RETIINOTRACK

AUTOMATED DETECTION AND CLASSIFICATION OF DIABETIC RETINOPATHY USING DEEP LEARNING

A MAJOR PROJECT REPORT

Submitted by

ABDUL SAMAD **Register No: TLY21IT002**
PRAJUL PRAKASH **Register No: TLY21IT049**
RANSHI FARHAN USMAN A K **Register No: TLY21IT051**

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Information Technology

Department of Information Technology

College of Engineering Thalassery

(Under Co-operative Academy of Professional Education)
Established by the Government of Kerala
Kannur Dt. - 670107

March 2025

DECLARATION

We undersigned hereby declare that the major project report "**RetinoTrack- Automated Detection and Classification of Diabetic Retinopathy Using Deep Learning**", submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Prof. **Nithya G P**. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Abdul Samad

Prajul Prakash

Ranshi Farhan Usman A K

Thalassery

28-03-2025

Department of Information Technology
College of Engineering Thalassery

(Under Co-operative Academy of Professional Education)
Established by the Government of Kerala
Kannur Dt. – 670107

CERTIFICATE

This is to certify that the major project report entitled

AUTOMATED DETECTION AND CLASSIFICATION OF DIABETIC RETINOPATHY USING DEEP LEARNING

Submitted by

ABDUL SAMAD	**Register No: TLY21IT002**
PRAJUL PRAKASH	**Register No: TLY21IT049**
RANSHI FARHAN USMAN A K	**Register No: TLY21IT051**

to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology is a bonafide record of the major project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Coordinator
Akhil Chandran Miniyadan
Assistant Professor, Dept. of IT

Project Guide
Nithya G P
Assistant Professor, Dept. of IT

Head of the Department
Shamal P K
Assistant Professor, Dept. of IT

ACKNOWLEDGEMENT

We are greatly indebted to all those who have rendered help for successful completion of this project work.

First, we wish to extend our sincere gratitude to our Principal, **Prof. (Dr.) Rajeev P**, for the help extended by him in providing all the college facilities for our project work.

We extend our genuine and heartfelt thanks to **Prof. Shamal P K**, Assistant Professor and Head, Department of Information Technology for having permitted us to undergo this project and for his valuable suggestions, help and support during the study, analysis design and implementation of this project.

We are thankful and profoundly indebted to our Project Coordinator **Prof. Akhil Chandran Miniyadan** and Project Guide **Prof. Nithya G P**, Assistant Professors, Department of Information Technology, for their exemplary guidance, criticism, monitoring and constant encouragement throughout the course and for showing us the right way.

Also, we would like to express our sincere voice of gratitude to all the faculty members of IT department who gave the support throughout the project.

We also express our deepest thanks to our family members and friends for their moral support rendered to us to make this project a success.

ABDUL SAMAD
PRAJUL PRAKASH
RANSHI FARHAN USMAN A K

ABSTRACT

Diabetic retinopathy (DR) is a severe complication of diabetes that affects the retina due to prolonged high blood sugar levels. Studies indicate that nearly 80% of individuals who have had diabetes for ten years or more develop some degree of DR, which, if left untreated, can lead to blindness. However, research suggests that at least 90% of new cases could be prevented with timely and effective monitoring. Traditional methods of DR diagnosis rely on manual screening by ophthalmologists, which can be time-consuming and resource-intensive. Given the increasing number of diabetes patients worldwide, there is an urgent need for an automated, accurate, and efficient method to detect and classify DR at different stages.

One of the major challenges in DR detection is the reliance on human specialists to analyze high-resolution retinal images. This manual process often results in delays in diagnosis, miscommunication, and loss of follow-up, which can lead to the progression of the disease. Clinicians typically identify DR by detecting lesions and vascular abnormalities in the retina, but this method requires extensive expertise and resources. To overcome these limitations, deep learning techniques, specifically convolutional neural networks (CNNs), have gained prominence in medical image analysis. These models can automatically learn patterns from vast amounts of data, making them well-suited for detecting DR in retinal images with high precision.

A popular deep learning architecture used for DR classification is VGG16, a deep CNN model known for its ability to extract complex features from images. VGG16 is particularly effective in analyzing high-resolution retinal images, as it consists of multiple convolutional layers that can identify key patterns associated with DR. By leveraging GPU acceleration, VGG16 can process large datasets efficiently, enabling quick and accurate classification of DR into different severity levels. This approach not only reduces the burden on healthcare professionals but also ensures faster diagnosis, leading to early intervention and better patient outcomes. Integrating deep learning models like VGG16 into DR screening can revolutionize ophthalmology by making detection more accessible and reducing the risk of blindness in diabetic patients.

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolutional Neural Network
AI	Artificial Intelligence
ML	Machine Learning
DFD	Data Flow Diagram
UML	Unified Modelling Language
ER	Entity Relationship

CONTENTS

CHAPTER 1

INTRODUCTION

Diabetic retinopathy (DR) is one of the primary causes of vision impairment and blindness among people with diabetes. As the prevalence of diabetes continues to rise worldwide, the demand for accessible and efficient DR screening has become increasingly urgent. Traditional diagnostic approaches, though effective, often require specialized training and can be time-consuming, leading to potential delays in diagnosis and treatment, especially in resource-limited areas. The need for scalable and automated screening solutions is critical to meeting the growing demand for early DR detection and intervention.

Recent advances in deep learning, particularly through convolutional neural networks (CNNs), have revolutionized medical imaging and disease diagnosis, providing powerful tools for automated DR detection. By training on extensive datasets of retinal images labeled with DR stages, CNNs can accurately detect and classify stages of DR by recognizing complex, subtle patterns in retinal images, achieving accuracy levels comparable to those of human specialists. These automated systems can rapidly process images, delivering consistent and immediate results that are invaluable in settings with limited access to specialized care.

This project explores the methodologies behind automated DR detection, including the datasets, model architecture, and evaluation techniques, and examines the potential impact on clinical practice. By enhancing the speed and accuracy of DR diagnosis, these deep learning systems provide significant benefits: they facilitate timely referrals, improve patient outcomes, and expand access to essential diagnostic services. Automated DR screening marks a significant advancement in ophthalmology, offering the potential to improve long-term outcomes for diabetic patients globally while addressing a critical need in diabetic eye care.

1.1 PROBLEM STATEMENT

- Develop a CNN model to automatically detect diabetic retinopathy (DR) from retinal images, helping to distinguish between healthy eyes and those affected by DR.
- Create a system that classifies the severity of DR into stages, allowing for a clearer understanding of disease progression
- Test the model's accuracy, precision, recall, and F1-score to ensure it performs reliably and effectively.
- Explore how this tool could support ophthalmologists in diagnosing DR faster and more affordably, making early detection more accessible and improving patient care outcomes.

1.2 MOTIVATION

- Prevent Vision Loss: Early detection can help prevent severe complications and blindness caused by diabetic retinopathy.
- Easy Access to Screening: A detection system can make eye exams more accessible and convenient for patients, especially those with limited healthcare access.
- Improved Health Management: Timely identification of the disease enables better management of diabetes and overall eye health, enhancing the quality of life for patients

1.3 PURPOSE AND NEED OF THE PROJECT

- Develop a CNN-based system for detecting and classifying diabetic retinopathy (DR).
- Improve efficiency and accuracy of DR detection.
- Reduce the burden on healthcare professionals.
- Enhance patient outcomes through early and accurate detection.

1.4 PROJECT GOALS

- Achieve high classification accuracy using deep learning models.
- Validate the model with existing datasets.
- Deploy the model for real-time diagnosis.
- Expand the model to detect other retinal diseases.
- Continuously improve model performance with new data and techniques.

1.5 PROJECT OBJECTIVES

- Develop a CNN-based deep learning model to detect and classify DR stages.
- Enhance diagnostic accuracy using deep learning.
- Minimize manual intervention in the diagnostic process.
- Ensure scalability and applicability to large datasets.
- Integrate the model with existing medical systems.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

Machine learning has brought significant advancements in detecting diabetic retinopathy (DR), fundamentally transforming medical image analysis. By automating the identification and classification of DR stages from retinal images, these systems assist healthcare providers in diagnosing and managing the disease more effectively, potentially preventing or delaying vision loss in diabetic patients. The technology is designed to identify even subtle changes in retinal images that indicate DR's progression, enabling timely and precise intervention. However, challenges remain in handling the natural variations across retinal images and ensuring consistent accuracy in diverse clinical environments, highlighting the need for further refinement.

Recent research has focused on improving detection accuracy and adapting systems to manage the diversity found in retinal images from different patients. By refining algorithms that recognize complex DR-related patterns, these models are making them more applicable to real-world healthcare settings. This progress brings the potential for reliable access to DR screening, offering timely support to clinicians, particularly in settings with limited specialized resources. As machine learning continues to evolve in this area, it holds the promise of enhancing early intervention, ultimately contributing to better long-term outcomes for patients with diabetic retinopathy.

The covered literature goes from August 2017 to May 2023. The goal is to give refining detection algorithms and addressing variability to improve practical implementation in medical settings. The goal is to make automated systems more accurate and adaptable across diverse healthcare environments, ensuring reliable DR detection.

2.2 AUTOMATED DETECTION OF DIABETIC RETINOPATHY USING SUPPORT VECTOR MACHINE

Paper [1] examines a common eye disease among diabetic patients, diabetic retinopathy, which is a leading cause of blindness worldwide. Diabetic retinopathy develops progressively, beginning with a non-proliferative stage, which can escalate to a proliferative stage if left undetected or untreated. The disease is marked by damage to the blood vessels in the retina, which can lead to severe vision loss as it advances. Early diagnosis and intervention are crucial, as they can prevent or delay further deterioration and allow for effective treatment strategies that protect against irreversible blindness. Detecting early symptoms, such as microaneurysms, hard exudates, and hemorrhages, is essential to the timely management of diabetic retinopathy.

The paper suggests Support Vector Machine (SVM) is used for identifying early signs of diabetic retinopathy through digital processing of retinal images. This system is designed to automatically classify the grade of non-proliferative diabetic retinopathy, using retinal image analysis as its primary diagnostic tool. By enabling the earlier identification and grading of the disease, this SVM could offer significant advantages in the clinical management of diabetic retinopathy, particularly in settings with limited access to specialist eye care.

In conclusion, the SVM proposed in paper [1] offers a promising approach to automated diabetic retinopathy screening and grading. By enabling early and accurate classification of disease severity, this system can play a vital role in reducing the incidence of vision loss among diabetic patients. Its successful implementation could also lessen the burden on healthcare providers by supporting more efficient and accessible screening processes, particularly in areas with limited access to specialized ophthalmic services.

2.3 DIABETIC RETINOPATHY DETECTION USING MACHINE LEARN ING AND TEXTURE FEATURES

Paper [2] examines the link between diabetes and diabetic retinopathy (DR), a serious medical condition in which prolonged high blood sugar levels lead to damage in the retina. This damage can result in blood leakage and progressive vision impairment. Diabetic retinopathy is commonly indicated by early signs such as Hemorrhages (HEM), Micro-aneurysms, hard Exudates, and other retinal abnormalities. Detecting Hemorrhages and Micro-aneurysms at an early stage is crucial to preventing irreversible blindness in diabetic patients. Early detection enables timely intervention, which can slow the progression of the disease and mitigate the risk of severe vision loss.

This study explores image analysis techniques that can identify and classify the early signs of DR using retinal fundus images. In particular, the researchers focus on feature extraction methods that use Local Ternary Patterns (LTP) and Texture-based Shape Histograms (LESH) for more accurate identification of retinal abnormalities. Previous studies have commonly used Local Binary Patterns (LBP) for feature extraction; however, this paper demonstrates that LTP-like features and LESH perform better than LBP-based techniques in identifying diabetic retinopathy markers. By capturing finer texture details and shape characteristics in retinal images, these advanced feature extraction methods improve the diagnostic accuracy of the computer-assisted diagnosis (CAD) system.

To classify the extracted histogram features, the study utilizes Support Vector Machines (SVM), which is an effective classifier for handling medical image data. Specifically, the study employs SVM with a radial basis function (RBF) kernel (SVM-RBF) to improve classification performance. In conclusion, this paper presents a CAD system that significantly enhances the accuracy of diabetic retinopathy detection through advanced feature extraction and classification techniques. By utilizing LESH and SVM-RBF, the system offers high precision and reliability, making it a valuable tool for early screening of diabetic retinopathy.

2.4 DIABETIC RETINOPATHY DETECTION AND CLASSIFICATION USING CAPSULE NETWORKS

Paper [3] examines diabetic retinopathy, a leading cause of blindness among individuals with diabetes. Diabetic retinopathy arises from prolonged high blood sugar levels, which can damage the retina's blood vessels and lead to severe vision impairment. Early and timely detection is crucial for improving prognosis, as it allows for appropriate interventions that can slow the disease's progression and prevent irreversible vision loss. Advances in machine learning and neural network techniques, particularly convolutional neural networks (CNNs), have become instrumental in automating the detection of diabetic retinopathy, enabling faster and more accurate diagnosis.

