



**COMSATS University Islamabad**  
**Abbottabad, Pakistan**

# **Detecting Parkinson's Disease using Machine Learning**

*By*

<b>Babar Ali</b>	<b>CIIT/FA18-BCS-043/ATD</b>
<b>Hassaan Moeed</b>	<b>CIIT/FA18-BCS-044/ATD</b>
<b>Mashaim Rehman</b>	<b>CIIT/FA18-BCS-051/ATD</b>

*Supervisor*

**Ms. Nudrat Habib**

***Bachelor of Science in Computer Science***  
***(2018-2022)***

**The candidate confirms that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others.**



**COMSATS University, Islamabad Pakistan**

# **Detecting Parkinson's Disease using Machine Learning**

**A project presented to  
COMSATS Institute of Information Technology, Islamabad**

**In partial fulfillment  
of the requirement for the degree of**

***Bachelor of Science in Computer Science  
(2018-2022)***

**By**

<b>Babar Ali</b>	<b>CIIT/FA18-BCS-043/ATD</b>
<b>Hassaan Moeed</b>	<b>CIIT/FA18-BCS-044/ATD</b>
<b>Mashaim Rehman</b>	<b>CIIT/FA18-BCS-051/ATD</b>

## **DECLARATION**

We thus state that neither the entirety of the software nor any individual components have been taken from any other source. It is additionally stated that we created the report and the accompanying software fully through our own efforts. If any portion of this project is found to be a reproduction of another or to have been taken verbatim from another source. We shall adhere to the results. No part of the work that is being presented has been submitted for any other degree or certification at this university or educational institution or any other.

Babar Ali

Hassaan Moeed

Mashaim Rehman

-----

-----

-----

## **CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “Detecting Parkinson’s Disease using Machine Learning ” was developed by **Babar Ali (CIIT/FA18-BCS-043)**, **Hassaan Moeed (CIIT/FA18-BCS-044)**, **Mashaim Rehman (CIIT/FA18-BCS-051)** under the supervision of “Ms. Nudrat Habib” and that in her opinion; it is fully adequate, in scope and quality for the degree of Bachelor of Science in Computer Sciences.

-----  
**Supervisor**

-----  
**External Examiner**

-----  
**Head of Department**  
**(Department of Computer Science)**

## **EXECUTIVE SUMMARY**

After Alzheimer, the most prevalent neurodegenerative disorder condition is reportedly Parkinson's whose discovery dates to the early 19th century, when James Parkinson initially referred to it as a neurological syndrome. Parkinson's disease is brought on by abnormal brain activity, which is caused when dopamine levels in the brain fall. Doctors mostly pay attention to classic motor symptoms. Parkinson's is a characteristically discriminatory disease that cannot be identified through blood pressure or cholesterol testing. Given that each patient has unique symptoms, early detection is the most difficult element of the diagnosis. Also, many Parkinson's disease patients find it extremely difficult to travel to medical facilities for treatment. Speech difficulties may be one of the disease's early-stage symptoms.

Our Android application, which runs on a server, focuses on detecting Parkinson's disease in its initial

stages so that the patient can become aware of their condition sooner and with less consequences. The proposed system uses a microphone that is mounted on the user interface to record audio. PRAAT that is an intermediate application extracts features from the audio signal to feed into ML Model.

KNN, SVM, Random Forest, and XGBoost are trained to distinguish between a healthy and an unwell person, with accuracy rates of 83%, 89%, 93%, and 94%, respectively.

XGBoost provides the maximum frequency so it was integrated with the app. The trained model and database are deployed on the server, and the features are delivered there to produce an output. The data set was acquired and utilised to test the model from UCI ML Repository. <sup>i</sup> The dataset has 23 attributes (name, MDVP:Fo(Hz), MDVP:Fhi(Hz), MDVP:Flo(Hz), MDVP:Jitter(%), MDVP:Jitter(Abs), MDVP:RAP, MDVP:PPQ, Jitter:DDP, MDVP:Shimmer, MDVP:Shimmer(dB), Shimmer:APQ3, Shimmer:APQ5, MDVP:APQ, Shimmer:DDA, NHR,HNR, status, RPDE,D2, DFA, spread1, spread2, PPE) and 195 number of instances.

## **ACKNOWLEDGEMENT**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are indebted to our project supervisor “Ms. Nudrat Habib.” Without her personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & challenging work.

Babar Ali

Hassaan Moeed

Mashaim Rehman

-----

-----

-----

## **ABBREVIATIONS**

<b>SRS</b>	Software Require Specification
<b>SDD</b>	Software Design Document
<b>PC</b>	Personal Computer
<b>PD</b>	Parkinson's Disease
<b>SDK</b>	Software Development Kit
<b>UC</b>	Use Case
<b>ML</b>	Machine Learning
<b>KNN</b>	K-nearest neighbors
<b>SVM</b>	Support vector machine
<b>SDLC</b>	Software Development Life Cycle
<b>API</b>	Application Programming Interface
<b>GDBT</b>	Gradient-boosted decision trees
<b>OOP</b>	Object Oriented Programming
<b>MPI</b>	Message Passing Interface
<b>CAD</b>	Computer Aided Diagnosis

## **TABLE OF CONTENTS**

<b><u>1</u></b>	<b><u>Introduction</u></b> .....	1
1.1	<u>Brief</u> .....	1
1.2	<u>Relevance to Course Modules</u> .....	2
1.3	<u>Project Background</u> .....	2
1.4	<u>Literature Review</u> .....	2
1.5	<u>Analysis from Literature Review (in the context of your project)</u> .....	2
1.6	<u>Methodology and Software Lifecycle for This Project</u> .....	5
1.6.1	<u>Rationale behind Selected Methodology</u> .....	6
<b><u>2</u></b>	<b><u>Problem Definition</u></b> .....	7
2.1	<u>Problem Statement</u> .....	7
2.2	<u>Deliverables and Development Requirements</u> .....	7
<b><u>3</u></b>	<b><u>Requirement Analysis</u></b> .....	8
3.1	<u>Use Cases Diagram(s)</u> .....	8
3.2	<u>Detailed Use Case</u> .....	9
3.3	<u>Functional Requirements</u> .....	15
3.4	<u>Non-Functional Requirements</u> .....	15
<b><u>4</u></b>	<b><u>Design and Architecture</u></b> .....	16
4.1	<u>System Architecture</u> .....	16
4.2	<u>Data Representation [Diagram + Description]</u> .....	21
4.3	<u>Design Models [along with descriptions]</u> .....	22
<b><u>5</u></b>	<b><u>Implementation</u></b> .....	23
5.1	<u>Algorithms</u> .....	23
5.2	<u>External APIs</u> .....	25
5.3	<u>User Interface</u> .....	26
5.4	<u>How to run the Application</u> .....	33
<b><u>6</u></b>	<b><u>Testing and Evaluation</u></b> .....	35
6.1	<u>Manual Testing</u> .....	35
6.1.1	<u>System testing</u> .....	35
6.1.2	<u>Unit Testing</u> .....	35
6.1.3	<u>Functional Testing</u> .....	37
6.1.4	<u>Integration Testing</u> .....	40
<b><u>7</u></b>	<b><u>Conclusion and Future Work</u></b> .....	42
7.1	<u>Conclusion</u> .....	42
7.2	<u>Future Work</u> .....	42
<b><u>8</u></b>	<b><u>References</u></b> .....	43



## **LIST OF FIGURES**

Fig 1.1 Incremental Model .....	6
Fig 3.1 Use Case Diagram .....	8
Fig 4.1 Sign-up.....	17
Fig 4.2 Login.....	18
Fig 4.3 Homepage .....	19
Fig 4.4 Voice record.....	20
Fig 4.5 Class Diagram .....	21
Fig 4.6 Design Model.....	22
Fig 4.7 System Block Diagram .....	23
Fig 5.1 Sign-up.....	26
Fig 5.2 Login.....	27
Fig 5.3 Homepage .....	28
Fig 5.4 Profile management .....	29
Fig 5.5 Audio Prediction .....	30
Fig 5.6 Server Homepage.....	31
Fig 5.7 Server Dashboard.....	32
Fig 5.8 Anaconda PowerShell.....	33
Fig 5.9 Ngrok .....	34
Fig 5.10 Android Studio.....	34

## **LIST OF TABLES**

Table 1.1 Comparison .....	3
Table 1.2 Comparison .....	4
Table 3.1 UC-1.....	9
Table 3.2 UC-2.....	10
Table 3.3 UC-3.....	11
Table 3.4 UC-4.....	12
Table 3.5 UC-5.....	13
Table 3.6 UC-6.....	14
Table 5.1 Important ML libraries.....	25
Table 5.2 APIs used.....	25
Table 6.1 Login Unit Testcase .....	35
Table 6.2 Signup Unit Testcase.....	36
Table 6.3 Get-Prediction Unit Testcase.....	37
Table 6.4 Login Functional Test Case.....	37
Table 6.5 Sign Up Functional Test Case.....	39
Table 6.6 Integration Test Case.....	40

# 1. Introduction

The second most prevalent form of brain disease is Parkinson's disease whose discovery dates to the early 19th century where it was first described as neurological syndrome by James Parkinson <sup>ii</sup>. Parkinson's disease is caused by the loss of the neurons in the brain that produce the chemical messenger dopamine. When the amount of the dopamine in the brain decreases, it causes unusual brain activities, which leads to Parkinson's disease. Although the precise cause of Parkinson's disease is unknown, several factors, such as genes, the environment, and triggers, seem to be involved <sup>iii</sup>. Doctors focus on Classic Motor Symptoms such as Resting Tremor, Stiffness, Slowness in movement and Speech changes to track down whether a person has Parkinson's or not. Typically, Parkinson disease symptoms appear after the age of 60, but hereditary factors can cause them -to appear earlier, in the 40s.

The application accepts audio input from the user because vocal alterations are one of the disease's most significant and early signs. Additionally, the application eliminates the inconvenience of going to the doctor. An estimated 90% of Parkinson's disease patients are thought to be vocally susceptible. In other words, speech difficulties may be one of the disease's early-stage symptoms. Voice signal recording is simple and non-intrusive, making it a noteworthy parameter for identifying and monitoring the evolution of symptoms. In this project, the subject's voice is recorded using a microphone and then particular aspects are extracted from the recorded signal and are examined using various techniques. One of the initial signs of early-stage PD is a pattern of atrophy and a linguistic impairment. The speech becomes repetitive and rapid while the voice becomes hushed. Over time, the voice becomes less audible, and in later stages of the illness, the patient may only whisper. The mumbling may be a diseases' first warning symptom.

## 1.1. Brief

The proposed system is an android-based application that runs through a python server Django, where the machine learning model is deployed. The app enables the user to get a prediction about his status of Parkinson's disease. Using the app interface, the user will create his profile and record his voice sample using a microphone and get the result of the prediction. The system will either predict it as 0 for healthy, or 1 for affected. The results of predictions and profile will be maintained using a database.

We have limited our input to voice only, as the dataset is based on voice features also pattern of atrophy and a linguistic impairment is one of the initial signs of early-stage PD. Features are extracted from the voice which are then given to the machine model as an input for further processing. The trained machine learning model then predicts and generates outputs, which categorizes a PD affected and a non-affected patient.

A database is used to keep records of users. The system focuses on attaining accuracy, precision and reducing errors. Suggesting a cure for the disease is not a part of this system.

The key features of PD detection app are:

- User will Sign up and login

- Users can conduct a disease prediction test.
- Users can view and update his profile.
- User can check out his previous history
- User can initiate a recording and send it for further processing.
- User can logout of the application.

### **1.1. Relevance to Course Modules**

Our final year project is an application that runs on an android device. Our project is related to various subjects that we studied in 4-years of study at university. Courses related to designing and software engineering concepts were key in developing the roadmap for implementation. The development process encompasses several technologies, which starts with front-end development in java, using Android Studio. The 4 key concepts of java are implemented there. In the later half, machine learning techniques and developing models included skills learned at Machine Learning relevant courses. Courses of Web Technologies is useful in deploying the model over a server. Principles of user experience goals are also implemented, studied in Human Computer Interaction course.

### **1.2. Project Background**

The idea behind the project is to develop an Android Application which allows people to conduct a Parkinson's disease test with some simple screen taps. The application focuses on generating accurate results based on audio features.

### **1.3. Literature Review**

**Research Paper/ Journal Name:** High-accuracy detection of early Parkinson's Disease using multiple characteristics of finger movement while typing<sup>iv</sup>

The research aims to examine those features of finger movements which are affected by PD, and then create an application based on machine learning that could classify a PD affected person correctly.

The data from the user was collected in a way that, Tappy application ran in background in parallel as the user used the computer and recorded some movements of the fingers of both right- and left-hand accessed keys and iteratively made a csv file every day and sent it to the server.

After taking data from 200+ people, only 53 participants were chosen in the final dataset, which dropped many with mild symptoms or taking some medication.

In the preprocessing phase 27 features were causing a curse of dimensionality, so dimensionality reduction technique "Linear Discriminant Analysis" was performed on it.

#### 1.4. Analysis from Literature Review (in the context of your project)

(Parkinson's Disease Detection from 20-Step Walking Tests Using Inertial Sensors of a Smartphone: Machine Learning Approach Based on an Observational Case-Control Study, n.d.)<sup>v</sup>

**Research Paper/ Journal Name:** Parkinson's disease detection from 20-step walking tests using inertial sensors of a smartphone: Machine learning approach based on an observational case-control study

The research question of this study consisted of finding feasibility in feature selection

and classifying the walk test performed at the clinic. It aimed to undergo various machine learning algorithms to find which one of them performed better in detecting PD patients.

