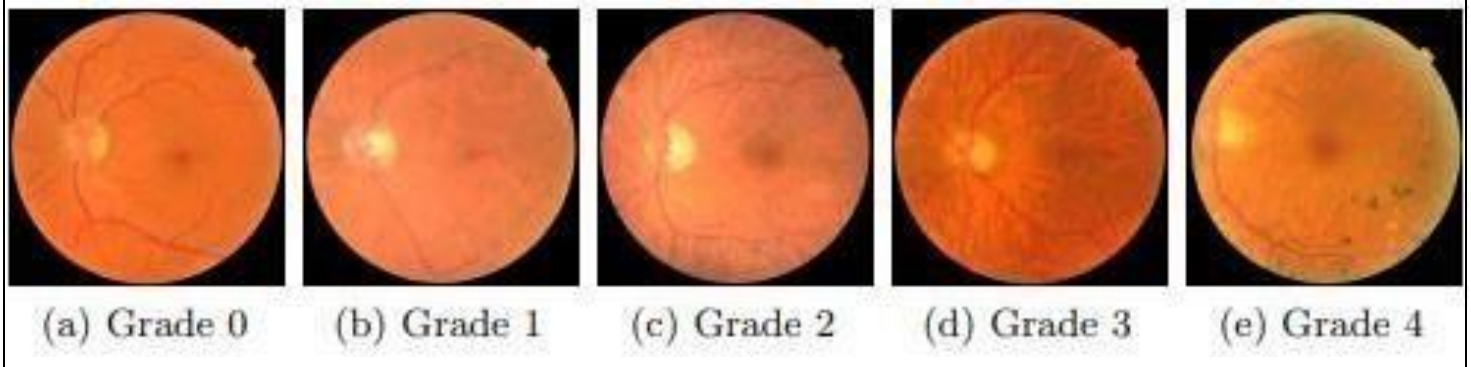
1. INTRODUCTION

1.1 Purpose of Project

Diabetic retinopathy (DR) is a type of ocular disease that can cause blindness due to damaged blood vessels in the back of the eye. The causes of DR are high blood pressure and high blood sugar concentration, which are very common in modern lifestyles. People with diabetes usually have higher risks of developing DR. In fact, one-third of diabetes patients show the symptoms of diabetic retinopathy according to recent studies. Therefore, early detection of DR is critical to ensure successful treatment. Unfortunately, detecting and grading diabetic retinopathy in practice is a laborious task, and DR is difficult to diagnose at an early stage even for professional ophthalmologists. As a result, developing a precise automatic DR diagnostic device is both necessary and advantageous.

Automated DR diagnosis systems take retinal images (fundus images) and yield DR grades. In the common retinal imaging dataset of DR, the grades of DR can be categorized into five stages: 0 - no DR, 1 - mild DR, 2 - moderate DR, 3 - severe DR, and 4 - proliferative DR. Specifically, the severity of DR is determined by taking the numbers, sizes, and appearances of lesions into account.



Grades of DR

1.2 Problems with Existing System

There are a few systems out there in the market which are trying to find out this type of ocular disease. We, with the help of the below mentioned modules will be implementing a system such that we can help in finding out the disease in the early stages.

The two main drawbacks of existing systems are Data Annotation and Scalability.

Data Annotation

The scarcity of high-quality annotated data remains a notorious challenge in medical image analysis due to the high cost of acquiring expert annotations.

Scalability

Supervised learning requires labeled data to predict outcomes for unknown data. However, it can need large datasets to build proper models and make accurate predictions. For large training datasets, manual data labeling can be challenging. Self-supervised learning can automate this process and handle this task with even massive amounts of data.

Proposed System

Analysis

Our project is based on detection of grade of DR. As discussed in the previous section there is not many ways to find and identify the grade of DR. We will be detecting this by getting an image of the back of the eye or called Retinography. Before detecting the grade of DR, we first must detect the back of eyes. This has 2 major steps:

1) Locating eyes
2) Determining the grade of DR

Methodology

The user provides the retro graphic image of the eye with a specialized camera and lenses. These images are then processed such that the model understands and recognizes the input provided by the user. These images are then predicted using the algorithms and the predefined data-sets which are highly capable of predicting the type and grade of the disease. These images can be further used to even identify whether the patient or the user might develop blindness. The output of this model is provided as a retinopathy grade output where the user can know all the information about the grade and condition of his eye or body.

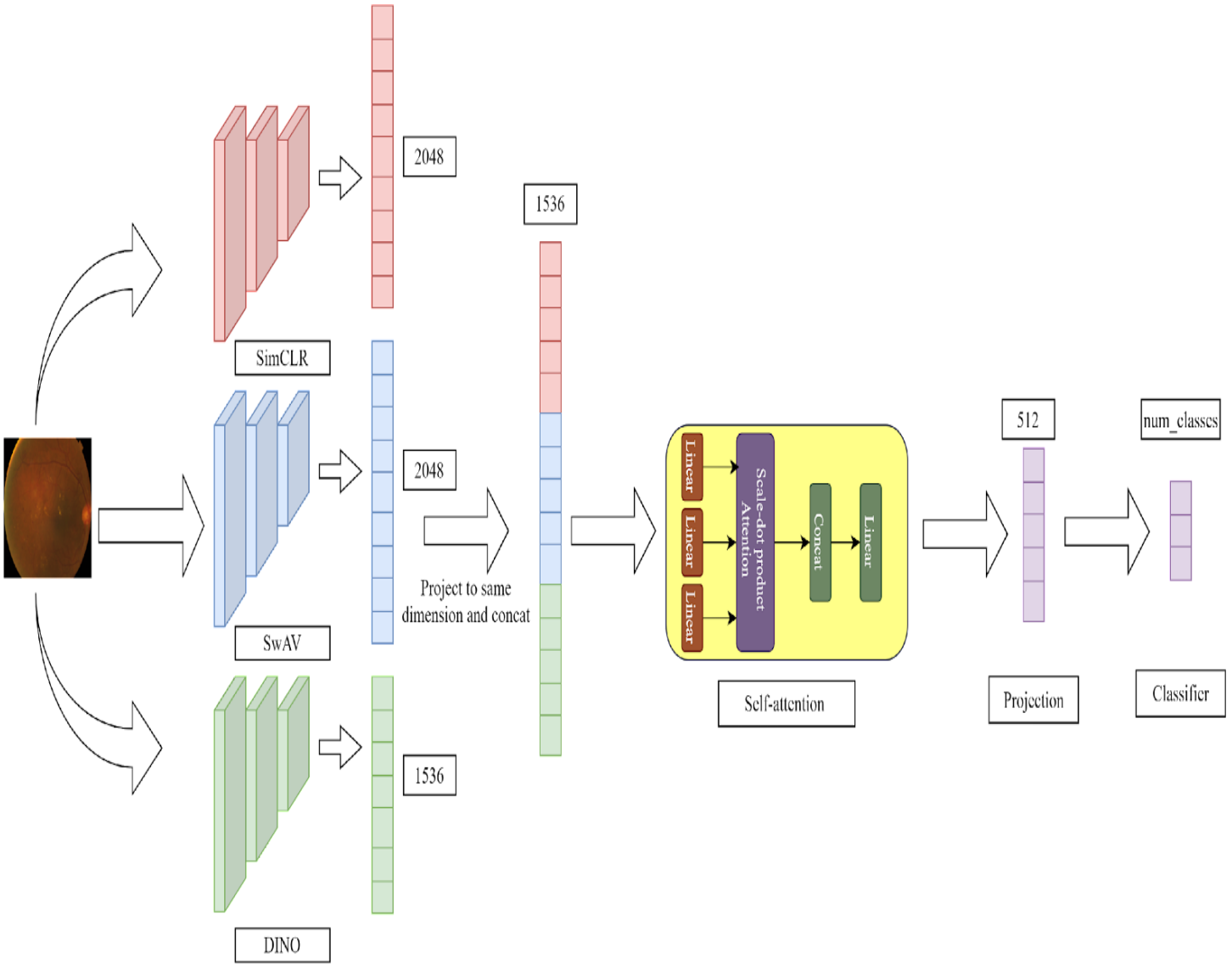
Our approach is by using Diabetic Retinopathy Datasets which takes the input images and then preprocesses these images such that the self-supervised pretraining can understand and define them. This supervised pretraining set images are then sent to downstream fine-tuning models where the tune the image and find the exact type and get the data from that image. This data is then used to classify the type and grade of the disease such that the user can get the exact information of the disease and can be used for further medical checks or records.

