CyberSight

An AI Powered Diabetic Retinopathy Detection System

Group Members

Ayesha Shamim (01-131222-011)

Muhammad Bilal Masood (01-131222-030)

Supervisor: Engr. Rafia Hassan

A Final Year Project submitted to the Department of Software Engineering, Faculty of Engineering Sciences, Bahria University, Islamabad in the partial fulfillment for the award of degree in Bachelor of Software Engineering

May 2026

THESIS COMPLETION CERTIFICATE

Student Name: Ayesha Shamim Enrolment No: 01-131222-011

Student Name: Muhammad Bilal Masood Enrolment No: 01-131222-030

Program of Study: Bachelor of Software Engineering

Project Title: CyberSight (An AI Powered Diabetic Retinopathy Detection System)

It is to certify that the above students' project has been completed to my satisfaction and my belief, it's that standard is appropriate for submission for evaluation. I have also conducted a plagiarism test of this thesis using HEC prescribed software and found a similarity index at _____ that is within the permissible limit set by the HEC. I have also found the thesis in a format recognized by the department.

Supervisor's Signature: _____

Date: _____ Name: _____

CERTIFICATE OF ORIGINALITY

This is certify that the intellectual contents of the project CyberSight are the product of my/our own work except, as cited properly and accurately in the acknowledgements and references, the material taken from such sources as research journals, books, internet, etc. solely to support, elaborate, compare, extend and/or implement the earlier work. Further, this work has not been submitted by me/us previously for any degree, nor it shall be submitted by me/us in the future for obtaining any degree from this University, or any other university or institution. The incorrectness of this information, if proved at any stage, shall authorities the University to cancel my/our degree.

Name of the Student: Ayesha Shamim

Signature: _____ Date: _____

Name of the Student: Muhammad Bilal Masood

Signature: _____ Date: _____

Abstract

CyberSight is an AI-powered clinical decision support system developed to assist ophthalmologists and general physicians in the early detection and severity grading of diabetic retinopathy using retinal fundus images. Diabetic retinopathy is one of the leading causes of blindness worldwide, especially among diabetic patients who do not receive timely screening and treatment. As the number of diabetes cases continues to rise globally, healthcare systems face increasing pressure to provide faster and more accessible retinal screening services. In many rural and under-resourced areas, the shortage of trained specialists makes regular eye examinations difficult, which often results in delayed diagnosis and permanent vision damage. CyberSight aims to address this challenge by providing an intelligent and accessible platform that supports doctors in identifying diabetic retinopathy at an earlier stage while ensuring that the final medical decision always remains under professional supervision.

The system is designed using a modern web-based architecture that combines a React frontend with a FastAPI backend for smooth communication and efficient request handling. The core AI component is based on a Vision Transformer (ViT) deep learning model trained to classify retinal fundus images into five diabetic retinopathy severity categories, ranging from no diabetic retinopathy to proliferative diabetic retinopathy. Before prediction, CyberSight performs several validation steps to improve reliability and reduce errors during inference. The uploaded image is checked for supported formats such as JPEG and PNG, image quality and resolution are verified, and a CLIP-based zero-shot validation mechanism confirms that the uploaded image is actually a retinal fundus image. These validation measures help prevent unrelated or poor-quality images from affecting prediction accuracy and improve the overall robustness of the system.

In addition to accurate prediction, CyberSight also focuses on transparency and explainability, which are important challenges in medical AI systems. Many deep learning models are often criticized for behaving like “black boxes” because doctors cannot easily understand how predictions are generated. To solve this issue, CyberSight integrates an occlusion-based explainable AI approach that generates visual heatmaps highlighting the image regions that most influenced the model’s decision. This allows doctors to better interpret and verify AI predictions, improving trust and usability in real clinical environments.

The platform also supports automatic diagnostic report generation that includes confidence scores, severity explanations, and medical disclaimers. These reports simplify communication between doctors and patients and make clinical documentation easier. Overall, CyberSight demonstrates how artificial intelligence can be integrated into healthcare systems as a supportive tool to improve early detection, enhance screening efficiency, and assist medical professionals in providing better patient care.

Dedication

Firstly, we dedicate this project to God Almighty, our creator, our strong pillar, our source of inspiration, wisdom, knowledge, and understanding. He has been the source of our strength throughout this FYP project and on His wings only have we soared.

Secondly, we would like to dedicate our project work to our beloved parents and respected teachers. Their constant support, encouragement, and prayers have been the backbone of our efforts and perseverance. We owe a special debt of gratitude to our esteemed supervisor, **Engr Rafia Hassan**, whose expert guidance, mentorship, and continuous motivation have been instrumental in shaping this work.

This achievement stands as a testament to the strength, patience, and belief of those who supported us unconditionally. We are truly grateful.

To our parents and our supervisor for their support

Acknowledgments

All praises and thanks to Almighty Allah, the Most Gracious and the Most Merciful, for providing us the strength, wisdom, and perseverance to successfully complete our Final Year Project titled CyberSight.

We would like to express our heartfelt gratitude to our supervisor, **Miss Rafia Hassan**, for her invaluable guidance, consistent encouragement, and continuous support throughout the duration of this project. Her expert advice and constructive feedback played a vital role in shaping the direction and quality of our work.

We are also deeply thankful to the Department of Software Engineering and our respected faculty members for creating an environment of academic support and innovation, and for providing us with the resources and knowledge necessary to complete this project.

Our sincere appreciation goes to our families for their unwavering support, patience, and prayers during challenging moments. Their belief in us gave us the confidence to keep moving forward.

Contents

List of Figures

List of Tables

Chapter # 1

Introduction

Chapter 1

Introduction

The swift expansion of Artificial Intelligence (AI) in healthcare has brought forth innovative possibilities to advance the diagnosis of diseases, medical image analysis, and clinical decision support. The use of AI-based systems to help healthcare professionals identify diseases more accurately and efficiently is on the rise. These advancements have been of special significance to ophthalmology among other medical disciplines, especially in automated diagnosis of retinal diseases through medical imaging.

Diabetic Retinopathy (DR) is one of the most critical retinal diseases in this respect. It is regarded as a severe complication of diabetes, and it is among the causes of blindness that are most common in the world. DR impacts the retina, the light-sensitive tissue of the eye situated at the back of the eye and vision. The prolonged high level of blood sugar starts to destroy the small blood vessels in the retina. With time, it may cause leakage, swelling, the proliferation of abnormal blood vessels as well as gradual loss of vision.

In case the disease is not diagnosed and treated during the initial stages, it can cause irreversible blindness. The International Diabetes Federation reports that over 537 million adults are already living with diabetes and many of them are in danger of getting retinal complications. A key problem of DR is that it progresses gradually and not necessarily with clear symptoms at the initial stages. The condition develops and progresses to a more severe level without the knowledge of many patients. Due to this, periodical fundus checkups and early diagnosis come in very handy in controlling the disease and avoiding the loss of sight.

1.1 Motivation

The motivation behind CyberSight is the problem of limited access to timely screening of Diabetic Retinopathy (DR) among the world population. With the rising cases of diabetes in the world, the demand of ophthalmologists has been on the rise. Nevertheless, the supply of experts has not been able to match this need. This loophole poses a critical bottleneck during diagnosis, and many patients fail to be screened promptly. This, in most instances, results in late detection, diagnosis at later stages and in some instances avoidable loss of vision. Under-resourced regions have an even more apparent issue, as the availability of healthcare facilities and qualified personnel is already low.

Most DR screening, at present, relies on experts viewing retinal images manually. Although this method is certain, it is time-consuming and can be costly. Consequently, this leads to a build-up of backlog cases in healthcare systems, thereby complicating the process of implementing early detection and making timely treatment difficult. This latency may have a direct effect on patient outcomes, particularly in illnesses such as DR which silently develop in the initial stages.

To address these issues, CyberSight will be developed as a web-based automated application that will make eye care more accessible. This should not be used to substitute experts, but to facilitate the

process of screening. The system can assist general practitioners to detect possible cases earlier by giving fast preliminary tests to lessen the workload of ophthalmologists. It can enhance the overall performance of the screening workflow, particularly in the high demand setting.
The project also utilizes state-of-the-art deep learning methods, especially Vision Transformers, and Explainable AI (XAI). This is not only aimed at getting right predictions but also to make predictions intelligible. Transparency is a value in a clinical setting; doctors must possess some degree of understanding of how the system is making decisions. CyberSight enhances trust and makes the system more realistic to the real world by incorporating explainability features.

Another important aspect of the system is its focus on safe and responsible use of AI. CyberSight is designed strictly as a decision-support tool, which means it assists doctors rather than replacing them. The final diagnosis always remains under the control of qualified medical professionals. This approach ensures that the system is used in a controlled and ethical manner, which is essential when dealing with healthcare applications.

1.2. Problem statement

The primary challenge with Diabetic Retinopathy is its asymptomatic nature in the early stages; many patients do not notice symptoms until the condition becomes vision threatening. While regular screening is critical, current clinical workflows face several bottlenecks:

- **Specialist Shortage:**
  Manual screening is time-consuming, and highly dependent on the availability of trained ophthalmologists, who are often scarce in under-resourced regions.
- **Input Vulnerability:**
  Many standard automated systems do not check that an uploaded image is a valid retinal fundus scan, and thus may make a misdiagnosis due to an out-of-distribution image.
- **The "Black Box" Problem:**
  Clinicians often hesitate to trust AI predictions that lack transparency regarding the spatial reasoning behind a diagnosis.
- **Hardware Constraints:**
  Many state-of-the-art AI models require expensive GPU hardware, making them inaccessible to smaller, local clinics.

1.3. Objectives

The main objectives of developing CyberSight are:

- **Clinical Decision Support System**
  To develop an AI-powered clinical decision support system capable of performing five-class diabetic retinopathy grading from retinal fundus images and providing clinically meaningful, calibrated outputs for doctor review.

- **Web-Based Screening Platform**

To design and implement a web-based platform for diabetic retinopathy screening that enables users to upload retinal images and receive automated analysis in an efficient and accessible manner.

- **Image Preprocessing and Validation**
  To ensure reliable model input by applying preprocessing and validation techniques, including image type and size checks along with CLIP-based fundus verification.
- **Explainability and Model Transparency**
  To improve trust and interpretability by generating visual explanations that help clinicians understand model predictions.
- **Secure Authentication and Access Control**
  To implement a secure authentication system using JWT-based sessions and role-based access control, enabling administrator-managed doctor accounts and restricted system access.
- **Report Generation and Clinical Support**
  To provide automated report generation and result-sharing features to support clinical documentation and streamline healthcare workflows.
- **Safety and Responsible AI Usage**
  To ensure safe usage of the system by incorporating strict validation safeguards and clear non-diagnostic disclaimers, positioning the AI as a decision-support tool rather than a replacement for medical professionals.

1.4. Main Contributions

The main contributions of the CyberSight project are as follows:

- **ViT-Based Inference Integration**
  An inference pipeline with a pretrained classifier architecture that matches the training notebook and a configurable model registry to enable disease extensions in the future.
- **CLIP-Based Fundus Validation Gate**
  Deployment of a CLIP based validation system, with OpenCLIP, implementing a probability threshold on retina specific prompts: to only process valid fundus images.
- **Efficient Explainable AI (XAI) Module**
  An occlusion-based explainability algorithm optimized to run on CPU resources was developed and implemented as part of the prediction service to produce heatmaps on requesting explanations.
- **Production-Ready API Design**
  Scalable backend API design, rate limiting, CORS setup, and error handling structure to ensure robust deployment.
- **Secure Identity and Administration System**
  Introduction of an authentication and authorization system based on Azure SQL, where the administrator has control over the management of doctor accounts, independent of the inference pipeline.

- **End-to-End User Workflow with Reporting**
  Creation of a React-based frontend workflow, which includes image upload, image analysis, display of the results, and the creation of printable reports with the help of the client-side PDF tools.

1.5. Report organization

The brief structure of our report is as follows:

Chapter:1
In Chapter 1, we provide a brief introduction of CyberSight, motivation, problem statement, objectives, contributions, and report organization.

Chapter:2
In Chapter 2, we present a literature-oriented discussion of medical fundus classification, transformer models for vision, explainability expectations, and limitations of prior screening automation.

Chapter:3
In Chapter 3, we explain about all the system requirements, use cases, functional requirement and nonfunctional requirement.

Chapter:4
In Chapter 4, we explain the overall design of the application. It includes class diagrams, sequence diagrams, ER diagrams, activity diagrams, and system architecture. The UI design and layout for both web and mobile versions are also discussed to show how users interact with the system.

Chapter:5
Chapter 5 provides a detailed explanation of the technologies, tools, and algorithms used for developing CyberSight.

Chapter:6
In this chapter, we describe the different phases of testing carried out to ensure system reliability. We elaborate on unit testing, integration testing, and system testing, along with test cases that cover every feature of the application, ensuring all components function as expected.

Chapter:7
Chapter 7 offers a conclusion to the entire report. It summarizes the work done throughout the project, outlines the challenges faced, and provides recommendations for future improvements and potential enhancements in CyberSight

Chapter # 2
Background study/Literature Review

Chapter 2

Literature Review

2.1 Introduction

Diabetic Retinopathy (DR) is a progressive eye disease that is brought about by a long-term diabetes condition, where the blood vessels in the retina are damaged and it may ultimately result in blindness. Retinal fundus imaging is crucial in the prevention of loss of vision early. Nevertheless, the conventional screening techniques are based on manual screening by ophthalmologists, which is time-consuming, expensive, and not always accessible in rural or poorly equipped communities.

As the field of Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) progress, automated systems have been created to aid in the detection and classification of DR. These systems are designed to enhance the accuracy of diagnosis, lessen workload and allow scalable screening solutions. The chapter will overview the traditional ML methods, the deep learning methods, the transformer-based methods and the explainable AI methods as well as analysis of existing systems.

2.2 Traditional methods for DR Detection

Early DR detection systems relied on classical image processing and rule-based techniques. These methods focused on identifying specific retinal abnormalities such as:

- Microaneurysms
- Hemorrhages
- Exudates
- Blood vessel irregularities

Common techniques included thresholding, edge detection, morphological operations, and segmentation. For example, early studies used pixel classification and handcrafted feature extraction to detect lesions in retinal images.

Limitations of Traditional Approaches

Despite initial success, traditional approaches suffer from several limitations:

- Heavy reliance on handcrafted features
- Poor generalization across datasets
- Sensitivity to image quality and illumination
- Inability to capture complex patterns

These challenges limited their effectiveness in real-world clinical applications, leading to the adoption of machine learning techniques.

2.3 Machine Learning Techniques for DR Detection

Machine Learning introduced a more structured approach to DR detection by enabling systems to learn patterns from data. A typical ML pipeline includes:
Image preprocessing

1. Image preprocessing
2. Feature extraction
3. Feature selection
4. Classification

Common ML Algorithms

- Support Vector Machines (SVM)
- K-Nearest Neighbour(KNN)
- Random Forest
- Decision Trees

These methods rely on handcrafted features such as texture descriptors (e.g., GLCM), shape features, and statistical measures.

Advantages of ML Techniques

- Improved accuracy compared to traditional methods
- Ability to learn patterns from data
- Lower computational cost compared to deep learning

Limitations of ML Techniques

However, ML-based approaches still suffer from several drawbacks:

- Dependence on manual feature engineering
- Limited capability to capture complex image patterns
- Performance degradation on large and diverse datasets
- Time-consuming feature extraction process

These limitations led to the adoption of deep learning techniques, which automate feature extraction.

2.4 Deep Learning Approaches for DR Detection

Deep Learning, particularly **Convolutional Neural Networks (CNNs)**, has revolutionized medical image analysis by enabling automatic feature extraction directly from raw images.

A landmark study by Gulshan et al. (2016) demonstrated that deep CNN models can achieve performance comparable to ophthalmologists in DR detection. Similarly, Abràmoff et al. developed automated systems with high sensitivity and specificity in clinical settings.

Common CNN architectures used include:

- ResNet
- VGGNet
- EfficientNet

Advantages of CNNs:

- Automatic feature extraction
- High accuracy
- Robust performance across datasets

Limitations:

- Require large, labelled datasets
- Computationally intensive
- Lack interpretability ("black box" problem)

While CNNs significantly improved detection accuracy, their lack of transparency remains a major barrier in clinical adoption.

2.5 Vision Transformers in Medical Imaging

Vision Transformers (ViTs) represent a recent advancement in deep learning, applying transformer-based architectures to image analysis. Unlike CNNs, which focus on local features, ViTs process images as sequences of patches and use self-attention mechanisms to capture global relationships.
Studies have shown that Vision Transformers:

- Capturing long-range dependencies in retinal images
- Improving detection of subtle abnormalities
- Providing competitive performance compared to CNNs

Advantages of Vision Transformers

- Better global feature representation
- Improved performance on complex datasets
- Scalable architecture

Limitations

- Requires large datasets for effective training
- High computational cost
- Complex architecture compared to CNNs

CyberSight leverages Vision Transformer-based models to improve classification accuracy and capture global retinal patterns.

2.6 Explainable AI in Healthcare

One of the major challenges in deploying AI systems in healthcare is the **lack of transparency**.
Explainable AI (XAI) techniques address this issue by highlighting important regions of input images that influence predictions.
A widely used method is:

- **Grad-CAM (Gradient-weighted Class Activation Mapping)**

Grad-CAM generates heatmaps that visually indicate regions contributing to the model's decision.

Importance in DR Detection

- Heatmaps highlight affected retinal regions
- Improve clinician trust
- Support decision-making

Other XAI techniques include:

- LIME (Local Interpretable Model-Agnostic Explanations)

- SHAP (SHapley Additive exPlanations)

However, Grad-CAM is preferred in image-based systems due to its simplicity and effectiveness.

2.7 Existing Systems and Their Limitations

Several diabetic retinopathy detection systems such as IDx-DR, Google DeepMind retinal AI, EyePACS-based models, and other CNN-based screening platforms focus primarily on automated analysis of retinal fundus images to detect the presence or severity of diabetic retinopathy. These systems use deep learning techniques to achieve high diagnostic accuracy and assist ophthalmologists in clinical decision-making. While these solutions are highly effective, many of them operate as black-box models and lacks sufficient explainability, making it difficult for clinicians to fully trust predictions. Additionally, most of these systems require high computational resources and are often deployed in controlled clinical environments, limiting their accessibility in rural or low-resource settings.

Furthermore, several embedded AI-based diagnostic systems developed in countries like China integrates retinal imaging devices with on-device inference capabilities. These systems enable real-time detection without relying on cloud infrastructure, which is beneficial in remote areas. However, they are heavily dependent on specialized hardware such as high-resolution fundus cameras and GPU-enabled embedded boards. This reliance increases cost, reduces flexibility, and make system scalability and maintenance more challenging. In contrast, many cloud-based systems require stable internet connectivity and large datasets for optimal performance, which may not always be available.

Additionally, much diagnostic research-based approaches tends to focus on binary classification of DR which provide a simpler but less informative model that enable quick screening of patients but does not gives a detailed diagnosis that would help plan therapy. On the other hand, classification algorithms that includes multi-classes provides a detailed staging of the disease but are more complex and computationally demanding.

CyberSight addresses these limitations by:

- Providing a web-based platform for easy access without dependency on specialized hardware
- Integrating deep learning models for accurate DR classification
- Supporting both detection and severity grading of diabetic retinopathy
- Incorporating explainable AI (heatmaps) to improve transparency and clinician trust Enabling automated report generation and sharing for efficient clinical workflows

Unlike many existing systems, CyberSight are designed to be accessible, scalable, and interpretable, offering a balanced solution that combine accuracy, usability, and reduced hardware dependency, making it suitable for both urban and resource-constrained healthcare environments.

2.8 Literature review:

▪ **Saima Akhtar et al. [1]**

This research presents a diabetic retinopathy severity grading system using transfer learning techniques. The authors utilized pre-trained convolutional neural networks such as VGG16, ResNet, and Inception for feature extraction and classification of retinal fundus images. Their approach significantly improved classification accuracy by leveraging pretrained weights and fine-tuning on DR datasets. The study demonstrated that transfer learning reduces training time while maintaining high performance. However, the system lacked interpretability, making it difficult for clinicians to understand the reasoning behind predictions.

▪ **Kursheed Aurangzeb et al. [2]**

This paper discusses the design of an AI-powered diagnostic system for glaucoma and diabetic retinopathy. This work describes the design of an image processing system using image pre-processing, feature extraction, and classification based on deep learning methods. The main strength of the designed system is its scalability and modularity. It should be noted that the designed algorithm is computationally heavy, and there is no explainable mechanism available in the model

▪ **M. Asif et al. [3]**

This paper provides an extensive insight into the timely diagnosis of diabetic retinopathy using both traditional and AI-driven approaches. It compares classical machine learning methods with deep learning techniques and highlights the advantages of AI in improving early detection rates. The study emphasizes that deep learning models outperform traditional approaches in accuracy and automation. However, challenges such as data imbalance, lack of interpretability, and dependency on large datasets remain unresolved.

▪ **Varun Gulshan et al. [4]**

This landmark research developed a deep learning model trained on a large dataset of retinal images, achieving performance comparable to ophthalmologists. The system demonstrates the effectiveness of CNN-based models in DR detection. Despite high accuracy, the model behaves as a black box, limiting interpretability in clinical environments.

▪ **D. Saproo et al. [5]**

This research proposed a deep learning-based binary classification system that classifies retinal images into **DR and No DR categories** using transfer learning. The study used multiple pretrained models and showed that binary classification is effective for early screening and large-scale deployment. However, it does not provide severity grading, which is essential for clinical decision-making

▪ **S. Buda et al. [6]**

This study investigated class imbalance in deep learning-based image classification. The authors showed that deep neural networks are highly sensitive to imbalanced datasets. They proposed data augmentation and weight loss functions to improve performance. These techniques are particularly important in DR detection where minority classes (severe DR) are underrepresented.

▪ **Y. Yuan et al. [7]**

This research explored Vision Transformers for medical image classification, demonstrating their ability to outperform CNNs in capturing global image features. The study highlights that ViT models are particularly effective for complex retinal patterns in DR detection. However, performance is highly dependent on dataset size and quality.

▪ **S. Gargeya and T. Leng [8]**

This research introduced a deep learning-based automated DR detection system capable of identifying referable diabetic retinopathy. The model achieved high sensitivity and specificity and is suitable for binary classification tasks (DR / No DR). The study highlights the potential of AI for large-scale screening. However, the lack of explainability and dependency on high-quality data remain limitations.

Chapter # 3

System Requirements

System Requirements

3.1 System Level Use Case Diagram

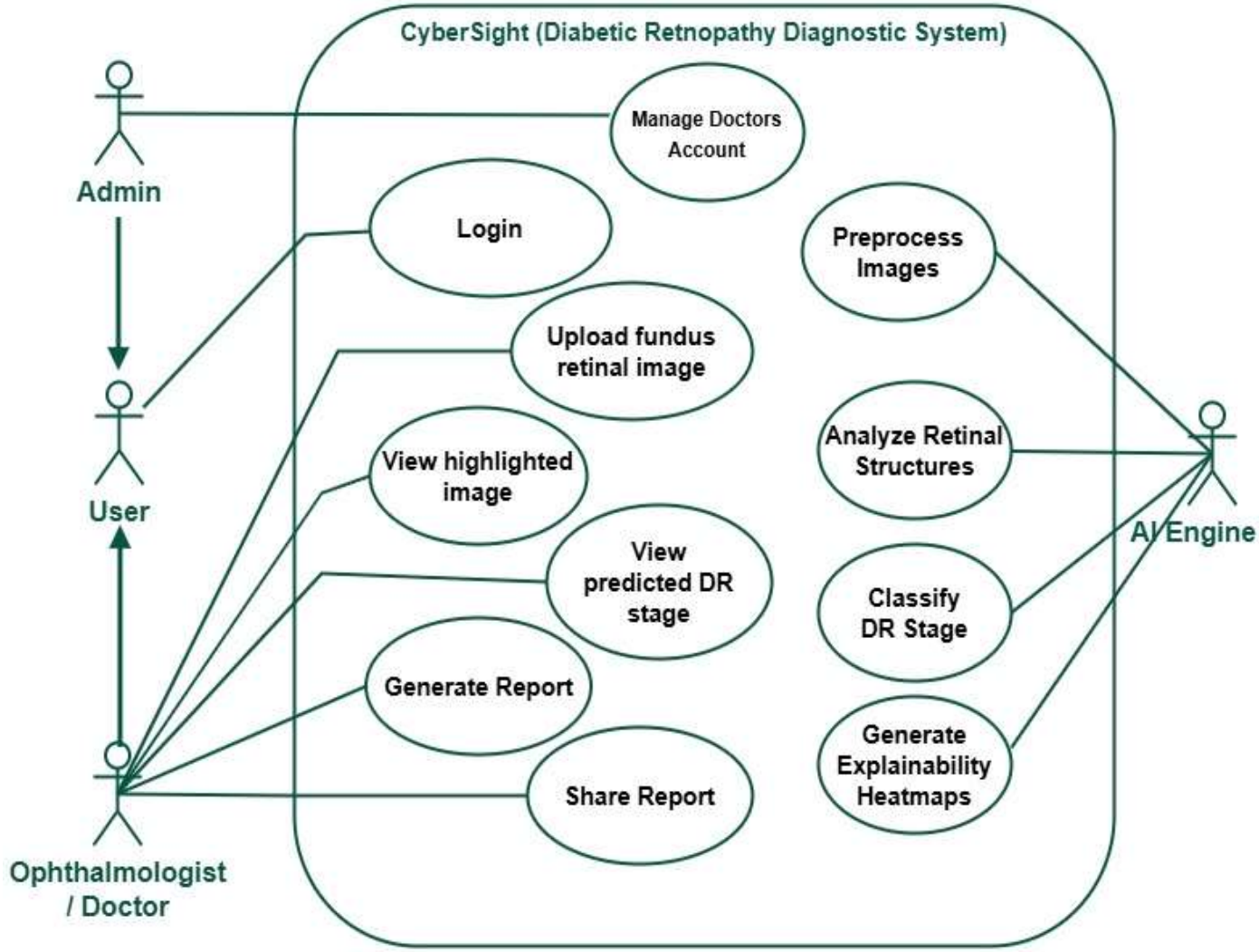

Figure 1: System Level Use Case Diagram

Description:

The use-case diagram illustrates the interaction between the Admin, Ophthalmologist/Doctor, and the AI Engine in the CyberSight diabetic retinopathy diagnostic system. The admin is responsible for managing doctor accounts and ensuring authorized access to the platform. Doctors log in to the system, upload retinal fundus images, and review AI-generated diagnostic results. After image upload, the AI Engine performs preprocessing, disease stage classification, and generates explainability heatmaps to support medical interpretation. Based on these results, doctors can analyze the severity of diabetic retinopathy, generate diagnostic reports, and share them with patients for further treatment and consultation.

3.2 Detailed Use cases:

3.2.1 Login

Use Case ID:	UC-01	
Use Case Name:	User Login	
Actor(s):	**Doctor/Admin**	
Pre-Conditions:	User has valid account credentials	
Priority:	High	
Basic Flow:	1. Enter Registered Email 2. Enter Password 3. Press Login	
Actor Actions		**System Response**
1	User enters username and pas Users click on "Login" button	System checks user's provided credentials. If user's cre authenticated and home screen is displayed.
Alternative Course of Action		
Actor Action		**System Response**
	If credentials are invalid, then Return to login screen	If user's credential is invalid an error message is disp shown/redisplayed.

Table 1: Use Case Description of User Login

3.2.2 Forgot Password

Use Case ID:	**UC-02**	
Use Case Name:	**Forgot Password**	
Actor(s):	Doctor	
Pre-Conditions:	Email must be registered	
Priority:	Medium	
Basic Flow:	1. Click "Forgot Password" 2. Enter registered email 3. Submit Email is sent on registered email for reset	
Actor Actions		**System Response**

1	The user enters the correct email and incorrect password.	Displays invalid username or password error message.
2	The user resets the password	Reset link sent if email exists
Alternative Course of Action		
Actor Action		**System Response**
1	User enters unregistered email	An error message will be displayed that this account doesn't exist.

Table 2: Use Case Description of Forgot Password

3.2.3 Logout

Use Case ID:	**UC-03**	
Use Case Name:	**Logout**	
Actor(s):	**Doctor/Admin**	
Pre-Conditions:	1. The user is logged in 2. The user no longer wants to be logged in.	
Priority:	Medium	
Basic Flow:	1. The user is done using the application 2. The user clicks on the logout button 3. The system logs the user out.	
Actor Actions		**System Response**
1	Users click on the logout button	The Login screen is display
Alternative Course of Action		
Actor Action		**System Response**
	NA	NA

Table 3: Use Case Description of User Logout

3.2.4 Upload Fundus Image

Use Case ID:	UC-04
Use Case Name:	Upload Fundus Retinal Image
Actor(s):	**Doctor**
Pre-Conditions:	User must be Logged In.

Priority:	High	
Basic Flow:	1. The user press upload Button 2. Select Fundus Image	
Actor Actions		**System Response**
1	User clicks the "Upload Image" button.	System opens file upload dialog.
2	User selects a JPG/PNG fundus image.	System validates file format and resolution.
3	User confirms upload.	System uploads and displays image preview. System uploads, preprocesses, and displays image preview
Alternative Course of Action		
Actor Action		**System Response**
	Invalid file format	System shows "Unsupported file type."
	Low resolution	System shows "Image quality too low for analysis."
	Upload fails	System displays "Upload failed. Try again."

