Software Design Document for Automated Detection of White Blood Cells Cancer Diseases

Rowan Omar, Hend Mohamed, Nermeen El saeed, Ali Essam

June 22, 2018

1 Introduction

1.1 Purpose

The main purpose of this software design document to describe the architecture and system design of our system Cancer chaser. Our system is a desktop application that mainly detects and classifies various white blood cells cancer diseases. We also provide a fulfilled description about each single stage input, output and algorithms used in this stage, along with a full illustration for each stage requirements and development process. It illustrates the system components and how they interacts with each other.

1.2 Scope

This document targets the hematologist experts that would use the Cancer Chaser Application to help them discriminating between different types of white blood cells cancer diseases (Myeloma, Leukemia ALL and AML) which will save much more time rather than manual inspection. Our proposed project is an automated system to diagnose such diseases. It is a recognition system applied on acquired microscopic images then performs pre-processing, segmentation, feature extraction and classification. The system has the ability of learning from misclassified tests to enhance the future accuracy of the system.

1.3 Overview

This SDD document includes 8 main sections. The first section is an introduction to our system including our scope and purpose. The second section is the system overview illustrating our system workflow. The third section includes the architecture design of the system, activity diagram, sequence diagram, state diagram and class diagram. The fourth section illustrates the database design in details. The fifth section illustrates our component design including the used algorithms and techniques. The sixth section illustrates the human interface design and describes how the user will interact with our system. The seventh

section is the requirement matrix that shows which components satisfy each of the functional requirements. The rest of the sections are appendices and references.

1.4 Definitions and Acronyms

Term	Definition
WBC	White blood cells are the cells of the immune system that are involved in protecting the body against both infectious disease and foreign invaders.
Hematologist	A hematologist is a specialist in hematology, the science or study of blood, blood-forming organs and blood diseases.
UI	User interface is the space where interactions between humans and machines occur.
GLCM	A statistical method of examining texture that considers the spatial relationship of pixels is the gray-level co-occurrence matrix.
YCBCR	is a family of color spaces used as a part of the color image pipeline in video and digital photography systems. Y is the luminance component and CB and CR are the blue-difference and red-difference chroma components.
AML	Acute myeloid leukemia, is a cancer of the myeloid line of blood cells, characterized by the rapid growth of abnormal cells that build up in the bone marrow and blood and interfere with normal blood cells.
ALL	Acute lymphoblastic leukemia is a type of cancer in which the bone marrow makes too many immature lymphocytes (a type of white blood cell).
Myeloma	is a cancer arising from plasma cells, a type of white blood cell which is made in the bone marrow.
ROI	Region of interest.
MVC	Model-View- Controller.
SGD	Stochastic Gradient Descent.
SVM	Support Vector Machine.
CNN	Convolutional Neural Network.
RF	Random Forest.
NB	Nave Bayes.

2 System Overview

In Cancer Chaser application we will implement a non-existing system with a high accuracy, speed differentiation and detection of various blood cancer diseases. Cancer Chaser goes through three essential stages. Firstly, uploading microscopic image for blood. Secondly, testing the image after applying different algorithms. Finally, classifying whether the disease is (AML, ALL, Myeloma). The application workflow is illustrated below in figure (1). Starting with pre-processing which will be performed on the uploaded image. Then, segmentation that converts images to YCBCR color space and constructs Gaussian distribution of CB and CR values. After that is feature extraction, mainly extracting 35 features of 4 types of features Morphologic, statistical, texture and

size ratio between nucleus and the cytoplasm. Final step is classification using different algorithms to train and test the dataset. The system stores medical records to save patients medical history. It also has the ability of learning from misclassified tests to enhance the future accuracy of the system.

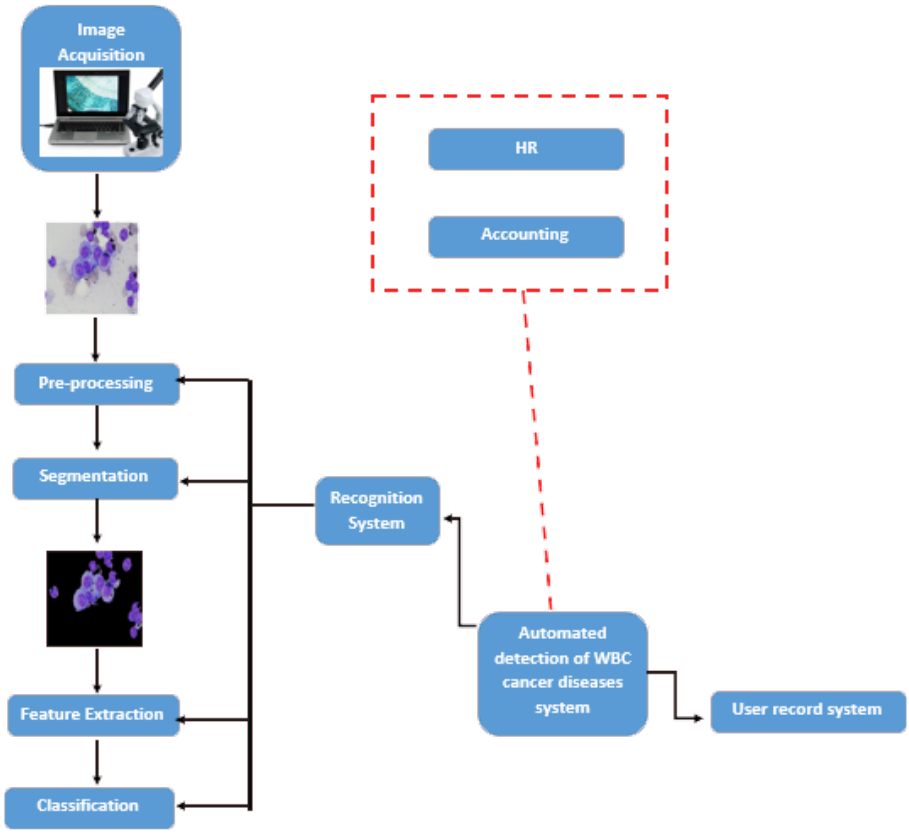


Figure 1: System Overview

3 System Architecture

3.1 Architectural Design

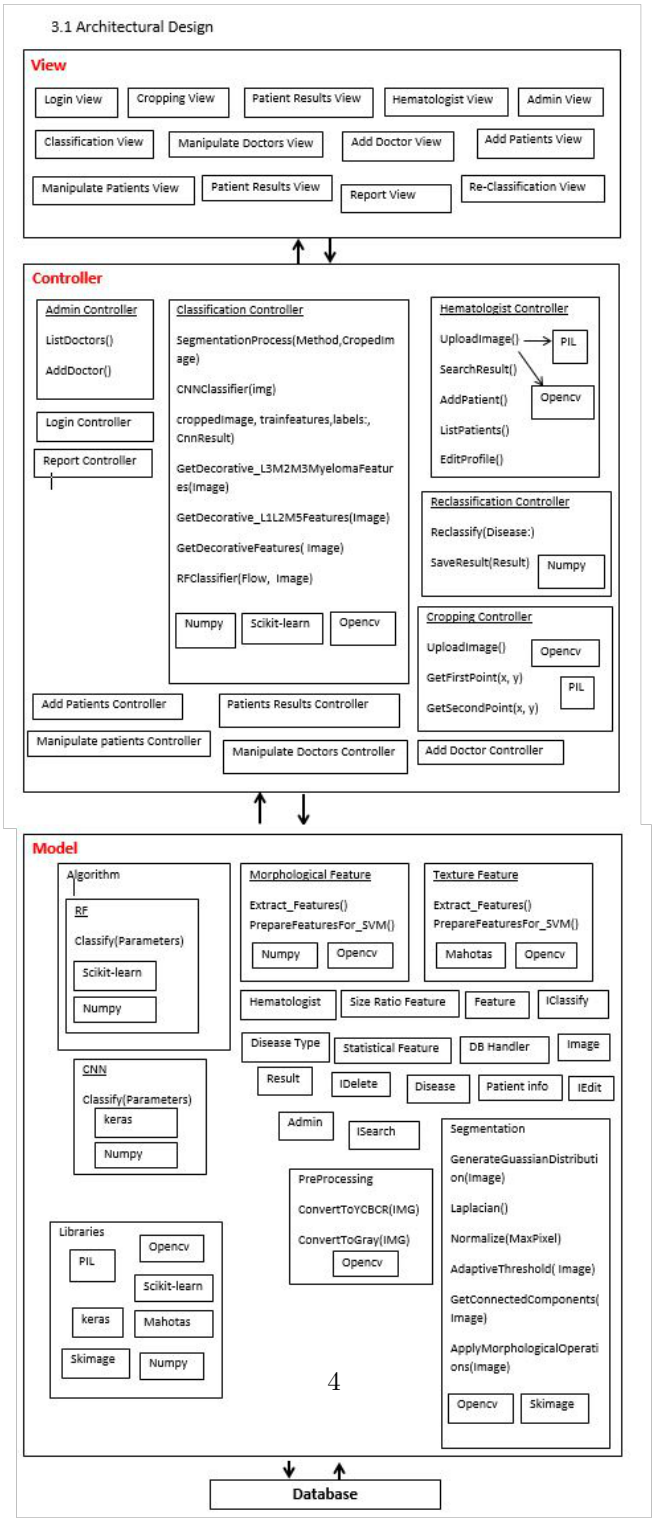


Figure 2: Architectural Design

3.1.1 View

It is responsible for the presentation of data and representing the User Interface (UI). We have two different interfaces one is responsible for Admin operations and the other one is responsible for representing the core part including cropping, diagnosis and classification.

3.1.2 Controller

It is responsible for binding the view and model. The interactions and requests made within the view are taken and sent to the database to fetch data with the use of models then it forward data to the view again to be shown. Some of the controllers we are having; Admin Controller that is responsible for handling interactions made within Admin Views, User Controller that is responsible for handling interactions made by the hematologist, Cropping, Diagnosis and Classification Controllers are responsible for core functionality and handling its interactions and finally the Notification Controller that is responsible for sending notifications to the doctor when the results are shown.

3.1.3 Model

Core
Admin, Hematologist and Patient are models handling information and main functions of users involved in the system.
Disease and Disease type are models handling information of the diseases we are considering and their sub-types.
Statistical, Morphological, Texture and Size Ratio Features are models handling information about each feature type that we extract from the test image.
DB Handler is the model responsible for inserting and fetching data from the database.

Algorithm
Random Forest: It starts with a decision tree as a machine learning technique. It models the data and draws multiple curves and then chooses the strongest curve which contains the greater number of curves assembled together. The data starts to move through different sub branches of the tree until it reaches a decision.

Libraries
Mahotas: This is the library containing on of our features which is texture.
Scikit-learn: This is the library containing the classifier we are using which is RF.
PIL: This library is for managing the images in the UI.
Opencv: This is our core library which handles the images, their color spaces and many operations on the image as calculating mean and mode.
Numpy: This library is responsible for handling arrays.

