**CHAPTER 1**

**INTRODUCTION**

**1.1 Introduction to the Project**

Accurate identification of skin problems and knowing which natural remedies to use can be difficult, especially in rural areas or when using traditional medicine. Usually, people rely on visiting a doctor or matching symptoms themselves, which can sometimes lead to wrong guesses and treatments that don’t work. Even though many plants are known to help with skin issues, most people don’t have an easy way to find the right plant or learn how to use it properly for their skin.

This project introduces HerbalLink, an intelligent, machine learning–powered mobile application that assists users in detecting common skin diseases and recommends suitable medicinal leaves as treatment options. Unlike existing solutions that rely on user descriptions or predefined medical databases, HerbalLink uses real-time image classification and user input to deliver personalized, nature-based healthcare guidance. Through a simple interface built with React Native and deployed via the Firebase, users can register, upload images of affected skin areas, or scan leaves to receive tailored disease identification, natural remedy suggestions, and detailed usage instructions.

The backend, developed using Flask and Python, leverages machine learning models trained on skin disease and leaf datasets. Image processing is handled using OpenCV, while classification tasks utilize CNN (Convolutional Neural Network) models built with TensorFlow. For the skin module, the system predicts conditions such as acne, eczema, or fungal infections based on image features and symptom inputs. For the leaf module, the system identifies the scanned leaf and retrieves a list of skin diseases it can treat, along with preparation and application guidance sourced from a curated database.

HerbalLink offers a new and complete way to care for your skin by combining traditional herbal treatments with modern AI technology. It helps people rely less on chemical medicines and gives them easy, natural options for treating skin problems. With just a few steps, users can get useful advice, making skin care more accessible and convenient. By using real-time image analysis and machine learning, HerbalLink not only improves the accuracy of skin condition detection but also ensures that users receive personalized herbal remedies based on their specific needs.

**1.2 Introduction to Technology used**

HerbalLink is a smart healthcare application that blends traditional medicine with modern technology to help users identify skin diseases and discover relevant medicinal herbs for treatment. This system leverages several technologies such as Python, Flask, MongoDB, Firebase, Convolutional Neural Networks (CNN), TensorFlow, OpenCV, and Flutter to deliver a seamless, intelligent, and accessible experience across platforms.

**1.2.1 Python with Flask**

Python with Flask serves as the core backend framework in the HerbalLink application. Python’s simplicity and rich ecosystem make it ideal for machine learning and web development tasks. Flask, a lightweight and modular web framework in Python, enables the creation of RESTful APIs that connect the mobile app to the backend logic and machine learning models. The backend receives images and user inputs from the frontend, processes them using Python scripts and CNN models, and sends back the predictions and medicinal leaf recommendations. This architecture ensures a fast and responsive interaction between the user and the system while maintaining flexibility and scalability for future enhancements.

**1.2.2 MongoDB**

MongoDB is a NoSQL database used in HerbalLink to store user information, scan history, and mapping data between diseases and medicinal leaves. Its document-based architecture allows the system to store flexible data structures, including JSON-like documents, which is particularly useful for applications that need to evolve and scale quickly.

**1.2.3 Firebase**

Firebase by Google is an all-in-one platform that makes it easy to deploy the entire project, including the frontend, backend, database, authentication, and file storage. Firebase Hosting is used to deploy the Flutter app, providing fast and secure delivery of static files. For backend logic, Firebase Cloud Functions typically run JavaScript, also interacts with external Python services to handle tasks such as image processing. Additionally, Firebase Storage allows secure storage of uploaded skin, leaf images and even trained machine learning model files. This unified setup simplifies deployment and management, making Firebase a recommended choice for HerbalLink.

**1.2.4 Convolutional Neural Networks (CNN)**

CNNs are the core technology behind the image classification feature of HerbalLink. They enable the automatic detection of skin diseases and identification of medicinal leaves from scanned images. CNNs are composed of three main layers:

**Convolution Layers**: These layers consist of several filters that scan across the input image to detect features such as edges, textures, and patterns. Each filter performs a 2D convolution operation on the image and produces a feature map. These filters act like neurons and learn to identify different features through training.

**ReLU Layers**: ReLU (Rectified Linear Unit) layers apply an activation function to introduce non-linearity into the model. The function is defined as f(x) = x when x >= 0, and f(x) = 0 when x < 0. It helps the model to learn complex patterns by allowing only positive values to pass through.

**Pooling Layers**: Pooling layers are used to reduce the dimensions of the feature maps. Max pooling is the most common type, where the highest value in a group of pixels is retained, and the rest are discarded. This reduces computational load and helps extract the most important features.

**1.2.5 TensorFlow**

TensorFlow is an open-source deep learning framework developed by Google. It is used to build, train, and deploy the CNN models used in HerbalLink. TensorFlow supports both CPU and GPU computation, making it suitable for running complex models efficiently. It is also integrated with tools like Keras to simplify the design of neural networks.

**1.2.6 OpenCV**

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. It is used in HerbalLink for preprocessing images, such as resizing, background removal, and noise reduction. This improves the quality of the input data for better prediction by the CNN model.

**1.2.7 Flutter**

Flutter is an open-source UI toolkit developed by Google for building beautiful, natively compiled applications from a single codebase. HerbalLink uses Flutter to develop the cross-platform mobile application, allowing deployment on both Android and iOS with the same code. Flutter provides smooth performance and flexible UI design, ensuring a user-friendly experience.

**1.3 unicodes**

In the HerbalLink application, Unicode ensures consistent and accurate representation of text data related to user inputs, such as selected symptoms and the names of medicinal plants. While the primary method of skin disease detection is through image scanning using machine learning, users are also required to choose their symptoms from a predefined list to support the prediction process. Additionally, after diagnosis, the app provides information about suitable medicinal leaves and instructions on how to use them—all in a textual format.

To maintain clarity and compatibility across various devices, operating systems, and platforms, HerbalLink uses Unicode for encoding and displaying this textual information. Unicode is a universal character encoding standard that assigns a unique code point to every character in nearly all of the world’s writing systems. Unlike older systems such as ASCII (which could only handle 256 characters and primarily supported English), Unicode uses standards like UTF-8 or UTF-16 to represent thousands of characters, including symbols, special characters, and scripts used globally.

Thus, while HerbalLink’s core prediction system is image-based, Unicode still plays a crucial role in standardizing the textual parts of the user interface, ensuring that all information, such as leaf usage instructions, is readable and reliable across devices and languages.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Introduction**

The literature survey for the HerbalLink project highlights advancements in skin disease detection using machine learning, particularly CNNs for image-based classification, and the digital integration of traditional medicinal knowledge. Existing solutions have successfully implemented both scanning of skin conditions to recommend suitable medicinal leaves, and scanning of leaves to display their uses in treating various skin diseases. HerbalLink builds on these innovations by combining AI-powered diagnosis with leaf recognition to recommend natural remedies and identify which conditions each medicinal plant can effectively treat.

**2.2 Literature Survey**

Sukhadia et al. (2021), [1] proposed a deep learning-based method for automated skin disease detection using the Fast R-CNN architecture. Their approach addresses limitations in traditional image classification techniques by integrating object detection and classification in a single framework. The system was trained on a custom skin disease dataset, and it demonstrated improved accuracy in detecting and localizing various skin conditions compared to basic CNN models. The Fast R-CNN model efficiently identifies multiple disease regions within dermatological images, even under challenging conditions such as low contrast and overlapping lesions. The authors also developed a user interface to simplify disease identification for non-expert users. Their results highlight the effectiveness of Fast R-CNN in medical image analysis, providing a foundation for real-time skin disease screening tools.

Alaa Haddad and Shihab A. Hameed (2018), [2] proposed a framework for automated skin disease detection using image processing and machine learning techniques. The system focuses on identifying multiple skin conditions—acne, psoriasis, melanoma, and heat rash—from mobile-acquired images. The methodology involves a sequence of preprocessing (using Gaussian and median filters), color space conversion (RGB to YCbCr), and segmentation (via K-means and Fuzzy C-means clustering) to isolate the skin region. Key features—both color and texture-based—are extracted and classified using a Support Vector Machine (SVM) with a radial basis function kernel. The proposed system achieved a classification accuracy of 90.09% with KNN and 82.5% with MSIM. A mobile interface was developed to capture images in real-time and initiate diagnosis, improving accessibility and user experience. This multi-disease detection approach offers significant advancement over prior models that focused on single diseases, enabling faster and more accurate dermatological assessments.

Esmaieeli Sikaroudi, A. Mohammad et al. (2021) [3] proposed a deep learning-based method for the automated detection of skin diseases using Fast R-CNN, a region-based convolutional neural network. Their approach leverages a dataset of labeled skin lesions to train a model that can accurately identify various types of skin diseases. The Fast R-CNN model outperforms traditional machine learning methods, achieving high accuracy in detecting skin abnormalities. The authors also implemented a preprocessing step to enhance the quality of skin images, improving detection performance under varying conditions. Experimental results showed a significant reduction in false positives compared to other existing techniques. The study's findings suggest that this deep learning-based approach has great potential in clinical practice for assisting dermatologists with skin disease diagnosis. A user-friendly interface was also developed to facilitate ease of use for medical practitioners.

A. Rajasekar et al. (2024) [4] developed an AI-based Skin Disease Detector integrating convolutional neural networks (CNN) and YOLOv8 for automated skin condition classification. The system processes both images and video frames through a series of preprocessing techniques such as contrast enhancement, segmentation, and hair removal to enhance lesion visibility and accuracy. Trained on a large dataset including the ISIC archive, the model achieved high performance metrics, including 90% accuracy and 0.85 AUC-ROC, demonstrating its diagnostic reliability. YOLOv8 enabled real-time lesion localization with 53 convolutional layers. Additionally, the system incorporates stress and fatigue analysis using physiological data. A robust user interface was also implemented to ensure accessibility and usability in varied lighting and clinical conditions. This approach supports efficient and accurate skin disease detection in both clinical and remote healthcare settings.

Miss Mukta Kamble et al. (2019) [5] propose a dual-stage skin disease detection system combining image processing and machine learning. The authors design a mobile-based application that utilizes computer vision techniques for image preprocessing and feature extraction, followed by a convolutional neural network (CNN) for classification. They categorize the detection process into two phases—image preprocessing and disease classification—and emphasize the system’s applicability in rural and resource-limited settings. The study reports an accuracy of up to 95% for detecting diseases like Psoriasis, Lichen Planus, and Pityriasis Rosea. Challenges such as varying skin tones and image quality are identified, with future work focusing on improving robustness and expanding disease coverage. This work highlights the potential of integrating clinically evaluated histopathological features with machine learning models for accessible and reliable dermatological diagnosis.

