Mini Project report on

# Eye Health Prediction System

**by**

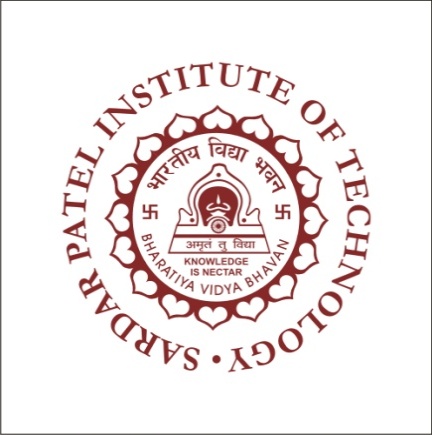
**Ghanshyam Gupta**

**Ashish Mishra**

Under the guidance of

**Internal Supervisor**

**Professor Mrs. Mayura Nagar**

****

Department of Master in Computer Applications

Sardar Patel Institute of Technology

University of Mumbai

2017-18

**CERTIFICATE OF APPROVAL**

This is to certify that the following students

Ghanshyam Gupta

Ashish Mishra

Have satisfactorily carried out work on the project entitled

“Eye Health Prediction System”

Towards the fulfillment of Mini project, as laid down by University of Mumbai during year 2017-18.

Project Guide

**PROJECT APPROVAL CERTIFICATE**

This is to certify that the following students

**Ghanshyam Gupta**

**Ashish Mishra**

Have successfully completed the Project report on **“Eye Health Prediction System”**, which is found to be satisfactory and is approved

At

SARDAR PATEL INSTITUTE OF TECHNOLOGY,

ANDHERI (W), MUMBAI.

INTERNAL EXAMINER EXTERNAL EXAMINER

Head of Department Principal

# Table of contents

Abstract ………………………………………………………………………………………….1

Objectives………………………………………………………………………………………..1

1. Introduction………………………………………………………………………………2
   1. Problem Definition
   2. Objective & Scope
   3. Existing System
   4. Proposed System
   5. System Requirements
2. Literature Survey………………………………………………………………………....3
3. Software Requirement Specification[SRS] & Design……………………………………3
   1. Introduction...…………………………………………………………………3
      1. Purpose
      2. Definitions
      3. System overview
   2. Overall description …………………………………………………………...4
      1. Product functions
      2. User characteristics
   3. Specific requirements ………………………………………………………...5
      1. External interface requirements (if any)
      2. Functional requirements (if any)
      3. Performance requirements (if any)
      4. Logical database requirement (if any)
      5. Software System attributes (if any)
      6. Other requirements (if any)
4. Project Analysis and Design…………………………………………………………………..6
   1. Methodologies Adapted………………………………………...……………………..6
      1. Detailed life Cycle of the Project
   2. Modules……………………………………………………………………………….9

4.2.1 UML Diagram

4.2.1 Use Case Diagram with Report

* 1. Architectural Design…………………………………………………………………13
  2. Database Schema/ Design……………………………………………………………14
     1. DFD diagram

5. Project Implementation & Testing…………………………………………………………….18

5.1 Work Break Down Structure ………………………………………………………..18

5.2 Gantt Chart…………………………………………………………………………...18

5.3 Code with reference to design………………………………………………………..19

5.7 Snapshot of UI & Reports……………………………………………………………20

5.8 Test cases & Report………………………………………………………………….25

5.8.1 Types of Testing……………………………………………………...……25

5.8.1.1 White Box Testing

5.8.1.2 Black Box Testing

5.8.2 Test Cases………………………………………………………………….26

5.8.3Test Report………………………………………………………………….28

5.8.4 Testing & Result………………………………………………………...…28

6. Documentation & Installation………………………………………………………………....32

* 1. User Manual with Installation procedure…………………………………………….32

1. Future Enhancements………………………………………………………………………...32
2. Limitations…………………………………………………………………………………...32
3. Conclusion…………………………………………………………………………………...32
4. Bibliography…………………………………………………………………………………32

**Abstract**

It might have happened so many times that you or someone yours need doctors help immediately, but they are not available due to some reason. The Health Prediction system is an end user support and online consultation project. Here we propose a system that allows users to get instant guidance on their eye health issues through an intelligent health care system online. The system is fed with various symptoms and the disease/illness associated with those systems. The system allows user to share their symptoms and issues. It then processes user’s symptoms to check for various illness that could be associated with it. Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with patient’s symptoms. If the system is not able to provide suitable results, it informs the user about the type of disease or disorder it feels user’s symptoms are associated with. If user’s symptoms do not exactly match any disease in our database, is shows the diseases user could probably have judging by his/her symptoms. It also consists of Feedback and administrator dashboard for system operations.

