# PROJECT REPORT ON

# Disease Detection System

*Submitted in partial fulfillment of the requirements for the award of the degree of*

## BACHELOR OF COMPUTERAPPLICATIONS



**Batch: 2020 - 23**

***Under the Guidance of Submitted By***

**Dr. Ruchi Sawhney** **Ajit Ekka 01128402020,**

**Aadarsh Prem 00128402020**

**Bosco Technical Training Society**

**Don Bosco Technical School, Okhla Road, New Delhi-110025**

**9643868820, 8527787221, 011-41033889**

**Affiliated to**



### Guru Gobind Singh Indraprastha University, Delhi

Sector - 16C, Dwarka, New Delhi - 110078 (India)

Phone: +91-11-25302170, Fax: +91-11- 25302111

E-Mail Id: ggsipu.pr@rediffmail.com, pro@ipu.ac.in

**CERTIFICATE**

This is to certify that this project entitled Disease Detection System is a qualify work done by Aadarsh Prem and Ajit Ekka completed under my supervision and guidance in partial fulfillment of the requirement for the award of degree of **Bachelor of Computer Application (BCA)** as a part of the curriculum bearing Course Code BCA-335 in **Guru Gobind Singh Indraprastha University, New Delhi-110078.**

Name of Guide

**Dr. Ruchi Sawhney**

Countersigned by:

Director,

Place:

Date:

**ACKNOWLEDGEMENT**

The note starts with thanks to Almighty who actually created this piece of work and helped us when things were not easy for us.

I am very grateful and indebted to my Guide Dr. Ruchi Sawhney who immensely helped and rendered their valuable advice, precious time, knowledge and relevant information regarding the collection of material. They had been a major source of inspiration throughout the project as they not only guided me throughout the Major Project on Disease Detection System but also encouraged me to solve problems that arose during this project.

Their guidance and suggestions about this Major Project have really enlightened me. It has been a great help to support to have them around.

And finally, I would like to mention appreciation to our parents and friends who have been instrumental throughout this period by providing unrelenting encouragement.

Name of Students Enrollment No. Signature

1. Ajit Ekka 01128402020

2. Aadarsh Prem 00128402020

**ABSTRACT**

Despite advances in modern medicine, diseases such as dengue, cancer, diabetes, and others continue to plague the human race. As we get older, our health becomes an important asset to us, and the decisions we make about it are usually expensive and risky. Health-related decisions are critical because they can cost a person his or her life. It happens all the time when we need doctors, but none are available, and the hospital's queuing system becomes chaotic. People nowadays do not have time to go to the doctor for a physical. The Disease Detection System is developed to analyze the symptoms and then give the conceivable disease symptoms of the patient given just like the doctor does. The system should respond to the circumstance of a doctor.

When the symptoms can be updated using the SVC, NB, and Random Forest Classifier algorithms, the system will become more dynamic. As a result, all of these algorithms will be implemented in this system to assist users in detecting diseases associated with their symptoms. The Disease Detection System will be a method for people to recognize health problems and efficiently detect diseases.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER** | **TITLE** | | **PAGE NO.** |
| **TITLE PAGE** | | | I |
| **CERTIFICATE** | | | II |
| **ACKNOWLEDGEMENT** | | | III |
| **ABSTRACT** | | | IV |
| **1** | **INTRODUCTION** | | 1 |
| 1.1 | Purpose | 1 |
| 1.2 | Scope | 2 |
| 1.3 | Objectives | 3 |
| 1.4 | SDLC Methodologies | 3 |
| **2** | **SYSTEM ANALYSIS STUDY** | | 7 |
| 2.1 | Introduction | 7 |
| 2.1.1 | Project Overview | 8 |
| 2.1.2 | Existing System Study Report | 8 |
| 2.2 | System Requirements Gathering | 8 |
| 2.2.1 | Prepared Questionnaire /Interview (With Filled Few Samples as Annexure) | 9 |
| 2.3 | Feasibility Study | 12 |
| 2.3.1 | Technical Feasibility | 12 |
| 2.3.2 | Economic Feasibility | 13 |
| 2.3.2 | Operational Feasibility | 14 |
| 2.4 | System Requirement Study Report | 15 |
| 2.4.1 | System Process Requirement | 15 |
| 2.4.2 | System Data Flow | 17 |
| 2.4.3 | Database Requirement | 17 |
| 2.4.4 | User Interface Requirement | 18 |
| **3** | **SYSTEMS DESIGN** | | 20 |
| 3.1 | Introduction | 20 |
| 3.2 | Physical Design: | 21 |
| 3.2.1 | System Flow Charts | 21 |
| 3.2.3 | Use Case Diagram | 22 |
| 3.2.4 | Data Flow Diagram ‟Up to Level 2” | 23 |
| 3.3 | Database Design | 25 |
| 3.3.1 | Database Schema | 25 |
| 3.4 | User Interface Design | 26 |
| 3.5.1 | Input Screen Designs | 27 |
| 3.5.2 | Report Layout Designs | 29 |
| **4** | **SOFTWARE CODING, TESTING & IMPLEMENTATION** | | 31 |
| 4.1 | Introduction | 31 |
| 4.1.1 | System Coding Environment and Standards Fo**l**lowed | 32 |
| 4.1.2 | Sample Code Layouts | 36 |
| 4.2 | Testing | 52 |
| 4.3 | Test Cases | 53 |
| **5** | **OUTPUT FORMS & REPORTS** | | 55 |
| 5.1 | Input/Output Screenshots with Sample Data | 55 |
| **6** | **CONCLUSION & FUTURE ENHANCEMENT(S)** | | 58 |
| 6.1 | Conclusion | 58 |
| 6.2 | Future Scope | 59 |
| **BIBLIOGRAPHY** | | | 60 |

**Chapter 1**

**INTRODUCTION**

A human being in this world is afflicted with numerous diseases. Diseases can have both physical and psychological effects on people.  Several diseases are cured after diagnosis and treatment, but chronic diseases are never cured regardless of treatment; however, treatment can prevent chronic diseases from worsening over time. As a result, it is extremely crucial to detect and treat disease as soon as possible. So, we're looking at a scheme where a patient requirement to see a doctor for an urgent illness like fever, cold, malaria, etc., so the doctor may or may not be capable to see the patient at that time. It is said that prevention is better than cure, so this system is proposed to help patients. The detection or diagnosis of a disease is critical in anyone's life. Some diseases, if detected early, can be completely cured or the disease state reversed. Treatment is also made easier, quality of life is preserved, and disease-related risks are avoided. The detection or diagnosis of a disease is critical in anyone's life. Some diseases, if detected early, can be completely cured or the disease state reversed. Treatment is also made easier, quality of life is preserved, and disease-related risks are avoided.

If we talk about the importance of diagnosing a disease early, the benefits are numerous, as it is a way to save lives and maintain the quality of living. Below are listed a few of the numerous benefits of early disease detection. If a disease is detected early, when the symptoms have not progressed to the point where it has caused damage to other organs, it is simple to treat with no complications. It reduces the individual's distress because he can return to his healthy state sooner.

A disease detection system has been proposed that detects the disease that gives the best possible prediction based on symptoms entered by the patient. The system contains various symptoms and their associated illnesses. The system allows users to share their symptoms and health complication.

**1.1 PURPOSE**

The main purpose is that when a patient visits a doctor for a viral illness such as dengue fever, fever, or malaria, the attending doctor may not be capable to criticize and demonstrate the patient at that time. Some diseases cause irreversible symptoms to develop over time, and in such cases, the patient can simply be given medicine to relieve the bothersome symptom. These people must deal with the disease. When such diseases are detected early, they can be reversed or eliminated from the body. As the disease progresses, so do the complications, as other organs, both related and unrelated, begin to be affected. As a result, the symptoms begin to mix, making it difficult to determine why a particular symptom has appeared. In such cases, early diagnosis aids in the treatment of the disease at an early stage without causing harm to other body organs.

As we all know, social distancing was important in the Covid-19 pandemic, fevers, coughs, and colds were all symptoms of Covid-19. So people were afraid that if they had a fever or a cold, they should not declare covid positive. For this reason, all patients were afraid to go to the hospital. If the pandemic strikes again, patients will automatically be able to recognize the disease, which is very beneficial for us. This scenario can affect patients who may be in the worst desirable condition and have health implications. A scenario where a patient has to wait in line to be seen at a hospital, and the worst happens when the patient shows up at the hospital at the wrong time. These are called the worst times to visit a hospital because of the long average waiting time and the inability to find an applicable doctor.

**1.2 SCOPE**

If a person has a life-threatening disease for which there is no treatment and only interventional measures are available, the family suffers as well. If the disease is detected early, a person can learn how to manage and cope with it. He can also plan effectively and use any available resource. Overall, if a disease is detected early, it is easier to treat and the quality of life is preserved. Stress is reduced for both the individual and family members.

A person's family is the closest to them, and their problem causes them to feel burdened and stressed. Early detection and diagnosis of the disease reduce stress and eliminates the feeling of regret for not becoming aware sooner. Disease detection systems are used by three types of users: admin, patients, and doctors. Admin manages the system as administrators.

The scope of,

1. Admin

Administrators can log into the system to rejuvenate doctor details and add doctors. Admins can also manage doctors and patients.

2. User

Users can access the system anytime, anywhere. User can manage their profile the way they like. When a user logs into the system, they can access the disease diagnosis, select symptoms based on the symptom record, and get their disease report. Users can book appointments with doctors and manage their appointments.

3. Doctor

Doctors can also access the system anytime, anywhere. Doctors advise their patients about this if the pain does not cure the disease. Otherwise, the doctor can make an appointment with the patient.

**1.3 OBJECTIVES**

Early detection may prevent rash decisions that would otherwise be made in ignorance and haste. The individual, alone or with family, can sit down and make the best decision about how to proceed with the treatment. He can plan ahead of time for his legal and financial obligations. Disease detection systems have been anticipated to accomplish some of the following principles:

1. Analysis of complications with existing disease detection systems.

2. Designing disease detection systems applying SVC, NB, and Random Forest Classifier algorithms.

3. Development of disease detection systems using SVC, NB, and Random Forest Classifier algorithms.

**1.4 SDLC METHODOLOGIES**

Software development methodologies (Software Development Life Cycle- SDLC Methodologies) are critical for software development. There are numerous development methods, each with its own set of advantages and disadvantages. To deliver a successful project, choose an appropriate development method for Project.

The disease detection system was matured based on selected methodologies from the Software Development Life Cycle (SDLC). The evolution of the system used to develop the project is the RAD model. This division details the phases that occur in this project development and the system requirements for developing the project.

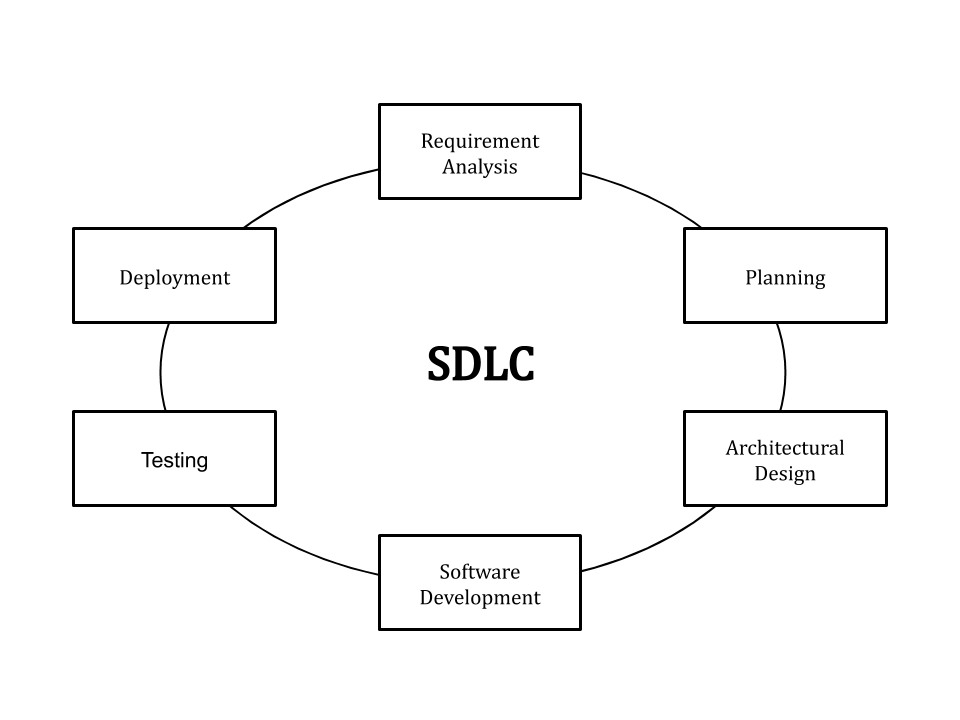
SDLC, or Software Development Life Cycle, is a process for producing software with the highest quality and lowest cost in the shortest amount of time. SDLC provides a well-structured flow of phases that enables an organization to quickly produce high-quality software that has been thoroughly tested and is ready for production use. SDLC reduces the cost of software development while improving quality and shortening production time. SDLC achieves these seemingly disparate goals by following a plan that eliminates the common pitfalls of software development projects. The plan begins by assessing existing systems for flaws. It then defines the new system's requirements. It then goes through the stages of analysis, planning, design, development, testing, and deployment to create the software. SDLC can eliminate redundant rework and after-the-fact fixes by anticipating costly mistakes such as failing to ask the end-user or client for feedback. It's also important to understand that the testing phase is prioritized. Because the SDLC is a repeatable methodology, you must ensure code quality at each cycle. Many organizations put little effort into testing, even though putting more effort into testing can save them a lot of rework, time, and money.

Figure 1: SDLC Steps

The stages are not clearly defined. Because the approach is iterative, different stages of application development can be reviewed and repeated. It is not necessary to know all of the requirements ahead of time. Changes are easier to accommodate. It requires very little documentation. Generally preferred for projects with shorter time frames and large enough budgets to allow for the use of automated tools and techniques. The use of reusable components contributes to a shorter project cycle time.

