Final Project Report

for

Prediction of various diseases

Prepared by

Doyel (20BCE2639)
Soumya Jha(20BCE2547)
Mansi (20BCE2604)
Abhay Rathi (20BCE2905)

(3rd yr. B.Tech CSE , VIT Vellore)
17th November 2022

Submitted to

Ushus Elizebeth Zachariah
(Assistant Professor Sr. Grade 1)
(SCOPE, VIT Vellore)

Table of Contents

1. Introduction

1.1 Purpose

After the spread of covid-19 , other diseases and ailments have been ignored. Earlier people used to get their Blood pressure and other common diseases checked regularly. But to reduce the burden on hospital staff , doctors insisted patients to not report to hospital for small things like blood tests and other unnecessary check-ups . Therefore, our prediction model can help people to predict diseases , which in turn can help patients to decide if or not to see a doctor or not. Our system reduces unnecessary pressure on hospitals and also eliminates panic situations from users point of view.

1.2 Document Conventions

The document is prepared using Google Docs and has used the font type 'Times New Roman'. The fixed font size that has been used to type this document is 12pt with 1.5 line spacing. It has used the bold property to set the headings of the document. All pages except the cover page are numbered, the numbers appear on the lower right hand corner of the page. Every image and data table are numbered and referred to in the main text Use case scenario is written according to Alistair Cockburn's template. Standard IEEE template is the template used to organise the appearance of the document and its flow.

1.3 Intended Audience and Reading Suggestions

The Covid-19 pandemic had an adverse effect on Indian Healthcare infrastructure. As India has a huge population many people were not able to have access to medical facilities owing to the lockdown. In such a situation the need of a remote platform arose which would provide easy medical access to people who need it the most. Hence our website was created to solve the above problem. Our website is a web based platform created to bring all the critical medical related solutions under a single umbrella. This platform will enable users to Predict disease using machine learning models based on the user data for heart , liver , kidney , cancer and diabetes.

1.4 Product Scope

Our main objective is to build a platform independent disease predictor which is easy to use , helps patients to predict risk of various diseases on the same site. Our system provides a login page which ensures security of users' data. Our platform will also assist hospital administration by enabling them to share important data to other administrations.
This project involves a platform independent disease predictor which is easy to use, helps patients to predict risk of various diseases on the same site. Our system also provides a login page which ensures security of user's data. The website consists of different disease predictors in different pages. For now, the website consists of Cancer prediction, diabetes prediction, heart disease prediction, kidney disease prediction, liver disease prediction. It

also has a page containing the disease index which presents in-depth information about each of the diseases. Also after the user inputs their data, they can press submit and the website will generate a PDF that provides the results of the risk assessment.

2. Literary Survey

a) Disease Prediction From Various Symptoms Using Machine Learning Rinkal Keniya , Aman Khakharia , Vruddhi Shah , Vrushabh Gada, Ruchi Manjalkar, Tirth Thaker, Mahesh Warang, Ninad Mehendale. Accurate and on-time analysis of any health-related problem is important for the prevention and treatment of the illness. The traditional way of diagnosis may not be sufficient in the case of a serious ailment. Developing a medical diagnosis system based on machine learning (ML) algorithms for prediction of any disease can help in a more accurate diagnosis than the conventional method. We have designed a disease prediction system using multiple ML algorithms. The data set used had more than 230 diseases for processing. Based on the symptoms, age, and gender of an individual, the diagnosis system gives the output as the disease that the individual might be suffering from. The weighted KNN algorithm gave the best results as compared to the other algorithms. The accuracy of the weighted KNN algorithm for the prediction was 93.5 %. Our diagnosis model can act as a doctor for the early diagnosis of a disease to ensure the treatment can take place on time and lives can be saved.

b) A Survey on Disease Prediction by Machine Learning over Big Data from Healthcare Communities K.DeepthiKrishnan 1 , B.Senthil Kumar2 M.Phil Scholar, Department of Computer Science, Sree Narayana Guru College, K.G.Chavadi, Coimbatore, Tamil Nadu, India 1 Associate Professor, Department of Computer Science, Sree Narayana Guru College, Coimbatore, Tamil Nadu, India 2 Corresponding Author: K.DeepthiKrishnan. The Enlarged and enhanced concept of big data is extracted in the medical field and the new concept is introduced in the paper concept. The survey concept is take the machine learning based disease prediction from medical field and uses the big data concept, which means the machine learning is a data mining techniques but this technique applied in disease prediction to come some difficulty such as, incomplete data, not suitable in large or big hospital and the some results are inaccurate, so this some difficulty are come in the existing, then, it will be move into the next level called the "Big Data". Big data is a large and huge data sets that holds, so this difficulty is overcome. The paper concept is machine learning based disease prediction over big data. Big data directly collects information in Healthcare communities, because big data is like, very knowledgeable concept. The proposed system aim is to (i) Analysis the optimal and accurate results on medical data, (iii) fast disease prediction (early predicts) from analysed the data for medical field (hospital medical data), (ii) take the *incomplete data and it move into the next level called "complete data" for disease analysis. The concept introduces the techniques namely, Multimodal Disease Risk Prediction (CNN-MDRP) based on Convolutional Neural Network. The paper additionally, describes the term Unimodal Disease Risk Prediction (UDRP) and it compares and analyzes the performance.*

c) A LITERATURE SURVEY OF PREDICTING HEART DISEASE M. Preethi1, Dr. J. Selvakumar 2 1M.Tech, Dept. of Computer Science and Engineering, Sri Ramakrishna Engineering College, Coimbatore-TamilNadu, India. 2Professor, Dept. of Computer Science and Engineering, Sri Ramakrishna Engineering College, Coimbatore-TamilNadu, India. This paper describes various methods of data mining, big data and machine learning models for predicting heart disease. Data mining and machine learning plays an important role in building an important model for the medical system to predict heart disease or cardiovascular disease. Medical experts can help the patients by detecting the cardiovascular disease before occurring. Now-a-days heart disease is one of the most significant causes of fatality. The prediction of heart disease is a critical challenge in the clinical area. But from time to time, several techniques are discovered to predict heart disease in data mining. In this survey paper, many techniques were described for predicting heart disease.

*d) DISEASE PREDICTION USING MACHINE LEARNING Anjali Bhatt*1, Shruti Singasane*2, Neha Chaube*3 *1,2Student, Dept. Of Computer Science Engineering, MIT-ADT University, Loni Kalbhor Maharashtra, India. *assistant Professor, Dept. Of Computer Science Engineering, MIT-ADT University, Loni Kalbhor Maharashtra, India. ABSTRACT This project is an attempt to help one to predict the disease he/she is having through the symptoms and the correct readings of the bodily vitals needed. There are times when people keep on ignoring health issues due to high medical fees. This may lead to severe issues later and even death. If not covered by insurance, medical bills can be a menace. This website is an approach in reducing the effort of a normal person by estimating the kind of disease one has and its severity. We have designed a disease prediction system using multiple machine learning algorithms. Based on the symptoms, age, and gender of an individual, the diagnosis system gives the output giving the information about whether the user is suffering from that particular disease or not. According to the severity, some diet plans and some exercises which can minimise the effects of the disease to some extent are also provided. It provides a simple yet effective approach for predicting the disease, if the provided values of vitals are accurate. The user will experience a simple yet effective User Interface and pleasing design. Keywords: Disease Prediction, Machine Learning Algorithm, Supervised Learning, Diagnosis Syst*

3. Proposed System Requirements Analysis,Users and Design

3.1 Product Introduction

According to the World Health Organization, every year 15 million deaths occur worldwide due to Heart Disease, liver ailments, breast cancer etc. Heart disease is one of the biggest causes of morbidity and mortality among the population of the world. Prediction of cardiovascular disease is regarded as one of the most important subjects in the section of data analysis. The load of cardiovascular disease is rapidly increasing all over the world from the past few years. Many researches have been conducted in an attempt to pinpoint the most influential factors of heart disease as well as accurately predict the overall risk. Heart Disease is even highlighted as a silent killer which leads to the death of the person without obvious symptoms. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduces the complications. Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry. This project aims to predict future Heart Disease by analysing data of patients which classifies whether they have heart disease or not using a machine-learning algorithm. Machine Learning techniques can be a boon in this regard. Even though heart disease can occur in different forms, there is a common set of core risk factors that influence whether someone will ultimately be at risk for heart disease or not. By collecting the data from various sources, classifying them under suitable headings & finally analysing to extract the desired data we can say that this technique can be very well adapted to do the prediction of heart disease.

3.2 Product Functions

The scope of this project is that our prediction model can help people to predict diseases , which in turn can help patients to decide if or not to see a doctor or not. Our system reduces unnecessary pressure on hospitals and also eliminates panic situations from users point of view. Our system must be able to predict the presence of disease (high risk) with 92% or greater accuracy. The project must have a user-friendly interface which is easy to navigate and displays results clearly. The work is divided in a way where all the teammates have equal opportunity to explore machine learning and front-end technologies in accordance with software engineering principles.

3.3 User Classes and Characteristics

As an example of classes of users, consider a computer-based catalog system for a library. We may decide that the users fall into four classes. Each class of users has particular goals and requirements for the system, as listed below. the specified

1. ***Novices***
    - ***Characteristics***
        - ***computer literate***
        - ***knowledgeable of library's system of filing materials***
        - ***familiar with the English language***
        - ***use system once per semester***
    - ***Goals/requirements***
        - ***Easy to learn - minimize amount of time for an individual to learn how to navigate through the system to achieving basic tasks.***
        - ***Easy to understand - display search results in a way that is easy to comprehend without cryptic reresentations of information.***
        - ***Minimum user options***
2. ***Average users***
    - ***Characteristics***
        - ***computer literate***
        - ***familiar with computer-based library systems***
        - ***use system 2-4 times/semester***
    - ***Goals/requirements***
        - ***High Retention Rate - quick relearning***
3. ***Frequent users***
    - ***Characteristics***
        - ***"power users"***
        - ***more than five times per semester***
    - ***Goals/requirements***
        - ***Usability without distractions***
        - ***Advanced user shortcuts***
        - ***Access to research tools, such as abstracts and indices of journals.***
        - ***Access to references not currently present in the specific library that they are using.***
4. ***Librarians***
    - ***Characteristics***
        - ***experts in the services and operations of the library.***
        - ***resource used by the other classes of users.***

- ***responsible for the maintenance of the contents/materials of the library.***

3.4 Operating Environment

Additionally, the system is made to be user-friendly. The programme will function along with the hardware platform, operating system, and versions, as well as any additional software parts or apps that it needs to get along with.

Software requirements

- *Windows 7 or above operating system*
- *MongoDB*

Hardware Requirements

- *Core i5 processor*
- *4GB Ram*
- *10GB of hard disk space in terminal machines*
- *20GB hard disk space in Server Machine*

3.5 Design and Implementation Constraints

1. Be prepared for challenges and constraints when it comes to system updates and changes owing to the coordination needed to shut down clinical systems that need to continue operating.

2. Have the capacity to manage a sizable number of transactions simultaneously.

3. Enable a high volume of concurrent electronic transactions because many healthcare experts might need to enter or edit information.

4. Always keep a record of all transactions so that you can comprehend what took place, replay events, identify flaws, and guarantee the accuracy of data.

5. Always confirm the accuracy of the data, especially when consulting simultaneously.

6. Whenever possible, even during concurrent consultation, make information available.

7. Regardless of how much information is searched for across numerous databases, provide a quick data display.

3.6 User Documentation

As a part of the system itself a user documentation is provided to the customers which gives an overview of the system. It will include the full description about the product and complete orderly followed steps to install the software. The users will get the opportunity to use the system without having any trouble. The user manual will include the email addresses to contact us in need. Tasks are listed alphabetically or logically grouped often using cross referenced indexes which helps the users to know exactly what sort of information they are looking for.

