Patients Cases Similarity (Skin Smart App)

[1]Yarragudi Akshath Kumar Reddy, [2]Perrisetty Uday Kiran, [3]P Abhinay Kumar, [4]P Jaswant

Guide: Dr. P Sudha Ass.Prof

Computer Engineering, Presidency University Bengaluru

ABSTRACT- This project entails the development of a mobile application for skin disease detection and management, utilizing Android XML for frontend design and Kotlin for backend functionality. The application is structured to serve three primary user roles: Admin, Hospitals, and Users, each with distinct capabilities. For the Admin, the application provides secure login, functionalities to add and manage hospital details, and view registered users, ensuring efficient oversight and administration of the system. Hospitals can log in to their dedicated interface to view patient appointments, facilitating streamlined appointment management and patient care. Users, the core beneficiaries, can log in to access various features aimed at improving skin health management. The application enables users to scan and detect skin diseases using advanced image recognition technology. Additionally, users can view nearby hospitals, add appointments with dermatologists, and maintain a history of their medical consultations and treatments. This feature-rich application ensures a seamless and integrated experience for managing skin health, from initial diagnosis to ongoing care. Overall, this mobile application aims to provide a comprehensive solution for skin disease management, leveraging the robustness of Kotlin for backend processes and the flexibility of Android XML for an intuitive user interface.

KEYWORDS: Mobile application, Android.

I. INTRODUCTION

In recent years, advancements in mobile technology and artificial intelligence have significantly transformed the healthcare industry. The development of mobile applications for health management has enabled individuals to access healthcare services more efficiently and conveniently. One such critical area of health that can benefit from technological innovation is dermatology. Skin diseases affect millions of people worldwide, necessitating a reliable, accessible, and efficient method for detection and management. This project aims to address this need by developing a comprehensive mobile application for skin disease detection and management, utilizing the Android platform with Kotlin for backend functionality and Android XML for frontend design. The application is designed to cater to three primary user roles: Admin, Hospitals, and Users. Each role is equipped with distinct capabilities to enhance the overall functionality and user experience of the system. Admins are provided with a secure login interface and functionalities to add and manage hospital details and view registered users.

II. RESEARCH ELABORATION

2.1 Motivation

This project aims to improve the care of dermatology through the use of mobile technology and AI in the management of skin diseases. It seeks to fill a gap in terms of easily accessible, efficient tools to identify and treat skin conditions for patients as well as for healthcare providers.

2.2 Problem Statement

Current methods for the management of skin diseases are inefficient because they still require manual processing that will inevitably delay diagnosis, treatment, and appointment schedules. Patient records are not kept centrally, and it's thus hard to gain access to specialized care.

2.3 Project Objective

The project will be based on the development of a mobile application that uses Android XML and Kotlin for early detection of skin diseases through image recognition, easy scheduling of appointments, and easy access to specialized healthcare services, which may improve the overall outcomes for skin health.

2.4 Scope

The app will feature secure logins for Admins, Hospitals, and Users, with functionalities including skin disease detection, hospital management, appointment scheduling, and tracking patient history. It will integrate advanced image recognition with a user- friendly interface to improve accessibility in healthcare.

2.5 Project Introduction

Develop a mobile application for skin disease detection and management using Android and Kotlin. The application will be used in three different roles: Admin, Hospitals, and Users. Admins will manage hospital details and user data. Hospitals will manage appointments while the users get access to skin disease detection, scheduling of appointments, and a record of their medical history. Users can now diagnose skin diseases from the comfort of their homes through advanced image recognition technology. The application makes sure of easy and streamlined care and easier access to dermatologists.

III. SYSTEM ANALYSIS

3.1 Current System:

The current systems associated with the management of skin diseases hardly use much of the technology for early detection and management. This causes the treatment to be delayed. Many times, scheduling appointments and tracking patient history is cumbersome, either on paper or through simple digital systems. Besides, finding nearby hospitals that specialize in such issues is a hassle, which does not integrate well or have user-friendly interfaces to provide holistic care.

3.2 Drawbacks

•Manual diagnosis and scheduling of treatment depends on which causes delay and inefficiency in patient care.

•There is no centralized digital record, which makes tracing the history of patients and managing appointments cumbersome.

•Difficulty in finding close-by specialized hospitals and no integrated user-friendly interface results in delayed and inefficient treatment.

3.3 Proposed System

The proposed system is a mobile application designed to provide comprehensive management of skin diseases. The frontend is Android XML, and the backend is written in Kotlin. There are three different user roles: Admin, Hospitals, and Users. Admin can log in; add hospitals; see a list of users. A Hospital can log in, thus viewing his patient's appointment.

