Final Project Report

Implementation of Deep Learning Method for Prediction of Cardiovascular Disease (IDMPCVD)

Project Supervisor
Dr. Saima Munawar

Submitted By

Group Project ID: F210271984

Zahir Ahmad MC200400105
Sayyada Aleena Bukhari MC190402109

Software Projects & Research Section,
Department of Computer Sciences,
Virtual University of Pakistan

CERTIFICATE

This is to certify that ZAHIR AHMAD (MC200400105), SAYYADA ALEENA BUKHARI (MC190402109) have worked on and completed their Software Project at Software & Research Projects Section, Department of Computer Sciences, Virtual University of Pakistan in partial fulfillment of the requirement for the degree of BS in Computer Sciences under my guidance and supervision.

In our opinion, it is satisfactory and up to the mark and therefore fulfills the requirements of BS in Computer Sciences.

Supervisor / Internal Examiner

Dr. Saima Munawar
Supervisor,
Software Projects & Research Section,
Department of Computer Sciences
Virtual University of Pakistan

_____
(Signature)

External Examiner/Subject Specialist
<<External Supervisor Name>>

_____
(Signature)

Accepted By:

_____
(For office use)

EXORDIUM

In the name of Allah, the Compassionate, the Merciful.

Praise be to Allah, Lord of Creation,
The Compassionate, the Merciful,
King of Judgment-day!

You alone we worship, and to You alone we pray for help,
Guide us to the straight path

The path of those who You have favored,

Not of those who have incurred Your wrath,
Nor of those who have gone astray.

DEDICATION

Dedicated to my Mother (RIP), my Family and Friends

~Zahir Ahmad

ACKNOWLEDGEMENT

All glory belongs to Allah (S.W.T). the One who created and sustains all visible and invisible worlds. We want to start by thanking Allah Ta'ala for the many benefits We have received that have enabled me to finish this task. Second, we want to extend my profound gratitude to my boss, Dr. Saima Munawar, for her invaluable advice and help. We greatly appreciate all of his help, inspiration, and research-related technical guidance. I offer up prayers for his continued prosperity and well-being. We owe a lot of gratitude to my parents and my family & friends for their encouragement, support, and help as I finished getting my degree.

PREFACE

In chapter 1 we have covered the Project Requirements, Usage Scenarios, Use Cases and Development Methodologies. Chapter 2 contains all designs of the project. Chapter 3 have the development plan, datasets, code, References and Appendix.

TABLE OF CONTENTS

Chapter 1 **Gathering & Analyzing Info**

1.1 Introduction

The heart is a vital organ that delivers blood to every region of our bodies. If it fails to function properly, the brain and other organs will stop operating, and the individual will die within minutes. Changes in lifestyle, work-related stress, and poor eating habits contribute to the rise in the number of cardiac ailments. Cardiovascular disease (CVDs) is a catch-all term for heart and blood vessel problems. CVDs are the leading cause of death globally, 38% deaths in 2019 were caused by CVDs [1]. The most common CVD is coronary artery disease (CAD) which causes 15.6% deaths worldwide in 2015 [2]. Prediction of CAD at early stage is a big challenge for doctors all around the globe. As a result, a reliable, accurate, and practical approach to diagnosis such disorders in time for adequate treatment is required. To automate the examination of big and complicated data, deep learning methods and techniques have been used for such problems. Several deep learning algorithms have recently been used by many researchers to aid the health care industry and experts in the detection of heart-related disorders.

We used deep learning methods on UCI machine learning repository [3] and Mendeley Data [4] datasets to predict the CAD. For structured or labelled data, we used Artificial Neural Network (ANN), and for unstructured or images dataset we used Convolutional Neural Network the training of model and achieved almost 90% accuracy in disease prediction. We used Streamlit Python Library to take inputs and show results.

1.2 purpose

The goal of this study is to develop cardiovascular and chronic respiratory disease prediction system for caregivers and medical experts. The system will be able to provide information about aggravating vital signs related to cardiovascular and chronic respiratory diseases. This early detection will help the medical advisors and caregiver to take necessary early actions.

1.3 scope

Scope of this chapter is to detailed study of requirements gathering, usage Scenarios and adopted methodology.

1.4 definitions, acronyms and abbreviations

DEFINITIONS:

Cardiovascular Disease: A type of disease that affects the heart or blood vessels [5].
Coronary Artery Disease: A disease in which there is a narrowing or blockage of the coronary arteries (blood vessels that carry blood and oxygen to the heart). Coronary artery disease is usually caused by atherosclerosis (a buildup of fatty material and plaque inside the coronary arteries). The disease may cause chest pain, shortness of breath during exercise, and heart attacks. The risk of coronary artery disease is increased by having a family history of coronary artery disease before age 50, older age, smoking tobacco, high blood pressure, high cholesterol, diabetes, lack of exercise, and obesity. Also called CAD and coronary heart disease [6].
Deep Learning: Deep learning (also known as deep structured learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning [7].
Loss Function: Machines learn by means of a loss function. It's a method of evaluating how well specific algorithm models the given data. If predictions deviates too much from actual results, loss function would cough up a very large number. Gradually, with the help of some optimization function, loss function learns to reduce the error in prediction. In this article we will go through several loss functions and their applications in the domain of machine/deep learning [8].
Data Augmentation: Data augmentation is a process of artificially increasing the amount of data by generating new data points from existing data. This includes adding minor alterations to data or using machine learning models to generate new data points in the latent space of original data to amplify the dataset [9].
Optimizers: While training the deep learning model, we need to modify each epoch's weights and minimize the loss function. An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improve the accuracy [10].

ABBREVIATIONS:

IDMPCVD	Implementation of Deep Learning Method for the Prediction of Cardiovascular Disease.
DS	Data Set
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
UC	Use Case

1.5 Project requirements

1.5.1 **Functional Requirements**

i. Define the problem and select the dataset as any type of Cardiovascular Disease
ii. Data Analysis and Preprocessing
iii. Feature Extraction
iv. Prediction
v. Build system
vi. Test System
vii. Tune System
viii. The program should have a knowledge-based system according to select data (Structured or Unstructured data).
ix. The program should have the deep learning algorithm to execute model prediction.
x. The program should evaluate the performance and update knowledge based on the requirement.

1.5.2 **Non-Functional Requirements**

1. **Performance:** Response time, process throughput, and capacity are all examples of performance criteria. They deal with response time, which refers to how long it takes for the system to load, reload, open and refresh the screen. The time it takes to process data owing to functions, calculations, imports, and exports. The system's performance is also affected by query and reporting time. System should be built with all essential and minimum features so that maximum performance with less time could be achieved.
2. **Maintainability:** The monitoring and maintenance of the system should be fundamental and focused in its approach. There should not be an excessive number of jobs running on several deeps, making it difficult to monitor if the jobs are operating smoothly.
3. **Usability:** Specifies how simple it is to understand, manage, and use the system. The term "usability" refers to how user-friendly a system is. This also covers needs for internationalization/localization, such as languages, spellings, and so on. The system should be simple enough for new users to pick up fundamental operations after just one day of use; users should only need two screens to accomplish a task and/or one measurement, and administrators should have direct access to the system.
4. **Reliability:** The level to which the system must work for users is described below. It also refers to the mean time between failures, which is the maximum amount of downtime possible. For instance, two hours every six months, and

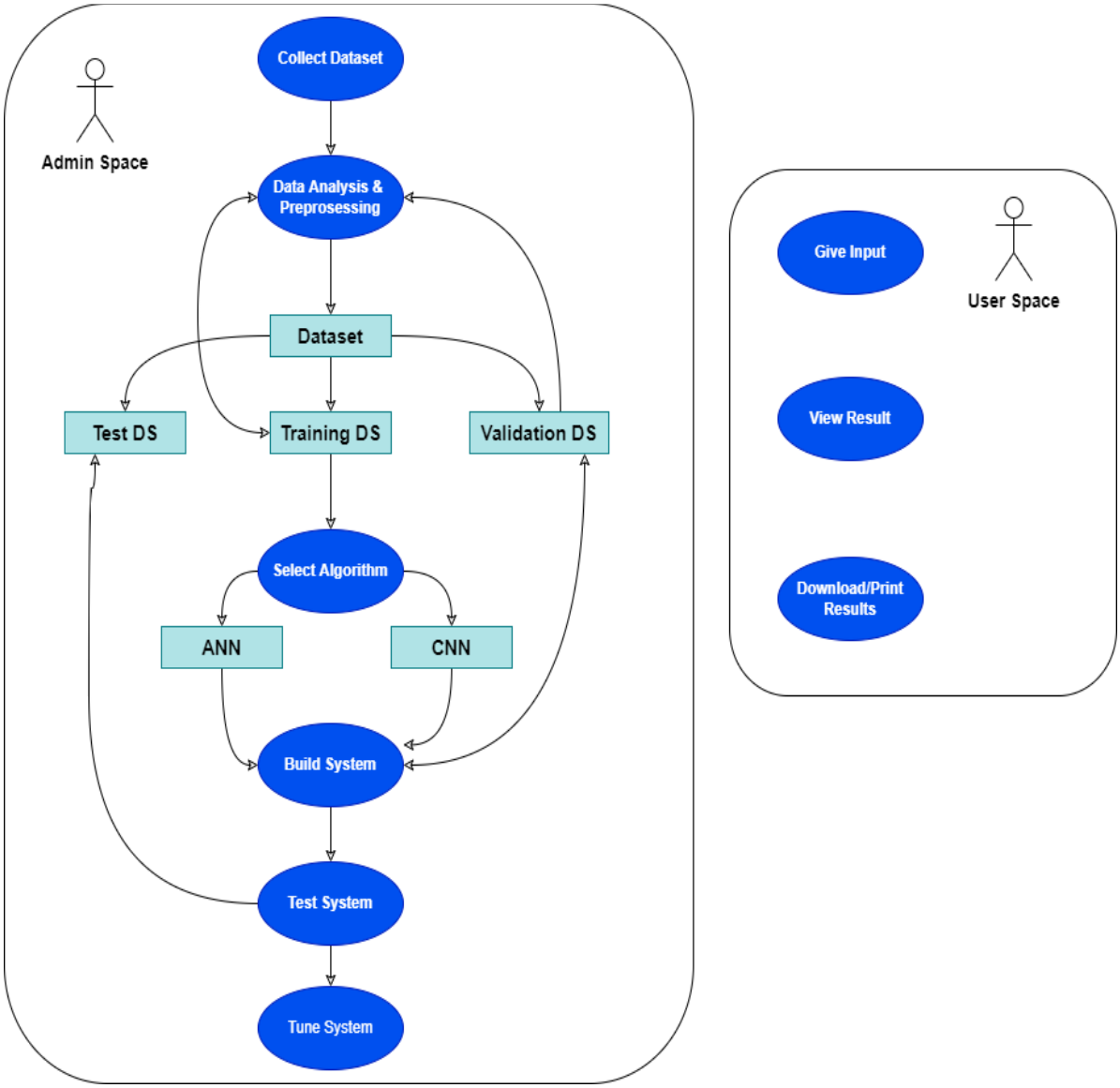
so on. It also addresses the time it takes for the system to recover from a failure; this is known as the mean time to recovery. As a result, reliability standards often refer to availability, downtime, time to repair, accuracy, and so on.

