Archives available at journals.mriindia.com

International Journal on Advanced Computer Engineering and Communication Technology

ISSN: 2278-5140

Volume 14 Issue 03s, 2025

AuraSnap: A Hybrid Model for Face-Based Photo Sorting and Organization

[1]Joyce M. Dsouza , [2]Parth Mhatre, [3]Pinak Meher, [4]Niharika Raut, [5]Seon D'silva
[1,2,3,4,5] Department of Computer Engineering
Vidyavardhini's College of Engineering & Technology(VCET)
Vasai, India
Email: [1]joyce.dsouza@vcet.edu.in, [2]parthmhatre2412@gmail.com, [3]pinakmeher@gmail.com, [4]niharikaraut2005@gmail.com, [5]seondsilva864@gmail.com

Peer Review Information

Submission: 05 Nov 2025

Revision: 25 Nov 2025

Acceptance: 17 Dec 2025

Keywords

PyTorch, TensorFlow, OpenCV, Customtkinter

Abstract

Aurasnap is an AI-powered photo management system that automates sorting and delivering event photos using face recognition. It detects and categorizes faces from large photo collections, saving photographers significant time and effort. The system also supports automatic watermarking to protect brand identity and integrates with PyWhatKit for direct WhatsApp photo delivery. Built with customtkinter for an easy interface and powered by OpenCV and PyTorch/TensorFlow for recognition, it offers both technical efficiency and user-friendliness. Overall, Aurasnap enhances productivity and customer satisfaction for photography studios.

Introduction

In the evolving domain of digital photography, the management and distribution of large volumes of images post-event remain a significant challenge, particularly in high-profile events such as weddings, birthdays, and corporate gatherings. Photographers often deal with thousands of images, and manually sorting these based on individual guests is a tedious and error-prone process. This not only results in inefficiencies but also delays the timely delivery of cherished memories to clients and guests.

To meet the growing demand for faster, smarter, and more personalized services, the need for an automated and intelligent solution becomes evident. Aurasnap addresses this gap by offering a streamlined photo management system powered by facial recognition technology. It eliminates manual intervention by automating the identification and categorization of individuals present in images, thus enhancing productivity and accuracy for professional photographers.

Aurasnap is developed as a desktop-based application tailored for photography studios. It integrates powerful computer vision frameworks such as OpenCV for image processing and utilizes advanced machine learning models through TensorFlow or PyTorch to ensure high-accuracy face recognition. The system also features a modern user interface built using CustomTkinter, ensuring accessibility and ease of use for users with minimal technical knowledge.

Among its key functionalities, Aurasnap includes automatic face detection, dynamic list-wise sorting of images, event-based watermarking, and automated delivery of photos through communication platforms using PyWhatKit. These features collectively minimize human effort, optimize workflow, and significantly reduce turnaround times. Ultimately, Aurasnap serves as a comprehensive, scalable, and user-centric solution that empowers photography businesses to deliver a more professional and efficient post-event experience. It not only improves operational efficiency but also enhances client satisfaction by ensuring that every guest receives their photographs accurately and promptly.

Literature survey

To understand the relevance of Aurasnap, it's essential to review existing systems that leverage AI and face recognition.

AI-powered Facial Recognition-Based Photo Retrieval System system integrates data collection, face detection, face matching, and security features to automate photo management efficiently. It uses MobileNet to balance detection speed and computational cost, along with algorithms like Euclidean Distance Finder and Bounding Box-Based Detection for accurate face matching. With an accuracy range of 90% to 99%, it delivers quick photo retrieval—averaging 2 seconds—even for large datasets. The system is scalable, reliable, and compliant with data protection standards like encryption and GDPR. Its user-friendly design ensures both strong performance and a secure, intuitive experience. The system faces some challenges despite its strong performance. Its accuracy may decrease under poor lighting, occlusions, or low-resolution images. While MobileNet offers efficiency, it may not match the precision of more advanced models like ResNet or FaceNet. Real-time scalability could require high-end hardware for large events, and managing biometric data poses privacy and legal challenges. Additionally, reliance on internet connectivity may affect performance in low-network environments[1].

