SMART HEALTH CARD USING MACHINE LEARNING

A PROJECT REPORT

In partial fulfilment of requirements for the Award of

DIPLOMA IN INFORMATION TECHNOLOGY

Submitted By :

Tanvi Vijay Surve. (SS22IF004)

Geeta Vitthal Satpute. (SS22IF005)

Meghana Samadhan Badgujar. (SS22IF012)

Akshata Anil Dicholkar. (SW22IF001)

Under The Guidance Of

Prof . Namrata Ashok Wankhede

DEPARTMENT OF INFORMATION TECHNOLOGY

Government Polytechnic Mumbai

GOVERNMENT POLYTECHNIC MUMBAI

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project report entitled, "**SMART HEALTH CARD USING MACHINE LEARNING**" is the bonafide work of "**Tanvi Vijay Surve (SS22IF004) , Geeta Vitthal Satpute (SS22IF005),Meghana Samadhan Badgujar(SS22IF012),Akshata Anil Dicholkar (SW22IF001)**"submitted in partial fulfilment of the requirements for the award of Diploma in Information Technology of Government Polytechnic Mumbai.

Dr. Sachin Bharatkar — **HEAD OF THE DEPARTMENT**

Prof .Namrata Ashok Wankhede — **PROJECT GUIDE**

Dr. Nitiket N. Mhala.

PRINCIPAL

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled “SMART HEALTH CARD USING MACHINE LEARNING” is being submitted by us towards the partial fulfilment of the requirement for the award of Diploma in Information Technology is a project work carried by us under the supervision of Prof. Namrata Ashok Wankhede and have not been submitted anywhere else.

We will be solely responsible if any kind of plagiarism is found.

Date :

Name of students	**Enrollment No.**	**Signature**
Tanvi Vijay Surve	SS22IF004	
Geeta Vitthal Satpute	SS22IF005	
Meghana Samadhan Badgujar	SS22IF012	
Akshata Anil Dicholkar	SW22IF001	

ACKNOWLEDGEMENT

The project is a huge team effort. Our team expresses our deepest gratitude and thanks to the following people to have helped us to achieve our work.

We would like to thank, Mr. Bharatkar, Prof. Namrata Wankhede for guiding us and helping in time of need.

Our team extends thanks to other faculties of our collage whom we have approached for the academic help with regards to our project.

Thanks to all our teachers in the past who have inculcated in us values and work habit, that have allowed us to create the level of success that we have achieved today in our project team work.

Tanvi Vijay Surve.
(SS22IF004)

Geeta Vitthal Satpute.
(SS22IF005)

Meghana Samadhan Badgujar.
(SS22IF012)

Akshata Anil Dicholkar.
(SW22IF001)

ABSTRACT

There is limited preventive health care and services to promote optimal health and wellness, and avert worsening of sequelae for children and adults with disabilities. Across the healthcare continuum, integrated approaches are needed to simultaneously address the many risk factors and conditions, as well as the medical. functional and societal limitations including determinants that influences the health and wellbeing of persons with disabilities. The prediction of disease at earlier stage becomes important task. But the accurate prediction on the basis of symptoms becomes too difficult for doctor. The correct prediction of disease is the most challenging task. To overcome this problem data mining plays an important role to predict the disease. Medical science has large amount of data growth per year. Due to increase amount of data growth in medical and healthcare field the accurate analysis on medical data which has been benefits from early patient care. With the help of disease data, data mining finds hidden pattern information in the huge amount of medical data. We proposed general disease prediction based on symptoms of the patient. For the disease prediction, we use K-Nearest Neighbor (KNN) and Convolutional neural network (CNN) machine learning algorithm After general disease prediction, this system able to gives the risk associated with general disease which is lower risk of general disease or higher. Suggesting diet and appropriate exercise is another merit of proposed system.

Keywords : Health Card, CNN, KNN, Machine learning, Disease Prediction.

INDEX

Chapter 1 :-

INTRODUCTION

1.1 Introduction

The prediction of disease at earlier stage becomes important task. But the accurate prediction on the basis of symptoms becomes too difficult for doctor. There is a need to study and make a system which will make it easy for end users to predict the harmonic diseases without visiting physician or doctor for diagnosis. Additionally, in terms of personalized healthcare and disease prevention services, these depend primarily on the strategy used to derive knowledge from the analysis of lifestyle factors and activities. Through the use of intelligent data retrieval and classification models, it is possible to study disease, or even predict any abnormal health conditions. To predict such abnormality, the Convolutional neural network (CNN) model is used, which can detect the knowledge related to disease prediction accurately from unstructured medical health records. However, CNN uses a large amount of memory if it uses a fully connected network structure. Moreover, the increase in the number of layers can lead to an increase in the complexity analysis of the model.

1.2 Background

1940s-1950s: Early Developments in Healthcare Data Systems :- The foundation of healthcare data management began with early computing

systems designed for large-scale scientific and medical research. These early computers, such as ENIAC, were primarily used for complex calculations but marked the start of technology's role in healthcare, albeit in a limited capacity.

1960s-1970s: Healthcare Information Systems and Data Storage :- In the 1960s, as mainframe computers became available to large organizations and research institutions, hospitals began using computer systems to store patient information. Mainframes enabled better record-keeping, and early networks like ARPANET set the stage for information sharing, which would later evolve into more interconnected health information systems.

1980s: Personal Computing and Electronic Health Records (EHRs) :- With the rise of personal computers and user-friendly graphical interfaces, healthcare facilities started exploring electronic health records (EHRs). Medical data could now be stored electronically, making patient information more accessible to healthcare providers. This period also saw the development of specialized software for healthcare, which allowed for more structured data management.

1990s: Digital Medical Records and the Internet :- The introduction of the World Wide Web opened new possibilities for healthcare, including easier access to medical information and the beginnings of telemedicine. EHR systems became more sophisticated, allowing hospitals and clinics to securely store and share patient data. Healthcare providers started using web-based tools for research, diagnostics, and patient education.

2000s: Emergence of Health Information Exchange (HIE) and Mobile Health :- As the internet evolved into Web 2.0, healthcare technology also advanced. Health Information Exchange (HIE) networks allowed patient data to be shared

across different healthcare systems, enhancing continuity of care. Mobile health apps emerged, providing patients with new tools for tracking health metrics and accessing medical information.

2010s: Big Data, Machine Learning, and Predictive Analytics :- The healthcare industry saw a rapid increase in the use of big data and machine learning. These technologies enabled the analysis of vast amounts of medical data, leading to better insights into disease patterns, predictive diagnostics, and personalized treatment options. Cloud computing allowed for scalable storage solutions, facilitating access to medical records and data analysis from anywhere.

2020s and Beyond: Smart Health Cards, Artificial Intelligence, and Remote Care :- The 2020s have brought advancements in artificial intelligence, Internet of Things (IoT) devices, and smart health solutions, making healthcare more accessible and data-driven. The COVID-19 pandemic accelerated the use of digital healthcare technologies, including telemedicine, remote monitoring, and AI-powered health predictions. Smart health card systems, like the project in focus, use machine learning algorithms to support personalized healthcare by providing disease predictions, treatment recommendations, and digital medical histories accessible via unique digital IDs. Future developments may include enhanced accuracy of health predictions, integration with biometric data, and seamless connections to government health systems, revolutionizing healthcare accessibility and preventive care for patients worldwide.

1.3 Objective

The prediction of disease at earlier stage becomes important task. But the accurate prediction on the basis of symptoms becomes too difficult for doctor. There is a need to study and make a system which will make it easy for end

users to predict the harmonic diseases without visiting physician or doctor for diagnosis.

In order to improve the comprehensiveness and pertinence of the medical examination, we use healthcare big data analysis combined with deep learning technology to provide patients with potential diseases which is usually neglected for lacking of professional knowledge, so that patients can do targeted medical examinations to prevent health condition from getting worse.

- **The Main objectives are discussed below:**

1. To predict disease from symptoms and health history using CNN
2. To calculate severity of disease with the help of machine learning
3. To assist user in choosing correct diet and exercise.

1.4 Scope and Purpose

1.4.1. Scope

✓ Disease Prediction & Classification
Using machine learning algorithms, primarily Convolutional Neural Networks (CNN), the system predicts diseases or abnormal health conditions at an early stage based on symptoms and user health data.

✓ Personalized Health Monitoring
The system provides users with personalized health insights, suggesting preventive measures, lifestyle changes, and monitoring progress based on ongoing health data.

✓ Data Collection & Integration
The system will collect and integrate user-provided data, including symptoms, medical history, and lifestyle factors, through an intuitive user interface.

✓ **Health Records Management**
Secure storage and management of health records, enabling easy access and analysis of historical health data for more accurate disease predictions.

✓ **User-Friendly Interface**
A web or mobile-based platform that allows users to easily input health data, view disease predictions, and track progress over time.

✓ **Real-Time Health Analysis**
Continuous monitoring and real-time analysis of health data to detect changes or potential risks, providing timely alerts and suggestions for preventive care.

✓ **Security & Privacy**
Implementation of encryption and secure data management protocols to ensure user data is protected in compliance with healthcare data regulations (e.g., HIPAA, GDPR).

✓ **Machine Learning Model Training & Evaluation**
Development and continuous refinement of the machine learning model using existing datasets, with ongoing validation to ensure prediction accuracy.

✓ **Disease Prediction Accuracy**
Focus on improving prediction accuracy, minimizing false positives and false negatives, and refining the model based on new data inputs.

✓ **Health Recommendations**
Based on predicted diseases, the system will offer recommendations such as lifestyle changes, preventive measures, or when to consult a healthcare professional.

✓ **Scalability & Performance**
Design of the system to handle large datasets and a growing number of users while maintaining high performance and speed.

The Smart Health Card Using Machine Learning project aims to create an accessible, efficient, and secure tool for early disease detection, personalized

health monitoring, and proactive disease prevention. It provides a seamless platform for users to input their health data, receive disease predictions, and act on preventive healthcare suggestions. The system's primary goals are to improve health outcomes, reduce the need for immediate physician consultations, and make personalized healthcare more accessible to users.

1.4.2. Purpose

As an important application of medical information, healthcare big data analysis has been extensively researched in the fields of intelligent consultation, disease diagnosis, intelligent question-answering doctors, and medical assistant decision support, and has made many achievements. In order to improve the comprehensiveness and pertinence of the medical examination, we use healthcare big data analysis combined with deep learning technology to provide patients with potential diseases which is usually neglected for lacking of professional knowledge, so that patients can do targeted medical examinations to prevent health condition from getting worse. Inspired by the existing recommendation methods, it proposes a novel deep-learning- based hybrid recommendation algorithm, which is called medical-history-based potential disease prediction algorithm.

The purpose is to harness the power of healthcare big data and deep learning technologies to provide an intelligent, user-friendly system for early disease prediction. This system aims to bridge the gap between medical knowledge and individual health awareness by analyzing a user’s medical history, symptoms, and lifestyle data. By predicting potential diseases that might otherwise go undetected, the system empowers patients to take preventive measures early on, before conditions worsen.

Chapter 2:

System Analysis

2.1 Limitations of Existing System :-

Data-Dependent Accuracy: Prediction accuracy is heavily reliant on the dataset's quality and diversity. Limited or biased data may lead to less reliable results in real-world applications.

High Memory Usage: The CNN model requires significant memory, especially with deep layers, which can impact performance and may need high-end hardware for smooth operation.

2.2 Requirement Analysis :-

The second chapter described the study of different papers and documents related to the proposed work. It specified the summary of each paper. The third chapter is Software requirement specification. The points included in this chapter are functional requirements, non-functional requirements, hardware and software requirements, external requirements, system requirements. This chapter also includes the software development lifecycle model which is to be used

2.3 Assumptions and Dependencies

User has basic knowledge of Neural Network.