3.7 Assumptions and Dependencies

- *Each user must have a valid user id and password*
- *Server must be running for the system to function*
- *Users must log in to the system to access any record.*
- *Only the Administrator can delete records.*

3.8 External Interface Requirements

3.8.1 User Interfaces

Different user kinds will have access to this programme, hence each of them will require a different user interface.
The following user groups are eligible to utilise this application:

• Common internet surfers who access our website by typing "disease prediction" into Google. They will be limited to using the public interface only.

• User will have a username and password to access the private interface where they can generate their result as a certificate and predict a disease.

3.8.2 Hardware Interfaces

- *Core i5 processor*
- *4GB Ram*
- *10GB of hard disk space in terminal machines*
- *20GB hard disk space in Server Machine*

3.8.3 Software Interfaces

*The software is developed for android, ios, windows 7, windows 8, windows 10, etc. Everything need not be created from scratch. For some tasks, you can save a tonne of time and money by using pre-made components (**SDKs, libraries, frameworks, etc.**).*

Having said that, several technologies have demonstrated their worth.

Mobile: *Java/Kotlin (Android), Swift (iOS), Flutter, React Native, Xamarin (hybrid).*

Web frontend: *JavaScript, React.js, Angular, Vue.js*

Web backend: *Java, Scala, C#, Python, .NET, Go.*

3.9 Communications Interfaces

Our website includes all channels of communication with the patient that leverage Information Technology platforms, including Voice, Audio, Text & Digital Data exchange. Any MedNet tool that is appropriate for carrying out technology-based patient consultations may be used by RMP, including the phone, video, devices connected over LAN, WAN, or the Internet, mobile or landline phones, chat platforms like WhatsApp, Facebook Messenger, etc., mobile apps, or internet-based digital platforms.

3.10 System Features

3.10.1 Easy User Registration

Patients regard patient registration to be simple and trustworthy when only the bare minimum of information is required, along with a solid authentication method.

Patients find it simple to use when the registration information needed is minimal and there are additional tools to upload pictures of other documents.

Patients can choose the doctor they need and schedule an appointment after completing an easy and quick registration process. Instead of typing, patients can upload their medical documents, insurance verification data, and old medications.

3.10.2 Users Dashboard

The users dashboard is a useful tool that helps the user to know about the various diseases and know about all the data required to predict a disease.
The user will be able to view information about various diseases and can also predict various diseases like heart disease.
The user will be able to generate a report and prediction based upon the data entered.

3.10.3 Disease prediction dashboard

Here the user will be able to enter various data like cp1, cp2 etc for a particular disease and then our model will predict the probability the user could have the given disease.

3.10.4 Report generation

After the user enters all the data in valid format and within valid range the system will generate a report for the user. The report can be downloaded as a pdf.

3.10.5 App Scalability

Any MedNet app must provide the ability for patients to rate and review their doctors. Patient feedback is seen as an essential element of quality improvement and career advancement. Additionally, it will assist a lot of new patients in locating the expert physician who best suits their needs.

To help the healthcare system, the scaling characteristics should be designed so that only the patient who receives the therapy can rate the physician.

3.11 Summary of Functional Requirements

There are a lot of software requirements included in the functional requirements for our project, which contains various processes, namely Registration, Report Generation, and Database.

➢ Registration process:

● Adding Users: The software enables the stakeholders to include new users to the system.

➢ Database requirements:

● Mandatory User Information: Every user has some necessary data like name, phone number, and current health status data.
● Updating users disease data points: The software system takes various data of the user for a particular disease and those data are updated each time in the database

➢ Report Generation

● Information of the User: The system application generates a report on every user regarding various information like users name, disease he/she was trying to predict and final result in the form of certificate.

3.12 Other Nonfunctional Requirements

3.12.1 Performance Requirements

● Response Time: The system provides acknowledgement in just one second once the patient's information is checked.
● Capacity: The system needs to support at least 1000 people at once.
● User-Interface: The user interface acknowledges within five seconds.
● Conformity: The system needs to ensure that the guidelines of the Microsoft accessibilities are followed.

3.12.2 Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed up log, up to the time of failure.

3.12.3 Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

3.12.3.1Preprocessing

The dataset preprocessing is the training system's primary process, which is vital in all processes since it directly affects the system's success rate. Since the real-world data are unclean, this decreases the data's complexity under examination to initially execute the preprocessing. The dataset can contain duplicate data. This step evades the training of the same information repeatedly for removing the redundant data via the execution of data deduplication.

3.12.3.2*Data Deduplication*

Data deduplication is among the methods that permit cloud users to efficiently administer their cloud storage space by avoiding repeated data storage and saving bandwidth. It evades the repeated training of duplicate data.

3.12.3.3*Disease Prediction System (DPS)*

In the proposed system, the disease prediction system (DPS) is the main process that predicts the chance of a disease's presence in a patient centered on their symptoms. The sensed values should be tested to find if the patient comprises the disease. First, train the disease dataset, before the values are tested. The training system has four phases: data collection, preprocessing, matrix representation, matrix reduction, and classification

3.12.3.4*Verification*

The verification procedure has been carried out after the system is logged in. The system would match this segment's user-id, username, password, and ciphertext.

3.12.3.5*Login*

Login is a credential set wielded for validating a user. Mostly, they comprise the username as well as the password. The login segment lets a user get accessibility to an application via entering their username and password. The patients ought to input the authentication data offered for authentication using the administrator while logging in to the system. The patient should enter the user-id, password, and ciphertext while logging in.

3.12.4 Software Quality Attributes

● **Availability:** *The system is available all the time.Whenever a user wants to register/ log in , they should be able to do so and input their details for the prediction.*

● **Correctness:** *The prediction result should be accurate and concise.*

● **Maintainability:** *The database should contain all the registered users .Errors: The system will track every mistake as well as keep a log of it. Back-Up: The system offers the efficiency for data back up.*

● **Usability:** *The prediction result should satisfy a maximum number of customer needs.*

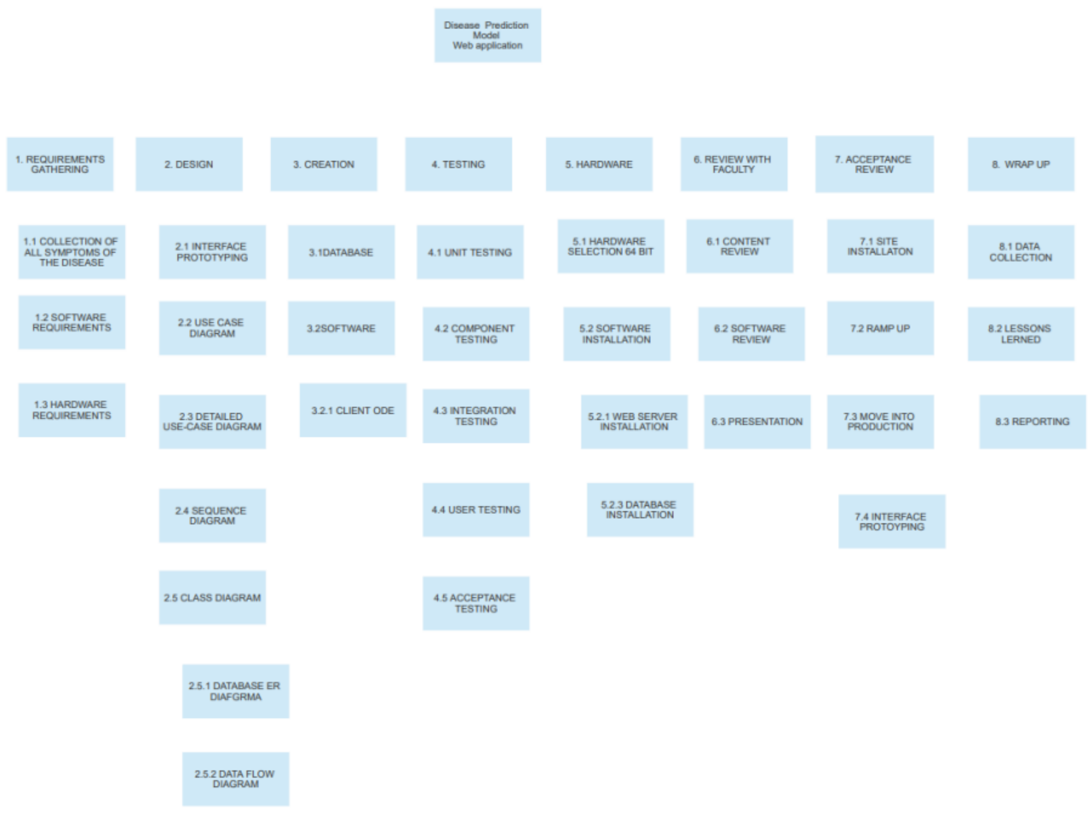
3.12.5 Business Rules

Our platform independent disease predictor which is easy to use , helps patients to predict risk of various diseases on the same site. But before providing such services, it is important to carefully research the state-specific regulations that have been put in place to govern the practice of such websites.. During deployment, doctors should be aware of any potential cybersecurity and legal issues.

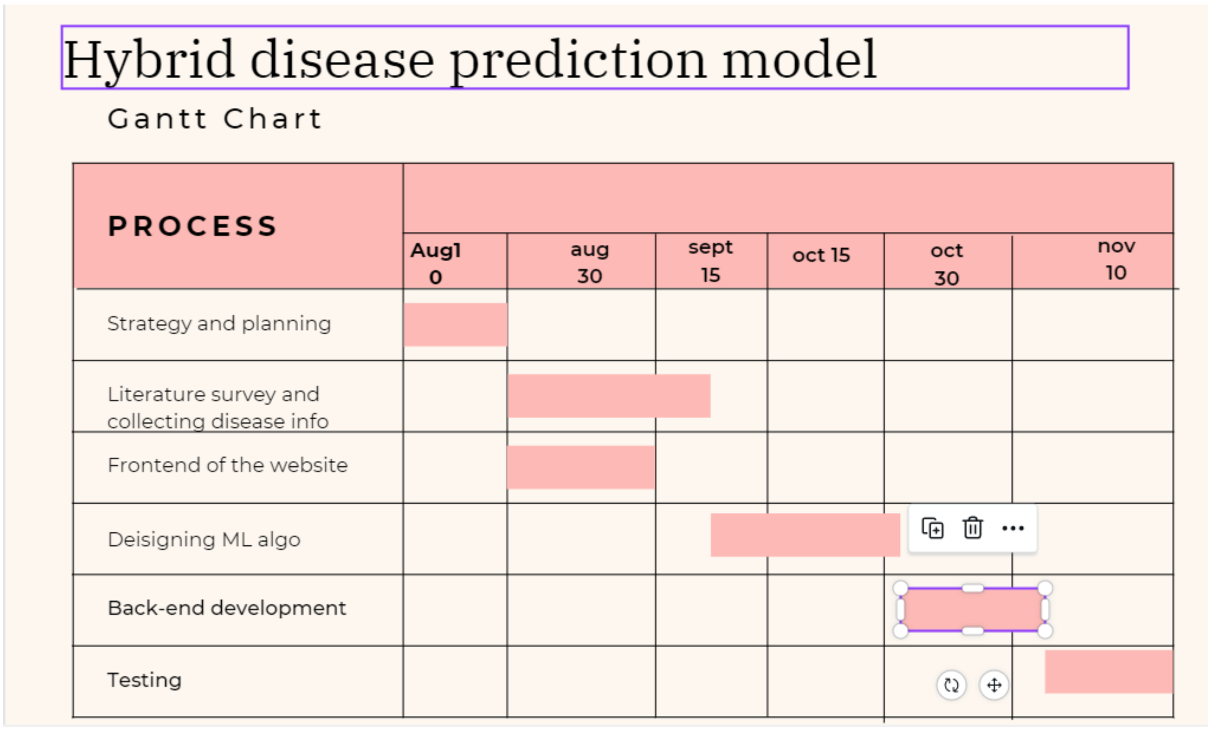
Recommended Actions

1.Review the laws governing the website in your state and the methods for paying for the services every year.
2.In your practice, establish precise policies and procedures for the delivery of services.
3,All employees and clinicians should receive training on your practice's policies, procedures, and utilisation of the selected platform.