5. **Supportability:** It relates to an application's capacity to be simply updated or modified in order to support common usage or modification scenarios. The software application, for example, will allow users to develop new workflows without having to programme them.
6. **Compatibility:** The ease with which a system can work with shared applications is referred to as compatibility. These shared programmes could also be third-party applications. This also includes the system's compatibility with other platforms. Hardware, software, or both can be used as platforms. Compatibility in the IDMPCVD project primarily pertains to the connection with medical devices.
7. **Scalability:** It refers to a software application's ability to scale the number of users or applications associated with the product. In the case of the IDMPCVD project, the software will be able to deliver more features to users in the future and will be able to support more medical equipment.
8. **Recovery:** The ability of a software system to recover after being damaged is known as recovery. The recovery time is the time it takes for the system to return to its original shape.
9. **Robustness:** Robustness refers to a computer system's ability to cope with faults during execution or an algorithm's ability to continue operating despite anomalies in input, calculations, and so on.

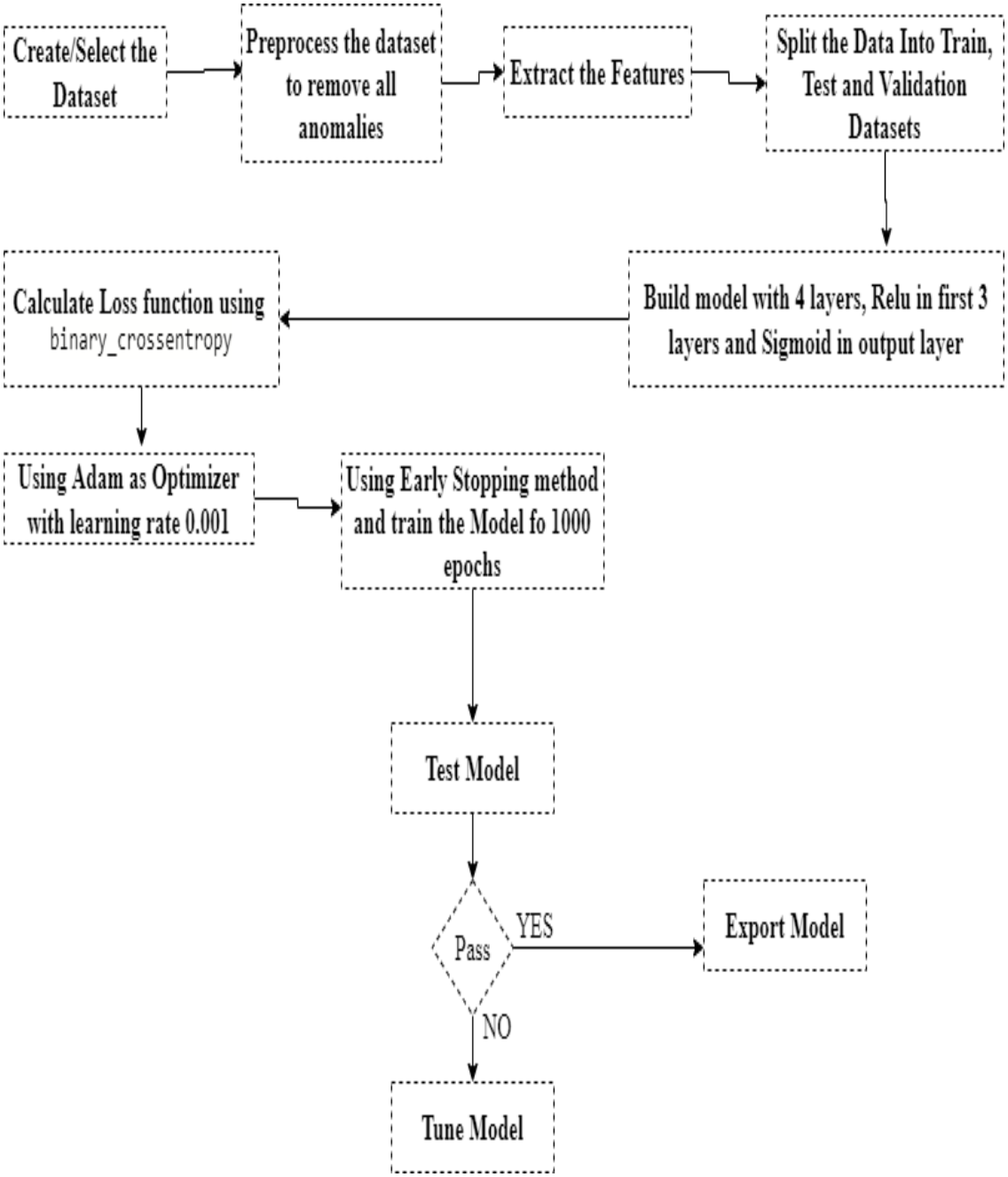
1.6 Use cases and usage scenarios

1.6.1 Use Case Diagrams

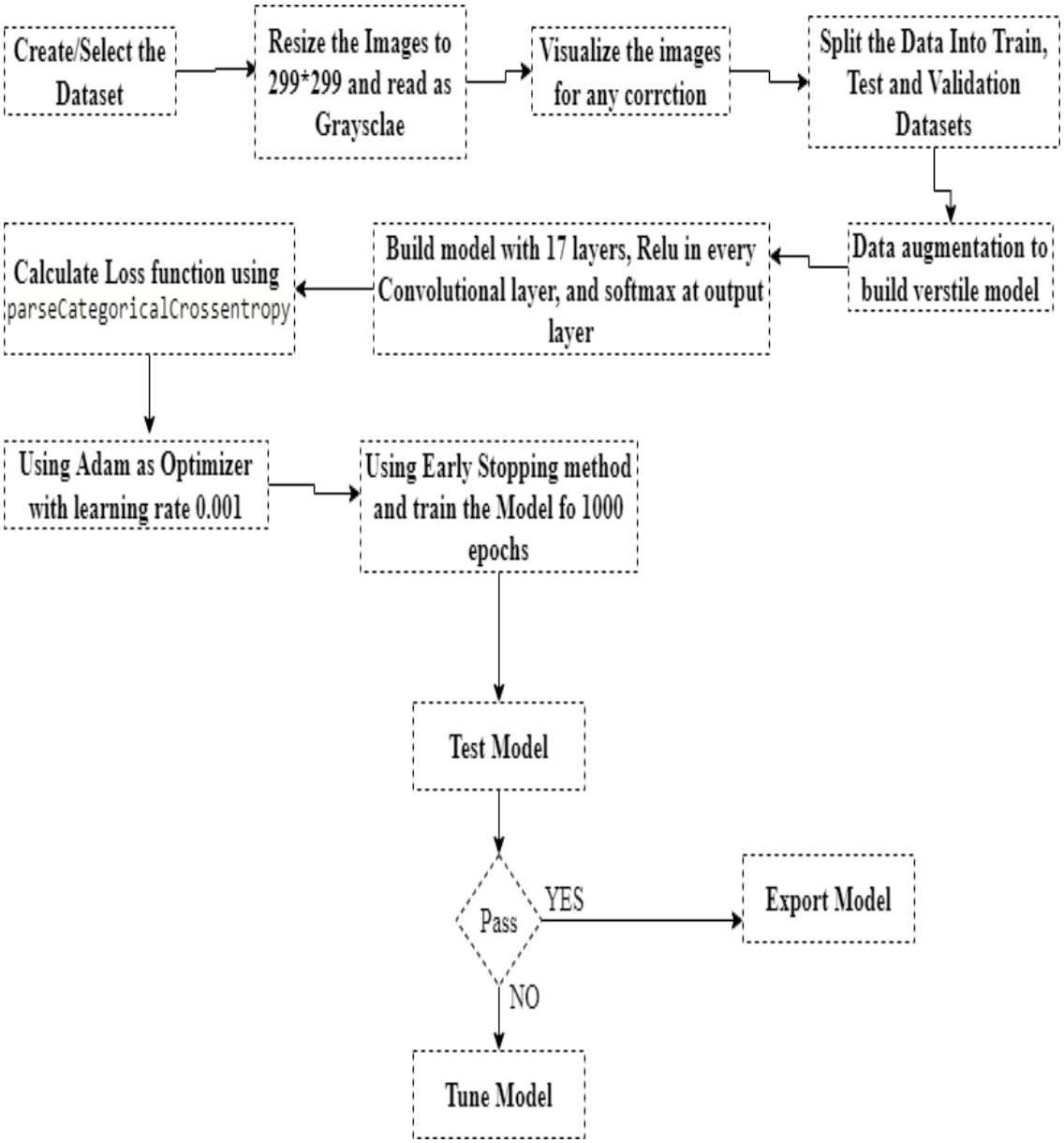
General System Use Case



ANN implementation Use-Case



CNN Implementation Use-Case



1.6.2 **Usage Scenarios**

Usage-Scenario_1

Use case related to Data collection/creation of Admin		
Use case title	Data Collection	
Abbreviated title	Data_colc	
Use case ID	IDMPCVD_UC1	
Actors	Admin	
Description	This use case enables Admin to Collect the required data for particular problem	
Pre-conditions	Admin must define problem area first	
Task sequence		Exceptions
1. Admin collects/creates data according to the requirement of defined problem		
Post condition	Valid Datasets collected	
Author	MC200400105	