Through parallelization, the study's new Java-based HDBSCAN* implementation (Tribuo Hdbscan) achieves a 25% quicker training time than the reference Java version while introducing a novel prediction mechanism for unseen data points. For streaming applications based on Java, it provides smooth integration and effective prediction. Python's hdbscan still outperforms the training performance, and more optimization is required to reach equivalent effectiveness. Furthermore, when labeling unknown data points, the new prediction method can generate approximation mistakes[2].

An improved HDBSCAN version called HDBSCAN(€) was created to better manage datasets with different densities. While effectively managing micro-clusters in dense locations, it preserves pertinent small clusters in sparse places. The method can be easily integrated with current implementations because it is completely FOSC-compliant and less sensitive to parameter choices like epsilon. HDBSCAN(€) may continue to encounter difficulties in fine-tuning parameters for highly heterogeneous datasets, and it may also increase computational burden during intricate clustering scenarios, despite its enhancements[3].

MultiFace is a new training method that improves facial recognition performance by combining various low-dimensional spaces to simulate high-dimensional areas. It accelerates training, resulting in cutting-edge performance on benchmark and large-scale datasets. In conclusion, MultiFace introduces an effective and general training mechanism that improves facial recognition performance by breaking down high-dimensional feature learning into a collection of low-dimensional subspaces. This method results in faster convergence, higher accuracy, and better interpretability across a variety of face recognition models and datasets. Managing several subspace models during training can increase overall resource usage, making the process more computationally demanding. The performance of the system is highly dependent on the selection of subspaces, which may vary across different datasets and require fine-tuning[4].

Methods to enhance semi-automatic person annotation in individual photo sets are investigated in this work. The best results are obtained when event-constrained person matching and event-based initial annotation are coupled; even with minimal initial labeling, excellent accuracy is achieved. To improve recognition, the system employs MPEG-7 and LBP descriptors in addition to face and body-patch features. Unbalanced datasets or events with few known identities may result in decreased performance. The method still necessitates initial annotation by hand, which can take a long time. Additionally, it could have trouble with different lighting or poses, and the use of several descriptors adds to the computational complexity[5].

The research investigates ways to enhance semi-automatic person annotation in individual photo collections. Even with limited first labeling, it finds that event-constrained person matching and event-based initial annotation produce the greatest results. In addition to MPEG-7 and LBP descriptors, the system employs face and body-patch features to improve recognition. Events with limited known identities or imbalanced datasets may result in decreased performance. Initial annotation is still done by hand with this method, which can take a long time. It might also have trouble with different lighting or poses, and using more than one description makes it more computationally complex[6].

A masked face recognition system that was trained using pictures of both masked and unmasked faces from a specially created dataset called MaFaR. Using deep learning models using the ArcFace loss function, such as MobileNetV2, DenseNet201, VGG16, and ResNet50V2, it improves accuracy through soft voting ensemble learning. In terms of identifying masked faces, the method outperformed individual models with a test accuracy of 93.65%.

Training many models raises the computational cost and complexity of the system, and it may have trouble with different types of masks or partial occlusions. The 75-person dataset size restricts generalization to broader populations, and camera or lighting conditions may affect performance[7].

In order to make managing big photo collections easier, a semi-automatic photo management system that combines precise face detection with user-assisted tagging. It reduces human labor by using a face recognizer to recommend comparable faces and a face-view interface for speedy labeling. The system efficiently tags and arranges people in photographs by striking a balance between automation and user input. For highly big datasets, the system's reliance on manual labeling can be time-consuming. Lighting and position changes limit its recognition accuracy, and even with bigger training sets, the model's precision plateaus. Furthermore, it can struggle with faces that look alike and need user experience to handle labeling effectively[8].

Nvidia's Omniverse, a potent 3D collaboration and simulation platform based on the USD architecture, is used in robotics, XR, digital twins, AI training, and other fields, it incorporates programs like Replicator, Isaac Sim, DriveSim, Audio2Face, and USD Composer. The platform facilitates realistic simulations, real-time collaboration, and the creation of synthetic data across a variety of sectors, such as smart cities, healthcare, and education. Omniverse restricts accessibility by requiring specialized technology and strong internet connectivity. It has issues with data management and cross-system and cross-application compatibility. Collaborative spaces can raise privacy and security issues, and their high cost may prevent widespread adoption[9].