To evaluate the effectiveness of the proposed network, the researchers conducted experiments using the Messidor dataset, a widely-used dataset for retinal image analysis. The results demonstrated that the capsule network achieved impressive accuracy rates in classifying diabetic retinopathy stages. Specifically, the network achieved accuracy rates of 97.98% for healthy retina images, 97.65% for stage one, 97.65% for stage two, and 98.64% for stage three fundus images. These high accuracy rates suggest that the modified capsule network is capable of reliably identifying and distinguishing between various stages of diabetic retinopathy, which is crucial for guiding treatment decisions.

In conclusion, the modified capsule network presented in Paper [3] demonstrates high accuracy in identifying diabetic retinopathy stages, showcasing its potential as an effective tool for early and accurate diagnosis of this sight-threatening condition. By leveraging the power of capsule networks, the proposed system provides a robust solution for detecting diabetic retinopathy in retinal fundus images, which could ultimately help in preserving vision and improving the quality of life for diabetic patients.

2.5 DIABETIC RETINOPATHY DETECTION USING VGG-NIN A DEEP LEARNING ARCHITECTURE

Paper [4] examines the rapid rise of diabetic retinopathy (DR), a major health concern that now affects approximately 382 million people globally. This number is projected to grow significantly, reaching an estimated 592 million by 2025, highlighting the urgent need for effective screening and diagnostic tools. Diabetic retinopathy is a serious eye condition that results from damage to the retina's blood vessels, caused by prolonged high blood sugar levels associated with diabetes mellitus (DM). DR can progress to more severe forms, such as proliferative diabetic retinopathy (PDR) and diabetic macular edema (DME), both of which can lead to substantial vision impairment or blindness if not managed effectively. To address this pressing issue, the research presented in Paper [4] proposes a computer-assisted diagnosis (CAD) system that utilizes deep learning architectures to classify different stages of diabetic retinopathy. Specifically, the study incorporates a VGG-16 model enhanced with a spatial pyramid pooling (SPP) layer and a network-in-network (NiN) structure.

The VGG-16 model, a widely-used convolutional neural network known for its depth and simplicity, is optimized in this study to improve the model's ability to analyze and classify high-resolution DR images. Experimental results show that the proposed SPP-based VGG-NiN model not only achieves higher classification accuracy than existing state-of-the-art methods but also optimizes computational resource usage. The study's findings suggest that this model is a promising solution for large-scale diabetic retinopathy screening, as it can process high volumes of retinal images efficiently without compromising performance. In conclusion, Paper [4] presents a CAD system that leverages a VGG-16 architecture enhanced with SPP and NiN layers, providing a scalable and accurate solution for detecting diabetic retinopathy. The proposed SPP-based VGG-NiN model demonstrates excellent performance in classifying different stages of DR, with both high accuracy and efficient computational resource usage. This system holds significant potential for large-scale DR screening and could play a crucial role in reducing the incidence of vision loss among diabetic populations.

2.6 DEEP LEARNING BASED DETECTION OF DIABETIC RETINO PATHY USING RETINAL FUNDUS IMAGES

Paper [5] examines a deep learning-based approach for detecting diabetic retinopathy (DR) from retinal fundus images. Diabetic retinopathy is a diabetes-related condition that progressively damages the blood vessels in the retina, leading to blurred vision and eventually partial or complete blindness as it worsens. This disease may begin without noticeable symptoms, affecting both eyes over time and often going undetected until vision impairment occurs. Early detection is crucial, as timely intervention can prevent the progression to full blindness. Therefore, the primary objective of this study is to develop an automated system for early DR detection using advanced image analysis techniques.

The research employs deep learning methods, particularly convolutional neural networks (CNNs), to identify and classify diabetic retinopathy by focusing on four key retinal features: microaneurysms, blood vessels, hemorrhages, and exudates. These features are essential indicators of DR and can be used to distinguish between various stages of the disease. The CNN model was designed to automatically extract and analyze these features from retinal images, providing a reliable, efficient means of screening for DR.

To improve the accuracy of the model, the researchers utilized DenseNet-169, a pre-trained CNN model, as part of a transfer learning approach. Transfer learning leverages the knowledge gained from training on large, diverse datasets, allowing the model to achieve high performance even with a relatively limited number of retinal images. In conclusion, the study presented in Paper [5] successfully demonstrates a deep learning approach to diabetic retinopathy detection using retinal fundus images. By employing DenseNet-169 with transfer learning and fine-tuning techniques, the model achieved high accuracy in identifying key DR features, offering a valuable tool for early screening. This automated system could aid in the early detection and treatment of diabetic retinopathy, potentially preventing vision loss and improving outcomes for diabetic patients.

2.7 AN AUTOMATED DETECTION AND MULTI-STAGE CLASSIFICATION OF DIABETIC RETINOPATHY USING CONVOLUTIONAL NEURAL NETWORKS

Paper [6] addresses diabetic retinopathy (DR), a frequent complication of diabetes mellitus that causes damage to the retina and can result in blindness if not detected early. DR leads to vision-impairing lesions that progressively worsen over time. Detecting DR in its early stages is essential to reduce the risk of permanent vision loss, and effective early diagnosis can enable timely intervention and treatment, significantly improving patient outcomes. Convolutional Neural Networks (CNNs), a form of deep learning, have become an essential tool in medical image analysis and classification due to their ability to recognize complex patterns in images.

This study proposes a novel technique for detecting diabetic retinopathy using a custom-designed CNN model, known as the DiaNet Model (DNM). The model employs several advanced image preprocessing and augmentation techniques to enhance its accuracy and efficiency. For preprocessing, a Gabor filter is applied to retinal fundus images to improve the visibility of blood vessels and enhance texture details. This filter plays a critical role in emphasizing essential features such as vessel patterns, which are crucial indicators of DR. By enhancing the visibility of these patterns, the Gabor filter aids in more accurate texture analysis, feature extraction, and object recognition, making it easier for the model to identify DR-related abnormalities.

The DiaNet Model achieved a mean accuracy rate of 90.02%, demonstrating its effectiveness for early DR detection. The research indicates that this model, with its Gabor filter-enhanced preprocessing and PCA-optimized architecture, is a promising solution for automated DR screening. By efficiently identifying diabetic retinopathy at early stages, the DiaNet Model can assist healthcare professionals in making timely interventions, potentially preventing vision impairment and improving the quality of life for diabetic patients.

2.8 COMPARISON BETWEEN THE RELATED WORK

Table 2.1 Comparison between the related work

Sl. No	Authors and Year	Aim of the Article/Research	Model/Algorithm/Method/Protocol Used	Characteristics/ Features	Merits/Findings	Limitations/ Drawbacks
1	E. V. Carrera et al. [1], 2017	To develop a system for early detection and grading of diabetic retinopathy.	Support vector machine (SVM).	Extraction for classification.	High accuracy in grading DR.	Limited to non-proliferative diabetic retinopathy.
2	M. Chetoui et al. [2], 2018	To develop a CAD system for early detection of DR.	SVM with Radial Basis Function (RBF) kernel.	Extraction using Local Ternary Pattern (LTP) and Local Energy-Based Shape Histogram (LESH.)	Effective at distinguishing subtle retinal features.	Challenges with HEM detection due to lighting and contrast.
3	Kalyani et al. [3], 2021	To detect and classify diabetic retinopathy using a reformed capsule network.	Reformed Capsule Network with Convolution and Primary Capsule Layers.	Extraction from fundus images.	Demonstrates the ability to handle varying stages of DR effectively.	Potential need for further optimization for diverse image conditions.
4	Z. Khan et al. [4], 2021	To classify different stages of diabetic retinopathy using an enhanced model.	VGG-16 with Spatial Pyramid Pooling (SPP) and Network-in- Network (NiN).	VGG-16 with Spatial Pyramid Pooling (SPP) and Network-in-Network (NiN).	Effective at processing DR images at varying scales.	Potential complexity in model training and deployment.

5	C. U. Kumari et al. [5], 2022	To detect diabetic retinopathy using deep learning on retinal fundus images.	Convolutional Neural Network (CNN) with DenseNet-169 and Conv2 layer.	Fine-tuning with DenseNet- 169.	Effective at detecting multiple key features of DR.	Model performance may vary with different datasets or image conditions.
6	N. S et al. [6], 2023	To develop a method for detecting diabetic retinopathy using the DiaNet Model (DNM).	DiaNet Model (DNM) with Gabor filter and PCA for image pre- processing and augmentation	Gabor filter for improving visibility of blood vessels and texture analysis.	Effective pre-processing and augmentation techniques.	The method's effectiveness may vary with different datasets or retinal image conditions.

2.9 CONCLUSION

Detailed research into machine learning methods for detecting diabetic retinopathy (DR) has revealed significant potential for early and accurate diagnosis of this sight-threatening condition. As DR progresses, it can lead to vision impairment and blindness if left undiagnosed and untreated. Machine learning, particularly deep learning models like Convolutional Neural Networks (CNNs), has shown great promise in accurately recognizing and categorizing the different stages of DR from retinal images. By automating this process, these models can detect subtle changes in the retina that may be overlooked by human observers, providing a more consistent and timely diagnosis. The use of CNNs, with their ability to automatically learn relevant features from large datasets of retinal images, significantly reduces the time and effort required for manual analysis.

The integration of advanced machine learning techniques for DR detection offers multiple advantages. One of the most notable is the high accuracy these systems can achieve in classifying DR stages, which is crucial for early intervention. These systems can process large amounts of image data efficiently, which is vital for large-scale screening programs that can reach populations at risk of DR, especially in regions with limited access to specialists. Furthermore, machine learning models can handle various image conditions, such as differences in image quality, lighting, and resolution, making them adaptable to real-world scenarios where retinal images may vary. Additionally, these models can analyze complex patterns and features related to DR that may be difficult for human clinicians to identify, enabling the detection of DR at earlier, more treatable stages. As a result, the use of machine learning for DR detection not only increases diagnostic efficiency but also enhances the ability to implement widespread screening programs, leading to earlier treatments, better patient outcomes, and a reduced risk of vision loss.

CHAPTER 3

PROJECT PLAN

A project plan is a formal document designed to guide the control and execution of a project. A project plan is the key to a successful project and is the most important document that needs to be created when starting any business project.

In IT, the term project plan refers to a Gantt chart or any other document that displays project activities along a timeline. However, considering these documents alone as a project plan is inaccurate. These particular documents can be more precisely termed as project schedules, and may be considered only a part of the actual project plan.

A project plan is used for the following purposes:

- To document and communicate stakeholder products and project expectations
- To control schedule and delivery
- To calculate and manage associated risks

3.1 GANTT CHART

A Gantt chart is a project management tool that illustrates a project plan. It is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

- What the various activities are
- When each activity begins and ends
- How long each activity is scheduled to last
- Where activities overlap with other activities, and by how much
- The start and end date of the whole project

To summarize, a Gantt chart shows you what has to be done (the activities) and when (the schedule).

Fig. 3.1 shows Gantt chart. It helps to plan work around deadlines and properly allocate resources. Here Gantt chart starts from Evaluation by Project Guide. It starts from 09th December 2024 to 28th March 2025. Project Evaluation by Project Guide starts only after topic selection is successfully completed. Formulation of Work Plan is done between 09th December 2024 and 31st December 2024. Implementation starts from 06th January 2025 and continues until 28th March 2025. Model Training is conducted from 06th January 2025 to 21st March 2025. Application Designing is done between 13th January 2025 and 25th March 2025. Testing is carried out from 13th January 2025 to 28th March 2025. First Interim Evaluation takes place from 09th January 2025 to 10th January 2025. Second Interim Evaluation is conducted from 25th March 2025 to 28th March 2025. Project Phase 2 Report is completed between 27th March 2025 and 02nd April 2025. Final Evaluation is planned from 07th April 2025 to 10th April 2025.

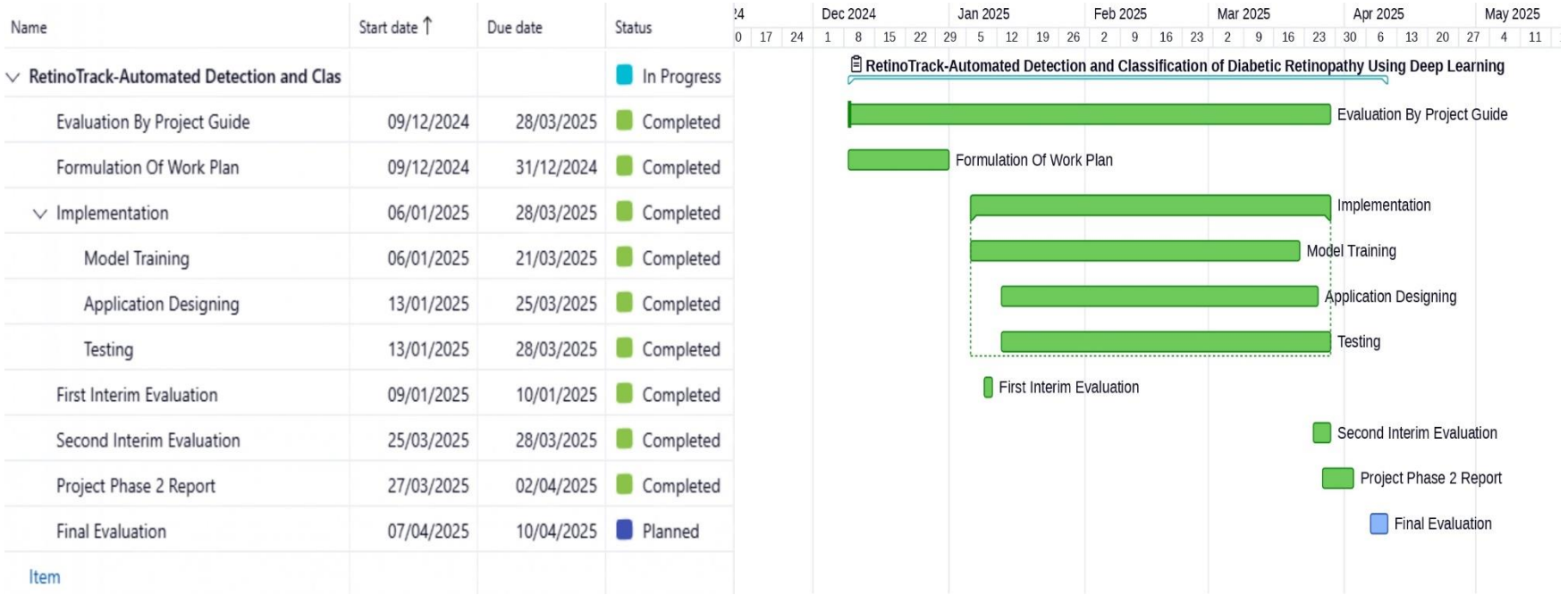


Fig. 3.1 Gantt Chart

3.2 WORK BREAKDOWN STRUCTURE (WBS)

Work breakdown structure (WBS) in project management is a method for completing a complex, multi-step project. It's a way to divide and conquer large projects to get things done faster and more efficiently.

