# ALZHEIMER DISEASE PREDICTION USING MACHINE LEARNING ALGORITHMS

*An Application Development – 2 (Project) Report Submitted*
*In partial fulfillment of the requirement for the award of the degree of*

*Bachelor of Technology*
*in*
*Computer Science and Engineering (Internet Of Things)*

**by**

**L.SREENIVAS REDDY** **20N31A6933**

*Under the Guidance of*

**Mrs. CH PAVITHRA**
**ASSISTANT PROFESSOR**
**Department of Emerging Technologies**
**MRCET (Autonomous Institution, UGC Govt. of India)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**(EMERGING TECHNOLOGIES)**
**MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY**
**(Autonomous Institution - UGC, Govt. of India)**
**(Affiliated to JNTU, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC – 'A' Grade, ISO 9001:2015 Certified)**
Maisammaguda (v), Near Dullapally, Via: Kompally, Hyderabad – 500 100, Telangana State, India
**2022-2023**

LIVE TO LEARN & LEARN TO SHARE

Estd : 2004

**MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous Institution – UGC, Govt. of India)**

(Sponsored by CMR Educational Society)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

( Affiliated to JNTUH,Hyderabad, Approved by AICTE- Accredited by NBA & NAAC– 'A' Grade - ISO 9001:2015 Certified )

UGC AUTONOMOUS

# CERTIFICATE

This is to certify that this is the bonafide record of the project titled **"Alzheimer Disease Prediction using Machine Learning Algorithms"** submitted by **L.Sreenivas Reddy (20N31A6933)** of **B.Tech III Year – II Semester** in the partial fulfillment of the requirements for the degree of **Bachelor of Technology** in **Computer Science and Engineering (Internet of Things)**, Dept. of CSE (Emerging Technologies) during the year 2022-2023. The results embodied in this project report have not been submitted to any other university or institute for the award of anydegree or diploma.

**Mrs.CH Pavithra**
**Assistant Professor**
**Department of CSE (ET)**

**Dr. P Dileep**
**Project Coordinator**
**Department of CSE (ET)**

**EXTERNAL EXAMINER**

**Dr. M V Kamal**
**Professor &**
**Head of the Department**

# DECLARATION

I hereby declare that the project entitled **“Alzheimer Disease Prediction using Machine Learning Algorithms”** submitted to **Malla Reddy College of Engineering and Technology,** affiliated t**o** Jawaharlal Nehru Technological University Hyderabad (JNTUH) as part of III Year B.Tech – II Semester and for the partial fulfillment of the requirement for the award of **Bachelor of Technology** in **Computer Science and Engineering (CSE-IOT)** is a result of original research work done by ourself.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

**L.SREENIVAS REDDY** **20N31A6933**

# ACKNOWLEDGEMENTS

I feel myself honored and privileged to place my warm salutation to my college "Malla Reddy College of Engineering and Technology (Autonomous Institution – UGC Govt. of India) and my Principal **Dr. S Srinivasa Rao,** Professor who gave us the opportunity to do the Application Development -2 (Project) during my III Year B.Tech and profound the technical skills.

I express my heartiest thanks to our Director **Dr. V S K Reddy,** Professor for encouraging us in every aspect of my project and helping me realize my full potential.

I also thankful to my Head of the Department **Dr. M V Kamal,** Professor for providing training and guidance, excellent infrastructure and a nice atmosphere for completing this project successfully.

I would like to express my sincere gratitude and indebtedness to my project supervisor **Mrs. CH Pavithra,** Assistant Professor for her valuable suggestions and interest throughout the course of this project.

I convey my heartfelt thanks to my Project Coordinator **Dr. P Dileep,** Professor for allowing for their regular guidance and constant encouragement during my dissertation work.

I would like to thank all our supporting **staff** of the Department of CSE (Emerging Technologies) and even all other department who have been helpful directly and in-directly in making our project a success.

Finally, I would like to take this opportunity to thank my **family** for their support and blessings for completion of my project that gave me the strength to do my project.

**L.SREENIVAS REDDY** **20N31A6933**

# ABSTRACT

Alzheimer's is the main reason for dementia that affects frequently older adults. This disease is costly especially, in terms of treatment. In addition, Alzheimer's is one of the deaths causes in the old-age citizens. Early Alzheimer's detection helps medical staffs in disease diagnosis, which will certainly decrease the risk of death. This made the early Alzheimer's disease detection a crucial problem in the healthcare industry. The objective of this research study is to introduce a computer-aided diagnosis system for Alzheimer’s disease detection using machine learning techniques. We employed data from the Alzheimer’s disease Neuro imaging Initiative (AONI). Common supervised machine learning techniques have been applied for automatic Alzheimer’s disease detection such as: support vector machine, random forest. Thus if the disease is predicted earlier, the progression or the symptoms of the disease can be slowed down. In this project we use machine learning algorithms to predict Alzheimer disease using psychological parameters like age, number of visits, MMSE and education.

Keywords: Alzheimer disease, machine learning algorithms, psychological parameters.

# TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

Alzheimer's Disease (AD) is a progressive neurological condition that leads to short-term memory loss, paranoia, and delusional ideas that are mistaken for the effects of stress or aging. In the United States, this Disease affects about 5.1 million people. AD does not have proper medical treatment. In order to control AD, continuous medication is necessary. AD (1) is chronic so that it can last for years or the rest of your life. Therefore, it is most important to prescribe medication at the appropriate stage so that the brain is not damaged to a great extent. Early detection of this Disease is a tedious and costly process since we must collect a lot of data and use sophisticated tools for prediction and have an experienced doctor involved. Automated systems are more accurate than human assessment and can be used in medical decision support systems because they are not subject to human errors. Based on previous research on AD, researchers have applied images (MRI scans), biomarkers (chemicals, blood flow), and numerical data extracted from the MRI scans to study this Disease. As such, they were able to determine whether a person was demented or not. In addition to shortening diagnosis time, more human interaction will be reduced by automating Alzheimer's diagnosis. In addition, automation reduces overall costs and provides more accurate results. For example, we can predict whether a patient is demented by analyzing MRI scans and applying prediction techniques. If a person has early-stage Alzheimer's Disease, they are considered demented. By doing so, we can achieve better accuracy.

When a person has Alzheimer's Disease in the early stages, they can usually function without any assistance. In some cases, the person can still work, drive, and partake in social activities. Although this is the case, the person may still feel uneasy or suffer from memory loss, such as not remembering familiar words and locations. People close to the individual notice that they have difficulty remembering their names. By conducting a detailed medical interview, a doctor may identify problems with memory and concentration in the patient.

Common challenges in early stage of Alzheimer's Disease include,

• It's hard to remember the right word or name.
• Having difficulty remembering names when meeting new people.
• Working in social settings or the workplace every day can be challenging.
• Having forgotten something that you have just read in a book or something else.
• Having trouble finding or misplacing a valuable object.
• Tasks and activities are becoming increasingly difficult to plan or organize.

## 1.1 MOTIVATION:

Under the current conditions, human instinct and standard measurements do not often coincide. In order to solve this problem, we need to leverage innovative approaches such as machine learning, which are computationally intensive and non-traditional. Machine learning techniques are increasingly being used in disease prediction and visualization to offer prescient and customized prescriptions. In addition to improving patients' quality of life, this drift aids physicians in making treatment decisions and health economists in making their analyses. Viewing medical reports may lead radiologists to miss other disease conditions. As a result, it only considers a few causes and conditions. The goal here is to identify the knowledge gaps and potential opportunities associated with ML frameworks and EHR derived data.

## 1.2 PROBLEM DEFINATION:

European e-health study shows that in each year, there are around 2.4 Million unnecessary healthcare center visits by patients. Elderly people, including Alzheimer disease patients, are among those who are unnecessarily visiting the hospitals. AD patients can diagnosed using different techniques but physical examination, laboratory test, and brain imaging scan require the physical appearance of the patient to the medical center. This results a large number of AD patients that visit the healthcare institutions. The increase in the number of visitors create a workload on professionals, high number of patients’ queues and extra cost in terms of time and money on both the patients and health institutions. On each visit, the healthcare centers register and stored massive patients' data for future follow up. Usually these massive data are used only when it is necessary to refer medical history of patients. Machine learning creates another possibility to diagnosis AD patients using the massive stored cognitive function and medical history data. In this research, machine learning and agent system will be combined with cognitive function and medical history data to diagnose patients based the existing 47,000 AD patients' data.

## 1.3 OBJECTIVE OF THE PROJECT:

- Identify and evaluate the existing AD diagnosis techniques
- Evaluate different machine leaning algorithm for classification purpose
- Identify a suitable machine learning classification algorithm to classify patient’s AD stages using NACC dataset
- Customize and implement the selected machine learning algorithm.
- Develop multi agent system model to improve the classification accuracy

# CHAPTER 2

# SYSTEM ANALYSIS

## 2.1 EXISTING SYSTEM:

Method of deep learning along with the brain network and clinical significant information like age, APOE gene and gender of the subjects for earlier examination of Alzheimer’s. Brain network was arranged, calculating functional connections in the brain region by employing the resting-state functional magnetic resonance imaging (R-fMRI) data. To produce a detailed discovery of the early AD, a deep network like auto encoder is used where functional connections of the networks are constructed and are susceptible to AD and MCI. The dataset is taken from the ADNl database. The classification model consists of the early diagnosis. initially pre-processing of raw R-fMR1 is done. Then, the time series data (90 x I 30matrix) is obtained and that indicates blood oxygen levels in each and every region of the brain and changes over a long peri0d. Then, a brain network is built and transformed to a 90 x90 time series data correlation matrix. The targeted autoencoder model is used which is a three layered model which gives intellectual growth of the nervous system then extracts brain networks attributes completely When a finite amount of data cases is taken, k-fold cross verification was implemented mainly to avoid the over fitting complication.

