**DETECTS DIABETIC EYE DISEASE IN RETINAL IMAGES**

## **[KCS-751]**

**B.Tech Project Submitted to**



**Dr. A.P.J. Abdul Kalam Technical University**

**In partial fulfillment of the requirement for the degree of**

# **BACHELOR OF TECHNOLOGY**

## **Session 2022—2023**

**Submitted By** **Submitted to**

Shagun Tyagi Mr. Asif

Satendra Assistant Professor

Manish Kumar



# **SUNDER DEEP ENGINEERING COLLEGE**

**NH-24, Delhi- Hapur Road, Dasna, Ghaziabad (UP)**

***Approved by AICTE and Affiliated to AKTU &***

**AKTU Code: 240**

**CANDIDATE DECLARATION**

We, **Shagun Tyagi**, **Satendra** and **Manish Kumar** hereby declare that the work presented in the Project Report entitled “**DETECTS DIABETIC DISEASE IN RETINAL IMAGES**” submitted in the Department of Computer Science & Engineering, SunderDeep Engineering College, Ghaziabad for the partial fulfillment of the requirement for award of the degree of Bachelor of Technology, is my authentic work carried out during the 8th semester.

The matter embodied in this project report has not been submitted elsewhere by anybody for the award of any degree, associateship, fellowship or any other similar titles.

Date:

Sign.

Shagun Tyagi

Sign.

Satendra

Sign.

Manish Kumar

**CERTIFICATE**

This is to certify that the work entitled “**DETECTS DIABETIC DISEASE IN RETINAL IMAGES**” submitted by the group in the partial fulfillment for the award of B.Tech degree in **Computer Science and Engineering, Sunderdeep Engineering college, Ghaziabad**, has been carried out under my supervision. The work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Sign Date:

Mr. Asif

Assistant Professor

Department of Computer Science and Engineering

Sunder Deep Engineering College

Ghaziabad

Sign

Mr. Varun Kumar Singh

Head of Department

Department of Computer Science and Engineering

Sunder Deep Engineering College

Ghaziabad

**ACKNOWLEDGEMENT**

We would like to convey our truthful gratitude to Mr. Asif for his tremendous support in the completion of our final year project. I would also like to thank our HOD (Head of the Department) - Mr. Varun Kumar Singh, for providing us this wonderful opportunity to work on a project i.e., “Detects Diabetic Disease in Retinal Images”. The completion of the project would not have been possible without their help and Assistance.

We are highly thankful to every individual for constant guidance, supervision and friendly environment for the successful completion of this project. We would like to thankful our project supervisor Mr. Asif who tool keen interest in our project and guided us throughout the project by providing all the great ideas and knowledge for developing a functional web application.

We are also very blessed to get constant support from our seniors and our staff of B.tech department which helped us to complete this great project timely.

Last but not the least, our thanks and appreciation to our colleagues for their encouragement and support in developing the project.

**ABSTRACT**

The Project on Detection of Diabetic Eye Disease in Retinal Images is developed to save the precious time of today's busy world. The purpose of this project is to provide facility to check infection in your eye.

As we know, this youth of digital and technical world is remaining in front of screens for long hours which is very harmful for our eyes. This web application is having an interactive user interface where you only need to upload an image of your eye and then you will get to know the percentage of Diabetic Retinopathy in your eye.

This online functional application is one of the easiest platforms to check the diabetic disease in your eye. Detection of diabetic disease in retinal images through this web application is very fast and easy to use. Our application promise to give accurate result in minimum possible time.

This web application is very useful for the person who is having some eye infection symptoms like itching, watery eye and all and, he/she is having very hectic schedule and he/she really wants to save his /her precious time. If so then, he/she will get to know the level of infection in his/her eye on interacting with this application. If he/she is having the same then he/she should manage his/her schedule and then, go for an appointment to Ophthalmologist.

**LIST OF FIGURES**

**S.NO** **TITLE NAME** **PAGE NO**

1 Retinal Fundus

2 Mild Non proliferative diabetic retinopathy

3 Moderate Non proliferative diabetic retinopathy

4 Severe Non proliferative diabetic retinopathy

5 Proliferative diabetic retinopathies

6 Current Diagnosis Process

7 Proposed Diagnosis Process

8 Sample Inputs

9 Front-End Folder Structure

10 Graphical Representation of Web Application

11 Python Backend Folder structure

12 Representation of Detection of Diabetic Fundus

13 Nodejs Backend folder structure

14 Flow Diagram of Proposed Technique

15 ResNet – 101 Architecture

16 Flow Diagram of Working of Detection of Diabetic Disease in Retinal Images

17 Sign In

18 Sign Up

19 Application Dashboard

20 Select and Upload Image

21 Output

22 Layout of recent Status / Activities of User

23 History

24 User Profile and Setting

**CONTENT**

CANDIDATES’ DECLARTION i

CERTIFICATE ii

ACKNOWLEDGEMENT iii

ABSTRACT iv

LIST OF FIGURES v

**CHAPTER –1 INTRODUCTION**

1.1 General

1.2 Types of Diabetic Retinopathy

1.2.1 Mild Non-Proliferative Diabetic retinopathy

1.2.2 Moderate Non proliferative diabetic retinopathy

1.2.3 Severe Non proliferative diabetic retinopathy

1.2.4 Proliferative diabetic retinopathy

1.3 Current Diagnosis Process

1.4 Proposed Diagnosis Process

1.5 Project Scope

1.6 Technology Used

**CHAPTER-2 REQUIREMENT SPECIFICATION**

2.1 Sample to take Input

2.2 Color Fundus Camera

2.3 Required Input Data

2.3.1 Methods Applied on Input Data

2.4 Hardware Requirements

2.5 Software Requirements

**CHAPTER-3 SYSTEM DESIGN**

3.1 Graphical Representation

3.2 Explanation of the Block Diagram

3.2.1 Start Application

3.2.1 Dashboard

3.3 Block Diagram of Detects Diabetic

**CHAPTER-4 TOOLS AND TECHNIQUES**

4.1 Terminologies to manage the project

4.2 Transfer Learning

4.3 Data Augmentation

4.4 Image Compression

4.5 Indexing

4.6 Caching

4.7 Code Optimization

**CHAPTER-5 WORKING OF THE PROJECT**

5.1 Something about Web Application

5.2 Flow Diagram of Working of Detection of Diabetic Disease in Retinal Images

5.2.1 Annotations

5.2.2 The Center Net

5.2.3 Detection Process

5.3 Flow Chart of Working of Detection of Diabetic Disease in Retinal Images

**CHAPTER-6 DESIGN SPECIFICATION**

6.1 Frontend Design

6.1.1 Sign In

6.1.2 Sign Up

6.1.3 Dashboard

6.1.4 Select and Upload Image

i Image Selection

ii Image upload

6.1.5 Response and Result

i Result Display

ii Result Visualization

6.1.6 Recent User Activities and Results

i Activity Log

ii Result Summary

6.1.7 History

6.1.8 User Profile and Settings

6.1.9 Sign-out the Application

CONCLUTION

FUTURE WORK

REFERENCE

**CHAPTER-1**

**INTRODUCTION**

**1.1 GENERAL**

Diabetic retinopathy (DR) is the leading cause of blindness in the working-age population of the developed world and is estimated to affect over 93 million people.

The US Center for Disease Control and Prevention estimates that 29.1 million people in the US have diabetes and the World Health Organization estimates that 347 million people have the disease worldwide. Diabetic Retinopathy (DR) is an eye disease associated with long-standing diabetes.

Currently, detecting DR is a time-consuming and manual process that requires a trained clinician to examine and evaluate digital color fundus photographs of the retina. By the time human readers submit their reviews, often a day or two later, the delayed results lead to lost follow up, miscommunication, and delayed treatment.

Some annotated examples from the literature to get an idea of what this really looks like.

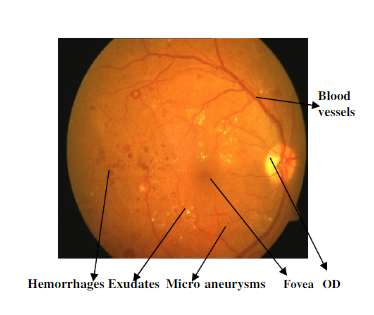


Fig 1. Retinal Fundus

**1.2 Types of diabetic retinopathy**

The condition has four stages of progression:

1. Mild Non proliferative diabetic retinopathy.
2. Moderate Non proliferative diabetic retinopathy.
3. Severe Non proliferative diabetic retinopathy.
4. Proliferative diabetic retinopathy.

**1.2.1 Mild Non proliferative diabetic retinopathy**

The earliest stage of diabetic retinopathy is the mild Non proliferative stage. During this stage, the presence of diabetes can cause the structure of the retina to change. At least one microaneurysm (a tiny red protrusion, or “bulb,” in a blood vessel in the retina) occurs during this stage.

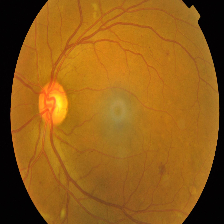


Fig 2 Mild DR

**1.2.1 Moderate Non proliferative diabetic retinopathy**

Blood vessels, which nourish the retina, are at risk of swelling and becoming blocked during the moderate Non proliferative stage of DR. Hemorrhages, microaneurysms and white spots called “cotton wool spots” also occur during this stage of DR.

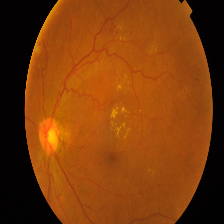


Fig 3 Moderate DR

**1.2.3 Severe Non proliferative diabetic retinopathy**

Blood vessels become increasingly blocked in the eye during the severe Non proliferative stage of DR. When this happens, the retina is unable to receive blood that it needs to function and is signaled by proteins called vascular endothelial growth factors (VEGFs) to grow new (but abnormal) blood vessels. Microaneurysms and hemorrhages occur during this stage of DR.

