tools & Technologies

- Backend Framework: Django 6.0 (Python)
- Database: SQLite (development) / PostgreSQL (production-ready)
- Frontend:HTML5, CSS3, Bootstrap 5, JavaScript (Froala client)
- Rich-Text Editor: Froala Editor (Django package)
- ORM: Django ORM
- Authentication: Django's built-in User model and authentication
- Image Processing:PIL/Pillow (for image resizing, if needed)
- Testing: Django TestCase, unittest
- Version Control: Git
- Documentation:Markdown, UML diagrams, PlantUML

Development Environment

- Python 3.8+
- Virtual environment (venv)
- Django development server
- SQLite for local development
- Code editor (VS Code)

6. Literature Review:

Modern web development heavily relies on secure, scalable frameworks like Django, which follows the MTV architecture and provides built-in mechanisms for authentication and data handling. Rich-text editors such as Froala enhance user experience by enabling intuitive content creation with advanced formatting and media support. Previous studies highlight the importance of organized media management and robust upload handling to maintain system security and performance. Responsive frontend design using frameworks like Bootstrap ensures accessibility across devices and improves usability. Overall, existing research emphasizes integrating modern UI tools, secure backend systems, and efficient file-management practices to build effective blogging platforms.

7. Implementation Plan:

Development Phases & Milestones

Phase 1: Project Setup & Infrastructure (Week 1)

Milestones:

- Set up Django project structure
- Configure database (SQLite for dev)
- Install dependencies (Django, Froala, Bootstrap, Pillow)
- Create apps: \`blog\`, \`users\`, \`core\`
- Prepare development environment

Deliverables:

- \`manage.py\`, \`settings.py\`, \`urls.py\`, \`requirements.txt\`
- Basic project scaffold

Phase 2: Database & Models (Week 2)

Milestones:

- Design and implement models: \`User\`, \`Post\`, \`Profile\`
- Create migrations
- Set up admin interface
- Define relationships and constraints

Deliverables:

- \`blog/models.py\`, \`users/models.py\`
- Migration files
- \`admin.py\` configurations

Phase 3: Authentication & User Management (Week 3)

Milestones:

- Implement user registration view
- Build login/logout functionality
- Create user profile view and update
- Add profile picture upload
- Set up session management

Deliverables:

- \`users/views.py\`, \`users/forms.py\`
- \`users/templates/\` (register.html, login.html, profile.html, logout.html)
- \`users/signals.py\` for profile creation on user signup

Phase 4: Post Management CRUD (Week 4)

Milestones:

- Create post creation view with Froala editor

- Build post list view with pagination
- Implement post detail view
- Add post update view (author-only)
- Implement post delete view (author/admin)

Deliverables:

- `blog/views.py` (PostCreateView, PostListView, PostDetailView, PostUpdateView, PostDeleteView)
- `blog/forms.py` (PostForm with FroalaEditor widget)
- `blog/templates/` (home.html, Post_form.html, Post_detail.html, user_posts.html)

Phase 5: Image Upload Integration (Week 5)

Milestones:

- Build custom Froala upload endpoint (`froala_image_upload` view)
- Implement file storage logic (organized folder structure)
- Add CSRF and file validation
- Return JSON with media URLs
- Ensure Froala editor embeds images correctly

Deliverables:

- `blog/views.py` (froala_image_upload function)
- `core/urls.py` (custom upload route)
- `core/settings.py` (FROALA_EDITOR_UPLOAD_PATH, MEDIA_ROOT, MEDIA_URL)

Phase 6: Frontend & UI (Week 6)

Milestones:

- Design and implement base template with Bootstrap
- Create page templates (home, about, post detail, forms)
- Add responsive navigation
- Implement CSS styling and custom branding
- Test on multiple devices

Deliverables:

- `blog/templates/` (base.html, about.html, home.html, post_confirm_delete.html, Post_detail.html, Post_form.html, user_posts.html)
- `blog/static/blog/` (main.css, bootstrap.min.css)

Phase 7: Testing & Debugging (Week 7)

Milestones:

- Write unit tests for models and views
- Perform end-to-end testing
- Test image upload functionality
- Security testing (CSRF, input validation)
- Performance optimization

Deliverables:

- `blog/tests.py`, `users/tests.py`
- Test coverage report

Phase 8: Documentation (Week 8)

Milestones:

- Create project synopsis and README
- Generate UML diagrams (Class, Use Case, DFD, Sequence, Component, Deployment, Activity)
- Document API endpoints and URL routing
- Prepare deployment guide
- Create user manual

Deliverables:

- README.md, Project_Synopsis.md
- UML diagrams (text + PNG)
- API documentation
- Deployment guide

Phase 9: Final Review & Presentation (Week 9)

Milestones:

- Code review and refactoring
- Final testing and bug fixes
- Prepare presentation slides
- Demo preparation

Deliverables:

- Presentation slides
- Demo-ready application
- Final code repository

8. Expected Outcomes:

Functional Outcomes

- Fully Operational Blog Application: A complete, working blog platform with user registration, post management, and image uploads.
- User-Friendly Interface: Intuitive UI with Bootstrap 5 responsive design, accessible on desktop and mobile devices.
- Secure Authentication System Django-based authentication with password hashing, session management, and role-based access control.

  Rich-Text Content Creation: Froala editor integration enabling users to create beautifully formatted posts with embedded images.

  Organized Media Storage:Structured file storage system (`media/blog_pics/<username>/post_<id>/images/`) for easy management and scalability.
- Admin Control Panel: Django admin interface for moderating users, posts, and system settings.

Technical Outcomes

- Scalable Architecture: Django MTV pattern enables easy feature additions and maintenance.
- Database Integrity: Proper schema design with relationships, constraints, and migrations ensuring data consistency.
- Security Implementation: CSRF protection, input validation, SQL injection prevention, and secure password handling.
- Error Handling: Comprehensive error handling and user feedback mechanisms.

Documentation Outcomes

- System Documentation: Complete UML diagrams (Class, Use Case, DFD, Sequence, Component, Deployment, Activity) and Entity Relationship Diagram.
- Code Documentation: Inline comments, docstrings, and README with setup instructions.
- Deployment Guide: Step-by-step instructions for deploying to production (Docker, traditional server, cloud platforms).
- Project Report: Comprehensive project synopsis and technical documentation.

Learning Outcomes

- Skills Development: Understanding of Django framework, web security, database design, and frontend development.

  Software Engineering: Experience with agile methodology, testing, documentation, and version control.

- Problem-Solving: Ability to identify and resolve technical challenges in full-stack web development.

Quality Metrics

- Code Coverage: Minimum 70% -unit test coverage
- Performance: Page load times < 2 seconds on standard network
  Responsive Design: Supports devices from 320px (mobile) to 1920px+ (desktop)
- Security: Zero critical vulnerabilities, OWASP compliance
- Usability: User satisfaction score > 4/5 in testing

9. Conclusion:

This project successfully delivers a modern, secure, and user-friendly blog platform leveraging Django and Froala editor. With comprehensive documentation, security implementation, and scalable architecture, it serves as both a functional application and an educational resource for web development. The organized image upload system and rich-text editing experience provide a solid foundation for content creators and demonstrate professional software engineering practices.

10. References:

Official Documentation

- Django Project (2025). "The Web Framework for Perfectionists." Retrieved from https://www.djangoproject.com/
- Django Documentation (2025). "Models, Forms, Views, Authentication." Retrieved from https://docs.djangoproject.com/
- Froala Editor (2025). "WYSIWYG HTML Editor." Retrieved from https://www.froala.com/wysiwyg-editor
- Bootstrap (2025). "The Most Popular CSS Framework." Retrieved from https://getbootstrap.com/

Security & Best Practices

- OWASP (2025). "OWASP Top 10 Web Application Security Risks." Retrieved from https://owasp.org/www-project-top-ten/
- Mozilla Developer Network (2025). "Web Security." Retrieved from https://developer.mozilla.org/en-US/docs/Web/Security

Web Development & Frontend

- Flanagan, D. (2020). "JavaScript: The Definitive Guide" (7th ed.). O'Reilly Media.
- Duckett, J. (2011). "HTML and CSS: Design and Build Websites." Wiley.
- Robson, E., & Freeman, E. (2012). "Head First Design Patterns." O'Reilly Media.

Additional Resources

- Real Python. "Django Tutorials." Retrieved from https://realpython.com/
- Full Stack Python. "Django." Retrieved from https://www.fullstackpython.com/django.html
- Stack Overflow. "Django Questions and Solutions." Retrieved from https://stackoverflow.com/questions/tagged/django