1. **INTRODUCTION**

Today’s modern and busy world consists of people who lack of time to take care of their health. This seems to be simple and small problem in reality it is a very serious problem the world is facing now. Many diseases have several stages. Depending upon each stage the patient’s condition, treatment, risk factor varies. Early detection of diseases can save many lives. It is necessary to have a system for checking the health regularly within a less time based on their symptoms.

When one suffers from a particular disease, then the person has to visit the doctor which is time consuming and costly too. Also if the user is out of reach of doctor and hospitals it may be completed using automated program which can save time and less money it could be easier to the patient which can make their process easier.

As the use of internet is growing every day, people are always curious to know different new things. People always try to refer to the internet if any problem arises. People have access to internet than hospitals and doctors. People do not have immediate options when they suffer with particular disease. So, this system can be helpful as they can check their health regularly.

There are lot of classification algorithms available now but it is not possible to conclude which one is superior to other. It depends on the application and nature of the dataset. The classification algorithm we used is Naive Bayes algorithm.

Our project allows patient to find a doctor based on specialization and book appointment based upon the respective time and date which can save patient’s precious time.

**MACHINE LEARNING**

Machine learning is a category of algorithm that allows software applications to becomes more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as a new data becomes available.

The process of learning begins with observations or data such as examples, direct experience or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human assistance.

**1.1 Literature survey**

MinChen et.al[1]A new convolution neural network based multimodal disease detection(CNN-MDRP) algorithm using structured and unstructured data from hospital is taken for their work. In their research, streamlining of machine learning algorithms for effective detection of chronic disease outbreak in disease-frequent communities is studied. Experiment is done on modified prediction models over real-life hospital data. Moreover, different regions exhibit unique characteristics of certain regional diseases, which may weaken the prediction of disease outbreaks. Compared to several typical prediction algorithms, the prediction accuracy of this proposed algorithm reaches almost highest percent with a convergence speed which is faster than that of the CNN-based uni-model disease risk prediction (CNN-UDRP) algorithm.

P. Siriyasatien et.al.[2]After the study of this work we came to know about the major components that can be used in a dengue prediction model. Their important research objective is to develop models that enable, or enhance, forecasting of outbreaks of dengue giving medical professionals the opportunity to develop plans for handling the outbreak, well in advance. Researchers have gathered and analyzed data to better identify the relational factors driving the spread of the disease, as well as the development of a variety of methods of predictive modeling using statistical and mathematical analysis of machine learning techniques, Decision tree, Regression Analysis, Artificial Neural Networks(ANN),Support Vector Machine(popularly known as “SVM”), k-nearest Neighbour are also used. Among Unsupervised machine learning techniques, K-means Clustering is used. Both the direct factors and indirect factors which results in dengue fever are considered and processed in this project.

Somaya Hashem et.al.[3]In studying this we understood that the research work is that they made a comparison between different machine learning approaches on prediction of advanced liver fibrosis in Chronic Hepatitis C patients. Particle swarm optimization, decision tree, multi-linear regression and genetic algorithm models were developed. This study concluded that prediction of advanced fibrosis stage for chronic HCV patients using different machine learning approaches with high accuracy is possible. The four parameters age, AST, albumin and platelets count found to be the most important features in prediction of the fibrosis as they have statistically significant relationship and accepted correlation greater coefficients with presence of advanced fibrosis as shown in the results.. The proposed models could be used as an acceptable, safe, and low cost alternating for predict advanced fibrosis rather than relatively risky alternative tools in chronic hepatitis C virus patients, but the system does not provide any information related to other hepatitis diseases.

A. Shah et.al.[4]They have combined the structure and unstructured data in healthcare fields that let us assess the risk of disease. The approach of the latent factor model for reconstructing the missing data in medical records which are collected from the hospital. And by using statistical knowledge, they could determine the major chronic diseases in a particular region and in particular community. To handle structured data, they consulted hospital experts to know useful features .In the case of unstructured text data, they select the features automatically with the help of k-mean algorithm. They proposed a k-mean algorithm for both structured and unstructured data.

D. W. Bates et.al.[5]Disease risk models involving machine learning and supervised learning algorithms which uses training data with the labels for the training of the models are used in their work. High-risk and Low-risk patient classification is done in groups test sets. But these models are only valuable in clinical situations and are widely studied. The information of patient’s statistics, test results, and disease history is recorded in EHR which enables to identify potential data-centric solutions which reduce the cost of medical case studies. Bates et al. propose six applications of big data in the healthcare field. The predictions of diseases have been non-specific and indefinite.

**1.1.1 Summary of Literature Study:**

We have studied about the different design methodologies used in various projects from the above papers. In the above papers, they used many supervised and unsupervised machine learning techniques such as Artificial Neural Networks, CNN, Decision tree, Regression Analysis and K-means Clustering. We observed that the design methodology is so complex and so we tried to achieve nearly the same accuracy in a simple way using the supervised learning algorithm called Naïve bayes classifier and we will consider the model with best accuracy when compared with remaining models.

**1.2 Existing System**

Traditionally, there was no such system developed from which we would be able to get the details of various illness/disease on a single website. In conventional system doctors make the decisions based on their knowledge. Even if they did have access to the massive amounts of data needed to compare treatment outcomes for all the diseases they encounter. Patient also have to meet the doctor personally which can take massive time to reach medical care. The problem here is it takes more time to result the disease corresponding the problem. So, for simple health disorders also the patient has to consult the doctor or they have to go to nearby hospitals.

**1.2.1 Disadvantages of Existing System**

* It consumes more time for processing the activities.
* Possibility of getting inaccurate results.
* Existing system is manual and need to be converted into automated system.
* Unavailability of medical resources.
* Patients have to go to hospitals every time for diagnosis.

**1.3 Proposed System**

To overcome the drawback of the existing system we have developed Enhanced Disease Prediction System. It is an online consultant project and user friendly. The system helps in knowing the health issues through an intelligent health care system through online. The system uses Naïve Bayes algorithm to display the most accurate results. The system contains the data of symptoms and the diseases associated with those symptoms. User will be asked to enter the symptoms, then system will process those symptoms for various illness or disease that user could be matched by using machine learning. . If user’s symptoms does not match any disease then it gives the related information that the patient could possibly have. Appointments can also be booked as per the convenient dates and time slot.

**1.3.1 Features of Proposed System**

The new system contains the following activities which try to automate the entire process:

* More Efficient
* There is no need for the patients to wait for long period of time
* User Friendly
* Makes the management much easier and flexible too
* Users can access this system from any part of the world
* Can book appointments for the respective specialists
* Flexible to book the appointments according to their respective dates and time.
* Patient can search to know about the overview of the predicted disease.

2.**REQUIREMENT ANALYSIS**

Requirement analysis is a software engineering task that bridges the gap between system software allocation and software design. It provides system engineer to specify software function and performance indicate software’s interface with other system elements and establish constraints that software must need .The basic is to obtain a clear picture of the needs and requirements of the end user and also the organization.

**2.1Hardware &Software Requirements:**

**2.1.1Hardware Requirements**:

When you plan for the hardware that is required for a project, as a starting point, you should determine the usage requirements for your project environment. These variables include the number of projects, tasks, users, average task per project, and s on.

* Processor-Intel i3, dual core
* RAM- 8GB
* Disk space -1GB

**2.1.2 Software Requirements:**

In addition to the hardware requirements, certain software ‘s are also essential for the successful execution of the project. In fact they can be considered as the heart of the project . They vary from project to the project . One set of software requirements exist for web-based project.

* Operating System-Windows 10
* Programming Language-python 3.8.1
* Machine learning algorithm

**2.2 Software Requirement Specification**

**2.2.1 Vision**

|  |  |
| --- | --- |
| **PROJECT ABSTRACT** | |
| **Name of the Project** | **ENHANCED DISEASE PREDICTION USING MACHINE LEARNING AND DOCTOR APPOINTMENT SYSTEM** |
| **Vision** | **For** the patient who seek instant guidance for their diseases according to their symptoms, to search the specialists dependent on the illness and also to book the doctor appointment the **ENHANCED DISEASE PREDICTION USING MACHINE LEARNING LEARNING AND DOCTOR APPOINTMENT SYSTEM** is a web application that provides elegant GUI based reports.  **For**  the doctor who can view the patient’s personal details, medical history and appointments the **ENHANCED DISEASE** PREDICTION **USING MACHINE LEARNING AND DOCTOR APPOINTMENT SYSTEM** is a web application that provides elegant GUI based forms.  **For** the admin who give instant guidance based on the patient symptoms , to suggest the doctors and to manage the bookings  made by the patients the **ENHANCED DISEASE PREDICTION USING MACHINE LEARNING AND DOCTOR APPOINTMENT SYSTEM** is a web application that provides elegant GUI based forms. |
| **Users/Actors of the System** | * Admin * Doctor * Patient |
| **System Features & Functional Capabilities** | * Based on symptoms provided by patient, it will show the related diseases/illness using Naïve Bayes, classifier algorithm. * It provides patients symptoms to trained module to make predictions. * Compare the given the symptoms with the input database. * Provides address and speciality of the doctors. * Disease symptoms search dashboard. * Patients can book appointment for their required date and time. |
| **Technologies/Tools to be Used** | * Python 3 * Django Framework * Html * Css * Bootstrap * Materializecss |
| **Third Party libraries /APIs/Services to be used** | * Scikit-learn * NumPy * Pandas |
| **Final Deliverable must Include** | * Complete source code * Data set |
| **Documents** | * Abstract * Vision Document * SRS document |

**2.2.2.Scope:**  
  
The ENHANCED DISEASE PREDICTION USING MACHINE LEARNING AND DOCTOR APPOINTMENT SYSTEM is an end user support project. It allows users to know about their health issuesthrough an intelligent health care system through online. The system contains the data of various symptoms and the diseases associated with those symptoms. The system processes those symptoms and checks the various illnesses that can be associated with it. The system uses intelligent supervised learning technique called Naïve Bayes classifier to guess the most accurate illness. If user's symptoms do not match any disease in the database, then it shows the diseases that user could probably have. Using structured and unstructured data from hospital it uses the algorithm for effective prediction of various disease data collected. The system will also allows the user to book the appointment to their respective specialized doctors, date and time.

**Inclusions:**  
i.GUIforinputsymptoms.  
ii. GUI to view the predicted disease.

iii. GUI to book the appointment.

iv.GUItoviewtheappointments.

**Assumptions:**  
i.ThedatasetisuploadedasanExcelfile.  
ii. The data set will be get trained.

**2.2.3 System Functions**

This system comprises of 3 major functions with their sub-functions as follows:

**1. Admin Login:**

* **Login:** Admin can login in to the system by using username and password.
* **Add / View Doctor:** Admin can registration new doctor into the system by entering the registration details and providing them the login credentials.
* **Upload dataset:** System allows admin to upload dataset.
* **Manage Bookings:** Can manage the appointment bookings.

**2. Patient:**

* **Registration / Login:** Patient need to register by providing basic registration details and create a login id and password. After successful registration, patient can now login into the system.
* **Predict Disease:** Here, the patient will enter the health condition such as headache and the symptoms will be displayed for selection. After selecting the symptoms, the will predict the disease from which the patient is suffering from.
* **Book Appointment:** Patient can also book appointment for a doctor of respective disease, date and time.
* **View Own Profile:** Can view their own profile details here.

**3.Doctor’s Login:**

* **Registration / Login:** Doctor need to register by providing basic registration details and create a login id and password. After successful registration, Doctor can now login into the system
* **View Appointments:** Can view the appointments made by the patients.
* **View Own Profile:** Can view their own profile details here.

**2.2.4 Detailed Software Requirements**

|  |  |
| --- | --- |
| **Actor Name** | Administrator |
| Actor Id | TC01 |
| Description | Handles all admin related tasks throughout the Application |
| Main Activities | Add and delete doctors patients, upload the training and testing datasets, manage bookings |
| Frequency of Use | Medium |
| Work Environment /  Location | Browser |
| Number of Users | Any number |

**Table 2.1 Administrator use case model**

|  |  |
| --- | --- |
| **Actor Name** | Patient |
| Actor Id | TC02 |
| Description | View the most accurate disease after prediction and to book the appointment. |
| Main Activities | View predicted disease, book appointments |
| Frequency of Use | High |
| Work Environment /  Location | Browser |
| Number of Users | Any number |

**Table 2.2 Patient use case model**

|  |  |
| --- | --- |
| **Actor Name** | Doctor |
| Actor Id | TC03 |
| Description | Know about the appointments made by the patients. |
| Main Activities | View appointments |
| Frequency of Use | High |
| Work Environment /  Location | Browser |
| Number of Users | Any number |

**Table 2.3 Doctor use case model**

**2.2.5 Detailed Use Case Description**

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Upload the training and testing dataset | |
| **Use Case ID** | UC\_01 | |
| **Actor(s)** | Admin | |
| **Goal** | Upload of dataset as a excel file and successful training of dataset | |
| **Preconditions** | The dataset contains real time symptoms and diseases associated with those symptoms. | |
| **Main flow** | 1. Goto administration page/database files | 1.1 It shows choose file option. |
| 1. Choose file type(training or testing) that you want to upload | 2.2 It shows upload option |
| 1. Choose file from computer | 3.1 click ok and save. |

|  |  |
| --- | --- |
| **Use Case Name** | Predicting \_disease |
| **Use Case ID** | UC\_02 |
| **Actor(s)** | Patient |
| **Goal** | To predict the disease according to the symptoms |
| **Summary** | The patient enter the atleast 4 symptoms so that the system will provides the accurate result. |
| **Preconditions** | The patient should have internet connection for using the software |
| **Main Flow** | |  |  | | --- | --- | | 1. The patient enters the use case. | 1.1 The system displays the options like search disease, book appointment. | | 2. Patient selects search disease | 2.1 The system asks for inputs like symptoms | | 3. Patient gives the atleast 4 symptoms one by one. | 3.1 The system asks for predict | | 4.Patient clicks on get predicted result | 4.1 The system gives the predicted disease with accuracy. | |

|  |  |
| --- | --- |
| **Use Case Name** | Book\_Appointment |
| **Use Case ID** | UC\_03 |
| **Actor(s)** | patient |
| **Goal** | To book the appointments according to their respective specialists, dates and time. |
| **Summary** | The Patient can book the doctor appointment to their respective specialists, date and time. |
| **Preconditions** | The previous format of dataset should be available to admin. |
| **Main Flow** | |  |  | | --- | --- | | 1. patient can login in to the system | 1.1 The system displays the options like search disease, book appointment. | | 2. Patient selects the Book appointment | 2.1 The system asks the inputs like city, speciality. | | 3. Patient enter the inputs. | 3.1 The system will displays the available doctors. | | 4. Patient can book the appointment to their respective date and time. | 4.1 The system asks for confirmation or cancellation. | | 5 Patient can confirm or cancel their booking. |  | |

**2.2.6 Functional Capabilities**

* The admin has to authenticate a user.
* The dataset which is uploaded is directly saved.
* Atleast four symptoms should be entered.

**2.2.6.1 Business Rules / Validations**

* Only the authorized person can upload the dataset.
* One can upload only one csv file at a time.
* Only the admin can add diseases/symptoms.

**2.2.7.SecurityRequirements**  
  
**2.2.7.1 User Management:**

* The user should work in pyCharm Editor.
* The user should install Django.
* The user should run server.

**2.2.7.2 User Authentication:**

* The data in the Excel file should not be modified.
* Only authorized users should use the application.
* Required fields must be filled.