Rapid Application Development (RAD) refers to an adaptive software development model that relies on prototyping and immediate feedback rather than detailed planning. In general, the RAD approach prioritizes development and prototyping over planning. Rapid application development allows developers to make multiple iterations and updates to the software without having to start from scratch. This helps to ensure that the final product is more quality-focused and meets the needs of the end users.

One of the primary benefits of rapid application development is the ability to change the design, add functionality, and iterate as often as possible without having to start from scratch each time.

A significant disadvantage of the waterfall model is that once the product enters the testing phase, the tester cannot go back and repeat and change the core functions and features. This essentially leaves teams with software that may or may not meet the evolving needs of end users.

RAD is the best method for quickly developing prototypes for testing software functionalities without affecting the final product. Businesses prefer the RAD approach because it requires little planning while allowing the team to quickly design, review, and iterate features and functionalities.

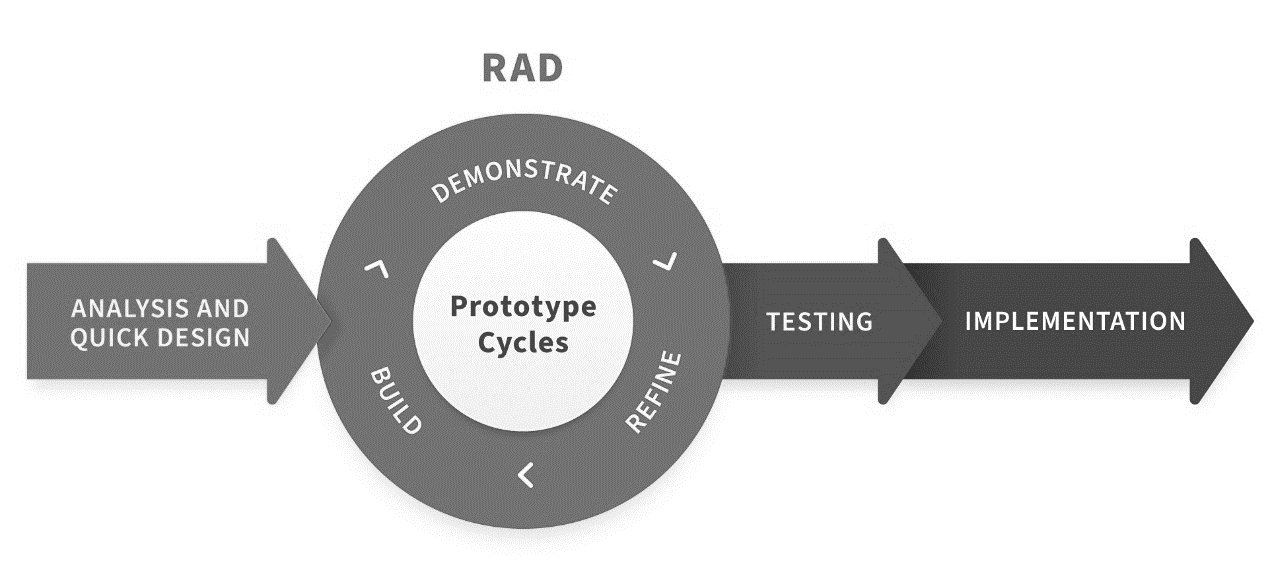


Figure 2: RAD Model

**Chapter 2**

**SYSTEM ANALYSIS STUDY**

**2.1 INTRODUCTION**

System analysis is the process of gathering and exhibiting facts, diagnosing problems and information about disease detection systems and recommending system improvements. A system analysis is a review of a technological system, such as a software package, for troubleshooting, development, or improvement. Analysts can discover code errors, end-user accessibility issues, and design incompatibilities through in-depth analysis. It is a problem-solving activity that requires intensive communication between system users and system developers, and system analysis or research is an important stage of the system development process. Systems analysis is the process of recognizing a problem, identifying relevant decision-making variables, analyzing and summarizing various factors, and determining an optimal or at least satisfactory solution or plan of action. The primary goals of systems analysis are to find answers to the following questions for each business process: What is done, How is it done, Who is doing it, When is he doing it, Why is it done, and How can it be improved? It is more of a thinking process that requires the System Analyst's creative abilities.

Analyzing a system is the first step in solving a problem that involves one. This entails breaking it down into its constituent parts and observing how those components interact with one another. Figuring out how a system works can sometimes involve turning off parts of it and seeing what happens, or changing parts of it and seeing what happens. How does changing what goes into a system affect what comes out? Essentially, systems analysis entails techniques that allow you to comprehend how a system operates. Systems analysis is a method of investigating our understanding of the world by breaking it down into its constituent parts. The foundations of systems analysis are determined by determining what those parts are and how they fit together.

**2.1.1 PROJECT OVERVIEW**

When working with multiple data sources, there are many chances for data to be incorrect, duplicated, or mislabelled. If data is wrong, outcomes and algorithms are unreliable, even though they may look correct. Data cleaning is the process of changing or eliminating garbage, incorrect, duplicate, corrupted, or incomplete data in a dataset. The motive of data cleaning services is to construct uniform and standardized data sets that enable data analytical tools and business intelligence easy access and perceive accurate data for each problem.

**2.1.2 Existing System Study Report**

Diagnostic decisions are commonly made based on the doctor's intuition and experience rather than the knowledge-rich data hidden in the database. This practice results in unintended biases, inconsistencies, and excessive medical costs, all of which have an impact on the quality of service provided to patients. A medical misdiagnosis can manifest itself in a variety of ways. A misdiagnosis of a serious illness, whether caused by a doctor or hospital staff, can have very severe and harmful consequences. Medical illnesses pose a significant threat to our healthcare profession. If they continue, people will be hesitant to go to the hospital for treatment. We can put an end to medical misdiagnosis by informing the public and filing claims and lawsuits against negligent medical practitioners.

**2.2 SYSTEM REQUIREMENT GATHERING**

Requirements gathering, also known as requirements elicitation, is the process of determining all of a project's requirements. There are two types of project requirements: business requirements and technical requirements. Business requirements describe what an organization will achieve with the project, whereas technical requirements describe how the project must be carried out. They are gathered during the project life cycle's initiation phase, but project managers must monitor them throughout the project timeline because they can change. A requirement document is used to explain what the product requires. It defines the product vision and how it must be achieved by the end of the project, among other things. However, it does not specify how it will be delivered. It's more about putting the product in context, such as why it's needed or what problem it solves. The specifics of how it will accomplish this are not provided.

How long will the project take? Plan your timeline, and then plan your requirements around it. This will be useful if some requirements are conditional on dependencies.

Who will be a part of the project? Will the entire design and development teams be involved, or just a few members from each? Which members of the team will be available? Which team members are experts in the issues that the project will address?

What are the risks associated with the requirement-gathering process? Define and document all assumptions and risks that may impact your requirements. Understand that your assumptions are typically classified as time, budget, and scope. They can range from assuming PTO, holidays, and sick days to assuming stakeholders will provide timely feedback.

What is our ultimate goal in comprehending our project specifications? What are the time-based, budget-based, and scope-based goals? Will it be to compete more directly in the market with a competitor? Will it be to resolve a customer issue or to address a bug?

**2.2.1 Prepared Questionnaire/ Interview (with Filled Few Samples as Annexure)**

**Q.** What is the operation of a Disease Detection System?

Once the Disease Detection System has been developed, doctors, users, or medical professionals select the disease for detection from the several disease options in the Disease Detection System's GUI. Then, using the system's graphical user interface, he or she enters the necessary patient data and completes the form.

**Q.** What information is used by the Disease Detection System?

The CSV dataset was used to generate the data for the experiments in this project. This dataset contains data on a variety of diseases, including liver disorders, hepatitis, heart disease, diabetes, and chronic kidney disease.

**Q.** How can the Disease Detection System's accuracy be improved?

After disease detection, inform users on how to prevent disease. In the future, we can apply this technique to other chronic diseases with large datasets. The accuracy can be improved by increasing the number of diseases and datasets used in the process.

**Q.** How did the Disease Detection System perform?

The proposed Disease Detection System can be used by doctors or medical professionals to detect diseases in patients via a Graphical User Interface (GUI). This system employs machine learning algorithms such as SCV, NB, and the Random Forest Classifier Algorithm to identify referred diseases with 70% accuracy, precision, and recall. The Disease Detection System GUI was created with Python support so that doctors, users, and medical professionals can easily detect disease symptoms.

**Q.** How does the Disease Detection System display diagnostic results?

Obtain Diagnosis Result: The system receives the diagnosis result after the classification calculation is completed. Show the User the Result: The user will then see the diagnostic result via the system's UI module.

**Q.** How is the Disease Detection System established?

The disease manifests itself differently in each patient. To implement the Disease Detection System, different datasets are extracted from the CSV dataset. The disease detection system is built with the AdaBoost classifier model: The Disease Detection System is built using the Adaboost classifier model. Python is used to create the Disease Detection System's Graphical User Interface (GUI).

**Q.** Why is it necessary to develop a Disease Detection System?

Because of the high number of deaths caused by chronic diseases such as heart disease, lung cancer, and brain tumors, doctors must develop a proper diagnosis system. We need to work on an accurate diagnosis of multiple diseases because incorrect diagnosis leads to human deaths.

**Q.** How precise is the Disease Detection System?

The analysis discovered that AI can correctly diagnose disease in 87% of cases, while detection by healthcare professionals yielded an 86% accuracy rate. Deep learning algorithms had a specificity of 93%, while humans had a specificity of 91%. Biases may overestimate AI performance.

**Q.** Is a disease detection system superior to human detection?

The analysis discovered that AI can correctly diagnose disease in 87% of cases, while detection by healthcare professionals yielded an 86% accuracy rate. Deep learning algorithms had a specificity of 93%, while humans had a specificity of 91%.

“Within that handful of high-quality studies, we found that deep learning could indeed detect diseases ranging from cancer to eye diseases as accurately as health professionals. But it’s important to note that AI did not substantially outperform human diagnosis.” -Prof. Alastair Denniston

**Q.** What is the impact of adding disease detection features using images to the Disease Detection System?

Skin diseases are more common than other types of illnesses. Skin diseases can be caused by a fungal infection, bacteria, allergies, or viruses, among other things. The advancement of laser and photonics-based medical technology has allowed for much faster and more accurate diagnosis of skin diseases. However, the cost of such a diagnosis is still limited and prohibitively expensive. Thus, image processing techniques aid in the development of an automated screening system for dermatology at an early stage. The extraction of features is critical in the classification of skin diseases. In a variety of techniques, computer vision plays a role in the detection of skin diseases.

**2.3 FEASIBILITY STUDY**

The feasibility study is a measure of the software product in terms of how beneficial product development will be for the organization from a practical viewpoint. A feasibility study is conducted for a variety of reasons, including determining whether a software product is appropriate for development, implantation, project contribution to the organization, and so on. A feasibility study is a thorough examination of all critical aspects of a proposed project, in business; success is primarily defined by return on investment, which means that the project will generate enough profit to justify the investment. However, many other important factors, such as community reaction and environmental impact, can be identified as positive or negative. Although feasibility studies can assist project managers in determining the risk and return on investment of pursuing. When a company considers starting a new business, adding a new product line, or acquiring a competitor, it may conduct a feasibility study.

When conducting a feasibility study, it's always a good idea to have a backup plan in place that you can test to ensure it's a viable option. A feasibility study is a project, so invest in project management software to help you complete it.

**2.3.1 Technical Feasibility**

Technical feasibility focuses on existing computer systems and how well they can support proposed additions. For example, if your computer is currently at 80% utilization and you're hitting the limit, running another application might overload the system or require additional hardware. This includes financial considerations to account for technical improvements. In Technical Feasibility, current resources, including hardware and software, as well as required technology, are analyzed to develop a project. This technical feasibility study determines whether the necessary resources and technologies are available for project development. A technical feasibility study is a comprehensive examination of the project's input, processes, output, fields, programs, and procedures. It is an extremely useful tool for long-term planning and troubleshooting. The technical feasibility study should primarily support an organization's financial information. can be defined as the formal process of determining whether a product or service is technically feasible to manufacture. Before launching a new offering or undertaking a client project, it is critical to plan and prepare for each stage of the process. Companies conduct feasibility studies before beginning work on a project, which is standard practise. Before launching a product or service, businesses conduct a technical feasibility study to determine its practicability and viability. Whether you work as a product engineer, product designer, or team manager, you may be required to prepare a technical feasibility study at some point in your career. In this article, we will define technical feasibility, explain how to conduct one and provide advice on how to write a feasibility study report.

**2.3.2 Economic Feasibility**

Economic analysis is the most commonly used method for evaluating the effectiveness of candidate systems. Commonly known as cost-benefit analysis, this process consists of determining expected benefits and savings from a candidate system and comparing them to costs. The cost and benefit of the project are examined in the Economic Feasibility study. This feasibility study includes a detailed analysis of the project's development costs, which include all required costs for final development such as hardware and software resources, design and development costs, and operational costs, among other things. Following that, it is determined whether the project will be financially beneficial to the organization. Economic Feasibility is a critical step in determining the costs, benefits, risks, and rewards of a new business venture. Feasibility studies assist businesses in planning operations, identifying opportunities and risks, and attracting investors. Economic feasibility elements include but are not limited to, increased or decreased revenue to other agencies or the general public, increased or decreased costs to other agencies or the general public, and increased and/or decreased costs to other agencies or the general public. They also provide independent project evaluation and boost project credibility.

Otherwise, if the proposed system is likely to be approved, further justification or alternatives should be made to the proposed system. This is an ongoing effort to improve accuracy at every stage of the system lifecycle.