In order to assist users locate their images quickly and safely, an AI-powered photo retrieval system for real-time event photography that makes use of FaceNet facial recognition, QR-based authentication, and cloud connectivity. Using FastAPI, React.js, and AWS cloud storage, the solution maintains scalability and efficiency while enhancing user experience through automation, privacy restrictions, and seamless access. In group or dimly lit photos, the system's accuracy decreases, resulting in uneven recognition. High processing power is necessary for real-time processing, and dependable internet access is necessary for cloud operations. Despite encryption, privacy issues still exist, and if used improperly, QR-based access could be dangerous. Furthermore, customization may be required for integration with other event configurations, which would increase the complexity of deployment[10].

Existing AI-powered photo tools face several key limitations that highlight the need for an advanced system like Aurasnap. Most current platforms, such as Google Photos or Apple Photos, still require manual confirmation of facial matches, making the process time-consuming and unsuitable for large-scale events. They also lack built-in mechanisms for automated photo delivery through messaging platforms like WhatsApp, forcing users to rely on manual sharing or external scripts.

Additionally, these tools do not support dynamic watermarking that can adapt based on guest identity or event type, limiting personalization and branding opportunities for photography studios. Heavy cloud dependency further restricts usability in areas with poor internet connectivity, as offline support is rarely available. Moreover, there are no dedicated desktop applications tailored for studio workflows that combine batch face recognition, real-time processing, watermark automation, and direct image delivery. As a result, studios often rely on fragmented toolchains for editing, recognition, and communication, leading to inefficiencies and a disjointed workflow.

Proposed System

Aurasnap is a stand-alone desktop program created in Python with the goal of automating the entire event photography management process. It identifies people in a set of event photos using facial recognition techniques, then uses minimum human intervention to classify and arrange the photos into person-specific directories.

For precise facial recognition, feature extraction, and identification matching, the program combines deep learning and computer vision frameworks. The technology automates the entire image categorization process by assigning relevant photos to dynamically generated lists once faces are recognized. Additionally, Aurasnap provides the ability to add editable watermarks to photos, making it simple for photo studios to add branding elements or event-specific information. The system's ability to use PyWhatKit to automatically send the processed photographs over WhatsApp is one of its main features[11].