Nahida Tabassum and Mariya Hamdani (2014), [6] explored the use of medicinal plants in the treatment of skin diseases by documenting 31 species traditionally employed in herbal remedies. The study emphasizes plant-based treatments for conditions such as eczema, acne, psoriasis, and skin infections. It presents Aloe vera, Allium cepa, Azadirachta indica, Curcuma longa, and Ocimum sanctum as key examples with antibacterial, antifungal, and anti-inflammatory properties. The authors highlight the pharmacological potential of these plants and advocate for scientific validation and clinical trials. This work contributes to the growing interest in herbal dermatology and supports the development of plant-based therapeutic solutions.

Rajasekar A., Shouvik C., and Mariya H. (2024), [7] proposed a multiattribute deep convolutional neural network (CNN) approach for accurately detecting medicinal plants and identifying their applications in treating skin diseases. The study integrates multiple attributes such as texture, color, and shape in the CNN architecture to enhance classification accuracy. Additionally, the paper explores the therapeutic relevance of detected plants by linking them to specific skin disease treatments, contributing to both automated plant recognition and practical medicinal use. The proposed method demonstrates improved performance compared to traditional techniques and highlights the potential of deep learning in botanical and medical informatics. This work sets a foundation for further research on AI-driven plant-based healthcare solutions.

Tabassum and Hamdani (2025), [8] explored the traditional use of medicinal plants in Norway for treating skin diseases and cosmetic applications. The study documents a variety of native plants known for their dermatological benefits, such as anti-inflammatory, antimicrobial, and wound-healing properties. It highlights key species like Aloe vera, Calendula officinalis, and Hypericum perforatum, emphasizing their active phytochemicals and therapeutic effects. The paper discusses preparation methods and cultural significance, linking ethnobotanical knowledge with modern pharmacological findings. This research underscores the value of integrating traditional plant-based remedies with contemporary skin disease treatment strategies and promotes further investigation into their efficacy and safety.

Smith et al. (2023), [9] investigates the identification of traditional medicinal plant leaves using an effective deep learning model and a self-curated dataset. They introduce a customized convolutional neural network (CNN) designed to accurately extract features from leaf images. To overcome limitations of existing datasets, the study compiles a diverse self-curated dataset representing various traditional medicinal plants. The research emphasizes the importance of high-quality and diverse data in improving classification accuracy. Advanced image preprocessing techniques are also applied to enhance leaf segmentation and clarity. The paper highlights the significance of combining domain-specific datasets with tailored deep learning architectures for reliable plant identification. This work lays the foundation for further AI-based advancements in botanical classification and traditional medicine.

Kale et al. (2024), [10] investigates the identification of Ayurvedic medicinal leaves and the recommendation of home remedies using state-of-the-art deep learning algorithms and a comprehensive dataset. The study introduces an Android-based application designed to classify 115 distinct species of medicinal leaves from a dataset of 6,541 images. The system employs well-known pre-trained neural networks including CNN, VGG16, MobileNet, and Inception to accurately identify leaves. In addition to leaf classification, the research addresses disease identification through a symptom-disease dataset encompassing 35 symptoms and 20 diseases. The conventional train-test split method ensures reliable model evaluation, preventing overfitting. The study emphasizes the integration of leaf image recognition with symptom-based Ayurvedic medicine and home remedy recommendations. Comprehensive performance metrics including accuracy, precision, recall, F1-score, and confusion matrices are utilized to assess the model's effectiveness. This work contributes significantly to bridging traditional Ayurvedic knowledge with modern AI techniques, enabling automated medicinal leaf identification and personalized treatment suggestions.

Rakib et al. (2023), [11] propose an automatic recognition system for medicinal plants based on multispectral and texture features using a hidden deep learning model. The research focuses on leaf-based identification, collecting images from five different medicinal plant species in natural environments. The study applies Convolutional Neural Networks (CNNs), particularly the VGG16 architecture, achieving a classification accuracy of 95.48%. The model leverages both multispectral and texture features to improve robustness and precision in plant identification. Image preprocessing plays a vital role in enhancing classification performance. The system aims to facilitate the medical sector by enabling accurate, automated recognition of medicinal plants, which can assist in medicinal plant awareness and preservation. Evaluation metrics include accuracy and model reliability through deep learning algorithms, contributing to the integration of AI with botanical research.

**2.3 Summary of Literature Survey**

Table 2.1 shows the summary of literature survey done.

**Table 2.1: Observations of Literature Survey**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author Name** | **Title of Paper** | **Methodology used** | **Advantages** | **Future Work** |
| Sukhadia, H., Patel, K., & Desai, S. | Deep Learning-Based Automated Skin Disease Detection using Fast R-CNN | The study proposes a deep learning method for automated skin disease detection utilizing the Fast R-CNN architecture. This model combines object detection and classification into a unified framework, enabling the system to accurately identify and localize multiple skin disease regions within dermatological images. | The Fast R-CNN model offers high accuracy in detecting and localizing multiple skin diseases within a single image, even under challenging conditions like low contrast and overlapping lesions. It improves over basic CNNs by combining object detection and classification in one framework. | Future improvements include expanding the dataset to include a wider variety of skin diseases, optimizing the model for real-time and mobile applications, and integrating treatment suggestions based on detected conditions. These enhancements aim to create a more robust and accessible tool for early skin disease screening and diagnosis. |
| Alaa Haddad and Shihab A. Hameed | Skin Disease Detection System Based on Image Processing and Machine Learning Techniques | The authors developed a framework for detecting multiple skin conditions using image processing and machine learning. The system applies preprocessing techniques, converts images from RGB to YCbCr color space, and segments skin regions using K-means and Fuzzy C-means clustering. | This method demonstrates high accuracy, efficiently handling multiple skin diseases rather than focusing on a single condition. The use of mobile image acquisition combined with a user-friendly interface enhances real-time diagnosis capabilities, making dermatological assessments more accessible to the public. | Future directions include improving classification accuracy through deeper neural network integration, expanding the dataset to include more skin disease classes, and enhancing the mobile app for real-time cloud-based diagnosis and treatment recommendation. |
| Esmaieeli Sikaroudi, A. Mohammad et al. | Deep Learning-Based Automated Detection of Skin Diseases Using Fast R-CNN. | The authors proposed an automated skin disease detection system based on Fast R-CNN, a region-based convolutional neural network architecture. The method involves preprocessing of dermatological images to improve quality, followed by training the Fast R-CNN model using a dataset of labeled skin lesions. | The Fast R-CNN model demonstrated superior accuracy and a notable reduction in false positives compared to traditional machine learning methods. It effectively handled variations in image quality and disease appearance. | Future improvements could include expanding the dataset to encompass more rare and complex skin conditions, incorporating additional deep learning models for better segmentation, and integrating telemedicine features to enable remote diagnosis and consultation capabilities. |
| A. Rajasekar et al. | AI-Based Skin Disease Detector Using CNN and YOLOv8 for Image and Video-Based Diagnosis | The authors proposed an advanced AI-driven framework combining Convolutional Neural Networks (CNN) with the YOLOv8 object detection algorithm to classify and localize skin diseases from both static images and video frames. | The integration of YOLOv8 enables rapid and accurate localization of skin lesions in real-time, while the CNN enhances classification precision. Achieving 90% accuracy and 0.85 AUC-ROC, the system demonstrates strong diagnostic capability. | Future enhancements could involve integrating more diverse and real-world skin datasets to increase generalizability, adding support for rare skin conditions, and implementing cloud-based diagnostics for telemedicine deployment. |
| Miss Mukta Kamble et al. | Skin Disease Detection Using Image Processing and Machine Learning on Mobile Devices | The authors developed a two-phase system for skin disease detection on mobile platforms. The first phase involves image preprocessing using computer vision techniques to enhance quality and extract key features, while the second phase uses a Convolutional Neural Network to classify skin diseases. | The system achieved up to 95% accuracy in diagnosing skin conditions such as Psoriasis, Lichen Planus, and Pityriasis Rosea. Its mobile integration and lightweight design make it ideal for use in resource-constrained environments. | Future directions include improving model robustness to handle diverse skin tones and image artifacts, expanding the dataset to cover more skin conditions, and incorporating clinically validated histopathological features to improve diagnostic precision. |
| Nahida Tabassum and Mariya Hamdani | Plants used to treat skin diseases | The study involves an ethnobotanical and pharmacological review of 31 medicinal plant species traditionally used for treating various skin diseases. The authors analyze the medicinal properties and applications of these plants through literature surveys. | This work bridges traditional knowledge and modern science by documenting the effectiveness of medicinal plants like Aloe vera, Azadirachta indica, and Curcuma longa. It showcases the therapeutic potential of these herbs in managing skin ailments. | The authors suggest the need for scientific validation through rigorous pharmacological studies and clinical trials. Future research should focus on standardizing dosages, understanding active compounds, and developing commercial herbal formulations for dermatological use. |
| Rajasekar A., Shouvik C., and Mariya H. | Multiattribute Deep CNN for Medicinal Plant Detection and Therapeutic Application in Skin Disease Treatment | The authors proposed a deep learning framework using a multiattribute Convolutional Neural Network (CNN) that incorporates texture, color, and shape features for improved medicinal plant classification. | This integrated approach enhances both the precision of plant identification and the reliability of linking plants to their medicinal uses. It offers a scalable solution for botanical diagnostics and supports the development of AI-driven tools. | The study encourages further expansion of the dataset to include more plant species and diseases, incorporation of regional medicinal knowledge, and the integration of mobile applications for real-time detection. |
| Tabassum, N., and Hamdani, M. | Ethnobotanical Insights into Medicinal Plants Used for Skin Diseases and Cosmetics in Norway | The authors conducted an ethnobotanical survey documenting native medicinal plants traditionally used in Norway for dermatological and cosmetic purposes. The study includes species identification, preparation techniques, and an analysis of the active phytochemicals responsible for therapeutic effects. | This work bridges traditional herbal knowledge and contemporary medicine by validating plant-based treatments for skin diseases. It identifies effective, low-cost alternatives to synthetic drugs and promotes the cultural heritage of herbal practices. | Future research will involve clinical testing of the documented plant-based remedies to evaluate their safety and efficacy. The authors also suggest developing standardized extraction and formulation protocols, and integrating these natural remedies into modern dermatological treatments and skincare products. |
| Smith et al. | Identification of Traditional Medicinal Plant Leaves Using Customized Deep Learning Models. | The study introduces a customized convolutional neural network (CNN) designed to accurately extract features from images of traditional medicinal plant leaves. To address limitations in existing datasets, the authors compiled a diverse self-curated dataset representing various medicinal plants. | The research demonstrates that combining domain-specific, high-quality datasets with tailored deep learning architectures significantly improves classification accuracy. This approach enables more reliable identification of medicinal plants, which is crucial for both botanical research and traditional medicine applications. | Future directions include expanding the dataset diversity and size to cover more species, optimizing the CNN architecture further, and integrating the model into practical applications such as mobile apps for real-time plant identification and medicinal use guidance. |
| Kale et al. | Automated Identification of Ayurvedic Medicinal Leaves and Home Remedy Recommendation Using Deep Learning. | This study presents an Android-based application that classifies 115 species of Ayurvedic medicinal leaves using a dataset of 6,541 images. The system leverages pre-trained deep learning models such as CNN, VGG16, MobileNet, and Inception for accurate leaf identification. | The integrated approach combines leaf image recognition with symptom-based disease identification, enabling personalized Ayurvedic remedy recommendations. Using multiple pre-trained models improves classification accuracy and generalizability. | Future work may include expanding the dataset to cover more species and diseases, improving model efficiency for real-time deployment, and integrating user feedback to refine recommendation accuracy. Further development could also focus on incorporating multi-modal data to enhance diagnosis and remedy suggestions. |
| Rakib et al. | Automatic Recognition of Medicinal Plants Using Multispectral and Texture Features with Deep Learning | This study develops an automatic medicinal plant recognition system based on leaf images collected from five different species in natural settings. The approach integrates multispectral imaging and texture feature extraction to enhance classification. | Combining multispectral data with texture features increases model robustness and precision, achieving a high classification accuracy of 95.48%. The system automates medicinal plant identification, aiding in awareness and conservation efforts. | Future directions include expanding the dataset to incorporate more species and environmental variations, refining multispectral imaging techniques, and developing a user-friendly mobile application for field use. |
|  |  |  |  |  |

**2.4 Comparison with Existing Systems**

The Herballink system presents a novel advancement over existing herbal medicine applications and skin condition diagnostic tools by integrating machine learning with both leaf recognition and skin disease analysis. Unlike conventional solutions that typically rely on static databases or keyword-based search engines, Herballink introduces image-based diagnosis, enabling users to interact with the system through real-world inputs such as photos of leaves or skin conditions.

