



PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Capstone Project Report Phase-2 on

SMART EYE CARE

Submitted by
PRATHAM SHANKAR - (PES1PG22CA152)

Feb 2024 – Jun 2024

under the guidance of

Guide Details

Ms. Archana A

Assistant Professor
Department of Computer Applications,
PESU, Bengaluru – 560085



PES
UNIVERSITY

**FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER APPLICATIONS
PROGRAM – MASTER OF COMPUTER APPLICATIONS**

CERTIFICATE

This is to certify that the project entitled

Smart Eye Care

is a bonafide work carried out by

Pratham Shankar-PES1PG22CA152

in partial fulfillment for the completion of Capstone Project Phase-2 work in the Program of Study MCA under rules and regulations of PES University, Bengaluru during the period Feb. 2024 – June 2024. The project report has been approved as it satisfies the academic requirements of 4th semester MCA.

Internal Guide

Archana A,
Assistant Professor
Department of Computer
Applications
PES University
Bengaluru - 560085

Chairperson

Dr. Veena S

Department of Computer
Applications
PES University
Bengaluru - 560085

**Dean- Faculty of
Engineering & Technology**

Dr. B K Keshavan

PES University
Bengaluru - 560085

DECLARATION

I, **Pratham Shankar**, bearing **PES1PG22CA152** hereby declare that the Capstone project Phase-2 entitled, ***Smart Eye Care***, is an original work done by me under the guidance of **Ms. Archana A**, *Designation*, PES University, and is being submitted in partial fulfillment of the requirements for completion of 4th Semester course in the Program of Study **MCA**.

All corrections/suggestions indicated for internal assessment have been incorporated in the report.

The plagiarism check has been done for the report and is below the given threshold.

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

PLACE: Bengaluru

DATE: 21-06-2024

PRATHAM SHANKAR
PES1PG22CA152

ACKNOWLEDGEMENT

I take a great pleasure in expressing my sincere gratitude to all those who have guided me and supported me to successfully complete this project.

I would like to express my sincere gratitude to the Vice Chancellor of PES University, **Dr. J Surya Prasad** and Chairperson **Dr. Veena S**, who gave me an opportunity to go ahead with this project.

I am grateful to my guide, **Ms. Archana A**, Assistant Professor, Department of Computer Applications, who provided me with guidance, encouragement and support, during the course of project.

PRATHAM SHANKAR
PES1PG22CA152

ABSTRACT

People suffering from ocular diseases such as glaucoma, keratoconus, corneal ulcer etc., often lose vision permanently, the main reason being that such diseases were not detected in their early stages and thus not enabling early treatment, with surgery the only possible option left as treatment. This cannot be afforded by everyone hence no treatment is provided. The project aims to tackle this problem by the use of machine learning models to detect such diseases in their early stages and provide early treatment. The project explores approaches that help doctors keep track of their patients easily by using a medication tracking system, and also allow patients to interact with their doctors frequently.

The project aims to make eye health-care more accessible and strategic for both the doctors and the patients using machine learning models to facilitate early detection of various eye-related diseases that could potentially lead to irreversible vision impairment. Utilizing machine learning models to detect ocular diseases and assess an individual's susceptibility to specific conditions, thus enabling timely intervention and treatment. Using image processing enabled by machine learning to analyze and interpret uploaded retinal scans. Add an additional feature that sends timely alerts to users to remind them to take their medications at the scheduled times.

Table Of Contents

Abstract

Sr. No	Particulars	Page No.
1	INTRODUCTION	
	1.1 PROJECT DESCRIPTION	1-2
	1.2 PURPOSE	
	1.3 SCOPE	
2	LITERATURE SURVEY	
	2.1 DOMAIN SURVEY	
	2.2 RELATED WORK	3-12
	2.3 EXISTING SYSTEMS	
	2.4 TECHNOLOGY SURVEY	
	2.5 FEASIBILITY STUDY	
3	HARDWARE AND SOFTWARE REQUIREMENTS	
	3.1 HARDWARE REQUIREMENTS	13
	3.2 SOFTWARE REQUIREMENTS	
4	SOFTWARE REQUIREMENTS SPECIFICATION	
	4.1 USERS	14-15
	4.2 FUNCTIONAL REQUIREMENTS	
	4.3 NON-FUNCTIONAL REQUIREMENTS	
5	SYSTEM DESIGN	16
	5.1 CONTEXT-FLOW DIAGRAM	
6	DETAILED DESIGN	
	6.1 PROCESS FLOW DIAGRAM WITH DESIGN METHODOLOGY	
	6.2 USE CASE DIAGRAM	17-22
	6.3 ACTIVITY DIAGRAM	
	6.4 DATABASE DESIGN	
7	IMPLEMENTATION	
	7.1 MODEL	23-39
	7.2 BACKEND	
	7.3 SCREEN SHOTS	
8	SOFTWARE TESTING	40-42
	8.1 MANUAL TESTING	
9	CONCLUSION	43
10	FUTURE ENHANCEMENTS	44
	APPENDIX A: BIBLIOGRAPHY	45-46
	APPENDIX B: PLAGARISM REPORT	47
	APPENDIX C: POSTER	48

LIST OF FIGURES

SL No	Figure Name	Page Number
1	CONTEXT DIAGRAM	16
2	PROCESS FLOW DIAGRAM	17
3	USE CASE DIAGRAM	19
4	ACTIVITY DIAGRAM	20
5	USER DATABASE	21
6	MEDICATION DETAILS DATABSE	22
7	IMAGE AUGMENTATION	23
8	CNN MODEL	24
9	MODEL ACCURACY	25
10	MODEL LOSS	25
11	FLASK SERVER CREATION	26
12	GLAUCOMA PREDICTION	27
13	GLAUCOMA PREDICTION MODEL SUMMARY	28
14	KERATOCONUS PREDICTION MODEL SUMMARY	29
15	CORNEAL ULCER PREDICTION MODEL SUMMARY	30
16	MEDICATION TRACKING	31
17	MONGODB CONNECTION	32
18	HOME PAGE	33
19	LOGIN PAGE	33
20	SIGNUP PAGE	34
21	GLAUCOMA PREDICTION PAGE	34
22	GLAUCOMA DETECTED	35
23	NO GLAUCOMA DETECTED	35
24	KERATOCONUS PREDICTION PAGE	36
25	KERATOCONUS DETECTED	36
26	NO KERATOCONUS DETECTED	37
27	CORNEAL ULCER PREDICTION PAGE	37
28	CORNEAL ULCER DETECTED	38
29	CORNEAL ULCER NOT DETECTED	38
30	MEDICATION REGISTRATION PAGE	39
31	MEDICATION DETAILS DISPLAY	39

LIST OF TABLES

SL NO	TABLE NAME	PAGE NUMBER
1	HARDWARE REQUIREMENTS	13
2	SOFTWARE REQUIREMENTS	13
3	IMAGE UPLOAD TEST CASE	40
4	MEDICATION TRACKING TEST CASE	41
5	DISEASE PREDICTION MODEL TEST CASE	42

INTRODUCTION

CHAPTER-1

INTRODUCTION

1.1 PROJECT DESCRIPTION

1.1.1 PROBLEM STATEMENT

In the domain of eye care, often due to late diagnosis of a particular disease leads to permanent vision loss in that particular patient. The existing methods are not very effective and may lead to incorrect diagnosis. This setback does not allow for patients to go for treatment methods other than surgery, which cannot be afforded by everyone.

The absence of a system that can detect such diseases in their early stages and also capable of tracking the recovery progress of a patient complicates the development of an advanced solution to this problem. The challenge lies in providing users with a reliable web application, capable of detecting diseases and also at the same time track a patient's medication status. Hence, the refined problem statement emphasizes the need for an advanced system focused on detecting the stage of a disease using retinal fundus images and corneal images, to enable early detection of diseases that will lead to permanent vision loss, and also capable of tracking the patient's medication enabling doctors to easily track the recovery progress.

1.1.2 PROPOSED SOLUTION

The Smart Eye Care is a web application that is integrated with multiple machine learning models to detect diseases that lead to permanent vision loss in their early stages, which allows the doctors to recommend the recovery progress for that particular stage of the disease and avoid permanent vision loss. The system employs a CNN model that runs on flask and processes the image uploaded by the user to detect the disease they are currently affected with and uses a Linear SVM model to detect the stage of the disease based on the features extracted from the image uploaded by the user. Also add a feature that allows users to enter their

medication detail and track their medication on a regular basis and send them scheduled reminders to remind them to take their medication at the prescribed time.

1.2 PURPOSE

The project aims to create a web application powered by machine learning models to detect the disease and the stage of the disease they are currently diagnosed with. Also provide the users with a system capable of tracking user's medication and updating them on the information of their medication, such as the name of the medication, the start date, the end date, the number of doses per day and the number of doses left. Based on this information it is easy for the doctors to track the recovery progress of the patient and schedule visits with them.

1.3 SCOPE

The project is designed to detect three diseases namely glaucoma, corneal ulcer and keratocunus using separate models and also it is limited to detecting the satges of the above mentioned diseases. It is also limited to tracking medications related to ocular diseases.

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 DOMAIN SURVEY

Machine Learning in healthcare is a domain that is expanding rapidly, machine learning models are being developed that are capable of detecting diseases in different parts of the human body. Users are able to use advanced machine learning algorithms to provide an accurate diagnosis to the patients, early detection of many life threatening diseases in their early stages is possible due to the introduction of machine learning into the healthcare domain.

Although machine learning in healthcare is a domain that is growing rapidly, very few systems exist that combine multiple models that detect diseases individually. One such field is eye care, there is a notable gap between developing individual models and combining them together under one system, that is also capable of tracking the patient's medication. The proposed system offers a solution to combine different models that are capable of detecting Glaucoma, Keratoconus, and Corneal Ulcer individually into a single system, thus improving the access of eye care and making it more easily available to patients and capable of tracking the patient's medication status, thus making it easier for doctors to track their patients' recovery progress.

While there have been many researches into developing such systems capable of detecting multiple diseases, very few have actually been implemented practically. This project seeks to successfully build such a system and thus reduce the gap in this field of research.

2.2 RELATED WORK

2.2.1 RESEARCH PAPER-1

Title: Convolutional Neural Networks for Automated Glaucoma Detection: Performance and Limitations

Authors: Akash Bayyana, Jeyanand Vemulapati, Sai Hemanth Bathula, Gangula Rakesh, Srilatha Tokala, Murali Krishna Enduri.

Publication Details: 06-08 July 2023, 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)

Summary: The proposed model uses Convolutional Neural Network(CNN), to process the image and detect whether or not glaucoma is present in that image. The proposed model in the paper seeks to evaluate the performance of different conventional methods against the CNN based model. The model explores the accuracy of different CNN architectures such as VGG16, Resnet or Inception, the detection of glaucoma based on retinal fundus images is used for this purpose. Using CNN the model achieved an accuracy value of 85.29%.

2.2.2 RESEARCH PAPER-2

Title: Glaucoma Detection Using Deep Learning Approach in Research Detection

Authors: Mr. Anil Pandurang Shinde, Ramandeep Singh, Mrs. Renuka Shantappa Anami, Noor Mohammed, R. Selvameena, Mrs. Ashwini Jotiram Patil.