User follows instruction as shown by the application.

Application should be able to display results based on current Symptoms'.

2.4 Functional Requirements

- Registration
- Login
- Enter Disease
- Enter Current Symptoms
- Pre-Processing
- Feature extraction
- CNN Training
- CNN Testing
- Predicated Results
- View Tablets, Diet Plan, Exercise and Doctor List
- Add History
- Logout

2.5 Non Functional Requirements

System should correctly execute process; display the result i.e. exact detection of disease. System should be able to suggest accurate pesticides.]Non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical non-functional requirements are reliability, scalability, and cost. Other terms for non-functional requirements are”constraints”,”quality attributes” and “quality of service requirements”.

1. Reliability

If any exceptions occur during the execution of the software it should be caught and there by prevent the system from crashing.

2. Scalability

The system should be developed in such a way that new modules and functionalities can be added, thereby facilitating system evolution.

3. Accuracy

System should correctly execute process; display the result i.e. exact detection of disease.

System should be able to suggest accurate pesticides.

1. Performance Requirements

The performance of the functions and every module must be well.

The overall performance of the software will enable the users to work efficiently.

Disease detection should be fast.

2. Software Quality Attributes

The software is user friendly while using it.

Image Quality in real time environment is clear (Noise free).

Disease should be accurately detected.

Diet plan, Exercise, Doctors List should be accurately suggested.

2.6 System Requirements

1. Software Requirements (Platform choice)

Windows 7 and above 3.5.3(Server Side)

1.Language – Python

2. Libraries and Package

(a)Numpy

(b)Pandas

(c)Opencv-python

(d)Cmake

(e)Dlib

(f)Face-recognition

(g)Mysql-connector-python

(h)Requests

(i)Tensorflow

(j)Keras

(k)Sklearn

It is recommended to use Anaconda Python 3.6 distribution and using a Django framework.

Client Side

Windows

Language-Python

IDE: Django framework

2. Hardware Requirements

Processor - Intel i3/i5/i7

Speed - 1.1 GHz

RAM - 2 GB(min)

Hard Disk - 40 GB

Floppy Drive - 1.44 MB

2.7 Literature Survey

1. History

Wenxing et al [1] : Medical-History-Based Potential Disease Prediction Algorithm -IEEE Access/2019 This paper proposed novel deep-learning-based hybrid recommendation algorithm, which predicts the patient's possible disease based on the patient's medical history and provides a reference to patients and doctors.It considers both, high-

-order relations as well as low order combination of disease among disease features and Improved comprehensiveness compared to previous system.

Dahiwade, D., Patle, G., Meshram,E. [2] : Designing Disease Prediction Model Using Machine Learning Approach -IEEE xplore/2019 Proposed general disease prediction, In which the living habits of person and checkup information consider for the accurate prediction.It also computes the risk associated with general disease. Low time consumption. Minimal cost possible. The accuracy of disease prediction is 84.5 percent.

2. Presently Available in System

The correct prediction of disease is the most challenging task. To overcome this problem data mining plays an important role to predict the disease. Medical science has large amount of data growth per year. Due to increase amount of data growth in medical and healthcare field the accurate analysis on medical data which has been benefits from early patient care. With the help of disease data, data mining finds hidden pattern information in the huge amount of medical data.General disease prediction based on symptoms of the patient. For the disease prediction, we use K-Nearest Neighbor (KNN) and Convolutional neural

network (CNN) machine learning algorithm for accurate prediction of disease. For disease prediction required disease symptoms dataset. In this general disease prediction the living habits of person and checkup information consider for the accurate prediction. The accuracy of general disease prediction by using CNN is 84.5memory requirement is also more in KNN than CNN. After general disease prediction, this system able to gives the risk associated with general disease which is lower risk of general disease or higher.

3. Presently Available DNN Algorithm

The network's architecture consists of 64 hidden neurons and 16 output neurons. The encoded information collected by the hidden layer's neuron is sent to the encryption layer, which then decodes it. The resulting output layer is composed of 16 hidden nodes. To control the output of the hidden layer, we need to combine the output with other compression techniques.]They are constantly in contact with infected person in hospital and also with health Hospital staffs are among the frontline workers who fight against contagious diseases. They are constantly in contact with infected person in hospital and also with health symptoms of different diseases is fed as input to system along with current symptoms of user and medical history of patient (when patient observed same type of symptoms before). Python based system used DNN algorithm to predict disease patient is suffering from. After predicting disease system classified disease into mild, moderate and severe conditions.

The network's architecture consists of 64 hidden neurons and 16 output neurons. The encoded information collected by the hidden layer's neuron is sent to the encryption layer, which then decodes it. The resulting output layer is composed of 16 hidden nodes.

Chapter 3 :-

Planning Phase

3.1 Feasibility Study

The feasibility study assesses the practicality of the "Smart Health Card Using Machine Learning" project by analyzing its technical, operational, economic, and scheduling aspects. This evaluation ensures that the project's objectives are achievable within available resources, budget, and timeline.

1. Technical Feasibility

The technical feasibility examines whether the necessary technology and resources are available to implement the project effectively.

- Technology Stack: The project relies on Python, Django, and essential machine learning libraries, including TensorFlow and Keras. Python's robust support for machine learning and Django's web framework provide a solid foundation for developing a reliable and scalable system.
- Algorithm Selection: Disease prediction will utilize the K-Nearest Neighbors (KNN) and Convolutional Neural Network (CNN) algorithms. KNN provides straightforward classification, while CNN offers detailed prediction based on complex patterns in data. However, CNN's high memory consumption can limit performance on devices with lower specifications.
- Hardware Requirements: The system requires a minimum of an Intel i3 processor, 2 GB of RAM, and 40 GB of storage space. While this configuration is sufficient for initial development, expanding the system for

larger datasets or real-time usage may necessitate more powerful hardware.

2. Operational Feasibility

Operational feasibility assesses the project's ability to meet user needs and fit within operational constraints.

- User Accessibility: The system is designed to be user-friendly, enabling users to input symptoms, view disease predictions, and receive guidance on diet, exercise, and relevant medical specialists. Proper design for clear instructions is essential to help users enter symptoms accurately.
- Healthcare System Integration: For future integration with healthcare facilities or electronic health records (EHRs), the system would need to comply with regulatory standards such as the General Data Protection Regulation (GDPR) or the Health Insurance Portability and Accountability Act (HIPAA) to ensure data privacy and security.
- System Maintenance: Maintenance would involve updating the medical dataset regularly to reflect new diseases and treatment options. Additional updates to the machine learning model will enhance accuracy and extend functionality.

3. Economic Feasibility

Economic feasibility considers the financial viability of the project by assessing both initial and potential future costs.

- Development Cost: According to the Constructive Cost Model (COCOMO) estimation, the development cost for this project is

approximately 12,665 INR, which covers essential resources for building and testing the system. This is a reasonable budget for a prototype.

- Future Costs: To scale the system for real-world applications, additional investments may be necessary. These may include costs for high-performance servers or cloud infrastructure, data security enhancements, and extensive datasets. While the initial prototype is financially feasible, larger deployments could incur higher operational costs.

4. Schedule Feasibility

Schedule feasibility assesses whether the project can be completed within the desired timeframe.

- Timeline: With defined project phases, including requirement analysis, design, implementation, and testing, the project is expected to take approximately 6.3 months. This timeline is achievable for a small team with a focused work plan and knowledge of Python, machine learning, and web development.
- Resource Availability: A dedicated team skilled in Python, Django, and machine learning can complete the project on time. Regular progress reviews will help ensure adherence to the planned schedule.

3.2 Analysis Models: SDLC Model to be applied

1. Software Model

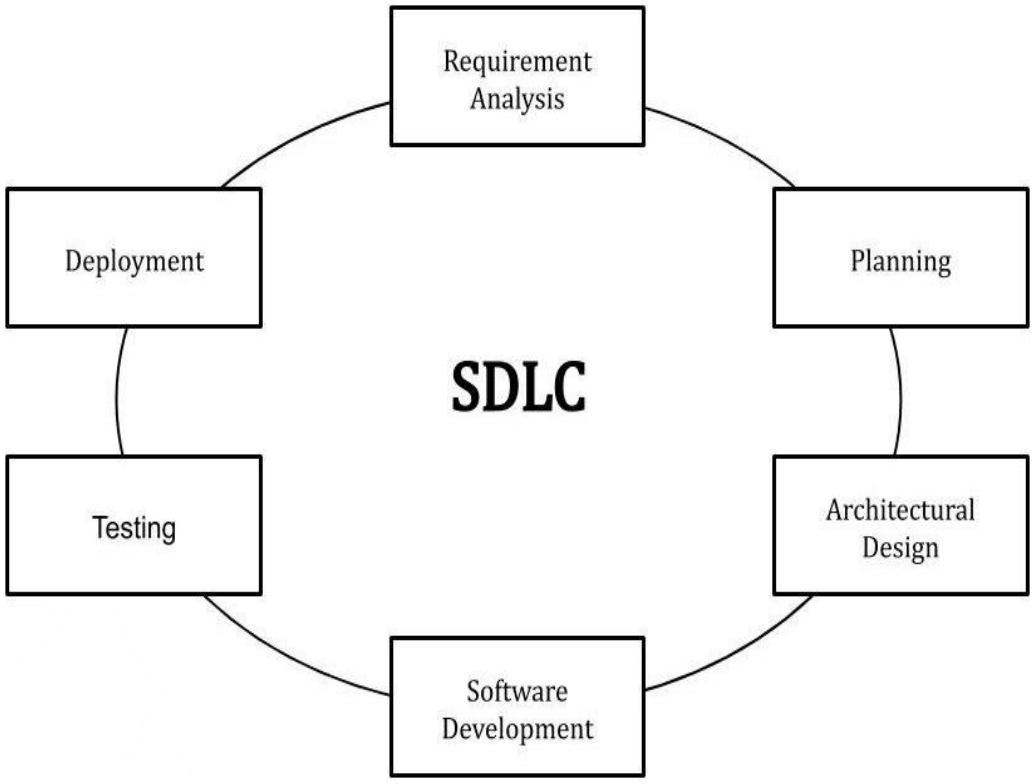
SDLC Software Development Life Cycle

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.
- ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC:



A typical Software Development Life Cycle consists of the following stages –

1. Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

2. Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

3. Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

4. Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code

5. Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

6. Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

Waterfall Model

The Waterfall Model shown in figure is a sequential design process, often used in Software development processes; where progress is seen as flowing steadily download through the phase of conception, Initiation, Analysis, Design, Construction, Testing, Production/Implementation, and Maintenance. There are 5 Phase of the waterfall model: image tag

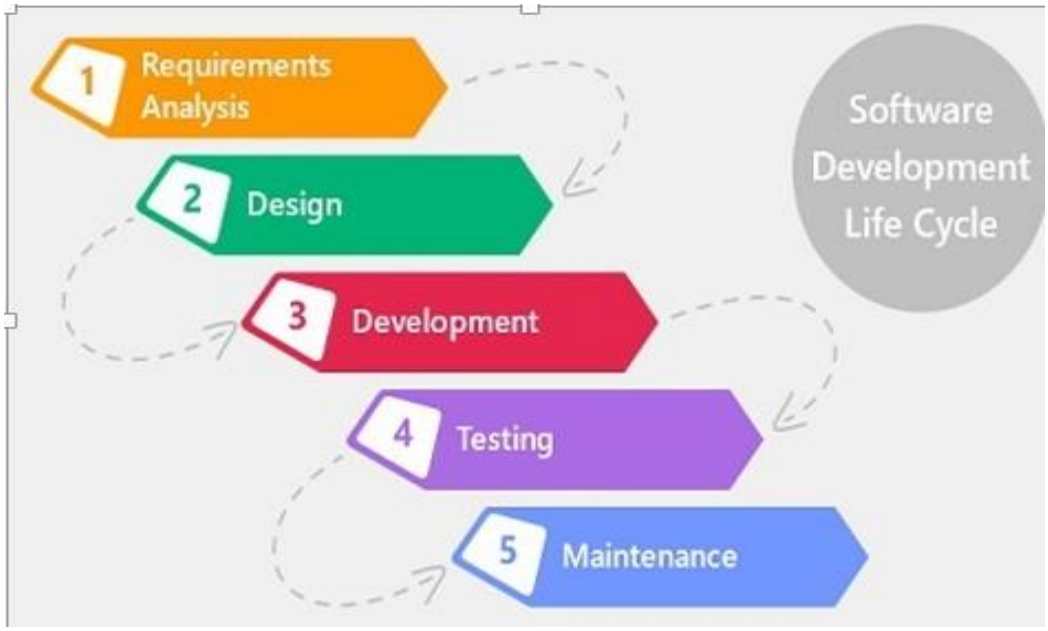


Fig. Waterfall Model

- **Requirement Gathering and analysis**

All the functional and non- functional requirements of the project were identified Interaction with the users and all other stakeholders of the project were conducted to identify all the requirements starting from important features like maintaining audit trail, etc. to the very basic features like the look and the feel of user interface.

