Health 360
Project Report

1. Introduction

Health 360 Platform is a mobile application that provides patients access to health-related services, resources, and information. The application can include various features, including health surveys, symptom trackers, medication reminders, appointment scheduling, telemedicine services, and access to health professionals.

Patients can use the platform to complete health surveys and assessments, which can help them monitor their health status and identify potential health problems. The platform can also offer personalised recommendations and resources based on the patient's health data.

Over time, new tasks or features can be added to the platform, such as integration with wearable devices or access to virtual health coaching services. These features can help patients manage their health more effectively and stay engaged with their health over time.

Overall, a Digital Health Diary can provide patients with a convenient and accessible way to manage their health and help healthcare providers deliver more effective and personalised care.

2. Tech Stack

1. **Frontend**:

   Flutter 3.3.9: (https://flutter.dev/)

   Flutter is an open-source mobile application development framework created by Google. It allows developers to build high-performance, beautiful, and natively compiled applications for mobile, web, and desktop platforms from a single codebase. Flutter has gained popularity among developers for its ease of use, flexibility, and ability to produce high-quality mobile apps for multiple platforms with a single codebase.

   Dart programming language:

   Flutter uses the Dart programming language, also developed by Google, and provides a rich set of pre-built widgets that enable developers to build user interfaces quickly and easily. The framework enables fast development cycles, quick iteration, and high-quality results.

Android Studio:

Android Studio is an Integrated Development Environment (IDE) designed specifically for developing Android applications. It was developed by Google and is the official IDE for Android app development.

Pub:

Pub is a package manager for Dart and Flutter that is used to manage and share packages (i.e., libraries or dependencies) in Dart projects.

Environment:

```
minSdkVersion 21
targetSdkVersion 30
ext.kotlin_version = ‘1.5.10’
dependencies {
      classpath ‘com.android.tools.build:gradle:4.1.0’
      classpath “org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version”
   }
```

2. **Backend**:

Currently, this application's backend and data model are implemented using a NodeJS and is hosted in Google cloud platform

.

Framework:We have used the Express framework which is a popular framework for building Node.js applications. Express is lightweight, flexible, and has a large community of developers.

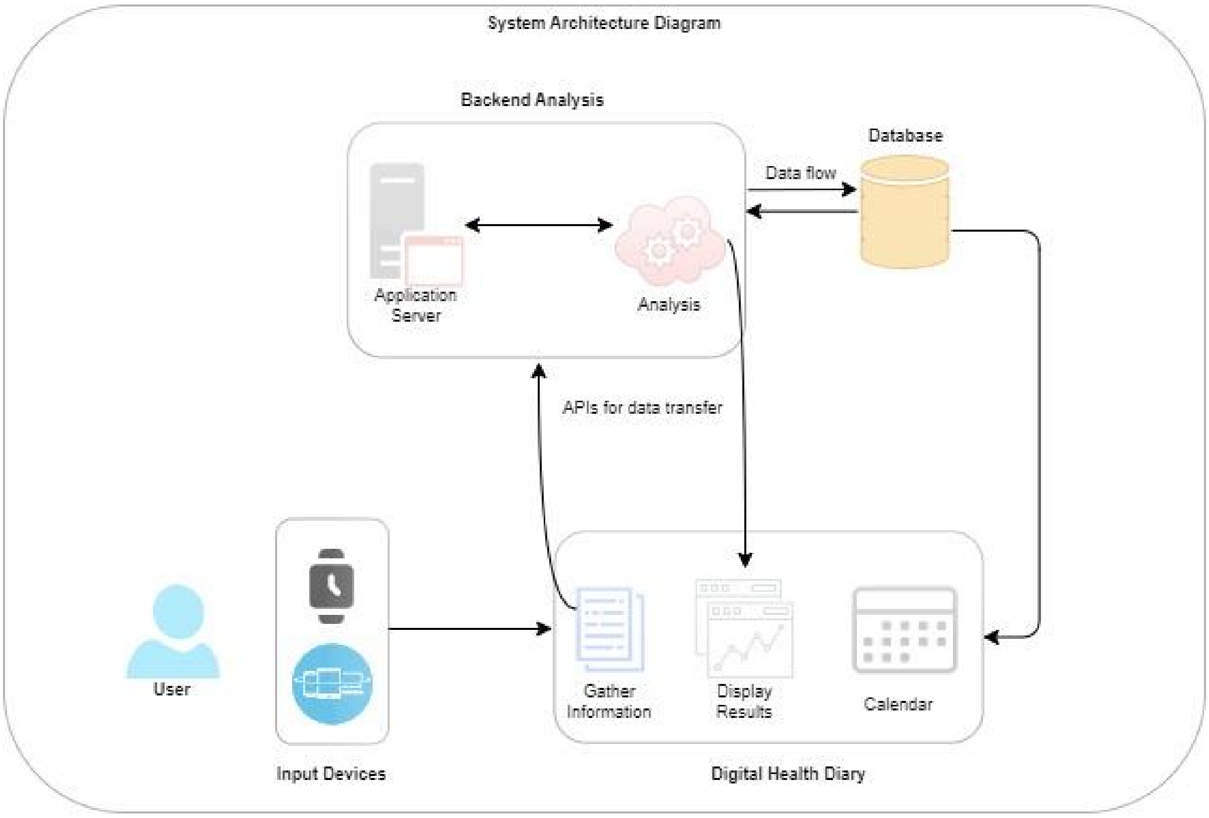
Database: We have used MongoDB, a NoSQL database for storing the data of the mobile application. MongoDB is a scalable, flexible, and document-oriented database that is well-suited for mobile applications. We have also used Google Cloud Storage, provided by Google Cloud Platform. It allows you to store and retrieve unstructured data, such as images, videos, documents, and backups, in a flexible and reliable manner. We have used GC Storage to store the audio received during Parkinson tests.

Authentication: We have used Passport.js for handling authentication in the mobile application. Passport is a widely-used authentication library in Node.js that provides a simple, modular, and customizable approach to authentication.

3. Connectivity for Backend and Frontend:

Flutter has built-in HTTP client packages such as http and dio that can be used to send HTTP requests to the backend. These packages can be used to send GET, POST, PUT, DELETE, and other types of HTTP requests.

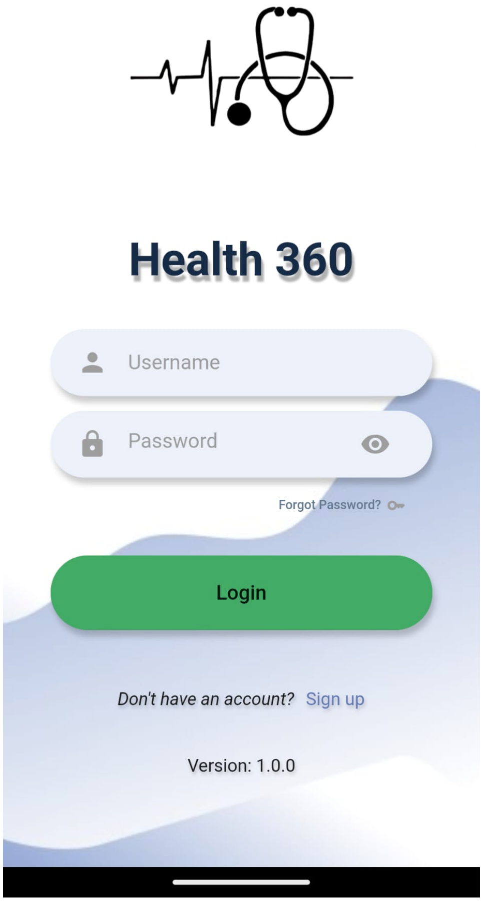
3. System Architecture Diagram



4. Features

1. Login

This is the first page of the application using which we authenticate the user and display the data, and tasks available to a particular logged-in user. A user with valid credentials can log into the application.



Health 360

Username

Password

Forgot Password?