**2.2.7.3 User roles and access control:**

* The user can enter the symptoms.
* The user can know the predicted disease
* The user can know the accuracy of the disease.
* The user can book the appointment.
* The user can view the appointments.

**2.2.8 Other Non-functional requirements**

* Minimum time required for file execution is 5 seconds. (Basing on the System Performance)
* User should have knowledge in python. (Usability).
* System must contain at least 4GB of RAM, 2GB of free disk space.(Supportability).
* It is reliable
* The internet connection should be available
* Only admin can upload the database.
* Avoid manipulation of the data in the dataset.

**3. SYSTEM DESIGN**

“SYSTEM DESIGN” phase follows analysis phase. Design is maintaining record of proof design divisions and providing a blueprint for the implementation phase. Design is the bridge between system analysis and system implementation.

System design is transition from a user oriented, document oriented to programmers. The design is a solution, a “how to” approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Design does through logical and physical stages of development; a logical design review the present physical system, prepare input and an output specification, detail the implementation plan, and prepares a logical design walkthrough.

**Design methodology:**

The design process for software system has two levels:

* System design or Top level design.
* Detailed design or Logical design.

**System design:** In the system design the focus is on deciding which modules are needed for the system, the specification of these modules and how these modules should be incorrected.

**Detailed design:** In the detailed design the interconnection of the modules or how the specification of the modules can be satisfied is decided. Some properties for a software system designs are

* Verifiability.
* Completeness.
* Consistency.
* Traceability.
* Simplicity/ Understand ability.
  1. **Data Dictionary**

A data dictionary is a file or a set of files that contains a database’s metadata. The data dictionary contains records about other objects in the database, such as data ownership, data relationships to other objects, and other data.

A Data Dictionary provides information about each attribute which is also known as fields of a data model. An attribute is a place in the database that holds information. It is typically organized in a spreadsheet format. Each attribute is listed as a row in the spreadsheet and each column label m element of information that is useful to know about the attribute.

A Data Dictionary, also called a Data Definition Matrix, provides detailed information about the business data, such as standard definitions of data elements, their meanings and allowable values. While a conceptual or logical Entity Relationship Diagram will focus on the high-level business concepts, a Data Dictionary will provide more detail about each attribute of a business concept.

**Elements of Data Dictionary:-**

Data dictionary is a table with data elements (columns) as rows and their attributes as columns. Specific attributes vary depending on the purpose of the data dictionary.

**Essential elements:-**

Data dictionary has 2 essential elements

1. List of tables (or entities)
2. List of columns (or fields, or attributes)

The key Elements of a Data Dictionary are as follows:

**Attribute Name**:- A unique identifier, typically expressed business language.

**Optional/Required**:- Indicates whether information is required in an attribute before a record can be saved.  
**Attribute Type:-** Defines what type of data is allowable in a field. Commontypesinclude text, numeric, date/time, enumerated list, lookups, Booleans and  
unique identifiers.

|  |  |
| --- | --- |
| TERM | DEFINITION |
| Admin | System admin is a person who is responsible for managing the whole system and who has full access to the system |
| Patient | A patient accessing the application using any standard web browser and internet connection |
| Doctor | A doctor accessing the application using any standard web browser and internet connection |
| Dataset | An .xslx file with dimensions 4921x133 |
| Kaggle | A website where datasets can be found |

**Table 3.1**. Data Dictionary

**Use of Data Dictionary**

* Consistency
* Clarity
* Reusability
* Completeness
* To Locate error and omissions in the system
* Increase in sharing and integration and
* Ease of use for the developer.

**Completeness and Clarity**: It helps the data programmer to know that data isclear, complete, and well defined.

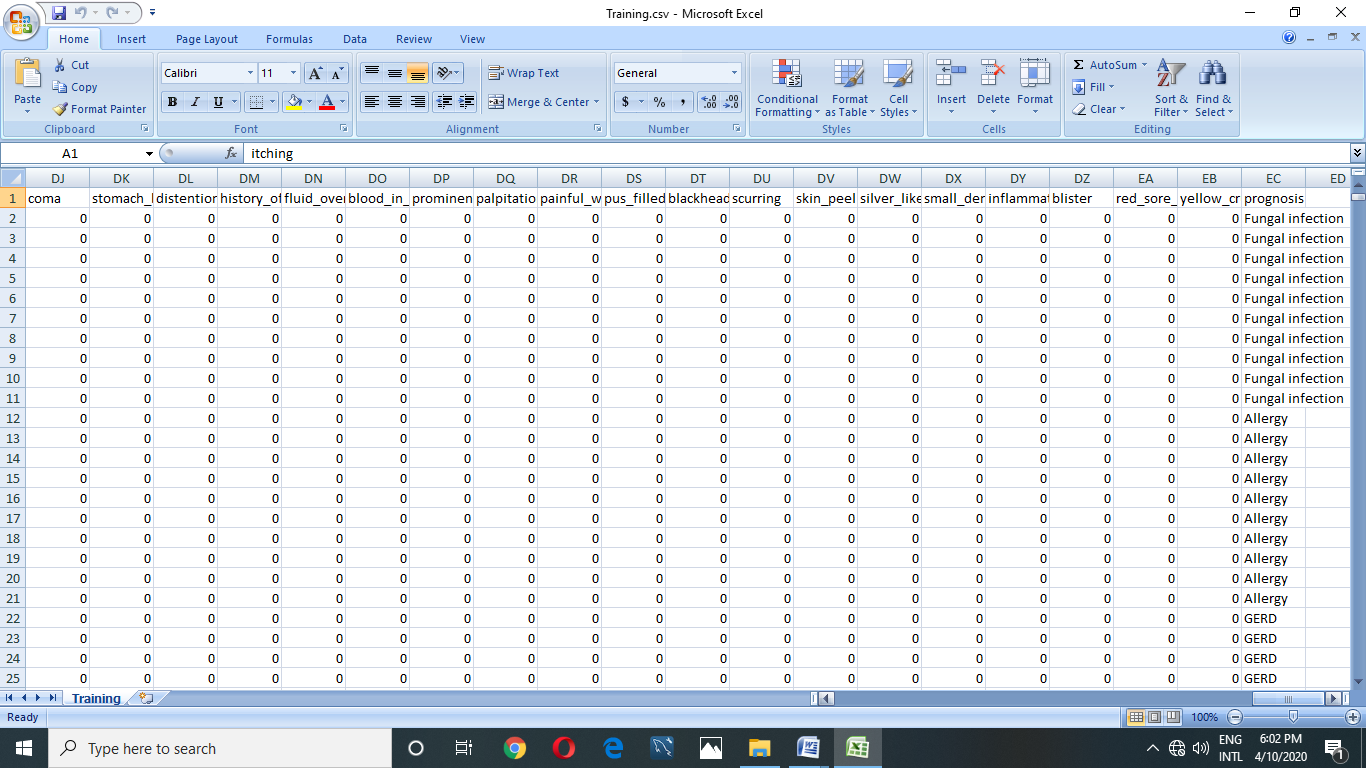
**Consistency**: To communicate a common meaning for all of the elements in the system.

Simply making sure that for all elements, the meaning will remain consistent.

**Reusability**: To Facilitates analysis of the details in order to evaluate characteristics and determine where system changes should be made and makes ease of use for developers.

**Diseases and their symptoms dataset**:-

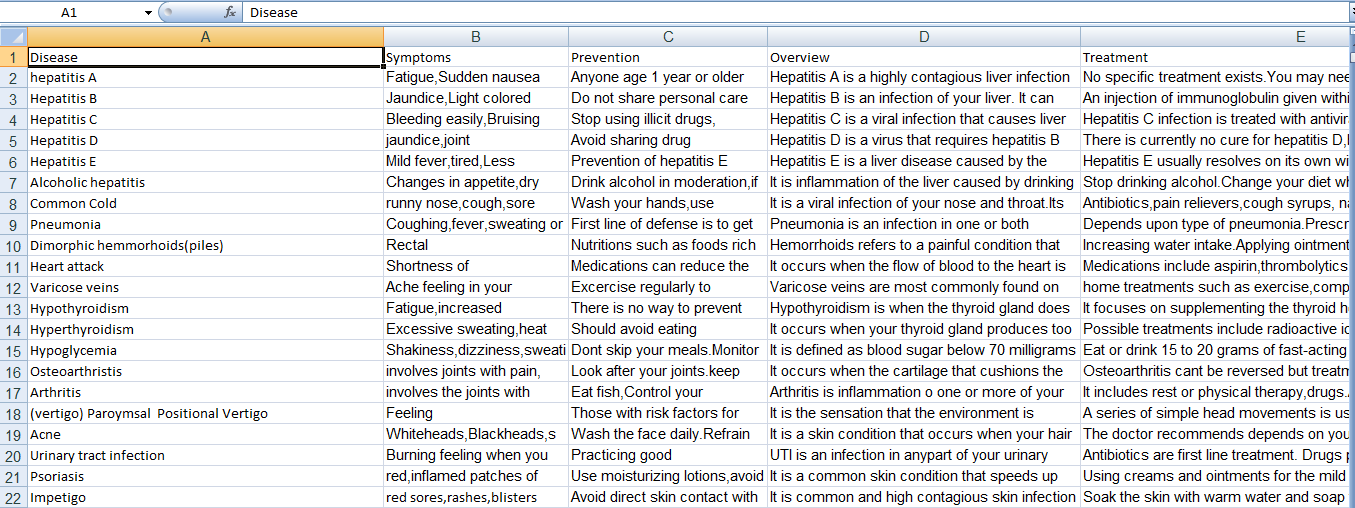
The dataset that we used in our project is taken from kaggle. The dataset is in the form of binary format. It consists of 4921x133 size of data. The data that we used for training the dataset contains the 42 diseases and 132 symptoms. Each row consists of Diseases and their symptoms. First 132 columns shows the symptoms in binary form and the last column shows the disease names.



**Fig 3.1 Diseases and their symptoms dataset**

**Disease information dataset:-**

The dataset is stored and fitted in a form of excel format. The dataset contains 5 attributes and 42 records(diseases) information. The labels or the elements that include in the dataset are disease, symptoms, prevention, overview, and treatment. The first column shows the disease name, the second column contains the symptoms that are associated the each disease, the third column contains the preventive measures to take for the particular disease, the fourth column contains the overview for the particular disease and the last columns contains treatment for the respective disease.



**Fig 3.2 Disease information dataset**

**3.2. Logical Database Design**

The data is stored in a tabular format for easy access and retrieval of the data. The data consists of following elements

**Data**: An 4921x133 size of data. Each row consists of Diseases and the Symptoms. The first 132 columns shows the symptoms that are associated with the respective disease and the last column shows the Disease name.

**Labels**: A list of 5 labels namely Disease names, symptoms, prevention, overview, treatment..

**Disease**: The disease names that are associated with the symptoms.

**Symptom**: The list of symptoms are associated with respective diseases.

**Prevention**: The preventive measures that should be taken for the particular disease.

**Overview**: The detailed information about the disease.

**Treatment**: The treatment for the respective disease.

**3.3. UML Diagrams**

Unified modeling language is the language used to visualize, specify, construct and document any component of software engineering.

The Unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows,

* User model view.
* Structural model view.
* Behavioral model view.
* Implementation model view.
* Environmental model view.

**User model view**: This view represents the system from the user's perspective. The analysis representation describes a usage scenario from the end users perspective.

**Structural model view**: This model the data and functionality are arrived from inside the system. This model view the static structures.

**Behavioral model view**: It represents the dynamic behavioral as parts of the system, depicting the interactions collections in between various structural elements described in the user model and structural model,

**Implementation model view**: In this structural and behavioral as parts of the system are represented as they are to be built.

**Environmental model view**: UML analysis modeling, which focus on the user model and structural model views, Behavioral and Implementation views.

Every complex system is best approached through a small set of nearly independent views of a model no single viewer is sufficient.

The UML diagrams include:

**State diagram:**

* Class diagram.
* Object diagram

**Dynamic diagram:**

* Use case diagram.
* Sequence diagram.

**Class diagram:** A class diagram shows a set of classes, interfaces, and collaborations and their relationships. These diagrams are the most common diagram found in modelling object-oriented systems, Class diagrams address the static design view of a system. Class diagrams that include active address the static process view of a system.

**Object diagram:** An object diagram shows a set of objects and their relationships. Objects diagrams represent static snapshots of instances of the things found in class diagram. These diagrams address the static view or static process view of a system as do class diagrams, but from the perspective of real or prototypical cases.

**Use case diagrams:** A use case diagrams shows a set of use cases and actors (a special kind of class) and their relationships. Use case diagram address the static view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system.

**Sequence diagram:** A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagrams shows a set of objects and the messages sent and received by those objects. Sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

**3.3.2 Use Case Diagrams**

**3.3.2.1 Use case diagram for entire system**

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform.



**Fig 3.3** Use case diagram for entire system

The above use case diagram represents the functionality of entire system. The system contains three actors patient, doctor and admin, where patient have the permissions to register the account, view profile, edit profile, change password, input symptoms, view predicted disease, book appointment, view appointment. In the same way, the doctor have the permissions to register the account, view and edit profile, view appointments. The admin have the permissions to upload training and testing datasets, add and delete doctors and patients and manages all the bookings.

**3.3.2.2 Use case for admin**

****

**Fig** 3.4 Use case diagram for admin

The use case diagram for admin describes the functionalities of the admin. The admin can login similarly like patient but admin gets redirected to admin panel where the admin can add doctors, patients, upload training and testing datasets, manage bookings, delete doctor and patients.

**3.3.2.3 Use case diagram for patient**

****

**Fig** 3.5 Use case diagram for patient

The above use case diagram describes about the functionality of the patient. Unlike admin patient cannot directly login into the system without registering. The patient needs to register initially before logging into the account. Once the patient logins after registration patient will be redirected to patient profile page where all the details of patient will be displayed. Here the patient can update their profile and change their password. The patient can input the symptoms, view the predicted disease and also book and view the appointments.

**3.3.2.4 Use case diagram for doctor**

****

**Fig** 3.6 Use case diagram for doctor

The above use case diagram describes about the functionality of the doctor. Unlike admin doctor cannot directly login into the system without registering. The doctor needs to register initially before logging into the account. Once the doctor logins after registration doctor will be redirected to doctor profile page where all the details of doctor will be displayed. Here the doctor can update their profile and change their password. The doctor can view the appointments made by the patients.

**3.3.3 Class Diagram**

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed. In the diagram, classes are represented with boxes that contain three compartments:



**Fig** 3.7 Class diagram for entire system

The above class diagram describes the functionality of the entire system. It contains five classes called admin, doctor, patient, prediction, appointments. The class diagram describes structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

**3.3.4 Sequence diagrams:**

A type of interaction diagram, a sequence diagram shows the actor of the object participating in an interaction and the events they generate arranged in a time sequence. Often a sequence diagram shows the events the results from particular instance of a use case but a sequence diagram represents time, with time preceding down the page the horizontal dimensions represent different actors.

**3.3.4.1 Sequence diagram for entire system**

****

**Fig 3.8** Sequence diagram for entire system

**3.3.4.2 Sequence diagram for admin**



**Fig 3.9** Sequence diagram for admin

The above sequence diagram describes the functionality between the user interface and admin. It describes the sequence of messages and interactions that happen between user interface and admin.

**3.3.4.3 Sequence diagram for patient**



**Fig 3.10** Sequence diagram for patient

The above sequence diagram describes the functionality between the user interface and patient. It describes the sequence of messages and interactions that happen between user interface and patient.