- **System Design**

The initial step was project design. The project was designed mainly in two parts: In first part, system captures an image of plant and processes it to detect diseases.

- **Implementation**

In this stage system is developed according to module wise.

- **Verification**

This stage all developed software are installed and they are tested with different way against system requirements.

- **Maintenance**

According to software's new version and there use them need to update.

3.3 Flow Of The System :-

User Registration and Login:

- Users (patients or doctors) begin by creating an account on the platform. They provide essential information (e.g., name, age, medical ID).

- After successful registration, users log in to access the system's features. Authentication ensures secure access.

Symptom Input and Data Collection:

- Once logged in, the user inputs current symptoms and relevant health details into the system.
- The system may also allow users to upload past medical history, which is stored in a centralized database for future reference.

Data Preprocessing:

- The system pre-processes the input data, ensuring that it is clean, formatted, and standardized for use in machine learning models.
- Preprocessing may include handling missing values, normalizing symptom data, and encoding text data into machine-readable formats.

Disease Prediction Using Machine Learning:

- The pre-processed symptom data is fed into machine learning models (e.g., CNN or KNN) to predict potential diseases based on the symptoms.
- The model processes the data and identifies diseases that are statistically likely, assigning a probability score to each potential disease.

Risk Assessment and Result Generation:

- The system evaluates the risk associated with the predicted disease(s), classifying the risk as high, medium, or low based on the model's output.
- The system generates a summary of results, including the probable diseases, risk levels, and the probability of each prediction.

Personalized Recommendations:

- Based on the predicted disease and associated risk, the system provides personalized recommendations for the user. These include:
    - Suggested medications
    - Diet and exercise plans
    - A list of relevant healthcare providers or specialists (if available)

Display and Smart Health Card Generation:

- The system displays the disease prediction results and recommendations on the user's dashboard.
- A Smart Health Card, containing the user's unique medical ID and health data, is generated and made available for download or sharing. This digital card allows the user to keep an updated health record.

Medical History Logging and Updates:

- If users receive a diagnosis or update their health status, they can add this information to their medical history, allowing the system to improve future predictions.
- Doctors, with authorized access, can also add relevant medical notes to a user's history for a more comprehensive record.

Logout and Data Security:

- Once done, the user logs out of the system. The system ensures that all sensitive health data is securely stored and accessible only to authorized users.

Chapter 4 :-

Testing

4.1 Software testing

Software testing is an important process in the software development lifecycle . It involves verifying and validating that a software application is free of bugs, meets the technical requirements set by its design and development , and satisfies user requirements efficiently and effectively.

White Box Testing: -

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

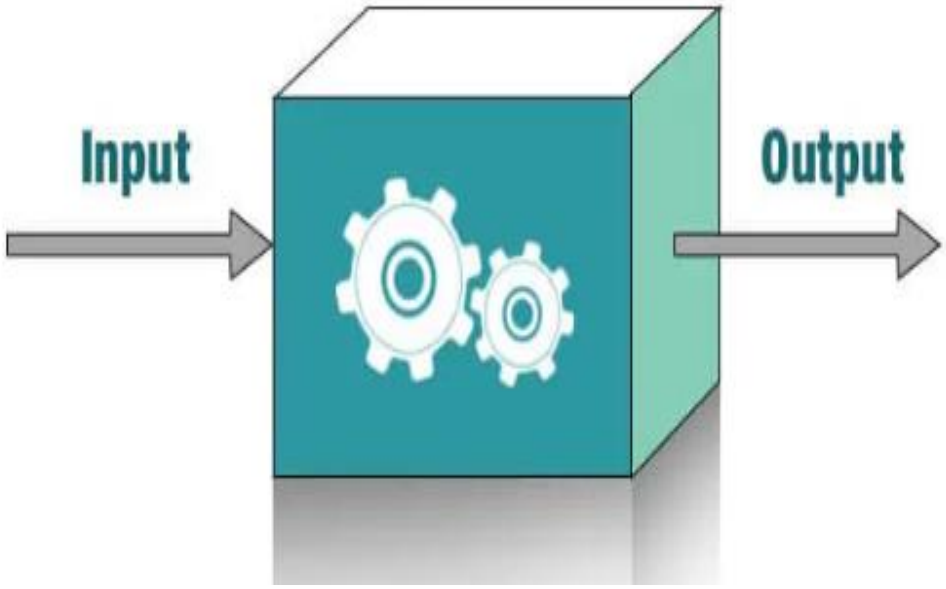
In this technique, the close examination of the logical parts through the software are tested by cases that exercises species sets of condition or loops, all logical parts of the software checked once error that can be corrected using this technique are typographical errors, logical expressions which should be executed once may be getting executed more than once and error resulting by using wrong controls and loops.

When the box testing tests all the independent part within a module a logical decision on their true and the false side are exercised, all loops and bounds

within their operational bounds were exercised and internal data structure to ensure their validity were exercised once.

White Box Testing Techniques:

- **Statement Coverage** - This technique is aimed at exercising all programming statements with minimal tests.
- **Branch Coverage** - This technique is running a series of tests to ensure that all branches are tested at least once.
- **Path Coverage** - This technique corresponds to testing all possible paths which means that each statement and branch is covered.



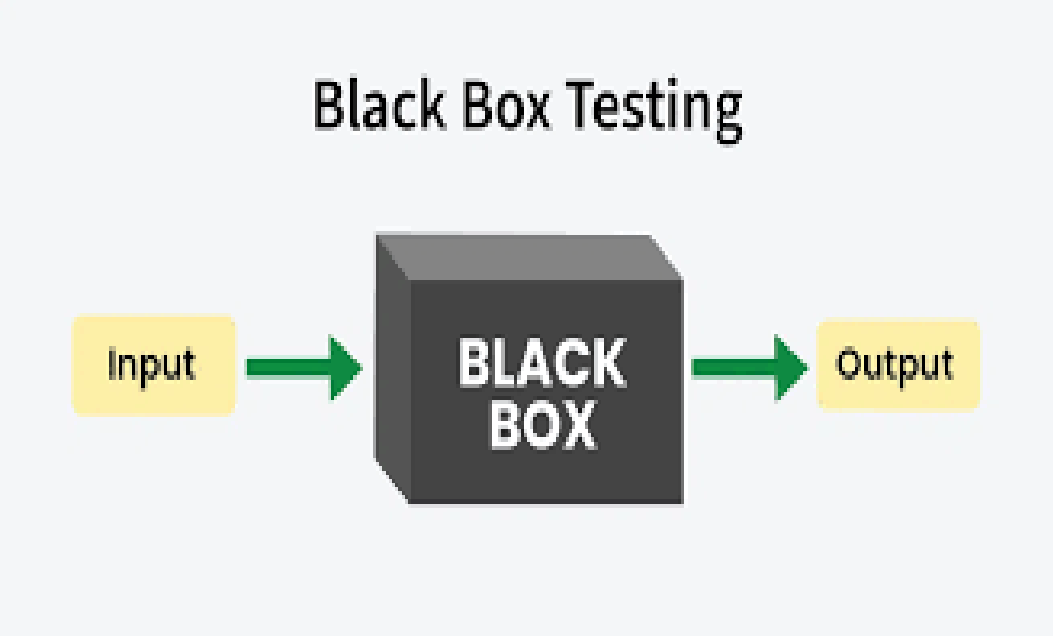
White Box Testing

Black Box Testing: -

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed.

The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.



Generic steps of black box testing

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.

- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested

Alpha Testing: -

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance testing. This is referred to as alpha testing only because it is done early on, near the end of the development of the software. Alpha testing is commonly performed by homestead software engineers or quality assurance staff. It is the last testing stage before the software is released into the real world.

Objective of Alpha Testing:

1. The objective of alpha testing is to refine the software product by finding the bugs that were not discovered during the previous tests.

2. The objective of alpha testing is to refine the software product by fixing the bugs that were not discovered during the previous tests.

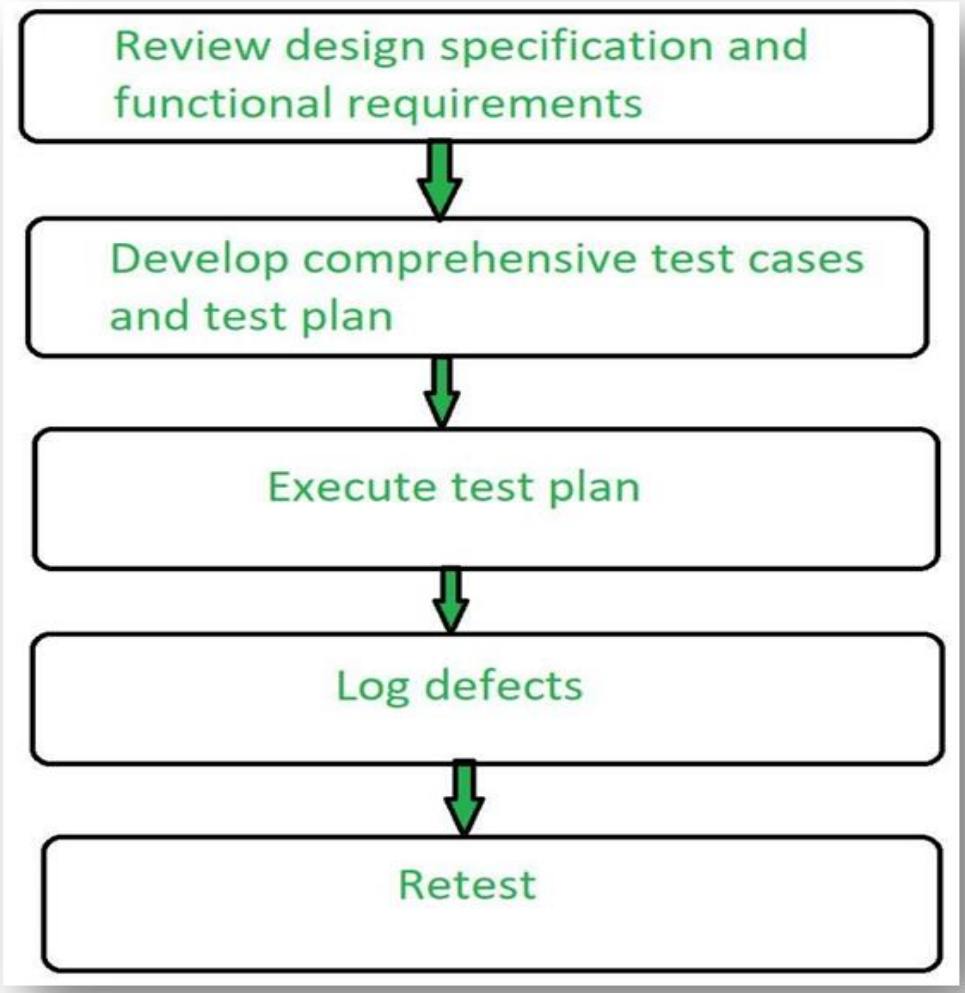
3. The objective of alpha testing is to involve customers deep into the process of development.