DISADVANTAGES OF EXISTING SYSTEM:

- The ability to collect, store, manage and process data has been difficult in existing methods.
- The stage of artificial intelligence is also defined as a discipline about knowledge, namely the technology about how to acquire and express the knowledge and convert it into practical applications

## 2.2 PROPOSED SYSTEM:

Proposed a method called multistage classifier by using machine learning algorithms like Support vector Machine, Naive Bayes and K-nearest neighbor to classify below different subjects. PSO(particle swarm optimization) which is technique that best selects the features was enforced to obtain best features. Naturally image retrieving process requires two stages: the first stage involves generating features so that it reproduces the query image and then later step correlate those features with those already gathered in the database. The PRO algorithm is used to select the finest biomarkers that show AD or MCI. The data is Alzheimer's disease Neuroimaging Initiative (ADNI) database. The MRI scans are pre-processed first after taking from the database. The feature selection includes volumetric and thickness measurements. Then the optimum feature lists were obtained from PSO algorithm. The Gaussian Naive Bayes, K- Nearest Neighbor, Support vector machine was used to distinguish between the subjects. Here a 2 stage c1assifier was used where in the initial stage GNB classified was used to classify the objects between AD, MCI and NC and in later stages SVM and KNN were used to analyze the object based on the performance of the initial one. Control Based Image Retrieval was used for retrieving images from the database.

ADVANTAGES OF PROPOSED SYSTEM:

- SVM is a directed study model that classifies by separating the objects using a hyper plane.
- It can be used for both classification and regression. The hyperplanes are drawn
- with the help of the margins.
- The main goal is to maximize the distance between the hyperplane and the margin.
- The margins are drawn with the help of support vectors that arc belonging to the objects.
- The main advantage of SVM is that it can distinguish linear and non-linear objects.

## 2.3 FUNCTIONAL REQUIREMENTS (HARDWARE AND SOFTWARE):

### SOFTWARE REQUIREMENTS:

| | |
|---|---|
| • Operating System | : Windows 7 |
| • Coding Language | : Python |
| • Tool | : Pycharm , Visual Studio Code |
| • Database | : Sqlite |

### HARDWARE REQUIREMENTS:

| | |
|---|---|
| • System | : Intel I-3,5,7 or above processor |
| • Hard Disk | : Min 500 GB |
| • Ram | : Min 4GB |
| • Floppy Drive | : 1.44 Mb |

# CHAPTER 3

# SOFTWARE ENVIRONMENT

## 3.1 PYTHON:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python:**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features:**

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

## PYTHON RESERVED WORDS:

Reserved words (also called keywords) are defined with predefined meaning and syntax in the language. These keywords have to be used to develop programming instructions. Reserved words can't be used as identifiers for other programming elements like name of variable, function etc.

Following is the list of reserved keywords in Python 3

| | | | |
|---|---|---|---|
| and | except | lambda | with |
| as | finally | nonlocal | while |
| assert | false | None | yield |
| break | for | not | |
| class | from | or | |
| continue | global | pass | |
| def | if | raise | |
| del | import | return | |
| elif | in | True | z |
| else | is | try | |

Table 1 : Python keywords

Python 3 has 33 keywords while Python 2 has 30. The print has been removed from Python 2 as keyword and included as built-in function.

## PYTHON IDENTIFIERS:

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, **Manpower** and **manpower** are two different identifiers in Python.

Here are naming conventions for Python identifiers −

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is private.
- Starting an identifier with two leading underscores indicates a strongly private identifier.
- If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

## PYTHON DATA TYPES:

Python Data Types are used to define the type of a variable. It defines what type of data we are going to store in a variable. The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters.

Python has various built-in data types which we will discuss with in this tutorial:

- Numeric - int, float, complex
- String - str
- Sequence - list, tuple, range
- Binary - bytes, bytearray, memoryview
- Mapping - dict
- Boolean - bool
- Set - set, frozenset
- None - NoneType

## PYTHON LIBRARIES:

The Python Standard Library contains the exact syntax, semantics, and tokens of Python. It contains built-in modules that provide access to basic system functionality like I/O and some other core modules. Most of the Python Libraries are written in the C programming language. The Python standard library consists of more than 200 core modules. All these work together to make Python a high-level programming language. Python Standard Library plays a very important role. Without it, the programmers can’t have access to the functionalities of Python. But other than this, there are several other libraries in Python that make a programmer’s life easier. Let’s have a look at some of the commonly used libraries:

1. **TensorFlow:** This library was developed by Google in collaboration with the Brain Team. It is an open-source library used for high-level computations. It is also used in machine learning and deep learning algorithms. It contains a large number of tensor operations. Researchers also use this Python library to solve complex computations in Mathematics and Physics.
2. **Matplotlib:** This library is responsible for plotting numerical data. And that’s why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.
3. **Pandas:** Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.
4. **Numpy:** The name “Numpy” stands for “Numerical Python”. It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations. Even libraries like TensorFlow use Numpy internally to perform several operations on tensors. Array Interface is one of the key features of this library.
5. **SciPy:** The name “SciPy” stands for “Scientific Python”. It is an open-source library used for high-level scientific computations. This library is built over an extension of Numpy. It works with Numpy to handle complex computations. While Numpy allows sorting and indexing of array data, the numerical data code is stored in SciPy. It is also widely used by application developers and engineers.

6. **Scrapy:** It is an open-source library that is used for extracting data from websites. It provides very fast web crawling and high-level screen scraping. It can also be used for data mining and automated testing of data.
7. **Scikit-learn:** It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.
8. **PyGame:** This library provides an easy interface to the Standard Direct media Library (SDL) platform-independent graphics, audio, and input libraries. It is used for developing video games using computer graphics and audio libraries along with Python programming language.
9. **PyTorch:** PyTorch is the largest machine learning library that optimizes tensor computations. It has rich APIs to perform tensor computations with strong GPU acceleration. It also helps to solve application issues related to neural networks.
10. **PyBrain:** The name “PyBrain” stands for Python Based Reinforcement Learning, Artificial Intelligence, and Neural Networks library. It is an open-source library built for beginners in the field of Machine Learning. It provides fast and easy-to-use algorithms for machine learning tasks. It is so flexible and easily understandable and that’s why is really helpful for developers that are new in research fields.

## 3.2 DJANGO:

Django is a web development framework that assists in building and maintaining quality web applications. Django helps eliminate repetitive tasks making the development process an easy and time saving experience.

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Django makes it easier to build better web apps quickly and with less code.

**Note** − Django is a registered trademark of the Django Software Foundation, and is licensed under BSD License.

## Django – Design Philosophies:

Django comes with the following design philosophies −

- **Loosely Coupled** − Django aims to make each element of its stack independent of the others.
- **Less Coding** − Less code so in turn a quick development.
- **Don't Repeat Yourself (DRY)** − Everything should be developed only in exactly one place instead of repeating it again and again.
- **Fast Development** − Django's philosophy is to do all it can to facilitate hyper-fast development.
- **Clean Design** − Django strictly maintains a clean design throughout its own code and makes it easy to follow best web-development practices.

## Advantages of Django:

Here are few advantages of using Django which can be listed out here −

- **Object-Relational Mapping (ORM) Support** − Django provides a bridge between the data model and the database engine, and supports a large set of database systems including MySQL, Oracle, Postgres, etc. Django also supports NoSQL database through Django-nonrel fork. For now, the only NoSQL databases supported are MongoDB and google app engine.
- **Multilingual Support** − Django supports multilingual websites through its built-in internationalization system. So you can develop your website, which would support multiple languages.
- **Framework Support** − Django has built-in support for Ajax, RSS, Caching and various other frameworks.
- **Administration GUI** − Django provides a nice ready-to-use user interface for administrative activities.
- **Development Environment** − Django comes with a lightweight web server to facilitate end-to-end application development and testing.

As you already know, Django is a Python web framework. And like most modern framework, Django supports the MVC pattern. First let's see what is the Model-View-Controller (MVC) pattern, and then we will look at Django's specificity for the Model-View-Template (MVT) pattern.

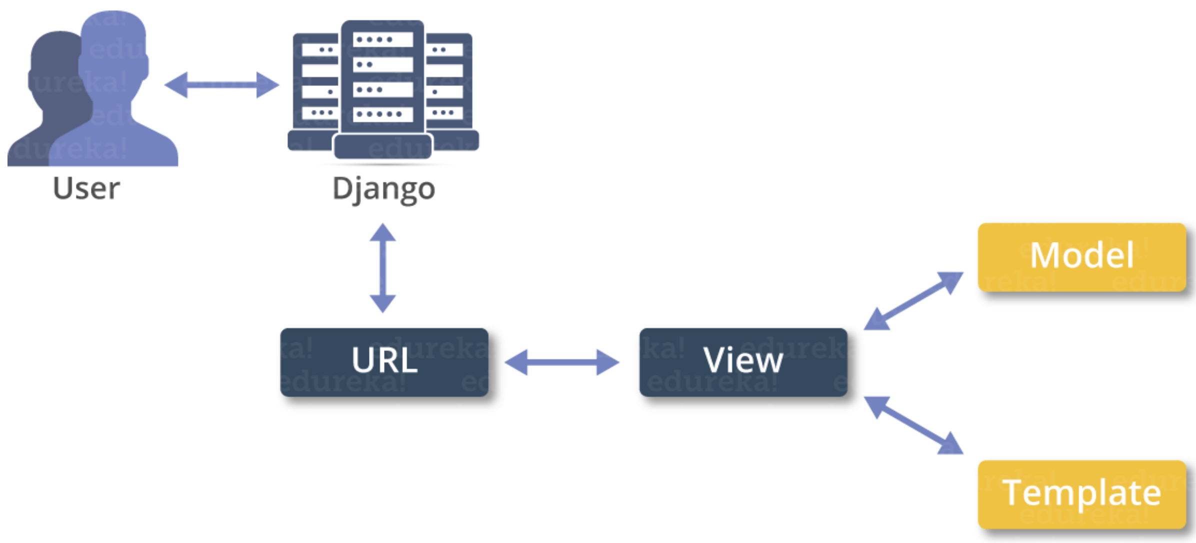
## MVC Pattern

When talking about applications that provides UI (web or desktop), we usually talk about MVC architecture. And as the name suggests, MVC pattern is based on three components: Model, View, and Controller.

## DJANGO MVC - MVT Pattern

The Model-View-Template (MVT) is slightly different from MVC. In fact the main difference between the two patterns is that Django itself takes care of the Controller part (Software Code that controls the interactions between the Model and View), leaving us with the template. The template is a HTML file mixed with Django Template Language (DTL).