**3.3.4.4 Sequence diagram for doctor**



**Fig** 3.11 Sequence diagram for doctor

The above sequence diagram describes the functionality between the user interface and doctor. It describes the sequence of messages and interactions that happen between user interface and doctor.

4**. SYSTEM IMPLEMENTATION**

**4.1 SELECTED SOFTWARE:**

**Software**

* PyCharm Editor

**Environment**

* Open Source Environment
* Django

**Technologies**

* HTML5
* CSS3
* BOOTSTRAP
* JAVASCRIPT
* MATERIALIZECSS
* JQUERY
* PYTHON3.8

**Libraries or Dependencies**

* NumPy
* Pandas
* Scikit-learn

**4.1.1 PyCharm Editor**

PyCharm editor is a powerful tool for creating and modifying source code. As any other IDE editor, it supports basic features like bookmarks, breakpoints, syntax highlighting, code completion, zooming, folding code blocks, etc. There are however, Plenty of advanced features like macros, highlighted TODO items, code analysis, intention actions, intelligent and fast navigation, and a lot more.

To configure your editing environment, use the editor settings page and its child pages. There is also a Quick Switch Scheme command that lets you change color schemes, themes, key maps, etc. with a couple of keystrokes. The editor is tab-based. All operations with the editor tabs are available from the context menu of a tab, or from window | Editor tabs node of the main menu.

When you open a file for editing, It opens in its own tab. The editor you are currently working in, is the active editor. You can change behavior of the active editor using the commands under View | Active Editor node of the main menu.

**1.Editor area**

Use this area to type and edit your source code. The editor suggests numerous coding assistance facilities. Refer to the sections under this node for details.

**2.Gutter area**

The left gutter provides additional information about your code and displays the various icons that identify the code structure, bookmarks, breakpoints, scope indicators, change markers and the code folding lines that let you hide arbitray code blocks. You can change the behavior of the left gutter. For example, it’s possible to make the left gutter thinner by hiding the gutter icons. This is done either for the active editor, or for all the newly created editors.

**3.Smart completion pop-up**

This is one of the key editing assistance features that suggests method names, functions, tags and other keywords you are typing.

**4.Document tabs**

Enable quick navigation across the multiple documents you are working on. Clicking a tab brings its contents to front and makes it available for editing in the active editor.

**5.Validation side bar / marker bar**

This is the bar to the right from the editing area, showing the green, red or yellow box on its top depending on whether your code is okay, or contains errors or warnings. This bar also displays active red, yellow, white, green and blue navigation stripes that let you jump exactly to the erroneous code, changed lines, search results, or TODO items.

**4.1.2 Django installation**

Django, also known as the ‘web framework for perfectionists with deadlines’ is taking the software & tech industry by storm. It is an intricate, advanced level Python web framework encouraging rapid development alongside clean, logical design and thinking.

**4.1.2.1 Features of developing with Django**

**Fast**

Django was designed with the intention to help developers travel from a concept to a finished project as quickly as possible. When speaking about speed one has to consider performance. Django partakes in lower memory consumption and fewer demands on the database/network, becoming much less of a burden than many other frameworks out there.

**The Complete Package**

It comes with an object-relational mapper in which user describe their database layout in python code. More than that, the data-model syntax provides many rich ways of representing your models. The migrate command glances over all available models and builds tables in your database for whichever ones that don’t exist. What’s more? It optionally provides richer schema control as well. In a nutshell, Django takes care of user authentication, site maps, content administration, RSS feeds and much more with simple commands.

**Highly Secure**

Its user authentication system provides a safe way to manage user accounts and passwords alongside ensuring developers stray away from making common mistakes such as: cross-site scripting, cross-site request, click jacking and forgery.

**Widely Scalable**

Some of the most crowded, busiest sites on the planet use Django framework.

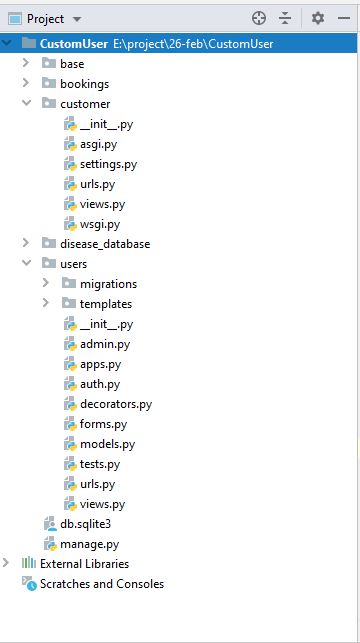


Fig 4.2: Arrangement of Django files

**4.1.9 Python 3:**

Python was developed by Guido van Rossum in early 1990’s and its latest version is 3.8.1, we can simply call it as python 3. Python 3.0 was released in 2008, it is a interpreted language i.e its not compiled and the interpreter will check the code line by line. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

**4.1.10 Packages used in python**

**Numpy**: It is the library for the python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into numeric , with extensive modifications. Num array is a open source software and has many contributors.

**Pandas: Pandas Data Frame** is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas Data Frame consists of three principal components, the**data, rows,** and **columns**. In the real world, a Pandas Data Frame will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, and Excel file. Pandas Data Frame can be created from the lists, dictionary, and from a list of dictionary etc.

**Scikit-learn:** Scikit-learn is probably the most useful library for machine learning in python. It is on Numpy, Scipy and matplot lib, this library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimenisionality reduction.

**Naïve Bayes Classifier**:

The proposed framework utilizes a Machine learning strategy "Naive Bayes classifier" for the development of the expectation framework. This framework includes a higher number of data indexes and characteristics which are legitimately gathered from specialist's data for the exact expectation of the symptom."Naive Bayes or Bayes" Rule is the reason for some, AI and information mining strategies. The standard is utilized to make models with prescient capacities. It gains from the "proof" by figuring the connection between's the objective (i.e., subordinate) and other factors

End

Find its probability against all output

Start

Attributes will belong to class variable with maximum value

Select the maximum value of probability

Multiply all the probabilities W.r.t. each attributes

For each input attribute

**Fig 4.1 Flowchart for Naïve Bayes**

***Naïve Bayes Algorithm:***

Following advances are actualized in Bayes calculation:-

Bayes’ Theorem : P(c | x) = P ( x | c) P(c) / P (x)

Where,

P (c | x) = Posterior Probability

P (c) = Prior probability

P (x | c) = probability of predictor

P(x) = Predictor’s prior probability

A lot of cases was taken in program was prepared with the data indexes to such an extent that the probabilities of the considerable number of classes with every one of the conditions were determined. Result was accumulate in database and when the test information was given we got the probabilities for the distinctive classes for the given side effect esteems based on which we infers that the patient fell into the class with the most elevated likelihood. Thus it is Naïve Bayes‟ order. By utilizing this stream diagram we can without much of a stretch presume that the patient has been experienced specific sickness or not .we will test this from the approaching ascribe which has a place with class variable with most extreme esteem. We initially process all conceivable individual probabilities adapted on the objective quality of specific illness contained all probabilities of trait of that malady. Register the conceivable probabilities for all condition choose that the p has part up into two cases one for Y and second for N. Subsequently, on the off chance that the contention of likelihood of P1 is more prominent than P2, at that point patient isn't having the illness. Our system shows maximum of 3 disease with their prediction accuracy which is higher than 20%.

**Implementation of Naïve Bayes using SKlearn library:**

For working of Naive Bayes algorithm in our project as of the work-flow shown in above section we have implemented some classed and functions, some of them are predefined which are provided by the **Sklearn** library and some of them are user defined.

The first function call will be from views to the **GenerateMlModel()** function

Which passes the list of user provided symptoms.

Function Call:

predictions = GenerateMLModel(symptom\_list)

Function Definition:

def GenerateMLModel(user\_symptom\_list):

This function processes the dataset file and converts it into ML Model which will be later used to make prediction.

Function used to generate model which is saved into variable gnb :

gnb = GaussianNB()

Then this model will be trained with Training dataset:

gnb.fit(Features\_Train, Target\_Train)

Then for prediction another function is called with parameters user symptoms, trained model and disease list. Which returns prediction result in the form of dictionary:

disease\_dict = DoPrediction(user\_symptom\_array, gnb, diseases)

Function definition:

def DoPrediction(features\_predict, gnb, diseases):

The main function which does the prediction:

predicts = gnb.predict\_proba(features\_predict)

The above function which is method of Gaussian Naive Bayes algorithm does the prediction where gnb in previously trained model. The method **predict\_proba** gives the all target class and the probability of each class. Then we are sorting those class based on probability with highest first. But we doesn’t need all the class so we are only returning top 3 prediction with probability in the form of dictionary.

d = dict()

for i in range(3):  
 if dicts[i][1] >= 0.2:  
 d[dicts[i][0]] = round(dicts[i][1] \* 100)  
return d

**4.2 Sample Code**

1. **Code for Disease Prediction**
   1. **Generate Gaussian Naive Bayes Model from Training Dataset**

**def GenerateMLModel(user\_symptom\_list):**  
# get all the item from DiseaseDatabaseModel  
ddb\_obj = DatabaseFiles.objects.all()  
 if ddb\_obj.filter(File\_Type='TestingFile') and ddb\_obj.filter(File\_Type='TrainingFile'):  
train\_obj = ddb\_obj.filter(File\_Type='TrainingFile').values('File\_Path')  
test\_obj = ddb\_obj.filter(File\_Type='TestingFile').values('File\_Path')  
train\_file\_path = str(train\_obj[0]['File\_Path'])  
test\_file\_path = str(test\_obj[0]['File\_Path'])  
# access by file reference  
train\_file = open('disease\_database/database\_files/'+train\_file\_path)  
test\_file = open('disease\_database/database\_files/'+test\_file\_path)  
train\_result = pd.read\_csv(train\_file)  
test\_result = pd.read\_csv(test\_file)  
Features\_Train = train\_result.drop(['prognosis'], axis=1).values  
Target\_Train = train\_result['prognosis'].values  
# get headings as numpy array  
Features\_Headings = train\_result.columns[:-1].tolist()  
 for i in range(len(Features\_Headings)):  
 if Features\_Headings[i] in user\_symptom\_list:  
 Features\_Headings[i] = 1  
 else:  
 Features\_Headings[i] = 0

# convert\_1D list to 2D list because predict function only takes 2D numpy array  
 Features\_Headings\_2D = [Features\_Headings]  
 # print(Features\_Headings\_2D)  
user\_symptom\_array = np.asarray(np.asarray(Features\_Headings\_2D))  
# print(type(Features\_Headings), len(Features\_Headings))  
 # build model  
gnb = GaussianNB()  
 # Train model  
gnb.fit(Features\_Train, Target\_Train)  
  
 # all the diseases that model can predict(41)  
 diseases = np.unique(Target\_Train)  
disease\_dict = DoPrediction(user\_symptom\_array, gnb, diseases)  
 return disease\_dict

* 1. **Make prediction with ML Model and user given symptoms in binary form**

**def DoPrediction(features\_predict, gnb, diseases):**  
 # predicting with probability for each disease  
 predicts = gnb.predict\_proba(features\_predict)  
  
 # create dictionary with key as disease and probability as value {'disease': 0.5}  
 # predict\_proba gives nested array,  
 # when single row of data provided, we have to access the internal array  
pred\_dict = dict(zip(diseases, predicts[0]))  
 # sort dictionary based on probability  
dicts = sorted(pred\_dict.items(), key=lambda x: x[1], reverse=True)  
  
 d = dict()  
 # print only top 3 disease with probability higher than 0.30  
 for i in range(3):  
 if dicts[i][1] >= 0.2:  
 print(dicts[i][0], ' --> Accuracy: ', round(dicts[i][1] \* 100), '%')  
 /’ d[dicts[i][0]] = round(dicts[i][1] \* 100)  
 return d

* 1. **Show results View**

**def disease\_list\_view(request):**  
 predictions = {}  
 if request.is\_ajax() and request.method == 'POST':  
symptom\_list = json.loads(request.POST.get('symptom\_list'))  
 print(symptom\_list)  
 predictions = GenerateMLModel(symptom\_list) # supply user symptoms as parameters  
 if not predictions:  
 # if dictionary is empty means one of the file is missing  
messages.error(request, 'Missing Files in DB')  
 else:  
 # predicted dictionary is here  
messages.success(request, 'Successfully predicted')  
 else:  
messages.success(request, 'This is not ajax request')  
 return render(request, 'ajax\_prediction\_result.html', {'predictions': predictions})

1. **Booking Appointment** 
   1. **Book new appointment**

**def bookingView(request):**  
specialist\_list = list(ExtendedDoctorsDetail.objects.values\_list('specialization', flat=True).distinct())  
cities\_list = list(User.objects.filter(is\_doctor=True).values\_list('city', flat=True).distinct())  
 if request.method == 'POST' and 'search\_doctors' in request.POST:  
 specialist = request.POST['specialist']  
 city = request.POST['city']  
doctors\_extd = ExtendedDoctorsDetail.objects.filter(specialization=specialist,  
 doctor\_\_city\_\_icontains=city).values('doctor',  
 'specialization')  
doctors\_base = User.objects.filter(doctor\_detail\_\_specialization=specialist,  
 city\_\_icontains=city).values('id', 'email', 'name',  
 'address', 'city',  
 'pin', 'state')  
 for doctor\_e in doctors\_extd:  
 for doctor\_b in doctors\_base:  
 if doctor\_e['doctor'] == doctor\_b['id']:  
doctor\_b.update(doctor\_e)  
filtered\_results = doctors\_base  
 return render(request, 'bookings.html',  
 {'spcs': specialist\_list, 'cities': cities\_list, 'doctors': filtered\_results,  
 'time\_slots': time\_list()})  
elifrequest.method == 'POST' and 'confirm-booking-button' in request.POST:  
 date = request.POST.get('selected\_date')  
 time = request.POST.get('selected\_time')  
 print(date, time)  
booked\_doctor\_mail = request.POST.get('confirm-booking-button')  
doctors\_extd = ExtendedDoctorsDetail.objects.filter(doctor\_\_email=booked\_doctor\_mail).values('doctor',  
 'specialization')  
doctors\_base = User.objects.filter(email=booked\_doctor\_mail).values('id', 'name',  
 'address', 'city')  
doctors\_extd = doctors\_extd[0]  
doctors\_base = doctors\_base[0]  
 if Booking.objects.filter(doctor\_email\_\_icontains=booked\_doctor\_mail, booked\_date\_\_icontains=date, booked\_time\_\_icontains=time).count() < 1:  
 book = Booking()  
book.p\_id = request.user.id  
book.patient\_name = request.user.name  
book.patient\_email = request.user.email  
book.d\_id = doctors\_extd['doctor']  
book.doctor\_name = doctors\_base['name']  
book.doctor\_email = booked\_doctor\_mail  
book.city = doctors\_base['city']  
book.specialization = doctors\_extd['specialization']  
book.booked\_date = date  
book.booked\_time = time  
book.save()  
messages.success(request,  
 'Successfully Booked Appointment. For date ' + date + ' with Doctor ' + doctors\_base[  
 'name'])  
 return redirect('bookings:show\_appointment\_patient')  
 else:  
messages.error(request, 'Time slot '+time+' is is not available, for Dr. '+doctors\_base["name"]+', Please select another slot !!')  
 context = {  
 'spcs': specialist\_list,  
 'cities': cities\_list,  
 }  
 return render(request, 'bookings.html', context)

* 1. **Display appointment**

**def ShowAppointmentsDoctors(request):**  
u\_email = request.user.email  
 bookings = Booking.objects.filter(doctor\_email=u\_email).values()  
 return render(request, 'show\_appointment\_doctor.html', {'bookings': bookings})

**5. TESTING**

**5.1 Introduction**

The purpose of testing is discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a word 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 the unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**5.1.1 Types of Tests**

* **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 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 specifcally aimed at exposing the problems that arise from the combination of components.