A local hospital in Finland was used to collect the data. Approximately 97 people participated in the entire process who lived nearby and from outer areas. A scale of 0-5 was given to the subjects based on severity of PD symptoms. 0 being unaffected and 5 being mostly affected. Subjects with milder symptoms were dropped. The subjects were divided into 29 healthy and 29 affected ratios.

MATLAB was used to perform data analysis. Data related to walking was collected by means of smartphone inertial sensors. Individual strides were created after preprocessing. The features were reduced to an exceedingly small number from 201 initially, to facilitate the feature selection process. Minimum Redundancy Maximum Relevance, Sequential forward feature selection, and Sequential backward feature selection were used as Feature Selection techniques.

KNN had the highest accuracy of 84.5% with 15 number of features selected in this study.

##### **20-Step Walking Test:**

<b>Feature Selection Method</b>	<b>No of Features</b>	<b>ML Algorithm</b>	<b>Accuracy (High to Low)</b>
Overall Classification	15	KNN k=49	84.5%

*Detecting Parkinson's Disease Using Machine Learning*

sequential feature selection (SFS)	7	Naive Bayes classifier	75.3%
sequential feature selection (SFS)	4	K Nearest Neighbors.	75.1%

*Table 1.1: Comparison*

**PD Detection using voice-based features:**

<b>Feature Selection Method</b>	<b>No of Features</b>	<b>ML Algorithm</b>	<b>Accuracy (High to Low)</b>
None	22	Random Forest	93.8%
None	22	Support Vector Machines	88.46
Heatmap (Comparing Coefficient correlation)	11	K Nearest Neighbors. K=5	84.61%
None	22	K Nearest Neighbors. K=7	87.17%

*Table 1.2: Comparison*

Following were the limitations of the studies research:

The mean age was between 60-70 years thus the patients may have shown slowness in movement due to other disabilities as well. The current walking test is useful for clinical data, it is not suitable for a wild environment as of now and the mobile phone needed to be gyroscope enabled.

### **1.5. Methodology and Software Lifecycle for This Project**

For “PD Detection Application” we use “Object Oriented Methodology” due to polymorphism's flexibility and inheritance's

- Ability to allow for code reuse.
- Successful problem-solving.
- Modularity to make troubleshooting simpler.

Python was used for the machine learning portion, which is an "Object-Oriented Approach" that adheres to bottom-

up programming. Bottomup programming focuses on developing an algorithm by starting at the most fundamental level and progressing upward. With this approach, every module is created and individually evaluated (unit tested) before being integrated to create a tangible solution. Utilizing low-level functions, unit testing is conducted.

Advantages:-

- Making up test conditions is simpler
- Examining test results is simpler.
- has reduced redundancy because data encapsulation and data-hiding are present.

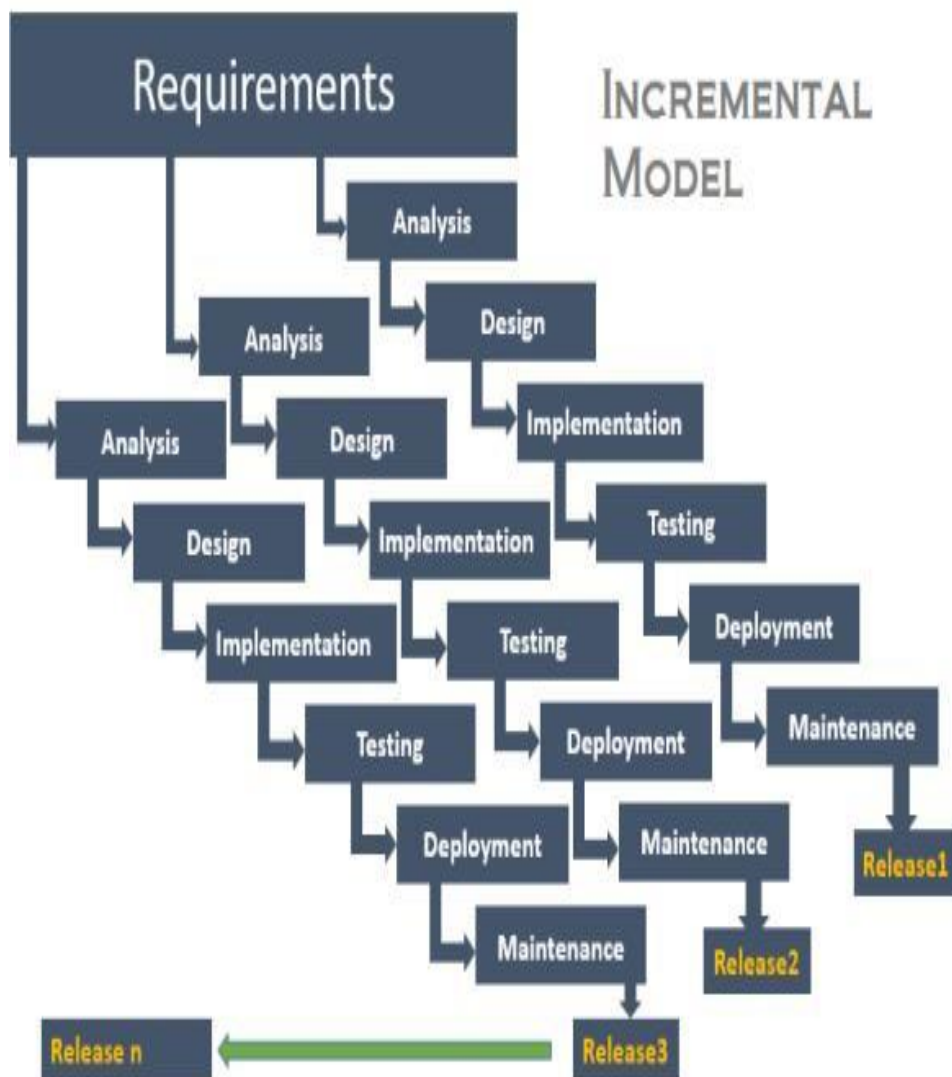
We used ML because it had built in algorithms. And machine learning in python was the most convenient amongst all as it had built in libraries providing specific functionalities. Code reusability was one key aspect, moreover it provided extensive support for deploying the model over the Django Server, and hosting services used for communication across Android application. For “PD Detection Application” we will use the process model called “Incremental model” as shown in Figure 1.1. In our project we will use an incremental model because in an incremental model the modules are developed in increments or in chunks and as we will deliver our project in the form of working modules or functionalities.

In an incremental model, requirements, design, coding, and testing are all completed at each iteration. With each new iteration, the system gains additional capabilities until it reaches its full potential. Therefore, the incremental model was the ideal choice for our project. If more iterations are necessary after the first, the application can be produced in new versions.

- Quickly and early in the software life cycle produces functional software.
- This paradigm is more adaptable and makes changing the requirements and scope less expensive.
- In a shorter iteration, testing and bug-fixing are simpler.
- In a shorter iteration, testing and bug-fixing are simpler.
- Because hazardous elements are detected and dealt with throughout each iteration, this method lowers the initial delivery cost and makes risk management simpler.
- The software will be generated quickly during the software life cycle
- Throughout the development stage, changes were easy to manage.

The workflow in-process model is given in the next section:

*Figure 1.1: Incremental model*





### **1.5.1. Rationale behind Selected Methodology**

In the “PD Detection” app we used Object Oriented Methodology. This helped us increase the project's productivity and quality. Also, Java follows object- oriented techniques that classify the data and behaviors. For Object Oriented Methodology we must use Android Studio for programming.

We added machine learning because it had built in algorithms. And machine learning in python was the most convenient amongst all as it had built in libraries providing specific functionalities. Code reusability was one key aspect, moreover it provided extensive support for deploying the model over the Django Server, and hosting services used for communication across Android application

Using Object oriented programming, a lot of benefits were yielded which are as follows:

- Re-usability.
- Data Redundancy.
- Code Maintenance.
- Security.
- Design be easily reverted.
- Can easily be tested.
- Better for debugging due to small modules.
- Follows SDLC (System Development Life Cycle)

The above-mentioned advantages made the development process flexible and more time saving, which then resulted in better productivity and code was comparatively easy to handle. The main importance of the Incremental model is that it divides the software development into submodules and each submodule Software requirements are divided or broken down into multiple stand-alone modules/increments in the SDLC (Software Development Life Cycle).

## **2. Problem Definition**

As we discussed previously, the second most prevalent type of brain ailment is Parkinson's disease and doctors tend to concentrate mostly on the classic motor symptoms. Few people avoid going to the doctor until there are no overt symptoms of disease. This app will therefore offer everyone a quick and simple technique of early detection without the necessity for a direct visit to the doctor.

### **2.1. Problem Statement**

The system we are proposing is an android app that targets one of the main symptoms, which is Change in Speech. It is different in its approach because the system will be able to detect a Parkinson's patient, by evaluating the parameters of his voice obtained through the microphone attached on the app. User inputs a voice on the app of which result is generated. This development will help the concerned patients and doctors to anticipate and take adequate measures for its prevention. Research have been continued various levels, to form a system that can detect the presence of Parkinson's disease, many of these included computer-aided diagnosis, but they had their demerits, which involved low accuracy rate, costly and inefficient sometimes. We expect to learn problem-solving (which in our case is Detection of the disease), implemented through machine learning, which will enable us to undertake real-life problems ahead in our future.

#### **2.1.1. Problem**

Parkinson's disease is said to be the second most common neurodegenerative disorder. Although it is mostly seen in people after 60 years of age, it can also be encountered in the 40 s due to genetic reason you need to visit a doctor for its diagnostic, also there has not been any work done for early detection of the disease.

#### **Possible Solution**

This system is an android-based application that runs through a python server Django, where the machine learning model is deployed. The app enables the user to get a prediction about his status of Parkinson's disease. Using the app interface, user will create his profile and record his voice sample using a microphone and get result of the prediction. The system will either predict it as 0 for healthy, or 1 for affected. The results of predictions and profile will be maintained using database.

### **2.2. Deliverables and Development Requirements**

- Laptop
- Android Studio
- Java

*Detecting Parkinson's Disease Using Machine Learning*

- Python Environment
- Django
- Android based Smartphone
- SQLite
- Windows 10
- Star UML
- Microphone
- Internet Connection
- Microsoft Word 365
- Microsoft PowerPoint
- Anaconda
- Android Application
- Trained Machine Learning Algorithms
- Django server
- Database
- SRS
- SDD
- Thesis.

### 3. Requirement Analysis

#### 3.1. Use Cases Diagram(s)

Fig 3.1 represents the use case diagram of the application, and interaction of the user with the system.

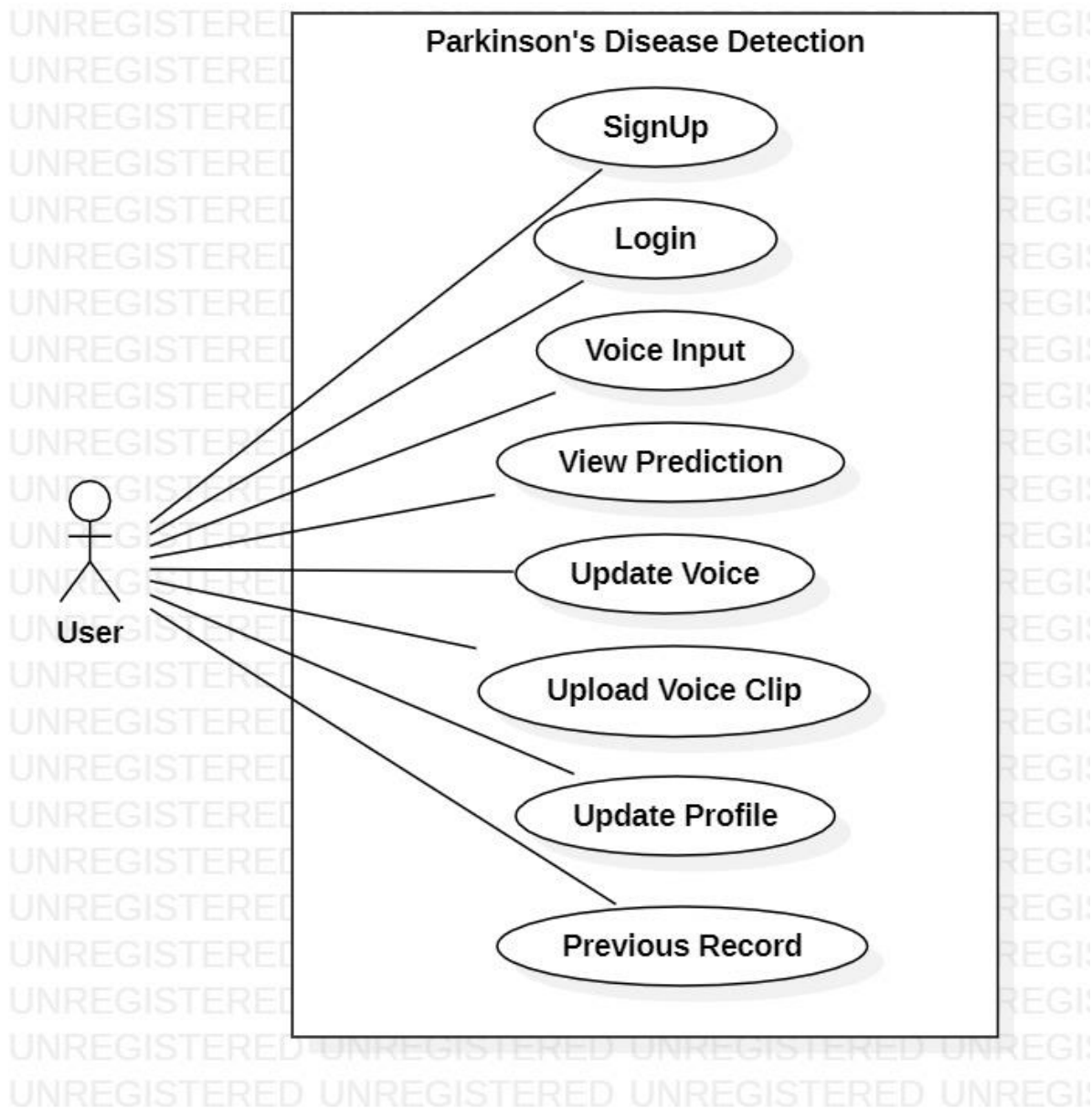


Figure 3.1: Use case diagram

### 3.2. Detailed Use Case

This section represents the detailed use case diagrams of the system.