**Objectives**

Create a web based platform to help any patient to efficiently diagnose their disease based on their symptom.

The system helps the administration to add a new disease into the system, view users’ feedback.

The user can also view his/her past queries about disease symptom and their prediction.

1. **Introduction**

1.1 **Problem Definition**

Sometimes it might happen that access to a doctor is not possible due to some issues. During that time this system will serve as an alternate option for the user to diagnose their condition. This system helps in achieving just that. It acts as an alternate option for the users to diagnose their disease based on their symptom.

1.2 **Objective & Scope**

Create a web based system that will help the user in diagnosing their disease by their symptom. Currently, most eye disease symptoms have been fed into the system so that it can identify or diagnose most of the type of eye problems.

1.3 **Existing System**

Currently, doctors diagnose their patients based on their symptom and past experience. Also, there exists few web based applications which predict disease only statically.

1.4 **Proposed System**

In our system, we propose an intelligent approach with the help of data mining and machine learning which learns from the symptoms and their diseases.

1.5 **System Requirements**

Hardware Requirement:

1. Memory of 2 GB RAM or more

2. 1 GB (or more) available hard disk space

3. Monitor resolution of 1024 x 768 or higher

4. Intel i3 or more

Software Requirement:

1. Java Runtime Environment 1.7

2. Java SE JDK v7.0

3. Web Browser

4. Internet Connection

Operating System Requirement:

1. Windows 7/Linux or any other OS.

1. **Literature Survey**

Currently, most health prediction system depend largely on RDBMS for predicting the disease based on health. These RDBMSs’ are fed with mapping of disease with their respective symptoms. On the basis of symptoms, disease gets predicted.

1. **Software Requirement Specification[SRS] & Design**

**3.1 Introduction**

**3.1.1** Purpose

To help people diagnose their symptoms in case of emergency and provide an alternate free option to check their health.

**3.1.2** Definitions

First, user must create an account to use the system. Then after logging in, user will enter his symptoms and then the system will predict the most likely disease of the symptom. Also, the user can view his past predictions. Also, the user can send feedback to the admin. Admin can view the feedback and update the system accordingly. Also, admin can add new disease with their respective symptoms.

**3.1.3** System overview

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product.  General description of the project is discussed this document.

* 1. **Overall description**

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product.  General description of the project is discussed in section 2 of this document.  Section 3 gives the functional requirements, data requirements and constraints and assumptions made while designing the Application.  It also gives the user viewpoint of product.  Section 3 also gives the specific requirements of the product.  Section 3 also discusses the external interface requirements and gives detailed description of functional requirements. Section 4 is for supporting information.

**3.2.1** Product functions

Product shall contain the following functionalities:

Login Screen: This screen will allow user to login. without login user won’t be able to Use any feature. if user is not signed up he/she can use Guest login to try out app.

Registration Screen: Any new user can create his account in order to use the system.

Home Screen: This page has 3 sub sections:

* Predict System: Here the user can select his symptoms from a list of symptoms and based on the input the system will predict appropriate disease associated with the symptom.
* Add Feedback: The user can send his feedback to the administrator regarding the system. The admin will view the feedback and may respond to the user.
* Prediction History: The user can view his prediction history over the period of time.

Log out Screen: Logout button will be provided on the home screen on pressing the logout button, the system will destroy any session attributes of the user during the course of logged in time.

**3.2.2 User characteristics**

User of the app will be all the type of any type.

* 1. **Specific requirements** 
     1. **External interface requirements**

The interface of the app needs to be intuitive and attractive to keep the attention of the user. Hence we have used a new concept in UI designing called as Material Design and Bootstrap. So we have used CDN of both these UI libraries.