* **Functional testing**

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

Functional testing is centered 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.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current test determined.

* **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, emphasizing pre-driven process links and integration points

* **white Box Testing**

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

* **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings structure or language of the module being tested. Black box tests, as most other kinds of tests must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software gender test is treated, as a black box you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

* **Stress Testing**

Stress testing is the testing to evaluate how system behaves under unfavorable conditions. Testing is conducted at beyond limits of the specifications. It falls under the class of black box testing.

* **Performance Testing**

Performance testing is the testing to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing

* **Acceptance Testing**

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

* **Regression Testing**

Regression testing is the testing after modification of a system, component, or group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

* **Beta Testing**

Beta testing is the testing which is done by end users, a team outside development, or publicly releasing full pre-version of the product which is known as beta version. The aim of beta testing is to cover unexpected errors. It falls under the class of black box testing.

* **Usability testing**

Usability testing is performed to the perspective of the client, to evaluate how the is user friendly? How easily can the client learn? After leaming how to use, how proficiently can the client perform? How pleasing is it to use its design?

**5.2 Test cases**

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected result |
| EDP\_TC01 | Valid Email  Valid Password | A valid email and valid password given by the user | Redirected to user profile page |
| EDP\_TC02 | Blank Email  Blank Password | A blank email and password given by the user | Provide email and password |
| EDP\_TC03 | Valid Email  Blank Password | A valid email and blank password given by the user | Provide password |
| EDP\_TC04 | Blank Email  Valid Password | A blank email and valid password given by the user | Provide email |
| EDP\_TC05 | Invalid Email  Valid Password | A invalid email and valid password given by the user | Provide valid email(invalid login) |
| EDP\_TC06 | Valid Email  Invalid Password | A valid email and password given by the user | Provide valid password(Incorrect password) |

Table 5.1 : Login page

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result |
| EDP\_TC01 | Valid Name  Valid Email  Valid Password  Valid Pin | A Valid Name, Email, Password, Pin given by patient | Redirected to Login Page |
| EDP\_TC02 | Blank Name  Blank Email  Blank Password  Blank Pin | A Blank Name, Email, Password, Pin given by the patient | Provide Name, Email, Password, Pin |
| EDP\_TC03 | Blank Name  Valid Email  Valid Password  Valid Pin | A blank Name and Valid Email, Password, Pin given by patient | Provide Name |
| EDP\_TC04 | Valid Name  Invalid Email  Valid Password  Valid Pin | A Valid Name, Invalid Email, Valid Password and Pin given by patient | Enter a Valid Email |
| EDP\_TC05 | Valid Name  Valid Email  Invalid Password  Valid Pin | A Valid Name, Email, Invalid Password and Valid Pin is given by patient | Password should be 8 characters long |
| EDP\_TC06 | Valid Name  Valid Email  Valid Password  Invalid Pin | A Valid Name, Email, Password and Invalid Pin is given by the Patient | Enter Valid Pin |

Table 5.2 Register Page for Patient

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result |
| EDP\_TC01 | Valid Name  Valid Email  Valid Password  Valid Specialization | A Valid Name, Email, Password,  Specialization | Redirected to Login Page |
| EDP\_TC02 | Blank Name  Blank Email  Blank Password  Blank Specialization | A Blank Name, Email, Password, Specialization given by the Doctor | Provide Name, Email, Password, Specialization |
| EDP\_TC03 | Blank Name  Valid Email  Valid Password  Valid Specialization | A blank Name and Valid Email, Password, Specialization given by Doctor | Provide Name |
| EDP\_TC04 | Valid Name  Invalid Email  Valid Password  Valid Specialization | A Valid Name, Invalid Email, Valid Password and Specialization given by Doctor | Enter a Valid Email |
| EDP\_TC05 | Valid Name  Valid Email  Invalid Password  Valid Specialization | A Valid Name, Email, Invalid Password and Valid Specialization is given by Doctor | Password should be 8 characters long |
| EDP\_TC06 | Valid Name  Valid Email  Valid Password  Invalid Specialization | A Valid Name, Email, Password and Invalid Specialization is given by the Doctor | Enter Valid Specialization |

Table 5.3 Register Page for Doctor

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result |
| EDP\_TC01 | Input real time symptoms | Dataset contains all real time symptoms | Successful add the symptoms |
| EDP \_TC02 | Input invalid symptoms | Dataset contains all real time symptoms | Generates an error as all the symptoms in the dataset are real time symptoms |
| EDP \_TC03 | Input one symptom | Enter only one symptom | The system will ask to enter atleast 4 symptoms |
| EDP \_TC04 | Input two symptoms | Enter the two symptoms | The system will ask to enter atleast 4 symptoms |
| EDP \_TC05 | Input three symptoms | Enter the three symptoms | The system will ask to enter atleast 4 symptoms |
| EDP \_TC06 | Input four symptoms | Enter the four symptoms | Successful prediction of disease |
| EDP \_TC07 | Input duplicate symptoms | Enter the same symptoms twice | Generates an error |
| EDP \_TC08 | Input five symptoms | Enter the five symptoms | Successful prediction of disease |

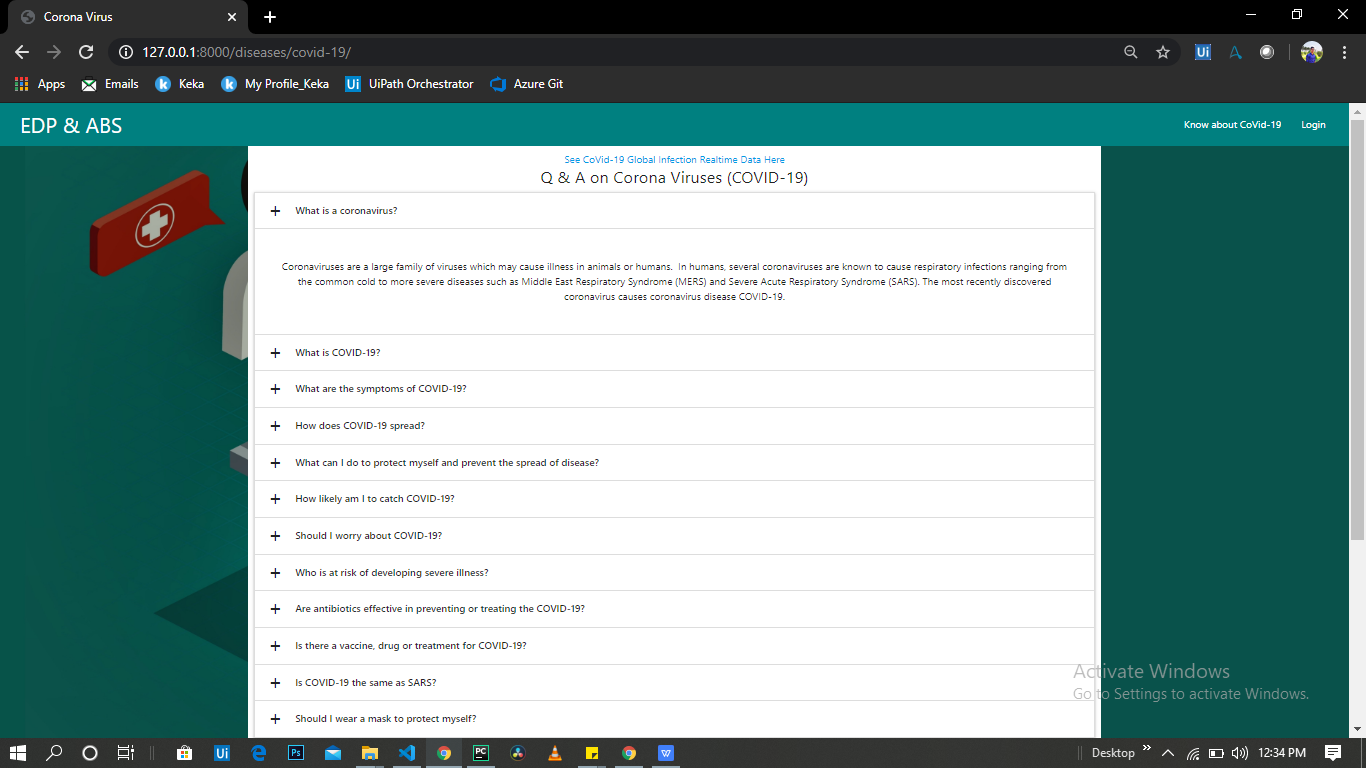
Table 5.4 Prediction Page

|  |  |  |  |
| --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result |
| EDP\_TC01 | Valid specialization, valid city, valid date, valid time | A Valid specialization, city, date and time given by the patient | Redirect to confirmation page |
| EDP\_TC02 | Blank specialization, Blank city, Blank date, Blank time | A Blank specialization, city, date and time given by the patient | Please fill out the fields |
| EDP\_TC03 | Valid specialization, Invalid city, valid date, valid time | A Valid specialization, date ,time and invalid city is given by the patient | No doctor available in this city |
| EDP\_TC04 | Valid specialization, valid city, Invalid date, valid time | A Valid specialization, city ,time and the slot is given after 30 days from that day | Generates an Error |
| EDP\_TC05 | Valid specialization, valid city,valid date, Invalid time | A valid specialization, city, date and the slot is given after 6 pm | Generates an Error |
| EDP\_TC06 | Valid specialization, valid city,valid date, Invalid time | A valid specialization, city, date and the slot is given before 6 am | Generates an Error |
| EDP\_TC07 | Valid specialization, valid city,valid date, Invalid time | A valid specialization, city, date and the time slot already booked for this doctor is given by patient | No slot available for this doctor |
| EDP\_TC08 |  |  |  |

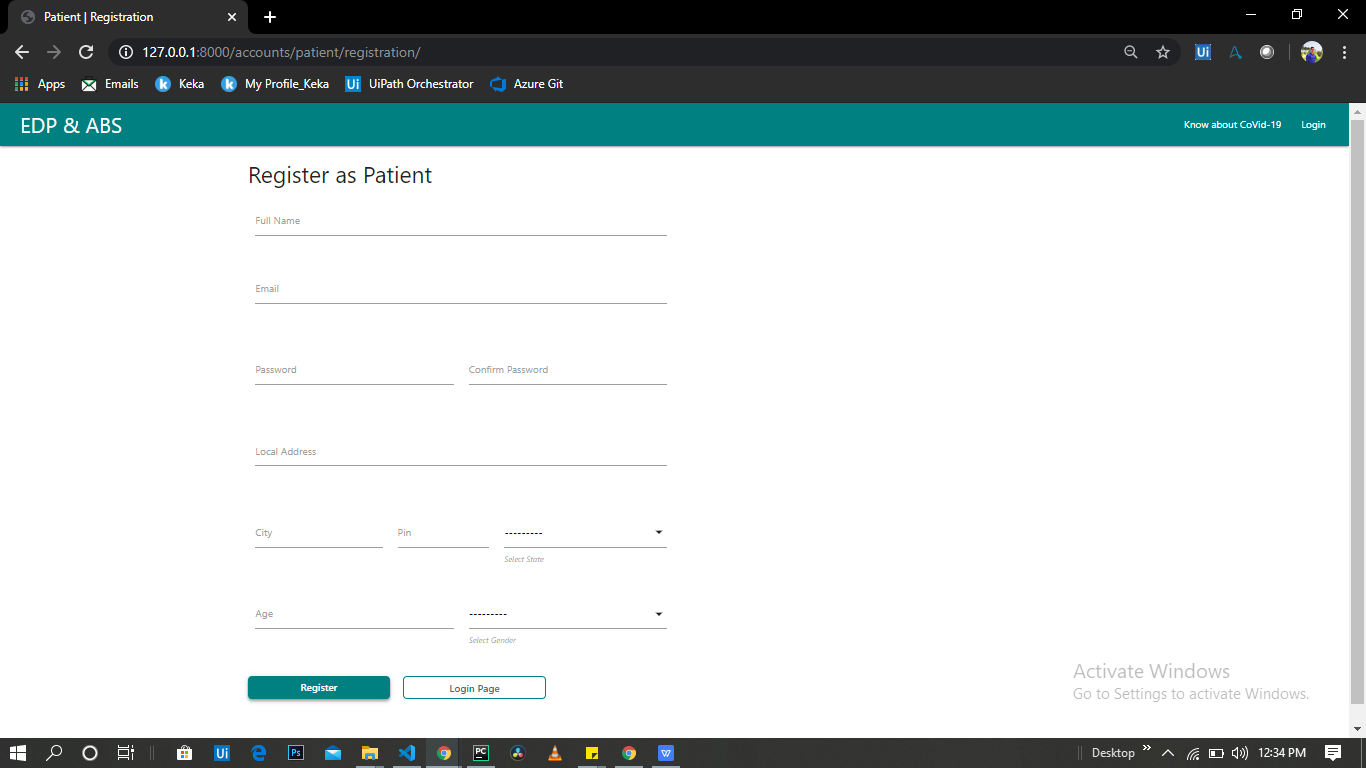
Table 5.5 Appointment Booking Page

**6. SCREENSHOTS AND REPORTS**

**6.1 Screenshots**

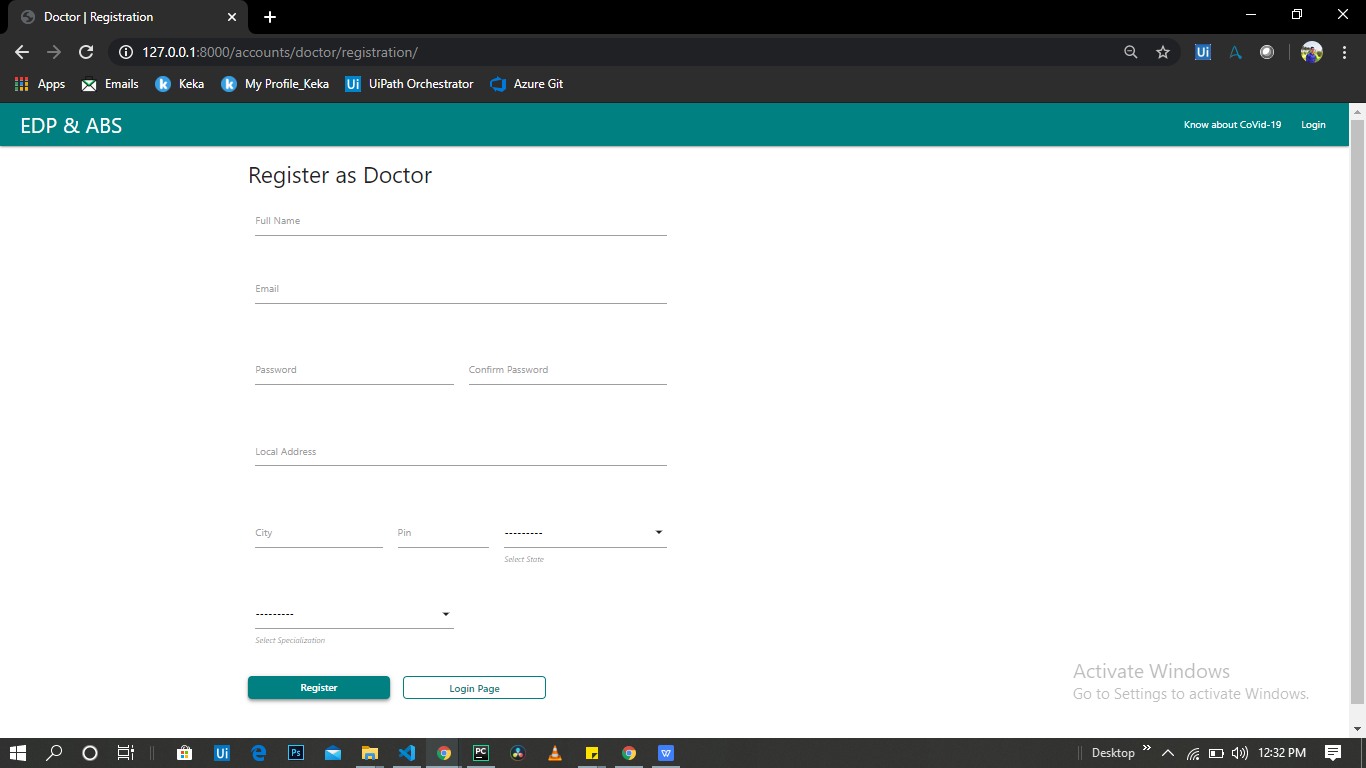
****

**Screen 6.1: Details about Coronavirus (COVID-19) from WHO page**

* It gives general details about Coronavirus epidemic, what it is? And How it transmits? And other questions also.