When we perform these techniques on 2 diabetic retinopathy datasets i.e., Aptos (contains 3662 images) and Eyepacs (contains 30,000 images). We first train the model with imagenet initialization on the datasets and then finetune to get the maximum accuracy. And when compared, the accuracy for supervised techniques the Eyepacs shows how easily it is scalable to more data.

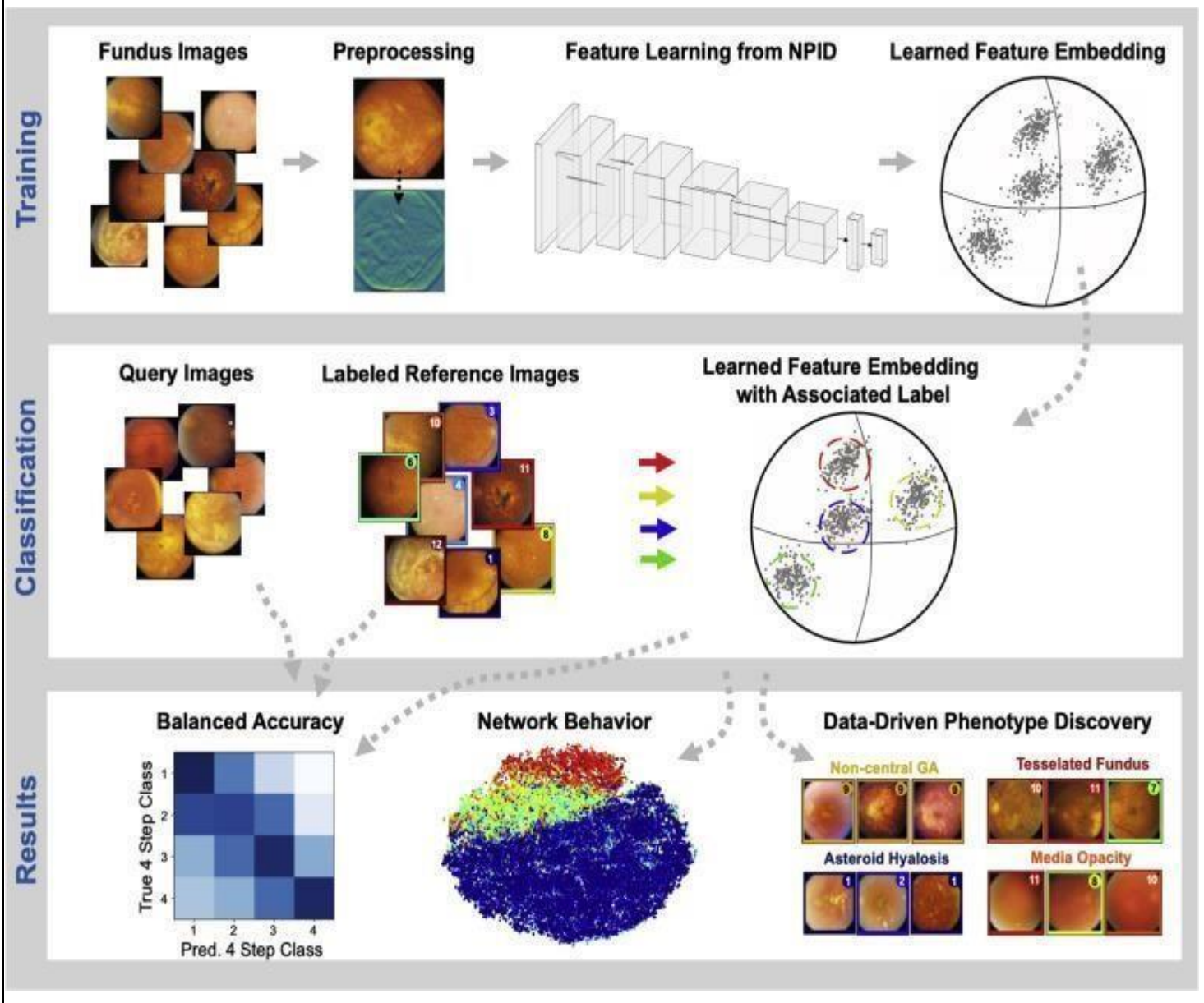
1.4 Scope of the Project

With the help of this model, we can help people to identify and find out whether they are currently suffering from the disease or what kinds of remedies or precautions they need to follow to avoid or prevent from these diseases.

1.5 Architectural Diagram



1.5 Architectural Diagram.



1.6 Schematic Diagram

CHAPTER - 2

2. LITERATURE SURVEY

A literature review is a survey of scholarly sources on a specific topic. It provides an overview of current knowledge, allowing you to identify relevant theories, methods, and gaps in the existing research that you can later apply to your paper, thesis, or dissertation topic.

1. A simple framework for contrastive learning of visual representations, T Chen, S Kornblith, M Norouzi (2020): Model used is SimCLR. It simplifies recently proposed contrastive self-supervised learning algorithms without specialized architectures or a memory bank. Advantage of this is a simple framework and disadvantage is needing more negative samples and more memory.

2. Barlow Twins Self-supervised learning via redundancy reduction, Jure Zbontar, Li Jing, Ishan Mishra, Yann Lecun (2021): Objective function that naturally avoids collapse by measuring the cross-correlation matrix between the outputs of two identical networks fed with distorted versions of a sample. Advantage of this is less dependent on Augmentation and disadvantages is dimensionality collapse and trivial solution.

3. Unsupervised learning of visual features by Contrasting Cluster assignments, Mathilde Caron, Ishan Mishra, Julien Mairal, Priya Goyal (2020): Unsupervised image representation have significantly reduced the gap with supervised pretraining, notably with the recent achievements of contrastive learning methods. Advantages of this is clusters the embeddings for a better loss function and disadvantages is needs more negative samples and more memory.

4. **Emerging properties in self-supervised vision transformers, Mathilde Caron, Hugo Touvron, Ishan Mishra, Herve Jegou (2021):** If self-supervised learning provides new properties to Vision Transformer (ViT) that stands out compared to convolutional networks. It contains explicit information about the sematic segmentation of an image, which does not emerge as clearly with supervised ViT, nor with convnets. Advantages of this is does not need negative samples and disadvantages is scales badly with higher dimensions.

CHAPTER - 3

3. SYSTEM REQUIREMENT SPECIFICATIONS

What is SRS?

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

Problem/Requirement Analysis:

The process is order and more nebulous of the two, deals with understand the problem, the goal, and constraints.

Requirement Specification:

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

Role of SRS

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

Requirements Specification Document

A Software Requirements Specification (SRS) is a document that describes the nature of a project, software, or application. In simple words, SRS document is a manual of a project provided it is prepared before you kick-start a project/application.

There are a set of guidelines to be followed while preparing the software requirement specification document. This includes the purpose, scope, functional and non functional requirements, software and hardware requirements of the project. In addition to this, it also contains the information about environmental conditions required, safety and security requirements, software quality attributes of the project etc.