* + 1. **Functional requirements**

Usability: Since this is a web based platform, the usability is simple and straight forward. User needs to have a web browser in order to use the system.

Accessibility: Since we will deploy it over the web, the system will be accessible to anyone having an internet connection.

Reliability: App is made using standard SDLC priciples, hence it is a reliable system.

Availability: System will be available 24x7.

Performance: System is efficiently made in order to avoid any kind of lag.

Interactivity: App is made interactive so that user feels intuitive using system.

Security: The app hashes users’ password to improve security.

And only registered user should be able to use the app.

1. **Project Analysis and Design**

4.1 Methodologies Adapted

Methodology adapted for the project is incremental model. The functionality of the products will be incremented with every new release. this model is suitable for the given product as there is immediate need for delivery of the core functionality.

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a “multi-waterfall” cycle. Cycles are divided up into smaller, more easily managed modules. Incremental model is a type of software development model like V-model, Agile model etc.

In this model, each module passes through the requirements, design, implementation and testing phases. A working version of software is produced during the first module, so you have working software early on during the software life cycle. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

Example of Incremental model in software testing. In the diagram above when we work incrementally we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it’s complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers. Likewise, in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.



**Advantages of Incremental model:**

* Generates working software quickly and early during the software life cycle.
* This model is more flexible – less costly to change scope and requirements.
* It is easier to test and debug during a smaller iteration.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

**Disadvantages of Incremental model:**

* Needs good planning and design.
* Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
* Total cost is higher than waterfall.
  + 1. Detailed life Cycle of the Project
* 1st release:

The product will have login page and registration screen.

* 2nd release:

The system will have home page created along with feedback page, history page of the user as well as the admin.

* 3rd release:

Data mining and machine learning algorithms will be used with the help of Weka for predicting.

* 4th release:

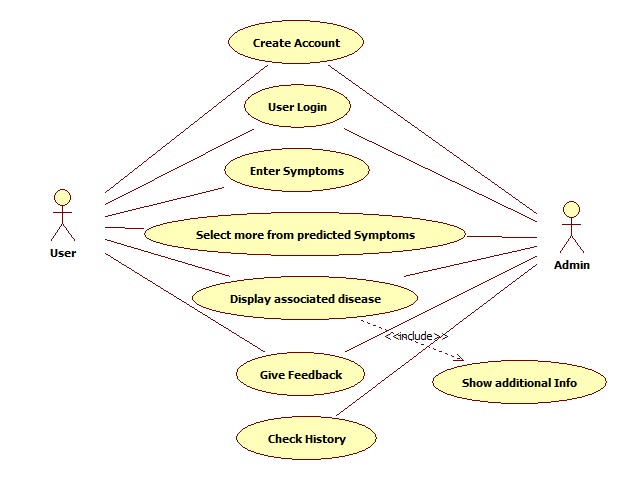
Database will be created for user authentication and details.

* 5th release:

Admin response and user feedback mechanism is added.

4.2 UML Diagram

4.2.1 Use Case Diagram with Report

****

Specification table:

|  |  |
| --- | --- |
| Use Case ID: | HD1 |
| Use Case Name: | Eye health prediction system |
| Created By: | Ghanshyam Gupta, Ashish Mishra |
| Date Created: | 28/7/2017 |

|  |  |
| --- | --- |
| Actor: | User, Admin |
| Description of use cases: | Login: User and Admin login to system.  Enter Symptom: User will enter his symptom into the system  Select More: User will select any other symptom in case he is not sure.  Display Prediction: System will predict the most possible symptom associated.  Provide Feedback: User can provide feedback into the system.  Check History: User can check his search history of the prediction. |
| Preconditions: | User and admin must log in to the system. |
| Post conditions: | User wil get his health condition diagnosed by the system. |
| Extends: | N/A |
| Includes: | Show additional Info: System will display additional information into the system. |
| Assumptions: | Any user can access the system at any time from anywhere. |