4. The objective of alpha testing is to give better insight into the software's reliability at the early stages of development.

Alpha Testing Process:

1. Review the design specification and functional requirements.
2. Develop comprehensive test cases and test plans.
3. Execute test plan

4. Log defects



Phases of Alpha Testing:

There are two phases in alpha testing:

1st Phase:

The first phase of testing is done by in-house developers or software engineers. They either use hardware-aided debuggers or debugger software. The aim is to catch bugs quickly. Usually while alpha testing, a tester comes across to lots of bugs, crashes, missing features, and docs.

2nd Phase:

The second phase of alpha testing is done by software quality assurance staff for additional testing in an environment. It includes a black box as well as white box testing.

Beta Testing: -

Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of **User Acceptance Testing**.

A Beta version of the software, whose feedback is needed, is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing helps in minimization of product failure risks and it provides increased quality of the product through customer validation. It is the last test before shipping a product to the customers.

Characteristics of Beta Testing:

1. Beta Testing is performed by clients or users who are not employees of the company.
2. Reliability, security, and robustness are checked during beta testing.
3. Beta Testing commonly uses black-box testing.
4. Beta testing is carried out in the user's location.
5. Beta testing doesn't require a lab or testing environment.

Types of Beta Testing:

There are different types of beta testing:

1. **Traditional Beta testing:** Product is distributed to the target market and related data is gathered in all aspects. This data can be used for Product improvement.
2. **Public Beta Testing:** Product is released publicly to the world through online channels and data can be collected from anyone. Based on feedback, product improvements can be done. For example, Microsoft conducted the largest of all Beta Tests for its operating system Windows 8 before officially releasing it.
3. **Technical Beta Testing:** Product is released to a group of employees of an organization and collects feedback/data from the employees of the organization.
4. **Focused Beta Testing:** Software product is released to the market for collecting feedback on specific features of the program. For example, important functionality of the software.
5. **Post-release Beta Testing:** Software product is released to the market and data is collected to make improvements for the future release of the product.

4.1.1. Unit Testing :-

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not.

It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested.

Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer.

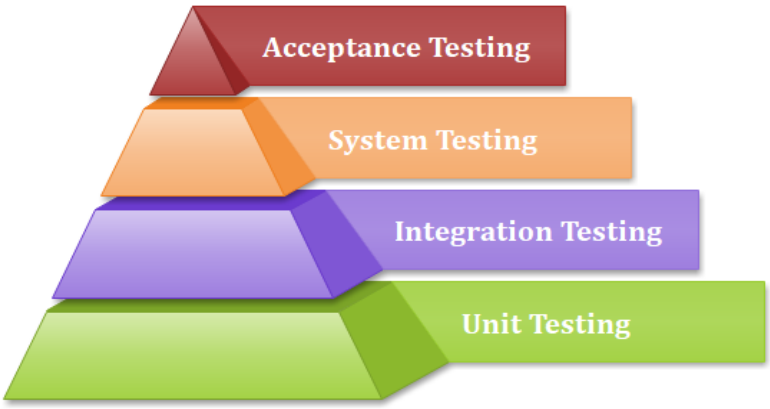
In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

Objective of Unit Testing:

The objective of Unit Testing is:

a. To isolate a section of code.
b. To verify the correctness of the code.
c. To test every function and procedure.

d. To fix bugs early in the development cycle and to save costs.
e. To help the developers to understand the code base and enable them to make changes quickly.
f. To help with code reuse.



Types of Unit Testing:

There are 2 types of Unit Testing: **Manual**, and **Automated**.

4.2.2. Integration Testing :-

Integration testing is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed. Integration testing can be done by picking module by module.

Integration test approaches – There are four types of integration testing approaches. Those approaches are the following:

1. Big-Bang Integration Testing –

It is the simplest integration testing approach, where all the modules are combined and the functionality is verified after the completion of individual module testing. In simple words, all the modules of the system are simply put together and tested. This approach is practicable only for very small systems. If an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during big bang integration testing is very expensive to fix.

2. Bottom-Up Integration Testing –

In bottom-up testing, each module at lower levels is tested with higher modules until all modules are tested. The primary purpose of this integration testing is that each subsystem tests the interfaces among various modules making up the

subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower level modules.

3. Top-Down Integration Testing –

Top-down integration testing technique is used in order to simulate the behavior of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First, high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

4. Mixed Integration Testing –

A mixed integration testing is also called sandwiched integration testing. A mixed integration testing follows a combination of top down and bottom-up testing approaches. In top-down approach, testing can start only after the top-level module have been coded and unit tested. In bottom-up approach, testing can start only after the bottom level modules are ready. This sandwich or mixed approach overcomes this shortcoming of the top-down and bottom-up approaches. It is also called the hybrid integration testing. also, stubs and drivers are used in mixed integration testing.

Validation Testing: -

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfils its intended use when deployed on appropriate environment.

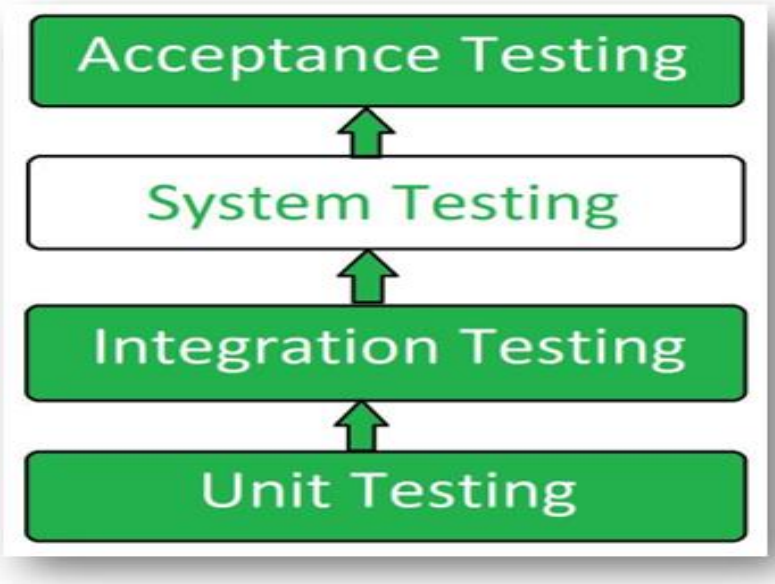
It answers to the question, are we building the right product?

Validation Testing - Workflow:

Validation testing can be best demonstrated using V-Model. The Software/product under test is evaluated during this type of testing.

4.3.3. System Testing :-

System Testing is a type of System Testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the (SRS) Software Requirements Specification.



System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. System Testing is a black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

System Testing Process:

System Testing is performed in the following steps:

- Test Environment Setup: Create testing environment for the better quality testing.

- Create Test Case: Generate test case for the testing process.
- Create Test Data: Generate the data that is to be tested.
- Execute Test Case: After the generation of the test case and the test data, test cases are executed.
- Defect Reporting: Defects in the system are detected.
- Regression Testing: It is carried out to test the side effects of the testing process.
- Log Defects: Defects are fixed in this step.
- Retest: If the test is not successful then again test is performed.
-

Types of System Testing:

- **Performance Testing:** Performance Testing is a type of software testing that is carried out to test the speed, scalability, stability and reliability of the software product or application.

- **Load Testing:** Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load.
- **Stress Testing:** Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads.
- **Scalability Testing:** Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

The system analysis consists of DFD diagram and UML diagram. It includes diagrams like system architecture, data flow diagram, use case diagram, activity diagram, and class diagram.

These diagrams help in understanding the functioning of the system.

Data flow Diagram

A data-flow diagram (DFD) is a way of representing a flow of a data of process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself.

1. DFD diagram level 0

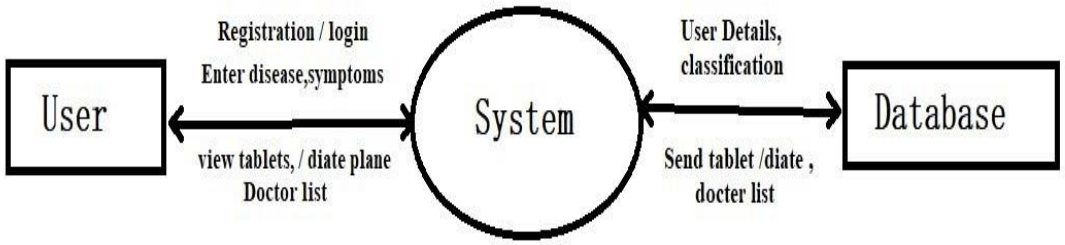


Fig. DFD diagram level 0

The DFD Diagram level 0 refer fig represent user register, login, and enter symptoms to detect disease after user get user details about disease , get tablet details , get diet plan

2. DFD diagram level 1

The DFD Diagram level 1 refer fig represent disease datastore, disease prediction, add disease, search doctor, admin.