#### 3.2.1. Signup:

Table 3.1 Represents the signup detailed Use Case.

Table 3.1: UC-1

Use Case ID:	UC-1
Use Case Name:	Sign Up
Actors:	Primary Actor : User
Description:	First, a user must register itself.
Trigger:	Join as a user
Pre-Conditions:	PRE1 - - Internet must be available PRE2- - The system must be idle. PRE3- The application must be installed on the device.
Post Conditions:	User registered successfully
Normal Flow:	System displays the Login.
Alternative Flows:	If in the basic flow the system validates an exception, an error message is displayed. The actor can re type the email in valid format or required information in correct format, at which point the use case ends
Exceptions:	E1-Registration unsuccessful. E2- Not connected to the server. E3- Invalid Email format.
Business Rules:	User should be able to complete the Register form.
Assumptions:	User is at the signup page.

### 3.2.2. Login:

Table 3.2 represents the detailed use case of Login.

Table 3.2: UC-2

Use Case ID:	UC-2
Use Case Name:	User Login
Actors:	Primary Actor : User
Description:	Users will enter their credentials and the system decides to give access or not by matching the criteria.
Trigger:	User indicates to login
Pre-Conditions:	PRE1- Internet must be available PRE2- The system must be idle showing login screen. PRE-3- User must be registered.
Post Conditions:	User has successfully logged in.
Normal Flow:	System displays the main menu.
Flows:	User clicks on the login button. User enters username and password. System validates the given data by sending it to the server. User logged into the system.
Exceptions:	E1- Incorrect username or password.
Business Rules:	User will be able to login.
Assumptions:	User is at the login page of the Application.

**3.2.3. Voice Input:**

Table 3.3 represents the detailed use case of voice operation performed in the application.

*Table 3.3: UC-3*

<b>Use Case ID:</b>	<b>UC-3</b>
Use Case Name:	Voice Input.
Actors:	Primary Actor : User
Description:	Users will enter his/her voice
Trigger:	User indicates the Get prediction button.
Pre-Conditions:	PRE1-User should be logged in. PRE2- System must be connected to the server. PRE3- System should be idle.
Post Conditions:	User can input his/her voice through the mic of the system.
Normal Flow:	System displays the voice recording, voice status.
Flows:	User clicks on the Get prediction button. User clicks to the start recording button. System validates the voice and shows the status of the voice being recorded.
Exceptions:	E1- voice is less than the described limit. E2- Voice input more than the limit would be exceeded.
Business Rules:	Users will be able to record voices.
Assumptions:	User is at the home page of the Application.

### 3.2.4. View Prediction.

Table 3.4 represents the detailed use case for the Disease status operation.

*Table 3.4: UC-4*

Use Case ID:	UC-4
Use Case Name:	View Prediction.
Actors:	Primary Actor : User
Description:	User can get the prediction of the disease through voice that has been recorded.
Trigger:	User Clicks on the predict button.
Pre-Conditions:	<p>PRE1-- User is logged into the system.</p> <p>PRE1-- User must have recorded or uploaded a voice recording to check the prediction.</p> <p>PRE2- The voice that has been recorded should be in the described limitations.</p>
Post Conditions:	System has predicted the disease successfully.
Normal Flow:	After clicking on the predict button the system shows the status of disease.
Alternative Flows:	None.
Exceptions:	<p>E1-No record found.</p> <p>E2- Voice must be in 15- 40 seconds time limitation.</p>
Business Rules:	User should be able to predict the disease.
Assumptions:	<p>User has logged in successfully.</p> <p>User is at the get prediction page.</p>



Business Rules:	User should be able to predict the disease.
Assumptions:	User has logged in successfully. User is at the get prediction page.

### **3.2.5. Previous Record:**

Table 3.5 represents the detailed use case of previous record operation.

*Table 3.5: UC-5*

Use Case ID:	UC-5
Use Case Name:	Previous Record.
Actors:	Primary Actor : User
Description:	User can check his previous record history of the disease .
Trigger:	User clicks the See previous information button.
Pre-Conditions:	PRE1- User must have logged into his own id. PRE 2-User must be on homepage of the application.
Post Conditions:	List of the record with date and time that when the previous user had predicted the disease along with the status of the disease.
Normal Flow:	User logged in successfully. User is at homepage of application. User clicked on the previous record button.
Alternative Flows:	None.
Exceptions:	E1-No previous record found.

Business Rules:	Users should be able to check the previous disease history.
Assumptions:	User logged in successfully into the system. User is at the home page.

### **3.2.6. Profile Management:**

Table 3.6 represents the detailed Use Case of profile management operation.

*Table 3.6: UC-6*

Use Case ID:	UC-6
Use Case Name:	Profile Management
Actors:	Primary Actor : User
Description:	Users can update the personal information such as name, email, or phone number.
Trigger:	User will click the Profile Management button.
Pre-Conditions:	PRE2- User must be registered to the system. PRE1- User must have a profile in the system record.
Post Conditions:	Changes updated successfully
Normal Flow:	After clicking the profile management button, a list of fields would be displayed that the user can change.
Alternative Flows:	None.
Exceptions:	E2- Enter the correct format. E1-Enter Correct info in the required field.

Business Rules:	Users should be able to update the profile.
Assumptions:	User has successfully registered to the system. User has successfully logged into the system. User is at the home page.

### **3.3. Functional Requirements**

- FR01 : User can create their account using sign-up page.
- FR02 : User can Login.
- FR03 : User can record voice and get prediction.
- FR04 : User can view prediction result, either as one for affected or 0 for healthy.
- FR05 : User can see previous history of predictions.
- FR06 : User can see the date and time of previous predictions results.
- FR07 : User can edit their first and last name
- FR08 : User can change their contact number.
- FR09 : User can logout of the application.

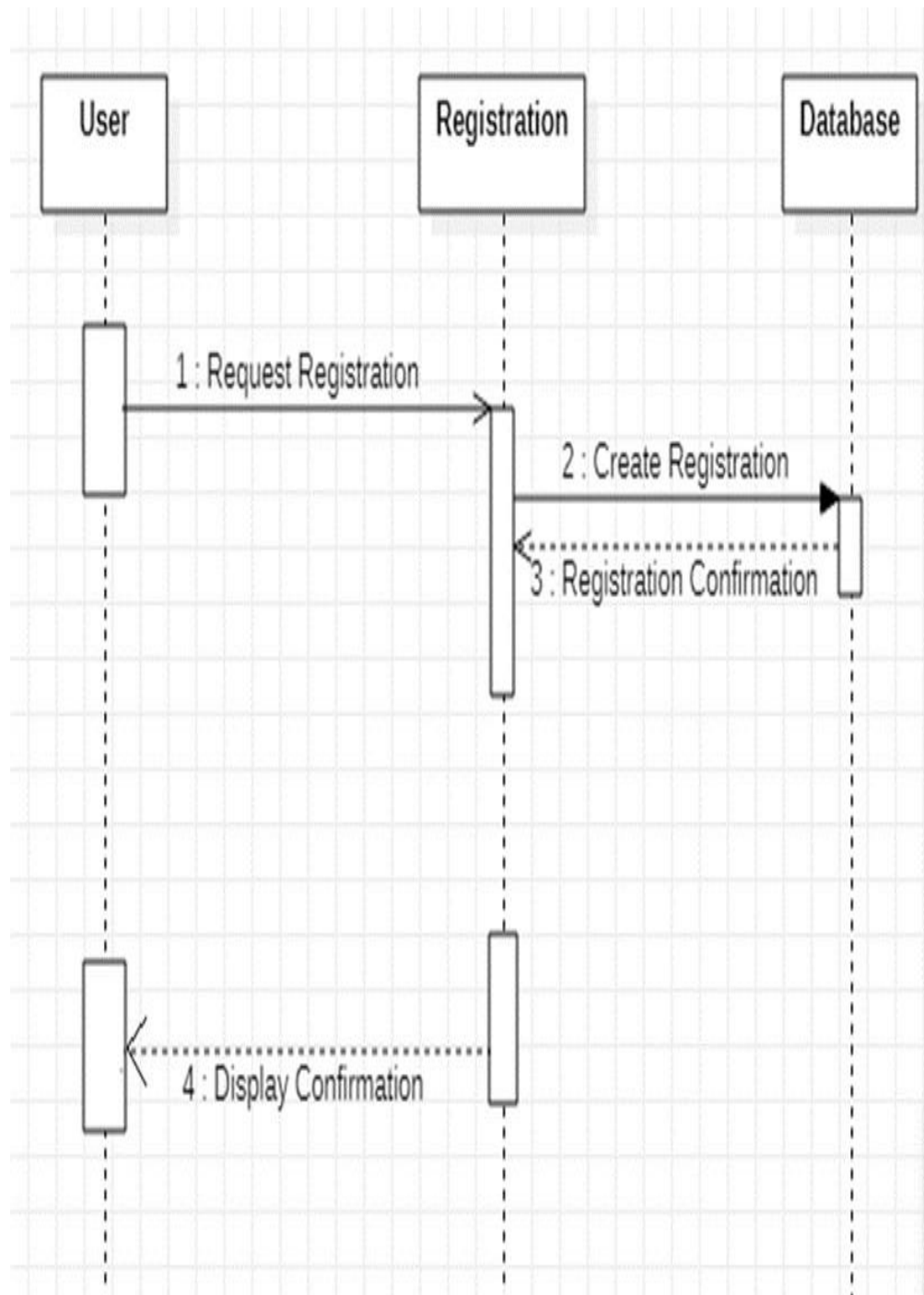
### **3.4. Non-Functional Requirements**

- NFR01 : The system must be compatible with the latest versions of Android.
- NFR02 : Users shall be prompted with empty fields check or improper format entered, at Sign-up and Login pages, respectively.
- NFR03 :Internet connection is mandatory.
- NFR04 :User shall be displayed with system status in between request and response time.
- NFR05 :User experience goals should t meet efficiently.
- NFR06 :Each username should be mapped to a unique id

## 4. Design and Architecture

This chapter included the important diagrams formulated in the SDD. These include System Sequence diagram, Class diagram