Publication Details: 25-26 May 2023, 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)

DOI: 10.1109/ACCAI58221.2023.10200098

Summary: The proposed model in the paper uses Convolution organisations, which is a type of deep learning method to detect whether glaucoma is present in the fundus image provided to the model. The model uses a split based approach, where the primary design is split and uses the Cup-to-Disc Ratio(CDR) values to determine whether glaucoma is present. The early

detection of glaucoma shields the users from permanent vision loss. The proposed model uses DenseNet-201 model, which gives an accuracy of 89.73 in training and 85.80 in testing.

2.2.3 RESEARCH PAPER-3

Title: Keratoconus Detection Algorithm using Convolutional Neural Networks: Challenges

Authors: Alexandru Lavric, Valentin Popa, Cristina David, Cristian Costel Paval.

Publication Details: 27-29 June 2019, 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)

DOI: 10.1109/ECAI46879.2019.9042100

Summary: The proposed model aims to help in the early detection of Keratoconus, which might lead to a safer and faster method for the detection of this rare disease and varies based on the geographical area. The model uses corneal topography pattern images, elevation patterns, pachymetry map, PTA index as parameter to train and test the proposed model. The model uses CNN to assess the accuracy of using machine learning in detecting Keratoconus. Using CNN the model achieved an accuracy of 97.5%.

2.2.4 RESEARCH PAPER-4

Title: Detecting Keratoconus From Corneal Imaging Data Using Machine Learning

Authors: Alexandru Lavric; Valentin Popa; Hidenori Takahashi; Siamak Yousefi

Publication Details: 12 August 2020

DOI: 10.1109/ACCESS.2020.3016060

Summary: The aim of the proposed model was to develop a machine learning model capable of detecting keratoconus at early stages. A total of 25 models were implemented for this purpose. The accuracy ranged from 64% to 94%, the highest accuracy was achieved when a linear SVM model was implemented.

2.2.5 RESEARCH PAPER-5

Title: High-Performance Detection of Corneal Ulceration Using Image Classification with Convolutional Neural Networks

Authors: Jan Gross, Johannes Breitenbach, Hermann Baumgartl, Ricardo Buettner.

Publication Details: Proceedings of the 54th Hawaii International Conference on System Sciences | 2021

Summary: The proposed model aims to detect corneal ulcers which could lead to permanent vision loss, if not diagnosed and treated on time. The model uses fluorescein staining images of cornea for training the model. The model built achieves an accuracy of 92.3% in identifying a normal eye and an infected eye.

2.2.6 RESEARCH PAPER-6

Title: An Efficient Automated Corneal Ulcer Detection Method using Convolutional Neural Network.

Authors: Ashrafi Akram, Rameshwar Debnath

Publication Details: 2019 22nd International Conference on Computer and Information Technology (ICCIT)

Summary: The proposed system utilizes Haar Cascade Classifiers for eye detection and segmentation, followed by CNN for detecting the presence of corneal ulcers. If an ulcer is detected, the system segments the ulcer area using GrabCut, Hough gradient, and active contour techniques. To enhance the limited dataset, the authors applied data augmentation techniques, which improved the CNN model's accuracy rate to 99.43%, with sensitivity and specificity of 98.78% and 98.60%, respectively.

2.2.7 RESEARCH PAPER-7

Title: KerNet: A Novel Deep Learning Approach for Keratoconus and Sub-Clinical Keratoconus Detection Based on Raw Data of the Pentacam HR system.

Authors: Ruiwei Feng, Zhe Xu, Xiangshang Zheng, Heping Hu, Xiuming Jin, Danny Z. Chen, Ke Yao, Jian Wu.

Summary: KerNet is an end-to-end deep learning model that processes raw data from the Pentacam HR system, which typically provides ophthalmologists with corneal topographic maps and indices. The model consists of a multi-branch convolutional neural network with a multi-level fusion architecture, including low-level and high-level fusion, and spatial attention modules to focus on critical areas of the cornea. The KerNet model was used to perform an analysis on the dataset containing corneal topography images of Keratoconus, it had an accuracy of 94%

2.2.8 RESEARCH PAPER-8

Title: An adaptive threshold based algorithm for optic disc and cup segmentation in fundus images

Authors: Ashish Issac, M. Parthasarathi, Malay Kishore Dutta

Summary: The proposed model presents an image processing technique for segmenting the optic disc and cup in fundus images using adaptive thresholding. The algorithm utilizes features from the image, mean, and standard deviation, to isolate the optic nerve head region in both red and green channels. The optic disc is segmented from the red channel, and the optic cup from the green channel based on a determined threshold from the preprocessed image's histogram. The results are compared with ground truth marked by doctors, demonstrating a high accuracy rate of 92.06%.

2.2.9 RESEARCH PAPER-9

Title: Transfer Learning for Early and Advanced Glaucoma Detection with Convolutional Neural Networks

Authors: Ali Serener, Sertan Serte

Summary: The proposed model focuses on implementing deep learning models, specifically ResNet-50 and GoogLeNet, for the early detection of glaucoma using fundus images. The dataset comprised of 1544 fundus images, augmented to increase data volume and the evaluation was performed on the RIM-ONE dataset. Results indicated that GoogLeNet exhibited superior performance compared to ResNet-50 in both early and advanced glaucoma classification. The accuracy of the GoogLeNet model was 91% and the accuracy of the ResNet-50 model was 90%.

2.2.10 RESEARCH PAPER-10

Title: AI-Driven Keratoconus Detection: Integrating Medical Insights for Enhanced Diagnosis

Authors: Mostafa Keshavarz, Mohammad Javad Ahmadi, Shima Sadat Naseri, Parisa Ghorbani, Maryam Mohammad Zadeh, Hossein Farokh Pour, Seyed Farzad Mohammadi

Summary: The proposed model is used for the proper diagnosis of keratoconus, which is a ocular disease that leads to permanent vision loss and also increases the risk of refractive eye surgeries. This study introduces an image dataset acquired by the Pentacam device, which is commonly used in keratoconus diagnosis. These images included four eye topography images. In the diagnosis of keratoconus, ophthalmologists perform a comprehensive evaluation of corneal topography maps and assign different levels of significance to each map. This research paper presents a novel algorithm that utilizes AI to imitate this medical insight. For this purpose, a regression network has been integrated into the classification algorithm to obtain the importance degree for each map. The model uses VGG as the backbone and it yielded an accuracy of 95%.

2.2.11 RESEARCH PAPER-11

Title: Automatic classification for corneal ulcer using a modified VGG network

Authors: Ningbiao Tang, Hao Liu, Keqiang Yue, Wenjun Li, Xueying Yue

Summary: In the proposed model, various preprocessing techniques were applied to corneal ulcer images, including masking, adaptive histogram equalization (AHE), normalization, and data augmentation. A weighted loss function was introduced to address the issue of uneven data distribution, with weights assigned based on the number of samples in each category. The modified VGG network, featuring 5 CCM layers and a feature fusion layer, achieved an 88.89% classification accuracy for different types of corneal ulcers.

2.2.12 RESEARCH PAPER-12

Title: Deep Learning Based Unsupervised and Semi-supervised Classification for Keratoconus

Authors: Nicole Hallett, Kai Yi, Josef Dick, Christopher Hodge, Gerard Sutton, Yu Guang Wang, Jingjing You

Summary: The proposed model focuses on accurately classifying different stages of Keratoconus (KC) using machine learning models. The study utilized both supervised and unsupervised models on patient data to develop a classification system. The Variational Autoencoder (VAE) with a Gaussian mixture classifier successfully clustered corneal data into four classes with a high accuracy of 80.3%. The Multilayer Perceptron (MLP) model demonstrated promising potential for accurate KC diagnosis.

2.2.13 RESEARCH PAPER-13

Title: Automated Glaucoma Diagnosis Using Deep and Transfer Learning: Proposal of a System for Clinical Testing

Authors: Mohammad Norouzifard, Ali Nemati, Hamid GholamHosseini, Reinhard Klette, Kouros Nouri-Mahdavi, Siamak Yousefi

Summary: The proposed model aimed to use advanced computer models to detect glaucoma by analyzing eye images from two datasets. One model, InceptionResNet-V2, achieved a high accuracy of 92.3%, while the other, VGG19, struggled with overfitting. By adapting the models through transfer learning due to limited data, they were able to improve their performance. Overall, the proposed model shows that these computer models can help accurately diagnose glaucoma, offering hope for better treatment decisions to protect people's eyesight and quality of life.

2.3 Existing Systems

A large number of research has been conducted to bring machine learning into eye care systems, as such there exist many such efficient models capable of detecting the diseases that a patient is suffering from. Despite the abundance of models available, there exists very few systems that are capable of combining these models and detect multiple diseases in a single application. The project aims to address the availability of such systems by combining models that detect diseases in different parts of the human eye. By doing so the project improves the availability of eye care to help doctors in early detection of diseases and to easily track their patient's recovery progress.

2.4 TECHNOLOGY SURVEY

- **Python(v3.11)**

Python is a high level programming language, it's design philosophy emphasizes on object oriented programming with the use of indentation. It has many pre-defined libraries that help in developing machine learning models with ease.

- **Tensorflow(v2.15)**

Tensorflow is a free and open source software library for machine learning and artificial intelligence. It is mainly used to build and train deep learning based neural network models.

- **Flask(v3.0.0)**

Flask is a web framework written in python, for facilitating the development of web applications. It uses simple and expressive syntax making it easy to connect web applications with machine learning models.

- **ReactJS(v18.2.0)**

ReactJS is a frontend development framework used to develop dynamic websites. It provides an interactive experience for the users to upload their images and view the results on the webpage.

- **NodeJS(v18.18.2)**

NodeJS is a cross platform open source javascript framework. NodeJS allows users to use javascript to write command line tools for server side scripting. It allows the react frontend to connect with databases.

- **MongoDB(v6.0.11)**

MongoDB is a source available, cross platform, document oriented database program. MongoDB is a nosql database, it uses JSON format to store the data in the backend. It is highly compatible with javascript frameworks.

- **Jupyter Notebook(v7.06)**

Jupyter Notebook is a versatile interactive computing environment widely employed for machine learning tasks. It provides a user-friendly platform to develop and store code disease and stage prediction models. Its interactive nature allows for iterative development and seamless integration of machine learning algorithms.

2.5 FEASIBILITY STUDY

The project aims to utilize Machine Learning (ML) models to detect diseases such as Glaucoma, Corneal Ulcer, and Keratoconus in their early stages. Early detection is critical to preventing permanent vision loss in patients, and by employing advanced technologies, the project seeks to make the detection process much more efficient. The project requires high-quality, annotated datasets of retinal scans and other related images, which can be sourced from publicly available databases. The quality and accuracy of these datasets are crucial for the successful training and performance of the ML models.

Convolutional Neural Networks (CNNs) are employed for image analysis, as they are well-suited for this task. The primary function of the model is to analyze the images and predict the presence of the disease. To enhance image quality and facilitate faster analysis, image processing techniques such as augmentation, normalization, and noise reduction are incorporated. The ML model is integrated with a user-friendly website, providing a platform for users to easily interact with the system. An additional feature allows doctors to track the recovery progress of their patients, making the platform beneficial for ongoing patient management.