**2.3.3 Operation Feasibility**

The degree of providing service to requirements is analyzed in Operational Feasibility, as well as how easy the product will be to operate and maintain after deployment. The willingness and ability of management, employees, customers, suppliers, and others to use and support a proposed system is referred to as operational feasibility. It is a measurement of how well a new proposed system solves problems. It aids in capitalizing on opportunities and meeting the requirements identified during the project's development. This feasibility is dependent on human resources and entails visualizing whether the software will function after it has been developed and will be operational after it has been installed. Other operational scopes include determining the usability of the product, determining whether or not the suggested solution by the software development team is acceptable, and so on. The above software's interface is far too simple to comprehend. All of the functionality is listed separately to provide the user with easy access. All of the pages are linked together so that the user can navigate to any page they want. The following tasks are also performed by operational feasibility.

Programs that reduce costs without sacrificing product quality are examples of operational feasibility. Studies are carried out to ensure that programs can be launched in the current production facility without the need for additional equipment or personnel. If more space, machinery, or personnel are required, the system must improve the way the product is perceived by customers. Because of the anticipated increase in sales and revenue, this allows for the additional expense of manufacturing the product.

One aspect of operational feasibility is economic feasibility. Everyone involved in the development and implementation of the system, product, or program must understand it and be able to use it in the manner intended. All governmental organizations conduct feasibility studies, not just the private sector. It is not considered feasible if the changes made to a product or program are not understood by the end users.

The ability to use, support, and perform the necessary tasks of a system or program is referred to as operational feasibility. Everyone who creates, operates, or uses the system is included. To be operationally feasible, the system must meet a business requirement.

**2.4 SYSTEM REQUIREMENT STUDY REPORT**

The use of software and hardware tools is essential for the creation of this project. To complete the project, the hardware and software facilities must be used. The use of these establishments is determined by what is already available or what has previously been used. These are the specifications for the disease detection system.

A Software Requirements Specification (SRS) is a document that describes the characteristics of a project, piece of software, or application. In simple terms, an SRS document is a project manual that is prepared before the start of a project/application. This document is also known as an SRS report and a software document. A software document is typically created for a project, software, or any other type of application. When creating the software requirement specification document, a set of guidelines must be followed. This includes the project's purpose, scope, functional and nonfunctional requirements, as well as software and hardware requirements. It also contains information about the required environmental conditions, safety and security requirements, software quality attributes of the project, and so on. A software requirements specification document describes the intended purpose, requirements, and nature of a piece of software that is being developed. It also includes the software's yield and cost.

**2.4.1 System Process Requirement**

Python is an object-oriented, high-level programming language with dynamic semantics that is interpreted. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very appealing for use as a scripting or glue language to connect existing components. Python is frequently embraced by programmers due to the increased productivity it provides. The edit-test-debug cycle is extremely fast because there is no compilation step. Debugging Python programs is simple: a bug or incorrect input will never result in a segmentation fault. Python's simple, easy-to-learn syntax emphasizes readability, lowering program maintenance costs. Python provides support for modules and packages, which promotes program modularity and code reuse.

**Hardware Requirement**: The physical computer resources, also known as hardware, are the most common set of requirements defined by any operating system or software application. In the case of operating systems, a hardware requirements list is frequently accompanied by a hardware compatibility list (HCL). An HCL is a list of hardware devices that have been tested, are compatible, and sometimes are incompatible with a specific operating system or application.

|  |  |
| --- | --- |
| Name of the component | Specification |
| Processor | Minimum i3 10th Gen with 2.0 GHz clock speed or more |
| RAM | Minimum 8GB |
| Hard Disk | Minimum 50GB |
| Monitor | 14 “color monitor” |

**Software Requirement**: Software requirements are concerned with defining the software resource requirements and prerequisites that must be installed on a computer for an application to function properly. These prerequisites or requirements are typically not included in the software installation package and must be installed separately before the software can be installed.

|  |  |
| --- | --- |
| Name of the component | Specification |
| Operating System | Windows 8,7, XP |
| Language | Python 3.11 |
| Database | Excel & MySQL |
| Modules | Scipy, Scikit-Learn, Pillow |

**2.4.2 System Data Flow**

The Information System/Data Flow Diagram captures the main components of an Information System, how data moves within the system, user interaction points, and the Authorization Boundary. Consider this diagram to be conceptual rather than technical; multiple systems can be abstracted together, and no network connections need to be detailed. The Authorization Boundary describes the Information System's limits - which components are currently being evaluated. Information Systems frequently rely on other information systems; however, those other information systems will be evaluated independently and their risk factored into the current information system.

A system flow diagram is a graphical representation of the data flow in a system. The diagram is comprised of several steps that identify where the system's input and output are located. The diagram allows you to control the system's event decisions as well as how data flows to the system. As a result, the system flow diagram is essentially a visual representation of data flow, excluding minor parts while including the major parts of the system sequentially. It provides a visual representation of data that non-IT people can understand without any special technical skills, and it is much easier to show the loops and branches in a diagrammatic way, as well as the conditions of the branches so that it is easily comprehensible for the user than using complex algorithms or pseudo-codes. A system flow diagram is required to detect flaws in a system, find a solution, and then execute the solution in very simple and sequential steps that are major while excluding minor ones that are unnecessary.

A system flow diagram is also required for large-scale industries to help visualize all of the system's major problems and how to find effective solutions. The diagram also aids in understanding the sequential flow of a process, where the solution is heading based on conditions, and identifying bottlenecks.

**2.4.3 Database Requirement**

**User Details**: If the user needs to access the database, the database will generate a new account id and password for him or her. The database is also responsible for database security, allowing only authorized users to access/modify the database. DBAs are to blame for issues such as security breaches and slow system response times. Parametric End Users are unsophisticated individuals who have no DBMS knowledge but frequently use database applications in their daily lives to achieve the desired results. Data Base Designers are users who create the database structure, which includes tables, indexes, views, triggers, stored procedures, and constraints, that are usually enforced before the database is created or populated with data. He or she determines what data must be stored and how data items are related. It is the responsibility of Database Designers to understand the needs of various user groups and then create a design that meets the needs of all user groups. Casual users are those who use/access the database on an irregular basis, but each time they do so, they require new information.

**Disease Data**: Depending on its scope, a disease database can be classified as general or specialized. General disease databases cover a broader range of diseases and medical conditions, whereas specialized databases focus on specific diseases such as cancer, tropical diseases, or genetic diseases. Disease databases are collections of data on diseases, syndromes, and other medical conditions. They may provide specific information on signs and symptoms, risk factors, treatment regimens, and/or the findings of studies conducted on known diseases and medical conditions. Disease databases can be found in both print and online publications. Many printed publications have already been replaced by online resources.

**2.4.4 User Interface Requirement**

**Tkinter**: Tkinter is the most widely used Python library for creating GUI (Graphical User Interface). It is a standard Python interface to the Python-supplied Tk GUI toolkit. Because Tk and Tkinter are available on most Unix platforms as well as Windows, developing GUI applications with Tkinter becomes the quickest and easiest option.

Python provides numerous GUI development options (Graphical User Interface). Tkinter is the most commonly used of all the GUI methods. It is a standard Python interface to the Python-supplied Tk GUI toolkit. Python with Tkinter is the quickest and most straightforward way to create GUI applications. Using Tkinter to create a GUI is a simple task.

To create a tkinter:

• Adding the module – Tkinter

• Make the main window (container)

• Add as many widgets as you want to the main window.

• On the widgets, use the event Trigger.

mainloop(): When your application is ready to run, a method called mainloop() is called. mainloop() is an infinite loop that is used to run the application, wait for an event, and process the event as long as the window is open.

Tkinter also provides access to the geometric configuration of widgets, which can be used to organize widgets in parent windows. There are primarily three geometry manager classes.

**Tkinter button widget:** In applications, button widgets represent a clickable item. Typically, you use text or an image to display the action that will be taken when the button is clicked.

Buttons can only display text in one font. The text, on the other hand, can span multiple lines. You can also make one of the characters underlined to indicate a keyboard shortcut.

To automatically invoke a function or method of a class when the button is clicked, assign its command option to the function or method. This is known as the bindinTkinter command.

To create a button, you use the ttk.Button constructor is as follows:

button = ttk.Button(container, \*\*option)

Code language: Python (python)

A button has numerous options. The typical ones, on the other hand, look like this:

button = ttk.Button(container, text, command)

Code language: Python (python)

In this syntax:

* The container is the parent component on which you place the button.
* The text is the label of the button.
* The command specifies a callback function that will be called automatically when the button is clicked.

**Chapter 3**

**SYSTEM DESIGN**

**3.1 INTRODUCTION**

System design refers to the process of defining a system's entire requirements, such as the architecture, modules, interface, and design. We can say that system design encompasses everything from discussing system requirements to product development. System development is the process of creating or altering a system to change the processes, practices, and methodologies used to develop the system. The design process begins with the most basic components and subsystems. These components are used to create or compose the next higher-level components and subsystems. The process is repeated until all of the components and subsystems have been combined into a single component, which is referred to as the complete system. As the design progresses to higher levels, the amount of abstraction increases.

When a new system must be created using existing basic information, the bottom-up strategy is appropriate. A good system design organizes the programmed modules so that they are simple to develop and change. Structured design techniques assist programmers in dealing with programmed size and complexity. Analysts write instructions for developers on how to write code and how to put pieces of code together to form a program.

It specializes in producing high-quality artwork while saving time and effort. This aids in the development of information system plans. It is used to solve internal issues, increase efficiency, and broadcast opportunities. It is also the basis of any business. It significantly contributes to achieving the desired results and makes work easier and simpler.

It is an abstract representation of the system's data flow, inputs, and outputs. It explains the sources, destinations, data stores, and data flows in a process that meets the needs of the user. A system's logical design is created with a level of detail in mind that virtually tells the information flow in and out of the system in mind. The data flow and E-R diagrams are employed.

**3.2 PHYSICAL DESIGN**

The physical design of the system is related to the process of actual input and output. The primary physical design criteria are to manage how data is verified, processed, and displayed as a result. It is primarily concerned with the user's interface design, process design, and data design.

In current physical DFD, process labels include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing labels including an identification of the technology used to process the data similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders "computer files" business forms or computer tapes.

Architecture Design: It is also known as the high level of design because it focuses on the design of system architecture. It explains the system's nature and origins.

**3.2.1 System Flow Charts**

**3.2.3 Use Case Diagram**

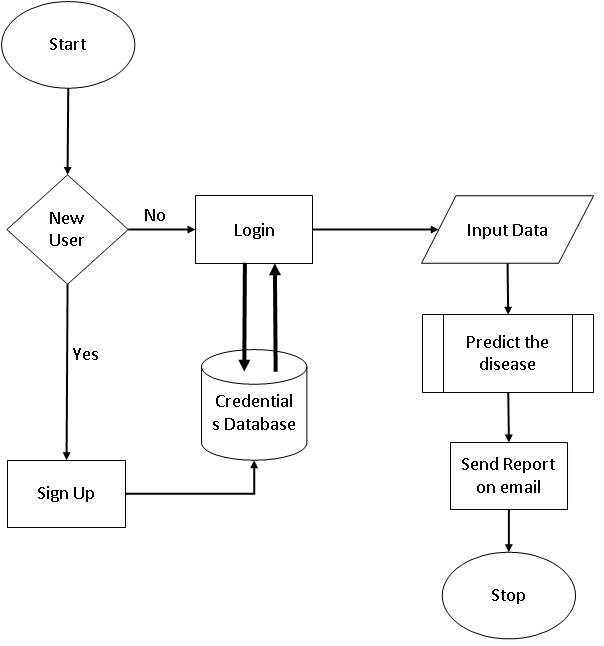
****At its most basic, a use case diagram is a representation of a user's interaction with the system that depicts the relationship between the user and the various use cases in which the user is involved. A use case diagram can identify different types of system users and use cases, and it is frequently accompanied by other types of diagrams.

Figure 3:System Flow Diagram

Although use case diagrams can be used for a variety of purposes, there are some general guidelines to follow when creating use cases. These include naming conventions, arrow direction, the placement of use cases, the use of system boxes, and the proper use of relationships. In a separate blog post, we went over these guidelines in depth. In a use case diagram, an actor is any entity that plays a role in a single system. This could be a person, an organization, or an external system, and is typically depicted as the skeleton shown below.

A use case is a representation of a system function or action. It's drawn in the shape of an oval and labeled with the function. To identify functions and how roles interact with them – The primary purpose of use case diagrams.

For a high-level view of the system – Especially useful when presenting to managers or stakeholders. You can highlight the roles that interact with the system and the functionality provided by the system without going deep into the inner workings of the system.

To identify internal and external factors – This might sound simple but in large complex projects a system can be identified as an external role in another use case. If you come across two or more use cases that share common functionality, you can extract the common functions and add them to a new use case. Then, using the include relationship, you can show that it is always called when the original use case is executed. Actors may be associated with similar use cases while triggering a few use cases that are unique to them. In such cases, you can generalize the actor to demonstrate function inheritance. You can do the same thing with use cases.

