

1. INTRODUCTION

1.1 Problem Definition

Alzheimer's Disease (AD) is a condition which affects brain functions. AD disease results from deterioration of neurons which do different functions such as controlling movements, processing sensory information and making decisions. Many times neuronal death begins long before the patient will experience any symptoms. It can be months or years before any effect is felt. Symptoms are noticed after lots of cells have died and certain parts of brain have been weakened to the point that they can no longer function properly. Early diagnosis is very important since number of neurodegenerative diseases are reversible only when these are diagnosed at early stage and are crucial to be able to develop more powerful treatments. This early diagnosis is very difficult because the symptoms appear after severe damage to the cells. This project uses supervised Artificial Neural Network, networks in which the result of the processing (the output desired) is already defined. Supervised ANNs calculate an error function that measures the distance between the desired fixed output (target) and their own output, and adjust the connection strengths during the training process to minimize the result of the error function. The principle advantages of artificial neural networks are that they are able to generalize, adapting to signal distortion and noise without loss of robustness. They are trained by example and do not require precise description of patterns to be classified or criteria for classification. A MLP classifier is used to create the model. The dataset used is Minimal Interval Resonance Imaging in Alzheimer's Disease (MIRIAD) database.

1.2 About the Organization

RISS Technologies, Ernakulam, an ISO 9001:2008 certified company focuses on transforming and running business processes and operations including those that are complex and industry-specific. Their loom is distinctive: through an unbiased, agile combination of smarter process science, targeted technology and advanced analytics, they help their clients become more competitive by making their enterprises more intelligent: adaptive, innovative, globally effective, and connected to their own clients. They provide

end-to-end expert IT solutions across entire software delivery cycle. RISS cater to every single need of our clients and help them in accomplishing their goals and objectives. RISS has significant expertise accumulated over these 17 years of specialized work with hundreds of enterprises, and they remain loyal to their heritage of operational excellence as an extension of their clients' business – reflected by the best client satisfaction scores in the industry.

1.2.1 Services

- Software Developing
- Web Developing
- Mobile Application
- Web Hosting
- Digital Marketing
- Training

1.2.2 Strength

Riss Technologies have extensive business and training acumen and experience within the IT field. Their greatest strength is a team of aspiring professionals who believe in quality, commitment and competency. The management team consists of IT industry veterans and they are highly trained in latest technologies.

1.2.3 Vision

To see ourself India's most celebrated & 200- company status with unique value added contributions in information technology and services by 2020.

2. LITERATURE SURVEY

2.1 Initial Investigation

Alzheimer's Disease (AD) is a progressive brain disorder which affects the brain cells. It can lead to memory loss and other cognitive skills due to changes in the brain caused by disease or trauma. The changes can affect thinking, memory and reasoning, and may occur gradually or quickly. Early detection of AD is very important as AD is a disease which is irreversible.

The existing system to diagnose AD, doctors use manual techniques by conducting tests to assess memory impairment and other thinking skills, judge functional abilities, and identify behavior changes. They also perform a series of tests to rule out other possible causes of impairment.

2.2 Existing System

Alzheimer's disease (AD) is a progressive brain disorder and the most common cause of dementia in the late life. AD leads to the death of nerve cells and tissue loss throughout the brain, thus reducing the brain volume in size dramatically through time and affecting most of its functions. The estimated number of affected people will double for the next two decades, so that one out of 85 persons will have the AD by 2050.

Magnetic Resonance Imaging permits an extremely detailed image of the brain's structure. When one image is placed over another, taken a few months' later, it is possible to see changes at an early stage in a certain part of the brain.

CT (Computed Tomography) Scanning measures the thickness of a part of the brain which becomes rapidly thinner in people with Alzheimer's disease.

SPECT (Single Photon Emission Computed Tomography) Scanning can be used to measure the flow of blood in the brain, which has been found to be reduced in people with Alzheimer's disease as a consequence of nerve cells not working properly.

PET (Positron Emission Tomography). The use of this scanning technique is often limited to research settings. It can detect changes in the way the brain of someone with Alzheimer's disease functions. It can, for example, detect abnormal patterns of glucose usage by the brain.

All the above scan images are processed by doctors by their own eyes manually. There is no automatic detection of disease detection.

2.3 Proposed System

The proposed system uses data mining techniques and is implemented with Artificial Neural Networks for Alzheimer's Disease Diagnostics. A patient's scan image is fed to the neural network. The image is firstly preprocessed by resizing, normalizing and reshaping the scan image. The steps of knowledge discovery of data process are implemented. The data is preprocessed and fed to the neural network for training it. The trained neural network is used to predict the Alzheimer's Disease percentage for a patient's scan image.

2.4 Feasibility Study

Feasibility study is a test of a system proposal according to its workability, impact on the organization, ability to meet users need and effective use of resources. The main aim of the feasibility study is to determine whether it would be financially and technically feasible to develop the product. The feasibility study was divided into four: Technical, Economical, Operational and Behavioral. It is summarized below.

2.4.1 Technical Feasibility

Currently Alzheimer's Disease is detected by doctor's manually by conducting tests to assess memory impairment and other thinking skills, judge functional abilities, and identify behavior changes. They also perform a series of tests to rule out other possible causes of impairment. Now we are offering a solution which is completely web based so

that they can use the system for early diagnostics of Alzheimer's Disease. Hence we can say that proposed system is technologically feasible. That means it is easy to handle by the user.

2.4.2 Economic Feasibility

The economic feasibility is the most important and frequently used method for evaluating the effectiveness of the proposed system. It is very essential because the main goal of the proposed system is to have economically better result along with increased efficiency. The doctor detect Alzheimer's Disease manually by conducting a series of tests and by examining the scan report of the person which is timing consuming and is not economically feasible. We are providing a web application with a provision for prediction which will help them to easily detect Alzheimer's Disease.

2.4.3 Operational Feasibility

The system operation is the longest phase in the development life cycle of a system. Therefore, operational feasibility is important. The users of the system do not need thorough training on the system. The "ALZHEIMER'S DISEASE DIAGNOSTICS" only expected to know how to operate the system, and needs only the basic net surfing knowledge. It has a user-friendly interface.

2.4.4 Behavioral Feasibility

In today's world, where computer is an inevitable entity, the system like searching site, which requires no special efforts than surfing the net are enjoying wide acceptance. Thus, the organization is convinced that the system is feasible.

3. SYSTEM ANALYSIS AND DESIGN

3.1 System Requirement Specification

Product Scope

The project is specifically designed for the prediction of Alzheimer's Disease. The main operations are doctor and patient management, train data, prediction and accuracy of the data. The main tasks like adding and modifying doctor details and patient details, prediction of Alzheimer's Disease can be easily handled by using this product. The Admin has the privilege to manage the doctor and patient details and also to view the model accuracy. Doctor has the privilege to view a patient and to handle patient's record, upload the Scan image of patient and to predict the AD percentage. A train data handles the data that is used to generate a model which is used in prediction. Data is trained by using Artificial Neural Network. The input data that is given for training is a set of MRI scan images available in Minimal Interval Resonance Imaging in Alzheimer's Disease (MIRIAD) dataset.

Product Features

The main operations that are performing in "Alzheimer's Disease Diagnostics" are doctor and patient management, train data, prediction and accuracy of the data. The main tasks like adding and modifying doctor details and patient details, prediction of Alzheimer's Disease can be easily handled by using this product.

The main tasks includes adding and modifying doctor details and patient details, model accuracy, view patient details, train data, prediction of AD.

- **Admin Module**

Admin handles one of the module in "Alzheimer's Disease Diagnostics". It controls operations like doctor management patient management, say adding and modifying doctor and patient details, train data. Admin also has the privilege to determine the score of the trained data.

Doctor Management

Doctor Management handles the doctor details and has the privilege to register, update, delete and view the details of a particular doctor. It include several details of the doctor such as name, personal details, enrollment number, qualification, area of specialization and so on.

Patient Management

Patient Management handles the patient details and has the privilege to register, update and delete a particular patient. It include several details of the patient such as name, personal details and so on.

Model Accuracy

Accuracy of data determine the score of the data after the data is trained. It is specified in terms of percentage.

- **Doctor Module**

Doctor handles one of the important module in "Alzheimer's Disease Diagnostics". It controls operations like view patient details, upload MRI scan and view report. Doctor also has the privilege to predict the value.

View Patient Details

Doctor can view the details of all the patients and search for a particular patient details.

Prediction

Prediction is handled by doctor on the basis of the scan report of the patient. The scan image is preprocessed i.e; the image will be resized, normalized and reshaped. After preprocessing the image is fed into the neural network which will be trained and the result will be obtained and the doctor can generate the report.

3.1.1 Functional Requirements

Hardware and software requirements for the installation and smooth functioning of this project could be configured based on the requirements needed by the component of the operating environment that works as front-end system here we suggest minimum configuration for the both hardware and software components.

Working off with this software is requirements concrete on system environments. It includes two phases

➤ Hardware Requirements:

- Input Device : Mouse, Keyboard
- Output Device : Monitor
- Memory : 8GB RAM
- Processor : intel core i7

➤ Software Requirements:

- Operating System : Windows 10
- Front End : PYTHON FLASK Framework
- Back End : MYSQL
- Software Used : PyCharm

3.1.2 Non-Functional Requirements

- Performance Requirements

For the efficient performance of the application, network must have high bandwidth so that the task of centralized management does not lead to network jam. Also the hard disk capability must be high so that data can be effectively stored and retrieved.

- Security Requirement

Security requirements of this application involves user authentication using user name and password so that invalid users are restricted from data access. For the security of

data, periodic database backups must be performed so that we can recover data in the case of data loss.

3.2 Unified Modeling Language (UML)

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development. UML stands for Unified Modeling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

- Improved integration between structural models like class diagrams and behaviour models like activity diagrams.
- Added the ability to define a hierarchy and decompose a software system into components and sub-components.
- The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed state chart diagrams to state machine diagrams, also known as state diagrams.