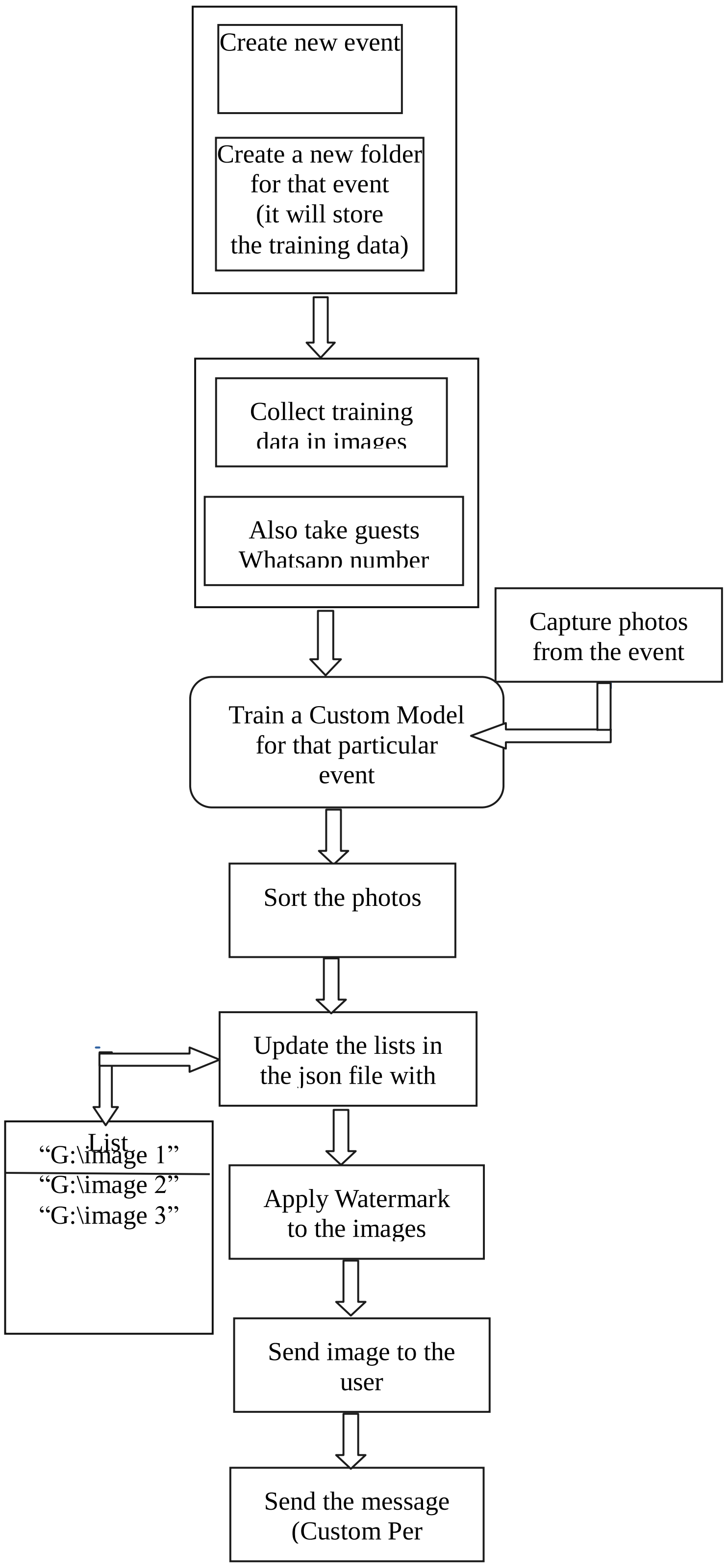

Create new event
Create a new folder for that event (it will store the training data)
Collect training data in images
Also take guests Whatsapp number
Capture photos from the event
Train a Custom Model for that particular event
Sort the photos
Update the lists in the json file with
List
"G:\image 1"
"G:\image 2"
"G:\image 3"
Apply Watermark to the images
Send image to the user
Send the message (Custom Per

Fig. 1: Proposed System Workflow

This streamlines the process of distributing photos to specific visitors straight from the system, doing away with the necessity for manual file transfers. A specified list of contacts can be used to initiate the distribution procedure in batch mode. Because Aurasnap is made to run offline, it can function fully even in places with poor connectivity, such on-site event venues. The application also offers optional real-time facial recognition via webcam connection for enhanced usefulness.

Customtkinter was used in the development of the graphical user interface, which provides both technical and non-technical users with a clear, responsive, and easy-to-use interface[12]. Overall, Aurasnap provides a cohesive solution that unifies face recognition, photo management, watermarking, and automated delivery into a single, efficient desktop application tailored for professional photography studios.

Using raw event photos, Aurasnap is a modular image-processing pipeline that organizes and classifies images, detects and recognizes faces, adds watermarks if desired, and sends the photos to guests (via WhatsApp automation). To enable independent testing, scaling, and component swapping, the pipeline is divided into several phases. Let us see these phases in detail.

A. Image Input

The system scans each image and gets it ready for processing in the Image Input stage. It does this by accessing a designated folder that contains raw event photos. It lists picture files and filters them by format using functions like os.walk[13]. Images are either streamed separately or processed in batches to maximize performance, particularly for huge events. Additionally, the system uses EXIF data to standardize image size and orientation. To speed up face detection while retaining image quality for later stages, it is important to use smaller thumbnails, bypass corrupted data, and preserve proper orientation.

B. Face Detection

Each image is examined to find faces in the Face Detection stage, after which the system produces bounding boxes that show where the faces are located. The CV2 feature of OpenCV can be used to accomplish this.For quicker, CPU-based detection, use CascadeClassifier; for more accuracy in a variety of scenarios, use more sophisticated deep learning models like MTCNN, RetinaFace, or YOLO[12]. To eliminate overlapping detections, a postprocessing step like non-max suppression may be used. While DNN-based detectors offer greater precision but demand more processing resources, Haar cascades offer speed but lower accuracy. Reliable results need tracking of detected face coordinates inside each image and proper parameter tweaking.

C. Face Embedding

The Face Embedding phase creates a numerical representation known as an embedding by cropping, preprocessing, and running each identified face through a deep learning model that has already been trained, such as ArcFace, FaceNet, or VGGFace2. Resizing the face image, converting it to RGB, leveling pixel values, and aligning facial landmarks for a consistent stance are common preprocessing steps. The resultant embedding, which is typically 128 or 512 dimensions, is a unique representation of the individual's facial features. Accuracy is increased by proper alignment, model selection should balance resource constraints and performance, and embedding are typically normalized (e.g., using L2 normalization) prior to comparison[11].