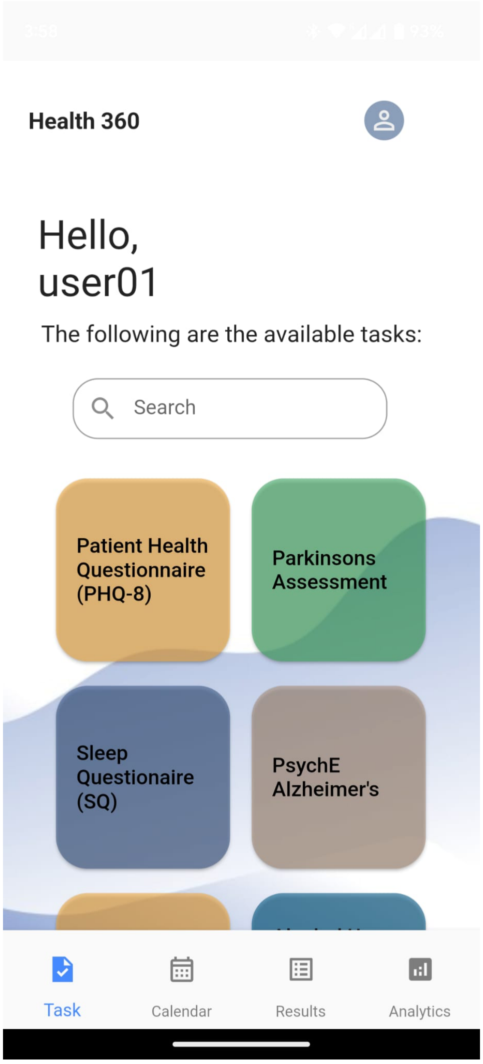
Login

Don't have an account? Sign up

Version: 1.0.0

2. Tasks

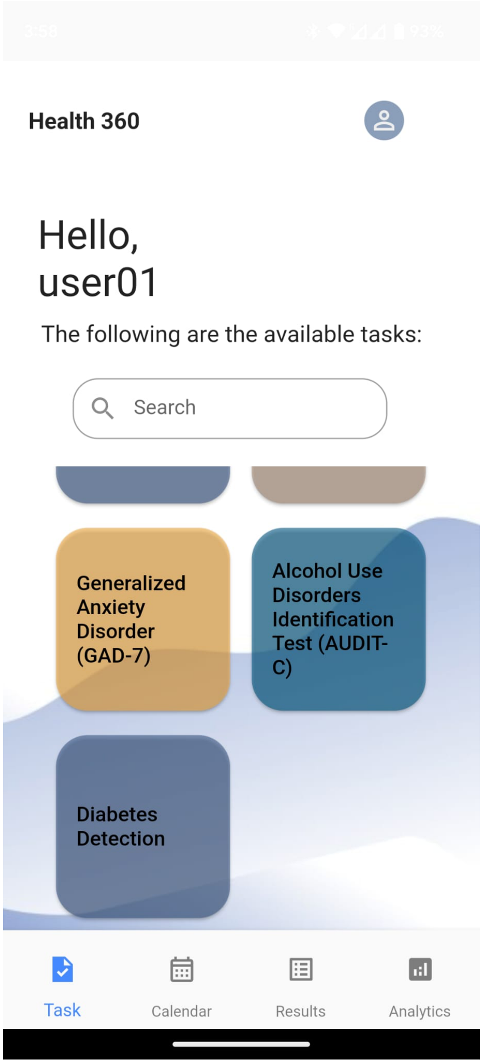
Once the user has successfully logged in he will be directed to the tasks page. In the context of this project tasks are some popular healthcare questionnaires like “Patient Health Questionnaire-8” , “Sleep Questionnaire”, “Parkinson’s Disease” etc.There are seven tasks available for now. The task page has all the tasks assigned to the user, and the user can select which task or test he wishes to perform. Selecting a particular task will take the user to that particular Task survey or activity or will ask him to perform certain actions.



Health 360

Hello,
user01

The following are the available tasks:



Health 360

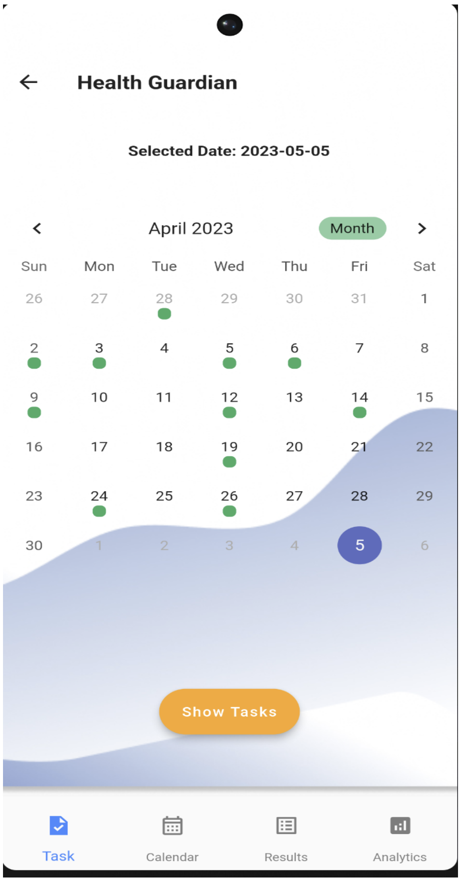
Hello,
user01

The following are the available tasks:

Search
Generalized Anxiety Disorder (GAD-7)
Alcohol Use Disorders Identification Test (AUDIT-C)
Diabetes Detection
Task
Calendar
Results
Analytics

3. Calendar

The calendar tab is one of the most important feature of Digital Health Diary. The calendar has highlighted all the tasks their results that the user undertook. After the user clicks on the calendar tab he will be directed to a calendar view in which he/she will see the days on the calendar displayed with a green dot. This green dot or highlight indicates that the user has taken some tests on this particular day. The user can further select this data and hit the show tasks button which will take him to a list of tasks that were taken on that particular day of the month where the user can see the results of the tests. Users can also change the calendar view to week, 2 weeks, or month as per their liking.



Health Guardian

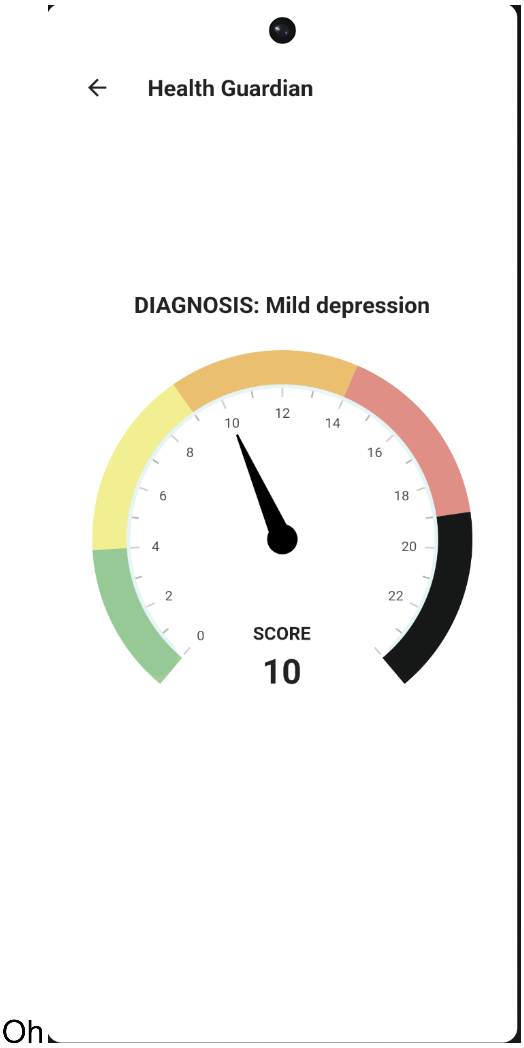
Selected Date: 2023-05-05

April 2023

Sun	Mon	Tue	Wed	Thu	Fri	Sat
26	27	28 ●	29	30	31	1
2 ●	3 ●	4	5 ●	6 ●	7	8
9 ●	10	11	12 ●	13	14 ●	15
16	17	18	19 ●	20	21	22
23	24 ●	25	26 ●	27	28	29
30	1	2	3	4	5	6

4. Results

The results page as the name suggests contains the results of all the tasks that the user has completed over time. The idea for this page was to provide users with easy access to the test outcomes. On the results page, the user can sort the results either based on task groups or date. Users can click on any task which takes them to a descriptive visual that will tell them the outcome of that task. This can help users or doctors to analyze and work on any particular tasks if needed.



5. Analytics

The Analytics page contains the systematic computational and statical analytics of the user data. This page can help users or doctors to get insights into user behaviour and see the patient's progress over time. The analytics page is like a dashboard view of patient data. The Analytics page has the following visuals:

a) Total Tasks: This donut chart indicated the total aggregation of each task category taken by the user. This sum will help the user see which task he does more frequently and what tasks he needs to do more regularly.

b) Average PHQ-8 Score: This is a gauge chart that displays the user's average PHQ-8 questionnaire score. This will give a quick highlight on how a patient is feeling whether he/she is depressed or have any symptoms of depression.

c) Average Sleep Score: This is a similar gauge as the PHQ-8 average. This one will highlight the user's average sleep scores which can help doctors diagnose if the patient has any sleep disorders.
d) Average GAD Score: This is a similar gauge as the PHQ-8 average. This one will highlight the user's average sleep scores which can help doctors diagnose if the patient has any sleep disorders.
e) Average AUDIT Score: This is a gauge chart that displays the user's average PHQ-8 questionnaire score. This will give a quick highlight on patients acohol consumption.

f) PHQ-8 Score: This is a scatter plot showing plotting all the user's PHQ-8 scores. The user or Doctor can take any further actions if required by analyzing the density of this scatter plot.

g) PHQ-8 Over Time: This is a simple line series indicating the patient's depression score over time. Looking at this chart patients or doctors can easily analyze whether the user is making any progress over time.

h) Sleep Score: This is a scatter plot showing plotting all the user's sleep scores. On this graph, the user will be able to see the results of all the sleep questionnaires and the outcomes that he/she has received.

i) Sleep Over Time: This line chart indicates the user's sleep score over time. Ideally this chart should be declining which will mean that user's sleep patterns are improving. However, this can give significant insights if there is a fluctuation in the patient's sleep routine.

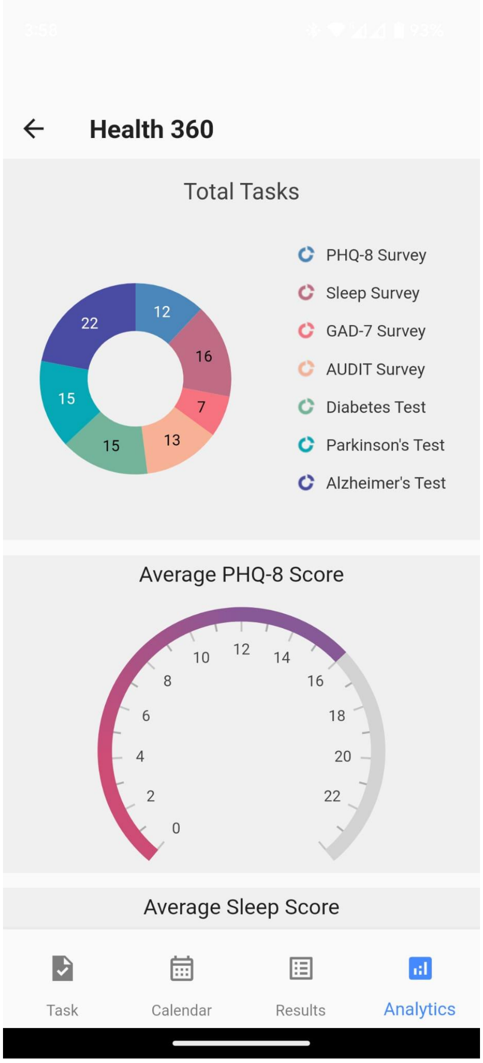
j) GAD-7 Score: This is a bar chart plotting all the user's GAD-7 scores. The user or Doctor can take any further actions if required by analyzing the density of this scatter plot.