3.2 Decomposition Description

3.2.1 Class Diagram

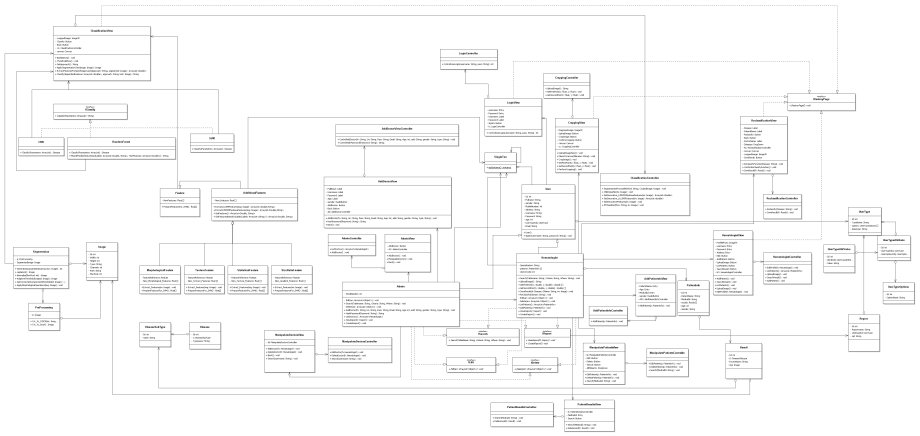


Figure 3: Class Diagram

Class name: User
Type: Concrete.
List of super classes: N/A.
List of sub classes: Hematologist, Admin.
Purpose: Class to encapsulate different user-types with their common attributes.
Collaboration:
This class aggregates class usertype.
Assists class LoginView.
Assisted by class SingleTon.
Extended by both classes, User and Admin.
Attributes: Id, Full name, username, password, Gender, Mobile Number, Address, age, user type object, Email.
Operations: SignIn(String username, String password).

Class name: UserType
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To allow scalability to add new user types.
Collaboration: This class is aggregated by class User, UserTypeAttribute and Report.
Attributes: id, TypeName, options[ ].
Operations: None.

Class name: UserTypeOptions

Type: Concrete.
List of super classes: N/A.
List of sub classes: None.
Purpose: Class to contain possible options for all user types.
Collaboration: This class is aggregated by class UserTypeAttribute.
Attributes: Id, optionname, datatype.
Operations: None.

Class name: UserTypeAttribute
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To represent what options each usertype has.
Collaboration: This class aggregates UserType, UserTypeOptions, aggregated by class UserTypeAttrValue.
Attributes: Id, UserType object, UserTypeOptions object.
Operations: None.

Class name: UserTypeAttrValue
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To contains the actual values of user-type options attributes.
Collaboration: This class aggregates class UserTypeAttribute.
Attributes: id, UserTypeAttribute Object, value.
Operations: None

Class name: Admin
Type: concrete.
List of super classes: User.
List of sub classes: None.
Purpose: To represent the admin.
Collaboration:
Extends class User.
Assisted by class Hematologist.
Associates classes AdminView, ManipulateDoctorsView.
Implements Both interfaces IReport and ISearch.
implements Interfaces (ISearch and IReport).
Attributes: Id,SerialNumber.
Operations:
AddDoctor(fn, Un, Pass, Email, Age, add, gender, Type).
EditDoctor (Hematologist H).
DeleteDoctor (Hematologist H).
Search (String Tablename, String Criteria, String whereClause).
HashPassword (String Password).
ListDoctors ().

ViewReport ().
CreateReport ().

Class name: Hematologist.
Type: concrete.
List of super classes: User.
List of sub classes: None.
Purpose: To represent the Hematologist Expert.
Collaboration:
This class extends class User.
associates class Admin.
Aggregates class Patientinfo.
Associates class AddPatientView, ManipulatePatientsView, PatientResultView.
Implements interfaces ISearch, IReport.
Assisted by class SingleTon.
Attributes: Id ,Specialization, Patientinfo[ ].
Operations:
Search (Tablename, Criteria, whereClause).
UploadImage ().
GetFirstPoint().
GetSecondPoint().
SaveResults(String Disease, String DocName, Image img).
ReClassify (String Disease).
EditPatient (Patientinfo p), DeletePatient (Patientinfo p).
ListPatients ().
AddPatient (Patientinfo p).
ViewReport ().
CreateReport ().

Class name: ISearch
Type: Interface.
List of super classes: None.
List of sub classes: None.
Purpose: To allow searching with different Criteria.
Collaboration:
Classes (Admin, Hematologist) implements this class.
Attributes: None.
Operations: Search (String TableName, String Criteria).

Class name: IReport
Type: Interface.
List of super classes: None.
List of sub classes: None.
Purpose: To allow creating and viewing reports.
Collaboration:
classes (Admin, Hematologist) implements this class.

this classis assisted by class Report.
Attributes: None.
Operations:
CreateReport (Report R).
ViewReport (Report R).

Class name: Feature
Type: Abstract.
List of super classes: None.
List of sub classes: class AdditionalFeatures.
Purpose: To contain the basic features of the Image to be wrapped with additional features layers.
Collaboration: This class is inherited by class AdditionalFeatures.
Attributes: Id , Newfeatures [ ].
Operations: PrepareFeaturesFor-SVM().

Class name: AdditionalFeatures.
Type: concrete.
List of super classes: Feature.
List of sub classes: class TextureFeature, StatisticalFeature, MorphologicalFeature and Size Ratio.
Purpose: to append to the basic features.
Collaboration:
This class is inherited by classes TextureFeature, StatisticalFeature, MorphologicalFeature and Size Ratio.
inherits class Feature.
Attributes: Id ,Newfeatures [ ].
Operations: ExtractL1L2M5Features(Image img), ExtractL3M2M3MyelomaFeatures(Image img),GetFeatures(), GetFeaturesBasedOnLabels(ArrayList¡String¿ Labels)

Class name: TextureFeature.
Type: concrete.
List of super classes: AdditionalFeatures.
List of sub classes: None.
Purpose: To extract the texture features of the Image.
Collaboration: This class inherits class AdditionalFeatures.
Attributes: Id , New-Texture-Features [ ].
Operations: Exctract-Features(), PrepareFeaturesFor-SVM().

Class name: MorphologicalFeatures.
Type: concrete.
List of super classes: AdditionalFeatures.
List of sub classes: None.
Purpose: extract the morphological features of the Image.
Collaboration: This class inherits class AdditionalFeatures.
Attributes: Id ,New-Gradient-Features [ ].

Operations: Exctract-Features(), PrepareFeaturesFor-SVM().

Class name: IClassify
Type: Interface.
List of super classes: None.
List of sub classes: None.
Purpose: To allow classification with different classifiers strategies.
Collaboration: Class SVM and RNN implements this class.
Attributes: None.
Operations: Classify(Parameters [ ]).

Class name: StatisticalFeature.
Type: concrete.
List of super classes: AdditionalFeatures.
List of sub classes: None.
Purpose: extract the statistical features of the Image.
Collaboration: This class inherits class AdditionalFeatures.
Attributes: Id ,New-Gradient-Features [ ].
Operations: Exctract-Features(), PrepareFeaturesFor-SVM().

Class name: SizeRatioFeature.
Type: concrete.
List of super classes: AdditionalFeatures.
List of sub classes: None.
Purpose: extract the size ratio features of the Image.
Collaboration: This class inherits class AdditionalFeatures.
Attributes: Id ,New-Gradient-Features [ ].
Operations: Exctract-Features(), PrepareFeaturesFor-SVM().

Class name: IClassify
Type: Interface.
List of super classes: None.
List of sub classes: None.
Purpose: To allow classification with different classifiers strategies.
Collaboration: Class SVM and RNN implements this class.
Attributes: None.
Operations: Classify(Parameters [ ]).

Class name: CNN.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To allow classification with CNN classifier.
Collaboration: This class implements class IClassify, assisted by class Features.
Attributes: None.
Operations: Classify(Parameters [ ]).

Class name: RandomForest.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To allow classification with Random forest classifier.
Collaboration: This class implements class IClassify, assisted by class Features.
Attributes: None.
Operations: Classify(Parameters [ ]), FitAndPredict(ArrayList¡Double, String¿ featuresAndLabels, ArrayList¡Double¿ TestFeatures).
Class name: Image.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To contain the image.
Collaboration: This class is aggregated by classes(preprocessing ,segmentation, Result).
Attributes: int Id , int Width, int Height, String Type, int Channels, String Path, int MaxPixel.
Operations: None.

Class name: Preprocessing.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To convert the color space and do some processing on the image before segmentation.
Collaboration: This class aggregates class Image, aggregated by class segmentation.
Attributes: Image img.
Operations: Cvt-To-YCBCR(), Cvt-To-GRAY().

Class name: Segmentation.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: To remark (segment) the parts of the doctors interest in the image (cells).
Collaboration:
This class aggregates class Image, preprocessing.
Associates class ClassificationView.
Attributes: Id, SegmentedIamgeObject, PreprocessingObject.
Operations:
GenerateGuassianDistribution (Image YcbcrImage).
Laplacian ().
Normalize (int MaxPixel).
AdaptiveThreshold (Image original).

GetConnectedComponents(Image thresholded).
ApplyMorphologicalOperations (Image im).

Class name: Disease.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to identify different diseases.
Collaboration: This class aggregates class DiseaseSubType.
Attributes: int Id, DiseaseSubTypeObject, String Typename.
Operations: None.

Class name: DiseaseSubType.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to identify different subtypes of diseases.
Collaboration: This class is aggregated by class Disease, Result.
Attributes: Id, name.
Operations: None.

Class name: Result.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the final result doctors received from the classification.
Collaboration: This class aggregates classes DiseaseSubType, Image.
Attributes: Id, DiseaseSubTypeObject, Image im, String Docname.
Operations: None.

Class name: Report.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to allow multiple reports in the system.
Collaboration: this class aggregates class userType, aggregated by class GUI-views, associates class IReport.
Attributes: int Id, String reportname, usertypeiD, String SQL.
Operations: None.

Class name: CroppingView.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the cropping screen.
Collaboration:

this class aggregates class CroppingController.
Assisted by class Hematologist.
Implements interface IDestroy.
Attributes:
Image DiagnosisImage.
Button UploadImage, CropImage, ConfirmCropping.
Canvas canvas.
CroppingController cc.
Operations:
UploadImageView ().
PlaceOnCanvas (String filename).
GetReadyForCroppingImage ().
GetFirstPoint(double x, double y).
GetSecondPoint(double x, double y).
PerformCropping().

Class name: CroppingController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Cropping view.
Collaboration:
this class aggregated by CroppingView.
Attributes: None.
Operations:
UploadImage ().
GetFirstPoint (double x, double y).
GetSecondPoint (double x, double y).