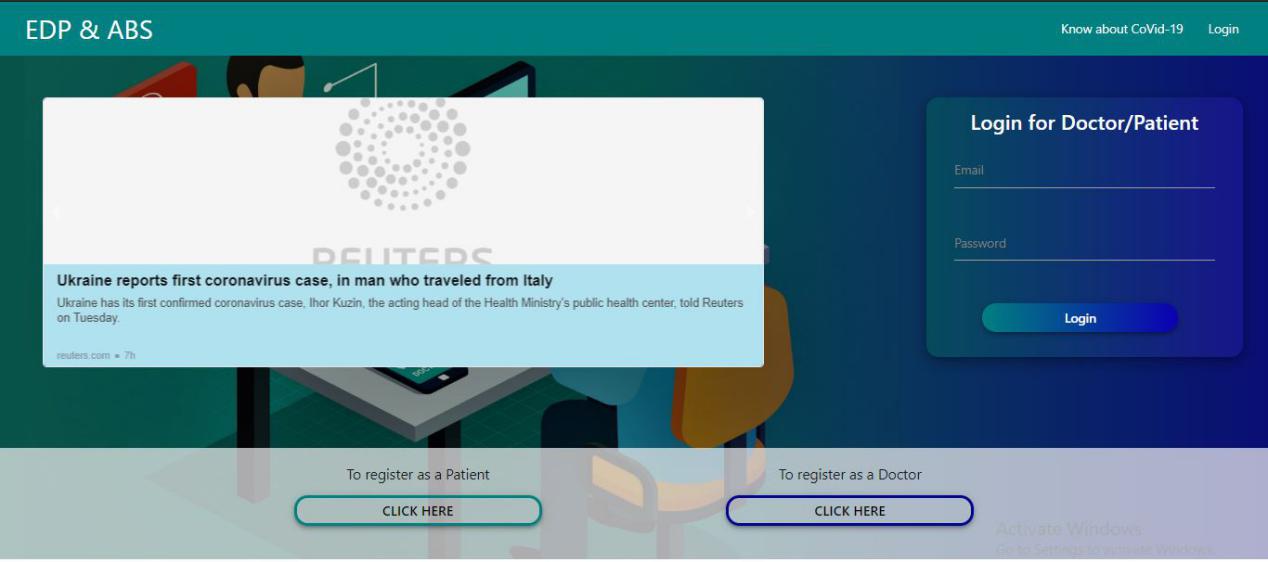
**Screen 6.2: Registration page for Patient**

* It takes all the user details and stores in database as user type **Patient.**
* All the input fields are required.
* It has front-end validation.



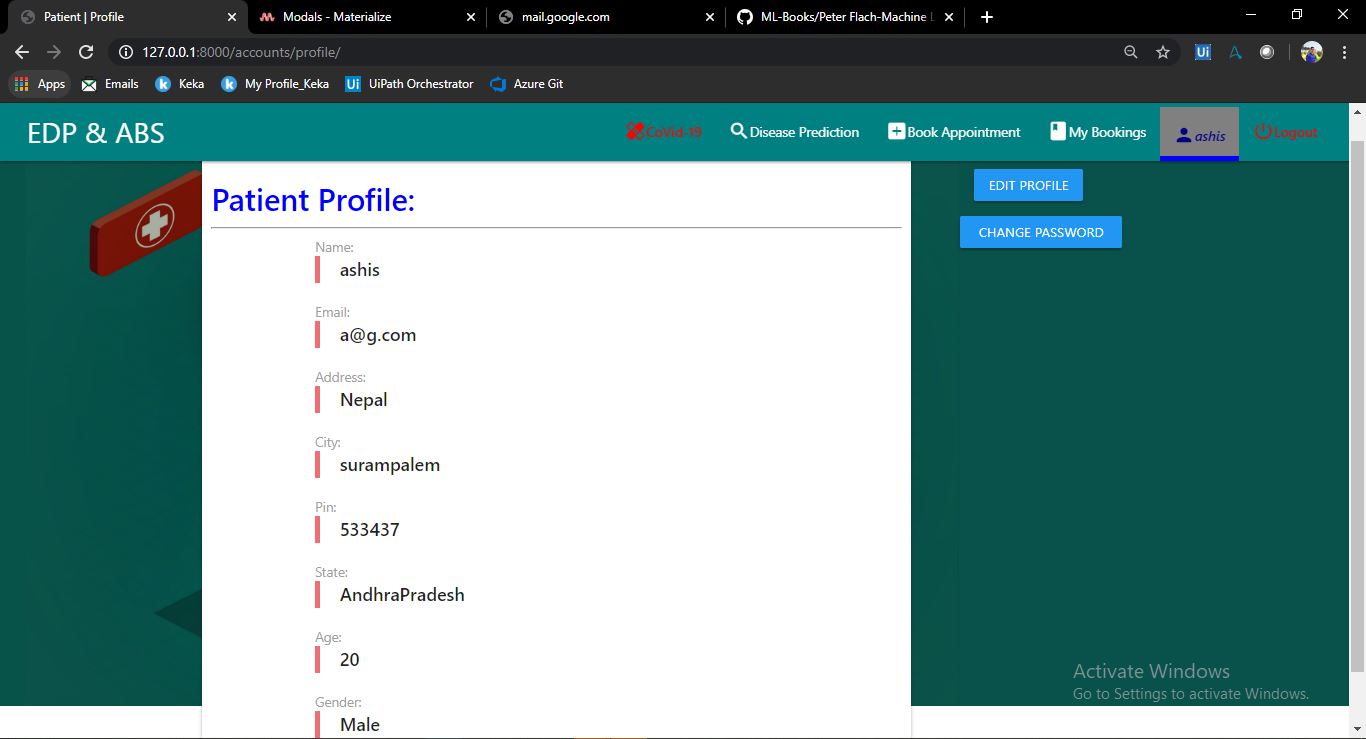
**Screen 6.3: Registration page for Doctor**

* It takes all the user details and stores in database as user type **Doctor.**
* All the input fields are required.
* It has front-end validation.



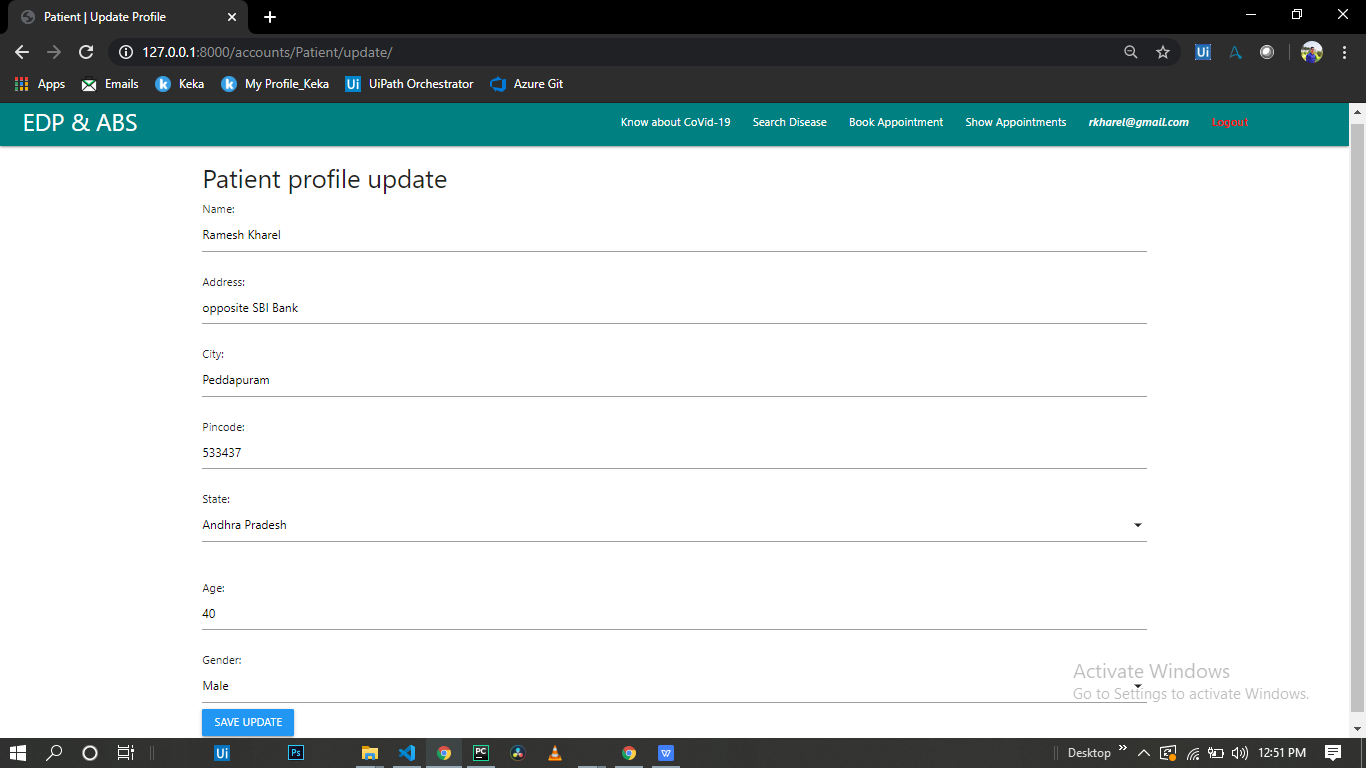
**Screen 6.4: Home and Login Page**

* Here we also have RSS news feed from Google Health News.
* The login form is same for doctor and patient, based on the registered type we redirect them to respective page.
* Link for Registration page also in the bottom side.

****

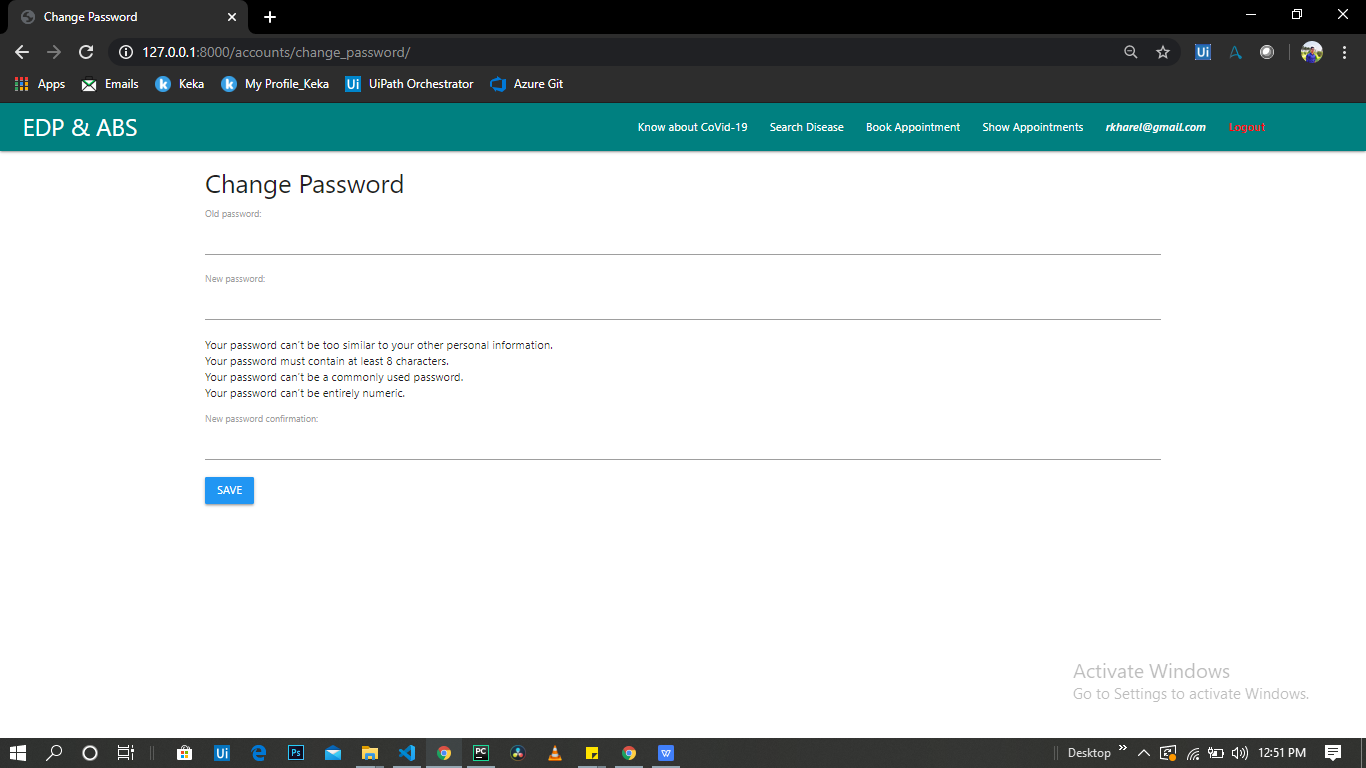
**Screen 6.5: Profile Page**

* This page layout is same for Patient and Doctor.
* From here we can change the details and password also.