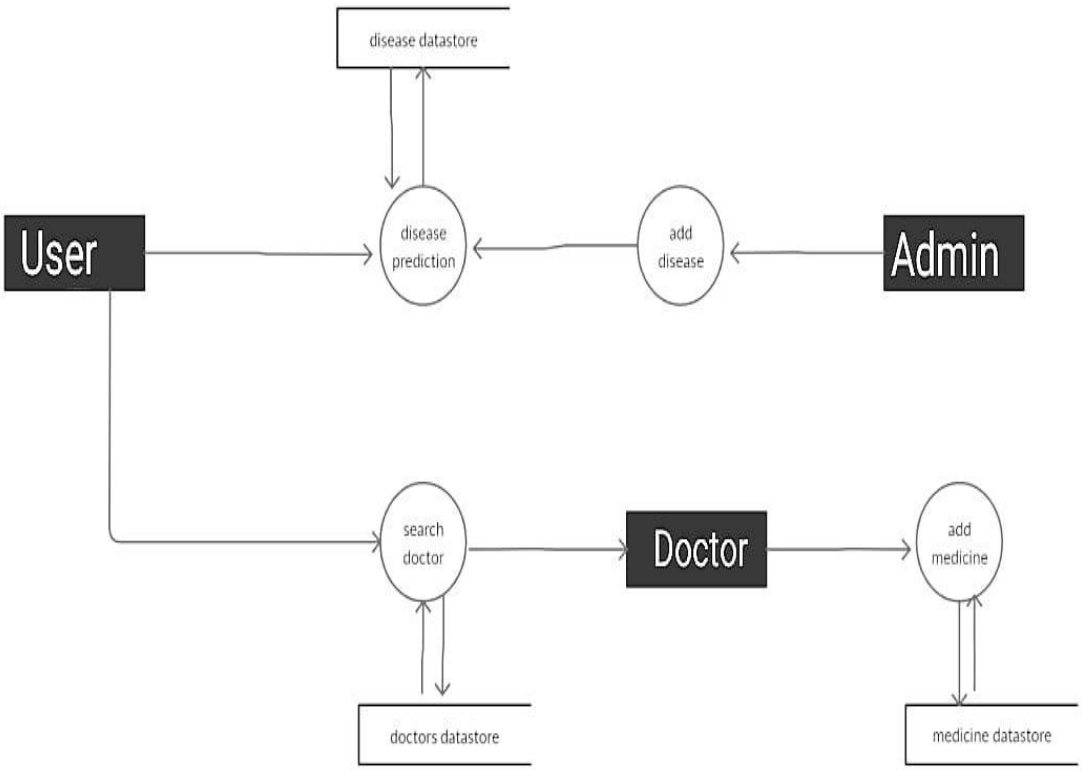


Fig. DFD diagram level 1

3. DFD diagram level 2

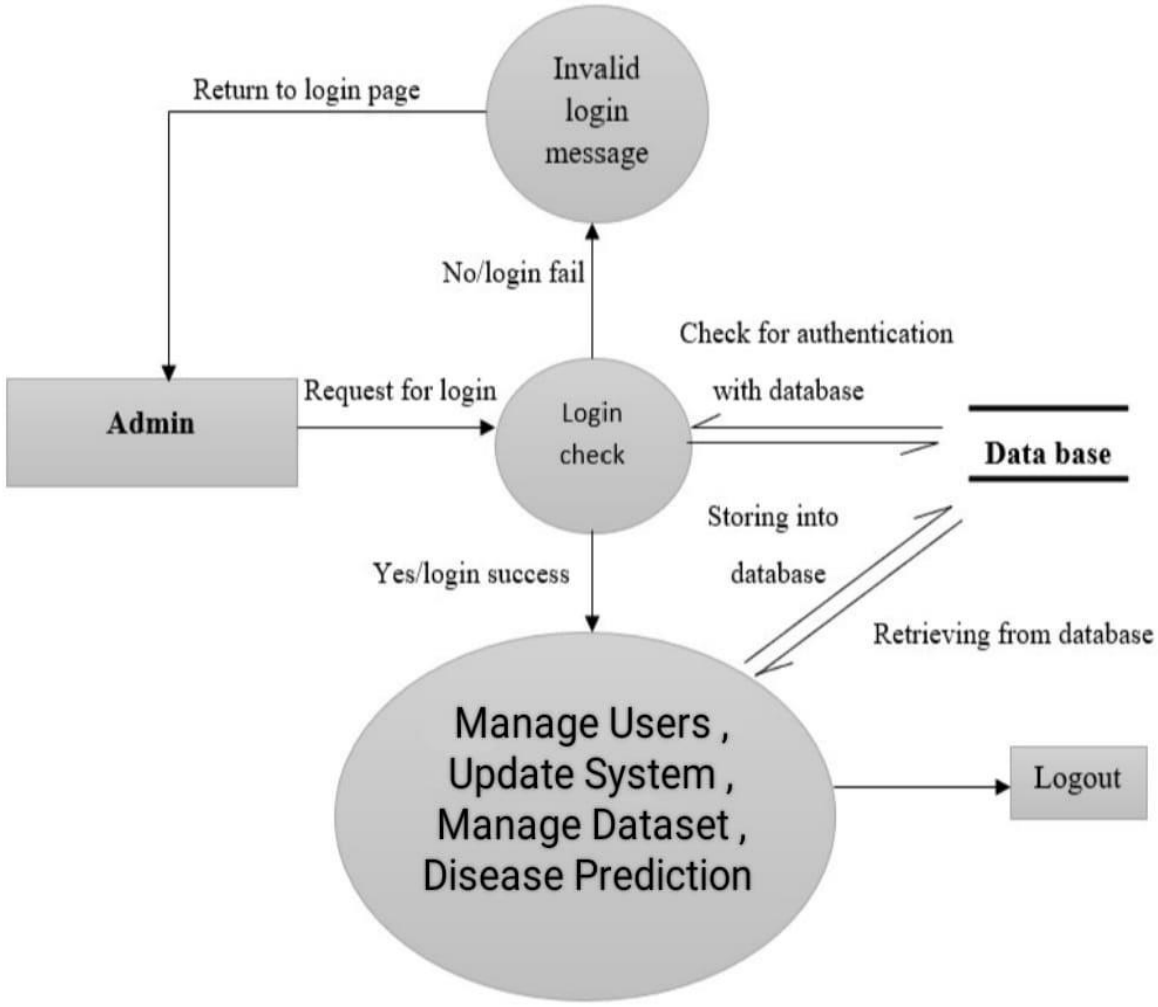


Fig. DFD diagram level 2

The DFD Diagram level 2 refer fig Admin Work on system as manage users, update system, manage dataset, add data about new disease.

UML Diagram

UML diagram is a diagram based on the UML (Unified Modelling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Class Diagram

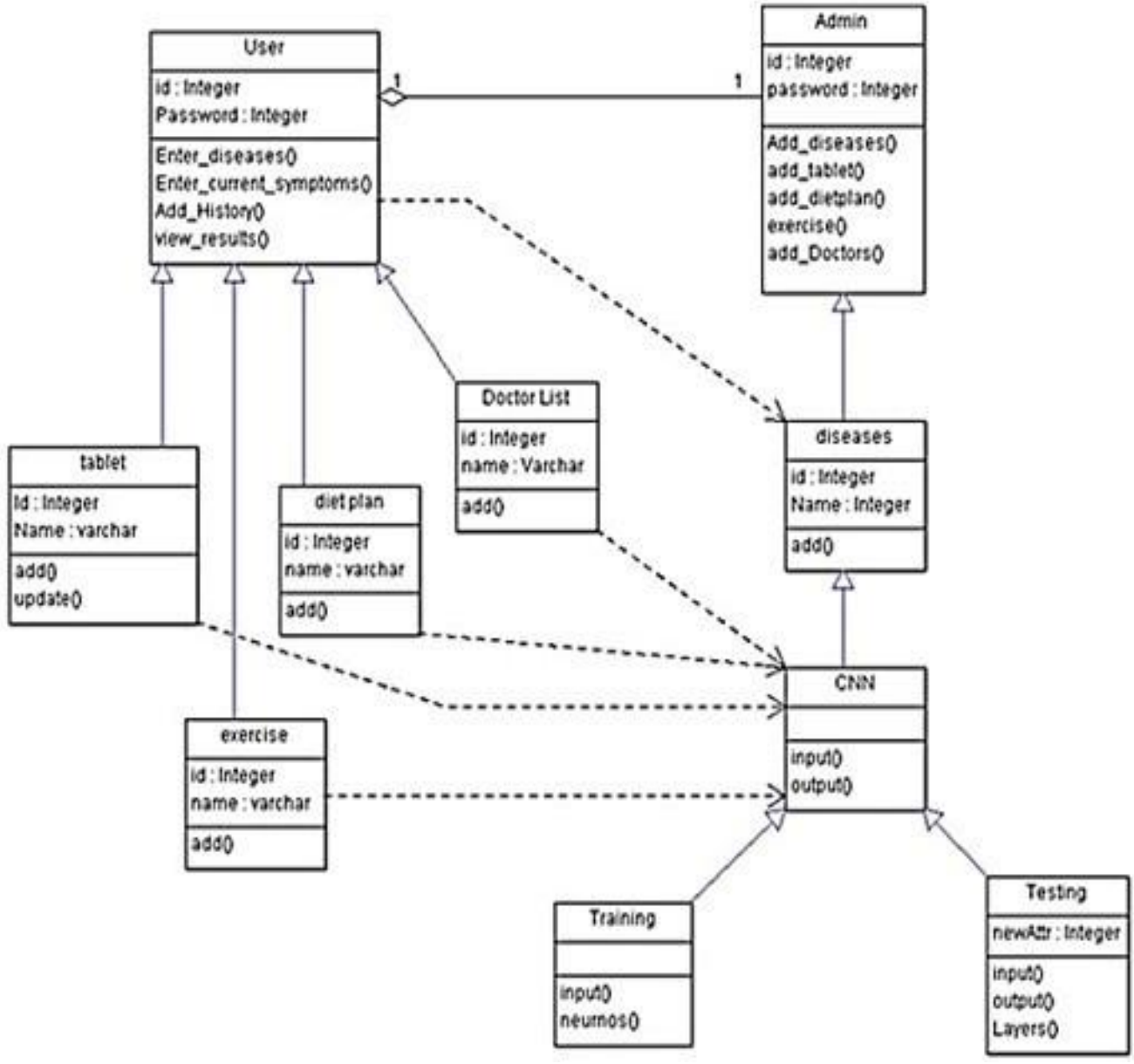


Fig Class Diagram

In class diagram it include the class that contain their attributes and their operations. In Smart Health prediction it's class diagram contain class like User, Admin, CNN, Disease, tablet, diet plan, exercise, testing, training. In User class contain attributes are "Id" and "Password" also it's have operation Add History, View Result, Enter disease, Enter Current symptoms. In Admin Class Contain

attributes like “Id” and “password” and their is operation Add Disease, add tablet, Add Doctor ;In Show

Fig 4.4 class diagram all class are connected each other and perform their work.

Use Case Diagram

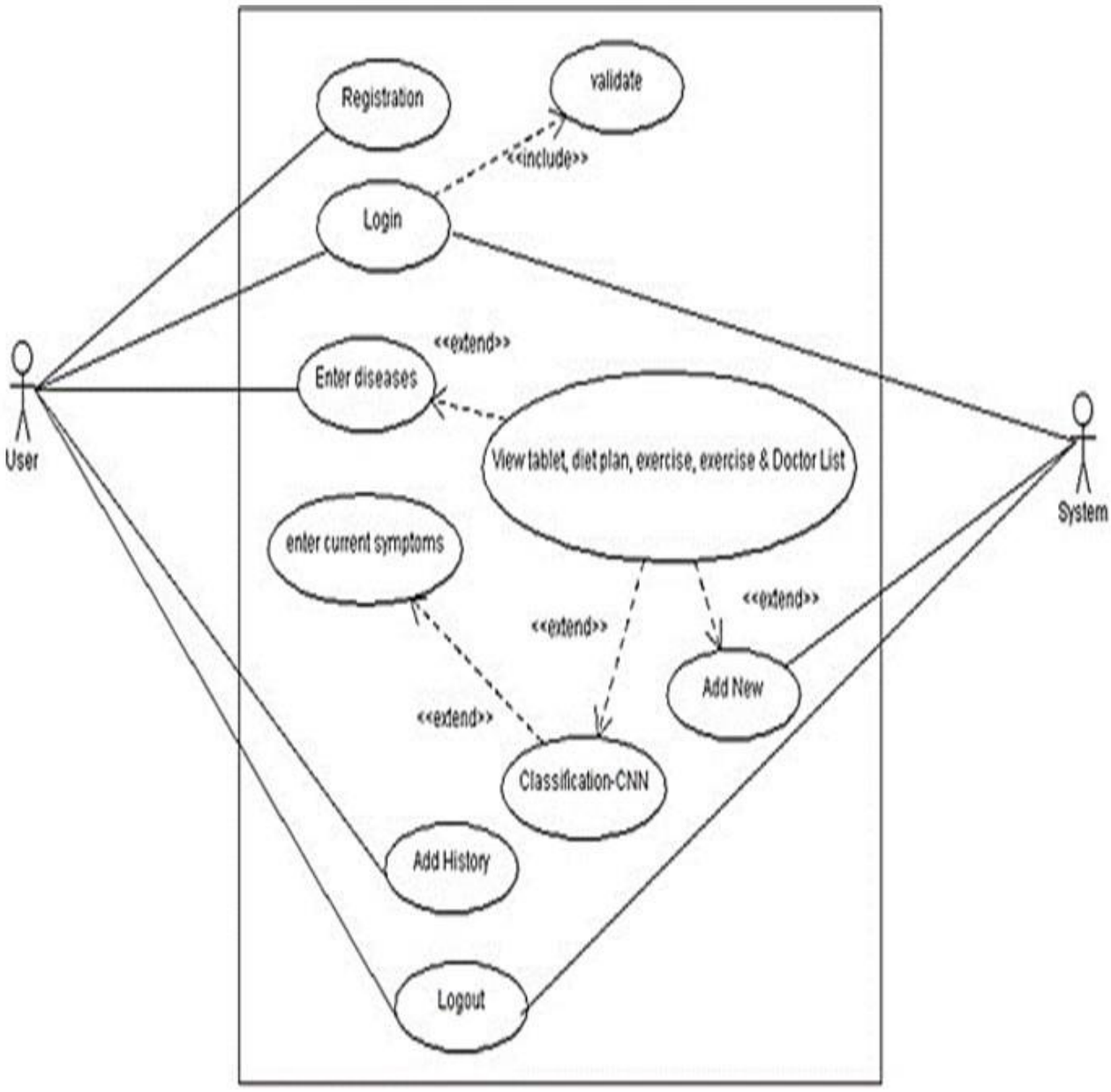


Fig Use Case Diagram

The use case diagram as shown in figure 4.5 summarizes the relationships between use cases and actors It’s Show general processes of Smart Health

prediction system.. The two main actors are user and system. User is as show in fig 4.5 first step is registration and after he can login , login is verify After login User Enter His Symptoms to detect his

Disease after user view Data as like tablet, diet plan, exercise and doctor list. User can add his past medical history and after all process user can Logout. Second Actor System can work as Admin. Admin Can Login and login validate. Admin Can Add disease, doctor list, diet plan. Admin main Work is Update Data of system after admin will logout.

