Department of Electrical & Computer Engineering

AIR UNIVERSITY

PROJECT REPORT

CE-362 Software Engineering

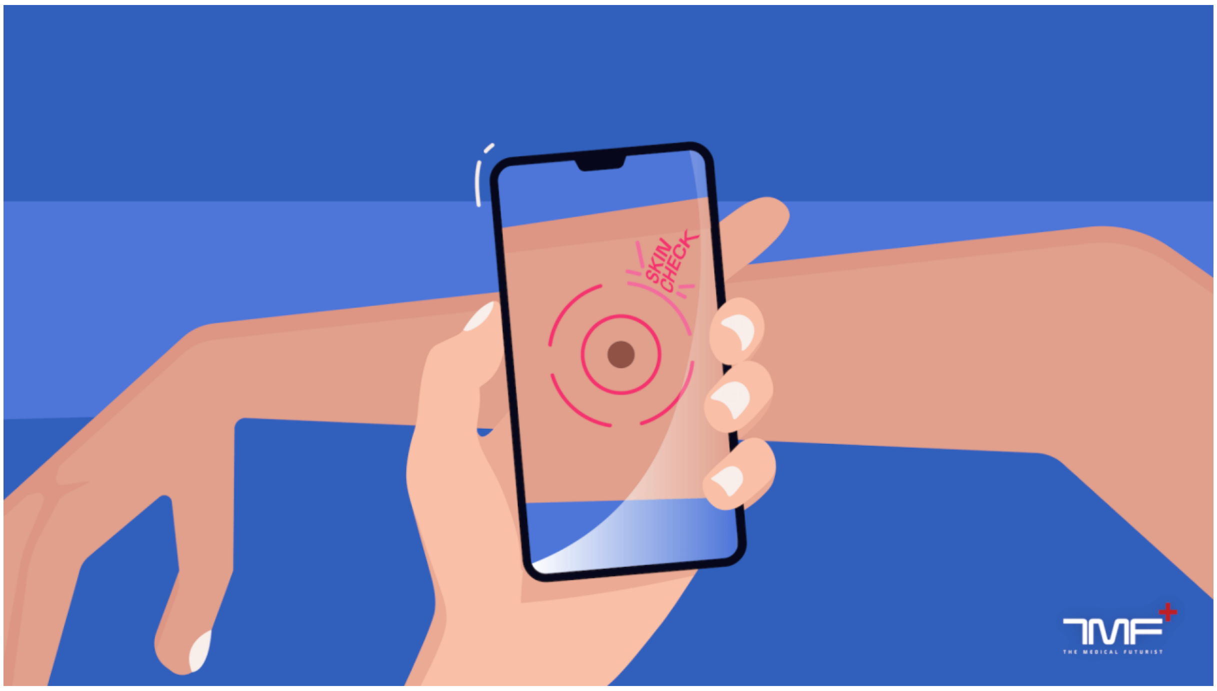
Instructor Name

Miss Marium Sabir

Submitted by

Rafia Arshad	210275
Muhammad Burhan Ahmed	210287
Noor ur Huda Rana	210294
Sadia Yaqoob	210284
Abdur Rehman	210312

06/01/2025



AI-Based Derma Checker Web Application

ABSTRACT

The **AI Derma Check Web Application** is an intelligent skin disease detection system designed to assist users in identifying dermatological conditions through image analysis. The application allows users to upload images of affected skin areas, which are then analyzed using a **YOLO11 model** trained on a specialized skin disease classification dataset from Roboflow. Upon detection, the system provides a detailed report including the predicted disease, confidence score, disease description, common symptoms, and practical care tips. By combining modern object detection with an easy-to-use web interface, the application aims to offer accessible preliminary screening, support early awareness, and promote proactive skin health management.

Contents

1 Introduction

1.1 Overview

Skin diseases represent a significant global health burden, affecting individuals of all ages and demographics. Prompt and accurate identification of dermatological conditions is vital to prevent disease progression and improve patient outcomes. Yet, in many regions, access to specialized dermatological care is constrained by geographic, economic, or infrastructural barriers. To bridge this gap, this project presents a web-based *AI Derma Check* platform that harnesses state-of-the-art deep learning for rapid, user-friendly skin disease screening.

At the core of the system lies a **YOLO11** object-detection model, meticulously trained on a diverse skin disease classification dataset sourced from **Roboflow**. Upon uploading a dermatoscopic or clinical image of a suspicious lesion, the application processes the input through the YOLO11 pipeline to accurately localize and classify the skin condition. Within seconds, users receive a comprehensive report that includes:

- **Predicted disease label** and **confidence score**
- A concise **disease description**
- Key **clinical symptoms** to watch for
- Practical **care and management tips**

Built with a responsive front-end framework, the platform delivers a seamless experience across desktop and mobile devices. By combining rapid object detection with an intuitive interface, the AI Derma Check Web Application empowers users with immediate, preliminary insights—supporting early awareness of skin conditions and encouraging timely consultation with healthcare professionals.

1.2 Purpose:

The AI Derma Check Web Application project aims to deliver an accessible, reliable, and efficient preliminary screening tool for skin diseases by using advanced deep learning techniques. By allowing users to upload images of skin disease and receive instantaneous feedback including the predicted disease label, confidence score, concise description, key symptoms, and practical care tips the system seeks to improve early detection and promote timely medical consultation. Designed as a web-based platform, it addresses geographical and economic barriers to dermatological expertise, making basic skin health assessment available to anyone with an internet connection and a camera-enabled device. Additionally, the project demonstrates the real-world applicability of the YOLO11 object-detection model, trained on a comprehensive Roboflow skin classification dataset, both as a proof of concept for AI-driven medical image analysis and as a potential augmentation tool for healthcare professionals. Through its user-friendly interface and rapid, informative reporting, the AI Derma Check Web Application fosters greater skin health awareness, supports proactive self-care, and underscores the transformative role of AI in modern healthcare.

1.3 Project Scope:

The AI Derma Check Web Application enables users to upload images of skin lesions via a responsive web interface and receive, in real time, an automated assessment powered by a YOLO11 model trained on a Roboflow skin disease classification dataset. The project scope includes data preprocessing, model training and validation, implementation of FastAPI endpoints for inference, and development of the front end to display the predicted disease label, confidence score, description, symptoms, and care tips. Deployment is limited to a standard web environment (desktop and mobile browsers). This project does not cover clinical diagnosis, treatment prescription, storage of sensitive personal data, or offline/on-device inference.

1.4 Technologies Used

The AI Derma Check Web Application integrates several modern tools and technologies to deliver accurate skin disease detection and a friendly web-based interface. Below are the primary technologies used:

- **YOLO11:** A state-of-the-art object detection model used for identifying and classifying different types of skin diseases from uploaded images. The model was trained on a labeled dataset using the YOLO11 architecture for high accuracy.
- **Roboflow (Skin Disease Dataset):** The project utilizes the "Skin Disease" dataset available on Roboflow[1]. Roboflow was used for dataset management, including annotation, preprocessing, and exporting the dataset in YOLO-compatible format for model training.
- **FastAPI:** A fast and lightweight web framework for building RESTful APIs in Python. It serves as the backend of the application, handling image upload requests, running the prediction model, and returning disease reports.
- **Python:** The core programming language used for backend development, integration of the detection model, and implementation of the inference pipeline.
- **TypeScript, TailwindCSS:** Advanced frontend technologies used to design and build the user interface of the web application, ensuring responsiveness and accessibility with major browsers, including Google Chrome and Mozilla Firefox.
- **Google Colab:** Used for training and testing the YOLO11 model with GPU acceleration.
- **GitHub:** Employed for version control and project collaboration, enabling structured and trackable development.
- **Jira:** Used for project management, task tracking, sprint planning, and ensuring organized collaboration among team members throughout the software development lifecycle.

