Journal of Technology and Informatics (JoTI)
Vol. 8, No. 2, Bulan Oktober Tahun 2026
P-ISSN 2721-4842
E-ISSN 2686-6102

Cloud-Based Mobile Application for Skin Lesion Screening Using InceptionV3

Habibullah Akbar[1*], Raynaldi Sandy[2]
[1*] Faculty of Computer Science, Master of Computer Science, Universitas Esa Unggul, Indonesia
[2] Faculty of Computer Science, Informatics Engineering, Universitas Esa Unggul, Indonesia
e-mail: habibullah.akbar@esaunggul.ac.id [1*], raynaldisandy77@student.esaunggul.ac.id [2]

Article Information

Article History:
Received : 13 May 2026
Revised : 3 July 2026
Accepted : 28 July 2026
Published : 16 September 2026

***Correspondence:**
Email
habibullah.akbar@esaunggul.ac.id

Keywords:
Skin Lesion Classification, Mobile Application, Cloud Computing, InceptionV3, Flutter, Google Cloud Platform

10.37802/joti.v8i2.1346

Journal of Technology and Informatics (JoTI)
P-ISSN 2721-4842
E-ISSN 2686-6102
https://e-journals.dinamika.ac.id/index.php/joti

***Abstract*:**

Skin-lesion studies have achieved promising classification accuracy; however, their implementation in mobile–cloud environments remains limited. This study presents Skin Alert, a cloud-based prototype for classifying seven skin-lesion categories in the HAM10000 dataset using an ImageNet-pretrained InceptionV3 model. Hyperparameters, including learning rate, dropout rate, batch size, and number of epochs, were optimized using grid search. The original class distribution was retained, while data augmentation and class-weighted learning were applied during training. The final model achieved an internal validation accuracy of 85.32%, weighted precision of 85.79%, weighted recall of 85.32%, weighted F1-score of 85.38%, macro F1-score of 72.93%, and macro-average one-vs-rest AUC of 96.84%. The model was deployed through a Flask API on Google Cloud Platform and integrated with a Flutter application, Firebase Authentication, Realtime Database, and Storage. Black-box and backend integration testing confirmed that predefined workflows operated as intended. The locally hosted API achieved approximately 250–300 ms response time per image, while cloud deployment required approximately 1.10 s per image. These findings demonstrate the technical feasibility of the proposed mobile–cloud prototype.

INTRODUCTION

Skin cancer remains an important public-health concern, and excessive exposure to ultraviolet radiation is a major preventable risk factor [1]. The visual similarity among different skin lesions can complicate assessment, even for trained practitioners. Convolutional Neural Networks (CNNs) have demonstrated the ability to extract discriminative features from dermoscopic images and have been applied using architectures such as MobileNet, ResNet,

DenseNet, and Inception-based networks [2]–[6]. Nevertheless, skin-lesion classification remains affected by class imbalance, limited minority-class samples, domain shift, and restricted external clinical validation [3], [5], [6]. Consequently, high performance on an internal dataset does not by itself demonstrate reliable clinical performance.

Mobile implementation of skin-lesion analysis is not entirely new. Previous studies have examined deep learning in combination with mobile technology [7], developed a real-time augmented-reality smartphone application [8], implemented mobile skin-cancer recognition [9], evaluated an AI-based application in a population setting [10], and developed a mobile CNN application for melanoma classification [11]. These studies demonstrate multiple deployment patterns, ranging from on-device inference to applications supported by remote services. The present study uses the HAM10000 dermoscopic dataset [12] and the InceptionV3 architecture [13]. Related work has also investigated web-based skin-lesion classification [14], lightweight models for cloud-assisted diagnosis [15], and on-device mobile inference [16]. Building on these studies, Skin Alert investigates a different system configuration: a Flutter mobile client integrated with Firebase services and a Flask prediction API hosted on Google Cloud Platform. The principal contribution is therefore the implementation and system-level evaluation of an end-to-end mobile–cloud prototype, including functional, backend-integration, and response-time testing.

METHOD

In this study, model development and mobile application development were conducted sequentially. The model development included dataset preparation, preprocessing, hyperparameter exploration, transfer learning, fine-tuning, and internal validation. In contrast, the application development used an iterative prototyping approach in which requirements, interfaces, backend connections, and prediction workflows were implemented and evaluated [17], [18]. The development process consisted of three phases. Firstly, the Requirement Analysis and Design phase defines user requirements, application workflows, system architecture, interface designs, and data relationships. Secondly, the mobile Prototype was developed using Flutter [19] and integrated with Firebase Authentication, Firebase Realtime Database, Firebase Storage, and a Flask API hosted on Google Cloud Platform [20]. Finally, the Evaluation and Refinement performs the black-box and backend-integration testing.

