Faculty of Computers and Artificial intelligence
Department of Computing and Bioinformatics

Cairo University

# OpticAI

## Revolutionizing eye care with AI


***Supervised by***
***Dr. Ibrahim Zaghloul***
***Dr. Desoky***
***TA. Mahmoud Hadad***

***Implemented by***

***Khaled Mohamed Ghanem***
***Anas Barakat Taha***
***Mahmoud Ahmed Mohamed***
***Ziad Ayman Mohamed***

Graduation Project

Academic Year 2022-2023

Documentation

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

1. **AI -** Artificial Intelligence
2. **CNN -** Convolutional Neural Network
3. **AMD -** Age-Related Macular Degeneration
4. **DR -** Diabetic Retinopathy
5. **UI -** User Interface
6. **UX -** User Experience
7. **ODIR5K -** Ocular Disease Recognition dataset
8. **APTOS 2019 –** Asia Pacific Tele Ophthalmology Society
9. **IDEs -** Integrated Development Environments
10. **GPUs -** Graphics Processing Units
11. **TPUs -** Tensor Processing Units
12. **ACC –** Accuracy
13. **GA -** Geographic Atrophy
14. **ERD –** Entity Relationship Diagram

# Chapter 1: Introduction

The field of medical diagnosis has witnessed significant advancements with the integration of artificial intelligence (AI) and machine learning techniques. These technologies have demonstrated great potential in improving the accuracy and efficiency of diagnostic processes. In this project, our focus is on developing and deploying five distinct Convolutional Neural Network (CNN) models for the diagnosis of various eye diseases, namely cataract, age-related macular degeneration (AMD), Mypoia, glaucoma, and diabetic retinopathy (DR).

The primary objective of this project is to leverage the power of CNN models to accurately classify eye images as either normal or indicative of a specific disease. By training individual CNN models for each disease, we aim to create a comprehensive diagnostic tool that can aid healthcare professionals in the early detection and treatment of these conditions. Early detection plays a crucial role in preventing further vision loss and ensuring appropriate medical interventions.

To ensure the accessibility and convenience of the diagnostic tool, the CNN models will be deployed on two platforms: a web application developed using Reactjs and a mobile application built with Flutter. This multi-platform approach allows users to seamlessly access the diagnostic capabilities of the CNN models from their preferred devices, be it a computer or a smartphone. By providing such accessibility, we aim to empower healthcare professionals to make faster and more accurate diagnoses, ultimately leading to timely interventions and improved patient outcomes.

The significance of this project lies in its potential to contribute to the field of ophthalmology by providing a reliable and user-friendly tool for eye disease diagnosis. By leveraging the power of CNN models and integrating them into user-friendly applications, we bridge the gap between cutting-edge technology and practical healthcare applications. The accurate diagnosis provided by these models can support healthcare professionals in making informed decisions and developing personalized treatment plans.

Throughout the documentation of this project, we will delve into the methodologies employed for model training, data collection and pre-processing, model evaluation, and the process of deploying the models on both the web and mobile platforms. By thoroughly documenting the project's implementation, we aim to facilitate knowledge

sharing and provide a valuable resource for future research and development in the field of AI-assisted medical diagnosis.

Furthermore, we will explore the potential impact of this project on the medical community. The integration of CNN models into the diagnostic process has the potential to revolutionize eye disease diagnosis by augmenting the capabilities of healthcare professionals and reducing diagnostic errors. By promoting accessible healthcare, we aim to extend the reach of diagnostic tools to underserved regions and populations, addressing the global challenge of limited access to specialized medical expertise.

In summary, this project presents an exciting opportunity to harness the power of deep learning and modern web and mobile technologies to develop a practical and impactful tool for eye disease diagnosis. By combining the capabilities of CNN models with user-friendly applications, we aim to improve diagnostic accuracy, enhance accessibility, and contribute to the advancement of healthcare in the field of ophthalmology.

## 1.1 Motivation

The motivation for this project stems from the significant impact of eye diseases on global health and the potential of artificial intelligence (AI) to improve diagnosis and treatment outcomes. Consider the following information:

- **Prevalence of Eye Diseases:** there are at least 2.2 billion people who have a near or distance vision impairment. Out of these, around 1 billion people, almost half of the total, have vision impairment that could have been prevented or remains unaddressed. This includes individuals with moderate or severe distance vision impairment or blindness caused by unaddressed refractive error (88.4 million), cataract (94 million), age-related macular degeneration (8 million), glaucoma (7.7 million), and diabetic retinopathy (3.9 million). Additionally, there are 826 million people with near vision impairment due to unaddressed presbyopia.[(1)]

- **Lack of Existing Diagnostic Solutions:** Despite the high prevalence of eye diseases and visual impairments worldwide, there is a lack of comprehensive diagnostic solutions that encompass the range of conditions mentioned earlier.

Many existing diagnostic methods focus on specific diseases or rely on manual assessment by ophthalmologists. However, a comprehensive AI-based diagnostic tool that covers multiple eye diseases, including unaddressed refractive errors, cataract, age-related macular degeneration, glaucoma, diabetic retinopathy, and unaddressed presbyopia, is yet to be developed.

- **Regional Differences:** The prevalence of distance vision impairment in low- and middle-income regions is estimated to be four times higher than in high-income regions. In terms of near vision impairment, rates of unaddressed cases are estimated to be greater than 80% in western, eastern, and central sub-Saharan Africa, while high-income regions of North America, Australasia, Western Europe, and the Asia-Pacific report lower rates, below 10%. **(1)**

- **Inconsistent Diagnoses:** In the field of ophthalmology, it is not uncommon for different doctors to provide contradictory diagnoses for the same patient. This inconsistency can arise due to variations in expertise, experience, and subjective interpretation of clinical findings. Such inconsistencies create confusion, delay in treatment initiation, and can potentially lead to suboptimal outcomes for patients.

- **Addressing Inconsistent Diagnoses and Silent Disease Progression:** One key aspect that sets this project apart is its aim to address the inconsistency of diagnoses provided by different healthcare professionals and the silent progression of certain eye diseases. Inconsistent diagnoses can lead to confusion, delayed treatment initiation, and suboptimal outcomes. Additionally, some eye diseases can progress silently, causing irreversible damage and blindness without noticeable symptoms. By developing an AI-based diagnostic tool, this project aims to provide standardized and generalized assessments, reduce inter-observer variability, and facilitate early detection to prevent avoidable vision loss.

- **Easy and Regular Diagnosis:** Regular diagnosis is crucial for managing eye diseases, particularly those that may progress silently. The project recognizes the importance of ease and accessibility in the diagnostic process. By implementing user-friendly AI-based diagnostic tools that can be deployed on various platforms, such as web and mobile applications, this project aims to

simplify the diagnostic process. This approach empowers individuals to perform self-assessments, seek timely medical attention, and enables healthcare professionals to conduct remote screenings. By promoting regular and easy diagnosis, the project strives to prevent irreversible vision loss, improve treatment outcomes, and enhance the overall quality of life for individuals at risk of eye diseases.

- **Comprehensive Diagnosis from a Single Image:** Traditionally, diagnosing different eye diseases often requires multiple tests and examinations, each focusing on a specific condition. However, this project aims to revolutionize the diagnostic process by leveraging AI technology to diagnose multiple eye diseases from a single fundus image. By training the AI-based diagnostic tool on a diverse dataset encompassing various conditions, including cataract, age-related macular degeneration, glaucoma, diabetic retinopathy, and unaddressed refractive errors, the system can learn to identify distinctive features and patterns associated with each disease.

- **AI Advancements in Medical Imaging:** Deep learning techniques, particularly Convolutional Neural Networks (CNNs), have revolutionized medical imaging analysis. CNN models excel in extracting meaningful features from images and making accurate predictions, making them well-suited for eye disease diagnosis. Studies have demonstrated the potential of CNN models in diagnosing various eye diseases with high accuracy and sensitivity.

## 1.2 Problem definition

### 1.2.1 Background:

Eye diseases such as cataract, age-related macular degeneration (AMD), Myopia, glaucoma, and diabetic retinopathy (DR) are major health concerns worldwide. Early detection and diagnosis of these diseases are crucial for effective treatment and prevention of vision loss. The current manual diagnostic process for eye diseases is often time-consuming, subjective, and can result in inaccuracies in diagnoses and treatments.

### 1.2.2 Problem Statement:

The problem that this project aims to address is the inefficiency and inaccuracies in the current manual diagnostic process for eye diseases. Despite advancements in medical technology, eye disease diagnosis is primarily performed by ophthalmologists manually examining patients' fundus images. This manual process is time-consuming, subjective, and can lead to incorrect diagnoses and treatments. Additionally, the increasing number of people suffering from eye diseases, particularly in developing countries, has resulted in a growing demand for efficient and accurate eye disease diagnosis systems. This demand has led to long wait times for patients and increased medical costs due to repeat visits and incorrect treatments.

### 1.2.3 Impact and Significance:

The project aims to have a significant impact by developing an artificial intelligence tool using Convolutional Neural Networks (CNNs) that can accurately classify different eye diseases. By automating the diagnostic process, the project intends to improve the accuracy and efficiency of eye disease diagnosis, reduce wait times for patients, and ultimately improve patient outcomes. This tool can provide a cost-effective solution, particularly in areas with limited access to ophthalmologists or where the demand for eye disease diagnosis exceeds the available resources. Early detection and timely treatment of eye diseases can significantly reduce the risk of vision loss and improve the overall quality of life for patients.

### 1.2.4 Scope and Limitations:

The scope of the project includes developing and training five distinct CNN models, one for each disease (cataract, AMD, Myopia, glaucoma, and DR). These models will be integrated into a web application using Reactjs and a mobile application developed using Flutter. The applications will allow users to upload fundus images and receive disease predictions. The project does not provide actual treatment or medical advice; rather, it serves as a screening tool for potential eye diseases.

However, it is important to note that the accuracy of the models' predictions may vary, and false negatives or false positives can occur. The effectiveness of the system will also depend on the quality of input images provided by users. Additionally, ethical considerations regarding data handling and user privacy must be addressed.

### 1.2.5 Success Criteria:

The success of the project will be evaluated based on the following criteria:

- **Accuracy:** The CNN models should achieve high accuracy in diagnosing the respective diseases, as measured by evaluation metrics such as precision, recall, and F1-score.

- **User Interface and Experience:** The web and mobile applications should provide a user-friendly and intuitive interface, allowing users to easily upload images and receive accurate disease predictions.

- **Performance and Scalability:** The deployed applications should demonstrate efficient performance and handle concurrent user requests without significant delays or crashes.

- **Feedback and Validation:** Collecting user feedback and validating the accuracy of predictions against expert opinions can help assess the real-world effectiveness of the system.

By meeting these success criteria, the project can significantly contribute to the early diagnosis and management of eye diseases, benefiting individuals and healthcare systems alike.

## 1.3 Project Objective (suggested solution)

To address the problem statement of diagnosing five different eye diseases (cataract, AMD, Myopia, glaucoma, and DR), we propose the development and deployment of five distinct CNN models, each specialized in diagnosing one specific disease. These models will be integrated into both a web application using the React framework and a mobile application developed with Flutter.

The suggested solution involves training and fine-tuning the CNN models using relevant datasets containing labelled images of normal and diseased eyes. The datasets will be pre-processed to ensure data quality and consistency. We will employ transfer learning techniques by leveraging pre-trained CNN architectures, such as VGG16 or ResNet, to expedite the training process and improve model performance.

The web application will be developed using React, a popular JavaScript library for building user interfaces. The application will provide an intuitive and user-friendly interface where users can upload or capture eye images for diagnosis. The web application will then utilize the trained CNN models to process the images and provide real-time diagnostic results.

Similarly, the mobile application will be developed using Flutter, a cross-platform framework that allows for seamless deployment on both iOS and Android devices. The mobile app will offer the same functionality as the web application, allowing users to easily access the diagnostic capabilities of the CNN models on their smartphones. The integration of the CNN models will ensure accurate and efficient disease diagnosis directly from mobile devices.

By deploying the CNN models on both web and mobile platforms, we aim to maximize the accessibility and usability of the diagnostic system. Users will have the flexibility to choose their preferred platform for accessing the application, making it convenient for both healthcare professionals and individuals concerned about their eye health.

The proposed solution not only addresses the challenge of diagnosing multiple eye diseases accurately but also leverages modern web and mobile technologies to provide a user-friendly and widely accessible platform. This approach has the potential to revolutionize the field of eye disease diagnosis, improving healthcare outcomes and facilitating early detection and intervention.