### 4.1. System Sequence Diagram



*Figure 4.1: Sign-up*

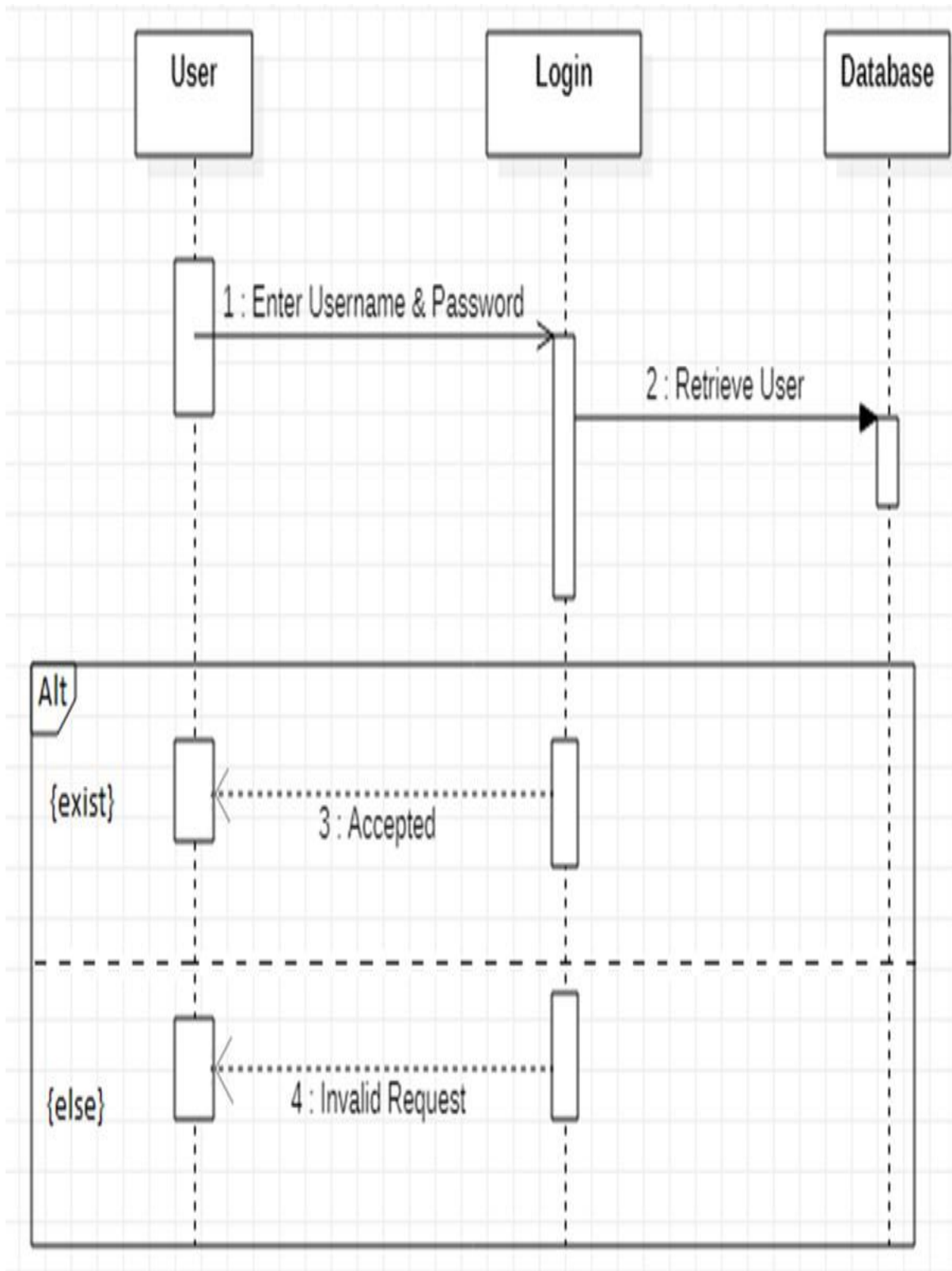
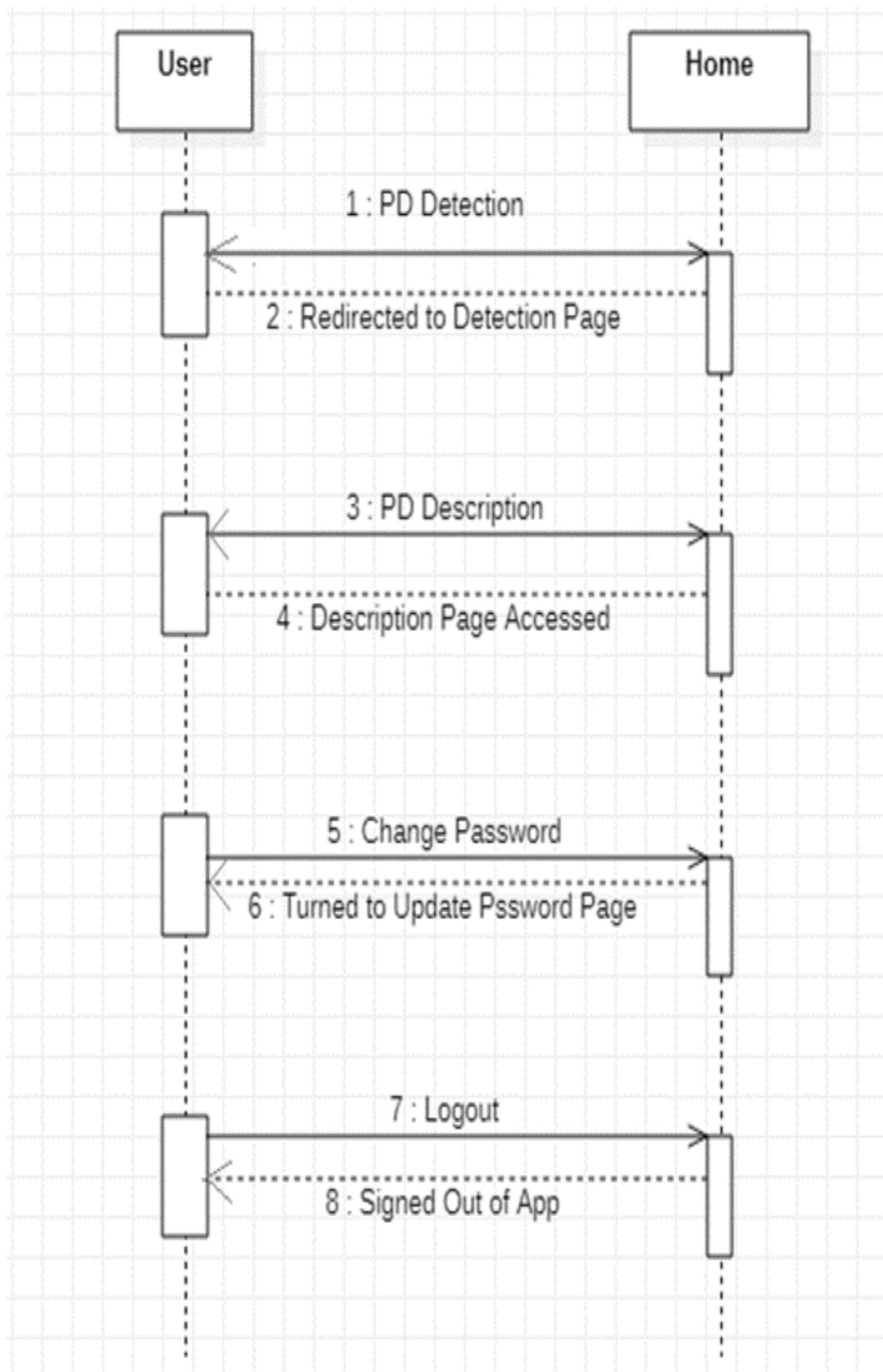


Figure 4.2: Login



*Figure 4.3: Homepage*

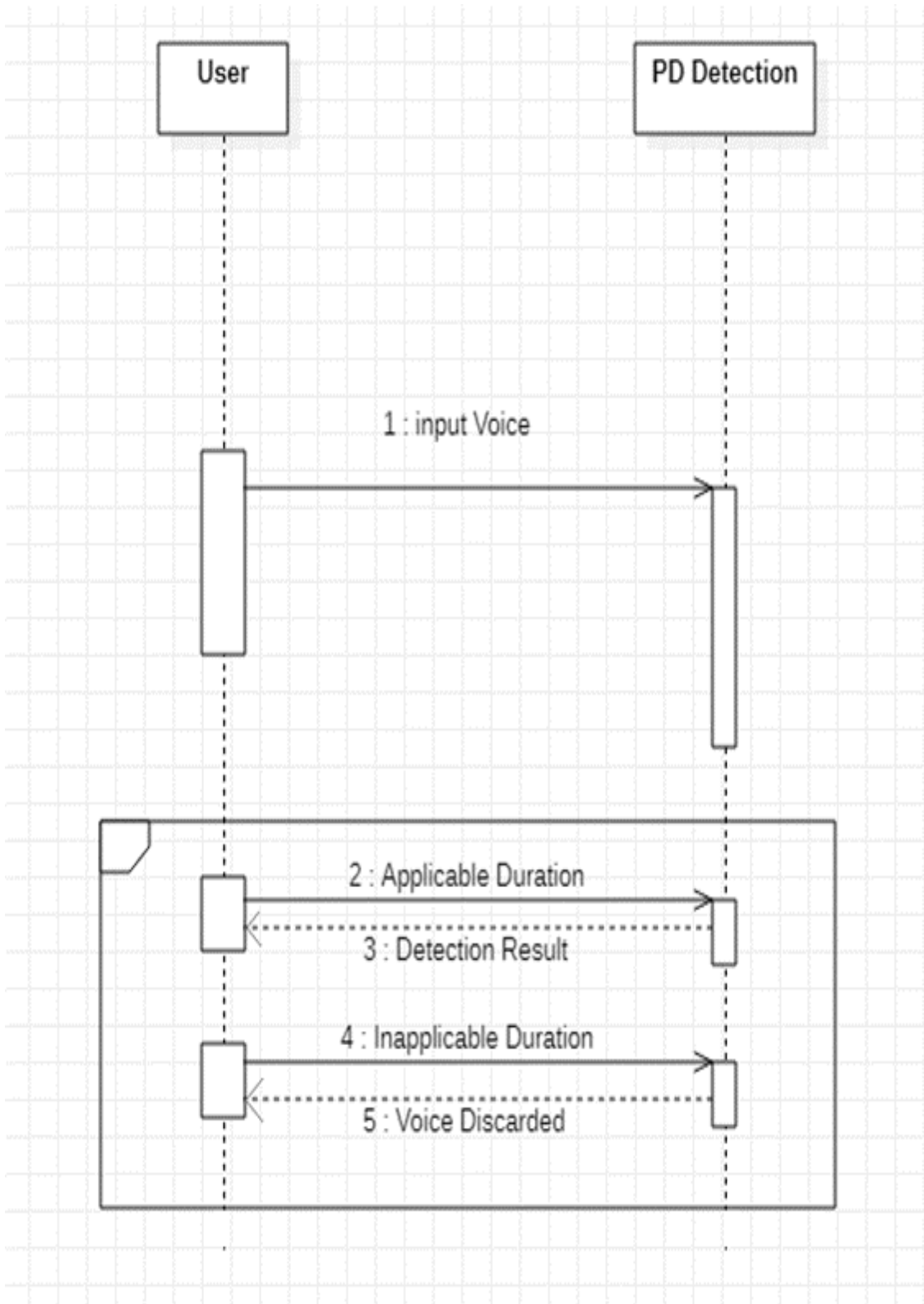


Figure 4.4: Voice record

## 4.2.Data Representation

### 4.2.1. Class Diagram

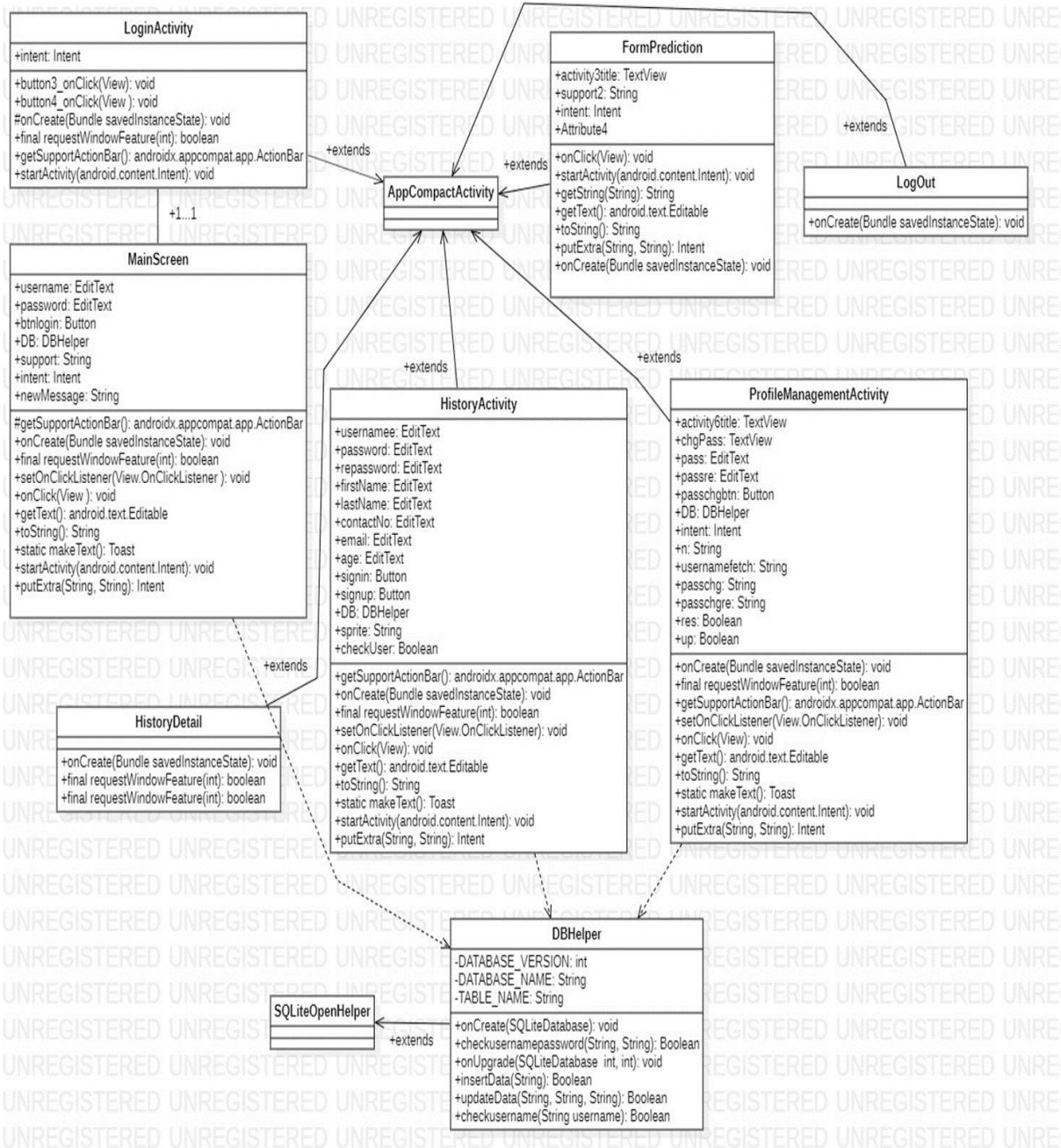
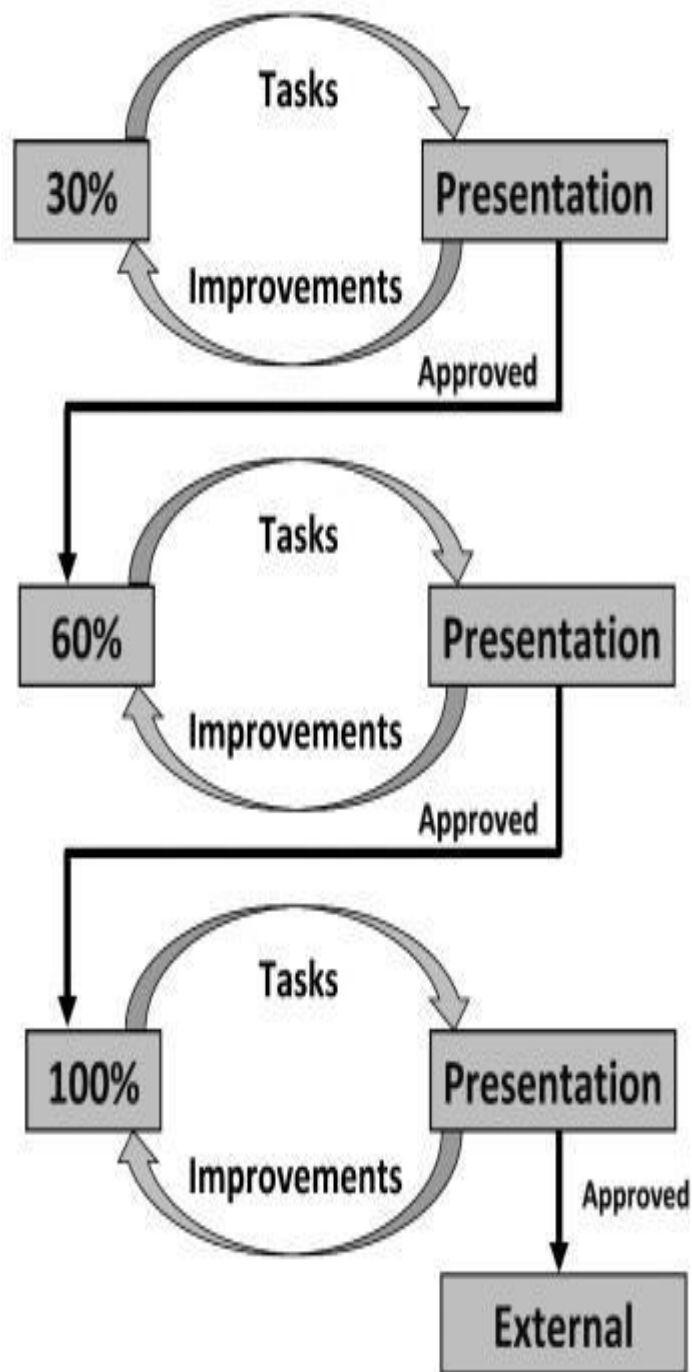