Research Stages

The HAM10000 dataset [12] contains 10,015 dermoscopic images distributed across seven categories: melanocytic nevi (nv, 6,705), melanoma (mel, 1,113), benign keratosis-like lesions (bkl, 1,099), basal cell carcinoma (bcc, 514), actinic keratoses (akiec, 327), vascular lesions (vasc, 142), and dermatofibroma (df, 115). An image-level stratified 80:20 split with a random state of 42 produced 8,012 training images and 2,003 internal validation images. The corresponding training/validation counts were nv 5,364/1,341, mel 890/223, bkl 879/220, bcc 411/103, akiec 262/65, vasc 114/28, and df 92/23. During the training phase, the class imbalance was mitigated using balanced class weights and then grid search was conducted on the class-balanced subset. The validation subset was used for performance monitoring, early stopping, checkpoint selection, and final internal assessment. Although grid-search training was using balanced dataset, the final internal-validation set retained the original imbalanced HAM10000 distribution.

The images were resized to 299×299 RGB and rescaled by 1/255. Training augmentation included rotations of up to 15°, width and height shifts of 0.10, zoom of 0.10, horizontal flipping, nearest-neighbor filling, and brightness variation of 0.9–1.1. Validation

images were only rescaled. The ImageNet-pretrained InceptionV3 architecture [13] used in this study includes canonical global average pooling, a 256-unit ReLU layer, dropout, and a seven-unit softmax output. Training used Adam and categorical cross-entropy. Fine-tuning used a learning rate of 10^{-5} and clipnorm of 1.0, together with early stopping, ReduceLROnPlateau, and checkpointing. Grid search evaluated 81 combinations of learning rate, dropout, batch size, and maximum epochs (see Table 1). Experiments were conducted in Google Colab using an NVIDIA Tesla T4 GPU.

Table 1. Hyperparameter Values for Grid Search Optimization

No	Parameter	Values
1	Learning rate	0.001; 0.0001; 0.00001
2	Dropout	0.3; 0.4; 0.5
3	Batch size	16; 64; 128
4	Maximum epochs	30; 40; 50

System Architecture

Figure 1 illustrates the mobile–cloud architecture of the Skin Alert prototype. The Flutter application provides authentication, image submission, prediction-result presentation, history access, and profile management. Firebase Authentication manages user access, Firebase Realtime Database stores structured user and screening-history records, and Firebase Storage stores uploaded images [19], [20]. When an image is submitted, the mobile application sends an HTTP request to the Flask API hosted on GCP App Engine. The server applies the required preprocessing, executes the InceptionV3 classifier, and returns the predicted class and confidence score in JSON format. Performing inference on the cloud reduces the computational workload on the mobile device, although it introduces dependence on network connectivity and remote-service availability [21].

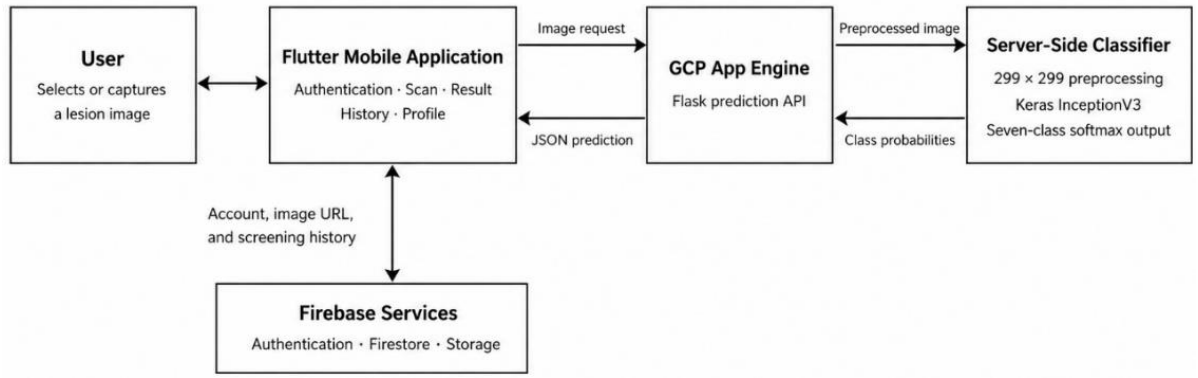

Figure 1. System Architecture of the Proposed Prototype

System Development Process

The development process began with the specification of system requirements, user workflows, architectural components, and data relationships using established modeling and software-engineering principles [22], [23]. The database design follows a one-to-many relationship in which one authenticated user may have multiple screening-history records [24]. Each history record stores the image reference, predicted lesion category, confidence value, and screening timestamp. These records are retrieved by the mobile application to display the user's previous screening results.

