Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

AquaID: An AI-Powered Web Application for Underwater Animal Species Identification Using the Gemini Vision-Language Model

Kirankumar J [4MG23CS028]
Dept. of Computer Science and Engineering
G Madegowda Institute of Technology
Mandya, India

Uday M G [4MG23CS059]
Dept. of Computer Science and Engineering
G Madegowda Institute of Technology
Mandya, India

Harish H C [4MG23CS018]
Dept. of Computer Science and Engineering
G Madegowda Institute of Technology
Mandya, India

Under the guidance of Prof. Anusha P Swamy

Abstract— AquaID is a full-stack web application that identifies underwater and aquatic animals from a user-uploaded photograph or short video clip and returns a structured species profile — common name, scientific name, full taxonomic classification, habitat, diet, typical size, lifespan, conservation status, danger-to-humans notes, fun facts, an honestly reported confidence score, and visually similar "look-alike" species where relevant. Unlike conventional underwater species-recognition systems that train or fine-tune a dedicated convolutional or transformer-based classifier on a curated, labelled underwater image dataset, AquaID performs no model training and uses no training dataset of its own: identification is delegated entirely to Google's Gemini 2.5 Pro multimodal vision-language model through a carefully engineered, schema-constrained prompt, accessed over the Gemini API. The application additionally supports optional free-text context from the user (e.g. location, approximate size, behaviour observed), video understanding via Gemini's File API, a conversational follow-up assistant grounded in the specific uploaded media and its identification result, automatic detection and graceful rejection of non-aquatic-animal uploads, and a conservation-reporting module that lists real, verifiable government and international wildlife bodies (e.g. IUCN, CITES, national wildlife authorities) relevant to the identified species together with a live map search for the nearest office. User authentication is handled with hashed credentials and session-protected routes. This paper describes the system's architecture, the prompt-engineering methodology used in place of model training, and the design choices made to keep the system both training-data-free and resistant to confident misidentification.

Keywords— *Underwater species identification; Vision-language models; Gemini API; Multimodal large language models; Zero-shot recognition; Flask web application; Prompt engineering; Marine conservation; No-training-data system; Open-set detection*

I. INTRODUCTION

The world's oceans are estimated to harbour between 700,000 and one million distinct animal species, yet fewer than one-third have been formally described by the scientific community [1]. This knowledge gap has significant consequences for marine conservation policy, fishery management, environmental impact assessment, and ecological research. Historically, underwater species documentation has relied on expert SCUBA divers, hand-drawn taxonomic sketches, and physical specimen collection — methods that are expensive, slow, often destructive to fragile reef ecosystems, and limited in geographic reach.

The proliferation of consumer-grade underwater cameras, smartphones in waterproof housings, and citizen-science platforms now generates millions of underwater photographs and video clips. Extracting structured biological knowledge from this visual data requires identification tools that ordinary, non-expert users can use in real time [2].

Underwater imagery also presents challenges distinct from terrestrial photography: wavelength-dependent light absorption causes severe colour distortion, scattering from suspended particles reduces contrast, and subjects are frequently partially occluded by coral or kelp [3]. Traditional deep-learning approaches to this problem — convolutional networks fine-tuned on labelled fish or marine-animal datasets — have shown strong results [4]–[6], but every such system requires a dataset: images collected, cleaned, and labelled for the specific set of species the model is meant to recognise, and re-collected whenever the species list changes. Building and maintaining such a dataset is itself a significant project.

In parallel, general-purpose multimodal foundation models — large vision-language models trained by their providers on web-scale image, video, and text corpora — have become capable enough to perform open-ended visual recognition and reasoning tasks without any task-specific training at all, given only a well-constructed prompt. This paper presents AquaID, a web application that takes this approach to its logical conclusion for the underwater species identification problem: rather than training, fine-tuning, or hosting any computer-vision model, AquaID sends the user's photo or video directly to Google's hosted Gemini 2.5 Pro model with a structured identification prompt, and uses the model's reasoning and prior knowledge — not a locally trained classifier — to produce the result. No underwater image dataset was collected, labelled, or used to train anything in this project.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

The primary contributions of this work are summarised as follows:

- A full-stack identification web application that performs no local model training and uses no curated training dataset, delegating all visual recognition to a third-party, pretrained multimodal foundation model (Gemini 2.5 Pro) accessed via API.
- A schema-constrained identification prompt that forces the model to return a single structured JSON species profile — including an honestly calibrated confidence score and a list of visually similar "alternative match" species — instead of free-text or over-confident guesses.
- Native support for both still images and short video clips, the latter processed through Gemini's File API for genuine temporal/video understanding rather than independent frame-by-frame classification.
- An optional free-text context field that lets the user supply observational notes (location, approximate size, behaviour) which the model is instructed to treat as supporting evidence.
- Model-driven rejection of non-aquatic-animal uploads (an open-set style safeguard implemented through prompt instructions rather than a separate statistical anomaly detector).
- A conservation-reporting module that lists only real, verifiable government/international wildlife bodies relevant to the identified species' conservation status, paired with a live, browser-geolocation-driven map search for a nearby office — with explicit instructions to the model never to fabricate addresses or phone numbers.
- A conversational follow-up assistant grounded in the specific uploaded media and identification result, and a secure, session-based authentication layer (hashed passwords, protected routes) built with Flask and SQLite.

The remainder of this paper is organised as follows. Section II reviews related work. Section III describes the system architecture. Section IV details the prompt-engineering methodology used in place of a trained model. Section V discusses evaluation and current limitations. Section VI concludes with directions for future work.

II. LITERATURE REVIEW

Early work on automated fish and marine species recognition relied on hand-crafted visual descriptors. Boom et al. [4] developed a long-term monitoring system for coral reef fish assemblages using HOG features and SVM classifiers on the Fish4Knowledge dataset. While pioneering, such systems struggled in open-water conditions with complex backgrounds and variable illumination.

The emergence of deep CNNs transformed marine computer vision. Qin et al. [5] demonstrated that fine-tuned VGG-16 achieved strong accuracy on multi-species fish datasets, establishing transfer learning from ImageNet as a dominant paradigm. Rauf et al. [6] subsequently showed that a progressively fine-tuned ResNet-50 improved on this benchmark while remaining computationally efficient. However, both studies, like most supervised approaches, operated on a fixed, curated set of species and required a labelled training dataset specific to that set.

Real-time detection requirements motivated the adoption of YOLO-family object detectors. Jäger et al. [7] applied YOLOv4 to video streams from ROVs, achieving high detection precision at interactive frame rates. YOLO-based systems, however, still require annotated bounding-box training data and tend to generalise poorly to species outside their training distribution, often producing high-confidence incorrect predictions rather than acknowledging uncertainty.

Vision-language foundation models have begun reshaping the field. Radford et al. [8] introduced CLIP, which learns aligned image and text embeddings from 400 million web image–caption pairs, enabling zero-shot classification by comparing an image embedding against text prompts rather than training a closed-set classifier. Oquab et al. [9] released DINOv2, a self-supervised Vision Transformer producing rich features without any labelled data. Both works demonstrated that large-scale pretraining can substitute for task-specific labelled datasets — a direction AquaID builds on directly, though at the level of a full general-purpose multimodal model rather than an embedding model that still requires an application-level classifier or reference-embedding registry to be built around it.

Open-set recognition — the ability to detect and gracefully handle inputs that do not belong to any class the system is meant to recognise — has received growing attention in deep learning [10]. Traditional open-set pipelines add a dedicated statistical anomaly-detection component (e.g. an isolation forest or a distance threshold over an embedding space) alongside the main classifier. In this work, that function is instead delegated to the foundation model itself through explicit prompt instructions, as described in Section III.

Most recently, large multimodal models offered as hosted APIs — such as Google's Gemini family [11] and OpenAI's GPT-4 with vision [12] — combine strong visual recognition with open-ended natural-language reasoning and generation in a single model, trained by their provider on web-scale data the application developer never has to collect. AquaID is, to the authors' knowledge, an application of this newest generation of models to the specific problem of underwater species identification: rather than training a bespoke classifier (as in [4]–[6]) or building an application-level zero-shot pipeline around embedding models (as enabled by [8], [9]), AquaID treats the hosted multimodal model as the entire recognition-plus-reasoning engine, and focuses engineering effort on prompt design, structured-output enforcement, and the surrounding application (authentication, media handling, conversational grounding, and responsible conservation signposting).

III. SYSTEM ARCHITECTURE

AquaID follows a three-tier architecture: (1) a browser-based frontend, (2) a Flask application server that handles authentication, media upload, and API orchestration, and (3) Google's hosted Gemini API, which performs all visual identification and conversational reasoning. No computer-vision model is trained, fine-tuned, or hosted by the application itself.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

A. Frontend

The client-facing interface is built with HTML5, CSS3, and vanilla JavaScript, rendered through Flask's Jinja templates. It consists of an animated landing page, login and signup pages, and a main "identify" page offering drag-and-drop upload of a still image (JPG, PNG, WEBP, BMP, HEIC) or a short video clip (MP4, MOV, WEBM, AVI, MKV, M4V, up to 50 MB), an optional free-text context box, and a chat panel that becomes active once an identification result is returned.