The following diagram illustrates how each of the components of the MVT pattern interacts with each other to serve a user request −



The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

## 3.3 <u>MODULES</u>:

- Doctor
- Patient
- Admin

**<u>MODULES DESCRIPTION</u>**:

**<u>DOCTOR</u>**:

- The Doctor can register the first.
- While registering he required a valid doctor email and mobile for further communications.
- Once the doctor registers, then the admin can activate the customer. Once the admin activates the doctor then the doctor can login into our system.
- After login he can see the view-patient data. based on patient symptoms, the doctor will give the precautions and he will give the doctor treatment.

**<u>PATIENT</u>**:

- The patient can register the first.
- While registering he required a valid patient email and mobile for further communications.
- Once the patient registers, then the admin can activate the patient.
- Once the admin activates the patient then the patient can login into our system.
- patient will submit his symptoms to the doctor.

**<u>ADMIN</u>**:

- Admin can login with his credentials.
- Once he logs in he can activate the doctors. The activated patient only login in our applications.
- Once he logs in he can see body details first he can get the data from the doctor. So this data user can perform the testing process.
- Admin performs svm implementation to the Alzheimer-disease.

# CHAPTER 4

# SYSTEM DESIGN AND UML DIAGRAM

## 4.1 DATAFLOW DIAGRAM:

**DFD** is the abbreviation for **Data Flow Diagram**. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart.

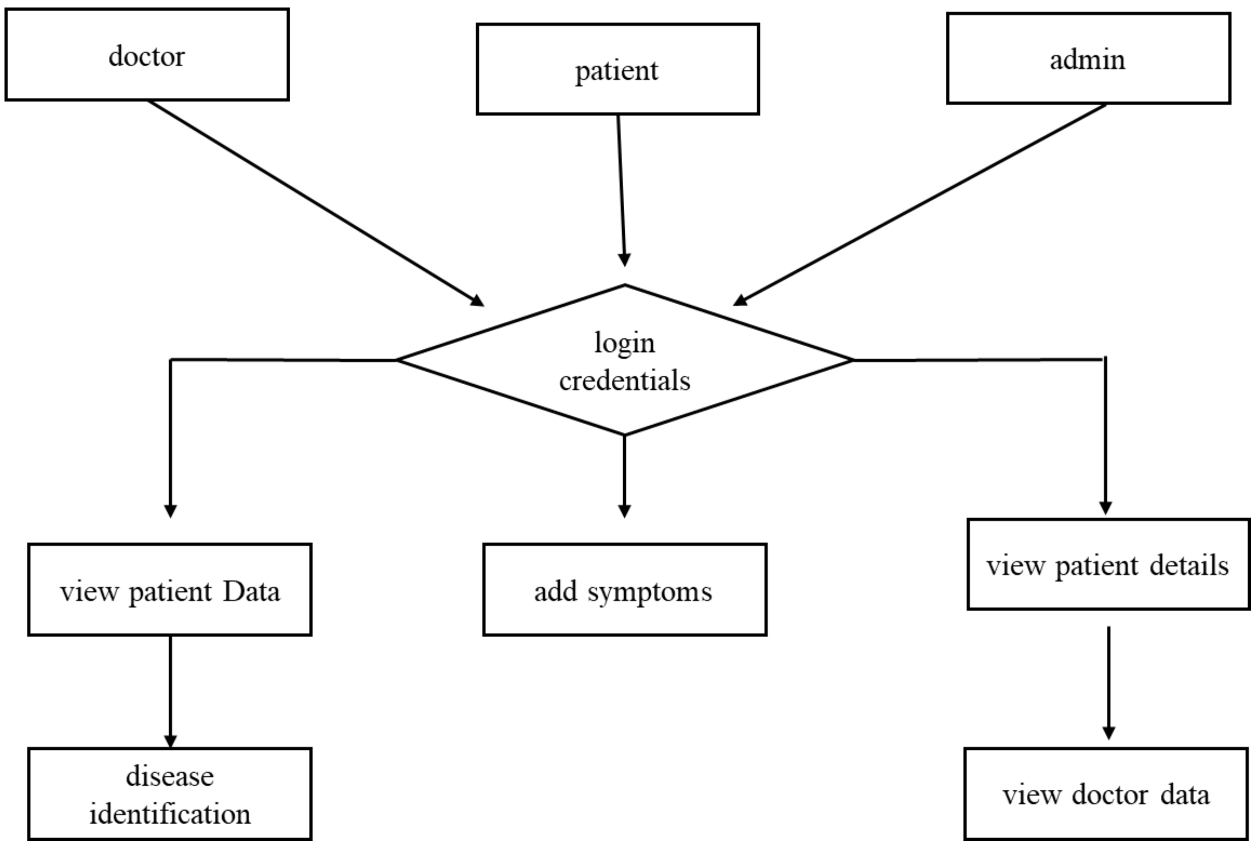


Fig 1: Dataflow Diagram

## 4.2 <u>ARCHITECTURE DIAGRAM</u>:

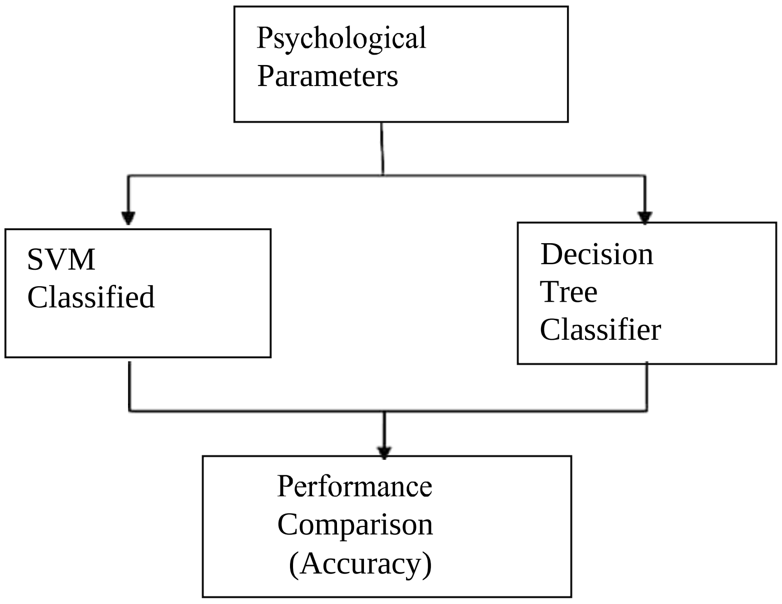


Fig 2: Architecture Diagram

## 4.3 <u>UML DIAGRAMS</u>:

### 4.3.1 <u>USE CASE DIAGRAM</u>:

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modeling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