System Evaluation

The system evaluation includes functionality and integration testing. To evaluate system behavior, we utilized black-box testing. Each major feature was tested, including the login and registration process, image uploading, data transmission to the API, receipt of prediction results, and storage of results in the user history. The testing results indicate that all application functions operated according to the defined scenarios and produced valid outputs. On the other hand, we used integration testing to verify communication among system components such as Firebase Authentication, Firestore Database, Firebase Storage, and Google Cloud Platform (GCP). This process ensured that user data, images, and prediction results could be synchronized in real time. The testing results demonstrated that all backend services interacted properly without connection failures or data loss, indicating that the system integration operated optimally.

RESULTS AND DISCUSSION

Model Performance Evaluation

The InceptionV3 model performance evaluation was conducted to assess its capability in classifying seven types of skin lesions using the HAM10000 dataset. Figure 2 presents the validation accuracy and validation loss curves during the training of the InceptionV3 model. The validation accuracy curve shows that, during initial training, the model achieved an accuracy of approximately 71%, which gradually increased to around 76%. During fine-tuning, the accuracy improved more rapidly and stabilized between 83% and 85%. In the deep fine-tuning stage, the initial accuracy was approximately 82% and increased to around 85–86% by the final epoch. These results indicate that fine-tuning improved the model's ability to recognize patterns in the validation data. The validation loss curve shows that, during initial training, the loss was relatively high at approximately 0.80 before decreasing to around 0.68–0.70, although some fluctuations remained. Fine-tuning produced a greater reduction in validation loss, reaching approximately 0.55, with more stable values than those observed during initial training. During deep fine-tuning, the validation loss remained stable within the range of 0.62–0.66. Although this loss was higher than that achieved during fine-tuning, the stage provided a balance between validation accuracy and generalization performance.

The final checkpoint achieved an internal validation accuracy of 85.32%, weighted precision of 85.79%, weighted recall of 85.32%, weighted F1-score of 85.38%, and macro-average one-vs-rest multiclass AUC of 96.84%. Although the AUC indicates strong probability ranking across classes, the lower macro F1-score and melanoma recall show that class-specific errors remain important.

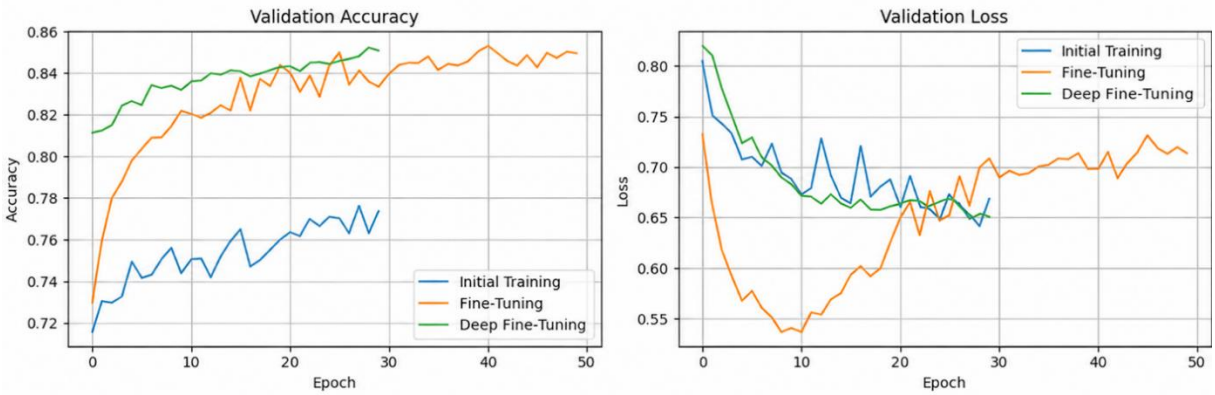

Figure 2. Training and Validation Performance of the InceptionV3 Model

Table 2 shows that performance varied across lesion classes. The highest F1-scores were obtained for melanocytic nevi (94%) and vascular lesions (89.66%), whereas dermatofibroma achieved the lowest score (57.14%) with only 23 validation images. Melanoma achieved precision, recall, and F1-score values of 64%, 64.57%, and 64.29%. The difference between macro F1 (72.93%) and weighted F1 (85.38%) indicates that class imbalance affected minority-class performance.