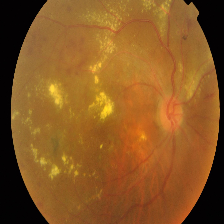


Fig 4 Severe DR

**1.2.4 Proliferative diabetic retinopathy**

Proliferative diabetic retinopathy is the final, most advanced stage of diabetic retinopathy. During this stage of DR, new blood cells can increase in the retina and make their way into the vitreous (the gel-like substance that fills the eye). These blood vessels are delicate and can eventually lead to retinal detachment.

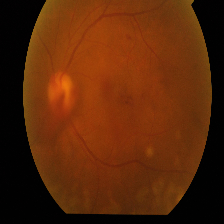


Fig 5 Proliferative DR

**1.3 CURRENT DIAGNOSIS PROCESS**

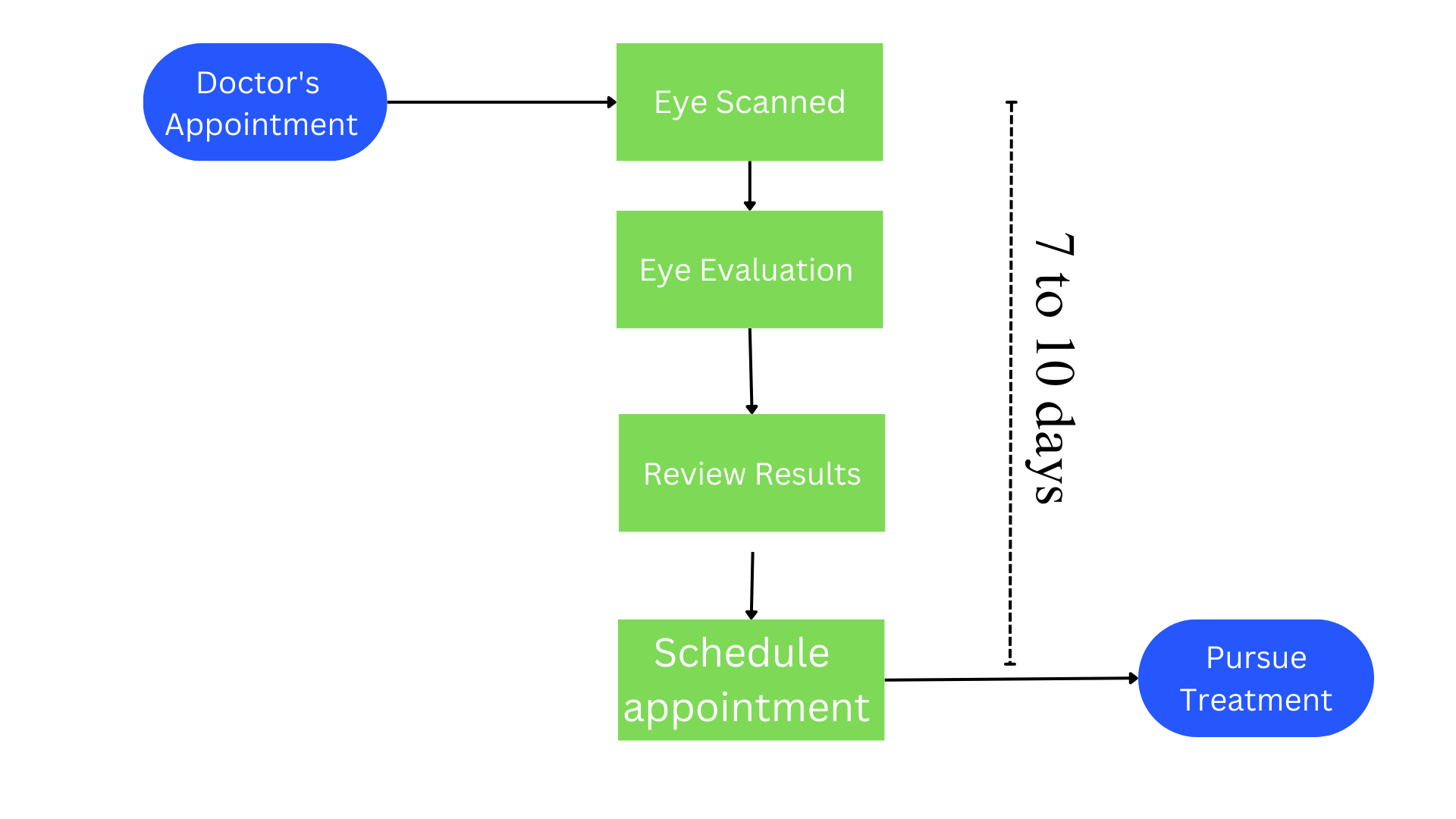


Fig 6 Current Diagnosis Process

**1.4 PROPOSED DIAGNOSIS PROCESS**

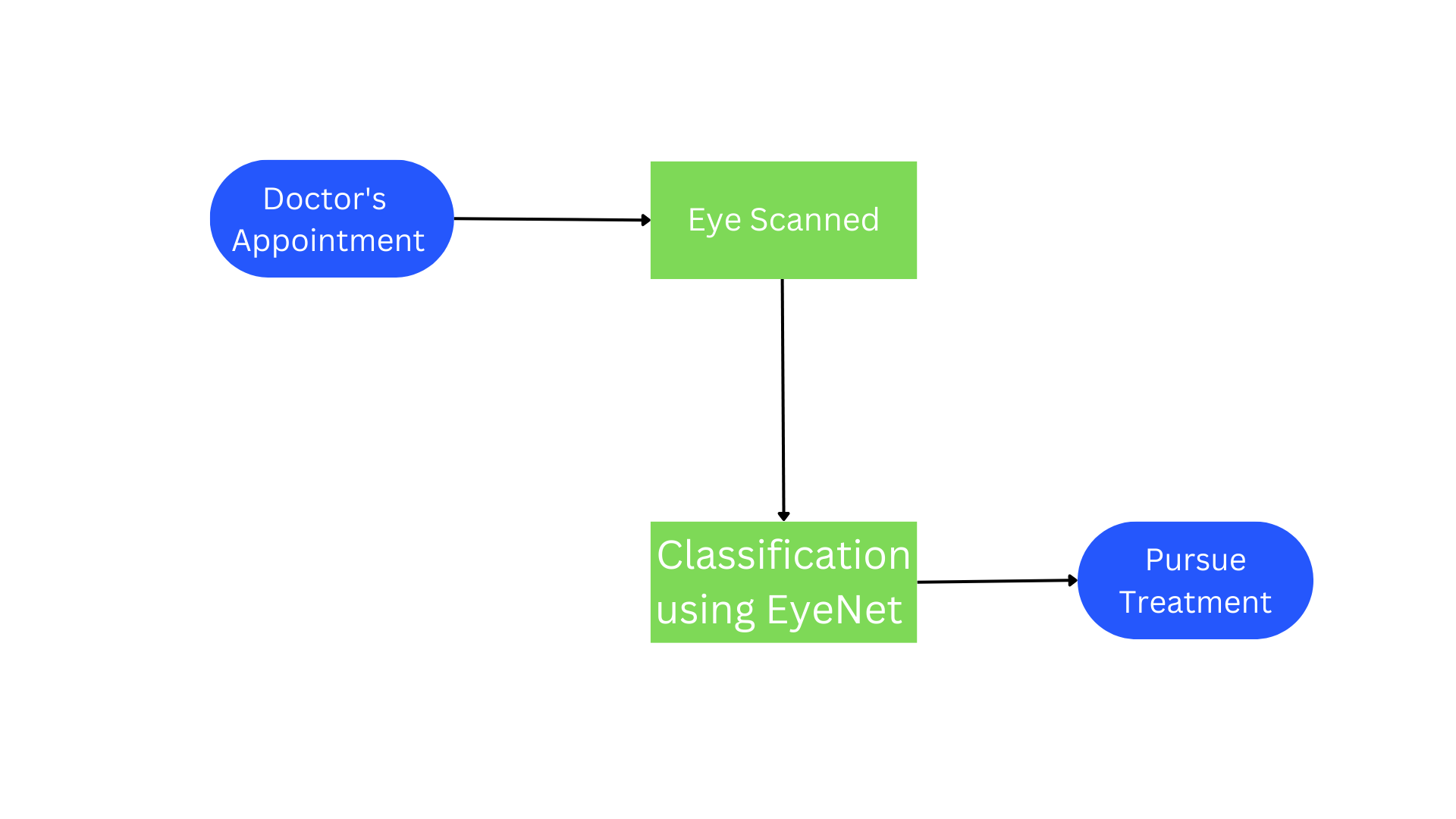


Fig 7 Proposed Diagnosis Process

**1.5 PROJECT SCOPE**

The scope of our project "Detects Diabetic Eye Disease in Retinal Images" includes the development of a web application that utilizes a pre-trained deep learning model to predict diabetic eye disease in retinal images. The project involves creating a Python backend using Flask to host the model and handle image processing requests. The React.js frontend with Redux for state management enables users to log in, upload retinal images, and view analysis results. User authentication is handled by a Node.js backend using JSON Web Tokens (JWTs) and MongoDB for database management. Cloudinary is integrated for image storage and processing. The project aims to provide an intuitive user interface, achieve high accuracy in disease prediction, and allow for future expansion to detect other eye diseases and incorporate additional features.