Figure 4.4: Class diagram

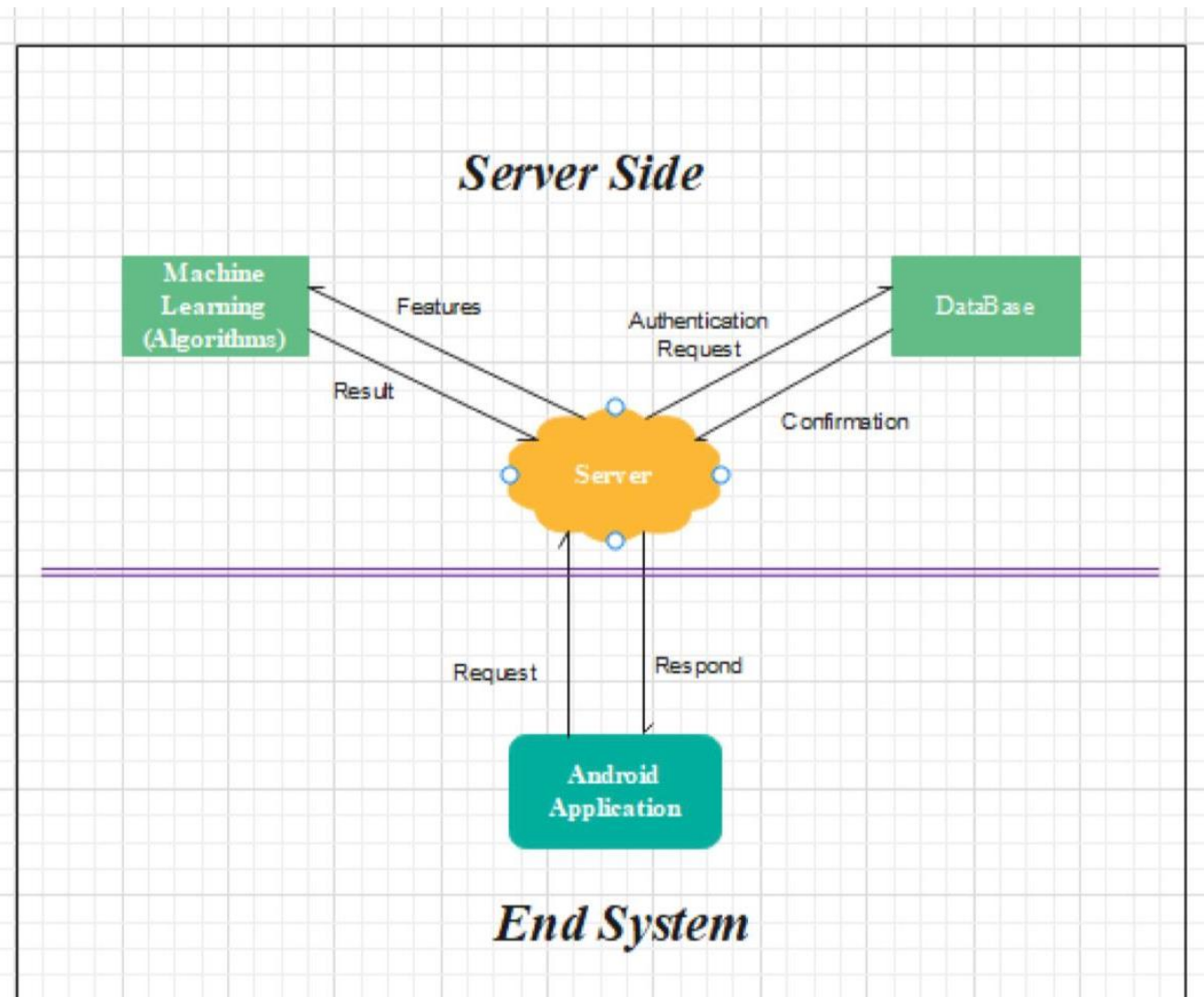


### 4.3.Design Model



*Figure 4.6: Design model*

#### **4.4. System Block Diagram**



*figure 4.7: System Block diagram*

## 5. Implementation

In this portion we will define the core libraries used as well as pseudocode of algorithms, along with concise description of each one.

### 5.1. Algorithm

Table 5.1: Important ML libraries

	Library	Function
1	NumPy	Used for mathematical functions and matrix processing
2	Pandas	It is used to read and store data in the form of dataframes. It helps in organizing the data
3	Scikit-learn	It had classical machine learning algorithms and functionalities to manipulate them
4	Matplotlib	It is used for plotting graphs and data analysis through mathematical values
5	Seaborn	It is based on Matplotlib, it helps in visualizing the data in a more descriptive from, for better visualization and understanding.

### **5.1.1 Random Forest:**

- For Random Forest, the dataset was loaded and then split into train and test. We import Random Forest Classifier from `sklearn.ensemble` to provide methods for both classification and regression via gradient boosted decision trees. Random forest manages the missing data so no hyper tuning is required.<sup>vi</sup>
- To check the relation between the numeric values and data frame heatmap is used.
- The association of the variables is demonstrated using Pearson's Correlation coefficient technique. Features that are highly correlated are at the top. Correlation value ranges from -1 - +1. This could be used to reduce the features that increase the efficiency of algorithm and accuracy in some respects. The table shows the parameters with their correlation values.  
vii
- To get more precise results we used `random_forest.fit` classifier used to provide higher accuracy through cross validation. Random forest will provide an output of the decision trees by averaging them and calculating mean more efficiently, precisely.
- 
- Predicted the model by entering values.
- The model yielded an accuracy of 93.8%.

### **5.1.2 KNN:**

KNN is a supervised machine learning algorithm used for classification purpose. KNN stores all available cases and classifies new cases or datapoints, based upon how its neighbors are classified. “K” is a parameter, which refers to the number of nearest neighbors to include in the majority voting process. Hence K is needed to be a positive integer. “K” is chosen by taking the square root of total data points to avoid high biased

or high variance.

- All-important libraries such as Pandas, Numpy, Scikit-Learn, MATPLOTLIB and Seaborn were used to import important functionalities.
- The dataset was split into input and output variables x and y, respectively.
- Output variable y will only have a ‘status’ column.
- Training size was set to 80% and Testing size was set to 20%.
- Performed the train-test split
- Pearson correlation coefficient was used to visualize the extent to which the features had impact on target variable “y.”
- from `sklearn.neighbors` imported the KNN Classifiers

- Predicted the model by entering values.
- The model yielded an accuracy of 87.17% using KNN where k was set to 7.
- Confusion Matrix was used to visualize important predictive analysis, such as: - recall, specificity, accuracy, and precision.
- Confusion Matrix[[ 25 16]  
[ 3 112]].

environment (Hadoop, SGE, MPI).

### **Procedure:**

We will load all the required libraries which are important. Then Parkinson disease's dataset was imported using `pd.read_csv`. Spearman correlation coefficient was used to visualize the extent to which the features had impact on target variable "y." After that **X** and **Y** variable will be **initialized**.

- a. Name and status attributes will be dropped while assigning rest of the attributes to X.
- b. Y will be assigned with status attribute only.

MinMaxScaler is used. Then dataset will be **divided** into **test(20%)** and **train(80%)**. **Scaler.transform** function will alter the data in a way that gives its distribution a mean value of 0 and a standard deviation of one. Confusion Matrix was used to visualize important predictive analysis, such as: - recall, specificity, accuracy, and precision. Classification report is calculated showing values of precision, recall and F1- score. I.e 0.86,0.86,0.86 respectively for 0 and 0.97,0.97,0.97 respectively for one. Accuracy of both **test** and **train data** is calculated. In the ultimate step model is evaluated to make predictions.

## **5.2.External APIs**

All the APIs are used for the Django Rest framework.

Table 5.1 shows the Details of APIs used in the project.

*Table 5.2: APIs used*

<b>Name of API</b>	<b>Description of API</b>	<b>Purpose of usage</b>	<b>List down the function/class name in which it is used</b>
API root	This API uses the default root view to the directed api with JSON format.	It is used to be directed towards main api for prediction.	/api/(Django)

API prediction	Prediction API would allow to get the voice file from system and to post the status of disease	To send the voice parameters and to post the disease status through the server	Formprediction.java
API admin	It provides the functionality to manage accounts.	It manages the admin account they can have access to the rest of get and post and it can manage the dashboard.	/api/admin(Django)
API login	API login used for the access of admin.	This api manages to secure the root access of the dashboard.	/api/login(Django)

### 5.1.3. Support Vector Machine:

Support vector machines (SVM) is a supervised learning approach for classifying, regression, and detecting outliers.

ixx

#### Benefits of support vector machines:

1. In high-dimensional spaces, it works well.
2. When the number of dimensions exceeds the number of samples, the method is still successful.
3. It is memory efficient because it uses a subset of training points (called support vectors) in the decision function.
4. Different Kernel functions may be given for the decision function, making it versatile. Common kernels are included, however custom kernels can also be specified.

#### Procedure:

First, we will **load** all the required **libraries** which are important. After importing the

libraries Parkinson disease's **dataset** was imported using `pd.read_csv`. Then we will use several functions to overview statistical analysis of the attributes. **Shape** which returns the shape of an array. An array's form is a tuple with the number of its members. (i.e., number of records and feature.)

- A. **.info()** function is used to get a concise summary of the dataframe.
- B. **.dtype** function will return the data type of all the attributes present in dataset.
- C. **.isnull ()** to check if there is any null value.
- D. **.describe()** to view the detailed description of attributes.

**X** and **Y** variable will be **initialized**. Name and status attributes will be dropped while assigning rest of the attributes to X. Y will be assigned with status attribute only. Pearson correlation coefficient was used to visualize the extent to which the features had impact on target variable "y." Then dataset will be **divided** into **test(20%)** and **train(80%)**. **Scaler.transform** function alters the data in a way that gives its distribution a mean value of 0 and a standard deviation of **svm.SVC** specifies the kernel type that is used in algorithm. Confusion Matrix was used to visualize important predictive analysis, such as: - recall, specificity, accuracy, and precision. Classification report is calculated showing values of precision, recall and F1- score. I.e 0.71,0.62,0.67 respectively for 0 and 0.91,0.94,0.92 respectively for Accuracy of both **test** and **train data** is calculated. In the ultimate step model is evaluated to make predictions.