Usage-Scenario_2

Use case related to Data analysis/preprocessing of Admin		
Use case title	Data Analysis	
Abbreviated title	Data_ana	
Use case ID	IDMPCVD_UC2	
Actors	Admin	
Description	This use case enables Admin to analyze the data and preprocess it for higher accuracy	
Pre-conditions	Admin must have collected the dataset	
Task sequence		Exceptions
1. Admin will analyze the dataset 2. Remove the anabolize in the dataset 3. Clean the dataset		1. Proper dataset is not collected
Post condition	Dataset is cleansed	
Author	MC200400105	

Usage-Scenario_3

Use case related to Data Selection of Algorithm of Admin	
Use case title	Select Algorithm
Abbreviated title	Select_Algo
Use case ID	IDMPCVD_UC3
Actors	Admin
Description	This use case enables Admin to select the suitable algorithm for the selected dataset
Pre-conditions	Admin must have cleansed the dataset
Task sequence	Exceptions
1. Admin will select the suitable Data learning Algorithm 2. ANN for Tabular data 3. CNN for images data	1.Code/syntax errors
Post condition	Suitable Algorithm Selected
Author	MC200400105

Usage-Scenario_4

Use case related to Model training of Admin	
Use case title	Train Model
Abbreviated title	Train_model
Use case ID	IDMPCVD_UC4
Actors	Admin
Description	This use case enables Admin to train the model
Pre-conditions	Proper algorithm must be selected
Task sequence	Exceptions
1. Admin will train the model.	1. Code/syntax errors
Post condition	Model is trained
Author	MC200400105

Usage-Scenario_5

Use case related to Testing of model of Admin		
Use case title	Test Model	
Abbreviated title	Test_model	
Use case ID	IDMPCVD_UC5	
Actors	Admin	
Description	This use case enables Admin to test the model	
Pre-conditions	Model must be trained	
Task sequence		Exceptions
1. Admin will test the model using Test Dataset		
Post condition	If Testing Pass: Export the model Else: Pass to Tune system phase	
Author	MC200400105	

Usage-Scenario_6

Use case related to Tune the Model of Admin		
Use case title	Tune the model	
Abbreviated title	Tune_model	
Use case ID	IDMPCVD_UC6	
Actors	Admin	
Description	This use case enables Admin to tune the model to achieve higher accuracy	
Pre-conditions	Model must be trained	
Task sequence		Exceptions
1. Admin will change the parameters in algorithm to tune the model to achieve higher accuracy 2. Admin will reclean the dataset to achieve the higher accuracy		
Post condition	If Testing Pass: Export the model Else: Re-Tune the model	
Author	MC200400105	

1.7 Development Methodology

1.7.1 **Chosen Methodology**

Software Development Methodologies

The terms "software development methodology" and "system development methodologies" are interchangeable. This is the system development life cycle, or System Development Process as it is also known. Software development techniques are frameworks for https://www.youtube.com/watch?v=28WlUHQfWlcstructuring, controlling, and planning the system development process.

There are multiple models for such processes, each giving a distinct approach to certain tasks or activities that occur within the process. The following are the names of some of the methodologies:

- Agile software Development
- Waterfall Model
- Spiral Model
- Incremental model
- Extreme Programming
- Crystal Methods

My adopted methodologies are as follows:

Waterfall methodology:

Other technical procedures were used to create the first published model of the software development process. This concept is known as the waterfall model because it cascades from one phase to the next. The linear sequential model is another name for this concept.

The waterfall model is a model that is driven by documentation. As a result, it generates thorough and detailed documentation, making maintenance considerably easier. It, on the other hand, suffers from the fact that customer input is only received after the product has been delivered, therefore any problems in the requirement specifications are not identified until the product has been given to the client.

As a result, there are significant time and expense implications.

Spiral Methodology:

Barry Boehm is the creator of this model. The fundamental goal of this paradigm is to reduce risk, which is always present in software development. For example, important individuals may resign at a critical time, the software

development manufacturer may go bankrupt, and so forth. In its most basic form, the Spiral model is a combination of the Waterfall model and risk analysis.

Each stage in this example is preceded by the identification of alternatives and risk analysis, and then evaluated and planned for the following phase. If the risks cannot be resolved, the project will be cancelled immediately. The Spiral Model's key strength is that it is extremely risk sensitive. It is simple to understand because of the spiral structure of development.

In the Spiral approach, each phase begins with a design goal and concludes with a client review of the process.