Types of UML Diagrams

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing, and deployment. These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

Structural UML diagrams

- Class diagram
- Package diagram
- Object diagram
- Component diagram
- Composite structure diagram
- Deployment diagram

Behavioral UML diagrams

- Activity diagram
- Sequence diagram
- Use case diagram
- State diagram
- Communication diagram
- Interaction overview diagram
- Timing diagram

Use Case Diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system. In software and systems engineering, a use case is a list of actions or event steps, typically defining the interactions between a role and a system, to achieve a goal. The actor can be a human or other external system. In this system, Admin, Branch Staff, Account Staff are the Actors. They are represented as follows,



: Usecase

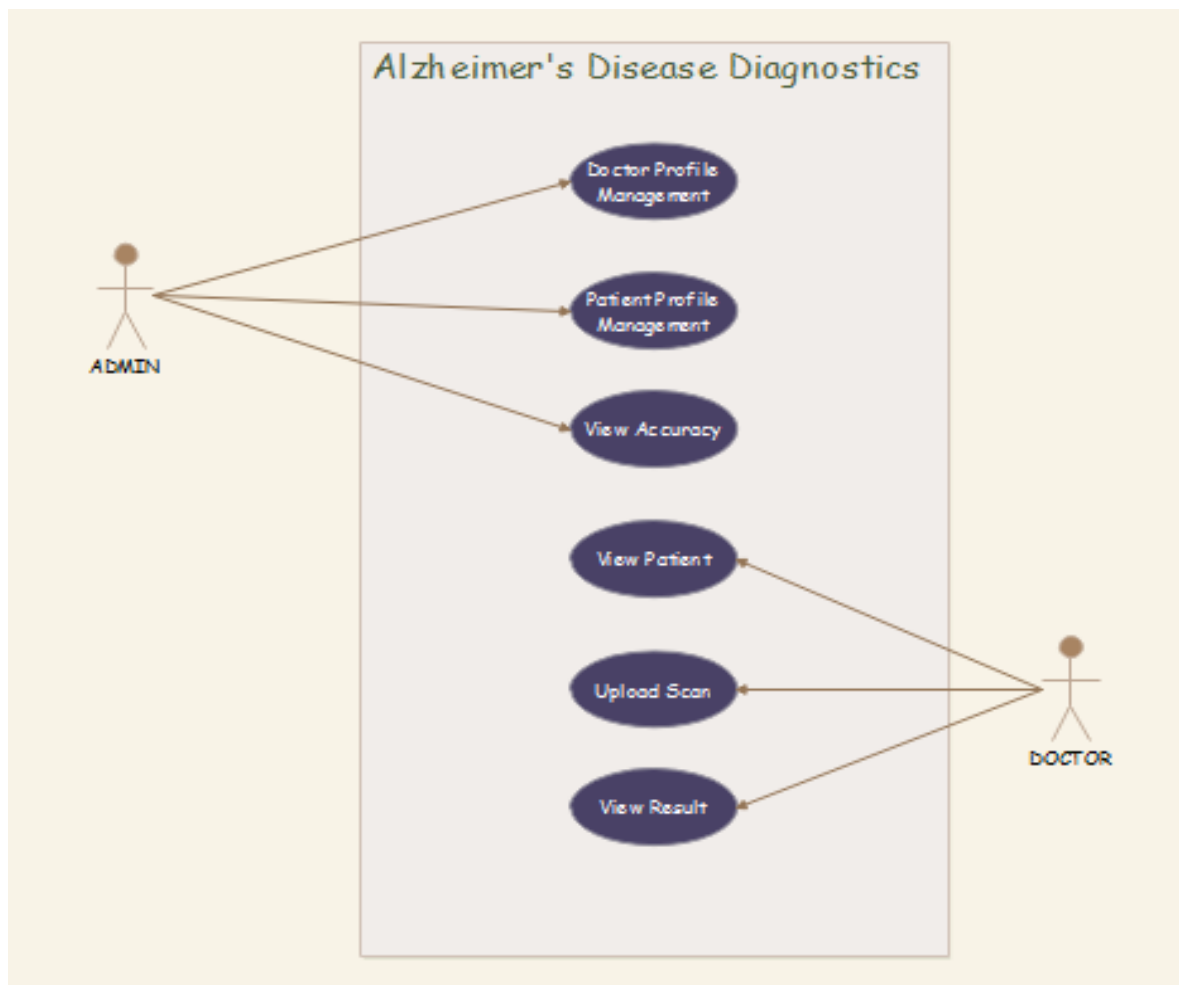


Figure 1: UML Use case

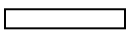
Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system, modelling its process aspects. Often they are a preliminary steps used to create an overview of the system which can later be elaborated. DFD can also be used for the visualization of data processing (Structured design).


A Data Flow Diagram, also known as “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. A DFD consists of a series of bubbles joined by lines. The bubbles represent data transformations and the lines represent data flows in the system.

Symbols used in DFD


1. External entity

A rectangle () indicates any entity external to the system being modelled. The function of external entity is to, supply data to, or receive data from the system.


2. Process

Bubbles or circles () are used to indicate where incoming data flows are processed and then transformed into outgoing data flows.

3. Dataflow

Arrows () marking the movement of data through the system indicate data flows. It's the pipeline carrying packet of data from an identified point of origin into a specific destination.

4. Data store

Open rectangles () are used to identify holding points for data. A data store denotes data in rest.

CONTEXT LEVEL DIAGRAM

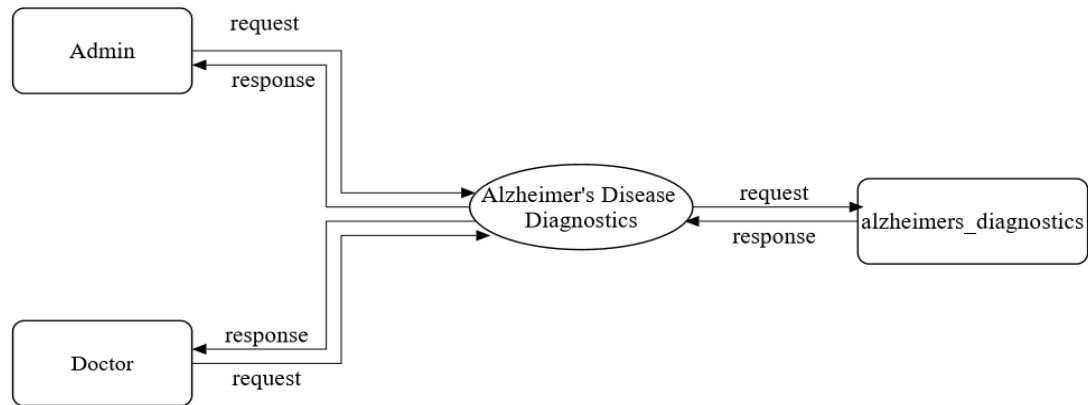


Figure 2: Level 0 DFD

LEVEL 1 ADMIN

LEVEL 1 - ADMIN

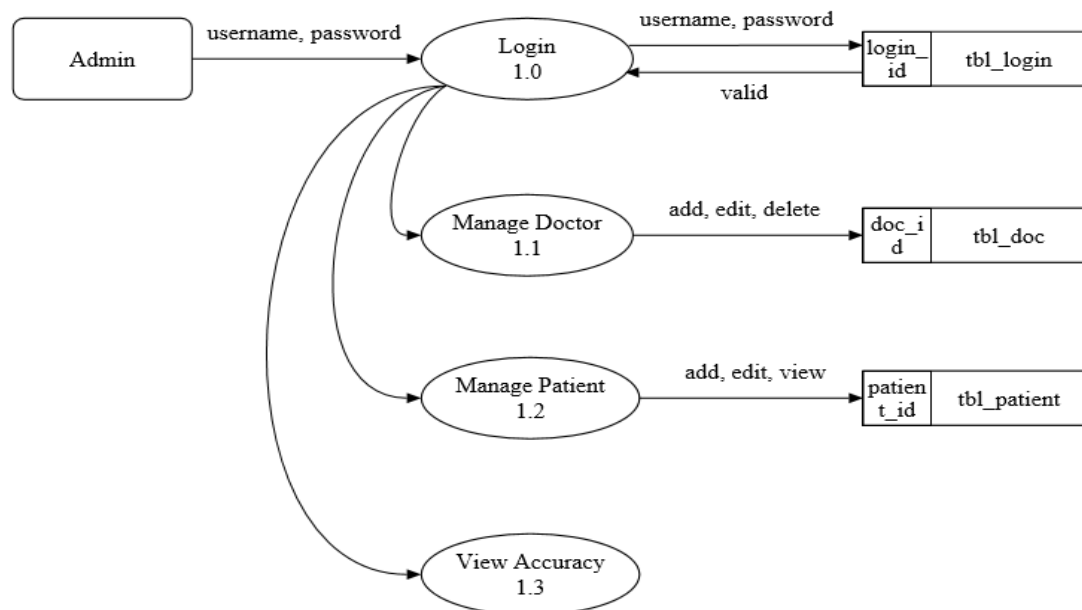


Figure 3: Level 1 Admin

LEVEL 1 DOCTOR

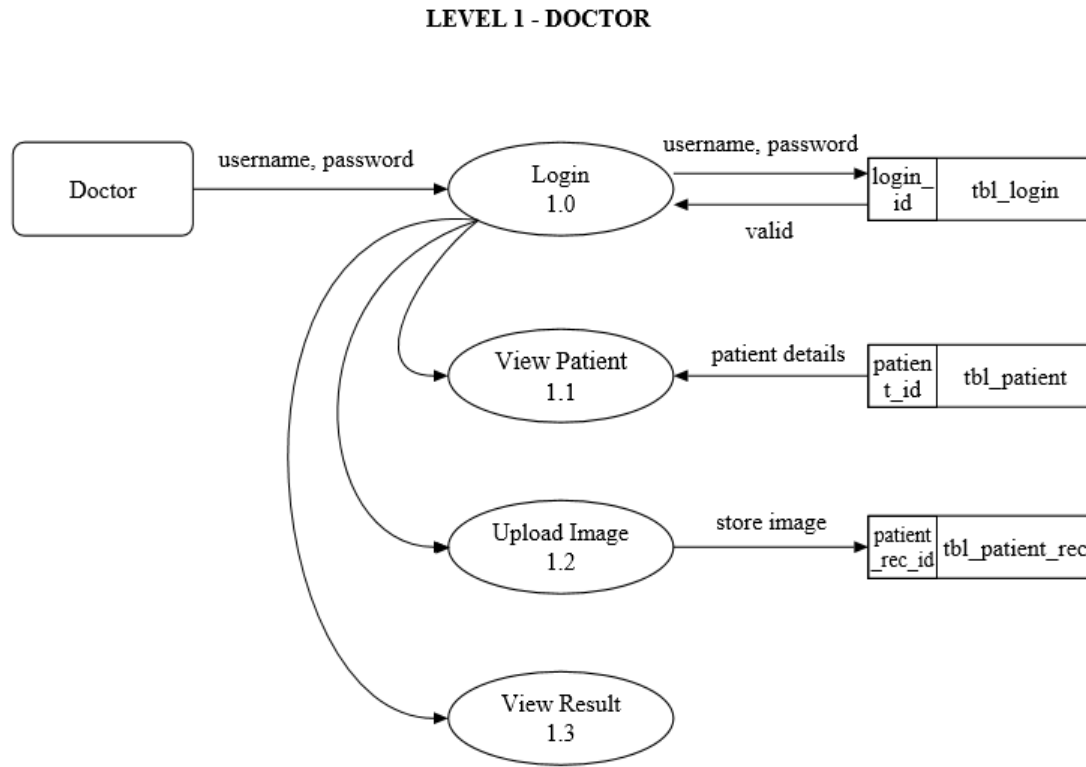


Figure 4: Level 1 Doctor

3.3 System Design

The most creative and challenging phase of the system lifecycle is the system design. The term design describes a final system and the process by which it is developed. It refers to the technical specifications that will be applied in implementing the candidate system. In system design, we move from the logical to the physical aspects of the life cycle.