## 1.4 Gantt chart of project time plan

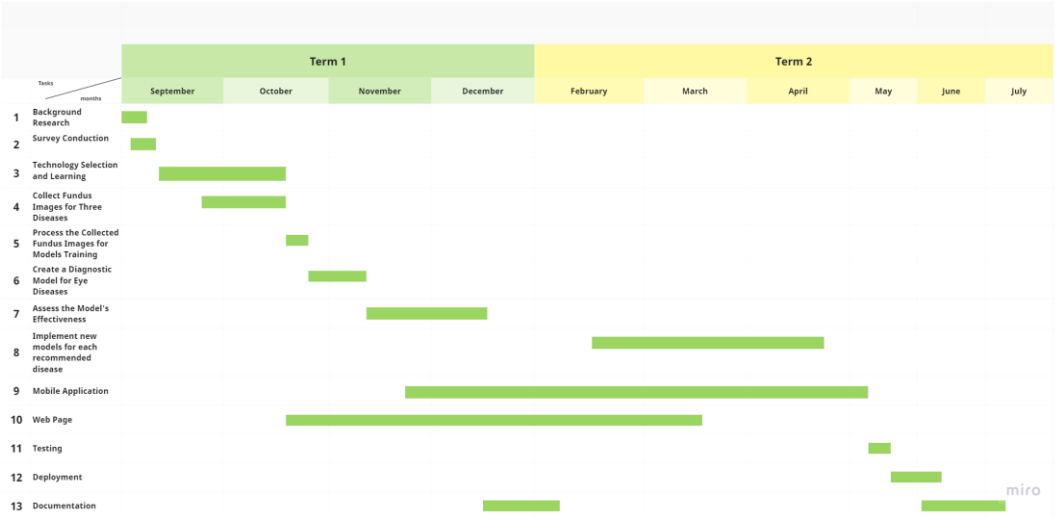



Figure 1

- **Background Research**

Before initiating the development process, we conducted comprehensive background research to gain a deeper understanding of the chosen eye diseases and the current state of CNN models in medical diagnostics. This research helped us lay a solid foundation for our project and make informed decisions throughout the development process.

Our research began by exploring each of the selected eye diseases individually. We studied the underlying causes, symptoms, and diagnostic approaches commonly employed by healthcare professionals. By gaining a thorough understanding of the diseases, we were able to identify the specific features and patterns that could be extracted from medical images. This knowledge served as a crucial guide in designing and training our CNN model for accurate and efficient diagnosis.

Throughout the research phase, we reviewed numerous academic papers, medical literature, and recent advancements in the field. This not only broadened our knowledge base but also enabled us to incorporate the latest findings into our model's architecture.

The research process also involved collaborating with domain experts, ophthalmologists, and medical researchers. Their valuable insights and feedback

further refined our approach and ensured the model's relevance and potential impact in real-world medical settings.

- **Survey Conduction**

Conduct surveys [(2)] to gather insights on the current methods of eye disease diagnosis. Identify the requirements for the project and develop the project concept.

- **Technology Selection and Learning**

To successfully complete the project, thorough research and careful selection of deep learning technologies and algorithms are crucial. This entails studying various deep learning frameworks and libraries like TensorFlow and Keras, as well as understanding neural network architectures and algorithms such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Analysing the project's goals, data types, and constraints will help make informed decisions about the most suitable technologies and algorithms.

Once the appropriate deep learning technologies and algorithms are selected, the next step is to acquire knowledge and expertise in those areas. This can be accomplished through self-study, online tutorials, or collaboration with experts in the field. By investing time and effort into learning and understanding the chosen technologies and algorithms, you will be better prepared to implement them effectively and efficiently in the project.

In addition, it is important to consider developing both a mobile app and a web application for the project. Flutter, a popular cross-platform framework developed by Google, is an excellent choice for mobile app development. Flutter allows you to write code once and deploy it on both iOS and Android platforms, saving development time and resources. By learning Flutter, you will gain the ability to create visually appealing and high-performing mobile applications with a single codebase.

For web development, React, a JavaScript library developed by Facebook, offers a robust and efficient solution. React enables the creation of interactive and dynamic user interfaces, making it ideal for building modern web applications. By learning React, you can leverage its component-based architecture and reusable UI elements to develop responsive and scalable web applications.

By combining knowledge and expertise in the selected deep learning technologies and algorithms with proficiency in Flutter for mobile app development and React for web development, you will have a solid foundation to create a comprehensive and successful project.

- **Collect Fundus Images for Three Diseases**

Obtain a comprehensive collection of fundus images from various online databases for each of the diseases: diabetic retinopathy, glaucoma, and cataracts.

- **Process the Collected Fundus Images for Models Training**

Clean and pre-process the gathered fundus images to remove any artefacts and irrelevant data. Ensure that the images are consistent in size and format, in preparation for training the models for each disease, including diabetic retinopathy, glaucoma, and cataracts, using appropriate pre-processing methods for each disease.

- **Create a Diagnostic Model for Eye Diseases**

Build deep learning models for each specific eye disease to diagnose fundus images. Fine-tune and test each model to enhance its accuracy.

- **Assess the Model's Effectiveness**

Evaluate the effectiveness of each model by using metrics such as accuracy, precision, recall, and F1-score. Compare the outcomes with previous models in the field to confirm the performance of the newly developed model.

- **Implement new models for each recommended disease**

Collect new fundus images for recommended disease such as Myopia, and age-related macular degeneration. And then apply appropriate pre-process the images to prepare them for model training, build deep learning models to diagnose each specific eye disease, and assess the models performance using metrics such as

accuracy, precision, recall, and F1- score and compare with previous models in the field.

- **Mobile Application**

The mobile application is designed to be user-friendly and intuitive, ensuring that users with varying levels of technical expertise can easily navigate through its features. The application is developed using the Flutter framework, which allows for seamless integration across different mobile platforms, including iOS and Android.

- **Web Page**

The web application is developed using the Reactjs framework, which provides a seamless and interactive user experience across different web browsers. The application follows a modern and user-friendly design, ensuring that users of all technical backgrounds can easily navigate its features.

- **Testing**

The testing phase of our disease diagnosis application is a critical step in ensuring its functionality, accuracy, and user experience. This phase involves a comprehensive series of tests conducted to identify and rectify any potential issues, validate the performance of the CNN models, and gather feedback to improve the application before its deployment.

- **Deployment**

The deployment phase is a critical step in bringing our disease diagnosis application to life and making it accessible to users. This phase involves preparing the application for deployment on both the web and mobile platforms, ensuring a smooth and seamless experience for users.

- **Documentation**

Documentation Create a detailed documentation of the project including the methodology, results, and conclusion. Document the code and algorithms used in the project.

## 1.5 Project development methodology

To ensure the successful development and deployment of the web application and mobile app for diagnosing eye diseases using CNN models, it is crucial to follow a systematic and efficient development methodology. The proposed development methodology for this project is as follows:

### 1.5.1 Project Planning and Scope Definition

The project's planning and scope definition involve the following key aspects:

- Clearly define the project's objectives, scope, and deliverables: This includes identifying the specific CNN models for each disease (cataract, AMD, Myopia, glaucoma, and DR) and their respective functionalities (diagnosing normal or patient). It is important to have a clear understanding of what the project aims to achieve and what will be delivered at the end.

- Set achievable milestones and timelines: The project plan will include a timeline that specifies milestones and deliverables for each phase of development. These phases may include model training, application design, integration, testing, and deployment. Setting realistic milestones and timelines helps ensure progress is made in a timely manner and keeps the project on track.

- Allocate sufficient time for key activities: It is crucial to allocate enough time for essential activities such as data collection and preprocessing, model training, and iterative development. Adequate time allocation allows for thorough and accurate completion of these tasks, which ultimately contributes to the success of the project.

- Employ project management techniques: Utilize project management methodologies such as Agile to facilitate effective collaboration, task tracking, and regular feedback cycles. Agile methodologies promote adaptability and flexibility, enabling the project team to respond to changes or challenges effectively. Communication channels, responsibilities, and decision-making processes should be clearly defined to ensure smooth progress throughout the project.

### 1.5.2 Data Collection and Pre-processing:

Data collection and pre-processing are critical steps in developing robust and accurate CNN models for disease diagnosis. Properly handling and preparing the dataset can significantly impact the model's performance. Here's an explanation of the pre-processing code provided, along with data augmentation:

- **Data Collection:**

  Gather a diverse and representative dataset containing images of healthy eyes and eyes affected by one of the five diseases. Combine the ODIR-5K[(0)] dataset for AMD, Cataract, and Myopia, the APTOS 2019 Blindness Detection[(3)] dataset for DR, and the Glaucoma[(0)] dataset for Glaucoma. The references provided above contain more information about the datasets and their usage for the respective diseases.

- **Data Pre-processing:**

  In the project, different pre-processing steps are applied to each disease's dataset to ensure optimal feature extraction and accurate disease diagnosis. Each disease may require specific pre-processing techniques tailored to its characteristics. Here are some examples of pre-processing steps commonly used for each disease:

  - **Convert the image to grayscale:** Grayscale images only have one channel (compared to three channels in RGB images) and are computationally more efficient for some image processing tasks.

  - **Unsharp Mask Filter:** The unsharp mask filter is used to sharpen the edges and enhance details in the image. It helps in improving the image's clarity and can be beneficial for the subsequent analysis by the CNN models.

  - **Crop Image from Gray:** This step likely involves removing any irrelevant background or border regions from the grayscale image, focusing on the critical part of the eye for disease diagnosis.

- **Resize Image:** Resizing the preprocessed image to a fixed image size (e.g., image_size x image_size) is essential to ensure all images fed to the CNN models have consistent dimensions, which is required for effective training and inference.

- **Shuffling the Dataset:** Randomly shuffling the dataset helps in creating a diverse training and testing set, reducing any bias that may be introduced due to the dataset's arrangement.

- **Data Augmentation:**

  Data augmentation is a technique used to increase the diversity of the training data by applying various transformations to the original images. This helps prevent overfitting and improves the model's generalization ability.

### 1.5.3 Model Training

Train individual CNN models for each disease using the collected dataset. Employ transfer learning techniques to leverage pre-trained models such as RESNET50 for Cataract, VGG19 for Glaucoma, InceptionV3 for DR, VGG19 for AMD, and VGG19 for Myopia. Fine-tune the pre-trained models on the specific eye disease classification task, adjusting the architecture and hyper-parameters as necessary to optimize performance. Training the models involves feeding the images into the network, computing the loss, and updating the model parameters using backpropagation and gradient descent optimization algorithms. Regularization techniques such as dropout and batch normalization may also be applied to prevent overfitting and improve generalization.

### 1.5.4 Model Evaluation

Evaluate the trained CNN models using appropriate evaluation metrics such as accuracy, precision, recall, and F1 score. Split the dataset into training, validation, and testing sets to assess the models' performance on unseen data. Perform rigorous testing to ensure the models can accurately classify between normal and disease-affected eyes for each specific eye disease. Additionally, conduct extensive analysis and comparison of the models' performance to identify the most accurate and reliable model for each disease.

### 1.5.5 Web Application Development

Build a web application using the Reactjs framework to provide an intuitive and user-friendly interface. Design a web page that allows users to upload eye images for diagnosis and display the results obtained from the trained CNN models. Ensure the web application is responsive and compatible with different devices and browsers.

### 1.5.6 Mobile App Development

Develop a mobile app using the Flutter framework, which enables cross-platform compatibility for both iOS and Android devices. Design an interface that facilitates the image upload process and displays the diagnosis results from the CNN models. Optimize the app for smooth performance and a seamless user experience.

### 1.5.7 Model Deployment

Deploy the trained CNN models on a server or cloud platform, allowing the web application and mobile app to interact with them for inference. Ensure the server infrastructure can handle concurrent requests and deliver fast response times to provide real-time diagnosis results.

### 1.5.8 Continuous Testing

Conduct rigorous testing throughout the development process. Perform unit testing, integration testing, and end-to-end testing to verify the functionality, accuracy, and performance of the system. Implement automated testing frameworks to streamline the testing process.

## 1.6 The used tools in the project (SW and HW)

The project employs a combination of software and hardware tools to develop and deploy the web application and mobile app for diagnosing eye diseases using CNN models. These tools provide the necessary infrastructure, development environment, and resources to ensure the smooth execution of the project. Let's explore the software and hardware tools utilized in detail.

### 1.6.1 Software Tools:

- **Programming Languages:** JavaScript and Dart The project utilizes JavaScript for developing the React web application and Dart for building the Flutter mobile app. JavaScript is a widely adopted programming language that is well-suited for web development, while Dart is the programming language specific to Flutter framework, offering simplicity and efficiency in mobile app development.

- **Frameworks:** React.js and Flutter React.js, a JavaScript library, is utilized for building the web application. It provides a component-based approach, allowing developers to create reusable UI components. Flutter, on the other hand, is an open-source UI toolkit by Google, used for developing cross-platform mobile applications. Flutter provides a rich set of widgets and offers a reactive programming style, facilitating the creation of visually appealing and responsive user interfaces.

- **Deep Learning Frameworks:** TensorFlow For training and deploying the CNN models, the project employs deep learning frameworks like TensorFlow. This framework offer extensive functionalities, including high-level APIs for building neural networks, model training, evaluation, and deployment. They provide pre-trained models, which can be used as a foundation for transfer learning, saving time and computational resources. TensorFlow is widely adopted in the deep learning community and offer excellent support for neural network development.