Traditional herbal identification platforms often require users to manually enter plant names, browse lists, or match physical features using textual descriptions or illustrations. This process is time-consuming and prone to human error. In contrast, Herballink uses leaf image processing and pattern recognition algorithms to identify medicinal plants accurately. This objective and automated approach significantly enhances both the speed and accuracy of identification, making it more accessible to non-experts.

Similarly, most existing systems for diagnosing skin conditions depend on user-filled symptom checklists or manual searches. Herballink offers a smarter alternative by allowing users to upload images of affected skin areas. The system then applies trained models to detect common dermatological conditions based on visual patterns such as color, texture, and shape. By incorporating optional symptom text input, the system can further refine its predictions, a feature rarely found in standard tools.

Another distinguishing feature is Herballink’s ability to cross-link plant properties with detected skin issues. Once a skin disease is recognized, the system suggests relevant Ayurvedic or herbal remedies derived from the identified plant dataset. This direct connection between problem and natural solution is lacking in most separate plant encyclopedia and skin care apps, which typically don’t integrate both domains into a unified flow.

Herballink respects user privacy by safely handling both images and any symptom details provided. The system only asks for the information needed to give accurate suggestions. Unlike other apps that collect a lot of personal or health-related data, Herballink keeps things simple and secure, so users can feel safe and confident while using it.

In a space where many applications focus narrowly on either botanical education and dermatological diagnosis, Herballink stands out by fusing the two with AI, offering a practical, user-friendly, and accessible tool. This system not only increases the accuracy of both plant and disease identification but also promotes the use of natural, accessible remedies, particularly benefiting rural and underserved communities.

**2.5 Proposed System**

The proposed system for the Herballink application combines machine learning, image analysis, and Ayurvedic knowledge to provide users with accurate, accessible, and natural health recommendations. The heart of the system is its Dual Image Processing Module, which processes both leaf images and skin condition photos uploaded by the user. These modules use advanced computer vision techniques to recognize the features of medicinal leaves, such as shape, texture, color, and analyze visible symptoms in skin images to detect common skin diseases. This scientific approach ensures precise identification and relevant treatment suggestions based on physical inputs rather than guesswork.

To make the system easy to use, a simple and intuitive interface will guide users step-by-step. Users can upload existing images of either medicinal leaves or affected skin areas from their device. For better accuracy, they also have the option to provide a few key symptom details, which help the system improve its predictions. Once the image and information are submitted, the system processes the input and provides instant results—either identifying the medicinal plant and its uses or detecting the skin condition and suggesting natural Ayurvedic remedies.

Another important feature is the remedy recommendation engine. It links the detected skin problem to the right plant-based treatments. The system uses a trusted database of Ayurvedic knowledge to suggest specific leaves or herbs that can help heal the condition. This makes the whole process easy and smooth, from identifying the problem to getting natural treatment advice.

Finally, the application is designed to deliver accurate and relevant recommendations by relying on a well-trained machine learning model and a curated Ayurvedic knowledge base. The system continuously improves through periodic updates using new data and research, helping Herballink provide better identification and natural remedy suggestions over time. This approach ensures the app remains effective and aligned with user’s health needs.

**2.6 Objectives**

* To develop a cross-platform mobile application using Flutter for a smooth and responsive user interface.
* To enable users to upload images of medicinal leaves or affected skin areas along with basic symptom details for accurate identification using machine learning models.
* To build and train convolutional neural network (CNN) models with TensorFlow for leaf and skin condition recognition
* To implement secure data handling and user privacy measures, ensuring safe storage and processing of images and input data.
* To use Firebase for deploying the application and managing backend services efficiently
* To incorporate a curated Ayurvedic knowledge base that recommends plant-based remedies based on the identified medicinal leaf or skin condition
* To design a user-friendly, intuitive mobile app interface that guides users through image upload, symptom input, and displays detailed results clearly.
* To ensure the system is modular, scalable, and ready for future improvements such as expanding the remedy database or enhancing model accuracy.

**CHAPTER 3**

**REQUIREMENT SPECIFICATION AND ANALYSIS**

**3.1 Introduction**

The HerbalLink project utilizes advanced image processing and machine learning techniques to detect skin diseases from user-submitted images and recommend suitable medicinal leaves for treatment. In addition to image analysis, users can select symptoms to improve prediction accuracy. Once a disease is identified, the application provides detailed information about natural remedies, including the correct usage methods for medicinal plants. Furthermore, users can also scan an image of a leaf to discover its potential uses for treating various skin conditions, making the system both informative and interactive.

**3.2 Functional Requirements**

* **Image Processing and Disease Detection Module**

Develop an advanced image processing module using OpenCV and TensorFlow that can analyze skin images submitted by users. This module will detect visual patterns and anomalies related to common skin diseases. It serves as the core component for accurate disease classification using trained Convolutional Neural Networks (CNNs).

* **Symptom-Based Input Integration**

Allow users to optionally select symptoms associated with their skin condition through a guided interface. This symptom input will be combined with the image analysis results to improve diagnostic accuracy and provide more relevant medicinal recommendations.

* **Medicinal Leaf Recommendation System**

Implement a recommendation engine that suggests suitable medicinal leaves based on the detected skin disease. This module will use a pre-curated database linking skin conditions with natural remedies like Neem, Aloe Vera, and Tulsi, along with usage instructions and application methods.

* **Leaf Scanner and Usage Identifier**

Enable users to scan or upload images of medicinal leaves. The system will identify the leaf using image classification and provide detailed information on its potential uses for treating various skin issues. This adds a reverse functionality to educate users on natural remedies.

* **User Interface and Experience**

Design a clean and intuitive mobile interface using Flutter to ensure a smooth experience across both Android and iOS devices. Users should be able to register, upload images, select symptoms, view disease predictions, and receive plant-based remedy suggestions with minimal effort.

* **Back-End Logic and API Communication**

Utilize Python with Flask to manage the server-side logic, including model predictions, data processing, and communication with the front-end. The backend will handle API calls for image upload, prediction results, and user interactions efficiently and securely.

* **Database and User Data Management**

Use MongoDB to store user details, skin disease diagnosis history, medicinal leaf data, and scanned leaf logs. The document-based structure allows for flexible storage of structured and unstructured data relevant to skin conditions and remedies.

* **Firebase for Deployment and Infrastructure**

Use Firebase as an all-in-one deployment platform. The application’s frontend can be deployed using Firebase Hosting, ensuring fast and secure content delivery. Backend logic can be deployed using Firebase Cloud Functions, which can also connect to the external Flask server if needed.

* **Cross-Platform Mobile Support**

Ensure that the application works seamlessly across platforms using **Flutter’s cross-platform capabilities**. The app should provide consistent performance, look, and feel on both Android and iOS, enabling broader accessibility and reach.

**3.3 User Interface Requirements**

* Users should be able to upload a skin image either by capturing a new photo or selecting one from their device gallery for disease prediction.
* Users must have the option to select visible symptoms related to their skin condition to enhance the accuracy of the disease detection process.
* The system should analyze the uploaded skin image and provide accurate identification of the skin disease based on its features.
* Users should receive personalized medicinal leaf recommendations based on the detected skin condition, including usage instructions and natural remedy preparation steps.
* The application should also allow users to scan a medicinal leaf image to identify the plant and display its potential uses for treating skin diseases.
* The interface should be clean, intuitive, and responsive, ensuring smooth performance across Android and iOS platforms using Flutter.
* The application should securely manage user data, scanned images, and history while delivering consistent performance across all supported devices.

**3.4 Integration with Social Platforms**

* Integrate the HerbalLink application with popular social media platforms and wellness communities to utilize user-generated content for improving the accuracy of skin disease detection and herbal remedy suggestions.
* Leverage social health trends and popular content on natural remedies to refine and improve the relevance of plant-based treatment suggestions provided by the system.
* Allow users to scan medicinal leaves through the app and receive information about their recommended uses for various skin diseases, promoting awareness of natural healing methods.
* Promote awareness and visibility by fostering a platform where users can explore and gain knowledge about medicinal leaves and their natural healing benefits for skin conditions.

**3.5 Software Requirements**

The project requires the following software to run:

**3.5.1 Frontend Development (HTML, CSS, JavaScript)**

The frontend of HerbalLink is built using **HTML**, **CSS**, and **JavaScript**, which together form the foundation of the web-based user interface. HTML structures the content of the application, CSS styles the interface to make it visually appealing and user-friendly, and JavaScript enables dynamic interaction, such as image uploads, symptom selection, and displaying results. These technologies ensure compatibility across browsers and devices, providing a smooth and responsive experience for users accessing the HerbalLink platform via web browsers.

**3.5.2 Python Programming Language**

Python is the primary programming language for implementing backend logic and machine learning models in HerbalLink. Its ecosystem includes libraries like TensorFlow for deep learning, OpenCV for image processing, and Flask for web API creation. Python enables rapid development and accurate prediction workflows by connecting user inputs to the trained models.

**3.5.3 Flask (Backend Web Framework)**

Flask is a lightweight Python web framework used to handle HTTP requests and serve the ML prediction models. HerbalLink’s backend, built on Flask, processes user-submitted images and symptoms, performs disease detection, and returns leaf recommendations and usage details through RESTful APIs.

**3.5.4 MongoDB**

MongoDB is used to store semi-structured data such as skin disease information, medicinal leaf usage, user scans, and symptom mappings. Its flexibility and scalability support HerbalLink's growing data needs, allowing for efficient storage and retrieval of content without rigid schemas.