D. Face Matching

The Face Matching stage identifies known persons by comparing each face embedding with embeddings from a labeled reference dataset. This is accomplished using similarity metrics like Euclidean distance or cosine similarity, and the existence of a match is determined by a predetermined threshold. The best or averaged score is utilized if more than one embedding represents the same person; otherwise, faces that don't match are labeled as "unknown." To balance false positives and negatives, it is essential to select a suitable threshold[10]. Fast search methods like FAISS or Annoy enhance performance for big datasets. All comparisons must adhere to privacy and consent policies, and ambiguous matches should be manually confirmed.

E. List Sorting

Within a JSON file that serves as the main manifest, the system records the paths of matching photographs into relevant categories—such as individual visitors, unidentified faces, or group photos—in order to arrange them in the List Sorting step. Additional metadata, such as timestamps, watermark status, similarity scores, and bounding box coordinates, may be included with each entry. Consistent identities (e.g., guest ID or phone number) should be used, but sensitive data must be protected or avoided. When multiple processes change the file at the same time, atomic writes or a lightweight database (like SQLite) should be used to avoid data conflicts[9]. The JSON structure should also be flexible enough to accommodate additional fields, such as delivery status or watermark version.

F. Watermarking

In the Watermarking stage, images are branded or personalized by adding a text or image watermark, such as the name of the event, the photographer's credit, or the name of a guest. This is usually accomplished with the Pillow (PIL) library, which uses alpha blending to overlay text or a semi-transparent logo onto the image before saving it as a new file. The layout, font, size, opacity, and position of the watermark can all be altered by users[8]. Original photos should be preserved, and copies should be the only ones with watermarks. When adding guest names, text handling must take length and special characters into consideration by utilizing the proper typefaces and encoding. The watermark should also scale appropriately across various image resolutions.

G. Image delivery (PyWhatKit automation)

The technology automatically delivers visitors categorized and watermarked photos through WhatsApp during the Image Delivery stage. The JSON manifest is used to get guest phone numbers, while PyWhatKit and other libraries are used to automate the delivery of images and messages via WhatsApp Web. After the user scans a QR code once, this procedure usually entails preparing the message content, launching WhatsApp Web in a browser session, and transmitting files[7]. Third-party messaging platforms like the WhatsApp Business API provide a more dependable and scalable option for production or large-scale settings. Managing rate restrictions using batching and retries, protecting guest privacy and consent, and recording delivery status (success or failure) in the JSON manifest for future use are important factors to take into account.

Result analysis

This section describes the result obtained by the Aurasnap System Workflow.

Figure 2 shows the dashboard or home screen that provides quick navigation to the major functionalities of the Aura Snap application.

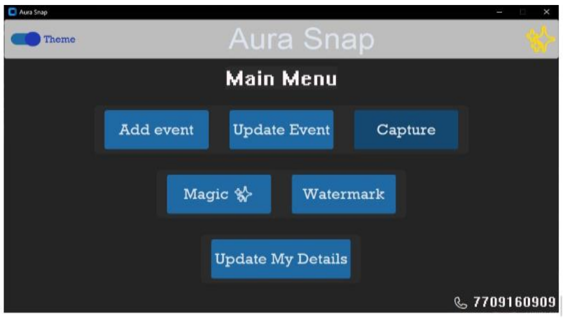


Fig. 2: Aura Snap Home Screen

The Aura Snap interface is modern, intuitive, and user-friendly, designed for both efficiency and visual appeal. A theme toggle at the top left lets users switch between light and dark modes, while the title "Aura Snap" and heading "Main Menu" clearly define the workspace. Functional buttons are neatly arranged — Add Event creates new events, Update Event edits existing ones, Capture handles photo input, Magic triggers automated features like sorting or face recognition, Watermark adds text or logo overlays, and Update My Details manages user information. A contact section with a phone icon and support number (7709160909) provides easy assistance. The consistent blue buttons, minimalist layout, and dark theme give the interface a clean, professional look that enhances usability.

Figure 3 "Add New Event" screen allows users to create and save new event entries that will be used for organizing and tagging captured or uploaded photos.