- **Development Environments:** Visual Studio Code, Android Studio, and Code Editors Development environments such as Visual Studio Code and Android Studio are used for web application and mobile app development, respectively. These integrated development environments (IDEs) provide code editors, debugging tools, and various plugins to enhance the development experience. Additionally, developers can use their preferred code editors to write, test, and debug the code.

- **Deployment Platforms:** Web Servers, App Stores, and Localhost
  The web application can be hosted on web servers or accessed locally through the localhost environment. Web servers, such as those provided by Amazon Web Services (AWS), Microsoft Azure, or DigitalOcean, offer cloud hosting services that ensure scalability and reliability for deploying web applications. Alternatively, developers can also run the application on their local machines using the localhost environment for testing and development purposes.

For mobile applications, distribution can still be achieved through app stores like the Google Play Store and Apple App Store, reaching a vast audience of mobile users. These app stores provide a centralized platform for users to discover, download, and update mobile applications.

- **Design and Prototyping:** Using Figma which provides a versatile and intuitive interface for designing the user interface (UI) and user experience (UX) of the web application and mobile app. With Figma's extensive library of design components, icons, and typography tools, the project team can create consistent and visually appealing UI elements. The collaborative nature of Figma allows multiple team members to work simultaneously on the design, ensuring efficient collaboration and real-time feedback.

- **Kaggle:** is an online platform that provides access to a wide range of datasets, competitions, and machine learning resources. In the project, Kaggle is utilized to acquire the Ocular Disease Recognition (ODIR5K) dataset and the Glaucoma dataset. Kaggle allows researchers and data scientists to explore, download, and contribute to various datasets, making it a valuable resource for gathering diverse and representative data related to eye diseases.

- **Google Colab:** is a cloud-based Jupyter notebook environment that offers free access to GPUs and TPUs. Colab provides a convenient platform for training CNN models as it offers powerful computational resources without the need for dedicated hardware. By utilizing Colab, the project can leverage the GPUs available on the platform to accelerate the model training process, reducing the time required for convergence.

- **Google Drive:** serves as a reliable cloud storage solution, offering ample space to store datasets, model weights, and other project-related files. By utilizing Google Drive, the project can efficiently manage and store the collected dataset, ensuring accessibility and data security. It also enables seamless collaboration between team members as they can easily share and synchronize files through shared Google Drive folders.

### 1.6.1 Hardware Tools:

- **Computers:** Powerful Computers or Servers The project requires a powerful computer or server with sufficient computational resources to handle tasks such

as training the CNN models and hosting the web application. These machines should have a suitable amount of RAM, processing power, and storage capacity to accommodate the dataset, perform intensive model training, and support the development environment.

- **Mobile Devices:** Android and iOS Devices To test and run the mobile app, physical devices or emulators/simulators are required. Physical devices running Android or iOS can be used to ensure the compatibility and functionality of the app across different platforms. Emulators or simulators allow developers to test the app on virtual devices with various configurations, screen sizes, and operating systems.

By utilizing these software and hardware tools effectively, the project can successfully develop and deploy the web application and mobile app for diagnosing eye diseases using CNN models. These tools provide the necessary resources, development environment, and infrastructure to streamline the development process, optimize model training, and ensure the seamless functioning of the application.

## 1.7 Report Organization (summary of the rest of the report)

The project stands out from previous works in eye disease diagnosis with its comprehensive disease coverage, integration with web and mobile platforms, improved accuracy, and the utilization of a single image for diagnosing multiple eye diseases. It covers a wide range of eye conditions, providing users with a thorough evaluation of their eye health. The integration with web and mobile platforms enhances accessibility and convenience for users. The project achieves high accuracy in diagnosing various eye diseases. Additionally, its unique approach of analysing a single image for multiple disease predictions saves time and effort, streamlining the diagnosis process and improving efficiency. Overall, this project offers a comprehensive, accessible, accurate, and efficient solution for assessing eye health.

The project specifications include functional requirements such as user account creation, login/logout, disease selection, image upload, pre-processing, segmentation, feature extraction, machine learning model utilization, disease prediction, result display, try without authentication, user management, data management, user feedback, confidence scoring, chat functionality, and search/filter functionality. The non-functional requirements encompass scalability, reliability, security, privacy, usability, accessibility,

performance, compatibility, maintainability, portability, support, and availability. These specifications ensure the development of a comprehensive, user-friendly, secure, and efficient system for eye disease diagnosis.

# Chapter 2: Related work

## 2.1 Closest examples of the project

- **Improved and robust Deep Learning agent for preliminary detection of Diabetic Retinopathy using public datasets [6]:**

This project focuses on utilizing a CNN model for the diagnosis of Diabetic Retinopathy (DR), a leading cause of preventable blindness among diabetic individuals worldwide, including India. The prevalence of DR in India has been increasing, and it is estimated that by 2030, India will have the highest number of diabetic patients globally. To address this issue, the project aims to enhance the early screening of DR by developing robust classification models using deep learning techniques.

The project utilized a publicly available dataset called EyePACS, consisting of 56,839 fundus images, for training and validation purposes. The problem was formulated as a binary classification task, distinguishing between DR of any grade (Grade 1-4) and No-DR (Grade 0). The trained models were evaluated on multiple datasets, including EyePACS, Messidor-2 (1,748 images), and Messidor-1 (1,200 images).

The achieved performance of the models was promising; with an Area under the Curve (AUC) of 0.92 on the Messidor-2 benchmark dataset. The sensitivity and specificity of the model were 81.02% and 86.09%, respectively. On the Messidor-1 dataset, the AUC, sensitivity, and specificity were 0.958, 88.84%, and 89.92%, respectively.

The project also addressed challenges related to automated disease detection in medical images using CNNs. These challenges encompassed the use of public datasets for training, pre-processing techniques, handling performance metrics for imbalanced classes, and a comparative analysis with existing studies.

- **Detection of Diabetic Retinopathy using Convolutional Neural Networks for Feature Extraction and Classification (DRFEC) [7]**

Diabetic Retinopathy (DR) is a condition that arises from Diabetes Mellitus and can cause damage to the retina, leading to vision impairment and, in severe cases, blindness. The manual process of diagnosing and detecting DR is time-consuming and unreliable due to resource limitations and the need for expert opinions. To address this, computerized diagnostic systems utilizing Deep Learning (DL) Convolutional Neural Network (CNN) architectures have been proposed to learn patterns from fundus images and classify the severity of DR.

This paper introduces a comprehensive model that evaluates the performance of 26 state-of-the-art DL networks for feature extraction and image classification of DR fundus images. The model compares the performance of different networks, such as ResNet50, Inception V3, EfficientNetB4, InceptionResNetV2, NasNetLarge, and DenseNet169, using the EyePACS fundus image dataset from Kaggle.

The results indicate that ResNet50 exhibits the highest overfitting, while Inception V3 shows the lowest overfitting during training. Among the evaluated models, EfficientNetB4 proves to be the most optimal, efficient, and reliable algorithm for DR detection, followed by InceptionResNetV2, NasNetLarge, and DenseNet169. EfficientNetB4 achieves a training accuracy of 99.37% and the highest validation accuracy of 79.11%. DenseNet201 achieves the highest training accuracy of 99.58%, but its validation accuracy of 76.80% is lower than the top-performing models mentioned above.

- **Deep Learning for Ocular Disease Recognition: An Inner-Class Balance** [8]

The identification of eye disorders through fundus pictures can be challenging for doctors. Manual diagnosis is time-consuming, prone to errors, and complex. Consequently, there is a need for an automated system aided by computer tools to detect various eye disorders using fundus images. Deep learning algorithms have significantly improved image classification capabilities, making such a system possible. This study introduces a deep learning-based approach for targeted ocular disease detection.

The study employed state-of-the-art image classification algorithms, including VGG-19, to classify the ODIR dataset. This dataset consists of 5000 fundus images categorized into eight different classes representing various ocular diseases. However, the dataset suffers from class imbalance issues. To address this, the study proposes converting the multiclass classification problem into a binary classification problem by using an equal number of images for each classification. The binary classifications were then trained using VGG-19.

The accuracy of the VGG-19 model was impressive, achieving 98.13% accuracy for normal versus pathological myopia, 94.03% for normal versus cataract, and 90.94% for normal versus glaucoma. Balancing the data improved the accuracy of all the models evaluated in the study.

In summary, the study highlights the challenges faced by doctors in early detection of eye disorders using fundus images. It emphasizes the need for an automated system supported by deep learning algorithms. By employing VGG-19 and addressing class imbalance, the study achieves high accuracy in classifying different ocular diseases.

- **Attention Based Glaucoma Detection: A Large-Scale Database and CNN Model [(0)]**

In recent times, the attention mechanism has been successfully used in convolutional neural networks (CNNs) to enhance computer vision tasks. However, its application in medical image recognition for glaucoma detection is limited. This study introduces an attention-based CNN for glaucoma detection (AG-CNN) to address this issue. A large-scale attention-based glaucoma (LAG) database is created, containing 5,824 fundus images labeled as positive (2,392) or negative (3,432) for glaucoma. Ophthalmologists' attention maps are collected through a simulated eye-tracking experiment. AG-CNN comprises an attention prediction subnet, a pathological area localization subnet, and a glaucoma classification subnet. Unlike other attention-based CNN methods, AG-CNN also visualizes the localized pathological areas, leading to improved glaucoma detection performance. Experimental results demonstrate the significant advancement of the proposed AG-CNN approach in glaucoma detection.

- **Bagging of Convolutional Neural Networks for Diagnostic of Eye Diseases [(10)]**

This paper focuses on the application of deep learning techniques, specifically convolutional neural networks (CNNs), for multi-class classification of eye diseases. The authors compare the performance of three models: CNN, VGG16, and InceptionV3, both individually and using a bagging ensemble approach. They utilize a dataset containing four types of eye diseases: Diabetic retinopathy, Glaucoma, Myopia, and Normal.

The experiments demonstrate that using a bagging ensemble improves the predictive efficiency of the models compared to standalone learning algorithms. The authors employ the confusion matrix to identify areas of confusion in the classifiers' predictions.

The study highlights the performance of bagging ensemble methods in combination with deep convolutional neural networks for eye disease classification. The authors observe that the ensemble approach outperforms traditional methods relying solely on learning algorithms. They compare the accuracy and effects of model assembly using CNN, VGG16, and InceptionV3, with InceptionV3-based bagging ensemble achieving the highest classification accuracy (87.20%).

Recommendations are made to utilize deep learning networks such as AlexNet or ResNet in combination with InceptionV3 for improved accuracy. The authors also emphasize the need to address confusion among the models, particularly in predicting Glaucoma, to optimize the classification process.

| Model | Type of ensemble | Prediction Accuracy |
|---|---|---|
| Three model of CNN | Bagging | 77.1% |
| Three model of VGG16 | Bagging | 79.8% |
| **Three model of InceptionV3** | **Bagging** | **87.2%** |
| CNN, VGG16 and InceptionV3 | Bagging | 86.5% |

Table 1

- **Age-related Macular Degeneration detection using deep convolutional neural network [(11)]**

The prevalence of Age-related Macular Degeneration (AMD) is increasing due to the aging population, necessitating early detection to prevent vision loss in the elderly. However, conducting comprehensive eye screenings for AMD in this population can be challenging. It was addressed by developing a deep Convolutional Neural Network (CNN) model with fourteen layers to automatically and accurately diagnose AMD at an early stage. The model's performance was evaluated using blindfold and ten-fold cross-validation strategies, achieving an accuracy of 91.17% and 95.45%, respectively. This novel model enables rapid and cost-effective screening for early detection of AMD in the elderly, and its portability allows for its use in various settings.

## 2.2 Main differences between them and this project

- **Comprehensive Disease Coverage**

One of the key differentiators of this project is its comprehensive disease coverage. While the other projects focused on diagnosing two or three specific eye diseases, this project stands out by offering a solution that covers a broader spectrum of eye conditions. By including six diseases, namely cataract, glaucoma, diabetic retinopathy, AMD, and myopia, this project provides users with a more comprehensive assessment

of their eye health. This comprehensive approach enables users to receive a more thorough evaluation of their eye conditions, leading to improved early detection, monitoring, and management of various eye diseases.

- **Integration with Web and Mobile Platforms**

  Unlike the other projects that primarily focused on developing CNN models for disease diagnosis, this project takes an additional step by integrating these models into a user-friendly web application and mobile app. This integration enhances the usability and accessibility of our solution. Users can conveniently access the eye disease diagnosis system through a web browser or a mobile device, allowing them to perform self-assessment of their eye health at their convenience. The integration with web and mobile platforms ensures that users can access the system from a wide range of devices, making it more accessible to a broader user base.

- **Improved Accuracy**

Accuracy is a crucial factor in the effectiveness of any disease diagnosis system. This project demonstrates impressive accuracy rates for each of the six diseases it diagnoses. With specific accuracies of 99% for cataract, 99% for glaucoma, 95% for diabetic retinopathy, 84% for AMD, and 96.5% for myopia, these models exhibit strong performance in correctly classifying and diagnosing these conditions. These high accuracy rates are a testament to the robustness and effectiveness of the CNN models used in this project. Achieving such accuracy levels is critical for building user trust and ensuring the reliability of the diagnosis system. It further emphasizes the practicality and usefulness of our project in assisting users in early disease detection and management.