[1] `https://universe.roboflow.com/skin-disease/skin-disease-vrvtv`

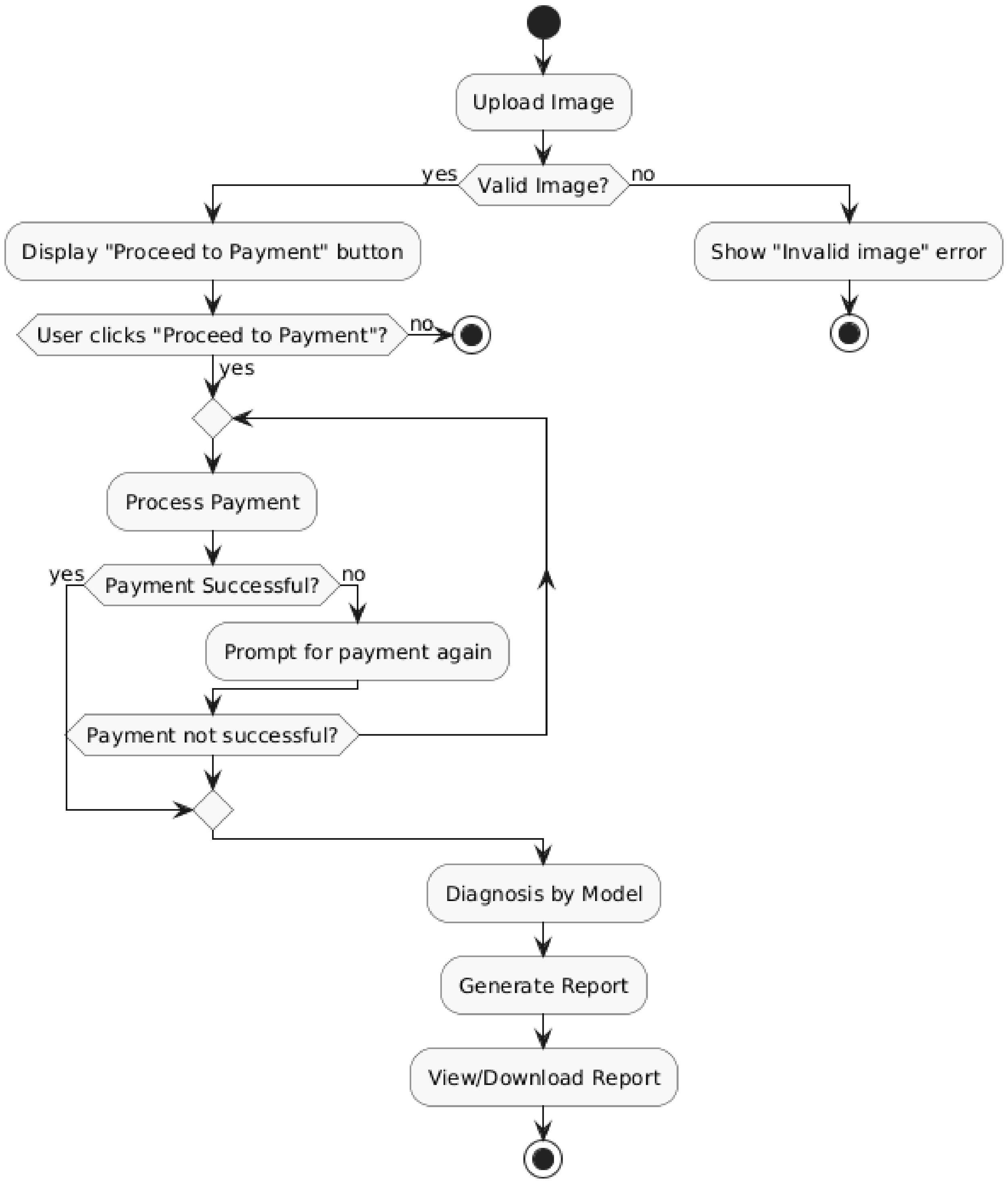


Figure 1: Activity diagram for AI Derma Checker

2 Overall Description

This section provides a high–level view of the AI Derma Check Web Application, outlining its context, major functions, and system constraints.

2.1 Product Perspective

The AI Derma Check Web Application is a standalone, web-based screening tool that brings rapid, AI-powered skin disease detection to users' fingertips. Unlike traditional dermatological diagnosis, which requires an in-person consultation, this system leverages a YOLO11 deep-learning model (trained on the Roboflow skin disease dataset) to deliver preliminary classification results within seconds. Users interact through any modern browser—uploading an image, viewing the predicted condition and confidence score, and, upon payment, downloading a detailed PDF report.

2.2 Dataset Description

The AI Derma Check Web Application is trained using the "Skin Disease" object detection dataset (v6, March 25 2023) from Roboflow Universe [2]. This dataset contains a total of 1,147 annotated images covering 10 distinct classes of skin conditions.

- Acne
- Atopic Dermatitis
- Chicken Skin
- Eczema
- Hansen's Disease Leprosy (mild)
- Hansen's Disease Leprosy (severe)
- Healthy Skin
- Psoriasis
- Ringworm
- Warts

[2] `https://universe.roboflow.com/skin-disease/skin-disease-vrvtv`

Table 1: Roboflow Skin Disease Dataset Attributes

Attribute	**Description**
Dataset Name	Skin Disease (v6) from Roboflow Universe
Total Images	1,147 annotated images
Image Size	Preprocessed to 416×416 pixels (RGB)
Image Format	JPEG / PNG
Source	Roboflow Universe: `https://universe.roboflow.com/skin-disease/skin-disease-vrvtv`
Class Labels	• Acne • Atopic Dermatitis • Chicken Skin • Eczema • Hansen's Disease / Leprosy (mild and severe) • Healthy Skin • Psoriasis • Ringworm • Warts
Dataset Split	Training: 83% (953 images) Validation: 10% (115 images) Test: 7% (79 images)

Each image is annotated with bounding boxes around the lesion of interest, enabling the YOLO11 model to learn both localization and classification. This well-annotated and diverse dataset underpins the system's ability to deliver accurate, real-time skin disease detection.

2.3 User Classes and Characteristics

The system supports the following types of users:

- **General Users (Patients):** Individuals who upload images of skin diseases for automated analysis. They interact with the web interface to receive preliminary detection results, confidence scores, disease descriptions, symptoms, and care tips. Their primary concern is ease of use, quick feedback, and clarity of the report.
- **System Administrators:** Personnel responsible for maintaining the application, including server management, model updates, and monitoring system health. They configure model versions, manage dataset imports, oversee payment gateway settings, and ensure that the application remains secure and up to date.

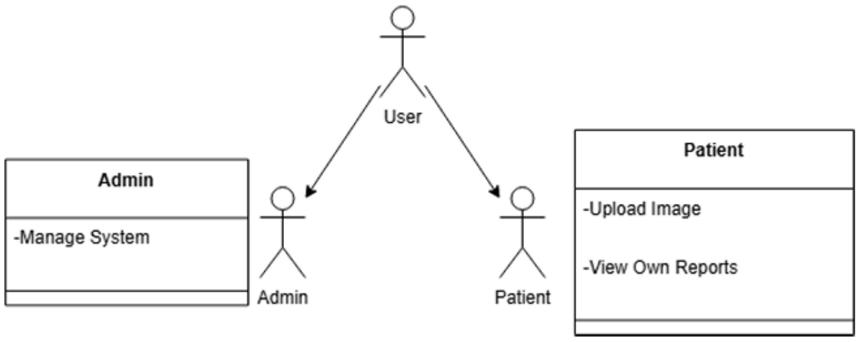


Figure 2: User Types

2.4 Product Functions

The system provides the following core functions:

- **Image Upload:** Users can upload images of affected skin areas for analysis.
- **Image Validation:** The backend verifies that the uploaded file is a valid medical image and appears to contain skin content.
- **Disease Detection:** Upon validation, the YOLO11 model analyzes the image and returns the predicted skin disease along with a confidence score.
- **Payment Processing:** Secure payment gateway integration for getting disease detection report and its analysis.
- **Result Display:** After payment, the system presents the PDF report summarizing disease name, confidence, brief description, key symptoms, and suggested care tips.