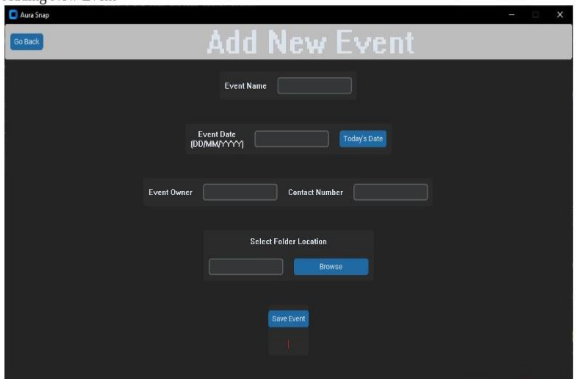


Fig. 3: Add New Event Screen

The event creation interface in Aura Snap is simple and efficient, designed for a smooth user workflow. A "Go Back" button at the top left returns users to the main menu, while the form includes key fields such as Event Name, Event Date (with a "Today's Date" shortcut), Event Owner, and Contact Number. Users can also select a folder location to store all event photos, ensuring organized image management.

After entering details, the "Save Event" button records the information—likely in a JSON file or database—for later use in sorting, watermarking, or delivery. The interface follows a dark theme with light text, blue buttons, and a clean, minimal layout consistent with the main menu's design.

Figure 4 shows the Update Events screen in Aura Snap which allows users to manage and modify event details efficiently.

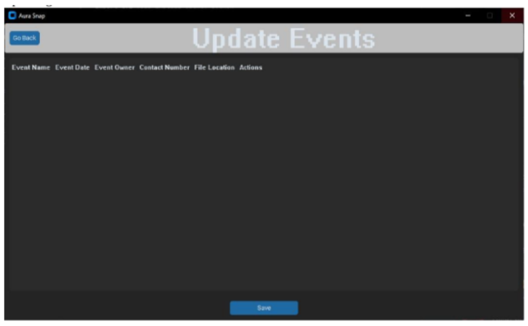


Fig. 4: Update Events Screen

At the top left, a “Go Back” button enables smooth navigation to the main menu without losing progress. The “Update Events” header clearly indicates that this section is meant for event management. Below the header, an event table displays all existing events in a structured format, including columns for event name, date, owner, contact number, file location, and actions. Users can directly edit details within the table, update folder paths using the “Browse” option, and have these changes automatically saved to the system’s JSON file or database.

At the bottom, a “Save” button commits all updates, ensuring other modules like Capture, Magic, and Watermark use the latest data. The screen follows a dark theme with a simple, user-friendly layout that supports quick editing and efficient workflow. This feature ensures event information stays accurate and synchronized across all stages of the Aura Snap system, preventing errors in image storage or delivery.

Figure 5 shows the Capture screen in Aura Snap that allows users to link guest photos with specific events efficiently. It includes a “Go Back” button for easy navigation and a clear “Capture” header indicating the active section.

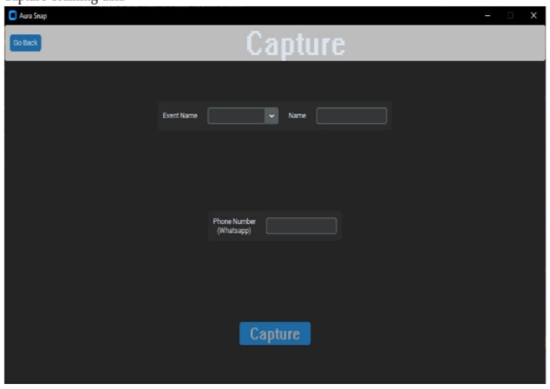


Fig. 5: Capture Training Data Screen

The Event Name dropdown lists previously created events, ensuring each photo is correctly associated. Users enter the guest’s Name and WhatsApp Number, which are stored in the system for later use during automated delivery. The “Capture” button starts the image capture process, either via webcam or by selecting an existing file, saving all details as metadata for further processing like face recognition and watermarking.

Designed with a dark theme, blue buttons, and simple layout, the interface ensures quick and error-free data entry during events. Within the overall workflow, this screen connects the event creation phase to later automation steps — detection, matching, watermarking, and delivery — enabling seamless organization and personalized photo distribution to guests.

Figure 6 shows the Magic screen in Aura Snap which serves as the control center for automated photo processing, where the system performs tasks like face detection, recognition, and image sorting.