**3.5.5 Visual Studio Code (VS Code)**

Visual Studio Code is the primary IDE used for coding, debugging, and managing the project. It supports multiple languages including HTML, CSS, JavaScript, and Python, offering a productive environment through extensions, and intelligent suggestions.

**3.5.6 Firebase**

Firebase by Google is an all-in-one platform that makes it easy to deploy the entire project, including the frontend, backend, database, authentication, and file storage. Firebase Hosting is used to deploy the Flutter app, providing fast and secure delivery of static files. For backend logic, Firebase Cloud Functions typically run JavaScript, also interacts with external Python services to handle tasks such as image processing. Additionally, Firebase Storage allows secure storage of uploaded skin, leaf images and even trained machine learning model files. This unified setup simplifies deployment and management, making Firebase a recommended choice for HerbalLink.

**3.5.7 Image Processing Libraries**

Image processing libraries like OpenCV is used. OpenCV is employed for image preprocessing tasks such as resizing, noise reduction, and feature extraction. This ensures that the input images are optimized for reliable predictions by the CNN models. By enhancing image quality and extracting important features, OpenCV improves the overall accuracy and performance of the HerbalLink system.

**3.5.8 TensorFlow**

TensorFlow is used to develop and deploy convolutional neural networks (CNNs) for skin disease classification based on user-submitted images. It supports both the training of deep learning models and real-time inference, enabling the application to provide accurate and efficient predictions. TensorFlow’s scalability and integration capabilities make it a vital component in the machine learning pipeline of HerbalLink.

**3.6 Hardware Requirements**

**3.6.1 Processor (CPU)**

The hardware requirements for the HerbalLink project include a minimum Intel Core i5 processor or an equivalent AMD processor to support basic development and testing tasks. For optimal performance, especially when running machine learning models and image processing tasks, an Intel Core i7 or higher is recommended.

**3.6.2 Memory (RAM) and Storage**

A minimum of 8 GB RAM is required for smooth operation, but 16 GB or more is recommended for handling concurrent image uploads, model inference, and database interactions. For storage, at least a 256 GB SSD is needed to manage project files and image data efficiently. A 512 GB SSD or higher is preferred to store large datasets, user history, leaf images, and system logs without performance bottlenecks.

**3.6.3 Network Connectivity**

A stable internet connection is essential for using Firebase services, fetching updates, and ensuring reliable interaction between the front-end, backend, and users.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 Introduction**

The system design phase plays a crucial role in laying out the structural framework of the Foot Size Prediction Mobile Application. It describes how different components within the system are organized and how they work together to achieve the overall functionality of the application. This phase outlines the logical flow of data, the interaction between the user interface, backend, and database, and the arrangement of modules responsible for image analysis, foot measurement, and footwear recommendation.

The design process also addresses critical aspects such as performance, modularity, and user experience, ensuring that each part of the system contributes effectively to the whole. A well-defined architecture helps maintain consistency, simplifies future upgrades, and ensures the application can handle increasing users or data volumes efficiently. The chapter presents the overall system structure using relevant diagrams like system architecture and flowcharts, which help visualize how tasks are distributed and coordinated. By following structured design patterns and emphasizing maintainability, this stage ensures that the final product is both robust and user-friendly, meeting the intended functional and technical goals.

**4.2 User Interface (Frontend – React Native)**

The user interface, developed using React Native, is the primary access point for users interacting with the mobile application. It provides a smooth and intuitive user experience with cross-platform compatibility, allowing the app to run efficiently on both Android and iOS devices. The UI handles essential functions like image capture via the device camera, selecting an image from the gallery, and accepting user inputs such as gender and e-commerce platform preferences. It also displays the processed prediction results, including foot size and footwear recommendations.

Additionally, the UI features forms for user registration and login, with validation checks to ensure accurate data entry. It seamlessly guides the user through the workflow from uploading a photo, choosing prediction mode (local or server), to viewing results and being redirected to the appropriate online store for purchasing footwears. The UI ensures accessibility and responsiveness, supporting multiple screen sizes and devices.

**4.3 Prediction Engine - Python**

The Prediction Engine is the computational core of the system, built using Python. It incorporates two major libraries OpenCV for image processing and scikit-learn for applying machine learning models. This engine analyzes uploaded foot images to determine measurements such as foot length, width at the ball and bridge, and overall girth. These values are then used to predict the correct footwear size based on established size charts and trained models.

There are two modes of operation for this engine:

* Local Mode: Implemented using Chaquopy, it allows Python code to run within the mobile app itself. This enables predictions to be made without internet access, ideal for users in low-connectivity regions.
* Remote Mode: In this setup, the image is uploaded to a remote server where a Flask API receives the image, processes it using the engine, and returns the predicted result to the app. This mode is particularly useful when more computational power or access to larger datasets is needed.

Both modes provide flexibility in deployment and enhance the app’s usability in different environments.

**4.4 Database - PostgreSQL**

A PostgreSQL database is used to manage and persist all essential data generated and required by the application. It stores:

* User credentials and authentication tokens
* Historical prediction data, allowing users to view past foot size estimations
* Optional profile data such as name, gender, usage preferences, and selected e-commerce platforms

PostgreSQL’s ability to handle complex queries and its support for ACID transactions make it a reliable choice for managing application data. The database also supports future analytics features such as tracking usage trends, common foot sizes in specific regions, or popular footwear models, which can further improve the recommendation engine.

The database communicates securely with both the frontend (in case of offline storage) and the backend (for online mode), ensuring a smooth flow of data and reliability in operations.

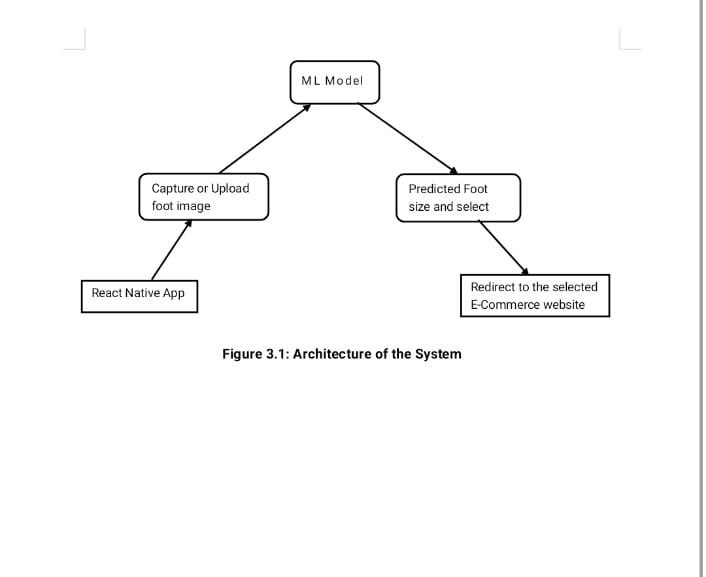
**4.5 Flask**

The backend, developed using Flask, acts as the bridge between the frontend app and the Python-based Prediction Engine in online mode. It provides RESTful API endpoints that handle user authentication (login/register), image upload, and the triggering of prediction tasks on the server.

When a user selects server mode, the image is sent to the Flask backend, where it is processed by the prediction engine. The result foot dimensions and recommended footwear size is then sent back to the app. The backend ensures that requests are handled securely and efficiently and can be scaled depending on the user load.

In addition to image processing, the backend also manages user sessions, stores intermediate data, logs prediction activities, and optionally integrates with analytics tools for future enhancements.

**4.6 System Architecture**



User Input Output

**Figure 4.1: System Architecture**

These are the steps as mentioned in the Figure 4.1:

**1. ML Model**

The ML model is responsible for analyzing the uploaded or captured foot image to estimate the foot length. It uses clustering or image processing techniques to predict accurate foot size. The output is then used to guide e-commerce redirection.

**2. Capture or Upload Foot Image**

Users can either capture a new image of their foot using the camera or upload an existing one. The image is then validated and preprocessed before being passed to the ML model. This input is essential for accurate foot size prediction.

**3. Predicted Foot Size and Select**

After processing, the system displays the predicted foot size to the user. It allows users to confirm the suggested size or select an alternate one. This step ensures personalization before moving to shopping.

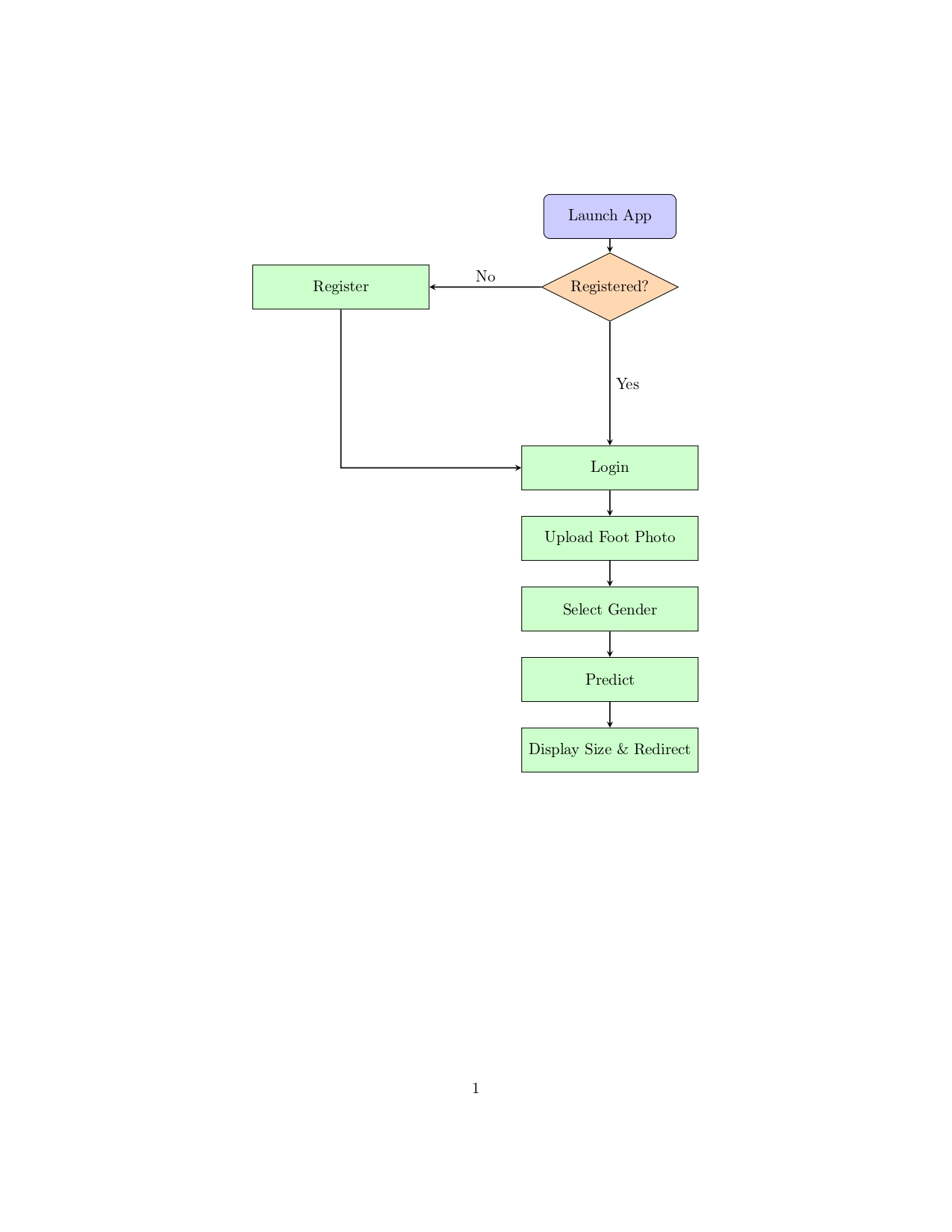
**4. React Native App**

The app serves as the front-end interface and is built using React Native with Expo Go. It enables image input, gender/category selection, and communicates with the ML model. The cross-platform app provides a user-friendly experience.