Figure 3 shows the functions of the system

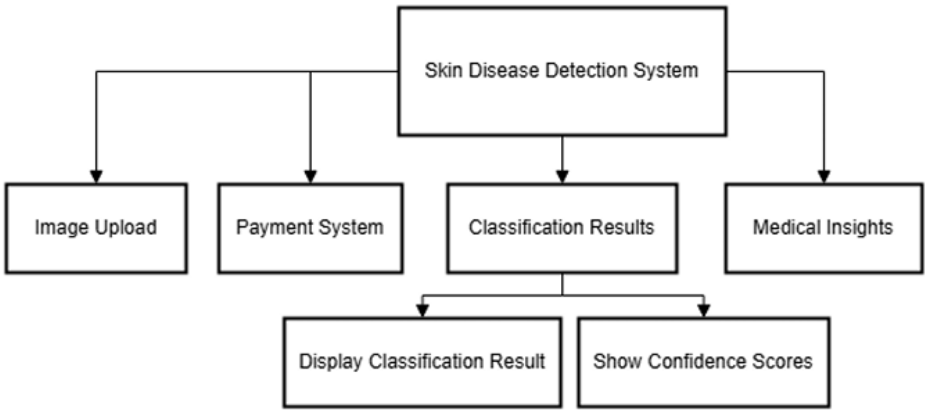


Figure 3: Functionalities of System

2.5 User Interaction Flow

The AI Derma Check Web Application guides users through a clear, step-by-step process from image submission to report generation. The steps involved in the user interaction are as follows:

1. **Navigate to Upload Page:** Upon using the application, the user first selects the "Upload Image" option to begin the analysis.
2. **Image Submission & Validation:** The user uploads a JPEG or PNG image of the skin affected area. The system validates the file format and ensures it contains a skin disease before proceeding.
3. **Preliminary Analysis:** Valid images are passed to the YOLO11 model, which performs disease detection. A brief preview of the predicted condition and confidence score is shown in the interface.
4. **Payment Prompt & Verification:** To unlock the full PDF report, the user is prompted to complete a secure payment. The system redirects to the payment gateway and awaits confirmation.
5. **Report Generation & Delivery:** After successful payment, the backend generates a comprehensive PDF report including disease name, description, symptoms, care tips, and confidence metrics then provides a download link.
6. **Error Handling:** If the image fails validation or the payment is declined, the user is shown an appropriate error message and offered the chance to retry.

2.6 Design and Implementation Constraints

- Model inference must complete within 5 seconds to meet performance requirements.
- All data exchanges must occur over HTTPS to ensure confidentiality.
- Uploaded images are limited to JPEG/PNG formats and a maximum size of 5 MB.
- The PDF generator must preserve report formatting across different operating systems.

2.7 Assumptions and Dependencies

- Users have access to a camera-enabled device and a stable internet connection.
- The external payment service remains available and maintains its API contract.
- Roboflow model datasets are accessible and compatible with the chosen model framework.
- No offline or on-device inference is required for this phase of the project.

2.8 Project Organization

2.8.1 Team Structure

The project team is organized to ensure smooth coordination and development:

- **CEO & Stakeholder (Mam Mariam Sabir)**: Oversees project vision and alignment with business goals.
- **Project Lead (Burhan Ahmed)**: Manages the overall project, coordinates team members, and ensures timely delivery.
- **AI/ML Engineer (Rafia Arshad)**: Builds and integrates the AI model for skin disease prediction.
- **Frontend Developer (Abdur Rehman)**: Develops the user interface using Next.js and TypeScript.
- **Backend Developer (Noor ul Huda Rana)**: Handles server-side logic, API integration, and PDF generation.
- **QA Engineer (Sadia Yaqoob)**: Conducts testing to ensure the system functions correctly and meets requirements.

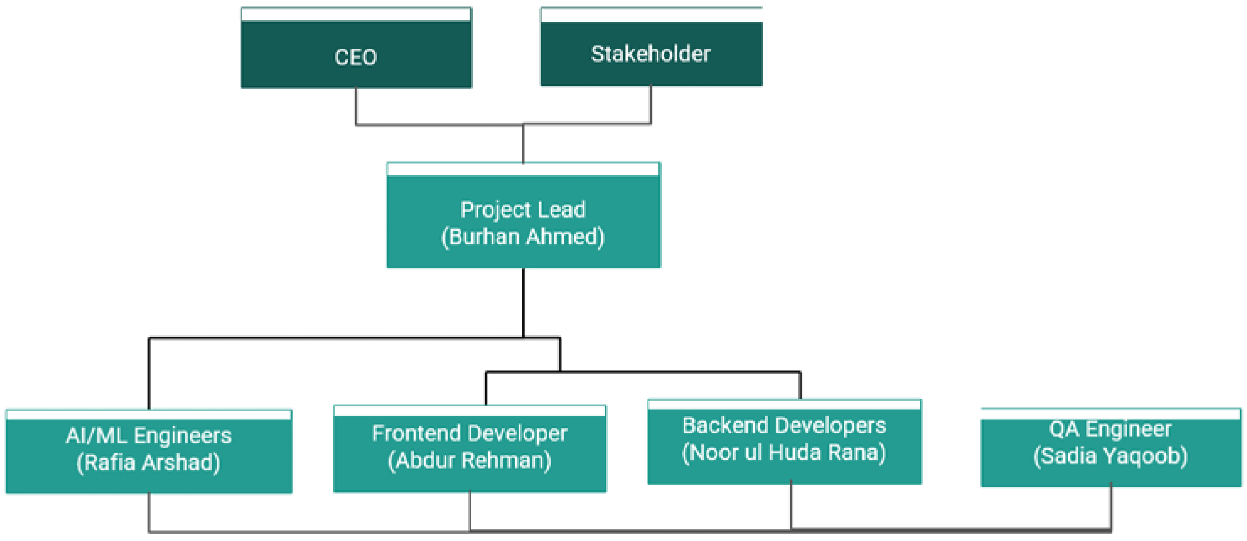

CEO
Stakeholder
Project Lead (Burhan Ahmed)
AI/ML Engineers (Rafia Arshad)
Frontend Developer (Abdur Rehman)
Backend Developers (Noor ul Huda Rana)
QA Engineer (Sadia Yaqoob)

Figure 4: Project Roles and team structure

3 Specific Requirements

3.1 3.1 Description and Priority

The proposed system, **AI Derma Check**, is a web-based application designed to allow users (patients) to upload images of visible skin conditions and receive AI-generated disease detection. The system leverages a YOLO11 model for disease detection and classification. Prior to receiving a full report, users must complete a payment process handled through a secure Express.js gateway. The backend is implemented using FastAPI and tested using Swagger UI and Postman. Payment processing is validated with Jest.

Key features and their priorities are listed in Table 2.

Table 2: System Features and Their Priority

Feature	**Description**	**Priority**
User Authentication	Enables user registration, login, and session management	Medium
Image Upload	Users can upload JPEG/PNG images of skin conditions	High
Image Cropping	Users can crop images before analysis for improved focus	Medium
Disease Detection (YOLO11)	Performs object detection on uploaded images to locate the disease region	High
Disease Classification	Predicts skin disease using a model trained on medical image datasets	High
Payment Gateway Integration	Secured payment handled through Express.js	High
PDF Report Generation	Provides downloadable report with disease info, confidence scores, care tips	High
System API Testing	Swagger UI and Postman used to test all API endpoints	Medium
Payment Testing	Jest used to simulate and validate payment functionality	Medium

3.2 3.2 Functional Requirements

- **FR1**: The system shall provide an interface for uploading skin images in PNG or JPEG format.
- **FR2**: The system shall allow users to crop the uploaded image before analysis.
- **FR3**: The system shall detect and extract the region of interest using a YOLO11 model.
- **FR4**: The system shall prompt users for payment before revealing results.
- **FR5**: The system shall integrate Express.js for secure payment processing.
- **FR6**: The system shall provide classification results with confidence metrics upon successful payment.
- **FR7**: The system shall generate a downloadable PDF report with relevant medical insights.