Table 4: Use Case Description of Upload Fundus Image

3.2.5 View Predicted DR Stage

Use Case ID:	UC-05	
Use Case Name:	View Predicted Diabetic Retinopathy Stage	
Actor(s):	Doctor	
Pre-Conditions:	Image must be successfully analyzed by AI model. User must be authenticated.	
Priority:	High	
Actor Actions		**System Response**
1	User navigates to the "Results" page.	System sends the preprocessed image to the trained AI model.
		AI model returns the predicted DR class (No DR → PDR).
		System displays the predicted DR stage along with confidence score.
Alternative Course of Action		
Actor Action		**System Response**
	Model cannot process image	System shows "Prediction unavailable."
	Timeout	System shows "Model response delayed."

Table 5: Use Case Description of View Predicted DR Stage

3.2.6 View Highlighted Image

Use Case ID:	UC-06
Use Case Name:	View highlighted image
Actor(s):	Doctor
Pre-Conditions:	AI must have completed DR prediction. Image must be valid and stored.
Priority:	Medium
Basic Flow:	

Actor Actions		System Response
1	User clicks “View Heatmap.”	**S**ystem generates heatmap.
		System overlays heatmap on the original image.
		System displays visual explanation.
Alternative Course of Action		
Actor Action		**System Response**
		If heatmap generation fails, System shows “Heatmap unavailable**.”** If model layers incompatible, System shows “Explainability error.”

Table 6: Use Case Description of View Highlighted Image

3.2.7 Generate Diagnostic Report

Use Case ID:	UC-07
Use Case Name:	Generate Report
Actor(s):	Doctor
Pre-Conditions:	User must be authenticated first. Image must be Analyzed and Heatmap must be generated
Priority:	High

Actor Actions		**System Response**
1	User clicks the “Generate Report” button.	System compiles all required data (image, prediction, heatmap).
		System generates a structured diagnostic report.
		System shows a “Download Report” button.
Alternative Course of Action		
Actor Action		**System Response**
		If required data missing, system shows “Unable to generate report. If PDF generation fails, system shows “Report generation error.”

Table 7: Use Case Description of Generate Diagnostic Report

3.2.8 Share Diagnostic Report

Use Case ID:		UC-08
Use Case Name:		Share Diagnostic Report
Actor(s):		Doctor
Pre-Conditions:		Diagnostic reports must already be generated. User must be authenticated.
Priority:		Medium
Actor Actions		**System Response**
1	User clicks "Share Report."	System opens sharing options (Email/Download Link).
2	User enters recipient details	System sends reports and confirms successful delivery.
Alternative Course of Action		
Actor Action		**System Response**
	No report exists	System shows "No report available to share."
	Network error	System shows "Failed to send report, try again."

Table 8: Use Case Description of Share Diagnostic Repo

3.2.9 Manage Doctors Account

Use Case ID:	UC-09	
Use Case Name:	Manage Doctors Accounts	
Actor(s):	System Administrator	
Pre-Conditions:	Admin must be authenticated.	
Priority:	Medium	
Basic Flow:		
Actor Actions		**System Response**
1	Admin opens "User Management" panel.	System displays list of registered doctors**.**
2	Admin selects "Add / Edit / Delete Account."	System displays required forms.
3	Admin submits updated data.	System saves changes and displays success message.
Alternative Course of Action		
Actor Action		**System Response**
	Admin searches user account which does not exist.	System generates message of "No user found".

Table 9: Use Case Description of Manage Doctors Account

3.3 Functional Requirements:

3.3.1. Doctor account provisioning

Identifier	FR1
Title	Doctor account provisioning
Requirement	An administrator shall be able to register a new doctor account by providing first name, last name, email, and role (e.g. Doctor). The system shall persist the account in the database and may issue a one-time set-password flow so the clinician can choose a password.
Source	Web Application (Admin Dashboard)
Rationale	Ensures only trusted administrators create clinician identities.
Restrictions and Risk	All required fields must be present and valid; duplicate emails must be rejected with a clear error. administrator credentials must be strongly protected because they gate all provisioning.
Dependencies	Azure SQL database configured, valid administrator JWT, SMTP for invite emails.
Priority	High

Table 10: Functional Requirement of Doctor Account Provisioning

3.3.2. Complete password registration (set password)

Identifier	FR2
Title	Complete password registration (set password)
Requirement	A doctor who has been provisioned without a password shall be able to complete registration by opening the set-password link, validating the one-time token, and submitting a new password that meets minimum length rules.
Source	Web Application

Rationale	
Restrictions and Risk	Token must be valid and unexpired; weak or short passwords rejected.
Dependencies	Valid set-password token stored server-side; link generation when emailing
Priority	High

Table 11: Functional Requirement of Complete password registration (set password)

3.3.3. Authentication (login)

Identifier	FR3
Title	Authentication (login)
Requirement	A registered user (doctor or administrator) shall be able to sign in by providing **email** and **password**.
Source	Web Application
Rationale	Secure access to personalized features
Restrictions and Risk	Email must be verified; spam filters may delay link delivery
Dependencies	User Must be Registered
Priority	High

Table 12: Functional Requirement of Authentication (login)

3.3.4. Forgot password

Identifier	FR4
Title	Forgot password
Requirement	An active doctor shall be able to request a new set-password link by submitting an email and the system shall return a link for setting a password.
Source	Web Application
Rationale	Recovery path when password is lost
Restriction and Risks	Inactive accounts must not receive a usable recovery
Dependencies	Check weather doctor exists or not
Priority	Medium

Table 13: Functional Requirement of Forgot Password

3.3.5 Logout

Identifier	FR5
Title	**Logout**
Requirement	A signed-in user can log out, which clears their session so the next person using the device can't access their account.
Source	Web Application
Rationale	Important for security on shared computers
Restrictions and Risk	The login token technically stays valid until it expires, a proper revocation system would be needed for stricter security.
Dependencies	N/A
Priority	Medium

Table 14: Functional Requirement of Logout

3.3.6 Manage doctors (view, update, block, delete)

Identifier	FR6
Title	Manage doctors (view, update, block, delete)
Requirement	An administrator shall be able to list doctors, fetch one record, update name/email/active flag, toggle block, and delete a doctor.
Source	Web Application
Rationale	Keeps the list of users accurate and allows the admin to suspend or remove access when needed.
Restrictions and Risk	Duplicate emails on update are rejected. Deletions should ask for confirmation to avoid accidents.
Dependencies	FR3
Priority	High

Table 15: Functional Requirement of Manage Doctors

3.3.7 Upload Fundus Image:

Identifier	FR7
Title	Upload fundus image
Requirement	The doctor selects or drags in a JPEG or PNG image of the patient's eye. A preview is shown before submitting.
Source	Web Application (Doctor Dashboard)
Rationale	This is the starting point for all AI analysis
Restrictions and Risk	Only JPEG and PNG are accepted. The doctor must stay on the page, navigating away may lose the uploaded file.
Dependencies	FR3
Priority	High

Table 16: Functional Requirement of Upload Fundus Image

3.3.8 Diabetic retinopathy classification

Identifier	FR8
Title	Diabetic retinopathy classification
Requirement	The system runs the processed image through a Vision Transformer AI model and returns: a grade (0–4), confidence percentage, per-grade probabilities, and a severity label.
Source	Backend
Rationale	Core DR grading output for clinical decision support
Restrictions and Risk	If the AI model file is missing, all analysis fails.
Dependencies	Image should be properly preprocessed
Priority	Medium

Table 17: Functional Requirement of DR classification

3.3.9. Explainability:

Identifier	FR9
Title	Explainability
Requirement	the system generates a visual heatmap showing which areas of the eye photo had the most influence on the AI's decision. This is included as an image in the results.
Source	Backend
Rationale	Shows which regions most influenced the predicted class.
Restrictions and Risk	Generating the heatmap takes extra time. If it fails, a placeholder is shown instead.
Dependencies	FR8
Priority	High

Table 18: Functional Requirement of Explainability

3.3.10. View analysis result

Identifier	FR10
Title	View analysis result
Requirement	After analysis, the doctor sees all results on screen: grade, confidence, interpretation, probability breakdown, and the heatmap.
Source	Web Application
Rationale	Presents the AI's output in a clear, clinical layout for the doctor.
Restrictions and Risk	If the page is refreshed, the results may be lost, and the doctor would need to re-upload.
Dependencies	FR8, FR9
Priority	High

Table 19: Functional Requirement of view analytics result

3.3.11. Generate diagnostic report

Identifier	FR11
Title	Generate diagnostic report
Requirement	The doctor can generate and download a PDF version of the analysis results from the report page.
Source	Web Application
Rationale	Produces a portable artifact for records or external distribution.
Restrictions and Risk	PDF generation happens in the browser, so it may be slow on older devices.
Dependencies	FR10
Priority	Medium

Table 20: Functional Requirement of Generating diagnostic report

3.4. Interface Requirements

3.4.1. User Interfaces

React.js-Web application was used to make the front end of the Diabetic Retinopathy Detection System. This will make sure the system is responsive and is easy to use. The user interface will follow rules for graphical user interfaces. These rules include giving the user feedback having a consistent layout and making it easy to navigate. Important actions like Analyze, Export Report and Manage Doctors will be easy to find on the screens.

- UI-1: The systems screens will be designed to be user-friendly and friendly to doctors and administrators who must adhere to brief workflows. For example, doctors will have to login, upload a file, analyze it, review it and then get a PDF report. Administrators will be required to log in and administer doctor accounts. The labels will be all in line with the screening terms.
- UI-2: Once the user logs in the system will display a dashboard which is role specific. Physicians will be capable of uploading and analyzing files on their dashboard. The administrators will have an option of controlling accounts of doctors in their dashboard. The systems logo, email of the users and a Disconnect option will appear on desktop computers or a similar menu on other devices.
- UI-3: The web pages will allow the user to navigate without having to reload the page. This will make the system feel smooth and easy to use. The system will retain the context until current user refreshes the page or scans.
- UI-4: The Analyze button will be the action on the upload screen. The report screen will be simple to locate the Export Report and PDF Download options.
- UI-5: The screen will display a preview of the fundus image, the estimated grade of Diabetic Retinopathy, a score of confidence, a clinical interpretation and an optional heatmap. The user will also be able to navigate to start a scan or access the report.
- UI-6: All forms, including the login and admin forms, will show validation messages to help the user.
- UI-7: The layout will be responsive to the size of the screens to ensure that the system is completely responsive. Protected Route and AUTH Context mechanisms will also redirect users that are not authenticated or lack the role to the appropriate page. The Diabetic Retinopathy Detection System will ensure that there are specific areas in the system which can be accessed by the authorized users.

3.4.2. Interfaces

CyberSight is created as a web-based application, which allows access from any device connected to the Internet such as laptop, tablet, and smartphone. It requires no special

hardware. CyberSight is created as a web-based application, meaning that it allows one to access the system using a simple web browser on various types of devices such as a laptop, tablet, or even smartphone as long as one has access to the Internet. The key benefit of such a solution is that one will not need any special hardware for its work.

3.4.3. Software Interfaces

Software tools and technology	Version	Rationale
Visual Studio Code	1.115	Primary IDE for React and Python development
React.js	19.2.5	Front-end UI Framework
AzureDB	-	SQL Database
Python	3.14.3	Backend Logic

Table 21: Software Interfaces

3.4.4. Communications Interfaces

The Diabetic Retinopathy Detection System comes with a communication interface which is based on standard web protocols so that the communication between the client and the server is reliable, safe and efficient.

- **HTTP (RESTful APIs):**
  RESTful HTTP APIs deal with all client-server communication. Login and API calls relating to the administration are done using JSON and multiform data is used when uploading fundus images along prediction routes. The API returns structured JSON response payloads with the fields of DR grade, class probabilities, clinical interpretation, disclaimer and explainability.
- **Ssynchronous Request–Response Model:**
  The clinician workflow follows a simple request-response model which involves upload, analyze and display the results. Long Extended response times may occur during CPU-based occlusion explainability processing.
- **AI Features:**
  The AI features of the system are diabetic retinopathy grading with a Vision Transformer model, validation of retinal fundus images with CLIP, and explainability with occlusions. These features are implemented within the Python FastAPI backend or delegated to a separate model HTTP service when enabled.

- **Transport & Access Control:**
  All communication between clients and servers should be used by HTTPS to guarantee the security of data. It has API calls at the admin level, which must be sent with a JWT token in the Authorization header after successful login.

3.5. Database Requirements

CyberSight system is managed by Microsoft Azure SQL Database to store its data. This is the database on which the system stores and organizes information. The system is linked to the database via pyodbc and SQL server ODBC Driver 18.

All the work with the database is done by the FastAPI backend. This means the front end and the database are separate which makes the system more secure. The React client does not connect to the database directly.
The database is mainly used for managing users, including who they're, how they log in what they are allowed to do and how doctors are added to the system. It stores usernames, passwords, what roles and the status of their accounts.
The database does not store images or the results of the AI predictions. These are processed in the AI pipeline. Either stored in memory for a short time or sent straight to the user.
In this manner the system does not hold much medical information that would make user data private and secure. CyberSight system and the Microsoft Azure SQL database are collaborative in ensuring that this occurs.Clinical information including fundus images and AI prediction outputs are not continuously updated in the relational database. These are executed in the AI pipeline in real-time and are either stored in memory temporarily or sent to the user as a response to the system. This design guarantees that there is little storage of sensitive medical information and compliance with privacy and security.

3.6. Non-Functional Requirements

3.6.1. Performance Requirements

CyberSight must be performed in hospitals. This implies that it must be able to run images quickly and generate AI-based predictions. This is significant because the system would react promptly and operate effectively when there are numerous users in it now.

- **Response Time:**

  Under normal operating conditions, the delay in a result showing when a button is pressed on the screen should not take more than 5 seconds.

- **Model Inference Time:**
  The AI model should be able to classify the fundus image and return the DR stage within 5 seconds**,** even during peak usage.
- **Image Processing Time:**

No more than 5 seconds per image should be used in image pre-processing operations. Image resizing, normalization and applying CLIP are included into image preprocessing time.

3.6.2. Safety Requirements

In a bid to make sure that CyberSight is safe to use in a clinical environment, then one must make sure that there is no possibility of any harm to the patients. To this end, it is crucial to have safety-related requirements that will make sure that AI predictions will not harm the patients in any way.

- **Patient Data Protection**
  The system must adhere to all the HIPAA and other data protection policies in healthcare.
- **Clinical Safety Compliance**
  The safety declaimer that AI prognoses are not substitutes to professional medical diagnoses should be included. The system should meet the healthcare software safety standards.
- **Error Prevention**
  Corrupted or/and low-quality images should be automatically identified and not allowed to be analyzed.

3.6.3 Security Requirements

These requirements ensure the secure work of the CyberSight system. It should protect medical data that flows through the system, in such a way that only authorized personnel can access the data. It should also be safeguarded against cyber-attacks. To do that, certain security practices have to be included to provide safety of medical data.

- **Role-Based Access Control (RBAC):**
  Access must be restricted based on user role:
  - Doctor: view/upload images, see predictions
  - Admin: managing users, view logs, configure system
  - Unauthorized actions must be blocked.
- **Password Security:**
  The system will have tough password policies:
  - Minimum 8 characters
  - Should contain uppercase, lower, number and special characters.
  - Passwords must be hashed prior to storage.
- **Session Security:**
  The system must have secure session management.
  The session tokens should expire after being inactive.
  Protected resources can only be accessed using authenticated sessions.

3.6.4. Software Quality Attributes

The attributes of the software quality are critical to making sure that CyberSight is easy to use, maintain, adapt and reliable in its life cycle. These attributes inform the behavior of the system in circumstances, its ability to evolve easily and the ease of interaction with it by the users. They make sure that the platform provides a high-quality experience whilst being adaptable to enhancement.

Usability:
The system must be simple to use to doctors who have simple computer knowledge. Novices should be capable of posting photos and seeing predictions with instructions. The system must also be available.

Metrics:

- **Task Completion Efficiency:**
  Users ought to be able to perform fundamental tasks such as posting pictures and seeing predictions fast and conveniently.
- **User Success Rate:**
  Workflow should be completed with no error.

Maintainability:
The system should be made up of parts with clear code explanations and follow coding standards.

Metrics:

- **Low Coupling Between Modules:**
  Components should be independent enough to ensure they can update separately.
- **Ease of Change:**
  Small improvements or corrections must be made with a low level of development.

Adaptability:
Improvements such as the addition of new diseases or AI models should be supported by the system with minimal changes.

Metrics:

- **High Code Reusability:**
  Existing components and logic should be usable when introducing new features.
- **Minimal Architectural Adjustments:**
  To add a new feature there is no need to reconstruct the architecture again.

Portability:
The system will operate on current web browsers and devices.

Metrics:

- **Cross-Platform Compatibility:**
  Core functionality should behave consistently across popular browsers and devices.

- **Acceptable Load Performance:**
  The web application would be loaded and responsive to the common hardware of the user.

Robustness:
The system must be able to deal with invalid inputs such as poor-quality images or incompatible formats. It ought to avoid crashes, save user information and display feedback.

Metrics:

- **Effective Error Handling:**
  Most invalid or unexpected inputs must result in meaningful error messages without causing a breakage.
- **Fast Recovery:**
  Once the system is faced with invalid inputs or operational problems, it should quickly resume normal operations.

3.7. Project Feasibility

Technical Feasibility:

- **Modular architecture: Web-based:**

  CyberSight is a web app, using React on the frontend and FastAPI on the backend. The project structure is organized in a modular fashion with UI, authentication, and the service component of the administrator and the AI inferencing component having their own layers. This helps with developing, scaling, and deploying solutions.

- **Use of mature and reliable technologies:**

  It is built on proven technologies such as React, Tailwind CSS, FastAPI, and Python-based machine learning systems. The technologies are broadly supported, well-documented and can be developed fast within a final year project schedule, minimizing the risk of implementation.

- **Risk considerations and mitigation:**

  The use cases also have certain possible risks that should be addressed and removed, like poor or inaccurate image inputs. This problem can be mitigated by validating the input files with respect to file format, image quality, size, etc., before running inferencing. An authentication-based methodology with the right role-based access controls is used to mitigate security risks. Others such as poor performance on mixed images and delays in explainability features are noted as possible, though possible.

Legal and Ethical Feasibility:

- **Decision support System:**

  CyberSight is a clinical decision support system and not the alternative to the opinion of the physician. All reports are preceded by a notice saying that these results have been produced by AI and require verification by certified healthcare practitioners before any action is taken.

- **Data privacy and minimization:**

  The system merely stores the valuable data on user accounts such as the information on what each user is allowed to do and the information on their logins. It does not store medical information, such as pictures of the eye and test results in the database for a long time. When this medical information is being processed, it is only utilized during that process. This assists in the privacy of user information and's a nice way to deal with medical information.

- **Ethical AI considerations:**

  The system enables the doctors to know how it makes predictions. This is achieved through making the predictions clearer. The doctors are then able to see the reason why the system made a decision. This assists the doctors to have more confidence in the system and apply it in a manner when attending to patients. The system will be transparent and will assist doctors in making decisions in using data privacy and minimization and artificial intelligence in their practice, medical data and data privacy and minimization.

3.8. Conclusion

CyberSight is a web-based AI-powered healthcare platform developed to assist in the early detection and grading of diabetic retinopathy using retinal fundus images. The system uses advanced deep learning models including Vision Transformers for diabetic retinopathy classification, CLIP for validating fundus images, and occlusion-based explainability techniques to highlight affected retinal regions.

The platform provides role-based access where ophthalmologists can upload images, review predictions, generate reports, and manage patient analysis, while administrators handle user management through a dedicated admin panel. CyberSight uses React.js for an intuitive frontend interface, FastAPI-based REST APIs secured with JWT authentication and HTTPS, and Microsoft Azure SQL Database for secure account and system data management.

The system is designed to be fast, secure, scalable, and reliable while maintaining patient privacy and ethical AI practices. By automating retinal image analysis, CyberSight helps healthcare professionals improve screening efficiency and support earlier diagnosis of diabetic retinopathy.

Chapter # 4

System Design

Chapter 4

System Design

4.1. Design Approach

The Design Approach of CyberSight provides the structure, behavior and interaction of the system parts. It guarantees that the system is efficient, scalable, reliable and flexible.

CyberSight follows a combination of:

- **Client–Server Architecture**
- **Modular AI-Based Pipeline**
- **Microservice-Ready Architecture (for future scalability)**

Such an architectural design isolates the user interface and backend logic and AI processing, so that each layer can be created, tested and deployed separately and the architecture has a high degree of modularity and loose coupling.

Client Layer (React Frontend):

The user interface is developed in React 18 and TypeScript and offers two interfaces that are role-based:

- **Doctor Dashboard**
  - Upload retinal fundus images
  - View AI-based analysis results
  - Display explainable heatmaps
  - Generate downloadable PDF reports
- **Admin Dashboard**
  - Manage doctor accounts (CRUD operations)

React Context API is used to manage state, such as AuthContext

- UploadContext
- AnalyzeContext
- DoctorContext

Every request to the backend is done via a centralized API Client, over the native fetch API with JWT Bearer token authentication.

Server Layer (FastAPI Backend):

FastAPI is used as the core of the application, and it is applied to the backend.

It offers versioned RESTful API endpoints of:

- Prediction
- Authentication
- Administration

The backend is responsible for:

- Request validation

- JWT-based authentication
- Rate limiting
- Delegating tasks to service modules

Core services include:

- PredictionService
- AuthService
- AdminService
- EmailService

The backend is the only entry point, and it orchestrates communication between the client, AI layer and the database.

AI Inference Layer (PyTorch):

The AI system will be constructed as a pipeline with a single task per component.

The image processing flow is as follows:

1. Image Validator (checks format, size, and dimensions)
2. CLIP Fundus Validator (verifies the image is a retinal fundus)
3. Image Preprocessor (resizes and normalizes the image)
4. Vision Transformer (ViT) Classifier (performs disease grading)
5. Saliency Generator (produces explainable heatmaps)

This modular pipeline enables updating or replacement of components without any impact on the entire system.

Database Layer (Azure SQL):

The database layer deals with persistent storage in Microsoft Azure SQL.

It stores:

- Doctor accounts
- Role assignments
- Password management tokens

This is done by importing pyodbc, and:

- Raw SQL queries
- Context-managed connections

4.2. Design Constraints

Reliability	The system is designed to maintain a stable and reliable connection between different parts of the application. This ensures smooth operation during image upload, prediction, and report generation without interruptions.
Criticality of the Application	CyberSight is a medical support system, accuracy is very important. The system provides results to assist doctors, but all outputs include a disclaimer to clarify that they are not a replacement for professional medical diagnosis.
Safety and security considerations	The system is designed to protect sensitive medical data by allowing access only to authorized users. Different access levels are provided for doctors and administrators, and measures are in place to prevent misuse and ensure secure and safe system usage.
Internet Connection	CyberSight requires a stable internet connection since it is a web-based system. The application depends on network connectivity for communication between different parts of the system and for accessing stored data. However, once the system is fully running, the core prediction process works independently without relying on external services.
Performance	The system is designed to provide results efficiently while keeping costs low. Predictions are generated within a short time, and the system remains responsive even during processing. Future upgrades can further improve speed and performance.
Image Constraints	The system is designed to accept valid image files in standard formats (such as JPG and PNG). Images must meet certain size and dimension requirements, and only proper retinal images are allowed for analysis to ensure accurate results.
Browser Compatibility	The system is designed to generate PDF reports directly in the user's browser. This requires a modern browser that supports advanced features. The reports are displayed clearly and can span multiple pages when needed.

Scalability	The system is designed to support one disease model, but it is designed to support multiple models in the future. Each model can run independently, allowing the system to grow easily and handle more diseases without affecting existing functionality.

Table 22: Design Constraints

4.3. System Architecture

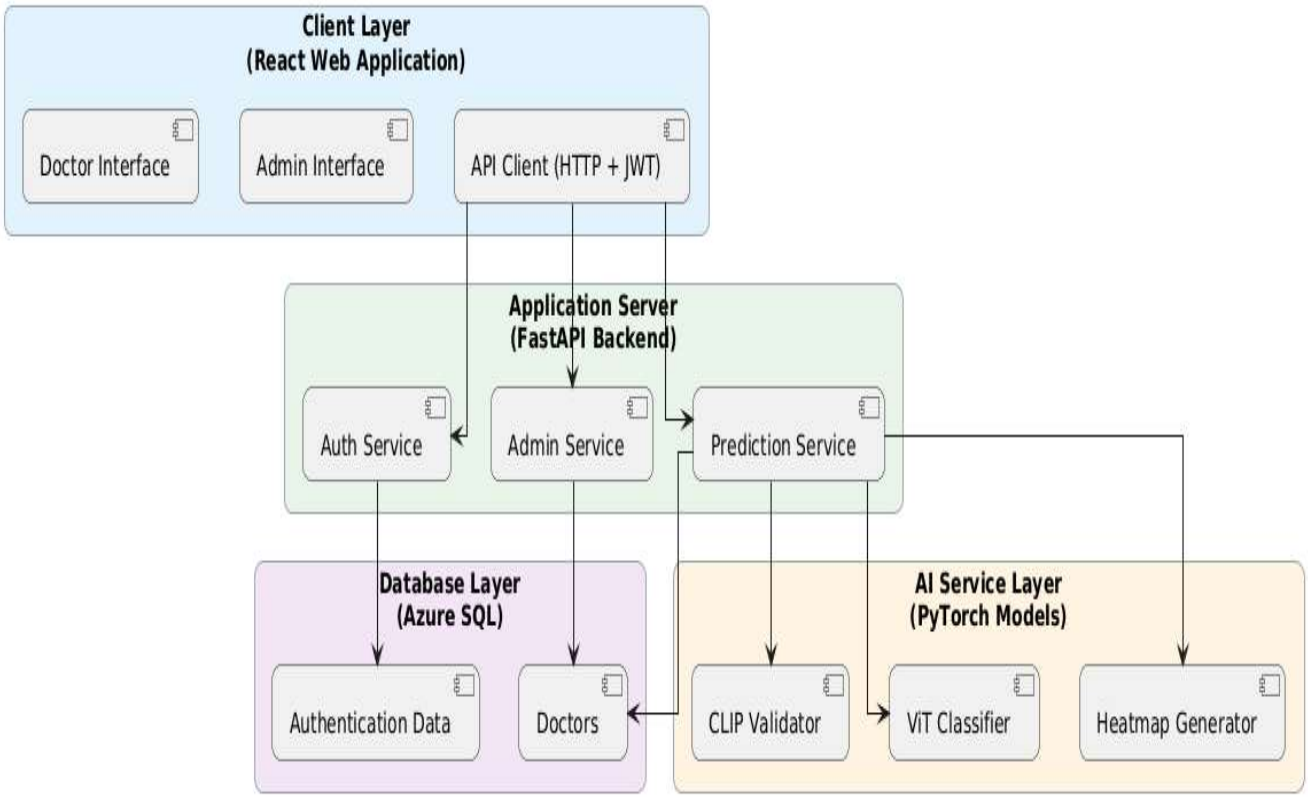

Figure 2: System Architecture

CyberSight is structured around a multi-layer client-server architecture, with a specific AI inference layer that is always well-separated in responsibilities and modular design of the system.

Client Layer (Frontend)

The client layer is built using React and provides user interfaces for both doctors and administrators.

- Doctors are able to upload retinal images, view analysis results, and generate reports
- Admins are able to manage doctor accounts
- The frontend communicates with the backend through API requests

Application Layer (Backend)

The backend is built using FastAPI and serves as the central controller of the system.

- Handles authentication and authorization
- Processes user requests from the frontend
- Validates input data
- Coordinates communication with the AI layer and database

AI Inference Layer

The AI layer takes a structured pipeline to process medical images.

- Validates uploaded images
- Analyzes images to generate predictions
- Produces visual explanations (heatmaps)

It is a modular layer and can be further extended to accommodate various disease models in future.

Database Layer

Data of the system are stored and managed in the database layer.

- Stores doctor accounts and roles
- Manages authentication-related information
- Ensures reliable data storage and retrieval

Conclusion

To sum up, the CyberSight system is developed based on a well-organized and modular design that guarantees efficiency, reliability, and scalability. It divides the system into layers of client, server, AI and database which facilitates effective communication within the components and clarifies responsibility. The design facilitates proper medical analysis and keeps the doctors and administrators secure and user-friendly. Additionally, the system is flexible enough to support future enhancements, such as adding new AI models and improving performance, making it a robust and future-ready solution.

The architecture of CyberSight is designed to support smooth integration between the frontend interface, backend services, AI inference modules, and database components. This layered structure improves maintainability and allows each module to operate independently while still communicating efficiently with the overall system. The modular approach also simplifies future

upgrades, testing, and deployment processes. Furthermore, the architecture can easily accommodate additional medical imaging features, larger datasets, and advanced AI functionalities without affecting the existing workflow. This makes CyberSight a scalable and adaptable platform for future healthcare and AI-based diagnostic applications.