The first step is to determine how the output is to be produced and in what format. Then input data and master files have to be designed at the next step and finally the impact of the candidate system on the user and organization are documented and evaluated by the management. After identifying the problem and the limitations of the existing system, a detailed design of the proposed system is conducted.

Free flow personnel interview and reference to previous records prepared manually were the only methods taken to collect necessary information. At present, the all organizations are on the path of computerization process.

Design is the phase that indicates the final system. It is the solution, the translation of requirements into ways of meeting them. In this phase the following elements were designed namely, dataflow, data stores, processes, procedures. Firstly, the logical design was done where the outputs, inputs, databases and the procedures was formulated in a manner that meet the project requirements. After logical design physical construction of the system is done.

The database tables, input screens, output screens, output reports are designed. After analyzing the various functions involved in the system the database, labels a dictionaries designed. Care is taken for the field name to be in self-explanatory form. Unnecessary fields are avoiding so as not affecting the storage of the system.

Care must be taken to design the input screen in the most user-friendly way so as to help even the novice users make entries approximately in the right place. This is being accomplished by the use of giving online help messages, which are brief and cleanly prompts users for the appropriate action.

Design is the only way that we can accurately translate a customer's requirements into a finished software product or system. Without design, risk of building an unstable system exist one that will fail when small changes are made, one that will be difficult to test.

All input screens in the system are user friendly and are designed in such a way that even a layman can operate. The sizes of all the screens are standardized. Reports generate in this software give the finer accepts of the required information, which helps in taking vital decisions.

The importance of the software design can be stated with a single word quality. Design is a place where quality is fostered in software development. Design is the only way where requirements are actually translated into a finished software product or system.

There are two users in the system

- Admin
- Doctor

Admin:

Administrator is one of the user of the system. He/she have the privilege to the overall system. Administrators are exempted from registration.

Doctor:

Doctor is the super user of the system. Doctor has the full authority for the prediction of the Alzheimer's disease.

3.3.1 Input Design

Input design is the process of converting user-oriented input into a computer based format. The goal of the designing input is to make data entry as easy and free from error. In Python, input to the system is entered through forms. A form is “any surface on which information is to be entered, the nature of which is determined by what is already on that surface.” If the data going into the system is incorrect, then processing and output will magnify these errors. So designer should ensure that form is accessible and understandable by the user. End users are people who communicate to the system frequently through the user-interface, the design of the input screen should be according to their recommendations. The following are the consideration given by the end-users for input design.

1. The screens should be user-friendly and easy to operate.
2. Proper validation of inputs to be provided.
3. The screens should be clear and enough information should be provided to guide the user to enter correct data.
4. List of valid values for the field should be provided wherever possible. An analyst must always assume that errors will occur. They must be found during input and corrected prior to storing or processing data.

In this project titled “Alzheimer’s Disease Diagnostics” initially as the application is invoked a login screen is shown which a security feature is built into the system since all the information stored is confidential and hence must be only to the administrator and doctor. The design decisions for handling input specify how data are accepted for computer processing. The design of inputs also includes specifying the means by which end-user and system operators direct the system in which the action to take. The goal of the input design is to make the data entry easier, logical and error free. Errors in the input data are controlled by input design. Complex name, figures etc. are avoided to make it user friendly. Security is provided in necessary areas.

The application has been developed in a user friendly manner. The system accepts the needs from the user with simple and understandable dialogs. The screens have been designed in such a way that during the processing the curser is placed in the position where the data must be entered.

The input screens of the design are designed with standard layouts, colors and with appropriate controls like option buttons, check boxes etc. for making the data entry process easy, error free, fast with less strain. Data validation methods are used to check the data entered and to display appropriate error messages if any errors are found.

3.3.2 Output Design

The most important thing about any system is what it produces. A system is judged to be a success or failure depending on whether its products are useful or not. So it is critical that we first specify what is required from the system. Once this had been done, we can concentrate on what is required from the system. Once this has been done we can concentrate on what is required to produce this output. In order to agree what results are to be produced by the system users are consulted to understand exactly what is required.

On decisions which need to be made, is which medium to use for a particular output.

The main media available are:

1. Print used for reports and for a permanent listing of the file contents.
2. Disk used for storing data files.

Other factors to be considered when we are designing output are usage, quality and cost. These factors are closely related and we normally seek a compromise involving all three. For instance, higher quality generally costs more and a document to be used by the public needs to be better quality than one used within an organization.

Thus where output is sent can be divided into two broad groups internal and external. Internal usage refers to use by employees within organization, whereas external output is desired for people outside the organization.

The main requirement of an internal document is that it contains the necessary information for it to be useful. There is no need for fancy document or top quality printing or very high quality paper.

As long as the information is presented in a readable format the most important criterion has been satisfied. External documents, on the other hand, can play an important role in determining the public image of the organization. Thus the emphasis here is a presentation as well as usefulness. Later quality appearance etc are given prime importance here.

Unfamiliar people use the external documents. So the terminology used must be simple. The higher the level of the employee, the lesser the details required in the report. How often a given report is needed or referenced can also influence its design. Some reports must be produced daily while others are less frequently required, in certain cases reports may be legal requirements. Sometime previous year report may be required so an appropriate output medium is selected for storing such reports.

3.3.3 Database Design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a Data Definition Language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

A database is an integrated collection of data and provides centralized access to the data. Usually the centralized data managing software is called RDMS. The main significant difference between RDMS and other DBMS is the separation of data as seen by the program and data as stored in direct access to storage device. This is the difference between logical and physical data.

Normalization

Normalization is a process of reducing redundancies of data in a database. It is technique that is used when designing and redesigning a database. The first step consists of transforming the data items into two-dimensional tables. Then remove the repeating occurrence of data items so that a flat file is obtained.

- **First Normal Form**

At the intersection of row and column, there is only one value in the tuple. Grouping of values are not allowed. A database is said to be 1 NF, if all the relations of the database are in 1NF.

- **Second Normal Form**

The relation is said to be in second normal form, every non-key attribute is fully functionally dependent on primary key.

- **Third Normal Form**

A relation is said to be in third normal form, there exist no transitive functional dependency between non-key attributes.

Database Tables

The efficiency of an application using SQL Server is mainly dependent upon the database tables, the fields in each table and joined using the fields contained in them to retrieve the necessary information. A table is a set of data elements that is organized using a model of vertical columns and horizontal rows. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a unique key index.

The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must:

- Determine the relationships between the different data elements.
- Superimpose a logical structure upon the data on the basis of these relationships.

Design Process

1. Determine the purpose of the Database: This helps prepare for the remaining steps.
2. Find and organize the information required: Gather all of the types of information to record in the database, such as product name and order number.
3. Divide Information into tables: Divide information items into major entities or subjects, such as Products or Orders. Each subject then becomes a table.
4. Turn information items into columns: Decide what information needs to be stored in each table. Each item becomes a field, and is displayed as a column in the table.
5. Specify the primary key: Choose each table's primary key. The primary key is a column, or a set of columns, that is used to uniquely identify each row. An example might be Birth ID.
6. Setup the table relationships: Look at each table and decide how the data in one table is related to the data in other tables. Add fields to tables or create new tables to clarify the relationships, as necessary.

7. Refine the design: Analyze the design for errors. Create tables and add a few records of sample data. Check if results come from the tables as expected. Make adjustments to the design, as needed.
8. Apply the normalization rules: Apply the data normalization rules to see if tables are structured correctly. Make adjustments to the table.

3.4 Tools and Platform

PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. It provides automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

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.

FLASK

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more regularly than the core Flask program.

MySQL

MySQL is the most popular Open Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various web-based software applications. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds. Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory but data fetching and writing would not be so fast and easy with those types of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because, all the data is

stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

A Relational DataBase Management System (RDBMS) is a software that –

- Enables you to implement a database with tables, columns and indexes.
- Guarantees the Referential Integrity between rows of various tables.
- Updates the indexes automatically.
- Interprets an SQL query and combines information from various tables.

RDBMS Terminology:

Before we proceed to explain the MySQL database system, let us revise a few definitions related to the database.

- Database – A database is a collection of tables, with related data.
- Table – A table is a matrix with data. A table in a database looks like a simple spreadsheet.
- Column – One column (data element) contains data of one and the same kind, for example the column postcode.
- Row – A row (= tuple, entry or record) is a group of related data, for example the data of one subscription.
- Redundancy – Storing data twice, redundantly to make the system faster.
- Primary Key – A primary key is unique. A key value can not occur twice in one table. With a key, you can only find one row.
- Foreign Key – A foreign key is the linking pin between two tables.
- Compound Key – A compound key (composite key) is a key that consists of multiple columns, because one column is not sufficiently unique.
- Index – An index in a database resembles an index at the back of a book.

- Referential Integrity – Referential Integrity makes sure that a foreign key value always points to an existing row.

MySQL Database

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons –

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

4. SYSTEM TESTING

Testing enhances the integrity of a system by identifying deviations in design and development of the expected end product. It should focus more on the error-prone areas of the application. This help in the prevention of errors in a system and builds confidence that the system will work without error after testing. It is the process of executing a program with the intent of finding an error. Testing also adds value to the product by conforming to the user requirements. Testing verifies that software deliverable conforms precisely and design phases.

Testing involves a series of operation of a system of application under controlled conditions and subsequently evaluating the result. The controlled condition should include both normal and abnormal conditions. It is planned and monitor for each testing level (e.g., unit, integration, system and acceptance). Testing is the major quality measure employed during software development. After the coding phase computer programs can be executed for testing purpose. Testing phase not only aim to uncover errors occurred during coding, but also locates error committed during the previous phase. Thus the aim of testing is to uncover requirements, design or coding errors in the programs .System testing of software is conducted on a complete, integrated system to evaluate the system's compliances with its specified requirements. System testing falls within the scope of black box testing, and should require no knowledge of the inner design of the code. As a rule, testing takes its input, all of the integrated software components that have successfully passed integration testing.

Testing is a process of executing a program with the interest of finding an error. A good test is one that has high probability of finding the undiscovered errors. The primary objective for the test case design is to drive a set of tests that has the highest likelihood for systematically uncovering different classes of errors in the software. A series of testing are performed for this project before the system is ready for acceptance.

