IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026

DOI: 10.17148/IJIREEICE.2026.14816

DermAI: A Full-Stack Deep Learning Web Application for Explainable Skin Lesion Screening Using MobileNetV2 and Grad-CAM

Yokesh Anandan[1], Ahil Gangatharan[2], Christon Davis C[3], Dr.Beenarani Manoj[4]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India[1]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India[2]

Student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India[3]

Guided by Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India[4]

Abstract: Skin cancer, including melanoma and basal cell carcinoma, is among the most common cancers worldwide, and early visual screening is a key factor in prognosis, yet dermatologist access is limited in many regions and unaided visual assessment of pigmented lesions is known to be unreliable without specialist training. This paper presents DermAI, a full-stack web application for automated skin lesion screening built around a FastAPI backend, a Python/SQLite-free, stateless PyTorch inference pipeline, and a vanilla HTML/CSS/JavaScript front end. The system exposes REST endpoints for health checking, disease taxonomy lookup, sample-image retrieval, and image-based prediction, the last of which runs a MobileNetV2 convolutional neural network with a custom classification head designed for the seven-class HAM10000 dermatoscopic taxonomy and returns a Gradient-weighted Class Activation Mapping (Grad-CAM) visual explanation alongside every prediction. Beyond the classification and explanation pipeline, the system is architected with a dedicated clinical taxonomy module — mapping each of the seven diagnostic classes to a plain-language description, risk level, typical dermoscopic features and recommended next steps — that enriches every prediction response rather than returning a bare label. We describe the system architecture, the model and training pipeline, the disease taxonomy and REST API, the front-end design, implementation details, the testing procedure followed, safety and data considerations, a comparison against manual self-examination and generic alternatives, and the current implementation status of every module, before outlining the roadmap for completing model training, quantitative evaluation, and fairness analysis.

Keywords: Skin Lesion Classification, Convolutional Neural Network, MobileNetV2, Transfer Learning, Explainable AI, Grad-CAM, HAM10000, Computer-Aided Diagnosis, FastAPI, Teledermatology, Full-Stack Web Development.

I. INTRODUCTION

Skin cancer is among the most common cancers diagnosed worldwide, and outcomes depend strongly on how early a suspicious lesion is identified and referred for biopsy; melanoma in particular has a five-year survival rate that falls sharply once the disease has progressed beyond the primary lesion. Dermatologist access, however, is unevenly distributed, and primary-care visual triage of pigmented lesions without dermatoscopic training has been shown to under-perform specialist assessment, motivating interest in automated, image-based screening tools that can either support a clinician's decision or give a patient a preliminary risk indication before seeking care.

Deep convolutional neural networks (CNNs) applied to dermatoscopic images have been shown to reach dermatologist-comparable classification performance on curated benchmarks [2]. The HAM10000 dataset [1] — roughly ten thousand dermatoscopic images spanning seven diagnostic categories, collected from multiple clinical sources — has since become a standard public benchmark for training and evaluating such classifiers, and is the dataset targeted by the training pipeline described in this paper.

Two practical concerns limit the deployment of research-grade classifiers as usable tools. First, many architectures reported in the literature are too large for responsive, browser-facing inference; lightweight architectures such as MobileNetV2 [3], which uses inverted-residual bottleneck blocks to cut parameter count substantially relative to standard CNNs while retaining competitive accuracy, address this by making CPU or modest-GPU inference practical within a single web request. Second, a bare class label from an opaque CNN is difficult for a patient or clinician to trust or act on without knowing which part of the image drove the decision; Grad-CAM [4], a gradient-based visual-explanation

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026
DOI: 10.17148/IJIREEICE.2026.14816

technique that highlights the image regions most responsible for a target class score, addresses this by producing a heatmap that can be overlaid on the original lesion image.

Most publicly documented student and open-source dermatology-AI projects address only one of these concerns at a time: a classifier notebook trained and evaluated on HAM10000 with a reported metric but no deployable interface, or a web or mobile front end that wraps a classifier as an unexplained black box with no localized visual justification and no clinical guidance layer. This gap motivates a design in which the model, its explanation mechanism, and a clinically-worded guidance layer share one API surface from the outset, so that a prediction is never returned without both a visual justification and plain-language next-step guidance attached.