4.4. Logical Design

4.4.1 Class Diagram:

Figure 3: Class Diagram

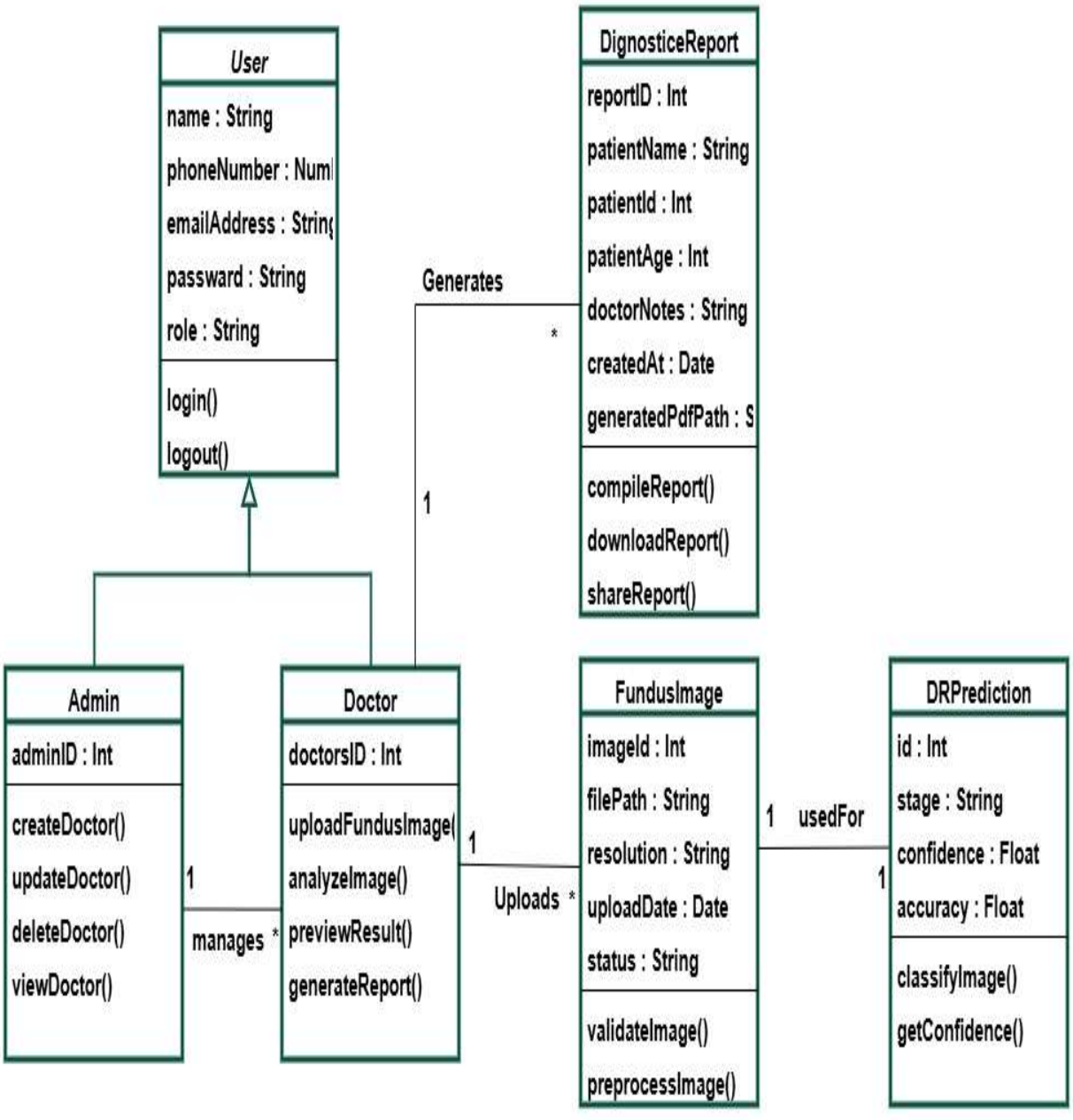

Figure 3: Class Diagram

Description:

- User Class:

  The User class is the parent class for all users in the CyberSight system. It contains common attributes such as name, phoneNumber, emailAddress, password, and role, along with functions like login() and logout(). Admin and Doctor classes inherit from this class through a generalization relationship, allowing shared properties and behaviors.

- Admin Class:

  The Admin class is a subclass of User that represents the system administrator. It includes the attribute adminID and functions such as createDoctor(), updateDoctor(), deleteDoctor(), and viewDoctor(). The Admin manages doctor accounts, controls access to the system, and ensures that only authorized medical professionals can use the platform. One Admin can manage multiple Doctors.

- Doctor Class:

  The Doctor class represents ophthalmologists who use the system for diabetic retinopathy diagnosis. It contains the attribute doctorID and operations including uploadFundusImage(), analyzeImage(), previewResult(), and generateReport(). Doctors upload retinal images, review AI-generated predictions, and create diagnostic reports. One Doctor can upload many Fundus Images and generate multiple Diagnostic Reports.

- FundusImage Class:

  The FundusImage class stores retinal images uploaded by doctors. It includes attributes such as imageID, filePath, resolution, uploadDate, and status. Functions like validateImage() and preprocessImage() ensure image quality before analysis. Each FundusImage is associated with one DRPrediction through a one-to-one relationship.

- DRPrediction Class:

  The DRPrediction class represents the AI-generated prediction result for a retinal image. It contains attributes such as predictionID, stage, confidenceScore, and accuracyRate, along with functions like classifyImage() and getConfidence(). The class determines the severity stage of diabetic retinopathy and provides prediction confidence values.

- DiagnosticReport Class:

  The DiagnosticReport class represents the final medical report generated after analysis. It includes attributes such as reportID, patientName, patientID, patientAge, doctorNotes, createdAt, and generatedPdfPath. Functions include compileReport(), downloadReport(), and shareReport(). The report summarizes patient details, AI predictions, and doctor observations.

4.5. Dynamic View

4.5.1. Sequence Diagram:

4.5.1.1 Login Flow

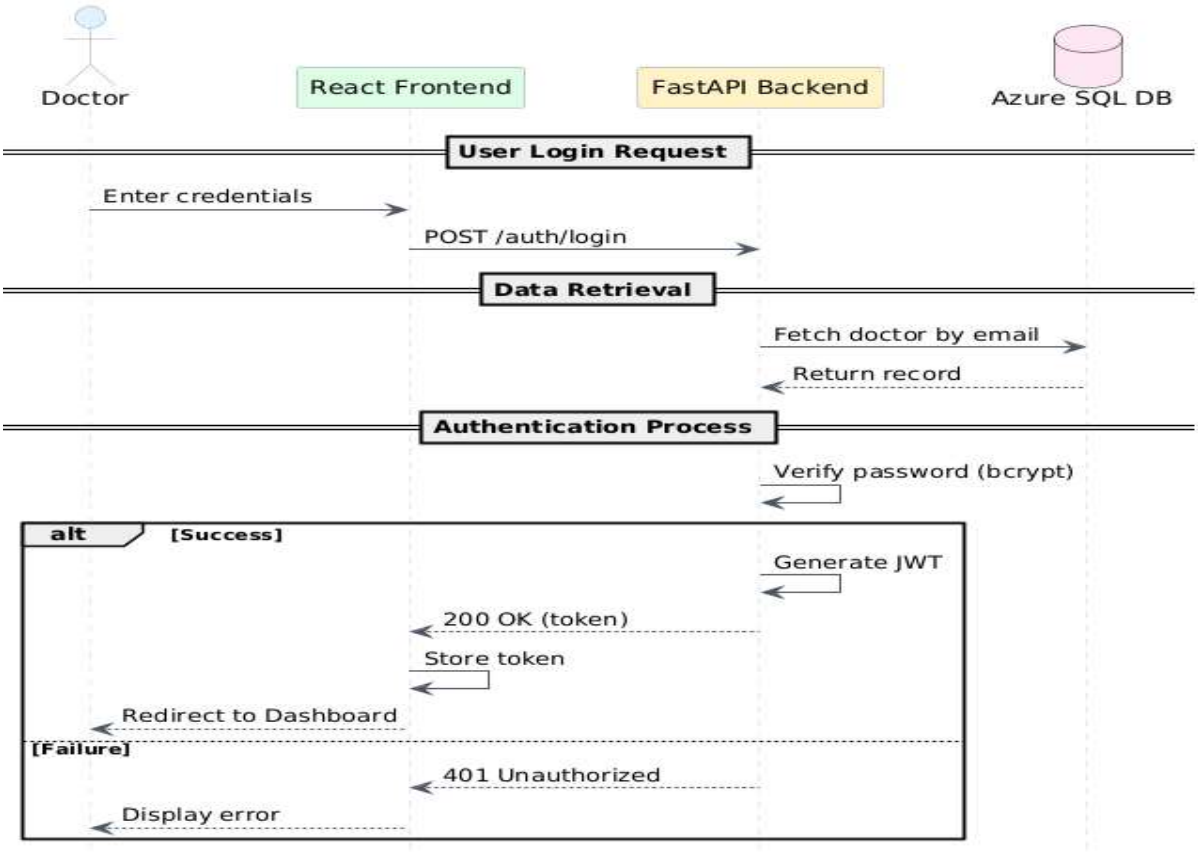

Figure 4: Sequence Diagram of Login Flow

Description:

The sequence diagram illustrates the authentication process of the CyberSight system, showing how a doctor securely logs into the platform. The doctor enters login credentials through the React frontend, which sends the authentication request to the FastAPI backend. The backend then retrieves the user record from the Azure SQL database and verifies the password using secure hashing techniques such as bcrypt. If the credentials are valid, the backend generates a JSON Web Token (JWT) and returns it to the frontend, allowing the doctor to access the system dashboard and features. If the credentials are invalid, the system returns an unauthorized response and displays an error message. This workflow ensures secure and role-based access control within the CyberSight system.

4.5.1.2 Upload Image

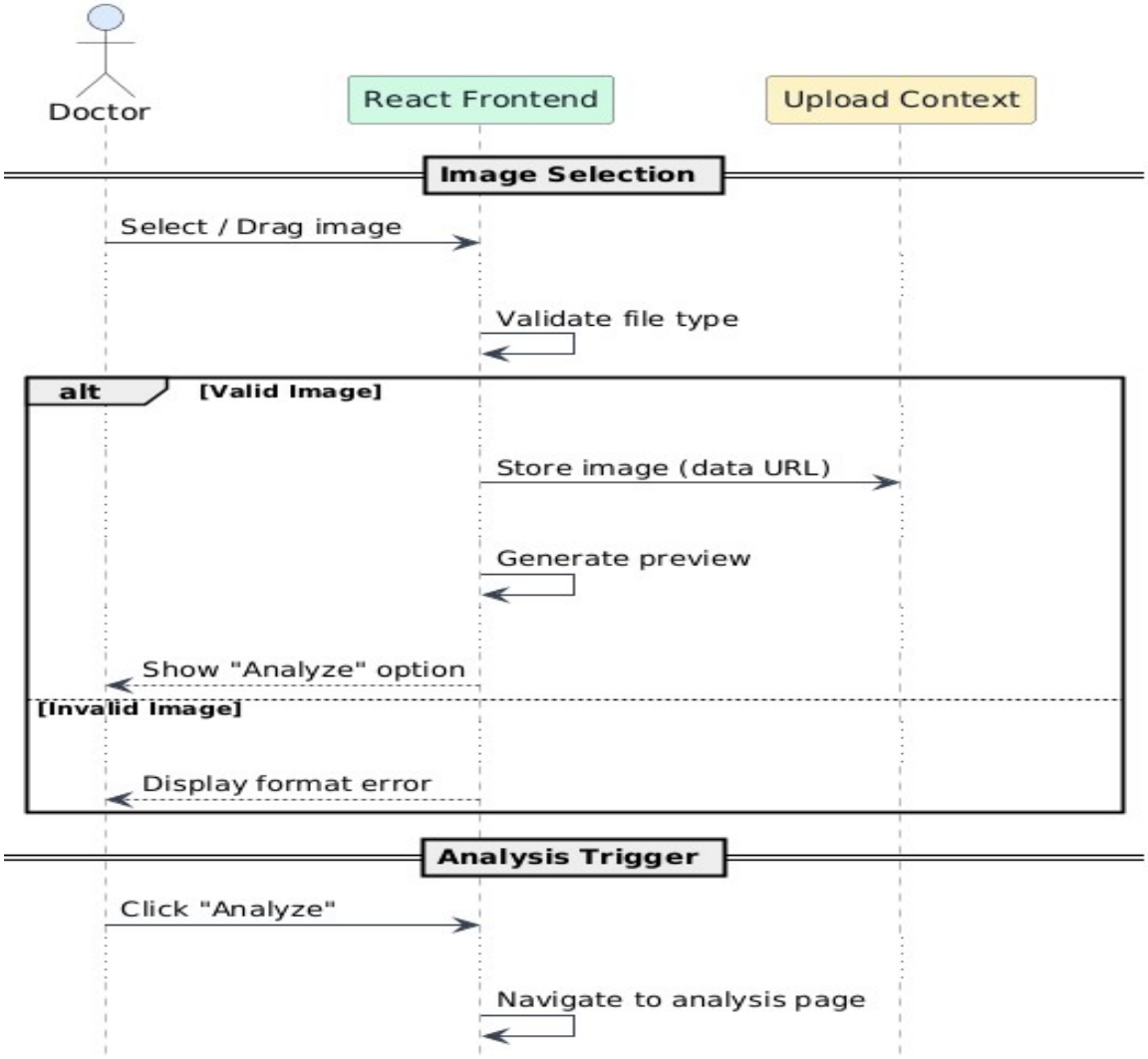

Figure 5: Sequence Diagram of Upload Image

Description:

The sequence diagram illustrates the image upload process in the CyberSight system, where a doctor uploads a retinal fundus image through the React-based frontend for diabetic retinopathy analysis. The system performs client-side validation to ensure that the uploaded file is in a supported format such as JPEG or PNG before processing begins.

After successful validation, the image is temporarily stored and a preview is displayed to provide visual feedback to the user. The doctor can then proceed with the analysis process, while invalid files trigger an error message and stop the workflow. This process ensures secure input validation, smooth user interaction, and reliable image handling within the CyberSight system.

4.5.1.3 Prediction Flow

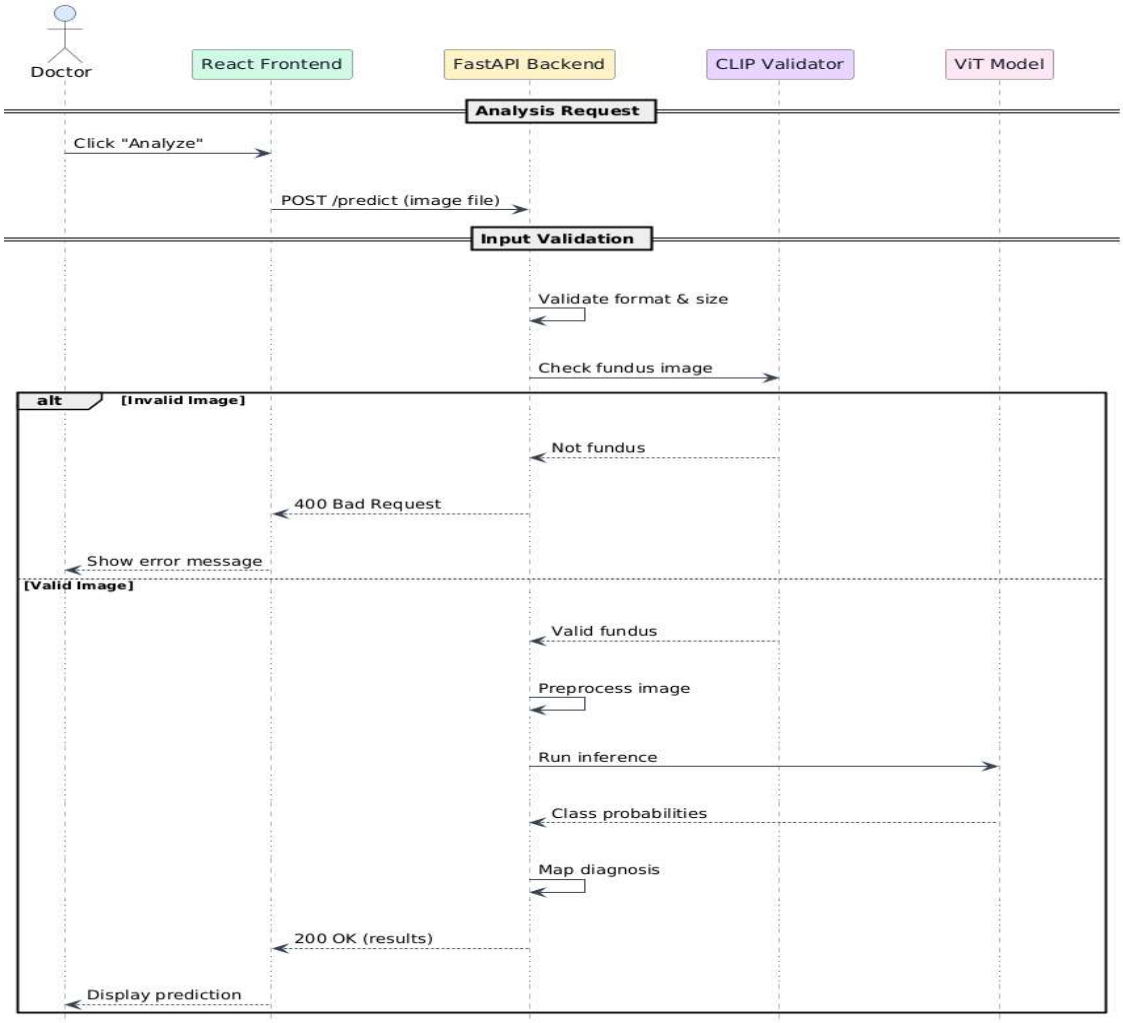

Figure 6: Sequence Diagram of Prediction Flow

Description:

The sequence diagram illustrates the prediction workflow of the CyberSight system for detecting and classifying diabetic retinopathy from retinal fundus images. The process begins when the doctor clicks the "Analyze" button on the frontend, which sends the uploaded image to the FastAPI backend for processing. The backend validates the image by checking its format, size, and dimensions, followed by a CLIP-based verification step to confirm that the uploaded image is a retinal fundus image.After successful validation, the image is preprocessed and passed to the Vision Transformer (ViT) model for inference. The model generates prediction probabilities for different diabetic retinopathy severity levels, which are then converted into a diagnostic result by the backend. Finally, the prediction results are sent back to the frontend and displayed to the doctor, ensuring an accurate and reliable diagnostic workflow within the CyberSight system.

4.5.1.4 Heatmap generation flow

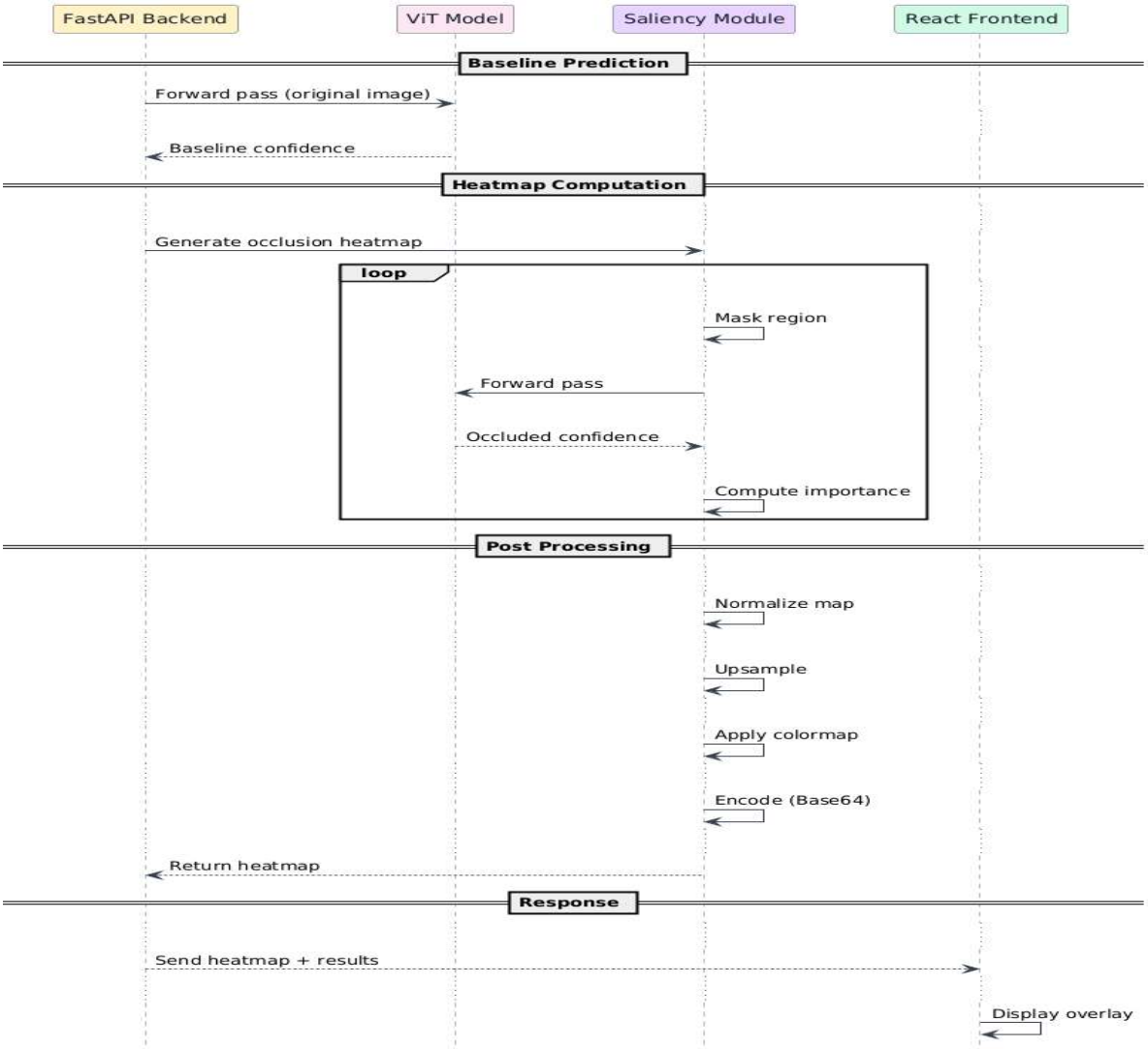

Figure 7: Sequence Diagram of Heatmap generation flow

Description:

The sequence diagram illustrates the heatmap generation process in the CyberSight system, which provides visual explanations for AI predictions. The process begins when the FastAPI backend performs a baseline prediction using the Vision Transformer (ViT) model on the original retinal image. The saliency module then applies an occlusion-based explainability method by masking different image regions and measuring their impact on prediction confidence.

The calculated importance values are combined to generate a heatmap, which is normalized and enhanced with a color overlay for better visualization. The generated heatmap is then returned to the frontend and displayed alongside the original retinal image, helping doctors understand which retinal regions influenced the AI prediction.

4.5.1.5 Report Generation Flow

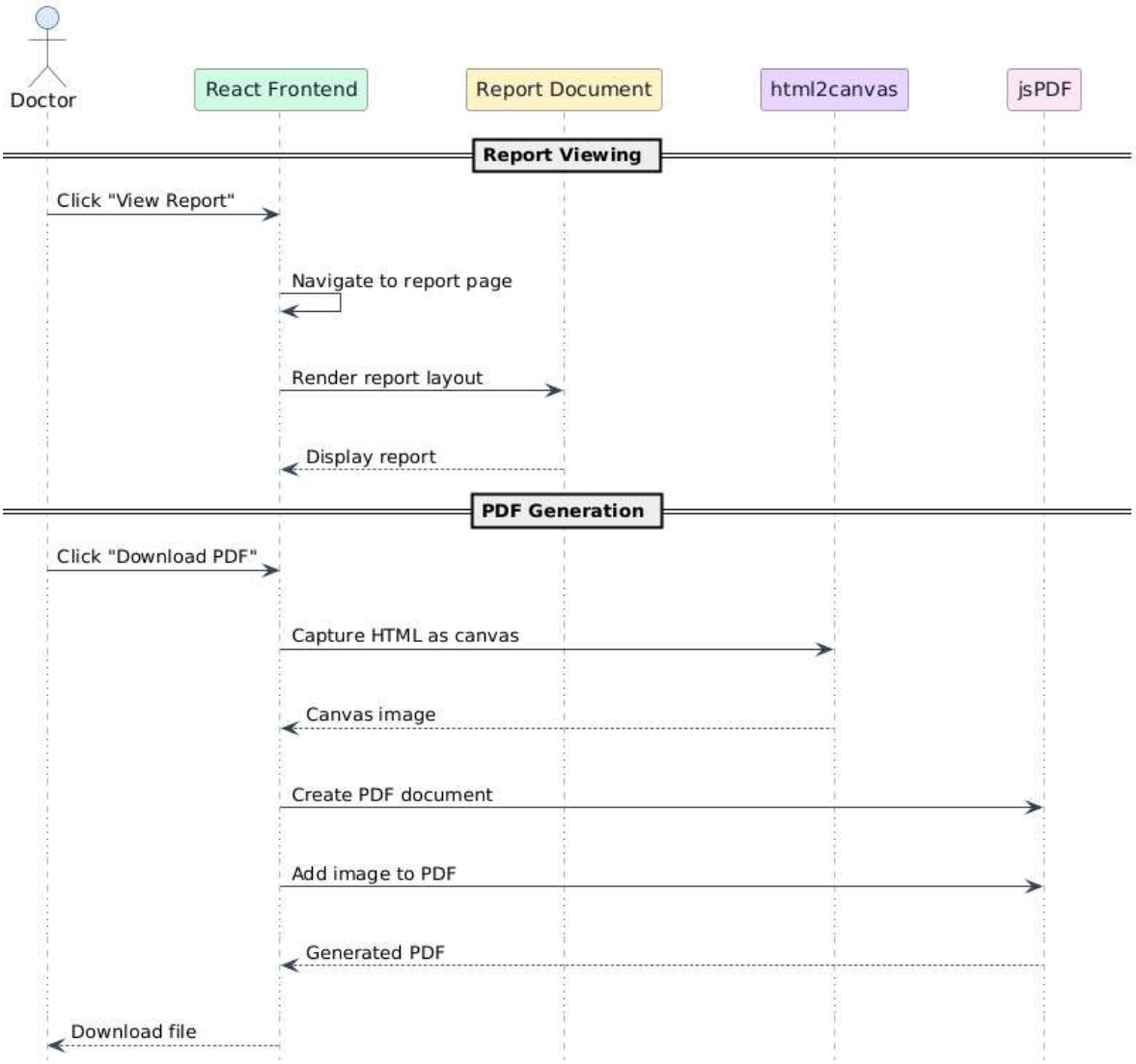

Figure 8: Sequence Diagram of Report Generation Flow

Description:

The sequence diagram illustrates the report generation workflow in the CyberSight system, where diagnostic results are displayed and exported as a downloadable PDF document. The process begins when the doctor selects the “View Report” option on the frontend, which displays patient details, prediction results, severity level, heatmap visualization, and clinical findings in a structured format. The doctor can then download the report in PDF format. The frontend uses the html2canvas library to capture the report layout and jsPDF to convert it into a properly formatted PDF document. This workflow allows easy documentation, portability, and sharing of diagnostic reports within the CyberSight system.

4.5.1.6 Admin Doctor Management Flow

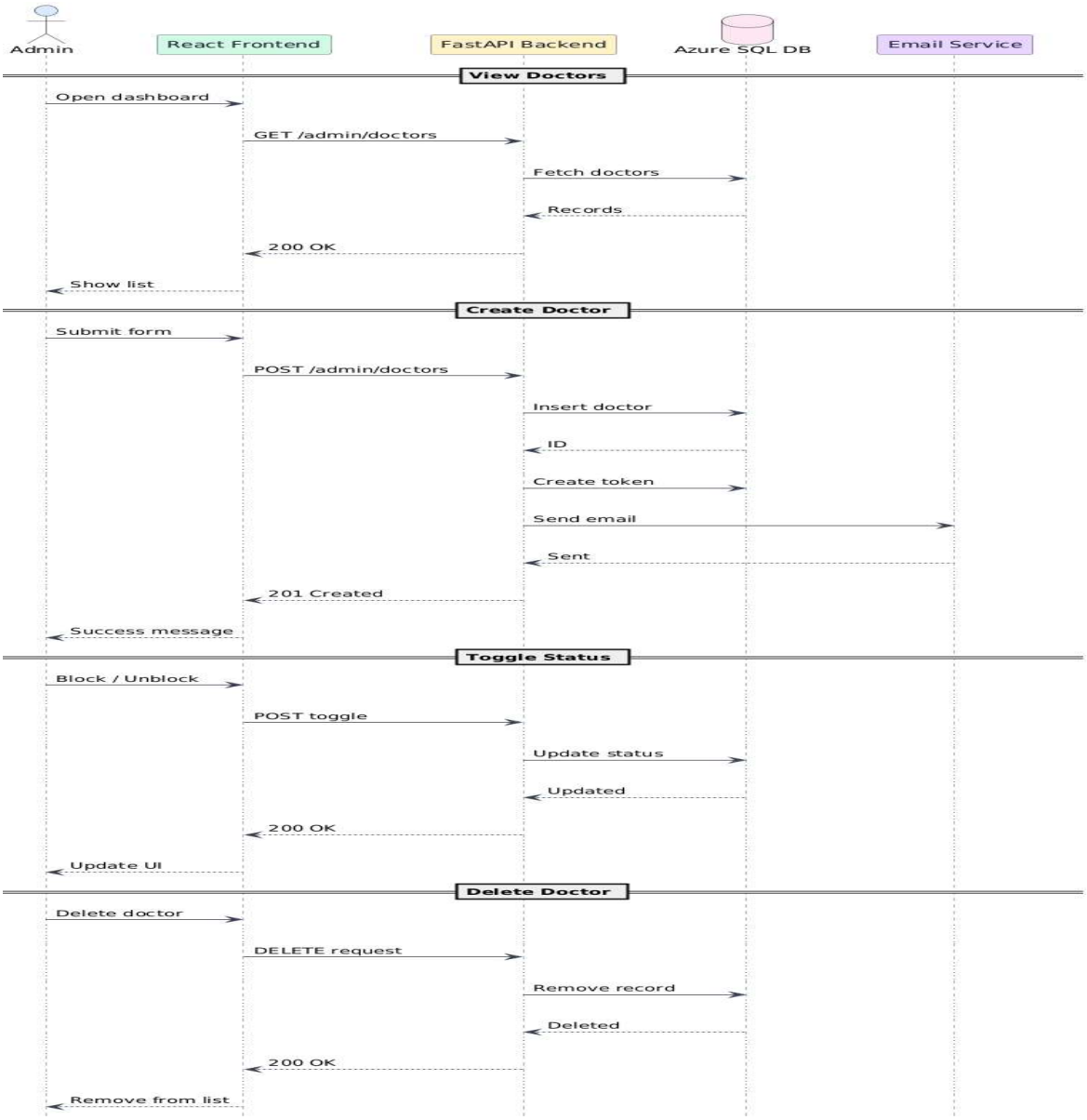

Figure 9: Sequence Diagram of Admin Doctor Management Flow

Description:

The sequence diagram shows the doctor account management process in the CyberSight system. The admin can view, create, block, unblock, or delete doctor accounts through the dashboard. The frontend communicates with the backend, which retrieves and updates doctor records in the database.For new accounts, the system stores doctor details, generates a secure password setup token, and sends an email for account activation. This workflow ensures secure and efficient management of doctor access within the CyberSight platform.