**1.6 TECHNOLOGY USED**

The software used in our project "Detects Diabetic Eye Disease in Retinal Images" includes the following:

1. **Python:** Python is used as the primary programming language for developing the backend of the application and hosting the deep learning model. It provides a wide range of libraries and frameworks for machine learning and web development.
2. **Flask:** Flask is a lightweight web framework for Python that is used to build the backend of the application. It allows for the creation of RESTful APIs, handling HTTP requests, and serving the machine learning model.
3. **React.js:** React.js is a JavaScript library used for building the frontend of the application. It provides a component-based architecture and allows for efficient rendering of UI components, resulting in a fast and interactive user interface.
4. **Redux:** Redux is a state management library for JavaScript applications, widely used with React.js. It helps manage the application's state in a centralized manner, ensuring predictable and scalable application behavior.
5. **Node.js:** Node.js is a JavaScript runtime environment used for developing the user authentication backend. It provides a non-blocking, event-driven architecture, making it ideal for building scalable and efficient server-side applications.
6. **MongoDB:** MongoDB is a popular NoSQL database used for storing user data, analysis results, and other relevant information. It offers flexibility in data modeling and scalability, making it suitable for our application's needs.
7. **Cloudinary:** Cloudinary is a cloud-based media management platform that provides tools and APIs for image storage, processing, and manipulation. It is used in our project for secure image upload and retrieval.

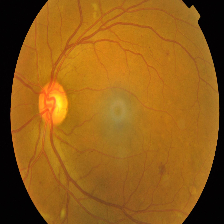
These software components work together to create a robust and efficient application for detecting diabetic eye disease in retinal images, with seamless integration between the frontend and backend components.

**CHAPTER 2**

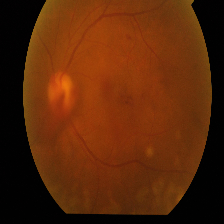
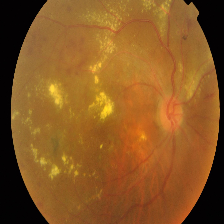
**REQUIREMENT SPECIFICATIONS**

**2.1 SAMPLE TO TAKE INPUT**

This project can be effectively used to detect the infection in a retinal image. To make successful use of this program user must upload an image of his/her eye. The Digital Image which is acceptable to this web application should be clicked by Digital Color Fundus Camera. This fundus camera is very well suitable to give colored and clear retinal images with best quality. Sample images are mentioned below.



SAMPLE 01 SAMPLE 02



SAMPLE 03 SAMPLE 04

Fig 8 Sample Inputs

**2.2 COLOR FUNDUS CAMERA**

Ophthalmology is one of the great photographic worlds which is very familiar with fundus camera and we are having a great reason behind this and i.e., Color fundus cameras are the best source to capture colorful retinal images without compromising the quality of an image. Fundus Camera is also capable to record various images of an interior surface of an eye based on different conditions. Images clicked by fundus camera are very useful to detect certain disorders in an eye and to monitor their change in different timestamps.

**2.3 REQUIRED INPUT DATA**

This web application is having one fixed criterion to examine the level of infection in any retinal image and that level is categorized into starting, moderate or final stage infection. Fundus images are provided as input to deal with this project. User is allowed to browse an image from the device to proceed further so that you will get to know the level and percentage of infection in your eye.

Project consists of attractive and interactive user interface where image of patient’s retina is uploaded to predict the percent of infection by using CNN based system. This CNN system describes two conditions of Diabetic Retinopathy I.e., Proliferative and Non – Proliferative stage of Diabetic Retinopathy. CNN based system has been used to analyze the fundus image and then predict the result. This technique is time-consuming and less cost effective.

**2.3.1 METHODS APPLIED ON INPUT DATA**

1. **PRE**-**PROCESSING**

This is the first most stage to get the final output. Pre-processing is the process of converting the inputted image into suitable form to pass that input to next step. This includes a conversion of original input into required one which is as similar as to the input data and this can be done only by converting one color space to another color space. Color space conversion can be done by using types of colors like RGB, HEX, HSL, RGBA, etc.

1. **IMAGE SEGMENTATION**

Image segmentation comes just after completing pre-processing on inputted data. Basically, image segmentation involves smoothing, masking and Bitwise AND.

**a) SMOOTHING**

This is the process of removing high frequency noises from an image. Smoothing also allows us to achieve blur free image of best quality.

**b) MASKING**

Masking is just a game of color conversion according to someone's needs and requirements. Firstly, this creates small pieces of an image and then, use them to modify an original image by applying masking of colors accordingly.

**c) BITWISE AND**

This is an operator which is used for image manipulation and it is very useful to extract essential parts in the image. Bitwise AND is proved as very beneficial for image masking also this bitwise operator is capable to enhance the properties to the input image.

1. **FEATURE EXTRACTION**

Feature Extraction is the second last step to manipulate an image. Extraction of features of an image is necessary to have best and most accurate predictions to get an output. This one is introduced to provide ease to next stage I.e., classification

1. **CLASSIFICATION**

Classification is having its own functions and algorithms to identify the dataset. Methods used to classify the data are SVM (Support Vector Machine), KNN (K- Nearest Neighbors) and Random Forest Algorithm. They all are having their own function of implementation. After getting classifiers the inputted data is ready to do further processing.

**2.4 HARDWARE REQUIREMENTS**

1. A computer or server capable of running the required software component.
2. Processor: Intel Core i5 or equivalent, or higher.
3. RAM: 8 GB or higher.
4. Storage: At least 100 GB of available storage.
5. Graphic Card: NVIDIA GeForce GTX 1070 or equivalent, or higher (for running the deep learning model).
6. Internet connection: High-speed internet connection for seamless user experience.

**2.5 SOFTWARE REQUIREMENTS**

1. **Operating system:** Windows XP / Windows 7,8,8.1,10/Linux.
2. **Tools:** Visual studio code.
3. **Browser:** Anyone of Mozilla, Opera, Chrome etc.
4. **Web Server Software Development Kit:** Node.js server.
5. **Python:** Python is used as the primary programming language for developing the backend of the application and hosting the deep learning model. It provides a wide range of libraries and frameworks for machine learning and web development.
6. **Flask:** Flask is a lightweight web framework for Python that is used to build the backend of the application. It allows for the creation of RESTful APIs, handling HTTP requests, and serving the machine learning model.
7. **React.js:** React.js is a JavaScript library used for building the frontend of the application. It provides a component-based architecture and allows for efficient rendering of UI components, resulting in a fast and interactive user interface.
8. **Redux:** Redux is a state management library for JavaScript applications, widely used with React.js. It helps manage the application's state in a centralized manner, ensuring predictable and scalable application behavior.
9. **Node.js:** Node.js is a JavaScript runtime environment used for developing the user authentication backend. It provides a non-blocking, event-driven architecture, making it ideal for building scalable and efficient server-side applications.
10. **MongoDB:** MongoDB is a popular NoSQL database used for storing user data, analysis results, and other relevant information. It offers flexibility in data modeling and scalability, making it suitable for our application's needs.
11. **Cloudinary:** Cloudinary is a cloud-based media management platform that provides tools and APIs for image storage, processing, and manipulation. It is used in our project for secure image upload and retrieval.
12. **Material UI (MUI):** Material UI is used for design the front-end of stand-alone website along with the React.js.

**CHAPTER 3**

**SYSTEM DESIGN**

**3.1 FRONT-END FOLDER STRUCTURE**

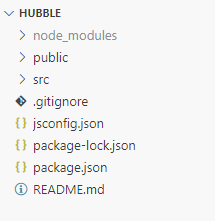


Fig 9 Front-End Folder Structure

1. **Src**

The "src" directory typically holds the source code files of a project, containing the main code and resources for development.

1. **App**.**js**

The "app.js" file is commonly used as an entry point or main file in a Node.js or web application, where the application logic and server setup are defined.

1. **Index.js**

The "index.js" file often serves as the entry point or main file in a JavaScript project, where the application starts and dependencies are imported.

1. **Package.json**

The "package.json" file is a configuration file that holds metadata about a Node.js project, including its dependencies, scripts, and project details.

1. **node\_modules**

The "node\_modules" directory is where external libraries and dependencies are stored when using Node.js, allowing the project to access and utilize these modules.

**3.2 GRAPHICAL REPRESENTATION**

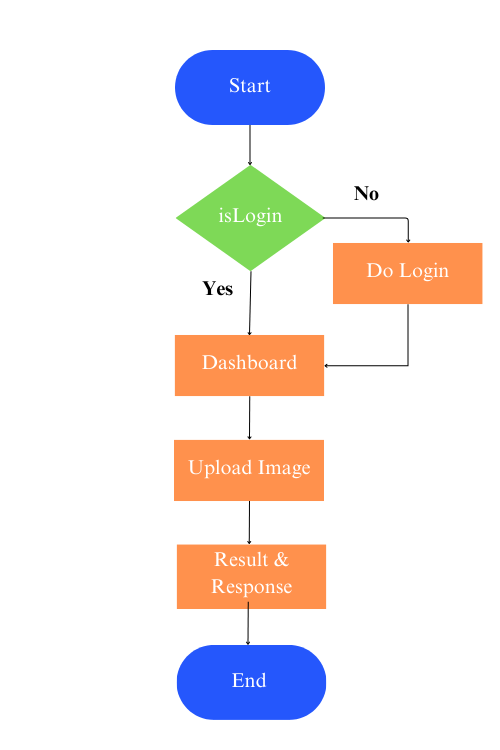


Fig 10 The graphical representation of web Application

**3.3 Explanation of the Block diagram**

**3.2.1 Start Application**

On starting the application, there are two options sign-in or sign-up.

1. **Sign In**

Users enter their credentials and click Sign In. The server verifies the information and navigates them to the dashboard.

1. **Sign Up**

Users provide required information, click Sign Up, and the server creates a new account. Users receive a success message and are redirected to the login screen.

**3.2.2 Dashboard**

After successful login, users are directed to the Dashboard where they can access various features and information related to the application. The Dashboard provides an overview of the user's uploaded images, analysis history, and relevant statistics. Users can also navigate to different sections, such as uploading new images, viewing analysis results, and managing account settings, to interact with the application.

* 1. **Dashboard**

Provides users with an overview of uploaded images, analysis history, and statistics. Users can upload images, view analysis results, and manage account settings.

