Photography Portfolio and Client Management System (FrameForge)

1st Aditya Kakade
MIT ADT University
Pune, India

2nd Saniya Shiradkar
MIT ADT University
Pune, India

3rd Himanshu Shirbhate
MIT ADT University
Pune, India

***Abstract*—FrameForge is a website that helps photographers and cinematographers manage their work and share it with others. It is a project that uses React for the parts you see Node.js for the behind the scenes work and SQLite to store all the information. FrameForge has some useful features like a way to upload your photos and videos in a organized way a calendar to keep track of events links to share your photos, with others and a way to download all your photos at once.**

The system also uses a cloud storage system called Amazon S3 which helps handle all the photos and videos and makes sure everything runs smoothly. FrameForge wants to be the one place where photographers and cinematographers can manage their work and talk to their clients.

***Index Terms*—Photography, Client Management, Portfolio System, Flask, React, Web Application, Cloud Storage**

I. Introduction

With the rapid digital transformation in the creative industry, photographers and cinematographers are required to manage multiple clients, projects, and vast media collections effectively. Conventional approaches rely on fragmented tools such as social media, spreadsheets, and cloud drives, leading to disorganization and inefficiency. FrameForge addresses these challenges by providing a unified web-based system that combines client management, portfolio presentation, and workflow automation.

The system aims to simplify portfolio display, automate scheduling, and centralize communication between clients and creative professionals. Built using modern frameworks—Flask for backend, React for frontend, and SQLite for database—it ensures a responsive interface and secure data handling. This paper discusses the design, architecture, and deployment of FrameForge.

II. Literature Survey

Various portfolio management and client coordination tools exist, but they are often designed for general business use rather than tailored creative workflows. The following literature sources provide technical and conceptual foundations relevant to the development of FrameForge.

- **Grinberg, M. (2018).** *Flask Web Development: Developing Web Applications with Python.*
- **Facebook Open Source. (2013).** *React – A JavaScript Library for Building User Interfaces.*
- **SQLite Consortium. (2022).** *SQLite Documentation: Lightweight Database for Embedded Applications.*
- **Pahl, C. (2015).** *Containerization and the PaaS Cloud: A Lightweight Alternative to Virtual Machines.*
- **Choudhary, A., & Patel, N. (2024).** *Full-Stack Development Using React and Flask – A Practical Approach.*
- **Sharma, V., & Rane, P. (2023).** *Designing Scalable Web Applications for Creative Professionals.*
- **GitHub Repository: Luap Pic Peak (2023).**

III. Proposed Methodology

The project follows a three-layered architecture consisting of frontend, backend, and storage layers. FrameForge's architecture combines React, AWS S3, and SQLite to ensure scalability, usability, and streamlined workflow.

A. System Design

FrameForge is made up of three parts: the Presentation Layer, the Application Layer and the Data Layer. This design helps keep everything organized and makes it easier to add features or make changes. Each part can be worked on separately which means we can make sure the whole system stays stable.

- **Presentation Layer:**
  The Presentation Layer is built using React.js, a tool that helps us create user interfaces that're easy to use and look good. This is where photographers and clients interact with FrameForge. They can use it to manage their projects upload files and talk to each other in time. React.js is great because it lets us reuse code and make sure everything works together. It also means that FrameForge works well on devices, like computers and tablets so users can access it from anywhere.
- **Application Layer:**
  The Application Layer is built using Flask, a framework that acts as the brain of the operation. It makes sure that only the right people can use features and handles all the requests from the Presentation Layer. Flask does things like create projects manage files and update the workflow. We use APIs to help the different parts of the system talk to each other. This makes it easy to add features or work with other tools. Flask also helps keep the system safe and secure by managing user sessions and handling errors.

- **Data Layer:**
  The Data Layer uses SQLite to store all the information. This includes things like user names and passwords, client information and project details. SQLite is a database that is perfect for systems like FrameForge. It helps us keep all the data organized. Makes it easy to retrieve when we need it. This is important for things like logging in and tracking projects.
- **System. Workflow:**
  The three main parts of FrameForge work together seamlessly. When a user does something the request goes to the Flask backend, which then talks to the SQLite database to get or update information. The response is then sent back to the user interface. This workflow helps everything run smoothly and quickly. It also makes it easier to find and fix problems.
- **Scalability and Future Enhancements:**
  FrameForge is designed to grow and adapt to technologies. We can easily add features like cloud storage, automatic backups and tools that help us analyze media. We can also use containerization technologies like Docker to make sure FrameForge works the same everywhere. The modular design means we can switch to advanced databases if we need to. This flexibility means FrameForge will keep working even as user needs change over time. FrameForge will be able to handle users and larger projects without any issues. The FrameForge system is ready, for whatever comes..