Table 2. Per-class Performance (2,003 Validation Images)

Class	Precision	Recall	F1-score	Support
Actinic keratoses	0.8095	0.5231	0.6355	65
Basal cell carcinoma	0.7526	0.7087	0.7300	103
Benign keratosis-like lesions	0.6302	0.7591	0.6887	220
Dermatofibroma	0.6316	0.5217	0.5714	23
Melanocytic nevi	**0.9457**	**0.9344**	**0.9400**	1,341
Melanoma	0.6400	0.6457	0.6429	223
Vascular lesions	0.8667	0.9286	0.8966	28
Macro average	0.7537	0.7173	0.7293	2,003
Weighted average	0.8579	0.8532	0.8538	2,003
Overall Accuracy		0.8532		2,003

Figure 3 presents the confusion matrix of the InceptionV3 model for seven skin-lesion classes. The model correctly classified 1,709 of 2,003 images (overall accuracy of 85.32%, as mentioned before). Overall, the model performed well on dominant classes, particularly melanocytic nevi, but its ability to distinguish melanoma from visually similar benign lesions requires further improvement.

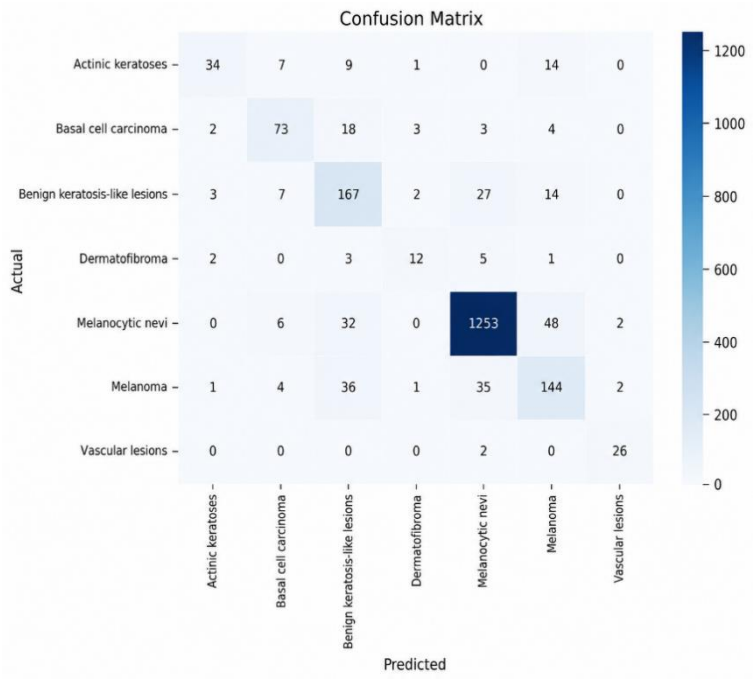

Figure 3. Confusion Matrix of the InceptionV3 Model

Table 3 contextualizes the proposed work relative to previous studies. Abdulredah et al. [26] reported 81.93% accuracy on a binary version of HAM10000 in which the original lesion categories were regrouped into benign and malignant classes. This result is not directly comparable with the seven-class task examined in the present study. Manzoor et al. [27] reported 82.36% accuracy for EfficientFormerV2 on the imbalanced HAM10000 data. Their overall framework was architecturally more elaborate because it combined U-Net with a VGG16 encoder for lesion segmentation and EfficientFormerV2 for classification. However, [27] did not report implementation or testing of the proposed framework within a mobile application; its experiments were conducted in a workstation-based Jupyter/TensorFlow environment, and edge deployment was identified as future work.

The present study achieved 85.32% internal validation accuracy while retaining the original class distribution and applying class-weighted learning. The distinct contribution of the present study is the implementation and evaluation of a Flutter–Firebase–GCP mobile–cloud workflow, including functional, integration, and response-time testing.