* 1. **History**

Provides a chronological record of previous image analyses for easy reference. Users can view past results, including diagnoses and relevant information for each analysis.

* 1. **Profile**

Allows users to view and update their personal information and preferences. Users can edit details such as their name, contact information, and profile picture.

**3.4 PYTHON BACKEND FOLDER STRUCTURE**

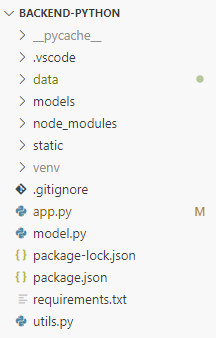


Fig 11 Python Backend Folder structure

**3.3 BLOCK DIAGRAM OF DETECTS DIABETIC**

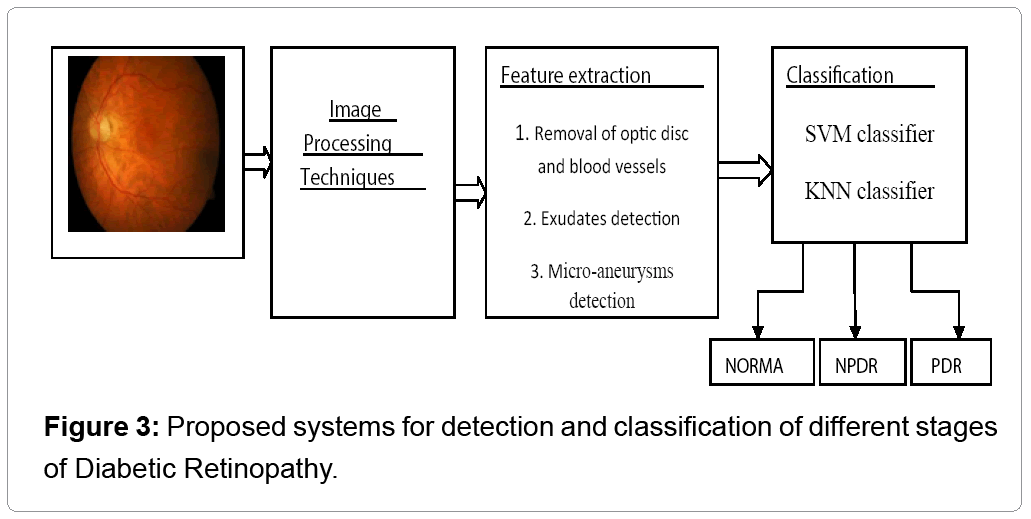


Fig 12 Representation of Detection of Diabetic Fundus

**3.4 NODEJS BACKEND**

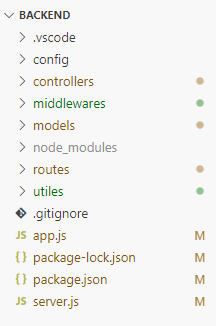


Fig 13 Nodejs Backend folder structure

**CHAPTER 4**

**TOOLS AND TECHNIQUES**

**4.1 TERMINOLOGIES TO MANAGE THE PROJECT**

In our project, we have employed several techniques to enhance the functionality and performance of the application. We leveraged transfer learning by utilizing a pre-trained deep learning model for diabetic eye disease prediction, allowing us to benefit from pre-existing knowledge and features. Data augmentation techniques were implemented to expand the dataset through various image transformations, improving the model's generalization. Additionally, we employed image compression techniques to optimize storage space and accelerate data retrieval. Indexing was utilized in MongoDB for efficient query performance, while caching mechanisms and code optimization techniques were employed to enhance overall application performance. Continuous integration and deployment pipelines were set up to automate testing, building, and deployment processes. Lastly, robust error handling and validation mechanisms were implemented to ensure a smooth and error-free user experience.

**4.2 TRANSFER LEARNING**

Incorporated transfer learning by utilizing a pre-trained deep learning model for diabetic eye disease prediction, leveraging pre-existing knowledge and features to improve accuracy and reduce training time. The model was fine-tuned on our specific dataset, enabling it to learn disease patterns effectively.

**4.3 DATA AUGMENTATION**

Implemented data augmentation techniques by applying image transformations such as rotation, scaling, and flipping. This artificially expanded the dataset, improving the model's generalization and robustness by exposing it to a wider range of variations and perspectives in retinal images.

**4.4 IMAGE COMPRESSION**

Employed image compression techniques to reduce the file size of uploaded images without compromising quality. This optimization reduced storage requirements, minimized network bandwidth usage, and improved the overall performance of the application.

**4.5 INDEXING**

Utilized indexing techniques in MongoDB to enhance query performance and accelerate data retrieval. By creating appropriate indexes on frequently accessed fields, we improved query response times and ensured efficient handling of large amounts of user data.

**4.6 CACHING**

Implemented caching mechanisms at both the browser and server levels to store frequently accessed data and computed results. This reduced the need for repetitive computations and database queries, resulting in faster response times and improved user experience.

**4.7 CODE OPTIMIZATION**

Employed various code optimization techniques, including algorithmic improvements, minimizing redundant operations, and optimizing resource utilization. This optimization enhanced the efficiency of the application, resulting in faster execution times and improved overall performance.

**CHAPTER 5**

**WORKING OF THE PROJECT**

**5.1 SOMETHING ABOUT WEB APPLICATION**

Detection of Diabetic Disease in Retinal Images is a user-friendly web application which is very easy to use and implement. This web application does not require any technical or professional skill to make successful use of this application.

**5.2 FLOW DIAGRAM OF WORKING OF DETECTION OF DIABETIC DISEASE IN RETINAL IMAGES**

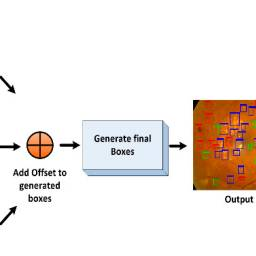
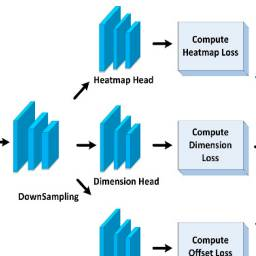
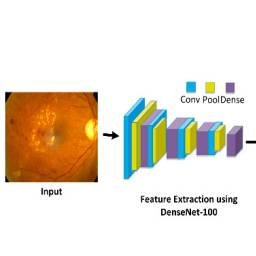


Fig 14 Flow Diagram of Proposed Technique

The above diagram comprises two main parts. First one is ‘dataset preparation’ and the other is an improved Center Net network trained for the classification of eye disease. Firstly, we develop analyses for images with all the disease, to clarify the region of interest, where the other part of the generated framework trains Center Net over the developed samples of images. We employed Custom Center Net with DenseNet-100 as its base network for feature computation. The feature extraction process of the Custom Center Net framework and DenseNet-100 accepts all the samples of images and location of the affected region in the input image. Illustrates the architecture of the presented method. In the beginning, an input sample, is passed to the DenseNet-100 framework along with their bounding box (b-box). The b-box recognizes the exact region of interest in the key points of CNN. Following this, the Custom Center Net is trained to classify the located areas. Finally, accuracies are estimated for all units as per metrics being used in the vision of computer.

**5.2.1 ANNOTATIONS**

This method is used to have an efficient training process. It is necessary to specify the position of the affected region from the input retinal samples of image. The generated annotations are stored in an XML file which holds two important details: class associated with each affected region and another one is values of box to draw a rectangular box over the detected region. In the next step, a train. record file is created from an XML file, which is used to train the model.

**5.2.2 THE CENTER NET**

The structure of Center Net employed ResNet is utilized to compute image key points and medical analysis of image. After all, ResNet framework use skip-connections and identity function of employes to confront the non-linear transformations, which cause the direct flow of gradient from the back layers to the front layer by implementing the identity function. Fig2 shows the structural view of the ResNet-101 framework. The ResNet-101 model comprises huge parameters which causes the vanishing gradient problem eventually. To deal with the issue of ResNet-101 framework we are having a convolution framework, namely and Dense Net, as the base network of the conventional Center Net technique by replacing ResNet-101 with DenseNet-100. The presented feature extractor, namely and DenseNet-100, has fewer parameters and has a thinner layer network in comparison to ResNet-101. which provides more cost-efficiency. Dense Net contains various dense blocks (DBs) which are joined successively to each other through additional convolutional and pooling layers between consecutive Dense blocks.



Fig 15 Resnet – 101 Architectures

**5.2.3 DETECTION PROCESS**

Center Net is a DL-based structure that is independent of the other approaches such as selective search and proposal creation. Therefore, the assumed image with its b-box is passed to the trained model as an input of which Center Net calculates the center points of the eye diseases portion, the offsets to the x and y coordinates and the dimensions of b-boxes along with the associated class.

**5.3 FLOW CHART OF WORKING OF “DETECTION OF DIABETIC DISEASE IN RETINAL IMAGES”**

This diagram mentioned below is the working of this project from taking input to getting output at the end. Where inputted image is processed and is used to identify the category of retinal infection I.e., Normal, Earlier, Moderate and Severe. This involves various methods and techniques to run the application successfully.