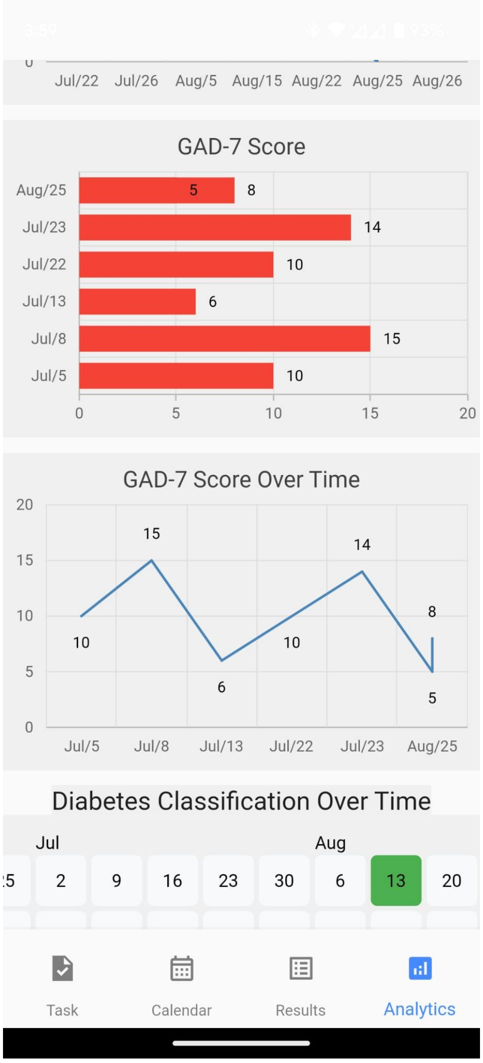
k) GAD-7 Over Time: This is a simple line series indicating the patient's anxiety score over time. Looking at this chart patients or doctors can easily analyze whether the user is making any progress over time.

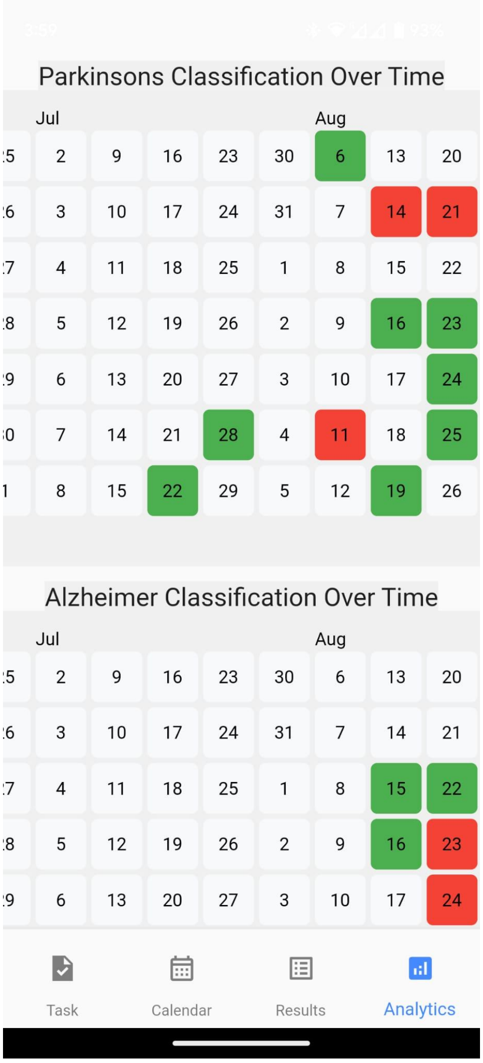
l) AUDIT Score: This is a bar chart plotting all the user's AUDIT scores. On this graph, the user will be able to see the results of all the sleep questionnaires and the outcomes that he/she has received.

m) AUDIT Over Time: This line chart indicates the user's alcohol consumption over time. Ideally this chart should be declining which will mean that user's sleep patterns are improving. However, this can give significant insights if there is a fluctuation in the patient's routine.

n) Diabetes Over Time: This is a heatmap that shows patients diabetes test classification. In this graph green means - “No-Diabetes”, orange means - “Pre-Diabeteic” and red symbolises “Diabetic”
o) Alzheimer's Over Time: This is a heatmap that shows patients dementia test classification. In this graph green means - “No-Alzheimers” and red symbolises “Alzheimer's”.
p) Parkinson's Over Time: This is a heatmap that shows patients' Parkinson's test results over time in calendar format. In this graph green means - “No-Parkinsons” and red means “Parkinsons”. This is a good representation of all the tests that patients took over time.


Health 360
Total Tasks
22
12
16
7
13
15
15
PHQ-8 Survey
Sleep Survey
GAD-7 Survey
AUDIT Survey
Diabetes Test
Parkinson's Test
Alzheimer's Test
Average PHQ-8 Score
0
2
4
6
8
10
12
14
16
18
20
22
Average Sleep Score
Task
Calendar
Results
Analytics


0
Jul/22
Jul/26
Aug/5
Aug/15
Aug/22
Aug/25
Aug/26
GAD-7 Score
Aug/25
Jul/23
Jul/22
Jul/13
Jul/8
Jul/5
5
8
14
10
6
15
10
0
5
10
15
20
GAD-7 Score Over Time
20
15
10
5
0
10
15
6
10
14
8
5
Jul/5
Jul/8
Jul/13
Jul/22
Jul/23
Aug/25
Diabetes Classification Over Time
Jul
Aug
5
2
9
16
23
30
6
13
20
Task
Calendar
Results
Analytics



Parkinsons Classification Over Time

	Jul					Aug		
:5	2	9	16	23	30	🟩 6	13	20
:6	3	10	17	24	31	7	🟥 14	🟥 21
:7	4	11	18	25	1	8	15	22
:8	5	12	19	26	2	9	🟩 16	🟩 23
:9	6	13	20	27	3	10	17	🟩 24
:0	7	14	21	🟩 28	4	🟥 11	18	🟩 25
1	8	15	🟩 22	29	5	12	🟩 19	26

Alzheimer Classification Over Time

	Jul					Aug		
:5	2	9	16	23	30	6	13	20
:6	3	10	17	24	31	7	14	21
:7	4	11	18	25	1	8	🟩 15	🟩 22
:8	5	12	19	26	2	9	🟩 16	🟥 23
:9	6	13	20	27	3	10	17	🟥 24

5. Disease Detection Mechanisms

1. Questionnaires

There are 4 questionnaires. Patient Health Questionnaire, Sleep Questionnaire, Generalized Anxiety Disorder and Alcohol use disorder identification test.

a. Patient Health Questionnaire: The Patient Health Questionnaire-8 (PHQ-8) is a self-reporting questionnaire used in the field of healthcare, particularly in mental health, to assess an individual's mental well-being and screen for symptoms of depression. There are 8 questions in this questionnaire. The options have a weightage of 0-3. The score is calculated by adding up the scores from all 8 questions. The final score can range anywhere from 0 to 24. Score of 0 to 4 classifies as minimal depression. Score 5 to 9 classifies as mild depression. 10 to 14 is moderate, 15 to 19 is moderately severe and above 19 is classified as severe depression.

b. Sleep Questionnaire: Sleep Questionnaire also known as The Stanford Sleepiness Scale is a single-item questionnaire that assesses a person's subjective level of sleepiness or alertness at a given moment. It is commonly used in research and clinical settings to quickly gauge a person's current level of sleepiness. The scale consists of seven points, each representing a different level of alertness.

c. GAD-7 Questionnaire: The Generalized Anxiety Disorder 7-item (GAD-7) is an easy to perform initial screening tool for generalized anxiety disorder. Score 0-4: Minimal Anxiety, Score 5-9: Mild Anxiety, Score 10-14: Moderate Anxiety and Score greater than 15: Severe Anxiety.

d. AUDIT Questionnaire: AUDIT stands for Alcohol Use Disorders Identification, this test is useful to identify if patients are over consuming alcohol and can help them track their weekly alcohol consumption. Score Interpretation for Men, a score of 4 or more is considered positive, optimal for identifying hazardous drinking or active alcohol use disorders. For Women, a score of 3 or more is considered positive, optimal for identifying hazardous drinking or active alcohol use disorders.

2. Diabetes Classification

Diabetes is a chronic (long-lasting) disease that affects millions of people around the globe. It is a metabolic disease identified by increased blood glucose levels that can lead to serious damage to the heart, blood vessels, eyes, kidneys, and nerves. There is a rise in diabetes and its consequences, so there is a need to develop effective treatment and prevention methods.