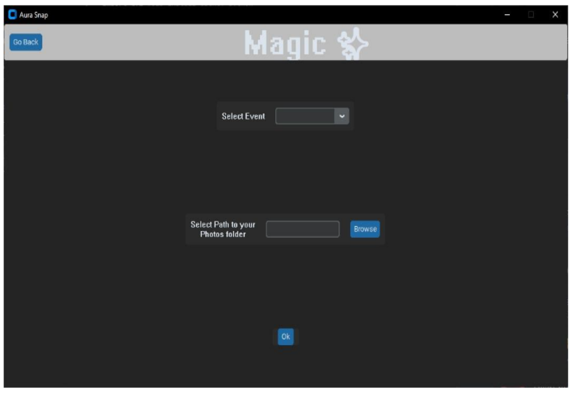


Fig. 6: Automation Screen

A “Go Back” button allows easy navigation to the main menu, while the “Magic” header highlights its automation purpose. Users can select an event from a dropdown list and specify the folder path containing raw photos using a text field or the “Browse” button. Clicking “Ok” starts the automation, which scans images, identifies faces, and organizes photos per guest, updating the event’s JSON manifest accordingly.

The interface follows Aura Snap’s dark theme with light text and blue buttons, maintaining a simple, focused layout. Overall, the Magic screen streamlines the workflow by turning unprocessed event images into neatly sorted, guest-tagged collections, ready for watermarking or delivery.

Figure 7 shows the Add Watermark screen in Aura Snap that allows users to customize and apply watermarks to their event photos for branding or personalization.

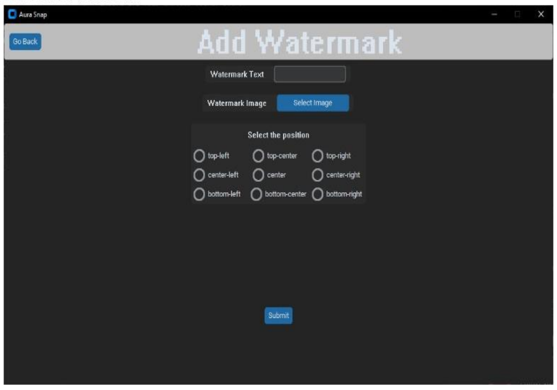
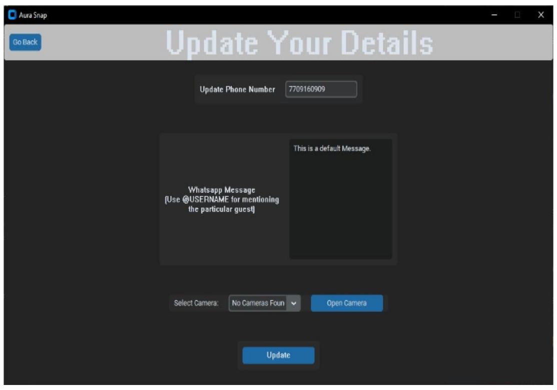


Fig. 7: Add Watermark Screen

A "Go Back" button at the top left provides easy navigation to the main menu. Users can add a Watermark Text or upload a Watermark Image using the "Select Image" button. The interface also includes multiple radio-button options under "Select the position", allowing placement of the watermark at various positions such as top-left, center, or bottom-right. Once the desired settings are chosen, the "Submit" button applies the watermark to the selected photos. The layout follows Aura Snap's consistent dark theme with blue buttons and a clean, minimal design focused on usability. Overall, this screen offers a straightforward way to add text or image watermarks to photos, ensuring professional presentation and protection of event images.

Figure 8 shows the Update Your Details screen in Aura Snap which allows users to manage and personalize their account and communication settings.



Update Phone Number 7709160909

Whatsapp Message
[Use @USERNAME for mentioning the particular guest]

This is a default Message.

Select Camera: No Cameras Foun

Fig. 8: Update Your Details Screen

A "Go Back" button at the top left lets users return to the main menu easily. The interface includes an Update Phone Number field where users can modify their registered contact number, ensuring that WhatsApp automation and notifications work correctly. A section labeled WhatsApp Message allows customization of the default message sent to guests, with a note suggesting the use of @USERNAME to personalize messages.