The purpose of SRS (Software Requirement Specification) document is to describe the external behavior of the application developed or software. It defines the operations, performance and interfaces and quality assurance requirement of the application or software. The complete software requirements for the system are captured by the SRS. This section introduces the requirement specification document for Word Building Game using Alexa which enlists functional as well as non-functional requirements.

Functional Requirements

For documenting the functional requirements, the set of functionalities supported by the system are to be specified. A function can be specified by identifying the state at which data is to be input to the system, its input data domain, the output domain, and the type of processing to be carried on the input data to obtain the output data. Functional requirements define specific behavior or function of the application. Following are the functional requirements:

FR1) To take the eye fundus image as input and process them accordingly.

FR2) To compare the image with the dataset using supervised learning.

FR3) To categorize the output to the nearest result.

Non-Functional Requirements

A non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. Especially these are the constraints the system must work within. Following are the non-functional requirements:

NR1) To give the input as an acceptable eye fundus image.

NR2) To have a complete dataset.

NR3) To provide the output in accurate and efficienst manner.

Performance:

The performance of the developed applications can be calculated by using following methods: Measuring enables you to identify how the performance of your application stands in relation to your defined performance goals and helps you to identify the bottlenecks that affect your application performance. It helps you identify whether your application is moving toward or away from your performance goals. Defining what you will measure, that is, your metrics, and defining the objectives for each metric is a critical part of your testing plan.

Performance objectives include the following:

Response time, Latency throughput or Resource utilization.

Software Requirements

Operating System	:	Windows 10.
Coding Language	:	Python 3.10.11
Software (Framework)	:	Pytorch lighting
IDE	:	VS Code, Linux system
Packages	:	resnet50, resnet18, vit-L, vit-B, EfficientNet-B1

Hardware Requirements

Processor	:	Intel core i5 and above.
Hard Disk	:	16 GB or above.
RAM	:	8 GB or above.
Internet	:	4 Mbps or above (Wireless)

CHAPTER - 4

4. SYSTEM DESIGN

4.1 Introduction to UML

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic, and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows:

1. User Model View

This view represents the system from the users' perspective. The analysis representation describes a usage scenario from the end-users' perspective.

2. Structural Model View

In this model, the data and functionality are arrived from inside the system. This model view models the static structures.

3. Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting he interactions of collection between various structural elements described in the user model and structural model view.

4. Implementation Model View

In this view, the structural and behavioral as parts of the system are represented as they are to be built.

5. Environmental Model View

In this view, the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

4.2 UML Diagrams

4.2.1 Use Case Diagram

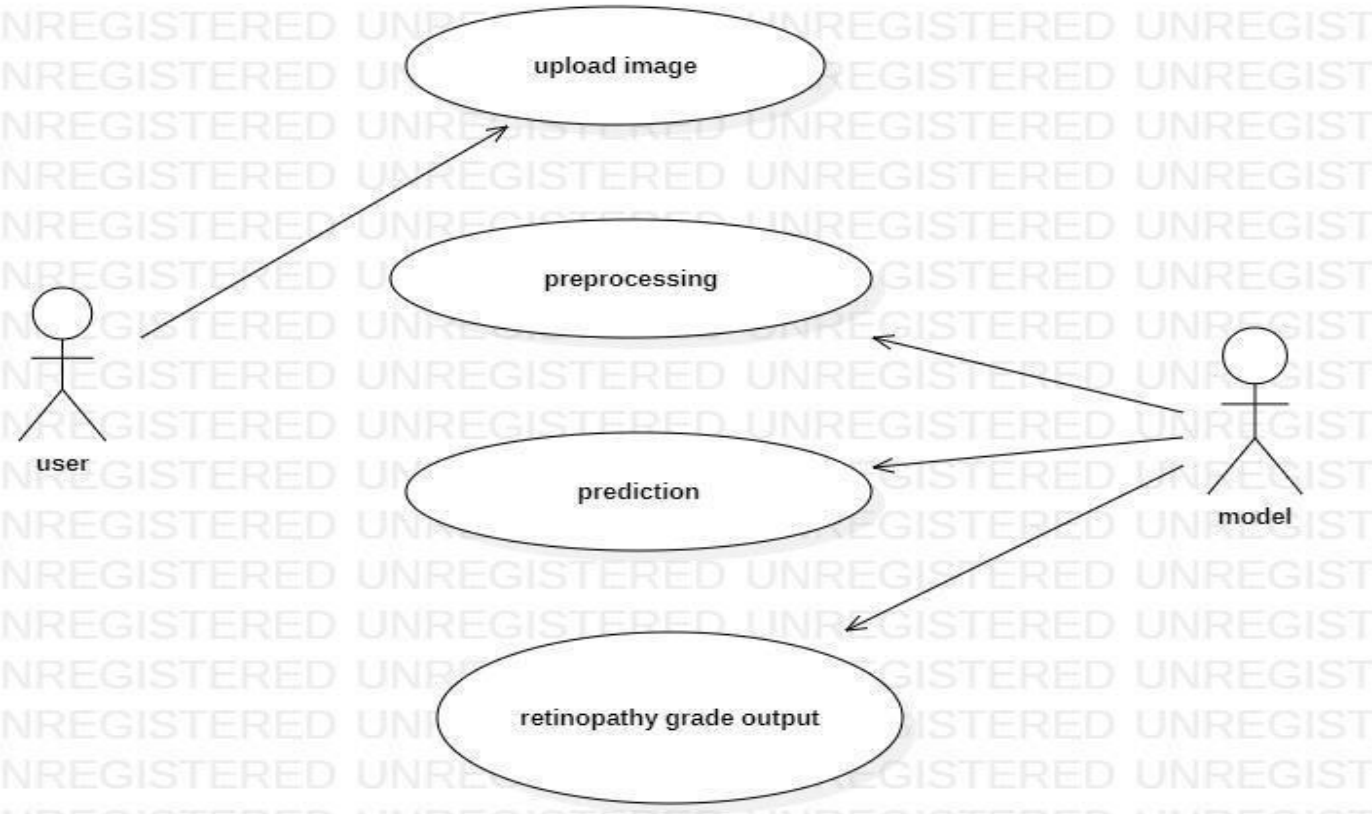
To model a system, the most important aspect is to capture the dynamic behavior. To clarify a bit in details, dynamic behavioral means the behavioral of the system when it is running/ operating.

So only static behavior is not sufficient to model a system rather dynamic behavior more important than static behavior. In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we must discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So use case diagrams are consisting of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So, to model the entire system numbers of use case diagrams are used.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So when a system is analyzed to gather its functionalities use cases are prepared and actors are identified. In brief, the purposes of use case diagrams can be as follows:

a) Used to gather requirements of the system.
b) Used to get an outside view of a system.
c) Identify external and internal factors influencing the system.
d) Show the interactions among the requirements are actors.