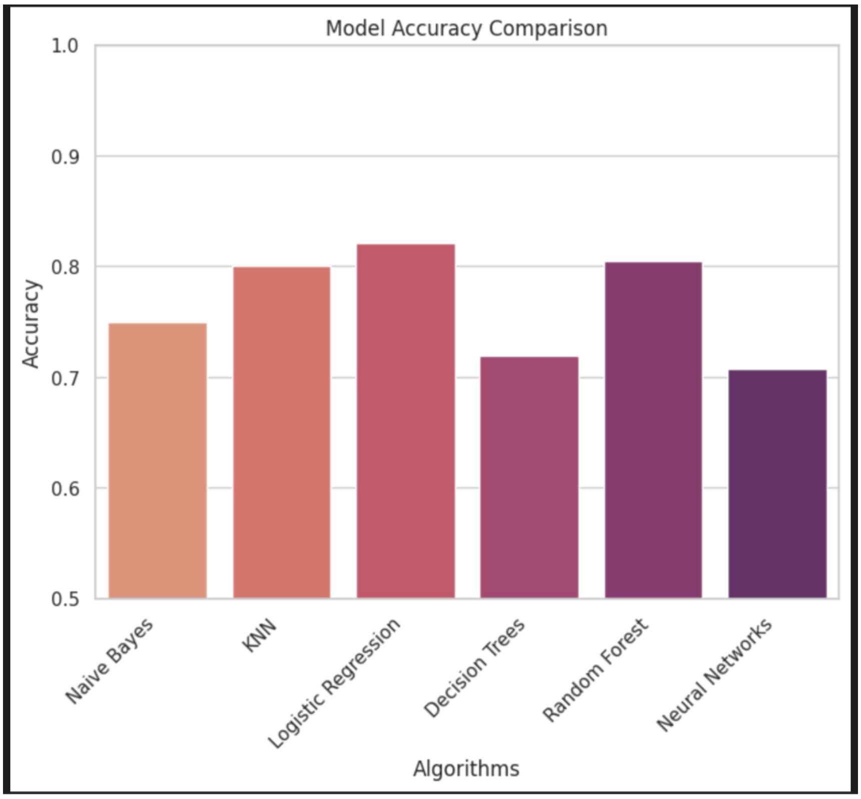
Data Source for Diabetes:

We are using the “Diabetes Health Indicators Dataset” from Kaggle. The description of the data as mentioned on Kaggle: https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

Machine Learning Model:

We trained various machine-learning models on the diabetes data set. The primary reason for using the Logistic Regression algorithm was its simplicity, it is easy to understand and implement compared to other advanced algorithms that we implemented like Random Forest and Neural Networks. In comparison with other models logistic regression performed better, especially in predicting “Pre-Diabetic” patients as the data for pre-Diabetic patients was relatively less.

Below is the Model Accuracy comparison chart for the diabetes data set:

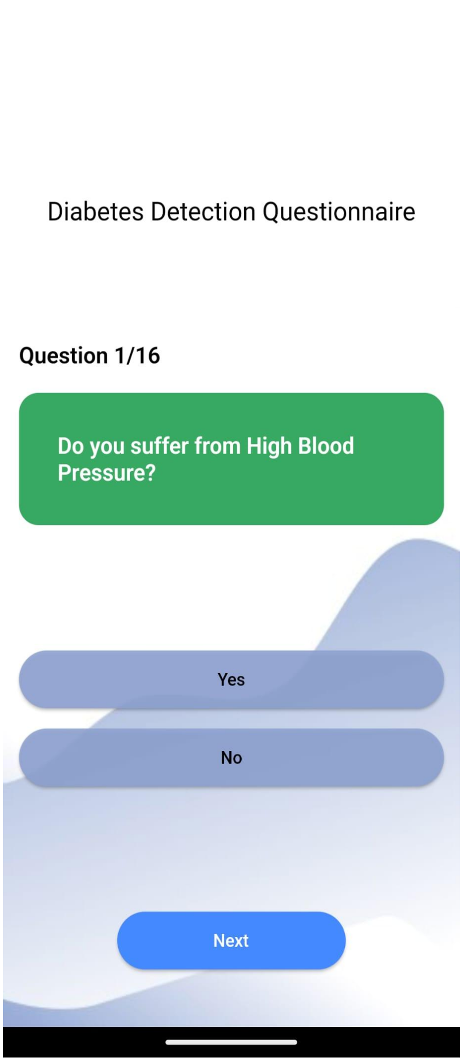


There are several other reasons for choosing Logistic Regression like the results from logistic regression are easy to understand. The model generates coefficients for each input variable, which represent the impact of each variable on the outcome that is predicted. Understanding the components of the classification can be simplified by this. Logistic regression models provide the probability of belonging to each class. In your case, you can obtain the probabilities of being "Diabetic," "Pre-Diabetic," or "No Diabetic" for each instance, allowing for a more neat and easy analysis.

Using logistic regression, we were able to evaluate the significance of each feature in the classification process. We were able to determine which variables have the greatest influence on the categorization by looking at the size and significance of the coefficients. Logistic regression is suitable for different types of data as it can handle both categorical and numerical input features. It is appropriate for circumstances in which there is a mixture of several types of variables.

Execution:

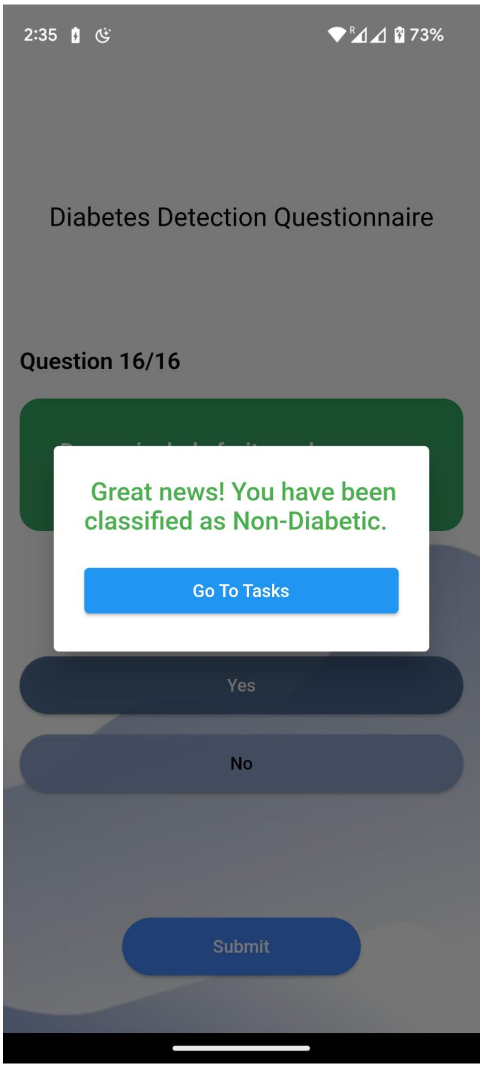
Flutter Health 360 app integrates the diabetes classification model using the **TFLite** package. With the TFLite package's efficient execution of machine learning models on-device, the app ensures data privacy by processing sensitive medical information locally. This integration enhances user experience by providing real-time results, fostering a proactive approach to managing diabetes within a user-friendly and secure environment.



Diabetes Detection Questionnaire

Question 1/16

Do you suffer from High Blood Pressure?



Diabetes Detection Questionnaire

Question 16/16

Great news! You have been classified as Non-Diabetic.

3. Alzheimer's Detection

Alzheimer's disease is a progressive neurodegenerative disorder that primarily affects the brain, leading to memory loss, cognitive decline, and behavioural changes. A key priority in Alzheimer's disease (AD) research is the identification of early intervention strategies that will decrease the risk, delay the onset, or slow the progression of the disease.

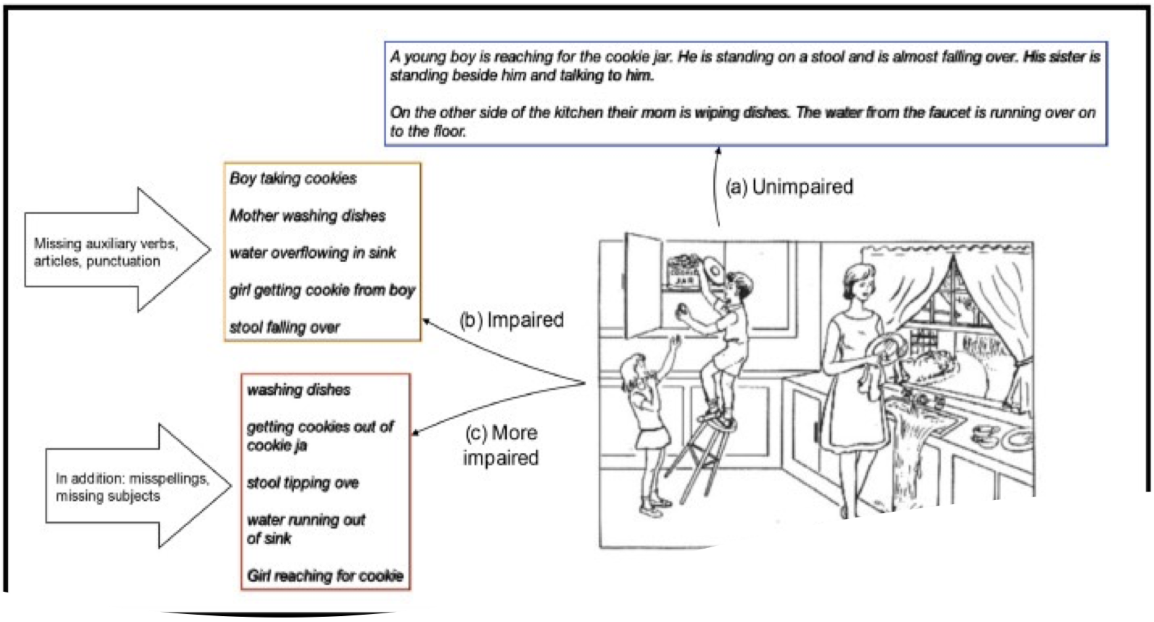
Data Source for Alzheimers:

This repository contains code for detecting Alzheimer's patients from linguistic cues. The dataset used is the “Dementia Bank dataset” which contains audio transcripts of various individuals on the “Recall Test” The dataset can be downloaded from the following link: https://dementia.talkbank.org/

Approach:

Prediction is based on data collected while the participants were cognitively healthy. Focus exclusively on variables readily attainable as part of the screening phase of an early-intervention trial and assess predictive performance using only linguistic metrics derived from a single administration of the **Cookie Theft Task**, a relatively simple and naturalistic language probe. It’s a machine learning approach to deal with a multivariate representation of linguistic performance comparing the predictive ability of language features with that of more traditional variables associated with the identification of high risk for AD.