Table 3. Contextual Comparison with Previous Studies

Study	Dataset	Model	Accuracy	Mobile-app evaluation
Abdulredah et al. (2025) [26]	Binary HAM10000	SWNet	81.93%	-
Manzoor et al. (2025) [27]	Imbalanced HAM10000	U-Net/VGG16 segmentation and EfficientFormerV2 classification	82.36%	-
Proposed study	Class-weighted HAM10000	Optimized InceptionV3 using Grid Search	**85.32%**	✓

User Interface Implementation

Figure 4 presents the eight main interfaces of the Skin Alert mobile application: splash screen, login, registration, home, scan, history, classification result, and user profile. Users first register or log in through Firebase Authentication before accessing the home page, which provides educational information and navigation to the application's main features. On the scan page, users select a skin-lesion image from the device gallery. The image is then sent to the Flask API hosted on Google Cloud Platform and processed using the InceptionV3 model.

The result page displays the uploaded image, predicted lesion category, prediction confidence, and a disclaimer emphasizing that the result is not a definitive medical diagnosis. Previous results are stored in Firebase Firestore and can be reviewed through the history page, while the profile page allows users to manage their personal information and log out. Overall, the figure demonstrates the complete application workflow, from user authentication and image submission to prediction, result storage, and profile management.

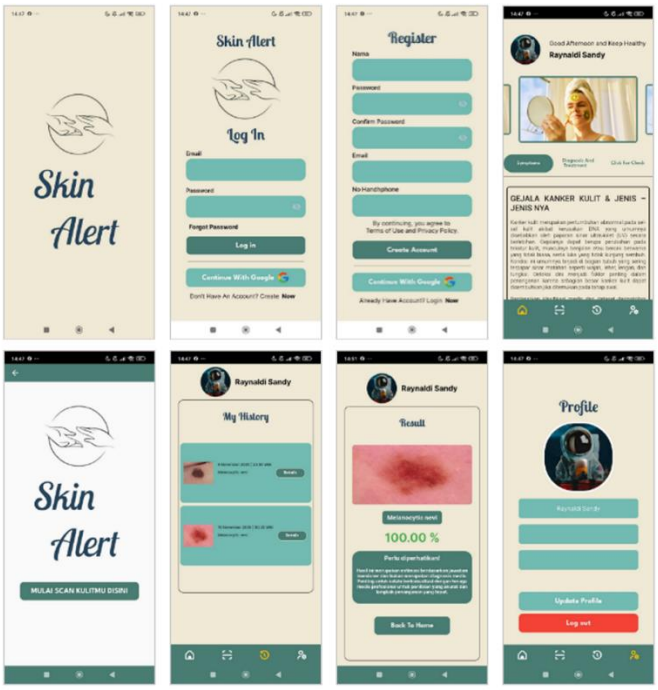

Figure 4. Skin Alert Application Interface

Backend Implementation

Figure 5 presents the Firebase services integrated into the Skin Alert application. The upper-left panel shows Firebase Authentication, which manages user registration, email-based login, and Google authentication. The upper-right panel displays Firebase Realtime Database, which stores structured application data, such as user profiles and detection records. The lower panel shows Firebase Storage, which stores uploaded skin-lesion images and user profile pictures. The stored image URLs can subsequently be linked to their corresponding database records. Together, these services support user authentication, data storage, image management, and retrieval of detection history within the application.