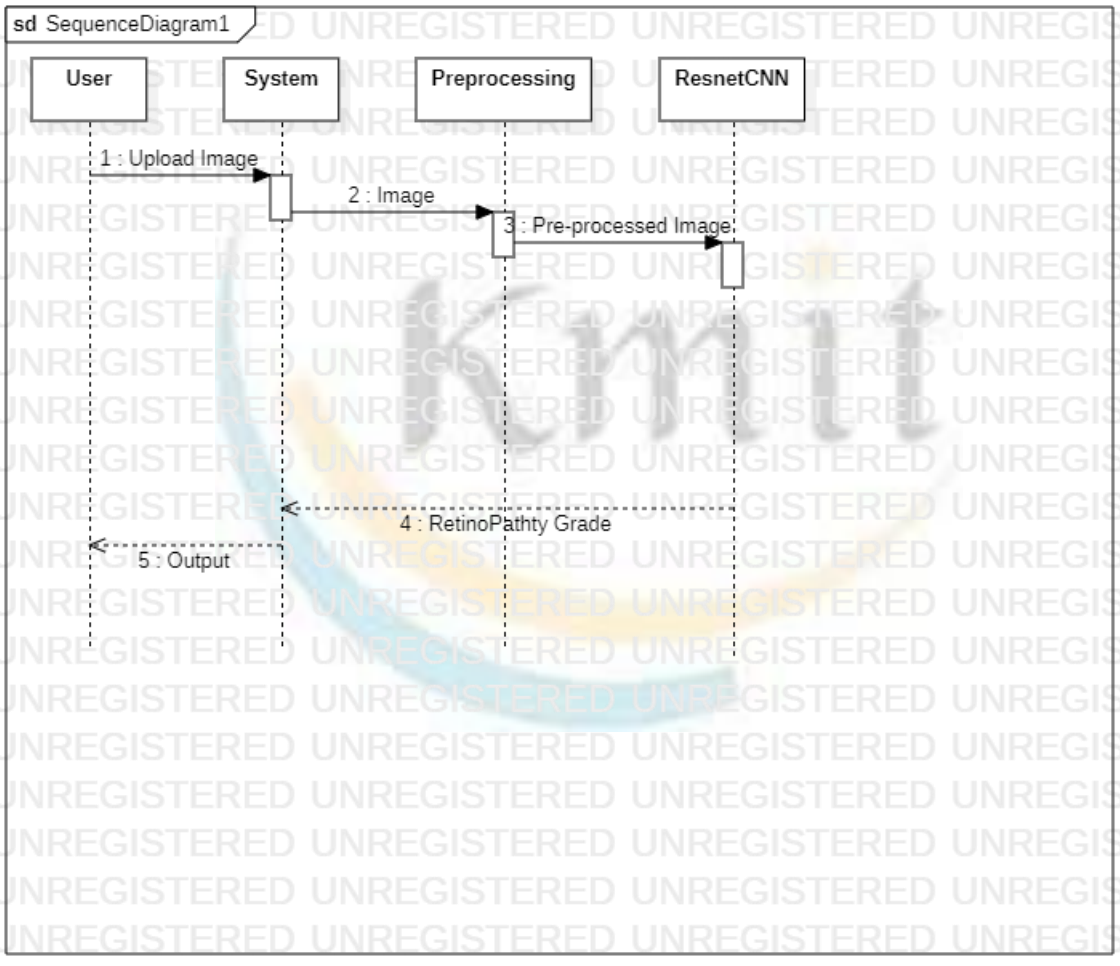
4.2.1 Use Case Diagram

4.2.2 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They are also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

Basic Sequence Diagram Notations:

- **Class role and participants**
- **Activation or Execution Occurrence**
- **Messages**
- **Lifelines**
- **Destroying objects**
- **Loops**

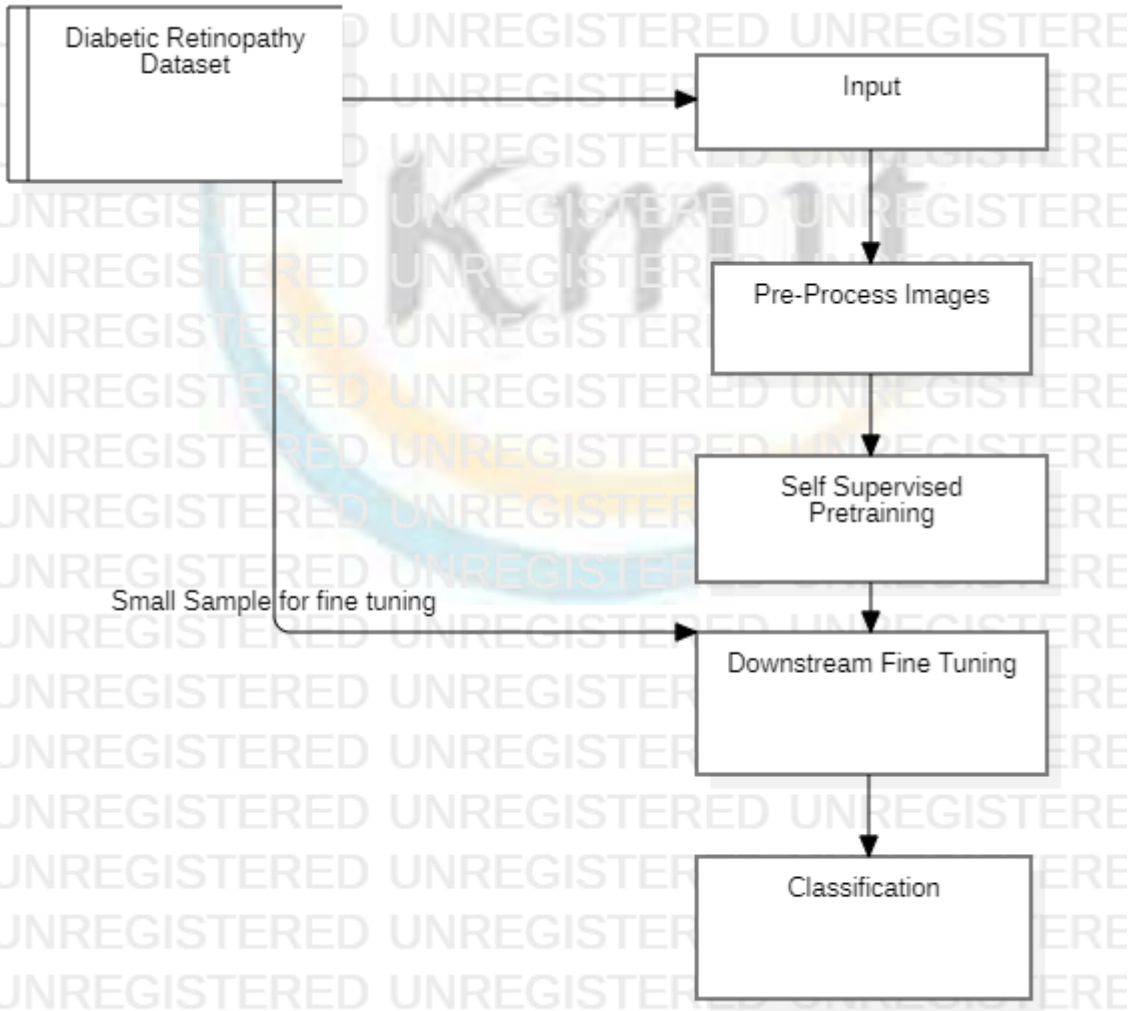


4.2.2 Sequence Diagram

4.2.3 Data Flow Diagram

A data flow diagram shows the way information flows through a process or system. It includes data inputs and outputs, data stores, and the various subprocesses the data moves through. DFDs are built using standardized symbols and notation to describe various entities and their relationships.

Data flow diagrams visually represent systems and processes that would be hard to describe in just words. You can use these diagrams to map out an existing system and make it better or to plan out a new system for implementation. Visualizing each element makes it easy to identify inefficiencies and produce the best possible system.