Use case specification Table:

|  |  |
| --- | --- |
| Usecase ID | 1 |
| Usecase Name | Login |
| Actor | Customer, driver and admin |
| Pre-condition | App should be downloaded and installed |
| Post-condition | User will land on home screen |
| Flow of events | Actor puts in login detail and logs in or he can sign up . |

|  |  |
| --- | --- |
| Usecase ID | 2 |
| Usecase Name | Booking |
| Actor | Customer, driver |
| Pre-condition | Actor must be logged in |
| Post-condition | Booking confirmed or view bookings |
| Flow of events | Customer can book driver by providing relative information or driver can view his bookings. |

|  |  |
| --- | --- |
| Usecase ID | 3 |
| Usecase Name | Cancel management |
| Actor | Customer, driver and admin |
| Pre-condition | Booking should be done. |
| Post-condition | Booking will be cancelled or new driver will be allotted |
| Flow of events | Customer will cancel booking and booking will be cancelled and if driver cancels booking then he would have to provide reason to admin then new driver will be allotted to the booking |

|  |  |
| --- | --- |
| Usecase ID | 4 |
| Usecase Name | Rating |
| Actor | Customer, driver |
| Pre-condition | Booking should be completed |
| Post-condition | Customer and drivers rating will be generated. |
| Flow of events | After booking customer and driver gets prompt to rate each other |

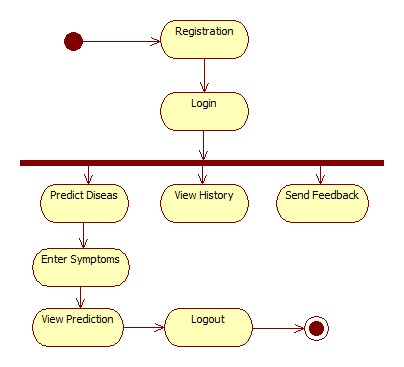
|  |  |
| --- | --- |
| Usecase ID | 5 |
| Usecase Name | Help and support |
| Actor | Customer, driver |
| Pre-condition | They should be logged in |
| Post-condition | Help will be received or query will be sent to admin |
| Flow of events | Actor can view FAQs or request for help |

|  |  |
| --- | --- |
| Usecase ID | 6 |
| Usecase Name | Data management |
| Actor | Admin |
| Pre-condition | Actor must be logged in |
| Post-condition | Changes to the data can be made |
| Flow of events | Admin can view update or delete data. |

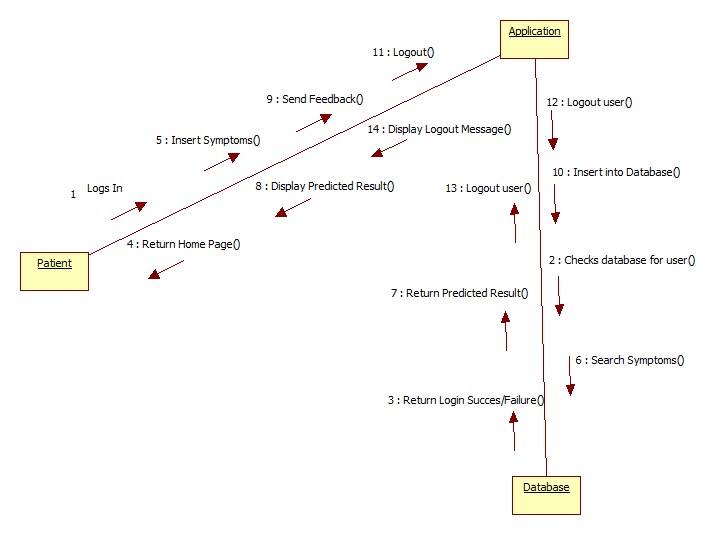
|  |  |
| --- | --- |
| Usecase ID | 7 |
| Usecase Name | Data verification |
| Actor | Admin |
| Pre-condition | Driver should have applied for providing service |
| Post-condition | Driver will be verified and available to customer |
| Flow of events | Admin will verify the documents of driver which he uploaded while signing up . |