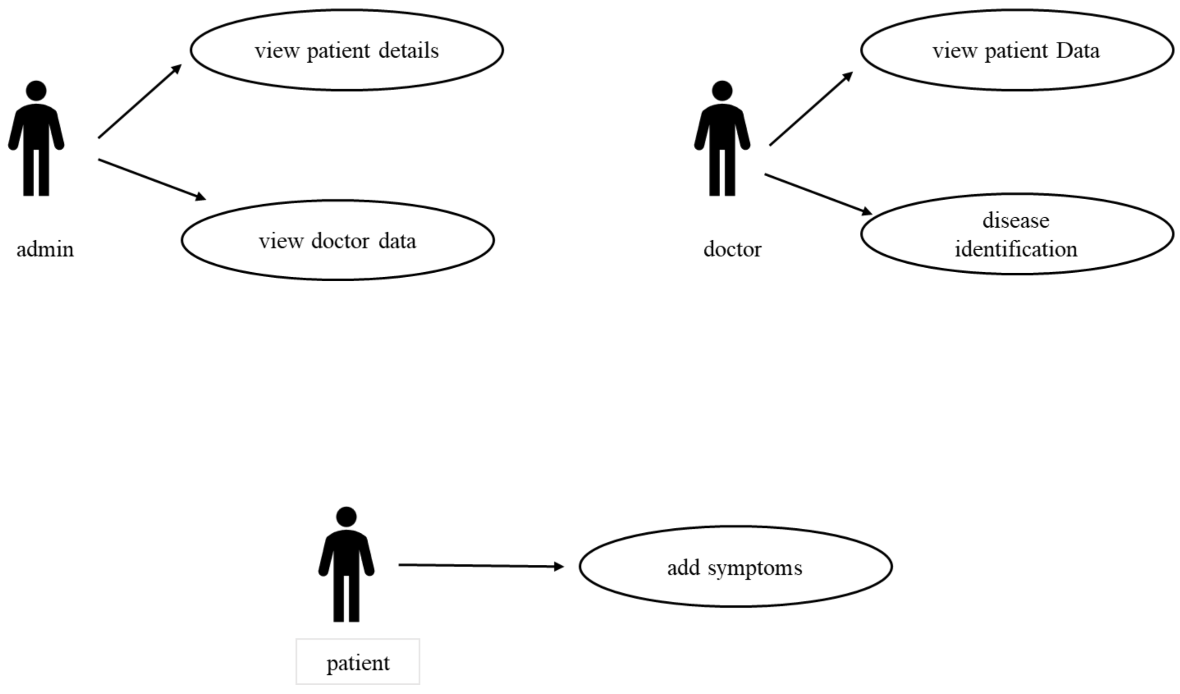


Fig 3: Use Case Diagram

### 4.3.2 CLASS DIAGRAM:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

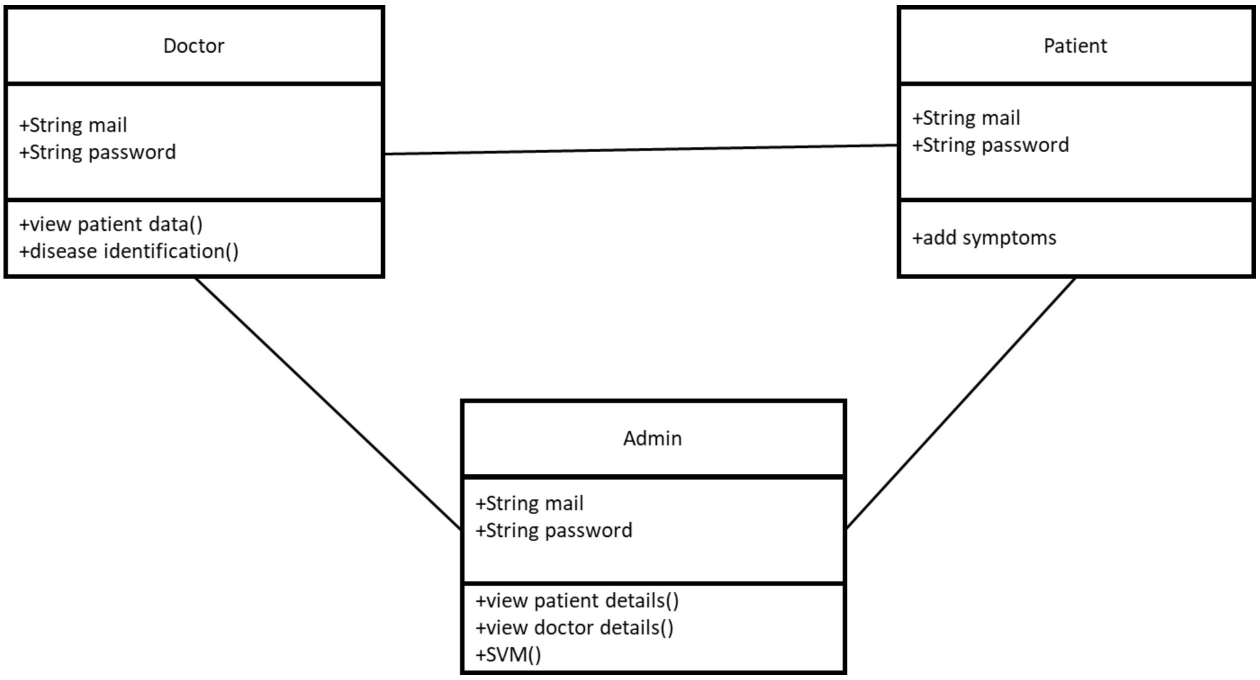


Fig 4: Class Diagram

### 4.3.2 SEQUENTIAL DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

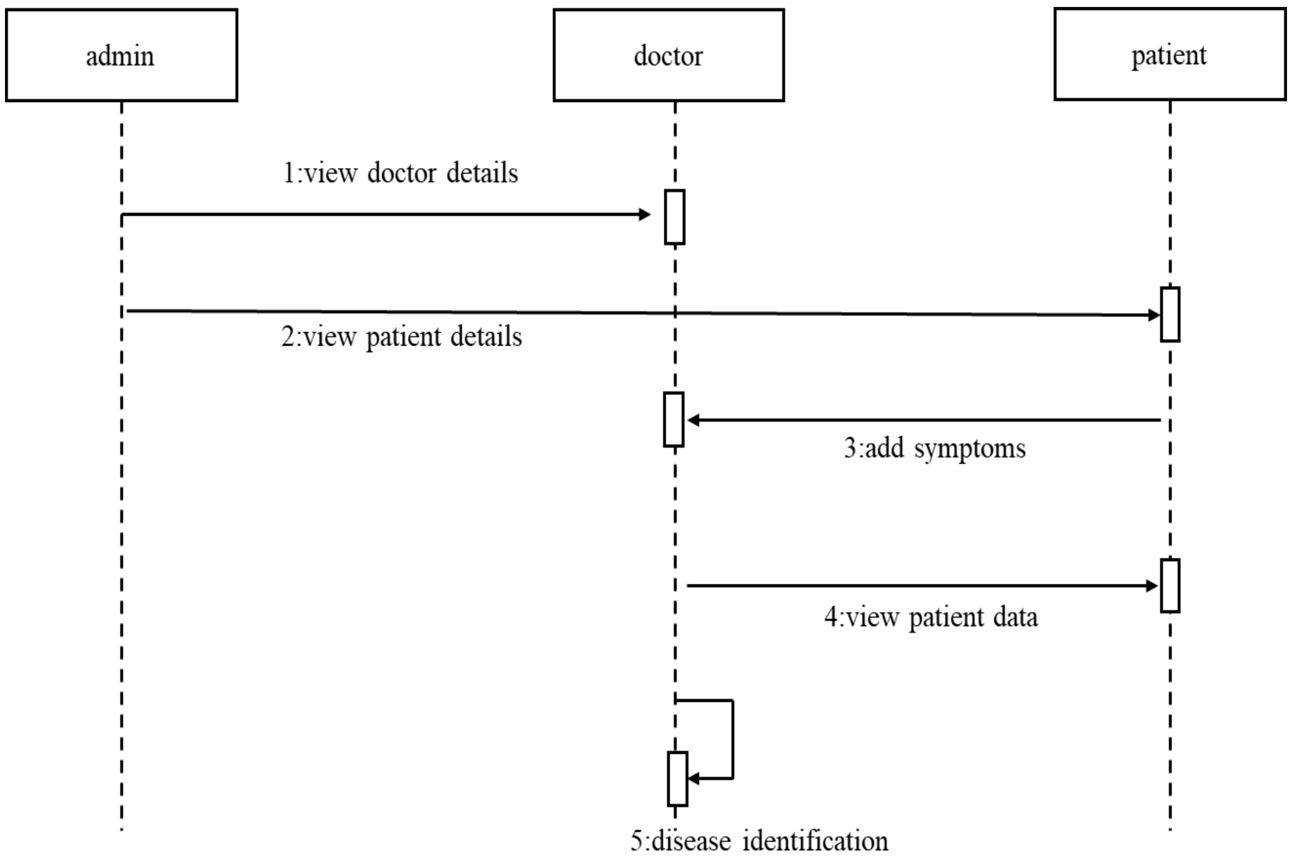


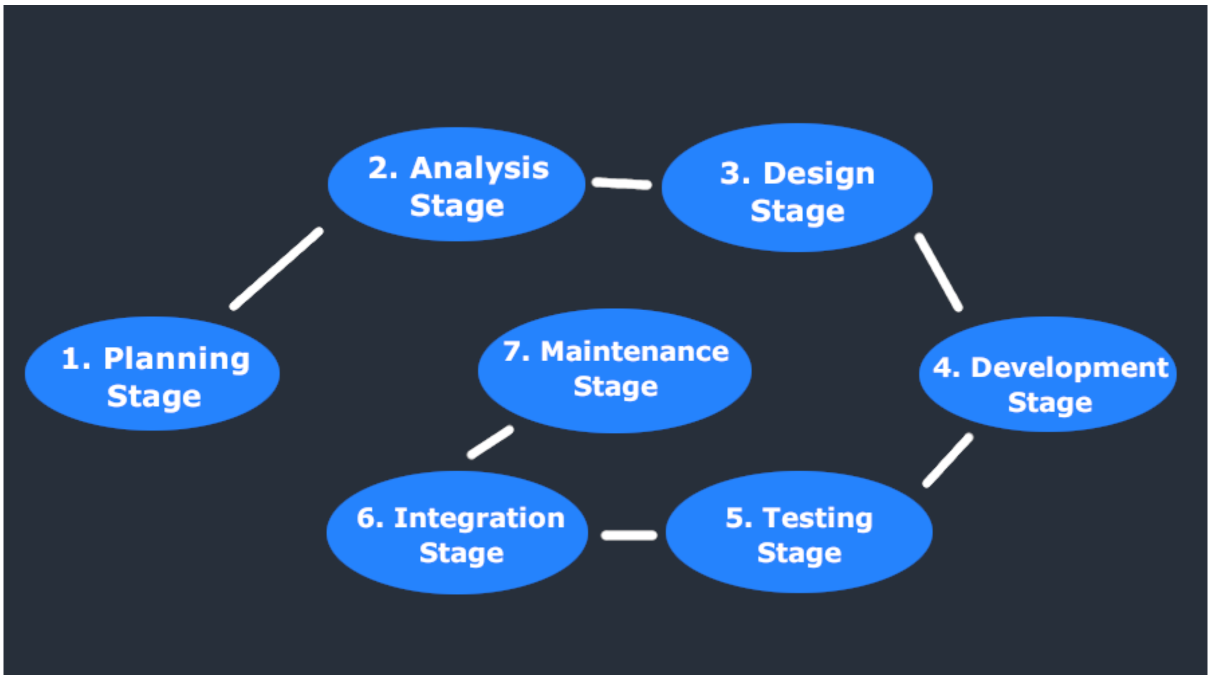
Fig 5: Sequential Diagram

# CHAPTER 5

# SOFTWARE DEVELOPMENT LIFE CYCLE

## 5.1 PHASES OF SDLC:

A system development life cycle or SDLC is essentially a project management model. It defines different stages that are necessary to bring a project from its initial idea or conception all the way to deployment and later maintenance.



### 7 Stages of the System Development Life Cycle:

There are seven primary stages of the modern system development life cycle. Here’s a brief breakdown:

- Planning Stage

- Feasibility or Requirements of Analysis Stage
- Design and Prototyping Stage
- Software Development Stage
- Software Testing Stage
- Implementation and Integration
- Operations and Maintenance Stage

Now let’s take a closer look at each stage individually.

## **Planning Stage**:

Before we even begin with the planning stage, the best tip we can give you is to take time and acquire proper understanding of app development life cycle.

The planning stage (also called the feasibility stage) is exactly what it sounds like: the phase in which developers will plan for the upcoming project.

It helps to define the problem and scope of any existing systems, as well as determine the objectives for their new systems.

By developing an effective outline for the upcoming development cycle, they'll theoretically catch problems before they affect development.

And help to secure the funding and resources they need to make their plan happen.

Perhaps most importantly, the planning stage sets the project schedule, which can be of key importance if development is for a commercial product that must be sent to market by a certain time.

## **Analysis Stage**:

The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

- Define any prototype system requirements
- Evaluate alternatives to existing prototypes
- Perform research and analysis to determine the needs of end-users

Furthermore, developers will often create a software requirement specification or SRS document.

This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

## **Design Stage**:

The design stage is a necessary precursor to the main developer stage.

Developers will first outline the details for the overall application, alongside specific aspects, such as its:

- User interfaces
- System interfaces
- Network and network requirements
- Databases

They'll typically turn the SRS document they created into a more logical structure that can later be implemented in a programming language. Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward.

Once complete, development managers will prepare a design document to be referenced throughout the next phases of the SDLC.

## **Development Stage:**

The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications.