4.2.3 Data Flow Diagram

CHAPTER - 5

5. Code Snippets

5.1 ssl_simclr.py

```
import torch
from torch import nn
import torchvision
import pytorch_lightning as pl
from pytorch_lightning.loggers import WandbLogger

from lightly.data import LightlyDataset
from lightly.data import SimCLRCollateFunction
from lightly.loss import NTXentLoss
from lightly.models.modules import SimCLRProjectionHead


class SimCLR(pl.LightningModule):
    def__init_(self):
        super().__init__()
        resnet = torchvision.models.resnet18()
        self.backbone = nn.Sequential(*list(resnet.children())[:-1])
        self.projection_head = SimCLRProjectionHead(512, 2048, 2048)

        # enable gather_distributed to gather features from all gpus
        # before calculating the loss
        self.criterion = NTXentLoss(gather_distributed=True)

    def forward(self, x):
        x = self.backbone(x).flatten(start_dim=1)
        z = self.projection_head(x)
        return z

    def training_step(self, batch, batch_index):
        (x0, x1), _, _ = batch
        z0 = self.forward(x0)
```

```
        z1 = self.forward(x1)
        loss = self.criterion(z0, z1)
        self.log("loss", loss, on_epoch=True, on_step=True)
        return loss

    def configure_optimizers(self):
        optim = torch.optim.SGD(self.parameters(), lr=0.06)
        return optim


wandb_logger = WandbLogger(project="ssl_aptos")

model = SimCLR()

dataset = LightlyDataset(input_dir="../aptos/train_images_resize/")
# or create a dataset from a folder containing images or videos:
# dataset = LightlyDataset("path/to/folder")

collate_fn = SimCLRCollateFunction(
    input_size=32,
    gaussian_blur=0.0,
)

dataloader = torch.utils.data.DataLoader(
    dataset,
    batch_size=256,
    collate_fn=collate_fn,
    shuffle=True,
    drop_last=True,
    num_workers=8,
)

gpus = torch.cuda.device_count()
```

```
# train with DDP and use Synchronized Batch Norm for a more accurate batch norm
# calculation
trainer = pl.Trainer(
   max_epochs=250,  gpus=gpus,  strategy="ddp",  sync_batchnorm=True,
    loggers=wandb_logger
)
    trainer.fit(model=model, train_dataloaders=dataloader)
```

5.2 ssl_swav.py

```
import torch
from torch import nn
import torchvision
import pytorch_lightning as pl
from pytorch_lightning.loggers import WandbLogger

from lightly.data import LightlyDataset
from lightly.data import SwaVCollateFunction
from lightly.loss import SwaVLoss
from lightly.models.modules import SwaVProjectionHead
from lightly.models.modules import SwaVPrototypes


class SwaV(pl.LightningModule):
   def__init_(self):
      super().__init__()
      resnet = torchvision.models.resnet50()
      self.backbone = nn.Sequential(*list(resnet.children())[:-1])
      self.projection_head = SwaVProjectionHead(2048, 512, 128)
      self.prototypes = SwaVPrototypes(128, n_prototypes=512)

      self.criterion = SwaVLoss(sinkhorn_gather_distributed=True)
```

```
    def forward(self, x):
        x = self.backbone(x).flatten(start_dim=1)
        x = self.projection_head(x)
        x = nn.functional.normalize(x, dim=1, p=2)
        p = self.prototypes(x)
        return p

    def training_step(self, batch, batch_idx):
        self.prototypes.normalize()
        crops, _, _ = batch
        multi_crop_features = [self.forward(x.to(self.device)) for x in crops]
        high_resolution = multi_crop_features[:2]
        low_resolution = multi_crop_features[2:]
        loss = self.criterion(high_resolution, low_resolution)
        self.log("loss", loss, on_epoch=True, on_step=True)
        return loss

    def configure_optimizers(self):
        optim = torch.optim.Adam(self.parameters(), lr=0.001)
        return optim


wandb_logger = WandbLogger(project="ssl_aptos")

model = SwaV()

dataset = LightlyDataset(input_dir="../aptos/train_images_resize/")

collate_fn = SwaVCollateFunction()

dataloader = torch.utils.data.DataLoader(
    dataset,
    batch_size=128,
    collate_fn=collate_fn,
```

```
    shuffle=True,
    drop_last=True,
    num_workers=8,
)

gpus = [4, 5, 6, 7]

trainer = pl.Trainer(
    max_epochs=150,  gpus=gpus,  strategy="ddp",  sync_batchnorm=True,
     logger=wandb_logger
)
     trainer.fit(model=model, train_dataloaders=dataloader)
```

5.3 ssl_dino.py

```
import copy

import torch
from torch import nn
import torchvision
import pytorch_lightning as pl
from pytorch_lightning.loggers import WandbLogger

from lightly.data import LightlyDataset
from lightly.data import DINOCollateFunction
from lightly.loss import DINOLoss
from lightly.models.modules import DINOProjectionHead
from lightly.models.utils import deactivate_requires_grad
from lightly.models.utils import update_momentum


class DINO(pl.LightningModule):
    def__init_(self):
        super().__init__()
```

```
    resnet = torchvision.models.resnet50()
    backbone = nn.Sequential(*list(resnet.children())[:-1])
    input_dim = 2048

    self.student_backbone = backbone
    self.student_head = DINOProjectionHead(
        input_dim, 512, 64, 2048, freeze_last_layer=1
    )
    self.teacher_backbone = copy.deepcopy(backbone)
    self.teacher_head = DINOProjectionHead(input_dim, 512, 64, 2048)
    deactivate_requires_grad(self.teacher_backbone)
    deactivate_requires_grad(self.teacher_head)

    self.criterion = DINOLoss(output_dim=2048, warmup_teacher_temp_epochs=5)

def forward(self, x):
    y = self.student_backbone(x).flatten(start_dim=1)
    z = self.student_head(y)
    return z

def forward_teacher(self, x):
    y = self.teacher_backbone(x).flatten(start_dim=1)
    z = self.teacher_head(y)
    return z

def training_step(self, batch, batch_idx):
    update_momentum(self.student_backbone, self.teacher_backbone, m=0.99)
    update_momentum(self.student_head, self.teacher_head, m=0.99)
    views, _, _ = batch
    views = [view.to(self.device) for view in views]
    global_views = views[:2]
    teacher_out = [self.forward_teacher(view) for view in global_views]
    student_out = [self.forward(view) for view in views]
    loss = self.criterion(teacher_out, student_out, epoch=self.current_epoch)
```

```
        self.log("loss", loss, on_step=True, on_epoch=True)
        return loss

    def on_after_backward(self):

     self.student_head.cancel_last_layer_gradients(current_epoch=self.current_epoch)

    def configure_optimizers(self):
        optim = torch.optim.Adam(self.parameters(), lr=0.001)
        return optim


model = DINO()

dataset = LightlyDataset(input_dir="../aptos/train_images_resize/")

collate_fn = DINOCollateFunction()

dataloader = torch.utils.data.DataLoader(
    dataset,
    batch_size=64,
    collate_fn=collate_fn,
    shuffle=True,
    drop_last=True,
    num_workers=8,
)

gpus = [0, 1, 2, 3]

wandb_logger = WandbLogger(project="ssl_aptos")

trainer = pl.Trainer(
    max_epochs=250,
    gpus=gpus,
```

```
    strategy="ddp",
    sync_batchnorm=True,
    replace_sampler_ddp=True,
    logger=wandb_logger,
)
     trainer.fit(model=model, train_dataloaders=dataloader)
```