The goal of a WBS is to make a large project more manageable. Breaking it down into smaller chunks means work can be done simultaneously by different team members, leading to better team productivity and easier project management.

Before we create a work breakdown structure, it's essential to first assess the project scope by talking to all stakeholders and key team members involved. We want to ensure that all critical input and deliverables are gathered and transparently prioritized. We may use Gantt charts, flow charts, spreadsheets, or lists to show the hierarchical outline of importance and connectivity between the tasks needed to complete the project.

A WBS structure must be constructed in a way that each new level in the hierarchy includes all the work needed to complete its parent task. This means that every parent task element must have more than one child task within it to consider the parent task element complete.

Work breakdown structure for each project can be different. We can select the WBS which works best for us and our team. The goal is to show the hierarchy of the projects and make progress clear to everyone involved — whether they are a team member or an external stakeholder.

Following are some work breakdown structure examples.

1. WBS spreadsheet: We can structure our WBS efficiently in a spreadsheet, noting the different phases, tasks, or deliverables in the columns and rows.
2. WBS flowchart: We can structure our WBS in a diagrammatic workflow. Most WBS examples and templates are flowcharts.
3. WBS list: We can structure our WBS as a simple list of tasks or deliverables and subtasks. This is the most straightforward approach to make a WBS.
4. Work breakdown structure Gantt chart: We can structure our WBS as a Gantt chart that represents both a spreadsheet and a timeline. With a Gantt chart-structured WBS, we can link task dependencies and show project milestones.

When created thoroughly, the work breakdown structure is a roadmap that guides a team when completing projects — whether simple or complex.

Here's our work breakdown structure. This diagram outlines a project workflow for a Automated Detection And Classification Of Diabetic Retinopathy Using Deep Learning, divided into four main stages: Topic Selection, System Development, and Test/Report. In the Topic Selection phase, the team identifies a topic, consults with a guide and coordinator, and gets approval. In System Design, they conduct a literature review, define the system architecture, create a flowchart, and complete UML and other diagrams. The Development phase involves designing the user interface, coding, and training datasets. Finally, the Test/Report stage includes testing the system, preparing a final report, and submitting it.

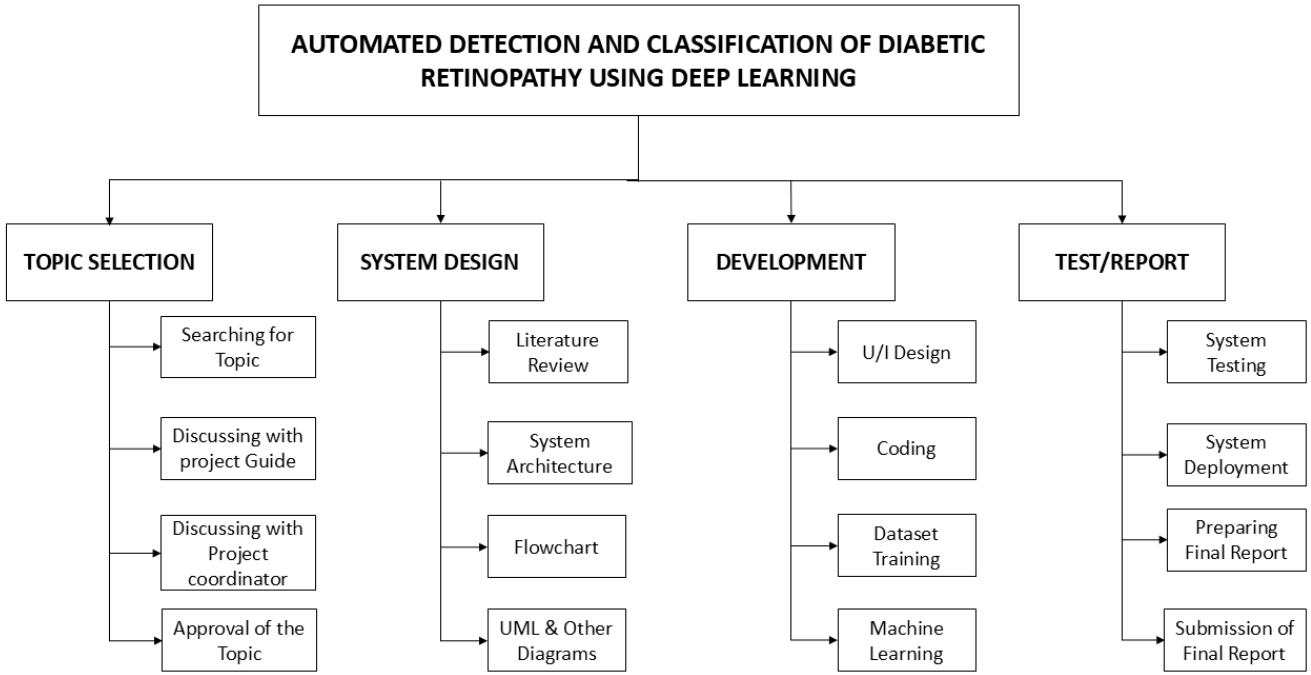


Fig. 3.2 Work Breakdown Structure

CHAPTER 4

METHODOLOGY

4.1 SYSTEM DESCRIPTION

- Initially, the user will upload a retinal image for analysis.
- After uploading, the system will preprocess the image to enhance its quality for accurate analysis.
- If the image quality is insufficient, the user will be prompted to upload a clearer image.
- Once the image is processed, the deep learning model will analyze the retinal features to detect signs of diabetic retinopathy (DR).
- Based on the analysis, the model will classify the stage of diabetic retinopathy (from no DR to severe stages) using the extracted features.
- Finally, the classified DR stage will be delivered to the user as the end Report, providing valuable insights for early intervention and treatment planning.

4.2 SYSTEM ARCHITECTURE

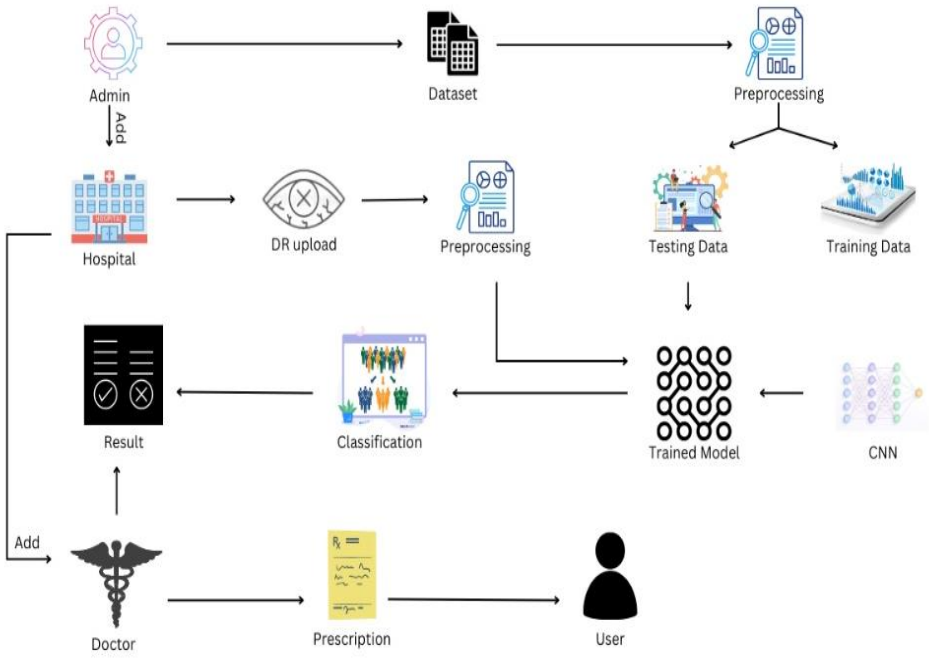


Fig 4.1 System Architecture

4.3 FLOW CHART

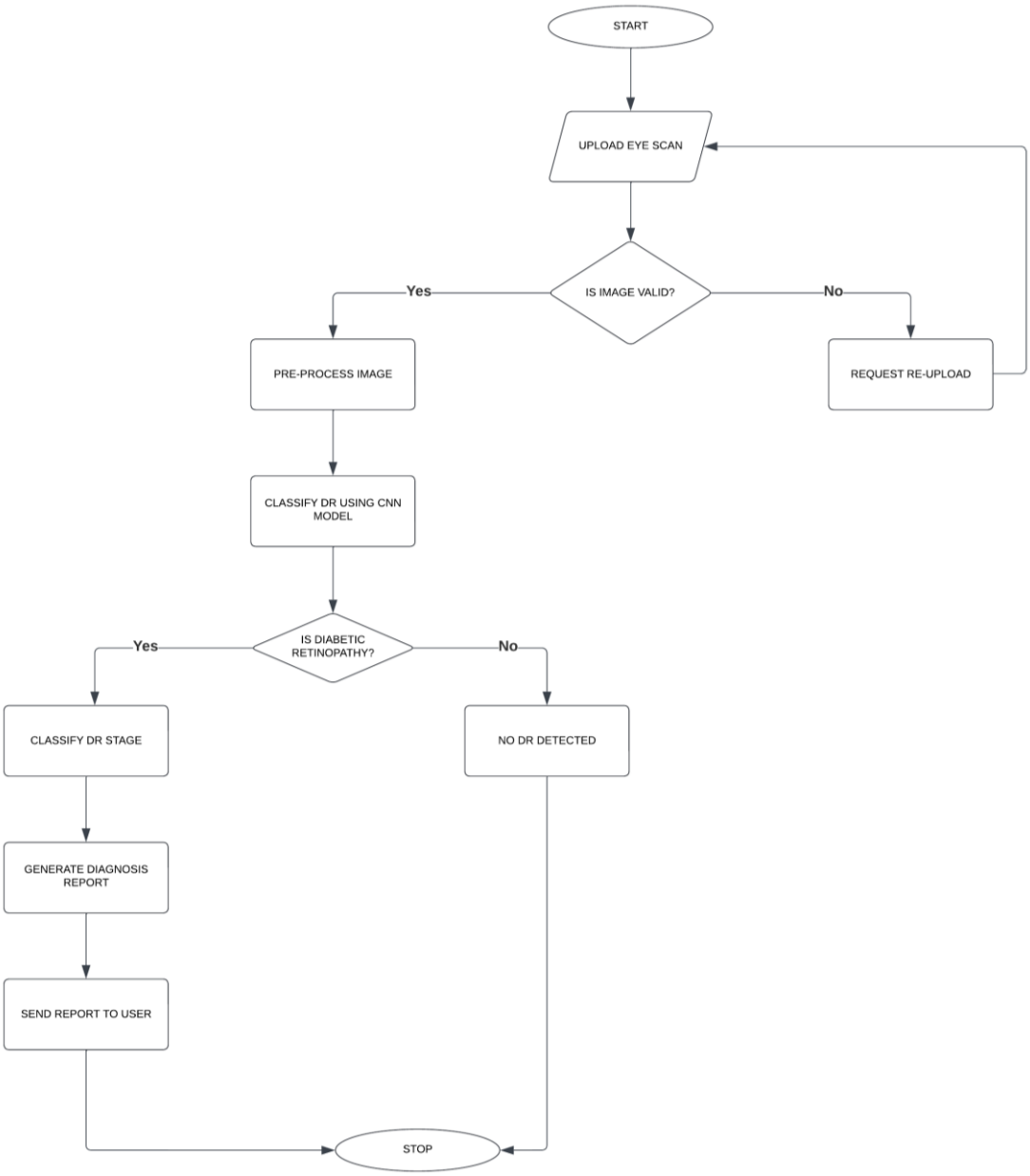


Fig 4.2 flow chart

CHAPTER 5

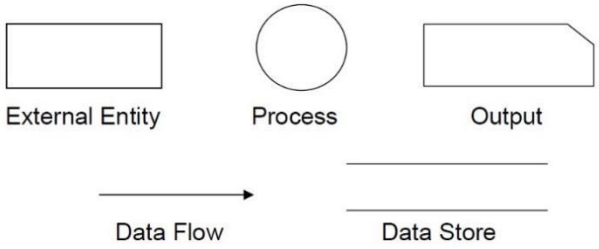
SYSTEM DESIGN

System Design is the process of designing the architecture, components, and interfaces for a system so that it meets the end-user requirements. We have created Data Flow Diagram level 0 and 1, Use Case diagrams, Class diagram, Sequence diagram, Activity diagram. Dataflow diagram visually represent systems and processes that would be hard to describe in a chunk of text. While Use case diagram is the written description of how users will perform tasks on your website. Class diagrams are fundamental to the object modeling process and model the static structure of a system. Sequence diagram is interactive based and Activity diagram a behavioral diagram.