This is where Static Application Security Testing or SAST tools come into play.

Product program code is built per the design document specifications. In theory, all of the prior planning and outlined should make the actual development phase relatively straightforward.

Developers will follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers, and interpreters.

Programming languages can include staples such as C++, PHP, and more. Developers will choose the right programming code to use based on the project specifications and requirements.

## **Testing Stage:**

Building software is not the end.

Now it must be tested to make sure that there aren't any bugs and that the end-user experience will not negatively be affected at any point.

During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed, and later retested.

t's important that the software overall ends up meeting the quality standards that were previously defined in the SRS document.

Depending on the skill of the developers, the complexity of the software, and the requirements for the end-user, testing can either be an extremely short phase or take a very long time.

## **Implementation and Integration Stage:**

After testing, the overall design for the software will come together. Different modules or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects.

The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for market and may be provided to any end-users.

**Maintenance Stage:**

The SDLC doesn’t end when software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end-users.

Furthermore, developers are responsible for implementing any changes that the software might need after deployment.

This can include handling residual bugs that were not able to be patched before launch or resolving new issues that crop up due to user reports. Larger systems may require longer maintenance stages compared to smaller systems.

# CHAPTER 6

# IMPLEMENTATION

## 6.1 SAMPLE CODE:

**urls.py**

```
from django.contrib import admin
from django.urls import path
from alzmierdisease import views as alzmr
from patient import views as patient
from doctor import views as doctor

urlpatterns = [
    # path('admin/', admin.site.urls),
    path('index/',alzmr.index, name="index"),
    path('base/', alzmr.base, name="base"),
    path('admin1/', alzmr.adminlogin, name="admin1"),
    path('adminloginaction/', alzmr.adminloginaction, name="adminloginaction"),
    path('activatepatient/',alzmr.activatepatient,name='activatepatient'),
    path('testing/', alzmr.testing, name="testing"),
    path('svm/', alzmr.svm, name="svm"),
    path('svm1/', alzmr.svm1, name="svm1"),
    path('logout/', alzmr.logout, name="logout"),
    path('adminDecisionTree/', alzmr.adminDecisionTree, name='adminDecisionTree'),


    path('doctorlogin/',doctor.doctorlogin,name='doctorlogin'),
    path('doctorregister/',doctor.doctorregister,name='doctorregister'),
    path('doctordetails/', doctor.doctordetails, name='doctordetails'),
    path('activatedoctor/',doctor.activatedoctor,name='activatedoctor'),
    path('doctorlogincheck/',doctor.doctorlogincheck,name='doctorlogincheck'),
    path('patientdata/',doctor.doctorviewpatientdata,name='patientdata'),
    path('addtreatment/',doctor.addtreatment,name='addtreatment'),
    path('report/',doctor.report,name='report'),



    path('patientlogin/', patient.patientlogin, name='patientlogin'),
    path('patientregister/', patient.patientregister, name='patientregister'),
    path('patientlogincheck/',patient.patientlogincheck,name='patientlogincheck'),
    path('patientdetails/', patient.patientdetails, name='patientdetails'),
    path('symptoms/', patient.patientsymptoms, name='symptoms'),
    path('patntsymptms/', patient.patntsymptms, name='patntsymptms'),


]
```

**doctormodel.py**

```
from django.db import models

class doctorModel(models.Model):
    name = models.CharField(max_length=50)
    email = models.EmailField()
    passwd = models.CharField(max_length=40)
    cwpasswd = models.CharField(max_length=40)
    mobileno = models.CharField(max_length=50, default="", editable=True)
    specialization = models.CharField(max_length=50, default="", editable=True)
    status = models.CharField(max_length=40, default="", editable=True)

    def  __str__(self):
        return self.email

    class Meta:
        db_table='doctorregister'
```

**doctorforms.py**

```
from django import forms
from doctor.models import *
from django.core import validators


class doctorForm(forms.ModelForm):
    name = forms.CharField(widget=forms.TextInput(), required=True, max_length=100,)
    passwd = forms.CharField(widget=forms.PasswordInput(), required=True,
max_length=100)
    cwpasswd = forms.CharField(widget=forms.PasswordInput(), required=True,
max_length=100)
    email = forms.CharField(widget=forms.TextInput(),required=True)
    mobileno= forms.CharField(widget=forms.TextInput(), required=True,
max_length=10,validators=[validators.MaxLengthValidator(10),validators.MinLengthValidat
or(10)])
    specialization= forms.CharField(widget=forms.TextInput(), required=True,
max_length=100)
    status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max_length=100)

    def__str_(self):
        return self.email

    class Meta:
        model=doctorModel
        fields=['name','passwd','cwpasswd','email','mobileno','specialization','status']
```

**doctorlogin.html**

```
{% extends 'base.html'%}
{% load static %}
{% block contents %}

<header id="gtco-header" class="gtco-cover">
        <div class="gtco-container">
            <div class="row">
                <div class="col-md-12 col-md-offset-0 text-left">
                    <div class="display-t">
                        <div class="display-tc">
                            <div class="row">
                                <h1><b>welcome to doctor page</b></h1>
                                <form action="/doctorlogincheck/" method="POST">
                             {% csrf_token %}<center>
                            <table><tr><td style="color:blue">
<!--                              <input  type="text" name="uname" placeholder="student Login"
><br/><br/>-->
                                <input type="email" name="email"  placeholder="Email">
                             <input  type="password" name="upasswd" placeholder="Password"
><br/><br/>
                                </td></tr></table> <input type="submit" style="color:#FF4500"
value="submit"><br/>
                            <br><br>New Doctor Register Here ?<a href="{% url 'doctorregister'
%}" style="color:blue">Click Here</a>
                        </center> </form> </h2>
                        {% if messages %}
                            {% for message in messages %}
                                <font color='darkred'> {{ message }}</font>
                            {% endfor %}
                        {% endif %}
                                </div>
                            </div>
                        </div>
```

```
              </div>
          </div>
      </div>

</header>

{% endblock %}
```

**Patientforms.py**

```
from django import forms
from patient.models import *
from django.core import validators

class patientForm(forms.ModelForm):
    name = forms.CharField(widget=forms.TextInput(), required=True, max_length=100,)
    passwd = forms.CharField(widget=forms.PasswordInput(), required=True,
max_length=100)
    cwpasswd = forms.CharField(widget=forms.PasswordInput(), required=True,
max_length=100)
    email = forms.CharField(widget=forms.TextInput(),required=True)
    mobileno= forms.CharField(widget=forms.TextInput(), required=True,
max_length=10,validators=[validators.MaxLengthValidator(10),validators.MinLengthValidat
or(10)])
    status = forms.CharField(widget=forms.HiddenInput(), initial='waiting', max_length=100)

    def__str_(self):
        return self.email

    class Meta:
        model=patientModel
        fields=['name','passwd','cwpasswd','email','mobileno','status']
```

**patientmodels.py**

```
from django.db import models


class patientModel(models.Model):
    name = models.CharField(max_length=50)
    email = models.EmailField()
    passwd = models.CharField(max_length=40)
    cwpasswd = models.CharField(max_length=40)
    mobileno = models.CharField(max_length=50, default="", editable=True)
    status = models.CharField(max_length=40, default="", editable=True)

    def  __str__(self):
        return self.email

    class Meta:
        db_table='patientregister'


from django.db import models
class patientsympmodel(models.Model):
    name = models.CharField(max_length=50)
    mobileno = models.CharField(max_length=50, default="", editable=True)
    gender = models.CharField(max_length=50, default="", editable=True)
    age= models.CharField(max_length=40, default="", editable=True)
    symptomps= models.CharField(max_length=40, default="", editable=True)
    familydata=models.CharField(max_length=40, default="", editable=True)


    def__str__(self):
        return self.name
    class Meta:
        db_table = 'patientdata'
```

**patientviews.py**

```
from django.shortcuts import render
from patient.forms import *
from django.http import HttpResponse

from builtins import Exception
from django.contrib import messages


def patientlogin(request):
    return render(request,'patient/patientlogin.html')

def patientregister(request):
        if request.method == 'POST':
            form=patientForm(request.POST)
            if form.is_valid():
                form.save()
                print("succesfully saved the data")
                return render(request, 'patient/patientlogin.html')
            else:
                print("form not valied")
                return HttpResponse("form not valied")
        else:
            form=patientForm()
            return render(request, "patient/patientregister.html", {"form":form})

def patientlogincheck(request):
    if request.method == 'POST':
        sname = request.POST.get('email')
        print(sname)
        spasswd = request.POST.get('upasswd')
        print(spasswd)
        try:
            check = patientModel.objects.get(email=sname,passwd=spasswd)
            # print('usid',usid,'pswd',pswd)
            print(check)
            request.session['name'] = check.name
            print("name",check.name)
            status = check.status
            print('status',status)
            if status == "Activated":
                request.session['email'] = check.email
                return render(request, 'patient/patientpage.html')
            else:
                messages.success(request, 'patient is not activated')
                return render(request, 'patient/patientlogin.html')
        except Exception as e:
            print('Exception is ',str(e))
            pass
        messages.success(request,'Invalid name and password')
        return render(request,'patient/patientlogin.html')

def patientdetails(request):
    qs=patientModel.objects.all()
    return render(request,'admin/patientdetails.html',{"object":qs})


def patntsymptms(request):
    name=request.POST.get('name')
    age=request.POST.get('age')
    mble=request.POST.get('mobile')
    gndr=request.POST.get('gndr')
    famly=request.POST.get('famly')
    symp=request.POST.get('symp')
    patientsympmodel.objects.create(name=name,age=age,mobileno=mble,gender=gndr,famil
ydata=famly,symptomps=symp)
    return render(request,'patient/patientsymptoms.html')


def patientsymptoms(request):
    return render(request,'patient/patientsymptoms.html')
```

**patientlogin.html**

```
{% extends 'base.html'%}
{% load static %}
{% block contents %}

<header id="gtco-header" class="gtco-cover">
            <div class="gtco-container">
                <div class="row">
                    <div class="col-md-12 col-md-offset-0 text-left">
                        <div class="display-t">
                            <div class="display-tc">
                                <div class="row">
                                    <h1><b>welcome to patient page</b></h1>
                                    <h2><form action="/patientlogincheck/" method="POST">
                                 {% csrf_token %}<center>
                                <table><tr><td style="color:blue">
<!--                                    <input  type="text" name="uname" placeholder="student Login"
><br/><br/>-->
                                    <input type="email" name="email"  placeholder="Email">
                                 <input  type="password" name="upasswd" placeholder="Password"
><br/><br/>
```

```
                              </td></tr></table> <input type="submit" style="color:#FF4500"
value="submit"><br/>
                           <br><br>New patient Register Here ?<a href="{% url 'patientregister'
%}" style="color:blue">Click Here</a>
                       </center> </form> </h2>
                        {% if messages %}
                           {% for message in messages %}
                              <font color='darkred'> {{ message }}</font>
                           {% endfor %}
                        {% endif %}

                              </div>
                           </div>
                        </div>
                     </div>
                  </div>
               </div>

</header>

{% endblock %}
```

# CHAPTER 7

# TESTING

## 7.1 SOFTWARE TESTING:

Software testing is a critical component of the software development lifecycle. It involves the process of evaluating software or a system's performance, functionality, and quality to identify and rectify defects, bugs, or errors. The primary purpose of software testing is to ensure that the software or system meets the specified requirements and performs as expected.