Sequence Diagram

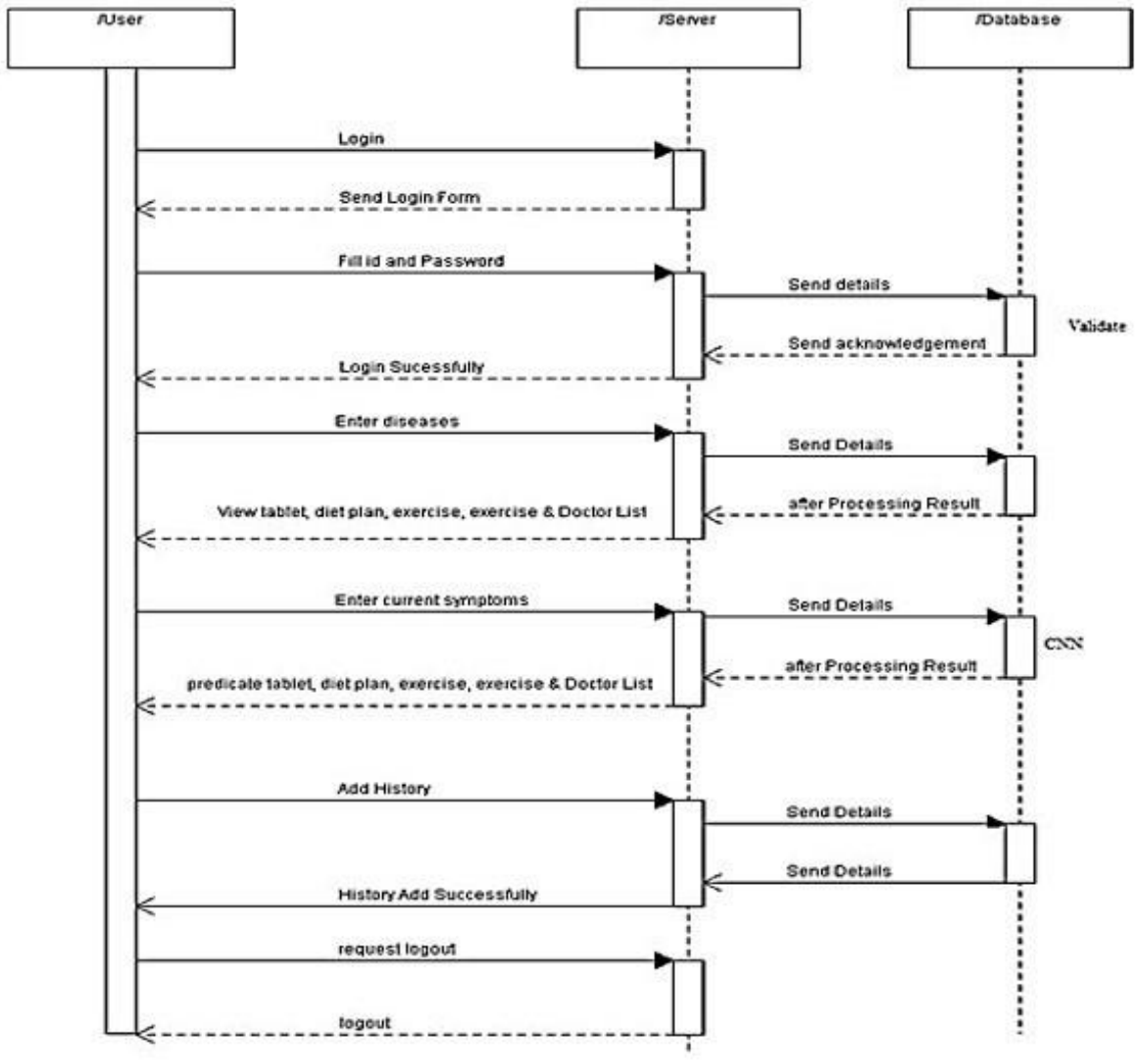


Fig Sequence Diagram

A sequence diagram shown in fig 4.6 is a type of interaction diagram because it describes how and in what order a group of objects works together. it show working of admin , server, database. All are Connected to each other as Show In fig 4.6 and work together. in sequence diagram User uploads an image and then according to his request a result is displayed.

Activity Diagram

Activity diagram fig is basically a flowchart to represent the flow from one activity to another activity. Activity show as fig.4.7 are like Login, enter Disease, enter current symptoms, add history and after completion Can Logout. The activity can be described as an operation of the system.

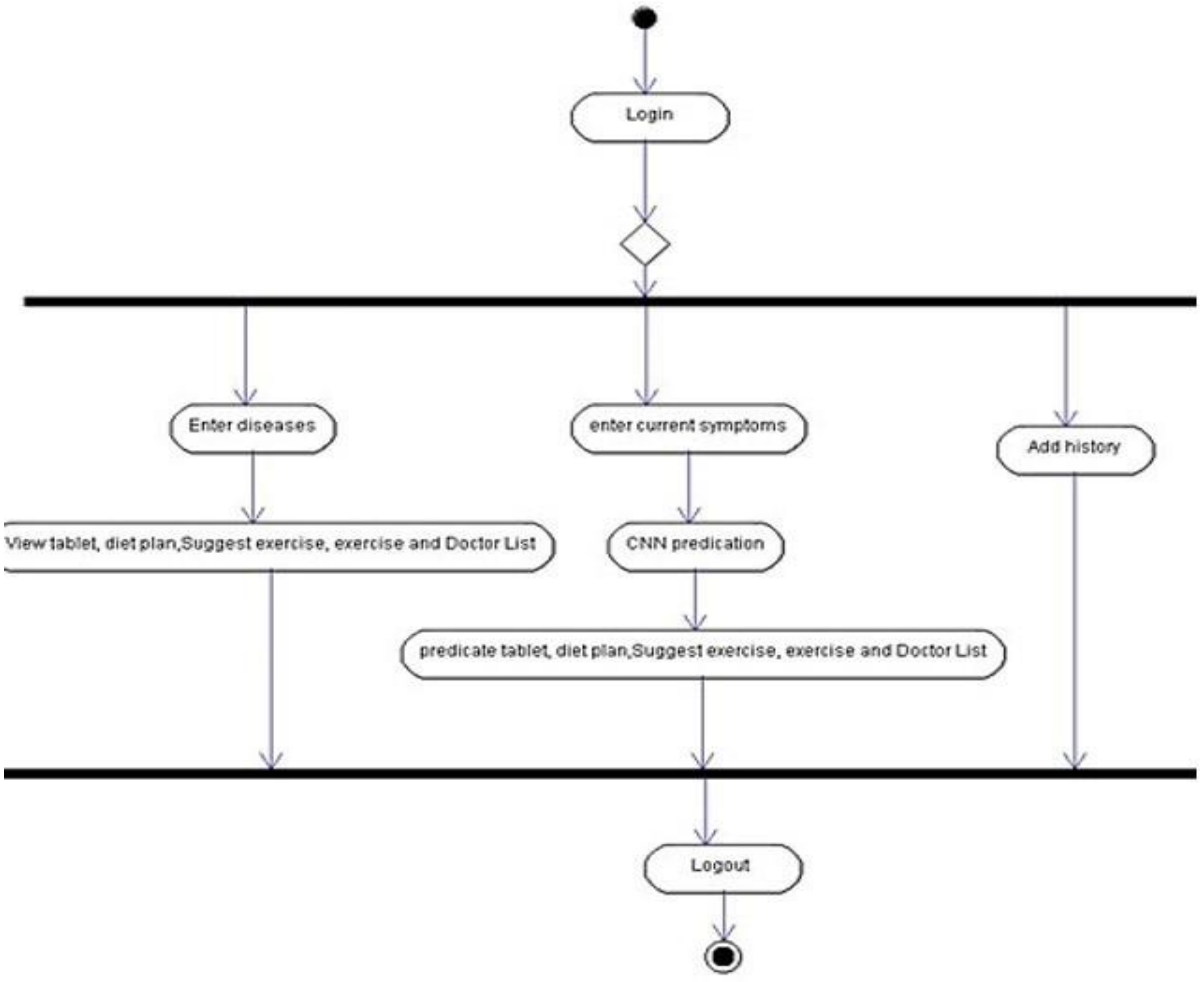
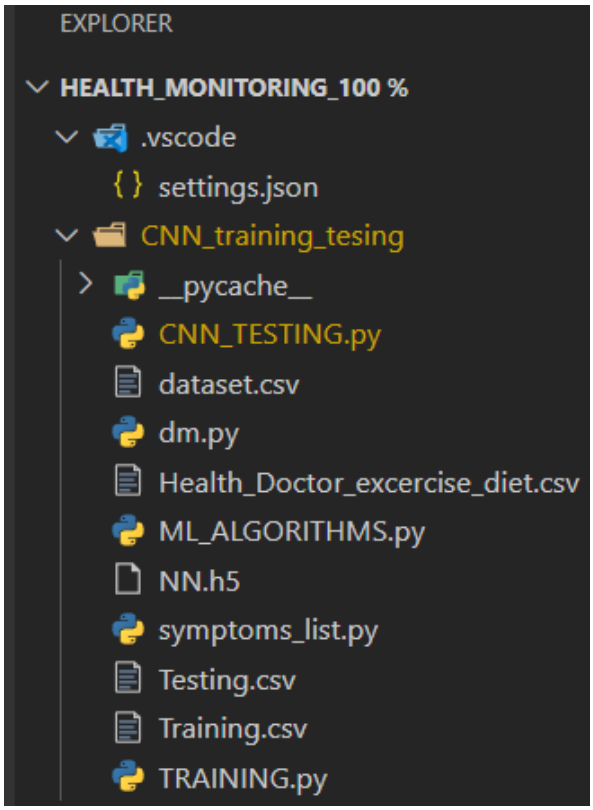


