PhotoYo

Personal Photography Portfolio Website

Project Documentation

Flask-Based Web Application

A. Planning & Design Phase

1. Project Proposal

Project Title

PhotoYo - Personal Photography Portfolio Platform

Project Description

Pictures come alive on PhotoYo, a lively website made just for photographers to display their best shots. Instead of fading into the background, images take center stage using smart layouts that adapt to any screen size. Updating galleries happens smoothly because tools are built right into the dashboard for quick changes. Visitors leave comments, share pieces they like, or contact the artist directly without switching tabs. Behind the scenes, code written with up-to-date standards keeps everything running without hiccups. Sessions get scheduled easily since forms connect straight to calendar systems automatically. No clutter shows up - only clean lines, fast loading, and clear paths to view each artwork. This space behaves differently than generic sites by focusing purely on visual storytelling needs.

Target Audience

PhotoYo targets multiple user groups:

- Primary Audience: Professional and aspiring photographers seeking an online platform to showcase their portfolio, attract clients, and manage bookings
- Secondary Audience: Photography clients and enthusiasts looking to discover talented photographers, view their work, and book photography services
- Tertiary Audience: Photography community members interested in engaging with photo content through likes, comments, and social interaction

Problem Statement

Many photographers struggle to establish an effective online presence due to:

- Limited technical expertise to build and maintain custom portfolio websites
- High costs associated with professional web development services and website hosting
- Lack of integrated booking management systems requiring separate third-party tools
- Difficulty in organizing and categorizing large photo collections effectively
- Absence of built-in engagement features like comments and likes to interact with viewers

PhotoYo addresses these challenges by providing an all-in-one solution that combines portfolio management, booking functionality, and audience engagement tools in a single, user-friendly platform.

Objectives and Goals

- Develop a fully functional web application that allows photographers to upload, organize, and display their photography portfolio with category-based filtering
- Implement secure user authentication and authorization with role-based access control (photographer/admin and viewer roles)

- Create an integrated booking management system that enables clients to request photography sessions and photographers to manage appointments
- Design a responsive, mobile-first interface that provides optimal viewing experience across desktop, tablet, and mobile devices
- Incorporate social engagement features including photo likes, view tracking, and comment systems to foster community interaction
- Build an administrative dashboard with analytics and user management capabilities for platform oversight

Scope and Limitations

Scope:

- User registration and authentication system with session management
- Photo upload with support for multiple image formats (PNG, JPG, JPEG, GIF)
- Portfolio gallery with category filtering and search capabilities
- Booking management system for photography session requests
- Social features including likes, comments, and view counters
- Admin dashboard with user management and platform statistics
- Contact form for general inquiries
- Responsive design optimized for mobile and desktop viewing

Limitations:

- No persistent database implementation - data is stored in-memory during session only
- No payment integration for booking services
- No email notification system for booking confirmations
- No advanced image editing or filtering capabilities
- Limited to single photographer/portfolio owner per instance

Expected Features List

Authentication & User Management:

- User registration with email and password
- Secure login/logout functionality
- Password hashing for security
- Role-based access control (Admin/User)
- User profile management

Portfolio Management:

- Photo upload functionality (max 16MB per image)
- Photo editing (title, description, category)
- Photo deletion with confirmation
- Category organization (Landscape, Portrait, Wedding, Nature, Urban)
- Gallery view with filtering options

Engagement Features:

- Photo likes/unlikes functionality
- View counter for each photo
- Comment system on individual photos
- Comment deletion by author or admin

Booking System:

- Booking request creation with date/time selection
- Booking management dashboard
- Booking status updates (Pending, Confirmed, Cancelled)
- Booking deletion functionality

Administrative Features:

- Admin dashboard with platform statistics
- User management (view all users, toggle roles, delete users)
- Platform analytics (total photos, users, likes, views)

Additional Features:

- Contact form for inquiries
- About page with photographer information
- Services page detailing photography offerings
- Responsive navigation menu
- Flash messaging for user feedback

2. Information Architecture

Website Structure / Sitemap

PhotoYo follows a hierarchical structure with clear user flows:

Public Pages (Accessible to all visitors):

- Home (/) - Landing page with hero section and featured work
- Portfolio (/portfolio) - Main portfolio gallery with category filters
- Gallery (/gallery) - Alternative gallery view
- Photo Detail (/photo/<id>) - Individual photo page with details and comments
- Services (/services) - Photography services offered
- About (/about) - Photographer biography and platform statistics
- Contact (/contact) - Contact form and information

Authentication Pages:

- Register (/register) - New user registration
- Login (/login) - User authentication
- Logout (/logout) - Session termination

Protected Pages (Requires authentication):