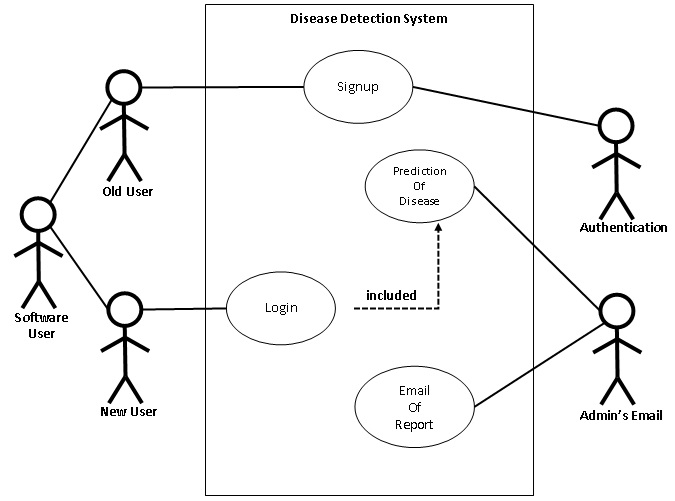
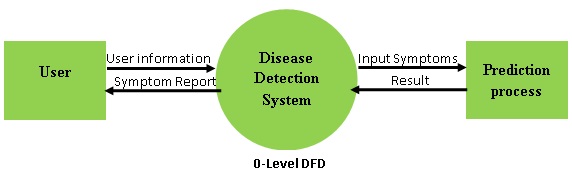
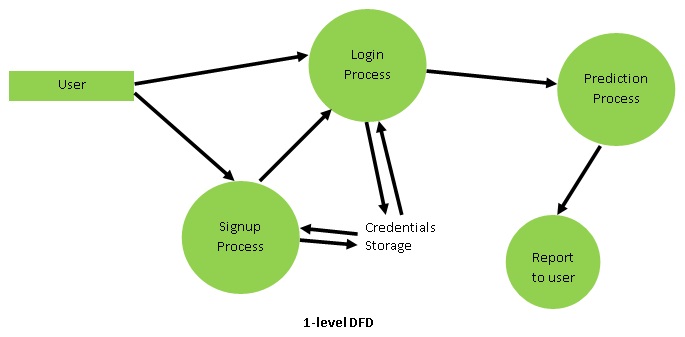
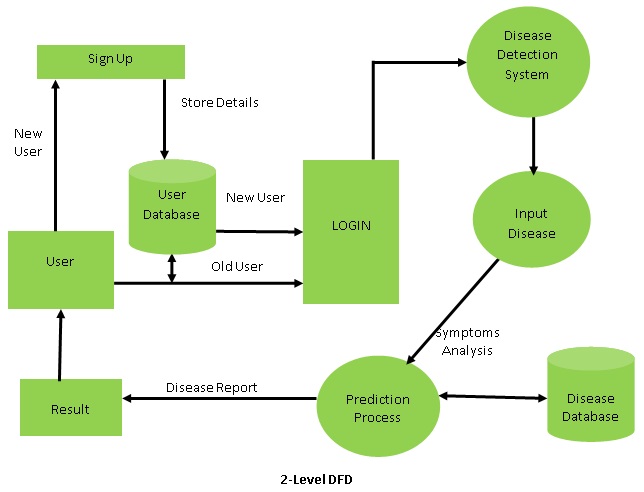


Figure 4: DDS Use Case Diagram

**3.2.3 Data Flow Diagrams**

**0-Level DFD:** Actors may be associated with similar use cases while triggering a few unique use cases. In such cases, the actor can be generalized to demonstrate function inheritance. The same can be said for use cases.

**1-level DFD:** At this level, we highlight the system's main functions and divide the high-level process of 0-level DFD into sub-processes.

**2-level DFD**: 2-level DFD delves deeper into elements of 1-level DFD. It can be used to plan or record specific/necessary details about how the system works. DFD level 2 is the highest extraction for the project, which means it is the most detailed diagram of the three. However, this level only specifies a sub-process or a portion of the system from DFD level 1. DFD level 2 can be used in multiple diagrams to specify all aspects of your project.

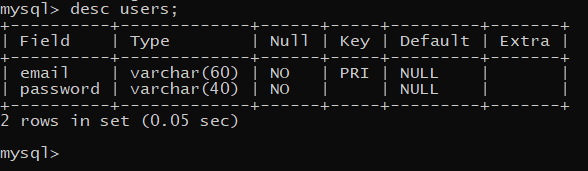
**3.3 DATABASE DESIGN**

Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of an enterprise data management system. Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space. Therefore, there has to be a brilliant concept for designing a database. The designer should follow the constraints and decide how the elements correlate and what kind of data must be stored.

**3.3.1 Database Schema**

A database is used to organize and store a collection of records. It enables us to organize data into tables, rows, columns, and indexes to quickly find relevant information. We can easily access and manage the records using the database. MySQL implements databases as directories that contain all files in the form of tables. MySQL is available under a variety of proprietary licenses and is free and open-source software under the terms of the GNU General Public License.

MySQL was owned and sponsored by MySQL AB, a Swedish company that was purchased by Sun Microsystems (now Oracle Corporation). When Oracle acquired Sun in 2010, Widenius forked the open-source MySQL project to form MariaDB.

****MySQL implements a database as a directory containing all files that correspond to the database's tables. In MySQL, you use the CREATE DATABASE statement with the following syntax to create a new database: First, after the CREATE DATABASE clause, specify the database name. Within the MySQL server instance, the database name must be unique.

The use of the database in this project is to store the email-ids and passwords of the users operating it and use that same database to get unique user email-ids on signup. And, non-duplicity is ensured by the key constraint in the table.

MySQL is a relational database management system that is free and open source (RDBMS). A relational database organizes data into one or more data tables, each of which can be related to another; these relationships help structure the data. SQL is a programming language that allows programmers to create, modify, and extract data from relational databases, as well as control user access to the databases. An RDBMS, in addition to relational databases and SQL, works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access, and facilitates testing database integrity and backup creation.

**3.4 USER INTERFACE DESIGN**

Designing user interfaces for hardware and software, such as computers, home appliances, mobile devices, and other electronic devices, to optimize usability and the user experience is known as user interface (UI) design or user interface engineering. Information architecture is the main emphasis of user interface (UI) design when it comes to computer or software development. The process of creating interfaces is what effectively conveys to the user what is crucial. Graphical user interfaces and other types of interface design are referred to as UI design. Making user interactions as easy to use and effective as feasible in achieving user goals is the aim of user interface design.

The goal of user interface (UI) design is to foresee what users would need to do and make sure that the interface has parts that are simple to use, access, and comprehend. Information architecture, interaction design, and graphic design ideas are all combined in UI. Users can only interact with your app or website through the user interface, thus how it is designed will largely affect how they feel about it. You already know how annoying a poorly designed user interface can be if you've used any low-effort apps or websites. You don't have much time to impress a consumer these days because apps are vying for their attention. In actuality, 71% of all app users churn after 90 days, according to Localytics.

User interface designers provide your software or website with a creative edge that helps it stand out from the many alternative options available. Creative UI design is how you may stand out, even if you're adhering to a more general template of what a ride-hailing app or booking website should look like. Users can only interact with your app or website through the user interface, thus how it is designed will largely affect how they feel about it. You already know how annoying a poorly designed user interface can be if you've used any low-effort apps or websites. You don't have much time to impress a consumer these days because apps are vying for their attention. In actuality, 71% of all app users churn after 90 days, according to Localytics.

User interface designers provide your software or website with a creative edge that helps it stand out from the many alternative options available. Creative UI design is how you may stand out, even if you're adhering to a more general template of what a ride-hailing app or booking website should look like.

**3.4.1 Input Screen Designs**

To avoid mistakes when entering data, it is crucial to establish acceptable data input techniques. These techniques vary depending on whether users directly enter data on PCs or whether users physically enter data in forms and afterward have it entered by data entry workers.

These are the goals of input design.

• to create data input and entry methods

• lowering the input volume

• create new data capture techniques or create source documents for data capture

• to create user interface displays, data entry screens, input data records, etc.

• to implement validation controls and provide efficient input controls. A system should guard against user errors by

• By providing adequate room for legible writing, forms are designed.

• Form filling instructions that are clear.

• simple form layout.

• fewer keystrokes.

• immediate feedback on errors.

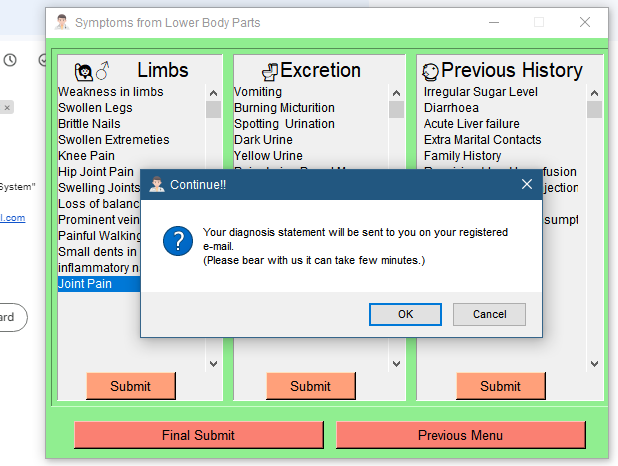
It is a two-way exchange. The system receives input data and instructions (what users wish to do) in a variety of ways. Responses from the system include outputs, error messages, feedback, warnings, help functions, etc. The interface also describes how users move about the program.

Figure 5: Input Interface Design

**3.4.2 Report Layout Designs**

A report layout determines the content and appearance of the report, including the order in which the data fields from a report dataset are displayed, the text style, the use of graphics, and more. You can create a new layout, alter an existing layout, or switch the layout being used on a report from Business Central.

**Extension patterns**

Layouts that are a component of an extension that has been deployed on the environment are called extension layouts. These layouts are often industry-standard layouts offered by companies like Microsoft in the base application. Alternatively, they might be designs that are incorporated into add-ons from different software developers. The extension name and publisher are displayed in the Extension column, making extension layouts on the Report Layouts page easy to identify.

**Individualized layouts**

The end-user is another source of layouts. A user with the right permissions can add new layouts using a variety of methods from within Business Central. One user-defined layout or an existing extension layout, for instance, could serve as a starting point. The Extension column in the user-defined layout on the Report Layouts will be empty.

**User-defined layout**

Layouts can also be obtained from end users. A user who has the necessary permissions within Business Central can add new layouts in several different methods. A current extension layout or another user-defined layout would be a good place to start. The Extension column will be empty in the user-defined layout on the Report Layouts.

**Custom layout**

Users can also design their layouts, known as custom layouts. The distinction is that these layouts can only be Word and RDLC types and are produced using the legacy Custom Report Layouts page. Custom layouts are still possible, although user-defined layouts are gradually replacing them.

Since the report or diagnosis statement is being sent by email the layout is the default mail design. The message is formatted as,

“*Dear {user’s email},*

*This is a probable disease predicted by our ML model.*

*The result of detection is {diagnosis predicted by the model}.*

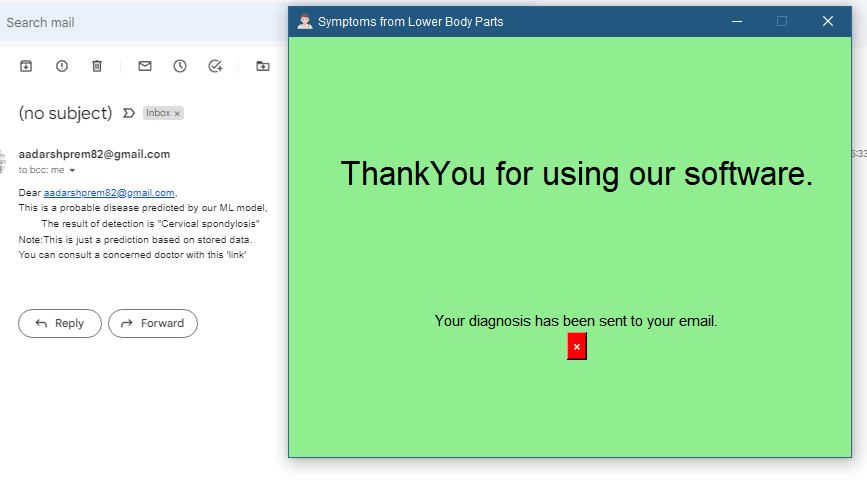
*Note: This is just a prediction based on stored data. You can consult a concerned doctor with this {link to the appointment page****}****.*”

Figure 6: Output Interface

**Chapter 4**

**SOFTWARE CODING, TESTING & IMPLEMENTATION**

**4.1 INTRODUCTION**

The process of designing, writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs is known as computer programming. A programming language is used to write this source code. The goal of programming is to create a program that performs a specific desired behavior. The process of writing source code frequently necessitates knowledge of multiple subjects, such as the application domain, specialized algorithms, and formal logic. In general, good programming is defined as the calculated application of all three, to produce an efficient and evolvable software solution (the criteria for "efficient" and "evolvable" vary considerably). In contrast to many other technical professions, programmers are not required to be licensed or pass any standardized (or governmentally regulated) certification tests to call themselves "programmers" or "software engineers."

* Efficiency/performance: the number of system resources consumed by a program (processor time, memory space, slow devices such as discs, network bandwidth, and, to a lesser extent, user interaction): the less, the better. This also includes proper resource disposal, such as cleaning up temporary files and avoiding memory leaks.
* Reliability is the frequency with which a program's results are correct. This is dependent on the conceptual correctness of algorithms and the minimization of programming errors, such as resource management errors (e.g., buffer overflows and race conditions) and logic errors (such as division by zero or off-by-one errors).
* Robustness refers to how well a program anticipates problems that are not caused by programmer error. This includes incorrect, inappropriate, or corrupt data, a lack of required resources such as memory, operating system services, and network connections, as well as user error.
* Usability is a program's ergonomics: the ease with which a person can use the program for its intended purpose, or in some cases, unexpected purposes. Regardless of other factors, such issues can make or break a company's success. This includes a variety of textual, graphical, and occasionally hardware elements that improve the clarity, intuitiveness, cohesiveness, and completeness of a program's user interface.
* The range of computer hardware and operating system platforms on which a program's source code can be compiled/interpreted and run. This is determined by differences in the programming facilities provided by different platforms, such as hardware and operating system resources, expected behavior of the hardware and operating system, and the availability of platform-specific compilers (and sometimes libraries) for the source code language.
* Maintainability is the ease with which a program's current or future developers can modify it to make improvements or customizations, fix bugs and security holes, or adapt it to new environments. In this regard, good practices during initial development make a difference. This quality may not be immediately apparent to the end user, but it can have a significant impact on the long-term success of a program.

**4.1.1 System Coding Environment and Standards Followed**

Different programming languages support various programming styles (called programming paradigms). Many factors influence the language used, including company policy, task suitability, the availability of third-party packages, and individual preference. In an ideal world, the programming language best suited to the task at hand will be chosen. Finding enough programmers who know the language to form a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute are all trade-offs from this ideal. Languages are roughly classified as "low-level" or "high-level"; "low-level" languages are more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute slower.