4.5.1.7 Set Password Flow

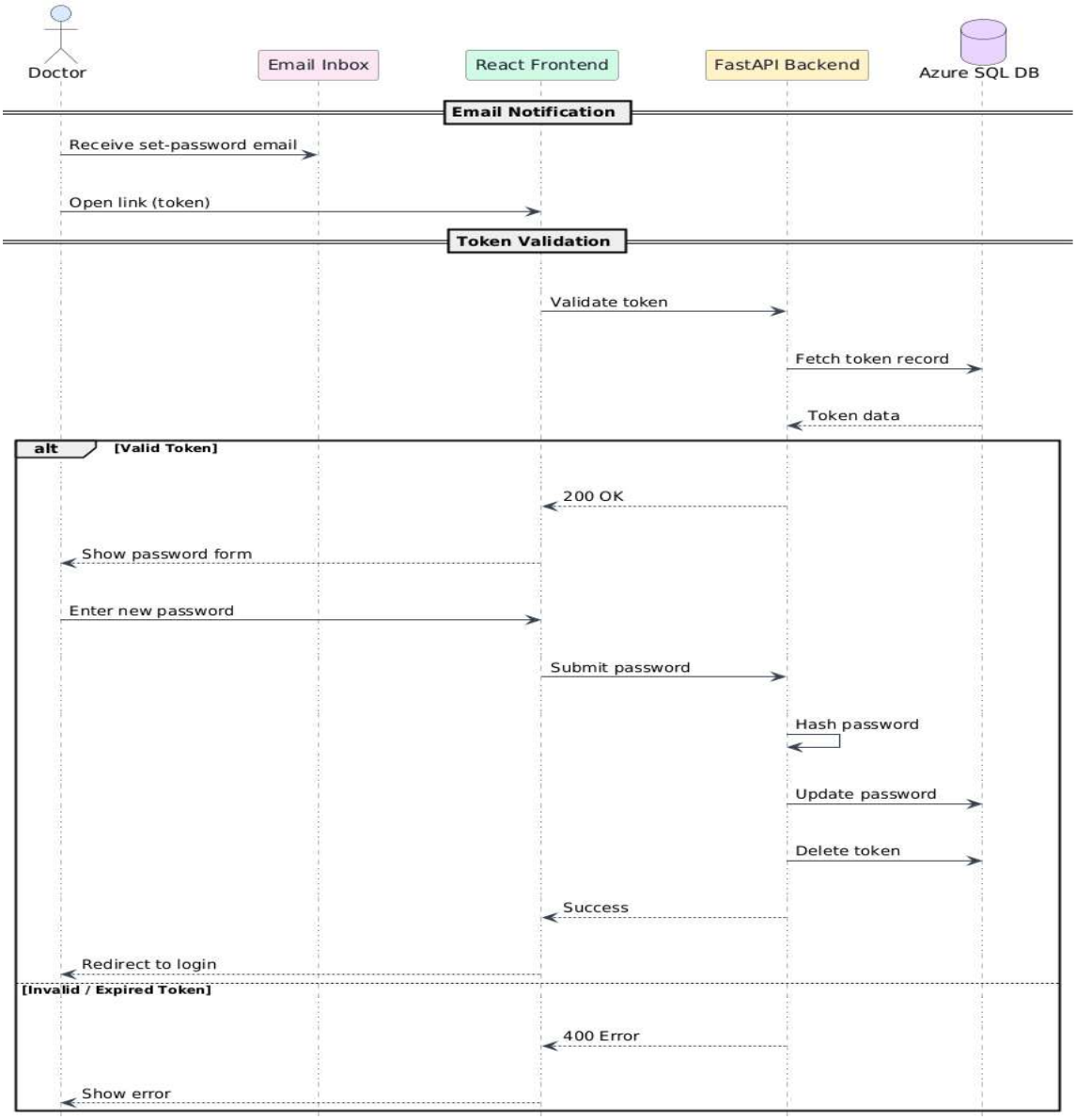

Figure 10: Sequence Diagram of Set Password Flow

Description:

The sequence diagram illustrates the password setup process in the CyberSight system. A doctor receives an email containing a secure password setup link with a unique token. When the link is opened, the frontend sends the token to the backend for validation against the database.
If the token is valid, the doctor can create a new password, which is securely hashed and stored in the database. The token is then removed to prevent reuse, and the user is redirected to the login page. Invalid or expired tokens display an error message to maintain system security.

4.5.1.8 Forgot Password Flow

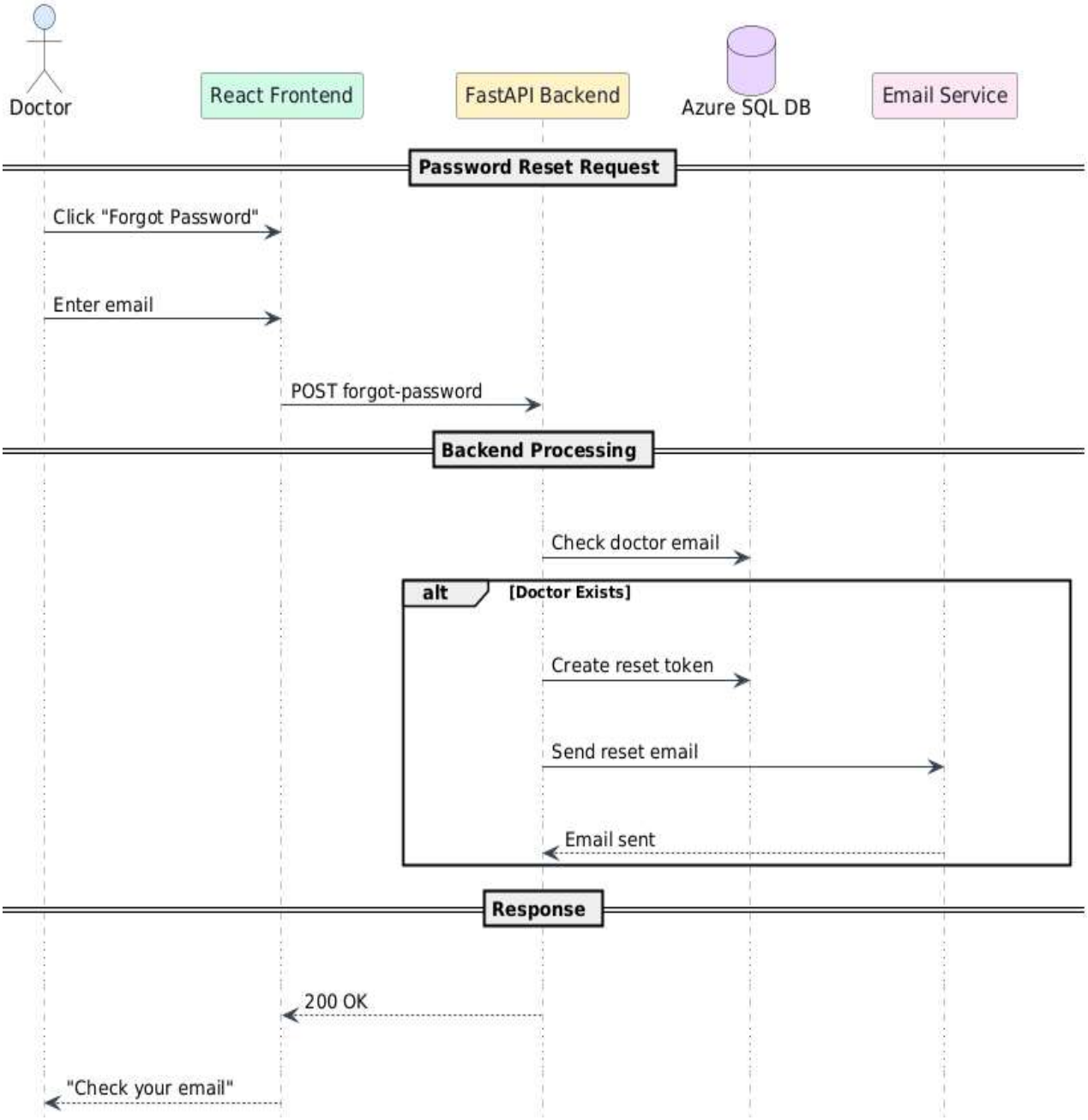

Figure 11: Sequence Diagram of Forgot Password Flow

Description:

The sequence diagram illustrates the forgot password process in the CyberSight system. The doctor enters a registered email through the frontend, which sends the request to the FastAPI backend for verification against the database. If the account exists, the backend generates a secure reset token and sends a password reset link through the email service. The system always returns a generic success message to protect user privacy and prevent unauthorized account discovery.

4.5.1.9 CLIP Fundus Validation Flow

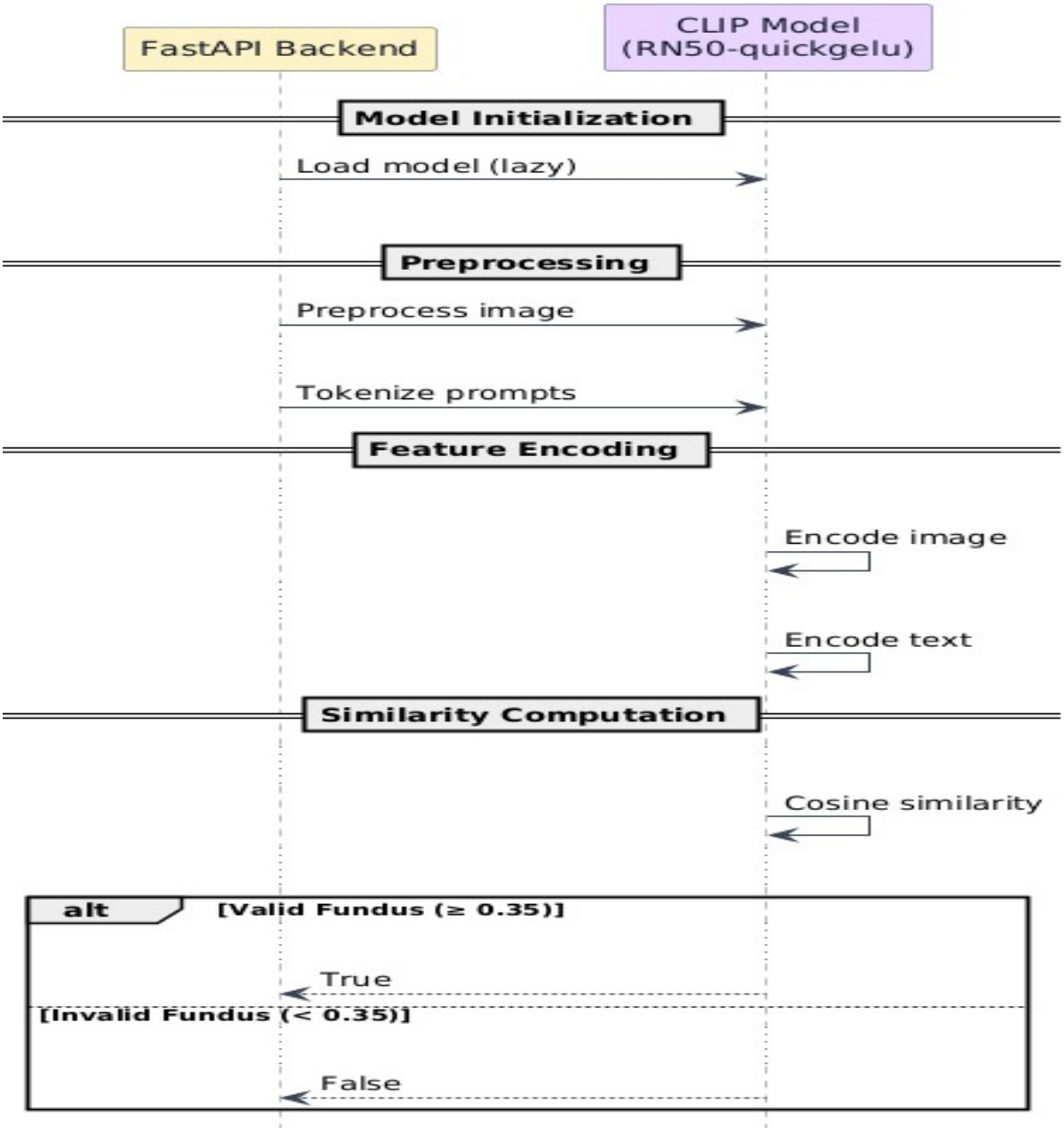

Figure 12: Sequence Diagram of CLIP Fundus Validation Flow

Description:

The sequence diagram illustrates the CLIP-based fundus validation process in the CyberSight system. The FastAPI backend loads the CLIP model and processes the uploaded image along with predefined retinal and non-retinal text prompts. The model compares image and text features using cosine similarity and generates probability scores.If the retinal probability exceeds the defined threshold, the image is accepted for further AI analysis; otherwise, it is rejected with an error message. This process improves system reliability by ensuring that only valid retinal fundus images are processed.

4.5.1.10 Role-Based Route Protection Flow

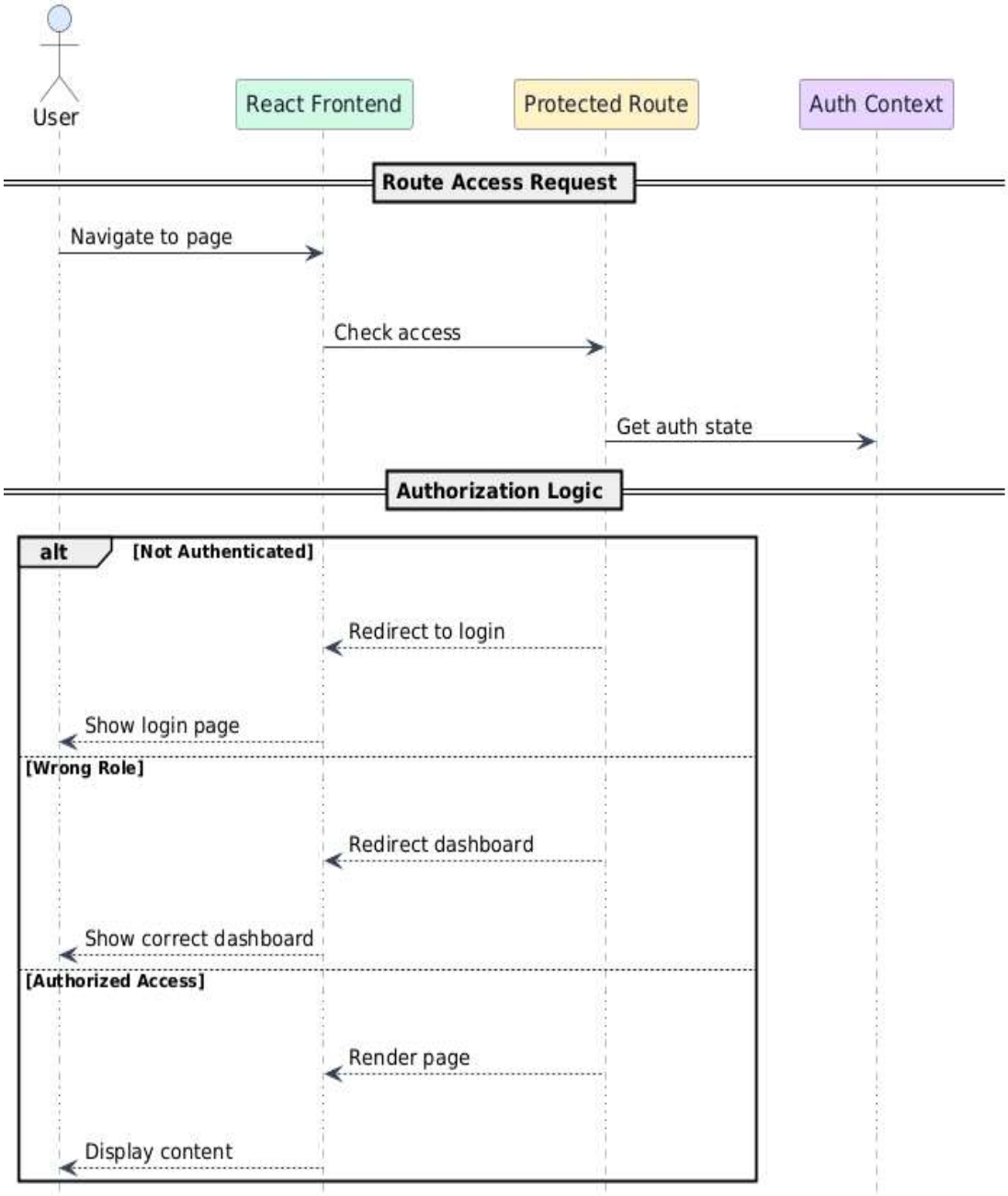

Figure 13: Sequence Diagram of Role-Based Route Protection Flow

Description:

The sequence diagram illustrates the role-based route protection mechanism in the CyberSight system. When a user attempts to access a protected route, the React frontend checks the user's authentication status and role through the authentication context. If the user is not authenticated, they are redirected to the login page. If the user lacks the required role, they are redirected to an authorized dashboard. Access is only granted when the user is both authenticated and authorized, ensuring secure role-based access control within the system.

4.5.2 Activity Diagram

4.5.2.1. Image Upload Workflow

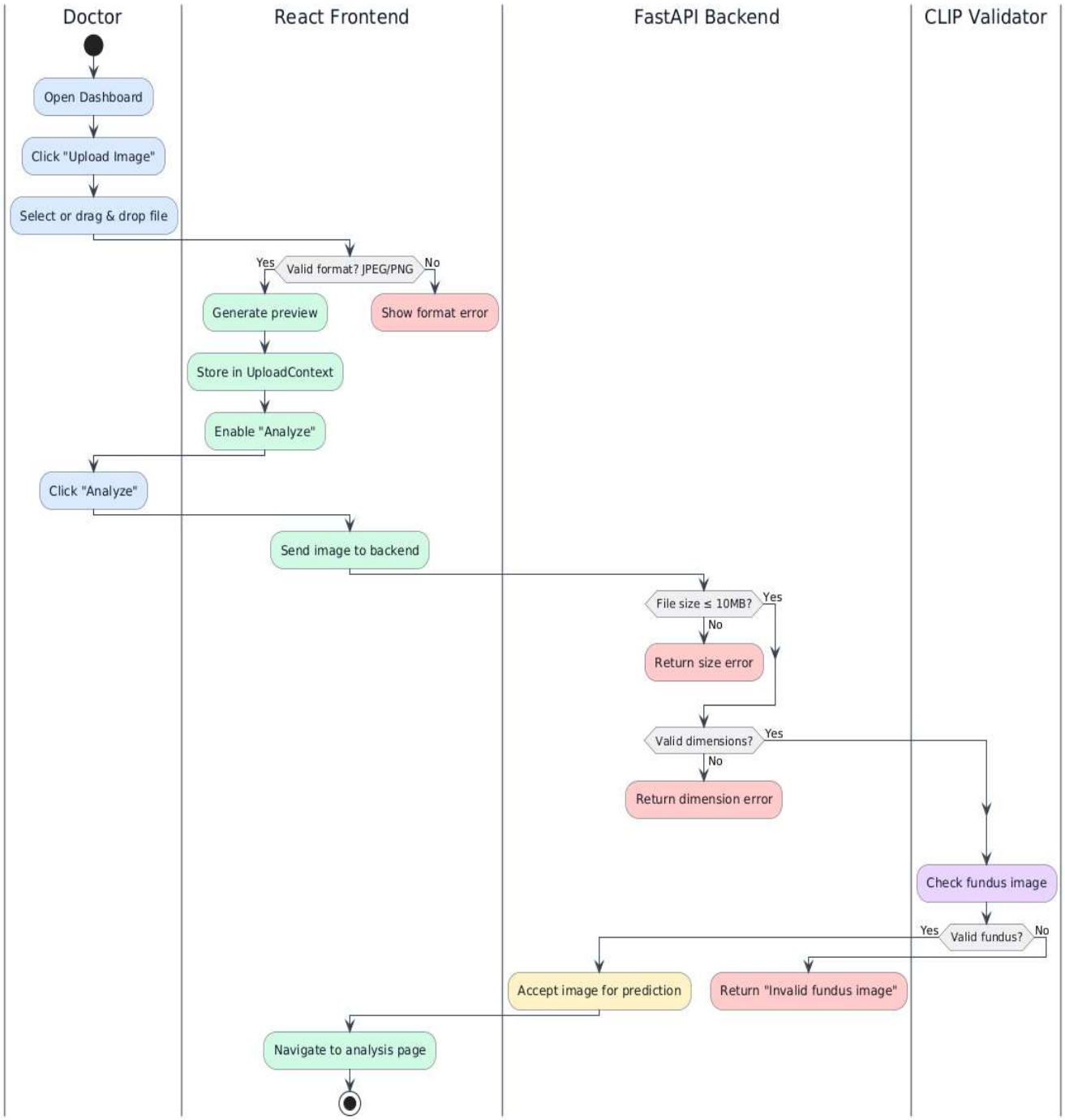

Figure 14: Activity Diagram of Image Upload Workflow

Description:

The activity diagram illustrates the complete image upload workflow in the CyberSight system, showing how a doctor interacts with the platform to upload a retinal fundus image for diabetic retinopathy analysis. The process begins when the doctor accesses the dashboard and selects or drags and drops an image into the React-based frontend. The frontend immediately performs client-side validation to ensure that the uploaded file is in a supported format such as JPEG or PNG. If the file format is invalid, the system displays an error message and terminates the process to prevent unsupported files from entering the diagnostic pipeline.

When the uploaded image passes the initial validation stage, the system generates an image preview and temporarily stores the file within the application state. This allows the doctor to visually confirm the selected image before proceeding with analysis. The Analyze option is then enabled, allowing the doctor to submit the image for further processing. This step improves user experience by providing immediate visual feedback and reducing the chances of uploading incorrect images.

After the doctor clicks the Analyze button, the image is securely sent to the FastAPI backend where additional validation checks are performed. These validations include checking file size, image dimensions, and image quality to ensure that the uploaded image meets the system requirements. The backend then forwards the image to the CLIP-based validation module, which verifies whether the uploaded image is actually a retinal fundus image. This additional verification step improves system robustness by preventing unrelated or invalid images from being processed by the AI model.

If any validation step fails, the system immediately returns an appropriate error message to the frontend and stops further execution. However, if the image successfully passes all validation stages, it is accepted for prediction and preprocessing. The frontend then redirects the doctor to the analysis page, where the prediction results, confidence scores, and explainability outputs are displayed. This workflow ensures secure image handling, reliable validation, improved diagnostic accuracy, and a smooth user experience within the CyberSight system.

4.5.2.2. AI Inference Pipeline

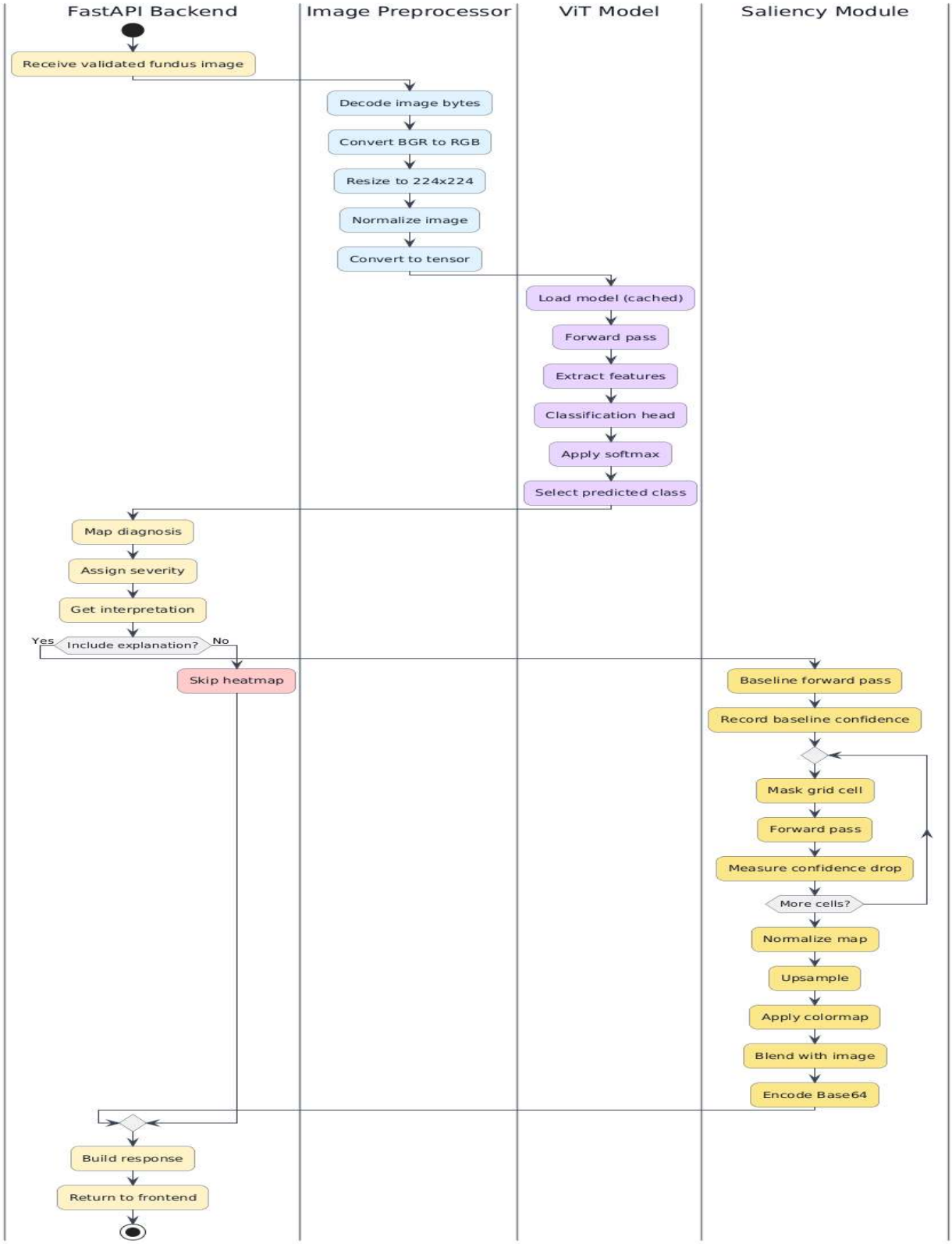

Figure 15: Activity Diagram of AI Inference Pipeline

Description:

The activity diagram represents the complete AI inference pipeline of the CyberSight system and demonstrates how a retinal fundus image is processed to generate a diabetic retinopathy diagnosis. The workflow starts when a doctor uploads a retinal image through the frontend interface. The FastAPI backend first validates the uploaded image to ensure that it is a retinal fundus image and meets the required quality standards. Once validation is completed successfully, the image is forwarded to the preprocessing module for further analysis.During preprocessing, multiple image enhancement and transformation operations are performed to prepare the image for deep learning inference. These operations include image decoding, conversion from BGR to RGB color format, resizing the image to the required input dimensions, normalization using ImageNet mean and standard deviation values, and tensor conversion for PyTorch compatibility. These preprocessing steps ensure consistency in image quality, reduce noise, and improve the reliability and stability of model predictions.