* 1. **Architectural Design**

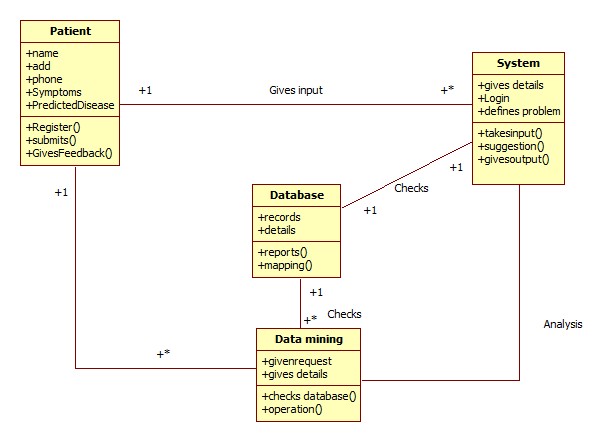
**Activity Diagram:**



**Collaboration Diagram:**

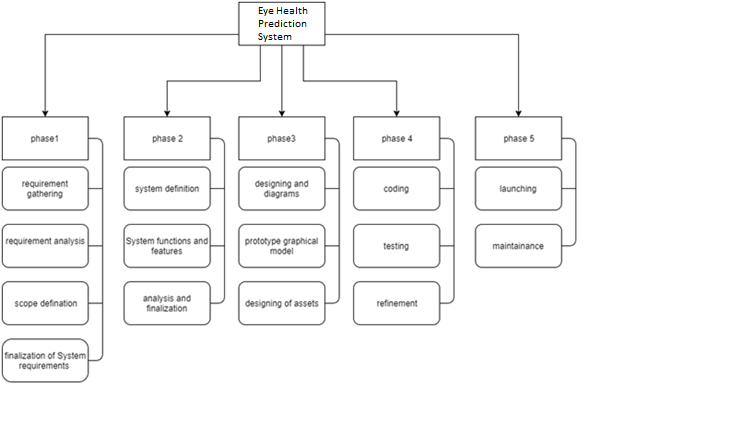


**c. Class diagram**



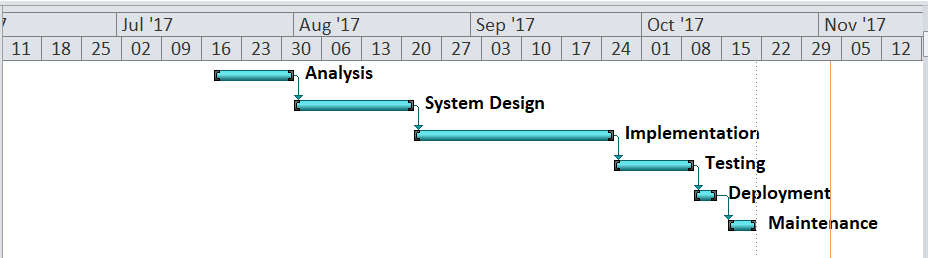
1. **Project Implementation & Testing**

**5.1 Work Break Down Structure**



**5.2 Gantt Chart**

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time.

****

* 1. **Code with reference to design**

static String predictHealth(Symp\_Disease s\_d, String comment) throws Exception {

try {

String filePath = "C:\\Users\\GHANSHYAM\\Desktop\\Project\\Mini\\HealthPrediction Dataset.arff";

BufferedReader br = new BufferedReader(new FileReader(filePath));

Instances trainingData = new Instances(br);

trainingData.setClassIndex(trainingData.numAttributes() - 1);

br.close();

Attribute attr1 = new Attribute(trainingData.attribute(0).name(), s\_d.symptoms);

Attribute attr2 = new Attribute(trainingData.attribute(1).name(), s\_d.diseases);

ArrayList<Attribute> attributes = new ArrayList<>(trainingData.numAttributes());

attributes.add(attr1);

attributes.add(attr2);

Instances inst = new Instances("Test", attributes, 1);

inst.setClassIndex(inst.numAttributes() - 1);

DenseInstance dInst = new DenseInstance(inst.numAttributes());

dInst.setValue(attr1, s\_d.symptoms.indexOf(comment));

inst.add(dInst);

StringToWordVector stwv = new StringToWordVector();

stwv.setInputFormat(trainingData);

stwv.setIDFTransform(true);

stwv.setTFTransform(true);

stwv.setLowerCaseTokens(true);

stwv.setOutputWordCounts(true);