Class name: LoginView.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the Login screen.
Collaboration:
this class aggregates class LoginController.
Assisted by class User.
Implements interface IDestroy.
Attributes:
Entry Username, Password.
Label username, password.
Button Signin.
LoginController lc.
Operations:
ControlUserLogin (String username, String pass).

Class name: LoginController.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Login view.
Collaboration:
this class aggregated by LoginView.
Attributes: None.
Operations:
ControlUserLogin (String username, String pass).

Class name: ClassificationView
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the Classification screen.
Collaboration:
this class aggregates class ClassificationController.
Assisted by class User, Svm, CNN, RandomForest, Segmentation, Feature.
Implements interface IDestroy.
Attributes:
ClassificationController cn.
Image croppedImage
Button Classify.
Button Back.
Canvas canvas.
Operations:
BackButton ().
TheWholeFlow().
GetApproach().
ApplySegmentation(Image TestImage).
ExtractFeaturesForSpecificApproach(String Approach, Image segmented)
Classify(ArrayList¡double¿ AppendedFeatures, String approach, Image test)

Class name: ClassificationController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Classification view.
Collaboration:
this class aggregated by ClassificationView.
Attributes: None.
Operations:
SegmentationProcess (String method, Image croppedImage).
CNNCLassifier (Image im).
RFClassifier (String flow, Image im).

GetDecorativeFeatures (Image im).
GetDecorative-L3M2M3MyelomaFeatures(Image im).
GetDecorative-L1L2M5Features(Image im).

Class name: ReClassificationView
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the ReClassification screen.
Collaboration:
this class aggregates class ReClassificationController.
Assisted by class Hematologist.
Implements interface IDestroy.
Attributes:
ReClassificationController cn.
Label Disease, patientName, DoctorName.
Button Reclassify, Save Result, Back.
DropDown Diseases.
Canvas canvas.
Image CroppedImage.
Operations:
ReClassifyFunction (String Disease).
ConfirmReClassifyFunction ().
SaveResult (Result r).

Class name: ReClassificationController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the ReClassification view.
Collaboration:
this class aggregated by ClassificationView.
Attributes: None.
Operations:
ReClassify (String Disease).
SaveResult (Result r).

Class name: AddPatientinfoiew
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the AddPatientinfo screen.
Collaboration:
this class aggregates class AddPatientinfoController.
Assisted by class Hematologist.
Implements interface IDestroy.

Attributes:
AddPatientinfoController APC.
Entry patientName, Age, Gender, MedicalId.
Operations:
AddPatient (Patientinfo p)

Class name: AddPatientinfoController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Add-Patient-info view.
Collaboration:
this class aggregated by AddPatientinfoView.
Attributes: None.
Operations:
AddPatient (Patientinfo p)

Class name: ManipulatePatientsView
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the Manipulate-Patients screen.
Collaboration:
this class aggregates class ManipulatePatientsController.
Assisted by class Hematologist.
Implements interfaces IDestroy.
Attributes:
ManipulatePatientsController lp.
Button Edit, Delete, Search.
DropDown AllPatients.
Operations:
EditPatient (Patientinfo p).
DeletePatient (Patientinfo p).
Search (String Medica-lId).

Class name: ManipulatePatientsController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the ManipulatePatients view.
Collaboration:
this class aggregated by ManipulatePatientsView.
Attributes: None.
Operations:
EditPatient (Patientinfo p).
DeletePatient (Patientinfo p).

Search (String Medica-lId).

Class name: PatientResultsView
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the Patient-Results screen.
Collaboration:
this class aggregates class PatientResultsController.
Assisted by class Hematologist.
Implements interfaces IDestroy.
Attributes:
PatientResultsController ls.
Button Search.
Entry medicalId.
Operations:
DeleteResult (Result r).
Search (String Medicald).

Class name: PatientResultsController.
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Patient-Results view.
Collaboration:
this class aggregated by PatientResultsView.
Attributes: None.
Operations:
DeleteResult (Result r).
Search (String Medicald).

Class name: ManipulateDoctorsView
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the Manipulate-Doctors screen.
Collaboration:
this class aggregates class ManipulateDoctorsController.
Assisted by class Admin.
Implements interfaces IDestroy.
Attributes:
ManipulateDoctorsController ld.
Button Edit, Delete, Search.
DropDown AllDoctors.
Operations:
EditDoctor (Hematologist H).

DeleteDoctor (Hematologist H).
Search (String Username).

Class name: ManipulateDoctorsController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Manipulate view.
Collaboration:
this class aggregated by ManipulateDoctorsView.
Attributes: None.
Operations:
EditDoctor (Hematologist H).
DeleteDoctor (Hematologist H).
Search (String Username).

Class name: IEdit
Type: interface.
List of super classes: None.
List of sub classes: None.
Purpose: to provide the Editing functionality in different subclasses.
Collaboration:
this class implemented by Classes Hematologist and Admin.
Attributes: None.
Operations:
Edit (*argv)

Class name: IDelete
Type: interface.
List of super classes: None.
List of sub classes: None.
Purpose: to provide the Deleting functionality in different subclasses.
Collaboration:
this class implemented by Classes Hematologist and Admin.
Attributes: None.
Operations:
Delete (*argv).

Class name: SingleTon
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to minimize number of connections.
Collaboration:
this class assists classes itself, User, AdditionalFeatures.
Attributes: None.

Operations:
getInstance ().

Class name: IDestroy
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to Switch between views.
Collaboration:
all views classes implements this class.
Attributes: None.
Operations:
DestroyPage ().

Class name: HematologistView
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the Hematologist profile screen.
Collaboration:
This class aggregates class HematologistController.
Assisted by class Hematologist.
Implements interface IDestroy.
Attributes:
Image Tk Profile photo
Entry username, password, Address.
Button Edit, AddPatient, UploadImage, ListPatients, SearchResult.
HematologistController HC.
Operations:
AddPatient ().
SearchResult ().
ListPatients ().
UploadImgae ().
EditProfile (Hematologist h).

Class name: HematologistController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Hematologist profile view.
Collaboration:
this class aggregated by HematologistView.
Attributes: None.
Operations:
AddPatient ().
SearchResult ().

ListPatients ().
UploadImgae ().
EditProfile (Hematologist h).

Class name: AdminView
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to represent the Admin screen.
Collaboration:
this class aggregates class AdminController.
Assisted by class Admin.
Implements interface IDestroy.

Attributes:
Button AddDoctor.
AdminController AC.
Operations:
AddDoctor ().
ManipulateDoctor ().
Back ().

Class name: AdminController
Type: concrete.
List of super classes: None.
List of sub classes: None.
Purpose: to control the Admin view.
Collaboration:
this class aggregated by LoginView.
Attributes: None.
Operations:
AddDoctor ().
ListDoctors ().