The types of software testing can be broadly classified into two categories: manual testing and automated testing. Manual testing involves executing test cases manually, while automated testing uses tools to automate the testing process.

Some of the most common types of software testing include:

**Unit Testing**: Unit testing is the process of testing individual units or components of the software or system in isolation. It involves testing each module or function of the software to ensure that it performs as expected and meets the specified requirements.

**Integration Testing**: Integration testing is the process of testing how different components or modules of the software interact with each other. It involves testing the interfaces between different modules to ensure that the software or system functions as expected when integrated.

**White-box Testing**: White-box testing, also known as structural testing or code-based testing, involves testing the internal structure and code of the software or system. It involves testing the software or system's logic, branches, loops, and paths to ensure that they function correctly.

**Black-box Testing**: Black-box testing, also known as functional testing, involves testing the software or system's functionality without knowledge of the internal workings. It involves testing the software or system's inputs, outputs, and interactions with other components to ensure that it performs as expected.

The need to specify detailed about these types of testing is crucial as it helps software testers and developers identify and rectify defects or errors at various stages of the software development lifecycle. It also ensures that the software or system meets the specified requirements and performs as expected, thereby improving the software or system's quality, reliability, and usability.

## 7.2 TEST CASES:

| S.NO | TEST CASE | DESCRIPTION | STATUS |
|---|---|---|---|
| 1 | Patient login | If the patient name and password is correct then it will be a valid page | pass |
| 2 | Doctor Register | If Doctor registration successfully then it is valid | pass |
| 3 | Admin login | If the admin name and password is correct then it will be a valid page | pass |
| 4 | Admin can activate the register doctors | Admin can activate the registered doctors id | pass |
| 5 | Admin can activate the register patients | Admin can activate the registered patients id | pass |
| 6 | SVM algorithm | We can implement prediction of SVM algorithm | pass |

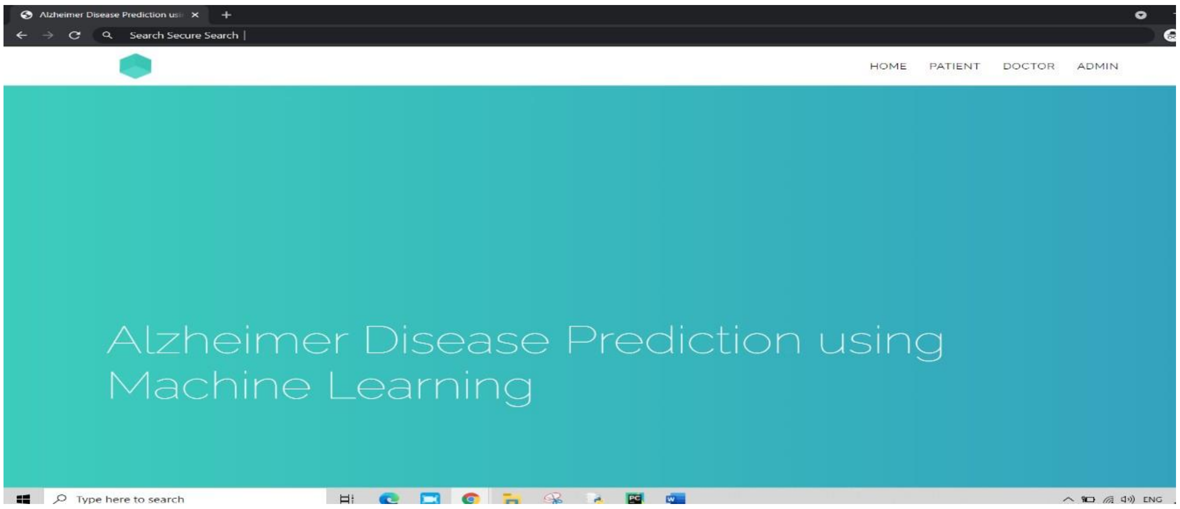
Table 2: Test Cases

# CHAPTER 8

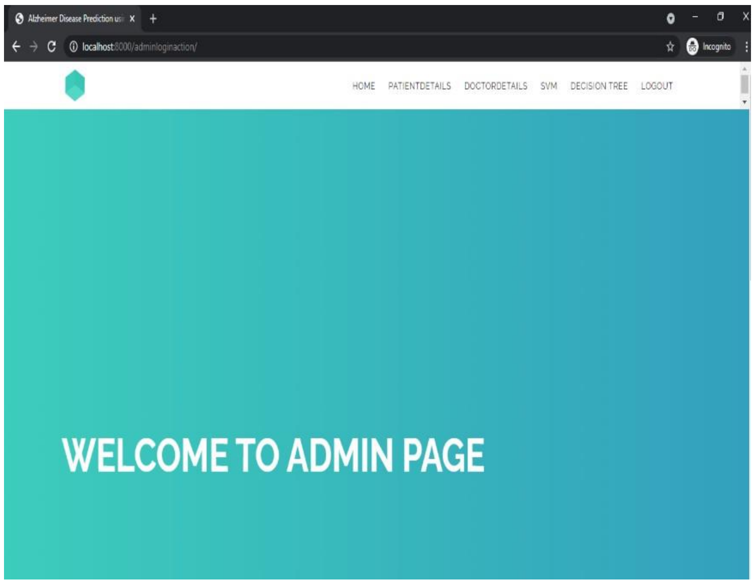
# OUTPUT SCREEN

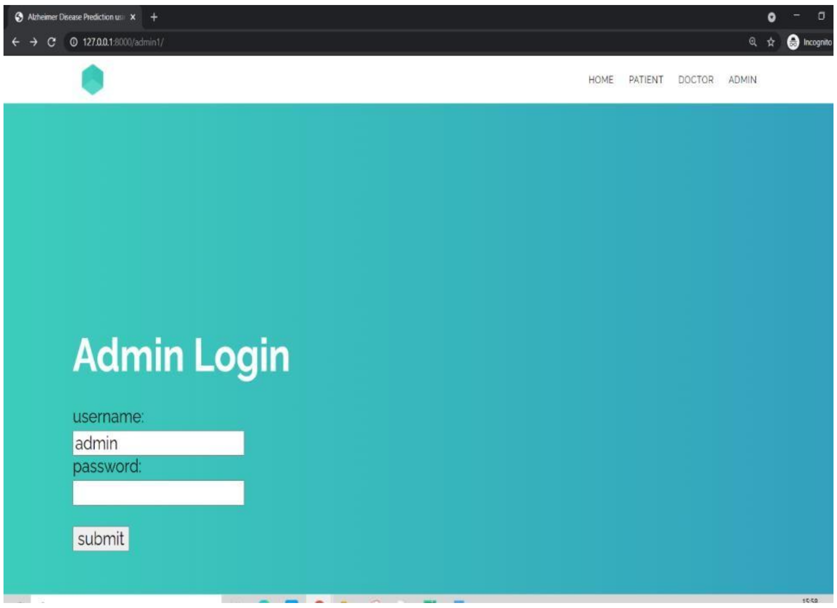
## 8.1SCREEN SHOTS:

### Home Page:


Alzheimer Disease Prediction usi
Search Secure Search
HOME
PATIENT
DOCTOR
ADMIN
Alzheimer Disease Prediction using Machine Learning
Type here to search

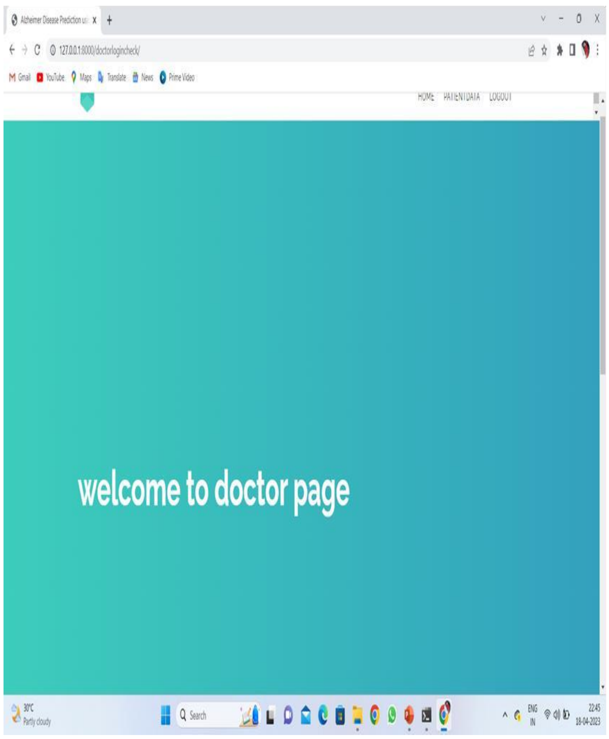
### Admin Login:



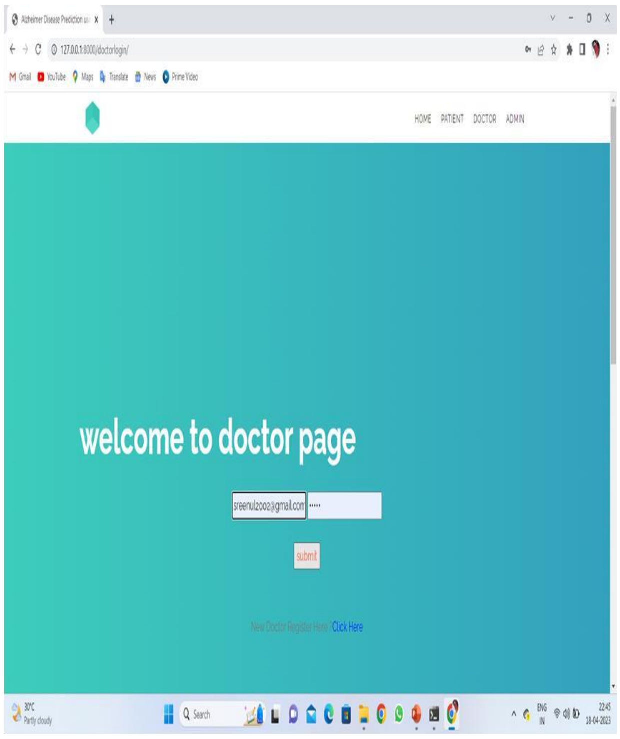


HOME
PATIENT
DOCTOR
ADMIN
Admin Login
username:
admin
password:
submit

**Doctor Page**:

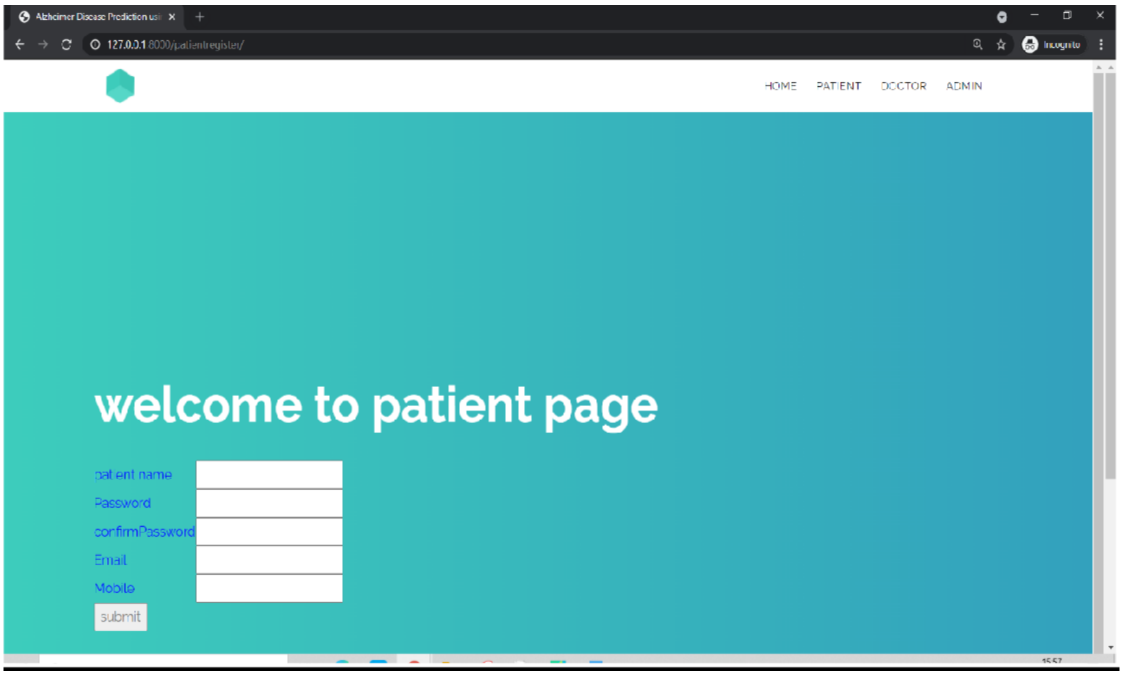


HOME
PATIENTDATA
LOGOUT
welcome to doctor page



HOME
PATIENT
DOCTOR
ADMIN
welcome to doctor page
sreenu12002@gmail.com
submit
New Doctor Register Here Click Here

## Patient Register:

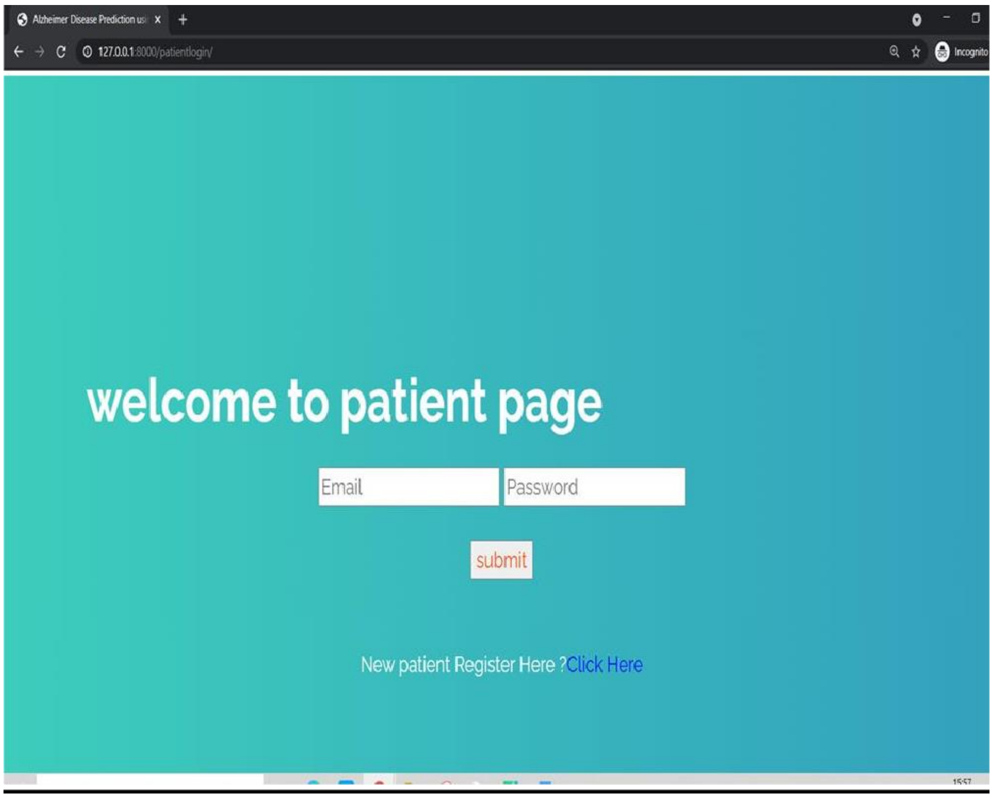

Alzheimer Disease Prediction usi
127.0.0.1:8000/patientregister/
Incognito
HOME
PATIENT
DOCTOR
ADMIN

welcome to patient page

| | |
|---|---|
| patient name | |
| Password | |
| confirmPassword | |
| Email | |
| Mobile | |