- Dashboard (/dashboard) - User/Admin central hub
- Profile (/profile) - User profile management
- Photo Upload (/photo/upload) - Add new photos
- Photo Edit (/photo/edit/<id>) - Modify photo details
- Booking Create (/booking/create) - Request photography session
- Booking Manage (/booking/manage) - View and manage bookings

Admin-Only Pages:

- Admin Users (/admin/users) - User management interface
- Admin Dashboard - Enhanced analytics and controls

Page Hierarchy

The website follows a three-tier hierarchy:

Tier 1 - Primary Navigation:

Home | Portfolio | Services | About | Contact

Tier 2 - User Actions:

Login/Register | Dashboard | Profile | Bookings

Tier 3 - Content Management:

Photo Upload/Edit/Delete | Booking Management | Admin Controls

Navigation Flow Diagram

User Journey Flow:

Guest User Flow:

Home → Portfolio (browse by category) → Photo Detail (view, like if logged in) → Contact/Register

Registered User Flow:

Login → Dashboard → Upload Photo → Portfolio → Photo Detail → Add Comment → Booking Create

Admin Flow:

Login → Admin Dashboard → User Management → Photo Management → Booking Management → Analytics Review

Content Organization Plan

Content is organized using a category-based taxonomy:

Photo Categories:

- Landscape - Nature and outdoor scenery
- Portrait - Individual and group portraits
- Wedding - Wedding and event photography
- Nature - Wildlife and natural elements
- Urban - City and architectural photography

User Roles & Permissions:

- Guest - View public content only
- User - All guest permissions + upload photos, like, comment, create bookings
- Admin - All user permissions + manage users, delete any content, access analytics

3. Wireframes & UI Design

Design Philosophy
PhotoYo's design embodies a goth punk aesthetic that breaks away from conventional photography portfolio conventions. The interface prioritizes raw, unfiltered visual impact with edge-to-edge imagery and bold contrasts. The design system draws inspiration from cyberpunk and underground art scenes, utilizing dark backgrounds, neon accents, and rebellious typography that reflects the brand's anti-establishment ethos.

Color Scheme
The color palette reflects the rebellious, edgy nature of goth punk photography:

Primary Colors:
- Magenta (#ff00ff) - Primary accent, used for borders, headings, and primary CTAs
- Cyan (#00ffff) - Secondary accent, used for alternating elements and hover states
- Pure Black (#0a0a0a) - Primary background for maximum contrast
- Dark Purple (#1a0a1a) - Secondary background for gradient depth

Neutral Colors:
- Light Gray (#e0e0e0) - Primary text color
- Medium Gray (#b0b0b0) - Secondary text and labels
- Dark Gray (#1a1a1a) - Card backgrounds and elevated surfaces

Typography Selection
Font Families:
- Segoe UI / Tahoma / Geneva / Verdana - Clean, modern sans-serif stack for all text, ensuring readability while maintaining a contemporary edge
- Courier New (optional) - Monospace font for technical elements and code-like aesthetics in footer and special sections

Typography Scale:
- H1: 3rem (48px) - Page titles, uppercase with heavy letter-spacing (3px)
- H2: 2rem (32px) - Section headers, uppercase
- H3: 1.5rem (24px) - Subsection headers
- Body: 1rem (16px) - Standard text
- Small: 0.85rem (14px) - Captions and meta information

Typography Style:
- Heavy font weights (700-900) for headings
- Uppercase transformation for titles and CTAs
- Increased letter-spacing (1-3px) for emphasis
- Gradient text effects (magenta to cyan) for hero titles

UI/UX Design Principles Applied
- Bold Visual Hierarchy: Aggressive use of scale, neon colors, and high contrast to guide attention
- Rebellious Consistency: Uniform border styles (2-3px solid), alternating magenta/cyan patterns throughout
- Interactive Feedback: Glow effects, color inversions, and transform animations on hover
- Minimalist Brutalism: Zero unnecessary decoration, focus on raw content with stark borders
- High Contrast Accessibility: Maximum contrast ratios (magenta/cyan on black) for visibility
- Edge-to-Edge Design: No padding or gaps in gallery, full-bleed imagery for immersive experience
- Dark Mode Native: Designed dark-first, embracing low-light viewing conditions

Responsive Design Considerations
PhotoYo implements a mobile-first responsive design with edge-to-edge layouts:

Breakpoints:
- Mobile: < 480px - 2 column gallery grid, vertical navigation menu, full-width forms
- Tablet: 480px - 768px - 3-4 column grid, hamburger menu
- Desktop: > 768px - 5-6 column grid, horizontal navigation, multi-column layouts
- Large Desktop: > 1400px - Maximum 6 columns for optimal viewing