5.4 app.py

```
import torch
import numpy as np
import matplotlib.pyplot as plt

import albumentations as A
from albumentations.pytorch import ToTensorV2
from PIL import Image

from pytorch_grad_cam import GradCAM
from pytorch_grad_cam.utils.model_targets import ClassifierOutputTarget
from pytorch_grad_cam.utils.image import show_cam_on_image

import streamlit as st

st.set_option("deprecation.showfileUploaderEncoding", False)


@st.cache(allow_output_mutation=True)
def load_model():
    return torch.load("dretino.pt")


model = load_model()
```

```
st.write(
    """

    # Web Demo for Self-Supervised Learning on Diabetic Retinopathy Images

    ### [GitHub](https://github.com/Maahi10001/D-Retino)

    """
)

transforms = A.Compose(
    [
        A.Resize(height=224, width=224),
        A.Normalize(
            mean=(0.5211, 0.2514, 0.0809),
            std=(0.2653, 0.1499, 0.0861),
            max_pixel_value=255.0,
        ),
        ToTensorV2(),
    ]
)

file = st.file_uploader("", type=["png", "jpg"])


with open("./assets/0.png", "rb") as file1:
    btn = st.download_button(
        label="Download Sample-1 Image", data=file1, file_name="0.png",
      mime="image/png"
    )

with open("./assets/1.png", "rb") as file2:
    btn = st.download_button(
```

```
        label="Download Sample-2 Image", data=file2, file_name="1.png",
      mime="image/png"
    )


@st.cache(allow_output_mutation=True)
def generate_cam():
    target_layer = [model.layer4[-1]]

    cam = GradCAM(model=model, target_layers=target_layer, use_cuda=False)
    return cam


cam = generate_cam()


def deprocess_image(img):
    img = img - np.mean(img)
    img = img / (np.std(img) + 1e-5)
    img = img * 0.1
    img = img + 0.5
    img = np.clip(img, 0, 1)
    return img


def dretino(img):
    img = transforms(image=img)["image"]
    img = torch.tensor(img)

    img = torch.unsqueeze(img, 0)

    model.eval()
    grayscale_cam = cam(input_tensor=img)
```

```
    rgb_img = np.transpose(deprocess_image(img[0].numpy()), (1, 2, 0))
    viz = show_cam_on_image(rgb_img, grayscale_cam[0], use_rgb=True)

    logits = model(img)

    preds = torch.nn.Softmax(dim=1)(logits.data)[0]
    res = {"No": 0, "Mild": 0, "Moderate": 0, "Severe": 0, "Proliferative": 0}
    for idx, key in enumerate(res.keys()):
        res[key] = preds[idx]

    return res, viz


if file is None:
    st.text("Please upload an image file")
else:
    st.text("Uploaded Image")
    image = np.asarray(Image.open(file))
    if image.shape[-1] == 4:
        image = image.convert("RGB")
    st.image(image, use_column_width=True)
    res, viz = dretino(image)
    fig, ax = plt.subplots()
    ax.barh(list(res.keys()), np.array(list(res.values())))
    ax.grid(visible=True)
    st.text("Predicted Result")
    res_ = {key: float("{:.2f}".format(item.item())) for key, item in res.items()}
    st.write(res_)
    st.write(fig)
    st.text("GradCam Output")
    st.write(
        """
    For more information about GradCam visit
```

[GradCam-Book](https://jacobgil.github.io/pytorch-gradcam-book/introduction.html)

"""

)

st.image(viz, use_column_width=True)

CHAPTER -6

6. TESTING

6.1 Introduction to Testing

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

Who does Testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing. In most cases, the following professionals are involved in testing a system within their respective capacities:

- **Software Tester**
- **Software Developer**
- **Project Lead/Manager**
- **End User**

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

- **Functional Testing**
- **Non-Functional Testing**

6.2 Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a

software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

6.3 Software Testing Life Cycle

The process of testing a software in a well-planned and systematic way is known as software testing lifecycle (STLC). Different organizations have different phases in STLC however generic Software Test Life Cycle (STLC) for waterfall development model consists of the following phases.

- **Requirements Analysis**
- **Test Planning**
- **Test Analysis**
- **Test Design**

Requirements Analysis

In this phase testers analyze the customer requirements and work with developers during the design phase to see which requirements are testable and how they are going to test those requirements. It is very important to start testing activities from the requirements phase itself because the cost of fixing defect is very less if it is found in requirements phase rather than in future phases.

Test Planning

In this phase all the planning about testing is done like what needs to be tested, how the testing will be done, test strategy to be followed, what will be the test environment, what test methodologies will be followed, hardware and software availability, resources, risks etc. A high level test plan document is created which includes all the planning inputs mentioned above and circulated to the stakeholders.

Test Analysis

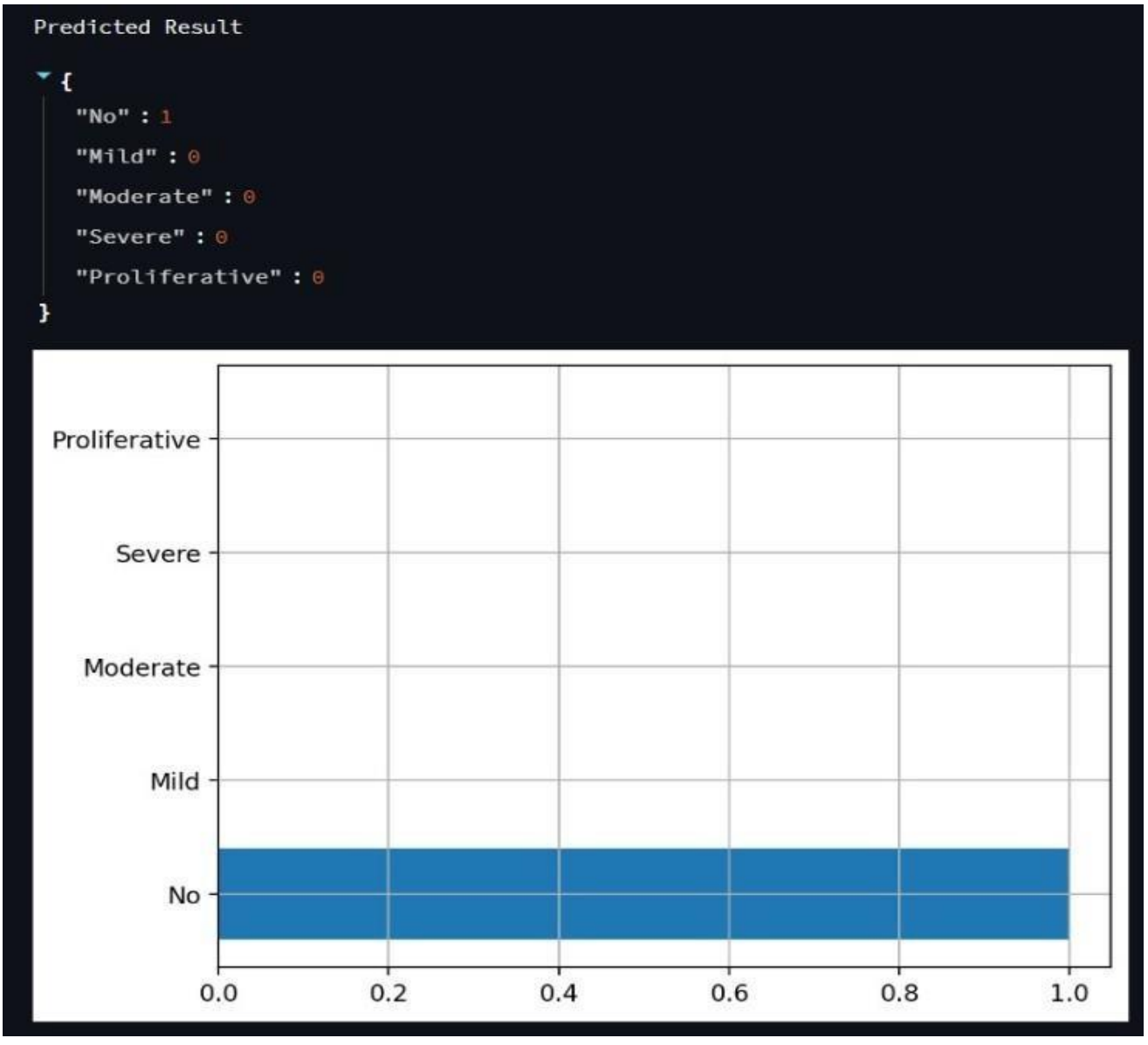
After test planning phase is over test analysis phase starts, in this phase we need to dig deeper into project and figure out what testing needs to be carried out in each SDLC phase. Automation activities are also decided in this phase, if automation needs to be done for software product, how will the automation be done, how much time will it take to automate

and which features need to be automated. Non functional testing areas(Stress and performance testing) are also analyzed and defined in this phase Test Design.