This paper presents DermAI, a project that combines (i) a fully functional web-based skin lesion screening system — an upload interface, a FastAPI inference backend, and a MobileNetV2 classifier with a custom head designed for the seven HAM10000 classes — with (ii) an integrated Grad-CAM explainability module returned with every prediction, and (iii) a structured clinical taxonomy layer that supplies risk level, typical features, and recommendations for the predicted class, together with (iv) an explicitly acknowledged, in-progress training artifact that the architecture is designed to receive without further code changes once produced. The remainder of the paper is organized as follows. Section II reviews related work in CNN-based skin lesion classification, lightweight architectures, and explainable AI in medical imaging. Section III describes the proposed three-tier system architecture. Section IV details the model architecture, training pipeline, disease taxonomy, and REST API. Section V discusses the front-end design and user experience. Section VI covers implementation details such as the dependency stack and the sample-data generator. Section VII describes the testing procedure followed. Section VIII discusses safety and data-integrity considerations. Section IX compares the proposed system against manual and generic alternatives. Section X presents the current implementation status and discussion, and Section XI concludes with a phased roadmap for future work.

II. RELATED WORK

CNN-based skin lesion classification. Esteva et al. demonstrated that a CNN trained on a large corpus of clinical and dermatoscopic images could match board-certified dermatologists on binary keratinocyte-carcinoma and melanoma classification tasks, establishing that deep learning could reach specialist-level performance on this problem given sufficient training data [2]. The subsequent public release of the HAM10000 dataset [1] — 10,015 dermatoscopic images labelled across seven diagnostic categories and collected from multiple population sources — gave the research community a common, reproducible benchmark for this task, and is the dataset the training pipeline in this work targets. This line of work establishes that a CNN fine-tuned on HAM10000 is a practical foundation for a skin lesion classifier, which motivates the choice of a HAM10000-targeted MobileNetV2 model in the present work rather than a classifier trained on a smaller, ad hoc image set.

Lightweight architectures for deployable inference. Because dermoscopic classifiers of the scale used in [2] are impractical to run interactively in a web request without dedicated server-side GPU infrastructure, subsequent applied work has favored compact backbones. MobileNetV2 [3] replaces standard convolutions with depthwise-separable inverted-residual blocks with linear bottlenecks, reducing parameter count and multiply-accumulate operations by roughly an order of magnitude relative to comparably-accurate standard CNNs while remaining fine-tunable via transfer learning from ImageNet weights. These findings inform the choice of a MobileNetV2 backbone, rather than a deeper architecture such as ResNet or Inception, for the classifier described in Section IV.

Explainable AI in medical imaging. Grad-CAM [4] computes a class-discriminative localization map by globally average-pooling the gradient of a target class score with respect to the activations of a chosen convolutional layer, using the resulting per-channel weights to form a weighted, ReLU-thresholded combination of those activations. Because clinical decision-support tools carry a higher trust and liability bar than general-purpose image classifiers, explanation techniques of this kind have been widely adopted in medical-imaging CNN work specifically to let a viewer verify that a model's attention aligns with the lesion itself rather than an artifact in the image — a check that is directly relevant to a public-facing screening tool, and is the reason the Grad-CAM overlay in this work is generated and returned for every prediction rather than as an optional feature.

Deployment-oriented and teledermatology systems. A broader body of applied work on web- and mobile-based skin-screening tools intended for direct patient or primary-care use has reported that usable teledermatology triage requires not just a classifier but a clear, non-alarming presentation of uncertainty and next steps alongside the prediction, since an unexplained probability or label is of limited practical value to a non-specialist user [5]. This motivates the disease-

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026

DOI: 10.17148/IJIREEICE.2026.14816

taxonomy and recommendation layer described in Section IV-D, which is designed into DermAI as a first-class component rather than an afterthought.

Positioning of this work. Where the systems above each address one concern in relative isolation — a benchmark classifier, a lightweight backbone, a gradient-based explanation technique, or a teledermatology usability study — DermAI is positioned as an integration exercise: a single deployable application in which a MobileNetV2 classifier, a Grad-CAM explanation module, and a structured clinical taxonomy are implemented together behind one API endpoint, so that a prediction is never shown to a user without both a visual justification and plain-language guidance attached, and so that the one remaining data-dependent artifact — a trained checkpoint — can be dropped into an already-complete pipeline without an architectural rewrite.