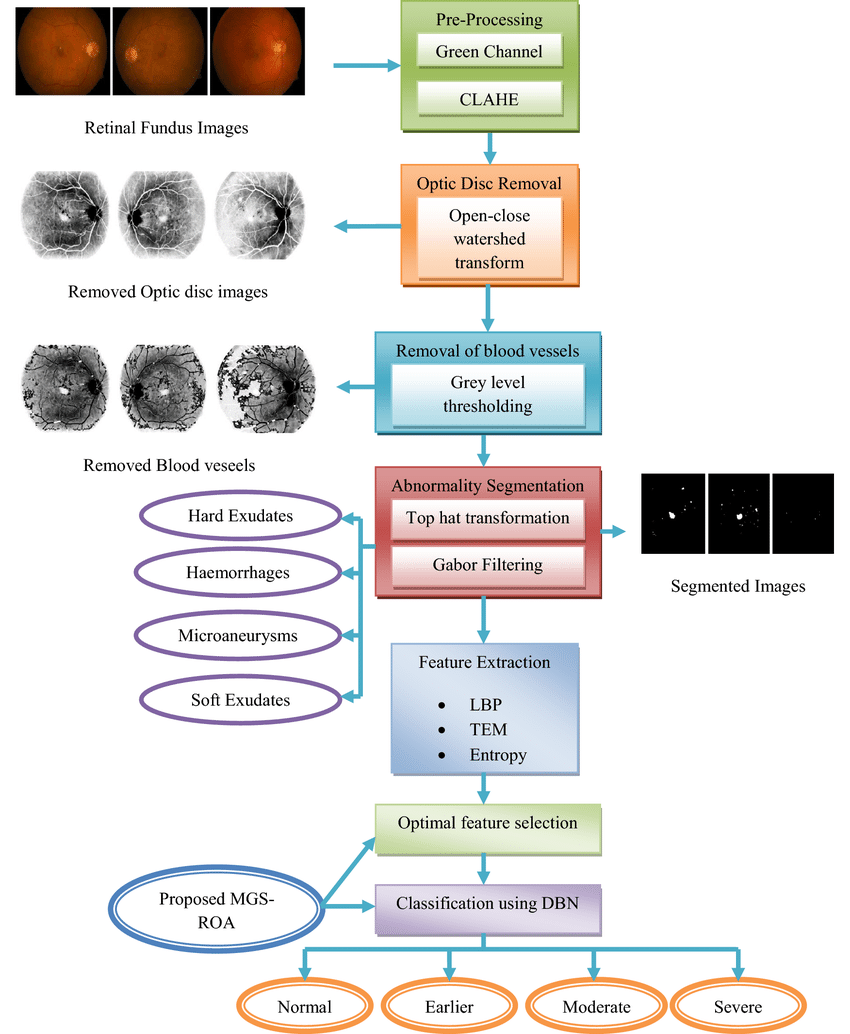


Fig 16 Flow Diagram of Working of Detection of Diabetic Disease in Retinal Images

**CHAPTER 6**

**DESIGN SPECIFICATION**

**6.1 FRONT-END DESIGN**

The frontend of the "Detects Diabetic Eye Disease in Retinal Images" application is built using ReactJS and Redux, with @mui/material used for UI components. Redux is used for state management to handle the state of the application and communicate with the backend. To handle user authentication, we used a library such as Passport to provide authentication middleware that can be used with NodeJS. To handle image uploading, we used a library such as React Drop zone to provide a drag-and-drop interface for users.

**6.1.1 SIGN-IN**

Users can securely sign in to the web application by entering their credentials. Upon clicking the sign-in button, a post request is sent to the server (Node.js backend) for authentication. Once verified, the user is directed to the dashboard, gaining access to personalized features and functionalities.

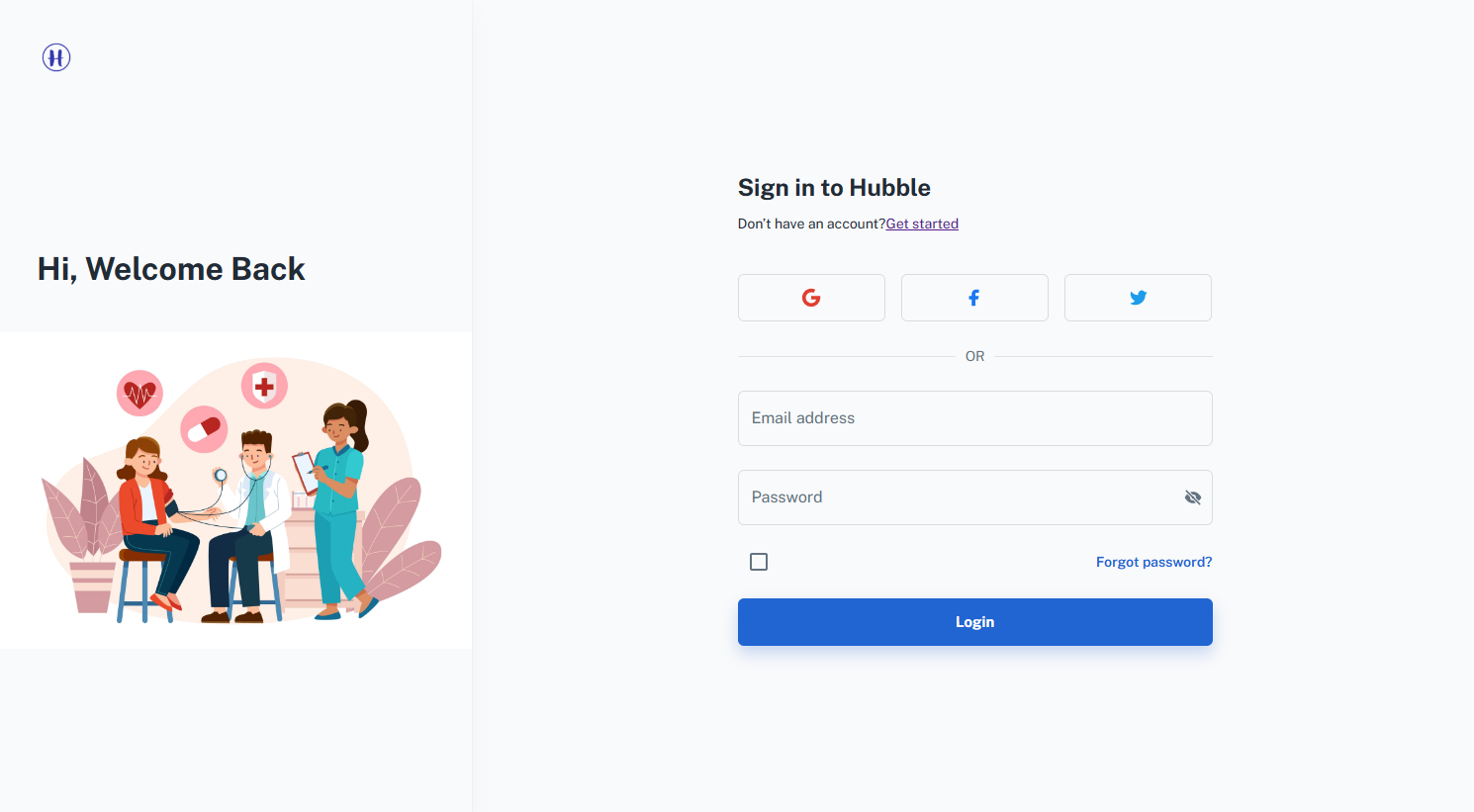


Fig 17 Sign-in Page

**6.1.2 SIGN-UP**

New users can create an account by providing their relevant information and choosing a unique username and password. The signup process involves sending a post request to the server, which handles user registration and stores the information securely in the database (MongoDB). Successful signup grants user's access to the application's features.

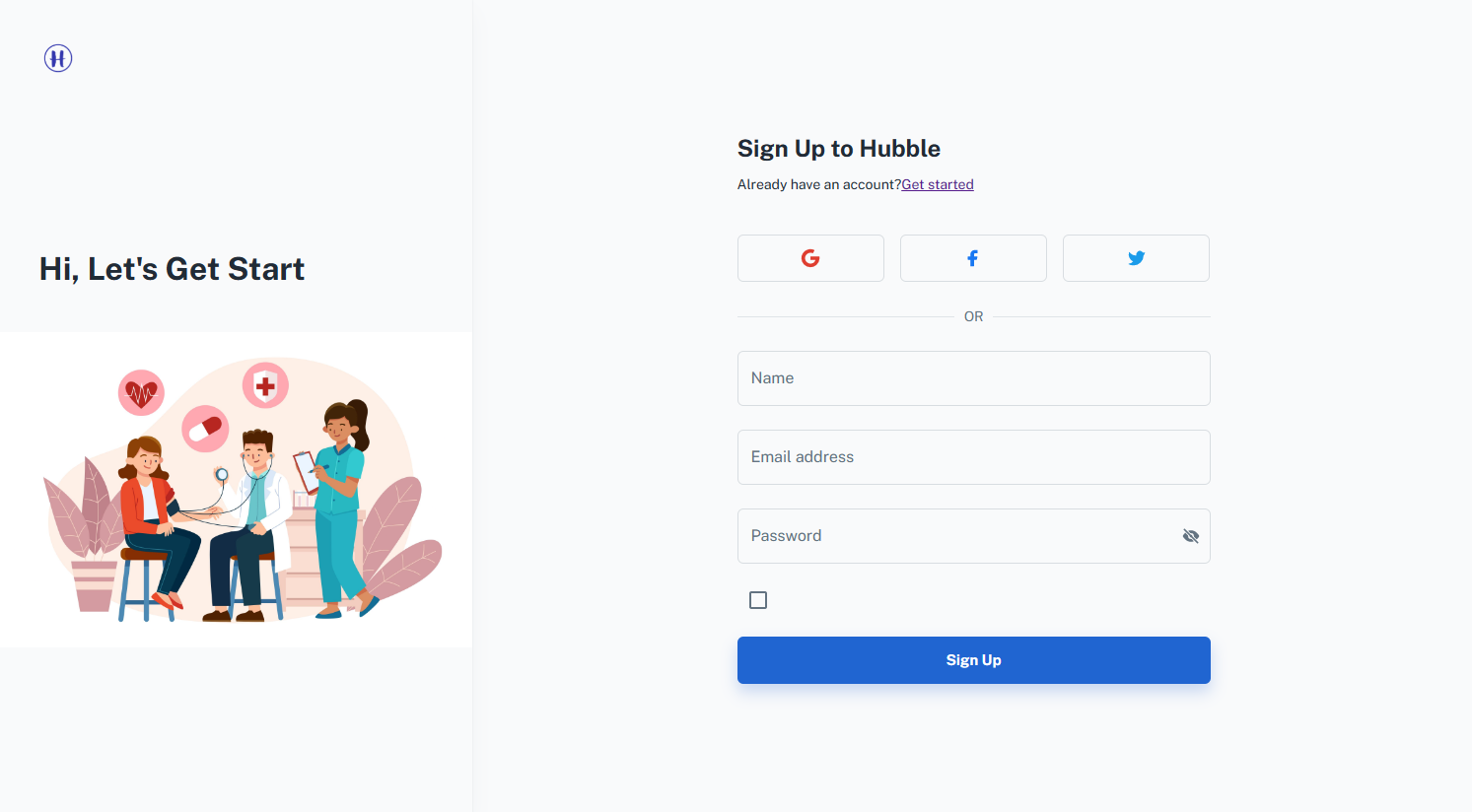


Fig 18 Sign Up Page

**6.1.3 DASHBOARD**

The dashboard serves as the central hub of the web application, providing users with an intuitive and personalized interface to access various features and information.