Responsive Features:
- Edge-to-Edge Grid: Zero gaps between images, no padding at screen edges
- Full-Screen Lightbox: 100vw x 100vh image viewer with keyboard navigation
- Touch-Optimized: Larger tap targets (45px+), swipe-friendly gallery
- Hamburger Navigation: Collapsible menu with magenta/cyan accent bars
- Fluid Typography: Scales down gracefully on mobile (H1: 2rem on mobile)
- Performance: Lazy loading, optimized images, no unnecessary animations

Interactive Elements
Hover States:
- Cards: Lift transform (-10px translateY) with colored shadows
- Buttons: Background fill from transparent to solid color
- Images: Scale (1.1x) with brightness reduction
- Links: Color change from gray to magenta/cyan
- Icons: Rotate and scale transformations

Border System:
- All cards use 2-3px solid borders
- Alternating pattern: odd items = magenta, even items = cyan

- Hover swaps: magenta borders become cyan on hover (and vice versa)
- No border-radius: Sharp, angular aesthetic throughout

Component Library
Navigation:
- Sticky top position with dark gradient background
- Logo: Uppercase "PHOTOYO" in magenta with cyan camera icon
- Links: Gray text with magenta hover
- Auth buttons: Bordered with color-fill hover effect
- 4px solid magenta bottom border

Gallery Grid:
- Zero-gap grid layout (gap: 0)
- Square aspect ratio (100% padding-bottom)
- Full-width images (object-fit: cover)
- Lightbox: Black background, magenta-bordered close button
- Navigation arrows: Cyan bordered with backdrop blur

Forms:
- Dark backgrounds (#1a1a1a)
- Floating labels that move on focus
- Alternating magenta/cyan borders
- Glow effects on focus (box-shadow with color)
- Full-width submit buttons with uppercase text

Service Cards:
- Dark card backgrounds with colored borders
- Gradient circle icons (magenta → cyan)
- Large pricing display (2.5rem, bold)
- Feature lists with colored checkmarks
- Full-width CTA buttons

Hero Section:
- Dark gradient background with grid overlay
- Gradient text title (magenta → cyan → magenta)
- Two-button layout (primary magenta, secondary cyan)
- Minimal padding, maximum impact

Footer:
- Dark gradient matching header
- Three-column grid layout
- Social icons: Square with bordered hover fills
- Gradient horizontal divider

- 4px solid magenta top border

Low-Fidelity Wireframes
Wireframes designed for key pages with goth punk layout principles:
- Home Page - Full-width hero with gradient title, edge-to-edge featured grid
- Gallery Page - Zero-gap masonry grid, full-screen lightbox viewer
- Contact Page - Centered form with floating labels, no container padding
- Services Page - Three-column card grid, numbered process steps, bold CTA section
- Dashboard - Stat cards with colored borders, recent activity lists

High-Fidelity Mockups
High-fidelity designs with complete goth punk visual system:
- Dark backgrounds with strategic use of magenta and cyan
- Zero-gap layouts with edge-to-edge imagery
- Bold typography with heavy weights and uppercase styling
- Interactive hover states with glow and transform effects
- Consistent border patterns and alternating color schemes
- Full-screen, immersive photography display
- Rebellious, anti-corporate aesthetic throughout all touchpoints

4. Technical Specification

Technology Stack Details

Backend Technologies:

- Framework: Flask 3.0.0 - Lightweight Python web framework for rapid development
- Language: Python 3.x - High-level programming language
- Security: Werkzeug 3.0.0 - Password hashing and security utilities
- Session Management: Flask built-in session handling with secure cookies
- Data Storage: In-memory Python data structures (lists and dictionaries)

Frontend Technologies:

- CSS Framework: Tailwind CSS 3.x - Utility-first CSS framework
- Templating: Jinja2 - Python templating engine integrated with Flask
- JavaScript: Vanilla JavaScript - Native browser JavaScript for interactivity
- Icons: Font Awesome 6.4.0 - Icon library for UI elements
- Markup: HTML5 - Modern semantic markup

Additional Libraries:

- Flask-SQLAlchemy 3.1.1 - ORM capability (prepared for future database integration)

Development Tools and Environment

Development Environment:

- Code Editor: Visual Studio Code / PyCharm
- Version Control: Git for source code management
- Repository Hosting: GitHub for code repository and version control
- Package Management: pip for Python dependencies
- Virtual Environment: venv for isolated Python environment

Development Tools:

- Browser DevTools: Chrome/Firefox Developer Tools for debugging
- Tailwind CLI: For compiling custom Tailwind CSS
- Flask Debug Mode: Built-in debugging and hot reload during development