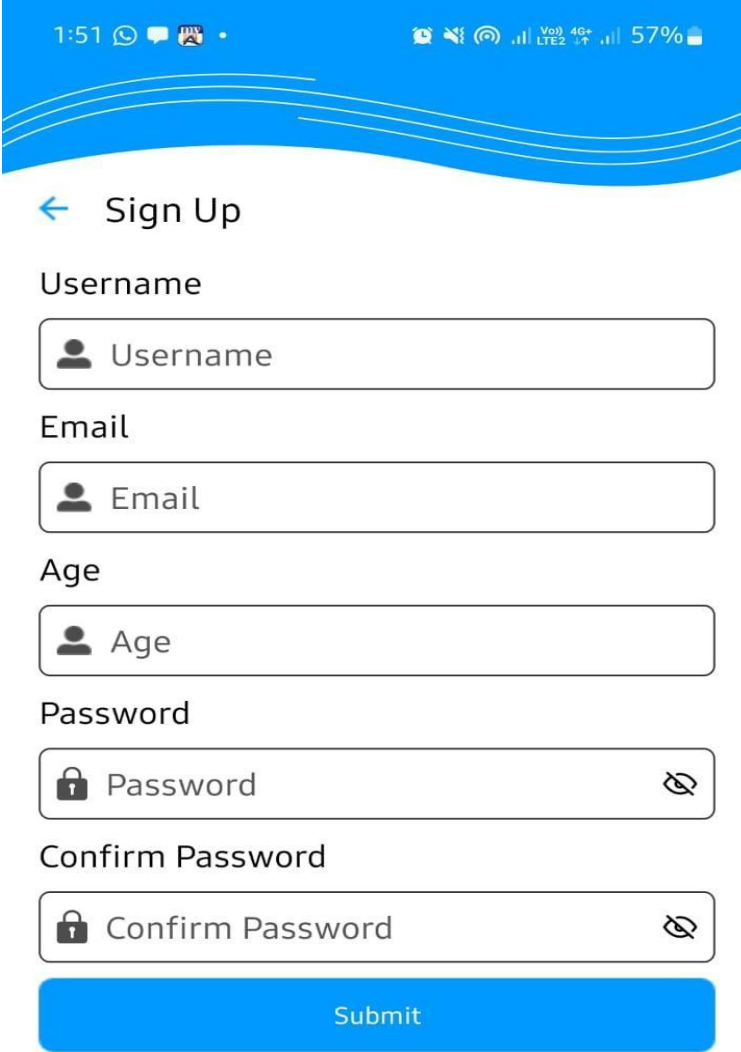
#### **5.1.4. XGBoost**

XGBoost is a distributed gradient boosting toolkit that is optimized for efficiency, flexibility, and portability. It uses the Gradient Boosting framework to implement machine learning algorithms. XGBoost uses parallel tree boosting (also known as GBDT or GBM) to tackle a variety of data science issues quickly and accurately. The same algorithm may tackle problems with billions of examples in a distributed

## **a. User Interface**

Information with descriptions regarding the user interface.

### **i. SignUp**



1:51 [notification icons] [status icons] 57%

← Sign Up

Username

[user icon] Username

Email

[user icon] Email

Age

[user icon] Age

Password

[lock icon] Password [eye icon]

Confirm Password

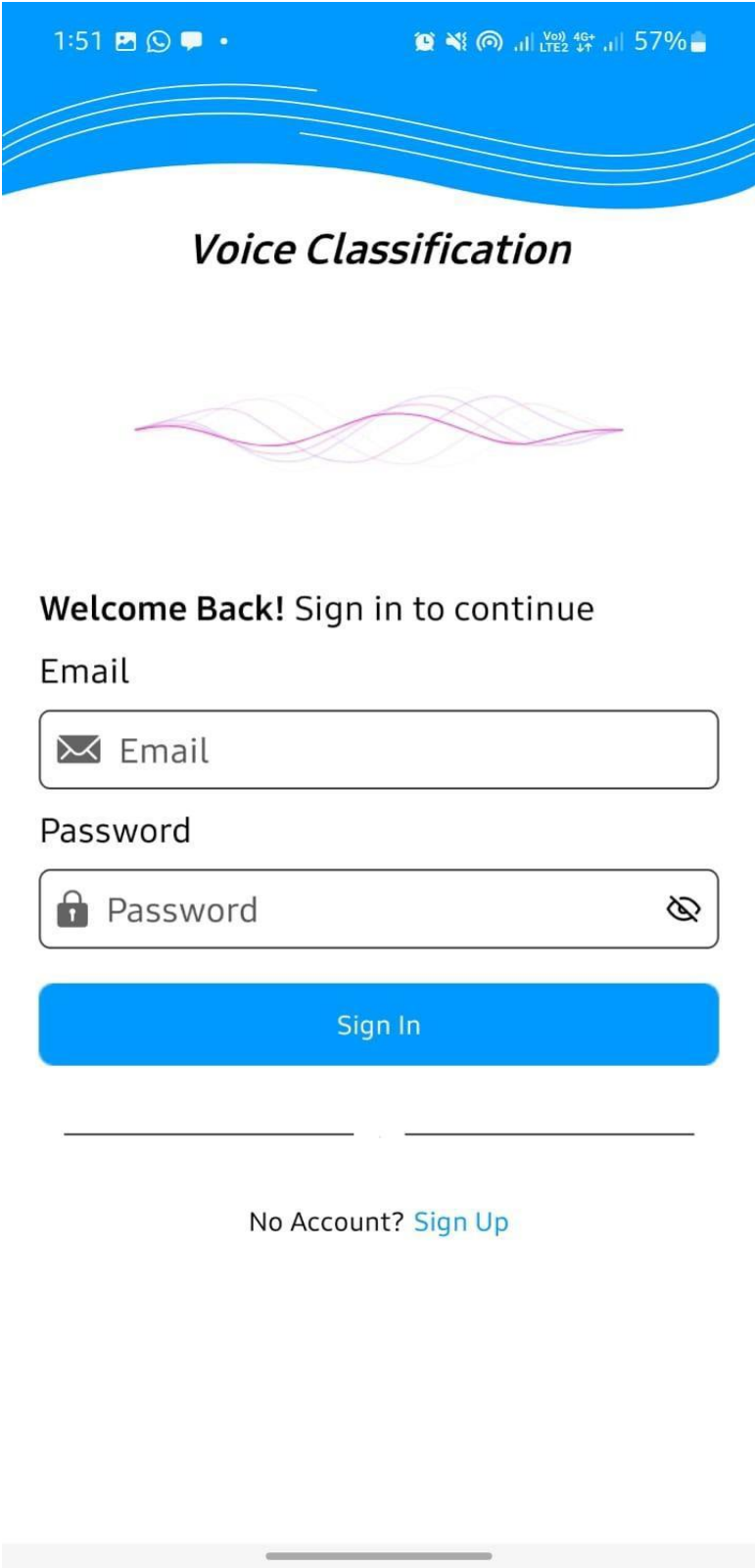
[lock icon] Confirm Password [eye icon]

Submit

*Figure 5.1: Sign-Up*



**ii. Login**



The image shows a mobile application interface for a login screen. At the top, there is a blue header with the title "Voice Classification" in white, italicized font. Below the header is a decorative graphic of overlapping wavy lines in shades of purple and pink. The main content area is white and contains the text "Welcome Back! Sign in to continue" in bold black font. Below this text are two input fields: "Email" and "Password". The "Email" field has an envelope icon on the left. The "Password" field has a lock icon on the left and an eye icon on the right to toggle visibility. Below the input fields is a large blue button with the text "Sign In" in white. At the bottom, there is a horizontal line and the text "No Account? Sign Up" in blue, where "Sign Up" is a link.

1:51 5G+ 57%

## Voice Classification

Welcome Back! Sign in to continue

Email

Email

Password

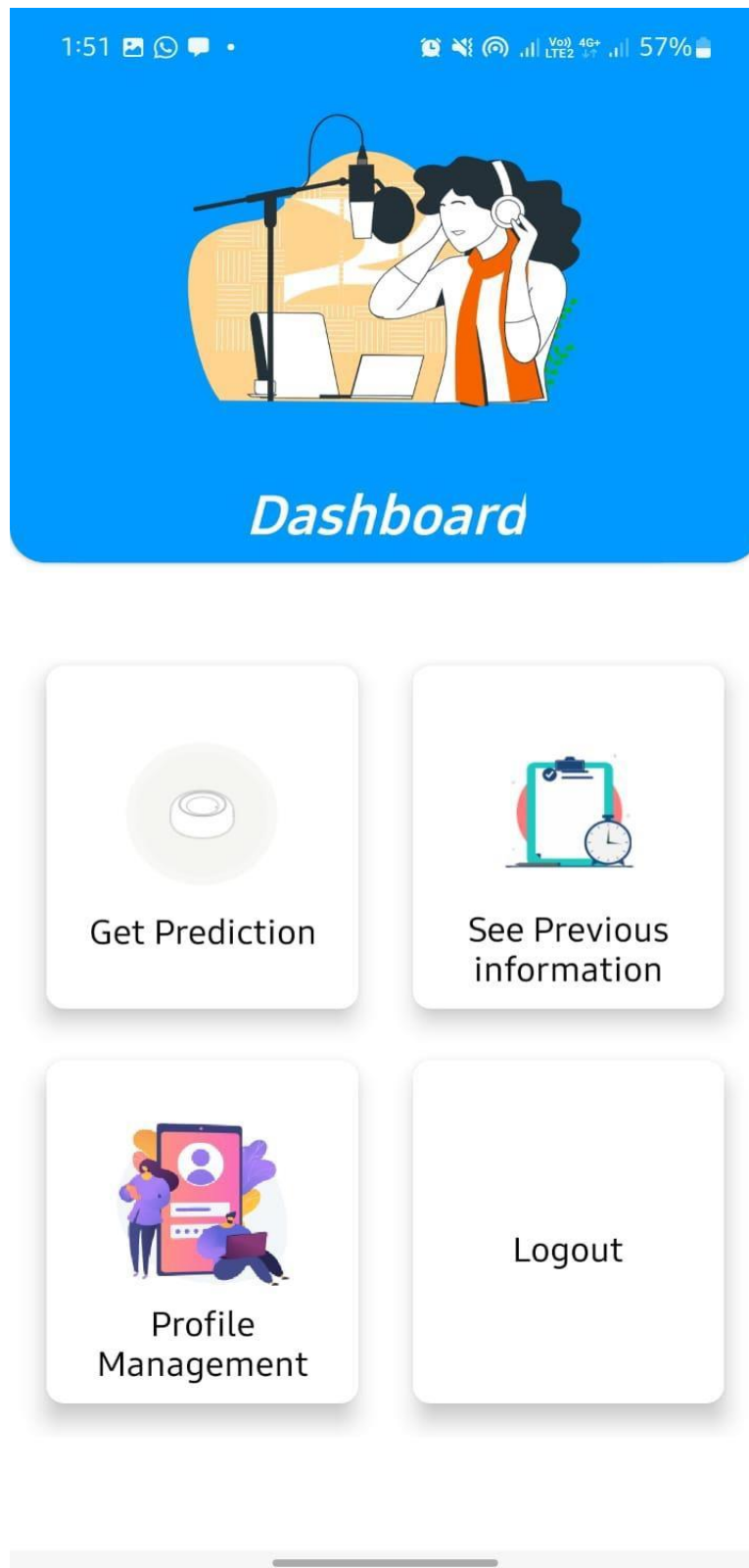
Password

Sign In

No Account? [Sign Up](#)