3.2.2 Activity Diagram

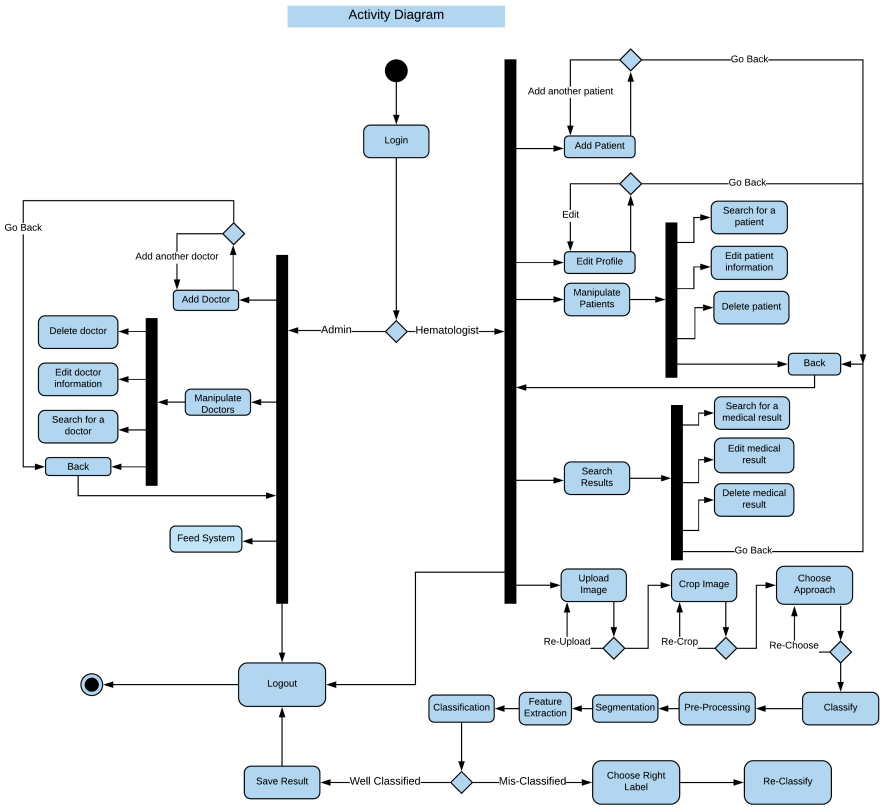


Figure 4: Activity Diagram

3.2.3 State Diagram

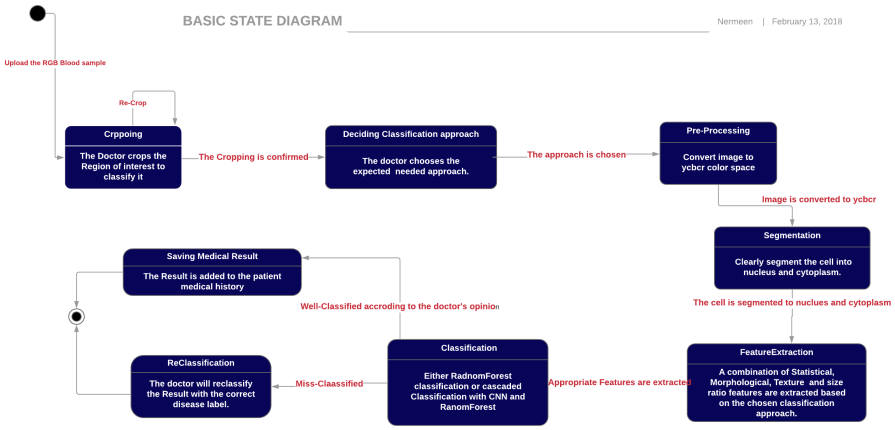


Figure 5: State Diagram

3.2.4 Sequence Diagram

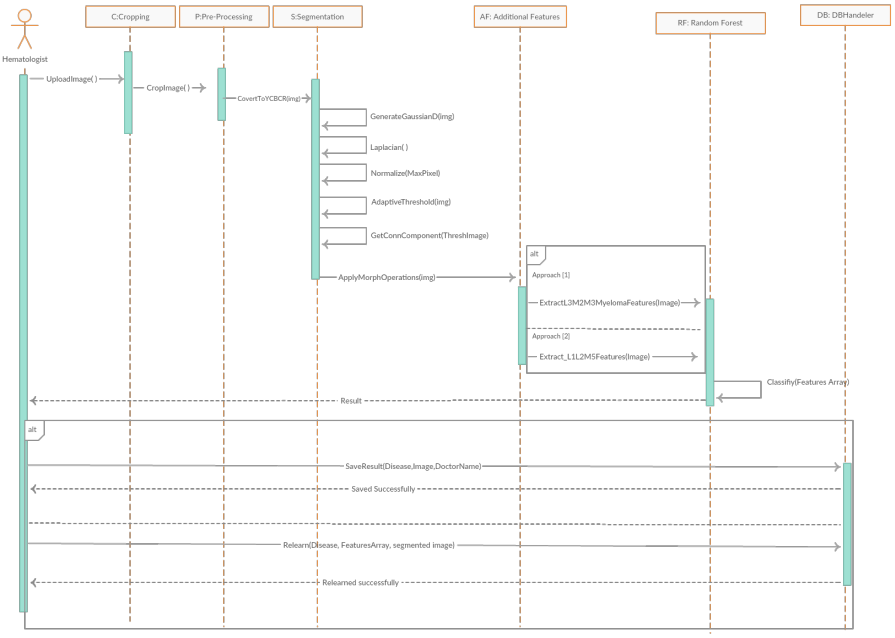


Figure 6: Disease Classification Sequence diagram

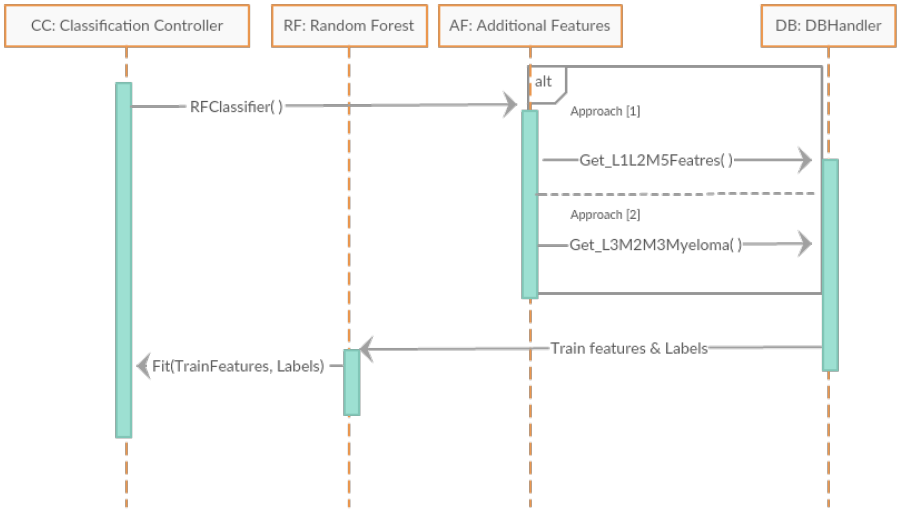


Figure 7: Classifier Training Sequence diagram

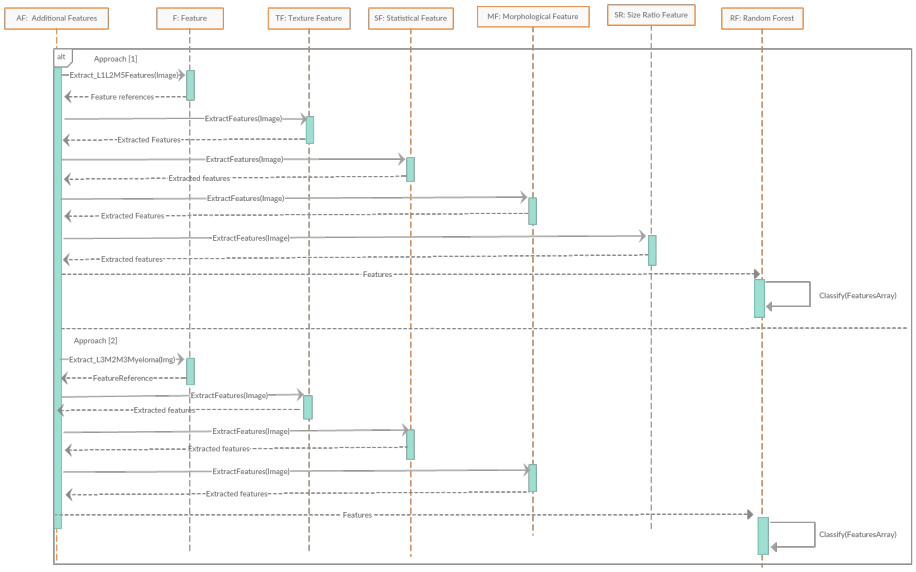


Figure 8: Feature Extraction Sequence diagram

3.3 Design Rationale

As mentioned previously, we have used Model-View-Controller (MVC) as our architectural model because it helped us separate the functionality and data of our system from the presentation. So, we can easily make modifications, re-use

and optimize functionality part as it is our core. Also, as we are developing software for medical use which is very sensitive when dealing with data and results so this part should be developed in very efficient and reliable way.
We had many alternatives concerning core algorithm so we have tested them and were able to choose the best that fits our problem. Those alternatives are SVM, CNN, RF, NB and SGD.

SVM: Linear SVM is given a set of train data which belong to a certain class to find an optimal separating line. It tries to maximize the distance between each class to avoid misclassification. Then a test data are given to be classified to one of the classes formed before[5].

CNN: Belongs to deep learning class, designed to require minimal preprocessing. It consists of two layers which are input and output as well as multiple hidden layers. These layers consist of convolutional layers, pooling layers, fully connected layers and normalization layers.

RF: It starts with a decision tree as a machine learning technique. It models the data and draws multiple curves and then chooses the strongest curve which contains the greater number of curves assembled together. The data starts to move through different sub branches of the tree until it reaches a decision[2,12,13].

NB: It works on conditional probability. It calculates the probability of the input relative to some decisions that was previously taken. It is well suited when the input has large number of dimensions[6].

SGD: Supports multi-class classification. In learning process the classifier run a binary classifier discriminating between one of the classes and the rest of them. At testing, it calculates the confidence score of the input with respect to all the classes and then chooses the class with the highest confidence.

We have chosen RF as our classifier because it was the best one that was able to differentiate between different types and the one which has given us the highest accuracy. Also, the architecture that this classifier was based on matches our problem as we have three parent disease classes and each one has many sub-classes as their sub-types.

4 Data Design

4.1 Data Description

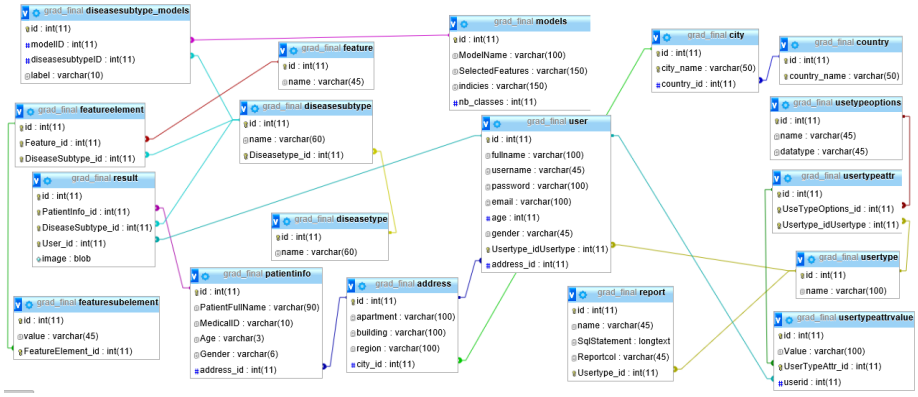


Figure 9: Databse

User: This table contains id, email, fullname, gender, age, username, password and it also contains UserType_IdUserType because we have two usertypes (Admin, Hematologist) and each type have different privileges. In addition, address_id is a foreign key from table Address.
UserType: This table contains id, name to be able to differ between the different users.
UserTypeOptions: This table contains id, name of the option and its datatype in which each user can have a couple of options.
UserTypeAttr: In this table each usertype have different options. So, we get the UserTypeOptions_id, UserType_IdUserType to know the different options for each usertype.
UserTypeAttrValue: This table contains id, Value of each option specified by UserTypeAttr_id and userid that is foreign key from table User.
Address: This table contains id, apartment, building, region and city_id that indicates which city does the user lives in and it's a foreign key from table City.
City: This table contains id, city_name and country_id which is foreign key from table Country.
Country: This table contains id and country_name.
Report: In this table the doctor may need to save an SQL statement to view specific data. This table contains an id, name of the report, the SQLStatement itself and Usertype_id because each type have different reports to view.
Feature: This table contains id, name of the feature.
FeatureElement: This table contains id, Feature_id that refers to the feature itself and the DiseaseSubType_id which is the disease for this feature.
FeatureSubElement: This table contains id, FeatureElement_id and the value. So, each Feature Element record will be mapped to various Feature Sub Ele-

ments.
DiseaseType: This table contains id, name of the disease whether its AML, ALL, Myeloma.
DiseaseSubType: This table contains the Subtypes of the diseases mentioned in the table (Disease Type). So, it contains id, name of the subtype and the DiseaseType_id of the disease itself.
Result: This table contains the results for each patient. It contains id, PatientInfo_id, DiseaseSubType_id, User_id that refers to the Hematologist_id and the sample Image itself.
Models: This table contains id, ModelName which is the saved classifier model, SelectedFeatures gained by using genetic algorithm, indices of the selected features and nb_classes are the number of trained classes in this model.
DiseaseSubTypeModels: This table shows that each model contains many diseasesubtypes. Each diseasesubtype contains a label. So, this table contains id, modelID foreign key from table Model, diseasesubtypeID foreign key from DiseaseSubType table and the label.
PatientInfo: This table contains id, PatientFullName, Age, Gender, MedicalID and address_id that is foreign key from Address table.

4.2 Data Dictionary

Security is achieved by hashing the passwords whereas Class Admin include a function called HashPassword (String pass).

Reliability is achieved by using singleton design pattern.

Extensibility is achieved by using strategy design pattern to switch between classifiers, interface IClassify encapsulate the classification behavior and classes Random Forest, CNN implement it. In addition, decorative design pattern achieves extensibility by appending the features vector dynamically.

Maintainability is achieved by using Entity Attribute Value Model (EAV) in the user classes (User, UserType, UserTypeOptions, UsertypeAttr, UserTypeAttrValue). The system is able to host different type of users with different options for each type.