Reference:
https://www.thelancet.com/journals/eclinm/article/PIIS2589-5370(20)30327-8/fulltext#seccesectitle0018

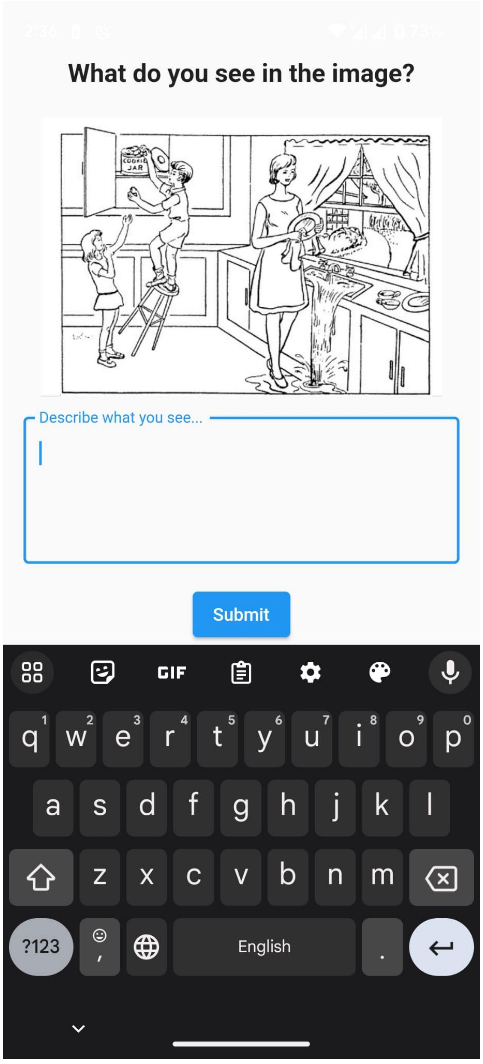

A young boy is reaching for the cookie jar. He is standing on a stool and is almost falling over. His sister is standing beside him and talking to him.
On the other side of the kitchen their mom is wiping dishes. The water from the faucet is running over on to the floor.
(a) Unimpaired
Missing auxiliary verbs, articles, punctuation
Boy taking cookies
Mother washing dishes
water overflowing in sink
girl getting cookie from boy
stool falling over
(b) Impaired
In addition: misspellings, missing subjects
washing dishes
getting cookies out of cookie ja
stool tipping ove
water running out of sink
Girl reaching for cookie
(c) More impaired
JAR

Machine Learning Model:

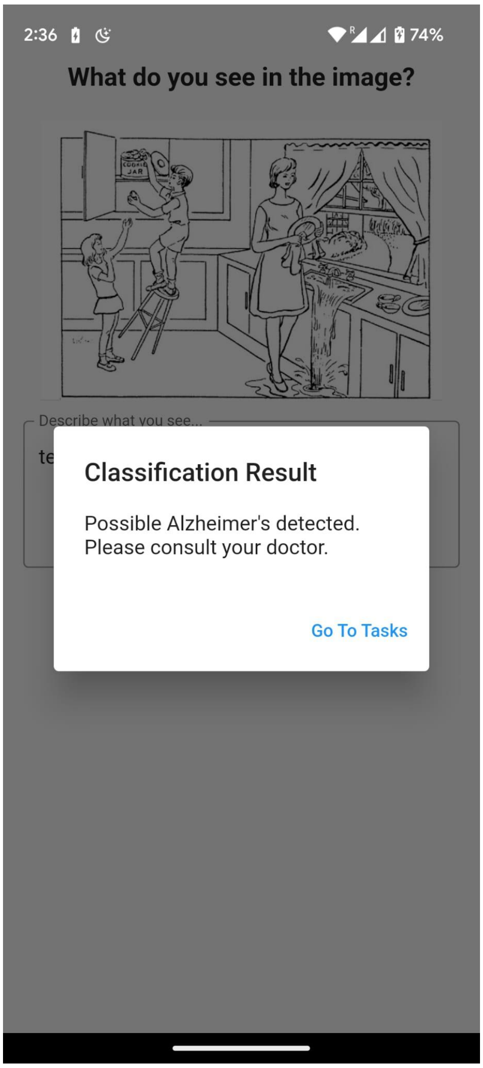
The machine learning model utilises the TensorFlow framework. The code employs a sequential architecture with an embedding layer to process textual data efficiently. Following the embedding layer, a sequence of dense layers progressively extracts features, leading to a final classification using sigmoid activation. The model is compiled with binary cross-entropy loss and optimised using the Adam optimizer, aiming to predict the onset of Alzheimer's disease from the dataset. The summarised architecture showcases its layers, dimensions, and activation functions, demonstrating its potential to provide valuable insights into Alzheimer's risk assessment.

Execution:

We have implemented a prediction tool into our app using a model file like and tokenizer file. When users type in text, like describing a picture of a cookie theft image, our backend code breaks down the text into important parts using the tokenizer. This organized text is then sent to the model, which quickly figures out if there's a chance of Alzheimer's disease. This helps users know about Alzheimer's risk just by typing, making it easier to catch early signs and raise awareness, all in a user-friendly app.



What do you see in the image?



2:36 74%

What do you see in the image?

Classification Result

Possible Alzheimer's detected.
Please consult your doctor.

4. Parkinson's Detection

Parkinson's disease is a neurological condition that largely interferes with how well the body moves and coordinates. Tremors, Bradykinesia, which causes slowness in movement and confusion, irregular speech, and many additional indications are some of the signs.

Data Collection

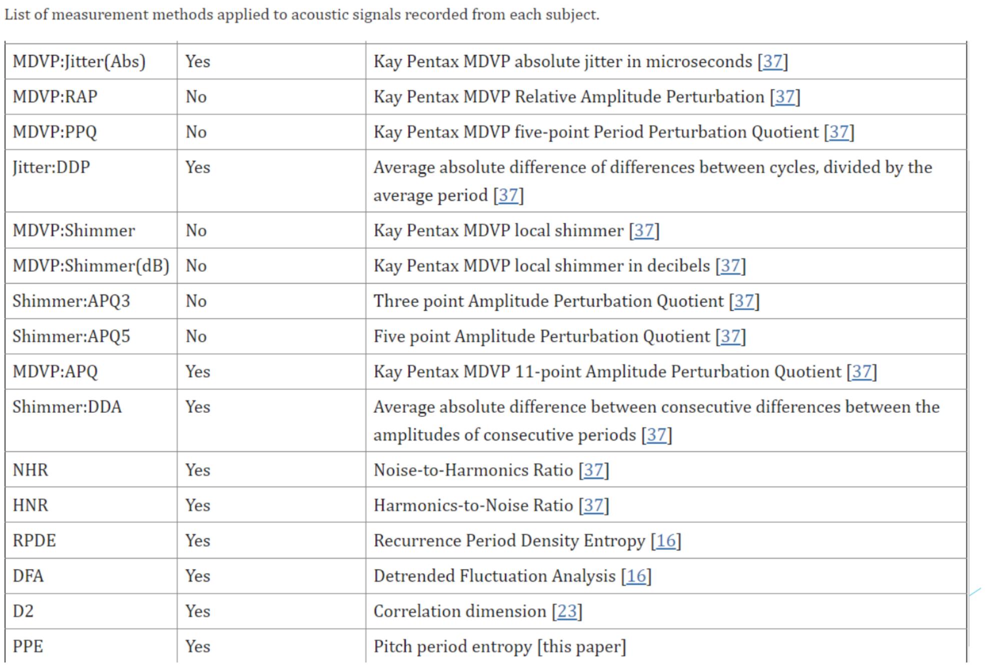
The dataset was created by Max Little of the University of Oxford, in collaboration with the National Centre for Voice and Speech, Denver, Colorado, who recorded the speech signals. The original study published the feature extraction methods for general voice disorders.
This dataset is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). Each column in the table is a particular voice measure, and each row corresponds one of 195 voice recording from these individuals.Further details are contained in the following reference
Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008), 'Suitability of dysphonia measurements for telemonitoring of Parkinson's disease', IEEE Transactions on Biomedical Engineering .Dataset can directly be downloaded from
https://www.kaggle.com/datasets/thecansin/parkinsons-data-set?resource=download

Approach of Detection

One of the main symptoms of Parkinson's disease is spasmodic dysphonia, which affects the control and coordination of the voice chords. In basic terms, it refers to a group of voice disorders distinguished by abnormal or compromised vocal quality, pitch, loudness, or resonance.Some of the speech alterations that Parkinson's patients have experienced were used in our model.