5.1 DATA FLOW DIAGRAM (DFD)

The DFD (also known as a bubble chart) is a hierarchical graphical model of a system that shows the different processing activities or functions that the system performs and the data interchange among these functions. Each function is considered as a processing station (or process) that consumes some input data and produces some output data. The system is represented in terms of the input data to the system, various processing carried out on these data, and the output data generated by the system. A DFD model uses a very limited number of primitive symbols to represent the functions performed by a system and the data flow among these functions.

DFD Symbols



A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

Level 0 DFD

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

Level 1 DFD

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.

DFD Level 2 then goes one step deeper into parts of Level 1. It may require more text to reach the necessary level of detail about the system's functioning.
We have to keep in mind following rules and tips:

- A context diagram should depict the system as a single bubble.
- All external entities interacting with the system should be represented only in the context diagram. The external entities should not appear at other levels of the DFD.
- It is a common oversight to have either too less or too many bubbles in a DFD. Only 3 to 7 bubbles per diagram should be allowed, i.e. each bubble should be decomposed to between 3 and 7 bubbles.
- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.
- All processes in a DFD go to another process or a data store.

Fig 5.1 represses DFD Level 0. User will input the Eye Scan Image. And image will send to Diabetic Retinopathy Detection System, after analyzing the image result will send to user.

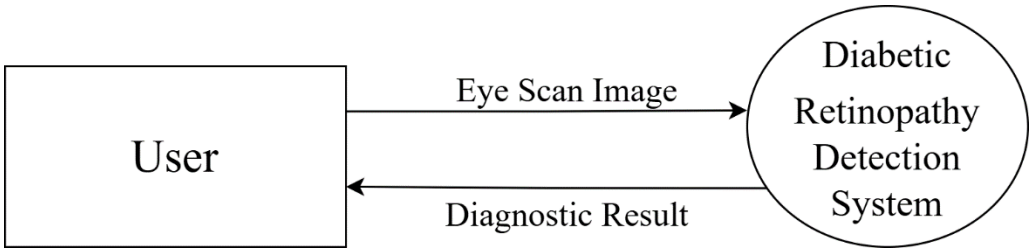


Fig 5.1 Data Flow Diagram Level 0

Fig. 5.2 shows Level 1. Here, the flow starts from the user. The user uploads an eye scan image. The system then processes the image, where pre-processing steps are applied to enhance the quality of the input. Following this, the system classifies the diabetic retinopathy (DR) stage based on the pre-processed image. After classification, a diagnosis report is generated, detailing the DR stage and any relevant findings. Finally, the user receives the diagnosis report as the output of the process.

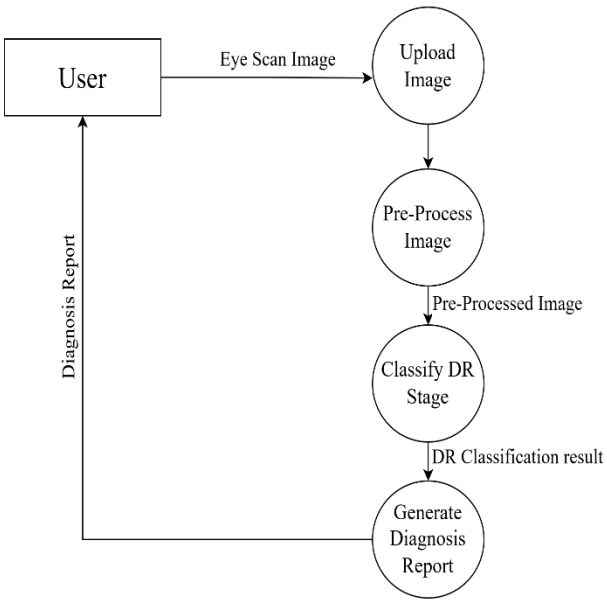


Fig 5.2 Data Flow Diagram Level 1

5.2 USE CASE DIAGRAM

Use case diagrams are considered for high level requirement analysis of a system. When the requirements of a system are analyzed, the functionalities are captured in use cases.

We can say that use cases are nothing but the system functionalities written in an organized manner. The second thing which is relevant to use cases are the actors. Actors can be defined as something that interacts with the system.

Actors can be a human user, some internal applications, or may be some external applications. When we are planning to draw a use case diagram, we should have the following items identified.

- Functionalities to be represented as use case
- Actors
- Relationships among the use cases and actors

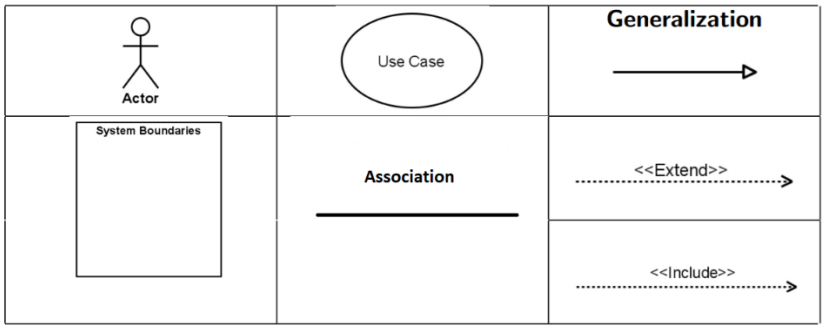
Use case diagrams specify the events of a system and their flows. But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output, and the function of the black box is known. These diagrams are used at a very high level of design. This high level design is refined again and again to get a complete and practical picture of the system.

A use case is a sequence of transactions performed by a system that yields an outwardly visible, measurable result of value for a particular actor. A use case typically represents a major piece of functionality that is complete from beginning to end.

In UML, a use case is represented as an ellipse. Give your use case a unique name expressed in a few words (generally no more than five words). An actor represents whoever or whatever (person, machine, or other) interacts with the system. The actor is not part of the system itself and represents anyone or anything that must interact with the system to:

- Input information to the system;
- Receive information from the system; or
- Both input information to and receive information from the system.

Arrows and lines are drawn between actors and use cases and between use cases to show their relationships. The default relationship between an actor and a use case is the «communicates» relationship, denoted by a line.



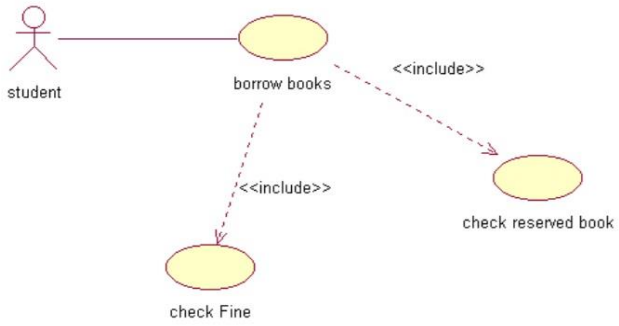
There are two other kinds of relationships between use cases (not between actors and use cases) that you might find useful. These are «include» and «extend». The «include» relationship is not the default relationship. Therefore, in a use case diagram, the arrow is labeled with «include» when

one use case makes full use of another use case. You use the «extend» relationship when you are describing a variation on normal behavior or behavior that is only executed under certain, stated conditions.

It is common to be confused as to whether to use the include relationship or the extend relationship. Consider the following distinctions between the two:

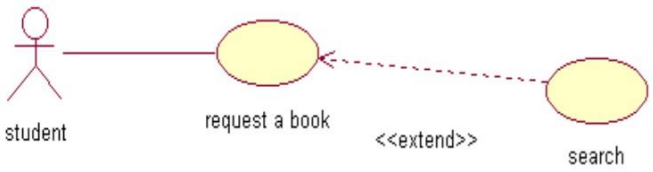
• Use Case X includes Use Case Y:

X has a multi-step subtask Y. In the course of doing X or a subtask of X, Y will always be completed.



• Use Case X extends Use Case Y:

Y performs a sub-task and X is a similar but more specialized way of accomplishing that subtask (e.g. closing the door is a sub-task of Y; X provides a means for closing a blocked door with a few extra steps). X only happens in an exception situation. Y can complete without X ever happening.



This figure illustrates an Automated Detection and Classification of Diabetic Retinopathy system involving two actors: Patient and System. The Patient uploads retina scans, views classification results, and receives a diagnosis. The System processes these images, detects anomalies, classifies diabetic retinopathy stages, analyzes data, and generates reports. The process begins with the patient uploading a scan, which the system processes to detect anomalies and classify the stage of retinopathy. The system then analyzes the data, generates a report, and presents the results to the patient. This enables early detection and classification, aiding patients in understanding their condition and seeking timely care.

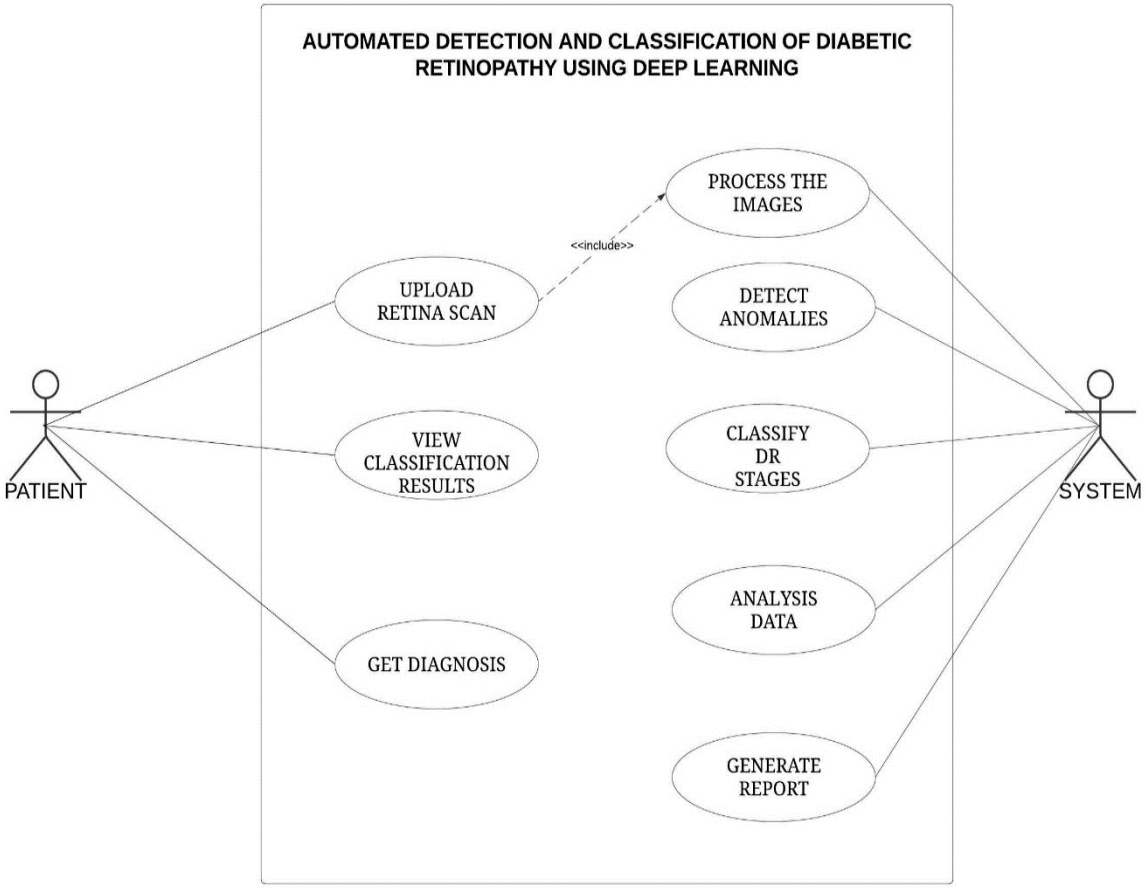


Fig. 5.3 Use Case Diagram

5.3 CLASS DIAGRAM

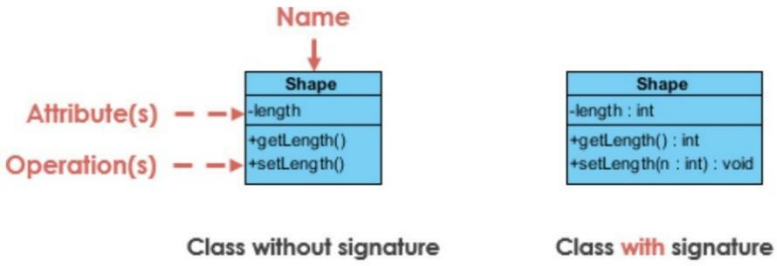
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

UML Class Notation

A class represent a concept which encapsulates state (attributes) and behavior (operations). Each attribute has a type. Each operation has a signature. The class name is the only mandatory information.