FilteredClassifier fc = new FilteredClassifier();

fc.setFilter(stwv);

fc.setClassifier(new SMO());

fc.buildClassifier(trainingData);

double prediction = fc.classifyInstance(inst.instance(0));

return inst.classAttribute().value((int) prediction);

} catch (Exception ex) {

// System.out.print("Error Occurred: " + ex.toString());

throw ex;

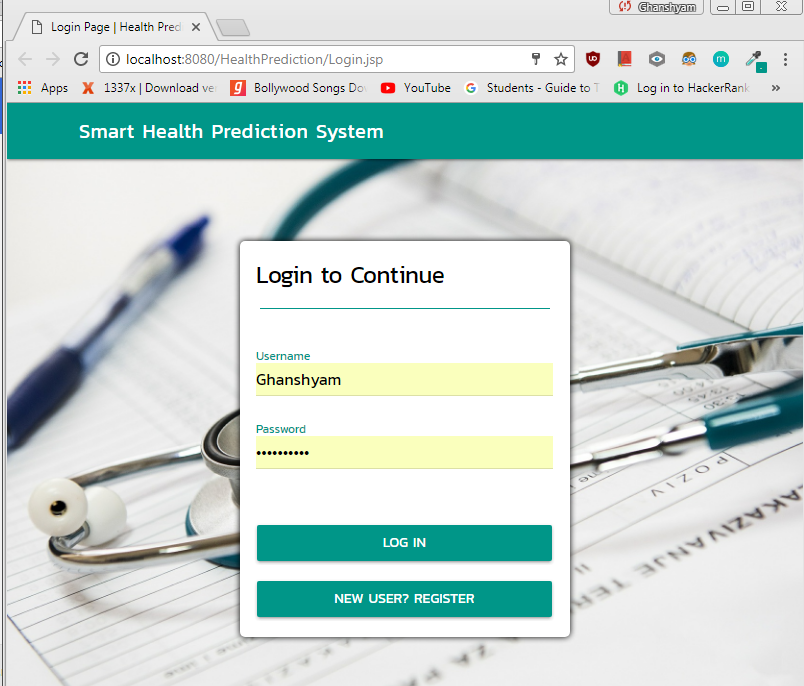
//JOptionPane.showMessageDialog(this, ex.getMessage(), ex.getClass().toString(), JOptionPane.ERROR\_MESSAGE);