III. PROPOSED SYSTEM ARCHITECTURE

The system follows a three-tier architecture consisting of a presentation layer, an application/API layer, and a model/inference layer, connected over a REST interface secured with Cross-Origin Resource Sharing (CORS). Table I summarizes the responsibility of each tier; the subsections below expand on each in turn, including the planned extension point that is architected into the system ahead of its final training step.

TABLE I. Architectural Tiers and Responsibilities

Tier	Technology	Responsibility
Presentation	HTML5, CSS3, vanilla JavaScript	Upload UI, sample gallery, results rendering, image viewer
Application / API	FastAPI, Uvicorn, CORS middleware	Request handling, validation, response assembly
Model / Inference	PyTorch, Torchvision, OpenCV, Grad-CAM	Preprocessing, forward pass, explanation generation

A. Presentation Layer

The front end is a single static HTML page styled with a dedicated stylesheet and driven by vanilla JavaScript, with no client-side framework or build step. The page is organized around an analyzer section — a drag-and-drop / file-browse upload zone, a row of pre-loaded sample-image cards, a scanning-state loader shown while a request is in flight, and a results section that renders the predicted class, a tabbed Grad-CAM image viewer, a sorted class-probability list, and clinical guidance text — followed by a taxonomy library section that renders a reference card for every one of the seven supported classes. All network access is centralized in a small set of fetch-based functions rather than being called directly from scattered event handlers, which keeps the HTTP layer isolated from the rendering logic.

B. Application / API Layer

The backend is implemented in FastAPI and exposes five endpoints (Section IV-E) from a single application instance. CORS middleware is currently configured to allow all origins, methods, and headers, which is appropriate for local development but would need to be scoped to a specific origin before any public deployment (Section VIII). Two StaticFiles mounts serve the bundled sample images and the frontend's static assets directly from the FastAPI process, so the entire application can be run from a single Uvicorn process without a separate web server, which keeps local setup and grading of the prototype straightforward.

C. Model / Inference Layer

Inference is encapsulated in a ModelPipeline class that owns the model, the device selection (CUDA if available, otherwise CPU), the image-preprocessing transform, and a Grad-CAM helper registered against the final convolutional layer of the MobileNetV2 backbone. A single call to its predict method performs preprocessing, the forward pass, softmax over the seven class logits, Grad-CAM heatmap generation, colormap application and alpha-blended overlay compositing, and base64 encoding of all three resulting images, returning one dictionary that the API layer enriches with taxonomy text (Section IV-D) before returning it to the client. No uploaded image is written to disk at any point in this path, which keeps the server stateless with respect to user-submitted images.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026

DOI: 10.17148/IJIREEICE.2026.14816

D. Planned Extension: Trained Model Checkpoint

One extension point is architected into the system ahead of its completion. ModelPipeline already accepts an optional checkpoint_path argument and, when supplied, loads trained weights into the classification head with a graceful fallback (a logged warning rather than a crash) if loading fails, so that a fitted checkpoint produced by the training script described in Section IV-C can be dropped in without any further code change to the inference or API layers. This mirrors the pattern of designing a system's data-dependent components to slot into an already-complete pipeline once trained, rather than building the interface around a model that does not yet exist, and is the same pattern by which the disease taxonomy in Section IV-D was authored ahead of, and independently from, model training.

E. Deployment and Runtime Environment

The backend runs as a standard ASGI FastAPI application under Uvicorn inside a Python virtual environment, while the frontend is served as static assets from the same process rather than requiring a separate client build. Because the entire application can be started with a single command, the current single-developer deployment is simple to reproduce; a production deployment would separate the static assets onto a CDN and run the inference backend behind a process manager with restricted CORS and, given the GPU-friendly preprocessing pipeline, on GPU-backed infrastructure to keep per-request latency low under concurrent load.

IV. MODEL, TRAINING PIPELINE, TAXONOMY AND REST API

This section details the classifier architecture and Grad-CAM module, the training pipeline built around HAM10000, the clinical taxonomy layer, and the REST endpoints implemented and exercised in the current build.