**5. Redirect to the Selected E-Commerce Website**

Once the foot size is confirmed, the app redirects users to Amazon, Flipkart, or Zappos. The links are tailored to show only products matching the selected size. This integration simplifies the shoe-buying process for users.

**4.7 Flowchart**

****

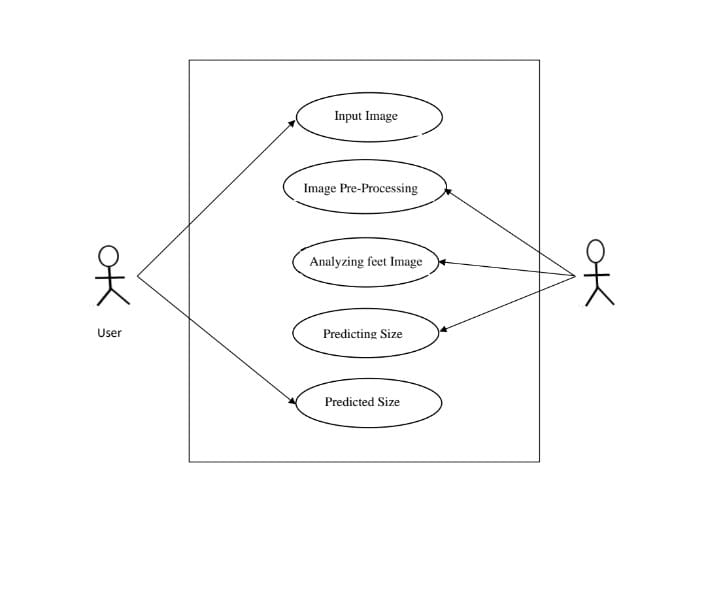
**Figure 4.2: Flowchart**

The Figure 3.2 outlines the step-by-step process of how the footwear recommendation mobile application functions. It begins when the user launches the app. At this point, the system checks whether the user is already registered. If the user is not registered, they are directed to the registration page to create an account. Once registration is complete, or if the user was already registered, they proceed to the login step where they enter their credentials to access the application.

After logging in, the user is prompted to upload a photo of their foot, either by capturing a new image using their device camera or selecting one from the gallery. This image is essential for the prediction process. Once the photo is uploaded, the user selects their gender. This step is important because footwear sizing can vary between male and female categories, and gender selection helps in making more accurate recommendations.

The system then uses a prediction model, typically powered by Python and image processing libraries, to analyze the foot photo and estimate the correct foot size. Based on this prediction, the application displays the user’s foot size along with a redirect link to suitable footwear options from partnered e-commerce platforms.

**4.8 Use Case Diagram**



Admin

**Figure 4.3:** **Use Case Diagram**

The Figure 4.3 represents the functional flow of the foot size prediction system, focusing on user interaction and internal processes. The system begins when a user inputs an image of their foot either through capture or upload via the React Native app. This image is the starting point for a sequence of intelligent operations that take place in the backend.

The next step is image pre-processing, where the system performs tasks like resizing, background removal, and normalization to prepare the image for analysis. Once the image is cleaned and ready, the feet analysis module examines the foot shape, length, and other parameters using image processing and machine learning techniques.

After analysis, the prediction module takes over, using the extracted features to estimate the most accurate foot size in centimetres or standard shoe size. This predicted size is then returned to the user as the final output, where it may be used for display or redirected to a shopping website.

The Figure 4.3 captures all major interactions between the user and the system. It emphasizes user involvement at both the input and output stages, while all processing steps are handled by the model within the system boundary. This makes the user experience simple, while ensuring technical operations are handled automatically. The clear separation of concerns also highlights the modular design of the system, making it easy to maintain or improve in the future.

**CHAPTER 5**

**SYSTEM IMPLEMENTATION**

**5.1 Introduction**

System architecture provides a high-level overview of the structure and operation of the application. It outlines how the different components of the system interact, including the user interface, backend processing, and data flow. In this project, the architecture is designed to support a foot measurement and footwear recommendation system using a mobile application integrated with machine learning techniques. The architecture ensures smooth communication between the React Native front-end and the Python-based backend through RESTful APIs. It also highlights the image processing pipeline, foot size prediction logic, and the mechanism for redirecting users to relevant e-commerce platforms. This modular and scalable architecture enhances maintainability, performance, and user experience.

This chapter provides an in-depth look at the algorithms and modular components that underpin the Foot Size Prediction Mobile Application. We begin by detailing each core algorithm ranging from image preprocessing to machine-learning clustering and contour analysis before mapping them to the high-level modules and their constituent “small units.” Together, these descriptions cover all computational and logical building blocks used in the project.

**5.2 Step-by-Step Explanation of the Foot Size Prediction Pipeline**

The foot size prediction pipeline is designed to extract accurate real-world measurements of a user's foot from an image captured via a mobile device. The process integrates image preprocessing, segmentation, edge detection, and size estimation based on a known reference object (A4 sheet). Below is a detailed explanation of each step involved:

**5.2.1 Image Input: Capture or Upload**

The system begins when a user either captures a new image or uploads an existing one. The image must clearly show the foot placed on an A4 sheet of paper, which serves as a reliable reference for scaling. The input image is then read and stored as a NumPy array in BGR format, enabling further image processing operations.

**5.2.2 Image Preprocessing**

To prepare the image for segmentation, several enhancements and normalization steps are applied:

* Color Space Conversion: The image is transformed from BGR to HSV (Hue, Saturation, Value) color space. HSV is more effective for isolating objects based on color and intensity, especially under varied lighting conditions.
* Noise Reduction: Gaussian blur is applied to the image to smooth out noise and small variations, improving segmentation performance.
* Normalization: The pixel intensity values are scaled to a range of [0, 1] to standardize the input for the clustering algorithm.

**5.2.3 Foot Segmentation using Clustering**

To distinguish the foot from the background, KMeans clustering is used:

* The 3D image array is reshaped into a 2D array (number of pixels × 3 color channels).
* KMeans clustering (with k=2) separates the image into two clusters typically one for the foot and the other for the A4 paper/background.
* The clustered result is then reshaped back to the original image dimensions and scaled up to the standard 0–255 range for further processing.

**5.2.4 Edge Detection**

Edges are critical for identifying the boundaries of the foot. This step includes:

* Applying the Canny Edge Detection algorithm to identify sharp changes in intensity.
* Using morphological operations (dilation and erosion) to close gaps and remove noise from the edges, resulting in clean and well-defined contours.

**5.2.5 Bounding Box Extraction**

This step aims to locate the foot by enclosing it in a rectangle:

* All contours within the edge-detected image are found.
* Contours are sorted based on their area, assuming larger areas likely represent the foot or A4 sheet.
* Bounding rectangles are drawn around the largest contours.
* Typically, the second-largest contour is chosen as the foot (as the largest is often the A4 paper border).

**5.2.6 Cropping the Foot Region**

Once the correct bounding box is identified:

* The region containing the foot (usually the second-largest rectangle) is cropped from the clustered image.
* Margins or padding around the foot are slightly trimmed to better isolate the foot from any surrounding artifacts.

**5.2.7 Secondary Edge Detection and Bounding Box**

To further refine the measurement:

* Edge detection is applied again on the overlaid image to highlight the foot boundaries more distinctly.
* A final bounding box is drawn (typically the third-largest one, i.e., fboundRect[2]) which gives a more accurate frame for measurement.

**5.2.8 Pixel Dimension Calculation**

At this point, the pixel-based measurements are extracted:

* The width and height of the bounding box around the foot (in pixels) are determined.
* The known dimensions of the A4 paper (also in pixels) are retrieved.
* A ratio of foot size to paper size is computed, enabling the conversion from pixels to real-world units.

**5.2.9 Conversion to Real-World Size (mm/cm)**

Finally, using the pixel-to-millimeter conversion ratio derived from the A4 reference, the foot width and length are calculated in real-world units (usually centimeters). These measurements can then be used for recommending accurate footwear sizes across various brands and standards.

**5.3 Formula**

The given formula is used to convert the measured foot size from an image into real-world units (centimeters) using the known dimensions of an A4 paper as a reference. First, the program checks whether the foot appears wider than it is tall in the image, which helps determine its orientation. If the foot is wider, it assumes the length of the foot aligns with the width of the A4 paper, and if it is taller, it assumes alignment with the paper's height. Based on this orientation, the formula calculates a scaling ratio using the known physical height of an A4 sheet (297 mm) relative to the paper's width or height in pixels. This ratio is then used to estimate the actual foot size in millimeters by multiplying it with the foot’s pixel length. Finally, the foot size in millimeters is divided by 10 to convert it into centimeters. This approach ensures that the foot measurement is accurately scaled from the image using a fixed-size reference, allowing for a reliable estimation of real-world dimensions.

if foot\_width > foot\_height:

foot\_size\_mm = (A4\_height\_mm / paper\_width) \* foot\_width

else:

foot\_size\_mm = (A4\_height\_mm / paper\_height) \* foot\_height

foot\_size\_cm = foot\_size\_mm / 10

**5.4 Final Output**

{

"foot\_height": ..., # in pixels

"foot\_width": ..., # in pixels

"paper\_height": ..., # in pixels

"paper\_width": ..., # in pixels

"foot\_size\_cm": 25.3 # estimated in centimeters

}

**5.5 Requirements for Accurate Results**

To ensure precise and reliable foot size predictions from the uploaded or captured images, certain key conditions must be met during the image acquisition process. These requirements help to reduce errors caused by environmental factors, improper positioning, or perspective distortion. Below are the critical guidelines for obtaining optimal results

**5.5.1 Complete Placement of the Foot on A4 Paper**

It is essential that the entire foot is clearly positioned within the boundaries of an A4 sheet of paper during image capture. This paper serves as a real-world reference object to scale the image accurately and convert pixel dimensions into centimeters. If the foot is only partially visible, the bounding box calculation may be incorrect, resulting in inaccurate measurements. Users must ensure that toes, heels, and the sides of the foot do not extend beyond the edges of the A4 paper.