3.13 Work Breakdown Structure



3.14 Gantt Chart

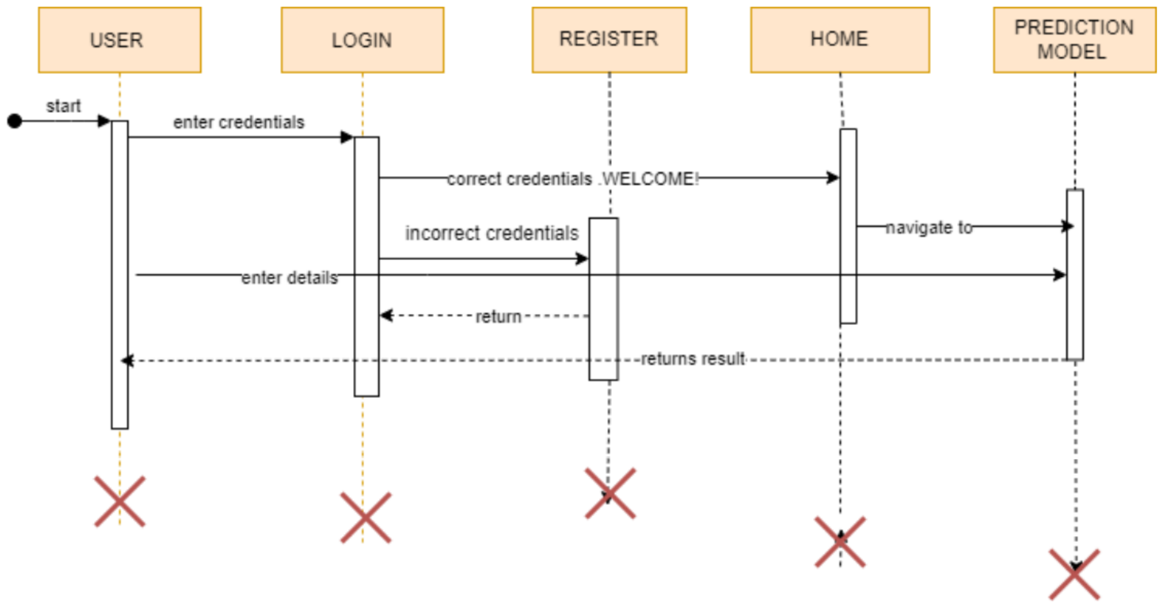


Hybrid disease prediction model

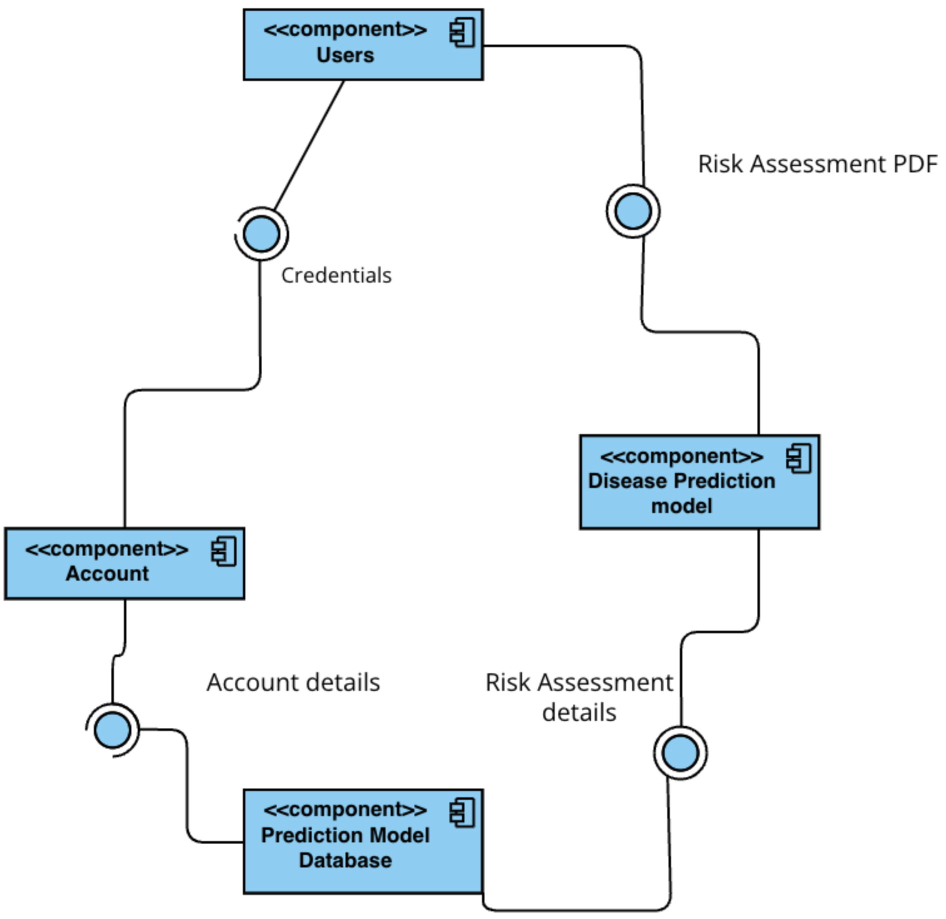
Gantt Chart

PROCESS	Aug1 0	aug 30	sept 15	oct 15	oct 30	nov 10
Strategy and planning	🟥					
Literature survey and collecting disease info		🟥	🟥			
Frontend of the website		🟥				
Deisigning ML algo			🟥	🟥		
Back-end development					🟥	🟥
Testing						🟥

Sequence diagram:



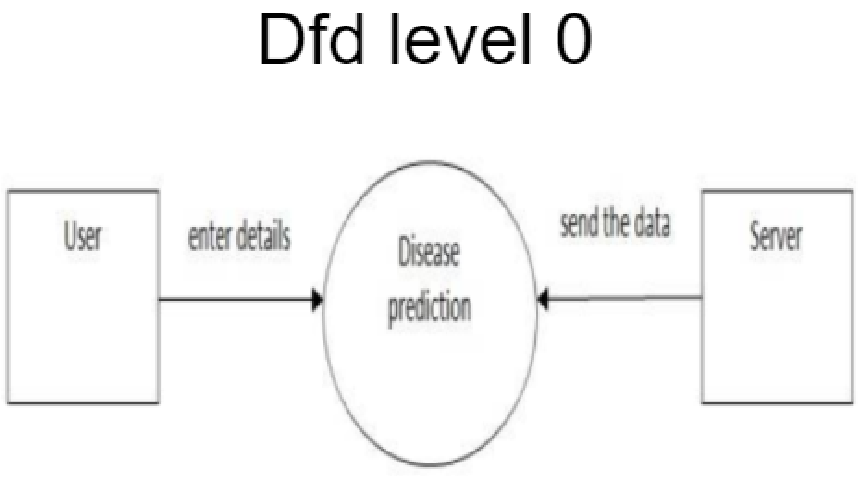
COMPONENT DIAGRAM



4. Design Of Proposed Model

4.1 High level Design

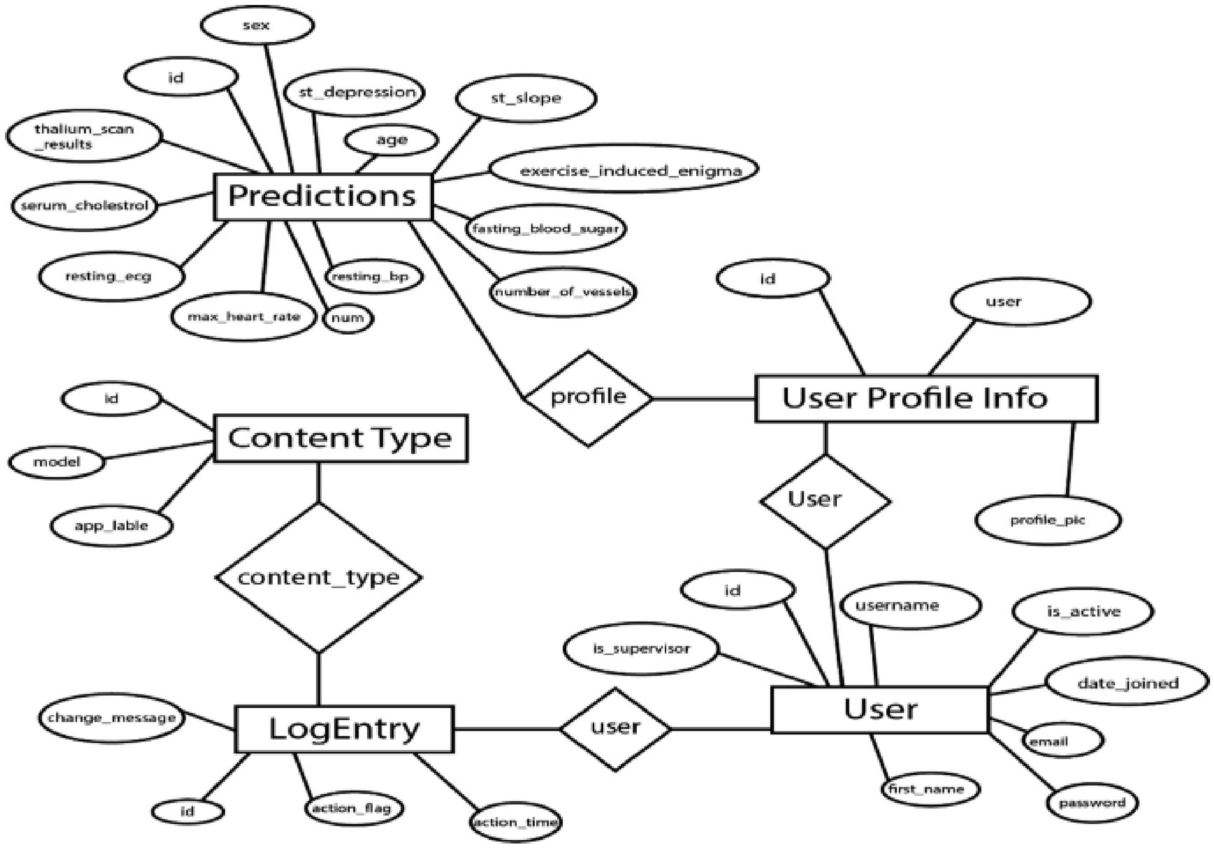
4.1.1 Data Flow Diagram



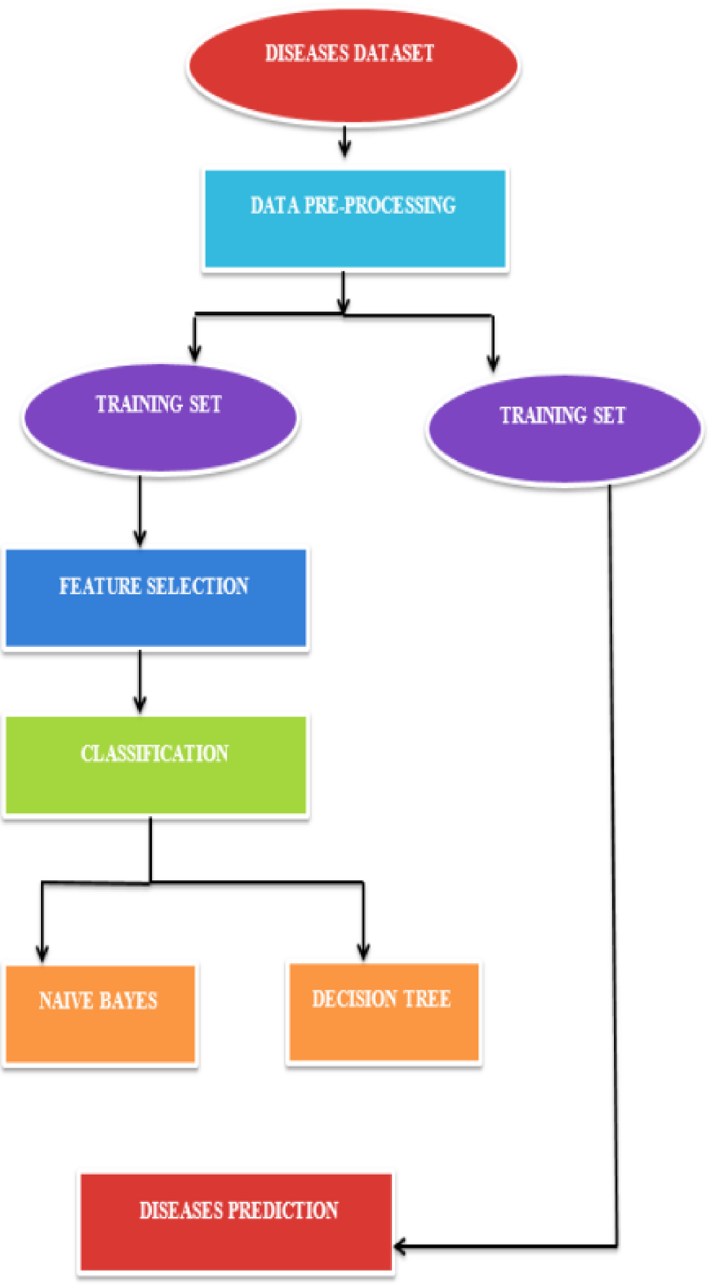
Dfd level 1 where the user's data will be used to produce results

4.2 Detailed Design

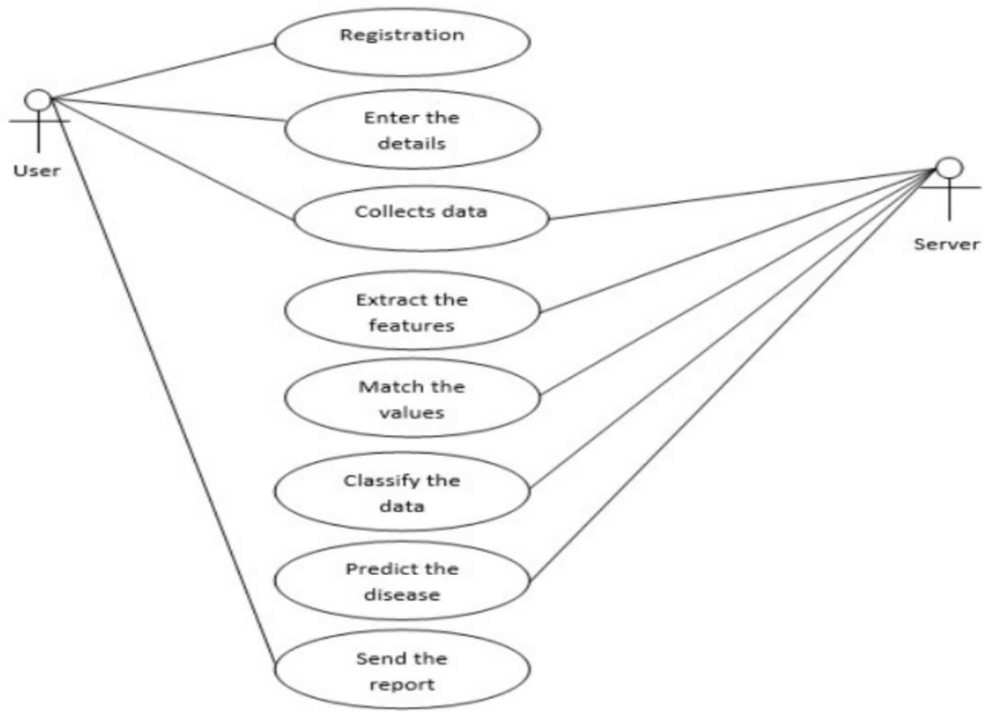
4.2.1 ER DIAGRAM



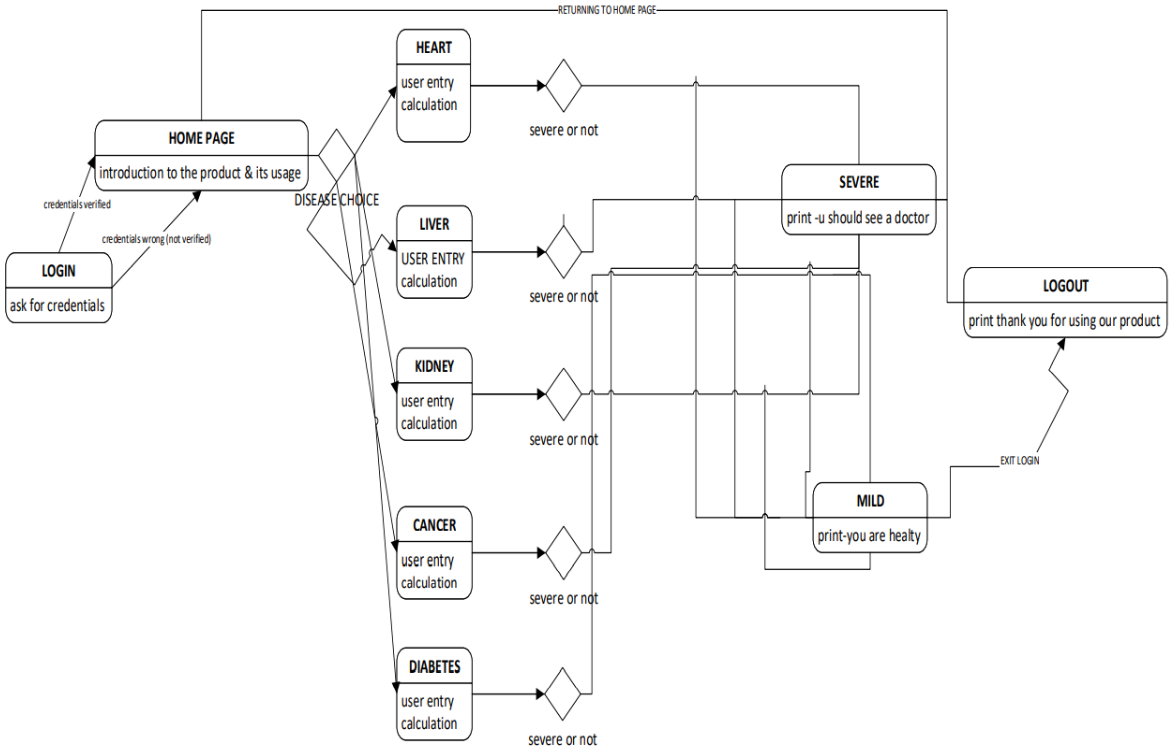
System architecture



4.2.2 UML DIAGRAMS

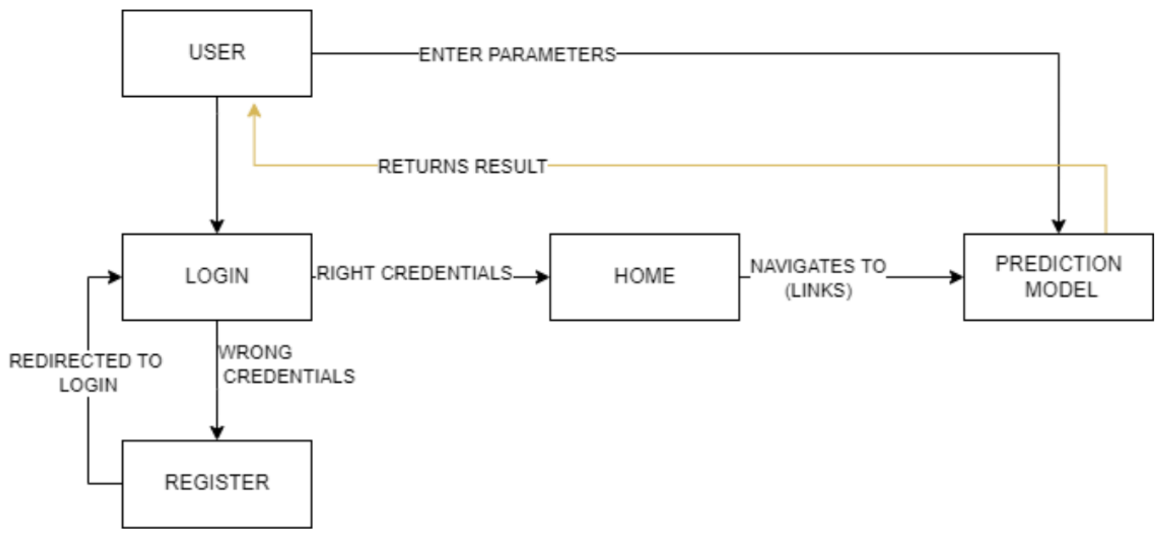


USE CASE DIAGRAM



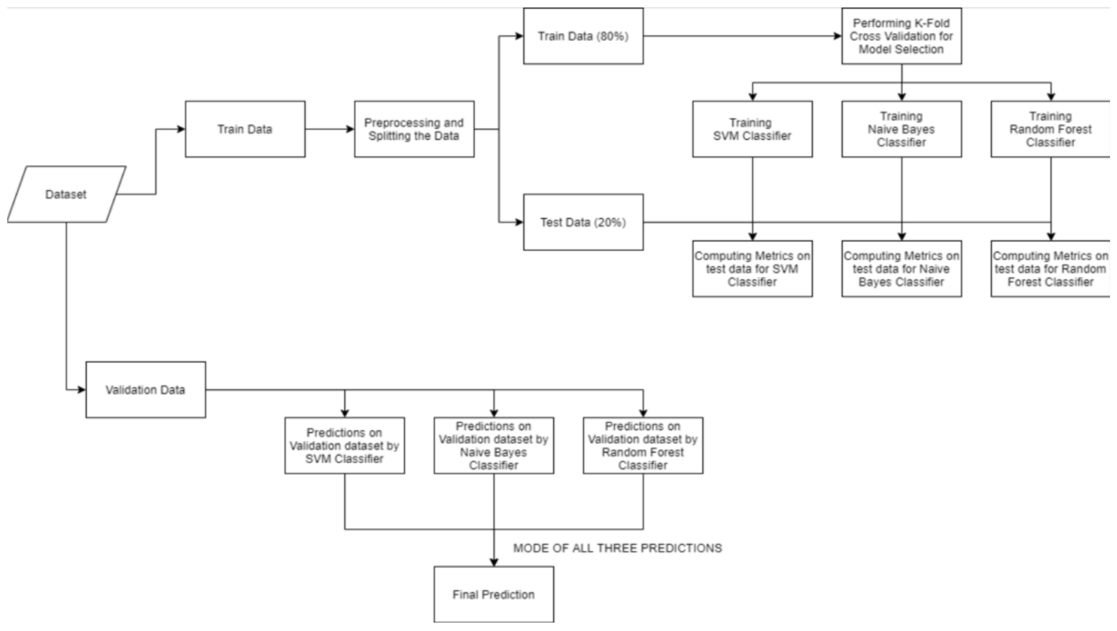
STATE DIAGRAM

ACTIVITY DIAGRAM



Collaboration diagram:

Preprocessing of data ..