5 Component Design

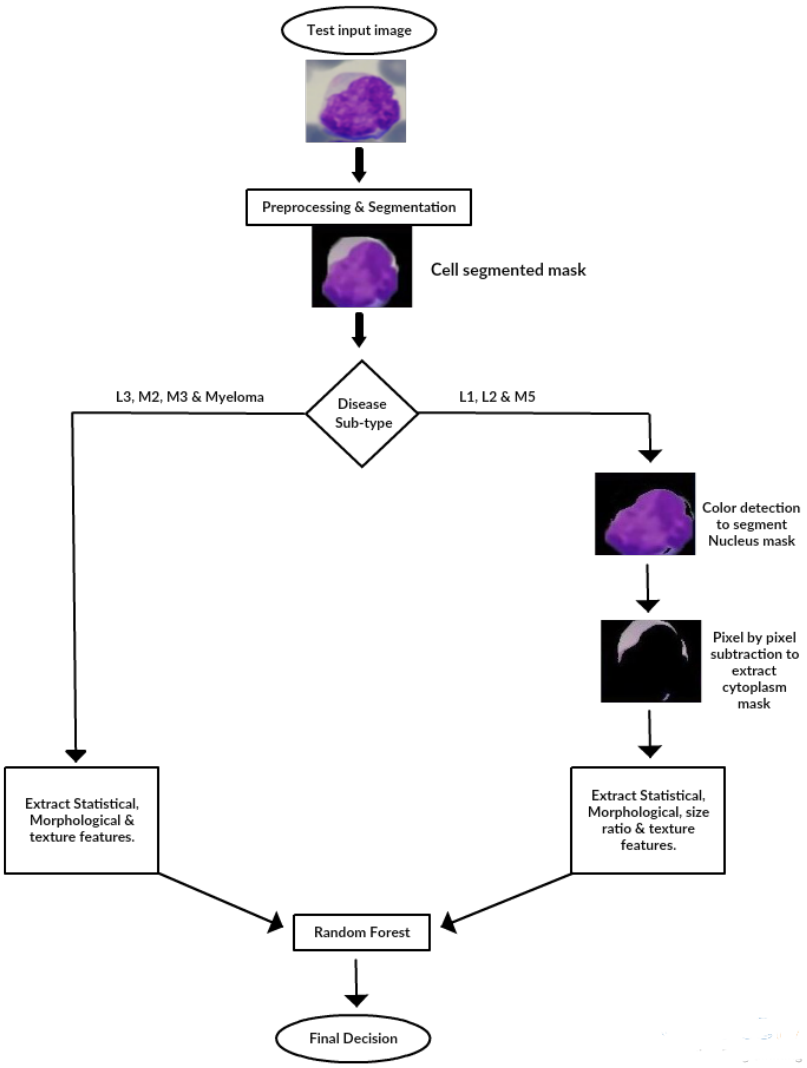


Figure 10: Flowchart of our system approach

5.1 Pre-processing

In this phase we prepare the blood sample image for the segmentation process by converting our input images from RGB color space to YCbCr space. Our choice to the YCBCR color space (Y: Luminance, CB: Blue Value, CR: Red Value) was due to the reddish and bluish colors of our blood samples. After converting

images to YCbCr space, the extracted Cb and Cr coefficients are used for cell segmentation process. Sample input image before and after conversion is shown in fig. 11.

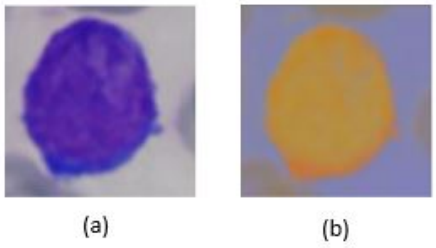


Figure 11: Input images before and after Preprocessing (a)RGB Color Space , (b)YCBCR Color Space

5.2 Segmentation

5.2.1 Cell Segmentation

The purpose of this phase is to segment the whole cell from the relative background as shown in Fig.12. The extracted Cb and Cr coefficients from our training images during preprocessing phase are now used to build a Gaussian Distribution as shown in equation ——.

——add the gaussian distribution model here please ——

This Gaussian distribution is applied on test images in the YCbCr space to extract our valuable pixels that are most probable included to our regions of interest ROI. After applying our defined distribution, normalization is applied followed by adaptive threshold algorithm.

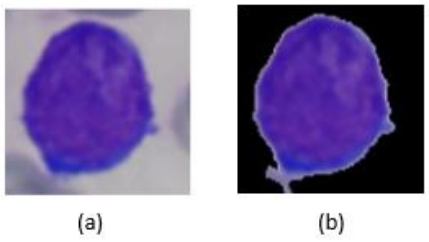


Figure 12: (a) Original RGB image before segmentation, (b)After Segmentation

5.2.2 Nucleus & Cytoplasm Segmentation

The result from cell segmentation is a mask containing only the cell. Color detection is applied on the cell mask with specified range of colors to segment

nucleus mask. By simple pixel to pixel subtraction of these two masks we can easily extract an accurate mask for the cytoplasm as shown in Fig.7

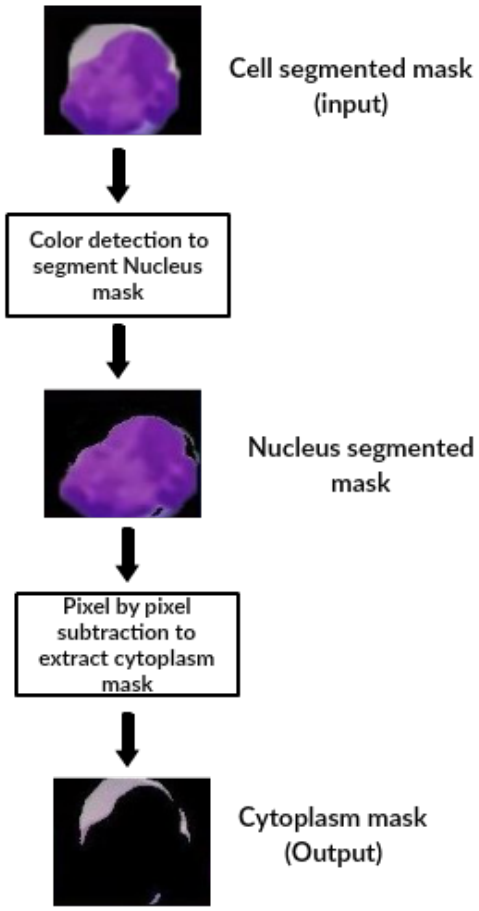


Figure 13: Nucleus and Cytoplasm Segmentation

5.3 Feature Extraction

The system mainly extracts 29 features of 4 types of features Morphological, statistical, texture and size ratio between nucleus and the cytoplasm. The differentiation between multiple types of both ALL, AML and Myeloma required different types of features to be considered as they are visually similar. Those features are morphological, statistical, texture and size ratio. We have different set of features calculated according to the approach chosen by the doctor. The first approach includes L1, L2, M5 as a set while the other set includes L3, M2, M3 and Myeloma, so based on the doctors decision one of the approaches will

be followed. We considered ratio features because they are invariant to scaling. Now we will discuss each feature in brief.

5.3.1 Morphological Features

These features represent shape of the cell and its dimensions.

Area to perimeter ratio It is the ratio between the actual number of pixels in the Region of interest and the distance between each adjoining pair of pixels around the border of the ROI.

$$AreaToPerimeterRatio = \frac{Area}{Perimeter} \tag{1}$$

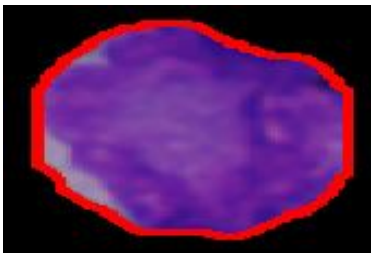


Figure 14: Region of interest

Circularity This feature measures the complexity of the perimeter of the circular object.

$$Circularity = \frac{Perimeter}{(4 * Area * pi)} \tag{2}$$

Elongation It is the ratio between length of the smallest rectangle containing the ROI and width of the smallest rectangle containing the ROI as shown in Fig. 15. It is also known as the growth in one direction of the ROI.

$$Elongation = \frac{LSR}{WSR} \tag{3}$$

where LSR is the length of the smallest rectangle containing the ROI and WSR is the width of the smallest rectangle containing the ROI

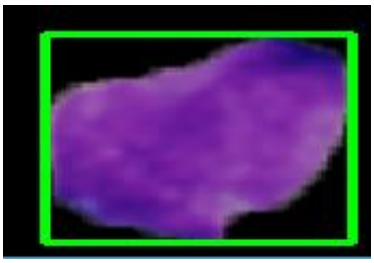


Figure 15: Smallest rectangle containing the ROI

Major to minor axis length ratio It is the ratio between the major axis of the ellipse containing the ROI and the minor axis of the ellipse containing the ROI as shown in Fig.16

$$MajorToMinorAxisLengthRatio = \frac{MajorAxisLength}{MinorAxisLength} \tag{4}$$

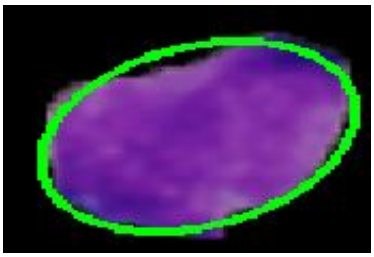


Figure 16: Ellipse containing the ROI

Extent It is the proportion of ROI area to the area of its bounding rectangle.

$$Extent = \frac{Area}{(Width * Length)} \tag{5}$$

Solidity It is the proportion of ROI area to area of its convex hull.

$$Solidity = \frac{Area}{Convexarea} \tag{6}$$

5.3.2 Statistical Features

These features also concern cell shape information but from different perspective. The calculated features are the following

Mode It is defined as most frequent value of the pixels intensity of the ROI.

Mean It is the average value of the pixels intensity of the ROI.

Standard deviation Standard deviation of the pixels intensity of the ROI.

Variance Variance value of the pixels intensity of the ROI.

Sum Sum of the pixels intensity of the ROI.

Gradient Angles' gradient is calculated by Canny edge detection.