4.1 Testing Methods

Testing is the phase where the bug in the programs was to be found and correct that bugs. One of the goals during dynamic testing is to produce a test suite. This is applied to ensure that the modification of the program does not have any side effects. This type of testing is called regression testing. Testing generally removes all the residual bugs and improves the reliability of the program.

The overall strategy for testing “Alzheimer’s Disease Diagnostics” application is described as follows.

- Unit Testing
- Integration Testing
- Validation Testing
- Output testing
- User Acceptance Testing

4.1.1 Unit Testing

Unit test comprises the set of tests performed by an individual programmer prior to integration of unit to large systems. Unit testing is done to testing the modules (classes) one by one in order to make sure that it works without any errors before it was put together with other modules. The tests are very simple. Each and every screen was put into testing by giving random values as input. Breakage testing was done with the boundary conditions too. The modules were checked to see that the methods return the expected result and that the classes handle the wrong input in a correct way. For example, error messages whenever needed and can handle exceptions effectively. After coding each dialogue is tested and run individually. All unnecessary coding were removed and it was ensured that all the modules worked, as the programmer would expect.

Logical errors were corrected. So, by working all the modules independently and verifying the output of each module was functioning as expected. This testing focuses on each module and individual software unit ensuring that they work properly. Unit testing checks for the changes made in the new system or any program in it. Unit testing includes white box testing. This is the first level of testing. In this different modules are tested

against the specification produced during the design of the modules. Unit testing is done for the verification of the code produced during the coding of single program modules in an isolated environment. Unit testing first focuses on the modules independently of one another to locate errors.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits; tests find problems early in the development cycle. This testing allows the programmer to refactor code at a later date, and make sure the module still works correctly. The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified and fixed.

In this project all the modules are tested. In our system database is connected using link and entity models, the main problem occurred in our system for database connection is whenever an entity model is created then if any further alteration in the table, we need to update the entity model, if we don't change the model then an updating error will be occurred. Another problem that occurred in our system are, this system will help to group students participating in the fest, students that are selected for a group have to be removed from the list, otherwise each group consist of same members, unit testing will help to identify and rectify this problem, So that each group consist of distinct members, So the unit testing is an important testing method.

4.1.2 Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. The purpose of integration testing is to verify functional, performance, and reliability.

Requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using Black box testing, success

and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Data can be lost access an interface, one module can have as adverse effort on another sub function when combined, may not produce the desired major functions. Integration testing is a systematic testing for constructing the program structure, while at the same time conducting tests to uncover errors associated within the interface. Here correction is difficult because the vast expenses of the entire program complicate the isolation of causes. Thus in the integration testing step all the errors uncovered are corrected for the next testing steps.

Some different types of integration testing are big bang, top-down, and bottom-up.

- **Big Bang**

In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing. A type of Big Bang Integration testing is called Usage Model testing. Usage Model testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use. Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components.

The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the

developers, and instead flesh out problems caused by the interaction of the components in the environment. For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives that the integrated environment will work as expected for the target customers.

- **Top –Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breathe first manner.

- **Bottom-Up Integration**

This method begins construction and testing with the module at the lowest level in the program structure. Since the module are integrated from the bottom up, processing required for module subordinate to a given level is always available and the need for the stubs is eliminated. Our system will work without any defects if it work independently but it incurred some errors when it combine all the modules together. If we edit any changes to event or student details then it won't display those changes until we have to stop and rerun the system again.

4.1.3 Validation Testing

It provides the final assurance that the software meets all the functional, behavioral and performance requirements. The software is completely assembled as a package. Validation succeeds when the software functions in a manner in which the user expects. Validation refers to the process of using software in a live environment in order to find errors.

System validation checks the quality of the software in both simulation and live put a lot of validation testing before finally implementing it. Thus the feedback from the validation phase generally produces changes in the software. The system objective, the

functional performance, requirements were looked into to see whether all these criteria are satisfying the system needs. The system is then presented before the manager along with the reports generated. The system then undergoes a testing phase with the sample test data provided by him. System testing in this manner would verify that all the modules work together and generate the intended results. All individual modules should be working in tandem so that the overall system function or performance is achieved.

Validation process is done in validation testing. If the username or password in the login is not correct, the validations such as, password is not valid or username is not valid etc. are occurred. Also in the registration process the validations can occur, like if the e-mail id is not in the correct format, it will display an error in that page.

4.1.4 Output Testing

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output generated or considered in two ways: one is on screen and another is printed format. The output format on the screen is found to be correct as the format was designed in the system design phase according to the user needs. For the hard copy also the output comes out as the specified requirements by the user. Hence output testing does not result in any correction in the system.

4.1.5 User Acceptance Testing

User acceptance testing is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing and making changes whenever required. This is done with regard to the following points. Input screen design, Output screen design and menu driven system.

It is formal testing conducted to determine whether or not the system satisfies its acceptance criteria to enable the customer to determine whether or not to accept the system. User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes

wherever required. Preparation of data plays a valid role in the system testing after preparing the test data the system under study is tested using the test data. While testing the system by using test data errors are again uncovered and corrected and the corrections are also noted for future use.

4.1.6 System Testing

System testing means testing all the different parts of the program together and sees how it will react under certain conditions. Some areas should test for are recovery and stress. When testing the all system there are few questions to be answered.

- How will you setup the system for testing?
- How do you perform the testing?
- What is the expected result?

4.1.7 Black box testing

The black box testing is a testing method in which test data are derived from the specified functional requirements without regard the final program structure. Because the functionality of the software modules is on concern, black box testing. Testing is mainly refers to functional testing method emphasized on executing the functions and examination of their input and output data .The tester treats the software under test as a black box only the input, output and specification are visible and the functionality is determined by observing the outputs to corresponding input .in testing various input are exercised and the outputs are compared against specification to validate the correctness. All test cases are derived from the specification. No implementation details of the code are considered. Here the functions for checking all the input values are written.

While user is logged and changing the password, entering or editing user information in the member profile, and member is added , if user forget to give any values then appropriate message are displayed or that errors. Testing performed on a complete, integrated system to verify that the system is compliant with its specifications and requirements. System testing is normally done before fully implementing if for the users. The testing done before implementing the system would help the developer to perform

any further operation on the system. Our developed system will be run in our college server by giving real time data and verify that the system is done without any errors.

4.2 Test Cases

Test Case ID	Test Objective	Precondition	Steps/Cases	Test Data	Expected Result
T1	Successful Login to Admin Home	1. There should be a valid user name and password for the Admin	1. Enter username and password 2. Click login button	Valid username and password	Logged in Successfully
T2	Unsuccessful Login to Admin Home	1. An Invalid User Account to Login	1. Enter username and password 2. Click login button	Invalid username and password	Login Unsuccessful
T3	Successful Login to Doctor Home	1. There Should be a valid user name and password for the Doctor	1. Enter username and password 2. Click login	Valid username and password	Logged in Successfully

			button		
T4	Unsuccessful Login to Doctor Home	1.An Invalid User Account to Login	1.Enter username and password 2.Click login button	Invalid username and password	Login Unsuccessful
T5	Edit Doctor Profile	Successful Login into The System	1.Go to Home. 2.Edit the details 3.Click 'Submit' button	Details of Doctor	Details saved successfully
T6	Edit Patient Profile	Successful Login into The System	1.Go to Home. 2.Edit the details 3.Click 'Submit' button	Details of Patient	Details saved successfully
T7	Accuracy	Successful Login into The System	1.Click on Model	Accuracy Calculated	Accuracy Score displayed

			Details Link		successfully
T8	Prediction	Successful Login into The System	1.Click on View Report Link	Prediction Details	Prediction Details displayed successfully

5. SYSTEM IMPLEMENTATION

Implementation is the stage of the project where the theoretical design is turned into a working system. The implementation of a system involves people from different sections and systems analysts are confronted with the practical problem of controlling activities of people outside their own data processing sections. Prior to this point in the project, system analysts have interviewed staff's different sections with the permission of their respective managers but the implementation phase involves staff of user departments carrying out specific task, which require supervision and control to critical schedules. Some of these problems may also need to be employed on the task of implementation in addition to their own present departmental tasks. Implementation is the stage of the project where the theoretical design is turned into a working system. At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned and controlled, it can cause chaos and confusion. Implementation includes all those activities that take place to convert from the old system to new system. The new system may be totally new, replacing an existing manual or automated system or it may be a major modification to an existing system. At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned and controlled, it can cause chaos and confusion. Depending upon the nature of the system, extensive user training may be required. Programming itself is a design work.

The initial parameters of the management information system should be modified as a result of programming efforts; programming provides a reality test for the assumptions made by the analyst. The implementation of a system involves people from different sections and systems analysts are confronted with the practical problem of controlling activities of people outside their own data processing sections. Prior to this point in the project, system analysts have interviewed staff's different sections with the permission of their respective managers but the implementation phase involves staff of user departments carrying out specific task, which require supervision and control to critical schedules. Some of these problems may also need to be employed on the task of

implementation in addition to their own present departmental tasks. Implementation includes all those activities that take place to convert from the old system to new system. The new system may be totally new, replacing an existing manual or automated system or it may be a major modification to an existing system. Proper implementation is essential to provide a reliable system to meet the organization requirements. Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it. The implementation stage involves the following tasks.

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Training of staff in the changeover phase.
- Evaluation of the changeover method.

The method of implementation and the time scale to be adopted are found out initially. Next the system is tested properly and the same time users are trained in the new procedures.

5.1 Implementation Procedure

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended users and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. People were not sure that the software is meant to make their job easier. In the initial stage, they doubt about the software but we have to ensure that the resistance does not built up as one has to make sure that,

- The active user must be aware of using the system.
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual processes won't take place.

5.2 Implementation Plan

Implementation includes all those activities that take place to convert from the old system to the new one. The new system may be totally new, replacing an existing manual or automated system. Proper implementation is essential to provide a reliable system to meet organization requirements. Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from old system to the new system. The system can be implemented only after thorough testing is done and if it is found to be working according to the specifications. The system personally checks the feasibility of the system. The implementation stages involve the following tasks.

- Investigation of system and constraints.
- Design of method to achieve the changeover.
- Training of the staff in the changeover phase.
- Evaluation of the changeover method.

The newly proposed system is implemented after the successful testing of the system.

Post Implementation Review: The final step of the systems approach recognizes that an implemented solution can fail to solve the problem for which it was developed. The results of implementing a solution should be monitored and evaluated. This is called post implementation review process, since the success of a solution is reviewed after it is implemented. The focus on this step is to determine if the implemented solution has indeed helped the firm and selected business units to meet their system objectives.