submit

## Patient Login:


Alzheimer Disease Prediction usi
127.0.0.1:8000/patientlogin/
Incognito

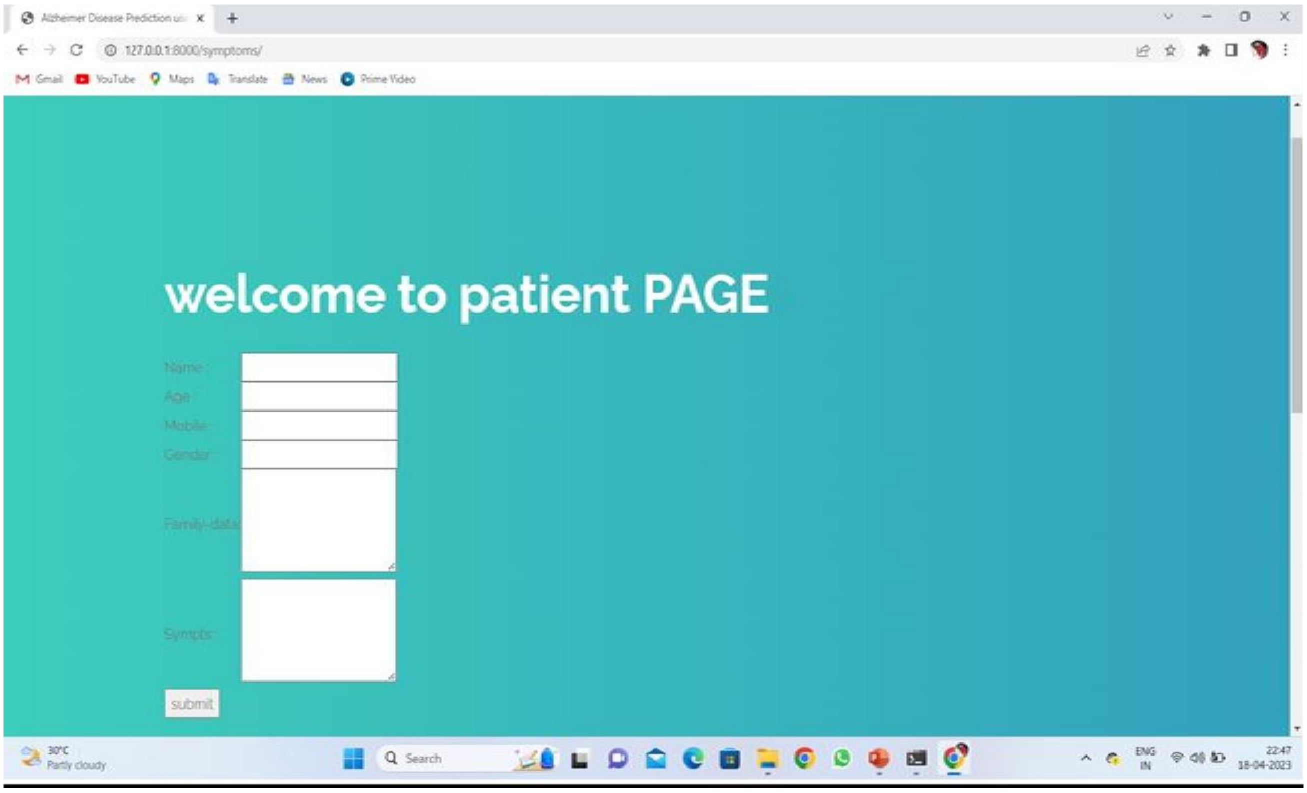
welcome to patient page

Email Password

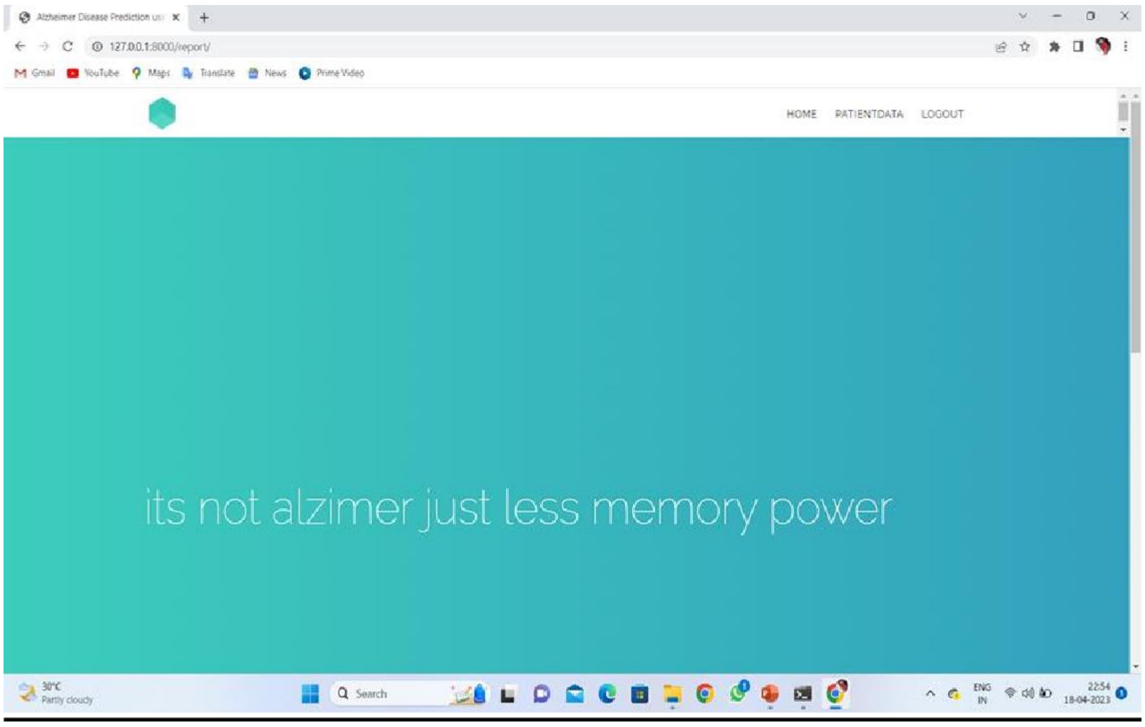
submit

New patient Register Here ?Click Here

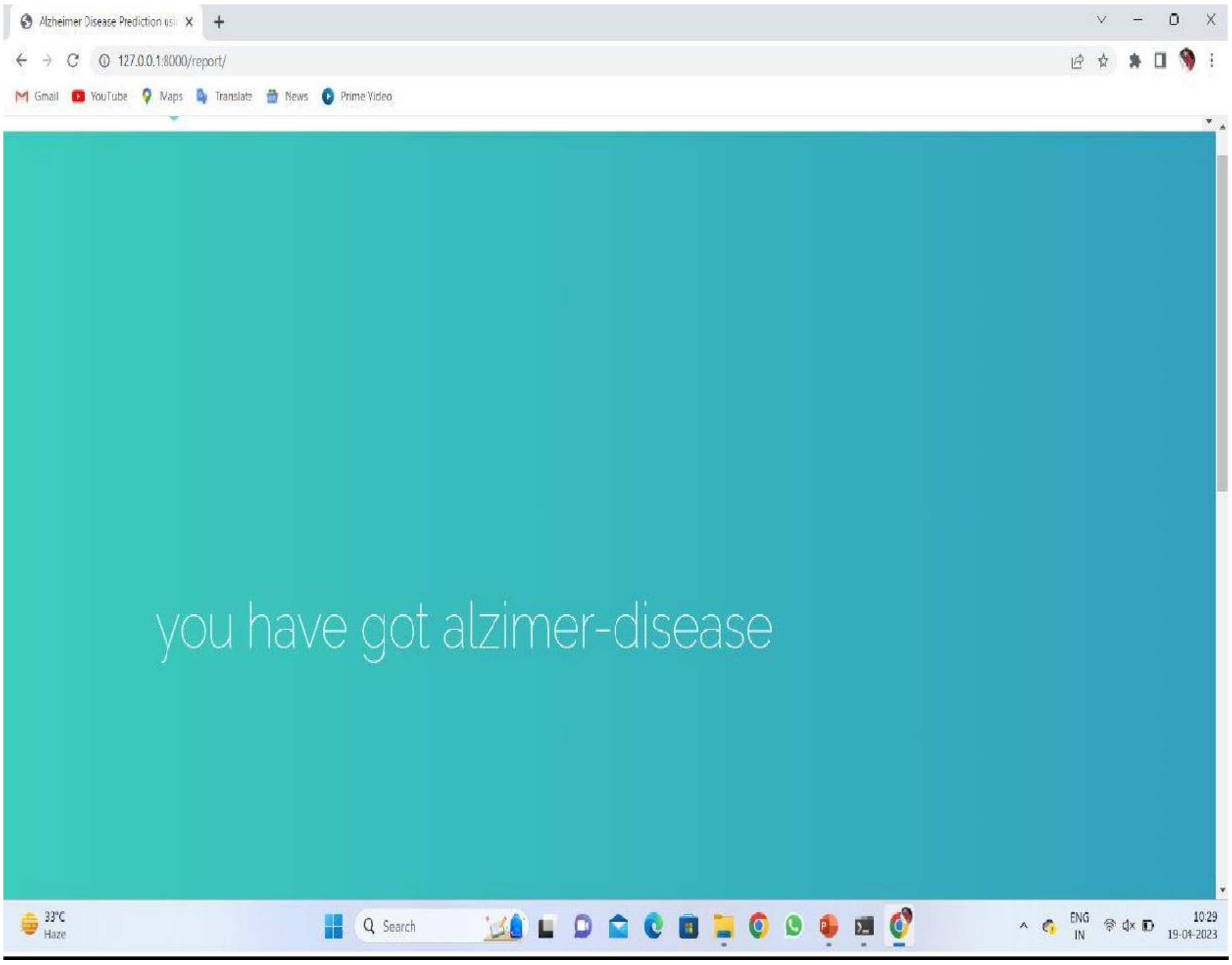
## Patient Page:



## Disease Identification:



its not alzimer just less memory power



you have got alzimer-disease

# CHAPTER 9

# CONCLUSION AND FUTURE SCOPE

## CONCLUSION:

- Machine learning approach to predict the Alzheimer disease using machine learning algorithms is successfully implemented and gives greater prediction accuracy results.
- The model predicts the disease in the patient.
- The future work can be done by combining both brain MRI scans and the psychological parameters to predict the disease with higher accuracy using machine learning algorithms.
- When they are combined, the disease could be predicted with a higher accuracy in the earlier stage itself.

## FUTURE SCOPE:

The future work can be done by combining both brain MRI scans and the psychological parameters to predict the disease with higher accuracy using machine learning algorithms. When they are combined, the disease could be predicted with a higher accuracy in the earlier stage itself.

# CHAPTER 10

# REFERENCES

[1] K.R.Kruthika, Rajeswari, H.D.Maheshappa, “Multistage classifier-based approach for Alzheimer’s Disease prediction and retrieval”, Informatics in Medicine Unlocked, 2019.

[2] Ronghui Ju , Chenhui Hu, Pan Zhou , and Quanzheng Li, “Early Diagnosis of Alzheimer’s Disease Based on Resting-State Brain Networks and Deep Learning”, IEEE/ACM transactions on computational biology and bioinformatics, vol. 16, no. 1, January/February 2019.

[3] Ruoxuan Cuia, Manhua Liu “RNN-based longitudinal analysis for diagnosis of Alzheimer’s disease”, Informatics in Medicine Unlocked, 2019.

[4] Fan Zhang , Zhenzhen Li , Boyan Zhang , Haishun Du , Binjie Wang , Xinhong Zhang, “Multi-modal deep learning model for auxiliary diagnosis of Alzheimer’s disease”, Neuro Computing, 2019.

[5] Chenjie Ge , Qixun Qu , Irene Yu-Hua Gu , Asgeir Store Jakola “Multi-stream multi-scale deep convolutional networks for Alzheimer’s disease detection using MR images”, Neuro Computing, 2019.

[6] Tesi, N., van der Lee, S.J., Hulsman, M., Jansen, I.E., Stringa, N., van Schoor, N. et al, “Centenarian controls increase variant effect sizes by an average twofold in an extreme case extreme control analysis of Alzheimer's disease”, Eur J Hum Genet. 2019;27:244–253

[7] J. Shi, X. Zheng, Y. Li, Q. Zhang, S. Ying, "Multimodal neuroimaging feature learning with multimodal stacked deep polynomial networks for diagnosis of Alzheimer’s disease", IEEE J. Biomed. Health Inform., vol. 22, no. 1, pp. 173- 183, Jan. 2018.

[8] M. Liu, J. Zhang, P.-T. Yap, D. Shen, "View aligned hypergraph learning for Alzheimer's disease diagnosis with incomplete multi-modality data", Med. Image Anal., 2017 vol. 36, pp. 123- 134.

[9] Hansson O, Seibyl J, Stomrud E, Zetterberg H, Trojanowski JQ,Bittner T, “CSF biomarkers of Alzheimer’s disease concord with amyloid-b PET and predict clinical progression: A study of fully automated immunoassays in Bio FINDER and ADNI cohorts”. Alzheimer’s Dement 2018;14:1470–81.

[10] Van der Lee SJ, Teunissen CE, Pool R, Shipley MJ, Teumer A,Chouraki V, “Circulating metabolites and general cognitive ability and dementia: Evidence from 11 cohort studies”, Alzheimer’s Dement2018;14:707–22