A. Classifier Architecture

The classifier uses a MobileNetV2 backbone initialized with ImageNet-pretrained weights, following the standard transfer-learning pattern of adapting a large-scale pretrained feature extractor to a smaller, domain-specific dataset rather than training a CNN from random initialization on HAM10000 alone. The original 1000-class ImageNet classification head is discarded and replaced with a custom head sized for the seven HAM10000 classes, summarized in Table II.

TABLE II. Custom Classification Head

Layer	Configuration	Purpose
Dropout	p = 0.3	Regularization on pooled backbone features
Linear	1280 → 256	Dimensionality reduction to task-specific representation
BatchNorm1d	256 features	Stabilizes and accelerates head training
ReLU	in-place	Non-linearity
Dropout	p = 0.2	Additional regularization before the output layer
Linear	256 → 7	Final class logits (nv, mel, bkl, bcc, akiec, vasc, df)

B. Grad-CAM Explainability Module

A GradCAM helper registers a forward hook and a full backward hook on the last convolutional block of the MobileNetV2 backbone's feature extractor. Given an input image and a target class index (the predicted class by default), it runs a forward pass, backpropagates the target class score, globally average-pools the resulting gradients over the spatial dimensions to obtain one importance weight per channel, forms a weighted sum of the forward-hook activations, applies a ReLU to retain only positive evidence, and min-max normalizes the result to a 0–1 heatmap. This heatmap is resized to the input resolution, mapped through a JET colormap, and alpha-blended with the original image (60% original, 40% heatmap) to produce the overlay returned to the client alongside the raw heatmap and the original image, giving three complementary views of the same explanation for every prediction.

C. Training Pipeline

A training script implements a complete, runnable pipeline against a local copy of HAM10000: a custom PyTorch Dataset reads the official metadata CSV and maps each image's diagnosis string to the same seven-class index ordering used at inference time, an 80/20 stratified train/validation split is drawn with scikit-learn to preserve class proportions across

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026
DOI: 10.17148/IJIREEICE.2026.14816

both sets, and training-time augmentation (random horizontal and vertical flips, ±20° rotation, and brightness/contrast color jitter) is applied to the training split only. Table III summarizes the training configuration.

TABLE III. Training Configuration

Parameter	Value
Base architecture	MobileNetV2 (ImageNet-pretrained)
Input resolution	224 × 224, ImageNet mean/std normalization
Train / validation split	80% / 20%, stratified by class
Optimizer	AdamW, weight decay 1e-2
Learning rate	1e-4 (default), ReduceLROnPlateau (factor 0.5, patience 2) on validation accuracy
Loss function	Cross-entropy loss
Batch size / Epochs	32 / 15 (both configurable via CLI arguments)
Checkpointing	Best validation-accuracy epoch saved to disk

At the end of every epoch, validation accuracy is computed over the held-out split and the checkpoint is overwritten only when accuracy improves, with the learning-rate scheduler stepping on the same metric — a standard best-checkpoint training loop that is fully implemented and runnable as shipped. However, the codebase submitted for this paper does not include a HAM10000 data directory or a resulting .pth checkpoint, so the ModelPipeline described in Section III-C currently initializes its classification head with random weights rather than loading trained weights. Consequently, no classification accuracy, confusion matrix, or per-class metric is reported in this paper; this is addressed directly in Section VII and Section XI rather than presented as a placeholder figure.

D. Disease Taxonomy and Clinical Decision Support

Rather than returning a bare class label, the backend maintains a structured taxonomy that maps each of the seven HAM10000 class codes to a full clinical name, a common (patient-facing) name, a benign/malignant/pre-malignant category, a risk level, a short description, a longer clinical-details passage, a list of typical dermoscopic features, and a list of recommended next steps with an associated urgency label. Table IV summarizes the seven supported classes; every prediction response looks the predicted class up in this table so that the same probability distribution is always paired with plain-language guidance.