6. CONCLUSION

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in PYTHON using FLASK Framework and HTML web based Application and MySQL, but also about all handling procedure related with “Alzheimer’s Disease Diagnostics”. It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

Benefits:

- The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
- The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updating so that the user cannot enter the invalid data, which can create problems at later date.
- From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned. That is, we can see that the project is user friendly which is one of the primary concerns of any good project.
- Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
- Through these features it will increase the efficiency, accuracy and transparency.

7. REFERENCES

- An Introduction to Database Systems Bipin.C. Desai
- Software Engineering Somer Viile
- Database system concept Silberschatze
- Learning Python Mark Lutz
- Python for Data Analysis Wes Mckinney
- Python Meachine Learning Sebastian Raschka
- Learning MySQL Seyed Tahaghoghi, Hugh Williams

Websites:

- <https://www.w3schools.com/python/>
- <https://www.tutorialspoint.com/python/>
- https://www.w3schools.com/html/html5_intro.asp
- http://scikitlearn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

8. APPENDICES

A. TABLE DESIGN

Database: **alzheimer's_disease**

Table: **tbl_login**

Field Name	Data Type	Constraints	Size	Description
login_id	int	Primary Key	10	To identify each user
username	varchar	Not Null	20	Name of the user
password	varchar	Not Null	20	Password of the user
login_type	varchar	Not Null	10	Type of user

Table: **tbl_doctor**

Field Name	Data Type	Constraints	Size	Description
doc_id	varchar	Primary Key	10	To identify each doctor
login_id	int	Foreign Key	10	To identify each user
fname	varchar	Not Null	20	First Name of doctor
lname	varchar	Not Null	20	Last Name of doctor
reg_no	varchar	Not Null	20	Doctor Enrollment ID
email	varchar	Not Null	30	Email ID of doctor
phone_no	varchar	Not Null	10	Phone Number of doctor
gender	varchar	Not Null	6	Gender
specialization	varchar	Not Null	30	Area of Specialization

qualification	varchar	Not Null	20	Academics
experience	varchar	Not Null	10	Experience
house	varchar	Not Null	20	House name
city	varchar	Not Null	20	City name
state	varchar	Not Null	20	State name
pin	int	Not Null	6	Pincode

Table: **tbl_patient**

Field Name	Data Type	Constraints	Size	Description
patient_id	varchar	Primary Key	10	To identify each patient
fname	varchar	Not Null	30	First Name of patient
lname	varchar	Not Null	30	Last Name of patient
gender	varchar	Not Null	6	Gender
age	int	Not Null	3	Age of patient
phone_no	int	Not Null	10	Phone Number of patient
house	varchar	Not Null	20	House name
city	varchar	Not Null	20	City name
state	varchar	Not Null	20	State name
pin	int	Not Null	6	Pincode

Table: **tbl_patient_rec**

Field Name	Data Type	Constraints	Size	Description
rec_id	int	Primary Key	10	To identify each patient record
patient_id	varchar	Foreign Key	10	To identify each patient
doc_id	varchar	Foreign Key	10	To identify each doctor
date	date	Not Null		Date
description	varchar	Not Null	50	Medicine Prescribed
image	varchar	Not Null	100	Path of the image