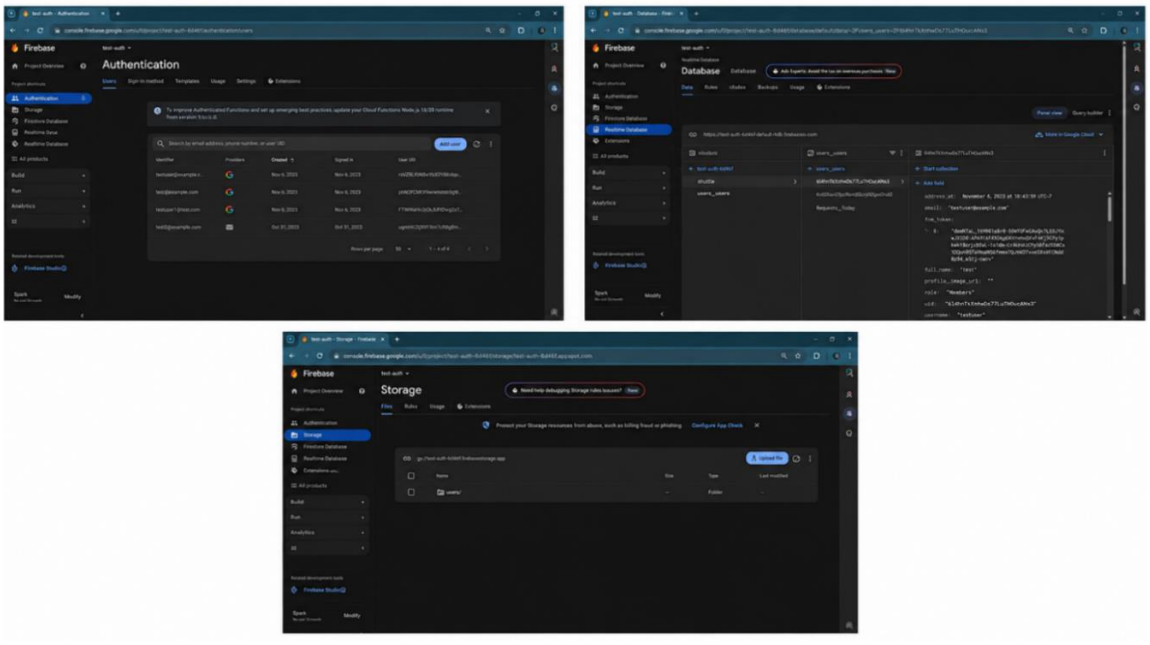

Figure 5. Firebase Services Integrated into the Skin

Google Cloud Platform (GCP) Integration

Figure 6 shows the Google Cloud Platform environment used to deploy the Skin Alert backend. The left panel shows the Cloud Shell Editor, where backend components such as the Flask API, app.py, model. keras, requirements.txt, and app.yaml are managed. The API receives skin-lesion images from the mobile application, performs preprocessing and InceptionV3 inference, and returns the predicted class and confidence score in JSON format. The right panel presents the GCP dashboard, which provides information about the project, App Engine, API activity, billing, service status, and monitoring.

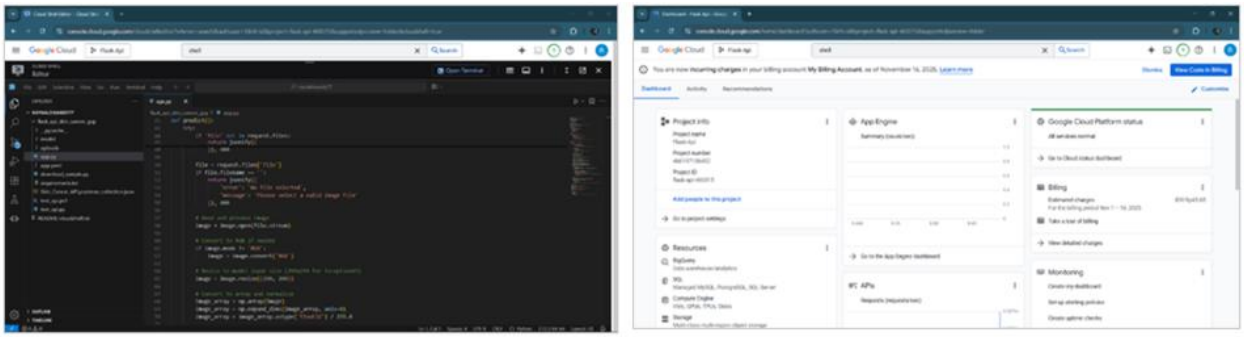

Figure 6. API and GCP Dashboard Implementation

System Evaluation Results

Table 4 summarizes the black-box functional testing of the Skin Alert application. Eight main features were evaluated: login, registration, image scanning, lesion detection, result display, detection history, profile updating, and logout. The results show that all tested functions operated successfully according to their expected scenarios. Users were able to authenticate, create accounts, upload skin-lesion images, receive prediction results from the API, review stored detection history, update profile information, and securely exit the application.

Table 4. Black Box Testing Results of the *Skin Alert* Application

No	Tested Feature	Testing Scenario	Result
1	Login	User enters email and password	Successful
2	Register	User creates a new account	Successful
3	Scan	User uploads an image from camera/gallery	Successful
4	Lesion Detection	System sends image to API and receives result	Successful
5	Result Page	System displays prediction result	Successful
6	History	Detection history is stored and displayed	Successful
7	Profile	Profile data can be updated	Successful
8	Logout	User exits the application	Successful

Table 5 summarizes the integration testing of the backend services supporting the Skin Alert application. Firebase Authentication successfully handled user registration and login, while Firebase Storage stored uploaded lesion and profile images. The database component successfully stored and retrieved detection-history data. The GCP-hosted API was also able to receive image-based prediction requests and return classification responses to the mobile application. Finally, frontend–backend integration testing confirmed that the complete

workflow from image submission and lesion classification to result display and history storage functioned as intended. Overall, the results indicate that the application components were successfully connected and exchanged data correctly. The locally hosted Flask API was tested using Postman and produced an average response time of approximately 250–300 ms per image. After cloud deployment, the average end-to-end prediction time was approximately 1.10 seconds per image.