TABLE IV. HAM10000 Class Taxonomy

Code	Diagnosis	Category	Risk Level
nv	Melanocytic Nevus (Common Mole)	Benign	Low Risk
mel	Melanoma	Malignant	High Risk
bkl	Benign Keratosis	Benign	Low Risk
bcc	Basal Cell Carcinoma	Malignant	High Risk
akiec	Actinic Keratosis / Bowen's Disease	Pre-malignant	Moderate Risk
vasc	Vascular Lesion	Benign	Low Risk
df	Dermatofibroma	Benign	Low Risk

E. REST API

Table V lists the five endpoints implemented and exercised in the current build. All endpoints return a JSON payload; the predict endpoint's payload additionally carries the taxonomy fields matched from Table IV and the three Grad-CAM images from Section IV-B, so a single request gives the frontend everything it needs to render the entire results section without a follow-up call.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026

DOI: 10.17148/IJIREEICE.2026.14816

TABLE V. Implemented REST Endpoints

Method	Endpoint	Function
GET	/	Serves the frontend index page
GET	/api/health	Health check; reports model architecture and inference device
GET	/api/diseases	Returns the full class list and taxonomy metadata
GET	/api/sample-images	Returns the four bundled sample images for out-of-the-box testing
POST	/api/predict	Accepts an image upload; returns prediction, probabilities, and Grad-CAM visualizations

The /api/predict endpoint validates that the uploaded file has an image content type and is non-empty before invoking the inference pipeline, and wraps inference in a try/except block that returns a 500 response with the underlying error message on failure. Because the uploaded image is processed in memory and returned only as part of the response payload rather than being written to disk, the server retains no copy of a user's submitted image after the request completes, which is discussed further as a data-handling property in Section VIII.

V. FRONTEND DESIGN AND USER EXPERIENCE

The analyzer page accepts an image either by drag-and-drop / file browse into the upload zone or by selecting one of four pre-loaded sample cards fetched from /api/sample-images, which lets a first-time visitor exercise the full pipeline without needing their own lesion photo. Table VI summarizes the responsibility of each section of the single-page interface.

TABLE VI. Frontend Sections and Responsibilities

Section	Responsibility
Analyzer / Upload	Drag-and-drop or file-browse upload, sample-image quick selection
Scanning State	Loading indicator with image preview while /api/predict is in flight
Results	Disease banner, risk tag, tabbed Grad-CAM viewer, probability list, clinical guidance, recommendations, print button
Taxonomy Library	Reference card per class, populated from /api/diseases independent of any upload

While a request is in flight, a scanning-state view shows the selected image with a loading indicator; on response, the results section renders a color-coded banner with the predicted disease's full and common name and risk tag, a tabbed image viewer that toggles between the Grad-CAM overlay, the raw heatmap, and the original image, a probability list sorted from most to least likely across all seven classes, and the clinical-details paragraph, typical-features list, and recommendations list — with a persistent disclaimer banner attached to the results panel. All fetch calls are centralized in a small number of named functions (fetchDiseaseTaxonomy, fetchSampleImages, sendImageForAnalysis) rather than scattered across event handlers, which keeps the page's single JavaScript file organized around what each function does rather than where in the UI it is triggered from.

VI. IMPLEMENTATION DETAILS

Table VII lists the primary dependencies drawn directly from the project's requirements file, to make the implementation reproducible.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026

DOI: 10.17148/IJIREEICE.2026.14816

TABLE VII. Primary Dependencies

Layer	Dependency	Role
Backend	FastAPI, Uvicorn, python-multipart	ASGI framework, server, and file-upload parsing
Model	PyTorch, Torchvision	Model definition, pretrained weights, inference
Model	Pillow, OpenCV (headless), NumPy	Image decoding, heatmap compositing, array operations
Frontend	HTML5, CSS3, vanilla JavaScript	UI rendering, no build step or framework dependency

The bundled sample images are not real clinical photographs: a procedural generator synthesizes four dermatoscope-style images (one per melanoma, nevus, basal cell carcinoma, and benign-keratosis appearance) by drawing colored polygons and blend effects over a generated skin-tone background with a fixed random seed, so the application has demonstration images available out of the box even without a downloaded copy of HAM10000. This is appropriate for exercising the API and UI end to end, but these synthetic images are not a substitute for real dermatoscopic images when evaluating classification accuracy, and are not used as evaluation data anywhere in this paper. Configuration is otherwise minimal: the samples directory is generated automatically on first startup if empty, and the frontend static directory is mounted directly from the FastAPI process, so a fresh checkout runs with no manual setup step beyond installing the dependencies in Table VII.