3.3 3.3 Non-Functional Requirements

- **Performance**: The system should deliver diagnostic results within 5 seconds of image submission.
- **Usability**: The user interface shall be responsive, accessible, and intuitive for non-technical users.
- **Security**: Payment, and user data must follow industry-standard encryption and validation protocols.
- **Scalability**: The backend (FastAPI) and database should handle concurrent requests efficiently.
- **Availability**: The system should maintain 99% uptime excluding scheduled maintenance.
- **Testing**: API endpoints shall be tested using Swagger UI and Postman; payment logic shall be verified via Jest unit tests.
- **Portability**: The application should run on any modern browser and support future deployment to cloud environments.

3.4 External Interface Requirements

3.4.1 User Interfaces

The AI Derma Check Web Application provides an intuitive and user-friendly interface accessible via modern web browsers. Key interface components include:

- **Image Upload Page:** Allows users to select and submit images of skin conditions (JPEG/PNG formats).
- **Analysis Display:** Shows a preview of the predicted disease along with a confidence score.
- **Payment Prompt:** Displays secure payment options via a web-based Express.js gateway.
- **Result Page:** Displays the final analysis and allows PDF download of the report.
- **Error Notifications:** Communicates image validation issues, payment failures, or upload errors.

3.4.2 Hardware Interfaces

The system is a web-based application and requires no specialized hardware. It can be accessed via:

- Desktop or Laptop with an internet browser (Chrome, Firefox, Edge).
- Mobile or tablet device with camera access and browser support.

3.4.3 Software Interfaces

- **Backend API:** Implemented using the `FastAPI` framework to serve RESTful endpoints for image analysis, payment verification, and PDF generation.
- **Payment Gateway:** Developed using `Express.js` to handle secure online payments.
- **Machine Learning Models:**
  - `YOLO11`: Used for object detection and initial skin disease prediction.
- **PDF Report Generator:** Backend integration to convert diagnostic results into downloadable PDF format.
- **API Testing Tools:**
  - `Swagger UI`: For testing and documenting FastAPI endpoints.
  - `Postman`: For manual API testing and verification during development.
- **Payment Testing Tool:**
  - `Jest`: Used to perform automated unit testing of the payment module developed with Express.js.

3.4.4 Communication Interfaces

The system communicates using standard web protocols and data formats:

- **Protocol:** HTTP/HTTPS for secure and reliable communication.
- **Data Format:** JSON for API requests and responses.
- **Endpoints:** RESTful APIs exposed by FastAPI and Express.js.

4 System Design

This section presents the design aspects of the AI Derma Check Web Application. It outlines how different components interact to ensure smooth functionality. The system is designed with modularity, scalability, and ease of use in mind. The diagrams included in this section provide visual representation of the internal and external working of the system.

4.1 System Architecture Diagram

The architecture of the system follows a modular design with separation between the frontend, backend, model server, and payment gateway.

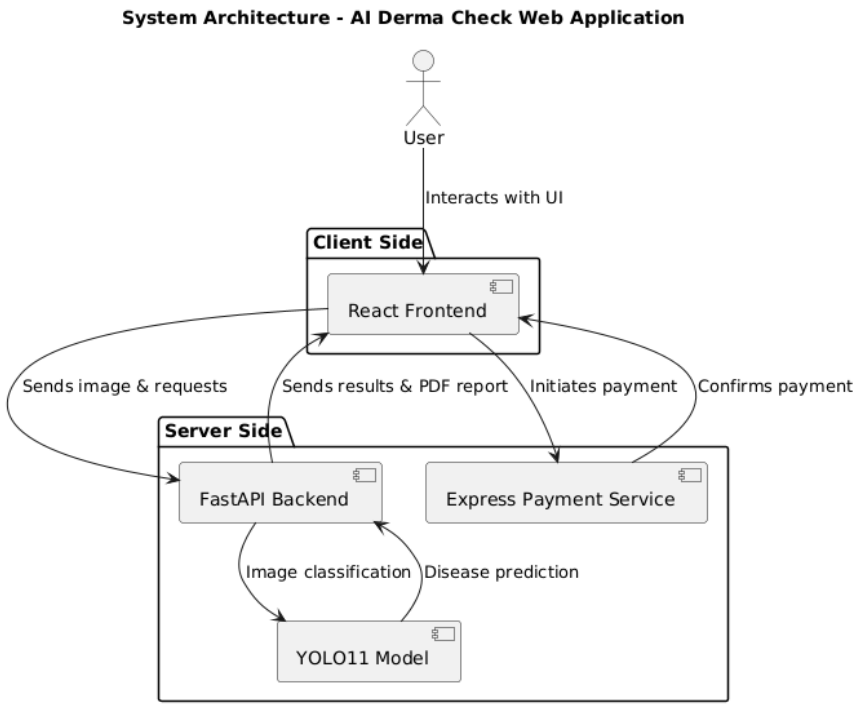


Figure 5: System Architecture of AI Derma Check Web Application

Description:

- **Frontend (React):** Provides the user interface for uploading images, viewing results, and managing payments.

- **FastAPI Backend:** Handles API requests, model interaction, image validation, and report generation.
- **YOLO11 Model Server:** Receives preprocessed images and returns prediction results.
- **ExpressJS Payment Service:** Manages secure payment handling and verification.

4.2 Context Diagram

The context diagram shows the high-level interaction of the system with external actors such as users and third-party services. Figure 6 illustrates the context diagram for the skin disease detection System.

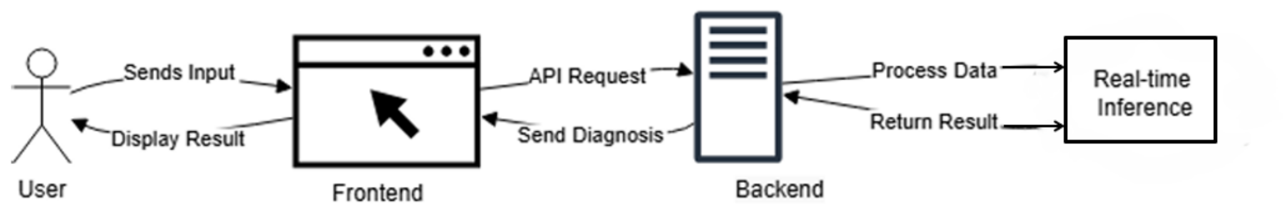


Figure 6: Context Diagram for AI Derma Checker

4.3 Use Case Diagram

The use case diagram describes the primary system functionality and how users and admin interact with it.

4.3.1 User Use Case Diagram

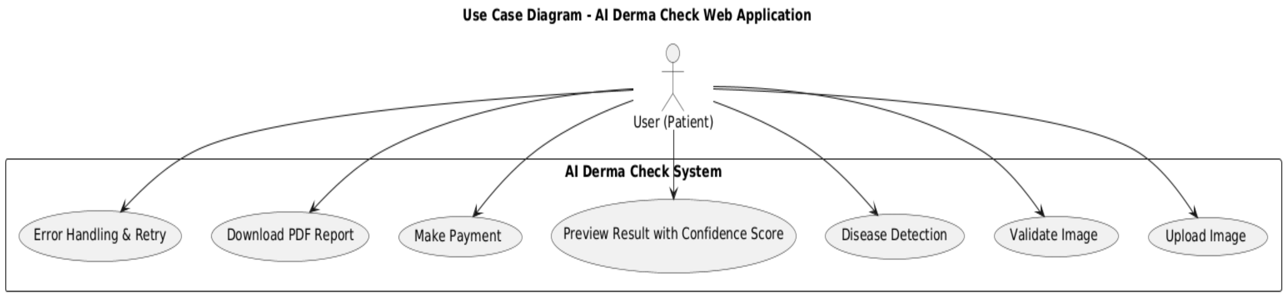


Figure 7: User Use Case Diagram

4.3.2 Admin Use Case Diagram

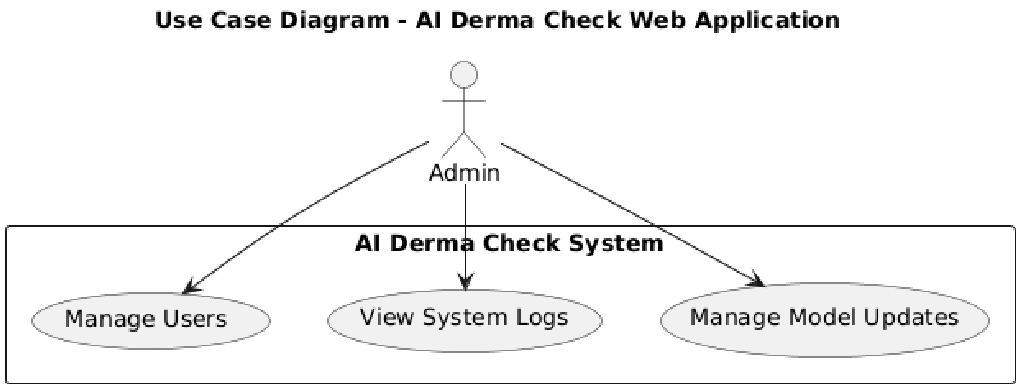


Figure 8: Admin Use Case Diagram

4.4 Class Diagram

The class diagram 9 shows the structure of the system and relationships among classes.