* Input: Get information from the keyboard, a file, or another device.
* Display data on the screen or send data to a file or another device as output.
* Perform basic arithmetic operations such as addition and multiplication.
* Conditional execution: When certain conditions are met, the appropriate sequence of statements is executed.
* Repetition: Repeating an action, usually with some variation.

If you've only recently begun writing code, you've almost certainly been asked by professionals or your teacher to make it neater and cleaner, to change the names of variables, or to improve the code's documentation. Because of the human mind, you would not understand your code even after a few weeks if it was written incorrectly, which is why Python code standards are developed to maintain quality.

Python is a well-structured language that uses indentation to separate code blocks. However, several Python code quality rules must be followed to produce beautiful and easily understandable code.

The process of converting a system's design into a computer language format is known as coding. This phase of software development is concerned with software translating design specifications into source code. It is necessary to write source code and internal documentation so that the code's conformance to its specifications can be easily verified.

Coding is performed by coders or programmers, who are independent of the designer. The goal is not to reduce the effort and cost of coding but to reduce the cost of a subsequent stage. With efficient coding, the cost of testing and maintenance can be significantly reduced. To convert a system design into a computer language format: Coding is the process of converting a system's design into a computer language format that can be executed by a computer and perform tasks as specified by the operation design during the design phase.

To convert a system's design into a computer language format: Coding is the process of converting a system's design into a computer language format that can be executed by a computer and perform tasks as specified by the design of operation during the design phase.

To cut the cost of subsequent phases: With efficient coding, the cost of testing and maintenance can be significantly reduced.

Making the program easier to read and understand: The program should be simple to read and understand. It improves code comprehension because having readability and understandability as a clear goal of the coding activity can help in producing more maintainable software.

**Space indentation and no tabs:** The 4-space rule is not always mandatory and can be overruled for continuation line.

def thank\_you():

clear\_everything()

tk.Label(MainWin, text="ThankYou for using our software.",height=7,width=30,

font=("Roboto bold",25), bg="lightgreen").grid(row=0, column=0)

tk.Label(MainWin, text="Your diagnosis has been sent to your email.",

font=("Roboto bold",11),bg ="lightgreen").grid(row=1, column=0)

tk.Button(MainWin, text="×", font=("Arial bold",10),

bg="red",fg="white",command=lambda:exit()).grid(row=2,column=0)

* **Use docstrings:** In Python, you can use both single and multi-line docstrings. The single-line comment, on the other hand, fits on one line; triple quotes are used in both cases. These are used to define a specific program or a specific function.

for i, val in enumerate(symptoms):

temp = val.replace("\_"," ").title()

all\_symptom = " ".join([temp])

symptoms\_index[all\_symptom] = i

* **Wrap lines so that they are no longer than 79 characters:** The Python standard library is conservative and requires that lines be no longer than 79 characters. Parenthesis, brackets, and braces can be used to wrap the lines. They should be used instead of backslashes.

OTP += dig[math.floor(random.random()\*10)]

gen\_OTP = "Welcome to \"Disease Detection System\" \nPlease verify your,\nEmail Id : " + email + "\nPassword : " + paswrd + " \nwith this OTP : " + OTP + "."

* **Regular and updated comments are valuable to both coders and users:** There are also various types and conditions that, if followed, can be of great assistance to both programs and users. Complete sentences should be formed from comments. Unless it is an identifier that begins with a lower case letter, the first word of a comment should be capitalized if it is a full sentence. The period at the end of short comments can be omitted. There are multiple paragraphs in block comments, and each sentence must end with a period. A single '#' can be used to separate block and inline comments.

#Sending Diagnosis

def final\_submit():

**4.1.2Sample Code Layouts**

#Modules

import tkinter as tk

from tkinter import \*

from tkinter import messagebox as msg

import os

import math

import random

import smtplib

import mysql.connector as sql

#ML\_Modules

import numpy as np

import pandas as pd

from scipy.stats import mode

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.svm import SVC

from sklearn.naive\_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix

#Image\_Modules

from tkinter import filedialog

from tkinter.filedialog import askopenfile

from PIL import Image, ImageTk, ImageChops

OTP = ""

input\_set = set()

mail = ""

#Clear Window

def clear\_everything():

    for i in MainWin.winfo\_children():

        i.destroy()

#Model Creation

def preparation(user\_symptoms):

    data = pd.read\_csv("Training.csv").dropna(axis = 1)

    encoder = LabelEncoder()

    data["prognosis"] = encoder.fit\_transform(data["prognosis"])

    X = data.iloc[:,:-1]

    Y = data.iloc[:,-1]

    svm\_model = SVC()

    nb\_model = GaussianNB()

    rf\_model = RandomForestClassifier(random\_state = 18)

    svm\_model.fit(X,Y)

    nb\_model.fit(X,Y)

    rf\_model.fit(X,Y)

    symptoms = X.columns.values

    symptoms\_index = {}

    for i, val in enumerate(symptoms):

        temp = val.replace("\_"," ").title()

        all\_symptom = " ".join([temp])

        symptoms\_index[all\_symptom] = i

    data\_dict = {"symptom\_index":symptoms\_index,"prediction\_classes":encoder.classes\_}

    return prediction(user\_symptoms, data\_dict, svm\_model, nb\_model, rf\_model)

#Prediction

def prediction(user\_symptoms, data\_dict, svm\_model, nb\_model, rf\_model):

    input\_symptoms = user\_symptoms.split(",")

    input = [0]\*len(data\_dict["symptom\_index"])

    for i in input\_symptoms:

        index = data\_dict["symptom\_index"][i]

        input[index] = 1

    input\_data = np.array(input).reshape(1,-1)

    svm = data\_dict["prediction\_classes"][svm\_model.predict(input\_data)[0]]

    nb = data\_dict["prediction\_classes"][nb\_model.predict(input\_data)[0]]

    rf = data\_dict["prediction\_classes"][rf\_model.predict(input\_data)[0]]

    result = mode([svm, nb, rf])[0][0]

    return result

##Login\_verifcation

def check\_details(email, password):

    global mail

    mail = email

    if validate\_login(email, password):

        flag = 0

        connection = sql.connect(host = "localhost", user = "root", password = "Dongle@123", database = "dds\_user")

        mysql = connection.cursor()

        mysql.execute("Select \* from users")

        for i in mysql:

            if i[0] == email:

                if i[1] == password:

                    flag = 0

                    main\_page()

                    break

                else:

                    flag = 0

                    msg.showwarning("Wrong!", "You have entered a wrong password.\nTry Again!!")

                    break

            else:

                flag = 1

        if flag == 1:

            msg.showinfo("Not Registered","We can't find you in our database.\nTry Signing Up!!")

#End Message

def thank\_you():

    clear\_everything()

    tk.Label(MainWin, text="ThankYou for using our software.",height=7,width=30,

             font=("Roboto bold",25), bg="lightgreen").grid(row=0, column=0)

    tk.Label(MainWin, text="Your diagnosis has been sent to your email.",

             font=("Roboto bold",11),bg ="lightgreen").grid(row=1, column=0)

    tk.Button(MainWin, text="×", font=("Arial bold",10),

              bg="red",fg="white",command=lambda:exit()).grid(row=2,column=0)

#Submission of Data

def click(event):

    print("clicked")

    selected = event.curselection()

    for i in selected:

        global input\_set

        if i not in input\_set:

            input\_set.add(event.get(i))

        print(event.get(i))

#Sending Diagnosis

def final\_submit(diagnosis):

    check = msg.askokcancel("Continue!!",

    "Your diagnosis statement will be sent to you on your registered e-mail.\n(Please bear with us it can take few minutes.)")

    if check:

        message = f"Dear {mail},\nThis is a probable disease predicted by our ML model,\n\tThe result of detection is \"{diagnosis}\"\nNote:This is just a prediction based on stored data.\nYou can consult a concerned doctor with this \'link\'"

##        smtp = smtplib.SMTP('smtp.gmail.com',587)

##        smtp.starttls()

##        smtp.login("some@gmail.com","okmmhluijiaaoaqe")

##        smtp.sendmail('some',mail,message)

        print(message)

        thank\_you()

    else:

        MainWin.destroy()

#Selected Inputs by User

def formatting():

    global input\_set

    input\_string = ",".join(input\_set)

    print("Input String is ",input\_string)

    diagnosis = preparation(input\_string.title())

    final\_submit(diagnosis)

def upper\_body\_symptoms():

    clear\_everything()

    msg.showinfo("Procedure","Please select all the symptoms you're having.\n(On the next page!!)")

    MainWin.title("Symptoms from Upper Body Parts")

    #Main Border after Main Window

    main\_border = tk.Frame(MainWin, bd=2, relief="sunken", background="lightgreen")

    main\_border.grid(row=0, column=0, padx=5, pady=4)

    eye\_neck\_border = tk.Frame(main\_border, relief="sunken")

    eye\_neck\_border.grid(row=0, column=0, padx=5, pady=4)

    #Eye Symptoms

    eye\_border = tk.Frame(eye\_neck\_border, bd=1, relief="sunken")

    eye\_border.grid(row=0, column=0, padx=5, pady=4)

    tk.Label(eye\_border, text="👀Eye", font=("Roboto bold",15)).grid(row=0, column=0)

    eye = tk.Listbox(eye\_border, width=20, height=7, selectmode=tk.MULTIPLE,

                     borderwidth=0, highlightthickness=0,

                     background=eye\_border.cget("background"))

    eye\_sb = tk.Scrollbar(eye\_border, orient="vertical", command=eye.yview)

    eye.configure(yscrollcommand=eye\_sb)

    eye\_sb.grid(row=1, column=1, sticky="ns")

    eye.grid(row=1, column=0)

    values = ['Sunken Eyes','Pain behind the eyes','Yellowing of eyes','Blurred And Distorted Vision','Redness of eyes','Watering From Eyes','Visual Disturbances']

    for i in values:

        eye.insert("end", i)

    tk.Button(eye\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(eye)).grid(row=2, column=0)

    #Neck Symptoms

    neck\_border = tk.Frame(eye\_neck\_border, bd=1, relief="sunken")

    neck\_border.grid(row=1, column=0, padx=5, pady=2)

    tk.Label(neck\_border, text="✨Neck", font=("Roboto bold",15)).grid(row=0, column=0)

    neck = tk.Listbox(neck\_border, width=20, height=7, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=neck\_border.cget("background"))

    neck\_sb = tk.Scrollbar(neck\_border, orient="vertical", command=neck.yview)

    neck.configure(yscrollcommand=neck\_sb)

    neck\_sb.grid(row=1, column=1, sticky="ns")

    neck.grid(row=1, column=0)

    values = ['Neck Pain','Enlarged Thyroid','Stiff Neck','Patches in throat','Cough', Phlegm','Throat irritation','Mucoid Sputum','Rusty Sputum', 'Blood in Sputum','Ulcers on tongue']

    for i in values:

        neck.insert("end", i)

    tk.Button(neck\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(neck)).grid(row=2, column=0)

    head\_in\_border = tk.Frame(main\_border, relief="sunken")

    head\_in\_border.grid(row=0, column=1, padx=5, pady=4)

    ##Head Symptoms

    head\_border = tk.Frame(head\_in\_border, bd=1, relief="sunken")

    head\_border.grid(row=0, column=1, padx=5, pady=4)

    tk.Label(head\_border, text="🧠Head", font=("Roboto bold",15)).grid(row=0, column=0)

    head = tk.Listbox(head\_border, width=21, height=18, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=head\_border.cget("background"))

    head\_sb = tk.Scrollbar(head\_border, orient="vertical", command=head.yview)

    head.grid(row=1, column=0)

    head\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Headache','Sinus Pressure','Dizziness','Slurred Speech','Unsteadiness','Depression','Irritability','Altered Sensorium','Lack of Concentration','Anxiety','Mood swings']

    for i in values:

        head.insert("end", i)

    tk.Button(head\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(head)).grid(row=2, column=0)

    nose\_face\_border = tk.Frame(main\_border, relief="sunken")

    nose\_face\_border.grid(row=0, column=2, padx=5, pady=4)

    ##Nose Symptoms

    nose\_border = tk.Frame(nose\_face\_border, bd=1, relief="sunken")

    nose\_border.grid(row=0, column=0, padx=5, pady=4)

    tk.Label(nose\_border, text="👃Nose", font=("Roboto bold",15)).grid(row=0, column=0)

    nose = tk.Listbox(nose\_border, width=20, height=7, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=nose\_border.cget("background"))

    nose\_sb = tk.Scrollbar(nose\_border, orient="vertical", command=nose.yview)

    nose.configure(yscrollcommand=nose\_sb)

    nose\_sb.grid(row=1, column=1, sticky="ns")

    nose.grid(row=1, column=0)

    values = ['Continuous Sneezing','Breathlessness','Runny Nose','Loss of smell', 'Red Sore around Nose']

    for i in values:

        nose.insert("end", i)

    tk.Button(nose\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(nose)).grid(row=2, column=0)

    face\_border = tk.Frame(nose\_face\_border, bd=1, relief="sunken")

    face\_border.grid(row=1, column=0, padx=5, pady=4)

    ##Face Symptoms

    tk.Label(face\_border, text="👩🏻‍🦲Face", font=("Roboto bold",15)).grid(row=0, column=0)

    face = tk.Listbox(face\_border, width=20, height=7, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=face\_border.cget("background"))

    face\_sb = tk.Scrollbar(face\_border, orient="vertical", command=face.yview)

    face.configure(yscrollcommand=face\_sb)

    face\_sb.grid(row=1, column=1, sticky="ns")

    face.grid(row=1, column=0)

    values = ['Puffy Face and Eyes','Drying and Tingling Lips','Pus filled pimples', 'Blackheads','Scarring','Skin Peeling']

    for i in values:

        face.insert("end", i)

    tk.Button(face\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(face)).grid(row=2, column=0)

    ##Button goes to mid\_body\_symptoms

    tk.Button(MainWin, text="Next..",background="salmon",

              font=("Roboto bold",10),width=65,

              command=lambda:mid\_body\_symptoms()).grid(row=1, column=0)

def mid\_body\_symptoms():

    clear\_everything()