B. Authentication Layer

User accounts are stored in a local SQLite database (instance/users.db), managed through a dedicated Flask blueprint (auth.py). Passwords are never stored in plain text: they are hashed using Werkzeug's PBKDF2-based password hashing before being written to the database. All identification, chat, and uploaded-media routes are protected by a login_required decorator and a server-side session cookie signed with a configurable secret key.

C. Identification Pipeline (/api/identify)

When a user submits media, the Flask backend saves it to a local uploads/ directory under a randomly generated filename, determines its MIME type, and prepares it for the Gemini API: an image is sent inline as raw bytes in the same request as the prompt, while a video is first uploaded through Gemini's File API and polled (up to 60 attempts, 2 seconds apart) until it reaches an ACTIVE processing state, after which it can be referenced in a generation request. This allows Gemini to reason over the video's actual temporal content — motion, behaviour, multiple viewing angles — rather than the application extracting and classifying individual frames itself.

The backend then constructs a single structured prompt (Section IV) and calls the Gemini 2.5 Pro model (configurable via an environment variable) with generation parameters set to a low temperature of 0.2 — to favour consistent, evidence-grounded answers over creative variation — and response_mime_type set to application/json. The returned text is parsed as JSON; a defensive fallback strips markdown code fences and extracts the content between the first "{" and the last "}" in case the model does not follow the no-code-fence instruction exactly.

D. Structured Species Profile

The parsed JSON result — including common name, scientific name, full taxonomic classification (kingdom through family), confidence percentage, habitat, diet, average size, lifespan, conservation status, danger notes, fun facts, the specific visual evidence that led to the identification, alternative look-alike species, and conservation-reporting fields — is returned to the frontend and rendered as an animated species-profile card. No part of this profile is looked up from a locally maintained species registry or external biology database API; the entire profile, including the factual species information itself, is generated by the Gemini model directly in response to the prompt, from its own pretrained knowledge.

E. Conservation-Reporting Module

For each identification, the model is asked to list two to four real, well-known government or international wildlife bodies relevant to the species (for example the CITES Secretariat, the IUCN Species Survival Commission, or a national wildlife/forest/fisheries authority), with an explicit instruction never to invent an organisation name, address, phone number, or email, and to omit an entry rather than guess if it is unsure a body is real. The frontend separately offers a "find an office near me" button that uses the browser's Geolocation API to open a live Google Maps search for the nearest relevant office; this location is used only to build the map-search URL and is never sent to Gemini or stored by the application.

F. Conversational Follow-Up Assistant

After an identification is produced, the backend starts a Gemini chat session seeded with the same uploaded media and the identification JSON as a system instruction, so that follow-up questions ("how big does it get?", "is it dangerous?") are answered in the context of that specific photo or video rather than generically. Chat sessions are currently kept in an in-memory dictionary keyed by the uploaded media's filename, which is adequate for a single-process development deployment but would need to be externalised (e.g. to Redis or a database) for a multi-worker production deployment.

IV. METHODOLOGY — PROMPT ENGINEERING IN PLACE OF MODEL TRAINING

A. Why No Training Dataset Was Used

AquaID does not collect, label, or train on any underwater image dataset, and does not fine-tune the Gemini model. Identification is performed zero-shot, purely through prompting: Gemini 2.5 Pro is a general-purpose multimodal model already pretrained by Google on a web-scale corpus that includes a very large amount of visual and textual material covering biological taxa, so the application-level engineering effort goes entirely into instructing the model correctly and structuring its output, rather than into collecting data or designing a network architecture. This is a direct extension of the zero-shot philosophy demonstrated by CLIP and DINOv2 [8], [9], but carried out with a single hosted general-purpose model that also produces the natural-language explanation, the structured metadata, and the conversational follow-up — tasks that would otherwise require separate components.

B. The Identification Prompt

The core design artifact of AquaID is a single, carefully engineered prompt template (sent to the model together with the image or video) rather than a trained model. It instructs Gemini to act as a meticulous marine biologist, to examine body shape, fin or limb placement, colour pattern, markings, eye position, skin/scale/shell texture, size cues, swimming or movement behaviour (for video), and background cues such as coral, substrate, depth, and water clarity before answering, and to reason step by step internally before producing only the final structured answer.