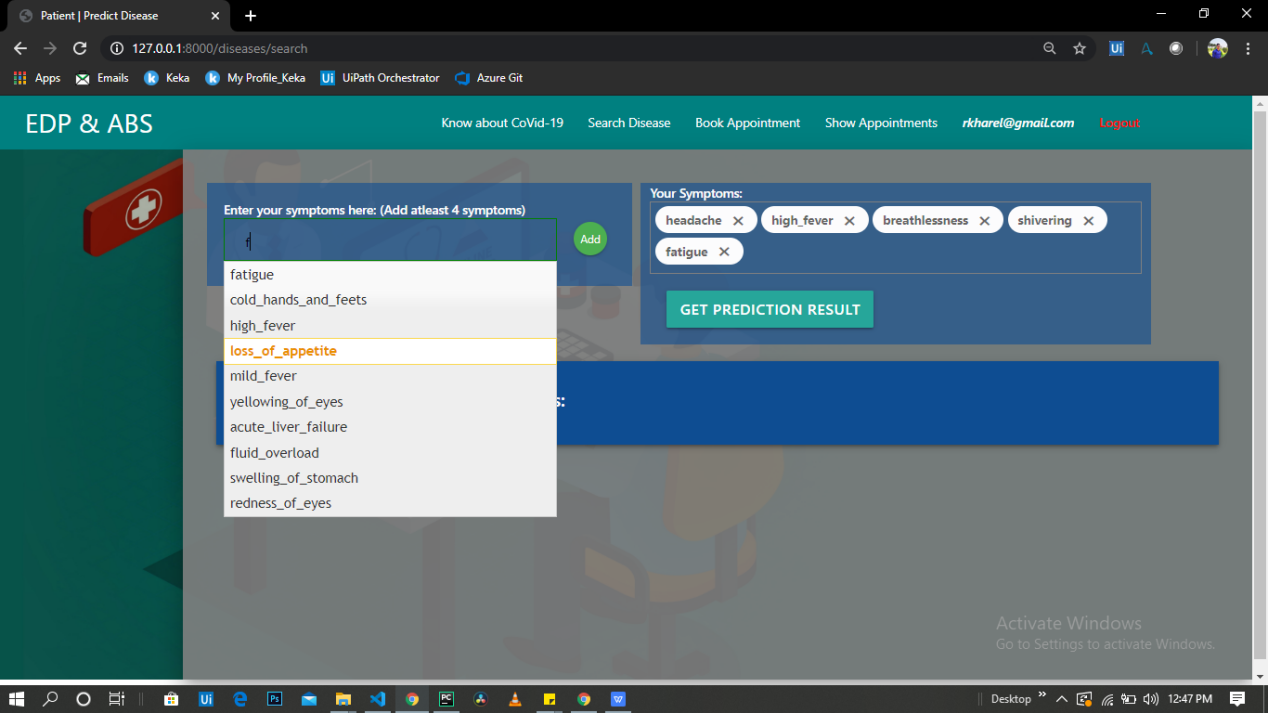
**Screen 6.6: Update Profile Details Page**

* Here you can update details except Email because its primary key in our database.



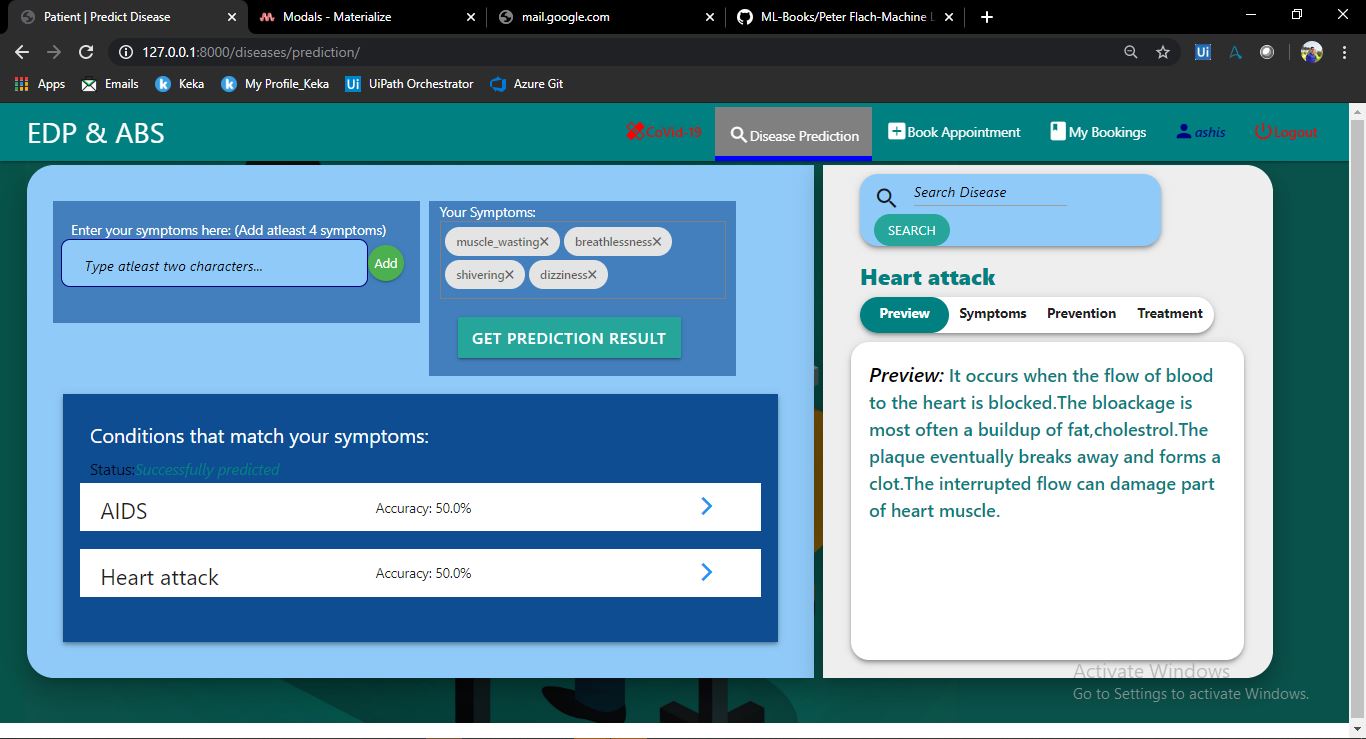
**Screen 6.7: Change Password Page**

* You can change your password by providing Old password.
* Here also all the password constraints should be passed.



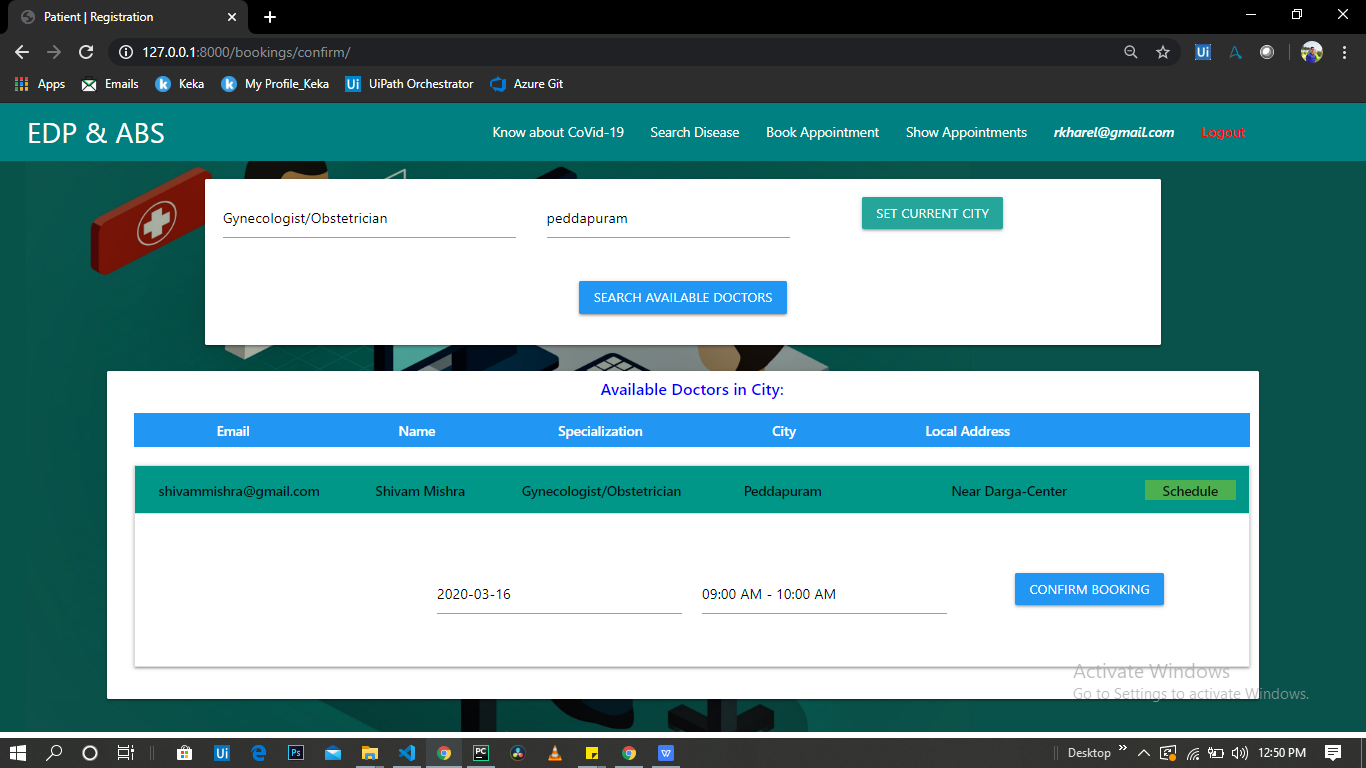
**Screen 6.8: Symptoms Input Page**

* It has symptom list from db on Auto-complete drop-down input.
* You have to provide at least 4 symptoms to get the prediction result.

****

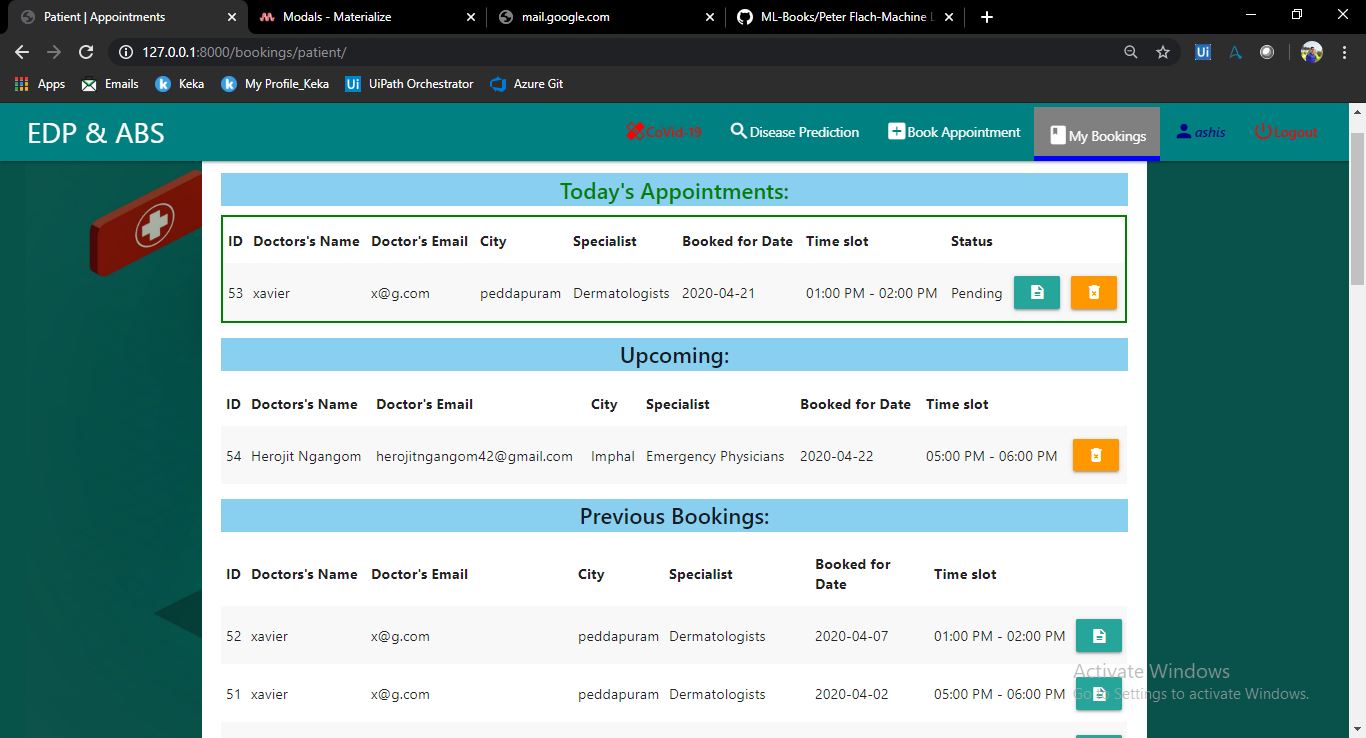
**Screen 6.9: Prediction Result Display Page**

* It shows maximum of 3 disease with their prediction accuracy which is higher than 20%.
* Here we can search for the overall description for predicted disease.



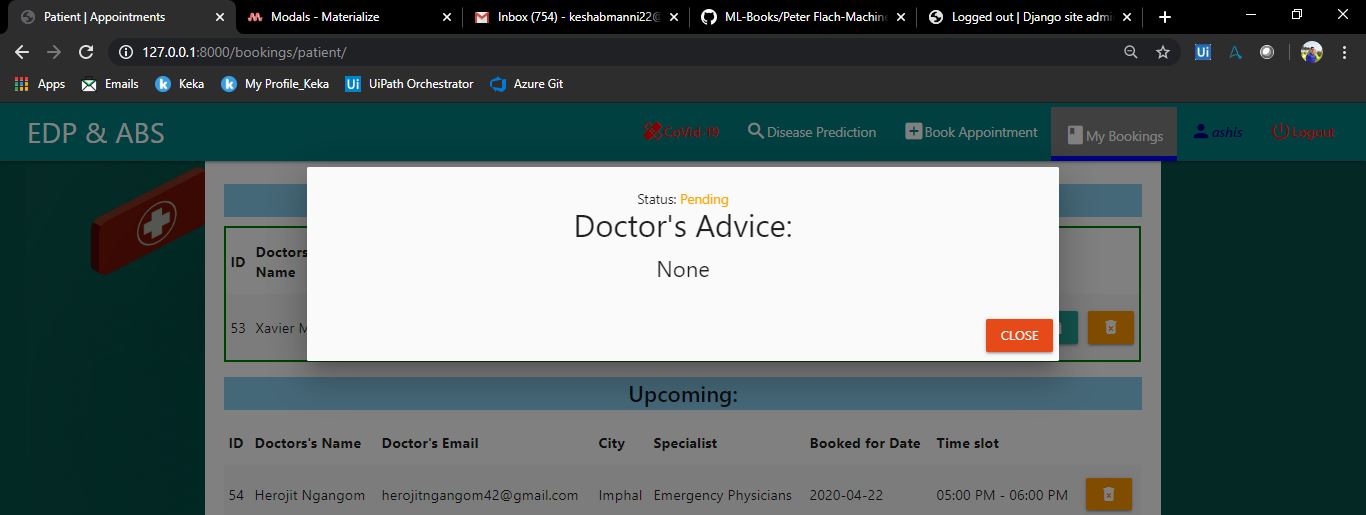
**Screen 6.10: Appointment Booking Page for Patient**

* Here you can book appointment with doctors by City and Specialization.
* You cam select any of the available date and time slot.
* Dates are from today to next 30 days.
* Time slots are from morning 8AM to evening 6PM with each 1 Hour slot.