**Accuracy comparison:**

| | Related work | | | Our work | | |
|---|---|---|---|---|---|---|
| **Disease** | **Data** | **Model** | **ACC** | **Data** | **Model** | **ACC** |
| **Cataract** | ODIR5K | VGG-19 | 94.03 | ODIR5K | ResNet50 | 99% |
| **Glaucoma** | ODIR5k | VGG-19 | 90.94% | kaggle | VGG19 | 99% |
| **DR** | Messidor | --------- | 95.8% | Aptos2019 | InceptionV3 | 95% |
| **AMD** | Private data | Nine-layer CNN model | 91.17% | ODIR5K | VGG19 | 84% |
| **Myopia** | ODIR5k | VGG-19 | 98.13% | ODIR5K | VGG19 | 96.5% |

Table 2

**Note**: in the AMD disease, the data used in the **related work** were acquired from the Ophthalmology Department of Kasturba Medical College (KMC), Manipal, India. And there are ethics approvals to collect the fundus images from Kasturba Medical Hospital, Manipal to conduct this study. It is evaluated 402 eyes with normal fundus, 583 retinal images with early, intermediate AMD, or GA and 125 retinal images with evidence of wet AMD.The images collected were acquired using Zeiss FF450 plus mydriatic fundus camera .

And this project depends on fundus images to classify all the 6 diseases, the previous one requires retinal images which is not optimal.

- **Utilization of a Single Image for Diagnosing Multiple Diseases**

Another significant distinction of this project is its utilization of a single image for diagnosing multiple eye diseases. In contrast to the other projects that focused on diagnosing specific diseases individually, our solution employs advanced image analysis techniques to assess various eye conditions simultaneously. By analyzing a single image of the user's eye, our system can provide insights and predictions for cataract, glaucoma, diabetic retinopathy, AMD, and myopia, all at once. This approach not only saves time and effort for the user but also reduces the need for multiple tests or examinations, making the diagnosis process more efficient and convenient.

# Chapter 3: System Analysis

## 3.1 Project specification

### 3.1.1 Functional requirement

- **User Account Creation**

  Users should be able to create a new account by providing personal information such as name, email address, and password. The account creation process should ensure data validation and secure storage of user credentials.

- **Login and Logout**

  Users should be able to log in to their account using their registered email address and password. A secure authentication mechanism should be implemented to protect user accounts from unauthorized access. Users should also have the ability to log out of their account when they are finished using the system.

- **Disease Selection**

  The system should allow users to select the type of eye disease they want to detect from a list of available options. This feature enables users to target specific diseases for diagnosis based on their concerns or symptoms.

- **Image Upload**

  Users should be able to upload images of their eyes for analysis. The system should support various image formats and provide a user-friendly interface for selecting and uploading images. Robust validation should be implemented to ensure the uploaded images are of appropriate quality and format.

- **Image Pre-processing**

  The system should perform pre-processing on the uploaded images to enhance their quality for accurate analysis. This may involve techniques such as noise reduction, image normalization, and image resizing. Pre-processing aims to optimize the input data for the subsequent analysis steps.

- **Image Segmentation**

  The system should accurately segment different parts of the eye from the uploaded images. This step is essential for isolating the relevant regions for disease detection. Techniques such as image thresholding, edge detection, or deep learning-based segmentation algorithms can be employed for accurate segmentation.

- **Feature Extraction**

  The system should extract relevant features from the segmented eye images. These features serve as input for the machine learning model. Feature extraction techniques can include extracting statistical, structural, or texture-based features that capture important characteristics of the eye and its diseases.

- **Machine Learning Model**

  The system should utilize a trained machine learning model, such as a Convolutional Neural Network (CNN), to classify different eye diseases. The model should be capable of handling multiple classes and trained on a diverse dataset of eye images representing various diseases.

- **Disease Prediction**

  Based on the input image and the extracted features, the system should predict the most likely eye disease. The machine learning model should provide accurate and reliable predictions, leveraging the trained knowledge from the dataset.

- **Result Display**

  The system should present the results of the disease prediction to the user in a clear and concise manner. This can include displaying the predicted disease, associated confidence scores, and any additional information or recommendations for further action.

- **Try without Authentication**

  The system should allow users to access the application and perform basic functionalities, such as uploading and analyzing images, without the need for authentication. This provides a convenient and seamless user experience for those who wish to quickly use the system without creating an account.

- **User Management**

  The system should provide administrative functionality to manage user accounts. This includes adding new users, editing user profiles, and removing user accounts if necessary. User management ensures proper access control and system administration.

- **Data Management**

  The system should effectively manage the data used by the machine learning model. This includes storing and organizing the uploaded images, extracted features, and disease predictions. Robust data storage and retrieval mechanisms should be implemented to ensure data integrity and availability.

- **User Feedback**

  The system should incorporate a feedback mechanism to gather user feedback regarding the accuracy and usability of the disease diagnosis. This feedback can be used to continuously improve the system's performance, address user concerns, and enhance user satisfaction.

- **Confidence Score**

  The system should calculate a confidence score for each disease detection, indicating the level of certainty in the classification. The confidence score provides users with additional information about the reliability of the predicted disease and helps them make informed decisions.

- **Chat Functionality**

  The system should provide a chat functionality that allows users to communicate with an ophthalmologist or healthcare professional in real-time. This feature enables users to discuss their results, seek further guidance, and receive personalized advice regarding their eye health.

- **Search and Filter Functionality**

  The system should offer search and filter capabilities to allow users to easily find and view their saved results. Users should be able to search for specific diagnoses, filter results based on disease type or date, and access previous reports for reference or follow-up purposes.

### 3.1.2 Non-functional requirement

- **Scalability**
  The application is designed to handle a large volume of users and data without experiencing performance degradation. It have the capability to scale up its resources, such as servers and databases, to accommodate increasing demands.

- **Reliability**:
  The application exhibit a high level of reliability by minimizing the occurrence of failures or crashes. It was built with fault-tolerant mechanisms and redundancy to ensure continuous availability and uninterrupted service.

- **Security**
  The applications prioritize the security of user data. It was implement robust security measures, including encryption, authentication, and authorization protocols, to protect against unauthorized access, data breaches, and other security threats.

- **Privacy**
  The application adheres to privacy laws and regulations to safeguard user privacy. It should employ measures to ensure the proper handling, storage, and processing of sensitive user data, and obtain user consent for data collection and usage.

- **Usability**
  The application should provide a user-friendly and intuitive interface, allowing users to easily navigate and perform tasks without confusion. It should have a clear layout, intuitive controls, and provide informative feedback to users.

- **Accessibility**
  The application should be accessible to users with disabilities, complying with accessibility standards and guidelines. It should provide features such as screen reader compatibility, keyboard navigation options, and adjustable font sizes to cater to users with diverse needs.

- **Performance**
  The application should deliver optimal performance, with fast response times and minimal latency. It should efficiently process user requests and handle concurrent operations, ensuring a smooth and responsive user experience.

- **Compatibility**
  The application should be compatible with various devices, operating systems, and web browsers. It should be responsive and adaptable to different screen sizes, resolutions, and input methods, providing a consistent experience across platforms.

- **Maintainability**
  The application's codebase should be well-structured, modular, and well-documented to facilitate ease of maintenance. It should follow coding best practices and standards, allowing developers to understand, modify, and enhance the application efficiently.

- **Portability**
  The application should be designed to be portable, enabling it to be deployed on different platforms and environments with minimal modifications or dependencies. It should be adaptable to various deployment configurations, such as on-premises or cloud-based setups.

- **Support**
  The application should have comprehensive documentation, including user guides and technical manuals, to assist users in utilizing its features effectively. It should also provide a reliable support system, with a dedicated support team available to address user inquiries, troubleshoot issues, and provide timely assistance.

- **Availability**
  The application should strive for high availability, aiming for minimal downtime and ensuring that users can access the application at all times. It should implement fault-tolerant mechanisms, such as load balancing and redundancy, to mitigate single points of failure and maximize uptime.

## 3.2 Use case diagram


Figure 2

The use case diagram represents the interactions between three main actors in the eye disease classification system: Patient, Doctor, and Admin. Each actor has specific roles and responsibilities within the system.

The Patient actor represents individuals using the mobile application to capture and upload fundus images for disease classification. The Patient can perform actions such as registering an account, logging in, choosing a service, uploading images, and viewing the classification results and these results can be viewed again in the results history, provide feedback, and communicate with doctors.

The Doctor Actor represents ophthalmologists who use the ophthalmologist interface can help patients by reviewing classification results and providing additional diagnoses or recommendations to patients. The Doctor can perform actions such as logging in, viewing patient data, reviewing classification results, and adding notes or comments.

The admin actor represents system administrators who have privileged access to manage system settings and user accounts. The admin can perform actions such as logging in, managing user accounts, adjusting system configurations, and monitoring system performance.

# Chapter 4: System Design

## 4.1 System component diagram

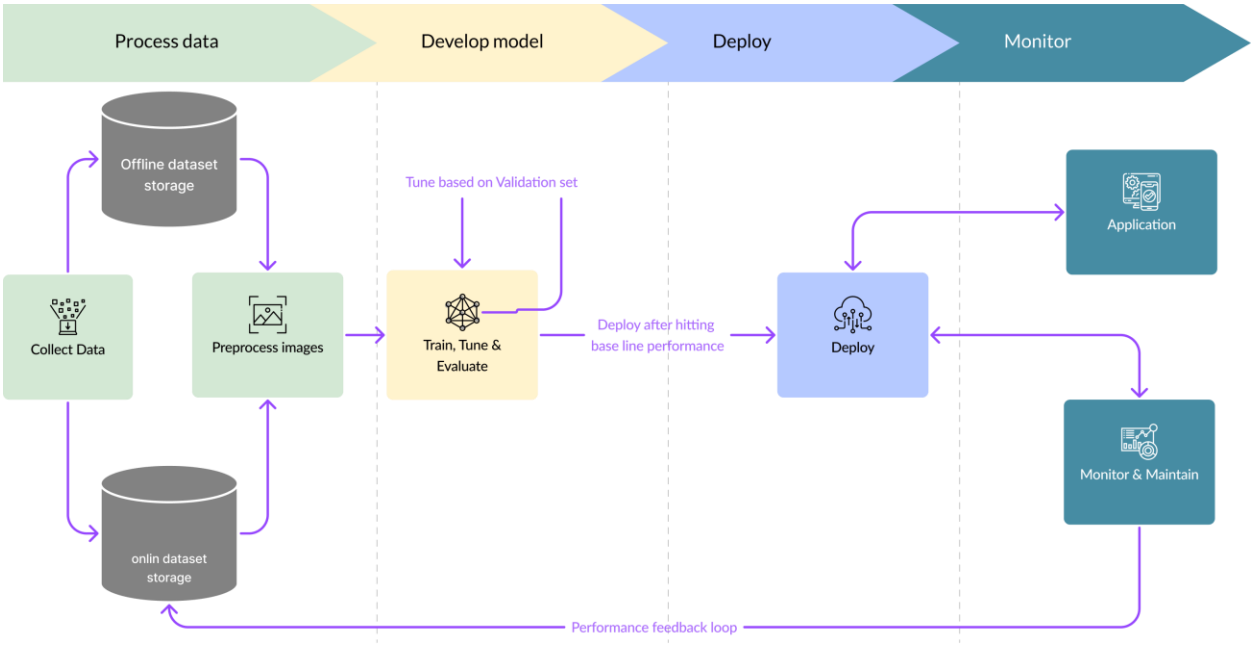



Figure 3

The component diagram states the steps of how the website and mobile app interact with the backend classification models, it firstly enters the processing data level where the data is either offline datasets from clinics or hospitals or online datasets from Kaggle or the images that the user used which is stored at firebase, the data is collected and processed to get ready for the next level which is the develop model level, in this level the pre-processed image enters the validation set to tune the model based on the dataset we have, after that the result and interactions occurs at the monitor level where the user interface occurs, this happens after the API calling and model deployment in the deploy level, monitoring and maintenance occurs gradually to update and improve the power of the prediction accuracy of the system.