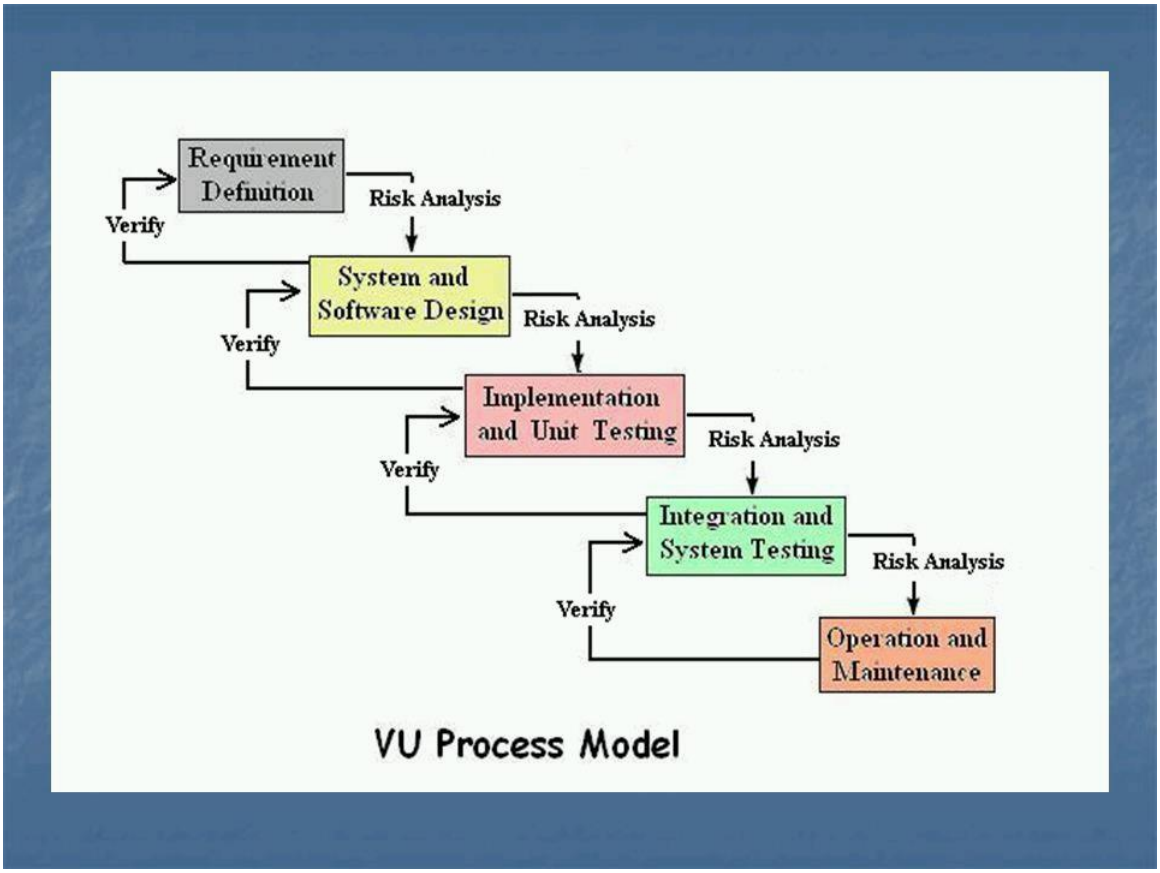


Image-1 VU Process Model

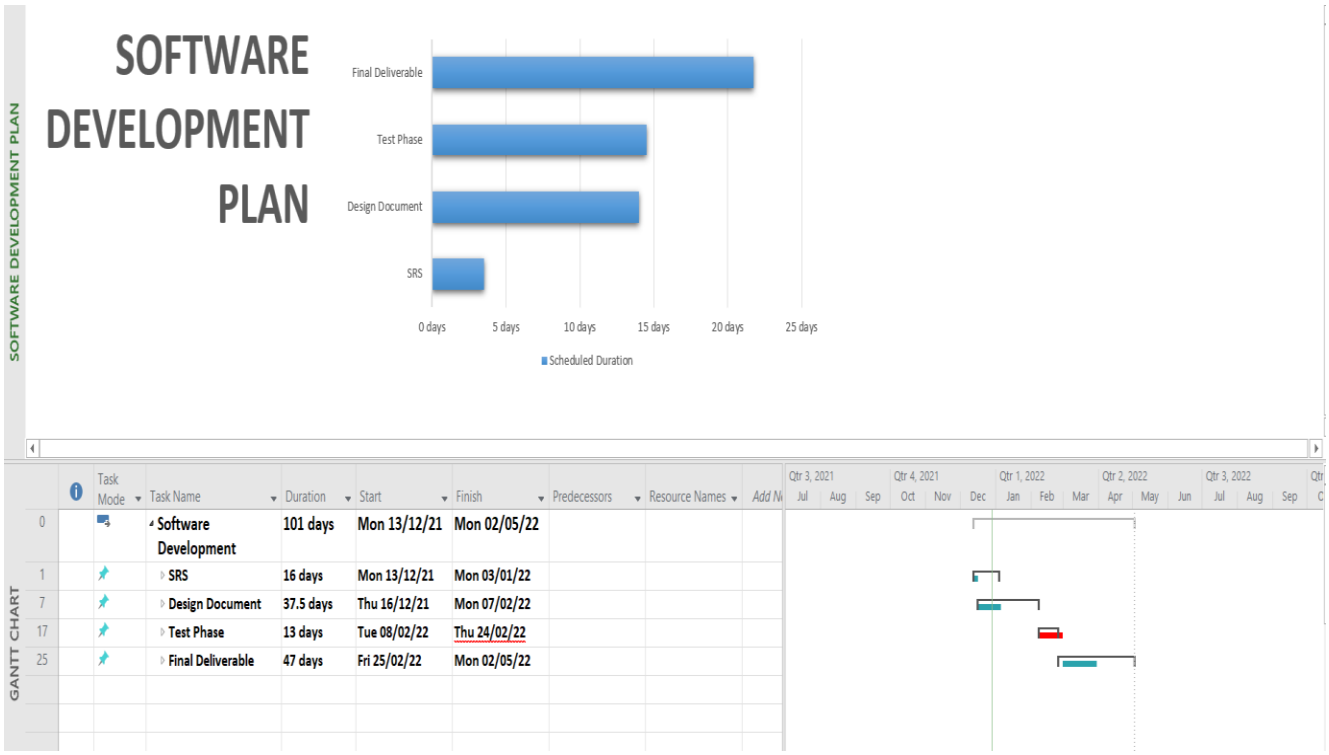
The VU process model, which combines waterfall and spiral models, is seen above. It is divided into five phases, the first of which is learning and evaluating requirements, the second of which is the creation of SRS, the third of which is Test Phase 1, the fourth of which is Test Phase 2, and finally the final deliverable. After a discussion with users, system goals, services, and limitations are developed in the requirement phase.

1.7.2 **Reasons for Chosen Methodology**

- This model is simple to comprehend and apply.
- Each phase has its own set of deliverables.
- Phases are finished and processed one at a time in this methodology.
- Risk avoidance is improved.
- Later on, further functionality can be added.

❖ This is a sequence model.

1.7.3 **Work Plan (Gantt Chart)**



Chapter 2 **Designing the Project**

2.1 Introduction

A design document is a collection of materials and resources that addresses every aspect of product design. A design document is a technique to explain your design decisions to others, particularly your client. The following elements make up a design document:

Entity relationship diagram (ERD): the graphical representation of data for an organization or for business area.

Sequence diagram: it shows how object interact with each other and emphasize time ordering of the messages by showing object interaction arranged in time sequence.

Architecture design diagram: it provides a tiered architecture of the system.

Class diagram: it describes the system by showing their classes, their attributes and relationships among the objects.

Database design: it is a process of creating a detailed data model of the database.

Interface Design: It is the proposed interface of the system

Test cases: These are the test case scenarios of use case scenarios.

You'll get stuck in a loop of fractious ambiguity if you don't have this document, with clients disputing what they told you or what you told them, angrily sending cut-and-pastes of previous communications, interpreting and arguing until the client demands that you make changes to bring the application into compliance with "what they actually asked for," and expects you to do so without pay.

When arguments emerge, you can refer to the specification that the client agreed to and signed off on, pointing out that you have followed it exactly. Instead of arguing, you'll make changes to the document and clarify things. If anything, the client will express regret for allowing the inaccuracy to occur in the first place.

2.2 Purpose

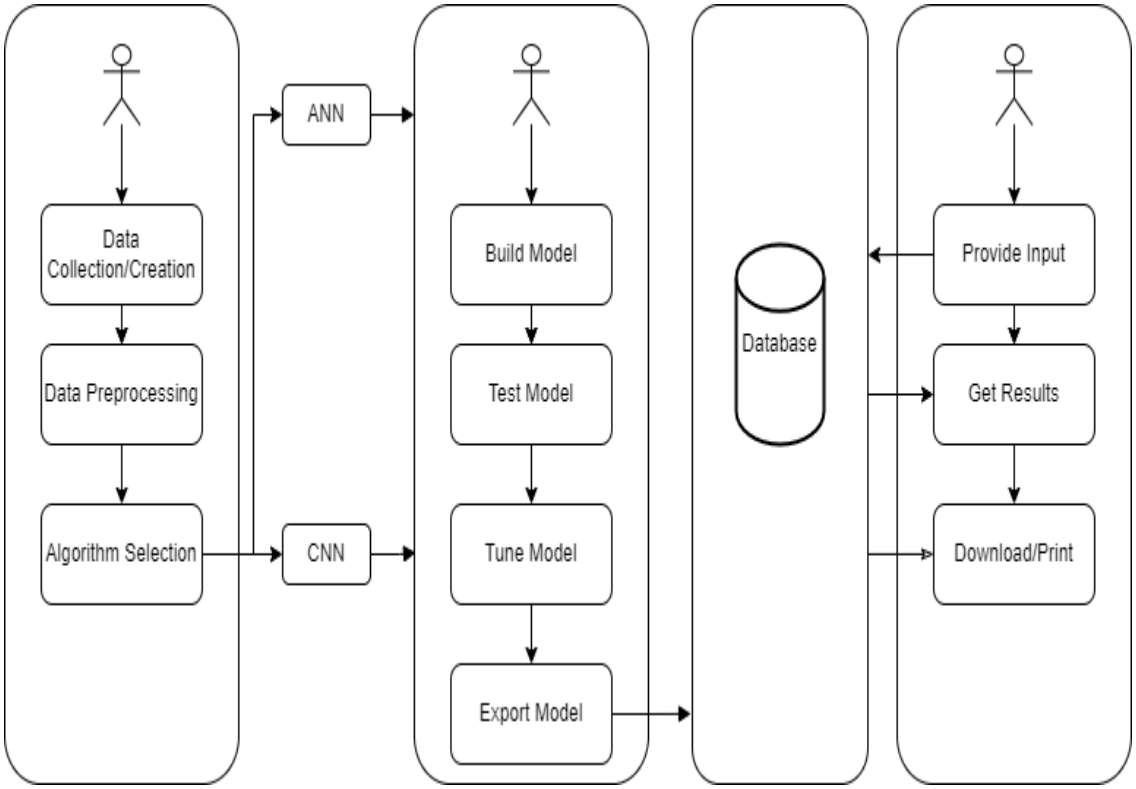
The goal of this chapter is to cover the detailed and depth knowledge of project design phase.

2.3 Scope

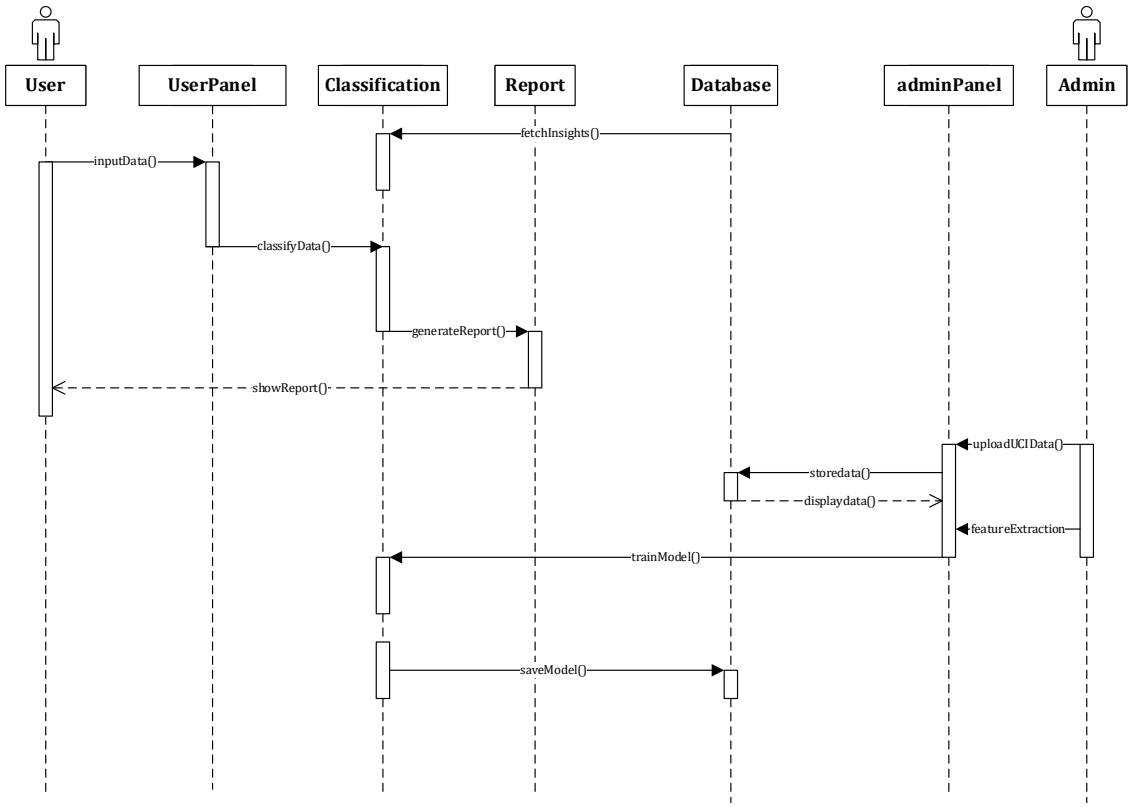
This chapter will cover the design phase of project.