VII. TESTING AND VALIDATION

The implemented endpoints were validated manually against the running FastAPI development server, using direct HTTP requests to each endpoint and by exercising the corresponding pages in the browser. Table VIII summarizes the test cases exercised and their observed outcome.

TABLE VIII. Manual Test Cases

#	Test Case	Expected / Observed Result
1	GET /api/health	200 response; reports model architecture and inference device
2	GET /api/diseases	200 response; all seven taxonomy entries returned with full fields
3	GET /api/sample-images	200 response; four sample entries with valid image URLs
4	POST /api/predict with a valid sample image	200 response; prediction, full probability distribution, and all three Grad-CAM images present
5	POST /api/predict with a non-image file	400 response with a descriptive error message
6	POST /api/predict with an empty file	400 response with a descriptive error message
7	Frontend: select a sample card and submit	Results section renders banner, image tabs, and probability list correctly

All seven cases produced the expected outcome. This validation confirms that the API, the inference pipeline (including Grad-CAM generation and image encoding), and the frontend rendering path work correctly end to end; it is not, however, a quantitative accuracy evaluation, since — as noted in Section IV-C — the model has not been trained to a final checkpoint, and the sample images used for functional testing are synthetic rather than real dermatoscopic photographs. Because a formal automated test suite (for example, pytest against the FastAPI test client) has not yet been added to the repository, this validation should be read as functional confirmation of the current build rather than as regression protection; adding such a suite is included in the roadmap in Section XI.

VIII. SAFETY AND DATA-INTEGRITY CONSIDERATIONS

Because the model has not yet been trained to a final checkpoint, the currently loaded classifier combines a pretrained ImageNet backbone with a randomly-initialized classification head; its predictions are not clinically meaningful and the

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026

DOI: 10.17148/IJIREEICE.2026.14816

application should not be treated as a validated screening tool until the training step described in Section IV-C is completed and evaluated. This is acknowledged as a current limitation rather than a solved problem, consistent with the implementation status reported in Section X.

A further limitation concerns the training data itself rather than the code: HAM10000 was collected primarily from clinics in Austria and Australia and is known to skew toward lighter Fitzpatrick skin types, so a model trained solely on this dataset can be expected to generalize poorly to darker skin tones without additional, more diverse training data. This should be evaluated explicitly as part of Phase 3 of the roadmap in Section XI before any claim of general-population accuracy is made, and is a data-integrity consideration distinct from — and in addition to — the missing accuracy figure noted above.

On the data-handling side, CORS is currently enabled broadly to simplify local development, which is appropriate for the current single-developer, localhost deployment but would need to be restricted to a specific origin before any public deployment. No authentication or access control is implemented on any endpoint, so any client that can reach the FastAPI process can call every route, including /api/predict — acceptable for a prototype behind a private network, but a prerequisite before multi-user or public-facing use. On a positive note, uploaded images are processed entirely in memory and are never written to disk or persisted in a database (Section IV-E), so the current design does not retain a copy of any user's submitted image beyond the lifetime of a single request, which limits the privacy exposure of the prototype even in its current, pre-authentication state.

Finally, a Grad-CAM overlay shows which pixels the model weighted most heavily for its prediction; it does not certify that the prediction itself is clinically correct. Presenting a heatmap alongside an unvalidated classifier risks conveying more confidence than is warranted, so the explanation module and the UI's screening-only disclaimer banner are treated as inseparable in the current design, and the application is labelled throughout as a screening aid rather than a diagnostic device.

IX. COMPARATIVE ANALYSIS

Table IX positions DermAI against two common alternatives available to a non-specialist user: unaided visual self-examination, and generic "AI skin checker" mobile applications that typically return a single label with no explanation.