Users can also login to scan for possible diseases in their skin due to image recognition technology available on the application. User also gets a list of close-by hospitals, as well as a scheduling of consultations history. This system looks for accelerating detection and treatment of the skin diseases. It is a whole single application which makes it comprehensive and user-friendly interface for users. Here, we're using the algorithm of the Mobile net. The very same model here is going to process with which the picture is to be detected

3.4 Advantages

• Android XML would provide an intuitive and an accessible frontend to all users.

• Users will get nearby hospitals, scheduled appointments, and detailed history on treatments.

•The backend of Kotlin provides high performance and smooth feature integration with a promise of a stable application experience.

IV. REQUIREMENT ANALYSIS

4.1 Function and non-functional requirements

Requirement analysis is a critical process for assessing the success of a system or software project. Requirements are categorized into functional and non-functional types.

Functional Requirements refer to the basic features the system must provide, directly requested by the end user. These requirements define the system's inputs, operations, and expected outputs. They are essential for the system's functionality and are visible in the final product.

Examples of functional requirements:

1. User authentication during login

2. System shutdown in case of a cyber- attack

Non-functional requirements: These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security

Examples of non-functional requirements:

1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
2) The processing of each request should be done within 10 seconds
3) The site should load in 3 seconds whenever of simultaneous users are > 10000

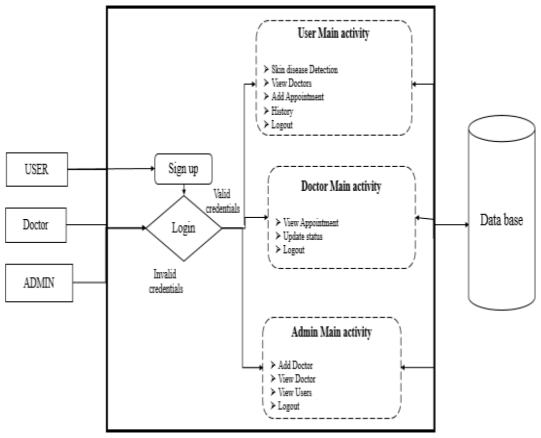
4.2 Hardware Requirements

Processor	- I3/Intel Processor
RAM	- 8 GB
Hard Disk	- 1TB

4.3 Software Requirements

Operating System	-Windows 10 JDK - java
Plugin	- Kotlin
SDK	- Android
IDE	-Android studio
Database	- sql

4.4 Architecture



V. SYSTEM DESIGN

5.1 Input Design

Input design binds the user to the information system; it handles pre-processing of data for handling. It assures that input into the system can be provided efficiently and reduces error to ensure security and privacy. Key points include;

- What needs to be entered
- How to organize or code the entered data

Provide mechanisms for input validation and error-handling 5.2 Objectives

Convert user input descriptions into a computer-based system which ensures accuracy of the data

Develop user-friendly screens that can make simple data entry and validation possible.

Develop designs that guide the user, hence not likely to err about input

5.3 Output Design

Output design ensures information from the system is communicated to the user. It will decide the form in which results get generated and ensure that outputs are intelligible as well as useful.

It includes:

- Convey past, current or future information.
- Signaling important events or actions
- Confirming or triggering an action

5.3 UML Diagram

UML is a standard language for modeling object-oriented software systems. It uses graphical notations to represent the software structures and behaviors and enables developers to communicate more easily. UML contains a Meta-model and a notation for developing software models.

5.4 Goals of UML

Provide a visual modeling language for software development

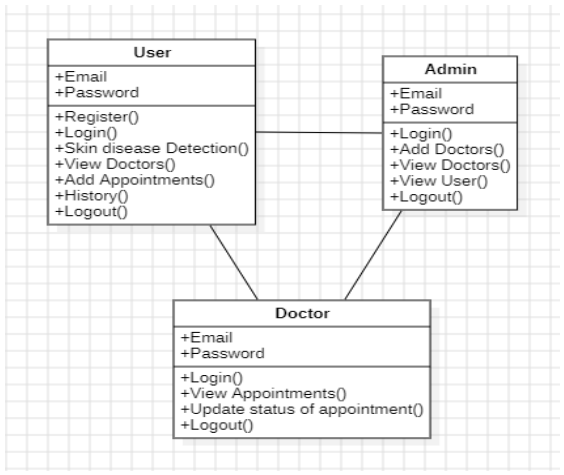
Be independent of programming languages and development processes