B. SCREEN SHOTS

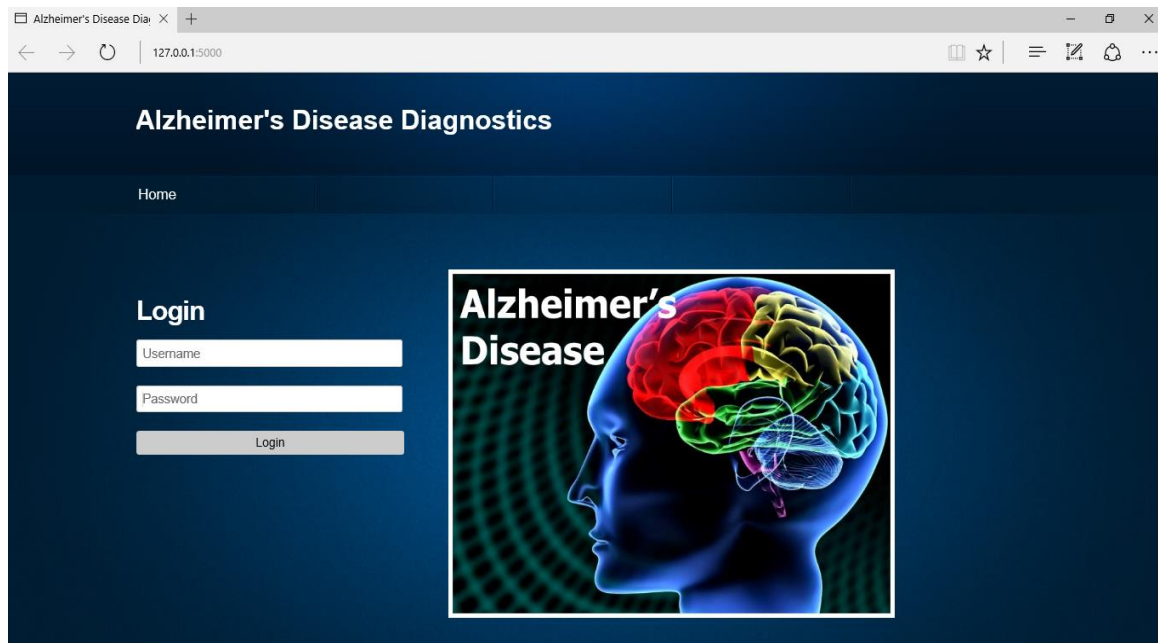


Figure 2: Login Page

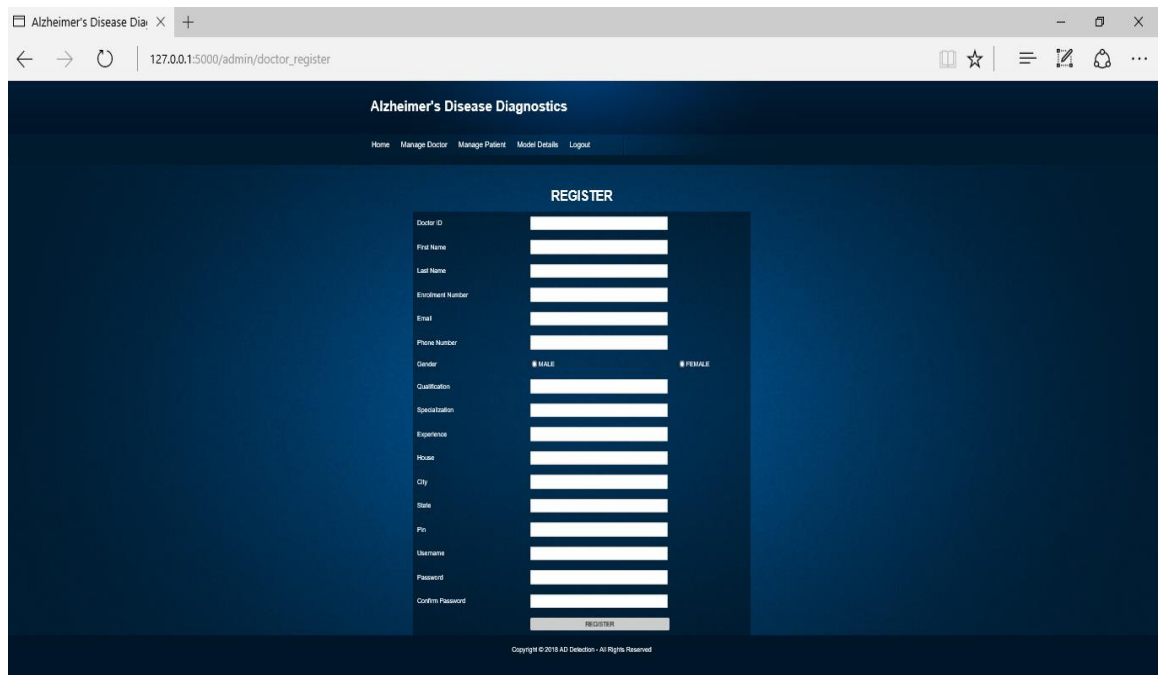


Figure 6: Doctor Registration Page

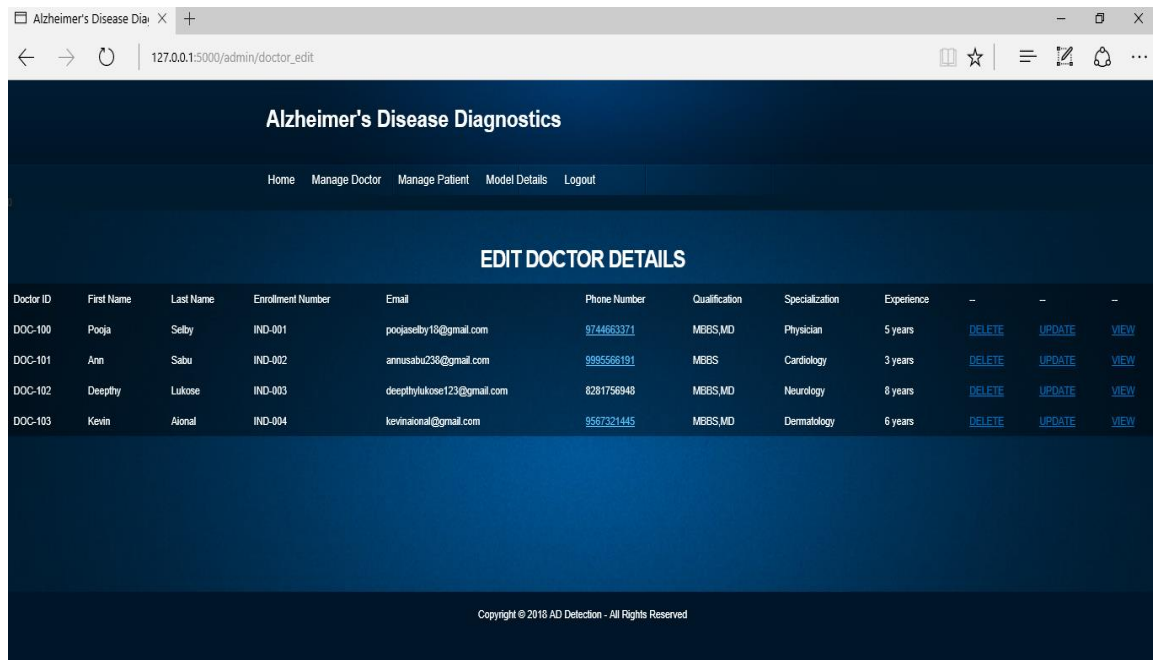


Figure 7: Modify Doctor Details

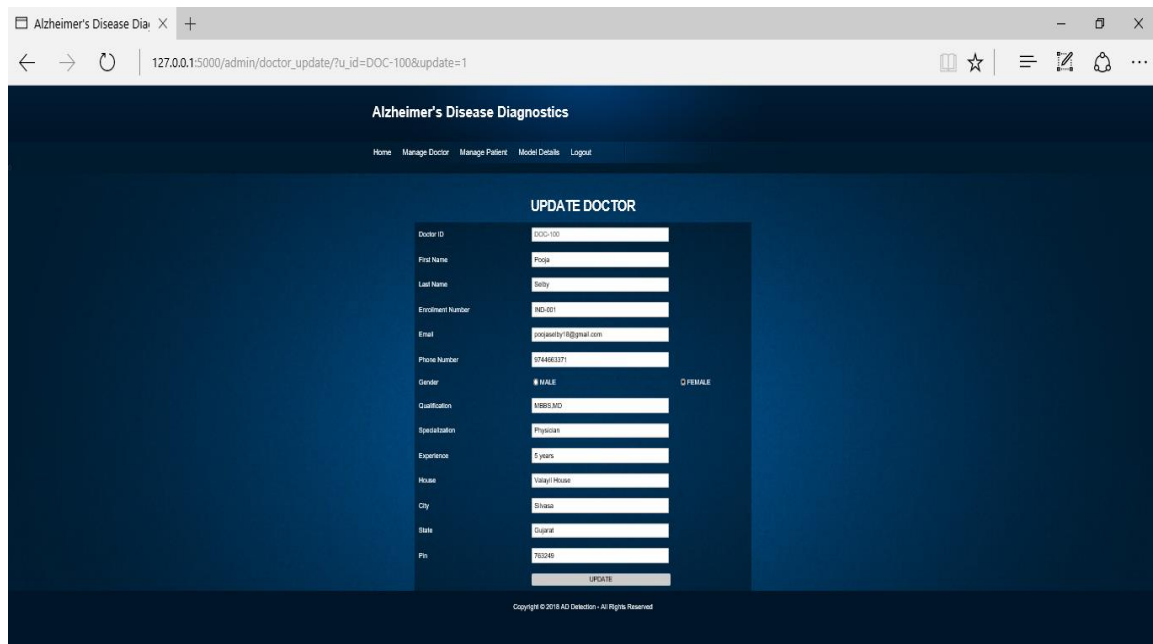


Figure 8: Doctor Profile Update

Alzheimer's Disease Diagnostics

Home Manage Doctor Manage Patient Model Details Logout

DOCTOR DETAILS

Doctor ID	DOC-100
First Name	Pooja
Last Name	Selby
Enrollment Number	IND-001
Email	poojaselby18@gmail.com
Phone Number	9744663371
Gender	Female
Qualification	MBBS MD
Specialization	Physician
Experience	5 years
House	Valayil House
City	Silvasa
State	Gujarat
Pin	763249

Copyright © 2018 AD Detection - All Rights Reserved

Figure 9: Doctor Details

Alzheimer's Disease Diagnostics

Home Manage Doctor Manage Patient Model Details Logout

PATIENT REGISTER

Patient ID	<input type="text"/>
First Name	<input type="text"/>
Last Name	<input type="text"/>
Gender	<input checked="" type="radio"/> MALE <input type="radio"/> FEMALE
Age	<input type="text"/>
Phone Number	<input type="text"/>
House	<input type="text"/>
City	<input type="text"/>
State	<input type="text"/>
Pin	<input type="text"/>
<input type="button" value="REGISTER"/>	