After preprocessing, the transformed image is passed to the Vision Transformer (ViT) model, which serves as the primary deep learning architecture for diabetic retinopathy classification. The model performs a forward pass to extract important retinal features such as lesions, hemorrhages, exudates, and abnormal blood vessel patterns. Based on these extracted features, the model classifies the retinal image into one of the diabetic retinopathy severity stages ranging from No DR to Proliferative DR. A softmax layer is then applied to generate probability scores for each class, enabling the system to determine the most probable diagnosis along with confidence values.The backend maps the predicted class index to a corresponding clinical severity level and generates a simplified interpretation that can easily be understood by ophthalmologists and medical staff. This interpretation assists doctors in understanding the seriousness of the disease and supports clinical decision-making during patient diagnosis and treatment planning.To improve transparency and trust in AI predictions, the CyberSight system also integrates an explainable AI module based on occlusion saliency analysis. When explainability is enabled, the system systematically masks different regions of the retinal image and repeatedly performs inference to observe changes in prediction confidence. Regions causing larger confidence drops are considered more important for the final prediction. These importance values are combined into a saliency map that highlights the retinal regions influencing the AI decision.The generated saliency map undergoes normalization, resizing, color mapping, and overlay processing to create a visual heatmap explanation. This heatmap is blended with the original retinal image so doctors can visually identify the critical retinal regions detected by the model. The explainability module improves interpretability, increases clinician trust, and helps reduce the black-box nature commonly associated with deep learning systems in healthcare applications.Once all processing and explainability steps are completed, the backend compiles the final response. The response includes the predicted diabetic retinopathy stage, confidence scores, severity level, generated heatmap visualization, and a medical disclaimer stating that the system provides AI-assisted decision support rather than replacing professional medical judgment. The final results are securely transmitted back to the frontend interface, where doctors can review the prediction, analyze the heatmap, generate diagnostic reports, and share findings with patients.

4.5.2.3. Report Generation Workflow

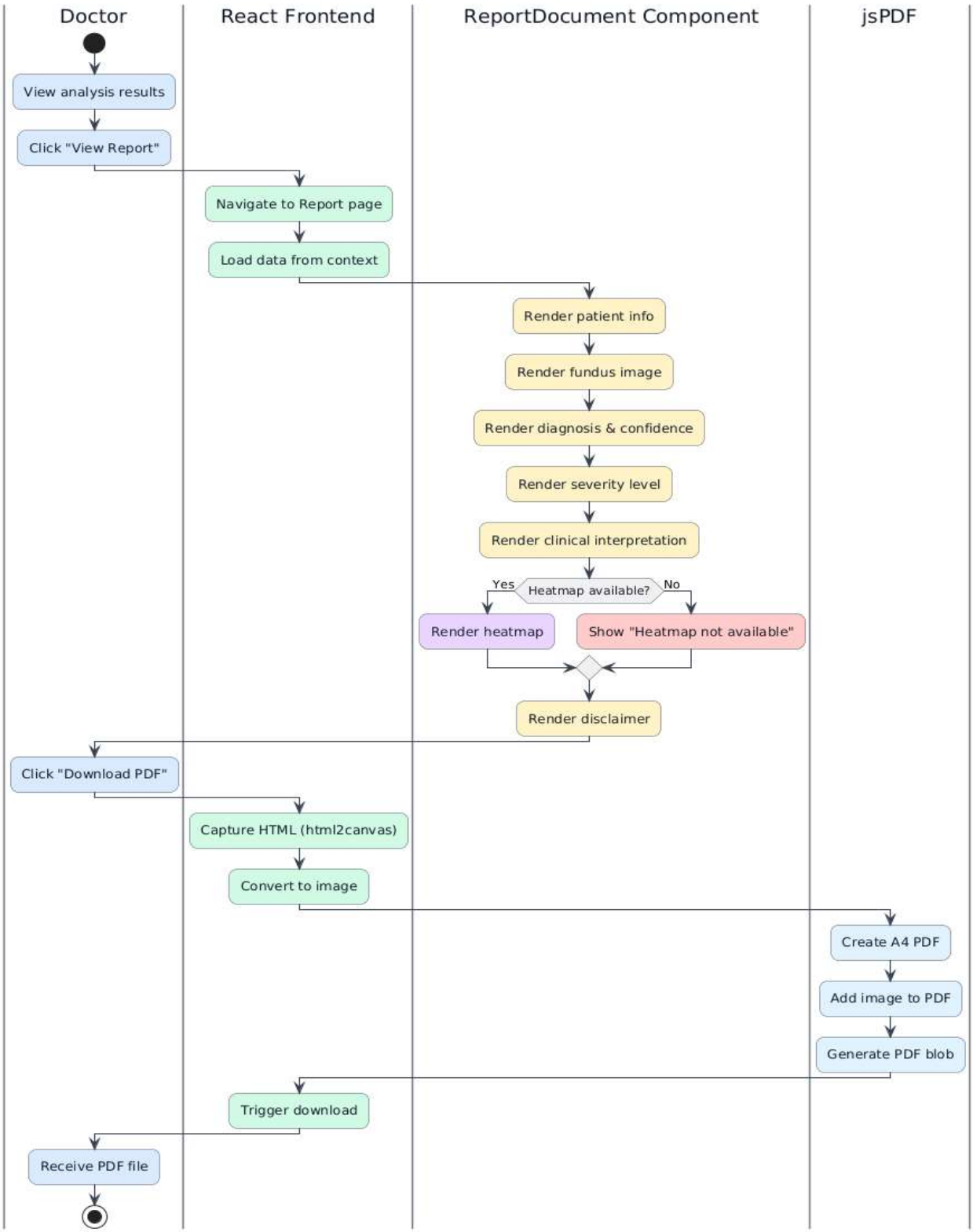

Figure 16: Activity Diagram of Report Generation Workflow

Description:

The activity diagram illustrates the complete report generation workflow in the CyberSight system, where diagnostic outputs are transformed into a structured and downloadable medical report. The process begins after the doctor reviews the AI-generated analysis results and selects the "View Report" option from the frontend interface. The React-based frontend then navigates to the dedicated report page and retrieves all required diagnostic data from the application context and backend response. The report layout is dynamically generated to present information in a clear and organized format suitable for clinical review and documentation.

The generated report includes multiple diagnostic elements such as patient information, the uploaded retinal fundus image, predicted diabetic retinopathy stage, confidence scores, severity level, and clinical interpretation. These details help doctors better understand the AI-generated diagnosis and provide supporting information during clinical evaluation. If explainable AI functionality is enabled, the generated heatmap visualization is also included within the report to highlight the retinal regions that influenced the model's prediction. In cases where explainability is not available, the system displays a suitable placeholder or informational message. This flexibility ensures that the report remains informative and consistent under different diagnostic conditions.

Once the report content is fully rendered on the frontend, the doctor is provided with an option to download the report as a PDF document. The frontend utilizes the html2canvas library to capture the rendered HTML report as a high-quality image representation. This image is then passed to the jsPDF library, which converts the captured content into a professionally formatted A4-sized PDF document. During this process, the system ensures proper scaling, spacing, and formatting so that the report remains readable and visually organized when printed or shared digitally.

After successful PDF generation, the completed report file is returned to the frontend, and the browser automatically triggers the download process. The doctor can then save, print, or share the report with patients or other healthcare professionals as part of clinical documentation and consultation. This workflow improves accessibility and portability of diagnostic information while reducing manual documentation effort within healthcare environments.

In addition to improving usability, the report generation process also contributes to transparency and accountability in AI-assisted diagnosis. By combining prediction results, visual explanations, confidence scores, and medical disclaimers into a single document, CyberSight ensures that diagnostic outputs remain interpretable and clinically meaningful. The structured reporting workflow enhances communication between doctors and patients while supporting efficient record keeping and future medical reference. Overall, the report generation module plays an important role in making the CyberSight system practical, professional, and suitable for real-world clinical deployment. Additionally, the report generation workflow follows a modular structure that separates report rendering, data processing, and PDF generation into independent components. This improves maintainability and allows future enhancements to be integrated more easily. The organized workflow makes the reporting module efficient and suitable for practical healthcare documentation.

4.5.2.4. Doctor Account Creation

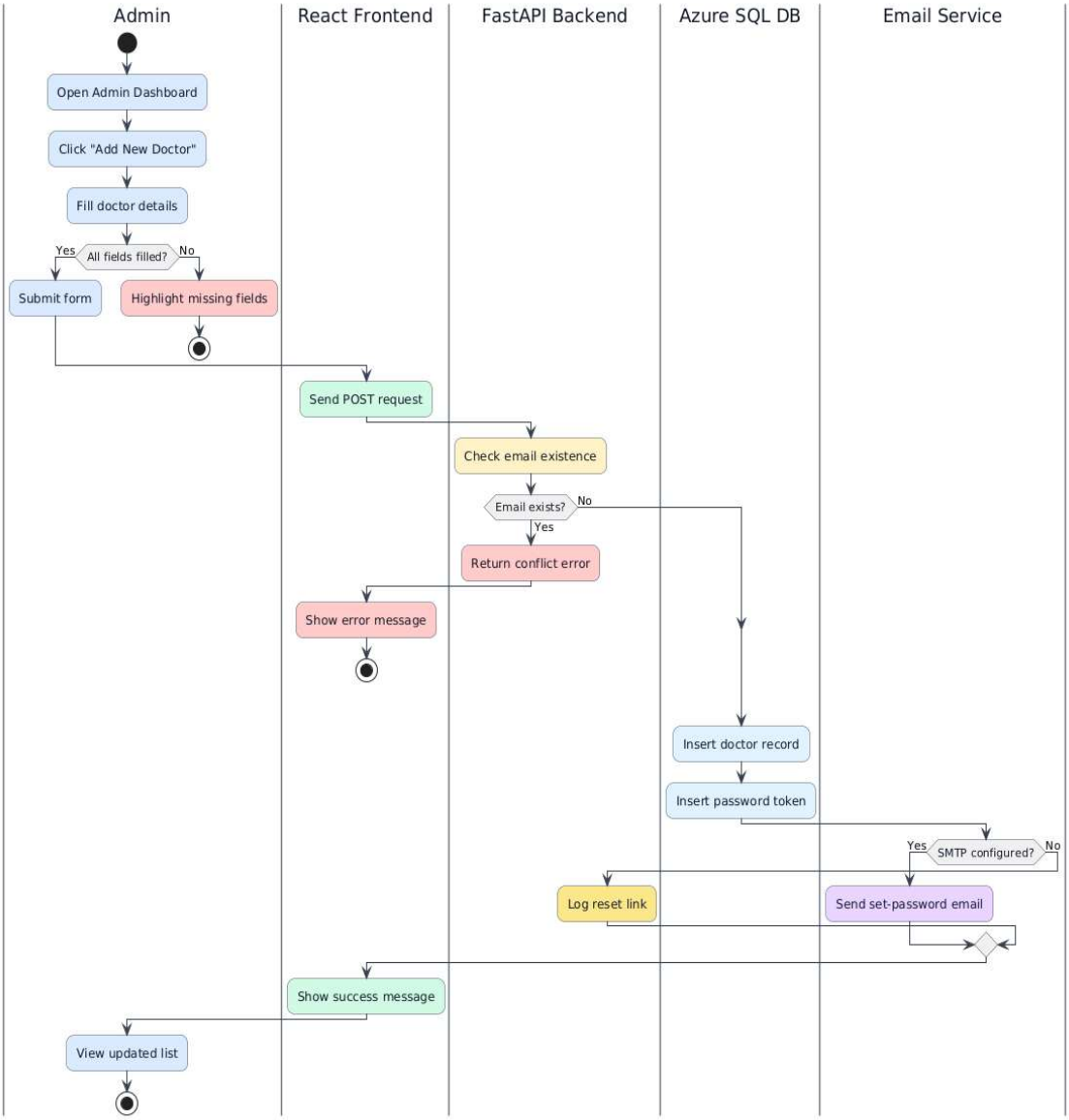

Figure 17: Activity Diagram of Doctor Account Creation

Description:

The activity diagram illustrates the process of creating a new doctor account in the CyberSight system. The admin enters doctor details through the dashboard, and the system validates the required fields before sending the request to the FastAPI backend.The backend checks whether the email already exists in the database. If the email is unique, the system creates the doctor account, generates a secure password setup token, and sends a setup email through the email service. Finally, the frontend displays a success message and updates the doctor list, ensuring secure and efficient account management within the CyberSight system.

4.5.2.5. Login & Authentication

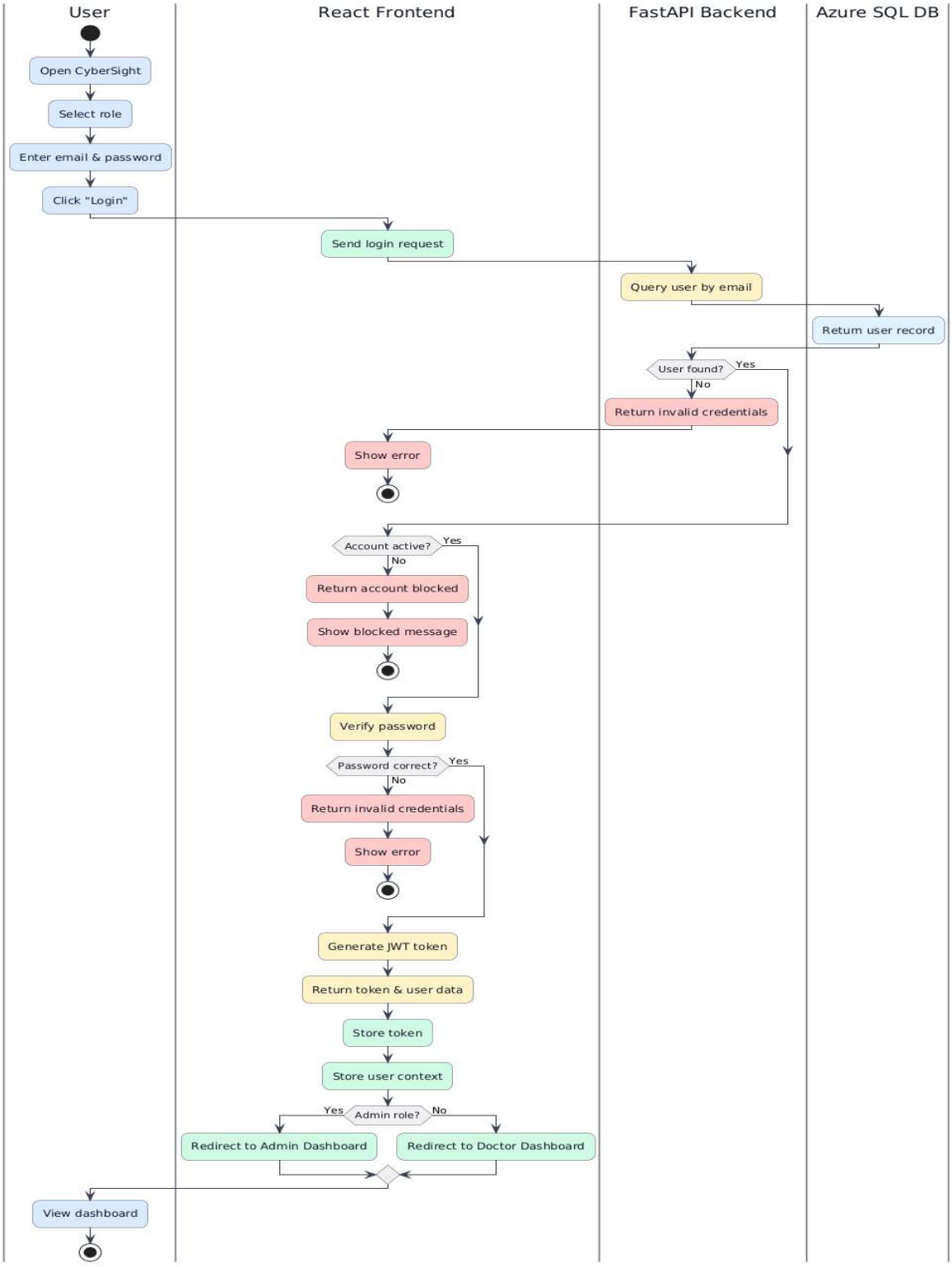

Figure 18: Activity Diagram of Login & Authentication

Description:

The activity diagram illustrates the complete login and authentication workflow in the CyberSight system, demonstrating how users securely access the platform based on their assigned roles and permissions. The process begins when a user opens the application and selects the appropriate role, such as doctor or administrator, from the login interface. The user then enters login credentials including email and password through the React-based frontend. Once the login form is submitted, the front end sends an authentication request to the FastAPI backend for verification and processing.

After receiving the request, the backend communicates with the Azure SQL database to retrieve the corresponding user record associated with the entered email address. The system then performs multiple validation checks to ensure that the login attempt is secure and legitimate. These checks include verifying whether the user account exists, confirming that the account is currently active, and validating the entered password against the securely hashed password stored in the database using strong encryption techniques such as bcrypt. These security measures help protect user credentials and reduce the risk of unauthorized access to sensitive medical information. If any validation step fails, the backend immediately generates an error response and sends it back to the frontend. The frontend then displays a suitable error message to inform the user about invalid credentials, inactive accounts, or unauthorized access attempts. When all authentication checks are successfully completed, the backend generates a JSON Web Token (JWT) containing authenticated user details and role information. The token is securely transmitted to the frontend, where it is stored safely for session management and future authorization requests.

The frontend then uses the received role information to redirect the user to the appropriate dashboard, such as the doctor dashboard or admin dashboard. This role-based routing mechanism ensures that users can only access features and resources permitted for their assigned roles. Overall, this workflow ensures secure authentication, proper access control, protected session management, and smooth user experience within the CyberSight system.

Additionally, the authentication workflow supports secure session handling and protected API communication throughout the system. Every authenticated request sent from the frontend to the backend includes the generated JWT token, allowing the backend to verify the user identity before granting access to protected resources and functionalities. The system also implements token validation and expiration mechanisms to prevent unauthorized session reuse and improve overall application security. By combining encrypted password storage, JWT-based authentication, role-based authorization, and secure backend validation, the CyberSight system maintains confidentiality, integrity, and controlled access to sensitive medical and diagnostic information.The authentication workflow is designed in a modular manner, where frontend validation, backend authentication services, and database verification operate as separate but connected components. This layered authentication structure improves maintainability and simplifies future enhancements related to user management and access control. The workflow also supports scalability by allowing additional user roles and authentication features to be integrated without significantly changing the existing system structure.

4.5.2.6. CLIP Fundus Validation

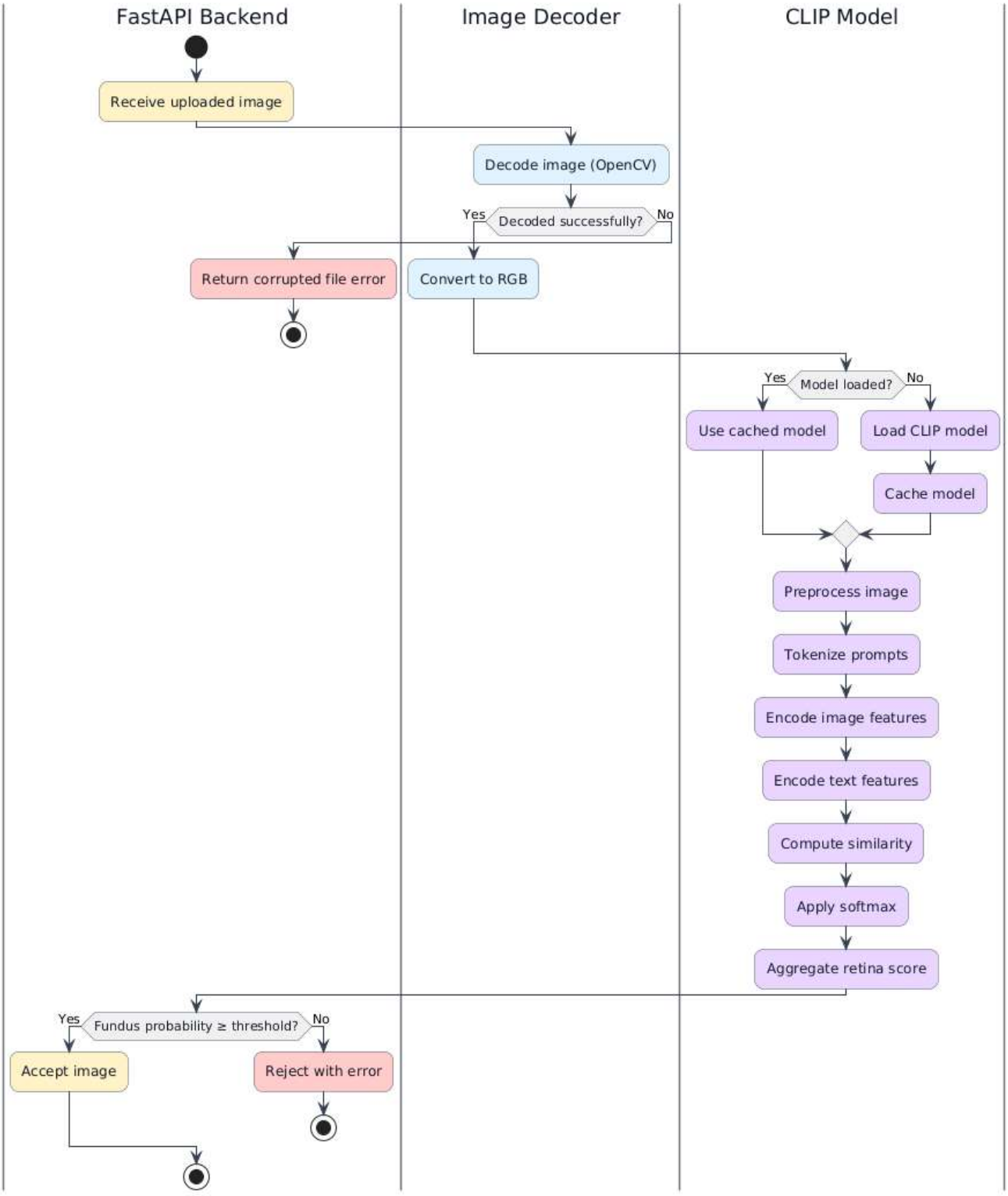

Figure 19: Activity Diagram of CLIP Fundus Validation

Description:

The activity diagram illustrates the CLIP-based fundus validation workflow in the CyberSight system, which ensures that only valid retinal fundus images are accepted for diabetic retinopathy diagnosis. The process begins when the FastAPI backend receives an uploaded image from the frontend interface. The image is first passed to the image processing module, where decoding operations are performed to verify that the file is readable and not corrupted. If the decoding process fails due to unsupported or damaged image data, the system immediately generates an error response and terminates the workflow to prevent invalid inputs from entering the AI pipeline.

Once the image is successfully decoded, the system converts it into the required format for further processing. This includes color conversion, resizing, normalization, and transformation into a format compatible with the CLIP validation model. These preprocessing operations ensure consistency in image quality and improve the reliability of the validation process. The processed image is then forwarded to the CLIP-based validation module for semantic verification.

Before inference begins, the backend checks whether the CLIP model is already loaded into memory. If the model is unavailable, the system loads and caches it dynamically to improve performance for future requests. The image is then preprocessed according to CLIP requirements, while a predefined collection of retinal and non-retinal text prompts is tokenized. These prompts represent various medical and non-medical image categories, allowing the system to compare the uploaded image against expected retinal image descriptions.

The CLIP model encodes both the image and text prompts into feature embeddings and calculates cosine similarity scores between them. These similarity values are processed through a softmax function to generate probability distributions for all prompt categories. The backend then combines the probabilities associated with retinal-related prompts to determine whether the uploaded image is a valid retinal fundus image. This probabilistic approach allows the system to perform zero-shot validation without requiring an additional custom classification model.

If the calculated retinal probability exceeds the predefined threshold, the image is accepted and forwarded to the diabetic retinopathy prediction pipeline for AI inference. However, if the probability falls below the threshold, the image is rejected and the system returns a suitable validation error message to the frontend. This message informs the user that the uploaded image is not a valid retinal fundus image and requests another upload.

This validation workflow significantly improves the robustness, reliability, and security of the CyberSight system by filtering irrelevant or incorrect inputs before diagnosis. It helps reduce prediction errors caused by unrelated images and ensures that the AI model processes only clinically relevant retinal images. Overall, the CLIP-based validation mechanism enhances diagnostic accuracy, strengthens system integrity, and contributes to a safer and more trustworthy AI-assisted healthcare environment.

4.5.2.7. Admin Doctor Management

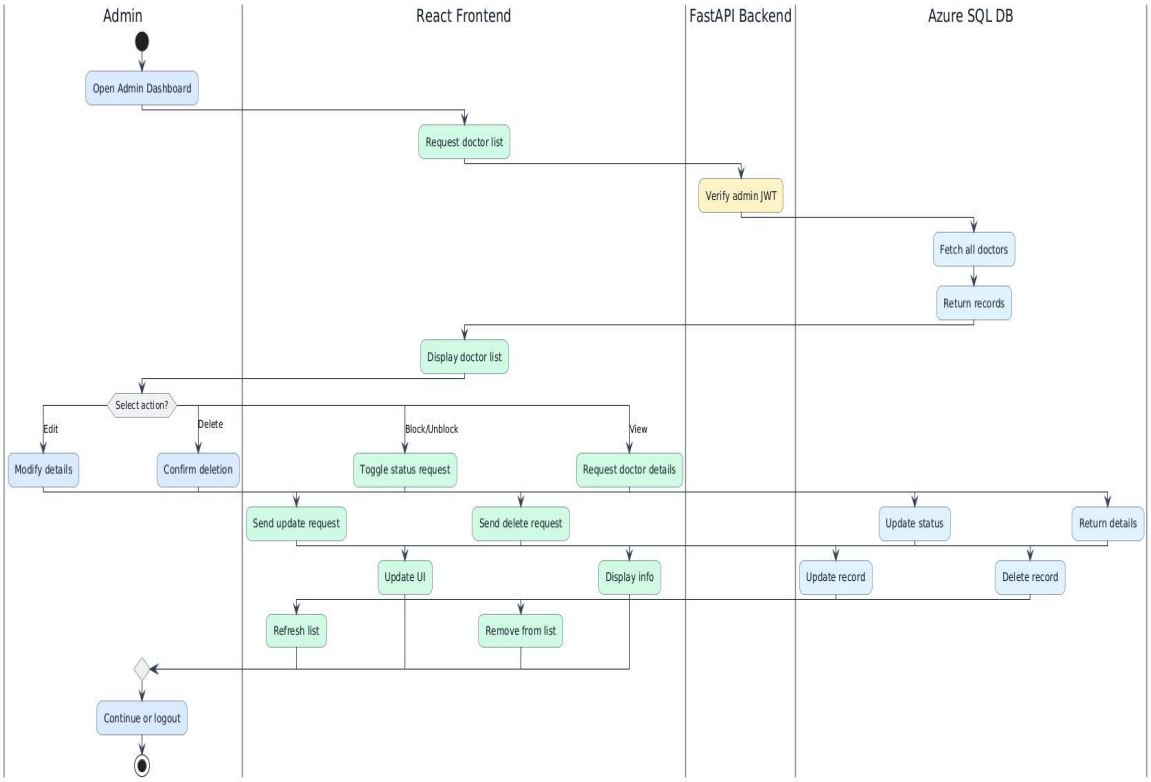

Figure 20: Activity Diagram of Admin Doctor Management

Description:

The workflow of the CyberSight system shows the workflow of the work of the administrators in terms of managing doctor accounts using the interface of the system, as shown in this activity diagram. The user can start by opening the dashboard, where the backend gets the list of registered doctors and the frontend requests it. The backend authenticates the admin provided with a JWT token and gets the doctor's records out of the database. These are then shown on the front end, and the admin can view and manage all the doctor's accounts effectively.

The administration is able to do various things such as updating doctor information, blocking or unblocking accounts, removing records or accessing specific information about a doctor. Every action makes a respective request to the backend, which makes changes to the database. The interface is subsequently updated or refreshed by the front end as the changes take place. The following workflow has guaranteed role-based control of doctor accounts and the full control of system users in the CyberSight system by administrators.