In this phase various black-box and white-box test design techniques are used to design the test cases for testing, testers start writing test cases by following those design techniques, if automation testing needs to be done then automation scripts also needs to written in this phase.

6.4 Test Cases

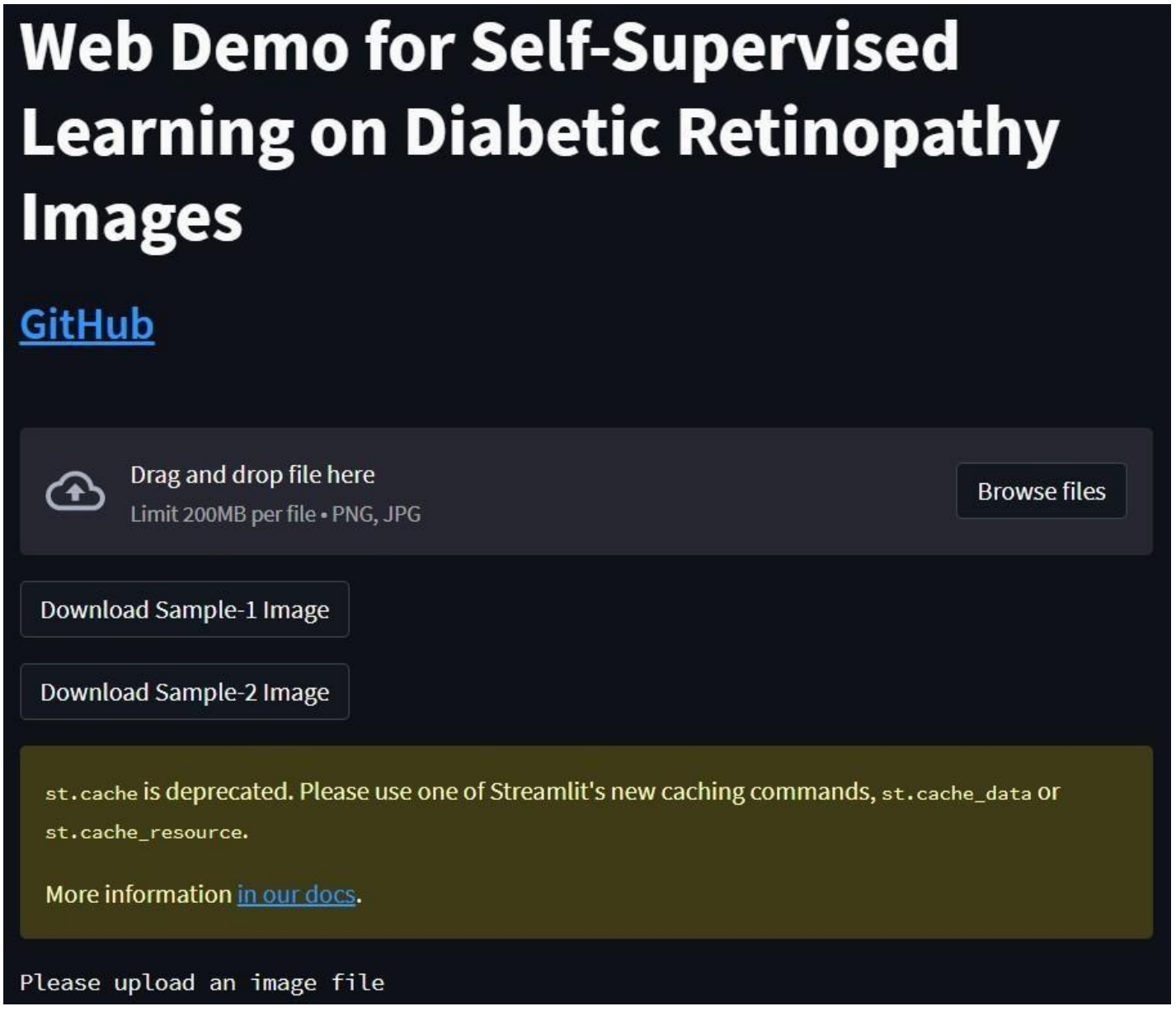
- **The results are displayed in the form of graphs where it shows the level of DR. In this case its "Not affecting."**