}

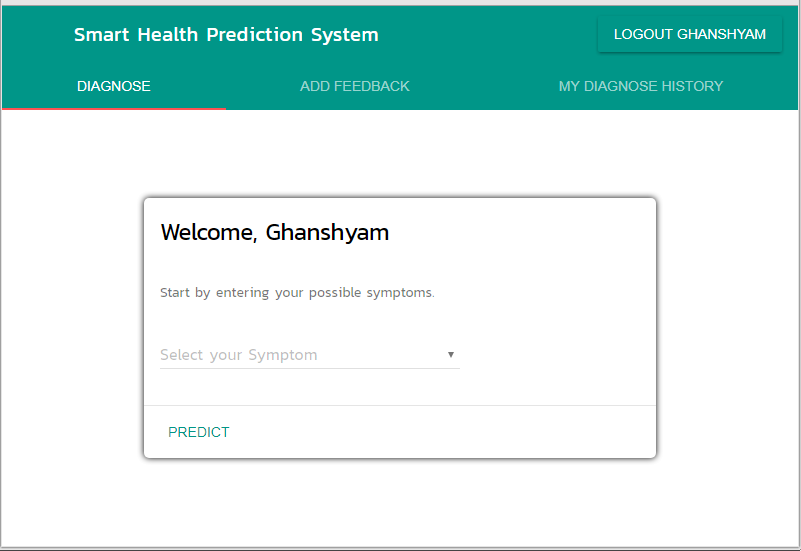
}

* 1. **Snapshot of UI & Reports**

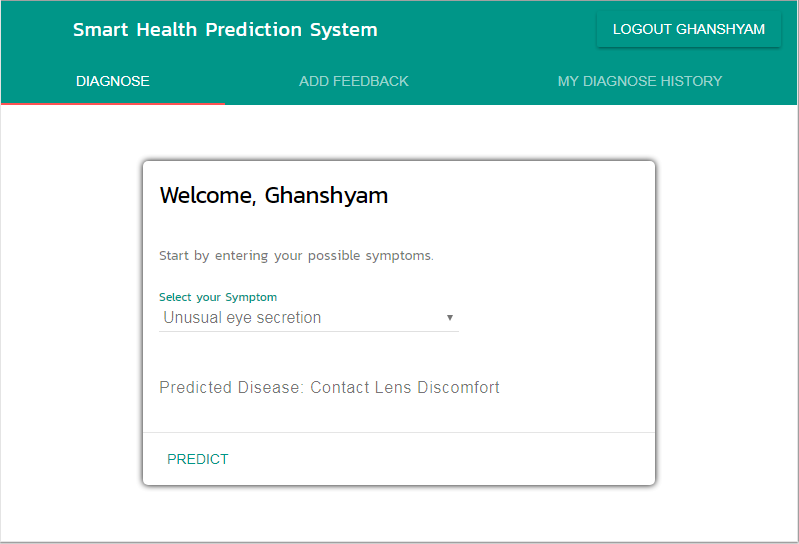
**Login Page**



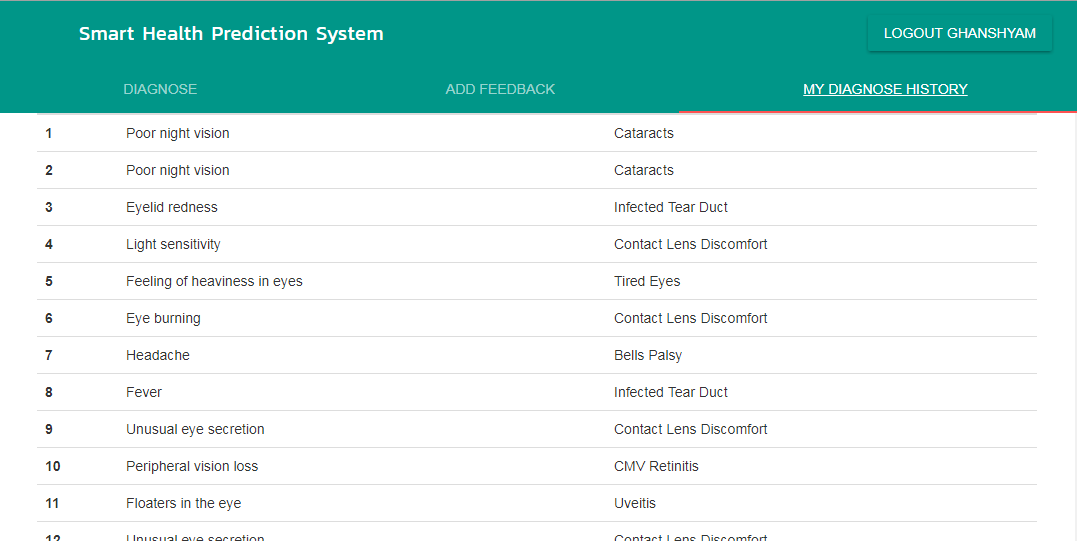
**Home Page**



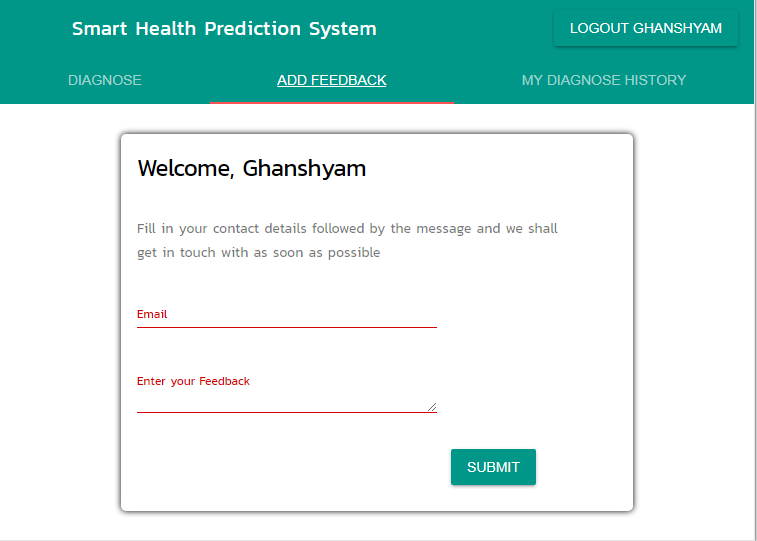
**Prediction**



**View History**



**Send Feedback:**



**5.8 Test cases & Report**

**5.8.1 Types of Testing**

**5.8.1.1 White Box Testing**

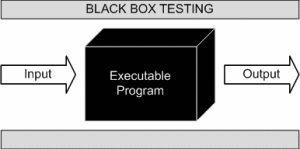
**White-box testing** (also known as **clear box testing**, **glass box testing**, **transparent box testing**, and **structural testing**) is a method of testing [software](https://en.wikipedia.org/wiki/Software) that tests internal structures or workings of an application, as opposed to its functionality (i.e. [black-box testing](https://en.wikipedia.org/wiki/Black-box_testing)). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. [in-circuit testing](https://en.wikipedia.org/wiki/In-circuit_test) (ICT). White-box testing can be applied at the [unit](https://en.wikipedia.org/wiki/Unit_testing), [integration](https://en.wikipedia.org/wiki/Integration_testing) and [system](https://en.wikipedia.org/wiki/System_testing) levels of the [software testing](https://en.wikipedia.org/wiki/Software_testing) process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

White-box test design techniques include the following [code coverage](https://en.wikipedia.org/wiki/Code_coverage) criteria:

* [Control flow](https://en.wikipedia.org/wiki/Control_flow) testing
* Data flow testing
* Branch testing
* Statement coverage
* Decision coverage
* [Modified condition/decision coverage](https://en.wikipedia.org/wiki/Modified_condition/decision_coverage)
* Prime path testing
* Path testing

**5.8.1.2 Black Box Testing**

**Black Box Testing**, also known as Behavioral Testing, is a method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.



This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors
* Initialization and termination errors