## 4.2 System class diagram

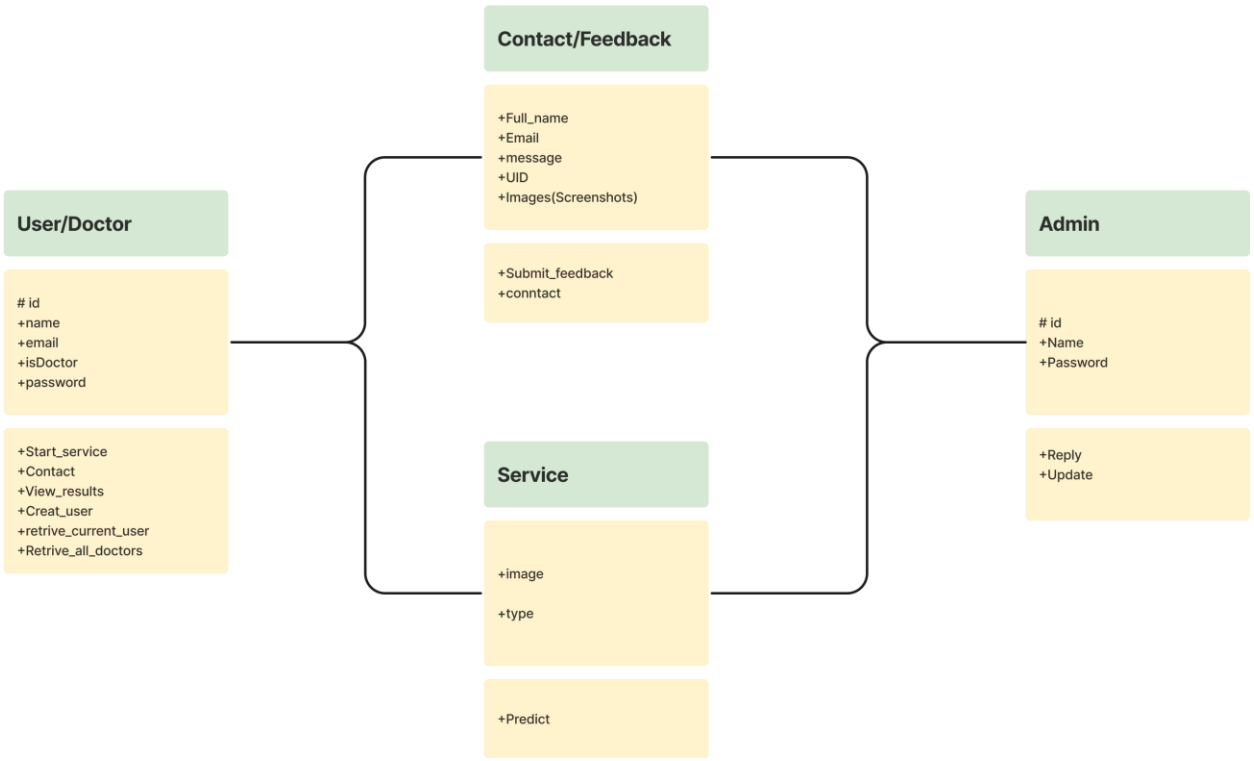



Figure 4

The class diagram depicts the architectural structure of the eye disease classification system, showcasing four main components: User/Doctor, Admin, Service, and Feedback/chat. The User/Doctor component represents the end-users, which include patients and ophthalmologists utilizing the mobile application and the ophthalmologist interface, respectively. The admin component represents the administrators responsible for managing system settings and user access, updating services and view feedbacks. The Service component serves as the central hub for data processing, including image analysis and disease classification using deep learning models and choosing the service (model) type. Lastly, the Feedback/chat component facilitates communication channels between users/doctors and the system administrators, allowing for feedback, inquiries, and support requests. This class diagram provides an overview of the system's key components and their interactions, highlighting the seamless flow of information and functionality among the User/Doctor, Admin, Service, and Feedback/chat components.

# 4.3 Sequence diagram

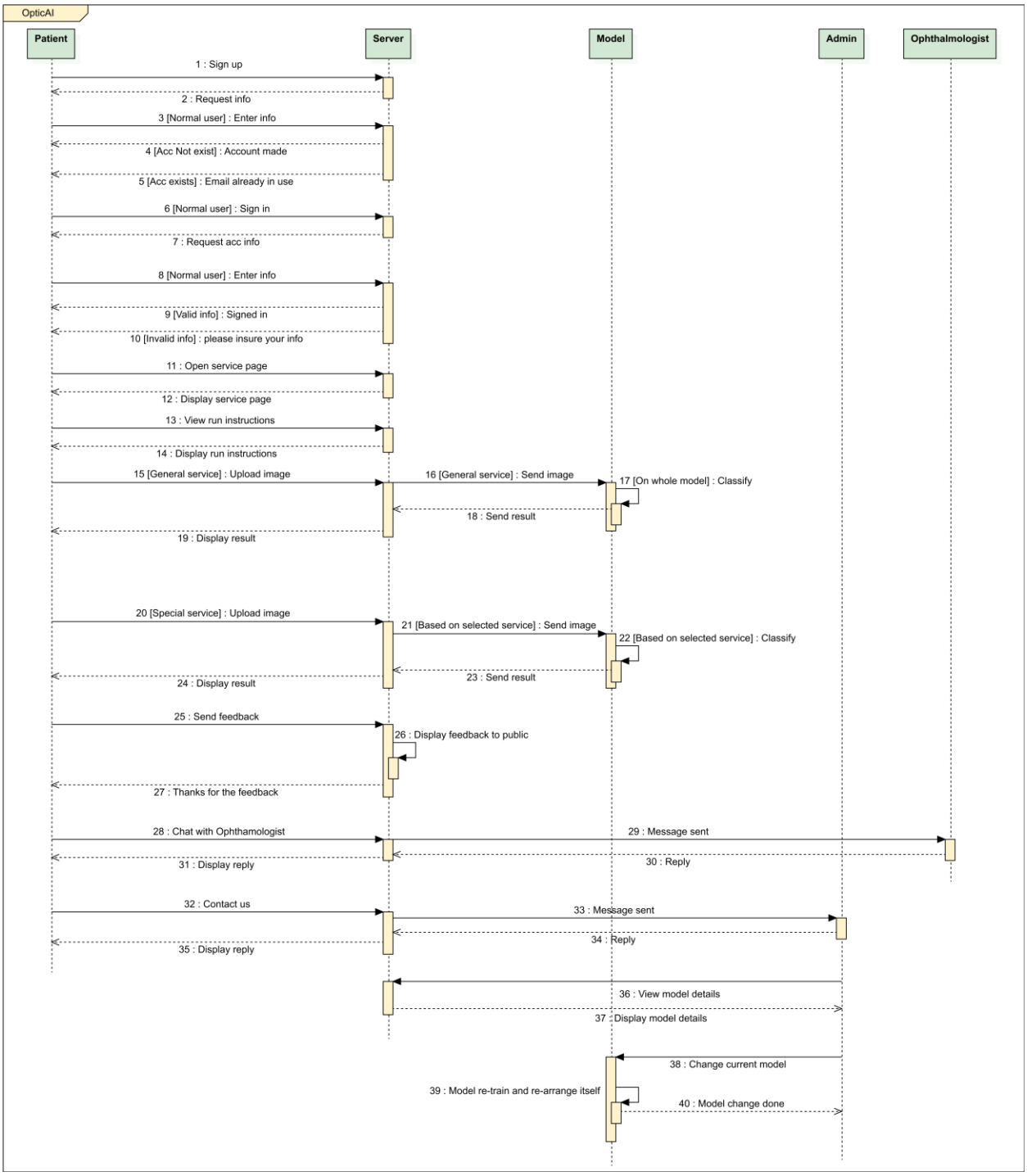



Figure 5

The sequence diagram illustrates the flow of interactions within the eye disease classification system, which comprises a mobile app for patients, ophthalmologists, and an administrative interface, all connected to a central server. The diagram showcases

the key interactions and messages exchanged between the various components. Initially, the patient captures a fundus image using the mobile app and submits it to the server for analysis. Upon receiving the image, the server invokes the deep learning model for disease classification. The classification result is then sent back to the app, where it is displayed to the patient. In parallel, the ophthalmologist can respond to provide diagnoses for patients asking for verification if the diagnosis of the app was right and communicate with patients. The ophthalmologist can also provide feedback or diagnosis through the app, the administrative interface is responsible for reading feedbacks and developing the models. This sequence diagram provides a comprehensive overview of the interactions and data flow within the eye disease classification system, highlighting the seamless communication between the mobile app, server, and stakeholders involved in the diagnosis process.

## 4.4 Project ERD

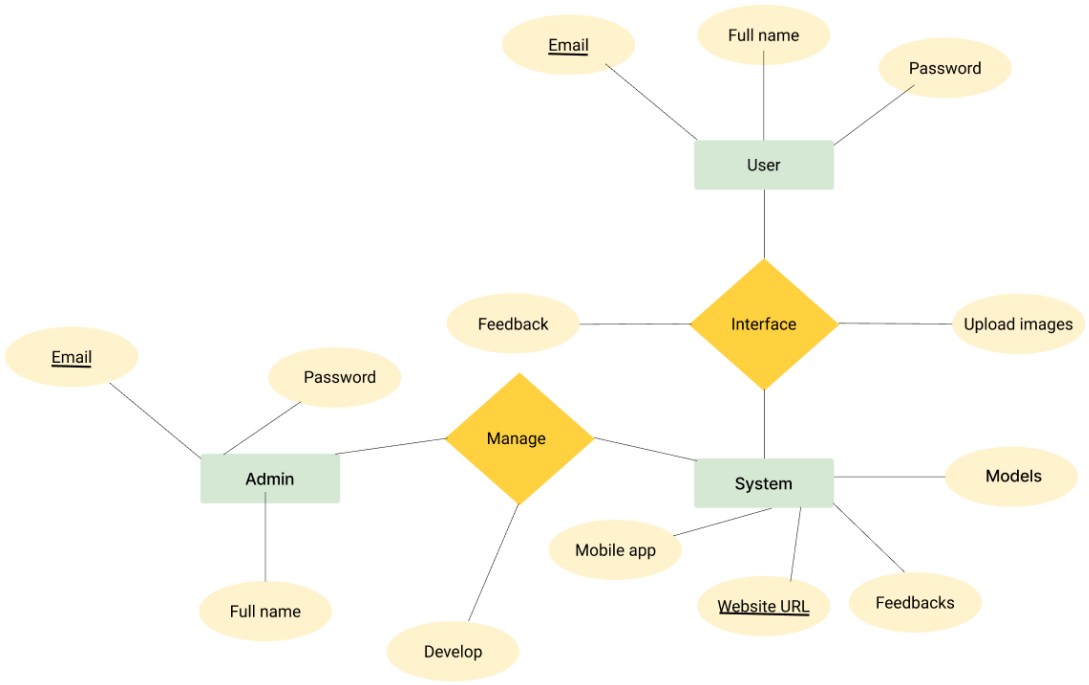



Figure 6

This entity relationship diagram (ERD) shows how the data moves and stored and its relationship to its entity, as for the user they carry a unique email address and full name and the password, same goes for the admin, the relationship between the user and the system is the interface by uploading image to run the classification model or providing a

feedback which is yet stored in the system entity, the system has a unique URL and has been developed in a mobile application, the relationship between admin and system is that they manage and develop the models or the website and its data.