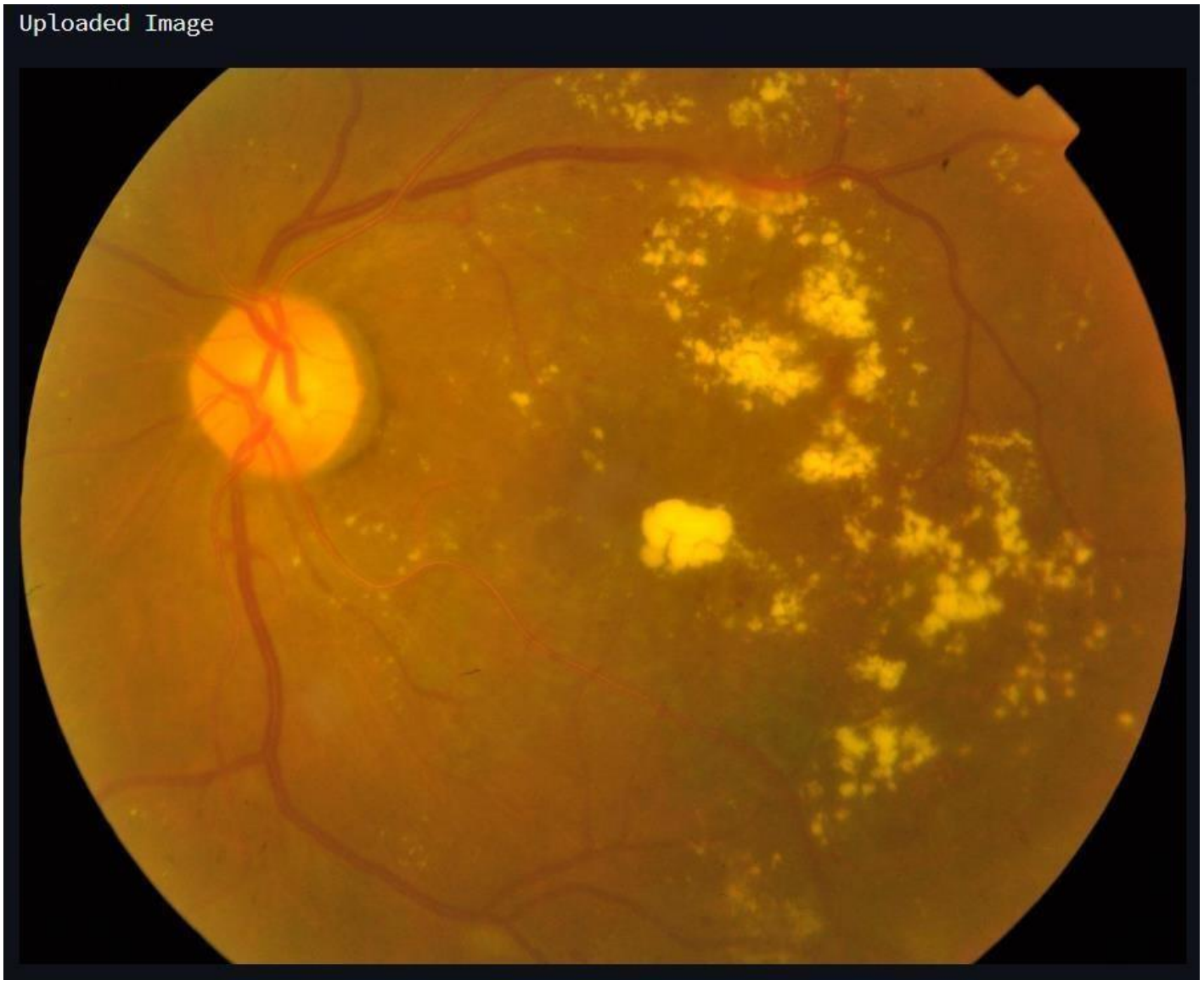
CHAPTER - 7

7. SCREENSHOTS

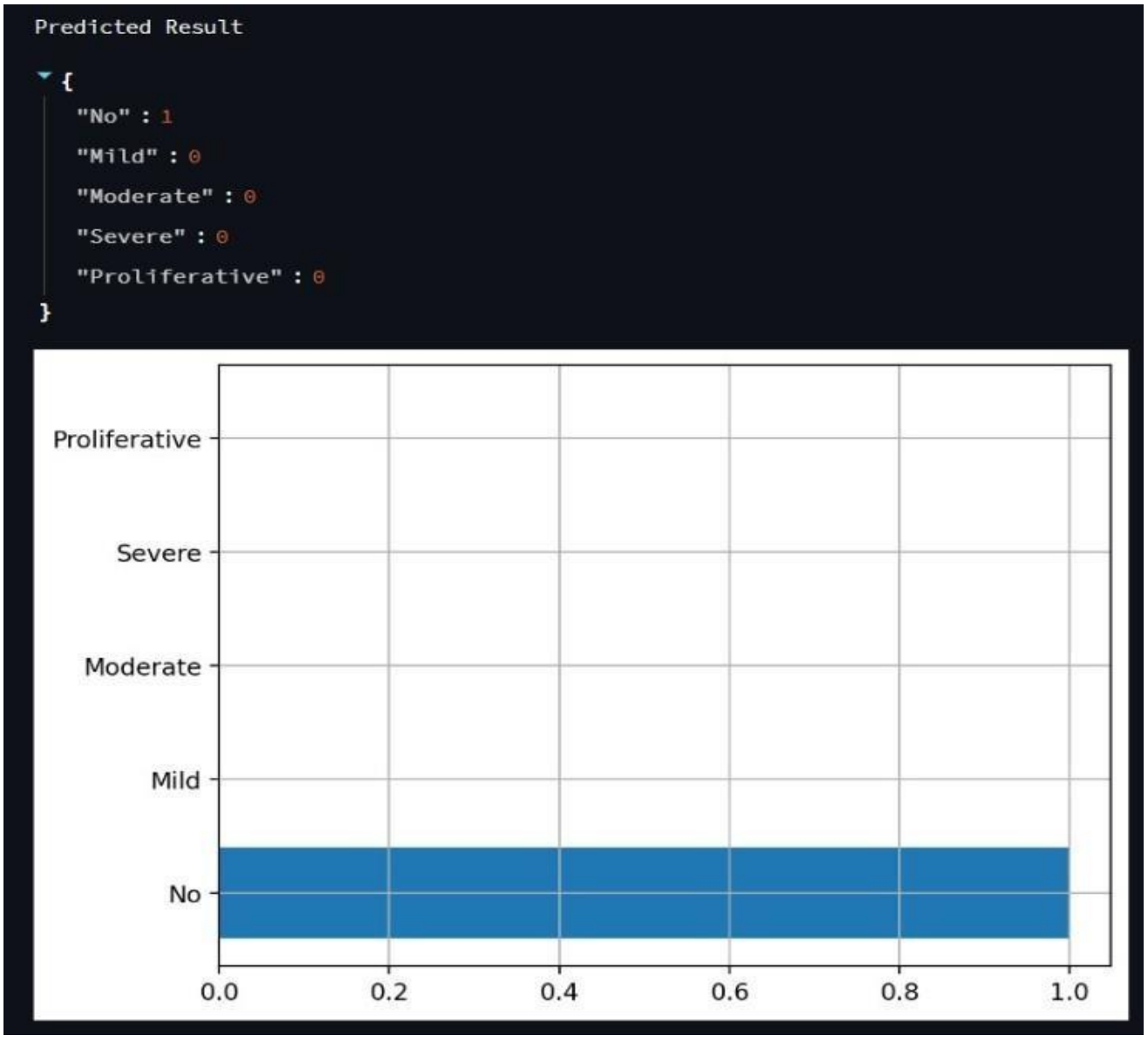
- **The Loading Page**



- **After uploading the image**



- **The results are displayed**



CHAPTER - 8

CONCLUSION & FUTURE OF THE PROJECT

Diabetic Retinopathy technique results are dependent on many factors such as intensity, texture, and image content. In our project a fast and efficient method is used for removing the disturbance from the eye fundus images and comparing with the data present in the model. The simulation results on retinal datasets that the proposed methodology can work with the eye fundus images to reduce the human errors or to provide services with a better model. Our project demonstrates an important screening tool for early detecting of a Diabetic Retinopathy. The proposed method exhibits less computational time to detect automatically the important clinical features of retinal images.

The future work can be concentrated by combining unsupervised and supervised detection methods in order to have better results. By achieving these methods, we can get a much intensified and a better image which can also be used to treat or check for other eye related diseases. This method also helps the ophthalmologists get a better look at the eye which can help them by minimizing those small human errors which might occur.

CHAPTER - 9

REFERENCES

[1] V.Kumari And N.Suriyanarayanan, "Feature Extraction For Early Detection Of Diabetic Retinopathy," In International Conference On Recent Trends In Information, Telecommunication And Computing, 2010, Pp. 359-361.

[2] S.Ravishankar, "Automated Feature Extraction For Early Detection Of Diabetic Retinopathy In Fundus Images" In Computer Vision And Pattern Recognition, 2009. CVPR 2009. IEEE Conference On 2009, Pp. 210-217.

[3] K.V.Kauppi T, Lensu L, Sorri I, Raninen A, "Diaretdb1: Diabetic Retinopathy Database And Evaluation Protocol. In: Medical Image Understanding And Analysis (MIUA)." Ed, 2007.

[4] Younis N, Broadbent Dm, Harding Sp, Vora Jp. "Incidence Of Sight-Threatening Retinopathy In Type 1 Diabetes In A Systematic Screening Programme." Diabet Med2003; 20:758-65.

[5] Berrichi Fatima Zohra, Benyettou Mohamed; "Automatic Diagnosis of Retinal Images Using The SupportVector Machine (SVM)".

[6] Sinthanayothin C, Boyce Jf, Williamson Th, Cook Hl, Mensah E, Lals. "Automated Detection Of Diabetic Retinopathy On Digital Fundus Image." Jdiabet Med 2002; 19:105–12.

[7] Niemeijer M, Van Ginneken B, Staal J, Suttorp-Schulten Ms, Abramoffmd. "Automatic Detection OfRed Lesions In Digital Color Fundus Trans Med Image Photographs." IEEE2005; 24:584–92.

[8] Neeraj Sharma And Lalit M.Aggarwal, "Automated Medical Image Segmentation Techniques" Journal Of Medical Physics, 2010 Jan-Mar.

[9] Gonzalez, R.C., R.E. Woods and S.L. Eddins, 2004. "Digital Image Processing 2nd Edn.,"Pearson Education India.