Operationally, the project is feasible due to the availability of the necessary technology and expertise. The key components required include high-performance servers and GPUs to train and run ML models efficiently, access to ML frameworks like TensorFlow and PyTorch, image processing libraries, and web development tools. A team of data scientists, ML engineers, software developers, and domain experts in ophthalmology will be essential. Data acquisition will be supported by establishing partnerships with medical institutions for access to high-quality retinal images and related data. Ongoing support will be necessary to update models, manage the website, and ensure data security and compliance with healthcare regulations.

Economically, the project shows strong potential for a high return on investment. Initial costs will include investment in hardware, software licenses, data acquisition, and personnel, while operational costs will cover ongoing expenses for server maintenance, cloud storage, and staff salaries. Revenue streams could come from subscription fees for the web platform, partnerships with healthcare providers, and licensing of the technology. Additionally, the scalable nature of the ML model allows for expansion to other ocular diseases and markets, increasing potential revenue.

In conclusion, the proposed project is feasible both operationally and economically. By leveraging ML models to detect ocular diseases, it has the potential to significantly improve the availability and accuracy of eye healthcare. This project not only aims to enhance early detection but also to provide a comprehensive tool for healthcare professionals to monitor and manage patient recovery.

HARDWARE AND SOFTWARE REQUIREMENTS

CHAPTER 3

HARDWARE AND SOFTWARE REQUIREMENTS

3.1 HARDWARE REQUIREMENTS

Table 3.1 Hardware Requirements

SPECIFICATION	DESIRED VALUE
Processor	Ryzen 5
Storage Space	256GB SSD or 1TB HDD
RAM	8GB

3.2 SOFTWARE REQUIREMENTS

Table 3.2 Software Requirements

SPECIFICATION	VERSION
ReactJS	18.2.0
NodeJS	18.18.2
Python	3.11
MongoDB	6.0.11
Jupyter Notebook	7.06
Tensorflow	2.15
Flask	3.0.0

SOFTWARE
REQUIREMENTS
SPECIFICATIONS

CHAPTER 4

SOFTWARE REQUIREMENTS SPECIFICATION

4.1 USERS

Includes both doctors and patients who wish to perform diagnosis based on the images uploaded and also track the status of their medication.

4.2 FUNCTIONAL REQUIREMENTS

- **User registration/login**

Users can create a new account by giving information such as name, email, etc. Once registered the users can login into the system by entering their email and password given during the time of registration.

- **Upload Images**

This feature allows users to upload images of their retinal scans, corneal topography images, etc., for disease prediction. If any format other than .jpg/.png/.jpeg is uploaded an error will be displayed

- **Prediction**

This feature allows user to predict the disease after the machine learning model analyzes it and then provides the result of the prediction from the image uploaded.

- **Enter Symptoms**

This feature allows users to upload any visual symptoms he is experiencing for detecting the stage of the disease. The format should be in text only, other values will not be accepted

- **Enter Medication Details**

This allows users to enter the details of their medication such as start date, end date, number of doses per day.

- **Track Progress**

Allows users to track the progress of their medication by displaying details such as number of days left and number of dosages remaining. This makes it easier for doctors and patients to track their recovery progress.

4.3 NON FUNCTIONAL REQUIREMENTS

- **Performance**

Refers to the speed with which the machine learning model provides predictions or responses to the user input. A high performance system responds promptly, ensuring a seamless user experience.

- **Security**

Includes verifying the user details during time of login and checking for the validity of the image uploaded by the user. Includes verifying the format for the images and ensuring the integrity of the images

- **Reliability**

Includes the reliability of the predictions made by the machine learning model. It should maintain the accuracy of predictions consistently over a period of time and varying conditions.

- **Usability**

The application can be used by everyone, as it does not have a very complicated process. Includes a userfriendly interface and intuitive navigation to ensure ease of users with varying levels of expertise.

SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

5.1 CONTEXT DIAGRAM

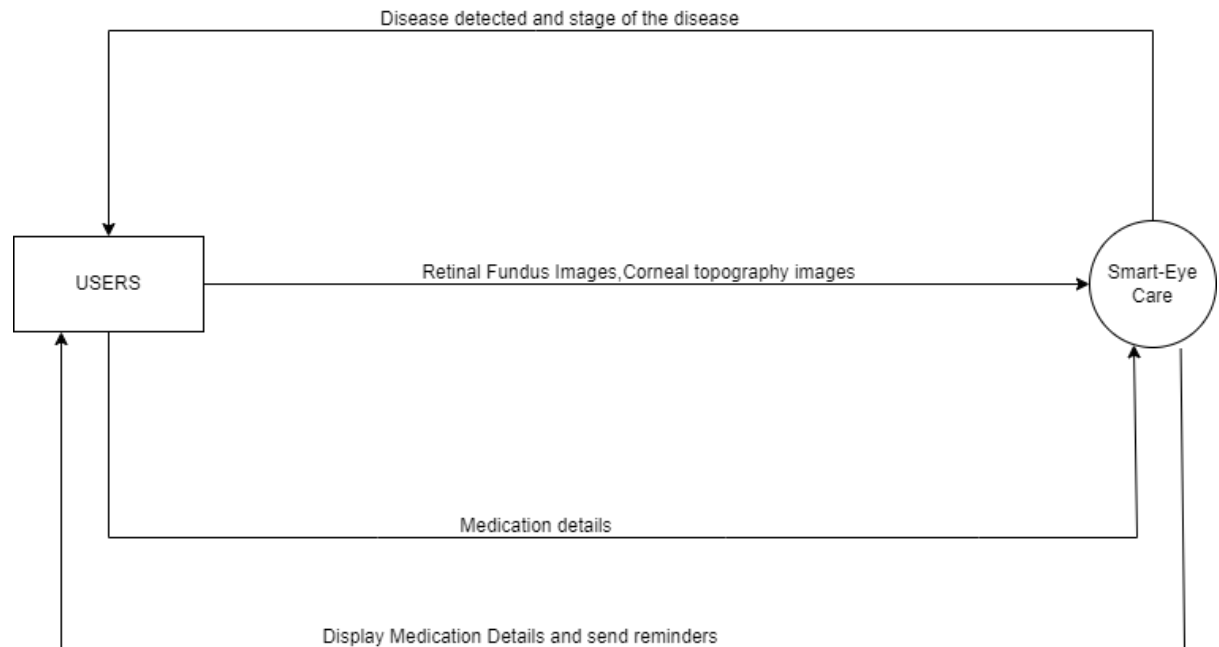


Fig 5.1: Context Diagram

The user uploads their fundus images or corneal images and visual symptoms to the website. The machine learning model processes the images uploaded and the visual symptoms entered by the user to predict the disease and the stage of the disease. The user can also upload their medication details and based on the medication details entered the system sends scheduled reminders to the users to remind them to take their medication.

DETAILED DESIGN

CHAPTER 6

DETAILED DESIGN

6.1 PROCESS FLOW DIAGRAM WITH METHODOLOGY

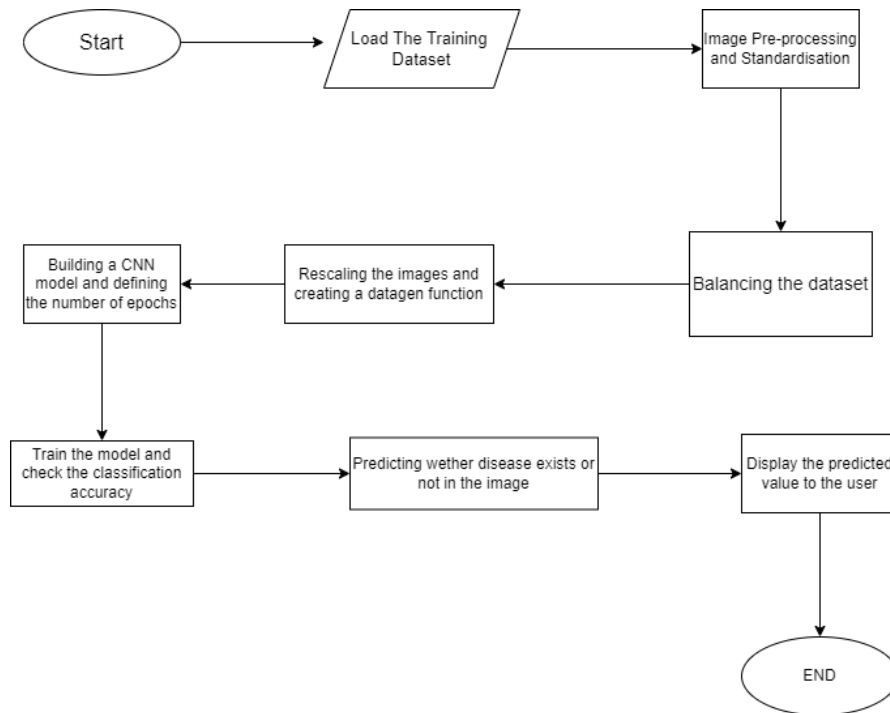


Fig 6.1: Process Flow Diagram for CNN

CNN MODEL METHODOLOGY

- **Load the Images dataset:** This involves providing the directory path where the images are present
- **Image Preprocessing:** Involves resizing the images to maintain uniformity, converting the images to grayscale(optional), enhancing the image, normalizing the images, and removing any unwanted watermarks etc.

- **Data Augmentation:** Involves generating new training samples by applying random transformations to the existing images such as rotation, flipping, zooming etc. This step is optional and is generally used to balance the data. It is a form of oversampling.
- **Train-test split:** Split the images into training and testing folders, which is then used to train and test the machine learning model.
- **CNN:** Apply the CNN model to train the images dataset by adding the required layers and specifying the number of epochs, and adding the required layers.
- **Predicting the values:** Using the trained model predict the values and display the output on the screen.

6.2 USE CASE DIAGRAM

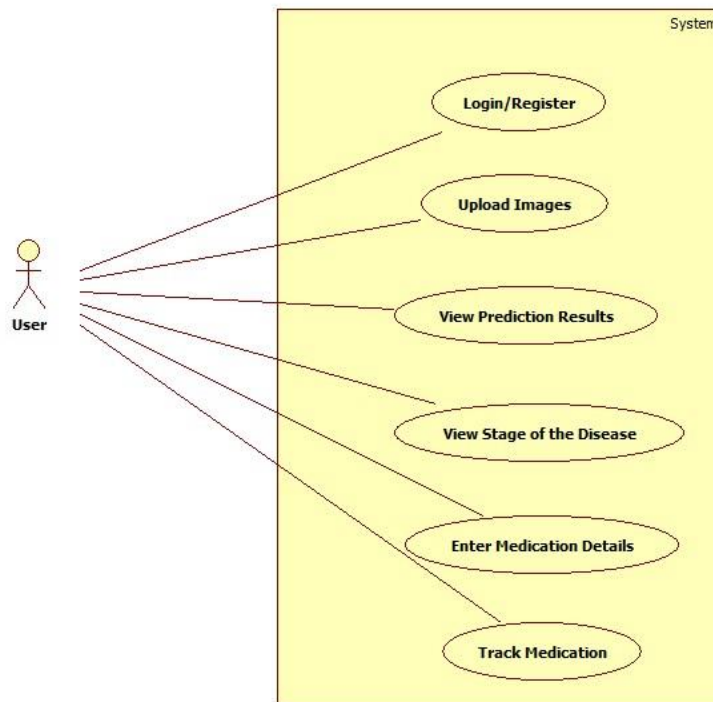


Fig 6.3 Use Case Diagram

The use case diagram illustrates the main functionalities of the system. The first functionality involves the user login or signup, the details entered during login are validated by checking in the database if such a user exists and during signup the data entered can be validated by checking if the data is entered in the correct format. After authentication the users can upload images to the system and the validity of the images is checked, once verified the images is processed by the machine learning model and the prediction result can be viewed by the user. Similarly the users can also enter the visual symptoms they are experiencing based on which the model processes the data and provides the stage of the disease. The user can also enter the details of the medication they are prescribed to the system. The users can also monitor their medication progress and get scheduled reminders from the system to remind them to take their medication.