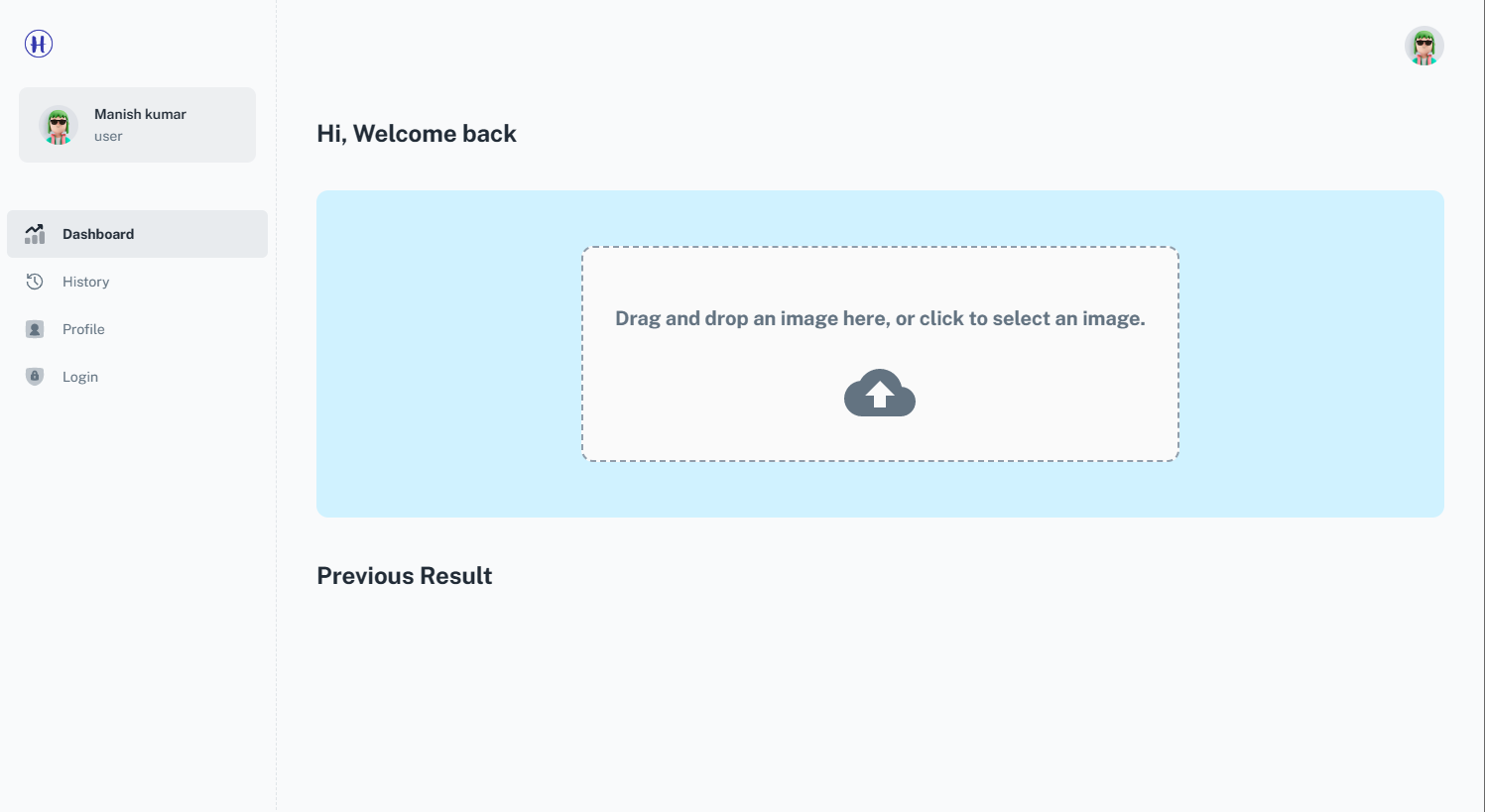


Fig 19 Application Dashboard

**6.1.4 SELECT AND UPLOAD IMAGE**

1. **Image Selection:** Users can conveniently select an image from their local device or choose from a gallery of previously uploaded images. This selection process enables users to pick the desired retinal image for disease prediction.
2. **Image Upload:** Once an image is selected, users can initiate the upload process by clicking a button or dragging and dropping the image onto the designated area. The system securely uploads the image to the server for further processing and analysis.

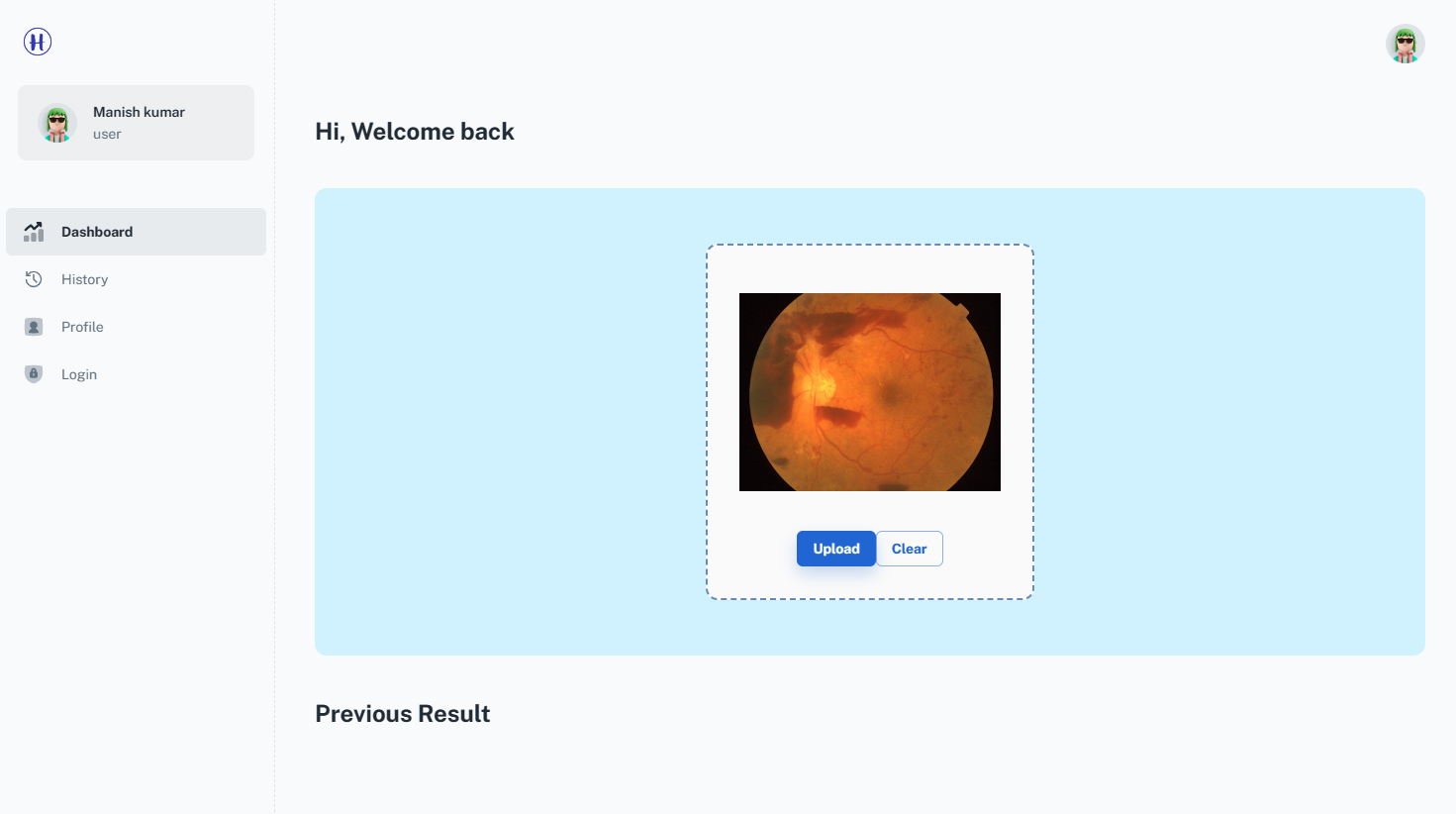


Fig 20 Select and Upload Image

**6.1.5 RESPONSE AND RESULT**

1. **Response Display:** After the image is uploaded and processed, the system provides a response to the user, indicating the outcome of the diabetic eye disease prediction.
2. **Result Visualization:** To enhance user understanding and engagement, the system may present the prediction results through visual representations, such as heatmaps, annotated images, or comparison charts. These visualizations help users interpret and analyze the prediction outcome in a more intuitive and informative manner.