Table 5. Backend Integration Testing Results

No	Tested Component	Testing Scenario	Result
1	Firebase Authentication	User registration and login	Successful
2	Firebase Storage	Image file storage	Successful
3	Firestore Database	Storage and retrieval of history data	Successful
4	GCP API	Send prediction requests and receive responses	Successful
5	Frontend–Backend Integration	Scan → detect → result → history workflow	Successful

CONCLUSION AND SUGGESTIONS

This study developed Skin Alert, a mobile–cloud research prototype integrating an InceptionV3 skin-lesion classifier with Flutter, Firebase, and Google Cloud Platform. Using an image-level internal validation split of HAM10000, the model achieved 85.32% accuracy, 85.38% weighted F1-score, 72.93% macro F1-score, and 96.84% macro-average one-vs-rest multiclass AUC across seven classes. The original class distribution was retained during final model development, while class-weighted learning was used to reduce majority-class dominance. The class-balanced subset was used only during hyperparameter grid search. Performance remained strongest for melanocytic nevi and vascular lesions, whereas dermatofibroma, actinic keratoses, and melanoma produced lower F1-scores. The mobile application supported authentication, image submission, cloud-based prediction, result presentation, history storage, profile management, and logout. Functional and backend-integration testing showed that the principal workflow operated as intended under the tested configuration. Local Flask API response times ranged from approximately 250 to 300 ms per image, while the mean end-to-end prediction time following cloud deployment was approximately 1.10 s per image. The principal contribution of this study is the implementation and system-level evaluation of the mobile–cloud prototype, without claiming clinical validation. However, the reported results remain limited by the absence of an independent test set and the lack of external clinical usability. Future work should use lesion-grouped data partitioning, independent and clinically representative datasets, improved minority-class sensitivity, and more rigorous latency, reliability, and usability evaluation.

REFERENCES

[1] World Health Organization, "Ultraviolet radiation," Jun. 21, 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/ultraviolet-radiation. [Accessed: Aug. 17, 2026].

[2] J. Vieira, F. Mendonça, and F. Morgado-Dias, "Deep Learning Approaches for Skin Lesion Detection," Electronics, vol. 14, no. 14, Art. no. 2785, 2025, doi: 10.3390/electronics14142785.

[3] Y. Wu, B. Chen, A. Zeng, D. Pan, R. Wang, and S. Zhao, "Skin Cancer Classification With Deep Learning: A Systematic Review," Frontiers in Oncology, vol. 12, Art. no. 893972, 2022, doi: 10.3389/fonc.2022.893972.

[4] B. Shetty, R. Fernandes, A. P. Rodrigues, et al., "Skin lesion classification of dermoscopic images using machine learning and convolutional neural network," Scientific Reports, vol. 12, Art. no. 18134, 2022, doi: 10.1038/s41598-022-22644-9.

[5] T. M. Alam, K. Shaukat, W. A. Khan, I. A. Hameed, L. A. Almuqren, M. A. Raza, M. Aslam, and S. Luo, "An Efficient Deep Learning-Based Skin Cancer Classifier for an Imbalanced Dataset," Diagnostics, vol. 12, no. 9, Art. no. 2115, 2022, doi: 10.3390/diagnostics12092115.

[6] B. C. R. S. Furriel et al., "Artificial intelligence for skin cancer detection and classification for clinical environment: a systematic review," Frontiers in Medicine, vol. 10, Art. no. 1305954, 2024, doi: 10.3389/fmed.2023.1305954.

[7] E. Goceri, "Diagnosis of skin diseases in the era of deep learning and mobile technology," Computers in Biology and Medicine, vol. 134, Art. no. 104458, 2021, doi: 10.1016/j.compbiomed.2021.104458.

[8] R. Francese, M. Frasca, M. Risi, and G. Tortora, "A mobile augmented reality application for supporting real-time skin lesion analysis based on deep learning," Journal of Real-Time Image Processing, vol. 18, pp. 1247–1259, 2021, doi: 10.1007/s11554-021-01109-8.

[9] I. Kousis, I. Perikos, I. Hatzilygeroudis, and M. Virvou, "Deep Learning Methods for Accurate Skin Cancer Recognition and Mobile Application," Electronics, vol. 11, no. 9, Art. no. 1294, 2022, doi: 10.3390/electronics11091294.