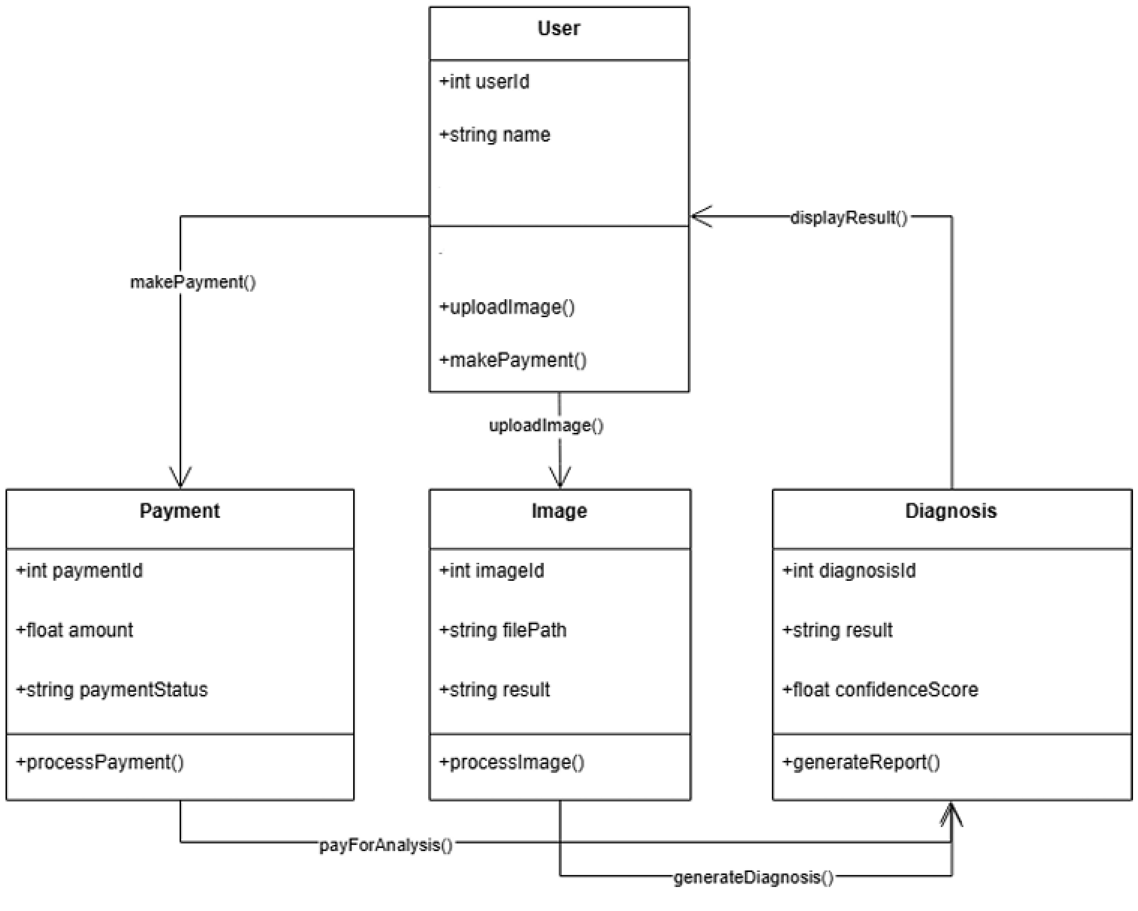


Figure 9: Class Diagram of the AI Derma Check System

4.5 Sequence Diagram

This diagram depicts the sequential interaction between components during a typical analysis and payment flow.

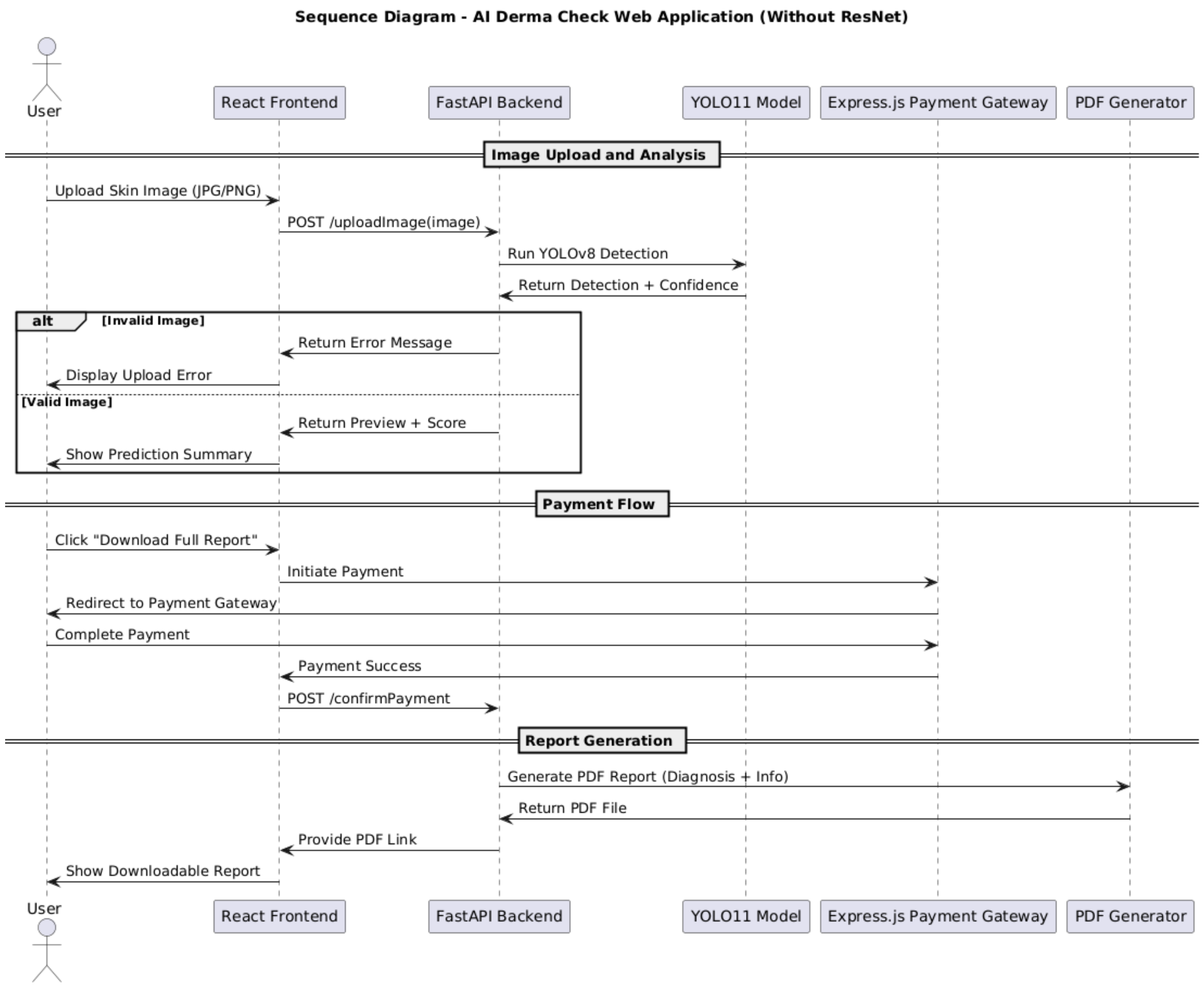


Figure 10: Sequence Diagram of Image Analysis and Report Generation Flow

5 System Implementation

This section outlines the implementation details of the AI Derma Check Web Application. It includes the selection of technologies, explanation of key modules, communication between system components, and testing strategies employed throughout the development process.