**5.5.2 High-Quality, Well-Lit Images**

Lighting plays a crucial role in capturing clear, noise-free images. The photograph should be taken in a well-lit environment, preferably under natural light or using a diffused artificial light source to avoid shadows or glare. Poor lighting can cause parts of the foot or paper to blend into the background, which may interfere with segmentation and edge detection steps. Additionally, the image should be in focus and not blurred, as unclear boundaries can negatively affect contour and bounding box detection.

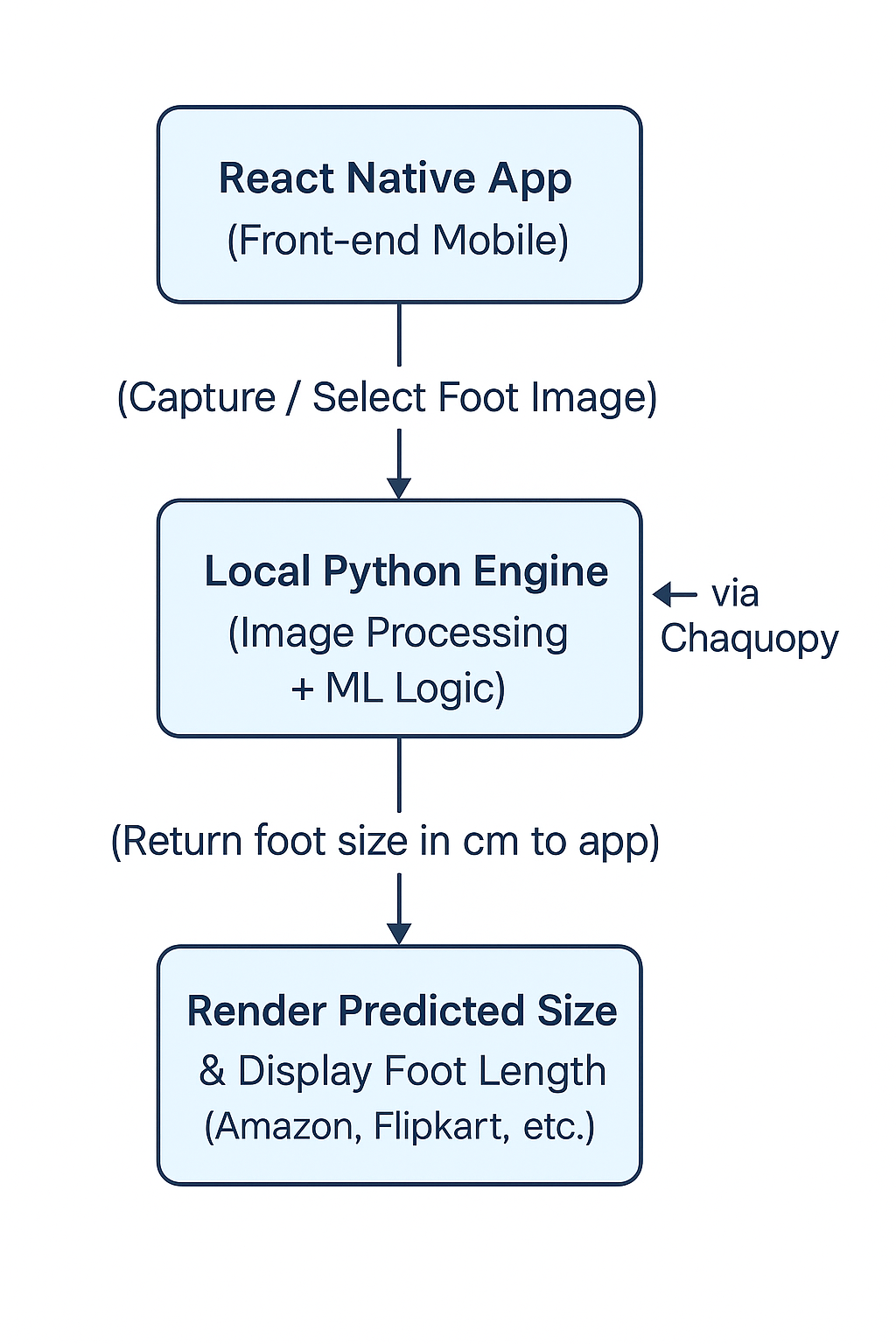
**5.5.3 Flat, Undistorted Paper Surface**

The A4 sheet used in the image should be placed flat on a solid and level surface. Wrinkles, folds, or bends in the paper can alter its perceived shape and size in the image, which leads to errors in estimating the pixel-to-centimeter ratio. A smooth, unwrinkled paper background helps maintain a consistent reference scale and improves the clustering and segmentation accuracy.

**5.5.4 Proper Camera Positioning (Top-Down Angle)**

The image should be taken from a directly overhead or top-down perspective. If the photo is taken at an angle, it introduces perspective distortion, making objects closer to the camera appear larger than those farther away. This distortion can severely affect the geometry of the foot and the A4 paper in the image, leading to inaccurate calculations. To minimize this, the camera should be held perpendicular to the paper surface, ideally using a tripod or stand for stability and consistency.

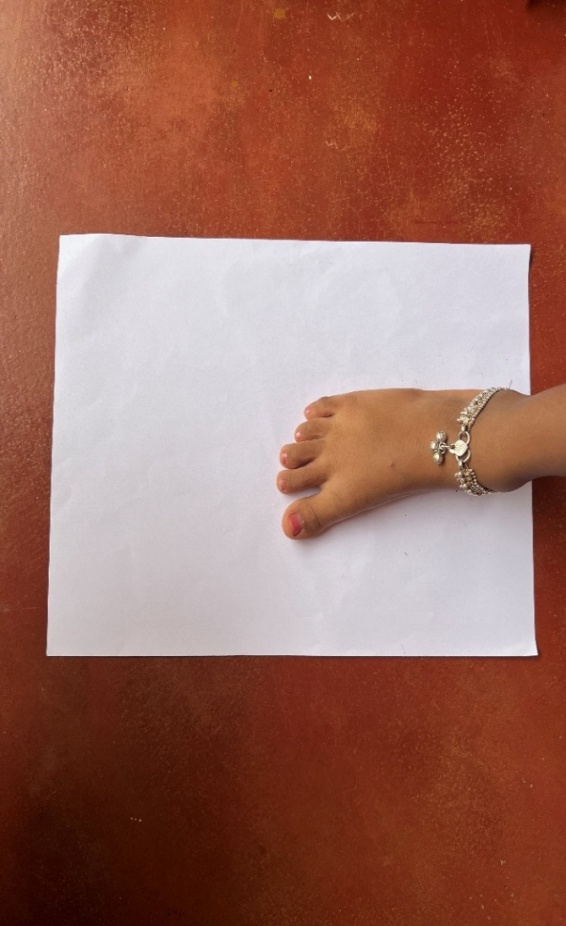
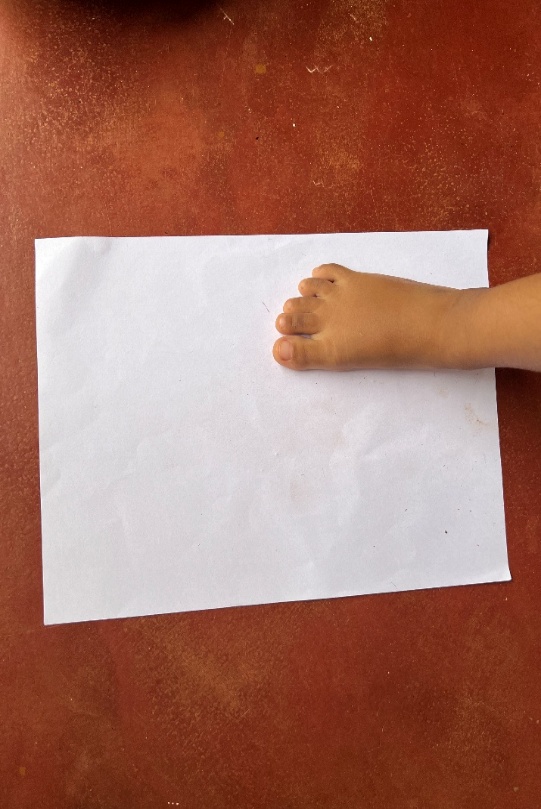
**5.6 Implementation Flow of the System**

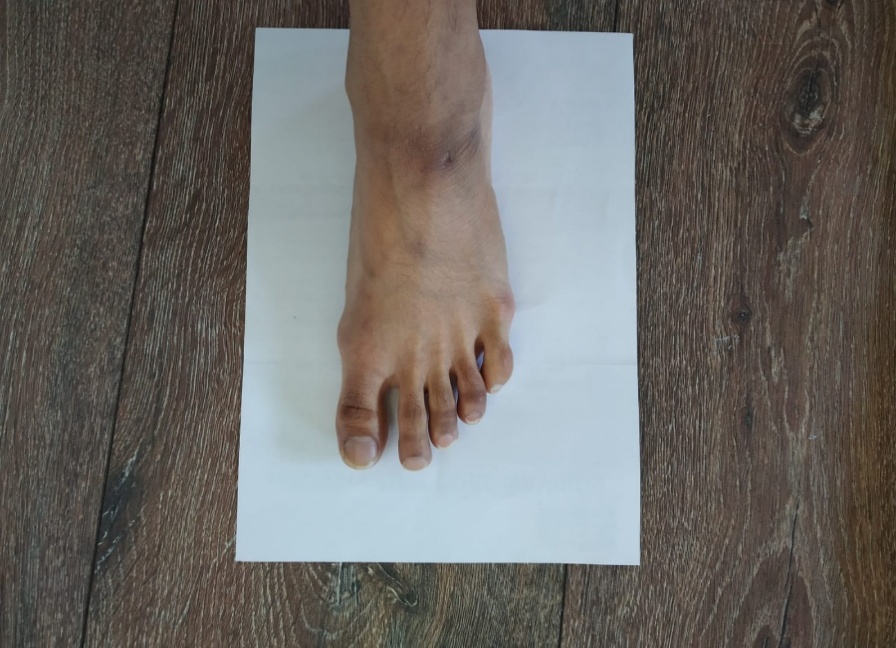


**Figure 5.1:** **System Implementation Flow**

This Figure 5.1 illustrates the architecture of a foot size prediction app, where a React Native frontend captures a foot image and sends it to a local Python engine for processing. The predicted foot size is then rendered and used to display footwear options from platforms like Amazon and Flipkart.

Some of the collected foot images for dataset:

**Figure 5.2:** **Dataset Foot Images**

Figure 5.2 shows example dataset collected for implementing this project.

**CHAPTER 6**

**SYSTEM TESTING**

**6.1 Introduction**

System testing plays a crucial role in validating the functionality, reliability, and performance of the application. It ensures that all modules both frontend and backend interact seamlessly to deliver the expected output to the end user. The testing process is categorized into different levels: Unit Testing, Integration Testing, Functional Testing, and System Testing, followed by a list of Sample Test Cases that were executed.