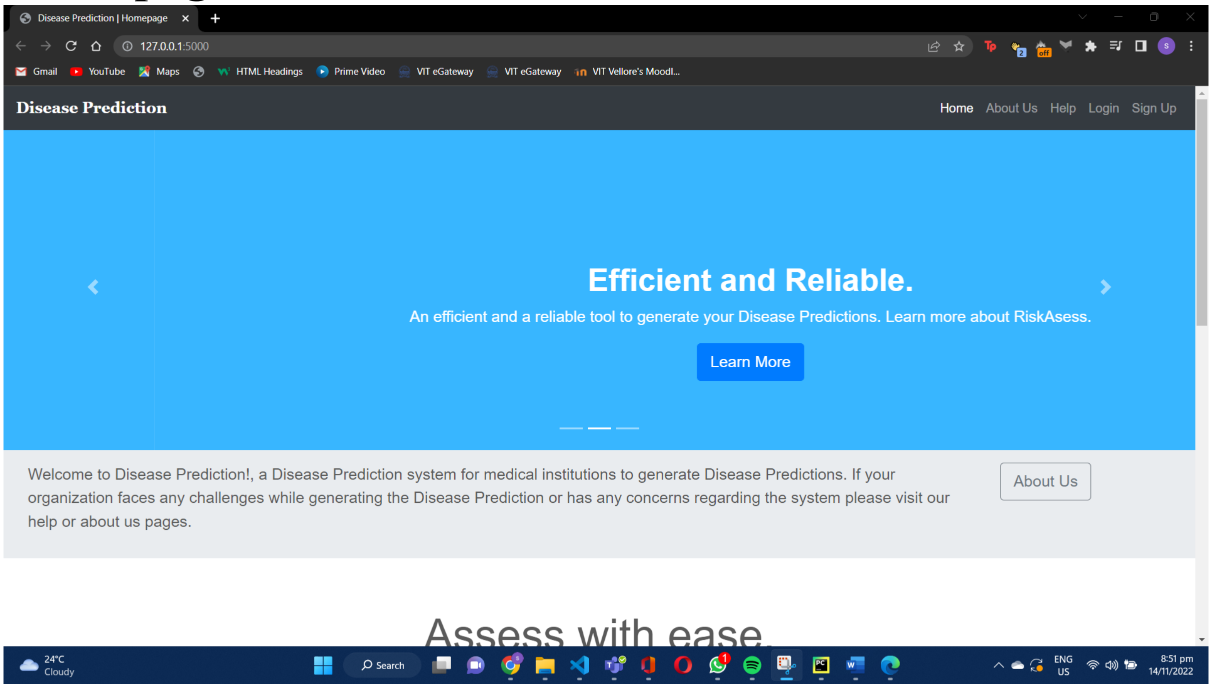

Dataset
Train Data
Preprocessing and Splitting the Data
Train Data (80%)
Performing K-Fold Cross Validation for Model Selection
Training SVM Classifier
Training Naive Bayes Classifier
Training Random Forest Classifier
Test Data (20%)
Computing Metrics on test data for SVM Classifier
Computing Metrics on test data for Naive Bayes Classifier
Computing Metrics on test data for Random Forest Classifier
Validation Data
Predictions on Validation dataset by SVM Classifier
Predictions on Validation dataset by Naive Bayes Classifier
Predictions on Validation dataset by Random Forest Classifier
MODE OF ALL THREE PREDICTIONS
Final Prediction

5. Implementation and Testing.

5.1. Implementation details (snapshots)

Output:

Home page:



Disease Prediction

Home About Us Help Login Sign Up

Efficient and Reliable.

An efficient and a reliable tool to generate your Disease Predictions. Learn more about RiskAsess.

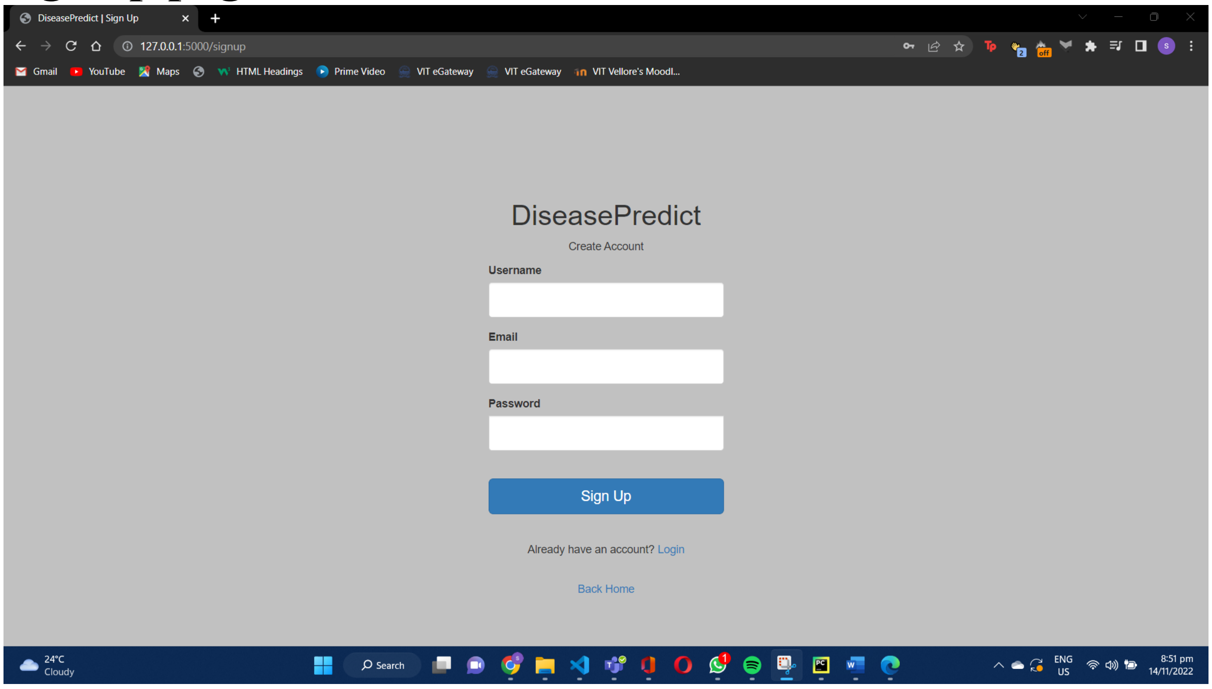
Learn More

Welcome to Disease Prediction!, a Disease Prediction system for medical institutions to generate Disease Predictions. If your organization faces any challenges while generating the Disease Prediction or has any concerns regarding the system please visit our help or about us pages.

About Us

Assess with ease

Sign up page:



DiseasePredict

Create Account

Username

Email

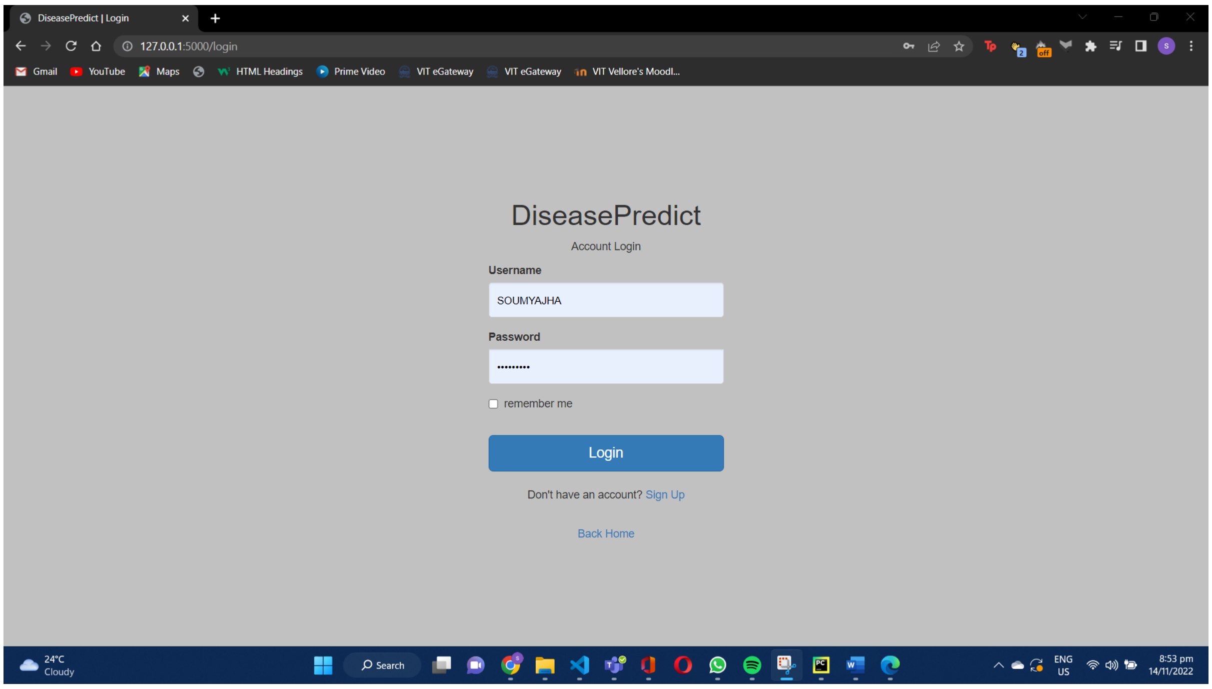
Password

Sign Up

Already have an account? Login

Back Home

Login page:


DiseasePredict

Account Login

Username

SOUMYAJHA

Password

••••••••

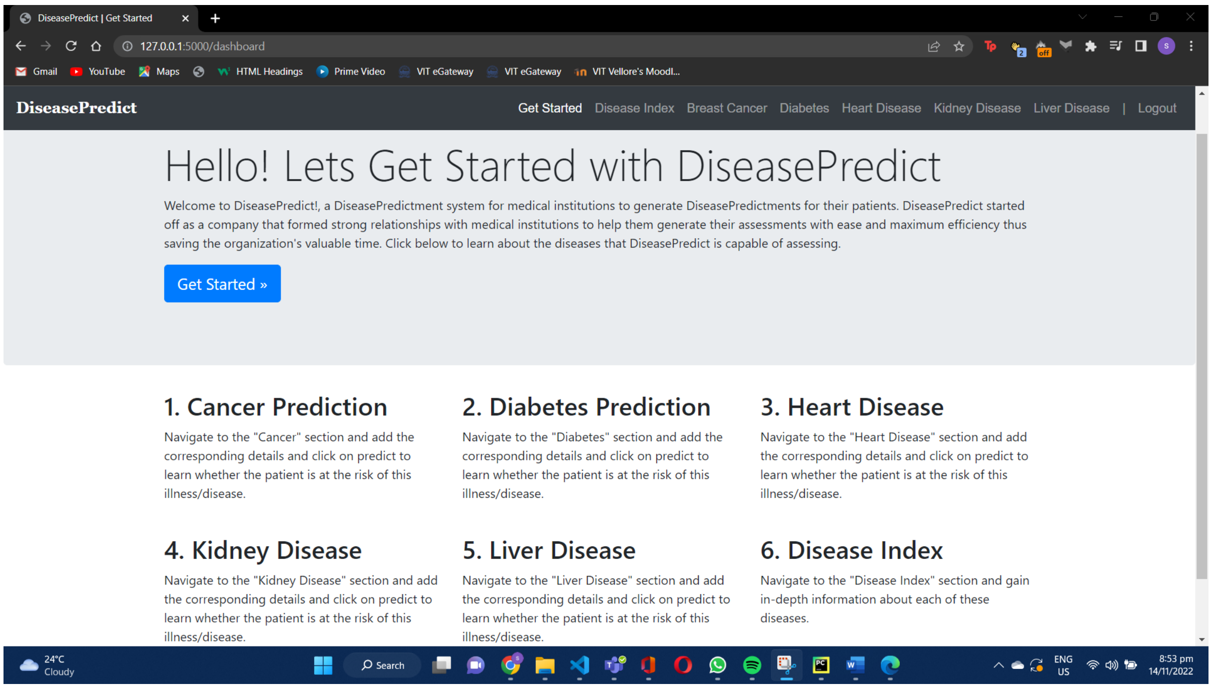
remember me

Login

Don't have an account? Sign Up

Back Home

Home page after login:



Hello! Lets Get Started with DiseasePredict

Welcome to DiseasePredict!, a DiseasePredictment system for medical institutions to generate DiseasePredictments for their patients. DiseasePredict started off as a company that formed strong relationships with medical institutions to help them generate their assessments with ease and maximum efficiency thus saving the organization's valuable time. Click below to learn about the diseases that DiseasePredict is capable of assessing.