2.4 Definitions, Acronyms and Abbreviations

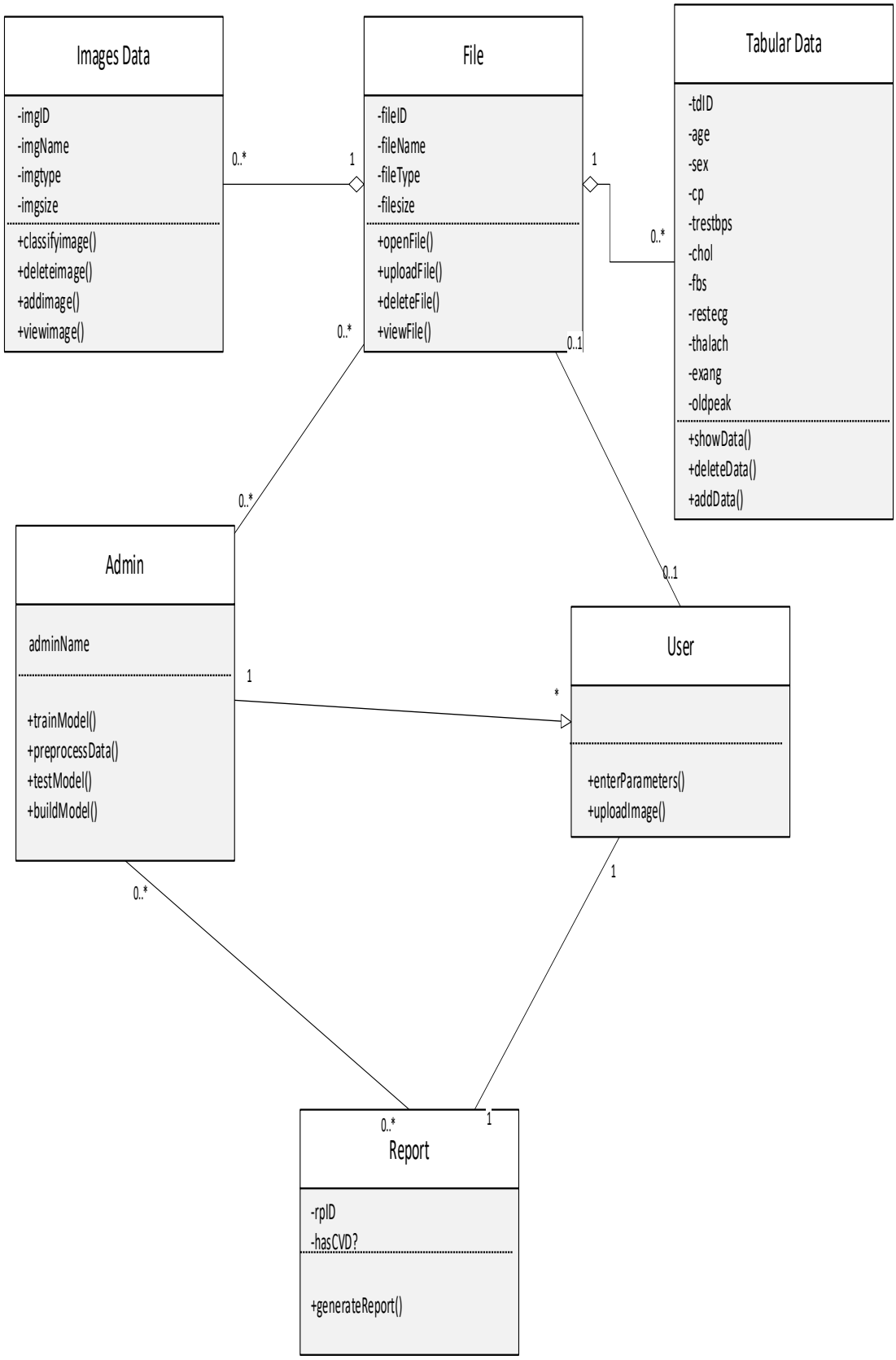
2.5 Architectural Representation (Architecture Diagram)



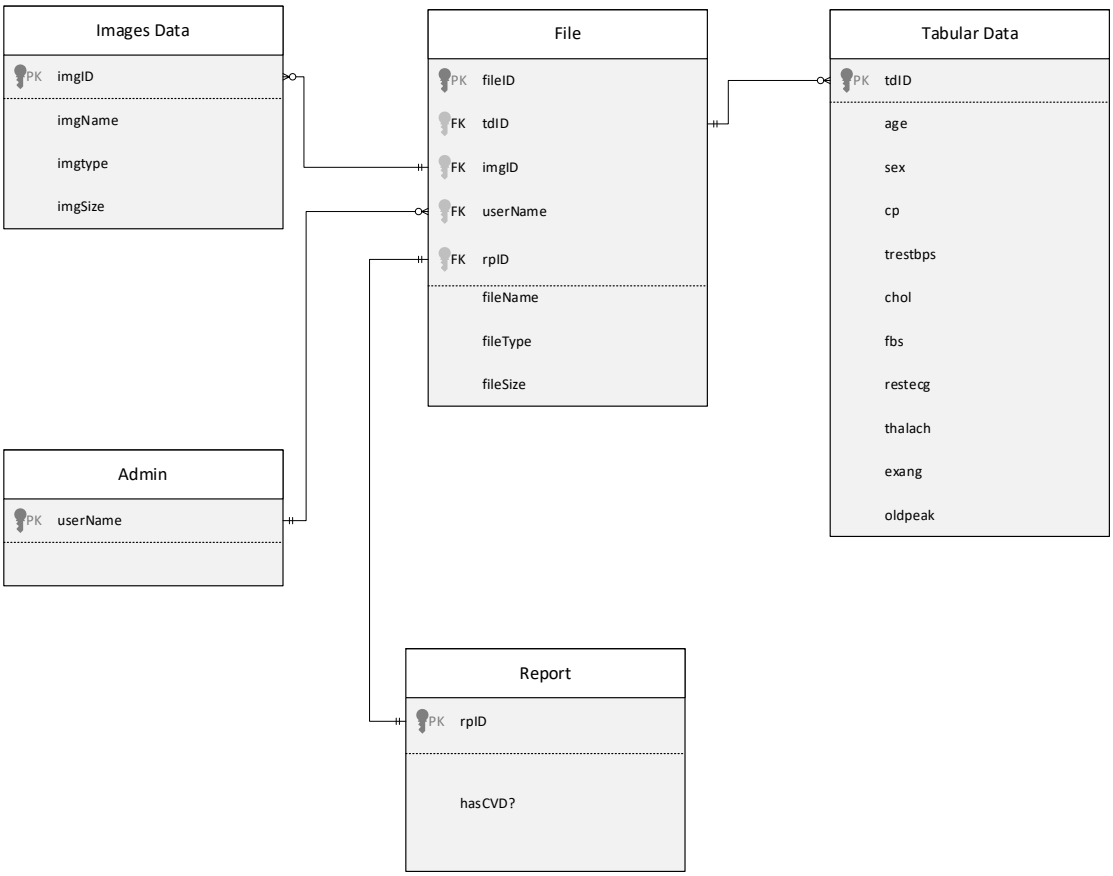
2.6 Dynamic Model: Sequence Diagrams



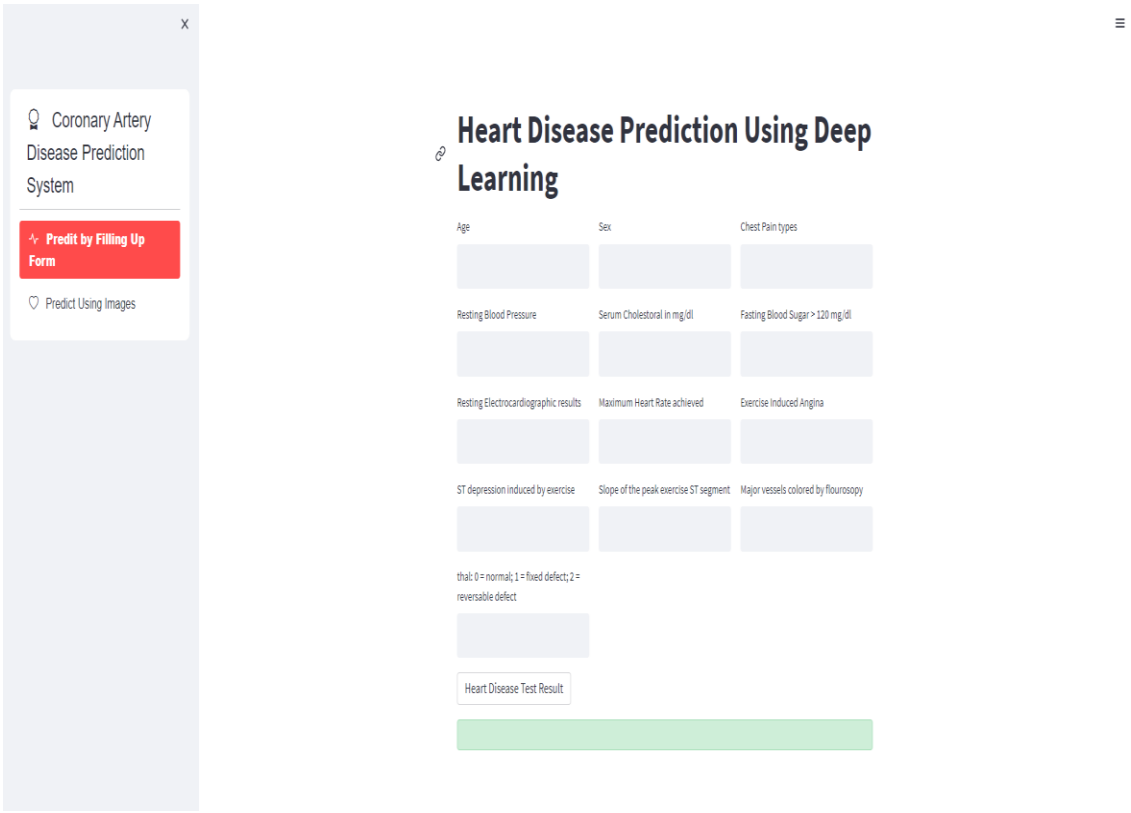
2.7 Object Model/Logical Model: Class Diagram