    MainWin.title("Symptoms from Mid-Body Parts")

    mid\_main\_border = tk.Frame(MainWin, bd=2, relief="sunken", background="lightgreen")

    mid\_main\_border.grid(row=0, column=0, padx=5, pady=4)

    skin\_chest\_border = tk.Frame(mid\_main\_border, relief="sunken")

    skin\_chest\_border.grid(row=0, column=0, padx=5, pady=4)

    ##Skin Symptoms

    skin\_border = tk.Frame(skin\_chest\_border, bd=1, relief="sunken")

    skin\_border.grid(row=0, column=0, padx=5, pady=4)

    tk.Label(skin\_border, text="✨Skin", font=("Roboto bold",15)).grid(row=0, column=0)

    skin = tk.Listbox(skin\_border, width=20, height=7, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=skin\_border.cget("background"))

    skin\_sb = tk.Scrollbar(skin\_border, orient="vertical", command=skin.yview)

    skin.grid(row=1, column=0)

    skin\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Itching','Skin Rash','Nodal Skin Eruptions','Yellowish Skin', 'Bruising','Internal Itching','Toxic Look (typhos)', 'Dischromic  Patches','Blister','Yellow Crust Ooze']

    for i in values:

        skin.insert("end", i)

    tk.Button(skin\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(skin)).grid(row=2, column=0)

    ##Chest Symptoms

    chest\_border = tk.Frame(skin\_chest\_border, bd=1, relief="sunken")

    chest\_border.grid(row=1, column=0, padx=5, pady=4)

    tk.Label(chest\_border, text="✨Chest", font=("Roboto bold",15)).grid(row=0, column=0)

    chest = tk.Listbox(chest\_border, width=20, height=7, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=chest\_border.cget("background"))

    chest\_sb = tk.Scrollbar(chest\_border, orient="vertical", command=chest.yview)

    chest.grid(row=1, column=0)

    chest\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Chest Pain', 'Fast heart Rate', 'Palpitations']

    for i in values:

        chest.insert("end", i)

    tk.Button(chest\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(chest)).grid(row=2, column=0)

    body\_in\_border = tk.Frame(mid\_main\_border, relief="sunken")

    body\_in\_border.grid(row=0, column=1, padx=5, pady=4)

    ##Body Symptoms

    body\_border = tk.Frame(body\_in\_border, bd=1, relief="sunken")

    body\_border.grid(row=0, column=1, padx=5, pady=4)

    tk.Label(body\_border, text="🏋🏻‍♂️Body", font=("Roboto bold",15)).grid(row=0, column=0)

    body = tk.Listbox(body\_border, width=21, height=18, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=body\_border.cget("background"))

    body\_sb = tk.Scrollbar(body\_border, orient="vertical", command=body.yview)

    body.grid(row=1, column=0)

    body\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Shivering', 'Chills', 'Fatigue', 'Weight Gain', 'Cold hands and Feet', 'Weight Loss', 'Restlessness', 'Lethargy', 'High Fever', 'Sweating', 'Dehydration' ,'Back Pain', 'Mild Fever', 'Fluid Overload', 'Swelled Lymph Nodes', 'Malaise', 'Congestion', 'Swollen blood vessels', 'Movement Stiffness', 'Spinning movements', 'Weakness of one body side', 'Red Spots over body', 'Abnormal Menstruation']

    for i in sorted(values):

        body.insert("end", i)

    tk.Button(body\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(body)).grid(row=2, column=0)

    stomach\_muscle\_border = tk.Frame(mid\_main\_border, relief="sunken")

    stomach\_muscle\_border.grid(row=0, column=2, padx=5, pady=4)

    ##Stomach Symptoms

    stomach\_border = tk.Frame(stomach\_muscle\_border, bd=1, relief="sunken")

    stomach\_border.grid(row=0, column=0, padx=5, pady=4)

    tk.Label(stomach\_border, text="🍔Stomach", font=("Roboto bold",15)).grid(row=0, column=0)

    stomach = tk.Listbox(stomach\_border, width=20, height=7, selectmode=tk.MULTIPLE, borderwidth=0, highlightthickness=0, background=stomach\_border.cget("background"))

    stomach\_sb = tk.Scrollbar(stomach\_border, orient="vertical", command=stomach.yview)

    stomach.grid(row=1, column=0)

    stomach\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Stomach Pain', 'Acidity', 'Indigestion', 'Nausea', 'Loss of appetite', 'Swelling of stomach', 'Excessive Hunger', 'Belly Pain', 'Increased Appetite', 'Stomach Bleeding', 'Distention of Abdomen']

    for i in values:

        stomach.insert("end", i)

    tk.Button(stomach\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(stomach)).grid(row=2, column=0)

    ##Muscle Symptoms

    muscle\_border = tk.Frame(stomach\_muscle\_border, bd=1, relief="sunken")

    muscle\_border.grid(row=1, column=0, padx=5, pady=4)

    tk.Label(muscle\_border, text="🦾Muscle", font=("Roboto bold",15)).grid(row=0, column=0)

    muscle = tk.Listbox(muscle\_border, width=20, height=7, selectmode=tk.MULTIPLE,

                         borderwidth=0, highlightthickness=0,

                         background=muscle\_border.cget("background"))

    muscle\_sb = tk.Scrollbar(muscle\_border, orient="vertical", command=muscle.yview)

    muscle.grid(row=1, column=0)

    muscle\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Muscle Wasting', 'Cramps', 'Obesity', 'Muscle Weakness',

              'Muscle Pain']

    for i in values:

        muscle.insert("end", i)

    tk.Button(muscle\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(muscle)).grid(row=2, column=0)

    lower\_border = tk.Frame(MainWin, relief="sunken", background="lightgreen")

    lower\_border.grid(row=1, column=0, padx=5, pady=4)

    ##Button to lower\_body\_symptoms

    tk.Button(lower\_border, text="Next..",background="salmon",

              font=("Roboto bold",10),width=30,

              command=lambda:lower\_body\_symptoms()).grid(row=0, column=0)

    tk.Label(lower\_border, text="  ", background="lightgreen").grid(row=0, column=1)

    tk.Button(lower\_border, text="Previous Menu",background="salmon",

              font=("Roboto bold",10),width=30,

              command=lambda:upper\_body\_symptoms()).grid(row=0, column=2)

def lower\_body\_symptoms():

    clear\_everything()

    MainWin.title("Symptoms from Lower Body Parts")

    lower\_main\_border = tk.Frame(MainWin, bd=1, relief="sunken", bg="lightgreen")

    lower\_main\_border.grid(row=0, column=0, padx=5, pady=10)

    limbs\_border = tk.Frame(lower\_main\_border, bd=1, relief="sunken")

    limbs\_border.grid(row=0, column=0, padx=5, pady=5)

    ##limbs Symptoms

    tk.Label(limbs\_border, text="🙋🏻‍♂️Limbs", font=("Roboto bold",15)).grid(row=0, column=0)

    limbs = tk.Listbox(limbs\_border, width=21, height=18, selectmode=tk.MULTIPLE,

                         borderwidth=0, highlightthickness=0,

                         background=limbs\_border.cget("background"))

    limbs\_sb = tk.Scrollbar(limbs\_border, orient="vertical", command=limbs.yview)

    limbs.grid(row=1, column=0)

    limbs\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Weakness in limbs', 'Swollen Legs', 'Brittle Nails', 'Swollen Extremeties', 'Knee Pain', 'Hip Joint Pain', 'Swelling Joints', 'Loss of balance', 'Prominent veins on calf', 'Painful Walking', 'Small dents in nails', 'inflammatory nails', 'Joint Pain']

    for i in values:

        limbs.insert("end", i)

    tk.Button(limbs\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(limbs)).grid(row=2, column=0)

    excretion\_border = tk.Frame(lower\_main\_border, bd=1, relief="sunken")

    excretion\_border.grid(row=0, column=1, padx=5, pady=4)

    ##Excretion Symptoms

    tk.Label(excretion\_border, text="🚽Excretion", font=("Roboto bold",15)).grid(row=0, column=0)

    excretion = tk.Listbox(excretion\_border, width=22, height=18, selectmode=tk.MULTIPLE,

                         borderwidth=0, highlightthickness=0,

                         background=excretion\_border.cget("background"))

    excretion\_sb = tk.Scrollbar(excretion\_border, orient="vertical", command=excretion.yview)

    excretion.grid(row=1, column=0)

    excretion\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Vomiting', 'Burning Micturition', 'Spotting  Urination', 'Dark Urine', 'Yellow Urine', 'Pain during Bowel Movements','Pain in Anal Region', 'Bloody Stool', 'Irritation in Anus', 'Bladder discomfort', 'Foul Smell of Urine', 'Passage of Gases', 'Polyuria']

    for i in values:

        excretion.insert("end", i)

    tk.Button(excretion\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(excretion)).grid(row=2, column=0)

    ##Previous History Symptoms

    prev\_border = tk.Frame(lower\_main\_border, bd=1, relief="sunken")

    prev\_border.grid(row=0, column=2, padx=5, pady=4)

    tk.Label(prev\_border, text="🔮Previous History", font=("Roboto bold",15)).grid(row=0, column=0)

    prev = tk.Listbox(prev\_border, width=22, height=18, selectmode=tk.MULTIPLE,borderwidth=0, highlightthickness=0, background=prev\_border.cget("background"))

    prev\_sb = tk.Scrollbar(prev\_border, orient="vertical", command=prev.yview)

    prev.grid(row=1, column=0)

    prev\_sb.grid(row=1, column=1, sticky='ns')

    values = ['Irregular Sugar Level', 'Diarrhoea', 'Acute Liver failure','Extra Marital Contacts', 'Family History','Receiving blood transfusion', 'Receiving Unsterile Injections','Coma', 'History of Alcohol Consumption']

    for i in values:

        prev.insert("end", i)

    tk.Button(prev\_border, text="Submit",background="lightsalmon",

              font=("Roboto bold",10), width=10,

              command=lambda:click(prev)).grid(row=2, column=0)

    lower\_border = tk.Frame(MainWin, relief="sunken", background="lightgreen")

    lower\_border.grid(row=1, column=0, padx=5, pady=4)

    ##Button to Final Last Submit

    tk.Button(lower\_border, text="Final Submit",background="salmon",

              font=("Roboto bold",10),width=30,

              command=lambda:formatting()).grid(row=0, column=0)

    tk.Label(lower\_border, text="  ", background="lightgreen").grid(row=0, column=1)

    tk.Button(lower\_border, text="Previous Menu",background="salmon",

              font=("Roboto bold",10),width=30,

              command=lambda:mid\_body\_symptoms()).grid(row=0, column=2)

#Comparision of Image with Defected Images

def check\_image(file):

    final = tk.Label(MainWin, text="Checking...", width=20,

             font=("Arial Bold", 24), padx=5, pady=0,bg="light green"

             ).grid(row=4, column = 0, padx=5, pady=10)

    image = Image.open(file)

    x = np.array(image.histogram())

    dis\_dict = {}

    tally = {}

    for j in range(47):

        y\_file = f"Eye\_diseases/Cataracts/{j}.jpeg"

        rest = Image.open(y\_file)

        y = np.array(rest.histogram())

        try:

            if len(x) == len(y):

                error = np.sqrt(((x-y) \*\* 2).mean())

                error = str(error)[:2]

                actual\_error = float(100) - float(error)

            diff = ImageChops.difference(image, rest).getbbox()

            if diff:

                tally[j] = actual\_error

            else:

                tally[j] = actual\_error

                continue

        except ValueError as identifier:

            print("Matching Images in percentage : ", actual\_error, " %")

    dis\_dict["Cataract"] = max(tally.values())

    tally = {}

    for j in range(17):

        y\_file = f"Eye\_diseases/Glaucoma/{j}.jpeg"

        template = Image.open(y\_file)

        y = np.array(template.histogram())

        try:

            if len(x) == len(y):

                error = np.sqrt(((x-y) \*\* 2).mean())

                error = str(error)[:2]

                actual\_error = float(100) - float(error)

            diff = ImageChops.difference(image, template).getbbox()

            if diff:

                tally[j] = actual\_error

            else:

                tally[j] = actual\_error

                continue

        except ValueError as identifier:

            print("Matching Images in percentage : ", actual\_error, " %")

    dis\_dict["Glaucoma"] = max(tally.values())

    tally = {}

    for j in range(26):

        y\_file = f"Eye\_diseases/Uveitis/{j}.jpeg"

        template = Image.open(y\_file)

        y = np.array(template.histogram())

        try:

            if len(x) == len(y):

                error = np.sqrt(((x-y) \*\* 2).mean())

                error = str(error)[:2]

                actual\_error = float(100) - float(error)

            diff = ImageChops.difference(image, template).getbbox()

            if diff:

                tally[j] = actual\_error

            else:

                tally[j] = actual\_error

                continue

        except ValueError as identifier:

            print("Matching Images in percentage : ", actual\_error, " %")

    dis\_dict["Uveitis"] = max(tally.values())

    diagnosis = ""

    for i in dis\_dict.keys():

        if dis\_dict[i] == 100:

            diagnosis = i+" or"

            break

        if dis\_dict[i] >= 85:

            diagnosis += i+" or "

    if diagnosis == "":

        final\_submit("Nothing")

    else:

        final\_submit(diagnosis[:-3])

#Loading Image Selection Page

def get\_image\_dialog():

    global img

    types = [("Jpeg Files","\*.jpeg"), ('Jpg Files','\*.jpg')]

    filename = filedialog.askopenfilename(filetypes = types)

    img = ImageTk.PhotoImage(file=filename)

    b2=tk.Label(MainWin, image=img, bg="light green")

    b2.grid(row=2, column=0)

    submit\_image = tk.Button(MainWin, text="Check", width=15,

            font=("Roboto bold", 12), command=lambda:check\_image(filename)

            ).grid(row=3, column=0, padx=5, pady=10)

##Detection\_through\_Images

def main\_image\_page():

    clear\_everything()

    tk.Label(MainWin, text = "Select the Image", width=29,

             font=("Arial Bold", 24), padx=5, pady=30,bg="light green"

             ).grid(row = 0, column = 0, pady=7)

    upload\_button = tk.Button(MainWin, text="Upload Image", width=25,

            font=("Roboto bold", 12), command=lambda:get\_image\_dialog()

            ).grid(row=1, column=0, padx=5, pady=10)

def camera():

    msg.showwarning("Comming Soon", "This feature will get added on next update.")