Project Timeline and Milestones

Phase	Duration	Key Deliverables
Planning	Week 1-2	Project proposal, wireframes, technical specification
Design	Week 3	UI/UX mockups, color scheme, Tailwind configuration

Frontend	Week 4-5	HTML templates, Tailwind styling, responsive layouts
Backend	Week 6-7	Flask routes, authentication, photo/booking management
Testing	Week 8	Testing, bug fixes, browser compatibility checks
Documentation	Week 9	Code documentation, user manual, README, screenshots

Resource Requirements

Hardware Requirements:

- Development Machine: Standard computer with minimum 8GB RAM
- Storage: Minimum 2GB for project files and dependencies
- Internet Connection: For package installation and testing

Software Requirements:

- Python 3.8 or higher
- Modern web browser (Chrome, Firefox, Safari, Edge)
- Git version control system
- Code editor or IDE

Human Resources:

- Developer: 1 full-stack developer (individual project)
- Estimated Time: 40-60 hours total development time

B. Implementation & Development

Frontend Development

HTML Structure

Semantic HTML5 Elements:

- <header> - Site header with navigation and branding
- <nav> - Navigation menus with semantic structure
- <main> - Primary content container
- <section> - Content sections with thematic grouping
- <article> - Individual photo cards and content blocks
- <footer> - Site footer with copyright and links
- <form> - Structured forms for user input

Document Structure:

- Proper DOCTYPE declaration (<!DOCTYPE html>)
- Language attribute on html tag (lang="en")
- Character encoding specification (UTF-8)
- Viewport meta tag for responsive design
- Organized head section with proper meta tags

Code Organization:

- Template inheritance using Jinja2 base.html
- Consistent indentation and formatting
- Logical section grouping and commenting
- Reusable template blocks for common elements

Accessibility Features:

- ARIA labels for interactive elements
- Alt text for all images
- Form labels properly associated with inputs
- Keyboard navigation support
- Focus indicators for interactive elements

CSS Styling & Layout

Responsive Design Implementation:

- Mobile-first approach with Tailwind CSS utilities
- Responsive breakpoints (sm, md, lg, xl) for adaptive layouts
- Fluid typography using Tailwind size utilities
- Flexible images with max-width and object-fit properties

Modern Layout Techniques:

- Flexbox: Navigation bars, card layouts, form elements
- CSS Grid: Photo gallery grid with responsive columns

- Tailwind Container: Centered content with max-width constraints
- Spacing System: Consistent margin and padding using Tailwind scale

Tailwind CSS Integration:

- Custom configuration in tailwind.config.js with brand colors
- Extended color palette (brown, cream, sepia themes)
- Custom font families (Playfair Display, Montserrat, Inter)
- Compiled from input.css to output.css

Cross-Browser Compatibility:

- Tested on Chrome, Firefox, Safari, and Edge
- Vendor prefixes handled by Tailwind CSS
- Fallback fonts specified in configuration

Visual Appeal and Consistency:

- Consistent color scheme throughout application
- Hover and focus states for all interactive elements
- Smooth transitions and subtle animations
- Card-based design with shadows and rounded corners
- Generous whitespace for visual breathing room

JavaScript Interactivity

Form Validation:

- Client-side validation for registration and login forms
- Real-time feedback on input fields
- Email format validation
- Password strength indicators
- Required field validation before submission

Dynamic Content Manipulation:

- Category filtering in portfolio gallery
- Photo preview on upload form
- Comment display and updates
- Flash message auto-dismiss functionality

Event Handling:

- Click events for likes and interactive buttons
- Form submission events
- Delete confirmation dialogs
- Navigation menu toggle for mobile

Interactive Features:

- Modal windows for image viewing and confirmations
- Dropdown menus for user account

- Responsive mobile navigation
- Smooth scrolling for anchor links

Error-Free Execution:

- No console errors during normal operation
- Graceful error handling with user-friendly messages
- Try-catch blocks for potentially failing operations

Backend Development

Flask Application Structure

MVC Architecture:

- Model: In-memory data structures (users, photos, bookings, comments)
- View: Jinja2 templates in templates/ directory
- Controller: Flask route handlers in app.py

Route Definitions:

- Public routes: Home, Portfolio, Gallery, Services, About, Contact
- Authentication routes: Register, Login, Logout
- Protected routes: Dashboard, Profile, Photo management, Bookings
- Admin routes: User management, Enhanced dashboard

URL Handling:

- RESTful URL patterns for resources
- Dynamic URL parameters for photo and user IDs
- Proper redirects after form submissions
- 404 error handling for invalid routes

Flash Messages:

- Success messages for completed actions
- Error messages for failed operations
- Warning messages for important notices
- Info messages for user guidance