Copyright © 2018 AD Detection - All Rights Reserved

Figure 10: Patient Registration Page

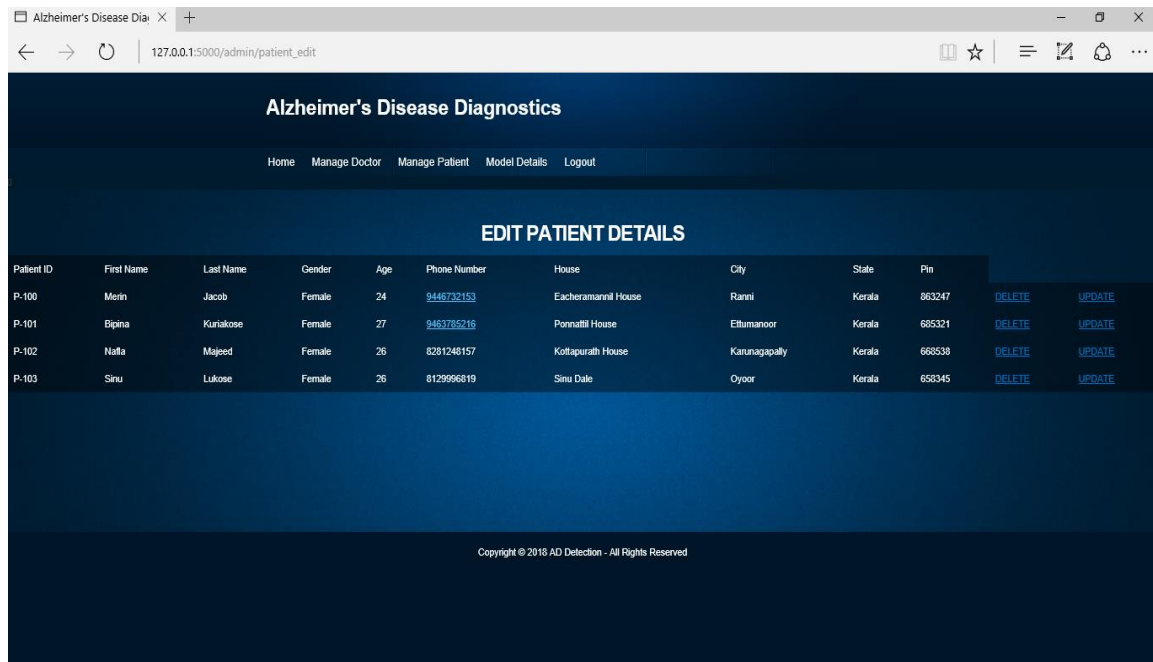


Figure 11: Modify Patient Details

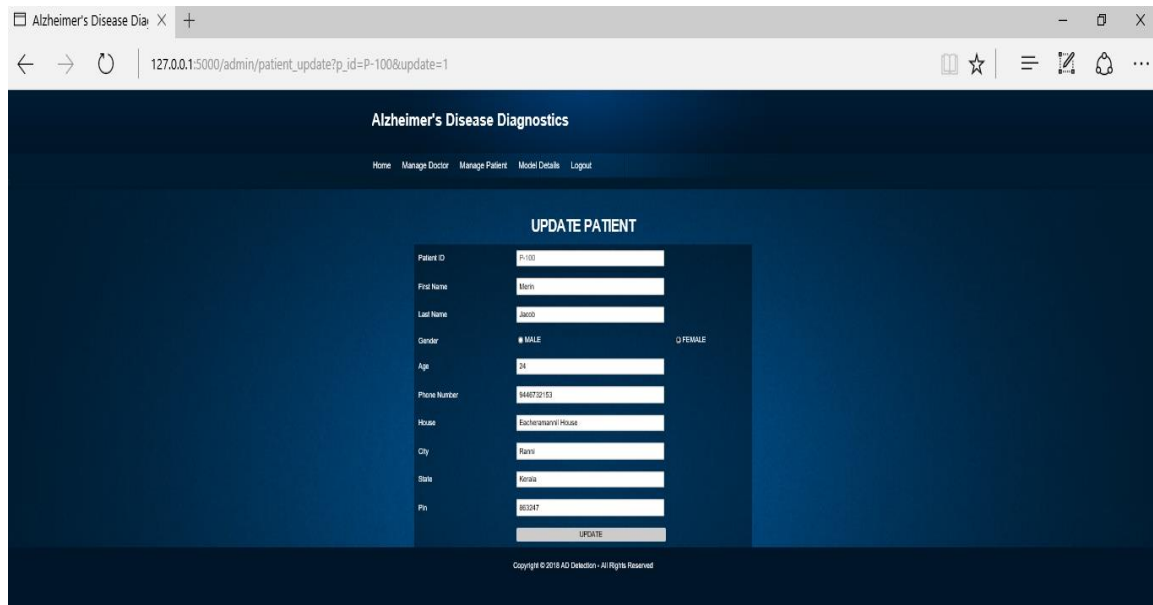
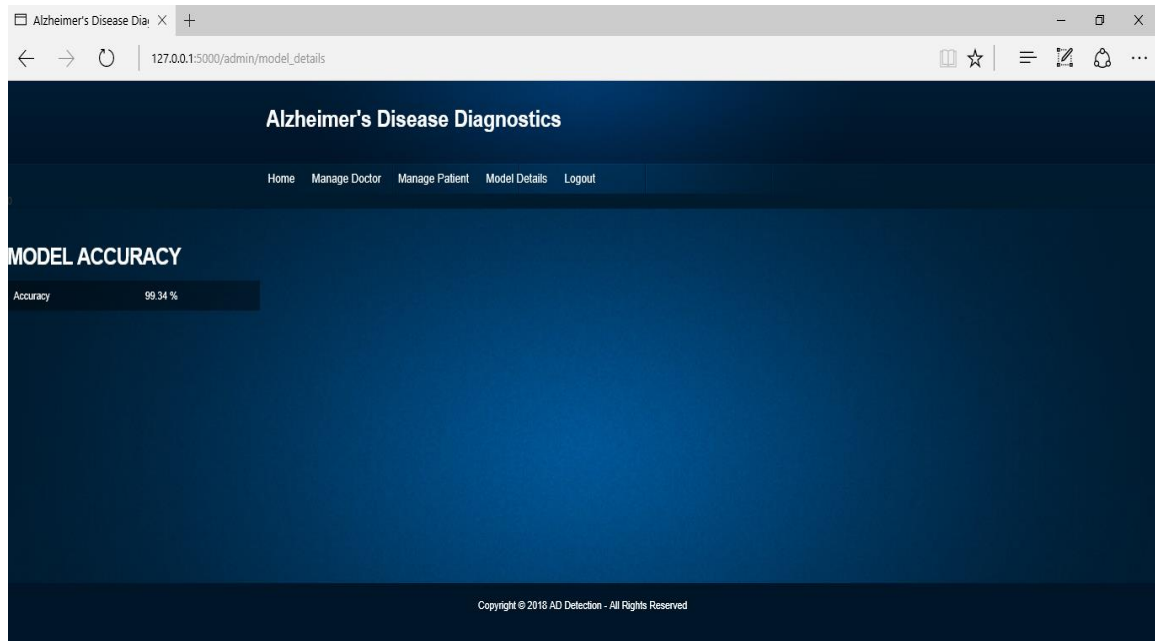
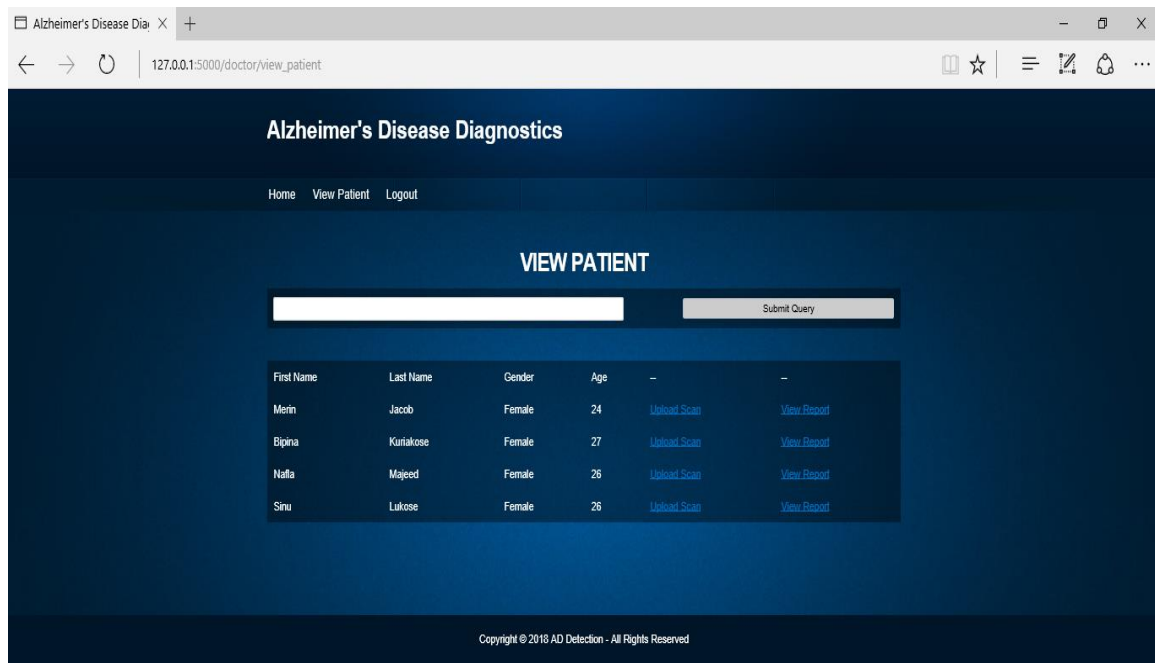
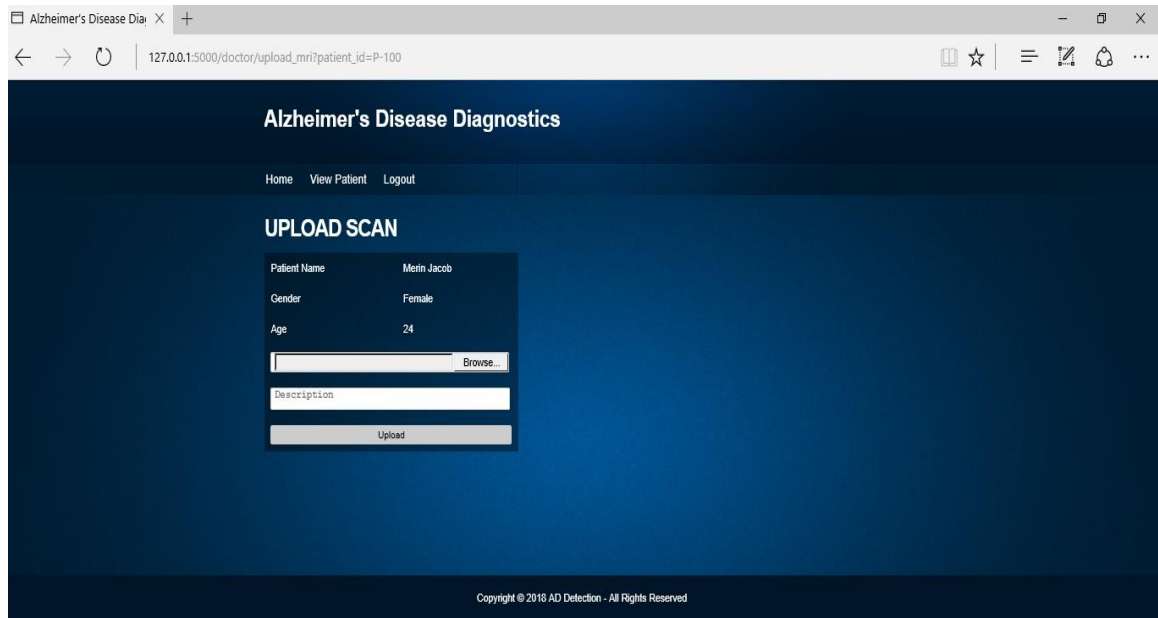
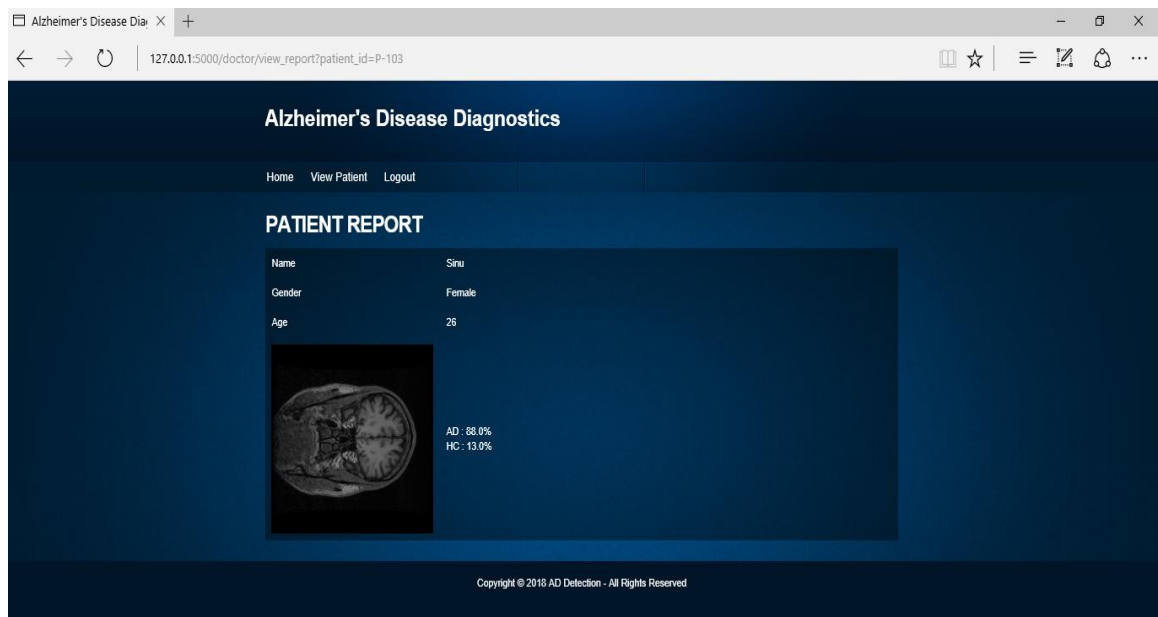


Figure 12: Patient Profile Update

**Figure 13: Model Accuracy****Figure 14: View Patient**

**Figure 15: Upload Scan****Figure 16: Prediction Result**

C. Sample Code

```

import Database as db
from flask import Flask, render_template, redirect, request, session, url_for, flash
import uuid
import core
app=Flask(__name__)
app.secret_key="SSFSSFS"

@app.route('/', methods=['get', 'post'])
def login():
    if "login" in request.form:
        username=request.form['username']
        password=request.form['password']
        log_id=db.select("select * from tbl_login where username='%s'and
password='%s'" %(username,password))
        if len(log_id)>0:
            session['id'] = log_id[0]['login_id']
            flash("LOGIN SUCCESSFUL")
            if log_id[0]['login_type']=='admin':
                return redirect(url_for("admin_home"))
            else:
                return redirect(url_for("doctor_home"))
        else:
            flash("UNSUCCESSFUL")
            return render_template('public/login.html')
    return render_template('public/login.html')

@app.route('/public/forgot_password', methods=['get', 'post'])
def forgot_password():
    if "forgot_password" in request.form:
        username=request.form['username']
        email=request.form['email']
        phone=request.form['phonenumber']
        fpass=db.select("select * from tbl_doctor inner join tbl_login
using(login_id) where username='%s' and email='%s' and phone_no='%s'"
%(username,email,phone))
        if len(fpass)>0:
            password=fpass[0]['password']
            return render_template('public/password_view.html', data=password)
        else:
            flash("Incorrect Data")
    return render_template('public/forgot_password.html')

@app.route('/public/password_view', methods=['get', 'post'])
def password_view():
    if "ok" in request.args:
        return render_template('public/login.html')
    return render_template('public/login.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))

@app.route('/admin/doctor_register/', methods=['get', 'post'])
def doctor_register():

```

```

if "register" in request.form:
    docid=request.form['docid']
    fname=request.form['firstname']
    lname=request.form['lastname']
    enrollemntno=request.form['enrollmentno']
    email = request.form['email']
    phone=request.form['phonenumber']
    gender=request.form['gender']
    qualification = request.form['qualification']
    specialization=request.form['specialization']
    experience=request.form['experience']
    house=request.form['house']
    city=request.form['city']
    state=request.form['state']
    pin=request.form['pin']
    username=request.form['username']
    password=request.form['password']
    confirmpass=request.form['confirmpassword']
    chk=db.select("select username from tbl_login where username='%s'"
%(username))
    if len(chk)>0:
        flash("Already Registered")
        return redirect(url_for("doctor_register"))
    else:
        if password==confirmpass:
            login_id=db.insert("insert into
tbl_login(username,password,login_type)values('%s','%s','doctor')")
%(username,password))
            doc_id=db.insert("insert into
tbl_doctor(doc_id,login_id,fname,lname,reg_no,email,phone_no,gender,qualificati
on,specialization,experience,house,city,state,pin)values('%s','%s','%s','%s','%
s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s')")
%(docid,login_id,fname,lname,enrollemntno,email,phone,gender,qualification,spec
ialization,experience,house,city,state,pin))
            if login_id > 0:
                flash("SUCCESSFULLY REGISTERED")
            else:
                flash("UNSUCCESSFUL")
        return render_template('admin/doctor_register.html')

@app.route('/admin/doctor_update/',methods=['get','post'])
def doctor_update():
    if "update" in request.form:
        u_id=request.args['u_id']
        fname=request.form['firstname']
        lname=request.form['lastname']
        enrollmentno=request.form['enrollmentno']
        email=request.form['email']
        phone = request.form['phonenumber']
        gender=request.form['gender']
        qualification=request.form['qualification']
        specialization=request.form['specialization']
        experience=request.form['experience']
        house = request.form['house']
        city = request.form['city']
        state = request.form['state']
        pin = request.form['pin']
        value=db.update("update tbl_doctor set
fname='%s',lname='%s',reg_no='%s',email='%s',phone_no='%s',gender='%s',qualific
ation='%s',specialization='%s',experience='%s',house='%s',city='%s',state='%s',
pin='%s' where doc_id='%s'"
%(fname,lname,enrollmentno,email,phone,gender,qualification,specialization,expe