Class Name:

The name of the class appears in the first partition.

Class Attributes:

Attributes are shown in the second partition.

The attribute type is shown after the colon.

Attributes map onto member variables (data members) in code.

Class Operations (Methods):

Operations are shown in the third partition. They are services the class provides. The return type of a method is shown after the colon at the end of the method signature. The return type of method parameters are shown after the colon following the parameter name. Operations map onto class methods in code

Class Visibility

The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.

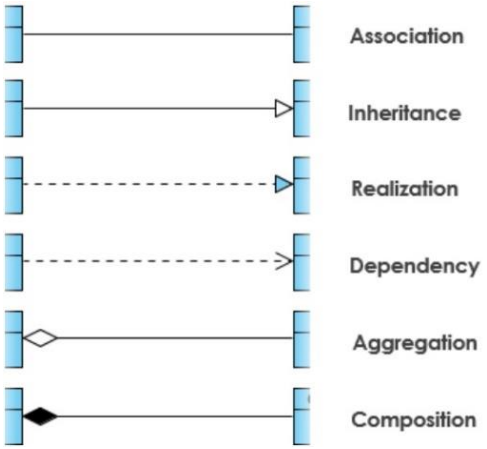
+ denotes public attributes or operations

- denotes private attributes or operations

# denotes protected attributes or operations

Relationships between classes

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:



Cardinality

How many objects of each class take part in the relationships and multiplicity can be expressed as:

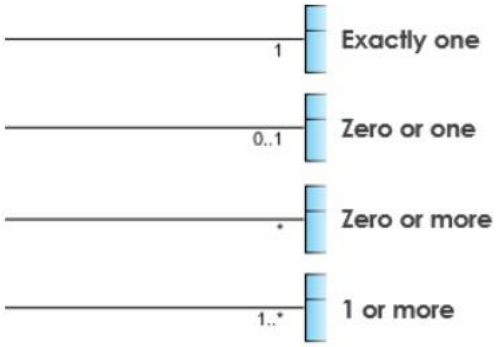


Figure 5.4 shows the class diagram, describing the attributes and operations of classes in the project: Patient, System, Image, Result, and Report. The Patient class stores patient information, including patientID, name, age, and email, and has methods like uploadImage() and viewResults(). The Image class, linked to Patient, represents uploaded eye images with attributes such as imageID and imageData, and includes methods like preprocess() for image preparation. The System class manages image processing and reporting through methods like classifyDR() and generateReport(). The Result class holds diagnostic outcomes, with fields like diagnosis and recommendations, while Report includes report details and provides methods like generatePDF() to share results with patients. Together, these classes facilitate image analysis and diabetic retinopathy diagnosis reporting for users.

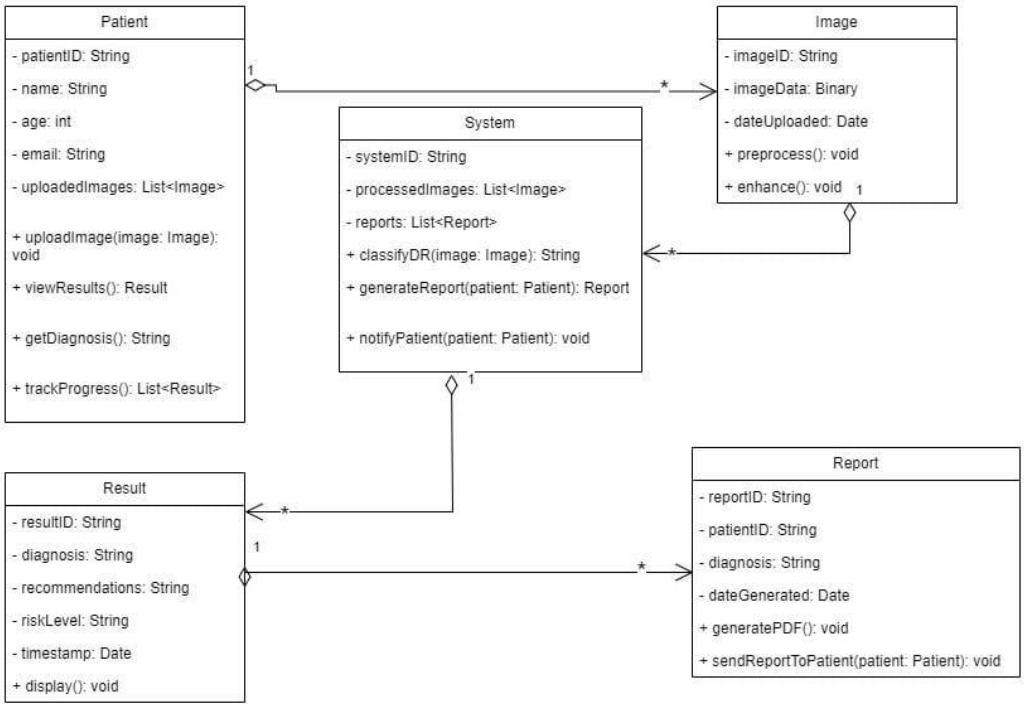


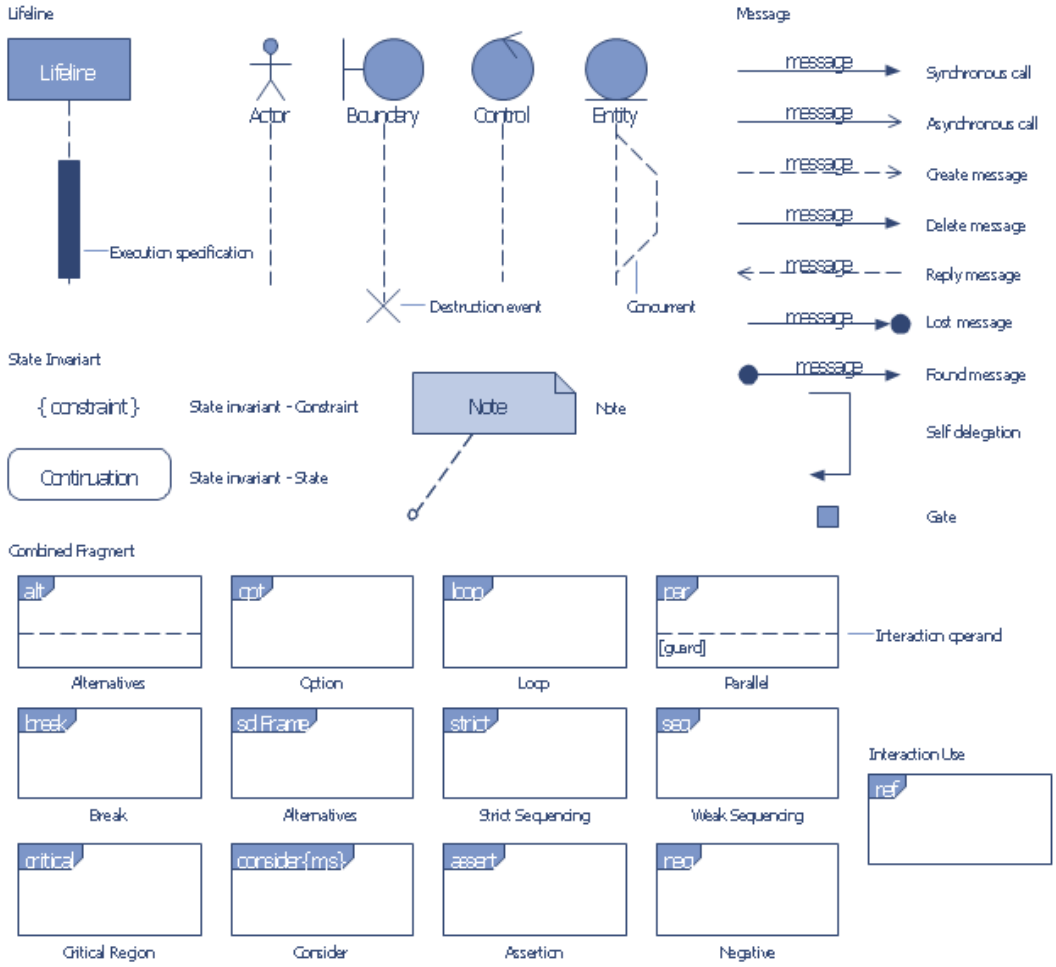
Fig. 5.4 Class Diagram

5.4 SEQUENCE DIAGRAM

The Sequence Diagram models the collaboration of objects based on a time sequence. It shows how the objects interact with others in a particular scenario of a use case. The diagrams are read left to right and descending.

In a sequence diagram, objects are shown in columns, with their object symbol on the top of the line. Similar to the class diagram, the object name appears in a rectangle. If a class name is specified, it appears before the colon. The object name always appears after a colon (even if no class name is specified). If an external actor initiates any interaction, the stick figure can be used rather than a rectangle.

A sequence diagram has two dimensions: the vertical dimension represents time; the horizontal dimension represents different objects. Initiation of the sequence starts in the top-left corner, and time proceeds down the page (from top to bottom). The vertical line is called the object's lifeline. There is no significance to the horizontal ordering of the objects.



A message sent from one object to another is shown as an arrow from the line of the sender to the line of the receiver. Each message is labeled at a minimum with message name. You can optionally include the arguments containing information that needs to be passed with the message. The reception of a message triggers a corresponding operation to execute. During this execution, other messages may be sent to other objects, and eventually the methods end. An object may send a message to itself. This is shown by an arrow from the object line to the same line. The method execution is represented in the sequence diagram by a thickening of the object line. Sequence diagram shows interaction in time sequence.

The sequence diagram illustrates the step-by-step process for automated diabetic retinopathy (DR) detection using an eye scan image. The user initiates the process by uploading an eye scan image to the system. The system validates the uploaded image for quality and clarity; if the image is deemed invalid, the user is prompted to re-upload a valid image. When the image is valid, it is sent to the Image Pre-Processor, which enhances and prepares it for further analysis. The pre-processed image is then passed to the CNN model, which analyzes the image to detect any signs of diabetic retinopathy. If DR is detected, the system moves to the Report Generator, which produces a detailed diagnosis report, including the identified DR stage (e.g., mild, moderate, or severe). If no DR signs are found, the Report Generator produces a report confirming no DR detected. This diagnosis report, whether positive or negative, is then sent back to the user for review. The sequence concludes with the user receiving an accurate and timely diagnosis report, allowing for further medical consultation if needed. This automated sequence provides a streamlined, efficient method for DR screening using deep learning.

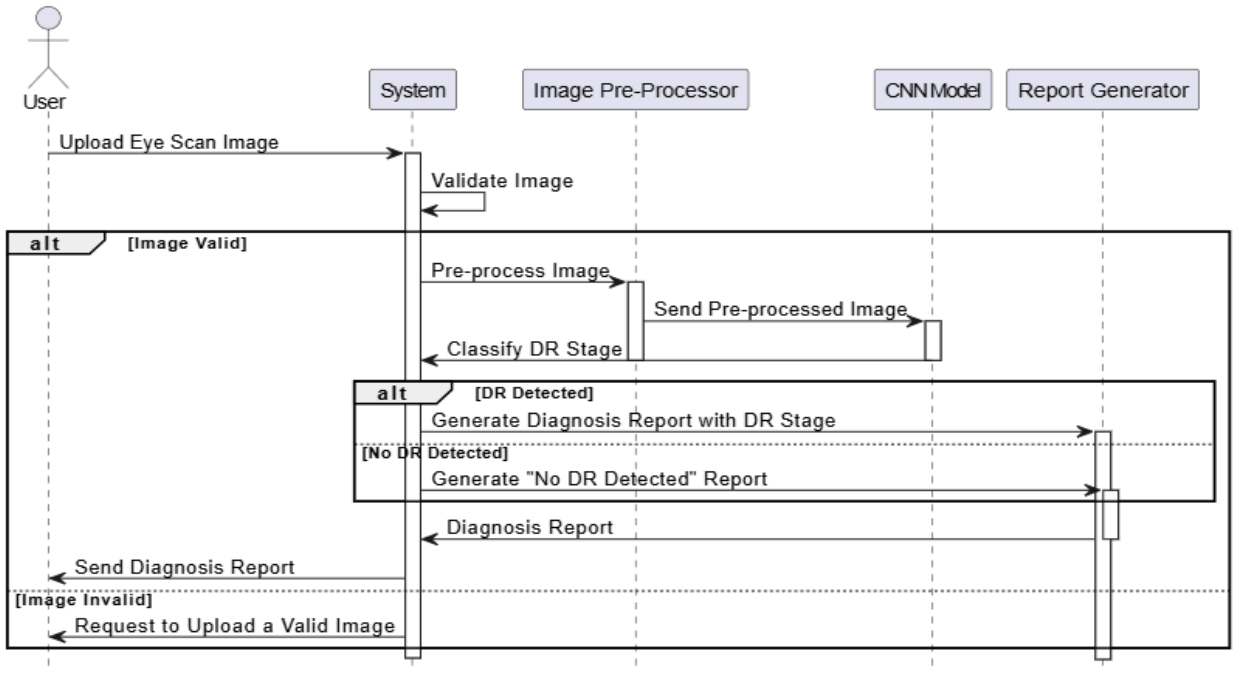
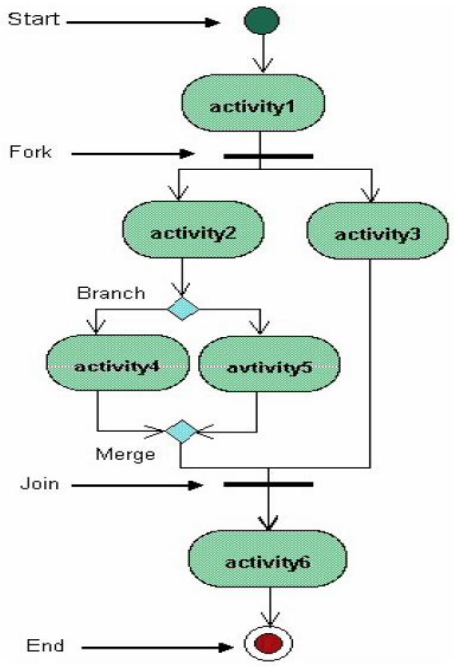


Fig. 5.5 Sequence Diagram

5.5 ACTIVITY DIAGRAM

Activity diagram captures the dynamic behavior of the system. It is is used to show message flow from one activity to another. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

Activity diagrams show the flow of activities through the system. Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities. A fork is used when multiple activities are occurring at the same time. The diagram below shows a fork after activity1. This indicates that both activity2 and activity3 are occurring at the same time. After activity2 there is a branch. The branch describes what activities will take place based on a set of conditions. All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch. After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.