6.3 ACTIVITY DIAGRAM

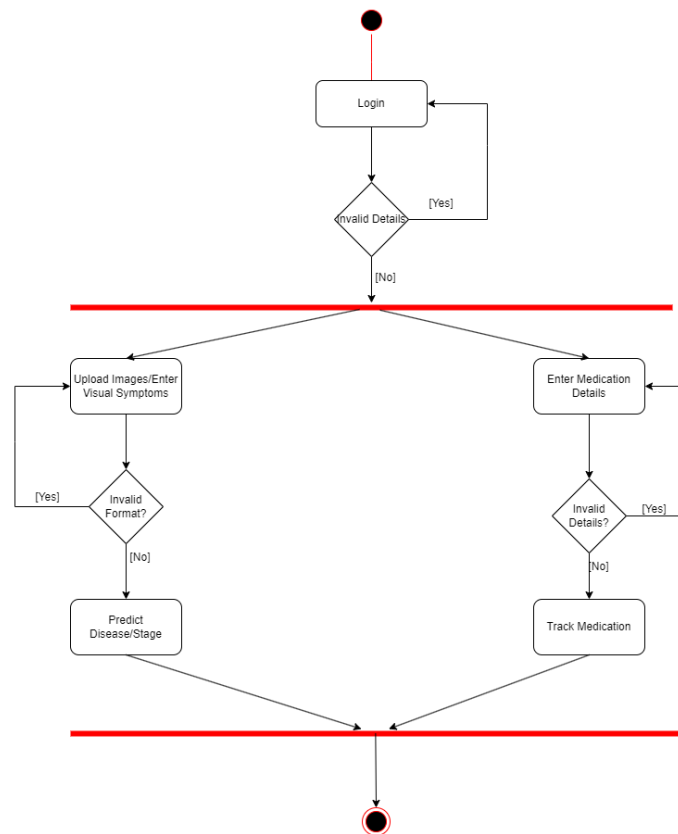


Fig 6.4 Activity Diagram

An activity diagram is a type of UML diagram that visualizes the flow of activities in a system or process. It uses rounded rectangles to represent activities, arrows to show transitions between activities, diamonds for decision points, bars for parallel processing, and nodes for the start and end of the process. Activity diagrams provide a concise and visual representation of how different actions interact within a system, making them valuable for modeling business processes, software workflows, and dynamic system behaviors.

The user enter their login details, if the entered details are invalid a message is displayed and the user is asked to enter the details again, if the entered details are correct the user can either choose to upload a image to predict the disease or they can enter the medication details prescribed to them. The system checks to see the format of the image uploaded, if it is an ivalid format no response is given and the user is askd to upload the image again. Similarly when entering the medication details, if the format of any field is not proper the user is asked to enter the details again.

6.4 DATABASE DESIGN

```
const UserSchema = new mongoose.Schema({
  patientid: {
    type: String,
    unique: true
  },
  name: {
    type: String
  },
  email: {
    type: String
  },
  password: {
    type: String
  },
  mobile_no: {
    type: Number
  },
  image: {
    data: Buffer,
    contenttype: String
  }
});
```

Fig 6.5 User Database

The Javascript code represents the structure of the user details database in MongoDB. Each user has a unique patientid and contains attributes such as name, email, password, mobile_no, image. This structure is designed to hold user details and the image uploaded by the user.

```
const MedicationSchema = new mongoose.Schema({
  patientid: {
    type: String,
    unique: true
  },
  medicationName: {
    type: String,
    required: true
  },
  startDate: {
    type: Date,
    required: true
  },
  endDate: {
    type: Date
  },
  numberOfDoses: {
    type: Number,
    required: true
  },
  dosesPerDay: {
    type: Number,
    required: true
  },
  daysToTake: {
    type: Number,
    required: true
  }
});
```

Fig 6.6 Medication Details Database

The above javascript code defines the structure of the medication details database stored in MongoDB. This database is used to track the medication of the patients, it consists of a unique patientid for every user and contains other attributes such as the name of the medication, the start date for the medication, the end date for the medication, the total number of doses, the number of doses to be taken each day and the number of days to take the medication.

IMPLEMENTATION

CHAPTER 7

IMPLEMENTATION

7.1 MODELS

IMAGE AUGMENTATION

```
data_gen=ImageDataGenerator(  
    rescale=1./255,  
    rotation_range=40,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    shear_range=0.15,  
    zoom_range=0.15,  
    horizontal_flip=True,  
    fill_mode='nearest')  
  
train_generator=data_gen.flow_from_directory(  
    train_dir,  
    target_size=img_size,  
    batch_size=batch_size,  
    class_mode='categorical',  
    shuffle=True,  
)  
  
validation_generator = data_gen.flow_from_directory(  
    val_dir,  
    target_size=img_size,  
    batch_size=batch_size,  
    class_mode='categorical',  
)  
  
test_generator = data_gen.flow_from_directory(  
    test_dir,  
    target_size=img_size,  
    batch_size=batch_size,  
    class_mode='categorical',  
)
```

Fig 7.1 Image Augmentation

This code defines the image augmentation which is a method of oversampling. It uses the ImageDataGenerator method to augment the images which is a method commonly used for keras models. It rescales all the images between 0 and 1, which is a common preprocessing step for classification models. It the randomly rotates the images by +/- 40 degrees, it also randomly shifts the images both width and height wise by 20%. It applies a random shear range, tilting the image vertically by 15 degrees, it also randomly zooms in or out on each image by 15%. It also randomly flips the image horizontally and it defines to fill the pixels lost during transformation by taking the value of the nearest pixel.

MODEL CREATION

```
model=Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(256,256,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

model.add(Flatten())

model.add(Dense(256, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dense(2, activation='sigmoid'))
```

Fig 7.2 CNN Model

This code defines the creation of a CNN model. The code defines a sequential model in which every layer are added in a linear stack. The model architecture includes 4 convolutional layers which are the basic blocks of a neural network, it defines the input size of the image as 256x256 and the ReLU activation function is used to introduce non linearity, initially 32 features are detected, then 64 features in the next convolutional layer, after that 128 features are detected and finally 256 features are detected. MaxPooling layer with a 2x2 pool size is added after every convolutional layer, this reduces the spatial dimensions, captures the dominant samples and controls overfitting. The BatchNormalization layer helps to stabilize training and speed up convergence. The flatten layer converts a 3D image to 1D format and prepares the image for dense layers. Two dense layers with ReLU activation and l2 regularizations are added, l2 regularization helps avoid overfitting. The dropout layer randomly drops out 25% of the connections during training and focuses the model to learn more on robust features. Final output layer with 2 units and sigmoid activation is added to produce probability like scores for each class.

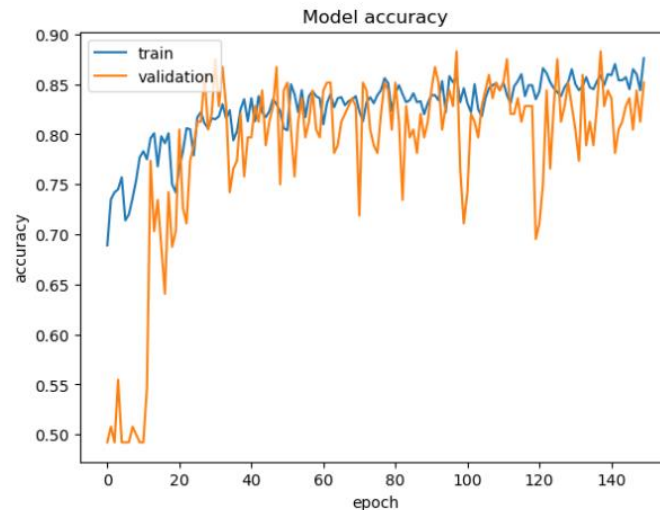


Fig 7.3 Model Accuracy

The above graph display the model accuracy for each epoch. Epoch is one complete pass over the dataset and that means every sample in the training dataset has been seen and used once to update the model's parameters. Multiple epochs are used to allow the model to learn and update the weights incrementally. The graph in the above figure displays the accuracy in each epoch and for different weights.

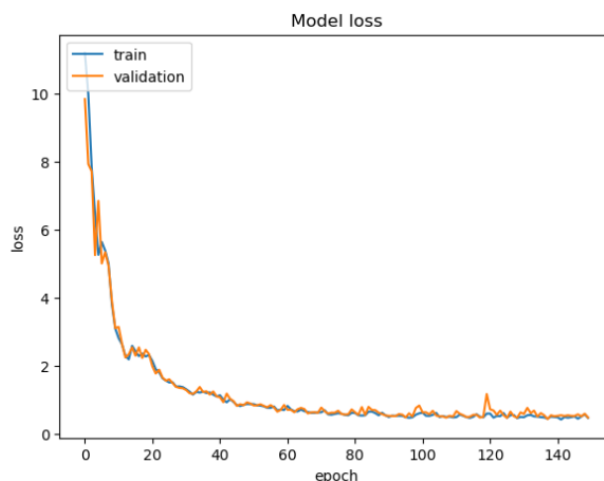


Fig 7.4 Model Loss

The above graph shows the loss for each epoch during both training and validation. Loss is used to measure how well the model's predictions match the actual target values. It represents the error between the predicted and the true outputs. The binary cross-entropy is used to calculate the loss in this case. There is a rapid decrease in the loss during the initial epochs and during the middle and later epochs the loss value stabilizes.