4.5.2.8. End-to-End System Flow

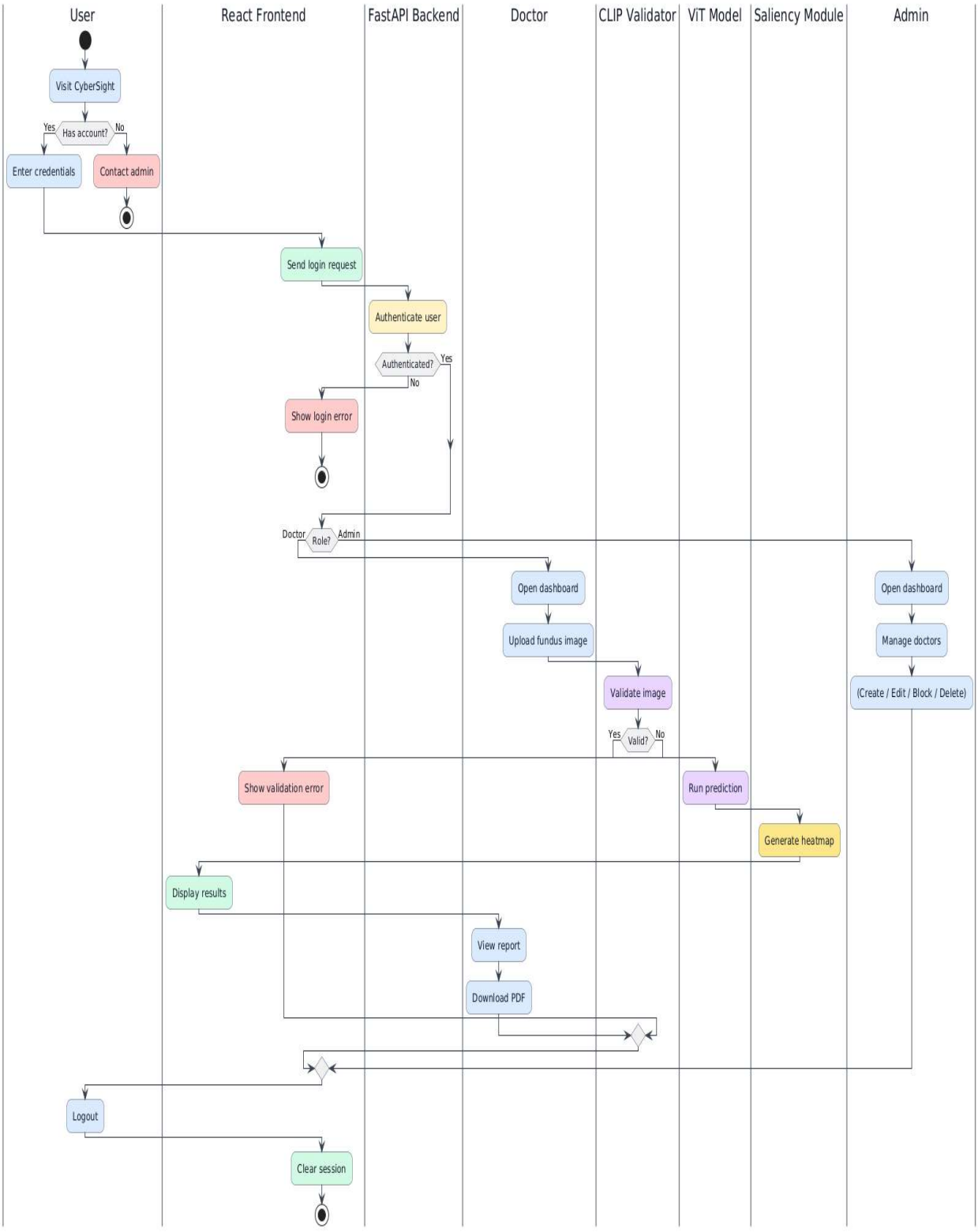

Figure 21: Activity Diagram of End-to-End System Flow

Description:

The activity diagram illustrates the complete end-to-end workflow of the CyberSight system, demonstrating how both doctors and administrators interact with the platform to perform clinical and management-related tasks. The process begins when a user accesses the application and logs in through the React-based frontend. The entered credentials are sent to the FastAPI backend, where the authentication module verifies the user by checking database records, account status, and encrypted password information. If authentication fails, the system displays an error message; otherwise, the user is redirected to the appropriate dashboard according to the assigned role.

For doctors, the workflow mainly focuses on diabetic retinopathy diagnosis using retinal fundus images. The doctor uploads a retinal image through the frontend interface, after which the system performs multiple validation steps including file format verification, image quality checks, and CLIP-based retinal fundus validation. These validations ensure that only valid and clinically relevant retinal images are accepted into the AI pipeline. Once validated, the image is preprocessed through resizing, normalization, and tensor conversion before being passed to the Vision Transformer (ViT) model for prediction.

The Vision Transformer model analyzes the retinal image and predicts the diabetic retinopathy severity stage. The backend then converts the prediction probabilities into diagnosis labels, confidence scores, and severity levels. To improve transparency and interpretability, the system can also generate explainability heatmaps using an occlusion-based saliency module, allowing doctors to visually understand which retinal regions influenced the prediction. The final results are displayed on the frontend, where doctors can review the diagnosis and generate downloadable PDF reports for clinical documentation and patient consultation.

The CyberSight system also includes secure session handling and role-based access control mechanisms to ensure that only authorized users can access protected functionalities. Administrators can manage doctor accounts, including account creation, activation, blocking, and deletion through the admin dashboard. When users log out, the system securely clears authentication tokens and session data to maintain platform security and protect sensitive healthcare information.

In addition, the activity workflow demonstrates how different system components coordinate with each other to ensure smooth and continuous operation throughout the diagnostic process. The frontend, backend, AI modules, and database remain interconnected through structured API communication and organized service layers.

The workflow also supports efficient handling of image processing, prediction generation, and report management without interrupting the user experience. Furthermore, the modular workflow structure allows future integration of additional diagnostic features, advanced explainability methods, and support for other retinal diseases. This makes the CyberSight activity architecture flexible, maintainable, and suitable for future healthcare AI enhancements.

4.6 Component Diagram:

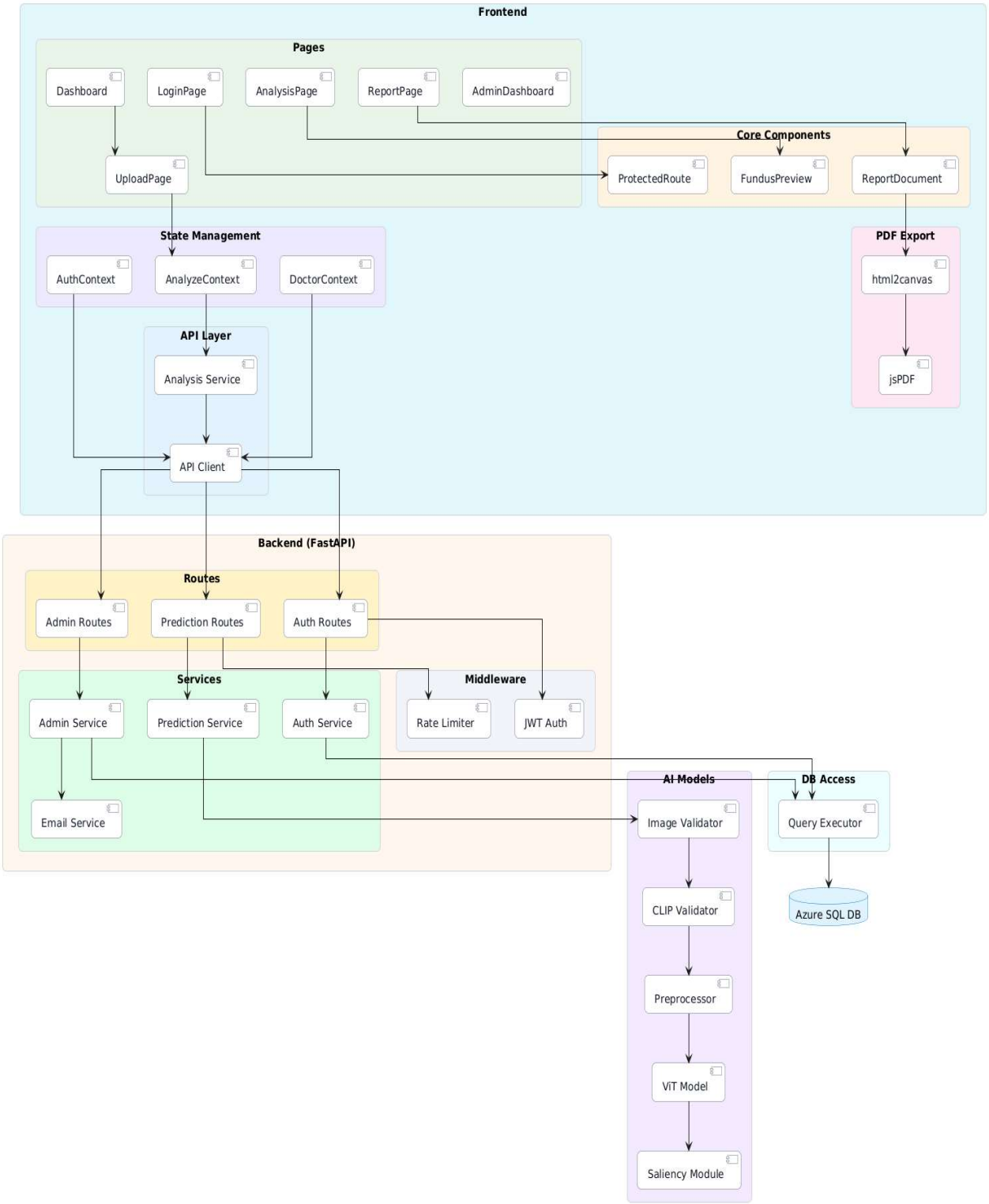

Figure 22: Component Diagram

Description:

The simplified component diagram represents the overall architecture of the CyberSight system while preserving all major components required to understand the system workflow and functionality. The system is divided into frontend, backend, AI processing modules, and database components, which together provide a modular and scalable healthcare application. This simplified architecture reduces unnecessary complexity while still clearly explaining how data flows between different parts of the system during authentication, image analysis, report generation, and administrative management.

The React-based frontend contains essential pages such as login, dashboard, image upload, analysis, and reporting modules. It also includes reusable components responsible for authentication handling, retinal image previews, heatmap visualization, and report rendering. State management within the frontend is maintained through context modules, which allow different parts of the application to securely share user session data, uploaded image information, and diagnostic results. API communication between the frontend and backend is managed using a centralized API client and analysis service, which simplifies request handling and improves maintainability.

The FastAPI backend serves as the core processing layer of the CyberSight system and is organized into routes, services, middleware, and utility modules. The routing layer manages incoming requests from the frontend, while service modules implement business logic such as authentication, image validation, prediction handling, and report generation. Middleware components provide additional functionality including request validation, error handling, token verification, and secure session management. This layered backend architecture improves modularity, security, and separation of responsibilities throughout the system.

The backend also communicates with the AI processing pipeline responsible for diabetic retinopathy diagnosis. The AI workflow begins with image validation and preprocessing modules that prepare retinal fundus images for deep learning inference. The processed image is then passed to the Vision Transformer (ViT) model, which predicts the diabetic retinopathy severity stage based on retinal image features. The system can also activate an explainability module that generates saliency heatmaps using occlusion-based methods to visually highlight important retinal regions contributing to the prediction. In addition to AI processing, the backend interacts with the Azure SQL database through a dedicated query execution layer. The database stores user credentials, authentication information, doctor account details, and other system-related metadata required for secure platform operation. This integration ensures reliable user management, authentication handling, and role-based access control within the system.

Overall, the simplified component architecture provides a clear, maintainable, and scalable design suitable for both practical implementation and academic system documentation. The separation of frontend, backend, AI modules, and database components improves system organization and simplifies future development activities. The architecture also supports easier integration of new features and AI functionalities without affecting the overall workflow of the system.

4.7 Data Models

4.7.1 ER Diagram:

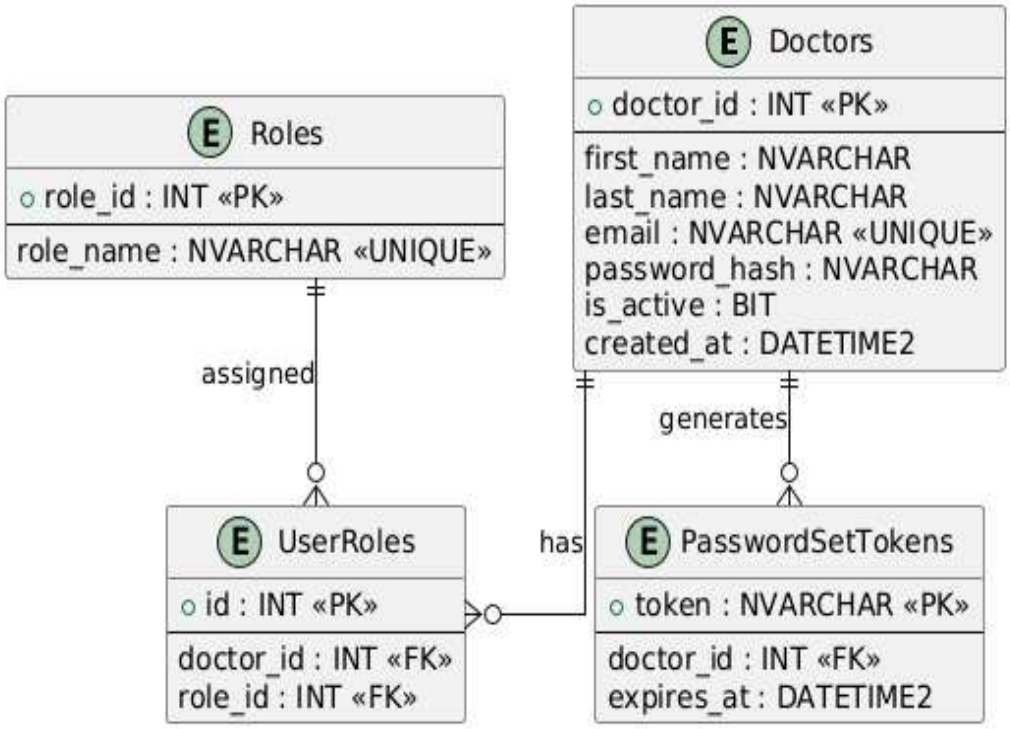

Figure 23: ER Diagram

Description:

The CyberSight ER diagram represents a secure and structured relational database designed for user management, authentication, and role-based access control. The Doctors entity is the central table and stores information such as doctor_id, name, email, password hash, account status, and creation timestamp. Passwords are stored in hashed form to improve security and protect user credentials.

The system supports role-based access control through the Roles entity, which defines user types such as Doctor and Administrator. Since one doctor can have multiple roles, a many-to-many relationship is implemented using the UserRoles junction table. This structure improves scalability and allows new roles to be added easily without modifying the database design.

The database also includes the PasswordSetTokens entity for secure password setup and reset functionality. Each token is linked to a specific doctor and contains an expiration time for improved security. Foreign key constraints and cascade delete operations maintain referential integrity and prevent orphaned records. Overall, the ER diagram demonstrates a normalized, scalable, and secure database design for the CyberSight system.

4.8 User Interface Design

4.8.1 Screenshot of Landing Page

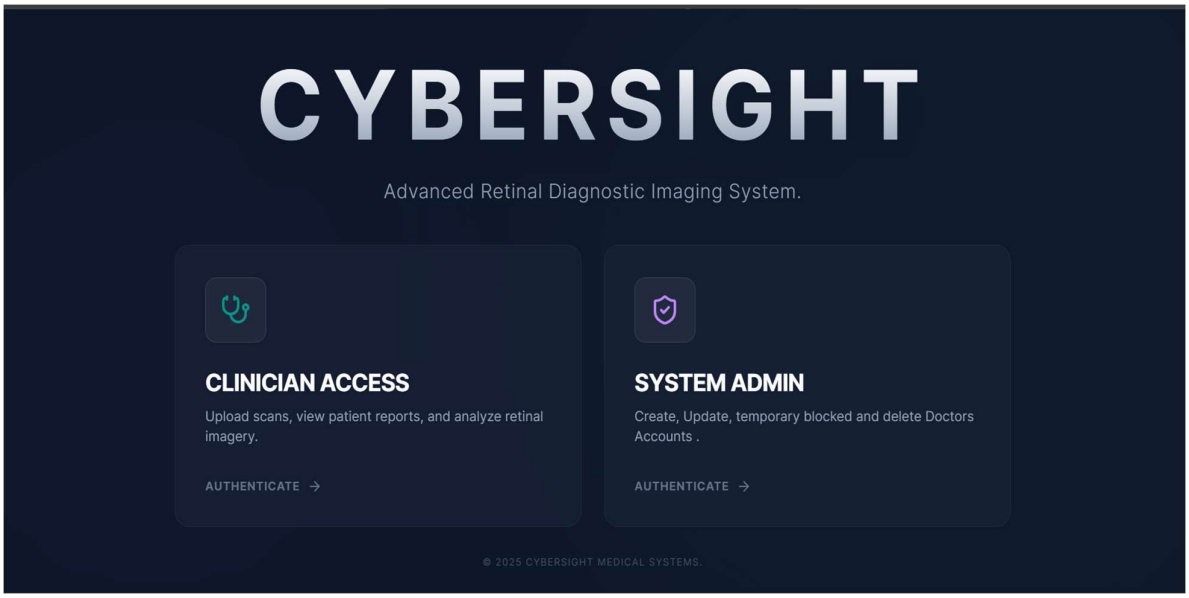

Figure 24: Screenshot of Landing Page

4.8.2 Screenshot of Login Page (Doctor Dashboard)

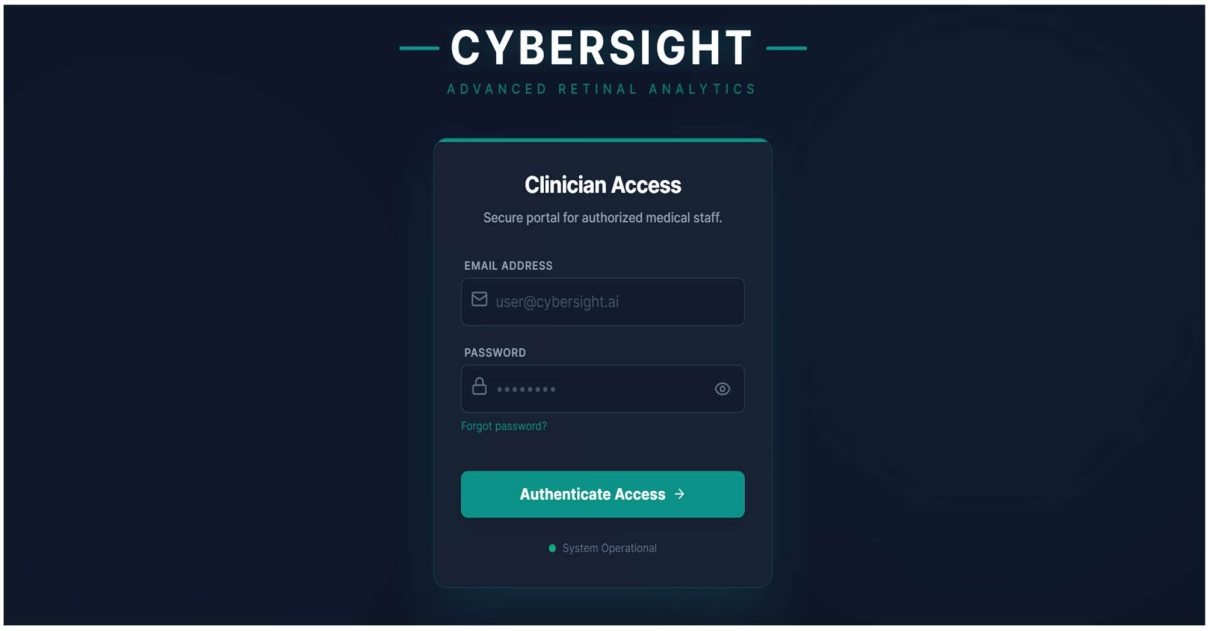

Figure 25: Screenshot of Login Page (Doctor)

4.8.3 Screenshot of Login Page (Admin Dashboard)

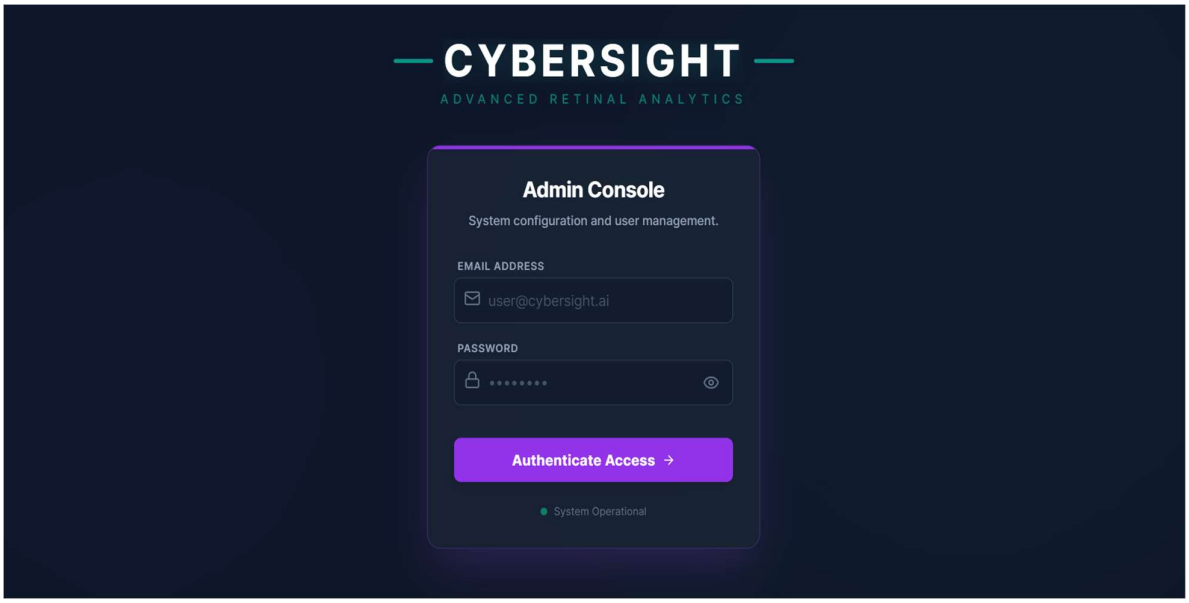

Figure 26: Screenshot of Login Page (Admin)