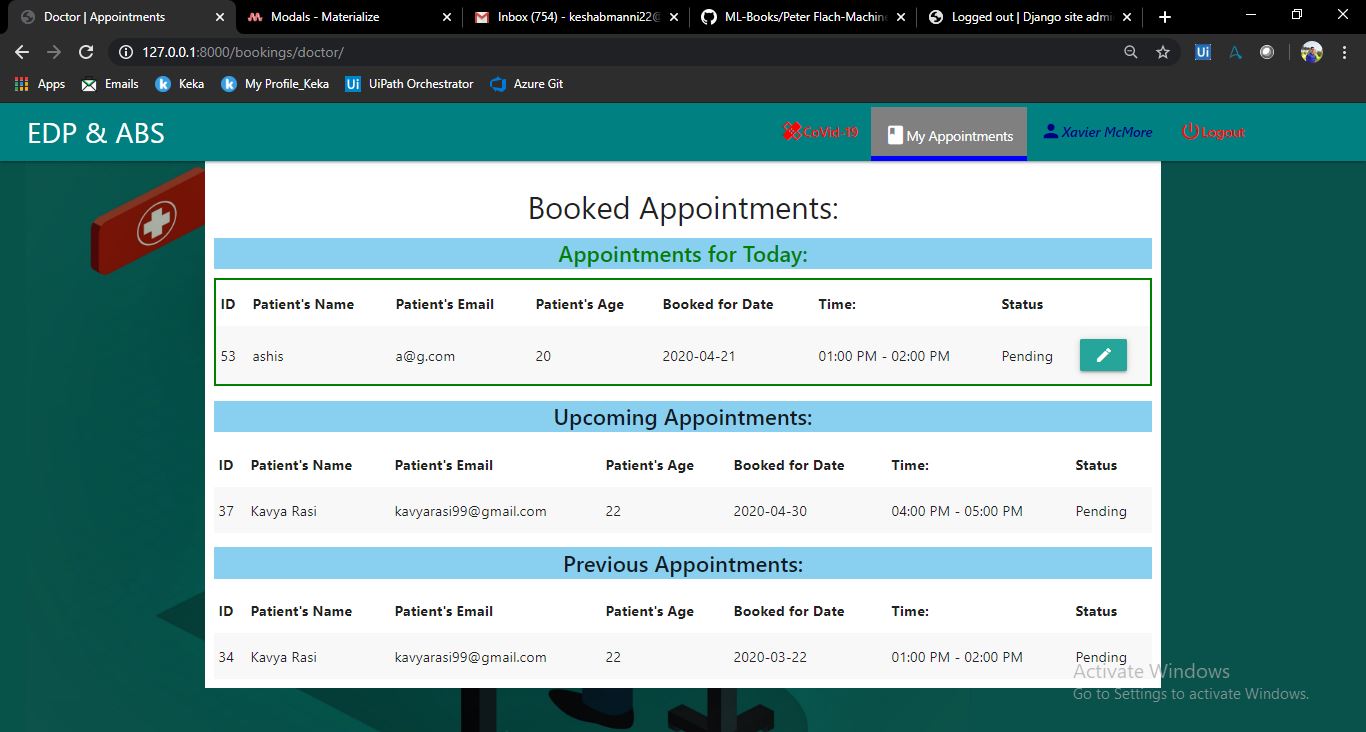
**Screen 6.11: Booked Appointments List Page for Patient**

* Here patient can see all his booked appointments with doctor and booking details.

****

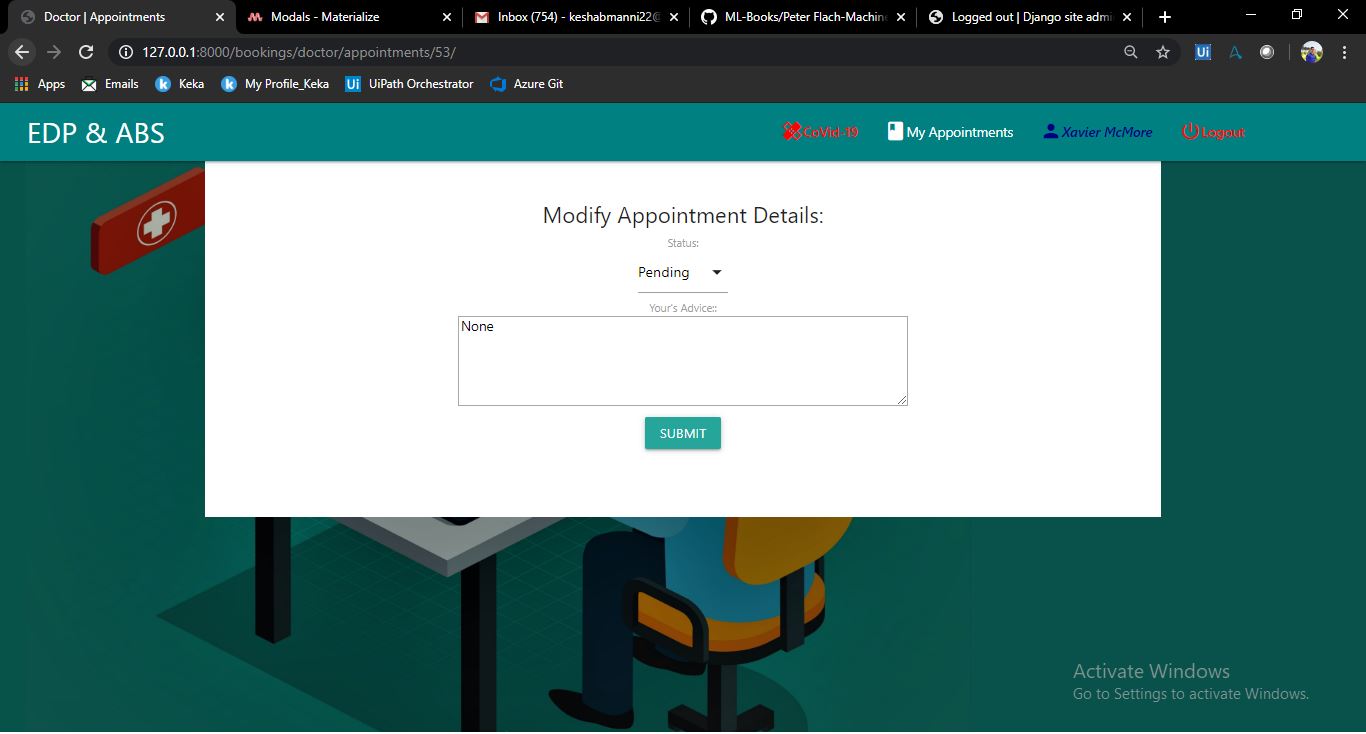
**Screen 6.12: Appointment status for Patient**

* Here the patient can see the appointment status details.
* The patient can able to see doctors advice.



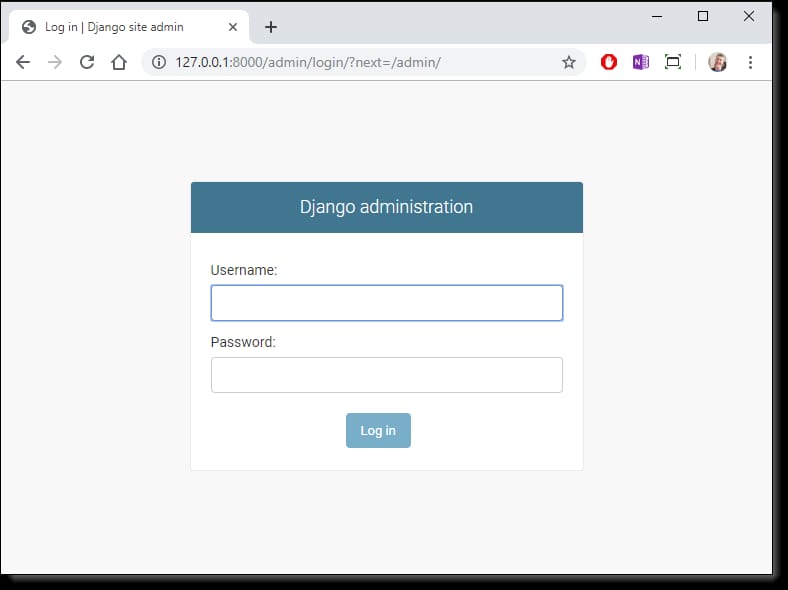
**Screen 6.13: Booked Appointments List Page for Doctor**

* Here doctor can see all the appointments booked for him/her with patient and booking details.

****

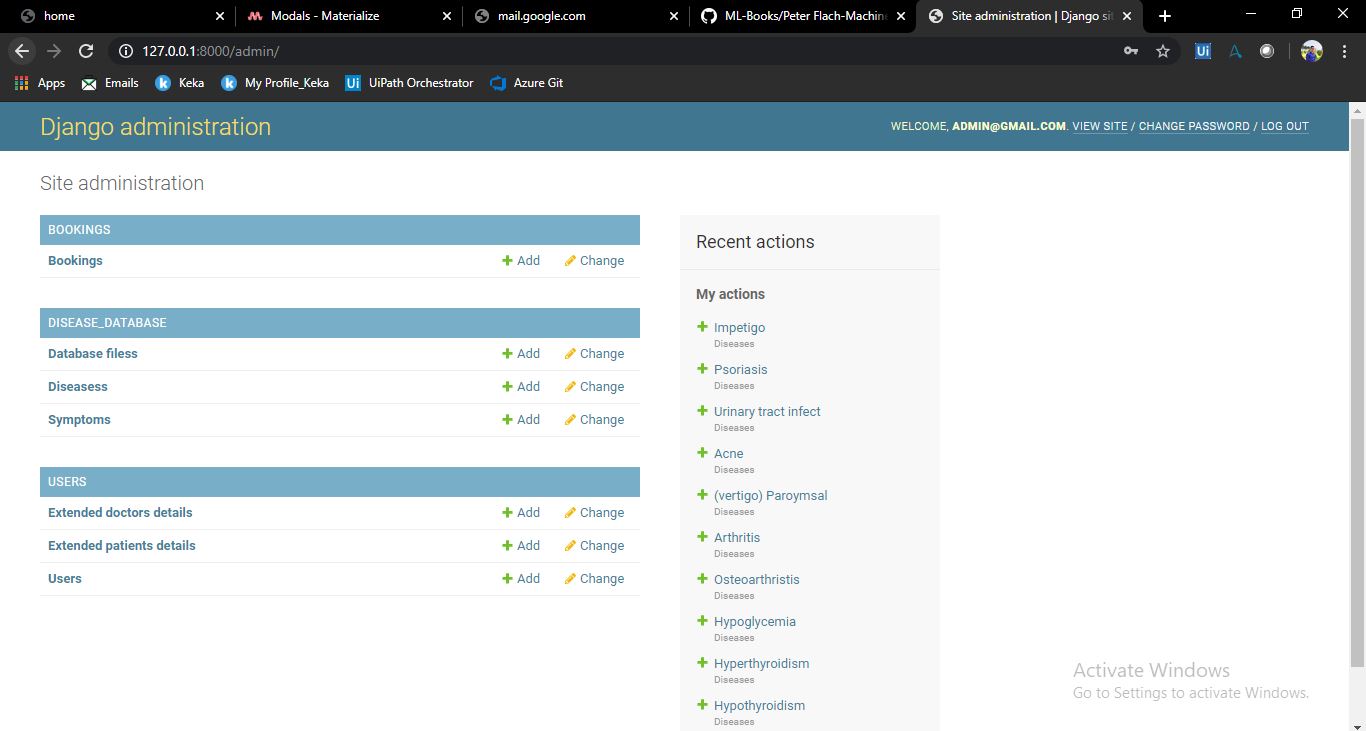
**Screen 6.14: Modify Appointment details Page for Doctor**

* Here in this page the doctor can change the appointment status and can write advice to concerned patients.

****

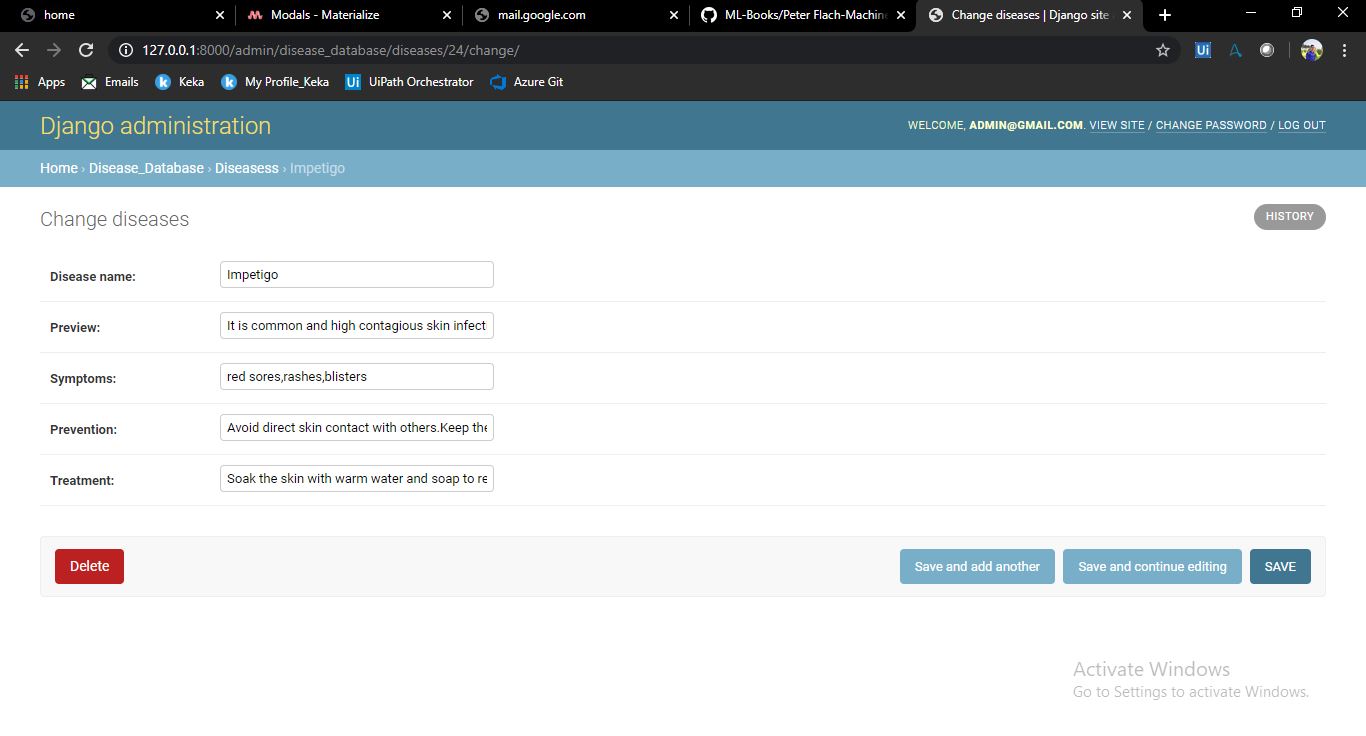
**Screen 6.15: Admin interface**

* Django provides a ready-to-use user interface for administrative activities. We all know how an admin interface is important for a web project. Django automatically generates admin UI based on your project models.

****

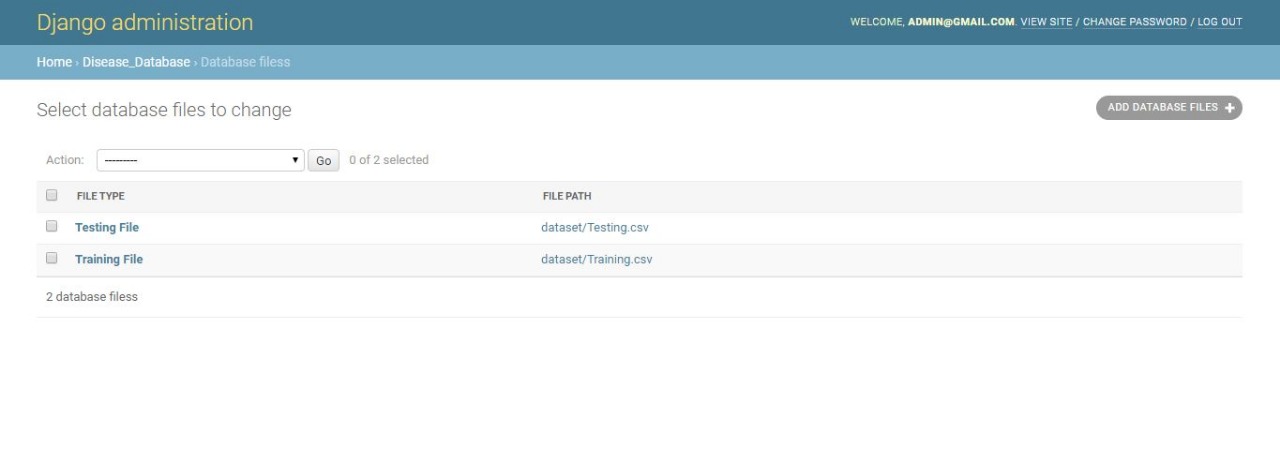
**Screen 6.16: Data base home page**

* This interface will let you administrate Django groups and users, and all registered models in the application. The interface gives you the ability to do at least the "CRUD" (Create, Read, Update, Delete) operations on your models.