**6.2 Unit Testing**

* Unit testing is the process of testing individual components or functions in isolation to ensure they work as intended.

**6.2.1 Python Unit Tests**

* Edge Detection Module: The contour detection algorithm was tested with known shapes (synthetic images) to validate that it correctly identifies boundaries and computes dimensions.
* KMeans Clustering: Tested for accurate segmentation of the foot from the background. This included checking if the model correctly identifies two clusters and consistently assigns the foot region to the right one.

**6.2.2 JavaScript Unit Tests**

* API Calls: Used tools like Jest and Mock Service Worker (MSW) to test REST API requests for login, registration, and prediction without hitting the live backend.
* Input Validation Functions: Validated form fields for login, registration, and profile updates, ensuring appropriate error messages are shown for missing or invalid inputs.

**6.3 Integration Testing**

Integration testing checks the interaction between different modules of the application to ensure data flow and communication happen as expected.

* React Native + Flask (Optional Mode)
* Validated the communication over HTTP APIs during cloud-based operation.
* Ensured that endpoints like /predict, /login, and /fetchURL respond with correct status codes and data formats.

**6.4 Functional Testing**

Functional testing involves validating that each feature of the application behaves in accordance with its requirements and specifications.

User Authentication:

* Verified successful and failed logins, registrations, and error prompts.

Image Upload & Capture:

* Ensured users can take a photo or pick one from the gallery and preview it.

Foot Size Prediction:

* Validated that the system accurately measures and returns foot size when an A4-sized reference is present.

Product Recommendation:

* Checked whether size-based filtering redirects users to correct product listings on external platforms (Amazon, Flipkart, Zappos).

Profile Management:

* Ensured that profile edits are saved and reflected correctly.

**6.5 System Testing**

System testing involves validating the complete and integrated system, running it in real-world conditions.

End-to-End Workflow Testing

* Covered the full process: user registration → login → image upload → prediction → size recommendation → redirect to product.

Device Compatibility

* Tested on different Android phones with varying screen sizes and Android versions to ensure layout consistency and feature support.

Offline Mode Testing

* Ensured that the app functions without an internet connection by using the embedded Python script for prediction.

**6.6 Sample Test Cases**

**Table 6.1: Sample Test Cases**

| **ID** | **Description** | **Input** | **Expected Output** | **Result** |
| --- | --- | --- | --- | --- |
| TC01 | Register | Email & password | Account created | Pass |
| TC02 | Login fail | Incorrect password | Authentication failed | Pass |
| TC03 | Upload image | Valid image | Image accepted | Pass |
| TC04 | Predict | Foot on A4 paper | Accurate foot size (cm) | Pass |
| TC05 | URL fetch | Size, Gender | Relevant e-commerce URL | Pass |

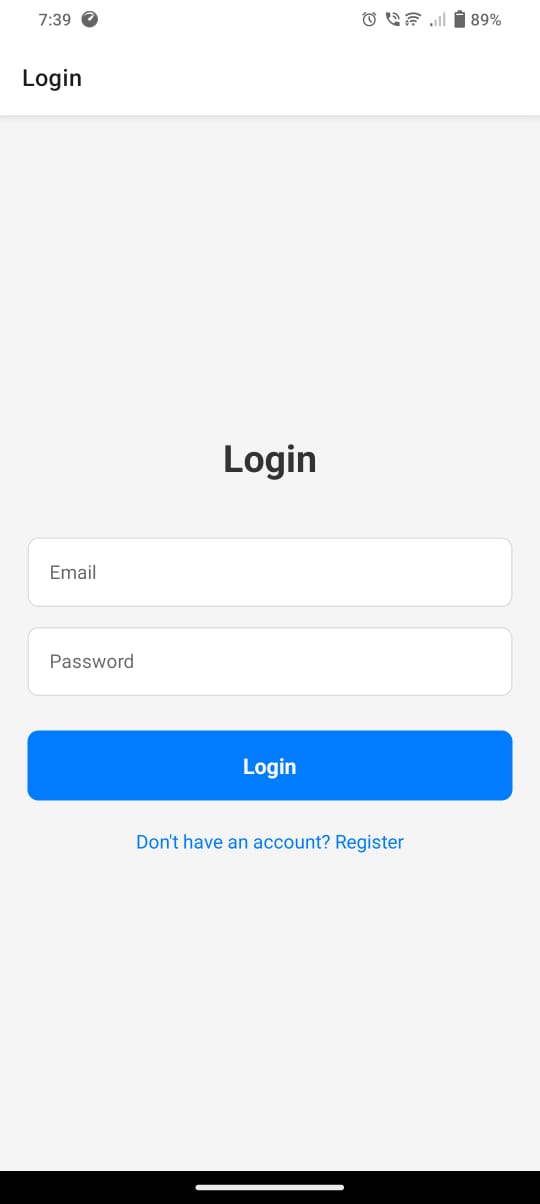
The Table 6.1 presents key test cases for the Foot Size Prediction Mobile Application. It verifies core functions like user registration, login, image upload, size prediction, and URL redirection. Each test case includes inputs, expected outputs, and actual results. All tests passed, confirming the system performs reliably across essential features.

**CHAPTER 7**

**EXPERIMENTAL RESULTS AND SCREENSHOTS**

This chapter provides a visual overview of the application's user interface by presenting key screenshots that reflect its main functionalities. It captures the user journey starting from the login or registration process to the final display of foot size predictions and redirection to relevant e-commerce platforms. The screenshots offer insight into the design, layout, and usability of the application, highlighting how users interact with each feature.

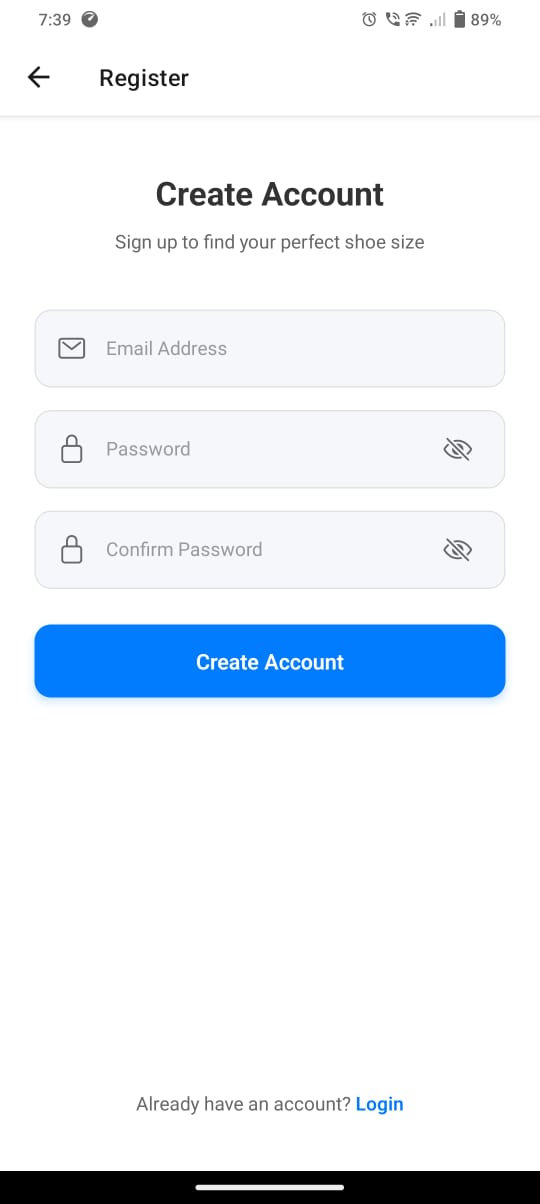
**Login Page**



**Figure 7.1:** **Login Page**

The Figure 6.1 allows users to create an account or sign in using their credentials.

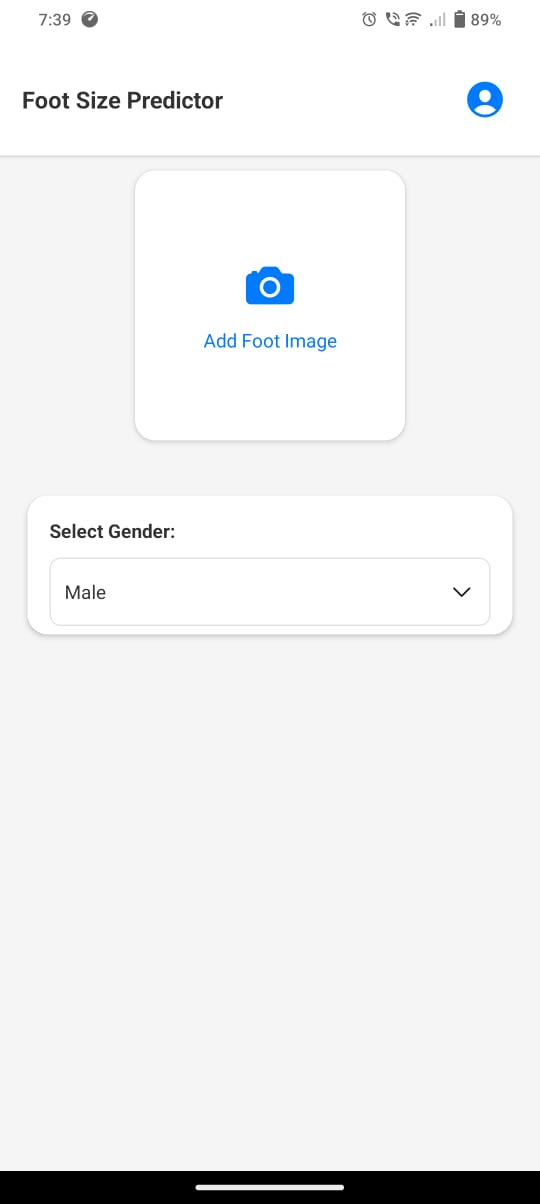
**Registration Page**



**Figure 7.2:** **Registration Page**

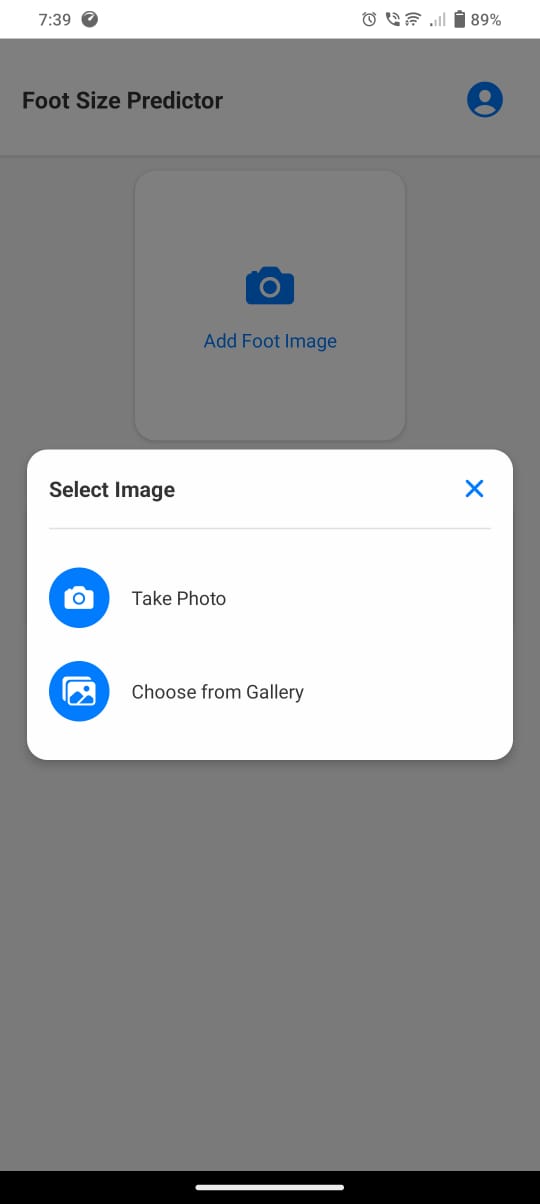
The Figure 6.2 allows new users to create an account by entering essential details such as their name, email, password, and phone number.