The interface also features camera configuration options — users can select a device from the "Select Camera" dropdown and activate it using the "Open Camera" button. Finally, the "Update" button saves all modifications. Maintaining Aura Snap's dark-themed, minimalist design, this screen helps ensure accurate contact details, personalized guest communication, and correct hardware setup for capturing or sending event photos.

Conclusion

AuraSnap effectively overcomes the major challenges photographers face in post-event workflows, such as time-consuming image sorting, manual tagging, and individual photo delivery. By integrating face recognition, automated watermarking, and WhatsApp-based delivery, the system provides a seamless, end-to-end solution that minimizes human effort while maximizing efficiency and accuracy.

This automation not only accelerates the overall process but also ensures that clients receive their photos quickly and with professional branding, thereby enhancing customer satisfaction. Furthermore, the project demonstrates that with optimized algorithms and thoughtful system design, machine learning and automation can deliver powerful, real-world benefits even on modest hardware setups, making AuraSnap a practical and scalable solution for modern photography professionals.

References

Vishwanath M. S., Ravina G., Saumya G., D. Linett Sophia, "AI-Driven Facial Recognition-Based Photo Retrieval System for Event Management," International Journal Of Scientific Research and Technology, vol. 12, pp. 53–57, April 2024.

Geoffrey Stewart and Mahmood Al-Khassaweneh, "An Implementation of the HDBSCAN* Clustering Algorithm," International Journal of Applied Sciences,vol. 12, February 25, 2022.

Claudia Malzer and Marcus Baum, "A Hybrid Approach To Hierarchical Density-based Cluster Selection", International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 223-228, 2020.

Jing Xu, Tszhang Guo, Yong Xu, Zenglin Xu, Kun Bai, "MultiFace: A generic training mechanism for boosting face recognition performance", Neurocomputing, pp. 40–47, 2021.

Rosa Andrie Asmara , Brian Sayudha , Mustika Mentari, Rizky Putra Pradana Bu-diman, Anik Nur

Handayani, Muhammad Ridwan and Putra Prima Arhandi7, "Face Recognition Using ArcFace and FaceNet in Google Cloud Platform For Attendance System Mobile Application," Annual Technology, Applied Science and Engineering Conference (ATASEC), pp.134-144, 2022.

Saman H. Cooray and Noel E. O'Connor, "Enhancing Person Annotation for Personal Photo Management Applications," 20th International Workshop on Database and Expert Systems Application, 2009, pp 251-257.

Arun Kumar R, V Anoosh Solayappan, Sree Sharmila T, Ram Prasad K, "Masked Deep Face Recognition using ArcFace and Ensemble Learning," 2021 IEEE 2nd International Conference on Technology, Engineering, Management for Societal impact using Marketing, Entrepreneurship and Talent (TEMSMET).

Bongwon Suh *, Benjamin B. Bederson, "Semi-automatic photo annotation strategies using event based clustering and clothing based person recognition, Interacting with Computers 19 (2007) 524–544.

Andreas Girgensohn, John Adcock, and Lynn Wilcox, "Leveraging Face Recognition Technology to Find and Organize Photos, 6th ACM SIGMM International workshop on Multimedis Information Retrieval, pp. 99-106, 2004.

Naveed Ahmed, Imad Afyouni , Hamzah Dabool, Zaher Al Aghbari "A systemic survey of the Omniverse platform and its applications in data generation, simulation and metaverse", Frontiers in Computer Science, vol. 6, 2024.

Kritika Patidar, Kaushal Rathore, Divya Kumawat, Mandakini Ingle, Gajendra Singh Rajput, "Photo Retrieval System based on Face Recognition with Cloud Integration", Vol. 16, Issue 2, April-June 2025.

Anushka Ajay Kamble, Triveni Tanaji Suryawanshi, Rutuja Shrihari Survase, Prof. Pooja Landage, "BULK WHATSAPP MESSAGE AUTOMATION USING ROBOTIC PROCESS AUTOMATION (RPA): A SCALABLE COMMUNICATION APPROACH", International Research Journal of Modernization in Engineering Technology and Science, Vol. 07, Issue:04, April-2025.

Jainam Shastri, Prishita Sinha, Vedang Shinde, Dr. Nilakshi Jain, Swati Gajbhiye, "Automation of WhatsApp using Python Selenium", International Journal of Advance Research, Ideas and Innovations in Technology,

vol. 8, Issue 3, 2022.