7.2 BACKEND

Creating Flask Server

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from tensorflow.keras.preprocessing.image import img_to_array, load_img
from tensorflow.keras.models import load_model
import numpy as np
import io

app = Flask(__name__)
cors = CORS(app, resources={r"/glaucoma_prediction": {"origins": "*"}})

try:
    loaded_model = load_model("D:/Project/ML/glaucoma_detection2.h5")
    print("Glaucoma Detection Model Loaded")
except Exception as e:
    print(f"Error loading model: {str(e)}")
```

Fig 7.5 Flask Server creation and loading the model

The above code creates a Flask Server and loads the glaucoma detection model from the device and it also defines the path for the model. It also defines CORS to allow image file uploaded from all origins to be taken as input. It also defines the route of the model, and tries to load the saved model and displays a message in case the model fails to load.

Glaucoma Prediction

```
@app.route('/glaucoma_prediction', methods=['POST'])
def glaucoma_prediction():
    try:
        print("Received request") # Print for debugging

        # Check for file in request
        if 'file' not in request.files:
            return jsonify({'error': 'No file provided'}), 400

        file = request.files['file']
        image_data = file.read()

        # Check for empty filename
        if file.filename == '':
            return jsonify({'error': 'No file selected'}), 400

        # Load and preprocess image
        test_image = load_img(io.BytesIO(image_data), target_size=(256, 256))
        print("Image loaded") # Print for debugging
        test_image = img_to_array(test_image)
        test_image = np.expand_dims(test_image, axis=0)
        test_image = test_image / 255.0 # Normalize pixel values

        # Make predictions
        predictions = loaded_model.predict(test_image)
        prediction_values = [float(value) for value in predictions[0]]
        print("Predictions made") # Print for debugging

        # Format result
        result = {
            'class_0_probability': prediction_values[0],
            'class_1_probability': prediction_values[1],
            'predicted_class': int(np.argmax(predictions))
        }

        return jsonify(result)
```

Fig 7.6 Glaucoma Prediction

This flask route is available at ‘/glaucoma_prediction’ and is using the POST method, it first checks if the file has been uploaded if no file has been uploaded it returns status 400 and displays that no file has been uploaded. Once the file has been uploaded, it is loaded and converted to an image array, then the pixel values are normalized and the image is sent to the loaded model for prediction. Once the image has been processed by the model, the result is converted to a JSON format and sent to the frontend page.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d_4 (MaxPooling2D)	(None, 127, 127, 32)	0
batch_normalization_4 (Batch Normalization)	(None, 127, 127, 32)	128
conv2d_5 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_5 (MaxPooling2D)	(None, 62, 62, 64)	0
batch_normalization_5 (Batch Normalization)	(None, 62, 62, 64)	256
conv2d_6 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_6 (MaxPooling2D)	(None, 30, 30, 128)	0
batch_normalization_6 (Batch Normalization)	(None, 30, 30, 128)	512
conv2d_7 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_7 (MaxPooling2D)	(None, 14, 14, 256)	0
batch_normalization_7 (Batch Normalization)	(None, 14, 14, 256)	1024
flatten_1 (Flatten)	(None, 50176)	0
dense_3 (Dense)	(None, 256)	12845312
dropout_1 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 128)	32896
dense_5 (Dense)	(None, 2)	258

Total params: 13268802 (50.62 MB)
 Trainable params: 13267842 (50.61 MB)
 Non-trainable params: 960 (3.75 KB)

Fig 7.7 Glaucoma Detection Model Summary

This is a summary of the model that has been implemented to detect glaucoma in the image provided by the user. It shows the total number of parameters, the total number of trainable parameters and the total number of non-trainable parameters. It also shows the different layers of the model such as conv2d, maxpooling, batchnormalization, flatten and dense layers. It shows the input shape of the image and the dimensions of the output from each layer.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_4 (MaxPooling2D)	(None, 111, 111, 32)	0
batch_normalization_4 (Batch Normalization)	(None, 111, 111, 32)	128
conv2d_5 (Conv2D)	(None, 109, 109, 64)	18496
max_pooling2d_5 (MaxPooling2D)	(None, 54, 54, 64)	0
batch_normalization_5 (Batch Normalization)	(None, 54, 54, 64)	256
conv2d_6 (Conv2D)	(None, 52, 52, 128)	73856
max_pooling2d_6 (MaxPooling2D)	(None, 26, 26, 128)	0
batch_normalization_6 (Batch Normalization)	(None, 26, 26, 128)	512
conv2d_7 (Conv2D)	(None, 24, 24, 256)	295168
max_pooling2d_7 (MaxPooling2D)	(None, 12, 12, 256)	0
batch_normalization_7 (Batch Normalization)	(None, 12, 12, 256)	1024
flatten_1 (Flatten)	(None, 36864)	0
dense_3 (Dense)	(None, 256)	9437440
dropout_1 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 128)	32896
dense_5 (Dense)	(None, 2)	258

Total params: 9860930 (37.62 MB)
 Trainable params: 9859970 (37.61 MB)
 Non-trainable params: 960 (3.75 KB)

Fig 7.8 Keratoconus Detection model Summary

The above image provides a summary of the CNN model used to detect Keratoconus in the image uploaded. It displays the layers used in the CNN model, the total number of parameters, the number of trainable parameters and the number of non-trainable parameters. It also displays the input shape of the image in this case it is (224,224) to the first layer of the CNN model. It also displays the output shape of the image from each layer to the next layer and the number of parameters taken as input in that particular layer. The different layers used are conv2d, maxpooling, batch normalization, flatten, dense and dropout.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d_4 (MaxPooling2D)	(None, 127, 127, 32)	0
batch_normalization_4 (Batch Normalization)	(None, 127, 127, 32)	128
conv2d_5 (Conv2D)	(None, 125, 125, 64)	18496
max_pooling2d_5 (MaxPooling2D)	(None, 62, 62, 64)	0
batch_normalization_5 (Batch Normalization)	(None, 62, 62, 64)	256
conv2d_6 (Conv2D)	(None, 60, 60, 128)	73856
max_pooling2d_6 (MaxPooling2D)	(None, 30, 30, 128)	0
batch_normalization_6 (Batch Normalization)	(None, 30, 30, 128)	512
conv2d_7 (Conv2D)	(None, 28, 28, 256)	295168
max_pooling2d_7 (MaxPooling2D)	(None, 14, 14, 256)	0
batch_normalization_7 (Batch Normalization)	(None, 14, 14, 256)	1024
flatten_1 (Flatten)	(None, 50176)	0
dense_3 (Dense)	(None, 256)	12845312
dropout_2 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 1)	129

Total params: 13268673 (50.62 MB)
 Trainable params: 13267713 (50.61 MB)
 Non-trainable params: 960 (3.75 KB)

Fig 7.9 Corneal Ulcer Detection Model Summary

The above image provides a summary of the CNN model used to detect Corneal Ulcer in the image uploaded. It displays the layers used in the CNN model, the total number of parameters, the number of trainable parameters and the number of non-trainable parameters. It also displays the input shape of the image in this case it is (256,256) to the first layer of the CNN model. It also displays the output shape of the image from each layer to the next layer and the number of parameters taken as input in that particular layer. The different layers used are conv2d, maxpooling, batch normalization, flatten, dense and dropout.

```

const MedicationDetails = () => {
  const [patientId, setPatientId] = useState('');
  const [medication, setMedication] = useState(null);
  const [error, setError] = useState(null);

  const handleChange = (e) => {
    setPatientId(e.target.value);
  };

  const handleSubmit = async(e) => {
    e.preventDefault();
    try {
      const response = await axios.get('http://localhost:3001/meddisplay?patientid=${patientId}');
      if (response.data.error) {
        setError(response.data.error);
        setMedication(null);
      } else {
        setMedication(response.data);
        setError(null);
      }
    } catch (error) {
      console.error("Error fetching medication:", error);
      setError("Server Error");
    }
  };

  const handleDoseTaken = async () => {
    if (medication.numberOfDoses > 0) {
      const updatedMedication = { ...medication, numberOfDoses: medication.numberOfDoses - 1 };
      try {
        const response = await axios.put('http://localhost:3001/med_update/${patientId}', updatedMedication);
        setMedication(response.data);
      } catch (error) {
        console.error("Error updating medication:", error);
      }
    }
  };
};

```

Fig 7.10 Medication Tracking

This code is used to search the medication a patient is taking using the unique patientid given to him. It displays the Medication Name, the Start date, the End Date and Number of Doses left. It also updates the doses after each day.

```

mongoose.connect('mongodb://127.0.0.1:27017/Smart_Eye_Care', {
});

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'MongoDB connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});

app.post('/register', (req, res) => {
  UserModel.create(req.body)
    .then(User1 => res.json(User1))
    .catch(err => res.json(err));
});

app.post('/login', (req, res) => {
  const {email, password} = req.body;
  UserModel.findOne({email:email})
    .then(user => {
      if(user) {
        if(user.password === password) {
          res.json("Success")
        } else {
          res.json("Incorrect Password")
        }
      } else {
        res.json("No such user Exists")
      }
    })
    .catch(err => res.json(err));
});

app.post('/med_register', (req, res) => {
  MedModel.create(req.body)
    .then(Med1 => res.json(Med1))
    .catch(err => res.json(err));
});

app.use(bodyParser.json());

app.get('/meddisplay', async (req, res) => {
  try {
    const { patientid } = req.query;
    if(!patientid) {
      return res.status(400).json({error: 'Patient ID is required'})
    }