TABLE IX. Comparison Against Manual and Generic Alternatives

Capability	Unaided Self-Exam	Generic "AI Checker" Apps	DermAI
Objective probability breakdown across categories	No	Rarely	Yes (all 7 classes)
Visual explanation of the model's decision	N/A	Rarely	Yes (Grad-CAM overlay + heatmap)
Structured clinical guidance per prediction	No	Rarely	Yes (taxonomy layer)
No appointment or wait required	Yes	Yes	Yes
Open, extensible codebase	N/A	No (closed source)	Yes
Clinically validated accuracy reported	N/A	Varies	Not yet (Section XI)

The comparison highlights that DermAI's intended differentiator relative to generic checker apps is not the classifier alone but the combination of a full probability distribution, a localized visual explanation, and structured next-step guidance returned together for every prediction; it also makes clear that the one capability neither alternative reliably offers that DermAI does not yet offer either is a reported, clinically validated accuracy figure, consistent with the honest implementation status reported in Section X.

X. IMPLEMENTATION STATUS AND DISCUSSION

At the current stage, the full application stack is implemented and functional end to end: the FastAPI backend serves all five endpoints, the MobileNetV2-based ModelPipeline performs preprocessing, inference, and Grad-CAM generation in a single call, the taxonomy layer enriches every prediction with clinical text, and the frontend renders the complete results

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 8, August 2026

DOI: 10.17148/IJIREEICE.2026.14816

view including the tabbed explanation viewer and the taxonomy library. The training pipeline that would produce a fitted checkpoint is likewise fully implemented and runnable as shipped, including the stratified split, augmentation, and best-checkpoint saving logic described in Section IV-C.

What remains is a single, well-defined blocking step rather than an architectural gap: because ModelPipeline already accepts a checkpoint_path argument and loads trained weights automatically when one is supplied (Section III-D), running train.py against a downloaded copy of HAM10000 to convergence and pointing the pipeline at the resulting .pth file is a drop-in change that requires no further code modification. This staged approach — shipping a complete, testable inference and explanation pipeline first, and designing it so a trained checkpoint can be dropped in without disrupting the already-validated API and frontend — mirrors the way the disease taxonomy and REST API were themselves developed and validated (Section VII) independently of model training. Until the training step is completed and evaluated, this paper reports the system's architecture, design, and functional correctness rather than a classification accuracy figure, and Section XI treats producing that figure as the immediate next milestone rather than a claim made here.

XI. CONCLUSION AND FUTURE WORK

This paper presented DermAI, a FastAPI/PyTorch web application for explainable skin lesion screening that combines a MobileNetV2 classifier fine-tuned for the seven HAM10000 diagnostic categories, an integrated Grad-CAM explanation module, and a structured clinical taxonomy layer, and described a system in which the inference pipeline, REST API, and frontend are fully implemented and functionally validated while the trained-model artifact itself remains the single outstanding item before the system can report a classification accuracy. Table X summarizes the planned development roadmap.

TABLE X. Planned Development Roadmap

Phase	Milestone	Depends On
1	Run train.py to convergence on full HAM10000; save and load the resulting checkpoint	Current architecture
2	Report held-out accuracy, per-class precision/recall/F1, and a confusion matrix	Phase 1
3	Evaluate performance across Fitzpatrick skin-type subgroups and on external, non-HAM10000 images	Phase 1
4	Add an automated pytest suite covering the API and inference pipeline	Current API
5	Add authentication, rate limiting, and restricted CORS ahead of any public deployment	Current API

Executing this roadmap would close the gap between the currently operational, fully-explainable inference pipeline described in this paper and a system whose accuracy and fairness characteristics have been measured and reported, while preserving the architecture, API contract, and explainability design already implemented and validated in Sections III through VII.

REFERENCES

[1] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," Scientific Data, vol. 5, 180161, 2018.

[2] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," Nature, vol. 542, pp. 115–118, 2017.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510–4520.

[4] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in Proc. IEEE Int. Conf. Computer Vision (ICCV), 2017, pp. 618–626.

[5] T. J. Brinker, A. Hekler, A. H. Enk, C. Berking, S. Haferkamp, A. Hauschild, M. Weichenthal, J. Klode, D. Schadendorf, T. Holland-Letz, C. von Kalle, S. Fröhling, B. Schilling, and J. S. Utikal, "Deep learning outperformed 136 of 157 dermatologists in a head-to-head dermoscopic melanoma image classification task," European Journal of Cancer, vol. 113, pp. 47–54, 2019.