The study is primarily based on the publication “Suitability of dysphonia measurements for telemonitoring of Parkinson's disease” from IEEE Trans Biomed Eng. 2009 Apr. (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3051371/)

From the collected phonations from 31 people of which 23 were with PD, the paper undertook analysis of various uncorrelation measures of the voices and figured out whether the person is suffering from the condition or not.Such impaired vocal symptoms included difference in pitch and frequencies,jitters(the extent of variation in speech F0 from vocal cycle to vocal cycle) ,shimmer (the extent of variation in speech amplitude from cycle to cycle) ,noise-to-harmonics ratios (the amplitude of noise relative to tonal components in the speech) and others like



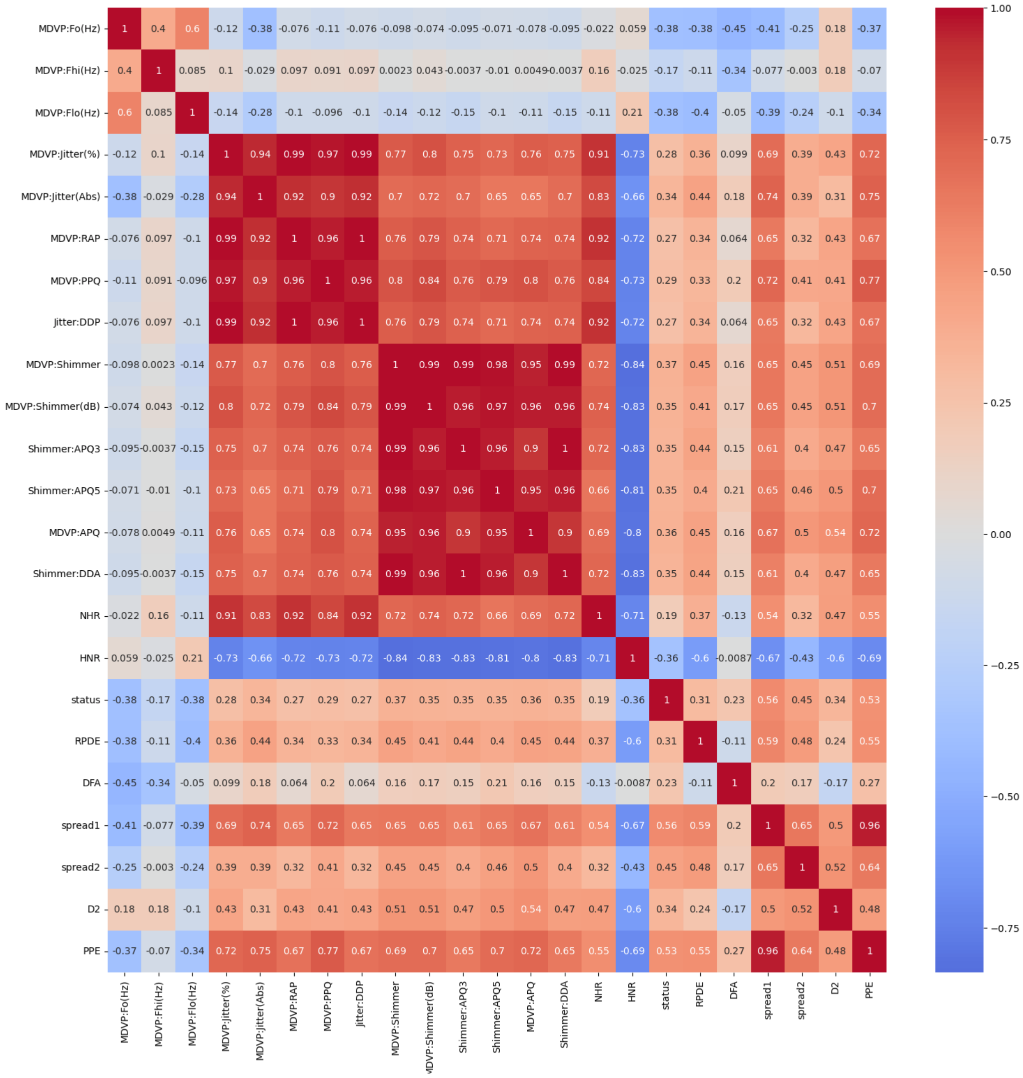
List of measurement methods applied to acoustic signals recorded from each subject.

MDVP:Jitter(Abs)	Yes	Kay Pentax MDVP absolute jitter in microseconds [37]
MDVP:RAP	No	Kay Pentax MDVP Relative Amplitude Perturbation [37]
MDVP:PPQ	No	Kay Pentax MDVP five-point Period Perturbation Quotient [37]
Jitter:DDP	Yes	Average absolute difference of differences between cycles, divided by the average period [37]
MDVP:Shimmer	No	Kay Pentax MDVP local shimmer [37]
MDVP:Shimmer(dB)	No	Kay Pentax MDVP local shimmer in decibels [37]
Shimmer:APQ3	No	Three point Amplitude Perturbation Quotient [37]
Shimmer:APQ5	No	Five point Amplitude Perturbation Quotient [37]
MDVP:APQ	Yes	Kay Pentax MDVP 11-point Amplitude Perturbation Quotient [37]
Shimmer:DDA	Yes	Average absolute difference between consecutive differences between the amplitudes of consecutive periods [37]
NHR	Yes	Noise-to-Harmonics Ratio [37]
HNR	Yes	Harmonics-to-Noise Ratio [37]
RPDE	Yes	Recurrence Period Density Entropy [16]
DFA	Yes	Detrended Fluctuation Analysis [16]
D2	Yes	Correlation dimension [23]
PPE	Yes	Pitch period entropy [this paper]

Model and Execution

In our model using libraries like Librosa,Parselmouth, scipy, nolds, Pyrpde and others,we loaded and analyzed all the features from recorded audio .There are 20 independent variables in total and Target variable is ‘status’.We have trained our model using Support vector machine yielding accuracy of .87 with test data.Status 0 will indicate patient is healthy but 1 suggests the patients is suffering from Parkinson Disease.

Correlation matrix for the dataset:



	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	DFA	spread1	spread2	D2	PPE
MDVP:Fo(Hz)	1	0.4	0.6	-0.12	-0.38	-0.076	-0.11	-0.076	-0.098	-0.074	-0.095	-0.071	-0.078	-0.095	-0.022	0.059	-0.38	-0.38	-0.45	-0.41	-0.25	0.18	-0.37
MDVP:Fhi(Hz)	0.4	1	0.085	0.1	-0.029	0.097	0.091	0.097	0.0023	0.043	-0.0037	-0.01	0.0049	-0.0037	0.16	-0.025	-0.17	-0.11	-0.34	-0.077	-0.003	0.18	-0.07
MDVP:Flo(Hz)	0.6	0.085	1	-0.14	-0.28	-0.1	-0.096	-0.1	-0.14	-0.12	-0.15	-0.1	-0.11	-0.15	-0.11	0.21	-0.38	-0.4	-0.05	-0.39	-0.24	-0.1	-0.34
MDVP:Jitter(%)	-0.12	0.1	-0.14	1	0.94	0.99	0.97	0.99	0.77	0.8	0.75	0.73	0.76	0.75	0.91	-0.73	0.28	0.36	0.099	0.69	0.39	0.43	0.72
MDVP:Jitter(Abs)	-0.38	-0.029	-0.28	0.94	1	0.92	0.9	0.92	0.7	0.72	0.7	0.65	0.65	0.7	0.83	-0.66	0.34	0.44	0.18	0.74	0.39	0.31	0.75
MDVP:RAP	-0.076	0.097	-0.1	0.99	0.92	1	0.96	1	0.76	0.79	0.74	0.71	0.74	0.74	0.92	-0.72	0.27	0.34	0.064	0.65	0.32	0.43	0.67
MDVP:PPQ	-0.11	0.091	-0.096	0.97	0.9	0.96	1	0.96	0.8	0.84	0.76	0.79	0.8	0.76	0.84	-0.73	0.29	0.33	0.2	0.72	0.41	0.41	0.77
Jitter:DDP	-0.076	0.097	-0.1	0.99	0.92	1	0.96	1	0.76	0.79	0.74	0.71	0.74	0.74	0.92	-0.72	0.27	0.34	0.064	0.65	0.32	0.43	0.67
MDVP:Shimmer	-0.098	0.0023	-0.14	0.77	0.7	0.76	0.8	0.76	1	0.99	0.99	0.98	0.95	0.99	0.72	-0.84	0.37	0.45	0.16	0.65	0.45	0.51	0.69
MDVP:Shimmer(dB)	-0.074	0.043	-0.12	0.8	0.72	0.79	0.84	0.79	0.99	1	0.96	0.97	0.96	0.96	0.74	-0.83	0.35	0.41	0.17	0.65	0.45	0.51	0.7
Shimmer:APQ3	-0.095	-0.0037	-0.15	0.75	0.7	0.74	0.76	0.74	0.99	0.96	1	0.96	0.9	1	0.72	-0.83	0.35	0.44	0.15	0.61	0.4	0.47	0.65
Shimmer:APQ5	-0.071	-0.01	-0.1	0.73	0.65	0.71	0.79	0.71	0.98	0.97	0.96	1	0.95	0.96	0.66	-0.81	0.35	0.4	0.21	0.65	0.46	0.5	0.7
MDVP:APQ	-0.078	0.0049	-0.11	0.76	0.65	0.74	0.8	0.74	0.95	0.96	0.9	0.95	1	0.9	0.69	-0.8	0.36	0.45	0.16	0.67	0.5	0.54	0.72
Shimmer:DDA	-0.095	-0.0037	-0.15	0.75	0.7	0.74	0.76	0.74	0.99	0.96	1	0.96	0.9	1	0.72	-0.83	0.35	0.44	0.15	0.61	0.4	0.47	0.65
NHR	-0.022	0.16	-0.11	0.91	0.83	0.92	0.84	0.92	0.72	0.74	0.72	0.66	0.69	0.72	1	-0.71	0.19	0.37	-0.13	0.54	0.32	0.47	0.55
HNR	0.059	-0.025	0.21	-0.73	-0.66	-0.72	-0.73	-0.72	-0.84	-0.83	-0.83	-0.81	-0.8	-0.83	-0.71	1	-0.36	-0.6	-0.0087	-0.67	-0.43	-0.6	-0.69
status	-0.38	-0.17	-0.38	0.28	0.34	0.27	0.29	0.27	0.37	0.35	0.35	0.35	0.36	0.35	0.19	-0.36	1	0.31	0.23	0.56	0.45	0.34	0.53
RPDE	-0.38	-0.11	-0.4	0.36	0.44	0.34	0.33	0.34	0.45	0.41	0.44	0.4	0.45	0.44	0.37	-0.6	0.31	1	-0.11	0.59	0.48	0.24	0.55
DFA	-0.45	-0.34	-0.05	0.099	0.18	0.064	0.2	0.064	0.16	0.17	0.15	0.21	0.16	0.15	-0.13	-0.0087	0.23	-0.11	1	0.2	0.17	-0.17	0.27
spread1	-0.41	-0.077	-0.39	0.69	0.74	0.65	0.72	0.65	0.65	0.65	0.61	0.65	0.67	0.61	0.54	-0.67	0.56	0.59	0.2	1	0.65	0.5	0.96
spread2	-0.25	-0.003	-0.24	0.39	0.39	0.32	0.41	0.32	0.45	0.45	0.4	0.46	0.5	0.4	0.32	-0.43	0.45	0.48	0.17	0.65	1	0.52	0.64
D2	0.18	0.18	-0.1	0.43	0.31	0.43	0.41	0.43	0.51	0.51	0.47	0.5	0.54	0.47	0.47	-0.6	0.34	0.24	-0.17	0.5	0.52	1	0.48
PPE	-0.37	-0.07	-0.34	0.72	0.75	0.67	0.77	0.67	0.69	0.7	0.65	0.7	0.72	0.65	0.55	-0.69	0.53	0.55	0.27	0.96	0.64	0.48	1