****

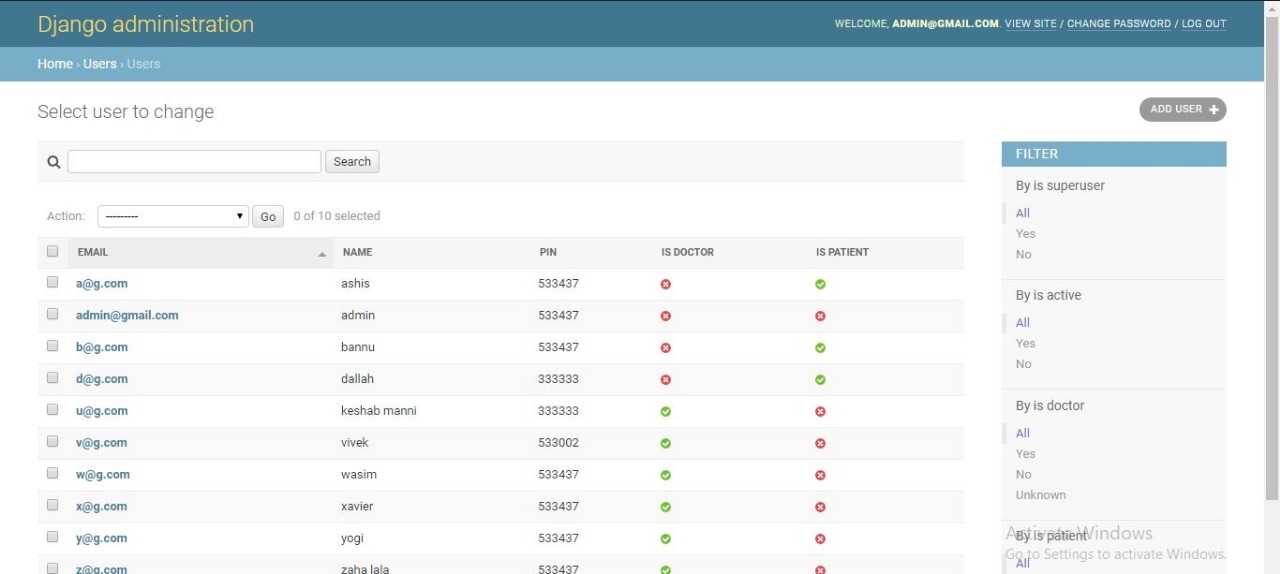
**Screen 6.17: Disease information page**

* Here the admin can able to view and change the disease information like disease name, preview, symptoms, prevention and treatment.

****

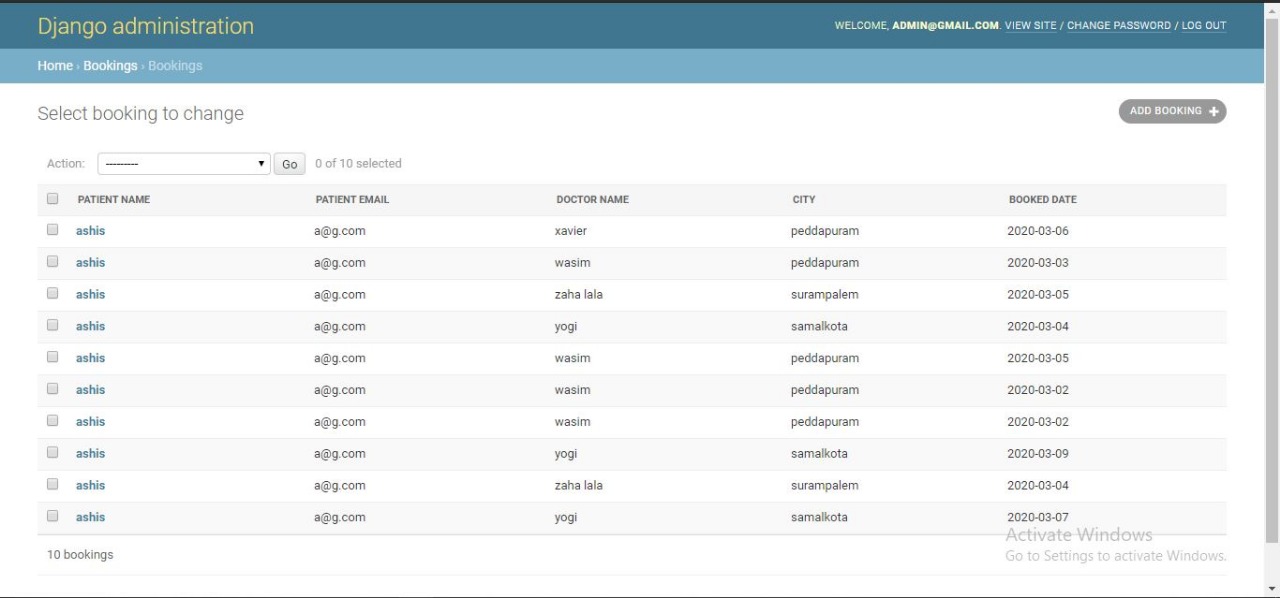
**Screen 6.18: database files page**

* Here in this page it stores the database files that we used for our application like testing and training files.

****

**Screen 6.19: User’s Table page**

* Here the users table stores the all the details of users like the Email, Name, Pin, it stores whether the user is patient or doctor.

****

**Screen 6.20: Booking Table page**

* Here the booking table stores the booked appointment details like patient name, patient email, doctor name, city and the booked date.

**6.2 Reports**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected result | Report |
| EDP\_TC01 | Valid Email  Valid Password | A valid email and valid password given by the user | Redirected to user profile page | Pass |
| EDP\_TC02 | Blank Email  Blank Password | A blank email and password given by the user | Provide email and password | Pass |
| EDP\_TC03 | Valid Email  Blank Password | A valid email and blank password given by the user | Provide password | Pass |
| EDP\_TC04 | Blank Email  Valid Password | A blank email and valid password given by the user | Provide email | Pass |
| EDP\_TC05 | Invalid Email  Valid Password | A invalid email and valid password given by the user | Provide valid email(invalid login) | Pass |
| EDP\_TC06 | Valid Email  Invalid Password | A valid email and password given by the user | Provide valid password(Incorrect password) | Pass |

Table 6.1 Login page

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result | Report |
| EDP\_TC01 | Valid Name  Valid Email  Valid Password  Valid Pin | A Valid Name, Email, Password,  Pin | Redirected to Login Page | Pass |
| EDP\_TC02 | Blank Name  Blank Email  Blank Password  Blank Pin | A Blank Name, Email, Password, Pin given by the patient | Provide Name, Email, Password, Pin | Pass |
| EDP\_TC03 | Blank Name  Valid Email  Valid Password  Valid Pin | A blank Name and Valid Email, Password, Pin given by patient | Provide Name | Pass |
| EDP\_TC04 | Valid Name  Invalid Email  Valid Password  Valid Pin | A Valid Name, Invalid Email, Valid Password and Pin given by patient | Enter a Valid Email | Pass |
| EDP\_TC05 | Valid Name  Valid Email  Invalid Password  Valid Pin | A Valid Name, Email, Invalid Password and Valid Pin is given by patient | Password should be 8 characters long | Pass |
| EDP\_TC06 | Valid Name  Valid Email  Valid Password  Invalid Pin | A Valid Name, Email, Password and Invalid Pin is given by the Patient | Enter Valid Pin | Pass |

Table 6.2: Registration page for patients

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result | Report |
| EDP\_TC01 | Valid Name  Valid Email  Valid Password  Valid Specialization | A Valid Name, Email, Password,  Specialization | Redirected to Login Page | Pass |
| EDP\_TC02 | Blank Name  Blank Email  Blank Password  Blank Specialization | A Blank Name, Email, Password, Specialization given by the Doctor | Provide Name, Email, Password, Specialization | Pass |
| EDP\_TC03 | Blank Name  Valid Email  Valid Password  Valid Specialization | A blank Name and Valid Email, Password, Specialization given by Doctor | Provide Name | Pass |
| EDP\_TC04 | Valid Name  Invalid Email  Valid Password  Valid Specialization | A Valid Name, Invalid Email, Valid Password and Specialization given by Doctor | Enter a Valid Email | Pass |
| EDP\_TC05 | Valid Name  Valid Email  Invalid Password  Valid Specialization | A Valid Name, Email, Invalid Password and Valid Specialization is given by Doctor | Password should be 8 characters long | Pass |
| EDP\_TC06 | Valid Name  Valid Email  Valid Password  Invalid Specialization | A Valid Name, Email, Password and Invalid Specialization is given by the Doctor | Enter Valid Specialization | Pass |

Table 6.3: Registration page for doctors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result | Report |
| EDP\_TC01 | Input real time symptoms | Dataset contains all real time symptoms | Successful add the symptoms | Pass |
| EDP \_TC02 | Input invalid symptoms | Dataset contains all real time symptoms | Generates an error as all the symptoms in the dataset are real time symptoms | Pass |
| EDP \_TC03 | Input one symptom | Enter only one symptom | The system will ask to enter atleast 4 symptoms | Pass |
| EDP \_TC04 | Input two symptoms | Enter the two symptoms | The system will ask to enter atleast 4 symptoms | Pass |
| EDP \_TC05 | Input three symptoms | Enter the three symptoms | The system will ask to enter atleast 4 symptoms | Pass |
| EDP \_TC06 | Input four symptoms | Enter the four symptoms | Successful prediction of disease | Pass |
| EDP \_TC07 | Input duplicate symptoms | Enter the same symptoms twice | Generates an error | Pass |
| EDP \_TC08 | Input five symptoms | Enter the five symptoms | Successful prediction of disease | Pass |

Table 6.4: Prediction page

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Testcase Id | Input | Description | Expected Result | Report |
| EDP\_TC02 | Blank specialization, Blank city, Blank date, Blank time | A Blank specialization, city, date and time given by the patient | Please fill out the fields | Pass |
| EDP\_TC03 | Valid specialization, Invalid city, valid date, valid time | A Valid specialization, date ,time and invalid city is given by the patient | No doctor available in this city | Pass |
| EDP\_TC04 | Valid specialization, valid city, Invalid date, valid time | A Valid specialization, city ,time and the slot is given after 30 days from that day | Generates an Error | Pass |
| EDP\_TC05 | Valid specialization, valid city,valid date, Invalid time | A valid specialization, city, date and the slot is given after 6 pm | Generates an Error | Pass |
| EDP\_TC06 | Valid specialization, valid city, valid date, Invalid time | A valid specialization, city, date and the slot is given before 6 am | Generates an Error | Pass |
| EDP\_TC07 | Valid specialization, valid city, valid date, Invalid time | A valid specialization, city, date and the time slot already booked for this doctor is given by patient | No slot available for this doctor | Pass |

Table 6.5: Appointment Booking Page

**7. CONCLUSION AND FUTURE SCOPE**

**7.1 Conclusion**

The ENHANCED DISEASE PREDICTION USING MACHINE LEARNING AND DOCTOR APPOINTMENT SYSTEM aims to predict the disease on the basis of symptoms and book the doctor’s appointment according to their preference. The Appointment booking process, has become a necessary burden in medical offices, healthcare facilities and wellness faced medical workers. This can work as per the requirements, specifications, and conditions mentioned in this documentation. In the proposed system, hidden knowledge will be extracted from the historical data by preparing datasets by applying Naïve Byes algorithm. The project is designed in such a way that the system takes symptoms from the user as input and produces output i.e. predict disease. The system is fed with various symptoms and the disease/symptoms associated with those symptoms. This system allows users to share their symptoms. If then processes user’s symptoms to check for various illnesses that could be associated with it. The system also allows users to search the doctors based on the specialization and city and can book the appointment as per the convenient dates and time slot. People always try to refer to the internet if any problem arises. People have access to internet than hospitals and doctors. The proposed system is aimed at simplifying the task of the patient and doctor. It will reduce long waiting time for patients and eradicate long queue. Patient also have freedom to fix their appointment and also book appointment according to their preference. The system will deliver timely and convenient access to health services for all patients. It reduces the stress of a patient and it also provide better guidance to the person who used the system. The system is feasible and can be easily understand by anyone.

**7.2 Future Scope**

Asa future work, we may do research on many various diseases and involve their symptoms for the prediction in our application. This web application can be additionally upgraded in an android application. This will be accessible to clients on versatile premise and its utilization can be additionally expanded. Likewise, highlight like getting the specialist online on a visit with the goal that patient can straightforwardly converse with the concerned specialists. Also, features like medical history of patients, geography can be include in our application. We can work on various machine learning algorithms for higher accuracy.

**BIBILOGRAPHY**

The knowledge required for developing this project is extracted from the following

* <https://www.webmd.com/>
* <https://stackoverflow.com/>
* <https://python-course.eu/>
* <https://www.djangoproject.com/>
* <https://materializecss.com/>
* <https://www.kaggle.com/>

**Base papers:-**

* [1]. Min Chen, Yixue Haol, Kai Hwang, Lu Wangi and Lin Wang, "Disease detection by Machine Learning Over Big Data From Healthcare Communities," IEEE 2017,vol.19,no.2,pp.171-209.
* [2]. P. Siriyasatien, S. Cahdsuthi, K. Jampachaisri, and K. Kesorn, "Dengue Epidemics detection: A survey of the State-of-the-Art based on Data Science Processes." IEEE 2018,vol.17,no.166,pp.122-185.
* [3]. Somaya Hashem , Gamal Esmat, Wafaa Elakel, Shahira Habashy, Safaa Abdel Raouf, Mohamed Elhefnawi, Mohamed I. Eladawy and Mahmoud ElHefnaw. “Comparison of Machine Learning Approaches for Prediction of Advanced Liver Fibrosis in Chronic Hepatitis C Patients”, IEEE/ACM transactions on computational Biology and Bioinformatics, vol.15, no.3,pp.64-135, 2018.
* [4]. A. Shah, G. Escobar, International Research Journal of Engineering and Technology (IRJET) –“Disease prediction using Machine Learning algorithms”,Volume: 06 Issue: 05,vol.2,pp.164-168.2019.
* [5]. D. W. Bates, S. Saria and L. Ohno-Machado ,“Big data in health care: using analytics to identify and manage high-risk and high-cost patients,” Health Affairs, vol.33.pp.123-13.
* [6]. Subasish Mohapatra, Prashanta Kumar Patra, Subhadarshini Mohanty, Bhagyashree Pati, International Conference on Information Technology (ICIT) 2018-“ Smart Health Care System using Data Mining”.