C. Schema-Constrained Structured Output

The prompt specifies an exact JSON schema the response must follow: an is_underwater_animal boolean, common and scientific name, a nested classification object (kingdom, phylum, class, order, family), a confidence_percent number,

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

habitat, diet, average_size, lifespan, conservation_status, a danger note, a list of fun facts, a distinguishing_features explanation tied to the specific media, an alternative_matches array, a government_bodies array with name/scope/role fields, a report_if_endangered note, and a free-text notes field for caveats. Combined with the API's response_mime_type=application/json setting and the backend's defensive parser, this keeps the output machine-readable and the frontend rendering deterministic, without the application needing to train a model to produce this structure itself.

D. Anti-Hallucination / Confidence Rules

A dedicated "accuracy rules" block in the prompt instructs the model never to state a species as certain unless the visible evidence truly supports it, to prefer a correct higher-level (e.g. genus or family) answer with honestly lower confidence over a wrong species-level guess at high confidence, and — when two or more species are visually near-identical — to say so in alternative_matches and lower confidence_percent accordingly rather than arbitrarily picking one. Confidence is explicitly tied to evidence quality: sharpness, how much of the animal is visible, occlusion, lighting, and how distinctive the visible features are.

E. Open-Set Rejection via Prompting

Rather than a separate statistical novelty/anomaly detector, the prompt instructs the model to set is_underwater_animal to false — and to fill the response with an explanation of what the media actually shows instead — whenever the upload is not an aquatic animal (e.g. a person, a land animal, an object, or unrelated scenery). This plays the functional role an explicit open-set detector would play in a traditional closed-set CNN pipeline, implemented instead as a reasoning instruction inside the foundation model.

F. Context-Conditioned Identification

If the user supplies free-text notes (for example, "found near a coral reef in the Maldives, about 40 cm long"), this text is inserted into the prompt as an explicit "additional context provided by the user" block, and the model is instructed to treat it as strong supporting evidence that can help resolve cases where the visual evidence alone is ambiguous.

G. Video Understanding via File API

For video uploads, the file is handed to Gemini's File API rather than being decomposed into frames by the application. The backend polls the returned file resource until its state transitions to ACTIVE before issuing the generation request, after which the model reasons directly over the video's visual and temporal content (motion, multiple angles, behaviour) as part of producing the same structured JSON profile used for images.

H. Security and Configuration

Passwords are hashed with Werkzeug's PBKDF2 implementation before storage; session cookies are signed with a configurable Flask secret key; all identification, chat, and media-serving routes require an authenticated session; uploaded-file size is capped at 50 MB and restricted to an explicit file-extension allow-list; and all runtime configuration (the Gemini API key, model names, the session secret, and a fallback country used for the conservation module) is read from environment variables via a .env file, kept out of version control.

V. EVALUATION AND DISCUSSION

Because AquaID performs no local model training, it was not evaluated against a held-out labelled test set of underwater images in the way a trained classifier conventionally would be — no such dataset was built or used, consistent with the project's training-data-free design. Evaluation to date has instead been functional and qualitative.

A. Functional Verification

The end-to-end flow — account signup and login, image upload and identification, video upload and identification through the File API, the optional context box influencing the returned profile, the grounded chatbot answering follow-up questions, the conservation-body listing, and the "find an office near me" map link — was exercised manually across a range of sample photos and clips of common aquatic species (reef fish, sea turtles, and similar) and of clearly non-aquatic subjects, and behaved as designed in each case: plausible species profiles with populated fields for genuine aquatic-animal media, and an explicit "not an underwater animal" response with an empty profile for unrelated media.

B. Qualitative Behaviour of the Confidence and Alternative-Match Design

Consistent with the prompt's accuracy rules, informal testing showed the model reporting lower confidence percentages and populating the alternative_matches field on images where two visually similar species could plausibly be confused, rather than committing to a single high-confidence label — the intended behaviour described in Section IV-D. A systematic, statistically meaningful measurement of this behaviour (e.g. calibration curves or precision/recall against ground-truth labels) was not performed, since it would require a labelled underwater benchmark dataset that this project deliberately does not build; this is identified as a direction for future work in Section VI.

C. Latency Characteristics

Because there is no local model to run, identification latency is dominated by the Gemini API's own response time for image or video reasoning, plus — for video — the time Gemini's File API takes to finish processing an uploaded clip before it becomes available for inference (observed informally to range from a few seconds for short clips up to roughly a minute for longer ones, depending on length and current API load). This differs fundamentally from the GPU/CPU inference-latency figures reported for locally hosted CNN pipelines in the literature reviewed in Section II, since no part of the identification runs on infrastructure controlled by the application.

D. Limitations

- The system depends entirely on a third-party, paid/rate-limited external API; it cannot identify anything, and the application has no fallback, if the Gemini API key is

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 10 , October - 2026

missing, the quota is exhausted, or the service is unreachable.