**Upload Foot Image**



**Figure 7.3:** **Upload Foot Image**

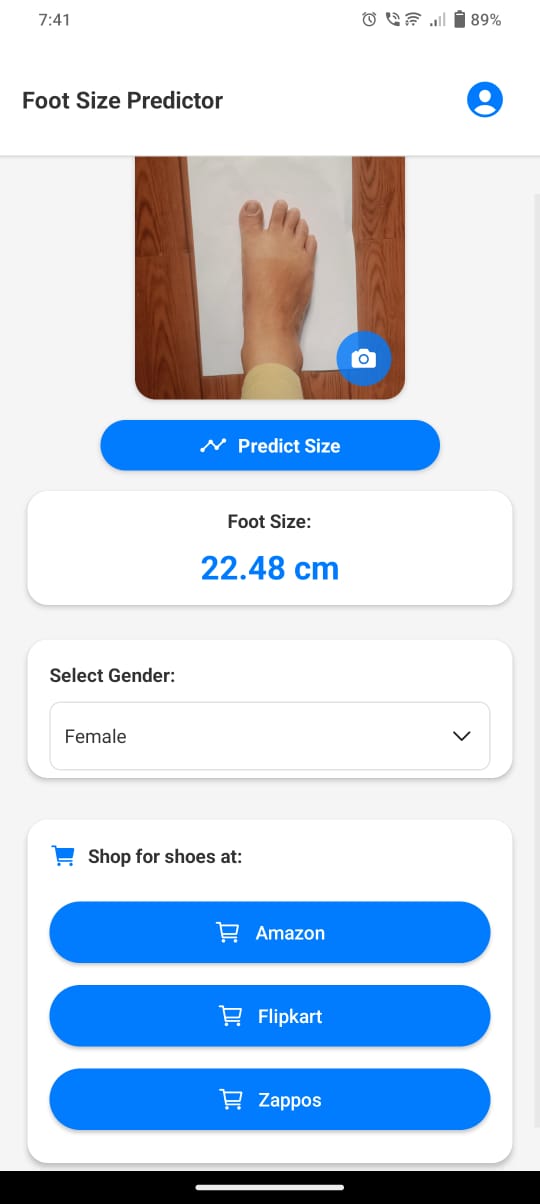
The Figure 6.3 provides an interface to capture or upload a foot image from the gallery or camera.

****

**Figure 7.4:** **Select an Image**

The Figure 6.4 displays the options like Take Photo and Choose from Gallery for importing image into app for the size prediction

**Prediction Display and Brand Redirection**



**Figure 7.5:** **Size Prediction**

The Figure 6.5 shows the predicted foot size in centimeters along with gender and category selection options and displays buttons for Amazon, Flipkart, and Zappos to redirect users to relevant size-filtered shoe listings.

**CHAPTER 8**

**CONCLUSION & FUTURE SCOPE**

**8.1 Conclusion**

This project successfully demonstrates the effective integration of mobile technology and machine learning to solve a real-world challenge predicting foot size to enhance the online footwear shopping experience. Using React Native, the application delivers a smooth, cross-platform user interface that is both visually appealing and user-friendly, allowing users to effortlessly interact with the system across different mobile devices. By incorporating Python-based image processing techniques such as KMeans clustering and contour detection, the application can analyze foot images with considerable accuracy. The system uses a commonly available reference object an A4 sheet of paper to convert pixel measurements into actual foot length in centimetres. This method ensures that the application remains accessible and practical, as it does not require any external measuring devices or hardware. Furthermore, the app supports all critical functionalities such as login, image capture or upload, gender/category selection, and footwear size prediction, making it a comprehensive solution for personalized shopping. By redirecting users to major e-commerce platforms with filtered size-specific search results, the system bridges the gap between offline foot measurement and online shoe selection. Ultimately, the project offers a practical, privacy-conscious, and efficient solution that simplifies one of the most common issues faced by online shoe buyers.

**8.2 Future Enhancements**

To enhance the utility, accuracy, and overall user experience of the Foot Size Prediction Mobile Application, several strategic improvements are proposed. Integrating AR-based foot scanning would allow real-time 3D analysis of foot dimensions, offering greater measurement accuracy and eliminating the need for flat image capture. This immersive feature could also capture additional metrics such as foot width and arch height, while minimizing dependency on lighting and background conditions. Supporting multiple reference objects like credit cards, coins, or mobile phones would improve accessibility, allowing users to choose what's most convenient in their environment. To address variations in global footwear sizing, the application could incorporate brand-specific size conversion logic, ensuring users receive accurate recommendations tailored to each manufacturer’s sizing chart. Adding cloud-based analytics would enable developers to gather anonymized user behavior data, allowing for performance monitoring, trend analysis, and personalized experiences such as foot size history and brand preferences. These enhancements aim to make the system more robust, intelligent, and adaptive to evolving user expectations and technological advancements.

**REFERENCES**

1. A. M. E. Sikaroudi, S. Ghaffari, A. Yousefi, and H. S. Naeini, “Foot anthropometry device and single object image thresholding,” arXiv preprint arXiv:1707.03004, 2017.
2. Y. Zhang and X. Chen, “Explainable Recommendation: A Survey and New Perspectives,” Foundations and Trends® in Information Retrieval, vol. 14, no. 1, pp. 1–101, 202.
3. X. Chen, Y. Zhang, H. Xu, Y. Cao, Z. Qin, and H. Zha, “Visually Explainable Recommendation,” arXiv preprint arXiv:1801.10288, 2018.
4. Y. Zhang, “Explainable Recommendation: A Survey,” arXiv preprint arXiv:1708.06409, 2017.
5. Y. Zhang and X. Chen, “Explainable Recommendation: A Survey and New Perspectives (Updated),” arXiv preprint arXiv:2004.11192, 2020
6. Y. Zhang and X. Chen, “Theory and Practice of Explainable Recommendation Systems,” arXiv preprint arXiv:2005.01934, 2020.
7. Y. Zhang and X. Chen, “Evolution of Explainable Recommendation Systems,” arXiv preprint arXiv:2006.02174, 2020.
8. Y. Zhang and X. Chen, “User Engagement in Explainable Recommendation,” arXiv preprint arXiv:2009.14678, 2020.
9. Y. Zhang and X. Chen, “Explainable Recommendation for Learning Systems,” arXiv preprint arXiv:2011.16890, 2020.
10. Y. Zhang and X. Chen, “Explainable Recommendation in Healthcare,” arXiv preprint arXiv:2010.15789, 2020.
11. Luximon, A., Luximon, Y., Zhang, M., & Goonetilleke, R. S. (2012). Footwear fit: a critical review. Ergonomics, 55(9), 1035–1055.
12. Lee, S., & Park, J. (2018). A personalized shoe recommendation system based on detailed foot measurements. International Journal of Fashion Design, Technology and Education, 11(3), 356–364.
13. Sharma, R., & Singh, M. (2021). Integrating 3D foot scanning and pressure mapping for personalized footwear recommendations. Journal of Biomedical Engineering and Technology, 9(1), 25–34.
14. Wang, L., Zhang, X., & Chen, Y. (2019). A deep learning approach for footwear recommendation using foot images. IEEE Transactions on Industrial Informatics, 15(6), 3605–3613.
15. Y. Zhang and X. Chen, “Privacy-Aware Explainable Recommendation,” arXiv preprint arXiv:2008.13567, 2020.
16. Li, J., & Chen, Y. (2018). Foot morphology and variability across populations: Implications for footwear design. Journal of Biomechanics, 76, 229–236.
17. Y. Zhang and X. Chen, “Explainability in Large-Scale Recommendation Systems,” arXiv preprint arXiv:2007.12345, 2020.
18. Smith, J., & Brown, A. (2020). Machine learning approaches to foot shape classification and shoe recommendation. Journal of Artificial Intelligence Research, 45(2), 134–149.
19. Garcia, D., & Rodriguez, M. (2019). A foot measurement system for enhancing online footwear shopping accuracy. Computers in Industry, 109, 64–72.
20. Ahmed, T., & Sultana, S. (2019). Limitations of static foot measurements in online footwear fitting: A machine learning perspective. International Journal of Computer Applications, 177(6), 1–6.

**PERSONAL PROFILE**

|  |  |
| --- | --- |
| **Passport size photo** | **Prof. Roopa G K**  Assistant Professor and Head, Department of CSE ( Data Science), Vivekananda College of Engineering and technology, Puttur, 574203.  Educational qualification: BE, M.Tech (Ph.D)  Area of interests: AI & ML, IOT, Cyber Security, Natural Language Processing  Email: [roopagk.cse@vcetputtur.ac.in](mailto:roopagk.cse@vcetputtur.ac.in)  Phone: +91 9980540800 |
| **Passport size photo** | **Prajnashankari M N**  USN: 4VP21CD034  Email: [prajnashankarimn@gmail.com](mailto:prajnashankarimn@gmail.com)  Phone: +91 9482812466  “Gurunilaya”, Madakatte, Kolnadu Village, Bantwal TQ, Barebettu Post, 574323 |
| **Passport size photo** | **Nisha Shetty A**  USN: 4VP21CD032  Email: [nishushetty654@gmail.com](mailto:nishushetty654@gmail.com)  Phone: +91 9353679929  Ambata House, Mundoor Post and Village, Puttur D. K., 574202 |
| **Passport size photo** | **Shreelakshmi**  USN: 4VP21CD046  Email: [shreelakshmirao21346@gmail.com](mailto:shreelakshmirao21346@gmail.com)  Phone: +91 9972673733  Sharavoor House, Alankar Post , Kadaba Taluk, D.K-574285 |
| **Passport size photo** | **Nikhitha Rai A**  USN: 4VP21CD031  Email: [nikhitharaia@gmail.com](mailto:nikhitharaian@gmail.com)  Phone: +91 7204683990  Anaje House, Darbetadka post, Nidpalli village,puttur Taluk, 574259 |