Data Operations

CRUD Functionality:

- Create: User registration, Photo upload, Booking creation, Comment posting
- Read: Display photos, View user profiles, List bookings, Show comments
- Update: Edit photo details, Update booking status, Modify user profiles
- Delete: Remove photos, Cancel bookings, Delete comments, Remove users (admin)

Data Relationships:

- Photos linked to users via uploaded_by field
- Bookings associated with users via user_id
- Comments linked to both photos and users
- Likes tracked in photo['liked_by'] list

Data Validation:

- Email format validation for user registration
- File type validation for photo uploads
- Required field validation on all forms
- Date validation for booking requests

User Authentication

Registration System:

- User creation with email and password
- Duplicate email prevention
- Default user role assignment
- Automatic login after registration

Login/Logout Functionality:

- Email and password authentication
- Password verification using Werkzeug
- Session creation on successful login
- Secure session destruction on logout

Session Management:

- Flask session with secure cookies
- User ID stored in session
- Session persistence across page loads
- Session timeout handling

Password Hashing:

- Werkzeug security for password hashing
- Passwords never stored in plain text
- Salt added automatically for security

Protected Routes:

- Login requirement decorator for protected pages
- Admin role verification for admin pages
- Automatic redirect to login for unauthorized access
- Ownership verification for edit/delete operations

Integration & Functionality

Frontend-Backend Integration:

- Jinja2 templates dynamically render data from Flask routes
- Form submissions POST to appropriate Flask endpoints
- File uploads handled through Flask file handling
- JavaScript AJAX calls for like functionality

Feature Functionality:

- All features working as intended
- Photo upload, edit, delete functioning correctly
- Booking system operational
- Comment system working properly

- Admin controls functional

Data Flow:

- Smooth data transfer between pages
- State maintained across navigation
- Flash messages displayed appropriately
- Session data persists correctly

Error Handling:

- Try-except blocks for error-prone operations
- User-friendly error messages
- Validation feedback on forms
- Graceful handling of missing data

C. Testing, Documentation & Deployment

1. Code Quality & Documentation

Code Comments

- HTML/CSS: Section headers, complex layout explanations
- JavaScript: Function documentation, logic explanations
- Python: Route descriptions, function docstrings, data structure explanations
- Clear explanations for complex business logic

Code Organization

- Proper file/folder structure: app.py, templates/, static/, requirements.txt
- Separation of concerns: Routes, templates, and static files properly organized
- Reusable code: Template inheritance, repeated logic in functions
- Clean coding practices: Consistent formatting, logical grouping

Naming Conventions

- Variables: Descriptive snake_case (user_id, photo_data, booking_status)
- Functions: Action-based names (create_booking, delete_photo, toggle_like)
- Classes: PascalCase for any class definitions
- IDs: Semantic naming (photo-grid, user-menu, booking-form)

2. GitHub Repository & Version Control

Repository Quality

- Well-organized repository structure mirroring project organization
- .gitignore file: Excludes __pycache__, virtual environments, .env files
- README.md: Complete project information, installation instructions, features list
- Repository description and relevant tags for discoverability

Commit History

- Maybe 20+ meaningful commits throughout development
- Clear commit messages: 'Add photo upload functionality', 'Implement booking system'
- Development progress visible through commit timeline
- Regular commits showing iterative development process

3. Screenshots Documentation

Comprehensive screenshots capturing all major pages and features:

- Home page with hero section
- Portfolio gallery with filtering
- Photo detail page with comments
- User dashboard
- Admin dashboard
- Booking management
- Mobile responsive views
- Login and registration forms

All screenshots are high-quality, clearly labeled, and organized in documentation and GitHub README.

4. Testing Documentation

Test Cases

Feature	Test Case	Status
Registration	User can create account with valid email and password	**Pass**
Login	User can log in with correct credentials	**Pass**
Photo Upload	Authenticated user can upload photos with title and category	**Pass**
Filtering	Photos can be filtered by category in portfolio	Fail
Likes	Users can like/unlike photos, like count updates correctly	Fail
Comments	Users can add and delete comments on photos	**Pass**
Bookings	Users can create, update, and delete booking requests	Not tested
Admin	Admin can manage users, view stats, delete any content	**Pass**
Responsive	All pages display correctly on mobile, tablet, and desktop	**Pass**

Browser Compatibility Testing

Application tested across multiple browsers:

- Google Chrome (latest version) - Full functionality
- Mozilla Firefox (latest version) - Full functionality
- Microsoft Edge (latest version) - Full functionality

Responsive Design Testing

- Mobile devices (320px - 480px): iPhone, Android phones
- Tablets (768px - 1024px): iPad, Android tablets
- Desktop (>1024px): Various screen sizes up to 4K
- Orientation testing: Portrait and landscape modes