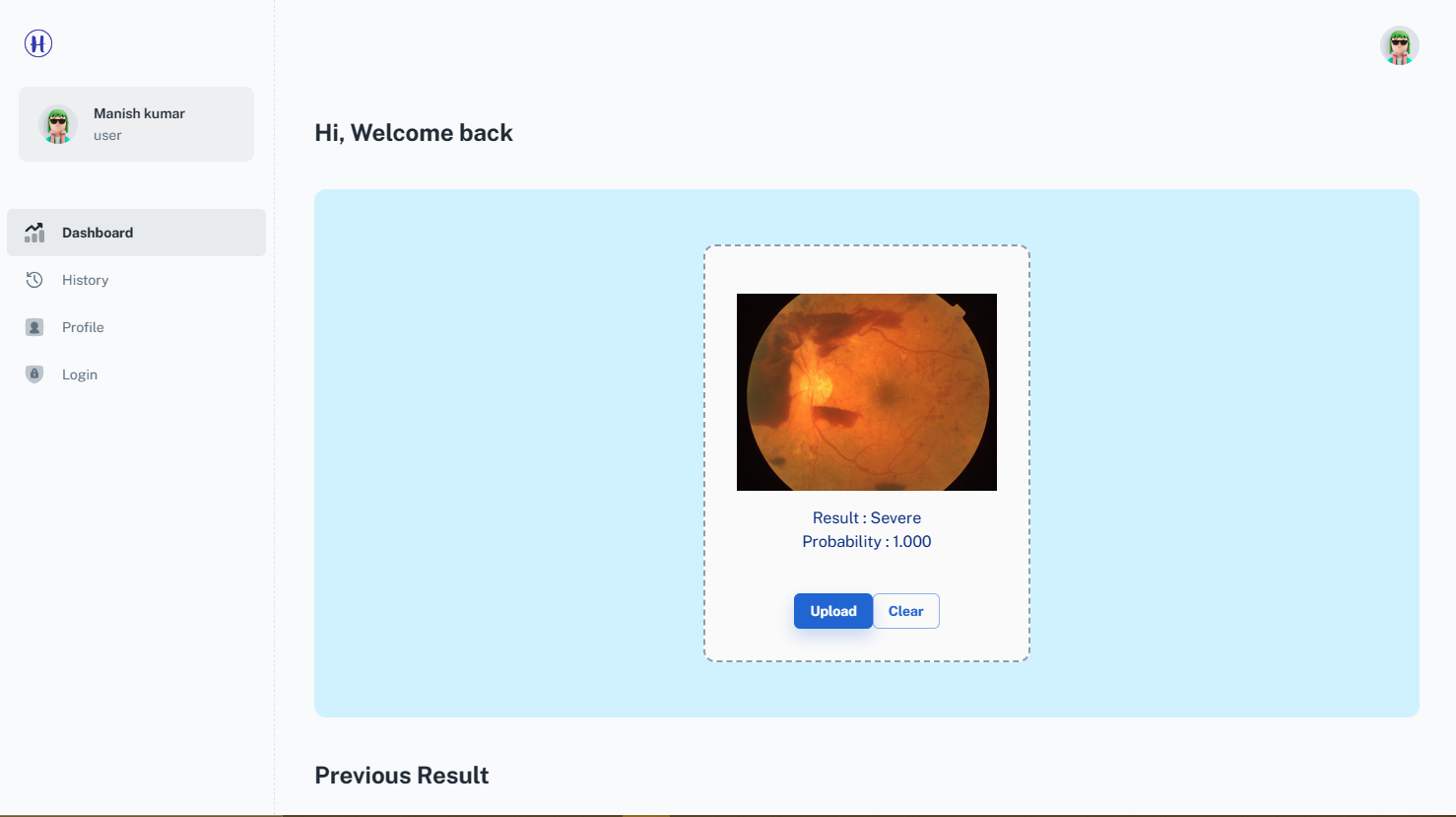


Fig 21 Output

**6.1.6 RECENT USER ACTIVITIES AND RESULTS**

1. **Activity Log:** The dashboard presents a chronological list of recent user activities, such as image uploads, disease predictions, profile updates, and any other relevant actions performed within the application. This log helps users review their recent interactions and track their engagement with the system.
2. **Result Summary:** Alongside each activity entry, a summary of the corresponding result or outcome is displayed. This summary may include the predicted disease condition, confidence score, any generated reports, or other pertinent information related to the activity. Users can quickly review their recent results and gain insights into their eye health status.

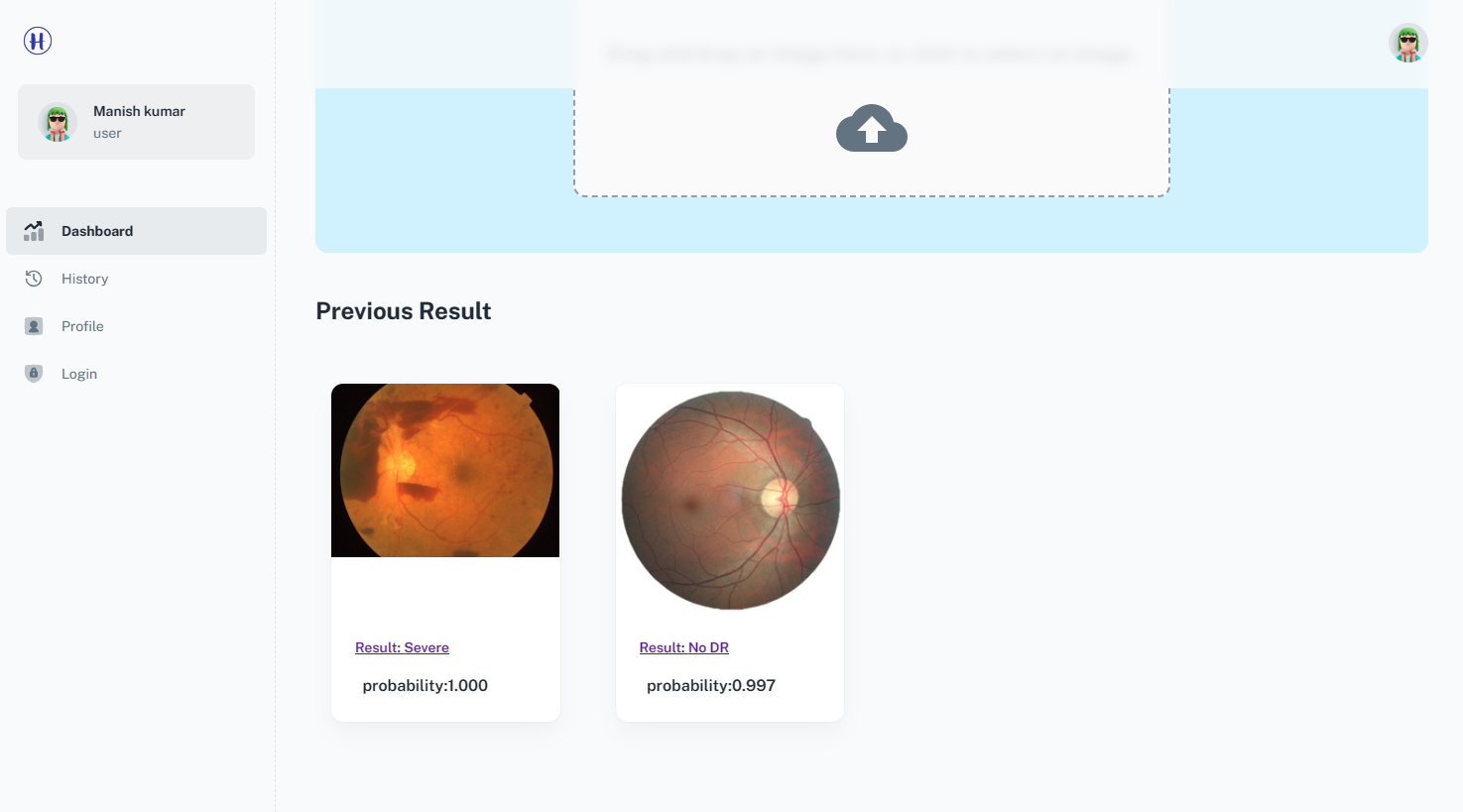


Fig 22 Layout of recent Status / Activities of User

**6.1.7 HISTORY**

The history section provides users with a detailed log or timeline of their previous actions and engagements within the application. This log may include activities such as image uploads, disease predictions, profile updates, and any other relevant actions performed by the user. It serves as a reference point for users to track their past interactions and monitor their usage history.