5.3.3 Haralicks Texture Features

These features concern details in the cell like holes and granules. We implemented Haralicks features [1]. It is a set of 14 texture features calculated from the gray level co-occurrence matrix using 4 directions of adjacency [3]. These features are angular second moment, contrast, correlation, variance, inverse different moment, sum average, sum variance, sum entropy, entropy, difference entropy, difference variance, measure of correlation 1, measure of correlation 2 and maximum correlation coefficient.

$$\mathbf{G} = \begin{bmatrix} p(1,1) & p(1,2) & \cdots & p(1,N_g) \\ p(2,1) & p(2,2) & \cdots & p(2,N_g) \\ \vdots & \vdots & \ddots & \vdots \\ p(N_g,1) & p(N_g,2) & \cdots & p(N_g,N_g) \end{bmatrix}$$

Figure 17: Gray level co-occurrence matrix[3]

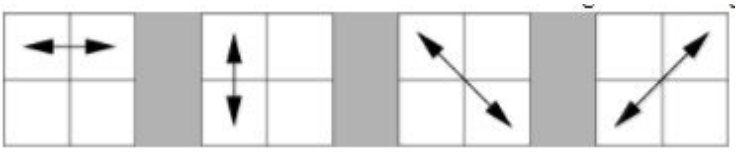


Figure 18: Four directions of adjacency to calculate Haralick's features[3]

Angular Second Moment	$\sum_i \sum_j p(i,j)^2$
Contrast	$\sum_{n=0}^{N_g-1} n^2 \{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)\}, \lvert i-j \rvert = n$
Correlation	$\frac{\sum_i \sum_j (ij)p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$ where μ_x , μ_y , σ_x , and σ_y are the means and std. deviations of p_x and p_y , the partial probability density functions
Sum of Squares: Variance	$\sum_i \sum_j (i-\mu)^2 p(i,j)$
Inverse Difference Moment	$\sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j)$
Sum Average	$\sum_{i=2}^{2N_g} i p_{x+y}(i)$ where x and y are the coordinates (row and column) of an entry in the co-occurrence matrix, and $p_{x+y}(i)$ is the probability of co-occurrence matrix coordinates summing to $x+y$
Sum Variance	$\sum_{i=2}^{2N_g} (i - f_8)^2 p_{x+y}(i)$
Sum Entropy	$-\sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\} = f_8$
Entropy	$-\sum_i \sum_j p(i,j) log(p(i,j))$
Difference Variance	$\sum_{i=0}^{N_g-1} i^2 p_{x-y}(i)$
Difference Entropy	$-\sum_{i=0}^{N_g-1} p_{x-y}(i) \log\{p_{x-y}(i)\}$
Info. Measure of Correlation 1	$\frac{HXY - HXY1}{\max\{HX, HY\}}$
Info. Measure of Correlation 2	$(1 - \exp[-2(HXY2 - HXY)])^{\frac{1}{2}}$ where $HXY = -\sum_i \sum_j p(i,j) \log(p(i,j))$, HX , HY are the entropies of p_x and p_y , $HXY1 = -\sum_i \sum_j p(i,j) \log\{p_x(i)p_y(j)\} HXY2 = -\sum_i \sum_j p_x(i)p_y(j) \log\{p_x(i)p_y(j)\}$
Max. Correlation Coeff.	Square root of the second largest eigenvalue of $\mathbf{Q}$ where $\mathbf{Q}(i,j) = \sum_k \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)}$

Figure 19: 14 Texture features of Haralick equations [3]

5.3.4 Size ratio features

Segmentation of nucleus and cytoplasm is necessary for this problem to be able to extract some features such as nucleus cytoplasm area, nucleus cell area and nucleus cell perimeter.

Nucleus cytoplasm area It is the ratio of the area between the nucleus and the cytoplasm.

$$NucleusToCytoplasmArea = \frac{NucleusArea}{CytoplasmArea} \tag{7}$$

Nucleus cell area It is the ratio of the area between the nucleus and the cell.

$$NucleusToCellArea = \frac{NucleusArea}{CellArea} \tag{8}$$

Nucleus cell perimeter It is the ratio of the perimeter between the nucleus and the cell.

$$NucleusToCellPerimeter = \frac{NucleusPerimeter}{CellPerimeter} \tag{9}$$

5.4 Random Forest Classification

Random Forest classifier is the best classifier that is able to differentiate between different types and the one which gives us the highest accuracy. Also, the architecture that this classifier is based on matches our problem as we have three parent disease classes and each one has many sub-classes as their sub-types.

5.4.1 Random forest algorithm

Random forest algorithm [2,12,13] is a supervised classification algorithm that constructs a forest with several decision trees. Highest accuracy results are achieved with the higher number of trees.

5.4.2 Random forest algorithm advantages

Random forest algorithm achieved successes in medical field as its one of the most powerful algorithms that is widely used in different applications [2]. It has many advantages as it can be used in different classification problems such as banking, stock market and E-commerce [2], it can be used for both classification and regression and it performs feature selection to only extracts the crucial features.

5.4.3 How Random forest algorithm works

Figure 20: How random forest algorithm works [2]

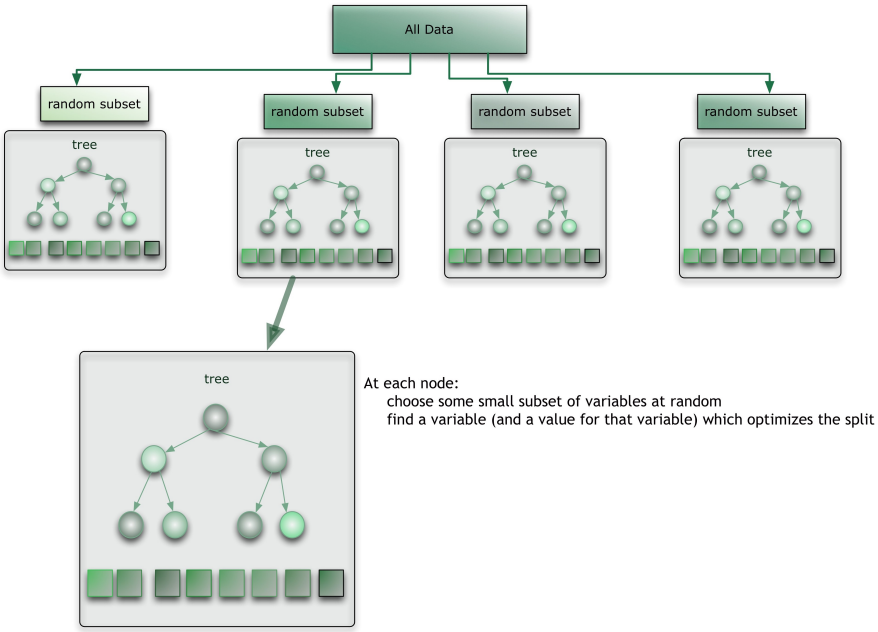


Figure 21: How random forest algorithm works

Pseudocode of the creation of Random Forest goes as shown in Fig. 22

1. Randomly select **"k"** features from total **"m"** features.
    1. Where **k << m**
2. Among the **"k"** features, calculate the node **"d"** using the best split point.
3. Split the node into **daughter nodes** using the **best split**.
4. Repeat **1 to 3** steps until "l" number of nodes has been reached.
5. Build forest by repeating steps **1 to 4** for "n" number times to create **"n" number of trees.**

Figure 22: Random Forest pseudocode [2]

Pseudocode of the prediction of Random Forest goes as shown in Fig. 23

1. Takes the **test features** and use the rules of each randomly created decision tree to predict the oucome and stores the predicted outcome (target)
2. Calculate the **votes** for each predicted target.
3. Consider the **high voted** predicted target as the **final prediction** from the random forest algorithm.

Figure 23: Random forest prediction pseudocode [2]

5.5 CNN Classification

[11] You might know about machine learning and deep learning. Deep learning is a subtype of machine learning. Convolutional Neural Network (CNN or ConvNet) is a specific kind of such deep neural network. To perform CNN classifier, you should create a model and include some layers to the model.

5.5.1 Convolutional Layer

[8] This layer is performed when we extract a segment from the image sized (k*k) centered at location (x,y). We then multiply the extracted segment element-by-element with convolution filter also sized (k*k) and add them all together to create a single output. As shown below in Fig.24

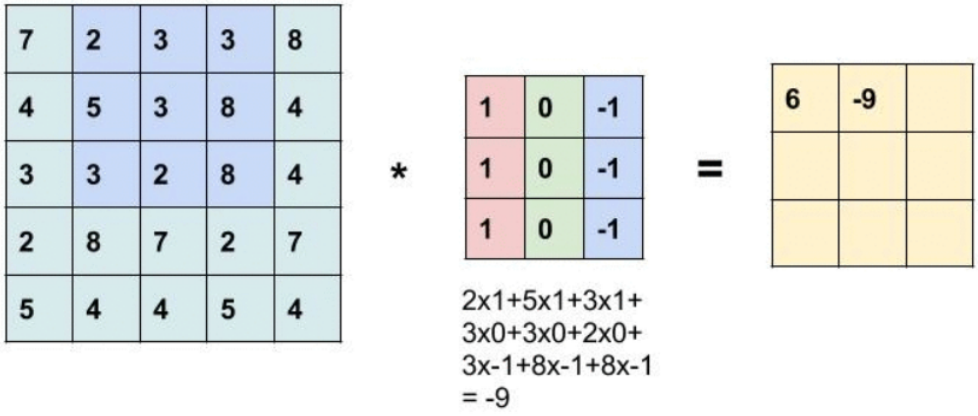


Figure 24: Convolutional Layer

5.5.2 Activiation Layer

[8][9] There are several activation functions each have a different function. For example, Softmax, Linear, Sigmoid, Tanh, Relu, Softsign, Softplus, Selu, Elu, Hard-sigmoid. Suppose we have filter size 3x3x3 and an input image 32x32x3.

In which 32x32 represents Width, Height and 3 represents the depth of the image. From this image we get 30x30x1 different locations. In fact, there is a single neuron corresponding to each location. The output from each location are called Activation and it serves as the input to the next layer.

5.5.3 Max Pooling Layer

[8] The pooling layer is mainly used to reduce the image size (Width and Height only, not Depth), computation and avoid overfitting. Note: Over Fitting is a condition occurs when the model accurately works on a given test data.
The common form of pooling is Maximum pooling shown in Fig. 25 Where we take a filter sized (k*k) and get the maximum value for each sized part of the image. The output obtained is a region that contains the maximum value from each part.