In Fig. 5.5, the flowchart illustrates the process of analyzing an eye scan for diabetic retinopathy (DR). The process begins with the user uploading an eye scan image, followed by a validation step to check image quality. If the image is invalid, the user is prompted to re-upload a clearer version. For valid images, preprocessing enhances clarity, highlighting important features for accurate analysis. The preprocessed image is then classified using a CNN model to detect signs of DR. If DR is detected, the system classifies its severity stage; if not, it ends the process by notifying the user of no DR. For detected DR, a detailed diagnosis report is generated and provided to the user, completing the analysis. This workflow enables automated detection and classification of DR, facilitating early diagnosis and intervention.

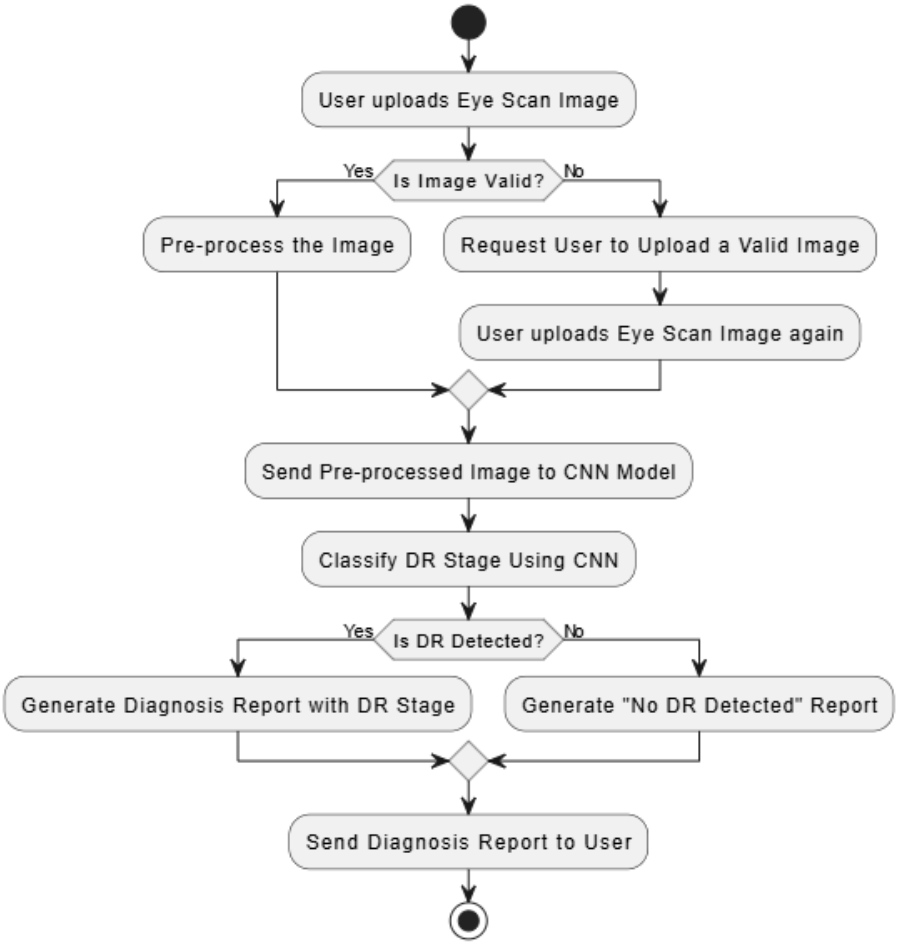


Fig. 5.6 Activity Diagram

5.6 ENTITY RELATIONSHIP DIAGRAM/DATABASE DIAGRAM

Entity Relationship Diagram, also known as ERD, ER Diagram or ER model, is a type of structural diagram for use in database design. An ERD contains different symbols and connectors that visualize two important information: The major entities within the system scope, and the inter-relationships among these entities. To fully utilize ER Diagram in database engineering guarantees you to produce high-quality database design to use in database creation, management, and maintenance.

ER models are mostly developed for designing relational databases in terms of concept visualization and in terms of physical database design.

Database diagrams graphically show the structure of the database and relations between database objects. It is a visual tool that helps you present a database using tables, columns, keys, and relationships. Using a database diagram, you can visualize and design any database to which you are connected. These diagrams are the foundation of database design and development, and it helps represent the basic structure of a database.

With such diagrams, you can plan how information is stored, categorized, and managed within a database. They create a graphical representation to understand various entities' attributes and relationships better. This further helps clearly understand the data structure and how you can minimize redundancy and other problems.

The database design process

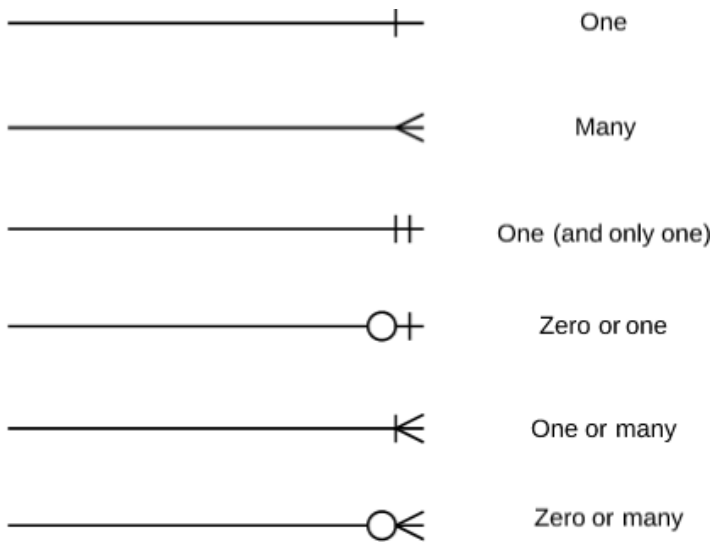
A well-structured database:

- Saves disk space by eliminating redundant data.
- Maintains data accuracy and integrity.
- Provides access to the data in useful ways.

Designing an efficient, useful database is a matter of following the proper process, including these phases:

1. Requirements analysis, or identifying the purpose of your database
2. Organizing data into tables
3. Specifying primary keys and analyzing relationships
4. Normalizing to standardize the tables

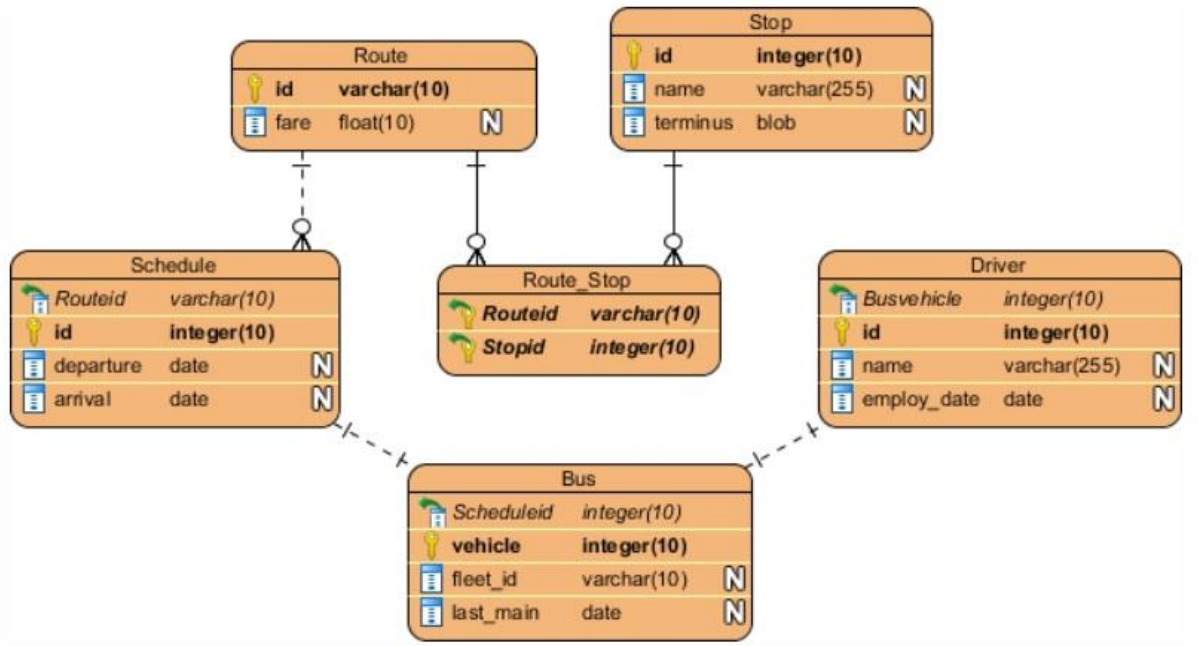
Cardinality in database:



Physical data model

Business analyst uses a conceptual and logical model to model the business objects exist in the system, while database designer or database engineer elaborates the conceptual and logical ER model to produce the physical ER model that presents the physical database structure ready for database creation.

Physical data model example:



Physical ERD represents the actual design blueprint of a relational database. A physical data model elaborates on the logical data model by assigning each column with type, length, nullable, etc. Since a physical ERD represents how data should be structured and related in a specific DBMS it is important to consider the convention and restriction of the actual database system in which the database will be created. Make sure the column types are supported by the DBMS and reserved words are not used in naming entities and columns.

This database diagram shows three entities: User, EyeScan, and Diagnosis. The User table stores user information, including user_id (primary key), name, email, role, and password. The EyeScan table contains data about each eye scan, with scan_id as the primary key and a foreign key user_id linking it to the User table. It also records the upload_time and image_path. The Diagnosis table holds diagnosis results for each scan, with diagnosis_id as the primary key and scan_id as a foreign key linking it to the EyeScan table. It includes fields such as dr_detected (a boolean indicating diabetic retinopathy detection), dr_stage (severity level), diagnosis_time, and report_path. This structure supports user management, eye scan tracking, and diagnosis storage for diabetic retinopathy classification.

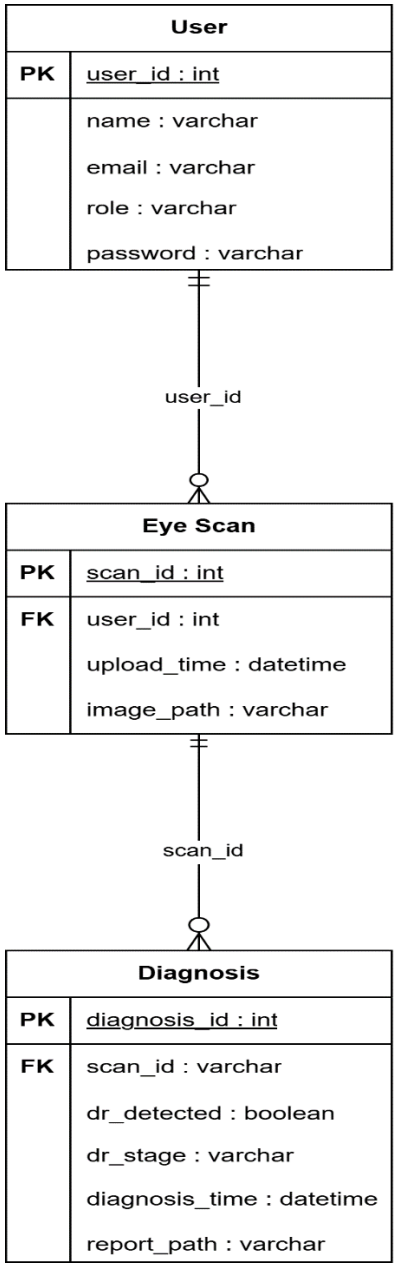


Fig. 5.7 Database Diagram

CHAPTER 6

EXPERIMENTAL SETUP

In our proposed system, we use deep learning to automate the detection of diabetic retinopathy (DR), a severe eye disease caused by prolonged high blood sugar levels that damage the retina. The system processes high-resolution retinal images to classify DR into different severity levels. The dataset used for training consists of labeled retinal images, which were preprocessed by resizing them to 224 × 224 pixels and normalizing pixel values to improve model performance. We employed the VGG16 convolutional neural network (CNN) for feature extraction and classification. The dataset was split into a training set and a validation set to ensure robust learning and evaluation.