##Main\_Page

def main\_page():

    clear\_everything()

    MainWin.title(f"Logged in as {mail}")

    tk.Label(MainWin, text = "Disease Detection System", width=29,

             font=("Arial Bold", 24), padx=5, pady=30,bg="light green"

             ).grid(row = 0, column = 0, pady=7)

    tk.Button(MainWin, text="Detection through Inputs", width=25,

            font=("Roboto bold", 12), command=lambda:upper\_body\_symptoms()

            ).grid(row=1, column=0, padx=5, pady=10)

    tk.Button(MainWin, text="Detection through Images",  width=25,

            font=("Roboto bold", 12), command=lambda:main\_image\_page()

            ).grid(row=2, column=0, padx=5, pady=10)

    tk.Button(MainWin, text="Detection through Camera",  width=25,

            font=("Roboto bold", 12), command=lambda:camera()

            ).grid(row=3, column=0, padx=5, pady=10)

##Login\_Page

def login\_widget():

    MainWin.title("Login..")

    #Labels and Textboxes

    tk.Label(MainWin, text = "Disease Detection System", width=29,

             font=("Arial Bold", 24),

             bg="light green").grid(row = 0, column = 0, pady=7)

    tk.Label(MainWin, text = "Login Page", bg="light green",

             font=("Roboto Bold", 12)

             ).grid(row = 1, column = 0)

    main\_border = tk.Frame(MainWin, bd=2, relief="sunken", bg="lightgreen")

    main\_border.grid(row=2, column=0, padx=5, pady=10)

    input\_border = tk.Frame(main\_border, relief="sunken",bg="lightgreen")

    input\_border.grid(row=0, column=0, padx=5, pady=10)

    #email

    E\_mail = tk.Label(input\_border,text = "Email ID ",bg="lightgreen", font=("Roboto Bold", 11))

    E\_mail.grid(row = 0, column = 0, padx=7, pady=7)

    email\_tex = tk.Entry(input\_border)

    email\_tex.grid(row = 0, column = 2, padx=7, pady=7)

    #password

    Pass\_word = tk.Label(input\_border, text = "Password",bg="lightgreen",font=("Roboto Bold", 11))

    Pass\_word.grid(row=2, column = 0, padx=7, pady=7)

    pass\_wrd = tk.Entry(input\_border, show = '\*')

    pass\_wrd.grid(row=2, column = 2, padx=7, pady=7)

    submit\_border = tk.Frame(main\_border, relief="sunken", bg="lightgreen")

    submit\_border.grid(row=3, column=0, padx=5, pady=10)

    tk.Button(submit\_border, text = "Login", width=26,

              font=("Roboto Bold", 12),

              command = lambda:check\_details(email\_tex.get(),pass\_wrd.get())

              ).grid(row=0, column = 0, padx=10, pady=10)

    tk.Button(submit\_border, text = "Sign Up", width=26,

              font=("Roboto Bold", 12),

              command = lambda:sign\_up\_widget()

              ).grid(row=0, column = 1, padx=10, pady=10)

##OTP widget

def show\_otp\_widget(email, paswrd, One\_tp, otp, submit\_otp):

    if validate(email,paswrd):

        #OTP generation

        global OTP

        dig = "0123456789"

        for i in range(6):

            OTP += dig[math.floor(random.random()\*10)]

        gen\_OTP = "Welcome to \"Disease Detection System\" \nPlease verify your,\nEmail Id : " + email + "\nPassword : " + paswrd + " \nwith this OTP : " + OTP + "."

##        smtp = smtplib.SMTP('smtp.gmail.com',587)

##        smtp.starttls()

##        user\_email = email

##        smtp.login("some@gmail.com","okmmhluijiaaoaqe")

##        smtp.sendmail('Some',user\_email,gen\_OTP)

        print(gen\_OTP)

        #OTP Widget

        One\_tp.grid(row=0, column=0)

        otp.grid(row=0, column = 1)

        submit\_otp.grid(row=1, column=1)

def verification(temp, email, passwrd):

    global OTP

    if temp.get() == OTP:

        go\_ahead(email,passwrd)

    else:

        msg.showwarning("Wrong!!","Entered Wrong OTP")

def validate\_login(email, pswrd):

    em\_txt = email

    if em\_txt == "":

        msg.showwarning("Empty!!","Email ID cannot be empty.")

    elif em\_txt[len(em\_txt)-10:len(em\_txt)] != '@gmail.com':

        msg.showwarning("Invalid!!","Please Enter a valid Email ID.\nEmail must include \"@gmail.com\"")

    elif pswrd == "":

        msg.showwarning("Empty!!","Password cannot be empty.")

    else:

        return True

def validate(email, pswrd):

    em\_txt = email

    flag = 0

    connection = sql.connect(host = 'localhost', user = 'root', password = 'Dongle@123', database="dds\_user")

    point = connection.cursor()

    point.execute("select \* from users")

    for i in point:

        if i[0] == email:

            msg.showwarning("Duplicate","Email already exists!!")

            flag = 1

            break

    if flag == 1:

        sign\_up\_widget()

    else:

        if em\_txt == "":

            msg.showwarning("Empty!!","Email ID cannot be empty.")

        elif em\_txt[len(em\_txt)-10:len(em\_txt)] != '@gmail.com':

            msg.showwarning("Invalid!!","Please Enter a valid Email ID.\nEmail must include \"@gmail.com\"")

        elif pswrd == "":

            msg.showwarning("Empty!!","Password cannot be empty.")

        else:

            return True

def go\_ahead(email, passwrd):

    connection = sql.connect(host = 'localhost', user = 'root', password = 'Dongle@123', database="dds\_user")

    point = connection.cursor()

    sql\_query = "insert into users (email, password) values(%s,%s)"

    point.execute(sql\_query,(email, passwrd))

    connection.commit()

    if point.rowcount != 0:

        clear\_everything()

        login\_widget()

    else:

        msg.showwarning("Error!","Something wrong with the insertion.")

#MainWin(Login or SignUp Page)

MainWin = tk.Tk()

MainWin.geometry("562x420+150+150")

MainWin.configure(bg="lightgreen")

MainWin.resizable(False,False)

MainWin.iconbitmap("doc.ico")

#properly centers the window

'''MainWin.eval('tk::PlaceWindow . center')'''

def sign\_up\_widget():

    clear\_everything()

    MainWin.title("Sign Up..")

    #Labels and Textboxes

    tk.Label(MainWin, text = "Disease Detection System", width=29,

             bg="light green", font=("Arial Bold", 24)

             ).grid(row = 0, column = 0, pady=7)

    tk.Label(MainWin, text = "Sign Up Page",

             bg="light green", font=("Roboto Bold", 12)).grid(row = 1, column = 0)

    main\_border = tk.Frame(MainWin, bd=2, relief="sunken",bg="lightgreen")

    main\_border.grid(row=2, column=0, padx=5, pady=10)

    input\_border = tk.Frame(main\_border, relief="sunken",bg="lightgreen")

    input\_border.grid(row=0, column=0, padx=5, pady=10)

    #email

    E\_mail = tk.Label(input\_border,text = "Email ID ", bg="lightgreen",

                      font=("Roboto bold",11))

    E\_mail.grid(row = 0, column = 0, padx=7, pady=7)

    email\_tex = tk.Entry(input\_border)

    email\_tex.grid(row = 0, column = 2, padx=7, pady=7)

    #password

    Pass\_word = tk.Label(input\_border, text = "Password", bg="lightgreen"

                         ,font=("Roboto Bold", 11))

    Pass\_word.grid(row=2, column = 0, padx=7, pady=7)

    pass\_wrd = tk.Entry(input\_border, show = '\*')

    pass\_wrd.grid(row=2, column = 2, padx=7, pady=7)

    #OTP\_Widget

    otp\_border = tk.Frame(main\_border, relief="sunken", bg="lightgreen")

    otp\_border.grid(row=3, column=0, padx=5, pady=10)

    One\_tp = tk.Label(otp\_border, text = "OTP ",font=("Roboto Bold", 11), bg="lightgreen")

    otp = tk.Entry(otp\_border)

    submit\_otp = tk.Button(otp\_border, text = "Submit",

                           font=("Roboto Bold", 11),

                           command = lambda:verification(otp, email\_tex.get(), pass\_wrd.get()))

    submit\_border = tk.Frame(main\_border, relief="sunken",bg="lightgreen")

    submit\_border.grid(row=2, column=0, padx=5, pady=10)

    #lambda is important here.

    tk.Button(submit\_border, text = "Sign Up", width=30,

              font=("Roboto Bold", 12),

              command = lambda:show\_otp\_widget(email\_tex.get(),pass\_wrd.get(),

                                             One\_tp, otp, submit\_otp)

            ).grid(row=0, column = 0, padx=10, pady=10)

login\_widget()

MainWin.mainloop()

##main\_page()

**4.2 TESTING**

The final phase of application development is software testing, during which software testers assess code by challenging it. This review might end quickly or keep going until all parties are happy. Software testing finds faults and problems early in the development process, allowing them to be repaired before the product is released. This strategy makes sure that only high-quality goods are offered to customers, which boosts their happiness and confidence. Although there are many different kinds of software testing, dynamic testing, and static testing are the two most common types. Static testing looks at the program's code and related documentation, whereas dynamic testing evaluates the software as it is being used. Static and dynamic approaches are frequently combined.

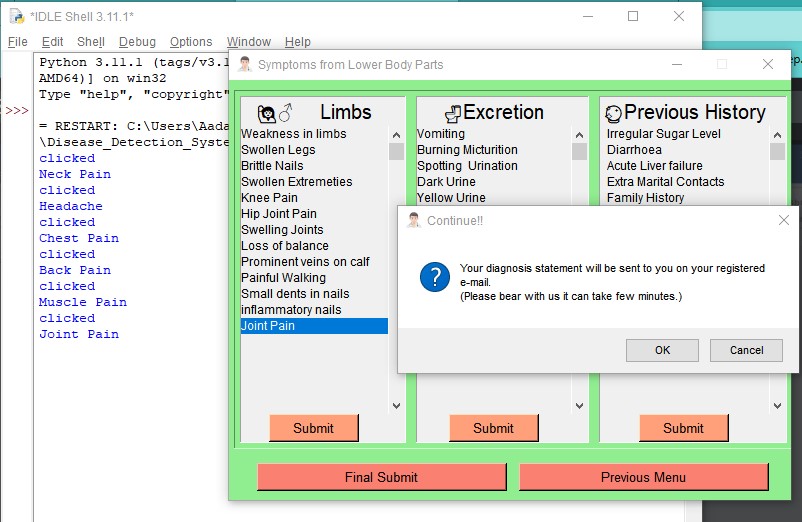
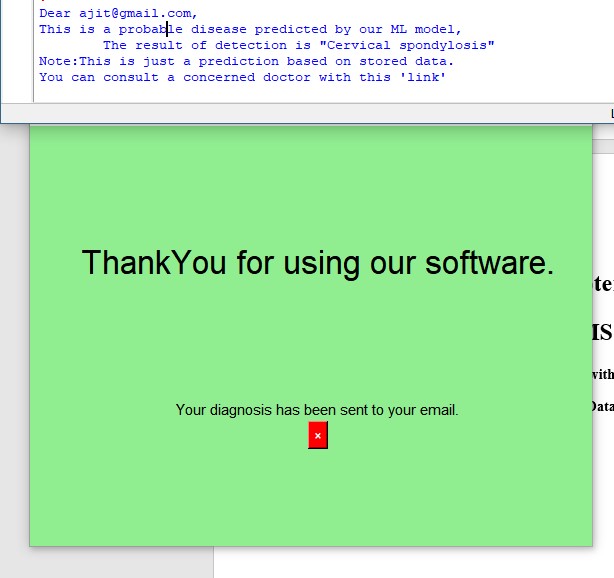
**UNIT testing**: The smallest testable components of an application, known as units, are separately and independently examined for appropriate operation as part of the unit testing phase of software development. Software developers and occasionally QA employees use this testing methodology when the software is still in the development stage. Unit testing's primary goal is to separate written code for testing to see if it functions as intended. Unit tests can be run automatically or by hand. An intuitive document outlining each stage in the process may be created for those using a manual method, however, automated testing is the more popular way to run unit tests. Automated methods frequently create test cases using a testi

Figure 7: Image of Unit Testing

ng framework. Additionally, these frameworks are configured to identify and report any failed test cases while also giving a list of all test cases.

Figure 8: Output sent to the user mail

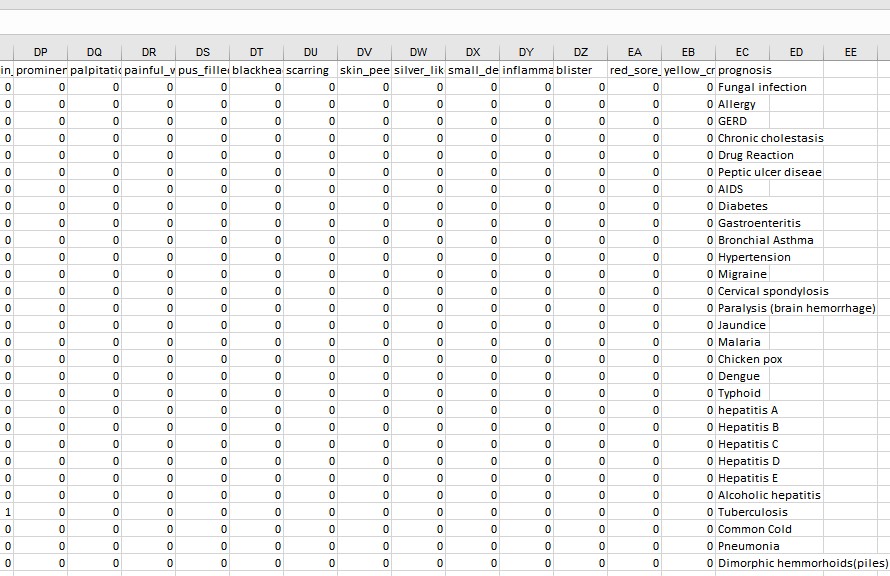
****

**4.3 TEST CASES**

How to test a system, piece of software, or application is specified in test cases. A test case is a unique set of instructions or actions that a tester must follow to verify a particular feature of a product's or application's functionality.