    const medication = await MedModel.findOne({patientid});
    if (!medication) {
      return res.status(404).json({error: 'No medication details found for this patient ID'});
    }
    res.json(medication);
  } catch (error) {
    console.error('Error fetching medication:', error);
    res.status(500).json({error: 'Server Error'})
  }
});

```

Fig 7.11 MongoDB connection

The above figure displays the NodeJs code to connect the ReactJS pages to the MongoDB databases. It also defines multiple functions with app routes to connect the login and signup pages to the User Details database and the Medication registration and Medication display pages to the Medication Details database.

7.3 Screenshots



Fig 7.6 Home Page

This is the home page of the web application it has links to access the login and signup pages. This the page the user lands on once the application is started

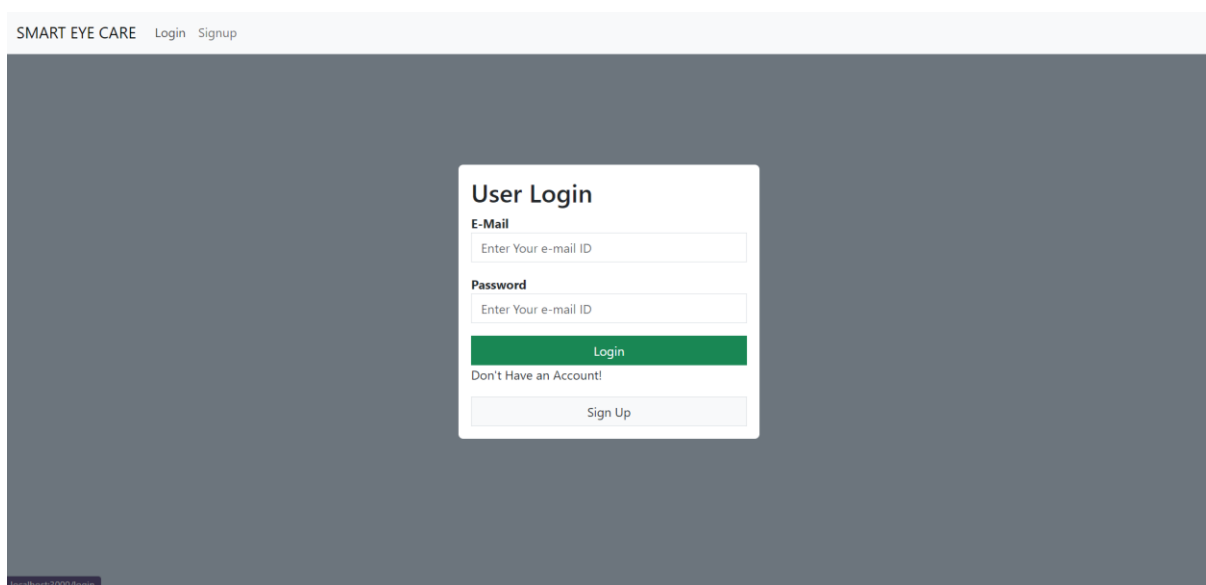
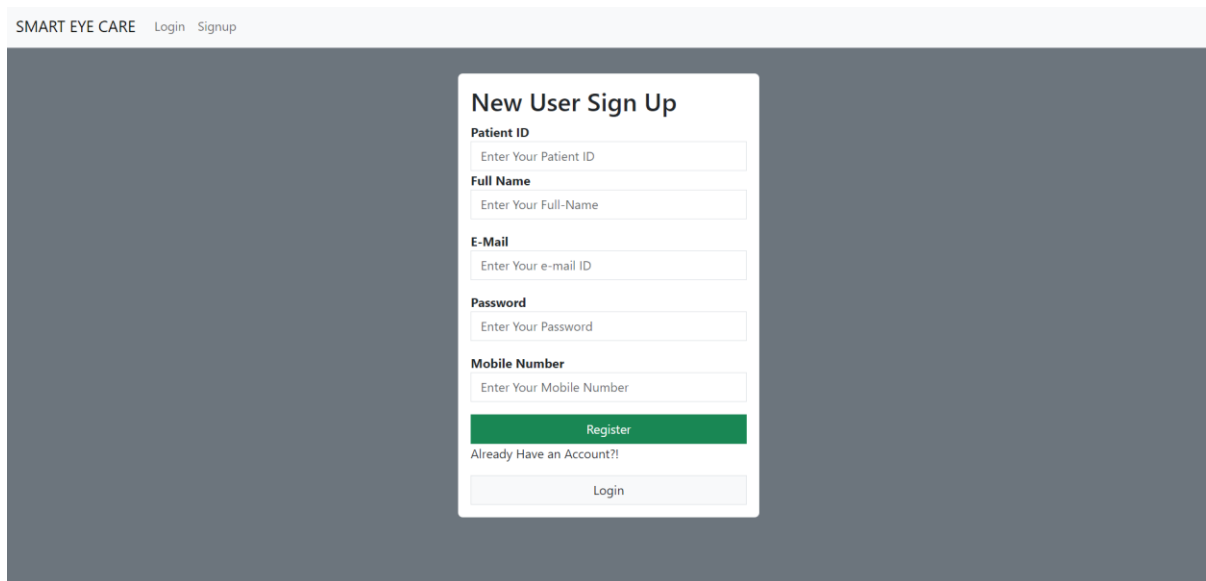


Fig 7.7 Login Page

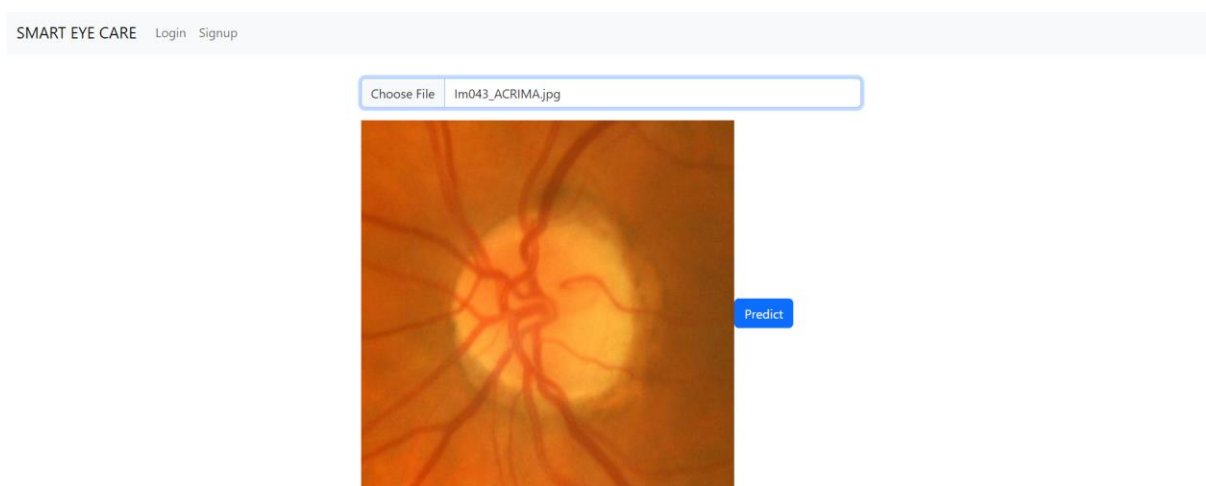
The above figger shows the login page, all the registered users can provide the registered email and password to login to the application and access the services of the application.



The image shows a web page titled "New User Sign Up" for "SMART EYE CARE". The page has a dark grey background. At the top, there is a navigation bar with "SMART EYE CARE", "Login", and "Signup". The main content area contains a white form with the following fields: "Patient ID" (with placeholder "Enter Your Patient ID"), "Full Name" (with placeholder "Enter Your Full-Name"), "E-Mail" (with placeholder "Enter Your e-mail ID"), "Password" (with placeholder "Enter Your Password"), and "Mobile Number" (with placeholder "Enter Your Mobile Number"). Below these fields is a green "Register" button. Under the "Register" button, there is a link "Already Have an Account?!" and a "Login" button.

Fig 7.8 Signup Page

The above figure shows the Signup page, through this page new users can be registered to the system. Once the user is registered he can login and access the services of the system.



The image shows a web page titled "Glaucoma Prediction Page" for "SMART EYE CARE". The page has a dark grey background. At the top, there is a navigation bar with "SMART EYE CARE", "Login", and "Signup". The main content area contains a white form with a "Choose File" button and a text input field containing "Im043_ACRIMA.jpg". Below the input field is a large orange image of a fundus. To the right of the image is a blue "Predict" button.

Fig 7.9 Glaucoma Prediction Page

The above figure shows the glaucoma prediction page. This page can be used to detect whether glaucoma is present in the image uploaded by the user. If the user uploads an invalid form of image the page displays an error.

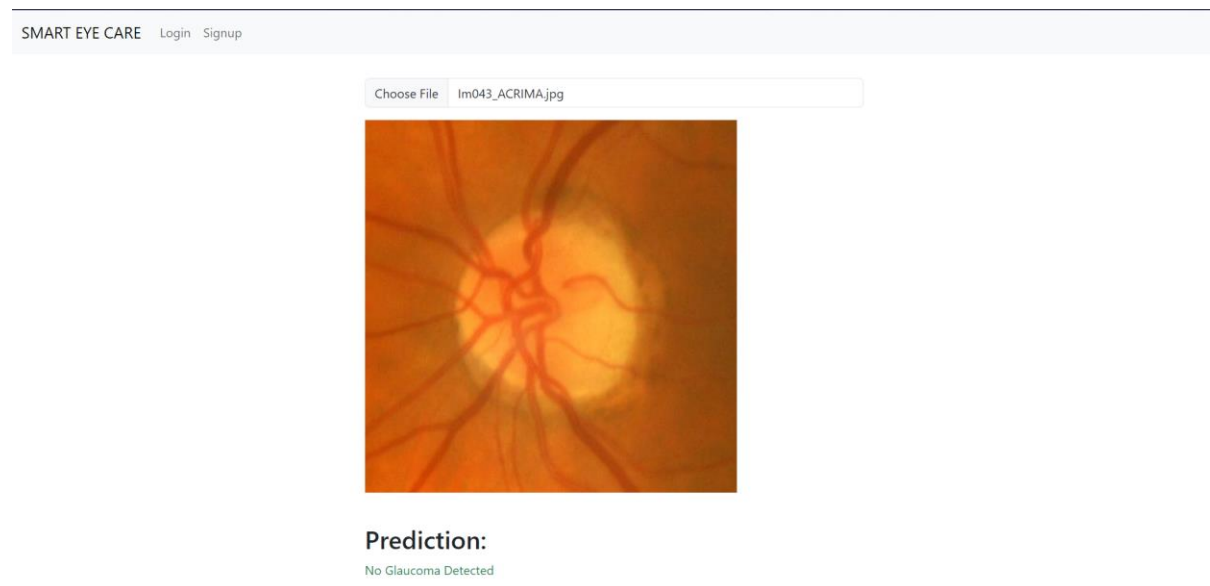


Fig 7.10 No Glaucoma Detected

The above figure shows an instance when the model does not detect glaucoma in the image uploaded by the user. In case glaucoma is not present the message is displayed in green

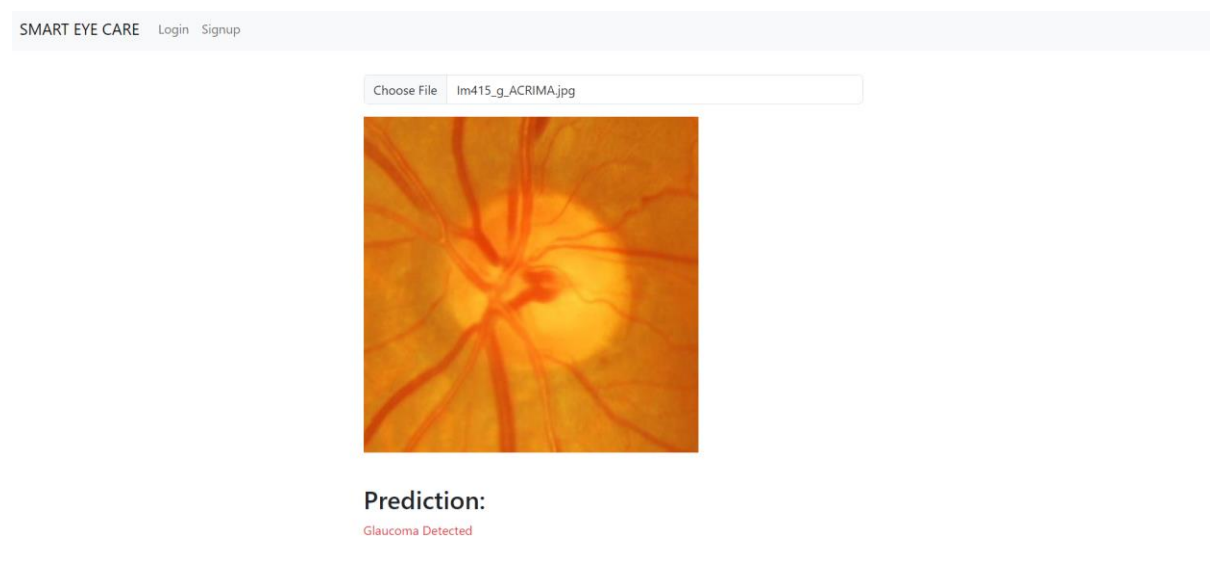


Fig 7.11 Glaucoma Detected

The above figure shows an instance when the model detects glaucoma in the image uploaded by the user. In case glaucoma is present the message is displayed in red.

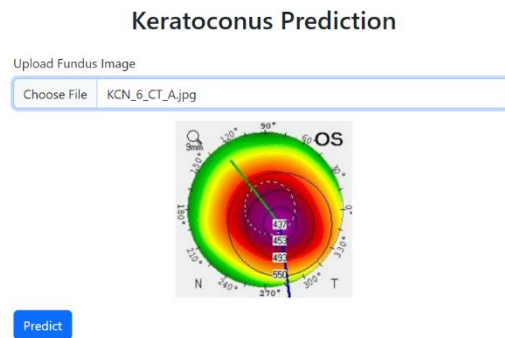


Fig 7.12 Keratoconus prediction page

The above figure shows the keratoconus prediction page. This page can be used to detect whether keratoconus is present in the image uploaded by the user. If the user uploads an invalid form of image the page displays an error.

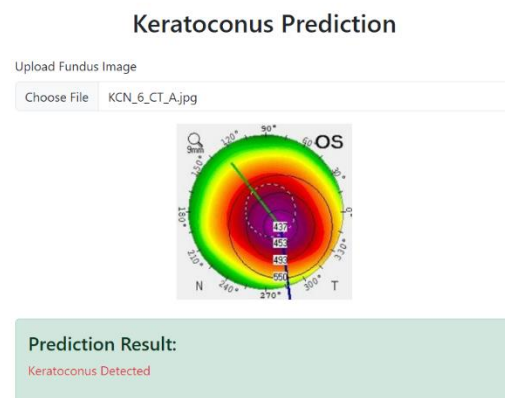


Fig 7.13 Keratoconus Detected

The above figure shows an instance when the model detects keratoconus in the image uploaded by the user. In case keratoconus is present the message is displayed in red.

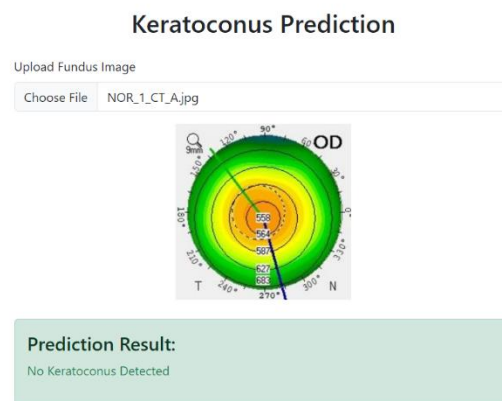


Fig 7.14 Keratoconus Not Detected

The above figure shows an instance when the model does not detect keratoconus in the image uploaded by the user. In case keratoconus is not present the message is displayed in green.

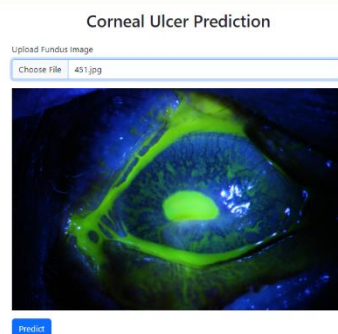


Fig 7.15 Corneal Ulcer Prediction Page

The above figure shows the corneal ulcer prediction page. This page can be used to detect whether corneal ulcer is present in the image uploaded by the user. If the user uploads an invalid form of image the page displays an error.

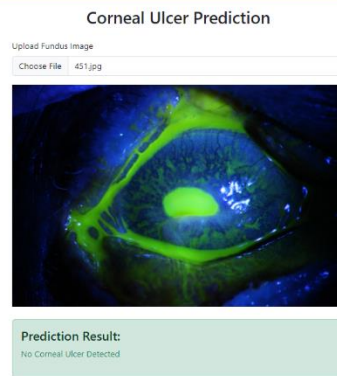


Fig 7.16 Corneal Ulcer Not Detected

The above figure shows an instance when the model does not detect corneal ulcer in the image uploaded by the user. In case corneal ulcer is not present the message is displayed in green.

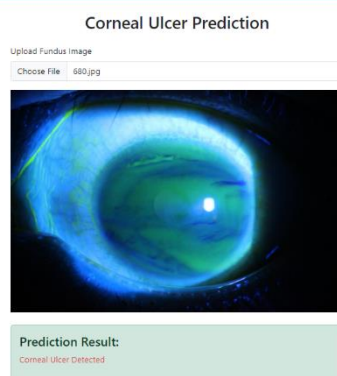


Fig 7.17 Corneal Ulcer Detected

The above figure shows an instance when the model detects corneal ulcer in the image uploaded by the user. In case corneal ulcer is present the message is displayed in red.

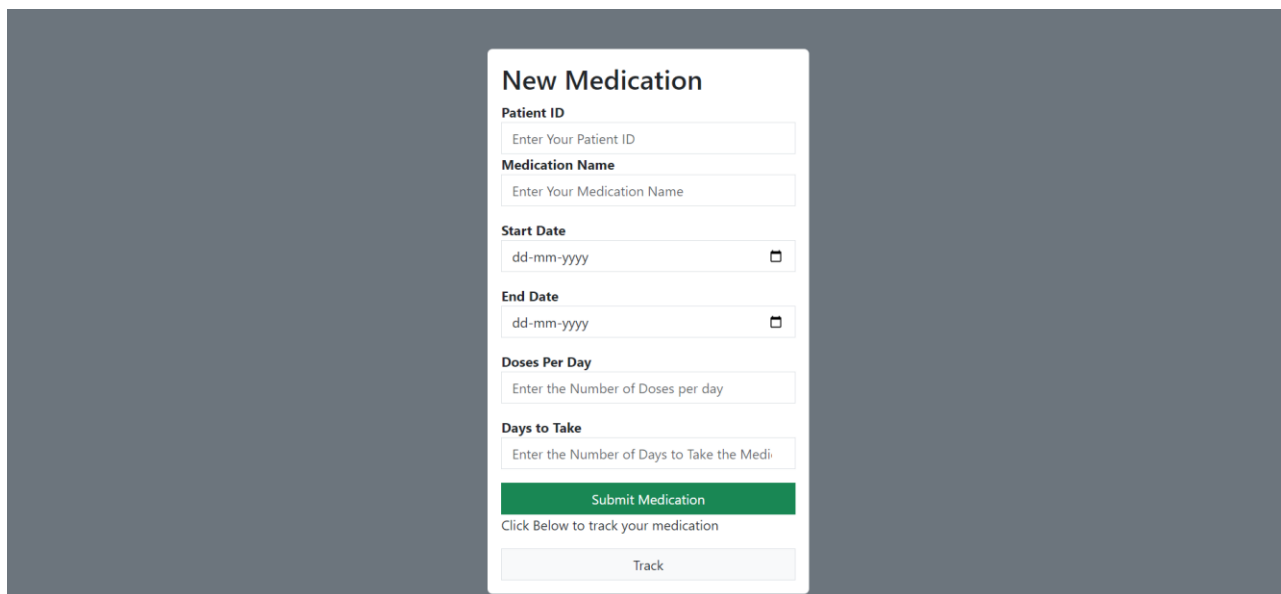