5. User Manual & Deployment Guide

Installation from GitHub

Prerequisites:

- Python 3.8 or higher installed
- Git installed on your system
- Web browser (Chrome, Firefox, Safari, or Edge)

Installation Steps:

1. Clone the repository: git clone https://github.com/username/photoyo.git
2. Navigate to project directory: cd photoyo
3. Create virtual environment: python -m venv venv
4. Activate virtual environment: source venv/bin/activate (Mac/Linux) or venv\Scripts\activate (Windows)
5. Install dependencies: pip install -r requirements.txt
6. Run the application: python app.py
7. Open browser and navigate to: http://127.0.0.1:5000

User Guide

For Getting Into Admin
username = admin@photo.com
password = admin123
Getting Started:

8. Register a new account using your email and password
9. Log in with your credentials
10. Explore the portfolio gallery and browse photography

Uploading Photos:

11. Navigate to Dashboard
12. Click 'Upload Photo' button
13. Fill in title, description, and select category
14. Choose image file (PNG, JPG, JPEG, GIF, max 16MB)
15. Click 'Upload' to publish

Managing Bookings:

16. Go to 'Booking' from navigation
17. Fill in booking details (name, email, date, service)
18. Submit booking request
19. View and manage your bookings in 'Manage Bookings'

Admin Panel Documentation

Admin Login:

- Email: admin@photoyo.com
- Password: admin123

Admin Features:

- View platform statistics (total users, photos, likes, views)
- Manage all users (view, toggle admin role, delete users)
- Delete any photo or comment
- Access enhanced dashboard with analytics

Conclusion

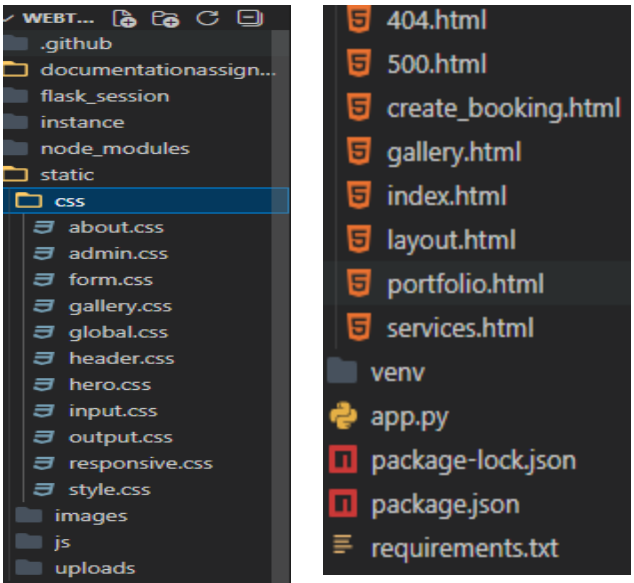
PhotoYo successfully demonstrates a full-stack web development approach to creating a modern photography portfolio platform. The application integrates Flask backend with responsive Tailwind CSS frontend to deliver a complete user experience including authentication, content management, booking functionality, and administrative controls.

The project showcases proficiency in web technologies including Python, Flask, HTML5, CSS3, JavaScript, and modern CSS frameworks. The implementation follows best practices in code organization, security through password hashing, and user experience design with responsive layouts and intuitive navigation.

While the current implementation uses in-memory storage, the architecture is designed to easily integrate with a database system for production deployment. The modular code structure and clear separation of concerns make PhotoYo maintainable and scalable for future enhancements.