The firm needs to adopt a multifaceted strategy that assesses both the front and back ends of the product to validate and verify system functionality. The many sorts of test cases can be categorized in a variety of ways. Starting with the formal and informal categories is a good place to start.

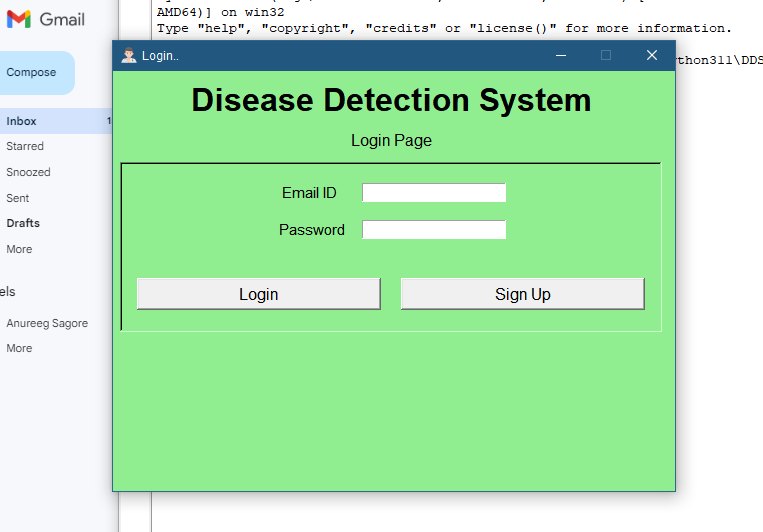
**Formal test**: With these kinds of test cases, the tester creates a test in which the inputs, such as the preconditions and test data, are all known and specifically described. Formal tests use specified input, which means that the inputs deliver an anticipated output that the test then tries to verify.

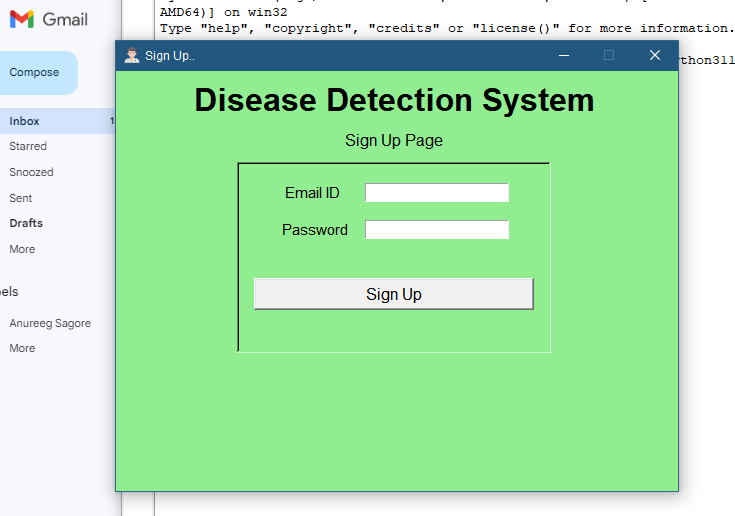
**Informal test**: Informal test cases, on the other hand, lack known inputs and outputs. These kinds of test cases are carried out by testers to learn and record the results, which can offer fascinating conclusions about digital quality.

**Chapter 5**

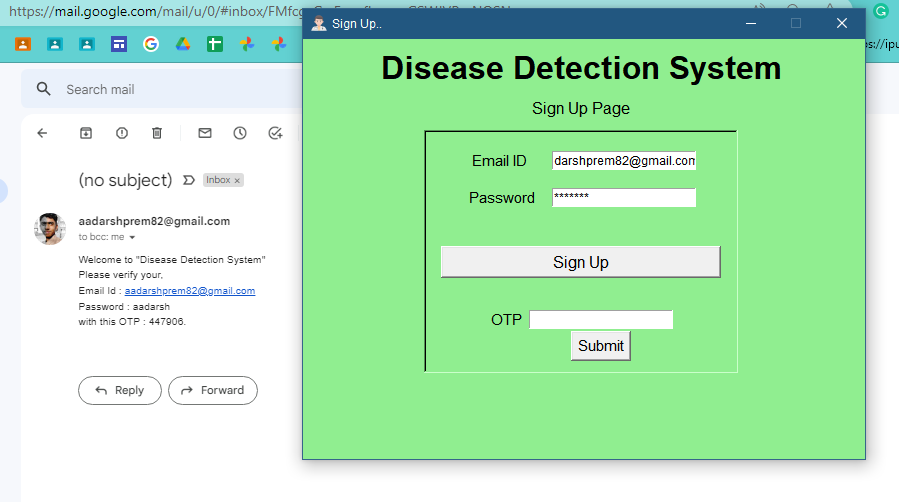
**OUTPUT FORMS & REPORTS**

**5.1 INPUT/OUTPUT SCREENSHOTS with SAMPLE DATA**

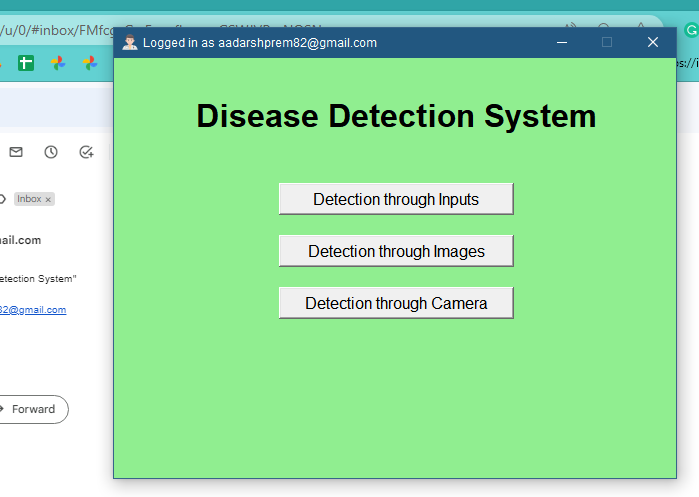
**Login Page**

**Signup Page**

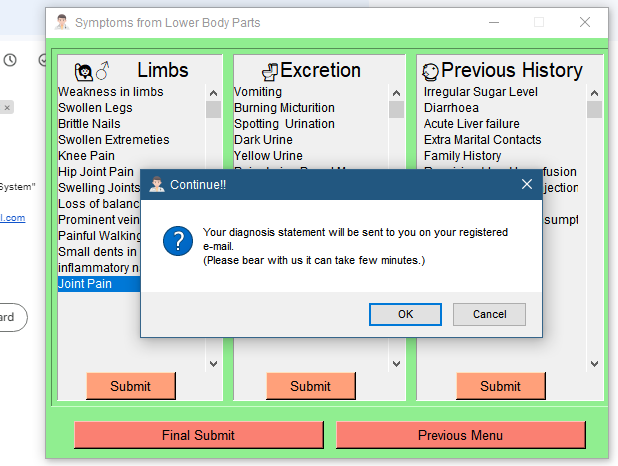
**OTP Submission Page**

****

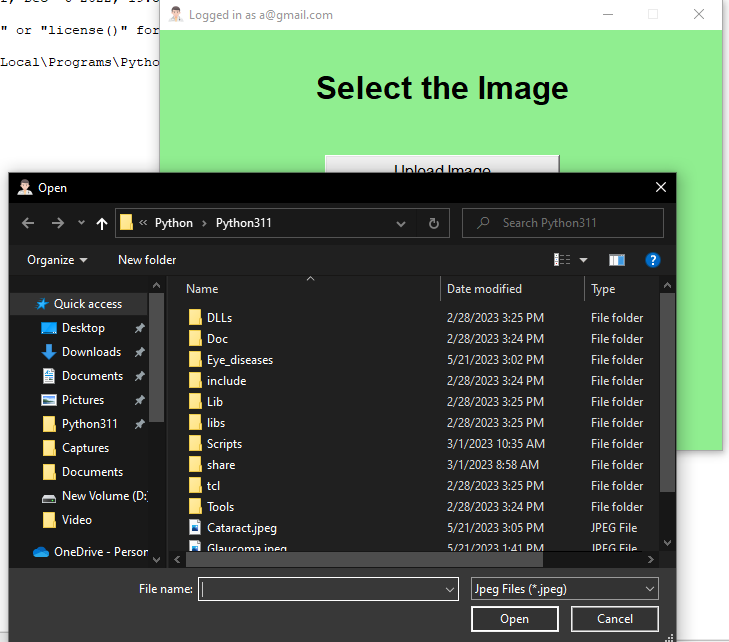
**Input Page**

****

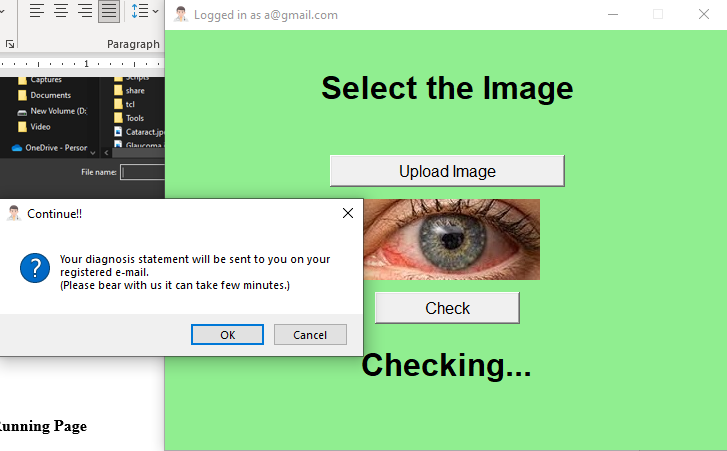
**Selection Page**

****

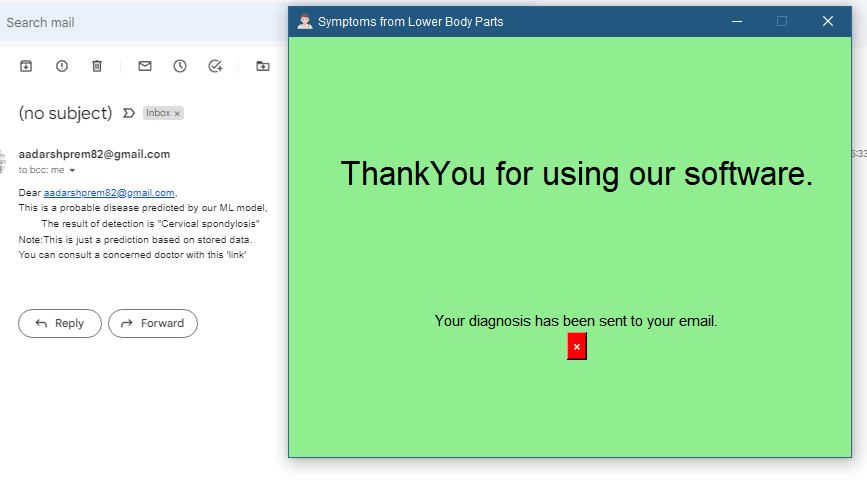
**Image Selection Page**

****

**Running Page**

****

**Ending Page**

****

**Chapter 6**

**CONCLUSION & FUTURE ENHANCEMENT(S)**

**6.1 CONCLUSION**

Ultimately, I would like to indicate that this project's Disease detection system using Python with machine learning is very useful in everyone's day-to-day life, but it is incredibly significant for the healthcare sector because they are the ones who use these systems daily to predict the diseases of the patients based on their general information and the symptoms that they have experienced.

Today's health industry plays an important role in curing the disease of the patient, so this is some kind of help for the help industry to tell the user, and it is also useful for the user if he/she does not want to go to the hospital or any other clinic, so simply through entering the symptoms.

All other valuable information the user can get to know the disease he/she is suffering from, and the entire health industry can benefit from the system by simply asking about the user's symptoms and entering them into the system, and in just a few seconds, they can tell the exact and, to a certain extent, accurate disease.

**6.2 FUTURE SCOPE**

In summary, the project's future scope revolves around the preservation of information about:

We can provide more advanced software or disease detection systems, as well as additional facilities.

* To make the platform available globally, we will host it on online servers.
* Categorize the data into different categories to reduce database query overload.
* Integrate a backup mechanism to take regular backups of the code base and database on different servers.
* Detection with the help of camera.

The above-mentioned improvements can be made to increase the applicability and usage of this project. Furthermore, as can be seen, today's players are versatile, implying that there is room for introducing a method to maintain the disease detection system.

We have left all options open so that if the user has any additional requirements for the system's enhancement, they can be implemented.

**BIBLIOGRAPHY**

**Python Installation**

[www.python.org/downloads](http://www.python.org/downloads)

**Package Installations**

Scikit Learn Library

<https://pypi.org/project/scikit-learn>

Scipy Library

<https://pypi.org/project/scipy>

Pandas Library

<https://pypi.org/project/pandas>

Pillow Library

[https://pypi.org/project/Pillow](https://pypi.org/project/pandas)

**Datasets for Training and Testing**

<https://github.com/aadarshprem82/Disease_Detection_System/blob/d646d7727410612697e785504b2fbfb1a8d8148d/Training.csv>

https://github.com/aadarshprem82/Disease\_Detection\_System/tree/d646d7727410612697e785504b2fbfb1a8d8148d/Eye\_diseases