[10] A. M. Smak Gregoor et al., "An artificial intelligence based app for skin cancer detection evaluated in a population based setting," npj Digital Medicine, vol. 6, Art. no. 90, 2023, doi: 10.1038/s41746-023-00831-w.

[11] R. Omirgaliyev, A. Arystanbekova, A. Alzhanov, and A. Makatov, "Mobile Application of Convolutional Neural Networks for Melanoma Classification," in 2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST), Astana, Kazakhstan, 2024, doi: 10.1109/SIST61555.2024.10629269.

[12] 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, Art. no. 180161, 2018, doi: 10.1038/sdata.2018.161.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.

[14] S. Aksoy, P. Demircioglu, and I. Bogrekci, "Deep Learning-Based Web Application for Automated Skin Lesion Classification and Analysis," Dermato, vol. 5, no. 2, Art. no. 7, 2025, doi: 10.3390/dermato5020007.

[15] H. W. Huang, B. W.-Y. Hsu, C.-H. Lee, and V. S. Tseng, "Development of a light-weight deep learning model for cloud applications and remote diagnosis of skin cancers," The Journal of Dermatology, vol. 48, no. 3, pp. 310–316, 2021, doi: 10.1111/1346-8138.15683.

[16] I. Valova, P. Dinh, and N. Gueorguieva, "Mobile Application for Skin Cancer Classification Using Deep Learning," Journal of Machine Intelligence and Data Science, vol. 4, pp. 15–26, 2023, doi: 10.11159/jmids.2023.003.

[17] L. Lasminiasih, G. E. Saputra, R. B. Utomo, and E. Wiseno, "Using Prototyping Method for Analysis and Design of Information Systems for Student Registration in Sekolah Master," International Journal Science and Technology, vol. 1, no. 2, pp. 19–29, 2022, doi: 10.56127/ijst.v1i2.140.

[18] D. J. C. Sihombing, "Exploring prototype methodology in land information system development: design and evaluation of an application," Jurnal Info Sains: Informatika dan Sains, vol. 14, no. 1, pp. 594–604, 2024. [Online]. Available: https://ejournal.seaninstitute.or.id/index.php/InfoSains/article/view/4007.

[19] Flutter, "Development," [Online]. Available: https://flutter.dev/development. [Accessed: Aug. 17, 2026].

[20] Google, "Firebase Documentation," [Online]. Available: https://firebase.google.com/docs. [Accessed: Aug. 17, 2026].

[21] S. Sachdeva et al., "Unraveling the role of cloud computing in health care system and biomedical sciences," Heliyon, vol. 10, no. 7, Art. no. e29044, 2024, doi: 10.1016/j.heliyon.2024.e29044.

[22] Object Management Group, "OMG Unified Modeling Language (OMG UML), Version 2.5.1," Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1/PDF.

[23] M. Anggraeni, H. Andhika F. R., H. Rante, and S. Sukaridhoto, "Indonesian Sign Language (SIBI) Learning Media Application Based on Deep Learning Technology for Deaf Children," Journal of Technology and Informatics (JoTI), vol. 5, no. 1, pp. 41–47, 2023, doi: 10.37802/joti.v5i1.384.

[24] M. R. D. Ulhaq, M. A. Zaidan, and D. Firdaus, "Pengenalan Ekspresi Wajah Secara Real-Time Menggunakan Metode SSD Mobilenet Berbasis Android," Journal of Technology and Informatics (JoTI), vol. 5, no. 1, pp. 48–52, 2023, doi: 10.37802/joti.v5i1.387.

[25] V. H. Sahin, I. Oztel, and G. Yolcu Oztel, "Human Monkeypox Classification from Skin Lesion Images with Deep Pre-trained Network using Mobile Application," Journal of Medical Systems, vol. 46, Art. no. 79, 2022, doi: 10.1007/s10916-022-01863-7.

[26] A. A. Abdulredah, M. A. Fadhel, L. Alzubaidi, Y. Duan, M. Kherallah, and F. Charfi, "Towards unbiased skin cancer classification using deep feature fusion," BMC Medical Informatics and Decision Making, vol. 25, Art. no. 48, 2025, doi: 10.1186/s12911-025-02889-w.

[27] K. Manzoor, N. U. Gilal, M. Agus, and J. Schneider, "Dual-stage segmentation and classification framework for skin lesion analysis using deep neural network," Digital Health, vol. 11, Art. no. 20552076251351858, 2025, doi: 10.1177/20552076251351858.