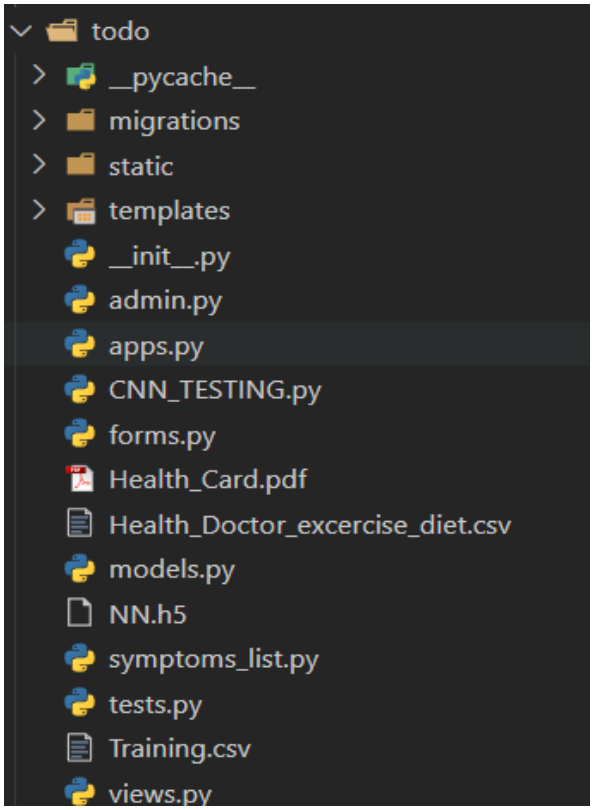
Fig Activity diagram

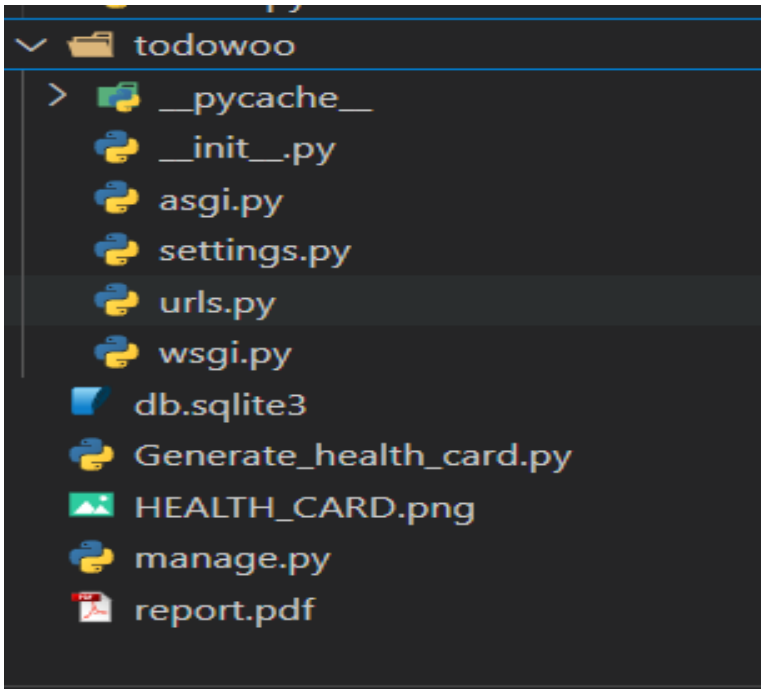
Chapter 5 :-
Conclusion and References

5.1 Appendices

Project Structure :-







CODE :-

Terminal

```
{
  "terminal.integrated.cwd":
"C:\\Users\\Meghna\\OneDrive\\Documents\\Desktop\\HEALTH_MONITORING
_100 %"
}
```

- CNN testing

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler
from keras.models import load_model
from symptoms_list import *
def get_result(symptoms_inserted):
  sym =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0]


  for k in range(0,len(sym)):
    for z in symptoms_inserted:
      if(z==symptoms[k]):
```

```
            sym[k-1]=1
##    print(sym)
    data=pd.read_csv("Training.csv")
    genre_list = data.iloc[:, -1]
    encoder = LabelEncoder()
    y = encoder.fit_transform(genre_list)
    model = load_model('NN.h5')
    ##sym =
[1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0]
    symptoms_to_algorithm = np.array([sym])
    predictions = model.predict(symptoms_to_algorithm)
    acc =  "%.2f" % round((max(predictions[0]) * 100),2)


    result_encoded = np.argmax(predictions[0])
    result = encoder.inverse_transform([result_encoded])
    data=pd.read_csv("Health_Doctor_excercise_diet.csv")
    doctor = data['Doctor']
    exercise = data['exercise']
    diet = data['diet']
```

```
    Diseases = data['disease_name']
    ##print(Diseases)
    for i in range (0,len(Diseases)):
        if(Diseases[i]==result[0]):
            break
    doctor_info = doctor[i]
    exercise_info = exercise[i]
    diet_info = diet[i]
    return result[0],acc,doctor_info,exercise_info,diet_info
```

- todo

```
 from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion
class Migration(migrations.Migration):
    initial = True
    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]
    operations = [
        migrations.CreateModel(
            name='User_info',
```

```
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('first_name', models.CharField(max_length=40)),
                ('last_name', models.CharField(max_length=40)),
                ('dob', models.DateField()),
                ('gender', models.CharField(max_length=40)),
                ('mobile', models.CharField(max_length=40)),
                ('Email_ID', models.CharField(max_length=40)),
                ('Address', models.CharField(max_length=40)),
                ('hin', models.CharField(max_length=40)),
                ('role', models.CharField(max_length=40)),
                ('user',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
related_name='user_name', to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='Todo',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
```

```
            ('title', models.CharField(max_length=100)),
            ('memo', models.TextField(blank=True)),
            ('created', models.DateTimeField(auto_now_add=True)),
            ('datecompleted', models.DateTimeField(blank=True, null=True)),
            ('important', models.BooleanField(default=False)),
            ('user',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
        ],
    ),
]
```

- Templates

```
{% load static %}
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
  <link rel="stylesheet" type="text/css" href="{% static 'todo/style.css' %}">
  <!-- Bootstrap CSS -->
```

```
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css
" integrity="sha384-
TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
   <link rel="icon" type="image/png"  href="{% static 'todo/logo.png' %}">
   <title>HEALTH CARE</title>
</head>
<body>
<nav class="navbar navbar-expand-md navbar-light bg-warning">
<div class="container">
   <a class="navbar-brand" href="{% url 'home' %}">
      <img src="{% static 'todo/logo.png' %}" width="30" height="30" class="d-
inline-block align-top">
      <span>HEALTH CARE</span>
   </a>
   <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
   </button>
   <div class="collapse navbar-collapse" id="navbarNav">
   {% if user.is_authenticated %}
```

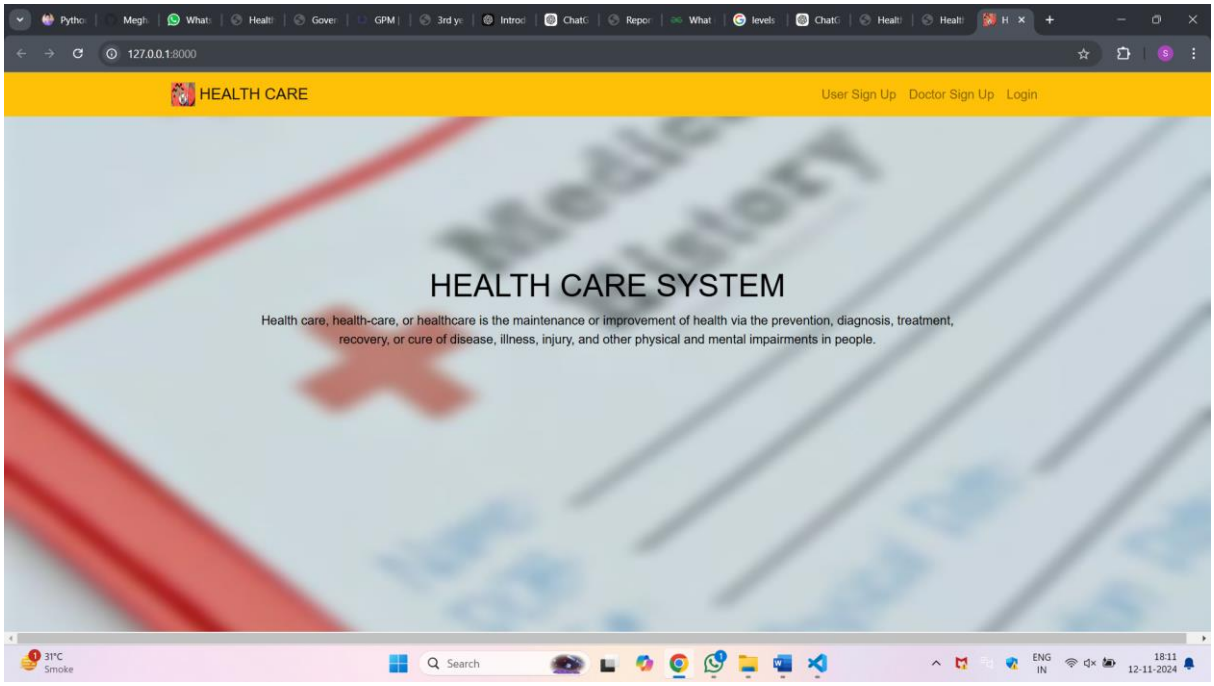
```
    <ul class="navbar-nav mr-auto">
        <li class="nav-item {{ current }}">
          <a class="nav-link" href="{% url 'your_profile' %}">Your Profile</a>
        </li>
        <li class="nav-item {{ current }}">
          <a class="nav-link" href="{% url 'currenttodos' %}">Current
Disease</a>
        </li>
        <li class="nav-item {{ completed }}">
          <a class="nav-link" href="{% url 'completedtodos' %}">Medical
History</a>
        </li>
        <li class="nav-item {{ create }}">
          <a class="nav-link" href="{% url 'createtodo' %}">Add Health
Issue</a>
        </li>
        <li class="nav-item {{ create }}">
          <a class="nav-link" href="{% url 'predict_disease' %}">Predict
Disease</a>
        </li>
    </ul>
  {% endif %}
```

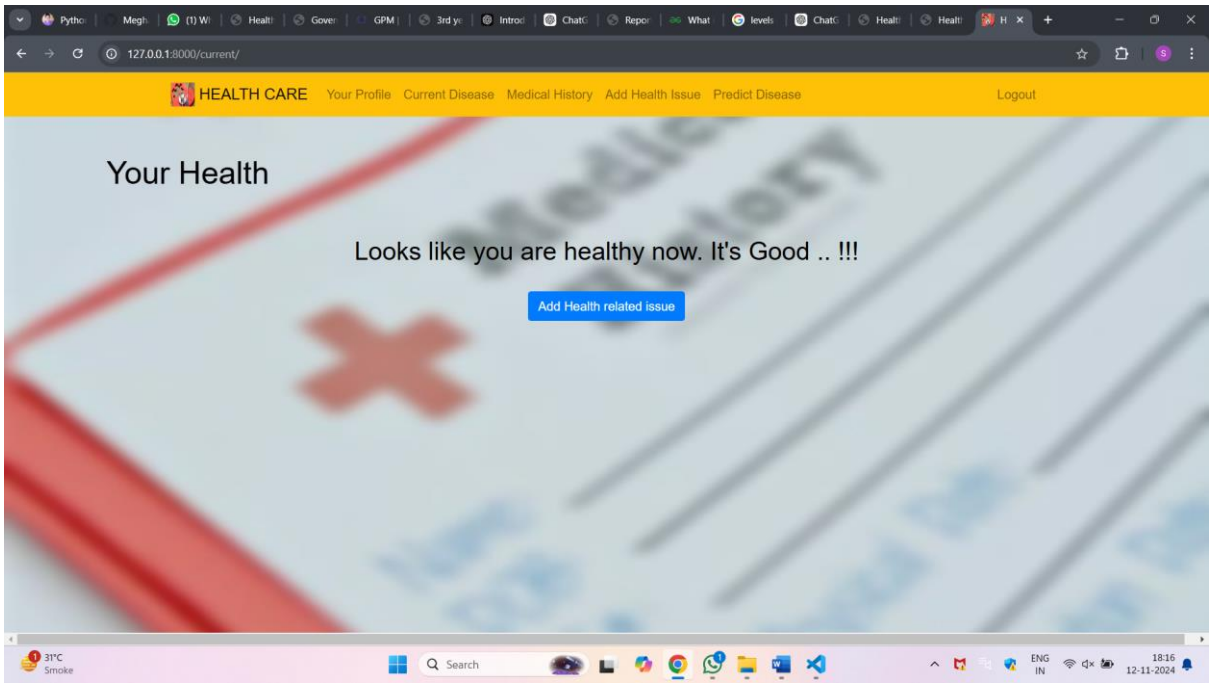
```
  <ul class="navbar-nav ml-auto">
      {% if user.is_authenticated %}
        <li class="nav-item">
          <a href="#" onclick="$('#signOutBtn').click()" class="nav-
link">Logout</a>
          <form style='display: none;' method="POST" action="{% url
'logoutuser' %}">
            {% csrf_token %}
            <button id="signOutBtn" type="submit">Logout</button>
          </form>
        </li>
      {% else %}
        <li class="nav-item">
          <a class="nav-link" href="{% url 'signupuser' %}">User Sign Up</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="{% url 'doctor_signupuser' %}">Doctor Sign
Up</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="{% url 'loginuser' %}">Login</a>
        </li>
```