4.8.4 Screenshot of Dashboard Page (Doctor)

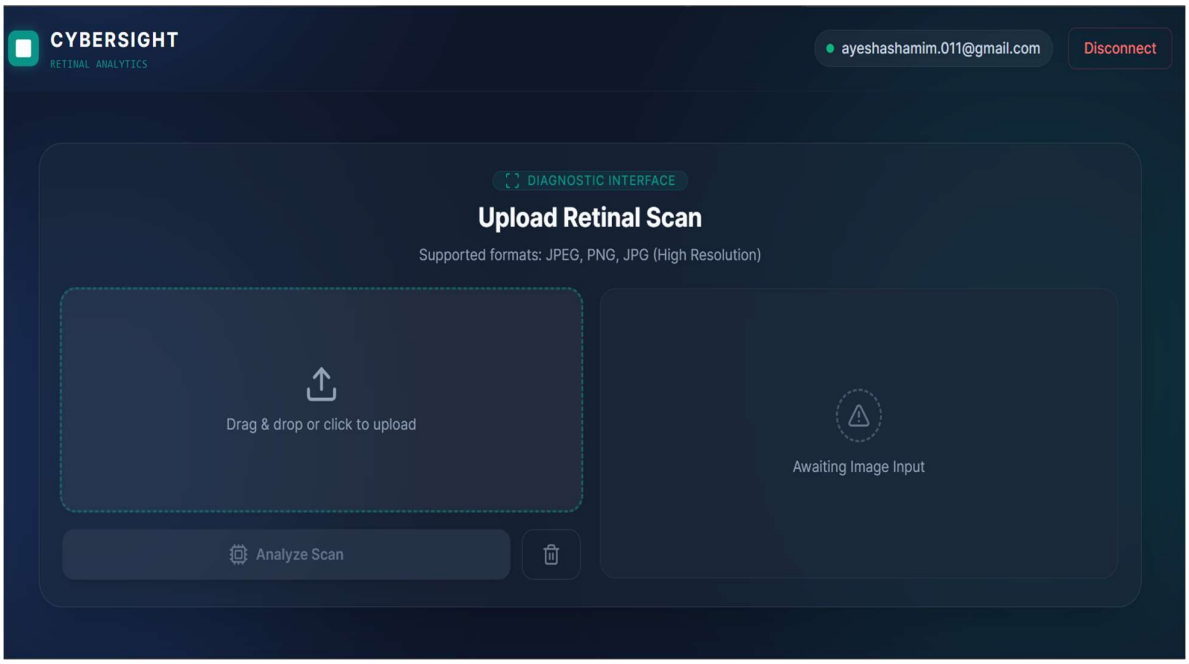

Figure 27: Screenshot of Doctor Dashboard Page

4.8.5 Screenshot of Dashboard Page after Image Upload

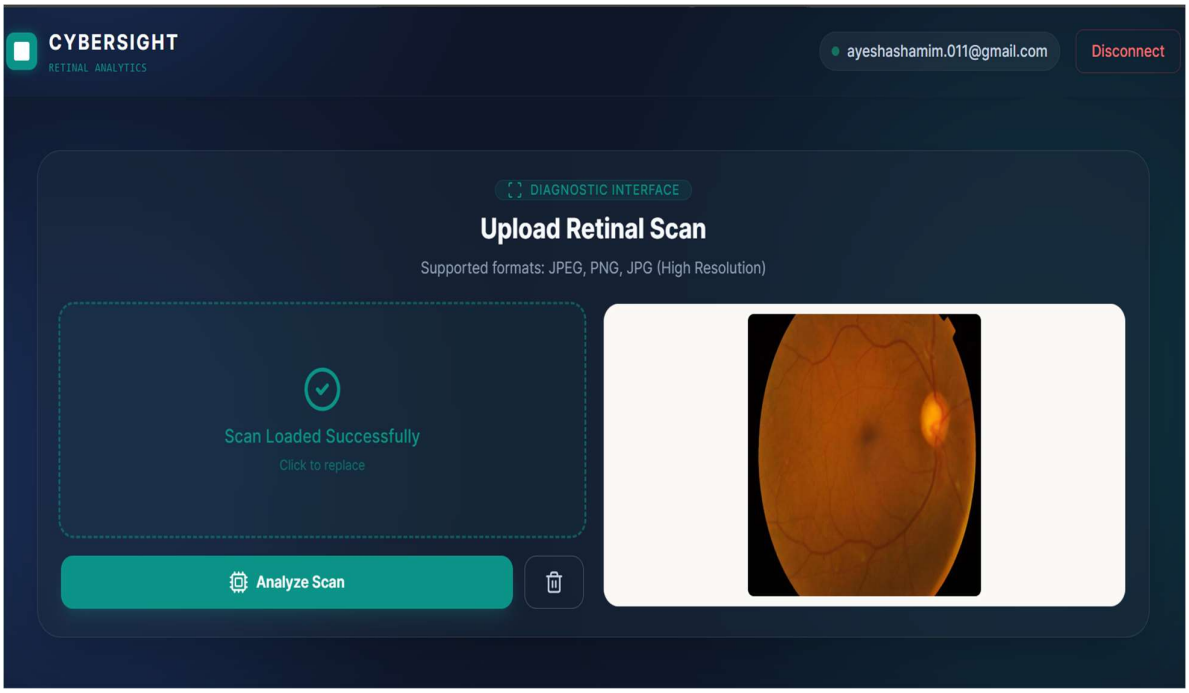

Figure 28: Dashboard page After Image Upload

4.8.6 Screenshot of Dashboard Page during Image Analysis

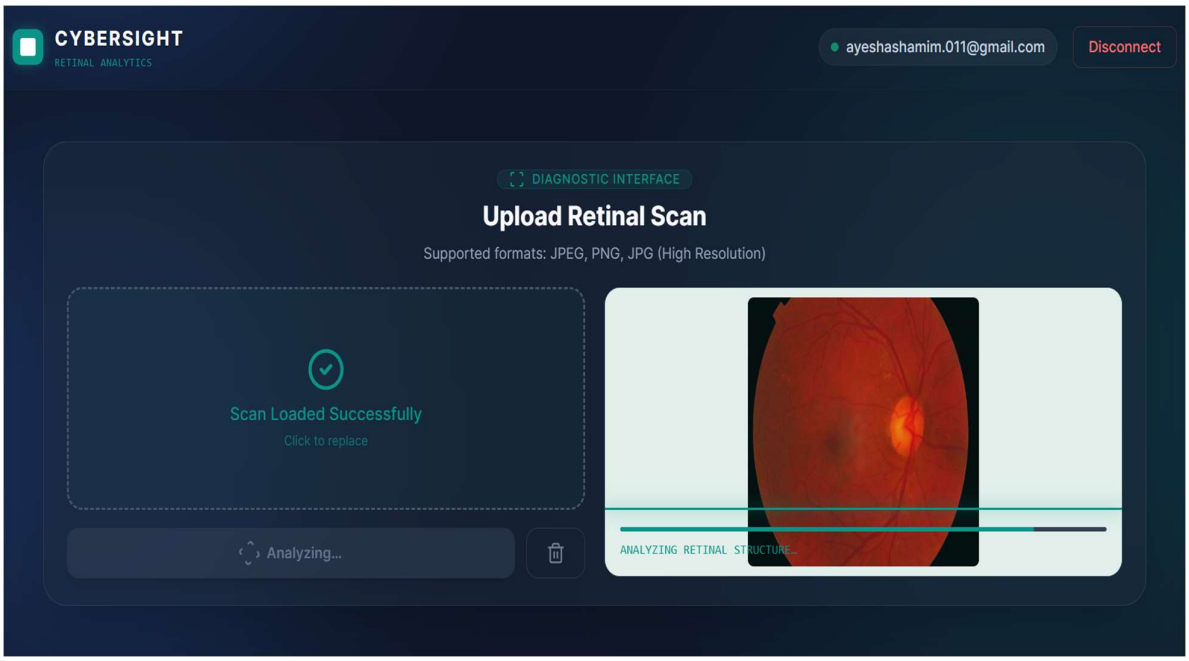

Figure29: Dashboard page During Image Analysis

4.8.7 Screenshot of Analysis Page

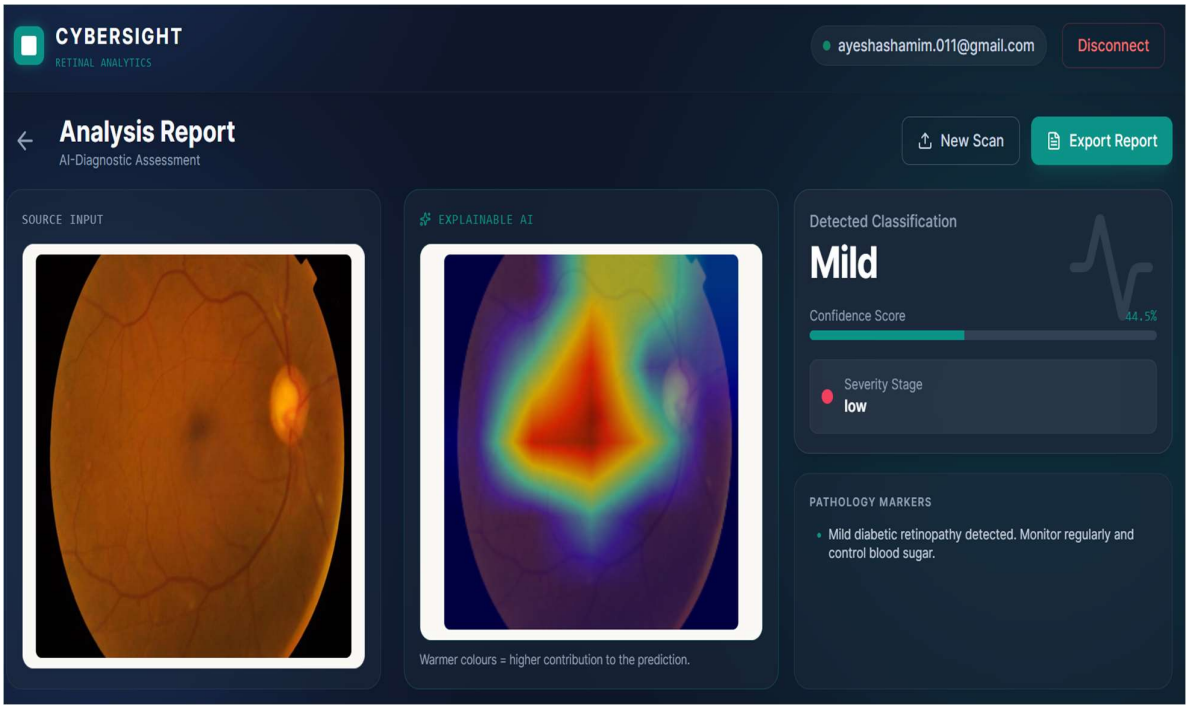

Figure 30: Screenshot of Analysis Page

4.8.8 Screenshot of Report Page

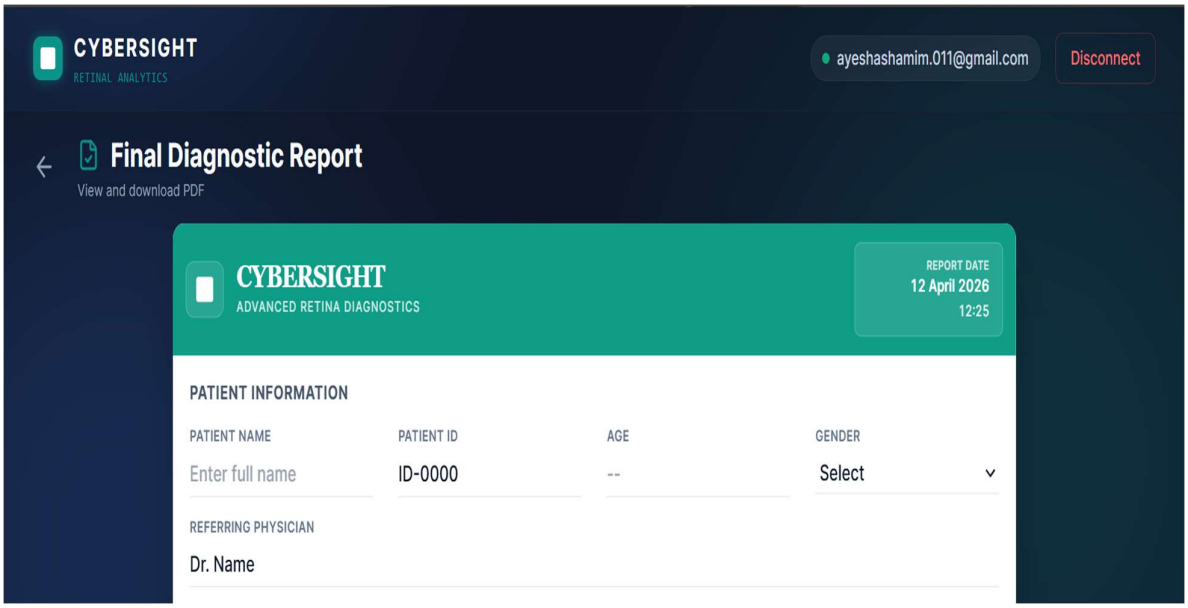

Figure 31: Screenshot of Report Page A

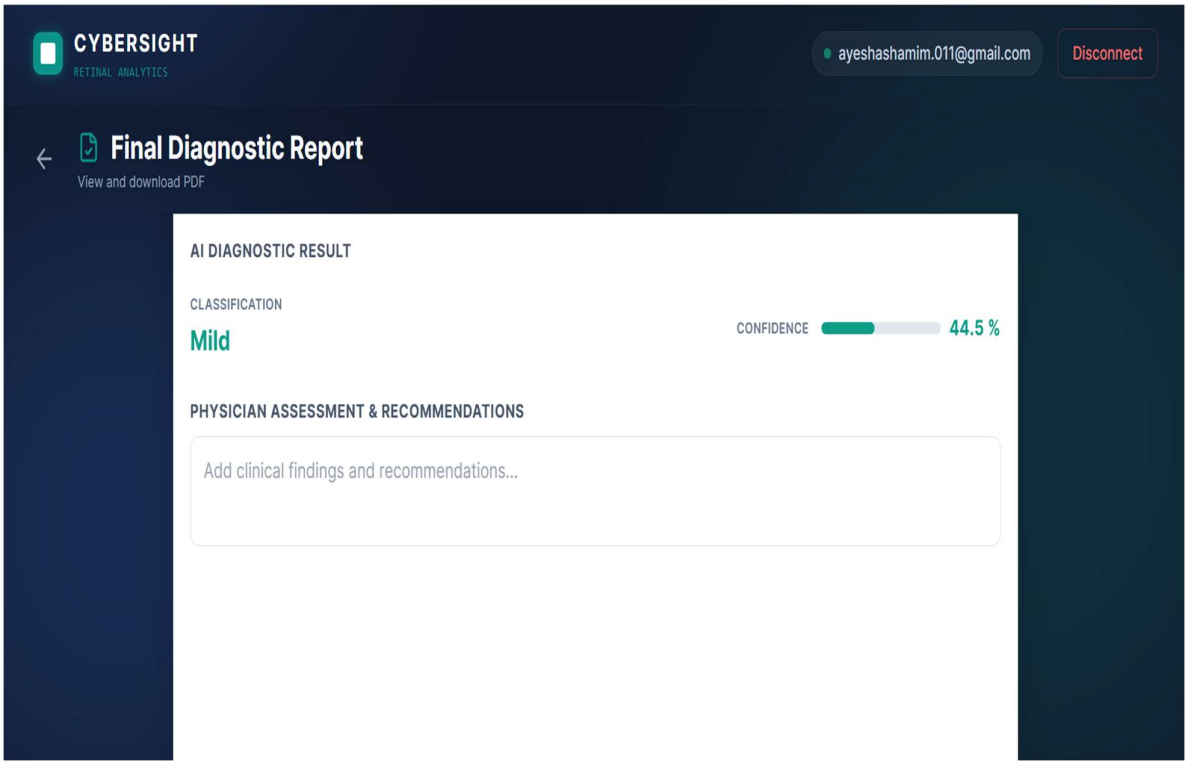

Figure 32: Screenshot of Report Page B

4.8.9 Screenshot of Dashboard Admin

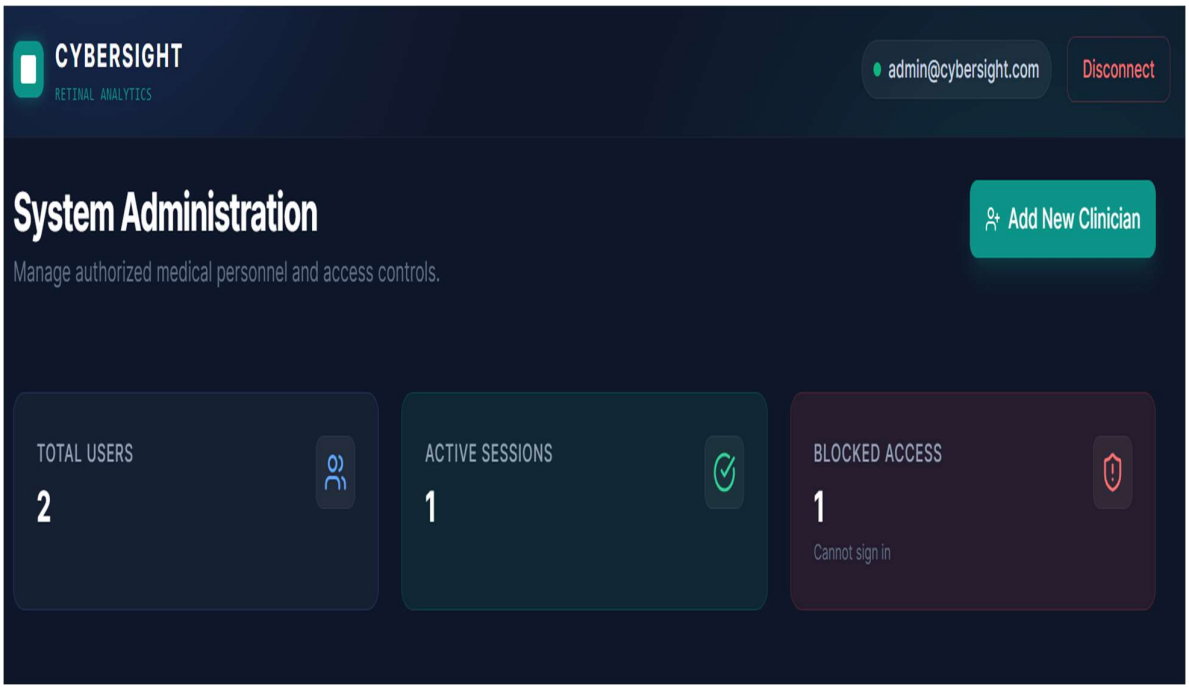

Figure 33: Screenshot of Admin Dashboard Admin

4.8.10 Screenshot of List of Doctors

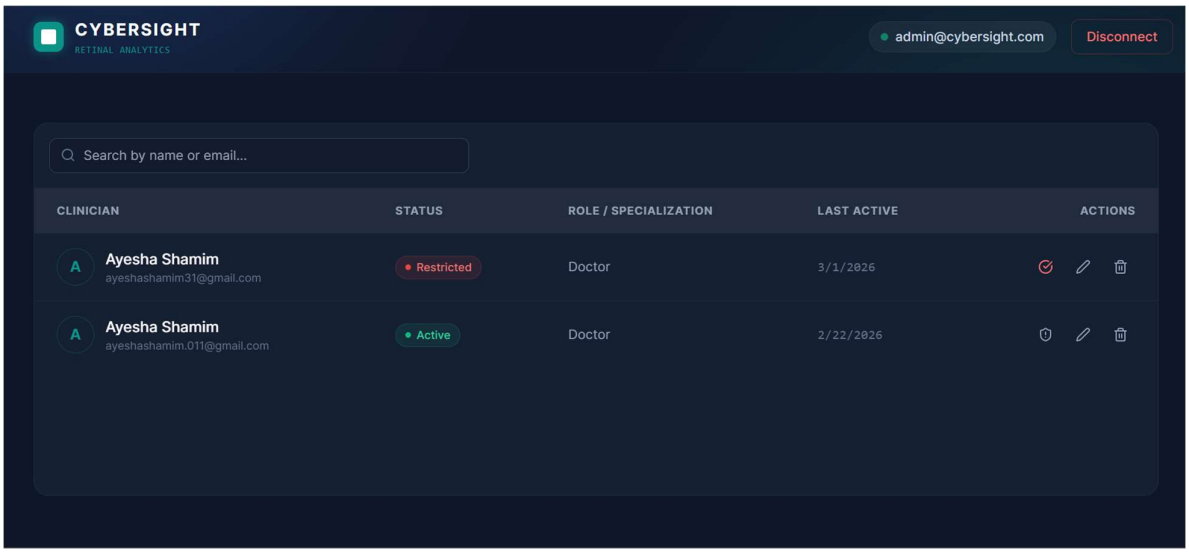

Figure 34: Screenshot of List of Doctors

4.8.11 Screenshot of Add New Doctor

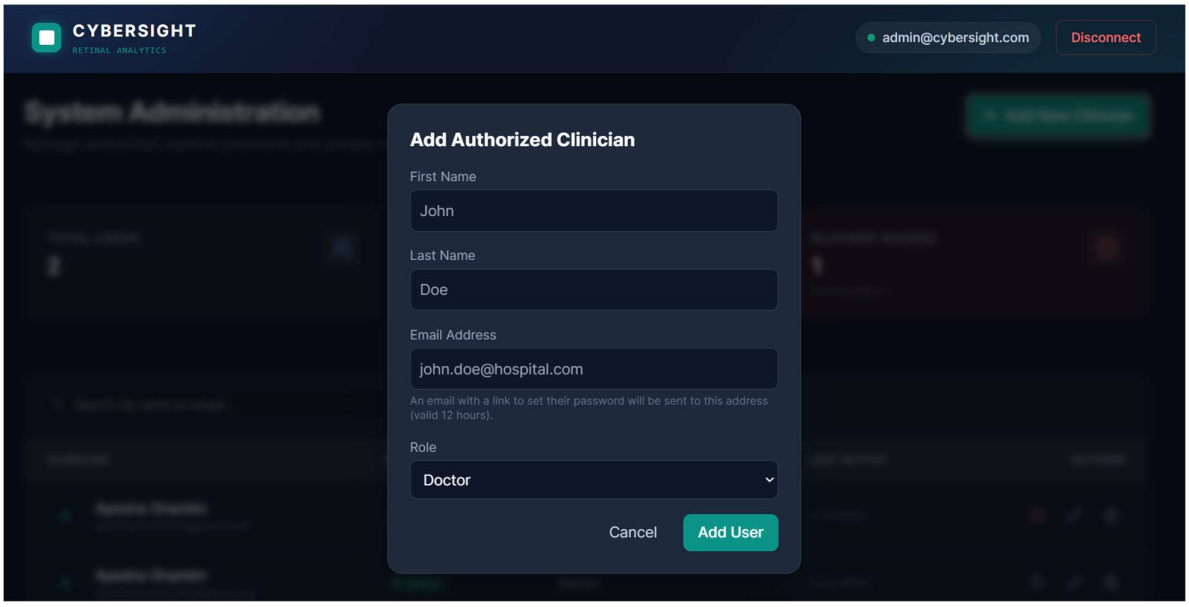

Figure 35: Screenshot of Add New Doctor

4.9 Conclusion

In this chapter, we have shown the design approach, system architecture, and user interfaces of our application.

Chapter # 5

System Implementation

Chapter no. 5

System Implementation

In this chapter, the implementation phase of the CyberSight system is described. The instruments and methods, which were employed in the creation and the successful implementation of the project, are discussed. The system was made to allow AI-based detection and grading of diabetic retinopathy based on the retinal fundus images, explainable AI-generated heatmap, detailed clinical report, and role-based access to doctors and administrators. It also provides image validation through CLIP based fundus validation, generation of PDF reports and secure authentication. Once fully implemented, the system has all functionalities as mentioned in the preceding chapters.

5.1 Strategy:

The CyberSight system was developed using a combination of the Incremental Model and Agile Development Methodology. Development started with core modules such as the Vision Transformer (ViT) training pipeline and FastAPI backend, followed by additional features including the CLIP-based fundus validator, explainable AI heatmaps, authentication system, admin dashboard, and PDF report generation in iterative cycles.

Initially, the deep learning model was trained using preprocessing, augmentation, and fine-tuning techniques on the EyePACS-APTOS-Messidor dataset. After successful model integration, RESTful APIs were developed for image upload and prediction. The system was later enhanced with CLIP-based image validation and occlusion-based saliency heatmaps for explainability.

In later stages, JWT-based authentication, password reset functionality, doctor management, and the React frontend were implemented. The frontend included role-based routing, image upload, result visualization, and responsive UI design using Tailwind CSS and Framer Motion. Continuous testing and feedback helped improve the overall functionality and usability of the CyberSight system.

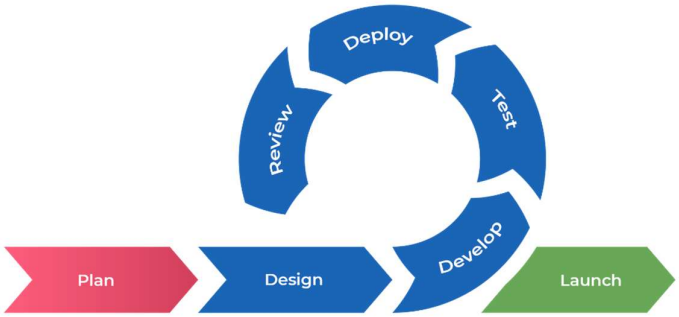

Figure 36: System Implementation

5.2 Tools and Technologies:

5.2.1 Development Tools

Tools	Version	Rationale
MS Word	2016	Documentation
MS PowerPoint	2016	Presentations
Visual Studio Code	1.86	Code Editor
Visual Paradigm	17.0	UML Diagrams Design
Git & GitHub	Latest	Version Control
Postman	Latest	API Testing
Figma	Web-Based	UI/UX Design

Table 23: Development Tools

5.2.2 Frontend Technologies

Technologies	Version	Rationale
React	18.2.0	Frontend SPA Development
TypeScript	4.9.5	Type-Safe JavaScript Superset
React Router DOM	6.22.3	Client-Side Routing
Tailwind CSS	3.4.18	Utility-First CSS Framework
Framer Motion	12.23.24	Page Transitions & Animations
Lucide React	0.554.0	Icon Library
html2canvas	1.4.1	HTML-to-Canvas for PDF Export
jsPDF	4.1.0	PDF Report Generation
React Scripts (CRA)	5.0.1	Build Tooling & Dev Server

Table 24: Frontend Technologies

5.2.3 Backend Technologies

Technologies	Version	Rationale
Python	3.12	Backend Programming Language
FastAPI	Latest	High-Performance Async REST API
Uvicorn	Latest	ASGI Server
Pydantic	Latest	Data Validation & Settings Management
PyTorch	Latest	Deep Learning Framework (Inference)
timm	Latest	Pretrained Vision Transformer Models
OpenCV	Latest	Image Processing & Heatmap Generation
OpenCLIP	Latest	CLIP-Based Image Validation
python-jose	Latest	JWT Authentication

Table 25: Backend Technologies

5.2.4 AI/ML Models

Models	Purpose
Vision Transformer (ViT-Base-Patch16-224) via timm	Pretrained backbone fine-tuned on diabetic retinopathy dataset to classify fundus images into 5 grades: No DR, Mild, Moderate, Severe, Proliferative.
CLIP RN50-quickgelu (OpenAI pretrained) via OpenCLIP	Zero-shot image-text matching to validate that uploaded images are retinal fundus photographs before passing them to the disease classifier.
Occlusion-Based Saliency (Custom Implementation)	Explainable AI technique that masks grid regions of the input image and measures confidence drop to generate visual heatmaps explaining predictions.

Table 26: AI/ML Model

5.3 Explanation:

- **MS Word:**
  Used for creating and editing documentation, including the project report and requirement specifications.

- **MS PowerPoint:**
  Used to design slides for presentations during the project proposal, mid-term, and final evaluation.

- **Visual Studio Code:**
  The main IDE used to write the code is selected because of its lightweight interface, the ability to extensively add extensions (Python, TypeScript, Tailwind CSS IntelliSense) and an integrated terminal to run both frontend and backend servers.

- **Visual Paradigm:**
  It is used for drawing UML diagrams like Use Case, Activity, Sequence, and Class diagrams, aiding in system modeling and architectural design.

- **Git & GitHub:**
  Enhanced versioning and collaboration, with team members able to monitor code changes, go through pull requests and handle branches of both the frontend and backend repositories.
- **Postman:**
  A RESTful endpoint validator API testing tool was used to test endpoints written in FastAPI. It aided in keeping the data integrity, authorization (JWT Bearer tokens), prediction responses, and error responses on all API routes.

- **Figma:**
  An online collaborative tool of UI/UX design that was utilized to develop wireframes, prototypes, and visual mockups to the clinical dashboard, upload interface, analysis result page, and administration panel.

- **Kaggle Notebooks:**
  Employed as the training environment with NVIDIA Tesla T4 GPUs. The entire model training process, consisting of dataset loading, dataset augmentation, two-phase training of ViT, evaluation, and checkpoint export was run on Kaggle with the EyePACS-APTOS-Messidor diabetic retinopathy dataset.

- **React + TypeScript + Tailwind CSS + Framer Motion:**
  A dynamic Single Page Application (SPA) based on react and TypeScript was developed, and it offers a static type safety in all its components. Tailwind CSS offered utility-based

styling, a custom brand color palette (teal primary, dark surfaces) and Framer Motion offered animations of page transitions and interactive user interface effects in every corner of the app.

- **React Router DOM:**
  Offered client-side routing with nested route layouts, role-based protected routes (doctor vs admin), and route transitions with AnimatePresence to provide a smooth user experience.

- **html2canvas + jsPDF:**
  Together these libraries enable client-side PDF report generation. html2canvas captures the clinical report HTML, as a canvas image, and jsPDF converts it into a downloadable A4 PDF document for clinical records.

- **React Context API (State Management):**
  Rather than Redux, there are four specific providers involved in the application, each being an instance of the built-in Context API of React: AuthContext(user session, JWT persistence in localstorage), UploadContext (state of uploaded fundus image), AnalyzeContext (results of the prediction and API calls) , and DoctorContext (CRUD operations of the administrator doctor). This slim design does not involve any external dependencies on state management and clean separation of concerns.
- **FastAPI + Uvicorn:**
  FastAPI was selected because of its intrinsic support of asynchronous operations, automatic OpenAPI / Swagger documentation generation, and high speed. Uvicorn is the ASGI server. The API uses versioned endpoints on /api/v1/ tto perform prediction, authentication, and administration with default rate limiting middleware.

- **Pydantic + Pydantic Settings:**
  Pydantic handles request/response schema validation across all API endpoints, while Pydantic Settings manages application configuration through environment variables and .env files, covering server settings, CORS origins, database credentials, JWT secrets, SMTP configuration, and model paths.

- **PyTorch + timm (PyTorch Image Models):**
  PyTorch is used as both the training and inference framework of the deep learning model. The pretrained vit_base_patch16_224 (Vision Transformer Base with 16x16 patches, 224x224 input) backbone is available in the timm library. A two-phase transfer learning approach was used during training: Phase 1 trained only the custom classification head (MLP: 768 512 256 5) with the backbone fixed and Phase 2 fine-tuned the whole model with a lower learning rate. During inference time, the trained .pth checkpoint is loaded and deployed to CPU to make the model easier to deploy.

- **OpenCV (cv2):**
  Applied to the pipeline to perform image I/O (image reading, decoding, resizing), color space operations (BGR RGB), image augmentation in training (flipping, rotating), model input preprocessing, and the explainable AI heatmap overlays with color maps (COLORMAP JET).

- **OpenCLIP (CLIP RN50-quickgelu):**
  Runs a one-shot image validation gate on the CLIP model of OpenAI. Images uploaded are cross-matched with 7 retinal fundus text prompts and 13 non-retina text prompts before any prediction. When the probability of the collective retina prompts is lower than the threshold (0.35), the image is discarded accompanied by a user friendly error message. This helps to avoid non-fundus image misclassification (e.g., selfies, X-rays, or random photos).

- **Occlusion-Based Explainable AI (Saliency Heatmaps):**
  An implementation that is custom, which produces visual explanations of model predictions. The algorithm splits the input image into a 5 x 5 grid (25 cells), masks a cell one at a time and counts the drop in confidence of the predicted class. Areas that masking has decreased the most confidence are identified by warm hues (red/orange) in a JET colormap overlay. It only takes 26 forward passes (25 occluded and 1 baseline) and is efficient on a CPU (about 30-90 seconds per image).

- **python-jose + passlib + bcrypt (Authentication):**
  HS256 signing with expiration of 24 hours (configurable) JWT (JSON Web Token) authentication. Passwords are hashed with bcrypt through passlib and any old plain-text passwords are automatically converted to a hashed password on logging in. The system facilitates an end-to-end password management process such as set-password through email token and forgotten password.

- **pyodbc + Microsoft Azure SQL:**
  Azure SQL is the relational database in cloud which is accessed through pyodbc using ODBC driver 18. Doctor accounts (Doctors table), role assignments (Roles, UserRoles tables), and password set tokens (PasswordSetTokens table) are stored in the database. Raw SQL is accessed via a custom database module containing context-controlled connections, implicit commit/rollback and connection timeouts.

- **SMTP Email Service:**
  Email service Python-based smtplib to send transactional emails with HTML format. It is used in the set-password flow in which new doctor accounts are sent an email containing a secure token link in which to set their password. Can use configurable SMTP providers (Mailtrap, Ethereal, or production SMTP). The set-password link will be logged on the console when SMTP is not configured so that it can be easily developed.

5.4 Model Training Pipeline:

A Kaggle Notebook was used to train the Vision Transformer model, accelerated by NVIDIA Tesla T4. The pipeline of the training was as follows:

- **Dataset:**
  EyePACS-APTOS-Messidor Diabetic Retinopathy dataset with 5 classes (No DR, Mild, Moderate, Severe, Proliferative DR).
- **Preprocessing:**
  - Images resized to 224×224 pixels.
  - Class 0 (No DR) undersampled to 2.5× the size of Class 1 to reduce majority class dominance.
  - Classes 1, 3, and 4 augmented (doubled) using horizontal flipping and random rotation (±15°).
  - Data split into 70% training, 15% validation, and 15% test sets with stratified sampling.
  - Balanced class weights computed for the weighted CrossEntropyLoss function.
- **Model Architecture:**
  - Backbone: vit_base_patch16_224 from timm (pretrained on ImageNet).
  - Custom classification head: Linear(768→512) → ReLU → Dropout(0.4) → Linear(512→256) → ReLU → Dropout(0.3) → Linear(256→5).
- **Two-Phase Transfer Learning:**
  - **Phase 1 (Frozen Backbone):**

    Only the classification head was trained for 5 epochs with learning rate 3×10^{-4} using AdamW optimizer and ReduceLROnPlateau scheduler. Early stopping with patience of 4 epochs.
  - **Phase 2 (Full Fine-Tuning):**

    The entire model (backbone + head) was unfrozen and trained for 5 additional epochs with a lower learning rate of 3×10^{-5}. Best model checkpoint saved based on validation accuracy.
- **Evaluation Metrics:**
  Accuracy, Precision, Recall, Weighted F1-Score, per-class F1 scores, confusion matrix, and classification report were computed on the held-out test set.
- **Outputs:**
  The final trained model checkpoint (vit_model_final.pth) was exported and deployed to the backend server for inference.

5.5 Model Training Pipeline Workflow:

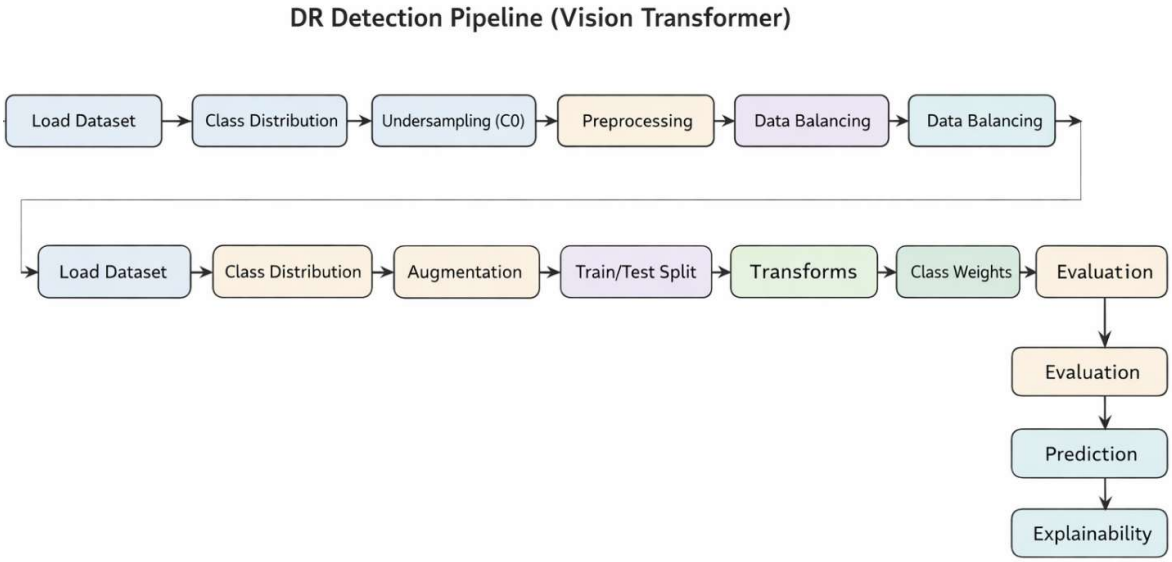

Figure 37: Model Training Pipeline

5.6 Key API End Points:

Method	Endpoint	Purpose
GET	/api/v1/	Health check, API version, available models
GET	/api/v1/models	List registered disease types
GET	/api/v1/models/{disease_type}	Model metadata (classes, image size, device)
POST	/api/v1/predict/{disease_type}	Upload image and get prediction + heatmap
POST	/api/v1/auth/login	Doctor/Admin login (returns JWT)
POST	/api/v1/auth/validate-set-password-token	Validate password setup token

POST	/api/v1/auth/set-password	Set password using email token
POST	/api/v1/auth/forgot-password	Trigger password reset email
GET	/api/v1/admin/doctors	List all doctors (admin only)
POST	/api/v1/admin/doctors	Create new doctor account (admin only)
GET	/api/v1/admin/doctors/{doctor_id}	Get doctor details (admin only)
PATCH	/api/v1/admin/doctors/{doctor_id}	Update doctor information (admin only)
POST	/api/v1/admin/doctors/{doctor_id}/toggle-block	Toggle doctor active status (admin only)
DELETE	/api/v1/admin/doctors/{doctor_id}	Delete doctor account (admin only)

Table 27: API End Points

5.7 Conclusion:

CyberSight was implemented with the help of the latest web technologies, deep learning models, and cloud services. Vision Transformer model was trained on a large diabetic retinopathy dataset with a two-step transfer learning strategy on Kaggle GPU hardware and served through a fastapi backend to make real-time predictions. All modules, including image validation (CLIP) to explainable AI (occlusion saliency) and clinical report generation (PDF export) were built and refined to guarantee stability, accuracy, and user-friendly operation. The system can now be used to detect diabetic retinopathy using AI on retinal fundus images, provide explainable visual heatmaps, create downloadable clinical reports, and deal with doctor accounts using a secure role-based access system.