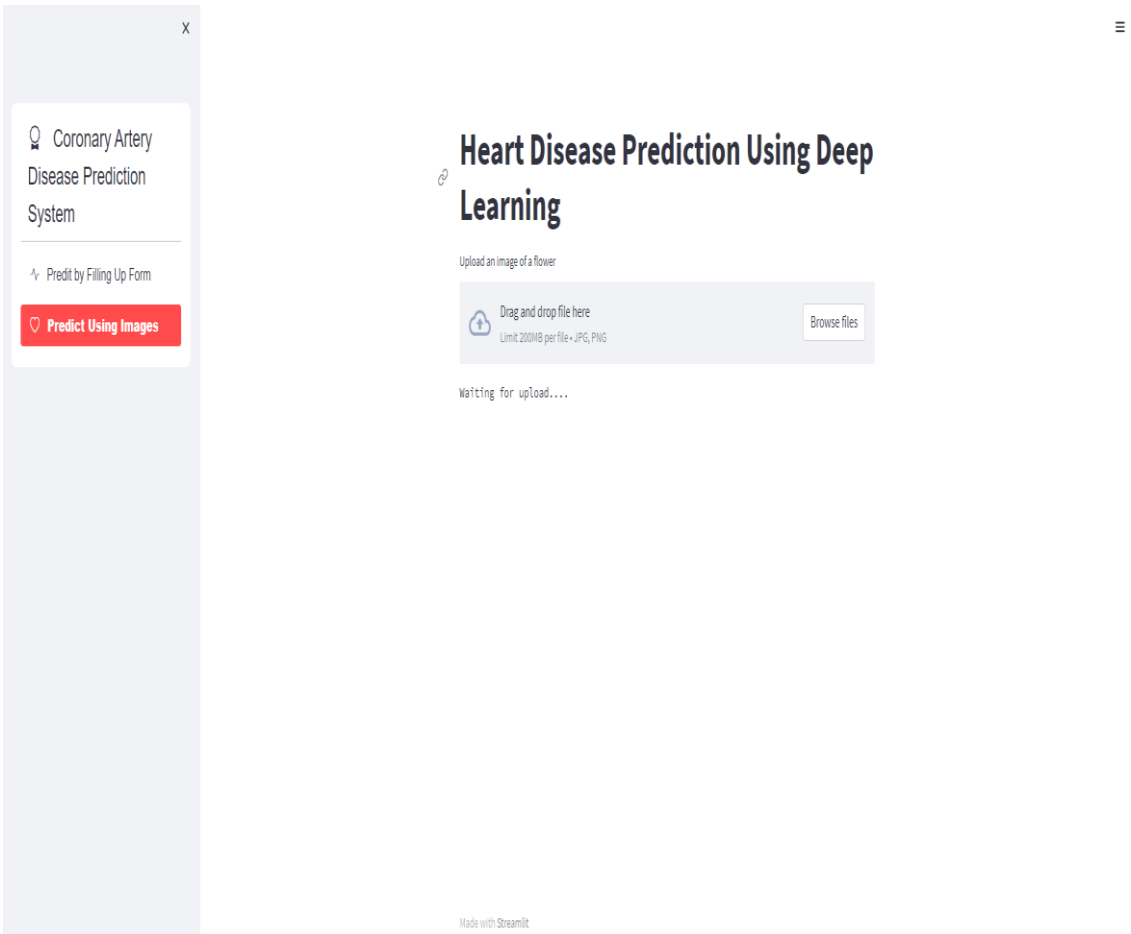


2.8 Database Model (Database Diagram)



2.9 Graphical User Interface

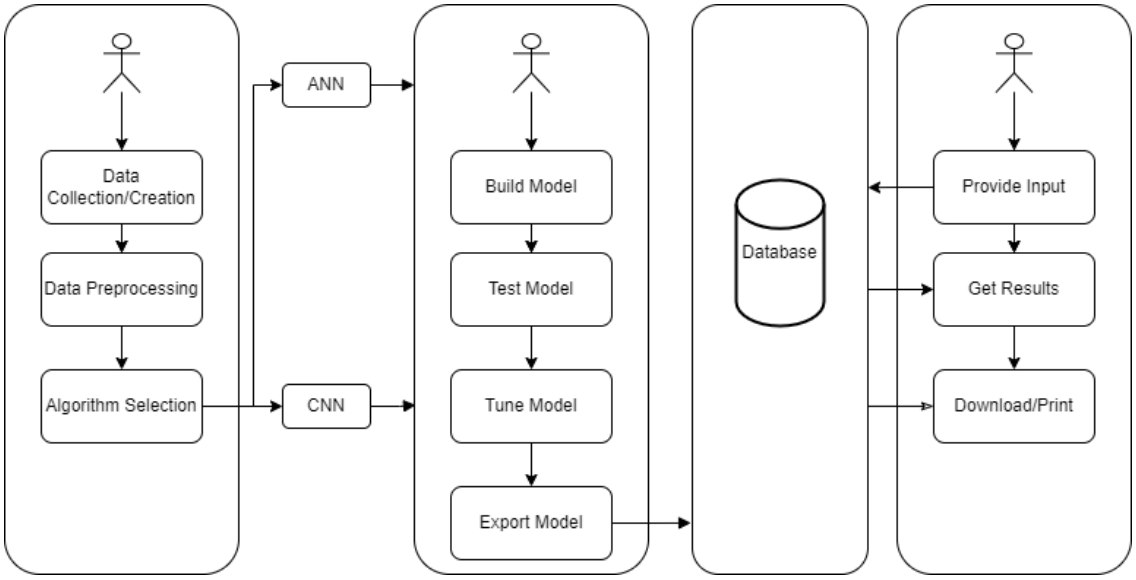




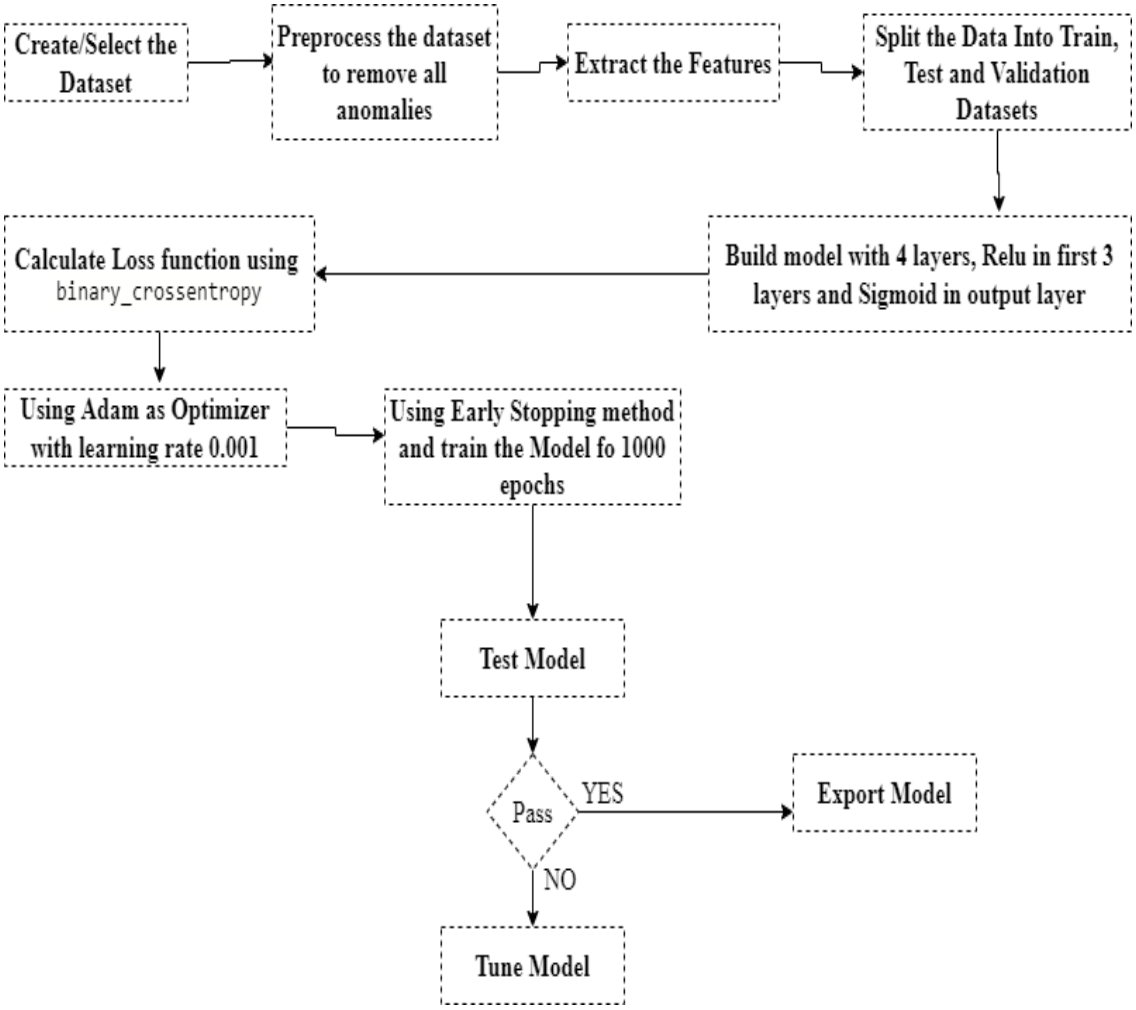
Chapter 3 **Development**

3.1 Development Plan (Architecture Diagram)

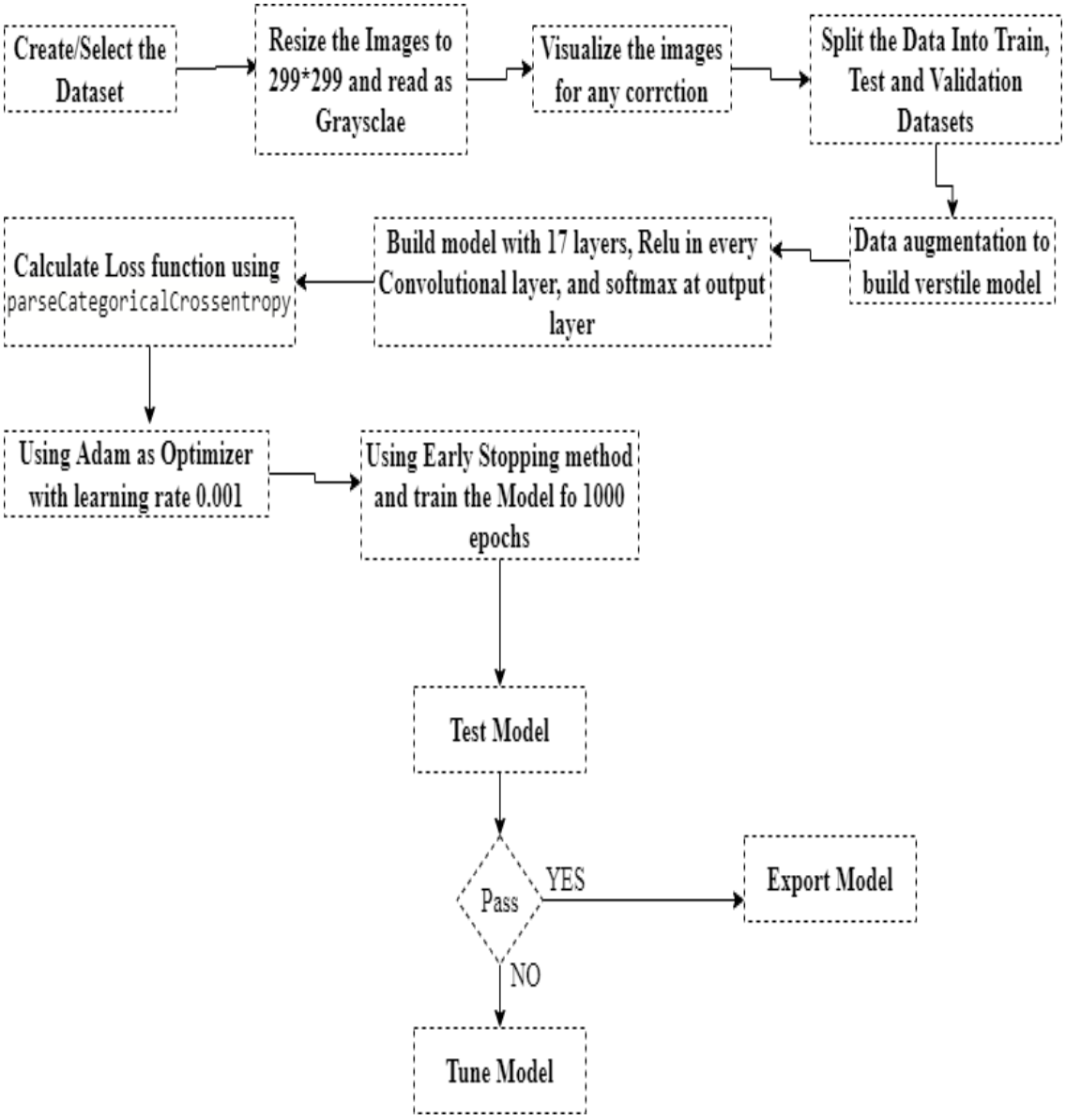
General Architecture Diagram



ANN Implementation diagram



CNN Implementation Diagram



3.2 Tabular Dataset (Link: https://www.kaggle.com/datasets/agsam23/coronary-artery-disease?select=data.csv)

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1.0
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1.0
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1.0
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1.0
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
303	58	0	1	140	130	1	0	115	0	1.3	1	2	3	0.5
304	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0.3
305	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0.6
306	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0.4
307	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0.6

308 rows × 14 columns

3.3 Code of Tabular Dataset:

```
# -*- coding: utf-8 -*-
"""IDMPCD.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1Vq6T4Xjaw8f37Cil4BV-0N59hzpfuD2p

## **Test Phase of IDMPCD**

Tabular Data
"""

#!pip install tensorflow-gpu

from google.colab import drive
drive.mount('/content/drive')

"""# **Importing Libraries**"""

## importing Tensorflow
import tensorflow as tf
print(tf.__version__)

# For importing Data I am using Pandas Python Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

"""# **Importing Dataset**
I have uploaded dataset on my Gdrive and importing it using Pandas.
"""

# Importing dataset from Gdrive
cad_data = pd.read_csv('/content/heart1.xlsx - heart.csv')
# Displaying Imported Dataset
cad_data

"""# **Data Preprocessing**"""

# Displaying Some information about the data
cad_data.info()

# Displaying missing values in the data
cad_data.isnull().sum()

"""We can see that, there are no missing values in the dataset, So we don't
need apply any missing value technique."""

# Printing some statistical insights of data
```

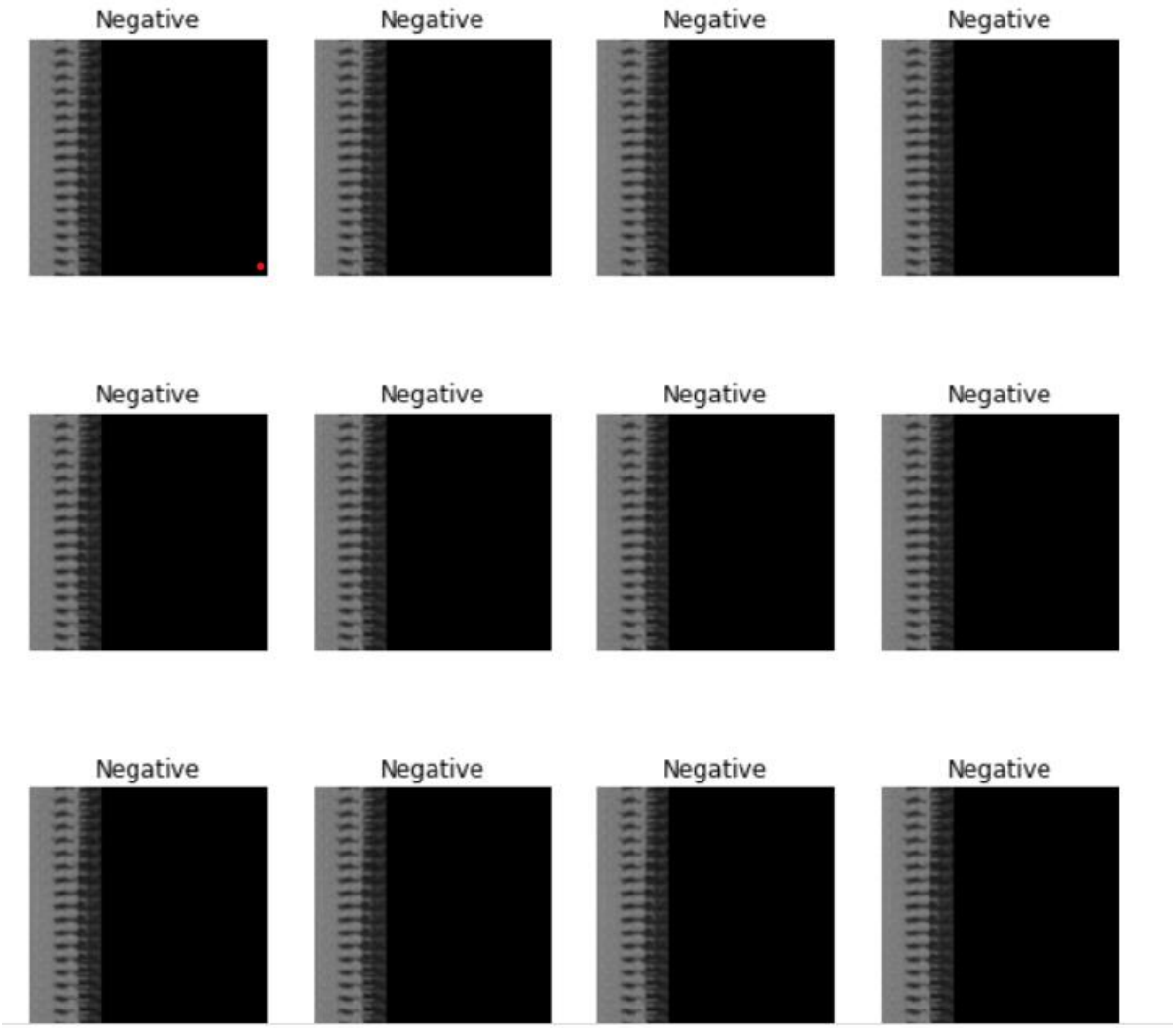
```
cad_data.describe()

# Checking Skewness of data columns, Acceptable values of skewness fall
between - 3 and + 3.
cad_data.skew()

"""We can see all columns of our dataset are normaly skewed."""

# Checking Kurtosis of dataset, kurtosis is appropriate from a range of - 10
to + 10
cad_data.kurtosis()

# Checking balance of target variable
cad_data['target'].value_counts()

"""We can see that from our target variable that our dataset is almost evenly
distributed
*   1 Represents: CAD present
*   0 Represents: CAD not present
"""

cad_data

"""# **Feature Extraction**"""

# Splitting data into features and target
# X contain featrues and Y contain the target variable
X = cad_data.drop(columns='target', axis=1) ## axis = 1 means we only want to
drop column, to drop row we'll write axis = 0
Y = cad_data['target']

# Printing Selected Features
print(X)

#Printing Target Variable.
print(Y)