```

```

rience,house,city,state,pin,u_id))
    flash("SUCCESSFULLY UPDATED")
    return redirect(url_for("doctor_edit"))
    if "update" in request.args:
        u_id=request.args['u_id']
        s=db.select("select
doc_id,fname,lname,reg_no,email,phone_no,gender,qualification,specialization,ex
perience,house,city,state,pin from tbl_doctor where doc_id='%s'" %(u_id))
        return render_template('admin/doctor_update.html',data=s[0])

```

```

@app.route('/admin/doctor_edit',methods=['get','post'])
def doctor_edit():
    if "delete" in request.args:
        dele_id=request.args['id']
        d=db.delete("delete from tbl_doctor where login_id='%s'" %(dele_id))
        l=db.delete("delete from tbl_login where login_id='%s'" %(dele_id))
        s=db.select("select * from tbl_doctor")
        return render_template('admin/doctor_edit.html',data=s)

```

```

@app.route('/admin/doctor_view',methods=['get','post'])
def doctor_view():
    if "view" in request.args:
        v_id=request.args['v_id']
        v=db.select("select * from tbl_doctor where doc_id='%s'" %(v_id))
        return render_template('admin/doctor_view.html',data=v[0])

```

```

@app.route('/admin/patient_register',methods=['get','post'])
def patient_register():
    if "register" in request.form:
        patientid=request.form['patientid']
        fname=request.form['firstname']
        lname=request.form['lastname']
        gender=request.form['gender']
        age=request.form['age']
        phone=request.form['phonenumber']
        house=request.form['house']
        city=request.form['city']
        state=request.form['state']
        pin=request.form['pin']
        patient_id=db.insert("insert into
tbl_patient(patient_id,fname,lname,gender,age,phone_no,house,city,state,pin)val
ues('%s','%s','%s','%s','%s','%s','%s','%s','%s','%s')")
        %(patientid,fname,lname,gender,age,phone,house,city,state,pin))
        if (patient_id>0):
            flash("SUCCESSFULLY INSERTED")
        else:
            print ("Unsuccessful")
        return render_template('admin/patient_register.html')

```

```

@app.route('/admin/patient_update',methods=['get','post'])
def patient_update():
    if "update" in request.form:
        p_id=request.args['p_id']
        fname=request.form['firstname']
        lname=request.form['lastname']
        gender=request.form['gender']
        age=request.form['age']
        phone=request.form['phonenumber']
        house=request.form['house']

```

```

        city=request.form['city']
        state=request.form['state']
        pin=request.form['pin']
        value=db.update("update tbl_patient set
fname='%s',lname='%s',gender='%s',age='%s',phone_no='%s',house='%s',city='%s',s
tate='%s',pin='%s' where patient_id='%s'"
%(fname,lname,gender,age,phone,house,city,state,pin,p_id))
        flash("SUCESSFULLY UPDATED")
        return redirect(url_for("patient_view"))
    if "update" in request.args:
        p_id = request.args['p_id']
        s = db.select("select
patient_id,fname,lname,gender,age,phone_no,house,city,state,pin from
tbl_patient where patient_id='%s'" %(p_id))
        return render_template('admin/patient_update.html', data=s[0])

@app.route('/admin/patient_edit',methods=['get','post'])
def patient_edit():
    if "delete" in request.args:
        dele_id=request.args['id']
        d=db.delete("delete from tbl_patient where patient_id='%s'" %(dele_id))
        s=db.select("select * from tbl_patient")
        return render_template('admin/patient_edit.html',data=s)

@app.route('/admin/view_report',methods=['get','post'])
def view_report():
    value=db.select("select fname,lname,gender,age from tbl_patient")
    return render_template('admin/view_report.html',data=value[0])

@app.route('/admin/model_details',methods=['get','post'])
def model_details():
    details = core.get_model_score()
    return render_template('admin/model_details.html', model_details=details)

@app.route('/admin/train')
def train():
    details = core.train_model()
    return "ok"

@app.route('/doctor/view_patient',methods=['get','post'])
def view_patient():
    v=db.select("select * from tbl_patient")
    if "search" in request.form:
        fname=request.form['firstname']
        q = "select * from tbl_patient where fname like '%s'" %
("%"+fname+"%")
        v=db.select(q)
        return render_template('doctor/view_patient.html',data=v)

@app.route('/doctor/view_report')
def doctor_view_report():
    patient_id = request.args['patient_id']
    q = "select * from tbl_patient left join tbl_patient_rec using(patient_id)
where patient_id='%s'" % patient_id
    res = db.select(q)
    for i in range(len(res)):

```

```

        row = res[i]
        if row['image'] != None:
            ad = core.get_ad_detection(row['image'])
            row['AD'] = ad
        res[i] = row
    return render_template('doctor/doctor_view_report.html', report = res)

    return "ok"

@app.route('/doctor/upload_mri', methods=['get', 'post'])
def upload_mri():
    patient_id = request.args['patient_id']
    if "upload" in request.form:
        image = request.files['scan']
        filename = "static/uploads/" + str(uuid.uuid4()) + "."
+ (image.filename).split(".")[1]
        image.save(filename)
        log_id = session['id']
        description = request.form['description']
        q = "insert into tbl_patient_rec
(patient_id,date,doc_id,description,image)values('%s',curdate(),(select doc_id
from tbl_doctor where login_id='%s'), '%s', '%s') " %
(patient_id, log_id, description, filename)
        db.insert(q)
        q = "select * from tbl_patient where patient_id='%s'" % patient_id
        res = db.select(q)
        return render_template('doctor/upload_mri.html', patient_details=res)

@app.route('/admin/admin_home')
def admin_home():
    if "id" in session:
        return render_template('admin/admin_home.html')
    else:
        return redirect(url_for('login'))

@app.route('/doctor/doctor_home')
def doctor_home():
    return render_template('doctor/doctor_home.html')

app.run(debug=True)

```

```

import pickle
import os
import numpy as np
import random
from sklearn.neural_network import MLPClassifier
import cv2

def preprocess_img(img):
    image_size = (224, 224)
    def image_resize(img, image_size):
        img = np.resize(img, image_size)
        return img

```

```

def normalize(img):
    img = img / 255
    return img
def reshape(img):
    img = np.reshape(img, (50176))
    return img

img = image_resize(img, image_size)
img = normalize(img)
img = reshape(img)
return img

def load_pickle(file_name):
    file = open(file_name, "rb")
    data = pickle.load(file)
    file.close()
    return data

def prepare_training_set():
    dataset = load_pickle("core/dataset.pickle")
    AD = dataset['AD']
    HC = dataset["HC"]
    dataset = []
    i = 0
    for person in HC:
        for j in range(33, 86, 1):
            scan_image = person[j]
            dataset.append((preprocess_img(scan_image), [1, 0]))
        i += 1

    for person in AD :
        for j in range(33, 86, 1):
            scan_image = person[j]
            dataset.append((preprocess_img(scan_image), [0, 1]))
        i -= 1
    if i == 0:
        break

    random.shuffle(dataset)
    training_input = dataset[0:int(len(dataset) * .75)]
    testing_set = dataset[int(len(dataset) * .75):len(dataset)]
    tr_input = []
    tr_output = []
    for i in training_input:
        tr_input.append(i[0])
        tr_output.append(i[1])
    ts_input = []
    ts_output = []
    for i in testing_set:
        ts_input.append(i[0])
        ts_output.append(i[1])
    tr_input = np.array(tr_input)
    tr_output = np.array(tr_output)
    ts_input = np.array(ts_input)
    ts_output = np.array(ts_output)
    return tr_input, tr_output, ts_input, ts_output

def create_model():
    clf = MLPClassifier(

        hidden_layer_sizes=(100,),
        activation='relu',

```

```

        solver='adam',
        alpha=0.0001,
        batch_size='auto',
        learning_rate='constant',
        learning_rate_init=0.001,
        power_t=0.5,
        max_iter=200,
        shuffle=True,
        random_state=None,
        tol=0.0001,
        verbose=False,
        warm_start=False,
        momentum=0.9,
        nesterovs_momentum=True,
        early_stopping=False,
        validation_fraction=0.1,
        beta_1=0.9,
        beta_2=0.999,
        epsilon=1e-08
    )

    return clf

def train(mlp,X_train,y_train,X_test,y_test,file_name):
    N_TRAIN_SAMPLES = X_train.shape[0]
    N_EPOCHS = 125
    N_BATCH = 128
    N_CLASSES = np.unique(y_train)
    scores_train = []
    scores_test = []
    epoch = 0
    while epoch < N_EPOCHS:
        print('epoch: ', epoch)
        random_perm = np.random.permutation(X_train.shape[0])
        mini_batch_index = 0
        while True:
            # MINI-BATCH
            indices = random_perm[mini_batch_index:mini_batch_index + N_BATCH]
            mlp.partial_fit(X_train[indices], y_train[indices],
classes=N_CLASSES)
            mini_batch_index += N_BATCH
            if mini_batch_index >= N_TRAIN_SAMPLES:
                break
            scores_train.append(mlp.score(X_train, y_train))
            score = mlp.score(X_test, y_test)
            print(score)
            file = open(file_name,"wb")
            pickle.dump(mlp,file)
            file.close()

            # SCORE TEST
            scores_test.append(score)
            file = open("core/score.pickle", "wb")
            pickle.dump(scores_test, file)
            file.close()
            epoch += 1

def get_model_score():
    scores = load_pickle("core/score.pickle")
    return scores[-1]

def get_ad_detection(path):

```



```
classifier = load_pickle("core/classifier.pickle")
if os.path.exists(path):
    img = preprocess_img(cv2.imread(path,0))
    result = classifier.predict([img])
    prob_result = classifier.predict_proba([img])
    print(prob_result)
return prob_result[0]

def train_model():
    tr_input, tr_output, ts_input, ts_output = prepare_training_set()

train(create_model(),tr_input,tr_output,ts_input,ts_output,"core/classifier.pickle")

#train_model()
```