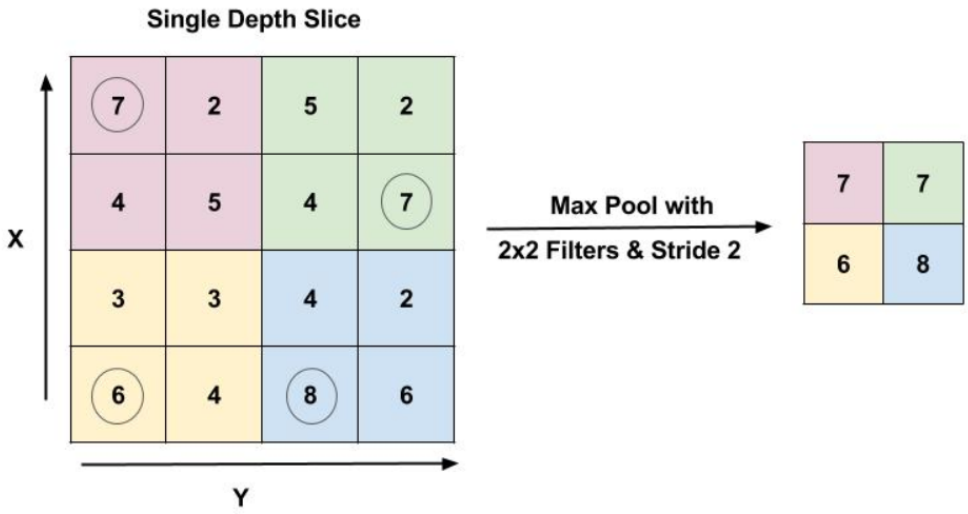


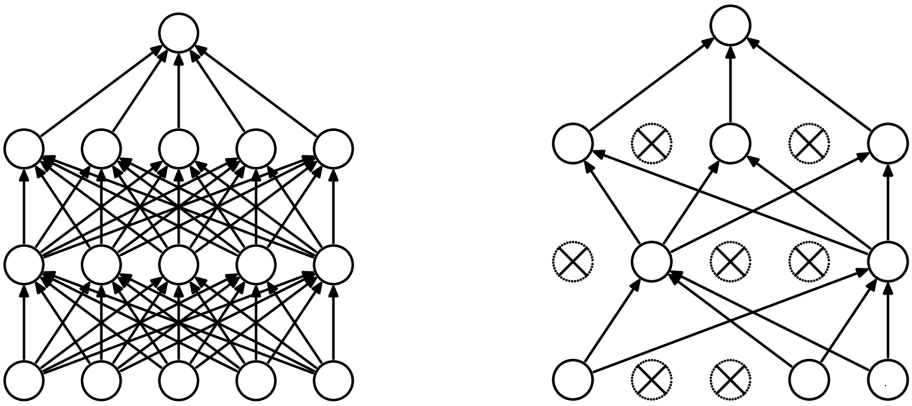
Figure 25: Pooling Layer

5.5.4 Flatten Layer

[10] This layer coverts the (3D or 2D) array to a single array.

5.5.5 Dropout Layer

[11] This layer improves the efficiency of neural networks mainly on supervised learning. And it also prevents the model from overfitting. The main idea is to drop random units along with their connections during the training phase as shown in Fig. **??**



(a) Before Dropout (b) After Dropout

Figure 26: Dropout Layer

5.6 Re-Learning

The system has the ability of learning from misclassified tests to enhance the future accuracy of the system. After the classification result is appeared, the doctor has the decision whether its misclassified or correctly classified. If the system misclassified, the doctor would has the ability to re-classify the cell. Experts are allowed to have the final decision either to accept the result from the proposed system or reclassify it with the correct label based on their medical experience. The system is capable of relearning from the misdiagnosed cases through fitting the system again with those newly classified samples by the expert doctors by repeating the training process.

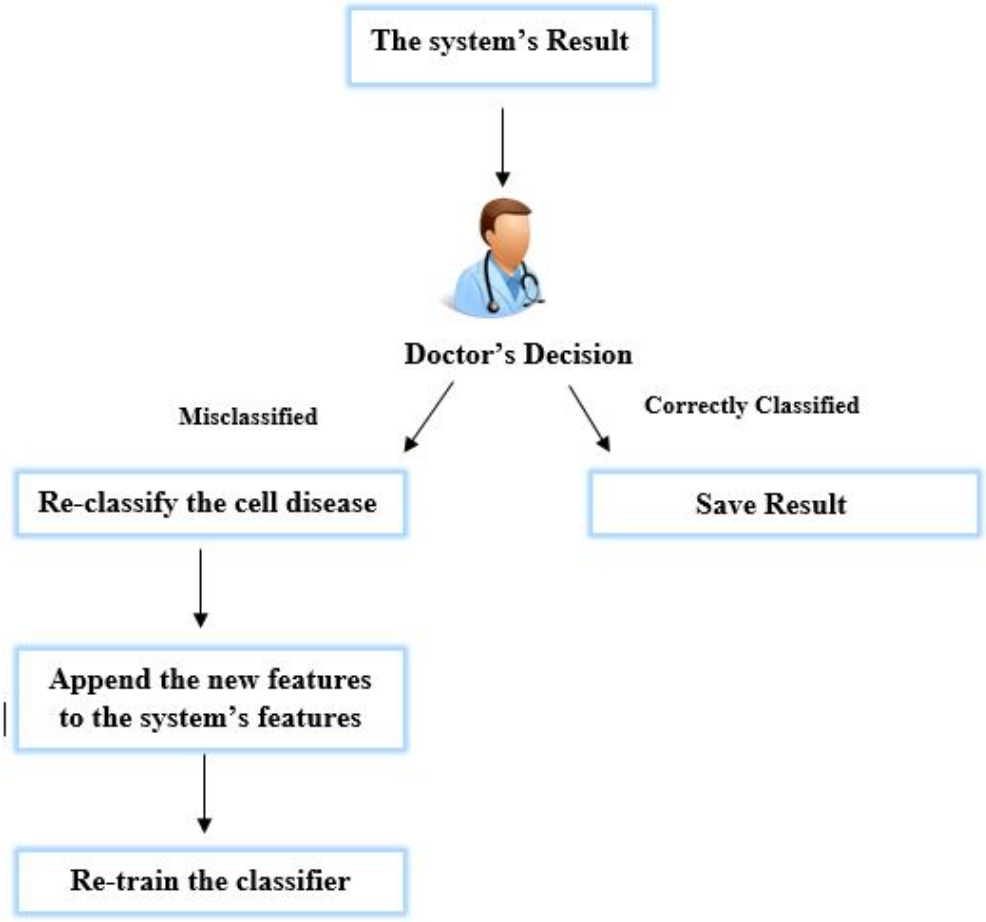


Figure 27: Re-learning Flowchart

6 Human Interface Design

6.1 Overview of User Interface

Our system Cancer Chaser user interface is very usable and clear. You can login whether you are an admin or a doctor. The system navigates you to different screens depends on your type. System admins have some duties such as manipulating doctors. While doctors are responsible for dealing with their patients. In fact, illustration for the whole system will be shown the upcoming sections (6.2, 6.3).