5.1 Technology Stack

The following technologies were used in developing different components of the system:

- **Frontend:** React.js was used to build a responsive and user-friendly interface, allowing users to interact with the system through image uploads, payments, and report downloads.
- **Backend API:** FastAPI, a modern and high-performance Python framework, was used for building RESTful APIs responsible for image processing, validation, AI integration, and report generation.
- **AI Model:** YOLO11, a real-time object detection model, was used for detecting skin conditions from the uploaded image.
- **Payment Gateway:** Express.js was integrated as a Node.js framework to handle secure payment processing.
- **Report Generation:** PDF reports were generated using libraries such as `ReportLab` containing disease predictions and related medical information.
- **Testing Tools:**
  - **Mnaual Testing:** Used for manually testing the drawbacks and issues.
  - **Swagger UI:** Used for testing and documenting REST API endpoints.
  - **Postman:** Used for manual testing of API responses and payloads.
  - **Jest:** Employed to write and run unit and integration tests for the Express-based payment module.

5.2 Modules Description

The application was modularly designed for scalability and maintainability. Below are the core functional modules:

5.2.1 Image Upload & Validation Module

Users upload images (JPEG/PNG) of the affected skin area. The backend validates:

- File format
- File size
- Image content (basic validation before passing to AI model)

5.2.2 Image Processing and Disease Detection Module

Once validated, the image is forwarded to the YOLO11-based AI model. The model detects the region of concern and classifies the skin condition. A prediction with a confidence score is returned to the frontend.

5.2.3 Payment Processing Module

To unlock full diagnostic details and the downloadable PDF report, users are prompted for payment:

- The payment API built using Express.js securely redirects to a payment gateway (e.g., Stripe).
- Payment status is confirmed through a callback mechanism.
- Jest is used for simulating and testing various payment states (success, failure, retry).

5.3 PDF Report Generation Module

Upon successful payment, the system dynamically generates a detailed PDF report containing:

- Disease Name
- Description and Symptoms
- Suggested Care Tips
- Confidence Score
- Date and Timestamp

A download link is provided on the frontend interface.

5.4 Error Handling Module

Robust error handling ensures that any failure (image rejection, model error, payment failure) is caught and a user-friendly message is displayed. The frontend provides retry options for both image upload and payment.

5.5 API Integration and Communication

5.5.1 Frontend and Backend Communication

The React frontend communicates with the backend via RESTful API endpoints using the Axios HTTP client. Examples include:

- `GET /api/check-live`
- `POST /api/create-checkout-session`
- `POST /api/predict-and-generate-report`

JSON is used as the standard data exchange format.

5.5.2 Security Measures

- APIs are protected.
- CORS policy and HTTPS encryption were implemented to secure data in transit.

5.6 Testing Strategy

Testing was conducted at different levels to ensure system reliability and correctness.

5.6.1 Frontend Testing

Manual testing was done to check:

- Image upload and cropping flow
- Response display logic and UI feedback
- Responsiveness across different devices

5.6.2 API Testing

- Swagger UI was used for automated API documentation and testing during development.
- Postman was used to test API endpoints with various input formats, headers, and response validations.

5.6.3 Payment Testing

- The payment workflow was tested using **Jest**, focusing on:
  - Valid payment confirmation
  - Invalid/missing payment response
  - Timeout or API error handling

5.7 Deployment

The system was deployed using:

- **Frontend:** Vercel
- **Backend:** Render for FastAPI + Express.js services

Continuous Deployment (CI/CD) was configured for automated updates from the GitHub repository.

6 Testing

A rigorous testing strategy was employed to validate the functionality, reliability, and performance of the AI Derma Check Web Application. Multiple complementary testing approaches were used, including automated API tests, manual UI/UX checks, end-to-end workflows, and specialized payment validation.

6.1 API Testing

6.1.1 Swagger UI

During development, **Swagger UI** provided an interactive interface to explore, execute, and document all FastAPI endpoints. Key activities included:

- Verifying endpoint contracts (paths, parameters, request/response schemas).
- Sending sample requests with valid and invalid payloads (e.g., image files, JSON bodies).
- Inspecting HTTP status codes, response headers, and error messages for consistency.
- Ensuring that authentication functioned correctly.

6.1.2 Postman Testing

Postman was used for comprehensive, iterative API validation:

- Automating collections of test cases for core workflows (image upload $\rightarrow$ prediction $\rightarrow$ report generation).
- Parameterizing requests to simulate a variety of inputs (edge cases, large file uploads).
- Asserting response times to meet performance targets (5 seconds per prediction).
- Generating test reports and monitors to catch regressions during development.

6.1.3 API Unit Testing with Jest

Jest was used to write unit and integration tests for the Express.js payment module:

- Mocked external payment gateway APIs to simulate success, failure, and timeout conditions.
- Verified that payment status was recorded and propagated back to the FastAPI backend via secure endpoints.
- Measured code coverage to ensure that edge cases in the payment flow were exercised.

6.2 Manual Testing

Manual testing ensured end-user usability and functional correctness across the entire application stack:

- **Frontend UI/UX:**
  - Verified layout and responsiveness on desktop, tablet, and mobile browsers (Chrome, Firefox, Edge, Safari).
  - Tested image upload component, prediction display, payment prompt, and PDF download.
  - Confirmed accessibility features (labels, focus states, error messages).
- **Backend Functionality:**
  - Confirmed correct handling of valid and invalid requests (format errors, missing tokens).
  - Simulated high-load scenarios to observe stability under concurrent uploads.
  - Inspected server logs for unhandled exceptions, memory leaks, or performance bottlenecks.
- **Error Handling:** Ensured that all failure modes (invalid image, network timeouts, payment declines) produced clear, actionable feedback and allowed the user to retry.

6.3 End-to-End Testing with Cypress

Cypress was employed to automate full user journeys in the browser, combining frontend interactions with real backend responses:

- Automated, image upload, and cropping workflows.
- Validated that a prediction summary appeared before payment.
- Simulated payment initiation and redirection to the Express.js gateway stub.
- Verified PDF report generation and download link presence.
- Ran tests as part of the continuous integration pipeline to catch UI regressions.

6.4 Payment Gateway Testing

Given the critical nature of payments, dedicated tests were executed in both sandbox and mock environments:

- **Successful Transactions:** End-to-end verification of payment capture and callback to the application.
- **Failure and Retry:** Simulated card declines, network errors, and user-initiated cancellations to confirm graceful recovery.
- **Security Checks:** Ensured PCI-compliant handling of payment tokens and encrypted transmissions (HTTPS).

6.5 Testing Summary

By combining automated API tests (Swagger UI, Postman, Jest), end-to-end browser tests (Cypress), and thorough manual validation, the AI Derma Check Web Application was verified for functional correctness, performance, security, and user experience before deployment. Continuous integration pipelines were configured to run all tests on each pull request, ensuring ongoing code quality and rapid feedback to the development team.

7 Results and Evaluation

This section presents the necessary results obtained from the AI Derma Check Web Application, including frontend interface verifications, payment testing, and quantitative evaluation of the underlying YOLO11 model. Where appropriate, images and plots are provided to illustrate performance and user experience. Detailed Test cases, Automated testing, user manual, Change Request & Bug/Defect Report, and API documentation is also designed and provided with this document.