Get Started »

1. Cancer Prediction

Navigate to the "Cancer" section and add the corresponding details and click on predict to learn whether the patient is at the risk of this illness/disease.

2. Diabetes Prediction

Navigate to the "Diabetes" section and add the corresponding details and click on predict to learn whether the patient is at the risk of this illness/disease.

3. Heart Disease

Navigate to the "Heart Disease" section and add the corresponding details and click on predict to learn whether the patient is at the risk of this illness/disease.

4. Kidney Disease

Navigate to the "Kidney Disease" section and add the corresponding details and click on predict to learn whether the patient is at the risk of this illness/disease.

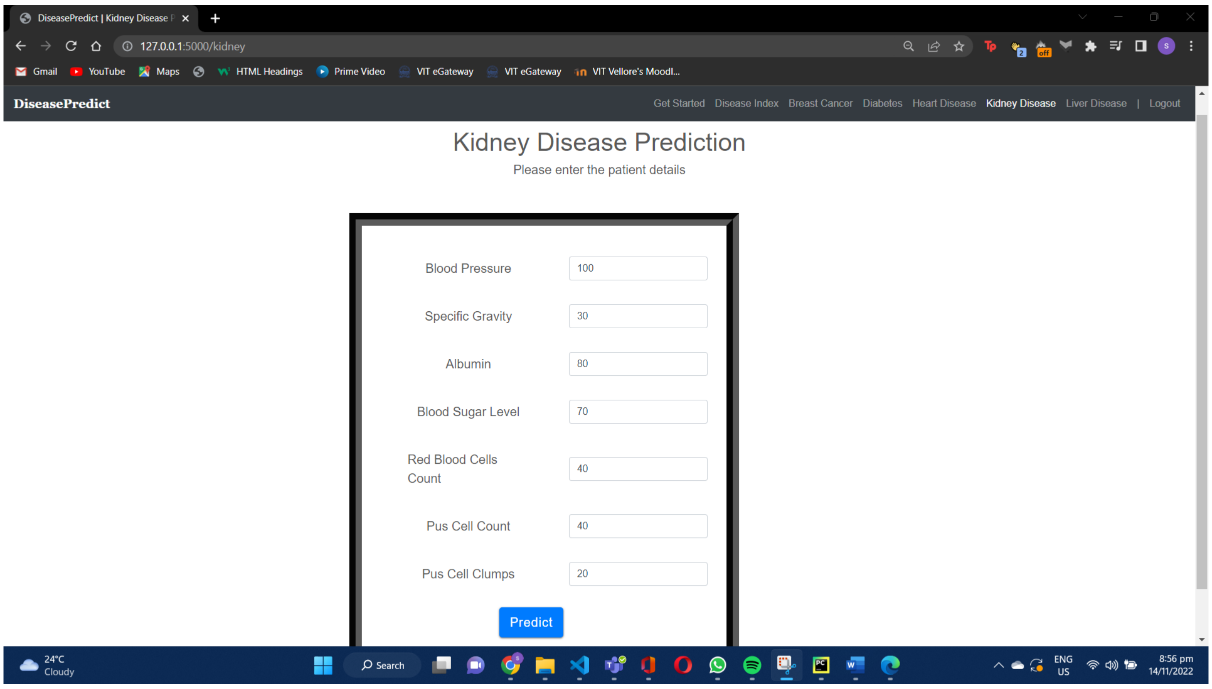
5. Liver Disease

Navigate to the "Liver Disease" section and add the corresponding details and click on predict to learn whether the patient is at the risk of this illness/disease.

6. Disease Index

Navigate to the "Disease Index" section and gain in-depth information about each of these diseases.

Test Case 1:
Kidney Disease Prediction
Patient's Details entered:


DiseasePredict

Get Started | Disease Index | Breast Cancer | Diabetes | Heart Disease | Kidney Disease | Liver Disease | Logout

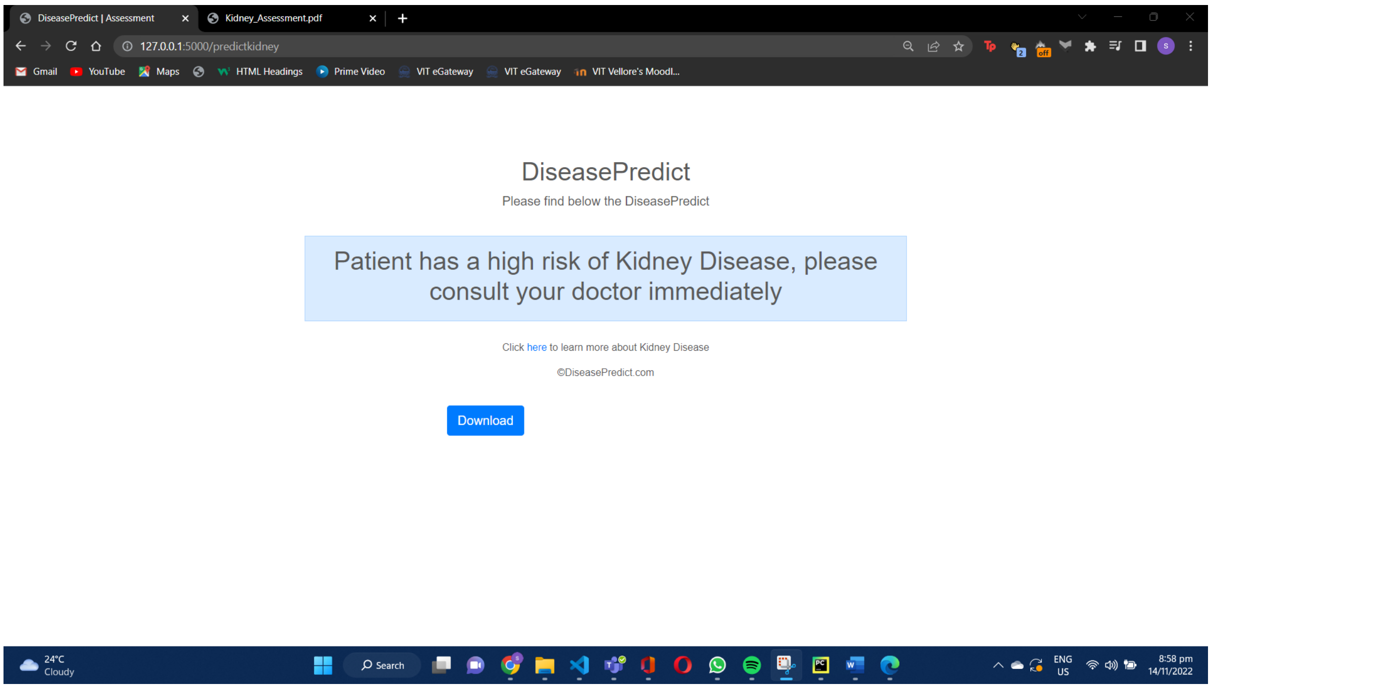
Kidney Disease Prediction

Please enter the patient details

Blood Pressure	100
Specific Gravity	30
Albumin	80
Blood Sugar Level	70
Red Blood Cells Count	40
Pus Cell Count	40
Pus Cell Clumps	20

Predict

Results:


DiseasePredict

Please find below the DiseasePredict

Patient has a high risk of Kidney Disease, please consult your doctor immediately

Click here to learn more about Kidney Disease

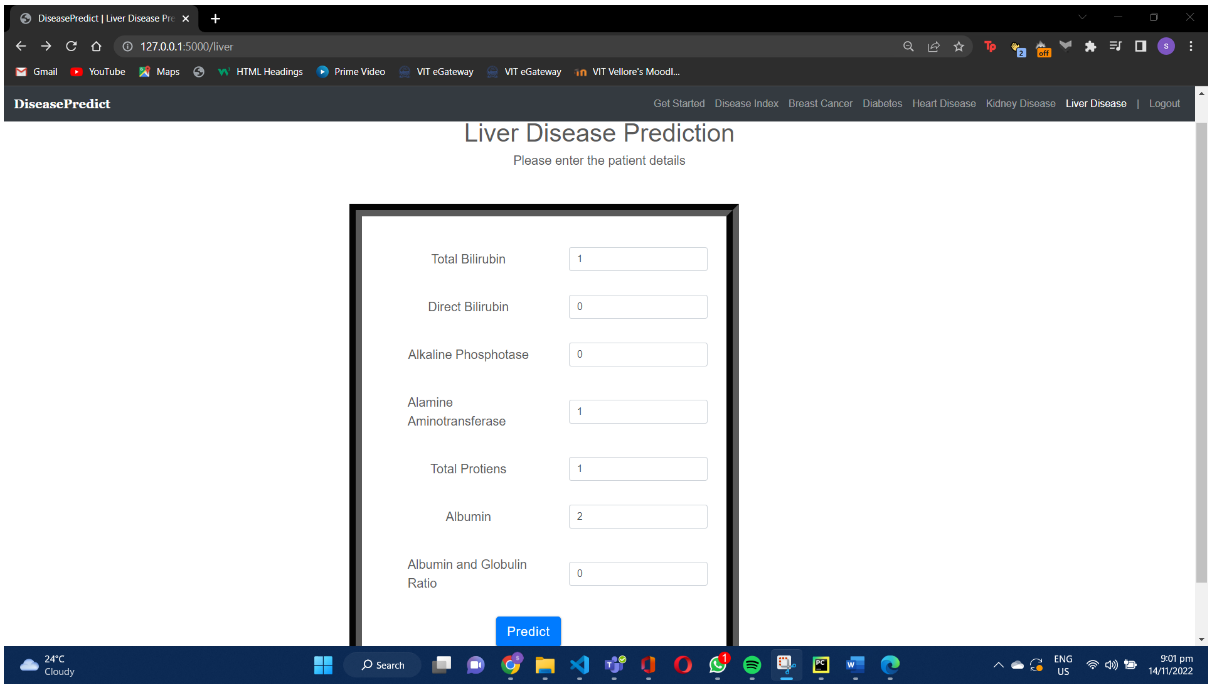
©DiseasePredict.com

Download

Can be downloaded as pdf file.