Audio processing libraries like Parsel mouth and Librosa are specifically used to extract the independent variables from an audio.Traditional measurements were calculated using inbuilt functions of Praat library through parselmouth.Calculations were mainly based on the methods as specified in the same paper mentioned for non-linear measurements like D2,PPE,Correlation dimension https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3051371.For audio processing purposes, audio has been used from a voice database.

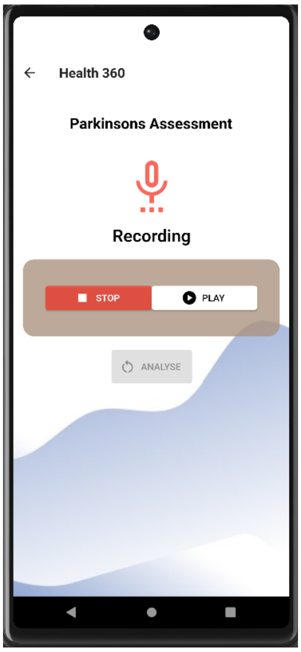
Link to the voice database : https://stimmdb.coli.uni-saarland.de/

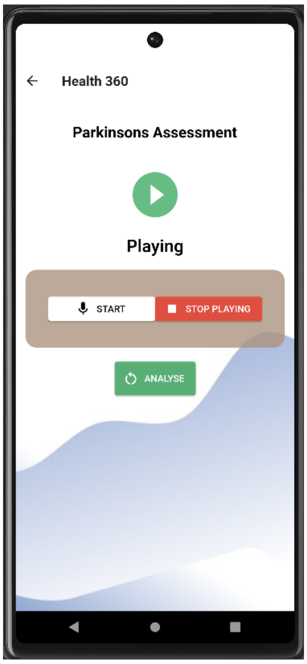
Link to the voice database manual :

https://stimmdb.coli.uni-saarland.de/help_de.php4

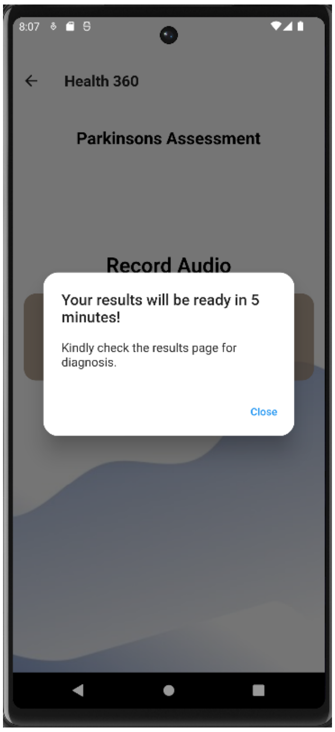
Frontend Functionalities

From the Task page of the Health 360 application, one can take the Parkinson assessment. It will direct us to a page where users can record audio. For fastest processing, we advise users to keep their recordings between 5 and 10 seconds long.After recording, the audio can be played for the user to listen to and review. The audio being recorded and played is visually represented on the assessment page.





After recording the audio and hitting the analyze button, a Post API call is performed to transfer the results to the server, which are then processed there.We retrieve the result by GET API call similar to the other tasks by sending the ‘hid’ in the API endpoint. Additionally saved in the backend are the audio files for each test. A prompt to examine the outcome after five minutes from Result page is displayed.



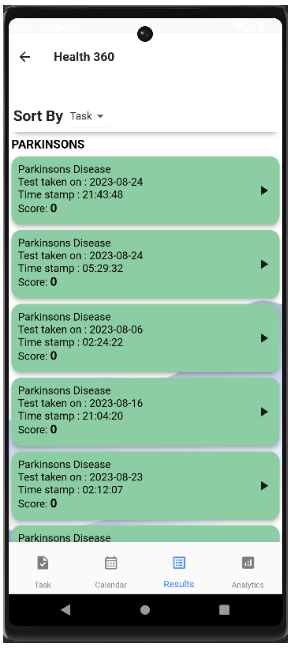
Parkinsons Assessment

Record Audio

Your results will be ready in 5 minutes!

Kindly check the results page for diagnosis.

The diagnosis can then be verified by users via the Results page.
The corresponding audio can also be played from the result page by clicking on the particular test which can be sorted according to Task and Date.All the Parkinson results are indicated with a play icon which show that it has audio.



Sort By Task

PARKINSONS

Parkinsons Disease
Test taken on : 2023-08-24
Time stamp : 21:43:48
Score: **0**

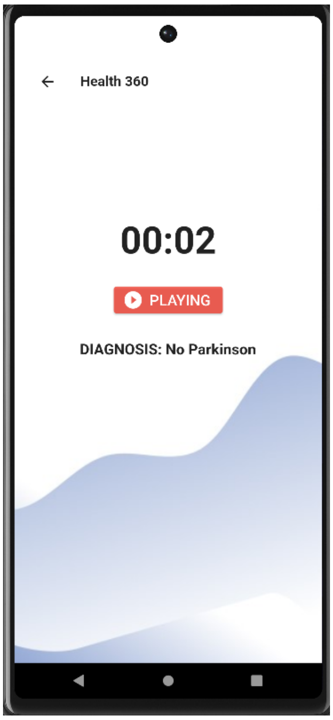
Parkinsons Disease
Test taken on : 2023-08-24
Time stamp : 05:29:32
Score: **0**

Parkinsons Disease
Test taken on : 2023-08-06
Time stamp : 02:24:22
Score: **0**

Parkinsons Disease
Test taken on : 2023-08-16
Time stamp : 21:04:20
Score: **0**

Parkinsons Disease
Test taken on : 2023-08-23
Time stamp : 02:12:07
Score: **0**

Parkinsons Disease


Health 360

00:02

PLAYING

DIAGNOSIS: No Parkinson

6. Backend

The backend of our application is powered by a Node.js server that utilizes the Express framework. The server is connected to MongoDB and Google Cloud Storage through their respective APIs. This integration allows our application to efficiently store and retrieve data from MongoDB and utilize Google Cloud Storage for various storage needs.
To implement the backend, we have used libraries like mongoose, express, multer, google cloud storage.

- Express: Its primary purpose is to simplify the process of building web applications and APIs (Application Programming Interfaces) by providing a set of tools and features like Middlewares, API routing, Error Handling etc.

- Mongoose: Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js. It simplifies the interaction between your Node.js application and a MongoDB database.

- Multer: Multer is a popular middleware for handling file uploads in Node.js web applications. It simplifies the process of receiving and storing files from client requests, such as file uploads via HTML forms.

- Google cloud Storage: This library is used to establish the connection between NodeJS and the cloud using some form of authentication.

In the app.js page, the routes are defined. Each route is given an unique name. Nodejs is connected to our MongoDB using mongoose. Each Database has its connection link which is available in mongodb. Error handling is also defined in this page.