7.1 Frontend Results

7.1.1 Home Page

Figure 11 shows the landing screen of the application. The homepage provides clear navigation to the upload workflow, user authentication controls, and brief instructions.

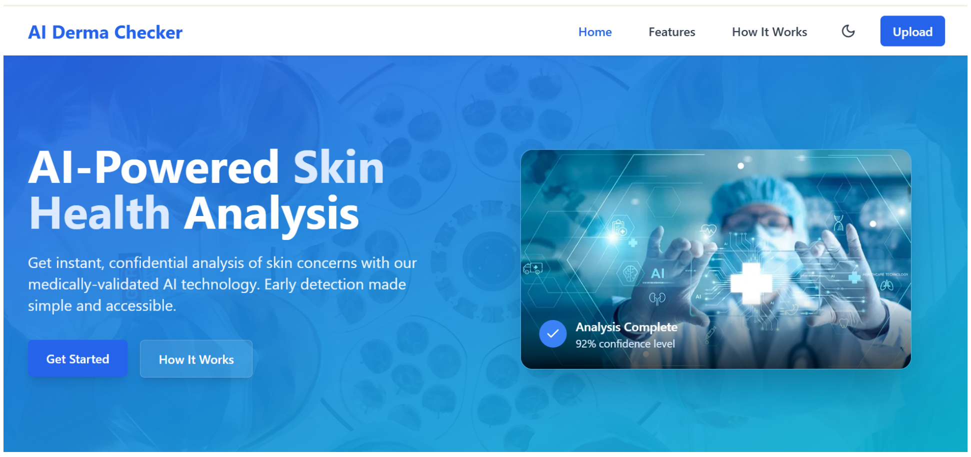

AI Derma Checker
Home
Features
How It Works
Upload
AI-Powered Skin Health Analysis
Get instant, confidential analysis of skin concerns with our medically-validated AI technology. Early detection made simple and accessible.
Get Started
How It Works
Analysis Complete
92% confidence level

Figure 11: Home Page of AI Derma Check Web Application

7.1.2 Upload Cropping Page

Figure 12 illustrates the image upload interface, including the file chooser and interactive cropping tool. Users can adjust the crop window to focus on the lesion before analysis.

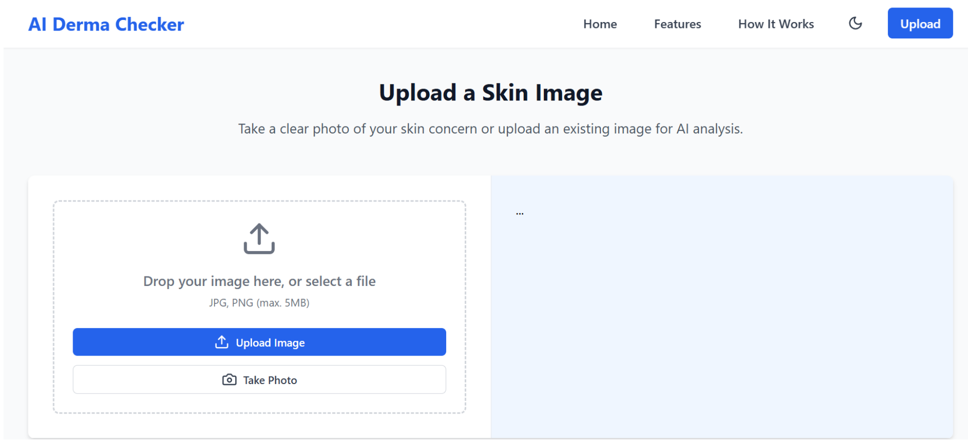
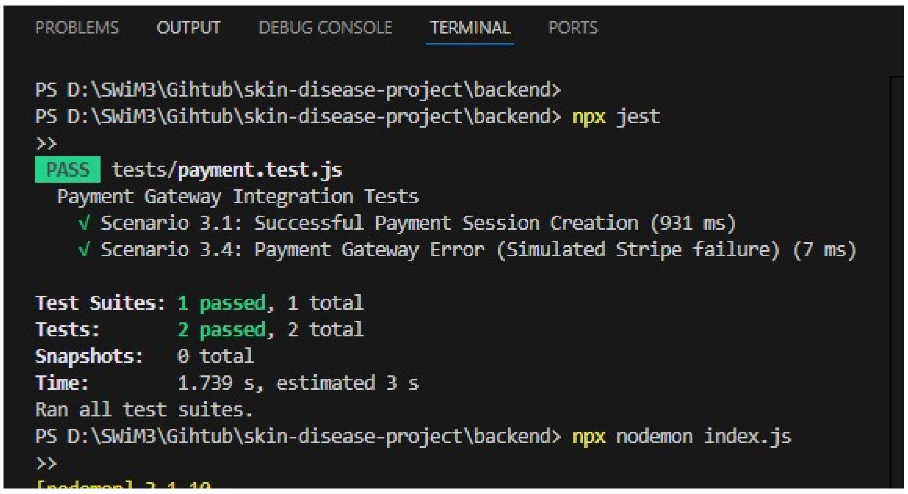

AI Derma Checker
Home
Features
How It Works
Upload
Upload a Skin Image
Take a clear photo of your skin concern or upload an existing image for AI analysis.
Drop your image here, or select a file
JPG, PNG (max. 5MB)
Upload Image
Take Photo

Figure 12: Image Upload and Cropping Interface

7.2 Payment Testing Results

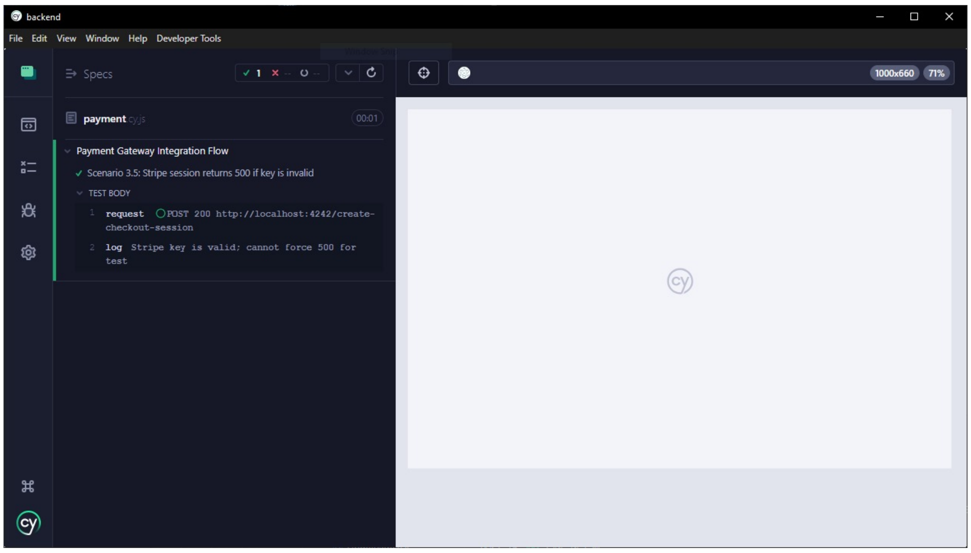
To verify the payment integration, test transactions were performed in the environment. Figure 14 shows a successful payment confirmation screen, indicating that the Express.js gateway is correctly invoked and returns status feedback to the frontend.



```
PS D:\SWiM3\Gihtub\skin-disease-project\backend>
PS D:\SWiM3\Gihtub\skin-disease-project\backend> npx jest
>>
 PASS  tests/payment.test.js
  Payment Gateway Integration Tests
    √ Scenario 3.1: Successful Payment Session Creation (931 ms)
    √ Scenario 3.4: Payment Gateway Error (Simulated Stripe failure) (7 ms)

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.739 s, estimated 3 s
Ran all test suites.
PS D:\SWiM3\Gihtub\skin-disease-project\backend> npx nodemon index.js
>>
[nodemon] 3.1.10
```

Figure 13: Payment Gateway Confirmation Screen



payment.cy.js 00:01

Payment Gateway Integration Flow

✓ Scenario 3.5: Stripe session returns 500 if key is invalid

TEST BODY

```
request  POST 200 http://localhost:4242/create-checkout-session
log  Stripe key is valid; cannot force 500 for test
```

Figure 14: Payment Gateway Confirmation Screen

7.3 Model Performance

7.3.1 Confusion Matrix

The performance of the YOLO11 model on the test set is summarized by the confusion matrix in Figure 15. This matrix shows the true vs. predicted class counts for each of the ten skin disease categories.