Support object-oriented tools and concepts

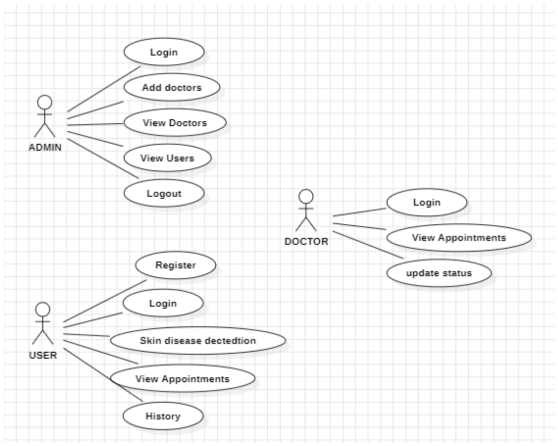
Integrate best practices for modeling complex systems

5.5 Class Diagram

A class diagram in UML describes the structure of a system by showing classes, their attributes, methods, and relationships. It provides a visual representation of how data and functionality are organized within a system.

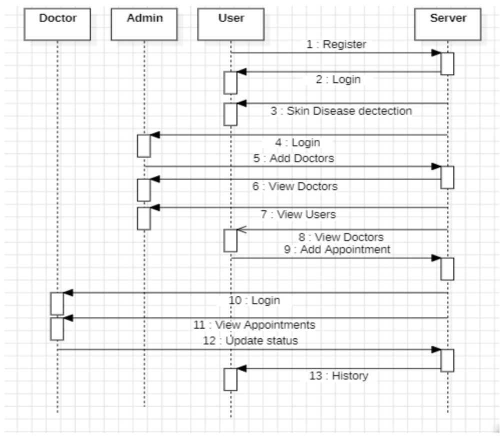


performed in the system by each of the actors and can further be used to represent roles played by the actors within the system.



5.6 SEQUENCE DIAGRAM:

In UML, a sequence diagram is an interaction diagram that illustrates how processes interact with each other and the order in which these interactions occur. One type of Message Sequence Chart, it is also called event diagrams or timing diagrams.



5.7 USE CASE DIAGRAM:

6 IMPLEMENTATION AND RESULTS

A use case diagram in UML is a behavioral diagram that graphically displays the functionality of a system. It involves actors, which are usually users or external systems and their goals, use cases, and relationships among them. The diagram shows the functions that are

USERS: Users can register, login, scan skin diseases using image recognition, view nearby hospitals, add appointments, and maintain a medical history all through a user-friendly mobile application.
ADMIN: Admins can securely login, add new hospitals to the system, and view registered users' details using the mobile application interface.

HOSPITAL: Hospitals can log in securely and access their dedicated interface to view scheduled appointments made by users through the mobile application.

VI. ANDROID ENVIRONMENT

7.1 Installation of Software:

How to Install Android Studio on Windows:

Download from the official site.

Run the installer and choose "Standard" or "Custom" installation.

Choose components like Android Studio, SDK, and AVD.

Installation location and Start Menu folder.

Complete the installation and open Android Studio.

Installation of SDK, download components, and set up emulator.

Start developing Android applications.

7.2 Software Development Life Cycle (SDLC - Agile):

Agile is the iterative model of software development focusing on short iterations, fast building, and customer participation. Agile SDLC includes phases such as requirement gathering, design, coding, unit testing, and acceptance testing. There is support for adaptive planning with rapid delivery and importance is placed on communication, collaboration, and swift delivery of working software.

Android Software Environment Key things include:

Operating System: Windows, macOS, or Linux IDE: Android Studio

Android SDK and JDK AVD for emulation Version Control (e.g., Git)

Gradle for build automation Physical device testing

Android supports hardware features like GPS, camera, and sensors. Libraries for graphics, media, and networking.

Architecture of Android:

Linux Kernel: Core services such as memory management.
HAL: Interface for hardware components

ART: Runtime for Java and Kotlin applications

Libraries: Core system functions (for instance, OpenGL, SQLite.)

Java API Framework: A high-level API for the UI, data storage, connectivity.

Application Framework: Activities and Services

UI Toolkit: Views and Layouts - UI components.

System Apps: Pre-installed apps like Phone, Contacts.

Hardware Running Android:

Android supports ARM architecture and works on various devices like phones, tablets, and TVs. Android applications are packaged as APKs and run in isolated processes for security.

Key Android Components:

Activities: UI screens.

Fragments: Modular UI components. Services: Background processes.

Broadcast Receivers: Respond to system messages.

Content Providers: Data sharing between apps. SQLite: Database management.

Creating an Android Emulator Device: Open Android Studio and go to AVD Manager.

Create a new virtual device, select hardware profile, system image, and configure settings.

Launch the emulator and test your app.