B. Technology Stack

The FrameForge system utilizes a modern set of technologies to ensure efficient performance, scalability, and maintainability.

Frontend Technologies

- **React.js:** Used to develop a dynamic and responsive user interface that enhances user interaction.
- **Vite:** Used as a build tool to improve development speed and optimize frontend performance.

Styling

- **Tailwind CSS:** Used to design responsive layouts and maintain a consistent visual theme across the application.

Backend Technologies

- **Node.js:** Provides the runtime environment for executing server-side operations.
- **Express.js:** Manages routing, API handling, and authentication mechanisms within the system.

Database

- **SQLite:** Used as a lightweight relational database for storing structured data.
- **better-sqlite3:** Facilitates efficient interaction with the SQLite database.

Authentication

- **JSON Web Token (JWT):** Provides secure authentication and session management.

Storage

- **Amazon S3:** Used for scalable cloud storage of media files.
- **Local Storage:** Temporary storage of files in uploads and archive directories.

DevOps

- **Docker:** Enables containerized deployment of the application.
- **Docker Compose:** Manages multiple containers within the system.

Reverse Proxy and Server

- **Nginx / Traefik / Caddy:** Used as reverse proxy servers to manage traffic routing and enable SSL-based secure communication.

Tools and Configuration

- **Node Package Manager (npm):** Used for dependency management.
- **Environment Variables (.env files):** Store configuration settings and sensitive credentials securely.

__

C. Core Features

FrameForge consists of multiple integrated modules that work collaboratively to improve the efficiency of media and event management operations. These modules ensure secure handling of digital media while maintaining system reliability.

Authentication Module

The authentication module ensures that only authorized users can access the system. User identity verification is performed before granting system access. JSON Web Token (JWT) technology is used to maintain secure login sessions and protect sensitive data.

Dashboard Module

The dashboard module provides administrators with real-time system insights. It displays information such as the total number of events, uploaded photos, view counts, and download statistics. Additionally, it highlights the top five most frequently accessed photos, enabling administrators to monitor usage patterns and system performance.

Upload and Media Management Module

This module enables users to upload media files efficiently. During the upload process, the system automatically generates new events when required. This automation simplifies file organization and improves workflow efficiency.

Event Management Module

The event management module allows administrators to create, view, and delete event records. This functionality helps maintain organized data and ensures quick access to stored media.

Archive System

The archive system allows inactive or completed events to be moved into storage, reducing clutter in the active workspace. Archived events can be restored whenever necessary.

Sharing and Access Module

This module enables administrators to generate unique shareable links that allow clients to view selected media content. The link-based access mechanism enhances usability while maintaining secure data access.

Download Module

The download module allows clients to download individual images or complete event albums. This functionality ensures convenient access to stored media.

Backup and Recovery Module

This module provides automated data backup and restoration features. Regular database backups prevent data loss and improve system reliability.

Navigation and User Interface Module

The navigation system provides an intuitive interface that allows users to access system features efficiently. The responsive design ensures compatibility across different devices and screen sizes.

Storage and Deployment Module

The storage module supports both local and cloud storage solutions. Docker-based deployment ensures system portability and simplifies installation across different platforms.

—

D. Workflow

The FrameForge system operates through a structured workflow designed to streamline media management operations.

1) **Administrator Login:** The administrator logs into the system using secure JWT authentication.
2) **Media Upload:** Images are uploaded into the system, and new events are automatically created.
3) **Event Management:** Events can be created, modified, or deleted based on project requirements.
4) **Gallery Sharing:** A shareable event link is generated for client access.
5) **Client Access:** Clients use the shared link to view available photos.
6) **Download Operations:** Clients download individual images or entire albums.
7) **Analytics Tracking:** The system records engagement metrics such as views and downloads.
8) **Archiving and Backup:** Completed events can be archived and restored when needed. Database backups are performed periodically.

—

E. Security

Security plays a vital role in protecting sensitive user and media data within FrameForge.

The system implements JWT-based authentication to validate user credentials and maintain secure sessions. Access control mechanisms restrict system operations to authorized users only.

Secure API communication protects data exchanged between frontend and backend components. Reverse proxy servers such as Nginx enable SSL encryption, ensuring secure HTTPS-based communication.

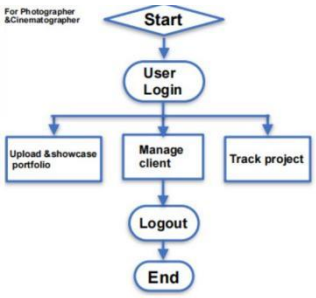

Fig. 1: Workflow

Environment variables stored in .env files protect sensitive credentials such as API keys and database configurations.