## 4.5 System GUI design

### 4.5.1 Website

- **Website home page**

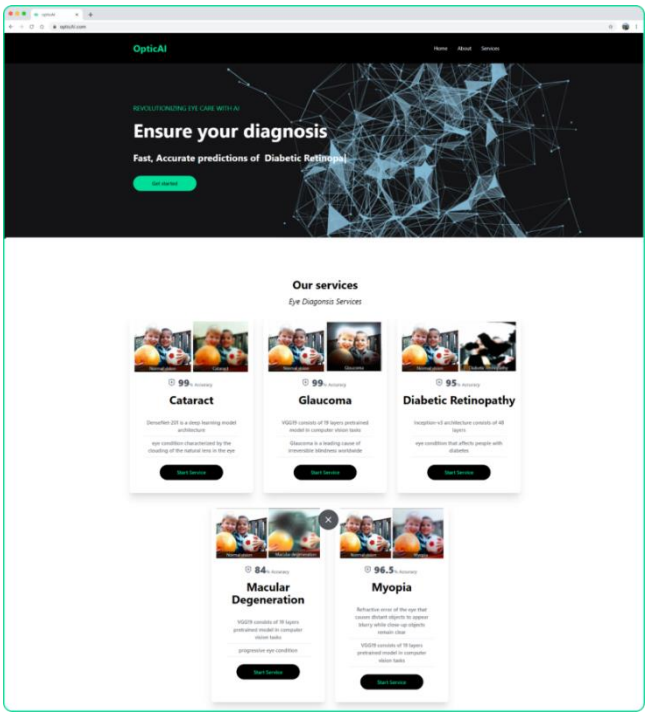



Figure 7

- **Service page**

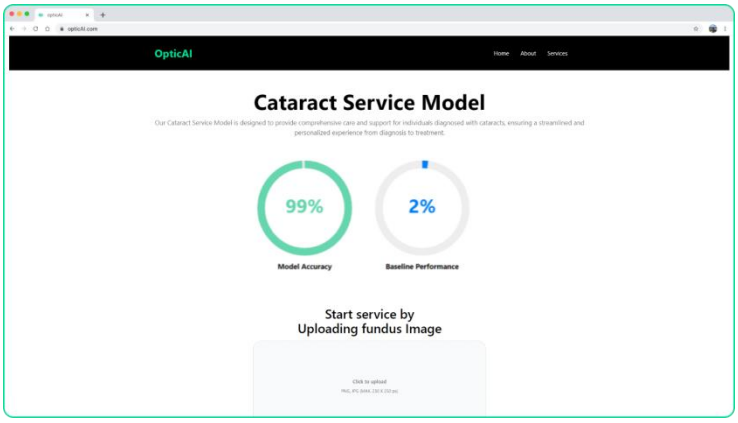



Figure 8

- **Try service**

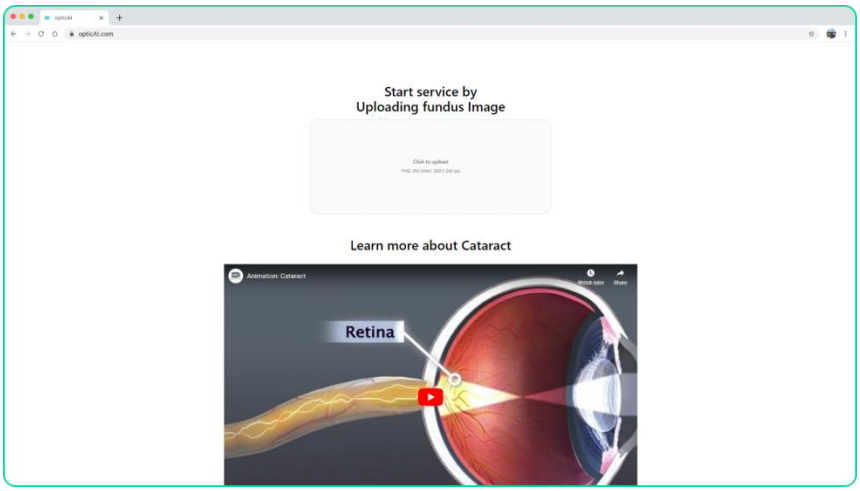



Figure 9

- **Prediction result**

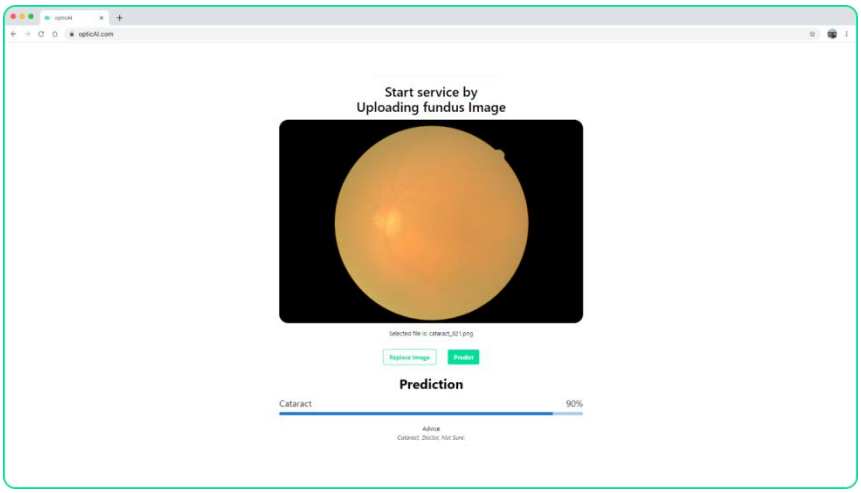



Figure 10

We provide the user with the ability to upload an image of their funds. The uploaded image is then sent as a request to a Flask API, which loads an AI model capable of accepting and pre-processing the funds image. The processed image is then added to the model for further analysis or processing. The result of the prediction is then sent back as a response from the API, providing the user with the outcome or information extracted from the funds image.

### 4.5.2 Mobile application

- **Registration and Login**

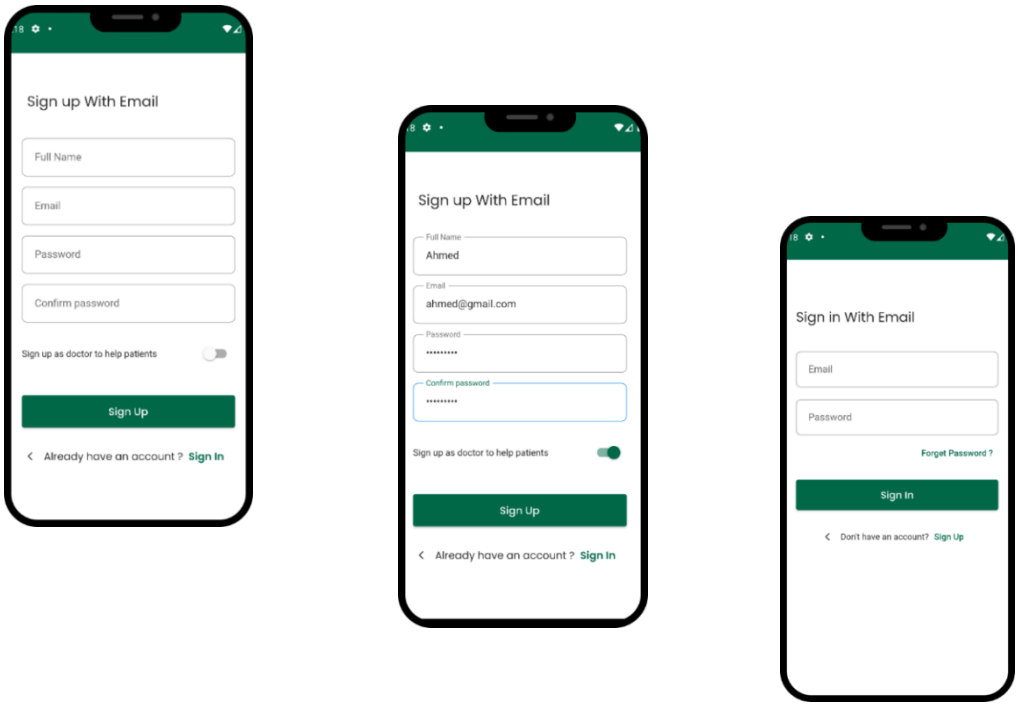



Figure 11

- **Try Services**

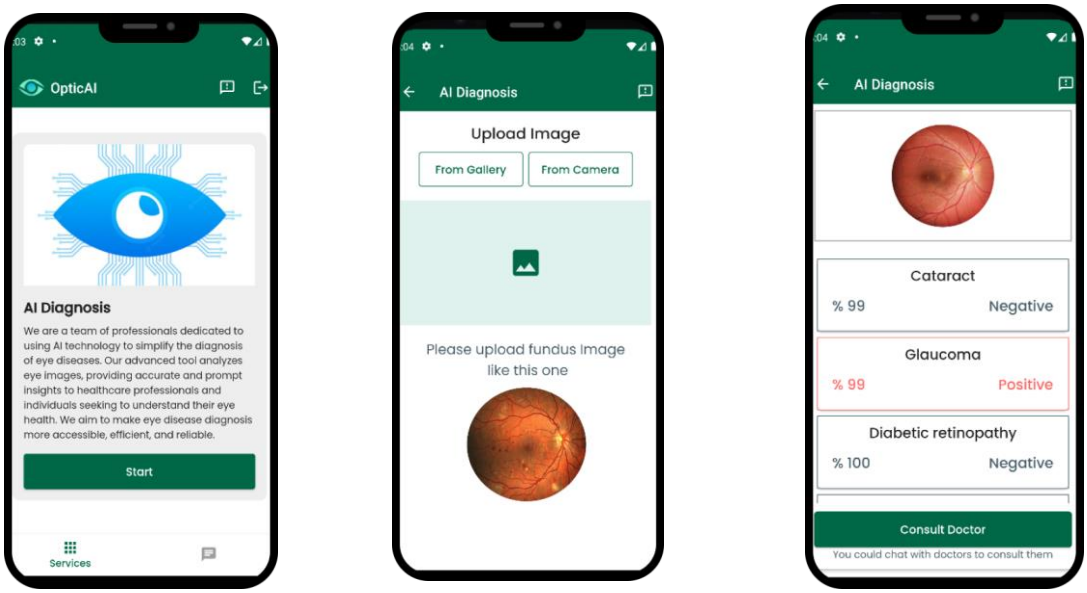



Figure 12

- **Chatting with doctors**

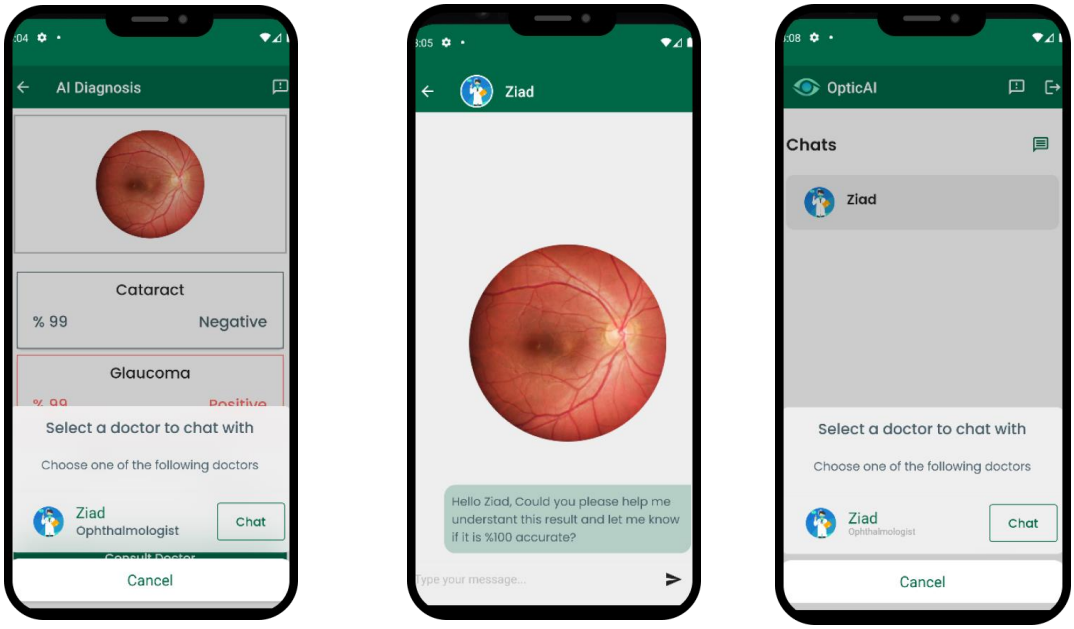



Figure 13

- **Feedback steps**

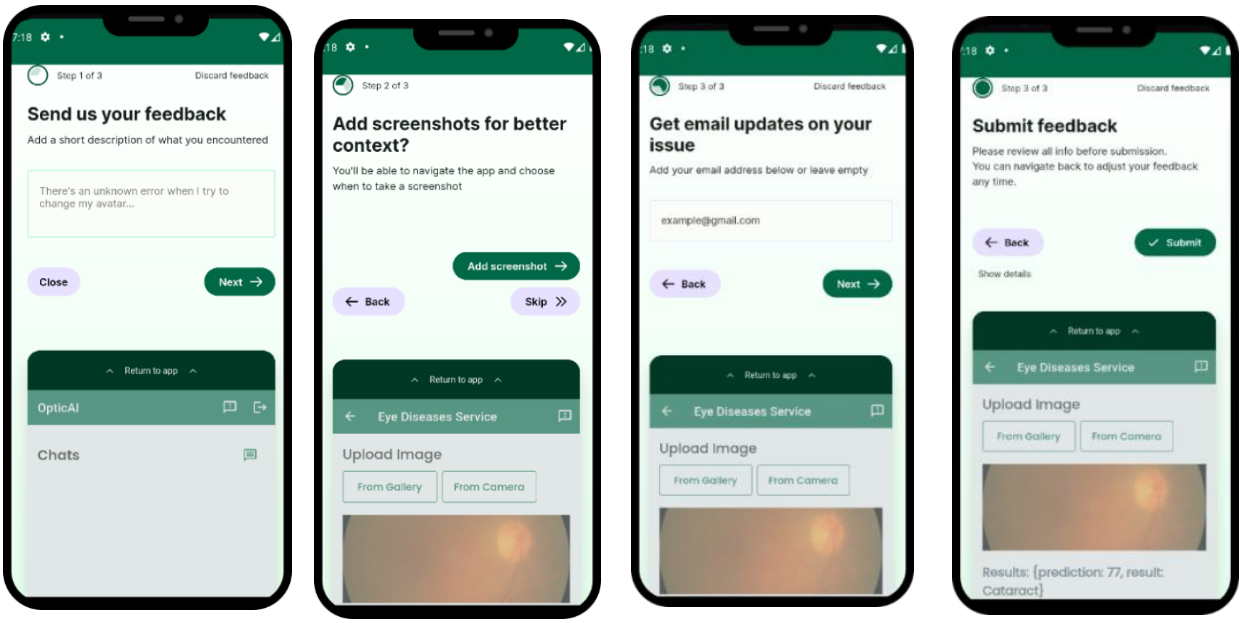



Figure 14

And here how we receive feedbacks from users:

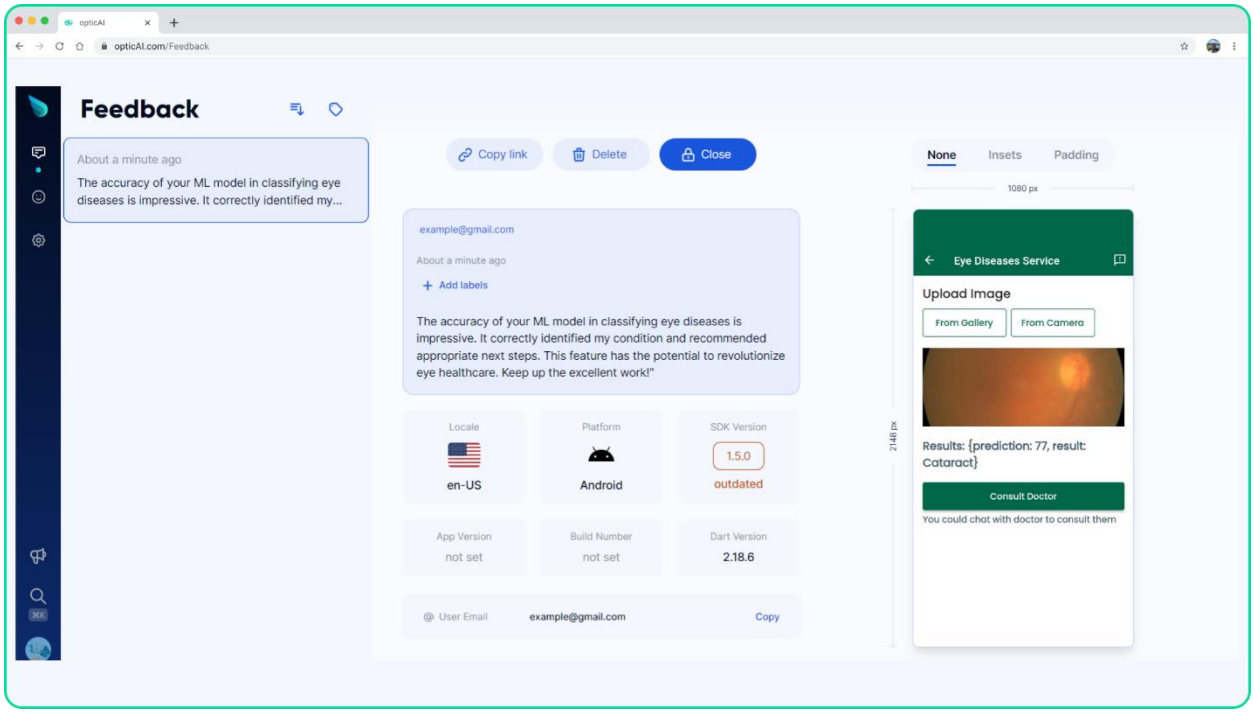



Figure 15

# Chapter 5: Implementation and Testing

## 5.1: Implementation

- **Implementation of cataract model.**

```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D, BatchNormalization
model = Sequential()

model.add(ResNet50(include_top=False, pooling='avg', weights="imagenet"))
model.add(Flatten())

model.add(BatchNormalization())

model.add(Dense(1, activation='sigmoid'))

model.layers[0].trainable = False
model.summary()
```

Figure 16

- **Implementation of Glaucoma, AMD and Myopia model.**

```
vgg = VGG19(weights="imagenet",include_top = False,input_shape=(224,224,3))
for layer in vgg.layers:
    layer.trainable = False
model = Sequential()
model.add(vgg)
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()
```

Figure 17

- **Implementation of diabetic retinopathy model.**

```python
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Sequential
from keras.layers import Conv2D, Dense, Dropout, BatchNormalization, Flatten, MaxPool2D,
 GlobalAveragePooling2D
from tensorflow.keras.regularizers import l2, l1
# Basic architecture model of InceptionV3
model_inception=InceptionV3(weights='imagenet',include_top=False, input_shape=(224, 224, 3))
x=model_inception.output
x= GlobalAveragePooling2D()(x)
x= BatchNormalization()(x)
x = Dense(256, activation='relu')(x)
x= Dropout(0.5)(x)
output=Dense(1,kernel_regularizer = tf.keras.regularizers.L2(0.01) ,activation='sigmoid')(x)
#FC-Layer
model_inception=tf.keras.Model(inputs=model_inception.input,outputs=output)
model_inception.summary()
# Freezing the base model
for layer in model_inception.layers[:-5]:
    layer.trainable=False
```

Figure 18

- **General Pre-processing (python-flask).**

```python
from flask import Flask, request, jsonify
from tensorflow import keras

from keras_preprocessing.image import load_img
from keras_preprocessing.image import img_to_array
import numpy as np
from PIL import Image
from flask_cors import CORS
import cv2
import numpy as np


app = Flask(__name__)
CORS(app)

cataract_classification_model = keras.models.load_model("
D:/Ocular/Catract/OcularCatract/my_model/ResNet50-Cataract_classification-98.98.h5")
glaucoma_classification_model = keras.models.load_model("
D:/Ocular/Catract/OcularCatract/my_model/VGG19-Glaucoma_classification-98.91.h5")
retinopathy_classification_model = keras.models.load_model("
D:/Ocular/Catract/OcularCatract/my_model/inception-DR_classification-95.00.h5")
amd_classification_model = keras.models.load_model("D:/Ocular/Catract/OcularCatract/my_model/VGG19-AMD_classification-84.21.h5")

def crop_image_from_gray(img,tol=7):
    if img.ndim ==2:
        mask = img>tol
        return img[np.ix_(mask.any(1),mask.any(0))]
    elif img.ndim==3:
        gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        mask = gray_img>tol

        check_shape = img[:,:,0][np.ix_(mask.any(1),mask.any(0))].shape[0]
        if (check_shape == 0):
            return img
        else:
            img1=img[:,:,0][np.ix_(mask.any(1),mask.any(0))]
            img2=img[:,:,1][np.ix_(mask.any(1),mask.any(0))]
            img3=img[:,:,2][np.ix_(mask.any(1),mask.any(0))]
            img = np.stack([img1,img2,img3],axis=-1)
        return img
```

Figure 19

- **Cataract API (python-flask).**

```python
@app.route('/cataract', methods=["POST"])
def Cataract():

    image_size=224

    def predict_class(image):


        # image = cv2.imread(image_path)
        image= crop_image_from_gray(image)
        image = cv2.resize(image,(image_size,image_size))

        image=np.array(image)
        image=image.reshape(-1,image_size,image_size,3)

        y = cataract_classification_model.predict([image])
        return y


    if 'imagefile' not in request.files:

        return "No image file found in the request."


    else:
        print ("Image file received and processed successfully.")

    imagefile = request.files['imagefile']


    image_bytes = imagefile.read()

    image_np = np.frombuffer(image_bytes, np.uint8)

    image = cv2.imdecode(image_np, cv2.IMREAD_COLOR)

    # Check if the image was loaded successfully
    if image is None:
        return 'Failed to load image', 400

    prediction = predict_class(image)


    print(prediction[0][0])
    result = ''
    if prediction[0][0] > 0.45:
        print("Cataract")
        result = 'Cataract'
    else:
        print("Normal")
        result = 'Normal'

    # to fix the prediction value when it is less than (Normal) ?
    if prediction[0][0] < 0.45:
        prediction[0][0] = 1 - prediction[0][0]

    prediction[0][0] = prediction[0][0] * 100
    prediction = prediction[0][0].tolist()  # Convert numpy array to a Python list
    prediction = round(prediction)
    # Clean up the image file if needed
    print(prediction)
    print("**********************************************************************")

    return jsonify({'prediction': prediction, 'result': result})
```

Figure 20

- **Flutter chasing.**

```dart
import 'package:camera/camera.dart';
import 'package:eyeclassify/auth/sign_in.dart';
import 'package:eyeclassify/auth/sign_up.dart';
import 'package:eyeclassify/bloc/bloc.dart';
import 'package:eyeclassify/components/color_style.dart';
import 'package:eyeclassify/components/components.dart';
import 'package:eyeclassify/components/text_style.dart';
import 'package:eyeclassify/helper/cache_helper.dart';
import 'package:eyeclassify/screens/Home_Layout.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:wiredash/wiredash.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();


  await CacheHelper.init();

  Widget? widget;

  if (token != null) {
    widget = Home_Layout();
  } else {
    widget = SignUpScreen();
  }

  runApp(MyApp(widget: widget));
}

class MyApp extends StatelessWidget {
  Widget? widget;

  MyApp({
    required this.widget,
  });
```

Figure 21

- **Calling cataract API (flutter).**

```dart
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:dio/dio.dart';
import 'package:image_picker/image_picker.dart';

class ApiHelper {
  static Dio? _dio;

  static Dio getDioInstance() {
    if (_dio == null) {
      BaseOptions options = BaseOptions(
        baseUrl: 'http://10.0.2.2:4000', // Replace with your Flask API URL
        connectTimeout: const Duration(milliseconds: 5000),
        receiveTimeout: const Duration(milliseconds: 3000),
      );
      _dio = Dio(options);
    }
    return _dio!;
  }

  static Future<String> uploadImage(File imageFile) async {
    try {
      Dio dio = getDioInstance();

      FormData formData = FormData.fromMap({
        'imagefile': await MultipartFile.fromFile(
          imageFile.path,
          filename: imageFile.path.split('/').last,
        ),
      });

      final response = await dio.post('/cataract', data: formData);

      final predictionResult = response.data.toString();

      return predictionResult;

    } catch (e) {
      throw Exception(e.toString());
    }
  }
}
```

Figure 22

- **Calling API and building the UI (React).**

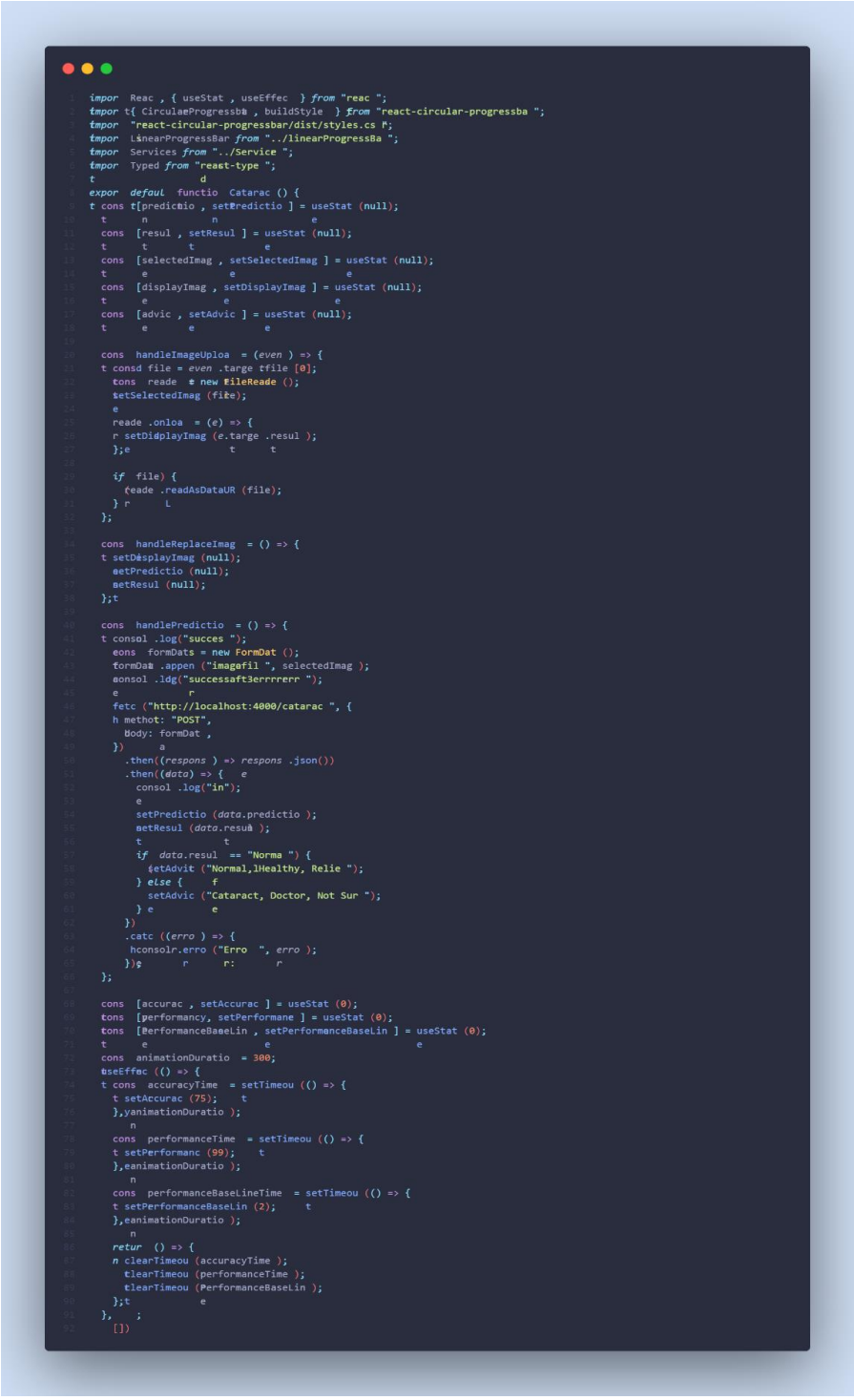


Figure 23

- **Calling API and handling prediction (React).**

```
cons  handlePredictio  = () => {
t consol .log("succes ");
  eons  formDats = new FormDat ();
  formDat .appen ("imagefil ", selectedImag );
  a        d       e          e
  fetc ("http://localhost:4000/catarac ", {
  h methot: "POST",
    body: formDat ,
  })      a
    .then((respons ) => respons .json())
    .then((data) => {   e
      consol .log("in");
      e
      setPredictio (data.predictio );
      setResul (data.resul );
      t              t
      if  data.resul  == "Norma ") {
        setAdvit ("Normal,lHealthy, Relie ");
      } else {     f
        setAdvic ("Cataract, Doctor, Not Sur ");
      } e          e
    })
    .catc ((erro ) => {
     hconsolr.erro ("Erro  ", erro );
    })$       r      r:       r
};
```