*Figure 5.2: Login*


**iii. Homepage**



*Figure 5.3: Homepage*

#### iv. Profile Management


1:52




57%

Profile management


First name

 babar


Last name

 babar


Username / Referral code

 babar

Email

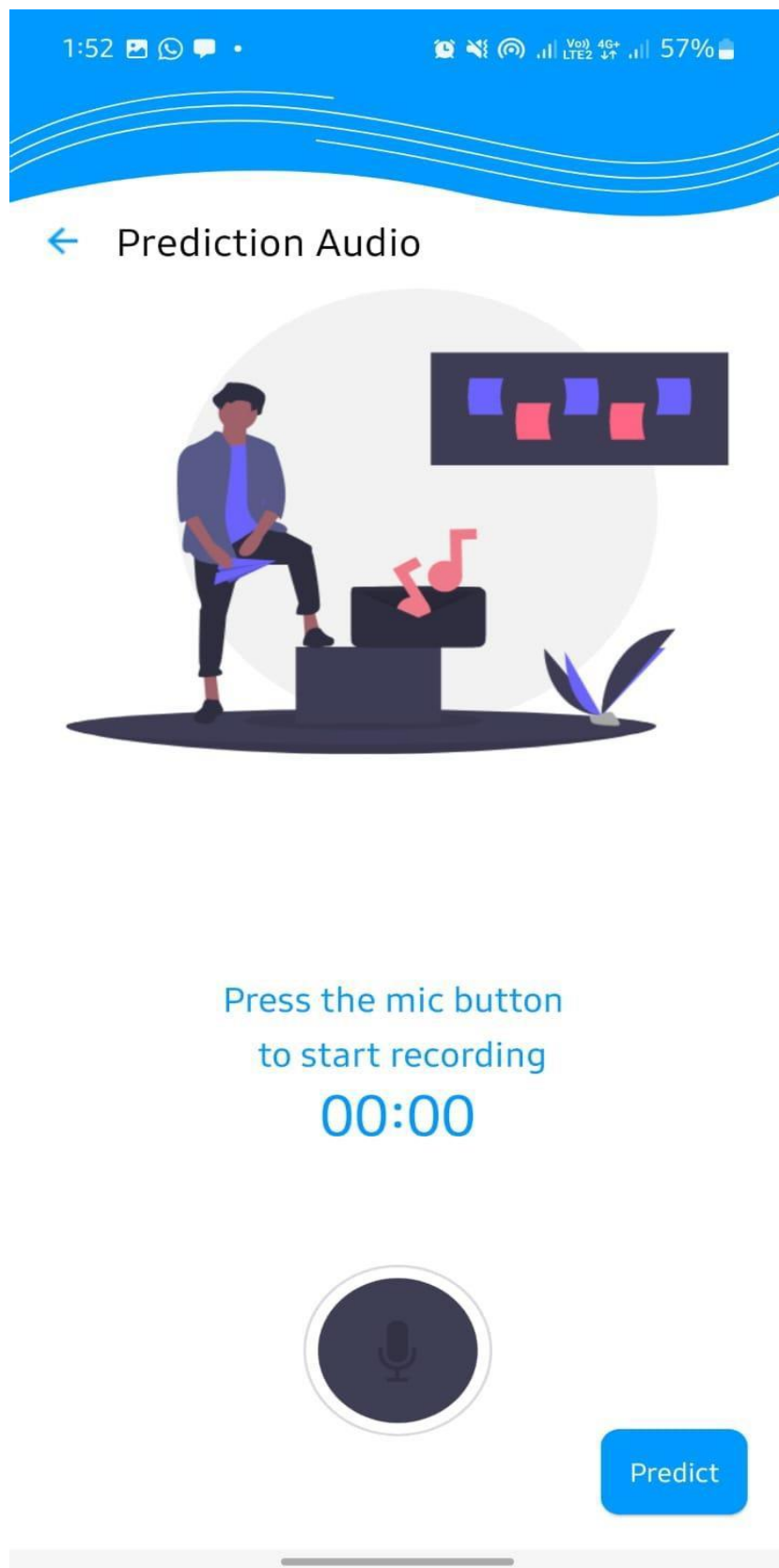
 babar@gmail.com

Phone no.

 +91 Phone no.

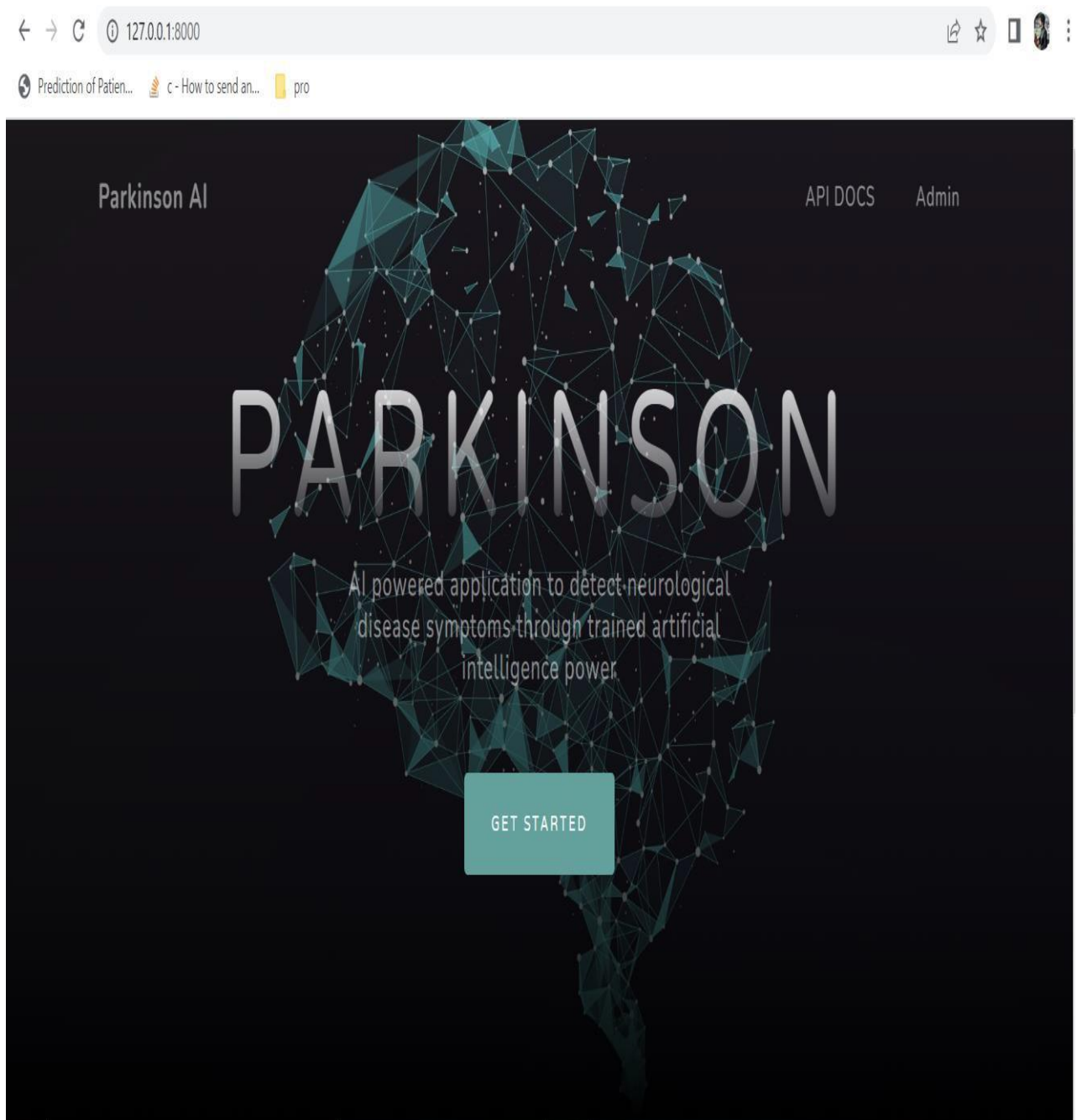
*Figure 5.4: Profile Management*

**v. Get Prediction**



*Figure 5.5: Audio Prediction*

**vi. Server(Django) Homepage.**



*Figure 5.6: Server Homepage*

**vii. Server(Django) Dashboard.**

The screenshot displays the Django Admin interface for managing user accounts. The top navigation bar shows 'Root Access' and a welcome message for the 'ADMIN' user. The breadcrumb trail indicates the current location: 'Home > Accounts > Users Accounts'. The left sidebar contains various management links, with 'Users Accounts' highlighted. The main area is titled 'Select User Account to change' and features a search bar, an action dropdown, and a table of users. The table lists two users: 'admin' and 'babar', both with email addresses ending in '@gmail.com' and marked as staff. A filter sidebar on the right allows filtering by staff status, superuser status, and active status.

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
<input type="checkbox"/> admin	admin@gmail.com			<input checked="" type="checkbox"/>
<input type="checkbox"/> babar	babar@gmail.com			<input checked="" type="checkbox"/>

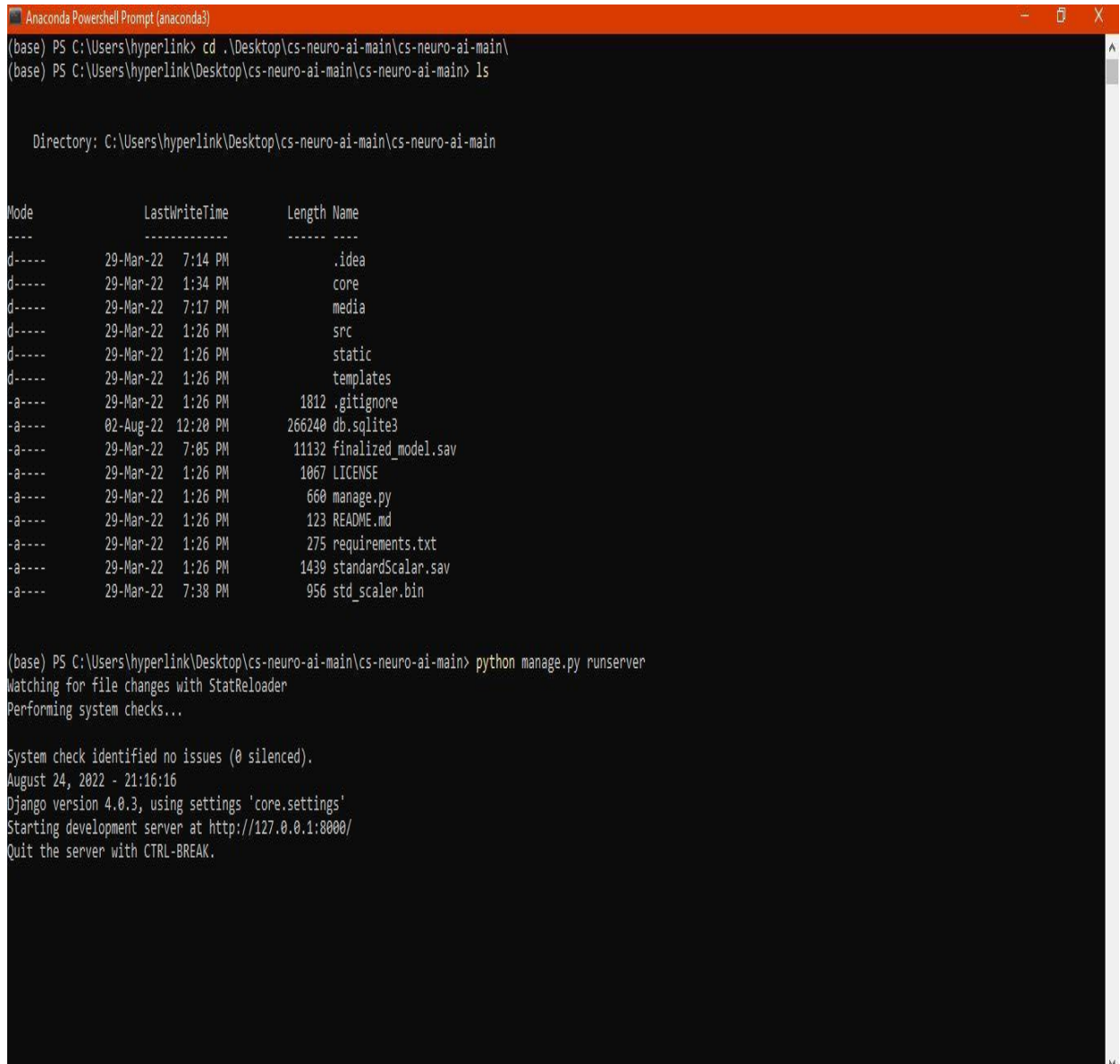
Figure 5.7: Server Dashboard

## b. How to run the Application

The application is developed using Android Studio (JDK) and Python 3.9.

### i. Python

Using Anaconda PowerShell, we ran the following commands to access the application directory and ran the server where machine learning model was hosted. 127.0.0.1, was used to communicate the user with its local machine. TCP Port 8000 was assigned to it for web software.

A screenshot of the Anaconda PowerShell Prompt window. The window title is "Anaconda PowerShell Prompt (anaconda3)". The terminal shows the following commands and output:

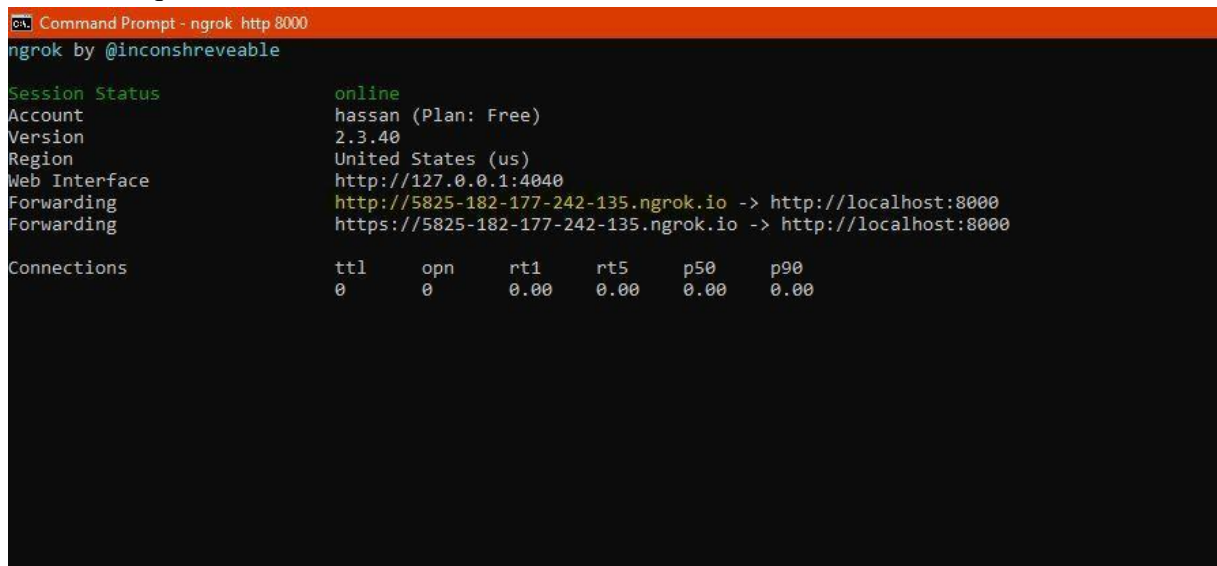
```
(base) PS C:\Users\hyperlink> cd .\Desktop\cs-neuro-ai-main\cs-neuro-ai-main\  
(base) PS C:\Users\hyperlink\Desktop\cs-neuro-ai-main\cs-neuro-ai-main> ls  
  
Directory: C:\Users\hyperlink\Desktop\cs-neuro-ai-main\cs-neuro-ai-main  
  
Mode                LastWriteTime         Length Name  
----                -  
d-----          29-Mar-22   7:14 PM          .idea  
d-----          29-Mar-22   1:34 PM          core  
d-----          29-Mar-22   7:17 PM          media  
d-----          29-Mar-22   1:26 PM          src  
d-----          29-Mar-22   1:26 PM          static  
d-----          29-Mar-22   1:26 PM          templates  
-a-----          29-Mar-22   1:26 PM        1812 .gitignore  
-a-----          02-Aug-22  12:20 PM    266240 db.sqlite3  
-a-----          29-Mar-22   7:05 PM    11132 finalized_model.sav  
-a-----          29-Mar-22   1:26 PM     1067 LICENSE  
-a-----          29-Mar-22   1:26 PM      660 manage.py  
-a-----          29-Mar-22   1:26 PM      123 README.md  
-a-----          29-Mar-22   1:26 PM      275 requirements.txt  
-a-----          29-Mar-22   1:26 PM    1439 standardScalar.sav  
-a-----          29-Mar-22   7:38 PM      956 std_scaler.bin  
  
(base) PS C:\Users\hyperlink\Desktop\cs-neuro-ai-main\cs-neuro-ai-main> python manage.py runserver  
Watching for file changes with StatReloader  
Performing system checks...  
  
System check identified no issues (0 silenced).  
August 24, 2022 - 21:16:16  
Django version 4.0.3, using settings 'core.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.
```

Figure 5.8: Anaconda PowerShell

## ii. Ngrok

Ngrok was used to connect the internet with the local server ports.