"""## Splitting data into Training and Test set

"""

from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=
0)

#feature Scaling

from sklearn.preprocessing import StandardScaler
sc =StandardScaler()
X_train = sc.fit_transform(X_train) ## Why to fit_transform in training
dataset only? Ans: To avoid data leakage, reference to krish naik video
```

```
X_test = sc.transform(X_test)

X_train

X_test

X_train.shape

X_test.shape

"""## Creating the ANN"""

## Sequential means a huge block, inside contains Neural network, in which we
can do complete forward
## propagation and backward propagation
from tensorflow.keras.models import Sequential
## Dense layer is used to create the neurons e.g input layer, hidden layer,
output layer
from tensorflow.keras.layers import Dense
## Activation functio is used inside the hidden layer to activate the neurons
from tensorflow.keras.layers import LeakyReLU,PReLU,ELU,ReLU
## When Training accuracy is high and test accuracy score is low, then we
dropout some neurons in hidden layers similar to Regularization in ML
from tensorflow.keras.layers import Dropout

## Initializing the ANN
classifier = Sequential()

## Adding the input layer
classifier.add(Dense(units=13, activation='relu'))

## Adding the first hidden layer
classifier.add(Dense(units=7, activation='relu'))

## Adding the second hidden layer
classifier.add(Dense(units=6, activation='relu'))

## Adding the output layer
classifier.add(Dense(1,activation='sigmoid'))

## Compiling the ANN
classifier.compile(optimizer='Adam', loss='binary_crossentropy',
metrics=['accuracy'])

"""## Training Model"""

## Early Stopping: when the accuracy of the model is not increasing, training
of the model will automatically stop.
import tensorflow as tf

earlyStop = tf.keras.callbacks.EarlyStopping(
              monitor="val_loss",
```

```
              min_delta=0.0001, ## Minimum change in the monitored quantity to
qualify as an improvement
              patience=20, ## Number of epochs with no improvement after which
training will be stopped.
              verbose=1, ## Verbosity mode, 0 or 1. Mode 0 is silent, and mode
1 displays messages when the callback takes an action.
              mode="auto",
              baseline=None,
              restore_best_weights=False)

model_history = classifier.fit(X_train, y_train, validation_split = 0.33,
batch_size=10, epochs = 1000, callbacks= earlyStop)

## To see parameters on which we are focused on during model training.
model_history.history.keys()

## Summarized history of accuracy
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc = 'upper left')
plt.show()

## Summarized history of loss
plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc = 'upper left')
plt.show()

## Making Prediction and evaluating the model

# Predicting the test set results
yPred = classifier.predict(X_test)
yPred = (yPred >= 0.5)

## Calculating the accuracy

from sklearn.metrics import accuracy_score
score = accuracy_score(yPred, y_test)

score

"""## Saving the Model"""



classifier.save('tb_mdl.h5')
```

```
ld_mdl = tf.keras.models.load_model('tb_mdl.h5')
```