The model was trained, tested, and validated using Python in Google Colab, a cloud-based Jupyter notebook environment that allows running Python without the need for complex local configurations. Colab provides pre-installed libraries and supports GPU acceleration, enabling efficient deep learning training. During training, we monitored key performance metrics, including training loss, validation loss, training accuracy, and validation accuracy. Our VGG16 model achieved an accuracy of 97.27% on the test dataset, outperforming other CNN architectures like MobileNet and ResNet.

To make the system accessible, we developed two mobile applications using MIT App Inventor. This visual programming tool allows the creation of fully functional apps using a block-based approach. Users can interact with the app to upload retinal images and receive diagnostic results. The backend is powered by Firebase, a cloud-hosted NoSQL database that stores data in JSON format and synchronizes information in real time. This integration ensures that DR detection is not only accurate but also efficient and accessible to users, helping in early diagnosis and timely medical intervention.

Fig. 6.1 MIT App Inventor Block Components

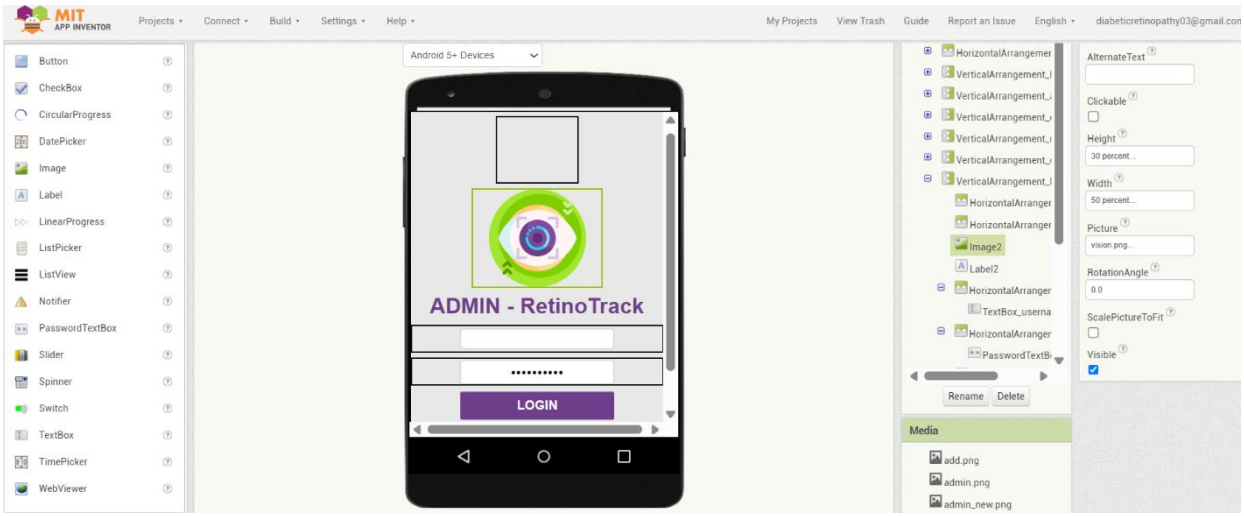


Fig. 6.2 MIT App Inventor Interface Design

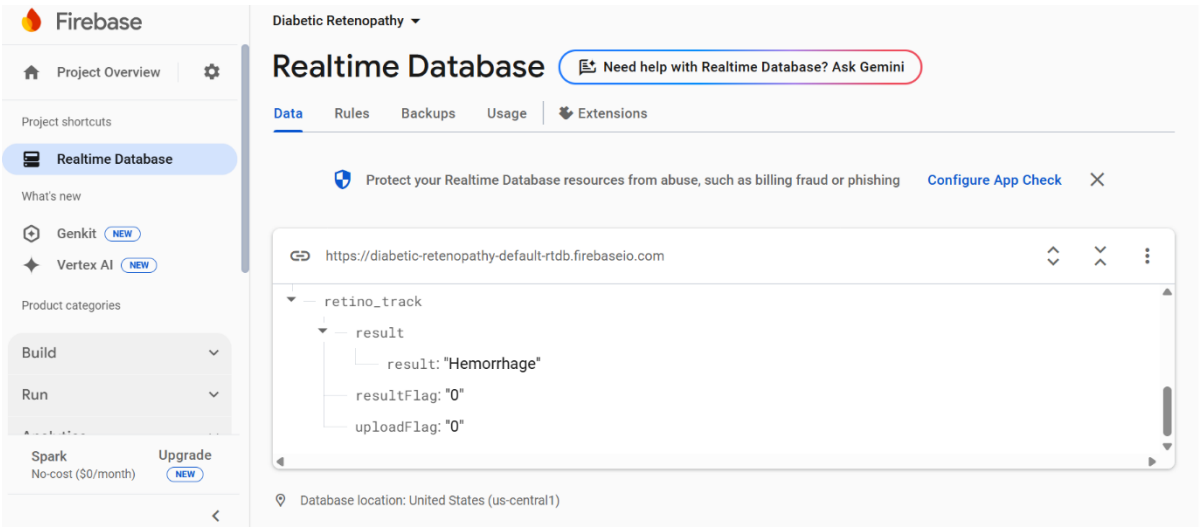


Fig. 6.3 Firebase real-time storage

CHAPTER 7

RESULTS AND DISCUSSIONS

In this project, we developed an automated diabetic retinopathy (DR) detection system that classifies retinal images into different severity levels. Users can upload retinal images, which are analyzed using the VGG16 convolutional neural network (CNN) to detect signs of DR. The dataset, collected from publicly available sources, was preprocessed by resizing images to 224 × 224 pixels and normalizing pixel values. The dataset was split into training and validation sets, and the model was trained in Google Colab using Python. Performance was monitored using training and validation loss graphs, along with accuracy metrics.

To evaluate the model, a confusion matrix was generated to analyze prediction outcomes. The system's accuracy, precision, recall, and F1-score were calculated, with VGG16 achieving 96.89% accuracy, outperforming other CNN architectures like MobileNet and ResNet. The project includes two mobile applications developed with MIT App Inventor, allowing users to upload images and receive diagnostic results. Firebase serves as the backend for secure real-time data storage and synchronization, ensuring an efficient and accessible DR detection system.

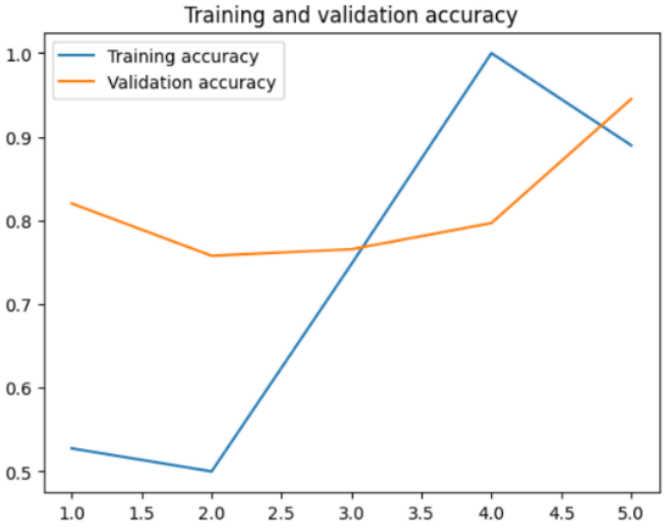


Fig. 7.1 Training and Validation Accuracy

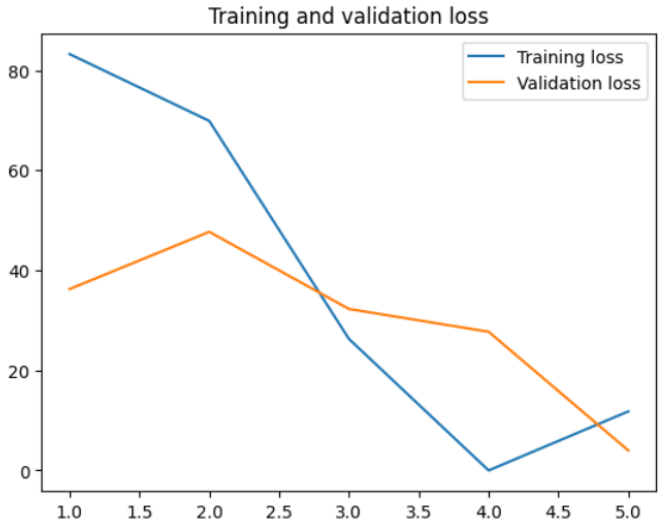
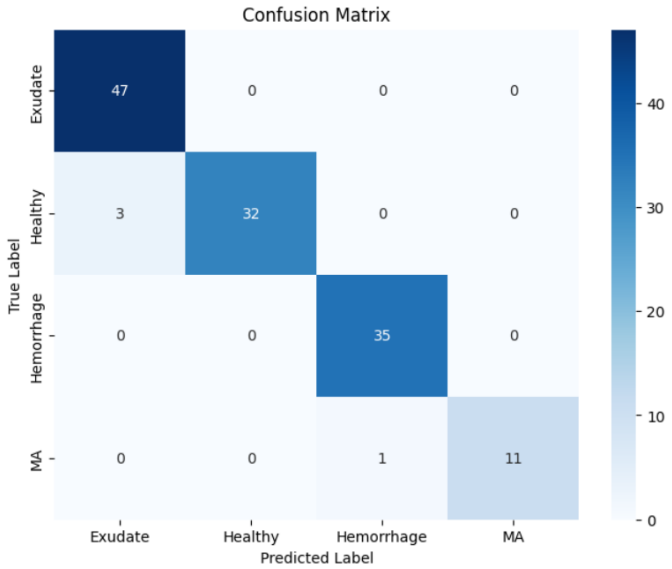


Fig. 7.2 Training and Validation loss



	precision	recall	f1-score	support
0	0.94	1.00	0.97	47
1	1.00	0.91	0.96	35
2	0.97	1.00	0.99	35
3	1.00	0.92	0.96	12
accuracy			0.97	129
macro avg	0.98	0.96	0.97	129
weighted avg	0.97	0.97	0.97	129

Test set accuracy: 0.9689922480620154

Fig. 7.3 Confusion Matrix and Results

CHAPTER 8

CONCLUSION

In this study, we investigated the automated detection and classification of diabetic retinopathy (DR) using deep learning methods, aiming to provide a reliable, non-invasive approach for early diagnosis and staging. Convolutional neural networks (CNNs) were utilized to analyze retinal images, enabling effective feature extraction and high classification accuracy across different stages of DR. The dataset was preprocessed by resizing images to 224 × 224 pixels and normalizing pixel values to enhance model performance. The VGG16 architecture was employed for classification, achieving a high degree of accuracy in detecting DR severity levels. Model training and evaluation were conducted in Google Colab using Python, with performance metrics such as accuracy, precision, recall, and F1-score analyzed to assess effectiveness.

The results establish a strong baseline performance, demonstrating that deep learning models can accurately classify DR and support early intervention. The CNN-based approach provides a significant advantage over traditional manual screening, reducing processing time and improving diagnostic accuracy. Additionally, mobile applications developed using MIT App Inventor enable users to upload retinal images and receive instant diagnostic results. Firebase serves as the backend for real-time data storage and synchronization, ensuring accessibility and efficient management of patient records.

Future research should focus on enhancing detection accuracy through advanced techniques such as refined image preprocessing, hyperparameter tuning, and ensemble modeling. Expanding the dataset with more diverse retinal images can improve model generalization and reliability across different patient populations. Additionally, integrating explainable AI techniques can increase model transparency, aiding healthcare professionals in understanding predictions. These improvements hold promise for developing a scalable, robust, and clinically applicable DR screening system, ultimately leading to better patient outcomes through early and accurate diagnosis.

REFERENCES

[1]. E. V. Carrera, A. González and R. Carrera, "Automated detection of diabetic retinopathy using SVM*," Proc. of the 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON),* Cusco, Peru, 2017.

[2] M. Chetoui, M. A. Akhloufi and M. Kardouchi, "Diabetic Retinopathy Detection Using Machine Learning and Texture Features", *Proc. of the 2018 IEEE Canadian Conference on Electrical & Computer Engineering (CCECE),* Quebec, QC, Canada, 2018.

[3] Kalyani, G., Janakiramaiah, B., Karuna, A. et al. Diabetic retinopathy detection and classification using capsule networks. *Complex Intelligent Systems*. 2021.

[4] Z. Khan et al., "Diabetic Retinopathy Detection Using VGG-NIN a Deep Learning Architecture," *in IEEE Access*, vol. 9, pp. 61408-61416, 2021.

[5] C. U. Kumari, A. Hemanth, V. Anand, D. S. Kumar, R. Naga Sanjeev and T. S. Sri Harshitha, "Deep Learning Based Detection of Diabetic Retinopathy using Retinal Fundus Images," *Proc. Of the 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT),* Kannur, India, 2022.