The status was online, and a forwarding address was assigned for forwarding requests.



```
Command Prompt - ngrok http 8000
ngrok by @inconshreveable

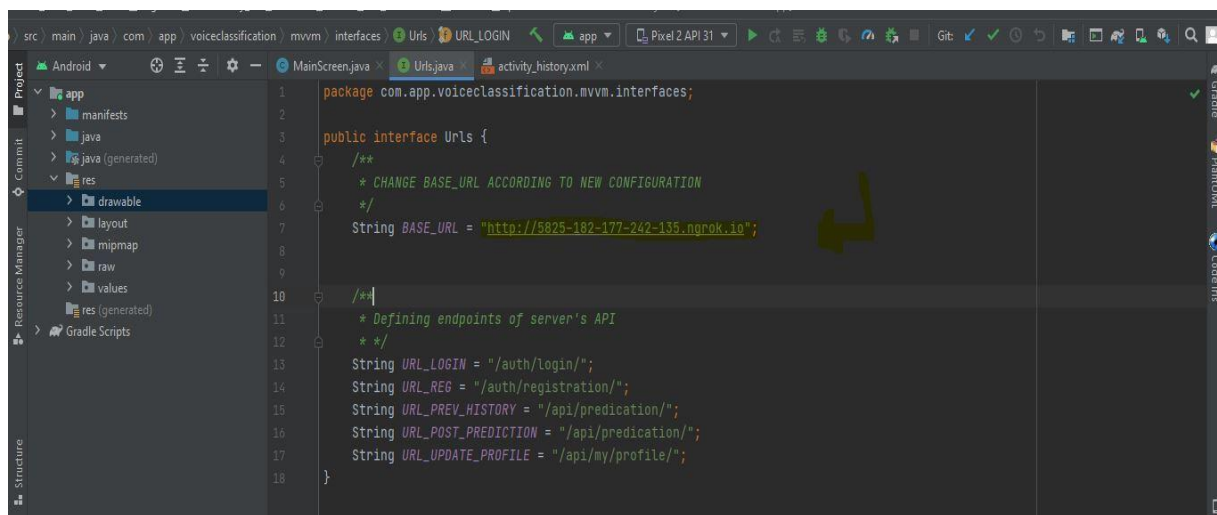
Session Status      online
Account             hassan (Plan: Free)
Version             2.3.40
Region              United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding           http://5825-182-177-242-135.ngrok.io -> http://localhost:8000
                   https://5825-182-177-242-135.ngrok.io -> http://localhost:8000

Connections
  ttl    opn    rt1    rt5    p50    p90
   0      0     0.00   0.00   0.00   0.00
```

Figure 5.9: Ngrok

## iii. Android

Android 12 with API level 31 was used for development of application. The first forwarding address obtained from ngrok was used to forward requests from android app. We copied the address and pasted it in the Urls.java page, as the base URL.



```
src > main > java > com > app > voiceclassification > mvvm > interfaces > Urls
package com.app.voiceclassification.mvvm.interfaces;

public interface Urls {
    /**
     * CHANGE BASE_URL ACCORDING TO NEW CONFIGURATION
     */
    String BASE_URL = "http://5825-182-177-242-135.ngrok.io";

    /**
     * Defining endpoints of server's API
     */
    String URL_LOGIN = "/auth/login/";
    String URL_REG = "/auth/registration/";
    String URL_PREV_HISTORY = "/api/predication/";
    String URL_POST_PREDICTION = "/api/predication/";
    String URL_UPDATE_PROFILE = "/api/my/profile/";
}
```

Figure 5.10: Android Studio



## 5. Testing and Evaluation

Testing is performed to validate the results against the defined set of functionalities. This chapter includes the Manual Testing method, every unit is treated as an independent system initially, followed by functionality testing of the core units and then performing Integration testing to check if the system performs as desired.

### a. Manual Testing

It is performed by the developer himself or separate testing team, in our case it was carried by us, using different roles.

#### i. System testing

Testing must be done to make sure the system is operating as planned after it has been successfully developed. This is done to make sure the system complies with the previous requirements. Additionally, system testing will assist in locating errors that may be concealed from the user. Unit testing, functional testing, and integration testing are a few of the several forms of testing. Before it is made available to users, testing must be finished.

#### ii. Unit Testing

Once the system has been successfully developed

##### 1. Unit Testing 1:

Login as FYP Committee as shown in Table 5.1

##### Testing Objective:

To ensure the login form is working correctly.

Table 6.1: Login Unit Testcase

No.	Test case/Test script	Attribute value and	Expected result	Result
1.	Verify login once user click on the 'Login' button on login field with correct input data.	Email: abc@gmail.com  Password: abc22	Successfully log into the main page of the system as user	Pass

## **2. Unit Testing 2:**

Sign Up as new user as shown in Fig:5.2

### **Testing Objective:**

To make sure that the Sign-Up page is working properly with all possibilities.

*Table 6.2: Sign Up Unit Testcase*

<b>No.</b>	<b>Test case/Test script</b>	<b>Attribute value and</b>	<b>Expected result</b>	<b>Result</b>
1.	Verify new sign-up after clicking on the 'Submit' button on sign-up form with correct input data.	Username: shoaib Email: shoaib@gmail.com Age: 39 Password: shoaib0992 Confirm Password: shoaib0992	A new user is successfully created	Pass

## **3. Unit Testing 3:**

Check if Prediction Page is working accordingly shown in Fig 5.3

### **Testing Objective:**

User is getting the prediction for his voice input.

Table 6.3: Get-Prediction Unit Testcase

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Verify that user is getting the Prediction	1.Click on record button  2.Stop the recording after 15 secs  3.Click on Predict Button	Prediction Result as 1 or 0.	Pass

### iii. Functional Testing

After the unit testing, the functional testing will take place. Each module's functioning is tested during this functional testing. This is done to guarantee that the system generated complies with the criteria and specifications

- Functional Testing 1:** Login with separate roles as shown in Table 5.3 **Objective:** To ensure that the correct page with the correct navigation bar is loaded.

Table 6.4: Login Functional Test Case

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Verify user login into the main page of the app which is the dashboard.	Email: abc@gmail.com  Password: abc22	User will log into the main page of the system as a user and every subsection is visible along-with the navigation bar.	Pass

2.	Verify by entering missing fields	Email: Password:	Prompt: Field Empty	Pass
3	Verify by entering both incorrect fields	Email:admin@gmail.com Password: 12345	Prompt: Incorrect field	Pass
4	Verify by entering one field incorrect .	Email:admin@gmail.com Password: abc22	Prompt: Incorrect field	Pass

## 2. Functional Testing 2:

Signup with separate roles as shown in Table 5.3

### Objective:

To ensure that the correct page with the correct navigation bar is loaded.

*Table 6.5: Sign-Up Functional Test Case*

No.	Test case/Test script	Attribute and value	Expected result	Result

1.	Verify that after creation of new user the user is directed to the login page to further proceeds	Username: shoaib Email: shoaib@gmail.com Age: 39 Password: shoaib0992 Confirm Password: shoaib0992	New user created and directed to the log-in activity page successfully	Pass
2..	Verify by clicking on Submit button without any data entered,	1.Open the app 2.Click on Submit Button None	Prompt :Field Empty	PASS
3.	Verify by entering missing fields	1.Enter Username 2.Enter email 3.Enter Age 4.Enter Password 1.Nikolas Tesla 2.NikolasTesla 22@gmail.com 3.78 4.123	Prompt :Fields Invalid Please Check	PASS

4.	Verify by entering incorrect fields	1.Enter Username 2.Enter email 3.Enter Age 4.Enter Password 5.Renter Password  1.Joe root 2.Joeroot123  gmail 3.2500 4.12345 5.13245	1. Prompt: Enter a valid username(No spaces allowed)  2. Prompt: Enter a valid email address  3. Prompt  Prompt :Fields Invalid Please Check	1.PASS 2.PASS 3. FAIL 4.PASS 5.PASS
----	-------------------------------------	--	--	---

#### **iv. Integration Testing**

Table 5.4 shows the integration testing

*Table 6.6: Integration Test Case*

No.	Test case/Test script	Attribute and value	Expected result	Result
1.	Login as user	Username: abc@gmail.com Password: abc22	Login successful and the homepage of application with its navigation bar is loaded. And main page with all the categories to access will appear	Pass

2.	Get Prediction Page selected	Mic button is enabled, and recording started	After a user records voice for more than fifteen secs and stops, selects the predict button and is redirected to the history page followed by the result of the prediction.	Pass
3.	See Previous Information	Clicks on See Previous Information Section	List of previous results of the user is shown	Pass
4.	Profile Management	Clicks on Profile Management Section. First name: Mark Last name: Burger Phone no: +92 123456789	User is directed to the Profile Management Page, updates record successfully	Pass
5.	Logout	Clicks on Logout Section	User is logged out and directed back to the login screen for new login	Pass

## **6. Conclusion and Future Work**

### **a. Conclusion**

- This project is going to help in understanding and solving the challenges involved in the detection of Parkinson's disease.
- The application will provide an ease for users to conduct a test easily at their home and it will be very user friendly.
- The Application provides early disease detection.
- Using computer aided technology for prediction of results makes it more accessible.

### **b. Future Work**

In the future, this project will be beneficial for doctors and individuals who want to detect Parkinson's disease accurately and easily. Further process the voice with better equipment and sound proof labs may result in yielding more precise results, making it a standalone technology of its type.

A much broader concept of machine learning is Deep Learning such as Neural Networks. It works in the form of layers to produce an output. Using voice-based input and then processing it through deep learning will improve the effectiveness and efficiency of the system. Deploying the system on a real-time environment will add up to its productiveness. Also achieving automation at various steps in the system will make it a self-sufficient app and may not have to rely on an interdependent process to initiate it.

This system also paves way for Computer-aided diagnosis, and research related to it. Using already developed systems such as CAD using brain imaging and walking test, in collaboration with this system to achieve precision and exploring novel approaches will be possible.

Developing the application further, keeping in view the modern-day requirements, it would be a very cost friendly option for concerned people over their Parkinson's disease status.



## 7. References

- 
- i <https://archive.ics.uci.edu/ml/datasets/Parkinsons>
  - ii Goetz CG. The history of Parkinson's disease: early clinical descriptions and neurological therapies. Cold Spring Harb Perspect Med. 2011 Sep;1(1):a008862. doi: [10.1101/cshperspect.a008862](https://doi.org/10.1101/cshperspect.a008862). PMID: 22229124; PMCID: PMC3234454.
  - iii <https://www.mayoclinic.org/diseases-conditions/parkinsons-disease/symptoms-causes/syc-20376055>
  - iv Adams WR. High-accuracy detection of early Parkinson's Disease using multiple characteristics of finger movement while typing. PLoS One. 2017 Nov 30;12(11):e0188226. doi: [10.1371/journal.pone.0188226](https://doi.org/10.1371/journal.pone.0188226). PMID: 29190695; PMCID: PMC5708704.
  - v Juutinen M, Wang C, Zhu J, Haladjian J, Ruokolainen J, Puustinen J, Vehkaoja A. Parkinson's disease detection from 20-step walking tests using inertial sensors of a smartphone: Machine learning approach based on an observational case-control study. PLoS One. 2020 Jul 23;15(7):e0236258. doi: [10.1371/journal.pone.0236258](https://doi.org/10.1371/journal.pone.0236258). PMID: 32701955; PMCID: PMC7377496.
  - vi <https://www.topcoder.com/thrive/articles/understanding-random-forest-and-hyper-parameter-tuning>
  - vii <https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/>
  - viii <https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989>
-