Figure 24

- **Run Flask API (Python-Flask).**

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=4000)
```

Figure 25

- **Testing API loaded model accuracies.**

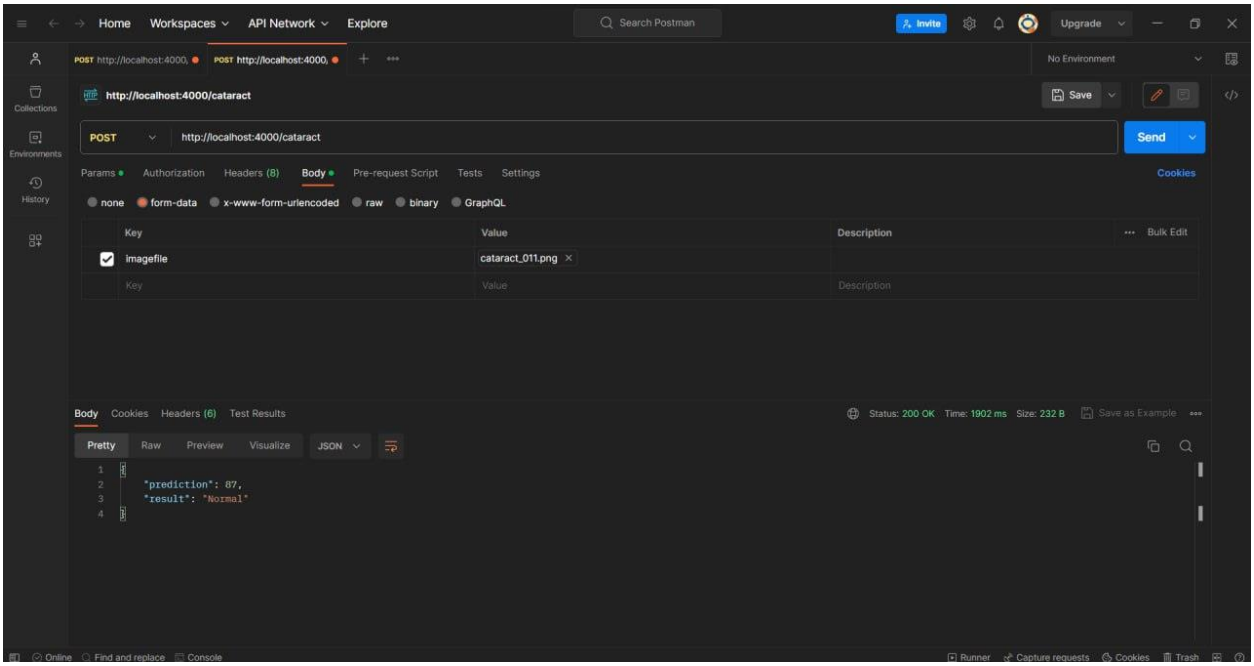
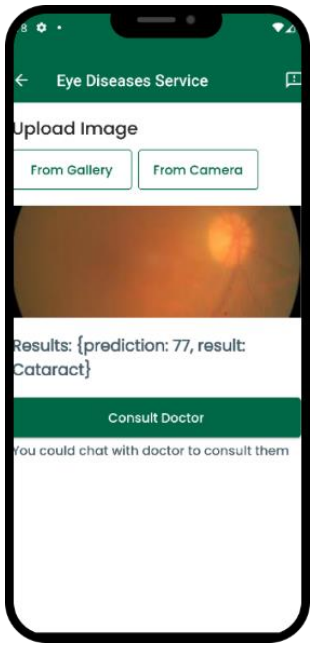



Figure 26

## 5.2: Testing

- **Showing Result.**

**Mobile Application:** **Website:**



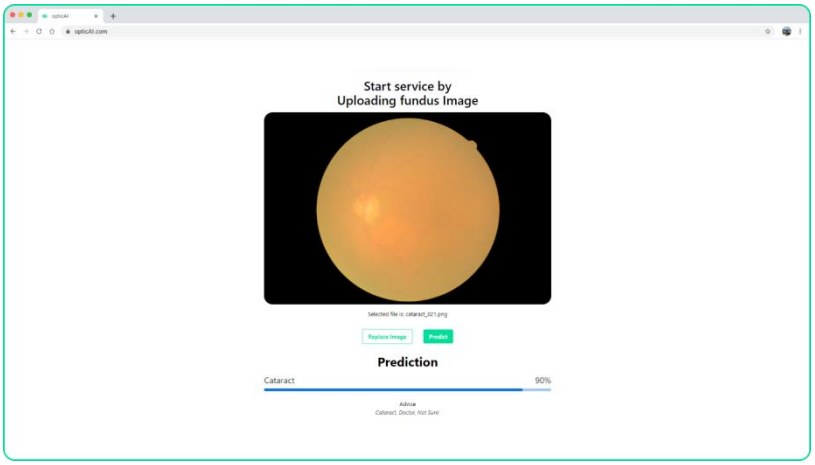




Figure 27

- **Not a strong password.**

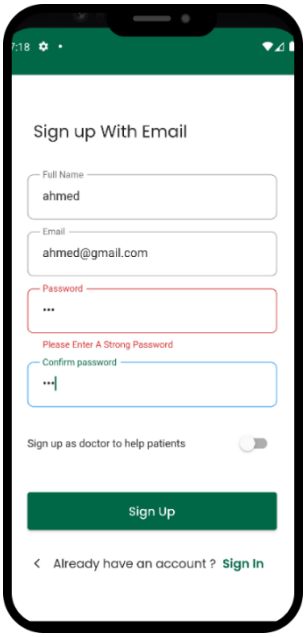



Figure 28

- **Password should be the same.**

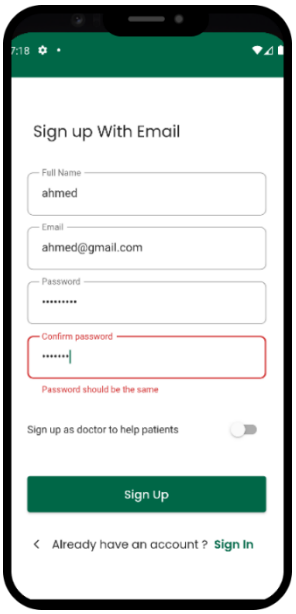



Figure 29

- **Email already exists.**

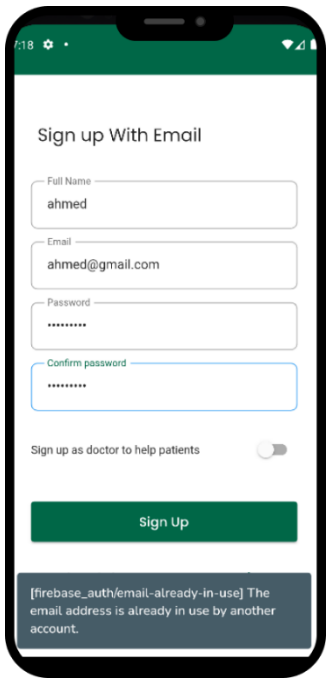



Figure 30

- **Email is invalid.**

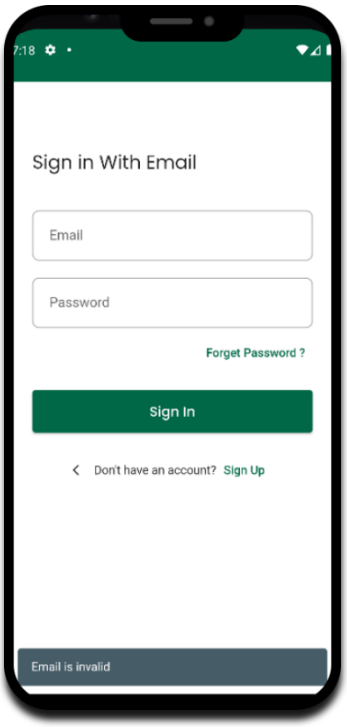



Figure 31

- **The required fields should be entered.**

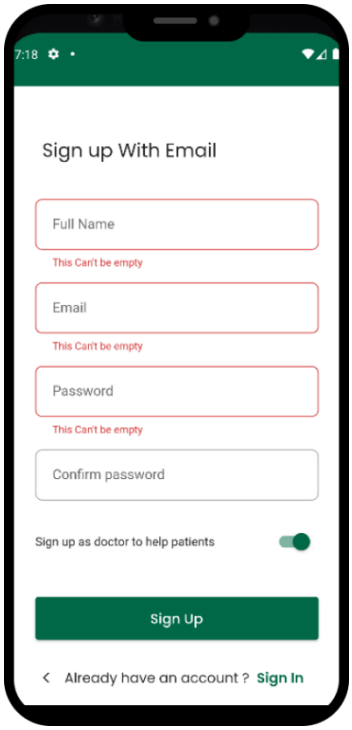



Figure 32

# References


1. World Health Organization. (2019). Blindness and Vision Impairment. Retrieved from https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

2. https://docs.google.com/forms/d/1QnN8H0s3C0dRLey9VfPemRVFUx-elg1lVWfx4VsiqLQ/edit?chromeless=1

3. ODIR-5K dataset: Larxel, "ODIR-5K: A Dataset for Classification of Retinal Diseases," arXiv:1902.04026 (2019). Dataset Link: https://www.kaggle.com/datasets/andrewmvd/ocular-disease-recognition-odir5k

4. APTOS 2019 Blindness Detection dataset from Kaggle. Reference: Brown JM, et al. "The APTOS 2019 Blindness Detection Challenge." arXiv preprint arXiv:1908.08413 (2019).Link: https://www.kaggle.com/competitions/aptos2019-blindness-detection/data

5. Glaucoma Detection dataset. Link: https://www.kaggle.com/datasets/sshikamaru/glaucoma-detection

6. Saxena, Gaurav & Rawat, Anil. (2020). Improved and robust Deep Learning agent for preliminary detection of Diabetic Retinopathy using public datasets. Intelligence-Based Medicine. 3-4. 10.1016/j.ibmed.2020.100022. https://www.researchgate.net/publication/346305946_Improved_and_robust_Deep_Learning_agent_for_preliminary_detection_of_Diabetic_Retinopathy_using_public_datasets

7. Das, Dolly, Saroj Kumar Biswas, and Sivaji Bandyopadhyay. "Detection of diabetic retinopathy using convolutional neural networks for feature extraction and classification (DRFEC)." Multimedia Tools and Applications (2022): 1-59. https://link.springer.com/article/10.1007/s11042-022-14165-4#Abs1

8. Khan, Md Shakib, et al. "Deep learning for ocular disease recognition: an inner-class balance." Computational Intelligence and Neuroscience 2022 (2022). https://www.hindawi.com/journals/cin/2022/5007111/#abstract

9. Li, Liu, et al. "Attention based glaucoma detection: A large-scale database and CNN model." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. https://openaccess.thecvf.com/content_CVPR_2019/html/Li_Attention_Based_Glaucoma_Detection_A_Large-Scale_Database_and_CNN_Model_CVPR_2019_paper.html

10. Smaida, Mahmoud and Serhii Yaroshchak. “Bagging of Convolutional Neural Networks for Diagnostic of Eye Diseases.” International Conference on Computational Linguistics and Intelligent Systems (2020). https://ceur-ws.org/Vol-2604/paper49.pdf

11. Tan, Jen Hong, et al. "Age-related macular degeneration detection using deep convolutional neural network." Future Generation Computer Systems 87 (2018): 127-135. https://www.sciencedirect.com/science/article/abs/pii/S0167739X17319167