```
        {% endif %}
      </ul>
    </div>
</div>
</nav>
<div>
<!-- <div class="container" style="background-image: url({% static 'todo/BG.jpg' %}) ">-->
{% block content %}{% endblock %}
</div>
<!-- Optional JavaScript -->
<!-- jQuery first, then Popper.js, then Bootstrap JS -->
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
```

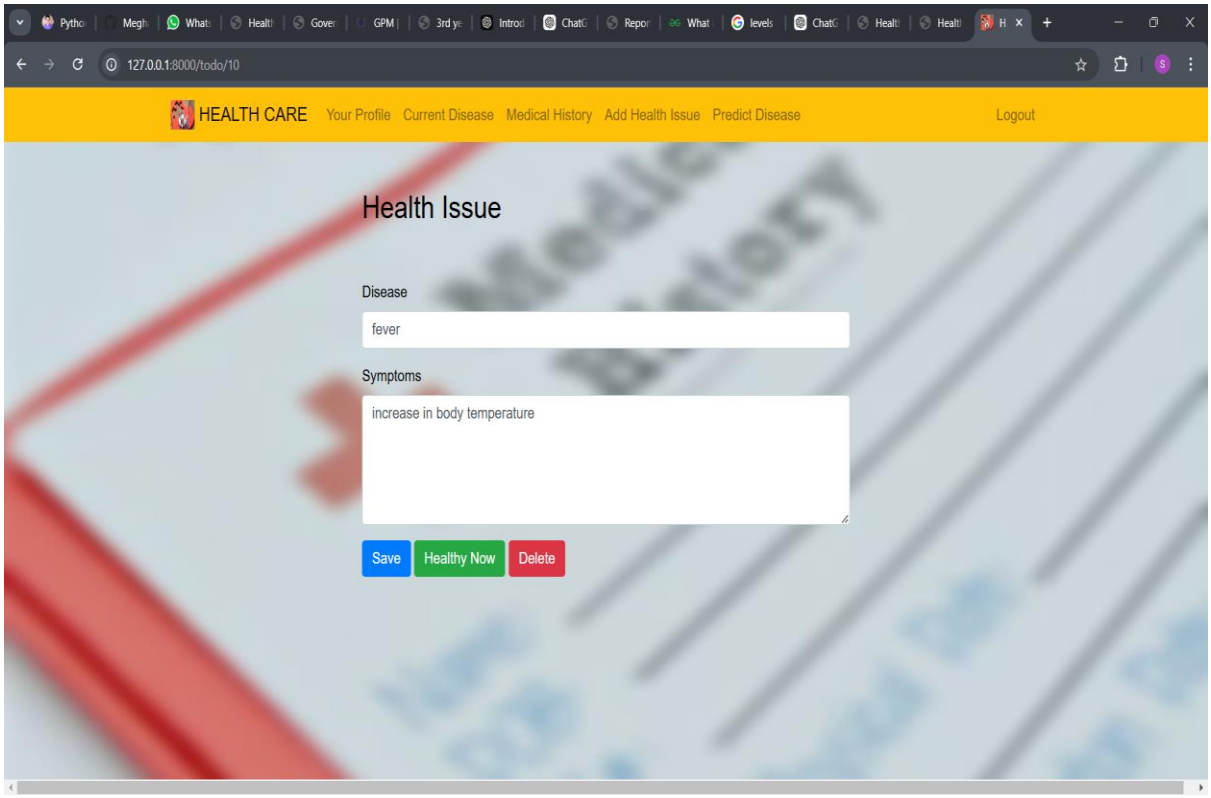
```
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6
" crossorigin="anonymous"></script>
</body>
</html>
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys
def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'todowoo.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
if __name__ == '__main__':
    main()
```

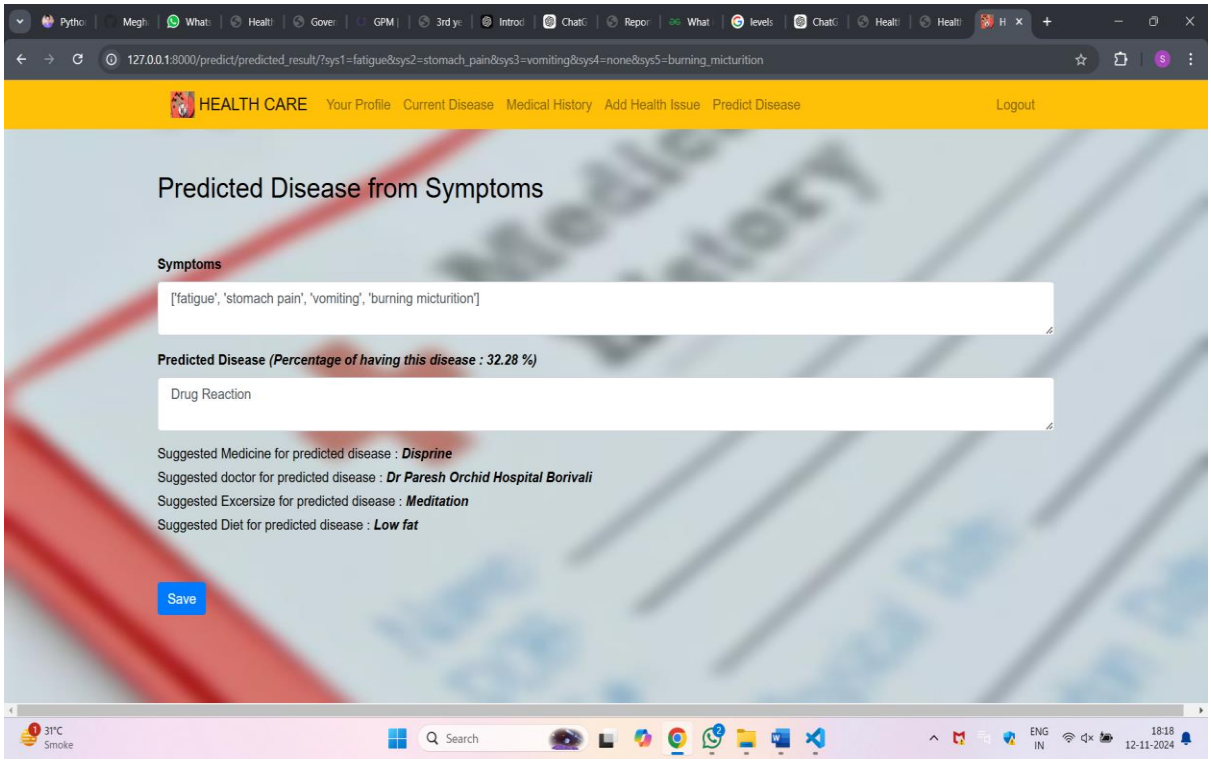
OUTPUT :





User Dashboard





Prediction of Disease

Smart Health Card

5.2 References

- Wenxing Hong, Ziang Xiong, Nannan Zheng, Yang Weng, “A Medical-HistoryBased Potential Disease Prediction Algorithm”, A Medical-History-Based Potential Disease Prediction Algorithm IEEE Access VOLUME 7, 2019, doi 10.1109/ACCESS.2019.2940644.

- Dahiwade, D., Patle, G., Meshram, E. (2019). Designing Disease Prediction Model Using Machine Learning Approach. 2019 Proceedings of the Third International Conference on Computing Methodologies and Communication (ICCMC 2019) IEEE Xplore doi:10.1109/iccmc.2019.8819782.
- For images/illustrations:
  https://undraw.co/
  https://lexica.art/
- For UI components:
  https://getwaves.io/
- For diagrams:
  https://practice.geeksforgeeks.org/
  https://www.javatpoint.com/ https://google.com/
- For theoretical explanation:
  https://www.javatpoint.com/
  https://practice.geeksforgeeks.org/ https://google.com/
- Logical help and innovative ideas:
  https://www.upgrad.com/
  https://tailwindcss.com/
  https://react.dev/

5.3 Future Scope :

The "Smart Health Card Using Machine Learning" project introduces an innovative approach to preventive healthcare by using machine learning algorithms to predict diseases based on users' symptoms. Developed by students, this system leverages the K-Nearest Neighbor (KNN) and Convolutional Neural Network (CNN) algorithms to provide accurate predictions and determine the risk level (high or low) associated with each diagnosis. The core goal of the project is to empower users to obtain early insights into their health conditions and promote timely intervention without requiring immediate physician visits.

In addition to predicting diseases, the system provides personalized recommendations for treatments, diet plans, and exercise routines tailored to the user’s specific health profile. An important feature of this project is the integration of a digital medical history log, where users can store and access past health data.

The Smart Health Card created for each user offers a unique digital ID, which can be used to link their entire medical history, making it easily accessible across different healthcare providers. Future expansions include enhancing the prediction accuracy of the system by increasing the dataset size and diversity, incorporating biometric authentication for added security, and potentially linking the Smart Health Card with government-issued IDs for seamless healthcare management.

Overall, this project demonstrates the potential of machine learning in revolutionizing healthcare by making it more accessible, personalized, and proactive, particularly in preventive care and early disease intervention.

5.4 CONCLUSION

Going to increment general disease prediction system based on machine learning algorithm. Smart health prediction system is software that help to people/user to detect disease by enter symptom and it show disease is high or low risk and user also show what type of medicine can implement user health and show best diet plan and if user want to go best doctor in their area or best doctor in India. User can go there or user can follow given tablet, diet, exercise by system. User able to add their medical history that help to doctor to get information about the patient. Develop smart health card user give their medical id to doctor, doctor login enter patient medical id and access patient data.

System give possibility or percentage accuracy of disease as result. To develop this System python software is first need. In python using Html design front end of software. Most important we use Django framework in python Django is important framework here to develop the project using django create server and database and develop this project we use must import CNN algorithm. CNN algorithm help to accuracy of disease. Because of this system may leads in low time consumption and minimal cost possible for disease prediction and risk prediction in future all use this system it would be easier for the doctors and medical researchers to find a remedy for the problem.

In future we can also improve accuracy of having this disease and daily update data of medicine and best doctor. In future we can add biomatrix system to smart health card or we can direct link our system to government issue id proof as like with Adhaar card so we can direct using biomatrix can find user details.