Chapter # 6

System Testing and Evaluation

Chapter 6

System Testing & Evaluation

In this chapter, we evaluate the CyberSight system and its components according to the expected outcomes. Each module of the system is tested against its expected results to ensure correctness and reliability. Testing is an important phase of the software development life cycle, that ensures the system is free from major errors and meets quality standards. Through testing, we verify that the system performs accurately in detecting diabetic retinopathy from fundus images and provides reliable outputs.

6.1 Test Strategy

Every software system must be tested before deployment to ensure it works as intended. Testing helps improve system reliability, usability, and performance by identifying and fixing bugs. For CyberSight, we followed a **code–implement–test strategy**. Initially, each module, was developed and tested individually to ensure it worked correctly. After that, the module was integrated into the complete system, where more detailed and rigorous testing was performed to verify its functionality within the full application. Different types of testing were carried out depending on the nature of each module. These tests ensured that the system correctly handles image uploads, performs accurate AI-based analysis, generates reports, and provides proper user interaction.

6.2 Component Testing

Component testing involves testing individual modules of the system independently. In CyberSight,, component testing focuses on verifying the functionality of key modules such as:

- User authentication (login, logout, password management)
- Fundus image upload module
- Image validation using AI (CLIP model)
- Diabetic retinopathy prediction module
- Heatmap generation (Grad-CAM visualization)
- Report generation module
- Admin panel for managing users/doctors

Each component is tested separately using sample images and mock data to ensure it behaves correctly. This helps identify issues early before integrating the modules into the full system.

6.3 Unit Testing

Unit testing is the first level of testing, where individual functions and methods are tested in isolation. In CyberSight, unit testing ensures that each small part of the system works correctly.

Key units tested include:

- API endpoints for image upload, prediction, and report generation
- Functions handling image preprocessing and validation
- AI model functions for diabetic retinopathy classification
- Heatmap generation functions (Grad-CAM)
- User authentication functions (login, logout, session handling)
- Database operations for storing reports and user data

Each unit is tested using test inputs and expected outputs to verify correct behavior. Edge cases such as invalid images, empty inputs., and system errors are also tested. This improves code reliability and makes debugging easier.

6.4 Integration Testing

Integration testing ensures that different modules of the system work correctly when combined. In CyberSight, this testing verifies the interaction between frontend, backend, and AI models.

The following integrations are tested:

- Frontend (React.js) communicating with Backend (Flask/Node.js) through APIs
- Backend interaction with the database for storing reports and user data
- Integration of AI models for image classification and validation
- Image upload module working with prediction and heatmap generation
- Report generation using prediction results and visualizations

Test cases are designed to ensure smooth data flow between modules, correct API responses, and proper system behavior, when all components are combined. This testing helps identify issues in communication between modules.

6.5 System Testing

System testing evaluates the complete CyberSight system as a whole to ensure it meets all requirements and works correctly in real-world scenarios.

Key areas tested include:

- Complete workflow: login, upload image, analyze, view result and generate report
- Fundus image upload and validation process
- AI-based diabetic retinopathy prediction accuracy
- Heatmap visualization for explainability
- Report generation and download functionality
- User authentication and session management
- Admin functionalities such as managing doctors/users

System testing ensures that all components of CyberSight work together seamlessly and deliver accurate, reliable, and user-friendly results.

6.6 Test Cases

Test Case: 01

Test Case	**01**
Objective	**Login with Valid Credentials**
Pre-Condition	Login page should be open
Flow	• User enters valid email • User enters valid password • Clicks Login button
Expected Output	User is successfully logged in and dashboard is displayed
Actual Output	User logged in successfully
Status	Passed

Table 28: Test Case 01

Test Case: 02

Test Case	**02**
Objective	**Login with Invalid Password**
Pre-Condition	Login page should be open
Flow	• User enters valid email • User enters incorrect password • Clicks Login
Expected Output	Error message "Invalid credentials" displayed
Actual Output	Error displayed
Status	Passed

Table 29: Test Case 02

Test Case: 03

Test Case	**03**
Objective	**Login with Empty Fields**
Pre-Condition	Login page should be open
Flow	• User leaves fields empty • Clicks Login
Expected Output	Validation messages for required fields
Actual Output	Validation messages displayed
Status	Passed

Test Case: 04

Test Case	**04**
Objective	**Forgot Password with Valid Email**
Pre-Condition	Forgot password page open
Flow	• User clicks "Forgot Password" • Enters registered email • Clicks Submit.
Expected Output	Password reset link sent to email
Actual Output	Reset link sent successfully
Status	Passed

Table 31: Test Case 04

Test Case: 05

Test Case	**05**
Objective	**Forgot Password with Invalid Email**
Pre-Condition	Forgot password page open
Flow	• User enters unregistered email • Clicks Submit
Expected Output	Error message “Invalid Email”
Actual Output	Error message displayed
Status	Passed

Table 32: Test Case 05

Test Case: 06

Test Case	**06**
Objective	**Logout Functionality**
Pre-Condition	User is logged in
Flow	User clicks Logout button
Expected Output	User redirected to login page
Actual Output	User redirected to login page successfully
Status	Passed

Table 32: Test Case 06

Test Case: 07

Test Case	**07**
Objective	**Upload Valid Fundus Image**
Pre-Condition	User logged in, dashboard open
Flow	□ User clicks Upload □ Selects JPG/PNG image □ Confirms upload
Expected Output	Image uploaded and preview displayed
Actual Output	Image uploaded successfully and preview displayed
Status	Passed

Table 33: Test Case 07

Test Case: 08

Test Case	**08**
Objective	**Upload Non-Fundus Image**
Pre-Condition	User logged in
Flow	• User clicks Upload • Selects random non fundus image • Confirms upload
Expected Output	Error "Invalid Image"
Actual Output	Error message displayed for invalid image
Status	Passed

Table 34: Test Case 08

Test Case: 09

Test Case	**09**
Objective	**Analyzing Image Successfully**
Pre-Condition	Valid fundus image uploaded
Flow	User clicks “Analyze” button
Expected Output	System processes image and shows prediction result
Actual Output	Image analyzed successfully and result displayed
Status	Passed

Table 35: Test Case 09

Test Case: 10

Test Case	10
Objective	**Analyzing Without Upload**
Pre-Condition	User logged in
Flow	User clicks "Analyze" without uploading image
Expected Output	Error message displayed
Actual Output	System displayed error for missing image
Status	Passed

Table 36: Test Case 10

Test Case: 11

Test Case	11
Objective	**Display DR Prediction**
Pre-Condition	Image analyzed
Flow	User navigates to result page
Expected Output	DR stage displayed
Actual Output	DR stage displayed correctly
Status	Passed

Table 37: Test Case 11

Test Case: 12

Test Case	12
Objective	**Display Confidence Score**
Pre-Condition	Image analyzed
Flow	User views prediction result
Expected Output	Confidence percentage shown
Actual Output	Confidence score displayed correctly
Status	Passed

Table 38: Test Case 12

Test Case: 13

Test Case	13
Objective	**Display Probability Breakdown**
Pre-Condition	Image analyzed
Flow	User checks result details
Expected Output	All class probabilities displayed

Actual Output	Probability breakdown displayed correctly
Status	Passed

Table 39: Test Case 13

Test Case: 14

Test Case	14
Objective	**Display Severity Level**
Pre-Condition	Image analyzed
Flow	User views prediction
Expected Output	Severity label shown
Actual Output	Severity level displayed correctly
Status	Passed

Table 40: Test Case 14

Test Case: 15

Test Case	15
Objective	**Display Clinical Interpretation**
Pre-Condition	Image analyzed
Flow	User checks result section
Expected Output	Interpretation text displayed
Actual Output	Clinical interpretation displayed correctly
Status	Passed

Table 41: Test Case 15

Test Case: 16

Test Case	16
Objective	**Model Failure Handling**
Pre-Condition	System issue simulated
Flow	User clicks Analyze
Expected Output	Error message displayed
Actual Output	System showed prediction error message
Status	Passed

Table 42: Test Case 16

Test Case: 17

Test Case	17
Objective	**Handle Analysis Timeout**
Pre-Condition	Slow processing
Flow	• User uploads image • Waits for response
Expected Output	Timeout message displayed
Actual Output	Timeout handled with proper message
Status	Passed

Table 43: Test Case 17

Test Case: 18

Test Case	18
Objective	**Display Loading Indicator**
Pre-Condition	Image uploaded
Flow	User click Analyze
Expected Output	Loading Spinner shown
Actual Output	Loading indicator displayed during processing
Status	Passed

Table 44: Test Case 18

Test Case: 19

Test Case	19
Objective	**View Heatmap**
Pre-Condition	Analysis completed
Flow	Users click “View Heatmap”
Expected Output	Heatmap displayed
Actual Output	Heatmap displayed successfully
Status	Passed

Table 45: Test Case 19

Test Case: 20

Test Case	20
Objective	**Heatmap Overlay on Image**
Pre-Condition	Heatmap generated
Flow	User views image
Expected Output	Heatmap overlay shown on image
Actual Output	Heatmap overlay displayed correctly
Status	Passed

Table 46: Test Case 20

Test Case: 21

Test Case	21
Objective	**Heatmap Accuracy**
Pre-Condition	Analysis completed
Flow	User reviews highlighted areas
Expected Output	Important regions highlighted
Actual Output	Relevant regions highlighted correctly
Status	Passed

Table 47: Test Case 21

Test Case: 22

Test Case	**22**
Objective	**Heatmap Load Time**
Pre-Condition	Heatmap requested
Flow	User opens heatmap
Expected Output	Heatmap loads properly
Actual Output	Heatmap loaded without delay issues
Status	Passed

Table 48: Test Case 22

Test Case: 23

Test Case	**23**
Objective	**Heatmap Failure Handling**
Pre-Condition	System error
Flow	User clicks heatmap
Expected Output	Error message displayed
Actual Output	Heatmap failure handled with message
Status	Passed

Table 49: Test Case 23

Test Case: 24

Test Case	**24**
Objective	**Toggle Heatmap View**
Pre-Condition	Heatmap available
Flow	User enables/disables heatmap
Expected Output	Toggle works correctly
Actual Output	Heatmap toggled successfully
Status	Passed

Table 50: Test Case 24

Test Case: 25

Test Case	**25**
Objective	**Generate Diagnostic Report**
Pre-Condition	Analysis completed
Flow	User clicks **Generate Report**
Expected Output	Report generated
Actual Output	Report generated successfully
Status	Passed

Table 51: Test Case 25

Test Case: 26

Test Case	**26**
Objective	**Download Report PDF**
Pre-Condition	Report generated
Flow	User clicks **Download**
Expected Output	PDF downloaded
Actual Output	PDF downloaded successfully
Status	Passed

Table 52: Test Case 26

Test Case: 27

Test case	**27**
Objective	**Verify Report Content**
Pre-Condition	Report generated
Flow	User opens report
Expected Output	Correct data displayed

Actual Output	Report content displayed correctly
Status	The Test Was Successfully Performed

Table 53: Test Case 27

Test Case: 28

Test case	**28**
Objective	**Report Includes Heatmap**
Pre-Condition	Heatmap generated
Flow	Generate report
Expected Output	Heatmap included in report
Actual Output	Heatmap included correctly in report
Status	Passed

Table 54: Test Case 28

Test Case: 29

Test case	**29**
Objective	**Generate Report Without Analysis**
Pre-Condition	No analysis performed
Flow	User clicks **Generate Report**
Expected Output	Error message displayed
Actual Output	System displayed error for missing analysis data
Status	Passed

Table 55: Test Case 29

Test Case: 30

Test case	**30**
Objective	**PDF Generation Failure Handling**
Pre-Condition	System issue simulated
Flow	User clicks Generate Report
Expected Output	Error message shown
Actual Output	Report generation error handled properly

Status	Passed

Table 56: Test Case 30

Test Case: 31

Test case	**31**
Objective	**Share Report Option Display**
Pre-Condition	Report generated
Flow	User clicks “Share Report”
Expected Output	Sharing options displayed
Actual Output	Sharing options displayed correctly
Status	Passed

Table 57: Test Case 31

Test Case: 32

Test case	**32**
Objective	**Share Report via Email**
Pre-Condition	Valid report available
Flow	• User selects Email option • Enters valid email • Clicks Send
Expected Output	Report sent successfully
Actual Output	Report shared via email successfully
Status	Passed

Table 58: Test Case 32

Test Case: 33

Test case	**33**
Objective	**Share Report with Invalid Email**
Pre-Condition	Report available
Flow	• Enter invalid email • Click Send

Expected Output	Error message displayed
Actual Output	System displayed invalid email error
Status	Passed

Table 59: Test Case 33

Test Case: 34

Test case	**34**
Objective	**Admin View Doctors List**
Pre-Condition	Admin logged in
Flow	Admin opens dashboard
Expected Output	List of doctors displayed
Actual Output	Doctor list displayed successfully
Status	Passed

Table 60: Test Case 34

Test Case: 35

Test case	**35**
Objective	**Add New Doctor**
Pre-Condition	Admin logged in
Flow	• Click Add Doctor • Enter details • Submit
Expected Output	Doctor added successfully
Actual Output	New doctor account created successfully
Status	Passed

Table 61: Test Case 35

Test Case: 36

Test case	**36**
Objective	**Add Doctor with Duplicate Email**
Pre-Condition	Admin logged in
Flow	• Enter existing email • Submit

Expected Output	Error message displayed
Actual Output	Duplicate email error shown correctly
Status	Passed

Table 62: Test Case 36

Test Case: 37

Test case	**37**
Objective	**Update Doctor Details**
Pre-Condition	Doctor exists
Flow	• Admin edits doctor info • Saves changes
Expected Output	Data updated successfully
Actual Output	Doctor details updated correctly
Status	Passed

Table 63: Test Case 37

Test Case: 38

Test case	**38**
Objective	**Delete Doctor Account**
Pre-Condition	Doctor exists
Flow	• Admin clicks Delete • Confirms action
Expected Output	Doctor removed from system
Actual Output	Doctor account deleted successfully
Status	Passed

Table 64: Test Case 38

Test Case: 39

Test case	**39**
Objective	**Block Doctor Account**
Pre-Condition	Doctor exists
Flow	Admin clicks Block

Expected Output	Doctor access restricted
Actual Output	Doctor account blocked successfully
Status	Passed

Table 65: Test Case 39

Test Case: 40

Test Output case	**40**
Objective	**Search Doctor**
Pre-Condition	Admin logged in
Flow	• Enter doctor's name • Click Search
Expected	Relevant results displayed
Actual Output	Search results displayed correctly
Status	Passed

Table 66: Test Case 40

Test Case: 41

Test case	**41**
Objective	**Unauthorized Access to Dashboard**
Pre-Condition	User not logged in
Flow	User tries to access dashboard
Expected Output	Redirect to login page
Actual Output	Unauthorized user redirected to login
Status	Passed

Table 67: Test Case 41

Test Case: 42

Test case	**42**
Objective	**Session Expiry Handling**
Pre-Condition	User logged in
Flow	Stay idle for long time

Expected Output	Session expires and logout
Actual Output	User session expired and redirected to login
Status	Passed

Table 68: Test Case 42

Test Case: 43

Test case	**43**
Objective	**Role-Based Access Control**
Pre-Condition	Doctor logged in
Flow	Doctor tries to access admin panel
Expected Output	Access denied
Actual Output	System blocked unauthorized access
Status	Passed

Table 69: Test Case 43

Test Case: 44

Test case	**44**
Objective	**Password Strength Validation**
Pre-Condition	Password setup page open
Flow	Password setup page open
Expected Output	Weak Password message displayed
Actual Output	Weak Password message displayed
Status	Passed

Table 70: Test Case 44

Test Case: 45

Test case	**45**
Objective	**Navigation Without Reload**
Pre-Condition	User logged in
Flow	Navigate between pages
Expected Output	Smooth navigation without reloads

Actual Output	Pages navigated smoothly
Status	Passed

Table 71: Test Case 45

Test Case: 46

Test case	**46**
Objective	**Responsive UI Check**
Pre-Condition	Application open
Flow	Open app on different devices
Expected Output	UI adjusts properly
Actual Output	UI responsive on all devices
Status	Passed

Table 72: Test Case 46

Test Case: 47

Test case	**47**
Objective	**Upload Page Navigation**
Pre-Condition	User logged in
Flow	Navigate to upload page
Expected Output	Upload page opens correctly
Actual Output	Upload page displayed successfully
Status	Passed

Table 73: Test Case 47

Test Case: 48

Test case	**48**
Objective	**Analysis Page Navigation**
Pre-Condition	Image uploaded
Flow	Navigate to analysis page
Expected Output	Analysis page displayed
Actual Output	Analysis page opened correctly

Status	Passed

Table 74: Test Case 48

Test Case: 49

Test case	**49**
Objective	**Report Page Navigation**
Pre-Condition	Analysis complete
Flow	Open report page
Expected Output	Report page displayed
Actual Output	Report page displayed successfully
Status	Passed

Table 75: Test Case 49

Test Case: 50

Test case	**50**
Objective	**Error Handling for Server Down**
Pre-Condition	Backend unavailable
Flow	Perform any action
Expected Output	Error message displayed
Actual Output	System handled server error properly
Status	Passed

Table 76: Test Case 50

Test Case: 51

Test case	**51**
Objective	**End-to-End Workflow Test**
Pre-Condition	User logged in
Flow	• Login • Upload image • Analyze • View result • Generate report
Expected Output	Complete workflow successful

Actual Output	Full workflow executed successfully
Status	Passed

Table 77: Test Case 50

6.7 Conclusion

In this chapter, we provide all the detailed information about the phases of testing that this project has been through. There was a total of three phases of testing i.e. Unit testing, component testing, and integration testing.

Chapter # 7

Conclusion

Chapter 7

Conclusion

CyberSight is meant to help ophthalmologists and clinicians in early detection and grading of diabetic retinopathy through the use of AI-based analysis of retinal fundus images. The system offers automated classification of diseases on five levels of severity, explainable AI-generated heatmaps indicating the areas contributing to each prediction and downloadable clinical reports in PDF format. The platform can help doctors make better clinical decisions and allows patients to have better outcomes because the platform can provide quick and AI-assisted diagnostic assistance and early detection and intervention.

This report started by providing motivation and background about CyberSight, then has provided a literature review of existing techniques and technologies in AI-based medical image analysis. It then outlined all the stages of the system development life cycle, such as requirements analysis, system design, architectural, implementation strategy, and system testing. The tools, technologies and design artifacts utilized in the project were also recorded in the report and therefore served as a full reference both to the developers as well as evaluators.

CyberSight is deployed as a web-based application, which combines the latest technologies, including React, TypeScript, FastAPI, PyTorch, Vision Transformers (via timm), OpenCLIP, OpenCV, and Microsoft Azure SQL. This combination enables the system to deliver a responsive, scalable, and intelligent medical image analysis platform. It was completed on time and reached its main goal, creating a correct, understandable, and available AI-based system to screen diabetic retinopathy.

7.1. Contributions

In this project report all technical and non-technical requirements were covered. This web application offers a convenient and intelligent means by which doctors can post retinal fundus images and get AI-assisted diagnoses based on deep learning models. The system addresses such issues as clinicians waste time with manual screening, there is a shortage of specialist ophthalmologists, and the visual explanation of available automated screening systems

7.2. Reflections

7.2.1. Strengths:

1. Clinical interface that is user-friendly and responsive and can be accessed via any current web browser.
2. Diabetic retinopathy classification with an AI-based fine-tuned Vision Transformer (ViT-Base) that is trained on a large-scale retinal dataset.
3. Image validation of CLIPs that automatically discard images not of the fundus prior to prediction, to avoid misclassification.
4. Visualize AI heatmaps with occlusion-based saliency that graphically and visually indicate which parts of the retinal picture were used to make the diagnosis.
5. Edible PDF clinical reports to keep records and document patients.

6. Role-based access control with doctor and administrator separate interfaces.
7. Microservice-friendly architecture allowing the addition of new disease models in the future without code modifications.

7.2.2. Weaknesses:

1. Now available only as a web-based solution, without native mobile support.
2. The system supports only one disease (diabetic retinopathy) at present, although the architecture is designed for multi-disease expansion.
3. AI inference runs on CPU, which makes heatmap generation slower (30–90 seconds) compared to GPU-based deployment.

7.2.3. Disciplined Project Management:

Proper project management was a key factor in CyberSight development that was successful and on time. The project was an incremental, agile one, beginning with the model training pipeline, and core backend and gradually extending it with the CLIP validator, explainability module, and authentication system, an admin panel and frontend interface. Tracking tasks and frequent team meetings served to make the development process run smoothly, and organize modules based on milestones.

7.2.4. Importance of Team Communication:

Open and transparent communication among team members was essential for collaboration. Feedback loops, regular check-ins, and active participation ensured that everyone contributed equally and that all ideas were considered. Good team communication contributed significantly to a productive and creative development environment, especially when coordinating between the AI model training, backend API development, and frontend interface design.

7.3. Future work

7.3.1. Other platforms

Although CyberSight is currently deployed as a web application, future enhancements could include the development of dedicated mobile applications for Android and iOS, allowing clinicians to capture and analyze fundus images directly from portable ophthalmoscope attachments on their smartphones.

7.3.2. Additional features

Future versions of CyberSight could include:

- **Multi-disease support:** Adding AI models for glaucoma detection, age-related macular degeneration (AMD), and cataract grading, each deployed as independent microservices using the existing Model Service Proxy architecture.
- **GPU-based deployment:** Migrating AI inference to dedicated GPU servers using the built-in `USE_MODEL_API` flag to significantly reduce prediction and heatmap generation time.
- **Patient record management:** Adding a patient database to store analysis history, enabling longitudinal tracking of disease progression over multiple visits.
- **Enhanced explainability:** Integrating additional XAI techniques such as Grad-CAM or attention map visualization from the Vision Transformer's self-attention layers for richer visual explanations.
- **Multi-language support:** Adding support for multiple languages in the user interface to serve clinicians in different regions.

REFERENCES

- [1] Akhtar, S., et al. (2025). Diabetic retinopathy severity grading using transfer learning techniques.
- [2] Aurangzeb, K., Alharthi, R. S., Haider, S. I., & Alhussein, M. (2023). System development of an AI-enabled diagnostic system for glaucoma and diabetic retinopathy. *IEEE Access*.
- [3] Asif, M., Rehman, F. U., Mirza, A., Rashid, Z., Hussain, A., & Qureshi, W. S. (2025). An insight on the timely diagnosis of diabetic retinopathy using traditional and AI-driven approaches. *IEEE Access*.
- [4] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., et al. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, vol. 316, no. 22, pp. 2402–2410. https://doi.org/10.1001/jama.2016.17216
- [5] Saproo, D., et al. (2024). Deep learning-based binary classification of diabetic retinopathy.
- [6] Buda, S., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, vol. 106, pp. 249–259. https://doi.org/10.1016/j.neunet.2018.07.011
- [7] Yuan, Y., et al. (2021). Vision transformers for medical image classification.
- [8] Gargeya, R., & Leng, T. (2017). Automated identification of diabetic retinopathy using deep learning. *Ophthalmology*, vol. 124, no. 7, pp. 962–969. https://doi.org/10.1016/j.ophtha.2017.02.008
- [9] Pratt, H., Coenen, F., Broadbent, D. M., Harding, S. P., & Zheng, Y. (2016). Convolutional neural networks for diabetic retinopathy. *Procedia Computer Science*, vol. 90, pp. 200–205.
- [10] Quellec, G., Charrière, K., Boudi, Y., Cochener, B., & Lamard, M. (2017). Deep image mining for diabetic retinopathy screening. *Medical Image Analysis*, vol. 39, pp. 178–193. https://doi.org/10.1016/j.media.2017.04.012
- [11] Ting, D. S. W., Cheung, C. Y. L., & Wong, T. Y. (2016). Diabetic retinopathy: Global prevalence and major risk factors. *Diabetes Care*, vol. 39, no. 10, pp. 1783–1792.
- [12] Li, Z., He, Y., Keel, S., Meng, W., Chang, R. T., & He, M. (2018). Efficacy of a deep learning system for detecting diabetic retinopathy. *JAMA Network Open*, vol. 1, no. 4, e181575.
  https://doi.org/10.1001/jamanetworkopen.2018.1575

- [13] Abràmoff, M. D., Lavin, P. T., Birch, M., Shah, N., & Folk, J. C. (2018). Pivotal trial of an autonomous AI-based diagnostic system for detection of diabetic retinopathy. *NPJ Digital Medicine*, vol. 1, no. 1, pp. 1–8. https://doi.org/10.1038/s41746-018-0040-6

- [14] Bhaskaranand, M., Ramachandra, C., Bhat, S., et al. (2016). The value of automated diabetic retinopathy screening with the EyeArt system. *Investigative Ophthalmology & Visual Science*, vol. 57, no. 13, pp. 5201–5209.

- [15] International Diabetes Federation. (2021). IDF Diabetes Atlas (10th ed.). Retrieved from https://diabetesatlas.org/

- **[16]** Canipek, A. S., et al. (2024). *EyePACS, APTOS, Messidor Diabetic Retinopathy Dataset*. Kaggle. Retrieved from https://www.kaggle.com/datasets/ascanipek/eyepacs-aptos-messidor-diabetic-retinopathy

APPENDIX A: GLOSSARY

Term	Definition
CyberSight	A web-based AI-powered clinical decision support system designed to assist in the detection and grading of diabetic retinopathy using retinal fundus images.
Vision Transformer (ViT)	A deep learning model that processes images as sequences of patches and uses self-attention mechanisms for classification tasks.
FastAPI	A high-performance Python web framework used as the backend of CyberSight for handling API requests, validation, and model inference.
React	A JavaScript library used to build the frontend interface for image upload, result visualization, and user interaction.
User (Doctor/Ophthalmologist)	A medical professional who uses the system to upload retinal images, view predictions, and generate reports.
Admin	A system-level user responsible for managing doctor accounts and controlling system access.
Artificial Intelligence (AI)	Technology that enables machines to simulate human intelligence, such as image recognition and decision-making.
Machine Learning (ML)	A subset of AI where models learn patterns from data to improve prediction performance.
CLIP (Contrastive Language–Image Pretraining)	A model used to verify whether an uploaded image is a valid retinal fundus image before analysis.
Fundus Image	A medical image of the retina captured using a fundus camera, used as input for DR detection.
Diabetic Retinopathy (DR)	A diabetes-related eye disease that damages retinal blood vessels and may lead to blindness.
DR Severity Levels	Categories of diabetic retinopathy: No DR, Mild, Moderate, Severe NPDR, and Proliferative DR.
Image Preprocessing	Techniques such as resizing, normalization, and validation applied before model input.
Explainable AI (XAI)	Techniques used to make AI predictions interpretable for users.
Occlusion-Based Heatmap	An explainability technique that highlights important regions of an image by masking parts and observing prediction changes.
Softmax Probabilities	Output values representing the likelihood of each prediction class.
API (Application Programming Interface)	A communication interface that allows frontend and backend components to exchange data.
JSON Response	A structured format used by the backend to return prediction results and metadata.
Model Inference	The process of generating predictions using a trained AI model on new data.
Dataset	A collection of labeled retinal images used to train and evaluate the model.
Data Augmentation	Techniques used to increase dataset size by modifying images (e.g., rotation, flipping).

Class Imbalance	A condition where some classes have significantly more data than others.
Stratified Sampling	A method of splitting data while maintaining class distribution across sets.
timm Library	A PyTorch-based library used to load and train Vision Transformer models.
ReduceLROnPlateau	A learning rate scheduling technique used to improve training performance.
JWT Authentication	A secure method for user authentication using JSON Web Tokens.
Azure SQL Database	A cloud-based relational database used to store system and user data.
Clinical Decision Support System (CDSS)	A system that assists healthcare professionals in decision-making without replacing them.
Medical Disclaimer	A statement clarifying that AI results are assistive and not a substitute for professional diagnosis.