6.2 Screen Images

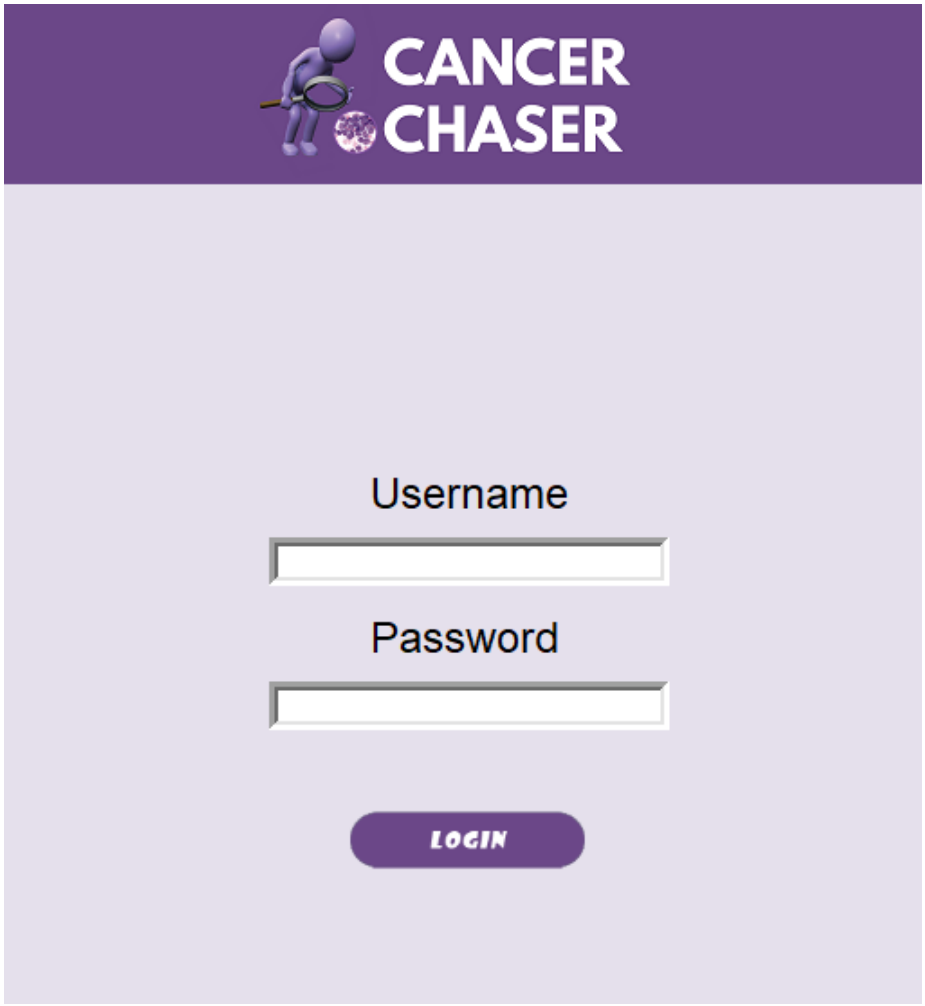


Figure 28: Login Screen

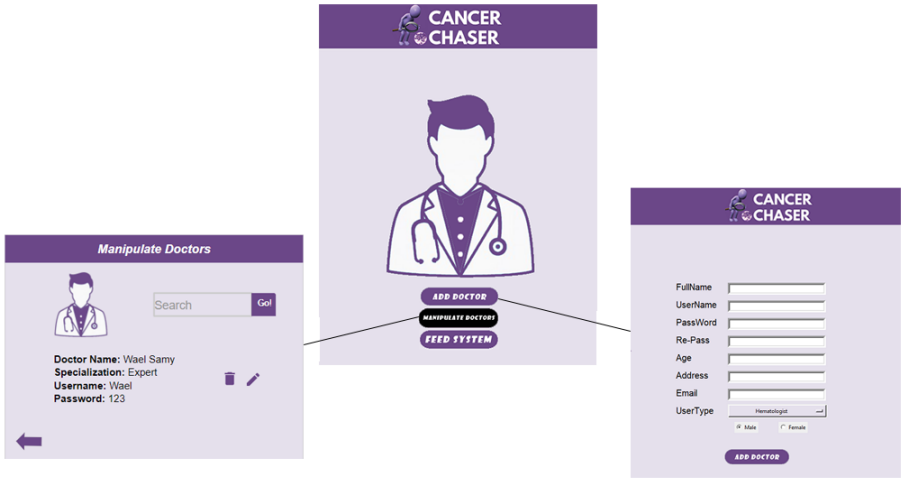


Figure 29: Admin Screen

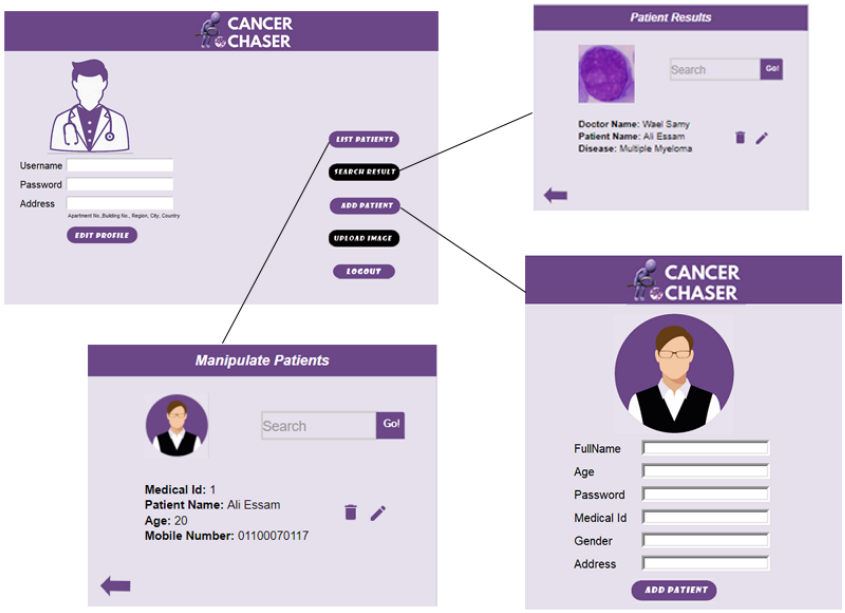


Figure 30: Hematologist Screen 1

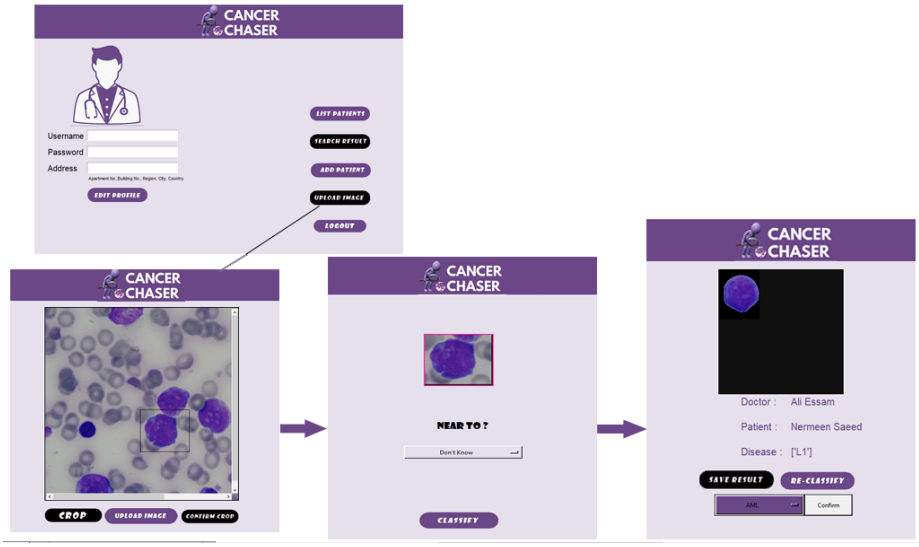


Figure 31: Hematologist Screen 2

Fig. 32 shows two text boxes. First is to enter username and second is for the password. If you logged in as an admin you will access the screens in Fig. 29, and if you logged in as a doctor youll access screens in Fig. 30, Fig. 31.

In Fig. 29, each admin have two options whether to add a new doctor to the system or manipulate existing doctors, such as viewing some personal information, searching for a doctor, editing doctors specialization or deleting a doctor from the system. Here comes the doctor screens in Fig. 30. First of all he can edit his personal information such as (Username, password, address, ..). The buttons shown at the right is (List Patients) in which the doctor can manipulate his patients through searching for a patient using his Medical Id, editing some information such as (Medical Id, Patient Name, Age, Gender, Mobile Number) and the doctor can also delete any patient from the system. Second button is (Search Result) and this opens a screen in which the doctor can view his medical history (Doctor Name, Patient Name, Disease) to perform some operations such as editing, deleting or searching for a specific medical history. Now it comes the third button (Add Patient) in which the doctor can add patients to the system. Moving to the final button (Upload Image) which is shown in Fig. 31. The initial screen is to upload the sample microscopic image for the patient. Then crop the region of interest and confirm this crop to move to the next screen. Now, the doctor will be able to choose an approach from the dropdown list whether the cropped image is nearly (L1-L2-M5) or (L3-M2-M3-Myeloma) or (dont know). After choosing an option and clicking classify, the last screen will now open. The final result is out and the doctor will have one of two options whether he needs to save the result if its accurate enough or re-classify the result by entering the right disease type to activate learning from misclassified results

module.

7 Requirements Matrix

Req. Id	Req. Type	Req. Name	Req. Description	Module	Status	Where In SDD
3.1.1	Required	Convert to YCBCR	Converts RGB image to YCBCR	Pre-Processing	Completed	Class diagram-Seqeunceof Testing
3.1.2	Required	Convert to Grayscale	Converts RGB/YCBCR image to Grayscale.	Pre-Processing	Completed	Class diagram
3.2.2	Required	Save test results	Hematologist will save his patient test results including his blood sample image, disease subtype, and doctor name.	Hematologist	In Progress	Class diagram, Sequence-of-Testing
3.2.7	Required	Upload Image	Hematologist will upload the blood sample image to be classified.	Hematologist	Completed	Class diagram , Activity Diagram, Sequence-of-Testing
3.2.8	Required	Crop Image	Hematologist will crop the cell to be classified.	Hematologist	Completed	Activity Diagram, Sequence-of-Testing
3.4.1	Required	Generate Gaussian Distribution and Get Max Pixel	Gaussian distribution will be generated using CB and CR components. And maximum Pixel is computed.	Segmentation	Completed	Class Diagram, Sequence-of-Testing
3.4.2	Required	Perform Laplacian Filter	Laplacian filter is applied on the image.	Segmentation	Completed	Class Diagram, Sequence-of-Testing
3.4.3	Required	Normalize	Divide image pixels over max pixel.	Segmentation	Completed	Class Diagram, Sequence-of-Testing

3.4.4	Required	Perform Adaptive Threshold	Adaptive Threshold is applied on the float image and any pixel less than the threshold value will be replaced by the corresponding pixel in the original image.	Segmentation	Completed	Class Diagram, Sequence-of-Testing
3.4.5	Required	Get Connected Components	Connected Components are calculated in order to remove components smaller than specific size.	Segmentation	Completed	Class Diagram, Sequence-of-Testing
3.4.6	Required	Apply Morphological Operations	Returns the image to its original state after segmenting the cell and removing noise.	Segmentation	Completed	Class Diagram, Sequence-of-Testing
	New Req	GetDecorative_ L1L2M5_Featurs	Append statistical, morphological, size ratio and texture features using decorative design pattern.	Feature Extraction	Completed	Class Diagram, Training Sequence
	New Req	GetDecorative_ L3M2M3Myeloma_Features	Append statistical, morphological and texture features using decorative design pattern.	Feature Extraction	Completed	Class Diagram, Sequence-of-Testing
3.7.4	Required	Classify	Extracts the segmented image features, evaluates the model, and predicts label to classify the disease.	Classification	Completed	Class diagram/ Activity diagram/
3.8.1	New Req.	Hash Password	Apply Hashing algorithm	Security	In Progress	Class Diagram
	New	GetReadyForDecorative	Returns the appended features array in a backward way using decorative design pattern.	FeatureExtraction	Completed	Class Diagram

Figure 32: Requirement Matrix

8 References

[1] Robert M. Haralick, "Statistical and structural approaches to texture," Proc. IEEE, vol. 67, no. 5, pp. 786-804, 1979.

[2] Polamuri, S. (2018). How the random forest algorithm works in machine learning. [online] Dataaspirant. Available at: http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/.

[3] Boland, M. V. (n.d.). Haralick texture features. Retrieved January 28, 2018, from http://murphylab.web.cmu.edu/publications/boland/boland$_n ode$26.$html$.

[4] Carolina Reta, Leopoldo Altamirano, Jesus A. Gonzalez, Raquel Diaz-Hernandez, Hayde Peregrina, Ivan Olmos, Jose E. Alonso, and Ruben Lobato. Segmentation and Classification of Bone Marrow Cells Images Using Contextual Information for Medical Diagnosis of Acute Leukemias, 2015.

[5] N. Cristianini, J. Shawe-Taylor, Support Vector Machines and other kernel-based learning methods. Cambridge University Press, 2000.

[6] Saxena, Rahul. How the Naive Bayes Classifier Works in Machine Learning. Dataaspirant, 6 Feb. 2017, dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/.

[7] 1.5. Stochastic Gradient Descent. 1.5. Stochastic Gradient Descent - Scikit-Learn 0.19.1 Documentation, 2017, scikitlearn.org/stable/modules/sgd.htmlimplementation-details.

[8] Image Classification Using Convolutional Neural Networks in Keras. Https://Www.learnopencv.com/Image-Classification-Using-Convolutional-Neural-Networks-in-Keras/, VIKAS GUPTA, 29 Nov. 2017, www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/.

[9] https://keras.io/activations/:

[10] Tata, Venkatesh. Simple Image Classification Using Convolutional Neural Network - Deep Learning in Python. Becoming Human: Artificial Intelligence Magazine, Becoming Human: Artificial Intelligence Magazine, 13 Dec. 2017,becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8

[11] Convolutional Neural Networks in Python. DataCamp Community, Aditya Sharma, 5 Dec. 2017, www.datacamp.com/community/tutorials/convolutional-neural-networks-python

[12] Liaw, Andy Wiener, Matthew. (2001). Classification and Regression by RandomForest. Forest. 23.

[13] StatSoft, Inc. (2013). Electronic Statistics Textbook. Tulsa, OK: StatSoft. WEB: http://www.statsoft.com/textbook/

[14] Application of Support Vector Machine and k-means Clustering Algorithms for Robust Chronic Lymphocytic Leukemia Color Cell Segmentation, nternational Conference on e-Health Networking, Applications and Services IEEE, 2013.

[15]J. R. Chaitali Raje, Detection of Leukemia in microscopic images us- ing image processing, Communications and Signal Processing (ICCSP), 2014 International Conference on IEEE, 2014.

[16] I. M. M. Sos Agaian Senior Member and I. Anthony T. Chronopoulos Senior Member, Automated Screening System for Acute Myelogenous Leukemia

Detection in Blood Microscopic Images, IEEE Systems Journal, 2014.

[17] Omid Sarrafzadeh1 , Hossein Rabbani1 , Alireza Mehri Dehnavi1 , Ardeshir Talebi2, DETECTING DIFFERENT SUB-TYPES OF ACUTE MYELOGENOUS LEUKEMIA USING DICTIONARY LEARNING AND SPARSE REPRESENTATION, Image Processing (ICIP), 2015 IEEE International Conference on, 2015.