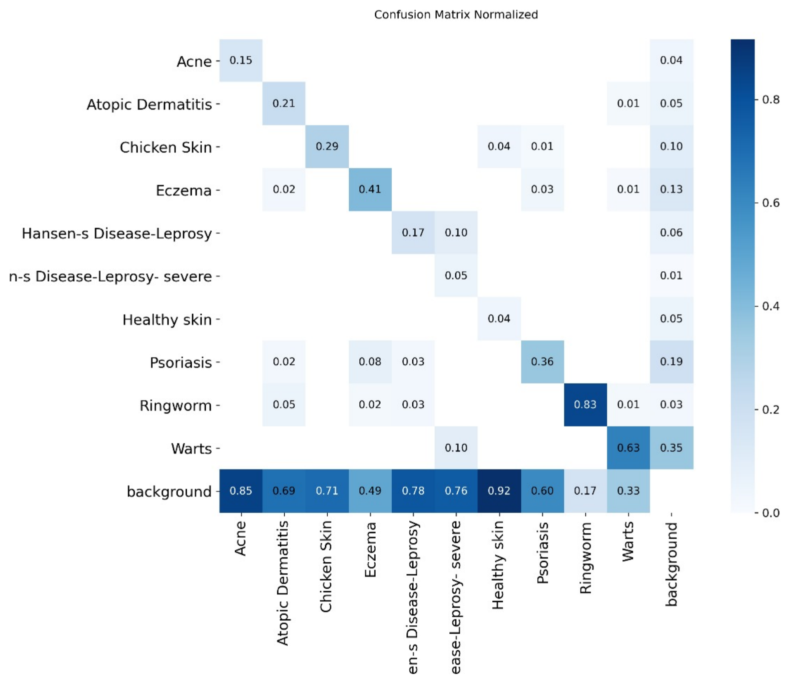
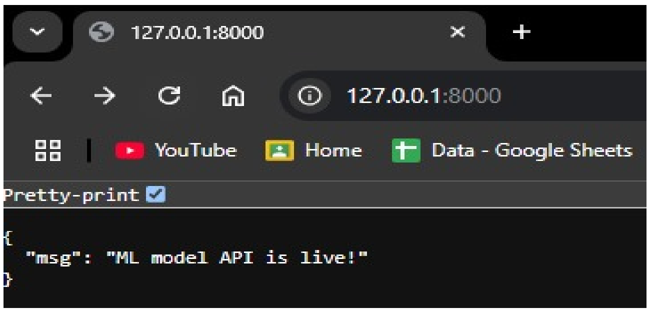


Figure 15: Confusion Matrix of YOLO11 Model on Test Data

These results meet the performance requirement of inference time and classification accuracy.

7.4 API Response

During API testing with Swagger UI and Postman, the `/api/get` endpoint returned a statement which indicates the local host is requesting correctly.

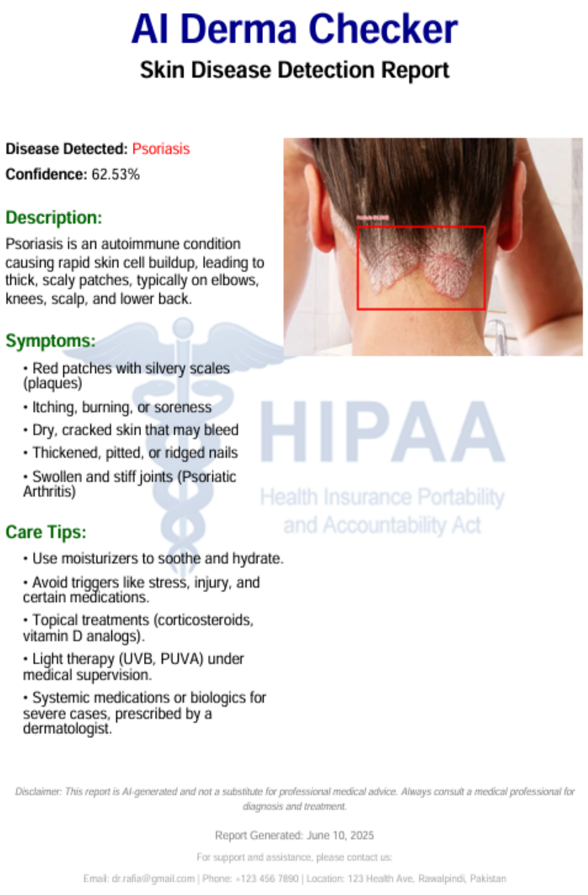


```
{
  "msg": "ML model API is live!"
}
```

Figure 16: API Response

7.5 Sample PDF Report Output

After successful payment, the backend generates a detailed PDF report encapsulating the diagnostic results and related information. Figure 17 shows a representative page from the report.



AI Derma Checker

Skin Disease Detection Report

Disease Detected: Psoriasis

Confidence: 62.53%

Description:

Psoriasis is an autoimmune condition causing rapid skin cell buildup, leading to thick, scaly patches, typically on elbows, knees, scalp, and lower back.

Symptoms:

- Red patches with silvery scales (plaques)
- Itching, burning, or soreness
- Dry, cracked skin that may bleed
- Thickened, pitted, or ridged nails
- Swollen and stiff joints (Psoriatic Arthritis)

Care Tips:

- Use moisturizers to soothe and hydrate.
- Avoid triggers like stress, injury, and certain medications.
- Topical treatments (corticosteroids, vitamin D analogs).
- Light therapy (UVB, PUVA) under medical supervision.
- Systemic medications or biologics for severe cases, prescribed by a dermatologist.

Disclaimer: This report is AI-generated and not a substitute for professional medical advice. Always consult a medical professional for diagnosis and treatment.

Report Generated: June 10, 2025

For support and assistance, please contact us:

Email: dr.rafia@gmail.com | Phone: +123 456 7890 | Location: 123 Health Ave, Rawalpindi, Pakistan

Figure 17: PDF Generated

8 Discussion

The frontend components function as expected across devices, providing intuitive workflows for image upload, cropping, and report retrieval. Payment integration was seamless, with Jest-based tests confirming robust handling of success and failure scenarios. The YOLO11 model demonstrates strong classification performance on diverse skin conditions, though rare classes (e.g., Hansen's Disease) indicating potential areas for dataset augmentation in future work.

9 Conclusion

The AI Derma Check Web Application delivers a comprehensive, end-to-end solution for preliminary skin disease screening. By integrating a YOLO11 detection model (trained on the Roboflow "Skin Disease" dataset) with a responsive React frontend and FastAPI backend, the system enables users to upload and crop lesion images, receive rapid predictions with confidence scores, complete secure payments via an Express.js gateway, and download detailed PDF reports. Rigorous testing including Swagger UI and Postman for API validation, Jest for payment module unit tests, Cypress for end-to-end UI workflows, and manual usability checks—ensured the application's reliability, performance , and security (HTTPS). Overall, the project successfully meets its objectives of enhancing accessibility to dermatological insights, maintaining high accuracy across ten skin conditions, and providing a user-friendly interface that encourages early self-assessment and timely professional consultation.

References

[1] Stripe Documentation,
`https://stripe.com/docs`

[2] Swagger UI — API Documentation Tool,
`https://swagger.io/tools/swagger-ui/`

[3] Postman — API Development Environment,
`https://www.postman.com/`

[4] Cypress Documentation,
`https://www.cypress.io/`

[5] GitHub — Development Platform,
`https://github.com/`

[6] Jira Software — Issue & Project Tracking,
`https://www.atlassian.com/software/jira`