A screenshot of a web form titled "New Medication". The form is centered on a dark gray background. It contains several input fields: "Patient ID" with a placeholder "Enter Your Patient ID", "Medication Name" with a placeholder "Enter Your Medication Name", "Start Date" with a placeholder "dd-mm-yyyy" and a calendar icon, "End Date" with a placeholder "dd-mm-yyyy" and a calendar icon, "Doses Per Day" with a placeholder "Enter the Number of Doses per day", and "Days to Take" with a placeholder "Enter the Number of Days to Take the Medi". Below these fields is a green "Submit Medication" button. Underneath the button is the text "Click Below to track your medication" and a light gray "Track" button.

Fig 7.15 Medication Registration Page

The above figure shows the medication registration page, the user can store details of the medications he is currently taking through this page. It takes details such as the name of the medication the start and the end date, etc. based on the doses per day and the the number of days to take the medication it calculates and stores the total number of doses.

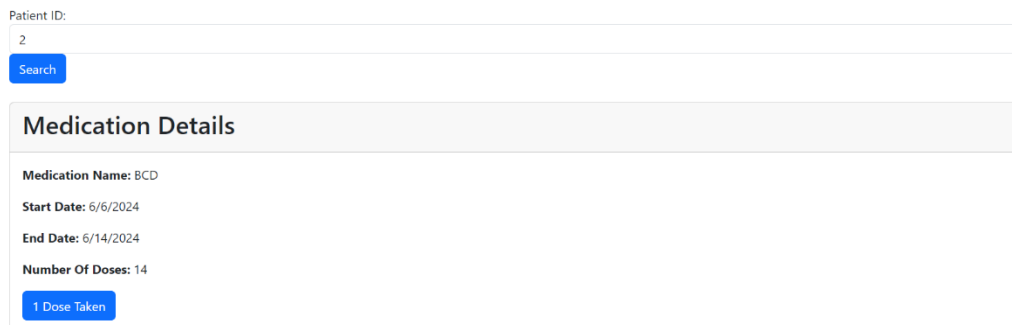
A screenshot of a web page for medication tracking. At the top, there is a "Patient ID:" label followed by a text input field containing the number "2" and a blue "Search" button. Below this is a section titled "Medication Details" in a light gray box. Inside this box, the following information is displayed: "Medication Name: BCD", "Start Date: 6/6/2024", "End Date: 6/14/2024", and "Number Of Doses: 14". At the bottom of this section is a blue button labeled "1 Dose Taken".

Fig 7.16 Medication Tracking Page

The above figure shows the medication tracking page. Through this page the user can daily update the number of doses left and the doctors can track the medication progress of the patients through this page.

SOFTWARE TESTING

CHAPTER 8

SOFTWARE TESTING

8.1 Manual Test Cases

Table 8.1 Image Upload Test Case

Test Scenario	Disease Prediction			
Test Case ID	Step Details	Expected Results	Actual Results	Pass/Fail
TC01	Navigte to http://localhost:3000/k_predict	Page should Open	Page Opened as Expected	Pass
TC02	Upload Image in the correct format	Disease must be predicted	Disease predicted as expected	Pass
TC03	Upload Image in the incorrect format	Error Message should be displayed	Error Message displayed as Expected	Pass
TC04	No Image Uploaded	Disease will not be predicted	Disease is not predicted as expected	Pass

Table 8.2 Medication Tracking

Test Scenario	Medication Tracking			
Test Case ID	Step Details	Expected Results	Actual Results	Pass/Fail
TC01	Navigate to http://localhost:3000/medreg	Page should open	Page opened as Expected	Pass
TC02	Enter a registered Patient ID	Medication Details should be displayed	Medication details displayed as expected	Pass
TC03	Enter unregistered Patient ID	Error message should be displayed	Error message displayed as expected	Pass

Table 8.3 Disease Prediction

Test Scenario	Disease Prediction Model			
Test Case ID	Step Details	Expected results	Actual result	Pass /Fail
TC01	Classify the unlabelled data	Unlabelled data should be stored in the labelled folders	Unlabelled data stored in the labelled folders	Pass
TC02	Split the labelled images into test,train and validation directories	Labelled images should be split into three directories	Labelled Images split into three directories	Pass
TC03	Define the input shape and use the images to predict the disease	Images should be resized and disease should be predicted without any bias	Images resized, but disease predicted with bias	Fail
TC04	Add regularizers and more layers to the model	There must be no bias	Bias does not exist	Pass

CONCLUSION

9. CONCLUSION

Traditionally, diagnosing eye diseases can be a lengthy process, requiring detailed examinations and interpretations by specialists. ML models can significantly speed this up. They can quickly analyze large volumes of images, providing initial diagnoses in just seconds. This rapid processing is particularly beneficial in community health clinics and other settings where access to specialized ophthalmologists might be limited. Moreover, these models can work alongside diagnostic tools, offering real-time feedback to healthcare providers. This not only speeds up the diagnostic process but also reduces waiting times for patients, making the whole experience more efficient and less stressful.

The impact of ML in eye care goes beyond just diagnosing diseases; it also opens up new possibilities for treatment. By analyzing a wide range of data—from genetic information to medical history and lifestyle factors—ML models can help create personalized treatment plans tailored to each patient. This personalized approach ensures treatments are more effective and come with fewer side effects. Additionally, ML plays a crucial role in the development of new therapies. By sifting through large datasets, ML can identify potential new drugs and predict how effective and safe they might be, speeding up the process of bringing new treatments to market.