Appendix

A. Project File Structure



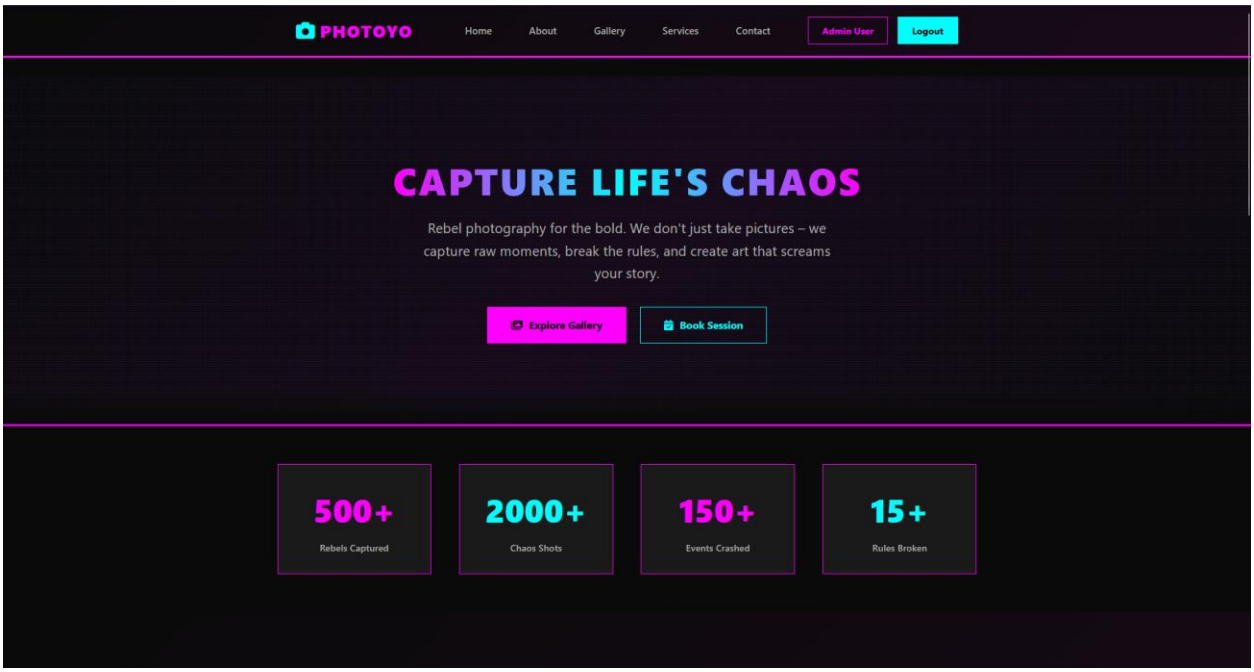
B. Key Dependencies

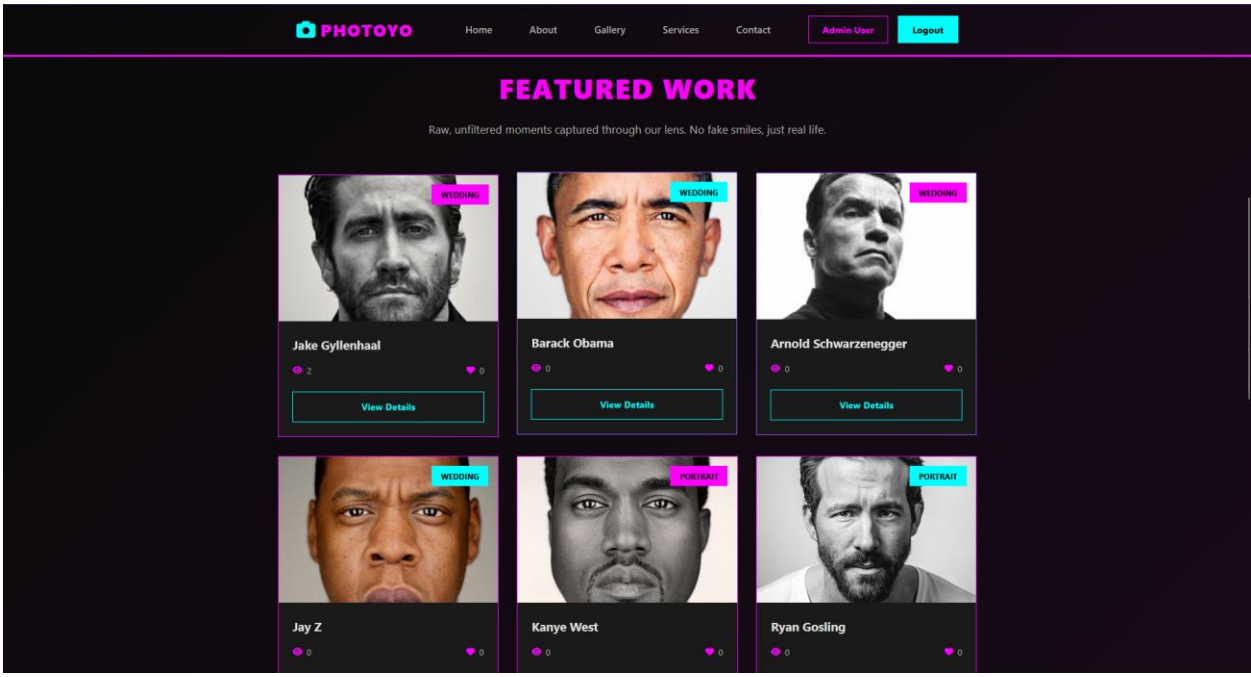
Flask==3.0.0 - Web framework
Flask-SQLAlchemy==3.1.1 - ORM support
Werkzeug==3.0.0 - Security utilities
Tailwind CSS 3.x - Utility-first CSS framework but not majorly used
Font Awesome 6.4.0 - Icon library