Test case 2:

Liver Disease:



DiseasePredict | Liver Disease Pr

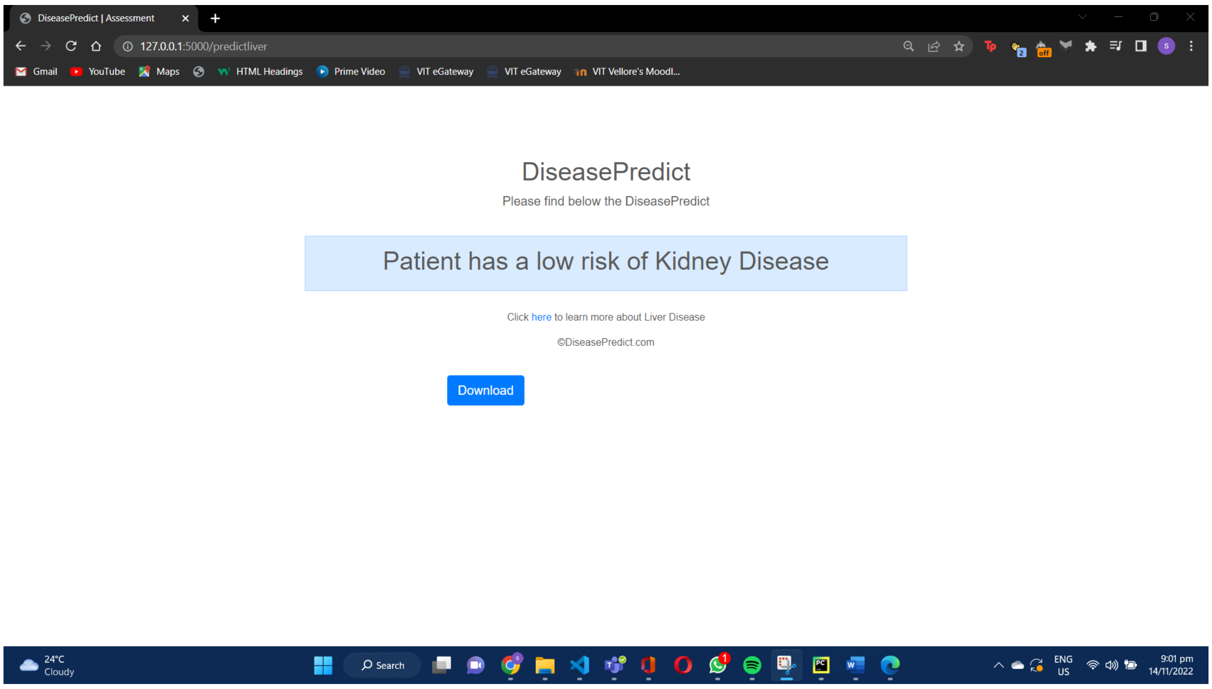
127.0.0.1:5000/liver

Liver Disease Prediction

Please enter the patient details

Total Bilirubin	1
Direct Bilirubin	0
Alkaline Phosphotase	0
Alamine Aminotransferase	1
Total Protiens	1
Albumin	2
Albumin and Globulin Ratio	0

Results:


DiseasePredict

Please find below the DiseasePredict

Patient has a low risk of Kidney Disease

Click here to learn more about Liver Disease

©DiseasePredict.com

5.2. Testing

Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

5.2.1. Types of Testing

- ***Unit testing:***

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit *tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.*

- ***Integration testing:***

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

- ***Functional testing:***

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input : *identified classes of valid input must be accepted.*
Invalid Input : *identified classes of invalid input must be rejected.*
Functions : *identified functions must be exercised.*
Output : *identified classes of application outputs must be exercised.*
Systems/Procedures: *interfacing systems or procedures must be invoked.*

- ***System Testing:***

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasising pre-driven process links and integration points.

- ***White Box Testing:***

  White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It has a purpose. It is used to test areas that cannot be reached from a black box level.

5.2.2. Test cases

Steps	***Teststeps***	***Expected result***	***Actual result***	***Notes***	***Status (Pass/ Fail)***
1	*Register a user in website*	*User should be registered successfully*	*user is registered successfully*	*Working as expected*	*Pass*
2					*Fail*
3					*Pass*
4	*Enter invalid data points for prediction of heart disease*	*The user should get a warning about the invalid range.*	*Successfully generates results and says chronic heart is detected.*	*Range of data points should be suggested*	*Fail*
5	*Enters valid data points for prediction of breast cancer*	*The user should get a warning about the invalid range.*	*Successfully generates results and says breast cancer.*	*Working as expected.*	*Pass*
6	*To download report as pdf*	*Should be to download report as pdf*	*Successfully downloads the report as pdf*		*Pass*

Accuracy table:

DISEASE	ALGORITHM	ACCURACY
Cardiac related	Decision Tree (Purusothaman, et al.)	76%
	Naïve Bayes (Purusothaman, et al.)	69%
Diabetic related	Decision Tree (Sisodia, et al.)	73.82%
	Naïve Bayes (Sisodia, et al.)	76.30%
Cancer related	Decision Tree (Sumbaly, et al.)	94.5%
	Naïve Bayes (Chaurasia, et al.)	97.36%

6. Conclusion, Limitations and Scope for future Work

Specifying the processing of natural healthcare data of various diseases information will help in the long term Saving human lives and early detection of abnormalities in serious conditions. Machine learning Techniques were used in this work to filter data and supply a replacement and novel understanding Towards heart situation.Disease projection is demanding and very significant in the medical field. However, the mortality ratio can be drastically regulated if the disease is observed at the first Stages and preventive criteria are accepted as soon as feasible. Further extension of this study is very desirable to direct the investigations to real-world datasets rather than just theoretical Approaches and simulations. The proposed hybrid HRFLM strategy utilises stirring the Characteristics of Random Forest (RF) and Linear Method (LM). HELM verified to be quite Accurate within the prediction of heart condition . The future course of this research can be performed Withdiverse mixtures of machine learning techniques to better prediction techniques. Furthermore, New feature selection methods are often developed to urge a broader perception of the many Features to extend the performance of heart condition prediction.

One of the considerable disadvantages is related to security. It is a usual matter of concern, if you want to go online without sufficient defence that can create a big security problem. The data violation is one of the substantial problems of the healthcare industry and is thought about as the most enlightened of all the problems. The procedure of software execution is a tough task. At any rate the process of getting training decently is another strain. It has been established that it's main to learn each item concerning the software to make use of it to the maximum. So aside from unification of the software for your organisational need, make sure that the employees are well trained on that.

As a part of future scope, we can add –
- *A Notification Alert System*
- *Live tracking*
- *Portal to Order Medicines from nearby Medical Stores*
- *We will extend the limit of our platform by involving other services.*
- *We will add functionality to directly import data from various*
- *health related electronic devices to our database.*

8. References

1. ***A. Yaganteeswarudu, "Multi Disease Prediction Model by using Machine Learning and Flask API," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 1242-1246, doi: 10.1109/ICCES48766.2020.9137896.***

2. ***Vijiyarani, S., and S. Sudha. "Disease prediction in data mining technique–a survey." International Journal of Computer Applications & Information Technology 2.1 (2013): 17-21.***

3. ***Shamrat, FM Javed Mehedi, et al. "An analysis on breast disease prediction using machine learning approaches." International Journal of Scientific & Technology Research 9.02 (2020): 2450-2455.***

4. ***Krishnaiah, V., G. Narsimha, and N. Subhash Chandra. "Diagnosis of lung cancer prediction system using data mining classification techniques." International Journal of Computer Science and Information Technologies 4.1 (2013): 39-45.***

5. ***Nikhar, Sonam, and A. M. Karandikar. "Prediction of heart disease using machine learning algorithms." International Journal of Advanced Engineering, Management and Science 2.6 (2016): 239484.***

6. ***A. Singh and R. Kumar, "Heart Disease Prediction Using Machine Learning Algorithms," 2020 International Conference on Electrical and Electronics Engineering (ICE3), 2020, pp. 452-457, doi: 10.1109/ICE348803.2020.9122958.***

7. ***Vijayarani, S., and S. Dhayanand. "Liver disease prediction using SVM and Naïve Bayes algorithms." International Journal of Science, Engineering and Technology Research (IJSETR) 4.4 (2015): 816-820.***

8. ***Ifraz, Gazi Mohammed, et al. "Comparative Analysis for Prediction of Kidney Disease Using Intelligent Machine Learning Methods." Computational and Mathematical Methods in Medicine 2021 (2021).***

9. ***Ghosh, Pronab, et al. "Optimization of prediction method of chronic kidney disease using machine learning algorithm." 2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP). IEEE, 2020.***

10. ***A. D. Gunasinghe, A. C. Aponso and H. Thirimanna, "Early Prediction of Lung Diseases," 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), 2019, pp. 1-4, doi: 10.1109/I2CT45611.2019.9033668.***

11. ***M. S. Kumar and K. V. Rao, "Prediction of Lung Cancer Using Machine Learning Technique: A Survey," 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-5, doi: 10.1109/ICCCI50826.2021.9402320.***