**5.8.2 Test Cases & Report:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr. No | To be test | Specification says | | Actual Result | | Status | |
| 1 | Availability of necessary software and network conditions | 1. Intenet connected Browser | | All requirement available | | Pass | |
| 2 | Configuration Testing | 64 bit operating system   1. Minimum 1 GB RAM 2. Minimum 20MB storage space. | | All requirement available | | Pass | |
| 3 | Compatibility | 1. Bootstrap and MDL Components | | All software’s are compatible | | Pass | |
| 4 | Security | 1. Password must not be readable | | Hashing is implemented to hide password. | | Pass | |
|  |  | | 1. Home page must not be visible after logout | | Session is included to provide this feature. | | Pass | |
| 5 | Error Messages | | 1. Appropriate error messages must be displayed on screen. | | Error messages are appropriately handled. | | Pass | |

**6. Documentation & Installation**

* 1. User Manual with Installation procedure

Since it is a web based application, no installation required. Only a web browser with an internet connection is required.

In order to use the system, we first need to login the system by creating our username and password.

A homepage will be displayed which will include 3 sections:

* Prediction: User needs to enter his/her symptoms and based on that the system will predict the disease.
* Send Feedback: User can send his feedback to the admin.
* View History: User can view his past prediction history.

1. **Future Enhancements**

* We plan to expand this system to the entire body of a human being.
* Improve prediction algorithm by extracting more data of health diseases.

1. **Limitations**

* Only limited to eye related diseases.

1. **Conclusion**

The system will greatly help people in need of medical emergency or in case of absence of doctors.

1. **Bibliography**

https://www.ijirset.com/upload/2016/august/32\_A%20Survey.pdf

http://platform.almanhal.com/GoogleScholar/Details/?ID=2-38086

https://pdfs.semanticscholar.org/fbd6/5a18f6653b56138cd5196d20e2f39de189e3.pdf

http://issc2016.system-safety.org/P02\_Zenga\_HTS\_SoS.pdf

https://en.wikipedia.org/wiki/Data\_mining