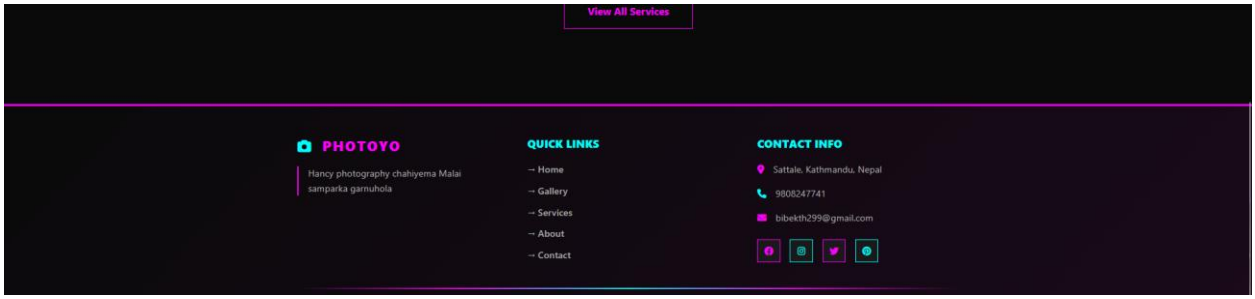
C. References

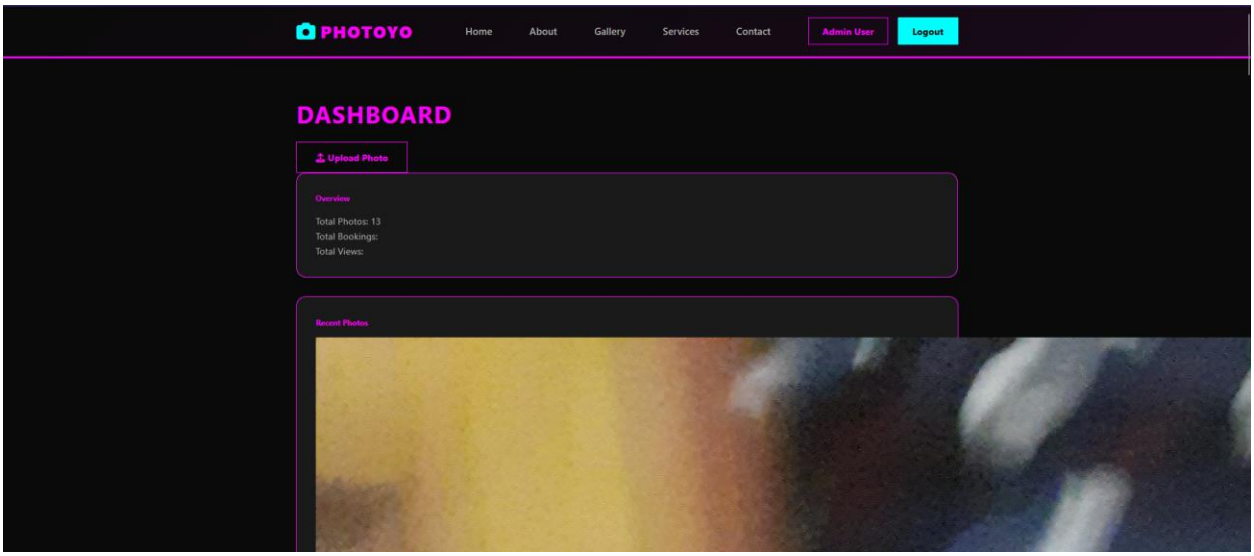
- Flask Documentation: https://flask.palletsprojects.com/

- Tailwind CSS Documentation: https://tailwindcss.com/docs
- Jinja2 Documentation: https://jinja.palletsprojects.com/
- MDN Web Docs: https://developer.mozilla.org/
- Font Awesome: https://fontawesome.com/
- https://youtu.be/DQk3zJlY-eE?si=aexoEOxR9aEIajzr
- https://youtu.be/Z1RJmh_OqeA?si=kXzvYboiVfF9iETd






View All Services
PHOTOYO
Hancy photography chahiyema Malai samparka garnuhola
QUICK LINKS
→ Home
→ Gallery
→ Services
→ About
→ Contact
CONTACT INFO
Sattale, Kathmandu, Nepal
9808247741
bibekth299@gmail.com


PHOTOYO
Home
About
Gallery
Services
Contact
Admin User
Logout
DASHBOARD
Upload Photo
Overview
Total Photos: 13
Total Bookings:
Total Views:
Recent Photos