3.4 Images DATASET (Link: https://data.mendeley.com/datasets/fk6rys63h9/1)



3.5 Code for Images Dataset:

```
# -*- coding: utf-8 -*-
"""IDMCAD (images).ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/14d-id2JMOXOf1Qi8lJFpVyY323Rgvr5X
"""

## Importing Some Useful Initial Libraries
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')

# Declaraing Some CONSTANTS for future use
imageSize = 299 # Defualt images size (width*height)
batchSize = 32 # To import the images in the groub of batches
CHANNEL = 3 # Defuatl images are in RGB (Three-Channel)

# Importing the Training Data set, placed in GDRIVE.
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "/content/drive/MyDrive/CAD_images_dataset",
    shuffle=True, # to shuffle the places of images every time, when dataset
loads
    image_size = (imageSize,imageSize),
    batch_size = batchSize
)

# To view the Classes in the datasets
class_names = dataset.class_names
class_names

# To view the number of images in a Single Batch
len(dataset)

# To view the Images Size and channel: (width, heigh, channel: 1 for
grayscale, 3 for RGB)
for image_batch, label_batch in dataset.take(1):
  print(image_batch[0].shape)

plt.figure(figsize=(10,10)) # Defining the Figure size in which each image
will be shown
for image_batch, label_batch in dataset.take(1):
  for i in range(12): # Definin the number of images that we want to show
    plt.subplot(3,4, i+1) # creating the subplot (matrix) so that each image
could be shown as a one elemet of matrix
```

```
    plt.imshow(image_batch[0].numpy().astype("uint8"))
    plt.title(class_names[label_batch[0]])
    plt.axis("off") # hides the axis number

# Splitting dataset into training, testing and validation
# 80% training
# 10 Testing
# 10 Validation

# To check how many numbers of batches are equal to 80% of dataset
train_size = 0.8
len(dataset)*train_size

# Declaring Training Dataset
train_ds = dataset.take(150)
len(train_ds)

# Declaring Test Dataset
test_ds = dataset.skip(150) # Skipping first 116 batches from the dataset and
saving remaining as test dataset
len(test_ds)

# Declaring validation data set
val_ds = test_ds.skip(19)
len(val_ds)

# Optimizing datasets for training performance,
# used Cache to prevent the repetative uploading of same image,
# used prefetch to enhance the training speed, by using prefetch,
# CPU first read the dataset batch--> pas it to GPU for training-->and CPU
will starts reading next batch, so on
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

# Creating a constant variable
# Resizing the all images to the same scale
# Rescalling the all images to binary (between 0 and 1)
resize_and_rescale = tf.keras.Sequential([
                     layers.experimental.preprocessing.Resizing(imageSize,
imageSize),
                     layers.experimental.preprocessing.Rescaling(1.0/255)
])

# Data augmentation is creating different images from the same image
# by flipping, zooming, rotating etc
# To achieve higher accuracy and verstile model that could predict the zoomed,
rotated images
# To increase the number of images to that trainig model could be feed well
enough for training
data_augmentation = tf.keras.Sequential([
```

```
                    layers.experimental.preprocessing.RandomFlip("horizontal_
and_vertical"),
                    layers.experimental.preprocessing.RandomRotation(0.2)
])

#Creating Model
input_shape = (CHANNEL, imageSize, imageSize, CHANNEL)
n_classes = 3
model = models.Sequential([
                        #Image Resize and Rescale layer
                        resize_and_rescale,

                        #Data Augmentation Layer
                        data_augmentation,

                        #Step - 1 : Convolutional
                        layers.Conv2D(32, (3,3), activation = 'relu',
input_shape = input_shape),

                        #Step - 2 : Pooling
                        layers.MaxPool2D((2,2)),

                        #Adding a second convulutional layer
                        layers.Conv2D(64, kernel_size=(3,3), activation =
'relu'),
                        layers.MaxPool2D((2,2)),

                        #Adding a third convolutional layer
                        layers.Conv2D(128, kernel_size=(3,3), activation =
'relu'),
                        layers.MaxPool2D((2,2)),

                        #Adding a fourth convolutional layer
                        layers.Conv2D(128, (3,3), activation = 'relu'),
                        # 10th max pooling layer
                        layers.MaxPool2D((2,2)),

                        #Step - 3 : Flattening
                        layers.Flatten(),

                        # Step - 4 : Full Connection
                        layers.Dense(64, activation ='relu'),
                        layers.Dense(n_classes, activation = 'softmax')
])

model.build(input_shape = input_shape)

# To check the summary of the built model

model.summary()

# Compiling the model
```

```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

## Early Stopping: when the accuracy of the model is not increasing, training
of the model will automatically stop.
import tensorflow as tf
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.0001, ## Minimum change in the monitored quantity to qualify
as an improvement
    patience=10, ## Number of epochs with no improvement after which training
will be stopped.
    verbose=1, ## Verbosity mode, 0 or 1. Mode 0 is silent, and mode 1
displays messages when the callback takes an action.
    mode="auto",
    baseline=None,
    restore_best_weights=False,
)

from gc import callbacks
# Training the model

history = model.fit(
    train_ds,
    epochs = 1000,
    batch_size = batchSize,
    verbose = 1,
    validation_data = val_ds,
    callbacks= early_stop
)

# Evaluating the model
scores = model.evaluate(test_ds)

scores

"""## Saving the Model"""

model.save('img_mdl.h5')
```

REFERENCES

1. Schreurs, J., Sacchetto, C., Colpaert, R.M., Vitiello, L., Rampazzo, A. and Calore, M., 2021. Recent Advances in CRISPR/Cas9-Based Genome Editing Tools for Cardiac Diseases. *International Journal of Molecular Sciences*, *22*(20), p.10985.

2. Akella, A. and Akella, S., 2021. Machine learning algorithms for predicting coronary artery disease: efforts toward an open source solution. Future science OA, 7(6), p.FSO698.

3. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

4. Demirer, Mutlu; Gupta, Vikash; Bigelow, Matthew; Erdal, Barbaros; Prevedello, Luciano; White, Richard (2019), "Image dataset for a CNN algorithm development to detect coronary atherosclerosis in coronary CT angiography", Mendeley Data, V1, doi: 10.17632/fk6rys63h9.1

5. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/cardiovascular-disease

6. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/coronary-artery-disease

7. https://en.wikipedia.org/wiki/Deep_learning

8. https://towardsdatascience.com/common-loss-functions-in-machine-learning46af0ffc4d23#:~:text=It's%20a%20method%20of%20evaluating,reduce%20the%20error%20in%20prediction.

9. https://www.v7labs.com/blog/data-augmentation-guide#:~:text=Data%20augmentation%20is%20a%20process,data%20to%20amplify%20the%20dataset.

10. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#:~:text=While%20training%20the%20deep%20learning,loss%20and%20improve%20the%20accuracy.

11.