Disease APIs

Below are the APIs for the diseases. Each disease is associated with a pair of APIs, one designed for POST requests and the other intended for GET requests. The HTTP POST request is equipped with a request body through which selections are submitted, while the subsequent HTTP GET request is employed for the purpose of retrieving the resultant data

All results are saved in the mongoDB using the same schema stated below. The schema can be found under api/models folder inside ‘diag.js’. We need to import the schema everytime we want to save records in mongoDB

{
id: mongoDB generated ID (string),

uid: user who took the test (string),

hid: this is the unique id of the test (string). The naming convention is
“uid”_”task id”_”date time created e format YYYYMMDDHHMMSSms”
Example, a phq8 test taken on 16th August 2023, at 9:18:23.456 pm by user1 will have
hid : “user1_PHQ-8_20230816211823456”

task_id: name of the task taken (string)

task_score: score of the questionnaire (integer)

classification: diagnosis according to the score (string).

createdAt: utc time when task was taken (Date).

updatedAt: utc time when result was last updated (Date).
}

1. **Questionnaires:** Health 360 app has 4 questionnaires. The POST request for all 4 of them work the same way. During POST request, once the score is

retrieved, the classification is saved and the result is saved in MongoDB.

a. **PHQ_8**: Patient Health Questionnaire or PHQ-8 sends the 8 answers of the questions via the body along with a user field.

   Once the scores are received, we compute the sum of options and save it in mongoDB. After that we return the mongoDB id as the response, with key “id”.

   Below are the fields and its descriptions.

   i. *user*: The user field contains a string stating the user who took the test.
   ii. *question1-question8*: These 8 fields contain integers stating the chosen option starting from 0 to 3. So, if the user chose the 3rd option for question1, the field question1 will have a value of 2.

b. **GAD_7**: Generalized Anxiety Disorder works the same as PHQ-8 . The only difference is that it has 7 questions instead of 8. The score calculation is the same as PHQ-8.

c. **Sleep_questionnaire(sleep_q)**: Sleep questionnaire is also a questionnaire like PHQ8 and GAD7 but has only 1 question. There is no score calculation. The option selected is the score.

d. **AUDIT**: Alcohol Usage Disorder Identification Test is one of the 4 questionnaires implemented in our app. This is similar to other questionnaires, but has 3 questions and 5 options per question.

2. **Diabetes(diab)**: For prediction of diabetes, the app is not using its backend server. Instead, it is using a tflite model to predict the value. Classification is also happening in the front end. For this reason, backend will be used to save the record to MongoDB. POST request will be used to pass the score as the body and diagnosis classification will be done in the backend (same logic as front end) and the result will be stored in mongoDB following the schema mentioned above. The body for this is as follows.
   a. *user*: the user who is taking the test (string).
   b. *score*: the class returned after prediction.(0,1 or 2)

3. **Alzheimer's(alzh_d)**: Alzheimers is a bit different than the previously mentioned diseases. In alzheimers, we will be using an extra module called “child-process”. It is inbuilt in nodejs and does not need installation. It allows one to create and manage child processes from within a Node.js application.

These child processes can run separate executable files. For this disease, we are taking a string and doing some calculations on it using a python script. After that, the python script will return an output 0 or 1, 0 being negative and 1 being positive. Since this is a nodejs app and we will host the docker in a ubuntu environment, we need to use child processes to execute the python file. We also need to have python installed in the ubuntu environment (more about this in docker section). For this POST request, a body is passed having 2 fields. “text” field has the main data. The text is then passed to the python scripts using child processes. Once the score is retrieved, the test is stored in mongoDB using the schema mentioned. The body for the POST request is as follows.

a. *user*: the user who is taking the test (string).
b. *text*: user input for the test (String).

4. **Parkinson’s(par_d)**: Parkinson's disease detection is the most complicated one in this group. It involves sending audio data from the front end. And then processing it using a python script, and feeding it to a ML model. After that the output is retrieved. For starters, apart from our regular libraries like mongoose and child processes, we will be using multer, google cloud storage and path(inbuilt) too. The audio is to be sent using a multipart POST request. Once the audio file is received, using multer library, the audio will be saved in a folder called uploads temporarily. The audio will have the same name as the hid of that test (this will help later while audio retrieval). We need to save the audio in a database since we will be keeping a record of the audio as well and users will be able to play the audio while retrieving the results. For that, i have chosen google cloud storage. One can use any other DB like mongoDB, amazon s3 etc. Setup for Google cloud storage will be below. Once the audio is stored in uploads folder, the python file is executed with the hid as input. Then the python script fetches the audio from the uploads folder using the filename and runs series of computations on it. Once that is done, it is being fed to the ml model and then we retrieve the output(0 or 1). If the python script is processed successfully, and we get a 200 status code, we proceed to save the audio in google cloud storage. The audio is then retrieved from the uploads folder and then saved in google cloud storage. Then the result is saved in mongoDB using the same schema. And, mongoDB id is returned as response. The body structure for the above POST request is mentioned below.

   a. *user*: user who took the test (Note, user field should always be before the audio).
   b. *audio*: the audio wav file recorded during the parkinsons test.

5. GET request: All diseases have similar get requests. Every POST requests mentioned above returns an id. To fetch the id, we need to give the id received along with the get URL. For example, if GET request of PHQ-8 returned a response {id: “632nas390rh1ijr0”}, then to fetch the respective score, we need to make a GET request using the get URL and id, like this : “get_URL”/632nas390rh1ijr0. The server will search through the mongoDB collections and return the record and other information about the test by matching the id. The response schema will be like:
   ```
   {
   data: {
             classification: the diagnosis (string),
             score: the score or classification (int)
           },
   metadata: {
                       creation_timestamp: time when test was taken (date),
                       task_id: mongoDB id of the result,
                       user_id: user who took the test
                 },
   type: type of test
   }
   ```

Other APIs

Apart from the disease APIs, we have some more APIs. One of them is ‘data’ which returns all the records in the mongoDB according to the user.

1. Data API: This is a GET request. The URL of the GET request will be “URL”/data/”user”. This request will go through the collections and return all the records whose uid will match with the “user” which is given to the URL.

The next api is the get_audio api. This is used to retrieve the audio requested by the user which was recorded during Parkinson's test.

2. Audio API(get_audio/file): In this GET request, we will append the hid(unique) of the parkinson's test, whose audio is required. The URL will be “URL”/get_audio/file/”hid”. The server will look for the file name matching with the hid inside google cloud storage. Then return the audio in wav format.

The users api is used for creating users, logging in, deleting users.

3. Users API: There are 3 APIs under this section. The schema for this will under the api/models folder inside ‘user.js’ file

    a. signup(POST): In this, we will pass a body which will contain the fields “username” and “password”. The backend will save the username and password in mongoDB and will return a response saying User has been created. The server will save the password as plaintext. We can upgrade the feature by using the **‘bcrypt’** library which will hash the password before saving it in mongoDB.
    Note: The user login details will be saved under the same database but different collections.

    b. login(POST): This is also a POST request, where we will send a body containing “username” and “password”. The server will search for the username in the database and then match the password. If the password matches, it will send a response with a status code of 200. Else, it will send a response saying Auth failed with a code of 401. If we use the bcrypt library, the authentication process will be a bit different.

    c. DELETE: This is a delete request, which will take the mongoDB id in the URL and delete the requested user.

Integration of Database with NodeJS

Our app is using two databases. One is mongoDB, another one is Google cloud storage. Both have different ways to connect to the server.
Database creation in MongoDB is an easy process. Once created, we can easily connect mongoDB with NodeJS by using mongoose library and mongoDB URI which is available under ‘connect’ option.

The second database we used is google cloud storage. For gc storage, once you have logged into your account, a bucket has to be created. Then, we need to create a service account. Once the account is created, we need to create a key if its not already created. While creating the key, google will download a json file under the name of the project. That file will be the authentication file. We have kept the file inside the uploads folder. But the file won't be visible due to its sensitive content. Once the file is obtained, one must add the filename to the nodeJS app.

Note: Anyone with the json file can connect to your google cloud storage.

The postman collection for the server is available in github under
Health 360/Backend/uploads folder

7. Deployment

We have deployed our server in google cloud run under google cloud platform. For this, we created a dockerfile. To create a dockerfile, we need to create an environment first. Since this is a nodeJS app, our environment should be based on node JS. But it should also have python since we are executing python scripts from the command line. There are two ways to execute this. 1. Create a node js environment and then install python inside that. In this method, we need to install python as we would do in a ubuntu machine, since our docker will be hosted in ubuntu. 2. We can import an environment with python pre-installed. For this app, I used the second method.

We used an existing docker image from the docker hub (ref: https://hub.docker.com/r/nikolaik/python-nodejs). This docker image has nodeJS with python pre-installed in it. For that, python installation was not necessary. After that, remaining important files were copied and a docker file was built. The docker container was then pushed to Google cloud platform using Google cloud sdk shell. Then the docker container was hosted in Google cloud run.

8. Task Distribution

Team Member	**Tasks**
Sagnik	1. Researched on Questionnaires like PHQ-8 and Sleep Questionnaire. 2. Modified the UI for PHQ-8 and Sleep Questionnaire. 3. Updated the task page. 4. Created the Backend for the app. 5. Introduced new APIs and features. 6. Integrated all the APIs with the app. 7. Created the server for the app and hosted it on Google cloud platform.
Namrata	1. Researched on different approaches for Parkinson disease detection 2. Developed ML model for parkinson detection using dysphonia 3. Developed python script for extracting audio

	features 4. Worked on UI for Parkinson assessment(Audio recording/playing/fetching) 5. Result page and Calendar list updates after API changes.
Saumitra	1. Researched new surveys GAD-7 and AUDIT 2. Researched on different approaches to predict onset of Alzheimer's Disease 3. Built the UI for GAD-7 and AUDIT Questionnaire 4. Developed an ML model to Detect Diabetes 5. Built the UI and Backend for Diabetes Detection 6. Processed and Cleaned the DementiaBank Dataset 7. Built the UI and Backend for Alzheimer's Disease 8. Added new visuals and fixed the backend for Analytics page