[6] N. S, S. S, M. J and S. C, "An Automated Detection and Multi-stage classification of Diabetic Retinopathy using Convolutional Neural Networks," *Proc. Of the 2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN),* Vellore, India, 2023.

APPENDICES

APPENDIX A

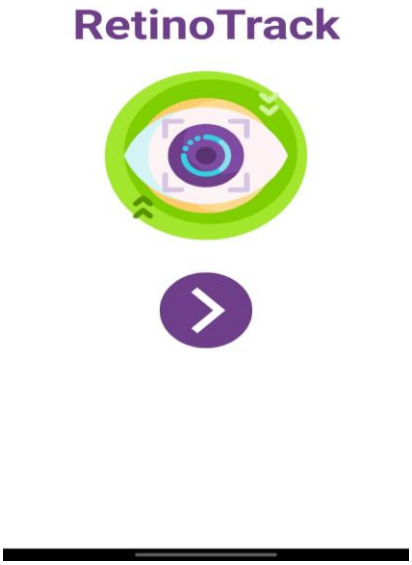


Fig. A.1 RetinoTrack Application

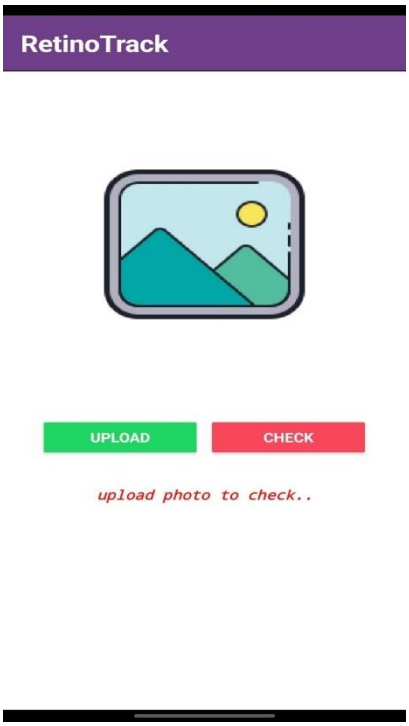


Fig. A.2 RetinoTrack Home Page

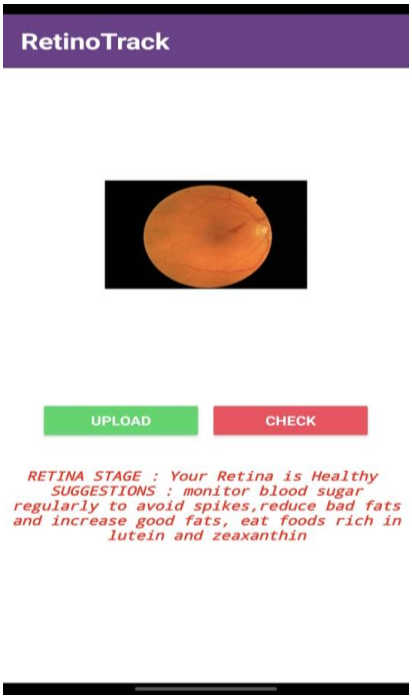


Fig. A.3 Healthy Stage

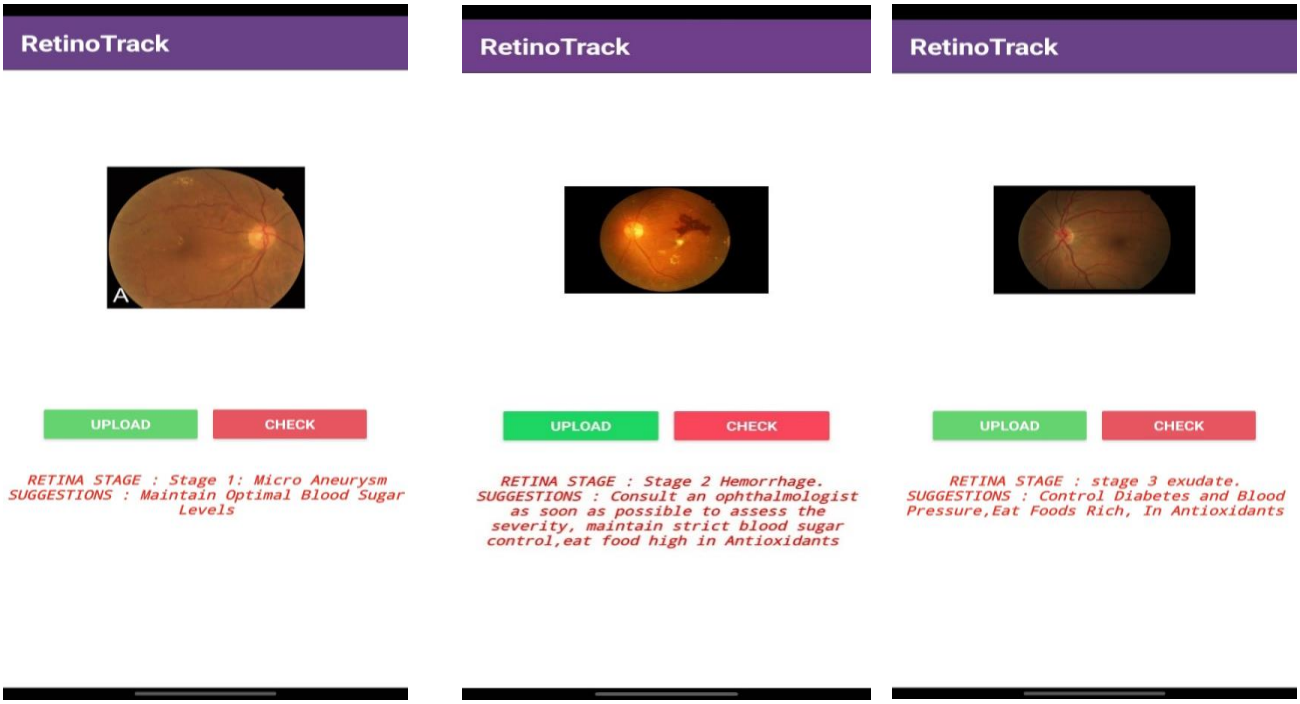


Fig. A.4 Different Stages of Diabetic Retinopathy

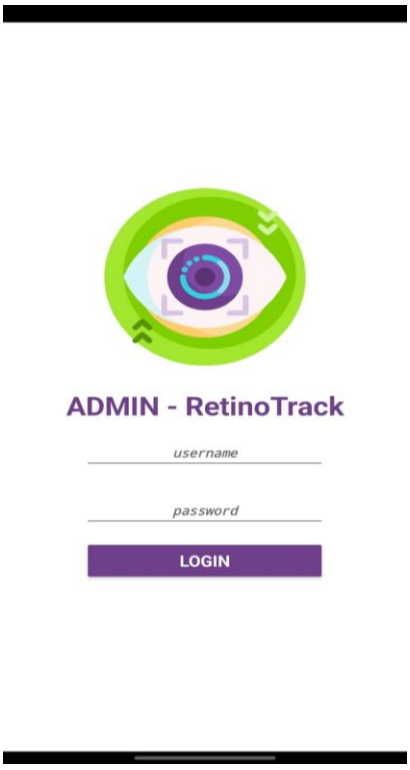


Fig. A.5 Admin's Application

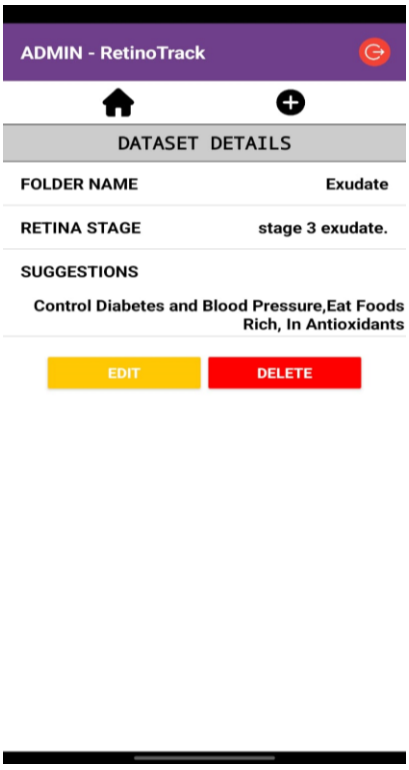


Fig. A.6 Admin's Home Page

APPENDIX B

install **libraries**

```
[ ] !pip install firebase==3.0.1
    !pip install python-jwt==4.0.0
    !pip install gcloud==0.18.3
    !pip install sseclient==0.0.27
    !pip install pycryptodome==3.17
    !pip install requests-toolbelt==0.10.1
    !pip install urllib3==1.26.14
```

Show hidden output

UPDATE FIREBASE CONFIG

```
[ ] firebase_config = {
      "apiKey": "AIzaSyCDYjf83lWXSAH7LmqAlcXNbiHDkZ6KEzE",
      "authDomain": "diabetic-retenopathy.firebaseapp.com",
      "databaseURL": "https://diabetic-retenopathy-default-rtdb.firebaseio.com",
      "storageBucket": "diabetic-retenopathy.firebasestorage.app"
    }


    childName = 'retino_track'
    childNameImg = 'DiabeticCOET2k25img'
```

```
[ ] firebase = Firebase(firebase_config)
    db2 = firebase.database()

    file_path = '/content/drive/MyDrive/Colab Notebooks/test_image/img.jpg'
    directory_path = os.path.dirname(file_path)
    if not os.path.exists(directory_path):
        os.makedirs(directory_path)
```

Library Installation and Firebase Configuration

```
import numpy as np
import matplotlib.pyplot as plt
# here we are working on Tensorflow version 2.1.0 so we need to write tensorflow.keras.
#keras is in built function in Tensorflow.
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Input, Dropout,Flatten, Conv2D
from tensorflow.keras.layers import BatchNormalization, Activation, MaxPooling2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.utils import plot_model
from IPython.display import SVG, Image
```

```
[ ] train_location = "/content/drive/MyDrive/Colab Notebooks/Diabetic Retenopathy/Final/New Dataset"
    test_location = "/content/drive/MyDrive/Colab Notebooks/Diabetic Retenopathy/Final/New Dataset"
    filepath = '/content/drive/MyDrive/Colab Notebooks/Diabetic Retenopathy/Model/F_model_VGG16.keras'
```

```
[ ] from tensorflow.keras.models import load_model
    Detection=load_model(filepath)
```

```
[ ] img_size=224
    batch_size=10
    num_class=4
```

```
[ ] # Complete Dataset images can be loaded using ImageDataGenerator function

    datagen_train=ImageDataGenerator(horizontal_flip=True)
    train_generator=datagen_train.flow_from_directory(train_location,target_size=(img_size,img_size),batch_size=batch_size,class_mode='categorical',shuffle=True)

    datagen_test=ImageDataGenerator(horizontal_flip=True)
    validation_generator=datagen_test.flow_from_directory(test_location,target_size=(img_size,img_size),batch_size=batch_size,class_mode='categorical',shuffle=True)
```

```
Found 129 images belonging to 4 classes.
Found 129 images belonging to 4 classes.
```

```
[ ] classes=train_generator.class_indices
    classes
```

```
{'Exudate': 0, 'Healthy': 1, 'Hemorrhage': 2, 'MA': 3}
```

```
category=[]
for i in classes:
          category.append(i)

category
```

```
['Exudate', 'Healthy', 'Hemorrhage', 'MA']
```

Model Training Using VGG16

import dataset

```
[ ] test_location = "/content/drive/MyDrive/Colab Notebooks/Diabetic Retenopathy/Final/New Dataset"
```

set parameters

```
[ ] img_size=224     #Vgg16 input layer size
    batch_size=10
    num_class=4      #Number of classification
```

load all images from directory

```
[ ] datagen_train=ImageDataGenerator(horizontal_flip=True)
    train_generator=datagen_train.flow_from_directory(test_location,
    target_size=(img_size,img_size),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True)
```

```
Found 129 images belonging to 4 classes.
```

live prediction

```
[ ] import time
    time.sleep(1)
    from IPython.display import clear_output
```

```
while True:
  uploadFlag = db.child(childName).child('uploadFlag').get().val()
  if uploadFlag == None:
    uploadFlag = '0'
  db2.child(childName).child('uploadFlag').set(uploadFlag)
  if uploadFlag == '1':
    clear_output()
    uploadFlag = '0'
    db.child(childName).child('uploadFlag').set(uploadFlag)
    db2.child(childName).child('uploadFlag').set(uploadFlag)
    db2.child(childName).child('resultFlag').set('0')
    time.sleep(2)
    storage.child(childName).child('img.jpg').download(file_path)
    with open(file_path, 'rb') as image_file:
      image_binary = image_file.read()
    image_base64 = base64.b64encode(image_binary).decode('utf-8')
    db2.child(childNameImg).child('photo').set(image_base64)
    #diseaseName = 'no disease'
    #img = mpimg.imread(file_path)
    test_img=image.load_img(file_path,target_size=(img_size,img_size))
    #test_img = cv2.resize(img, (img_size,img_size))  # Resize using cv2.resize

    result = image_prediction(test_img)
    db.child(childName).child('result').set({'result':result})
    db.child(childName).child('resultFlag').set('1')
    db2.child(childName).child('result').set({'result':result})
    db2.child(childName).child('resultFlag').set('1')
```

Importing Dataset and Live Prediction