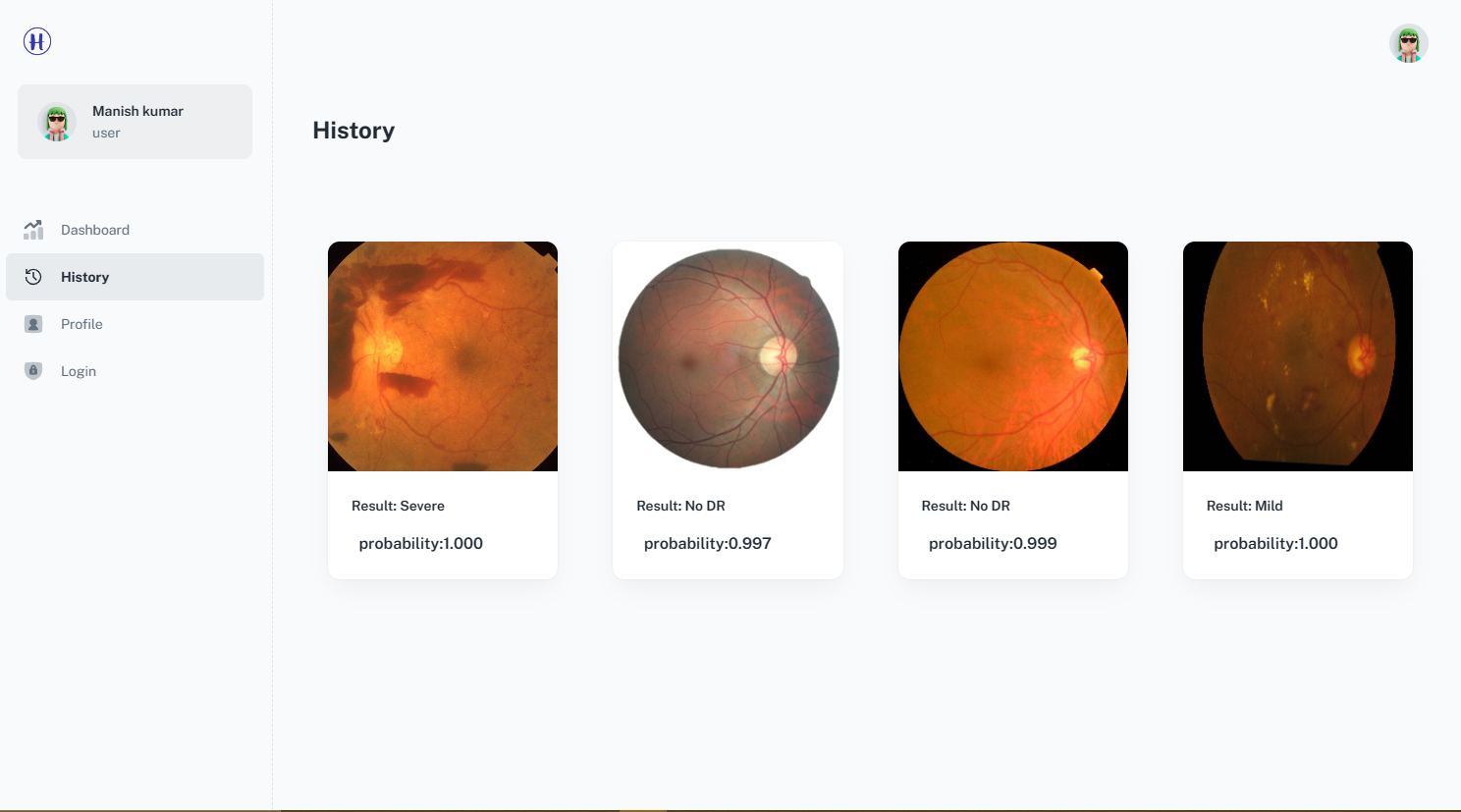


Fig 23 History

**6.1.8 USER PROFILE AND SETTINGS**

The user profile section allows individuals to manage their personal information, preferences, and settings within the web application. It provides a dedicated space for users to customize their profile and maintain accurate details.

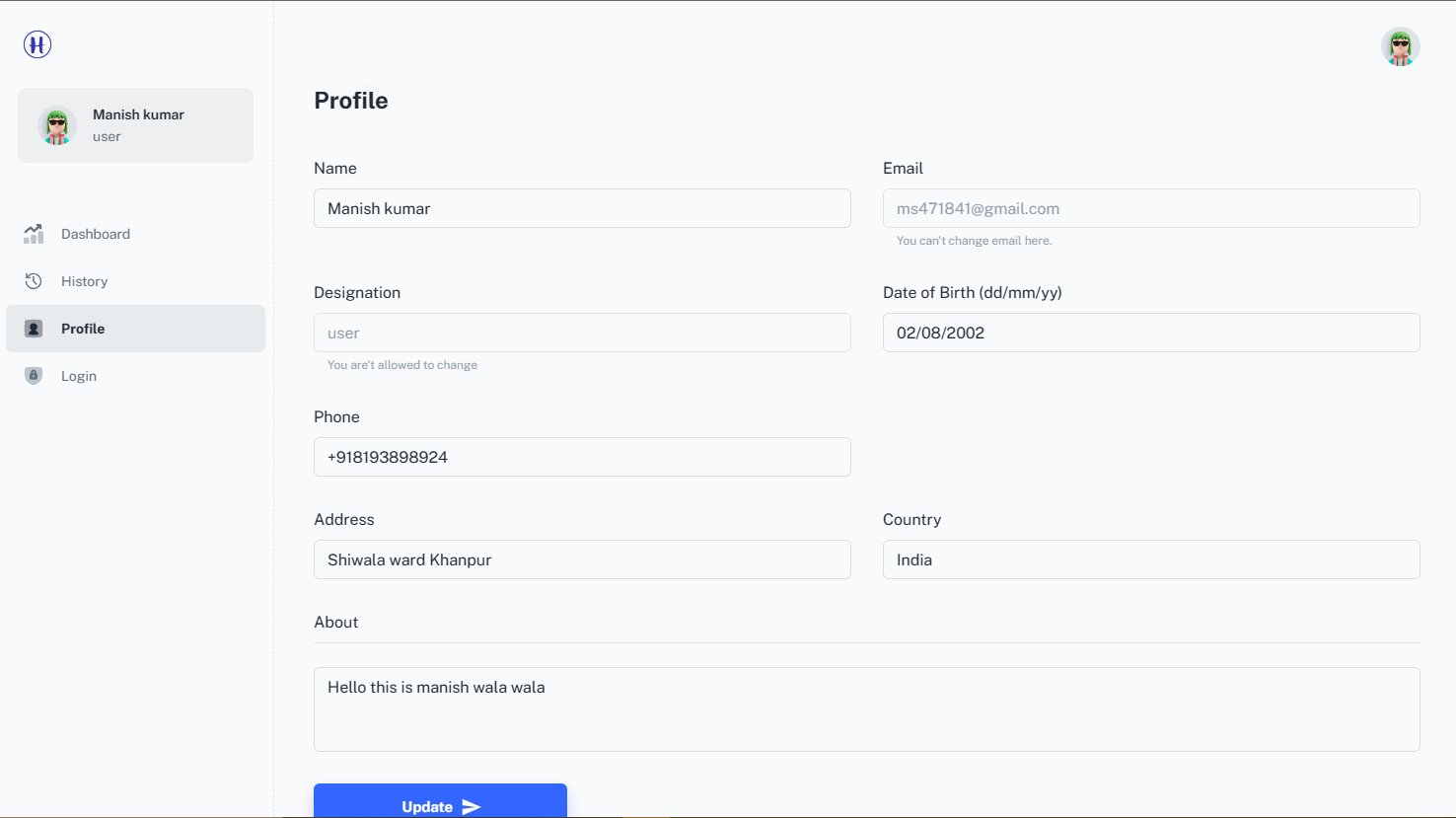


Fig 24 User Profile and Setting

**6.1.9 SIGN-OUT THE APPLICATION**

Upon selecting the sign-out option, the system initiates the log out procedure. This involves clearing any session or authentication tokens associated with the user's session, effectively terminating their active session. After successful sign-out, the user is redirected to the login page, preventing unauthorized access to their account. Sign-out ensures the privacy and security of user information and allows for a safe transition when ending their session within the application

**Conclusion**

In this project, "**Detects Diabetic Eye Disease in Retinal Images**," we have developed a web application that allows users to upload retinal images and receive predictions about diabetic eye disease using a pretrained machine learning model. By using ReactJS and Redux for the frontend, NodeJS for user authentication and database interactions, Flask for the Python backend, MongoDB for the database, and @mui/material for UI components, we were able to build a robust and scalable application that can handle multiple users and multiple images. We encountered several challenges during development, including integrating the machine learning model, managing user authentication and database interactions, but we were able to overcome these challenges and deliver a working application. In the future, we could add features such as patient management and doctor reporting to further enhance the application. Overall, this project demonstrates the potential of machine learning, web development technologies, and databases to create innovative healthcare.

**Future Work**

1. **Expanded Disease Detection:** Extend the application's capabilities to detect and predict other eye diseases, such as glaucoma, macular degeneration, or retinopathy, to provide a more comprehensive solution for users.
2. **Integration with Healthcare Systems:** Collaborate with healthcare institutions to integrate the application into existing diagnostic workflows and electronic health record (EHR) systems for seamless information exchange and improved patient management.
3. **Mobile Application Development:** Develop a mobile application version of the project to increase accessibility and convenience, allowing users to upload images, receive notifications, and access their eye health information on their smartphones.
4. **User Education and Engagement:** Provide educational resources and personalized recommendations within the application to raise awareness about eye diseases, promote proactive eye care practices, and engage users in their own health management.

By pursuing these future work directions, the application can enhance its performance, expand disease detection capabilities, integrate with healthcare systems, improve accessibility through a mobile app, enable remote consultations, educate users, and ultimately improve the diagnosis and management of eye diseases.

**REFERENCE**

1. <https://defauw.ai/diabetic-retinopathy-detection/>
2. <https://www.researchgate.net/publication/351066356_Diabetic_Retinopathy_Detection_Using_Retinal_Images>
3. <https://www.mdpi.com/1424-8220/21/16/5283>
4. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8398326/>
5. <https://www.researchgate.net/publication/353727114_Detection_of_Diabetic_Eye_Disease_from_Retinal_Images_Using_a_Deep_Learning_Based_CenterNet_Model>
6. <https://www.researchgate.net/publication/275347879_Fundus_Image_Analysis_for_the_Detection_of_Diabetic_Eye_Diseases-A_Review>
7. <https://www.wikipedia.org/>
8. [An introduction to machine learning with scikit-learn — scikit-learn 1.2.2 documentation](https://scikit-learn.org/stable/tutorial/basic/tutorial.html)
9. [Developer News | 404 - Page not found (freecodecamp.org)](https://www.freecodecamp.org/news/web-development-with-node-express-js-learn-by-building-89f5352018ca/)
10. <https://www.nature.com/articles/s41467-021-25138-w>
11. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10088017>
12. Some other resources