Additionally, backup and recovery mechanisms safeguard against data loss and enhance overall system reliability.

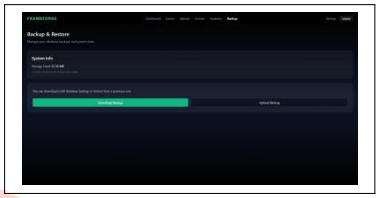

Fig. 2: Backup & Restore

F. Cloud Deployment

FrameForge supports cloud-based deployment to enhance performance, scalability, and availability.

The system integrates Amazon S3 cloud storage to store media files securely and enable fast retrieval operations. This ensures reliable storage of large media datasets.

Docker-based containerization enables consistent deployment across different cloud environments. The application can be deployed on various cloud platforms without significant configuration changes.

Reverse proxy configurations combined with SSL certificates provide secure remote access to the system. Cloud-based deployment improves system reliability and enables the platform to support multiple users simultaneously.

IV. Results and Discussion

The FrameForge system was successfully implemented and tested to evaluate its performance, efficiency, and usability. The testing process focused on analyzing system functionality, response time, and user interaction experience.

The system demonstrated reliable performance during media upload, event creation, and gallery sharing operations. The automatic event creation feature significantly reduced manual effort and simplified workflow management for administrators.

The analytics tracking module functioned effectively by recording engagement metrics such as the number of views and downloads. Additionally, the storage monitoring feature provided valuable insights into disk usage, allowing users to manage available storage efficiently.

Integration of Amazon S3 enhanced system scalability by enabling efficient storage and retrieval of large media files. The use of Docker ensured consistent system behavior across different environments, improving deployment reliability and portability.

Security testing confirmed that JWT-based authentication and SSL encryption mechanisms effectively restricted unauthorized access. The backup and recovery functionality further ensured data safety and reliability in case of unexpected failures.

User evaluation indicated that the system improved workflow efficiency compared to traditional file storage methods. The intuitive interface design and responsive layout enhanced usability and overall user satisfaction.

Overall, the FrameForge system successfully achieved its intended objectives. It simplified event-based media management and improved accessibility for both administrators and clients.

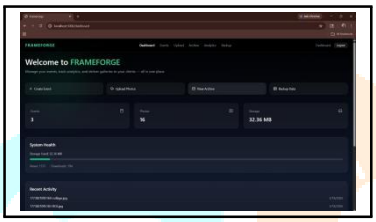

Fig. 3: Login Interface

V. Conclusion

FrameForge provides an integrated platform designed to enhance event-based media management and client interaction processes.

The system utilizes modern web technologies including React, Node.js, SQLite, and Docker to create a secure, scalable, and user-friendly environment. Cloud storage integration using Amazon S3 enables reliable handling of large media files and supports scalable storage capabilities.

Security mechanisms such as JWT-based authentication, SSL encryption, and controlled access policies ensure safe data management. The containerized deployment approach improves portability and simplifies system maintenance.

Experimental evaluation confirms that FrameForge improves workflow efficiency, reduces manual effort, and enhances accessibility for users. The centralized platform enables administrators and clients to manage media content more effectively.

Future enhancements may include artificial intelligence-based image categorization, advanced analytics features, mobile application support, and multi-user collaboration capabilities. These improvements will further expand the functionality and scalability of the FrameForge system.

Acknowledgment

The authors express sincere gratitude to the faculty of MIT ADT University for their guidance and support throughout the project.

References

[1] Flask Documentation, ”Flask Web Framework,” 2024. [Online]. Available: https://flask.palletsprojects.com/
[2] React Documentation, ”React – A JavaScript Library for Building User Interfaces,” 2024. [Online]. Available: https://react.dev/
[3] SQLite, ”SQLite: Lightweight Relational Database Engine,” 2024. [Online]. Available: https://www.sqlite.org/
[4] Amazon Web Services (AWS) Documentation, ”AWS for Web App Deployment,” 2024. [Online]. Available: https://aws.amazon.com/documentation/
[5] Mozilla Developer Network (MDN), ”Web Development Guides,” 2024. [Online]. Available: https://developer.mozilla.org/
[6] Bootstrap Documentation, ”Responsive Frontend Framework,” 2024. [Online]. Available: https://getbootstrap.com/
[7] W3C, ”Web Accessibility and Standards Guidelines (WAI),” 2024. [Online]. Available: https://www.w3.org/WAI/
[8] Python Software Foundation, ”Python Programming Language Documentation,” 2024. [Online]. Available: https://docs.python.org/
[9] DigitalOcean, ”Web Application Deployment Tutorials,” 2024. [Online]. Available: https://www.digitalocean.com/community/tutorials