- No systematic accuracy evaluation against a labelled underwater-species benchmark has been carried out; reported confidence and alternative-match behaviour has only been checked qualitatively, as described above.
- Prompt-level anti-hallucination instructions reduce but cannot fully eliminate the risk of the underlying model producing an incorrect or overconfident answer, or an incorrect conservation-body entry despite the explicit "only real organisations" rule.
- Uploaded photos and videos, which may contain sensitive location or personal information, are sent to a third-party API provider as part of normal operation; this should be disclosed to end users and considered in any production deployment.
- In-memory chat-session storage, keyed per process, does not persist across server restarts and is not safe for a multi-worker production deployment without being moved to shared storage.

VI. CONCLUSION

This paper presented AquaID, a full-stack web application for underwater animal species identification that performs no local model training and uses no curated underwater-image training dataset. Instead of the conventional pipeline of a CNN or embedding model trained or fine-tuned on a labelled dataset, AquaID delegates the entire visual-recognition task to Google's Gemini 2.5 Pro multimodal model through a single, carefully engineered, schema-constrained prompt — supplemented by application-level engineering for secure authentication, image and video upload handling (including genuine video understanding via Gemini's File API), structured-output parsing, a conversation-grounded follow-up assistant, and a conservation-reporting module that surfaces only real, verifiable wildlife authorities rather than fabricated contact details.

The system demonstrates that recent general-purpose multimodal foundation models can substitute for an entire bespoke computer-vision pipeline in a practical identification task, shifting the engineering effort from dataset collection and model architecture design toward prompt design, structured-output enforcement, and responsible-use safeguards. Future work includes systematic quantitative evaluation against a labelled benchmark test set (precision/recall and confidence-calibration analysis), a cached or locally hosted fallback for when the external API is unavailable, externalised (database-backed) chat-session storage suitable for multi-worker deployment, and explicit user-facing disclosure and consent controls around sending uploaded media to a third-party API.

ACKNOWLEDGMENT

The authors gratefully acknowledge the guidance and continuous support of Prof. Anusha P Swamy, Department of Computer Science and Engineering, G. Madegowda Institute of Technology, Mandya, throughout the course of this project. The authors also thank the GMIT faculty and lab staff for providing resources, and acknowledge Google's Gemini API, on which the identification functionality of this application depends.

REFERENCES

[1] C. Mora et al., "How Many Species Are There on Earth and in the Ocean?" PLOS Biology, vol. 9, no. 8, e1001127, Aug. 2011.

[2] H. Voss, M. Matz, and P. Biber-Freudenberger, "Citizen Science and Deep Learning for Coral Reef Fish Monitoring," Remote Sensing, vol. 14, no. 5, p. 1230, 2022.

[3] S. Han, Y. Liu, and P. Wang, "Underwater Image Enhancement Using Deep CNN with Colour Correction and CLAHE," IEEE Access, vol. 8, pp. 93712-93728, 2020.

[4] A. Boom, J. He, S. Palazzo, P. Spampinato, and C. Kavasidis, "A Research Tool for Long-term and Continuous Analysis of Fish Assemblage in Coral Reefs," Ecological Informatics, vol. 23, pp. 83-97, 2014.

[5] B. Qin, T. Cao, and Y. Zhang, "Deep Fish: A Deep Learning Framework for Fine-grained Fish Species Identification," J. Marine Science and Engineering, vol. 9, no. 6, p. 640, 2021.

[6] H. Rauf, M. Saleem, and R. Khan, "Fine-Grained Fish Species Classification Using ResNet-50 with Progressive Transfer Learning," Applied Sciences, vol. 12, no. 10, p. 4994, 2022.

[7] S. Jäger, K. Fricke, and M. Bochow, "Real-time Fish Detection in Turbid Underwater Video Using YOLOv4," Frontiers in Marine Science, vol. 8, 2021.

[8] A. Radford et al., "Learning Transferable Visual Models from Natural Language Supervision," in Proc. ICML, pp. 8748-8763, 2021.

[9] M. Oquab et al., "DINOv2: Learning Robust Visual Features without Supervision," Transactions on Machine Learning Research, 2024.

[10] S. Vaze, K. Han, A. Vedaldi, and A. Zisserman, "Open-Set Recognition: A Good Closed-Set Classifier is All You Need?" in Proc. ICLR, 2022.

[11] Google DeepMind, "Gemini: A Family of Highly Capable Multimodal Models," arXiv:2312.11805, 2023.

[12] J. Achiam et al., "GPT-4 Technical Report," arXiv:2303.08774, 2023.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)