VII. SYSTEM STUDY AND TESTING

8.1 Feasibility Study

Feasibility of the project is determined by carrying out a feasibility study in terms of impact and budget. There are three types of feasibility:

- Economic Feasibility: The system has to be affordable. Most technologies are free and only custom components cost a little.
- Technical Feasibility: Checks the feasibility of the system's technical

requirements without burdening available resources.

- Social Feasibility: Deals with acceptance by users through training to make them comfortable with the system.

System Testing

Testing finds out errors and ensures that the system satisfies all the requirements and expectations. Some of the testing types include;

- Unit Testing: verifies the individual program logic and outputs.
- Integration Testing: Ensures that the integrated components work.
- Functional Testing: verifies that the system meets business and technical requirements.
- System Testing: Tests the whole system to ensure it meets integrated requirements.
- White Box Testing: Testing based on knowledge of the system's internal structure.
- Black Box Testing: Tests without any information regarding how the system works.
- Testing Strategy and Objectives:
- Unit Testing: This tests the functionality at the component level.
- Integration Testing: This tests interaction between components.
- Acceptance Testing: This test whether the system meets user requirements.

All tests were passed without defects

VIII. CONCLUSION

In conclusion, this mobile application represents a significant advancement in the field of dermatology by integrating advanced image recognition technology and comprehensive healthcare management features. By utilizing Kotlin for backend development and Android XML for frontend design, the application ensures robust performance and an intuitive user experience. Admins can efficiently manage hospital details and user registrations, hospitals can streamline patient appointments, and users can access reliable skin disease detection, nearby hospital information, and maintain their medical history.

This multi-faceted approach not only enhances the efficiency of dermatological care but also empowers users to take proactive steps in managing their skin health. The seamless integration of these features creates a holistic platform that bridges the gap between patients and healthcare providers, ensuring continuous and comprehensive care. Overall, this application stands as a testament to the potential of mobile technology in revolutionizing healthcare management.

IX. FUTURE ENHANCEMENT

Future enhancements for this mobile application could include the integration of telemedicine features, allowing users to have virtual consultations with dermatologists directly through the app. Implementing machine learning algorithms to continually improve the accuracy of skin disease detection based on user data and feedback is another potential enhancement. Additionally, expanding the database to include more diverse skin types and conditions would increase the application's reliability and inclusivity. The incorporation of personalized treatment plans and reminders for medication and follow-up appointments can further enhance user engagement and health outcomes. Adding multilingual support would make the application accessible to a broader audience, ensuring that language barriers do not hinder access to essential dermatological care. Finally, integrating with wearable health devices could provide real-time monitoring and alerts, offering a more comprehensive approach to skin health management. These future enhancements would further solidify the application's role as a leading tool in dermatological care

X. REFERENCES

- Abuzaghleh, O., Barkana, B. D., & Faezipour, M. (2023). Mobile-based skin lesion detection using deep learning and smart feature selection. In 2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI) (pp. 1-4). IEEE.
- Mishra, R., & Ghorai, S. (2022). Skin lesion detection using machine learning: a systematic review. Journal of Ambient Intelligence and Humanized Computing, 13(2), 1021-1034.
- Nguyen, Q. T., & Nguyen, B. P. (2022). AI- based mobile application for skin cancer detection using image processing techniques. Journal of Medical Imaging and Health Informatics, 12(2), 345-352.
- Moleanu, I., Diaconu, R., & Marcu, L. (2021). Mobile application for skin cancer detection using artificial intelligence. In 2021 International Conference on e-Health and Bioengineering (EHB) (pp. 1-4). IEEE.
- Gupta, A., Tomar, D., & Gautam, A. (2021). Skin disease detection using convolutional neural network. Procedia Computer Science, 167, 2419-2428.
- Han, S. S., Kim, M. S., Lim, W., Park, G.
H., Park, I., & Chang, S. E. (2020).
Classification of the clinical images for benign and malignant cutaneous tumors using a deep learning algorithm. Journal of Investigative Dermatology, 140(7), 1538-

1546.

- Tschandl, P., Rinner, C., Apalla, Z., Argenziano, G., Codella, N., Halpern, A., & Kittler, H. (2020). Human–computer collaboration for skin cancer recognition. Nature Medicine, 26(8), 1229-1234.
- Liu, Y., Jain, A., Eng, C., Way, D. H., Lee,
K., Bui, P., & Coz, D. (2020). A deep learning system for differential diagnosis of skin diseases. Nature Medicine, 26(6), 900-
908.
- Esteva, A., Kuprel, B., Novoa, R. A., Ko,
J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.
- Brinker, T. J., Hekler, A., Enk, A. H., Berking, C., Haferkamp, S., Hauschild, A.,
... & von Kalle, C. (2019). Deep neural networks are superior to dermatologists in melanoma image classification. European Journal of Cancer, 119, 11-17.