9. Code:

```
import joblib
from flask import Flask, render_template, redirect, url_for, request
from flask_bootstrap import Bootstrap
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField
from wtforms.validators import InputRequired, Email, Length
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user
import pandas as pd
import pickle
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

filename = 'diabetes-prediction-rfc-model.pkl'
classifier = pickle.load(open(filename, 'rb'))
model = pickle.load(open('model.pkl', 'rb'))
model1 = pickle.load(open('model1.pkl', 'rb'))

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
bootstrap = Bootstrap(app)
db = SQLAlchemy(app)
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'


class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(15), unique=True)
    email = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(80))


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


class LoginForm(FlaskForm):
    username = StringField('Username', validators=[InputRequired(), Length(min=4,
max=15)])
```

```
    password = PasswordField('Password', validators=[InputRequired(), Length(min=8,
max=80)])
    remember = BooleanField('remember me')


class RegisterForm(FlaskForm):
    email = StringField('Email', validators=[InputRequired(), Email(message='Invalid email'),
Length(max=50)])
    username = StringField('Username', validators=[InputRequired(), Length(min=4,
max=15)])
    password = PasswordField('Password', validators=[InputRequired(), Length(min=8,
max=80)])


@app.route('/')
def index():
    return render_template("index.html")


@app.route('/about')
def about():
    return render_template("about.html")


@app.route('/help')
def help():
    return render_template("help.html")


@app.route('/terms')
def terms():
    return render_template("tc.html")


@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user:
            if check_password_hash(user.password, form.password.data):
                login_user(user, remember=form.remember.data)
                return redirect(url_for('dashboard'))

        return render_template("login.html", form=form)
    return render_template("login.html", form=form)
```

```
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    form = RegisterForm()
    if form.validate_on_submit():
        hashed_password = generate_password_hash(form.password.data, method='sha256')
        new_user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
        db.session.add(new_user)
        db.session.commit()

        return redirect("/login")
    return render_template('signup.html', form=form)


@app.route("/dashboard")
@login_required
def dashboard():
    return render_template("dashboard.html")


@app.route("/disindex")

def disindex():
    return render_template("disindex.html")


@app.route("/cancer")
@login_required
def cancer():
    return render_template("cancer.html")


@app.route("/diabetes")
@login_required
def diabetes():
    return render_template("diabetes.html")


@app.route("/heart")
@login_required
def heart():
    return render_template("heart.html")


@app.route("/kidney")
@login_required
def kidney():
    return render_template("kidney.html")
```

```
def ValuePredictor(to_predict_list, size):
    to_predict = np.array(to_predict_list).reshape(1, size)
    if size == 7:
        loaded_model = joblib.load('kidney_model.pkl')
        result = loaded_model.predict(to_predict)
    return result[0]


@app.route("/predictkidney",  methods=['GET', 'POST'])
def predictkidney():
    if request.method == "POST":
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(float, to_predict_list))
        if len(to_predict_list) == 7:
            result = ValuePredictor(to_predict_list, 7)
    if(int(result) == 1):
        prediction = "Patient has a high risk of Kidney Disease, please consult your doctor
immediately"
    else:
        prediction = "Patient has a low risk of Kidney Disease"
    return render_template("kidney_result.html", prediction_text=prediction)


@app.route("/liver")
@login_required
def liver():
    return render_template("liver.html")


def ValuePred(to_predict_list, size):
    to_predict = np.array(to_predict_list).reshape(1,size)
    if(size==7):
        loaded_model = joblib.load('liver_model.pkl')
        result = loaded_model.predict(to_predict)
    return result[0]


@app.route('/predictliver', methods=["POST"])
def predictliver():
    if request.method == "POST":
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(float, to_predict_list))
        if len(to_predict_list) == 7:
            result = ValuePred(to_predict_list, 7)
    if int(result) == 1:
        prediction = "Patient has a high risk of Liver Disease, please consult your doctor
immediately"
    else:
        prediction = "Patient has a low risk of Kidney Disease"
    return render_template("liver_result.html", prediction_text=prediction)


@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('index'))


@app.route('/predict', methods=['POST'])
def predict():
    input_features = [int(x) for x in request.form.values()]
    features_value = [np.array(input_features)]
    features_name = ['clump_thickness', 'uniform_cell_size', 'uniform_cell_shape',
'marginal_adhesion',
                    'single_epithelial_size', 'bare_nuclei', 'bland_chromatin', 'normal_nucleoli',
'mitoses']
    df = pd.DataFrame(features_value, columns=features_name)
    output = model.predict(df)
    if output == 4:
        res_val = "a high risk of Breast Cancer"
    else:
        res_val = "a low risk of Breast Cancer"

    return render_template('cancer_result.html', prediction_text='Patient has
{}'.format(res_val))



df1 = pd.read_csv('diabetes.csv')

# Renaming DiabetesPedigreeFunction as DPF
df1 = df1.rename(columns={'DiabetesPedigreeFunction': 'DPF'})

# Replacing the 0 values from ['Glucose','BloodPressure','SkinThickness','Insulin','BMI'] by
NaN
df_copy = df1.copy(deep=True)
df_copy[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] = df_copy[['Glucose',
'BloodPressure',
                                                                                    'SkinThickness', 'Insulin',
                                                                                    'BMI']].replace(0, np.NaN)
```

```
# Replacing NaN value by mean, median depending upon distribution
df_copy['Glucose'].fillna(df_copy['Glucose'].mean(), inplace=True)
df_copy['BloodPressure'].fillna(df_copy['BloodPressure'].mean(), inplace=True)
df_copy['SkinThickness'].fillna(df_copy['SkinThickness'].median(), inplace=True)
df_copy['Insulin'].fillna(df_copy['Insulin'].median(), inplace=True)
df_copy['BMI'].fillna(df_copy['BMI'].median(), inplace=True)

# Model Building

X = df1.drop(columns='Outcome')
y = df1['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)

# Creating Random Forest Model

classifier = RandomForestClassifier(n_estimators=20)
classifier.fit(X_train, y_train)

# Creating a pickle file for the classifier
filename = 'diabetes-prediction-rfc-model.pkl'
pickle.dump(classifier, open(filename, 'wb'))



@app.route('/predictt', methods=['POST'])
def predictt():
    if request.method == 'POST':
        preg = request.form['pregnancies']
        glucose = request.form['glucose']
        bp = request.form['bloodpressure']
        st = request.form['skinthickness']
        insulin = request.form['insulin']
        bmi = request.form['bmi']
        dpf = request.form['dpf']
        age = request.form['age']

        data = np.array([[preg, glucose, bp, st, insulin, bmi, dpf, age]])
        my_prediction = classifier.predict(data)

        return render_template('diab_result.html', prediction=my_prediction)



@app.route('/predictheart', methods=['POST'])
def predictheart():
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]
    features_name = ["age", "trestbps", "chol", "thalach", "oldpeak", "sex_0",
                     "  sex_1", "cp_0", "cp_1", "cp_2", "cp_3", "  fbs_0",
                     "restecg_0", "restecg_1", "restecg_2", "exang_0", "exang_1",
                     "slope_0", "slope_1", "slope_2", "ca_0", "ca_1", "ca_2", "thal_1",
                     "thal_2", "thal_3"]

    df = pd.DataFrame(features_value, columns=features_name)
    output = model1.predict(df)

    if output == 1:
        res_val = "a high risk of Heart Disease"
    else:
        res_val = "a low risk of Heart Disease"

    return render_template('heart_result.html', prediction_text='Patient has {}'.format(res_val))



if __name__ == "__main__":
    app.run(debug=True)


import joblib
from flask import Flask, render_template, redirect, url_for, request
from flask_bootstrap import Bootstrap
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, BooleanField
from wtforms.validators import InputRequired, Email, Length
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user
import pandas as pd
import pickle
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

filename = 'diabetes-prediction-rfc-model.pkl'
classifier = pickle.load(open(filename, 'rb'))
model = pickle.load(open('model.pkl', 'rb'))
model1 = pickle.load(open('model1.pkl', 'rb'))

app = Flask(__name__)
app.config['SECRET_KEY'] = 'secret'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
bootstrap = Bootstrap(app)
db = SQLAlchemy(app)
```

```
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'


class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(15), unique=True)
    email = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(80))


@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))


class LoginForm(FlaskForm):
    username = StringField('Username', validators=[InputRequired(), Length(min=4,
max=15)])
    password = PasswordField('Password', validators=[InputRequired(), Length(min=8,
max=80)])
    remember = BooleanField('remember me')


class RegisterForm(FlaskForm):
    email = StringField('Email', validators=[InputRequired(), Email(message='Invalid email'),
Length(max=50)])
    username = StringField('Username', validators=[InputRequired(), Length(min=4,
max=15)])
    password = PasswordField('Password', validators=[InputRequired(), Length(min=8,
max=80)])


@app.route('/')
def index():
    return render_template("index.html")


@app.route('/about')
def about():
    return render_template("about.html")


@app.route('/help')
def help():
    return render_template("help.html")
```

```
@app.route('/terms')
def terms():
    return render_template("tc.html")


@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        if user:
            if check_password_hash(user.password, form.password.data):
                login_user(user, remember=form.remember.data)
                return redirect(url_for('dashboard'))

        return render_template("login.html", form=form)
    return render_template("login.html", form=form)


@app.route('/signup', methods=['GET', 'POST'])
def signup():
    form = RegisterForm()
    if form.validate_on_submit():
        hashed_password = generate_password_hash(form.password.data, method='sha256')
        new_user = User(username=form.username.data, email=form.email.data,
password=hashed_password)
        db.session.add(new_user)
        db.session.commit()

        return redirect("/login")
    return render_template('signup.html', form=form)


@app.route("/dashboard")
@login_required
def dashboard():
    return render_template("dashboard.html")


@app.route("/disindex")

def disindex():
    return render_template("disindex.html")


@app.route("/cancer")
@login_required
```

```
def cancer():
    return render_template("cancer.html")


@app.route("/diabetes")
@login_required
def diabetes():
    return render_template("diabetes.html")


@app.route("/heart")
@login_required
def heart():
    return render_template("heart.html")


@app.route("/kidney")
@login_required
def kidney():
    return render_template("kidney.html")


def ValuePredictor(to_predict_list, size):
    to_predict = np.array(to_predict_list).reshape(1, size)
    if size == 7:
        loaded_model = joblib.load('kidney_model.pkl')
        result = loaded_model.predict(to_predict)
    return result[0]


@app.route("/predictkidney",  methods=['GET', 'POST'])
def predictkidney():
    if request.method == "POST":
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(float, to_predict_list))
        if len(to_predict_list) == 7:
            result = ValuePredictor(to_predict_list, 7)
    if(int(result) == 1):
        prediction = "Patient has a high risk of Kidney Disease, please consult your doctor
immediately"
    else:
        prediction = "Patient has a low risk of Kidney Disease"
    return render_template("kidney_result.html", prediction_text=prediction)


@app.route("/liver")
@login_required
```

```
def liver():
    return render_template("liver.html")


def ValuePred(to_predict_list, size):
    to_predict = np.array(to_predict_list).reshape(1,size)
    if(size==7):
        loaded_model = joblib.load('liver_model.pkl')
        result = loaded_model.predict(to_predict)
    return result[0]


@app.route('/predictliver', methods=["POST"])
def predictliver():
    if request.method == "POST":
        to_predict_list = request.form.to_dict()
        to_predict_list = list(to_predict_list.values())
        to_predict_list = list(map(float, to_predict_list))
        if len(to_predict_list) == 7:
            result = ValuePred(to_predict_list, 7)

    if int(result) == 1:
        prediction = "Patient has a high risk of Liver Disease, please consult your doctor
immediately"
    else:
        prediction = "Patient has a low risk of Kidney Disease"
    return render_template("liver_result.html", prediction_text=prediction)


@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('index'))


@app.route('/predict', methods=['POST'])
def predict():
    input_features = [int(x) for x in request.form.values()]
    features_value = [np.array(input_features)]
    features_name = ['clump_thickness', 'uniform_cell_size', 'uniform_cell_shape',
'marginal_adhesion',
                     'single_epithelial_size', 'bare_nuclei', 'bland_chromatin', 'normal_nucleoli',
'mitoses']
    df = pd.DataFrame(features_value, columns=features_name)
    output = model.predict(df)
    if output == 4:
        res_val = "a high risk of Breast Cancer"
    else:
        res_val = "a low risk of Breast Cancer"

    return render_template('cancer_result.html', prediction_text='Patient has
{}'.format(res_val))



df1 = pd.read_csv('diabetes.csv')

# Renaming DiabetesPedigreeFunction as DPF
df1 = df1.rename(columns={'DiabetesPedigreeFunction': 'DPF'})

# Replacing the 0 values from ['Glucose','BloodPressure','SkinThickness','Insulin','BMI'] by
NaN
df_copy = df1.copy(deep=True)
df_copy[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']] = df_copy[['Glucose',
'BloodPressure',
                                                                                    'SkinThickness', 'Insulin',
                                                                                    'BMI']].replace(0, np.NaN)

# Replacing NaN value by mean, median depending upon distribution
df_copy['Glucose'].fillna(df_copy['Glucose'].mean(), inplace=True)
df_copy['BloodPressure'].fillna(df_copy['BloodPressure'].mean(), inplace=True)
df_copy['SkinThickness'].fillna(df_copy['SkinThickness'].median(), inplace=True)
df_copy['Insulin'].fillna(df_copy['Insulin'].median(), inplace=True)
df_copy['BMI'].fillna(df_copy['BMI'].median(), inplace=True)

# Model Building

X = df1.drop(columns='Outcome')
y = df1['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)

# Creating Random Forest Model

classifier = RandomForestClassifier(n_estimators=20)
classifier.fit(X_train, y_train)

# Creating a pickle file for the classifier
filename = 'diabetes-prediction-rfc-model.pkl'
pickle.dump(classifier, open(filename, 'wb'))



@app.route('/predictt', methods=['POST'])
def predictt():
    if request.method == 'POST':
        preg = request.form['pregnancies']
        glucose = request.form['glucose']
        bp = request.form['bloodpressure']
        st = request.form['skinthickness']
        insulin = request.form['insulin']
        bmi = request.form['bmi']
        dpf = request.form['dpf']
        age = request.form['age']

        data = np.array([[preg, glucose, bp, st, insulin, bmi, dpf, age]])
        my_prediction = classifier.predict(data)

        return render_template('diab_result.html', prediction=my_prediction)



@app.route('/predictheart', methods=['POST'])
def predictheart():
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]

    features_name = ["age", "trestbps", "chol", "thalach", "oldpeak", "sex_0",
                    "  sex_1", "cp_0", "cp_1", "cp_2", "cp_3", "  fbs_0",
                    "restecg_0", "restecg_1", "restecg_2", "exang_0", "exang_1",
                    "slope_0", "slope_1", "slope_2", "ca_0", "ca_1", "ca_2", "thal_1",
                    "thal_2", "thal_3"]

    df = pd.DataFrame(features_value, columns=features_name)
    output = model1.predict(df)

    if output == 1:
        res_val = "a high risk of Heart Disease"
    else:
        res_val = "a low risk of Heart Disease"

    return render_template('heart_result.html', prediction_text='Patient has {}'.format(res_val))



if __name__ == "__main__":
    app.run(debug=True)
```