ML also enhances the interaction between doctors and patients. Applications powered by ML can provide patients with easy-to-understand explanations of their conditions and treatment options, making complex medical information more accessible. Telemedicine platforms using ML enable remote consultations, allowing patients to get expert advice without having to travel, which is especially beneficial for those in remote or underserved areas. Furthermore, ML tools can help doctors communicate more effectively with patients by generating clear, concise summaries of medical reports, making it easier for patients to understand their health and treatment plans.

In summary, the integration of machine learning into eye care is transforming the field by enabling early detection of diseases, speeding up diagnoses, expanding treatment options, and improving communication between doctors and patients. This technological advancement promises more effective, timely, and personalized care, ultimately leading to better health outcomes for patients with eye diseases. As technology continues to advance, the partnership between ML and ophthalmology will likely grow even stronger, pushing the boundaries of what is possible in eye care and bringing hope to countless individuals worldwide.

FUTURE ENHANCEMENTS

10. FUTURE ENHANCEMENTS

Streamlining the integration of machine learning (ML) models into eye care can significantly enhance efficiency and accessibility. Here's how we can achieve this by making the system more compact, expanding its capabilities, and releasing a versatile mobile application.

To make the system more compact, we can reduce the number of models by creating a single model capable of identifying multiple eye conditions simultaneously. This approach simplifies the system, making it easier to manage and more efficient to run, while still maintaining high accuracy in detecting various diseases.

Expanding the disease detection capabilities involves training the model to recognize a broader range of ocular conditions. This can be done by gathering a large and diverse set of images representing different eye diseases and using advanced techniques to ensure the model learns effectively from these images. By doing so, we can ensure that the model is well-equipped to identify many different eye conditions accurately.

Developing a mobile application compatible with both Android and iOS is essential to ensure that this technology is accessible to as many people as possible. Using cross-platform development tools allows us to create an app that works seamlessly on both platforms. The app should have a user-friendly design, making it easy for users to navigate, capture retinal images, and understand their results.

APPENDIX A

BIBLIOGRAPHY(REFERENCES)

[1] Convolutional Neural Networks for Automated Glaucoma Detection: Performance and Limitations

<https://ieeexplore.ieee.org/document/10307182>

[2] Glaucoma Detection Using Deep Learning Approach in Research Detection

<https://ieeexplore.ieee.org/document/10200098>

[3] Keratoconus Detection Algorithm using Convolutional Neural Networks: Challenges

<https://ieeexplore.ieee.org/document/9042100>

[4] Detecting Keratoconus From Corneal Imaging Data Using Machine Learning

<https://ieeexplore.ieee.org/document/9165721>

[5] High-Performance Detection of Corneal Ulceration Using Image Classification with Convolutional Neural Networks

<https://scholarspace.manoa.hawaii.edu/handle/10125/71031>

[6] An Efficient Automated Corneal Ulcer Detection Method using Convolutional Neural Network.

<https://ieeexplore.ieee.org/document/9038389>

[7] KerNet: A Novel Deep Learning Approach for Keratoconus and Sub-Clinical Keratoconus Detection Based on Raw Data of the Pentacam HR system.

<https://ieeexplore.ieee.org/document/9429895>

[8] An adaptive threshold based algorithm for optic disc and cup segmentation in fundus images

<https://ieeexplore.ieee.org/document/7095384>

[9] Transfer Learning for Early and Advanced Glaucoma Detection with Convolutional Neural Networks

<https://ieeexplore.ieee.org/document/8894965>

[10] AI-Driven Keratoconus Detection: Integrating Medical Insights for Enhanced Diagnosis

<https://ieeexplore.ieee.org/document/10412506>

[11] Automatic classification for corneal ulcer using a modified VGG network

<https://ieeexplore.ieee.org/document/9361241>

[12] Deep Learning Based Unsupervised and Semi-supervised Classification for Keratoconus

<https://ieeexplore.ieee.org/document/9206694>

[13] Automated Glaucoma Diagnosis Using Deep and Transfer Learning: Proposal of a System for Clinical Testing

<https://ieeexplore.ieee.org/document/8634671>

Other References:

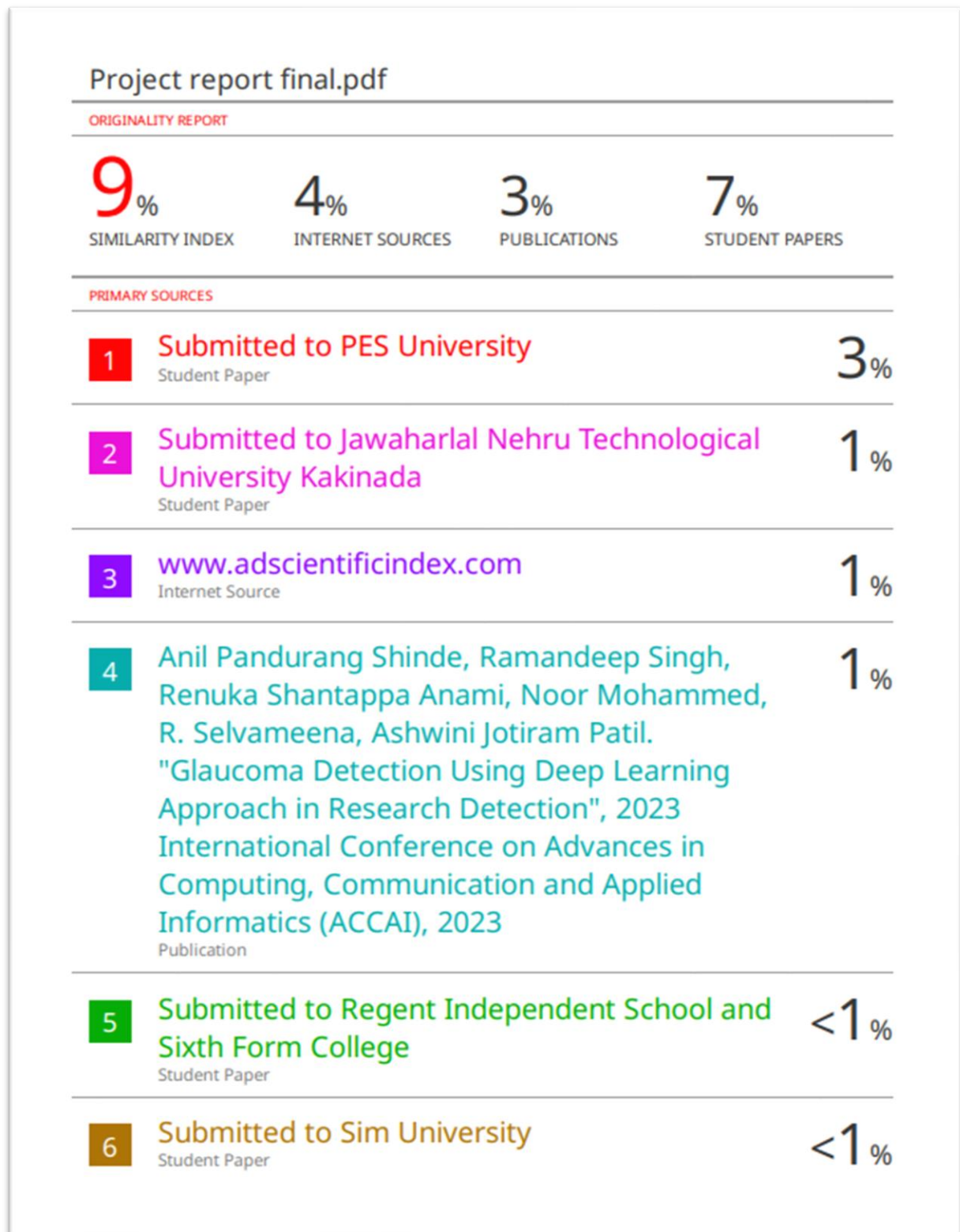
[1] IEEE Xplore.

<https://ieeexplore.ieee.org/Xplore/home.jsp>

[2] Google Scholar.

<https://scholar.google.com>

APPENDIX-B



APPENDIX C

SMART EYE CARE



PES
UNIVERSITY

By:

Pratham Shankar

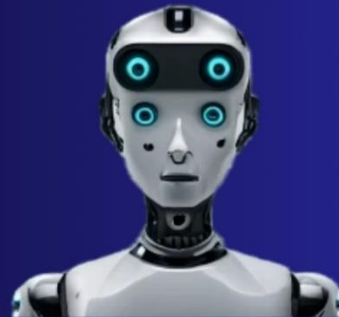
PES1PG22CA152

Guide:

MS Archana A
Assistant Professor

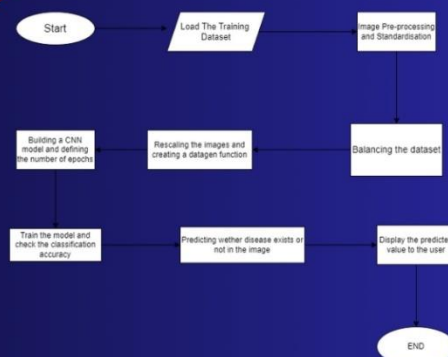
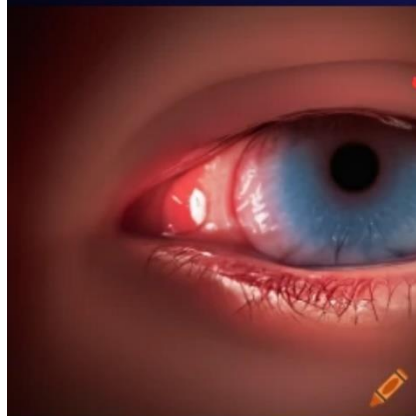
ABSTRACT:

Smart Eye Care is a ML based Web Application that aims to detect diseases in different parts of the human eye

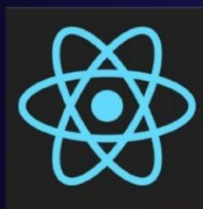


MACHINE LEARNING

Methodology



INPUT IMAGE



Conclusion and Results:

Smart Eye care System is an application that can be used to predict diseases and track the recovery progress of the patient with accuracy and ease

DISEASE DETECTED